Dr. Mars Caroline Wibowo. S.T., M.Mm.Tech

PRESPEKTIF BARU DALAM DESAIN WEB

YAYASAN PRIMA AGUS TEKNIK

PRESPEKTIF BARU DALAM
DESAIN WEB

Dr. Mars Caroline Wibowo. S.T., M.Mm.Tech

YAYASAN PRIMA AGUS TEKNIK

PENERBIT :
YAYASAN PRIMA AGUS TEKNIK
Jl. Majapahit No. 605 Semarang
Telp. (024) 6723456. Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

ISBN 978-623-8642-06-9 (PDF)
9 786238 642069

PRESPEKTIF BARU DALAM DESAIN WEB

Penulis :
Dr. Mars Caroline Wibowo. S.T., M.Mm.Tech

ISBN : 9 886238 642069

Editor :
Dr. Ir. Agus Wibowo, M.Kom, M.Si, M.M.

Penyunting :
Dr. Joseph Teguh Santoso, M.Kom.

Desain Sampul dan Tata Letak :
Irdha Yunianto, S.Ds., M.Kom.

Penebit :
Yayasan Prima Agus Teknik Bekerja sama dengan
Universitas Sains & Teknologi Komputer (Universitas STEKOM)

Anggota IKAPI No: 279 / ALB / JTE / 2023

Redaksi :
Jl. Majapahit no 605 Semarang
Telp. (024) 6723456
Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

Distributor Tunggal :
Universitas STEKOM
Jl. Majapahit no 605 Semarang
Telp. (024) 6723456
Fax. 024-6710144
Email : info@stekom.ac.id

KATA PENGANTAR

Puji Syukur penulis panjatkan atas kehadirat Tuhan Yang Maha Esa atas terselesaikannya buku yang berjudul "***Prespektif Baru Dalam Desain Web***". Keindahan Web terletak pada umur panjang dan fleksibilitasnya. Waktu tidak pernah berhenti dalam pengembangan Web, dan semakin banyak waktu yang Anda habiskan di Web, kecepatannya akan semakin cepat. Beberapa dari Anda mungkin sudah mulai mengembangkan situs web beberapa waktu lalu, yang lain baru saja terjun ke desain Web; tetapi Anda semua mungkin pernah melihat teknik dan praktik datang dan pergi, alat dan perpustakaan dipuji dan diabaikan, tren muncul dan gagal. Web itu dinamis dan serbaguna teknik pengkodean tidak hitam atau putih, dan keputusan kita selalu muncul dari area abu-abu tersebut. Tidak ada solusi yang sempurna, dan biasanya yang terpenting adalah mencapai kompromi yang baik dalam batasan yang ada. Desain web saat ini membutuhkan pendekatan pragmatis dan berpikiran terbuka.

Buku ini tidak akan mengubah semua pengetahuan Anda tentang desain dan pengembangan Web. Ini juga tidak akan merevolusi alur kerja atau alat Anda. Namun mudah-mudahan ini akan menantang Anda untuk berpikir sedikit berbeda tentang cara Anda mendekati masalah desain dan cara menyelesaikannya secara bermakna dalam kehidupan nyata. Buku ini adalah buku praktis untuk desainer profesional dan pengembang Web. Itu tidak membahas desain datar atau skeuomorfisme, dan tidak membahas gaya atau tren visual.

Dengan Perspektif Baru tentang Desain Web, penulis ingin mengeksplorasi teknik praktis dan strategi cerdas yang diperoleh dari proyek sebenarnya. Mencakup aspek teknis dan desain desain Web, buku ini memberikan wawasan tentang arsitektur front-end yang dapat diskalakan, trik back-end yang tidak jelas, desain Web responsif yang bertanggung jawab, dan optimalisasi kinerja, tetapi juga antarmuka adaptif, tipografi Web, strategi konten, dukungan pelanggan, dan proses kreatif, dan psikologi perilaku manusia di Web.

Dalam bab 1 buku ini akan membahas mengenai arsitektur CSS modern dan Pengembangan sebuah front-end. Bab 2 buku ini membahas prinsip-prinsip organisasi dalam pengembangan perangkat lunak, seperti konvensi kode, arsitektur yang kokoh, dan dokumentasi yang baik. Selain itu, strategi untuk memperbaiki dan memelihara kode lama juga dibahas, termasuk menambahkan dokumentasi dan pengujian. Bab ini juga menekankan pentingnya memahami bahwa basis kode memiliki kehidupannya sendiri, dan pertumbuhannya dipengaruhi oleh perawatan, struktur, dan kenyamanan dalam melakukan modifikasi. Dan pada bab ke 3 membahas pentingnya fleksibilitas dalam pengembangan web dan mengingatkan kita untuk kembali kepada prinsip-prinsip dasar yang membuat web sukses. Penulis juga menekankan bahwa web bukanlah platform untuk membatasi diri, tetapi sebagai wadah yang dapat menampung berbagai konten dan media dengan mudah. Daripada mencari solusi yang membatasi potensi web, penulis mendorong untuk mengadopsi gagasan bahwa masa depan web akan menjadi semakin beragam. Bab ke 4 membahas pentingnya merancang kinerja situs web secara cermat dan menyeluruh untuk menghindari masalah

seperti obesitas web. Hal ini menekankan perlunya memasukkan kinerja dalam proses pengembangan sejak awal, termasuk menetapkan anggaran kinerja, simulasi kecepatan koneksi yang buruk, dan menggunakan browser serta perangkat nyata. Lebih lanjut, bab tersebut menekankan bahwa perhatian terhadap kinerja bukan lagi tanggung jawab hanya bagi pengembang, tetapi juga bagi desainer, yang dapat mempengaruhi kinerja situs sejak tahap awal. Dengan perhatian yang tepat, situs web dapat menjadi cepat dan menyenangkan bagi pengguna, tanpa harus menjadi berat atau lambat.

Selanjutnya pada bab ke 5 buku ini akan membahas skeptisisme yang muncul terhadap desain web responsif, yang dianggap telah mendapat reputasi yang tidak sepadan. Penulis menyatakan tujuannya untuk menunjukkan cara membangun situs web responsif yang ramping, kaya fitur, dan mudah diakses untuk semua pengguna. Mereka mengakui bahwa sejak Ethan Marcotte memperkenalkan konsep ini, banyak artikel dan diskusi telah mengkritiknya, menyebutnya tidak efektif atau tidak cukup. Namun, penulis menegaskan bahwa desain web responsif dapat menjadi solusi yang efektif jika dilakukan dengan benar, tanpa ketergantungan berlebihan pada skrip atau komponen sisi server. Mereka mengingatkan pembaca bahwa desain web yang responsif seharusnya tidak memakan waktu sebanyak persiapan Natal, menyuarakan kebutuhan akan pendekatan yang praktis dan efisien. Bab 6 ini menyoroti pentingnya pengalaman bebas jank dalam pengembangan web seluler untuk mendekati standar pengalaman aplikasi asli. Selain keahlian desain dan CSS, penulis menekankan pentingnya memperhatikan kinerja, termasuk animasi halus yang mempertahankan kecepatan bingkai. Mereka menyarankan penggunaan alat pengembang browser untuk memantau kinerja antar browser yang berbeda dan memastikan perubahan gaya tidak membebani halaman. Penulis juga mengingatkan tentang pentingnya memperkecil ukuran area cat dan mengurangi perubahan ukuran gambar yang tidak perlu untuk menjaga performa halaman. Dengan mengoptimalkan kinerja, pengembang dapat memberikan pengalaman yang lebih lancar kepada pengguna, melewati batas bebas jank, dan memberikan pengalaman web yang lebih menyenangkan, apa pun perangkat yang digunakan pengguna.

Pada bab ke 7 membahas kompleksitas desain yang melibatkan keseimbangan antara daya tarik visual dan fungsi yang mudah dimengerti. Desain yang sukses harus memadukan kedua aspek ini dengan baik. Ketika estetika mendominasi kegunaan, desain cenderung menjadi lebih tentang ekspresi diri daripada memenuhi kebutuhan pengguna. Sebaliknya, ketika fokus pada kegunaan mengatasi estetika, desain bisa menjadi kurang menarik dan kurang memikat bagi pengguna. Penting untuk mencapai keseimbangan yang tepat agar desain dapat diakses dan disukai oleh pengguna. Bab 8 ini menggambarkan pengalaman yang menegangkan ketika sebuah situs web baru yang baru dikembangkan tiba-tiba mengalami kegagalan saat diakses. Mulai dari pengalaman bangun pagi yang penuh harap hingga kekecewaan saat menemui masalah, bab ini berjanji untuk membahas berbagai aspek perbaikan teknis back-end web. Dari masalah pada router hingga server, dari analisis log kesalahan hingga pertimbangan tentang peretasan PHP, penulis berjanji untuk menyelami skenario terburuk dan memberikan banyak tip dan perintah kecil sepanjang proses perbaikan. Bab ini bertujuan untuk memberikan wawasan baru tentang bagaimana infrastruktur internet

dan pembuatan server web dapat memengaruhi kinerja dan stabilitas situs web, serta memberikan pandangan tentang bagaimana mengatasi masalah yang muncul.

Selanjutnya bab 9 ini mengajak pembaca dalam perjalanan memperbaiki teknik back-end web dan mengungkap rahasia terminal. Dimulai dengan gambaran dramatis tentang kegagalan situs web yang baru saja dikembangkan, bab ini menjanjikan pembahasan dari router hingga server, dari log kesalahan hingga peretasan PHP. Penulis akan menjelajahi skenario terburuk dan memberikan wawasan mendalam tentang infrastruktur internet dan pembuatan server web. Di sepanjang bab, pembaca akan diberikan banyak tips dan perintah kecil untuk membantu memahami bagaimana situs web bisa mengalami masalah dan bagaimana cara memperbaikinya. Bab 10 buku ini membahas dua strategi dalam konten: mempertimbangkan kebutuhan pengguna dan memahami perspektif editor. Penulis menekankan tanggung jawab mereka untuk menciptakan situs web yang berguna bagi keduanya. Selanjutnya bab 11 membahas pentingnya dukungan pelanggan dalam mendukung produk. Meskipun produk mungkin sangat berguna dan bebas bug, banyak pelanggan, terutama desainer web yang belum berpengalaman dengan CMS atau database MySQL, dapat mengalami kesulitan dalam mengatur atau menggunakan produk. Penulis menyoroti betapa pentingnya memberikan dukungan yang baik kepada pelanggan, bahkan lebih dari sekadar fitur produk itu sendiri. Perilaku dan tanggapan perusahaan terhadap pertanyaan atau masalah pelanggan dapat membuat perbedaan besar dalam persepsi mereka terhadap produk atau layanan. Memberikan pengalaman pelanggan yang positif dalam dukungan adalah kunci untuk membangun hubungan yang kuat dan memastikan kepuasan pelanggan.

Selanjutnya dalam bab 12 ini bertujuan untuk mengajak pembaca untuk mempertimbangkan peran manusia sebagai pusat dari setiap keberhasilan atau kegagalan dalam hidup. Namun, pendekatan ini tidak mengambil jalan yang klise, seperti bersimpati di media sosial atau mempublikasikan postingan blog tentang hambatan birokrasi. Sebaliknya, penulis mengajak untuk menggali akar masalah dan berhenti hanya memperindah atau mengelak dari realitas yang sulit. Ini adalah panggilan untuk berbicara tentang inti masalah, bukan sekadar gejala yang terlihat.Dalam bab 13 yang merupakan bagian bab terakhir buku ini akan membahas mengenai tekanan yang sering kali muncul dalam memulai proyek baru, di mana klien seringkali mengharapkan hasil yang cepat dan dalam waktu singkat. Kondisi ini seringkali mendorong kita untuk mengambil jalan pintas dan menggunakan ide-ide yang sudah ada, daripada menghasilkan ide baru yang orisinal. Penulis mengajak kita untuk melangkah mundur dari dunia digital dan mencari inspirasi di dunia analog, seperti perpustakaan atau toko kopi. Mereka menekankan pentingnya membangun budaya ide yang melibatkan tiga faktor utama: mempersiapkan otak dengan pengetahuan, menemukan konduktor yang memfasilitasi kerja otak secara harmonis, dan mendorong terjadinya benturan ide yang tidak disengaja. Dengan menggabungkan elemen-elemen ini, penulis meyakini bahwa ide-ide akan mengalir deras.

Demikian buku yang berjudul "**Perspektif Baru Dalam Desain Web"** ini dengan harapan pembaca dapat mencermati perkembangan saat ini, dan terus meningkatkan alur

kerja dengan penyempurnaan kecil dan bertahap belajar untuk mengimbangi, memperkaya dan meningkatkan Web yang menakjubkan. Terima Kasih.

Selamat Membaca

Semarang, Juni 2024
Penulis

Dr. Mars Caroline Wibowo. S.T., M.Mm.Tech

DAFTAR ISI

Halaman Judul ..... **i**
Kata Pengantar ..... **ii**
Daftar Isi ..... **v**
BAB 1 ARSITEKTUR CSS MODERN DAN PENGEMBANGAN FRONT-END **1**
1.1. Pendahuluan 1
1.2. Tiga Pemangku Kepentingan 3
1.3. Beberapa Kesalahpahaman Tentang CSS 4
1.4. Pemisahan Isi Dan Gaya 6
1.5. Perhatikan Developer 7
1.6. Struktur Kode 9
1.7. Mengorganisasi Semuanya 11
1.8. Mengelola Semuanya 12
1.9. Anatomi Pemilih CSS 15
1.10. BEM: Blok, Elemen, Modifier 20
BAB 2 PENULISAN KODE YANG DAPAT DIPERTAHANKAN **26**
2.1. Konvensi Kode 26
2.2. Komunikasi Adalah Kunci 27
2.3. Arsitektur 32
2.4. Mengelola Komponen Pihak Ketiga 37
2.5. Komponen Pihak Ketiga 39
2.6. Tambahkan Uji 43
BAB 3 PRODUK WEB **46**
3.1. Kode Khusus Browser Tidak Dapat Dipercaya 53
3.2. Ajukan Pertanyaan “If” 55
3.3. Kegunaan Mengalahkan Konsistensi Di Seluruh Browser 60
3.4. Menganalisis Efek Beats Menambahkan Fx 62
BAB 4 BUDAYA PERTUNJUKAN **65**
4.1. Tentang Kinerja 66
4.2. Dampak Anggaran Kinerja 72
4.3. Pramuat Konten Secara Adaptif 82
4.4. Penyelesaian Tugas 83
BAB 5 DESAIN WEB YANG KUAT, BERTANGGUNG JAWAB, DAN RESPONSIF **85**
5.1. Peningkatan Presumtif 86
5.2. Peningkatan Progresif 87
5.3. CSS 92
5.4. Javascript 95
BAB 6 MENEMUKAN DAN MEMPERBAIKI MASALAH RENDERING WEB SELULER **105**

6.1. Jaringan 105
6.2. Animasi CSS..... 109
6.3. Alat Kinerja Devtools 111
6.4. Pintasan Untuk Menyembunyikan Elemen Dom Dengan Cepat 113
6.5. Penghitung FPS 116
6.6. Alur Kerja Optimasi 118
6.7. Alat Pengembang..... 126
BAB 7 MERANCANG ANTARMUKA ADAPTIF 132
7.1. Pendahuluan 133
7.2. Inti 137
7.3. Mempertimbangkan Kendala 137
7.4. Blok Bangunan 151
7.5. Membuat Komponen Adaptif 152
BAB 8 MEMPERBAIKI TEKNIK BACK-END WEB DAN RAHASIA TERMINAL 156
8.1. Infrastruktur 156
8.2. Gerbang Default 158
8.3. Server DNS Lokal 160
8.4. Router Broadband Dikembalikan 163
8.5. Telnet Dan Netcat 165
8.6. Kegagalan Layanan 170
8.7. Log Kesalahan Server Web 172
8.8. Antarmuka Kontrol Apache 180
8.9. Kesalahan Situs Web 183
8.10. Log Akses 192
BAB 9 LANGKAH SELANJUTNYA UNTUK TIPOGRAFI WEB 195
9.1. Gambaran Besar: Tipografi Universal 195
9.2. Mesin Rendering OS 203
9.3. Menggabungkan Tipografi 211
9.4. Sejarah Tipografi 213
9.5. Jaringan Dasar 215
9.6. Menggabungkan Elemen Pseudo Dengan Properti Konten 224
BAB 10 DUA SISI STRATEGI KONTEN 235
10.1. Penemuan: Siapa Dan Mengapa? 238
10.2. Mengaudit Proses Yang Ada 241
10.3. Menentukan Peran 243
10.4. Pendidikan Dan Iterating 250
BAB 11 MENDUKUNG PRODUK ANDA 254
11.1. Nilai Tersembunyi Dari Dukungan Pelanggan 225
11.2. Mengelola Permintaan Fitur 257
11.3. Masalah Klien Akhir 262
11.4. Strategi Untuk Meminimalkan Dukungan 267

11.5. Alat Untuk Dukungan 269
BAB 12 DESAIN ORANG 278
12.1. Efek Psikologi Sistem Adil 279
12.2. Menyatakan Perilaku Yang Baik Dalam Tinjauan Desain 281
12.3. Uji Sally–Anne 288
BAB 13 TENTANG SEMANGAT KREATIF 297
13.1. Menempatkan Teori Ke Dalam Praktek 297
13.2. Mitos Inovasi 299
13.3. Teknik Menghasilkan Ide 300
Daftar Pustaka 315

BAB 1
ARSITEKTUR CSS MODERN DAN PENGEMBANGAN FRONT-END

1.1 PENDAHULUAN

Revolusi Industri, yang terjadi antara pertengahan tahun 1700an hingga pertengahan tahun 1800an, menandakan perubahan besar dalam teknik manufaktur komersial dunia; pengerjaan dan pengerjaan tangan tradisional membuka jalan bagi proses-proses baru yang melibatkan mesin dan otomatisasi. Pada awal abad ke-19 di Inggris, meningkatnya mekanisasi memunculkan kelompok yang dikenal sebagai Luddites, sebuah formasi pekerja tekstil artisanal pengrajin dan pekerja terampil yang pekerjaannya perlahan tapi pasti menjadi usang karena gelombang baru sistem yang dipimpin oleh mesin. manufaktur1. Sayangnya, kaum Luddite bukanlah sebuah produk sampingan yang tidak terduga dari Revolusi Industri.

Mesin yang diperkenalkan dapat melakukan pekerjaan sepuluh pekerja terampil, dan kecepatannya sepuluh kali lipat. Pekerjaan mereka yang lamban dan buatan tangan digantikan dengan keinginan akan alat produksi yang lebih kuat dan menguntungkan. Pergeseran ini, tentu saja, tidak diterima dengan baik oleh para pekerja, yang semakin merasa tersingkir dari peran yang telah mereka emban selama berpuluh-puluh tahun. Kaum Luddite adalah contoh cemerlang dari orang-orang yang menolak perubahan dan membayar mahal perubahan tersebut. Perubahan itu baik, perubahan itu sulit, perubahan itu perlu, dan perubahan akan terjadi yang terpenting adalah cara kita menghadapinya.

Web sedang berubah. Apakah kamu mengikuti?

Web Lalu

Ada suatu masa (sebenarnya belum lama ini) ketika situs web dibuat menggunakan tabel untuk strukturnya, dan markup seperti `<fontcolor="red">Foo</font>` digunakan untuk menerapkan fitur visual semata. Teknik-teknik ini bertahan cukup lama. Saya mulai mendalami pengembangan Web baru-baru ini, sekitar tahun 2007 atau lebih. Saya sangat beruntung karena zaman tabel untuk tata letak sudah lama berlalu, dan saya langsung memasuki arena di mana teknologi seperti CSS dianggap sebagai standar. Setiap artikel yang saya baca berbicara tentang semantik, HTML bersih, dan menggunakan ID dan kelas sesedikit mungkin.

CSS muncul bertahun-tahun sebelum minat saya dalam membangun situs web. Tabel dihapuskan dan diganti dengan bahasa yang jauh lebih kuat dan sesuai. Bagi banyak orang, hal ini merupakan perubahan besar dari apa yang biasa mereka alami; bagi saya itu hanya bagaimana hal-hal dilakukan ketika saya tiba di tempat kejadian. Peralihan dari tata letak berbasis tabel ke tata letak berbasis CSS, secara keseluruhan, merupakan perubahan yang disambut baik. Butuh beberapa waktu bagi beberapa pengembang untuk melakukan langkah ini karena, seperti yang selalu terjadi pada pengembangan front-end, sebagian besar orang bergantung pada browser yang digunakan oleh audiens mereka.

Terlepas dari berapa lama (atau tidak) transisi tersebut berlangsung, orang-orang pada umumnya tampaknya menyukai CSS. Ini merupakan perubahan besar dalam alur kerja

pengembangan dan produksi yang hampir semua orang tentu saja setiap orang yang saya kenal secara pribadi menyambutnya dengan tangan terbuka. CSS memungkinkan kita melakukan lebih banyak hal daripada biasanya: kita dapat membangun sesuatu dengan lebih cepat, lebih cerdas, lebih besar, dan lebih baik.

Kami menerima perubahan tersebut dan segalanya menjadi lebih baik. Meskipun ada perubahan dalam teknologi, situs web pada umumnya masih berukuran kecil. Belasan tahun yang lalu, situs web sering kali hanya berisi beberapa halaman yang digunakan untuk memamerkan gambar kucing Anda dan memutar lagu Rush favorit Anda secara otomatis; sekarang situs web seringkali berukuran ratusan, bahkan ribuan halaman dan menghasilkan puluhan miliar dolar per tahun.

Peralihan dari tabel ke CSS merupakan perubahan teknologi, namun hanya ada sedikit perubahan dalam lanskap; situs-situs lima belas tahun yang lalu biasanya memiliki ukuran dan tujuan yang relatif sama, terlepas dari apa dasar pembangunannya. Web saat itu merupakan tempat kecil yang jauh lebih sederhana, dan saran yang diberikan kepada kami mengenai standar Web, CSS, semantik, dan kode bersih serta menghindari markup dan kelas tambahan memang benar adanya. Tapi itu dulu; Web telah berubah dan masih terus berubah namun saya khawatir kita sudah berhenti mengubahnya. Praktik terbaik kami telah dirombak sekitar satu dekade yang lalu, namun sepertinya kami tidak lagi meninjaunya sejak saat itu.

Web Sekarang

Web saat ini sangatlah berbeda: situs web biasanya jauh lebih besar, jauh lebih kompleks, jauh lebih penting (sering kali situs tersebut menjadi satu-satunya sumber pendapatan masyarakat), dan menghasilkan lebih banyak uang. Dengan perusahaan seperti Amazon yang membukukan pendapatan lebih dari Rp.60 miliar pada tahun 20122, Twitter memiliki 200 juta pengguna aktif, Google mempekerjakan hampir 54.000 orang4, dan bahkan perusahaan tempat saya bekerja Sky Bet menghasilkan lebih dari seratus juta pound per tahun5 dan mempekerjakan 450 orang. bagi orang-orang semata-mata dari uang yang diperoleh secara online, sangatlah naif jika kita berpikir bahwa kita dapat memperlakukan situs web saat ini seperti yang kita lakukan sepuluh atau bahkan lima tahun yang lalu. Namun, masih banyak dari kita yang melakukannya.

Dengan semakin besarnya situs web, tim pengembang mereka semakin besar, dan tujuan mereka menjadi lebih jelas dan, menurut saya, yang lebih penting, kita perlu melihat lagi bagaimana sebenarnya kita membangunnya. Tidak lagi praktis untuk membuat kode dengan tangan, dan mengejar kemurnian semantik serta markup yang bersih; diperlukan pendekatan yang jauh lebih cepat, kuat, dan pragmatis. Kita perlu berubah lagi.

Buku-buku, artikel dan posting blog yang ditulis berdasarkan standar Web selama sepuluh tahun terakhir ini semuanya berisi nasihat yang relevan dan masuk akal tidak dapat disangkal lagi – namun Web telah berubah dengan kecepatan yang jauh lebih cepat dibandingkan dengan praktik terbaik yang kita lakukan. Kita sudah melampaui metode-metode yang ada selama bertahun-tahun dan kini terserah pada kita untuk memperbarui peraturan tidak tertulis dalam pengembangan Web agar sesuai dengan lanskap baru. Situs yang lebih besar dan mirip aplikasi yang melayani jutaan pengguna setiap hari dan

menghasilkan pendapatan miliaran dolar memerlukan pendekatan yang tidak dapat ditawarkan oleh saran di awal tahun 2000-an. Kita perlu menerima perubahan sikap lainnya, yang mempertimbangkan perbedaan sifat Web saat ini, dan juga orang-orang yang terlibat di dalamnya: pemangku kepentingan Anda.

1.2 TIGA PEMANGKU KEPENTINGAN

Secara umum, dalam setiap proyek Web, terdapat tiga pemangku kepentingan utama, yaitu tiga kelompok orang yang paling mementingkan situs tersebut. Tentu saja ini merupakan generalisasi besar-besaran, namun menurut saya, sebagian besar, hal ini benar adanya:

1. Klien: Orang yang membayar Anda untuk membangun situs, orang yang membayar tagihan Anda.
2. Pengguna: Orang-orang yang akan menggunakan situs yang dibangun oleh klien yang membayar Anda.
3. Anda: Pengembang yang harus bekerja dengan, menskalakan, dan memelihara situs.

Anda perlu mengingat para pemangku kepentingan ini dan memastikan Anda melakukan hal yang benar untuk orang yang tepat dengan alasan yang tepat.

Klien

Klien tidak peduli dengan semantik. Klien tidak peduli berapa banyak ID dan kelas yang Anda miliki atau belum gunakan. Mereka tidak ingin tahu bahwa Anda tidak dapat dengan cepat menduplikasi suatu konten karena Anda menggunakan ID dan ID tidak dapat digunakan kembali. Bukan masalah mereka untuk mengkhawatirkan hal-hal seperti itu, dan tanggung jawab Anda adalah melindungi mereka dari hal-hal tersebut. Klien peduli dengan seberapa cepat dan efisien (dan murah) Anda dapat memperbarui situs mereka, mereka peduli dengan seberapa andal dan kuatnya situs tersebut, seberapa baik kerjanya di berbagai browser dan perangkat.

Pengguna

Pengguna tidak peduli dengan kode: kita semua menggunakan situs hari demi hari, yang menggunakan kode yang tampak buruk (meskipun kode itu sendiri belum tentu buruk). Bahkan sebagai pengembang, kami tidak peduli seberapa baik markup Facebook atau Google ditulis; kami peduli bahwa situs dan layanan yang ingin kami gunakan cepat dan dapat diandalkan. Kode bagi situs web sama seperti batu bata dan mortir bagi sebuah bangunan: sangat penting, namun tidak benar-benar sesuai dengan apa yang dilihat orang.

Pengembang

Anda peduli dengan seberapa baik basis kode terdokumentasi, betapa nyamannya digunakan, seberapa alami basis kode tersebut dapat ditingkatkan, seberapa mudahnya dipelihara, seberapa cepat Anda dapat melakukan perubahan, seberapa efektif Anda dapat memenuhi permintaan klien. Klien tidak peduli jika Anda berhasil menghindari penggunaan kelas apa pun dan tidak menulis markup yang asing, tetapi jika menggunakan dua kelas dan div tambahan membuat pekerjaan Anda lebih mudah, dan membuat situs lebih mudah dipelihara, Anda harus memilih kode tambahan setiap waktu. Tidak ada gunanya membuat

hidup Anda lebih sulit karena Anda berpikir bahwa menghindari beberapa karakter tambahan HTML adalah cara yang tepat. Bersikaplah baik kepada diri sendiri dan tim Anda.

Dengan mengingat ketiga pemangku kepentingan ini, kita dapat mulai membuat keputusan yang lebih tepat tentang cara kita menulis kode front-end: pengguna tidak peduli jika kita telah menulis kelas semantik; klien tidak peduli berapa banyak ID yang kami gunakan; dan hanya kami yang peduli betapa bagusnya kode kami untuk digunakan.

Faktanya, pengembang adalah satu-satunya orang yang benar-benar peduli dengan kode. Tulislah untuk Anda sendiri, tetapi tulislah agar cepat (untuk pengguna) dan kuat (untuk klien). Setiap orang mendapat manfaat dari kode yang baik, tetapi pengguna dan klien hanya mendapat manfaat dari hasilnya. Pengembang mendapat manfaat langsung dari kualitas kode sebenarnya (konvensi penamaan yang masuk akal, banyak komentar, dan sebagainya).

1.3 BEBERAPA KESALAHPAHAMAN TENTANG CSS

Masih ada beberapa aspek CSS (dan HTML) yang sering disalahpahami yang — pada tingkat mendasar akan memengaruhi cara Anda membangun situs web. Kesalahpahaman ini perlu diakui dan diperbaiki. Dengan melakukan hal ini, kita akan terbebas dari dogma lama dan merasa lebih terbebas serta berdaya untuk membangun front-end yang lebih besar, lebih cepat, dan lebih kuat.

Sejujurnya saya yakin bahwa saya menyadari, memahami, dan kemudian membuang kesalahpahaman yang mengubah saya dari sekadar menulis CSS menjadi mampu menskalakan CSS untuk situs web besar.

Semantik

Kesalahpahaman pertama kami melibatkan semantik. Semantik - setidaknya dalam kaitannya dengan pengembangan front-end - adalah bidang yang sangat disalahpahami dan dipertahankan secara berlebihan. Semantik dalam pengembangan front-end berkaitan dengan penggunaan div atau header, atau apakah teks harus berupa paragraf atau judul, atau apakah daftar harus diurutkan atau tidak, dan sebagainya.

Semantik dalam pengembangan front-end adalah tentang elemen dan bukan atribut. Ini tentang mesin (pembaca layar, browser, teknologi bantu, bot) yang mengumpulkan makna dari elemen yang kami gunakan untuk menandai dokumen kami. Semantik bukan tentang bagaimana kita memberi nama kelas atau ID kita; satu-satunya hal yang mengumpulkan makna dari kelas dan ID adalah manusia. Apa pun yang membaca atau mengakses kelas atau ID hanya mencocokkannya; itu tidak mendapatkan arti apa pun darinya sama sekali. Ini, langsung dari spesifikasi HTML(5):

Makna tertentu tidak boleh diturunkan dari nilai atribut ID.

Ambil dua cuplikan berikut:

```
<div class="heading-one">My page title</div>
```

Dan:

```
<h1 class="red">My page title</h1>
```

Yang pertama adalah implementasi semantik yang tidak tepat: penulis harus menggunakan elemen h1 untuk menandai judul halamannya, bukan div . Meskipun kelas yang digunakan sangat deskriptif, namun semantiknya tidak terletak di sini. Mesin tidak akan mendapatkan apa pun dari potongan markup tersebut.

Cuplikan kedua jauh lebih semantik: mesin yang membaca ini akan mengetahui bahwa itu adalah judul tingkat atas. Fakta bahwa penulis menggunakan kelas merah sama sekali tidak relevan dengan mesin mana pun. Jika ada aturan .red{} yang disetel di CSS, maka browser akan memberi gaya pada judul ini, apa pun yang terjadi.

Semantik VS. Kepekaan

Ketika kebanyakan orang berbicara tentang semantik, yang mereka maksud sebenarnya adalah kewajaran. Seperti yang telah kami pastikan, kelas dan ID tidak ada hubungannya dengan semantik sama sekali. Kelas merah bukanlah semantik dan bukan semantik, karena kelas tidak ditemukan pada spektrum semantik. Namun, kelas warna merah mungkin kurang tepat.

Saat memilih nama untuk kelas Anda, lupakan gagasan semantik dan lebih memperhatikan umur panjangnya. Sampai kapan nama ini akan terus masuk akal? Berapa lama hal itu akan tetap relevan? Seberapa masuk akalnya hal itu dalam enam bulan?

Mengambil contoh .red lebih jauh, jelas terlihat mengapa ini adalah nama yang tidak bijaksana untuk sebuah kelas. Bayangkan Anda sedang mengerjakan situs e-niaga dan Anda perlu menentukan harga secara berbeda karena ini merupakan penawaran khusus, seperti:

```
<strong class="red">$99.99</strong>

.red {
     color: red;
 }
```

Jika gaya tersebut kebetulan melibatkan warna merah maka kelas dan pemilih warna merah mungkin digunakan, mungkin tidak disarankan. Jika, karena alasan apa pun, Anda ingin mengubah semua harga penawaran khusus menjadi biru, maka Anda akan mendapatkan hasil seperti ini:

```
<strong class="red">$99.99</strong>
.red {
     color: blue;
}
```

Mesin tidak akan kesulitan dengan hal ini sama sekali satu-satunya tugas mesin adalah mencocokkan dan menata kelas. Namun, yang menjadi canggung adalah ketika manusia (yaitu pengembang) mulai bekerja dengan kode ini. Sungguh membingungkan melihat kelas warna merah yang sebenarnya membuat sesuatu menjadi biru. Ini bukanlah kelas yang masuk akal, kelas yang umurnya pendek dan tidak selalu masuk akal.

Pilihan yang jauh lebih baik adalah:

```
<strong class="special-offer">$99.99</strong>

.special-offer {
     color: blue;
 }
```

Tidak masalah apakah penawaran spesialnya berwarna merah, hijau, biru, atau ungu pemilih tersebut akan selalu masuk akal bagi manusia. Ini adalah kelas yang masuk akal, meskipun tidak semantik. Kepekaan adalah tentang kemudahan pemeliharaan, umur panjang, dan seberapa masuk akal suatu hal bagi manusia, karena manusia adalah satu-satunya hal yang dapat mengambil makna apa pun dari nama kelas.

1.4 PEMISAHAN ISI DAN GAYA

Secara tradisional, pemisahan konten dan gaya mengacu pada pemisahan markup dan gaya konten ke dalam bahasa yang berbeda. Hal ini tidak berarti memisahkan kedua bahasa secara fisik.

Dulu, kami menulis HTML seperti `<font color="red">Foo</font>`.

Masalahnya di sini adalah HTML menandai dan menata konten ini, padahal kita harus menggunakan bahasa yang berbeda untuk masing-masingnya. Inilah sebabnya CSS lahir: untuk memisahkan kekhawatiran ini. Pemisahan konten dan gaya mengacu pada penggunaan teknologi yang terpisah dan independen untuk setiap peran, bukan untuk menghindari kelas presentasi dalam markup Anda.

Contoh sebelumnya menunjukkan kurangnya konten dan gaya yang terpisah; contoh berikut konten dan gayanya terpisah dengan sempurna:

```
<a href="#" class="big-red-button-with-white-text">Foo</a>
```

Namanya sangat buruk. Pemisahan konten dan gaya adalah tentang memisahkan konten dan bahasa gaya, bukan lokasinya. Dengan memahami hal ini, Anda mulai melihat bahwa kesalahpahaman ini telah menyebabkan kebencian dogmatis dan menyeluruh terhadap segala bentuk kelas presentasi. Hanya karena kelas mungkin dibaca sebagai

presentasi tidak berarti mencampurkan konten dan gaya; sebenarnya hal ini kembali ke poin kami di atas: ini semua tentang seberapa bijaksana Anda melakukan sesuatu.

Cuplikan HTML seperti `<div class="red"> ... </div>` telah memisahkan konten dan gaya dengan sempurna, hanya saja tidak terlalu masuk akal.

Kesimpulan: Kami masih memiliki beberapa kesalahpahaman seputar CSS. Mengenali dan menghapuskan hal-hal tersebut sangat membebaskan dan membuka kemungkinan untuk membangun front-end yang lebih baik.

Perubahan Sikap

Saat ini, kita sudah memiliki kerangka berpikir yang lebih siap untuk membangun front-end yang lebih kuat dan pragmatis. Jika kita menerima bahwa Web adalah lingkungan yang jauh lebih serius dibandingkan sepuluh tahun yang lalu, dan jika kita menerima bahwa kode yang kita tulis harus melayani sejumlah orang yang berbeda dalam beberapa cara yang berbeda, maka menurut saya itu tidak benar. terlalu berlebihan untuk berasumsi bahwa kita juga menerima bahwa kita memerlukan perubahan sikap.

Seperti yang pernah dikatakan Nicole Sullivan, “Praktik terbaik (CSS) kami membunuh kami”; keinginan kita untuk menulis markup yang tidak menggunakan kelas apa pun tidak bermanfaat bagi siapa pun, dan sering kali menimbulkan masalah bagi diri kita sendiri.

Mengejar markup yang bersih dan kelas semantik dan semua yang menyertainya memiliki niat baik tetapi tidak membantu siapa pun. Hal ini menyebabkan style sheet yang bertele-tele, sulit dirawat, dan kusut. Akibat dari menghilangkan beberapa kelas adalah harus menulis penyeleksi yang besar dan berbelit-belit untuk menargetkan elemen DOM yang tidak memiliki nama dan tidak disebutkan namanya ini. Seperti kata pepatah, kami merampok Peter untuk membayar Paul. Kami menulis markup yang bersih dengan mengorbankan penulisan CSS yang bertele-tele, berantakan, dan berbelit-belit. Kami baru saja memindahkan kekacauan itu ke tempat lain.

Web saat ini memerlukan pandangan yang lebih informatif dan tidak terlalu dogmatis mengenai hal-hal seperti semantik. Kita perlu menyadari bahwa kode kita yang sebenarnya hanya menguntungkan pengembang lain, jadi kita perlu menulisnya untuk mereka. Kita perlu menyadari bahwa pengguna dan klien hanya menginginkan situs web yang cepat, andal, dan tangguh. Web tumbuh semakin besar, semakin cepat. Kode bukan lagi sebuah kerajinan, melainkan alat yang berdaya.

Saya percaya bahwa semakin cepat kita membuang cita-cita yang bertujuan baik namun salah arah, maka kita bisa lebih bebas dan mampu membangun situs yang lebih besar, berkualitas lebih baik, dengan cara yang lebih tepat waktu dan responsif. Seperti yang saya sebutkan sebelumnya, mengakui dan membuang cita-cita kuno inilah yang menjadikan saya seorang pengembang seperti sekarang ini.

1.5 PERHATIKAN DEVELOPER

Salah satu perubahan terbesar dalam sikap yang saya lakukan adalah beralih dari front-end developer yang berorientasi pada desain menjadi front-end developer yang berorientasi pada teknik. Beberapa tahun yang lalu, saya adalah seorang ahli semantik garis keras. Saya

percaya bahwa markup saya harus dibuat dengan tangan, bersih, dan sempurna, dan menghindari kelas serta menggunakan penyeleksi yang cerdas adalah hal yang luar biasa, dan merupakan tanda pasti dari pengembang yang hebat!

Kemudian saya memulai pekerjaan saya sebagai pengembang UI senior di BSkyB, di mana saya adalah satu-satunya pengembang front-end di sebuah perusahaan yang penuh dengan insinyur perangkat lunak yang sangat berbakat. Saya telah pindah dari sebuah perusahaan yang mirip agensi di mana pengembang front-end lainnya memiliki cita-cita yang sama kerinduan akan lean markup dan semantik dan sekarang saya berada di sebuah perusahaan yang tidak ada orang yang berpikiran seperti itu sama sekali, dan saya sedang mengerjakan situs yang harus saya skalakan dan pelihara selama bertahun-tahun yang akan datang. Berada di lingkungan ini, dikelilingi oleh para insinyur, sungguh membuka mata saya.

Saya belajar banyak dari orang-orang ini, hal-hal seperti abstraksi, KEKERINGAN, pengembangan berorientasi objek, kinerja, ekstensibilitas. Saya belajar langsung dari beberapa orang terpintar yang pernah saya temui bahwa kode bukanlah tentang kecantikan, ekspresif, atau menghindari sesuatu hanya karena alasan tertentu. Saya belajar bahwa Anda tidak mendapatkan poin karena menjadi cantik, Anda mendapatkan poin karena menjadi kuat. Pergeseran pendekatan ini, bagi saya, adalah hal yang memulai segalanya. Saya mulai bertanya tentang konsep seperti prinsip tanggung jawab tunggal, abstraksi, pemikiran tingkat tinggi di balik orientasi objek, dan masih banyak lagi. Kemudian saya mulai bertanya-tanya bagaimana menerapkan prinsip-prinsip pemrograman yang telah dicoba dan diuji ini pada kode front-end: yaitu CSS.

Kesimpulan: Sudah waktunya kita melihat kode kita dengan pandangan baru untuk menerima dan mengelola perubahan fokus, ukuran dan sifat proyek front-end.

CSS Berorientasi Objek

OOCSS, istilah dan metodologi yang diciptakan oleh Nicole Sullivan banyak meminjam dari dunia pengembangan perangkat lunak dan pemrograman. Pemrograman OO adalah paradigma yang bertujuan untuk memisahkan kode menjadi potongan-potongan logis yang terpisah yang kemudian dapat berinteraksi satu sama lain. Alih-alih menggunakan satu potongan kode monolitik untuk menyelesaikan tugas ABC, Anda akan mendapatkan masing-masing A, B, dan C yang diabstraksi ke dalam objeknya sendiri. Sekarang kita punya A yang bisa kita gabungkan dengan B atau C dalam urutan apa pun yang kita pilih.

Ide ini, yang dibawa ke CSS, memungkinkan kita menghilangkan kumpulan aturan rumit yang melakukan satu hal tertentu, sehingga kita dapat menggunakan beberapa kumpulan aturan yang lebih kecil dan sederhana yang dapat digabungkan untuk mencapai hasil yang sama.

Selain abstraksi, OOCSS juga memperkenalkan prinsip struktur dan kulit: apa yang membentuk struktur suatu komponen harus ada secara terpisah dari apa yang membentuk fitur visualnya. Kerangka dasar yang sama dapat diulang berulang kali di situs web, namun dikuliti agar terlihat sangat berbeda di setiap kasus. Inilah keuntungan mengabstraksi objek yang berulang: kemampuan untuk memanipulasi dan menggunakannya kembali dalam berbagai skenario berbeda.

Kesimpulan: Temukan pola desain yang berulang, abstrak, dan gunakan kembali. Pisahkan komponen ke dalam struktur dan kulit dan terapkan paradigma pemrograman yang telah teruji. Untuk informasi lebih lanjut tentang OOCSS saya akan merekomendasikan membaca GitHub Wiki10 Nicole Sullivan.

1.6 STRUKTUR KODE

Saat menulis CSS dalam jumlah besar, penting untuk merencanakan bagaimana Anda akan mengaturnya. Seperti kata pepatah lama, gagal merencanakan berarti merencanakan kegagalan, dan sangat penting bagi Anda untuk menghabiskan banyak waktu memikirkan arsitektur Anda sebelum terjun langsung dan mulai membuat kode.

Jika Anda mempunyai hipotek atau hutang pelajar, atau pernah melakukan pembersihan musim semi, Anda akan terbiasa membagi hal-hal ini menjadi bagian-bagian yang lebih kecil dan lebih mudah dikelola. Anda mungkin memiliki hipotek sebesar Rp.2.000.000, tetapi kemungkinan besar Anda tidak akan pernah membayarnya kembali sekaligus; Anda membayar beberapa ratus dolar sebulan agar jumlah sebesar itu lebih mudah dikelola. Saat Anda membersihkan rumah di musim semi, Anda membaginya menjadi tugas-tugas kecil: membersihkan kamar mandi, menyedot debu karpet, mencuci cat. Merinci semuanya hanyalah pendekatan yang masuk akal untuk mengatasi hambatan yang lebih besar selangkah demi selangkah, dan hal yang sama berlaku untuk membangun situs web.

Saat dihadapkan pada pembangunan front-end berskala besar, penting untuk tidak melihat gambaran yang lebih besar. Perbesar dan lihat tugas-tugas kecil yang, saat Anda menyelesaikannya, akan mengurangi proyek secara keseluruhan. Meminjam pepatah lain, jagalah uang dan pound akan mengurus dirinya sendiri. Atau, seperti yang sering dikatakan oleh teman baik saya Jamie Mason, "*jagalah bit-bitnya, maka byte-byte akan menjaga dirinya sendiri*."

Lego

Salah satu analogi favorit saya untuk memecah kode berasal dari Nicole Sullivan. Dia berkata bahwa Anda harus memperlakukan kode seperti Lego; banyak bongkahan yang lebih kecil dan sederhana dapat digabungkan dan disusun untuk membuat berbagai macam struktur. Dengan sekotak potongan Lego yang berbeda-beda, kamu bisa membuat mobil, atau pesawat terbang, atau Menara Eiffel, atau Big Ben, atau apa pun! Semuanya dengan menggabungkan potongan-potongan kecil yang sama dalam jumlah berbeda, dengan cara berbeda, dan dalam urutan berbeda.

Jadi, daripada melihat proyek Anda dan bersiap membuat halaman kontak monolitik, pikirkan tentang menggabungkan serangkaian masukan dan tombol di dalam area konten, yang berada di sebelah area subkonten, yang dikelilingi oleh header dan sebuah catatan kaki. Berhentilah berpikir dalam kerangka halaman dan mulailah berpikir dalam kerangka komponen. Jika Anda perlu membuat formulir pencarian, Anda mungkin dapat menggunakan kembali dan menggunakan kembali masukan yang sama dari formulir kontak. Memecah kode Anda menjadi potongan-potongan seperti Lego memberi Anda kemampuan untuk melakukan lebih banyak hal dengannya.

Analogi lain yang pernah saya gunakan sebelumnya adalah analogi Subway. Jika Anda pernah memesan di toko sandwich Subway, Anda pasti familiar dengan cara mereka menawarkan makanan. Mereka memecah semuanya menjadi bahan-bahan terpisah, sehingga Anda dapat menggabungkan daging pilihan Anda dengan salad favorit Anda dan menambahkan saus favorit Anda di atasnya.

Menguraikannya ke tingkat ini menghasilkan sejumlah besar kombinasi bahan dasar yang sama, dan memungkinkan pelanggan untuk memilih bagian yang mereka suka dan tidak inginkan. Suatu hari Anda bisa makan tuna dan mentimun dengan jagung manis dan mayo; keesokan harinya Anda bisa mengawetkan daging dengan mayo, bawang bombay, selada, dan mentimun. Ada beberapa bahan yang digunakan bersama pada keduanya, namun karena masing-masing bahan berbeda, Anda cukup menukar bahan dan membuat sandwich yang benar-benar baru. Jadi, daripada membuat kode sandwich, kodekan semua bahannya dan buatlah sandwich Anda dari bahan-bahan tersebut.

Dengan memikirkan kode Anda dengan cara yang sama seperti Anda memikirkan bahan Lego dan sandwich, Anda seharusnya bisa mulai melihat gambar yang lebih kecil dan fokus pada tugas mikro yang lebih mudah. Hasil dari tugas ini adalah keseluruhan potongan kode yang dapat digabungkan.

Dimana Harus Memutuskannya

Memang benar bahwa kita perlu memecah kode menjadi bagian-bagian yang lebih kecil, tapi di mana kita memecahnya? Sayangnya, tidak ada jawaban benar atau salah. Tetapi! Saya merasa ada beberapa analogi dan petunjuk yang bermanfaat. Saya telah menyebutkan sebelumnya bahwa memecah-mecah sesuatu hanyalah hal yang masuk akal, jadi pikirkan sekarang tentang bagaimana Anda, sebagai pengembang Web yang mungkin tidak pernah bekerja di bidang konstruksi, akan membangun rumah. Saya membayangkan kita semua membangun rumah dengan urutan sebagai berikut:

1. Landasan untuk membangun segala sesuatunya.
2. Struktur seperti dinding, untuk menopang bangunan.
3. Perlengkapan seperti pintu, jendela dan tangga.
4. Dekorasi seperti kertas dinding atau cat, karpet atau lantai kayu.
5. Ornamen seperti lukisan, kanvas dan poster.

Ini adalah perintah yang masuk akal untuk membangun rumah; kita tidak bisa menggantung foto sebelum kita mempunyai tembok, dan kita tidak bisa mendirikan tembok tanpa pondasi, jadi kita harus menyusun barang-barang dalam urutan tertentu. Itu tidak berarti kita harus membuat bagian-bagian individual dalam urutan tertentu, tentu saja tidak. Ornamen kita mungkin merupakan barang antik yang dibuat ratusan tahun yang lalu, namun mereka tidak dapat menjadi bagian dari rumah sampai kita memiliki rak di tempatnya.

Kalau begitu, mari kita bawa analogi ini ke situs web kita. Fondasi kami adalah penyetelan ulang CSS kami; sistem jaringan kita bersifat terstruktur; komponen dan widget adalah perlengkapan dan kelengkapan kami; desain kami adalah dekorasi; dan, yang terakhir, skinning atau pemberian tema apa pun (misalnya, lencana penawaran khusus pada suatu produk, atau logo bertema Natal) adalah hiasan kami.

Sekali lagi, kita dapat mendesain komponen dan widget jauh sebelum dimasukkan ke dalam sebuah halaman, namun ketika menyangkut urutan perakitan, perlu ada halaman di sana untuk memasukkan komponen tersebut. Kita perlu membangun sesuatu dalam urutan tertentu, jadi mengapa tidak mulai memecah kode kita di sana?

Jadi sekarang kita harus bisa mulai memecah build-up kita menjadi beberapa bagian yang logis, misalnya:

1. Saya mau pakai normalize.css, jadi harus masuk dulu.
2. Sistem grid inilah yang akan saya gunakan.
3. Saya ingin tampilan h2 yang tidak berkelas seperti ini.
4. Seperti inilah tampilan widget pencari lokasi toko.
5. Seperti inilah tampilan widget pencari lokasi toko jika tidak ada toko di wilayah Anda.

Kita dapat menjaga semua hal ini tetap baik dan terpisah, jadi kita harus melakukannya. Kami memiliki titik-titik yang nyaman dan alami untuk mulai memecah kode kami menjadi beberapa bagian.

Potongan-potongan kecil ini memberi kita kode yang menganut apa yang, dalam bidang ilmu komputer, dikenal sebagai prinsip tanggung jawab tunggal. Definisi prinsip tanggung jawab tunggal saya yang encer dan sangat tinggi pada dasarnya adalah: kode yang melakukan satu pekerjaan, satu pekerjaan saja, dan satu pekerjaan dengan sangat baik.

Pikirkan kembali bahan-bahan Subway kami: wortel sangat bagus untuk menjadi wortel, karena hanya itulah yang dimaksudkan untuk dilakukan. Wortel benar-benar menyebalkan sebagai ayam, tapi tidak apa-apa karena tidak ada hubungannya dengan ayam. Idenya adalah kita memiliki banyak potongan kecil kode yang masing-masing memiliki satu tanggung jawab. Tanggung jawab ini tidak boleh menyatu atau bocor satu sama lain, namun harus berjalan dengan baik jika digabungkan.

1.7 MENGORGANISASI SEMUANYA

Sekarang kita memiliki potongan kode SRP yang indah, Subwayesque, mirip Lego, kita harus mulai memikirkan cara membuat semuanya berfungsi dengan baik bersama-sama. Untungnya, cara kita menguraikan bangunan rumah kita cukup sesuai dengan urutan ideal untuk membentuk style sheet kita.

Beberapa tahun yang lalu, sebelum saya mulai memikirkan tentang SRP dan CSS granular, saya sering menata style sheet saya untuk secara kasar mencerminkan struktur halaman pada umumnya. Mungkin terlihat seperti ini:

1. Setel ulang
2. Gaya tingkat tinggi (latar belakang tubuh, dll.)
3. Gaya tajuk
4. Gaya halaman (konten, subkonten, dll.)
5. Isi (bentuk, tipografi, gambar, tabel)
6. Gaya catatan kaki

Ini jelas merupakan penyederhanaan besar, tapi saya benar-benar memesan style sheet tanpa memahami bagaimana CSS disusun, dan tanpa memperhatikan kaskade atau warisan. Sejak

mulai bekerja di situs yang lebih besar, saya menemukan bahwa cara paling efektif untuk mengurutkan kumpulan aturan adalah dengan urutan warisan. Artinya, setiap aturan yang ditetapkan harus mewarisi dan menambahkan aturan sebelumnya. Anda mulai dengan hal-hal paling mendasar, fondasi Anda, dan Anda menambahkan struktur, lalu komponen, lalu desain visual. Kumpulan aturan Anda harus diurutkan dari yang paling umum hingga yang paling tidak umum. Hal ini tidak hanya berarti style sheet Anda lebih waras, mungkin juga berarti style sheet Anda juga jauh lebih kecil dan lebih efisien. Jika setiap aturan yang ditetapkan hanya menambah dan memperluas aturan yang telah ditentukan sebelumnya, maka kemungkinan untuk membatalkan penataan gaya akan jauh lebih kecil, kemungkinan terjadinya masalah kekhususan akan jauh lebih kecil, dan kemungkinan tersandung akan jauh lebih kecil.

Jonathan Snook menulis tentang hal semacam ini dengan cara yang jauh lebih baik daripada yang pernah saya harapkan dalam bukunya SMACSS: Scalable and Modular Architecture for CSS13. Jika Anda belum membaca SMACSS maka berhentilah mendengarkan saya sekarang (saya tidak akan tersinggung, saya berjanji) dan dapatkan salinannya sendiri. Saya sangat yakin ini adalah salah satu publikasi terbaik yang pernah dilihat oleh perkembangan CSS modern.

Jadi, sekarang CSS proyek saya mungkin terlihat seperti ini:

1. Reset/normalisasi.css
2. Sistem jaringan
3. Penataan gaya tingkat elemen: elemen tanpa kelas, seperti judul, daftar, dan elemen html dan isi
4. Komponen dan elemen yang dikelompokkan: elemen navigasi, galeri gambar, tombol, formulir
5. Gaya mengalahkan: hal-hal seperti status kesalahan, tema musiman, dll.

Semua hal ini berlapis satu sama lain, memberikan skalabilitas yang logis dan terencana. Segala sesuatu harus termasuk dalam salah satu kategori tersebut dan ditempatkan secara bijaksana pada tempatnya, mampu memperluas apa pun yang telah terjadi sebelumnya, dan membuka jalan bagi apa pun yang mungkin terjadi berikutnya.

1.8 MENGELOLA SEMUANYA

Sekarang Anda memiliki kerangka proyek besar yang terstruktur dengan baik, bagaimana Anda sebenarnya akan mengelolanya? Saran sederhana saya adalah menggunakan banyak file di banyak direktori, ditambah preprosesor.

Saya tahu banyak orang yang masih ragu mengenai nilai praprosesor, namun kelebihannya adalah Anda dapat menggunakan sebanyak atau sesedikit yang Anda perlukan. Dalam hal pengelolaan proyek besar, menurut saya penggabungan praprosesor (@import) sangat berharga. Anda tidak perlu menggunakan mix-in, atau variabel, atau apa pun, namun kemampuan untuk membagi basis kode Anda menjadi banyak penyertaan kecil benar-benar berguna saat mengerjakan proyek besar.

Saya menggunakan Sass, jadi dengan memiliki file seperti normalize.scss, grids.scss, form.scss, dan sebagainya, saya dapat mengimpor semua yang saya perlukan kapan pun dan, yang terpenting, ke tempat yang saya inginkan. Karena kita sekarang memiliki potongan granular dari SRP CSS, kita cukup memindahkan include kita ke atas dan ke bawah file master Sass kita untuk menyusun ulang seluruh style sheet kita dengan sangat cepat. Sebagai contoh, bayangkan ini adalah file Sass utama kita yang mengimpor CSS seluruh proyek:

```
@import "generic/normalize",
      "base/grids",
     "base/headings",
     "base/forms",
     "gui/page-head",
     "gui/page-foot",
     "gui/image-gallery",
     "gui/date-picker",
     "gui/blog-posts";
```

Yang perlu kita lakukan hanyalah mendorong beberapa baris ke atas atau ke bawah dan kita dapat sepenuhnya merancang ulang hasil kompilasi kita. Kemampuan untuk mengelola seluruh komponen sebagai satu baris, bukan keseluruhannya, membuatnya sangat mudah untuk membuat perubahan besar dengan cepat pada struktur CSS jika diperlukan. Ini juga berarti Anda dapat menghapus potongan CSS yang tidak lagi diperlukan hanya dengan mengomentari salah satu file yang disertakan. Tidak memerlukan galeri gambar lagi? Sederhana:

```
@import "gui/page-foot",
      // "gui/image-gallery",
     "gui/date-picker";
```

Mengatur File

Struktur direktori mungkin akan mencerminkan pemisahan pembangunan rumah dalam kode Anda. Seperti inilah struktur direktori CSS di BSkyB, misalnya:

```
vars.scss
generic/
     *.scss
base/
     *.scss
 objects/
 *.scss
gui/
     *.scss
style.scss
```

Perlu diketahui bahwa hal di atas ditulis berdasarkan urutan penerapan, bukan berdasarkan abjad, seperti yang mungkin biasa Anda lihat. Perintah itu berarti:

1. Kami memiliki file variabel yang menampung hal-hal seperti warna merek dan ukuran font.
2. Kemudian kita memiliki direktori generik dengan reset, clearfiX dan seterusnya.
3. Selanjutnya adalah direktori dasar kita, yang menampung elemen-elemen yang tidak dikelaskan seperti

1. h2s, tabel, dll.

4. Kemudian kita mempunyai rangkaian objek dan abstraksi, seperti objek media.
5. Di atas itu kita meletakkan lapisan GUI kita: carousel, akordeon, header, footer dan sejenisnya.
6. Semuanya kemudian dikompilasi ke dalam file CSS khusus produk, yang dibuat dari file master style.scss Sass kami.

Jadi sekarang kita memiliki direktori file Sass yang sangat terorganisir, masing-masing berisi potongan kecil CSS. Potongan-potongan inilah, yang disusun dengan cara ini, yang akan memungkinkan kita untuk menggabungkan dan pada akhirnya menskalakan CSS kita tanpa batas.

Kesimpulan: Pecahkan kode Anda menjadi bagian-bagian yang lebih kecil dan terpisah yang kemudian dapat ditambahkan, dihapus, digabungkan, dan dilapis dalam urutan yang masuk akal.

Pemilih CSS

Salah satu cara paling sederhana untuk membuat CSS Anda lebih skalabel dan cocok untuk front-end modern yang lebih besar adalah dengan lebih memperhatikan pemilih CSS Anda. Ini benar-benar merupakan kemenangan yang sangat cepat. Pedoman seukuran gigitan untuk penyeleksi CSS yang baik pada dasarnya adalah: beri nama yang baik, sesingkat mungkin, dan jaga kekhususan tetap rendah.

Pemilih CSS, seperti ID dan kelas dan seterusnya, adalah salah satu hal pertama yang Anda pelajari saat memasuki dunia pengembangan front-end. Kemampuan untuk mencocokkan sesuatu seperti `<div id="foo">` dengan #foo {} adalah pengembangan Web. Hal sederhana, saya yakin kita semua setuju.

Namun, penyeleksi CSS, meskipun sederhana, memegang salah satu kunci terbesar untuk menulis UI yang skalabel, modular, dapat diperluas, dan cepat. Semua penyeleksi CSS berdampak, dalam beberapa hal:

- Ketergantungan lokasi
- Portabilitas kode
- Kekokohan
- Kekhususan
- Maksud pemilih
- dan bahkan efisiensi

1.9 ANATOMI PEMILIH CSS

Sebelum melanjutkan, kita harus membiasakan diri dengan anatomi kumpulan aturan CSS. Saya tidak akan membahasnya secara mendetail, tetapi mari kita lihat yang berikut ini:

```
foo .bar .baz {
     float: left;
 }
```

Baris 1–3 keseluruhan blok disebut kumpulan aturan. Baris 1 adalah pemilih (gabungan) kami. Keseluruhan baris 2 adalah deklarasi, terdiri dari float yang merupakan properti, dan kiri yang merupakan nilai. .baz, pemilih terakhir sebelum { adalah pemilih kunci kita. Jika penyeleksi mewakili sebagian besar kunci CSS yang dapat diskalakan, pemilih kunci pasti memegang kunci penyeleksi secara umum.

PEMILIH MANA YANG HARUS ANDA GUNAKAN?

Jawabannya sebenarnya cukup mudah: versi singkatnya adalah kelas. Mereka menawarkan granularitas, spesifisitas rendah dan dapat digunakan kembali, mereka dapat digabungkan, mereka (tentu saja) didukung dengan sangat baik dan mereka hebat!

Jika Anda membagi kode menjadi potongan-potongan modular yang lebih kecil, mematuhi prinsip tanggung jawab tunggal, seperti Lego, maka masuk akal jika style sheet Anda sebagian besar akan terdiri dari kelas-kelas. Saya tidak dapat memikirkan selector yang lebih cocok dengan cita-cita spesifisitas rendah, dapat digunakan kembali, portabilitas, dan kejelasan lebih baik daripada kelas sederhana. Seperti akronimnya: “tetap sederhana, bodoh.” Kelas adalah pemilih yang lugas, sederhana, dan telah teruji yang sesuai dengan kebutuhan arsitek CSS dengan sempurna.

iD

Siapapun yang pernah melihat saya berbicara atau membaca artikel saya akan langsung bisa menebak apa yang akan saya katakan selanjutnya: jangan gunakan ID di CSS. Masalah dengan ID bermacam-macam. Pertama, mereka tidak dapat digunakan kembali. Hanya satu ID yang harus ada pada halaman HTML tertentu: melakukan hal sebaliknya adalah tidak valid. Menariknya, bagaimanapun, CSS Anda akan tetap mencocokkan dan menata gaya semua kemunculan ID yang berulang, namun JavaScript hanya akan mencocokkan elemen pertama yang ditemukannya, lalu berhenti. Inilah sebabnya ID dalam JavaScript bagus dan cepat.

Sekarang, mungkin saja Anda tidak menginginkan apa pun lebih dari sekali pada satu halaman, dan Anda tidak memerlukan pemilih yang dapat digunakan kembali sama sekali. Ini semua baik dan bagus, jadi ID mungkin cocok di sini, tapi seperti yang akan kita lihat, ada kerugian yang jauh lebih besar.

Masalah kedua dan yang terbesar mengenai ID adalah kekhususannya. Jika Anda ingin menulis front-end yang lebih mudah, lebih mudah diperluas, dan seringkali lebih besar, maka Anda selalu ingin menjaga kekhususan Anda serendah mungkin. Bagaimanapun juga, kekhususan adalah alasan mengapa kita merasa !penting.

ID memiliki kekhususan yang jauh lebih tinggi dibandingkan jenis pemilih lainnya (kecuali gaya sebaris). Faktanya, mereka jauh lebih spesifik daripada kelas. Tidak ada jumlah kelas yang dirantai yang sespesifik hanya satu ID.

Peningkatan spesifisitas ini dapat menimbulkan beberapa efek samping yang tidak terduga dan tentu saja tidak diinginkan. Ambil contoh:

```
#sidebar {
     background-color: #09f;
 }

 #sidebar,
     #sidebar a {
          color: #fff;
 }
.twitter-widget {
     background-color: #fff;
 }
.twitter-widget,
     .twitter-widget a {
          color: #09f;
 }
```

Di sini kita hanya memiliki sidebar (#sidebar {}) yang memiliki latar belakang biru (#09f) dan teks serta link apa pun di dalamnya berwarna putih. Kami juga memiliki widget Twitter yang dapat digunakan kembali (.twitter {}) yang memiliki latar belakang putih dan teks serta tautannya berwarna biru, kebalikan dari sidebar.

Menurut Anda apa yang akan terjadi jika kita meletakkan widget Twitter di sidebar? Ya, tautannya akan menjadi putih pada latar belakang widget yang putih. Tentu bukan hal yang kita inginkan. Hal ini terjadi karena #sidebar yang merupakan pemilih {} jauh lebih spesifik daripada .twitter yang merupakan pemilih {}, sehingga mengalahkannya. Di sinilah ID bisa menjadi sangat menyusahkan.

Apa pun yang dapat Anda lakukan dengan ID, dapat Anda lakukan dengan kelas, dan banyak lagi. Jika Anda ingin menggunakan komponen satu kali pada suatu halaman, Anda dapat melakukannya dengan sebuah kelas. Jika Anda ingin menggunakannya ribuan kali pada satu halaman, Anda dapat melakukannya dengan kelas. Kelas memiliki semua manfaat ID yang sama tetapi tidak ada kekurangannya.

Tetapkan Pemilik Anda Singkat

Ketika saya mengatakan jaga agar penyeleksi Anda tetap pendek, yang saya maksud bukan namanya, yang saya maksud adalah ukuran pemilih gabungan. Pemilih CSS gabungan adalah pemilih yang terdiri dari penyeleksi yang lebih kecil, misalnya:

```
.a-css-selector {}
 #a .compound .css [selector] {}
```

Hindari penyeleksi majemuk sedapat mungkin. Ada sejumlah alasan mengapa hal ini masuk akal.

Ketergantungan Lokasi

Penyeleksi bersarang atau gabungan kemungkinan besar menggunakan banyak penyeleksi turunan, memilih sesuatu yang hidup di dalam benda lain. Misalnya:

```
.sidebar .button {}
```

Satu masalah di sini adalah gaya .button sekarang hanya dapat digunakan di sidebar Anda. Mungkin ada saatnya klien ingin menggunakan tombol yang sama di tempat lain, namun mereka tidak bisa. Untuk melakukan itu, Anda perlu menulis beberapa CSS lagi:

```
.sidebar .button,
 .footer .button {}
```

Hal ini mungkin tidak tampak terlalu buruk pada awalnya, namun jelas ini bukanlah solusi yang mudah dipertahankan, terukur, atau tahan terhadap masa depan; Anda harus terus menambahkan lebih banyak CSS untuk mencapai tujuan Anda. Ini jauh dari ideal. Mengikat pemilih ke suatu lokasi mengurangi cakupannya untuk digunakan kembali.

Pemilih yang jauh lebih baik adalah:

```
.button--secondary {}
```

Ini sekarang bisa hidup di mana saja tanpa kita harus menyentuh (dan mengasapi) CSS kita. Semua penyeleksi harus sebebas mungkin bergerak. Anda mungkin tidak ingin memindahkannya, namun tidak ada gunanya mengikatnya jika tidak perlu.

Portabilitas

Kita baru saja membahas bagaimana kita dapat memindahkan elemen DOM kita dengan lebih bebas tanpa penyeleksi turunan, namun kita juga dapat meningkatkan elemen DOM mana yang dapat kita terapkan penyeleksinya. Ambil contoh:

```
input.button {}
```

Ini sepertinya sedikit CSS yang tidak mencolok, bukan? Salah! Kita tidak boleh mengkualifikasi penyeleksi kita dengan elemen.

Bayangkan kita ingin menerapkan gaya .button pada tautan. Itu tidak akan berhasil karena kita telah mengikat pemilih kita ke elemen masukan. Dengan menghilangkan pemilih kualifikasi utama ini, kami langsung membuka kemungkinan penerapan kelas ke beragam elemen HTML. Kapan pun Anda melihat sesuatu seperti berikut:

```
ul.nav {}
div.header {}
p.comment {}
a.button {}
```

…dan seterusnya, bertujuan untuk menulis ulang menjadi:

```
.nav {}
.header {}
.comment {}
.button {}
```

Kekokohan

Jika Anda memiliki penyeleksi panjang dengan banyak hal yang terjadi di dalamnya, maka masuk akal bahwa secara statistik, ada kemungkinan lebih tinggi terjadinya kesalahan. Mari kita lihat sebuah contoh:

```
section.content p:first-child {
 font-size: 1.333em;
 }
```

Di sini kita memiliki empat pemilih yang lebih kecil dalam satu pemilih gabungan. Artinya, ada empat tempat yang berpotensi untuk memecahkan kode ini. Jika kita mengubah bagian tersebut menjadi artikel, pemilihnya akan rusak. Jika kita mengubah kelas .content menjadi .page, pemilih akan rusak. Jika kita memasukkan elemen sebelum paragraf itu, pemilihnya akan rusak. Pada dasarnya, ada banyak kesalahan karena ukuran dan jumlah bagian dalam pemilih ini.

Pengganti yang jauh lebih kuat adalah dengan menggunakan:

```
.intro {
 font-size: 1.333em;
 }
```

Tidak ada yang salah di sini. Satu-satunya cara agar pemilih ini tidak berfungsi adalah dengan menghapusnya seluruhnya dari markup, dan jika kami melakukannya maka kami bermaksud untuk merusaknya. Memperpendek penyeleksi membuat mereka jauh lebih kuat, semata-mata dengan menurunkan peluang statistik bahwa mereka dapat dipatahkan.

Kurangi Spesifisitas

Seperti yang saya sebutkan sebelumnya, kekhususan adalah mimpi buruk yang mutlak. Kekhususan inilah yang menyebabkan CSS menjadi tidak terkendali. Kekhususan adalah alasan

mengapa kita memiliki !penting. Untungnya, dengan mempersingkat penyeleksi, Anda menjaga spesifisitas Anda tetap rendah. Ini adalah tempat yang bagus untuk dikunjungi!

Selain menghindari ID, kita juga perlu menghindari penambahan apa pun yang tidak perlu ke pemilih kita. Apa pun yang dapat dihapus dari pemilih harusnya. Selain menurunkan portabilitas dan ketahanan, menambah panjang pemilih juga meningkatkan spesifisitas ini adalah kondisi terburuk yang pernah ada.

Prinsip Tanggung Jawab Tunggal

Kami kembali ke SRP lagi! Kini setelah kita memiliki short selector kecil berbasis kelas ini, kita dapat menggabungkannya dengan lebih mudah satu sama lain. Ini adalah keuntungan lain dari mempersingkat penyeleksi kami. Sifatnya yang kecil, hanya satu pekerjaan dan satu pekerjaan saja membuat mereka sangat mudah untuk digabungkan, ditambah, dan dikurangi satu sama lain.

Kesimpulan: Buat penyeleksi Anda sesingkat mungkin, pertahankan spesifisitasnya tetap rendah, jaga agar penyeleksinya dapat digabungkan, pertahankan di SRP dan pastikan penyeleksinya memiliki maksud pemilih yang baik.

Konvensi Penamaan

Dengan lusinan kelas SRP yang telah diabstraksikan ini, kita memerlukan cara yang layak, konsisten, dan bisa diterapkan untuk memberi nama pada kelas tersebut. Anda akan terkejut betapa merepotkannya kelas yang diberi nama buruk, tetapi saya tidak menyarankan untuk mencari tahu dengan cara yang sulit. Seperti yang pernah dikatakan Phil Karlton:

Hanya ada dua hal sulit dalam Ilmu Komputer: pembatalan cache dan penamaan sesuatu. Untuk waktu yang lama, kita telah diberitahu bahwa nama kelas kita harus semantik. Seperti yang telah kita bahas sebelumnya, ini semua adalah hal yang bodoh: hanya manusia yang memahami kelas, jadi tulislah kelas untuk manusia.

Memberi daftar kelas .blog-posts adalah mubazir. Kami dapat mengetahui dari kontennya bahwa ini adalah daftar postingan blog sehingga tidak perlu menyatakannya di kelas Anda, yang hanya sekedar gaya saja. Daripada membatasi diri sendiri untuk hanya menggunakan kelas ini untuk menata daftar postingan blog, mengapa tidak memberi nama yang lebih netral? Anda kemudian dapat menerapkannya ke daftar mana pun: daftar produk, daftar item berita terbaru, daftar nama pengguna, apa saja.

Jaga agar nama kelas Anda tetap relevan namun netral, masuk akal namun portabel. Daripada menulis nama kelas yang mendeskripsikan konten, cobalah menulis nama yang dapat diterapkan pada semua jenis konten, dan yang menggambarkan perlakuan visual yang akan ditambahkan oleh kelas. Kelas menulis yang mendeskripsikan konten adalah mubazir dan tidak membantu siapa pun.

Hal yang berguna adalah mengetahui bahwa suatu kelas tidak terikat pada jenis konten tertentu, dan bahwa kelas tersebut cukup diabstraksi untuk digunakan kembali di tempat lain. Objek media Nicole Sullivan14 adalah contoh sempurna dari cara berpikir ini. Nama kelas yang tidak menyinggung sama sekali jenis kontennya sangat dapat digunakan kembali.

Apa Namanya?

Jadi, kita telah membahas bagaimana kelas kita harus ditulis untuk pengembang, dan mendeskripsikan tujuan penataan gaya, bukan kontennya, dan bagaimana kelas tersebut harus diberi nama senetral mungkin tetapi bagaimana sebenarnya kita memberi nama pada kelas tersebut?

Mengutip pengembang Google, Jens O. Meiert: *"Gunakan nama yang sesingkat mungkin namun sepanjang diperlukan"*. Kelas yang berhubungan dengan pola desain umum dan abstraksi harus tidak jelas, abstrak dan netral. Objek media, sekali lagi, adalah contoh sempurna dari hal ini. Saya mendapat ide tentang objek pulau: pola desain yang tugasnya hanya membuat kotak empuk. Alasan saya memilih "pulau" adalah karena dapat diterapkan pada segala macam elemen. Pulau adalah sesuatu yang terpisah dari lingkungannya di semua sisi; objek pulau menghilangkan beberapa konten.

Untuk abstraksi dan pola desain tingkat tinggi, gunakan nama abstrak yang cocok untuk digunakan kembali, dan jangan terlalu menyesuaikan diri dengan jenis konten tertentu. Hubungan yang tidak jelas dan abstrak ini memungkinkan portabilitas yang lebih besar dan penggunaan kembali kelas Anda.

Untuk komponen tertentu (seperti carousel, atau header atau akordeon), Anda perlu menggunakan nama yang spesifik dan tidak ambigu yang memberi tahu pengembang banyak hal tentang komponen dari kelas di markup saja. Mari kita gunakan contoh singkat:

```
<div class="widget foobar-widget">
 <h2 class="widget-title"></h2>
 <div class="widget-body">
 <img src="" alt="" class="widget-thumbnail">
 </div>
 </div>
```

Di sini kita dapat melihat bagaimana semua kelas kita diberi nama dengan baik, sangat eksplisit dan jelas. Melihat kelas-kelas ini memberi tahu kita banyak hal tentang komponen: komponen tersebut dirancang untuk memiliki judul dan gambar, misalnya. Tapi tunggu, masih ada lagi…

1.10 BEM: BLOK, ELEMEN, MODIFIER

BEM artinya blok, elemen, pengubah adalah metodologi penamaan front-end yang dipikirkan oleh orang-orang di Yandex, yang berbasis di Rusia. Ini adalah cara cerdas untuk memberi nama kelas CSS Anda agar lebih transparan dan bermakna bagi pengembang lain. Mereka jauh lebih ketat dan informatif, sehingga membuat konvensi penamaan BEM ideal untuk tim pengembang pada proyek besar yang mungkin memerlukan waktu lama.

Cita rasa BEM yang saya gunakan sebenarnya merupakan modifikasi yang dipikirkan oleh Nicolas Gallagher, dan mengikuti pola berikut:

```
.block {}
```

```
.block__element {}
.block--modifier {}
```

- .block mewakili tingkat abstraksi atau komponen yang lebih tinggi.
- Elemen .block mewakili turunan dari .block yang membantu membentuk.block secara keseluruhan.
- .block--modifier mewakili status atau versi .block yang berbeda.

Mari kita tulis ulang contoh sebelumnya dengan BEM:

```
<div class="widget widget--foobar">
 <h2 class="widget__title"></h2>
 <div class="widget__body">
 <img src="" alt="" class="widget__thumbnail">
 </div>
 </div>
```

Di sini kita dapat melihat blok itu adalah .widget; .widget--foobar merupakan modifikasi dari widget tersebut; dan .widget title, .widget body, dan .widget thumbnail merupakan elemen widget.

Keuntungan di sini mungkin tidak langsung terlihat, tapi sekarang ambil contoh serupa:

```
<div class="widget foobar-widget">
      <div class="media">
           <img src alt class="img thumbnail">
           <div class="body">
                <h1 class="heading"></h1>
                    <p></p>
           </div>
      </div>
 </div>
```

Bagaimana hubungan kelas .img dan .widget satu sama lain? Apakah mereka berhubungan satu sama lain? Bagaimana dengan .media dan .thumbnail? Nah, jika kita menulis ulang dengan BEM:

```
<div class="widget widget--foobar">
      <div class="media">
      <img src alt class="media__img widget__thumbnail">
          <div class="media__body">
               <h1 class="widget__heading"> </h1>
               <p> </p>
```

```
            </div>
        </div>
    </div>
```

Di sini kita dapat langsung melihat bagaimana semua kelas ini berhubungan satu sama lain. Jika kita ingin menghapus semua gaya yang terkait dengan widget, kita dapat dengan cepat melihat kelas terkait. BEM memberi pengembang gambaran yang sangat mendetail tentang keseluruhan markup hanya dengan melihat sekilas kelas-kelasnya.

BEM memang terlihat jelek, dan bertele-tele, namun kekuatan yang diberikannya jauh melebihi dampak negatifnya. Jika Anda mengupayakan kode yang cantik dibandingkan kode yang kuat, saya yakin Anda berfokus pada hal yang sepenuhnya salah.

KELAS LONGGAR

Kami telah membahas dua jenis kelas sebenarnya di sini. Pertama, kita melihat kelas-kelas yang sangat abstrak dan sangat dapat digunakan kembali dengan nama-nama yang sangat abstrak yang tidak sesuai dengan sesuatu yang khusus. Kedua, di sisi lain, kita melihat kelas-kelas yang melakukan pekerjaan tertentu dan dengan demikian menerima nama yang eksplisit, bertele-tele, dan diberi spasi nama. Menggabungkan keduanya adalah ide yang sangat buruk. Memberi komponen tertentu kelas yang sangat longgar sangatlah tidak bijaksana.

Nama kelas yang longgar adalah nama yang tidak diberi nama secara eksplisit sesuai tujuan yang dimaksudkan. Bayangkan Anda bekerja di situs besar dengan banyak pengembang dan Anda melihat kelas .card . Apa fungsinya? Nama kelas seperti itu sangat longgar, dan nama kelas yang longgar sangat buruk karena dua alasan utama.

Pertama, Anda tidak bisa serta merta mengetahui tujuannya dari kelas saja (apakah itu kartu profil pengguna, kartu elektronik, kolom dalam formulir yang menerima nomor kartu kredit?). Kedua, hal ini sangat tidak jelas sehingga dapat dengan mudah didefinisikan ulang secara tidak sengaja oleh pengembang lain. Kelas yang longgar itu seperti variabel global: sulit untuk dilacak, mudah (secara tidak sengaja) ditimpa, dan mudah untuk menyelinap ke dalam pekerjaan Anda di tempat yang tidak Anda duga.

Nama kelas yang longgar bagi CSS sama dengan cakupan global bagi JavaScript. Variabel dalam lingkup global dapat secara tidak sengaja dipindahkan atau secara tidak sengaja dimasukkan ke bagian program yang tidak terkait. Inilah sebabnya mengapa pengembang menghindari cakupan global jika memungkinkan. Dengan memperhatikan nama kelas, dan menghindari penggunaan penyeleksi yang longgar untuk pekerjaan tertentu, kita dapat mengontrol cakupan penyeleksi dengan lebih baik.

Rekap

Jadi, mari kita ulas. Abstraksi tingkat tinggi harus memiliki nama yang sangat abstrak. Mereka tidak boleh menyesuaikan diri dengan konten tertentu, dan makna yang mereka sampaikan harus mengacu pada pola visual daripada mendeskripsikan konten yang terpengaruh.

Komponen tertentu harus memiliki konvensi penamaan yang sangat eksplisit, dan memanfaatkan hal-hal seperti BEM dapat sangat membantu dalam hal ini. Kelas-kelas seperti itu, karena mereka biasanya bekerja sama dengan kelas-kelas lain, harus memberikan gambaran umum yang bermanfaat dan mendalam kepada pengembang tentang kode yang mereka kerjakan.

Terakhir, Anda sebaiknya tidak menggabungkan keduanya. Jika suatu kelas melakukan pekerjaan tertentu, kelas tersebut harus memiliki nama yang sama eksplisitnya (misalnya, nama profil pengguna, bukan .nama).

Kesimpulan: Pastikan nama yang Anda gunakan untuk penyeleksi sudah sesuai, dan nama tersebut memaparkan informasi sebanyak mungkin. Jangan khawatir bertele-tele dengan kelas (saya baru-baru ini menulis kelas .accordion__ trigger--closed), dan beri nama sehingga pengembang lain akan menganggapnya praktis dan masuk akal untuk digunakan.

Verbositas dalam HTML

Mari kita bicara tentang gajah di dalam ruangan. Semua pembicaraan tentang granularitas, abstraksi, dan objek tentu berarti lebih banyak kelas dalam markup kita, dan lebih banyak HTML yang bertele-tele, bukan? Benar. Banyak orang enggan menggunakan lebih banyak kelas dalam HTML mereka karena kesalahpahaman seputar semantik dan markup yang bersih. Seperti yang telah kita bahas, hal ini dapat dimengerti tetapi sedikit salah kaprah. Alasan utama untuk menghindari penggunaan terlalu banyak kelas adalah kemampuan pemeliharaan. Hal ini penting, namun perlu adanya pragmatisme dan keseimbangan.

Sehubungan dengan overhead pemeliharaan lebih banyak kelas, kita sering memikirkan biaya karena harus mengubah kelas di banyak file HTML, tampilan, atau templat. Oleh karena itu, menghindari menambahkan lebih banyak kelas berarti kita memiliki lebih sedikit perubahan pada file HTML jika diperlukan. Tentu saja ini sangat benar. Lebih banyak kelas berarti lebih banyak HTML yang berpotensi dipertahankan.

Namun, menghindari kelas berarti Anda mungkin menulis CSS yang kurang terperinci, sehingga CSS Anda tidak terpisah-pisah dan akan lebih sulit untuk dipertahankan. Anda cukup memindahkan overhead pemeliharaan dari HTML ke CSS Anda, dan mengubah beberapa kelas dalam markup Anda jauh lebih mudah daripada memfaktorkan ulang style sheet yang kusut.

Sebagai persiapan untuk ceramah yang saya berikan pada pertengahan tahun 2013, saya memutuskan untuk melakukan eksperimen yang sangat lo-fi dan tidak ilmiah. Di tempat kerja saya telah membangun situs yang cukup besar selama enam bulan sebelumnya, berisi banyak file tampilan menggunakan banyak kelas. Saya harus membuat perubahan besar yang melibatkan pengeditan kelas yang muncul di banyak tempat di banyak file tersebut. Saya memutuskan untuk mengatur waktu proses ini untuk melihat seberapa besar upaya yang diperlukan dalam mengubah beberapa kelas markup yang digunakan di seluruh situs. Butuh waktu dua belas menit. Itu saja. Saya kehilangan waktu berhari-hari untuk memperbaiki CSS yang rusak. Seharian penuh. Dengan menggunakan kelas-kelas tambahan ini, saya memiliki CSS yang sangat terperinci dan menyenangkan untuk digunakan, dan membuat perubahan di seluruh proyek ini membutuhkan waktu kurang dari seperempat jam.

Alasan mengapa hal ini begitu cepat sederhana saja: otomatisasi. Anda dapat mengotomatiskan pencarian dan penggantian tetapi Anda tidak dapat mengotomatiskan pemfaktoran ulang CSS. Proses saya sederhana. Saya baru saja berlari:

```
$ git grep "$CLASS_I_WANTED_TO_FIND"
```

Ini memberi saya daftar semua file yang berisi kelas itu di proyek saya. Saya kemudian membuka semua file ini di editor teks saya dan menjalankan pencarian dan penggantian global. Mudah.

Biaya tambahan dari kelas-kelas tambahan ini tidak seberapa dibandingkan dengan upaya yang dilakukan dalam pengerjaan ulang CSS. Ambil cara mudah dengan menambahkan lebih banyak kelas dan gunakan otomatisasi (grep, ack, sed) saat Anda perlu mengubahnya.

Hal lain yang sering saya katakan dalam situasi ini adalah: Anda adalah seorang pengembang Web, tugas Anda adalah memelihara kode. Daripada mengeluh karena harus mengubah beberapa kode yang Anda tulis, mulailah mencari cara untuk melakukannya dengan lebih cepat.

Jadi jika kelas tidak berdampak pada semantik, dan tidak ada yang peduli tentang seberapa bersih markup Anda, dan semakin banyak kelas tidak terlalu sulit untuk dipertahankan, saya pikir inilah saatnya kita lebih menerapkannya.

Sekarang, saya menganjurkan penggunaan lebih banyak kelas tetapi, tentu saja, hal ini dapat diambil terlalu jauh:

```
<div class="red-text brand-face rounded margin-bottom">
```

Tingkat granularitas ini akan segera membengkak dan menjadi mimpi buruk yang harus dipertahankan. CSS Anda akan terlalu terperinci dan tidak koheren. CSS dan HTML Anda akan menjadi lebih sulit untuk dirawat.

Dimana Menggambar Garis

Sayangnya, sangat sulit untuk mengidentifikasi titik di mana kita harus mulai menggunakan lebih banyak atau berhenti menggunakan begitu banyak kelas dalam HTML kita. Tes lakmus kecil saya adalah jika sesuatu harus melakukan N hal, maka harus ada N pengait yang diterapkan padanya. Seperti halnya prinsip tanggung jawab tunggal, Anda dapat menghilangkan berbagai aspek komponen hanya dengan menghapus kelas terkait. Mari kita ambil contoh:

```
<a href="" class="btn btn--purchase btn--full js-button"
id="js-purchase-button" data-user-id="2893348">Purchase</a>
```

Kita dapat dengan cepat dan jelas melihat markup ini melakukan beberapa hal. Pertama, ini adalah tautan (<a>) dan ditata seperti tombol (.btn), khususnya tombol pembelian (.btn--beli) lebar penuh (.btn--penuh).

Kedua, kita juga dapat melihat bahwa kita mengikatnya melalui JavaScript karena itu adalah sebuah tombol (.js-button), dan juga karena itu secara khusus adalah tombol pembelian (#js-purchase-button). Terakhir, kita dapat melihat bahwa tombol tersebut memiliki atribut data (data-user-id="2893348") yang menyimpan beberapa data tentang pengguna. Markup ini memang terlihat sangat bertele-tele, namun hanya bertele-tele sesuai kebutuhan. Kami mungkin bisa menulis markup itu seperti ini:

```
<a href="" id="purchase-button" data-user-id="2893348">Purchase</a>
```

Kami dapat melampirkan semua gaya kami dan mengikat semua JavaScript kami ke satu ID tersebut, dan masih mengakses atribut data seperti biasa. Ini akan berhasil, dan jauh lebih ringkas, namun sangat tidak dapat digunakan kembali. Kami telah melampirkan terlalu banyak fungsi ke satu pemilih yang terlalu spesifik yang berarti CSS kami tidak terlalu KERING (jangan ulangi lagi) dan kami tidak akan dapat dengan mudah membuat perubahan terpisah pada HTML tersebut (seperti menghapus pengikatan JavaScript, misalnya).

Prinsip tanggung jawab tunggal telah dicoba dan diuji, dan menjaga tugas kode Anda tetap terperinci dan terpisah membuat aplikasi Anda jauh lebih fleksibel, serta lebih cepat untuk digunakan.

Kesimpulan: HTML lebih mudah diperbarui daripada CSS untuk difaktorkan ulang. Jika sebuah markup harus melakukan lima pekerjaan, ia memerlukan lima hook, baik itu kelas, atribut data, atau apa pun.

BAB 2
PENULISAN KODE YANG DAPAT DIPERTAHANKAN

2.1 KONVENSI KODE

Ketika saya belajar ilmu komputer di perguruan tinggi, saya mempunyai seorang profesor yang sangat tangguh. Namanya Dr. Maxey dan dia mengajar mata kuliah yang lebih rumit seperti struktur data dan arsitektur komputer. Dia adalah seorang guru yang luar biasa dengan bakat mengartikulasikan konsep-konsep sulit, tetapi juga seorang siswa kelas yang sangat tangguh. Dia tidak hanya akan memeriksa kode Anda untuk memastikan kode tersebut berfungsi, dia juga akan mengurangi poin untuk masalah gaya. Jika kode Anda tidak memiliki komentar yang sesuai, atau bahkan jika komentar berisi satu atau dua kata yang salah eja, ia akan mengurangi poin. Jika kode Anda berantakan (menurut standarnya), dia akan mengurangi poin. Pesannya jelas: kualitas kode Anda tidak hanya dalam eksekusinya tetapi juga dalam penampilannya. Itu adalah pengalaman pertama saya dengan gaya coding.

Gaya Apa Itu?

Gaya pengkodean adalah tampilan kode Anda, polos dan sederhana. Dan yang saya maksud dengan "milik Anda" adalah kode yang ditulis oleh Anda, orang yang membaca bab ini. Gaya pengkodean sangat pribadi dan setiap orang memiliki pendekatan pilihannya sendiri. Anda dapat menemukan gaya pribadi Anda dengan melihat kembali kode yang Anda tulis ketika Anda tidak memiliki panduan gaya untuk dipatuhi. Setiap orang memiliki gayanya masing-masing karena cara mereka belajar coding. Jika Anda menggunakan IDE seperti Visual Studio untuk mempelajari pengkodean, gaya Anda mungkin cocok dengan yang diterapkan oleh editor. Jika Anda belajar menggunakan editor teks biasa, kemungkinan besar gaya Anda berevolusi dari apa yang Anda anggap lebih mudah dibaca. Anda bahkan mungkin memperhatikan bahwa gaya Anda berubah dari satu bahasa ke bahasa lainnya. Keputusan yang Anda buat di JavaScript mungkin tidak terbawa ke CSS Anda. Misalnya, Anda mungkin memutuskan string JavaScript harus menggunakan tanda kutip ganda sedangkan string CSS harus menggunakan tanda kutip tunggal. Hal ini biasa terjadi karena kita cenderung berpindah konteks saat berpindah-pindah bahasa. Tetap saja, ini merupakan latihan observasi diri yang menarik.

Gaya pengkodean terdiri dari banyak keputusan kecil berdasarkan bahasa:

- Bagaimana dan kapan menggunakan komentar,
- Tab atau spasi untuk lekukan (dan berapa banyak spasi),
- Penggunaan ruang putih yang tepat,
- Penamaan variabel dan fungsi yang tepat,
- Kode yang mengelompokkan suatu organisasi,
- Pola yang harus digunakan dan pola yang harus dihindari.

Ini bukanlah daftar yang lengkap, karena gaya pengkodean bisa sangat terperinci, seperti Google JavaScript Style Guide1, atau lebih umum, seperti jQuery Core Style Guidelines.

Sifat Pribadi

Sifat pribadi dari gaya pengkodean merupakan tantangan dalam suasana tim. Seringkali, untuk menghindari perdebatan panjang, tim menunda pembuatan panduan gaya dengan alasan tidak ingin menghambat inovasi dan ekspresi. Beberapa orang melihat panduan gaya yang ditentukan tim sebagai cara untuk memaksa semua pengembang untuk bersikap sama.

Beberapa pengembang memberontak ketika diberikan panduan gaya, percaya bahwa mereka tidak dapat melakukan tugasnya dengan baik jika seseorang memberi tahu mereka cara menulis kode.

Saya mengibaratkan situasinya seperti sekelompok musisi yang mencoba membentuk sebuah band. Masing-masing orang percaya bahwa cara mereka melakukan sesuatu adalah yang terbaik (metode atau proses mereka). Band ini akan kesulitan selama semua orang berusaha melakukan hal mereka sendiri. Tidak mungkin menciptakan musik yang bagus kecuali semua orang di band sepakat mengenai tempo, gaya, dan siapa yang harus memimpin dalam sebuah lagu. Siapa pun yang pernah mendengar pertunjukan band sekolah menengah tahu bahwa ini benar. Kecuali semua orang mempunyai pemikiran yang sama, Anda tidak akan mencapai banyak hal. Itu sebabnya saya sangat merekomendasikan panduan gaya untuk tim pengembangan perangkat lunak. Sulit untuk membuat semua orang memiliki pemikiran yang sama, dan panduan gaya adalah tempat yang bagus untuk memulai. Dengan meminta semua orang menulis kode yang terlihat sama, Anda dapat menghindari banyak masalah.

2.2 KOMUNIKASI ADALAH KUNCI

Program dimaksudkan untuk dibaca oleh manusia dan hanya secara kebetulan untuk dijalankan oleh komputer.

— Harold Abelson dan Gerald Jay Sussman,
Struktur dan Interpretasi Program Komputer, 1984

Hal terpenting ketika bekerja dalam tim adalah komunikasi.

Orang-orang harus dapat bekerja sama secara efektif dan satu-satunya cara untuk melakukannya adalah dengan berkomunikasi. Sebagai pengembang, kami berkomunikasi terutama melalui kode. Kami berkomunikasi dengan bagian lain dari perangkat lunak melalui kode dan kami berkomunikasi dengan pengembang lain melalui kode.

Meskipun perangkat lunak yang berkomunikasi dengan kode Anda tidak peduli dengan tampilan kodenya, pengembang lain di tim Anda pasti peduli. Tampilan kode menambah pemahaman kita tentangnya. Berapa kali Anda membuka sepotong kode yang ditulis orang lain dan, sebelum melakukan hal lain, memasukkannya kembali ke dalam sesuai keinginan Anda? Itu berarti otak Anda tidak dapat memahami kodenya karena tampilannya. Ketika setiap orang menulis kode yang terlihat berbeda, setiap orang terus-menerus mencoba mengurai kode tersebut sebelum dapat memahaminya. Ketika semua orang menulis kode yang terlihat sama, otak Anda bisa sedikit rileks karena pemahamannya menjadi lebih cepat.

Ketika Anda mulai menganggap kode sebagai komunikasi dengan pengembang lain, Anda mulai menyadari bahwa Anda tidak sekadar menulis kode, Anda sedang membuat kode. Anda memberikan pemikiran ekstra pada setiap penekanan tombol karena kode tersebut tidak lagi hanya dapat dieksekusi oleh komputer, namun juga dapat dipahami oleh orang lain. Kode Anda harus mengkomunikasikan tujuannya dengan jelas kepada pengamat biasa. Ingatlah, kode Anda ditakdirkan untuk dikelola oleh orang lain selain Anda. Anda tidak hanya berkomunikasi dengan anggota tim Anda yang lain saat ini, Anda juga berkomunikasi dengan anggota tim Anda di masa depan.

Saya baru-baru ini menerima email dari seseorang yang mengerjakan kode yang saya tulis 10 tahun lalu. Rupanya, saya terkejut dan ngeri, kode saya masih digunakan di produk. Dia merasa terdorong untuk mengirim email kepada saya untuk mengatakan bahwa dia menikmati bekerja dengan kode saya. Aku tersenyum. Rekan satu tim saya di masa depan sebenarnya mengapresiasi gaya pengkodean yang saya ikuti.

Tinggalkan Petunjuk Diri Dengan Komentar

Jika Anda mengenal musuh Anda dan mengenal diri Anda sendiri, Anda tidak akan terancam dalam ratusan pertempuran.

—Sun Tzu, Seni Perang

Mengenal diri sendiri penting dalam kehidupan dan juga coding. Namun, Anda tidak akan pernah cukup mengenal diri sendiri untuk mengingat dengan tepat apa yang Anda pikirkan saat menulis setiap baris kode. Sebagian besar pengembang telah berpengalaman melihat potongan kode lama yang mereka tulis dan tidak tahu mengapa mereka menulisnya. Bukan berarti ingatan Anda buruk, hanya saja Anda membuat begitu banyak keputusan kecil saat menulis kode sehingga tidak mungkin untuk melacak semuanya.

Menulis kode ke panduan gaya mentransfer informasi tersebut ke dalam kode itu sendiri. Saat Anda memutuskan kapan dan di mana menggunakan komentar, serta pola mana yang boleh dan tidak boleh digunakan, Anda meninggalkan jejak kecil bagi diri Anda di masa depan untuk menemukan jalan kembali ke tujuan kode. Sangat menyegarkan untuk membuka potongan kode lama dan membuatnya terlihat seperti potongan kode baru. Anda dapat menyesuaikan diri dengan cepat, menghindari proses yang membosankan dalam mempelajari kembali fungsi kode sebelum Anda dapat mulai menyelidiki masalah sebenarnya.

Panduan gaya apa pun yang baik menentukan kapan dan bagaimana meninggalkan komentar dalam kode. Sebagian besar pengembang enggan berkomentar karena sepertinya seperti menulis dokumentasi dan pengembang cenderung benci menulis dokumentasi. Hal ini biasanya karena menulis dokumentasi dipandang sebagai waktu, bukan menulis kode. Namun, dokumentasilah yang memungkinkan Anda kembali ke kode yang Anda tulis sebelumnya dan dengan cepat mendapatkan kembali kecepatannya.

Argumen umum lainnya yang menentang penggunaan komentar adalah bahwa kode harus didokumentasikan sendiri. Pada kenyataannya, tidak ada kode yang dapat mendokumentasikan diri sendiri. Kode yang mendokumentasikan diri sendiri adalah mitos yang diabadikan oleh orang-orang yang tidak suka menulis dokumentasi. Argumen umum

yang menentang penulisan komentar adalah, “jika Anda tidak dapat memahami kode saya, maka Anda tidak cukup pintar.” Pada kenyataannya, memahami kode tidak ada hubungannya dengan seberapa pintar seorang pengembang, melainkan memberikan konteks yang cukup sehingga kode tersebut masuk akal. Tanpa komentar, atau diskusi langsung dengan pembuat kode, sangat sulit mendapatkan konteks yang memadai.

Kode yang paling mudah dipahami dipenuhi dengan komentar yang menjelaskan bagian-bagian penting. Anda tentu tidak menginginkan komentar untuk setiap baris kode, dan komentar harus memberikan konteks dan informasi tambahan yang tidak dapat diperoleh dengan membaca kode. Berikut ini contoh komentar buruk:

```
// set initial count to 0
var count = 0;
```

Komentar ini tidak memberikan konteks atau informasi tambahan pada baris kode yang dijelaskannya. Komentar seperti ini harus dihindari bagaimanapun caranya. Komentar yang baik mencakup informasi seperti:

- Deskripsi tentang apa yang dilakukan kode tersebut,
- Mengapa kode tersebut melakukan hal tersebut dengan cara ini (dan bukan dengan cara alternatif),
- Referensi ke bug atau masalah apa pun yang terkait dalam pelacak masalah.

Misalnya, berikut adalah komentar bagus dari basis kode jQuery:

```
// #8138, IE may throw an exception when accessing
// a field from window.location if document.domain has been set

try {
     ajaxLocation = location.href;
          } catch( e ) {

// Use the href attribute of an A element
// since IE will modify it given document.location
     ajaxLocation = document.createElement( "a" );
     ajaxLocation.href = "";
     ajaxLocation = ajaxLocation.href;
}
```

Komentar tersebut merujuk pada nomor masalah dan memberikan deskripsi bug asli. Komentar kedua menjelaskan pendekatan yang memperbaiki masalah tersebut. Sangat mudah bagi siapa pun yang membaca kode untuk memahami mengapa kode ini disertakan dan ke mana harus pergi jika diperlukan informasi lebih lanjut. Berikut contoh bagus lainnya dari perpustakaan YUI CSS:

```
h1 {
/*18px via YUI Fonts CSS foundation*/
 font-size:138.5%;
}
```

Tanpa komentar, Anda mungkin bertanya-tanya mengapa 138,5% adalah angka yang signifikan. Dengan adanya komentar, Anda mengetahui dua informasi penting. Pertama, file ini memerlukan fondasi CSS YUI Fonts agar dapat berfungsi dengan baik. Kedua, 138,5% sama dengan 18pX berdasarkan persyaratan tersebut. Apa yang tadinya bisa menjadi sumber kebingungan kini menjadi sumber informasi dan pemahaman.

Bagaimana Anda tahu jika kode memerlukan komentar? Bayangkan komentar sebagai catatan Post-it dalam kode Anda. Kapan pun Anda takut lupa apa yang dilakukan kode tersebut atau bagaimana cara kerjanya, tambahkan komentar. Kapan pun Anda menemukan sesuatu yang mungkin membuat pengembang lain tersandung, seperti peretasan khusus browser, tinggalkan komentar. Jika Anda menerapkan algoritme tertentu, tinggalkan komentar. Tinggalkan komentar kapan pun Anda merasa kehilangan beberapa informasi penting jika Anda pergi selama enam bulan dan kemudian perlu mengerjakan kodenya lagi.

Pengembang yang baik akan menggunakan komentar dengan bijaksana dan tidak mengharapkan kodenya berbicara sendiri. Anda tidak perlu membaca seluruh kode hanya untuk memahami apa yang terjadi. Komentar mempersingkat kebutuhan dengan memberikan narasi yang lebih ringkas menggambarkan apa yang sebenarnya dilakukan kode. Dan itu sangat berharga untuk kebersihan kode Anda dalam jangka panjang.

Buat Kesalahan Jelas

Salah satu alasan terpenting untuk memiliki panduan gaya yang koheren adalah untuk membantu memperjelas kesalahan. Panduan gaya melakukan ini dengan membiasakan pengembang dengan pola tertentu. Setelah Anda terbiasa, pola asing muncul dari kode saat Anda melihatnya. Pola seperti itu tidak selalu merupakan kesalahan, namun pasti memerlukan pengamatan lebih dekat untuk memastikan tidak ada yang salah.

Misalnya, pertimbangkan pernyataan switch JavaScript. Merupakan kesalahan yang sangat umum jika membiarkan satu kasus jatuh ke kasus lain, seperti ini:

```
switch(value) {
     case 1:
          doSomething();
     case 2:
          doSomethingElse();
   break;
   default:
   doDefaultThing();
}
```

Kasus pertama masuk ke kasus kedua jadi jika nilainya 1, maka doSomething() dan doSomethingElse() akan dieksekusi. Dan inilah pertanyaannya: apakah ada kesalahan di sini? Ada kemungkinan bahwa pengembang lupa menyertakan jeda pada kasus pertama, namun ada kemungkinan juga bahwa pengembang bermaksud agar kasus pertama dimasukkan ke kasus kedua. Tidak ada cara untuk mengetahuinya hanya dengan melihat kodenya.

Sekarang anggaplah Anda memiliki panduan gaya JavaScript yang berbunyi seperti ini: *Semua kasus pernyataan peralihan harus diakhiri dengan break, throw, return, atau komentar yang mengindikasikan fall-through.*

Dilihat dari pedoman ini, pasti ada kesalahan gaya dan itu berarti mungkin ada kesalahan logika. Jika kasus pertama seharusnya masuk ke kasus kedua, maka akan terlihat seperti ini:

```
switch(value) {
     case 1:
          doSomething();
          // falls through
     case 2:
          doSomethingElse();
          break;
     default:
          doDefaultThing();
}
```

Jika kasus pertama tidak seharusnya gagal, maka kasus tersebut harus diakhiri dengan pernyataan seperti break. Dalam kedua kasus tersebut, kode asli tidak cocok dengan panduan gaya dan itu berarti Anda perlu memeriksa ulang fungsionalitas yang diinginkan. Saat melakukannya, Anda mungkin menemukan bug.

Contoh lainnya, anggota salah satu tim saya memutuskan bahwa mereka tidak suka menggunakan tiga nilai untuk padding atau margin di CSS, seperti:

```
.box {
     padding: 5px 10px 6px;
}
```

Konsensusnya adalah bahwa ketiga nilai tersebut tidak secara jelas menunjukkan maksud penulis. Apakah jelas bahwa nilai keempat sengaja dihilangkan? Apakah ini kecelakaan? Meskipun Anda mungkin tidak menganggap ini sebagai kemungkinan kondisi kesalahan, tim saya menganggapnya demikian, jadi kami membuat aturan bahwa Anda boleh memiliki satu, dua, atau empat nilai untuk padding dan margin, tetapi tidak boleh tiga. Dengan begitu, jika kita hanya melihat tiga nilai saja, kita akan tahu bahwa mengabaikan nilai keempat adalah suatu kesalahan.

Jika Anda memiliki panduan gaya, kode yang tampaknya tidak berbahaya akan langsung memunculkan tanda karena gaya tersebut tidak diikuti. Ini adalah salah satu aspek panduan gaya yang paling diabaikan: dengan menentukan seperti apa kode yang benar, Anda akan lebih mudah mengidentifikasi kode yang salah dan oleh karena itu menghindari potensi bug sebelum terjadi.

Gaya pengkodean menangani tampilan kode, yang merupakan langkah pertama yang penting. Langkah selanjutnya adalah mengatur cara kode diatur, dan di situlah arsitektur yang baik berperan.

2.3 ARSITEKTUR

Ada ketegangan yang menarik antara pengembang dan arsitek di sebagian besar organisasi rekayasa perangkat lunak. Pengembang melihat arsitek sebagai ahli teori yang suka menggambar diagram dan membuat pernyataan tentang bagaimana perangkat lunak harus dibangun, tanpa memperhitungkan bahwa dunia ini tidaklah sempurna. Arsitek sering kali dipandang sebagai orang yang tidak mampu mengimplementasikan desainnya sendiri karena kurangnya perspektif di lapangan dan di dalam parit.

Sejujurnya, ada arsitek yang cocok dengan gambaran tersebut, namun arsitek yang baik adalah komoditas tak ternilai yang meningkatkan nilai seluruh organisasi melalui visi dan perspektif tingkat tinggi mereka. Visi dan perspektif seperti itu mengubah sebuah arsitektur.

Peran arsitektur diabaikan di banyak tempat padahal arsitektur merupakan bagian terpenting dari perangkat lunak. Arsitektur yang kokoh:

- Menyediakan cara mudah untuk menyelesaikan tugas-tugas umum.
- Memastikan segala sesuatu mempunyai tempat dan tujuan.
- Memungkinkan kami menambah atau menambah fungsionalitas dengan cepat.
- Mendorong desain perangkat lunak yang baik.

Sebagian besar kode yang sangat tidak dapat dipelihara yang pernah saya temui dalam karier saya dapat ditelusuri kembali ke kurangnya arsitektur yang baik. Tanpa struktur seperti itu, kita akan bingung dimana dan kapan harus melakukan perubahan tertentu. Jika hal ini terjadi, kita akhirnya akan meretas solusi yang bukan merupakan solusinya dan hal ini akan menyebabkan penurunan kualitas kode. Setiap kali seseorang tidak dapat menjawab pertanyaan “Kemana perginya?”, itu berarti kode tersebut berakhir di lokasi yang tidak terduga dan, pada gilirannya, melanggengkan masalah.

Kode memiliki kebiasaan berlipat ganda saat Anda tidak melihatnya. Jika ada saat komponen baru-baru ini melakukan sesuatu dalam basis kode, maka akan segera ada dua contoh pola yang sama. Dua mengarah ke empat, dan ini berlanjut hingga pola itu meresap ke seluruh sistem. Hal ini terjadi ketika pengembang mencari contoh bagaimana orang lain mencapai fungsionalitas tertentu. Ketika mereka menemukan sebuah contoh, contoh tersebut akan disalin ke tempat lain untuk tujuan serupa. Jika polanya bagus, Anda mendapatkan hasil yang diinginkan; jika polanya buruk, maka kode Anda menjadi kurang dapat dipelihara seiring berjalannya waktu.

Itu sebabnya arsitektur terbaik mempunyai tempat untuk segalanya. Kapan pun Anda perlu melakukan perubahan atau penambahan, Anda tahu persis di mana harus melakukannya. YUI, misalnya, memiliki beberapa tipe objek berbeda yang tersedia untuk memperluas perpustakaan. Jika Anda ingin menambahkan fungsionalitas yang benar-benar baru, buatlah modul. Jika Anda ingin menambahkan lebih banyak metode ke elemen DOM, buatlah plugin node. Selalu ada jawaban untuk "Bagaimana cara melakukan ini?" di YUI, dan itu membuatnya mudah untuk dikerjakan dan diperluas.

Ingatlah bahwa sistem YUI berfungsi dengan baik untuk perpustakaan JavaScript, tetapi Anda memerlukan pendekatan yang berbeda tergantung pada apa yang Anda coba buat. Ada arsitektur perpustakaan (seperti sistem plugin YUI dan jQuery), arsitektur kerangka kerja (sistem modul Node.js) dan arsitektur aplikasi (Model-View-Controller atau MVC). Ada arsitektur untuk setiap jenis perangkat lunak dan memilih salah satunya tidak selalu mudah. Di sisi lain, tidak memilih salah satu adalah cara terbaik untuk memastikan kode Anda menjadi tidak dapat dikelola dalam waktu singkat.

Pengembang web biasanya tidak terlalu memikirkan arsitektur, tetapi hal itu mulai berubah. Semakin banyak perpustakaan dan kerangka kerja dengan arsitektur tertentu yang masuk ke dalam proyek. JavaScript dan CSS untuk aplikasi besar, khususnya, telah memperoleh manfaat dari banyak penelitian tentang bagaimana kode harus disusun. Saat ini, ada sejumlah pendekatan dan solusi bawaan untuk membantu Anda membuat aplikasi dengan cara yang logis.

Tulang punggung.js

Backbone.js3 dikreditkan dengan memulai gerakan arsitektur MV di JavaScript. Bukan kerangka MVC tradisional karena kurangnya pengontrol, Backbone.js menyediakan pola umum untuk memanipulasi tampilan (HTML) dengan data terstruktur. Tampilan dapat diperbarui secara otomatis ketika data yang direpresentasikan dalam tampilan berubah. Backbone.js sendiri berukuran cukup kecil dan tidak mewakili keseluruhan arsitektur, namun dapat menjadi landasan yang baik untuk desain yang lebih besar.

JavaScript yang dapat diskalakan

Sebuah pendekatan yang saya rancang untuk arsitektur aplikasi JavaScript lengkap yang berkembang dan berkembang seiring dengan berkembangnya aplikasi Anda. Pendekatan utamanya adalah memisahkan aplikasi menjadi serangkaian bagian kecil, masing-masing dengan tanggung jawab dan batasan tertentu. Arsitekturnya dapat dibangun di atas pustaka JavaScript mana pun dan diperluas melalui penambahan plugin di setiap level. Tidak ada perpustakaan JavaScript tunggal untuk pendekatan ini, karena pendekatan ini disajikan sebagai desain arsitektur tingkat tinggi (meskipun ada banyak implementasi yang dapat ditemukan secara online). Lihat slide dan transkrip saya di SlideShare4 untuk informasi lebih lanjut.

Ember.js

Ember.js5 adalah kerangka aplikasi lengkap yang mencakup semua yang Anda perlukan untuk membangun aplikasi JavaScript. Ini menyediakan kerangka kerja MVC ditambah kemampuan perutean. Ini semua digabungkan dengan Handlebars6, sebuah bahasa

templating. Ember.js memiliki pendapat — ada cara untuk melakukan segalanya: cara Ember.js. Melakukan hal ini memungkinkan Anda untuk fokus membangun aplikasi Anda daripada mengkhawatirkan desain arsitektur di sekitarnya.

CSS Berorientasi Objek

OOCSS adalah pendekatan yang diciptakan oleh Nicole Sullivan untuk membuat CSS lebih mudah dikelola. Ide umumnya adalah membuat objek kecil yang dapat digunakan kembali (kombinasi kelas markup dan CSS) yang mewakili pola umum. Dengan potongan-potongan kecil yang cukup, Anda dapat membuat halaman dalam jumlah tak terbatas yang terlihat berbeda meskipun menggunakan pola dasar yang sama.

Menyusun gaya CSS ke dalam struktur, skin, konten, dan pengelompokan lainnya membawa keteraturan pada CSS. Lihat "Pengantar CSS Berorientasi Objek" karya Louis Lazarus untuk primer yang bagus. Nicole juga menawarkan perpustakaan berdasarkan prinsip-prinsip ini.

Arsitektur Scalable dan Modular untuk CSS

SMACSS9 dirancang oleh Jonathan Snook untuk secara jelas menguraikan tanggung jawab setiap bagian CSS. Dia mengkategorikan aturan ke dalam dasar, tata letak, modul, keadaan dan tema, dan masing-masing kategori memberikan pedoman mengenai properti mana yang dapat digunakan dan untuk tujuan apa. Tidak ada perpustakaan yang sejalan dengan pendekatan ini karena SMACSS adalah deskripsi arsitektur tingkat tinggi dan bukan implementasi spesifik.

Ini hanyalah contoh arsitektur yang tersedia untuk JavaScript dan CSS. Lakukan riset untuk mengetahui arsitektur mana yang paling cocok untuk Anda. Jangan membuat kesalahan dengan terlambat mengkhawatirkan arsitektur itulah resep utang teknis sejak awal. Memilih arsitektur sangat mirip dengan meletakkan fondasi untuk sebuah rumah. Jika fondasinya kuat, Anda bisa membangun apa saja di atasnya; jika pondasi lemah atau tidak ada, maka kualitas seluruh rumah terancam.

Meskipun Anda tidak dapat menemukan arsitektur yang sempurna untuk proyek Anda, pilih saja satu. Memiliki beberapa organisasi jauh lebih baik daripada tidak memiliki organisasi. Ketika Anda memutuskan bagaimana menyusun kode, Anda telah mengambil langkah penting menuju pembuatan basis kode yang berkelanjutan. Setiap kali keputusan penting dibuat, ada baiknya untuk menuliskan cara kerjanya dan mengapa keputusan tersebut dirancang dengan cara ini. Mendokumentasikan semua ini memudahkan pengembang baru untuk bergabung.

Dokumentasi

Dokumentasi adalah bagian pekerjaan yang paling tidak disukai pengembang, namun sering kali sama pentingnya dengan kode itu sendiri. Jika Anda melihat kesuksesan perangkat lunak sumber terbuka utama, Anda biasanya dapat menarik garis lurus antara kesuksesan itu dan kehadiran dokumentasi yang sangat baik. Keberhasilan jQuery sebagian besar berkat dokumentasi luar biasa yang ada di sekitar perpustakaan, yang sebagian besar disumbangkan oleh komunitas yang penuh semangat. Selain itu, pengguna jQuery lainnya membuat blog mereka sendiri dengan tips dan trik, serta tutorial yang berlimpah. Itu bahkan sebelum buku

jQuery mulai bermunculan. Saat ini Anda dapat melakukan pencarian cepat untuk segala hal yang berhubungan dengan jQuery dan menemukan ratusan contoh dan tutorial.

Twitter Bootstrap adalah perpustakaan lain yang mendapat manfaat dari dokumentasi yang sangat baik. Saat Anda tiba di situs tersebut, Anda akan mendapatkan banyak informasi tentang cara memulai. Semua pola didokumentasikan dengan kode dan contoh langsung sehingga Anda dapat melihat dengan tepat apa yang akan Anda dapatkan dengan menerapkan kelas tertentu pada HTML. Popularitas Bootstrap sebagian disebabkan oleh kesederhanaan dalam memulainya, dan hal ini disebabkan oleh dokumentasi berkualitas tinggi.

Ada alasan mengapa perangkat lunak sumber terbuka yang populer meminta dan menghabiskan waktu untuk berkontribusi pada dokumentasinya: jika perangkat lunak sulit digunakan atau terlalu buram, orang tidak akan mempermasalahkannya. Namun pengembang yang sering mengeluh tentang kurangnya dokumentasi pada beberapa perangkat lunak adalah orang yang sama yang mencari alasan untuk tidak menulis dokumentasi untuk perangkat lunak mereka sendiri. Perangkat lunak yang baik adalah perangkat lunak yang terdokumentasi dengan baik, dan perangkat lunak yang buruk hanya memiliki sedikit dokumentasi. Tidak ada kode yang dapat dipelihara yang tidak didokumentasikan.

Bahkan perangkat lunak yang ditulis paling buruk pun menjadi lebih mudah dikelola jika ada dokumentasinya. Dokumentasi ini mengangkat tabir keajaiban di sekitar kode dan memungkinkan pengembang untuk bekerja lebih efektif dengannya. Itu sebabnya tidak ada perangkat lunak yang dianggap lengkap tanpa dokumentasi yang menyertainya. Menulis dokumentasi yang baik tidaklah sulit, ini hanya masalah mentransfer pemikiran Anda ke dalam teks dan gambar. Bergantung pada jenis perangkat lunak yang Anda buat, mungkin masuk akal untuk menyusun dokumentasi Anda dengan cara yang berbeda. Namun, ada beberapa pendekatan umum terhadap dokumentasi yang harus diketahui semua orang.

mulai

Panduan memulai cepat menjelaskan cara memulai dan menjalankan. Ini adalah contoh tradisional “Halo dunia” yang dimiliki banyak perpustakaan. Panduan memulai cepat yang baik menjelaskan cara mendapatkan dan menyiapkan perpustakaan, dan cara segera mulai menggunakan fungsionalitas tersebut. Twitter Bootstrap memiliki panduan memulai yang bagus

Tutorial

Ada kasus penggunaan umum yang sering dibutuhkan pengguna perpustakaan sehingga penting untuk menunjukkan kepada mereka cara menyelesaikan tugas tersebut. Tutorial harus dalam bentuk narasi, menjelaskan setiap langkah proses dan hasil prototipe fungsional pada akhirnya. jQuery memiliki banyak tutorial13 yang ditulis dengan sangat baik.

Dokumentasi API

Jika Anda menawarkan API untuk digunakan orang lain, maka dokumentasi API sangatlah penting. Ini menjelaskan setiap antarmuka publik API dengan sangat detail, termasuk nama fungsi, jenis argumen yang diharapkan fungsi, nilai kembalian, kelas yang tersedia, dan banyak lagi. Perpustakaan YUI memiliki kumpulan dokumentasi API yang sangat baik dan dapat dicari sepenuhnya.

Dokumen Desain

Dokumen desain menjelaskan arsitektur dan opsi yang tersedia dalam beberapa perangkat lunak. Ini sering ditulis sebelum pengkodean dimulai dan diperbarui setelah pengkodean selesai. Dokumen desain menjawab pertanyaan, “Bagaimana cara kerjanya?” Sangat umum untuk melihat diagram serta diskusi seputar filosofi desain, pola desain yang digunakan, dan asumsi yang dibuat oleh perangkat lunak. Chromium, proyek sumber terbuka yang menjadi dasar Google Chrome dan Opera, memiliki serangkaian dokumen desain yang sangat baik.

Ketika Anda mewarisi beberapa kode yang harus mulai Anda pertahankan, dokumen desain untuk kode tersebut harus menjadi tempat pertama yang Anda tuju untuk mendapatkan pemahaman yang baik tentang kode tersebut.

Secara umum, panduan dan tutorial memulai diperlukan saat Anda membuat perpustakaan untuk digunakan orang lain. Dokumentasi API dan dokumen desain bagus, apa pun perangkat lunak yang Anda tulis. Struktur sebenarnya dari dokumen-dokumen ini bervariasi berdasarkan perangkat lunak yang Anda buat dan audiens yang dituju.

Dokumentasi API adalah dokumentasi minimum yang harus dimiliki suatu proyek. Sebagian besar bahasa, termasuk JavaScript dan CSS, memiliki alat untuk menghasilkan dokumentasi dari kode sumber. Generator dokumentasi ini menggunakan komentar yang tertanam dalam kode untuk membuat dokumentasi mandiri (biasanya dalam format HTML) yang menjelaskan kode tersebut. Gaya komentar yang tepat bergantung pada alat yang digunakan. Berikut beberapa alat yang perlu diselidiki:

1. **JSDoc**: generator dokumentasi JavaScript asli. Ia menggunakan komentar gaya JavaDoc untuk membuat dokumentasi API.
2. **YUIDoc**: generator dokumentasi JavaScript dari tim YUI. Itu juga menggunakan komentar gaya JavaDoc untuk membuat dokumentasi API.
3. **Docco**: generator dokumentasi narasi JavaScript. Alih-alih membuat dokumentasi API, alat ini membuat narasi di mana deskripsi kode muncul di sebelah kiri dan kode sebenarnya muncul di sebelah kanan.
4. **KSS**: generator panduan gaya CSS. Mengekstrak komentar di dalam file CSS, SCSS, atau LESS dan menghasilkan panduan gaya dengan contoh keluaran.

Ada generator dokumentasi untuk hampir semua bahasa yang Anda gunakan untuk membangun aplikasi Web. Teliti, pelajari, dan gunakan. Cara terbaik untuk memastikan komentar yang baik dalam kode adalah dengan mengetahui bahwa komentar tersebut akan dimasukkan ke dalam dokumentasi sebenarnya. Saya telah melihat hal ini terjadi beberapa kali: segera setelah dokumentasi mulai dibuat dan hasilnya dapat dilihat semua orang, lebih banyak waktu dihabiskan untuk menyusun komentar yang muncul.

Tidak ada yang namanya terlalu banyak dokumentasi untuk kode, tapi ada yang namanya terlalu sedikit. Cara terbaik untuk mendorong penulisan dokumentasi adalah menjadikannya bagian dari penyampaian fitur. Sebuah fitur tidak boleh dianggap lengkap sampai dokumentasi yang memadai telah ditulis dan ditempatkan di lokasi yang sesuai. Mewajibkan dokumen desain sebelum pengkodean dimulai juga membantu menjaga

dokumentasi selalu diingat oleh semua orang. Terlepas dari bagaimana Anda memutuskan untuk mengaturnya, dokumentasi harus menjadi bagian dari penyampaian setiap kali kode ditulis. Jenis dokumentasi yang tepat akan bergantung pada jenis kodenya, namun semua kode memerlukan dokumentasi, meskipun hanya penambahan satu kalimat pada dokumen yang sudah ada.

Memiliki beberapa panduan gaya pengkodean yang baik, arsitektur yang terdefinisi dengan baik, dan sejumlah besar dokumentasi akan menyiapkan proyek apa pun untuk sukses. Bagian yang benar-benar menantang muncul ketika Anda ingin memasukkan kode yang tidak ditulis oleh tim Anda, dan memiliki panduan tentang cara melakukannya adalah hal yang penting untuk kesehatan aplikasi Anda secara keseluruhan.

2.4 MENGELOLA KOMPONEN PIHAK KETIGA

Kecuali Anda sedang mengerjakan proyek pribadi, kemungkinan besar aplikasi Web Anda akan bergantung pada satu atau lebih komponen pihak ketiga agar dapat berfungsi dengan baik. Bahkan pengembang terbaik dan paling berpengalaman pun beralih ke komponen pihak ketiga ketika ada aspek aplikasi Web yang tidak ingin mereka miliki atau pelihara. Tidak masuk akal bagi setiap orang untuk menciptakan cara mereka sendiri dalam melakukan segala sesuatu sehingga komponen pihak ketiga membantu membuat aplikasi Web aktif dan berjalan lebih cepat sementara melakukan outsourcing pemeliharaan kepada orang lain.

Misalnya, ketika browser baru keluar, Anda dapat yakin bahwa jQuery akan diperbarui untuk mendukungnya. Yang perlu Anda lakukan hanyalah memasukkan versi terbaru dan aplikasi Web Anda terus berfungsi dengan baik. Jika Anda telah membuat perpustakaan abstraksi browser Anda sendiri, Anda harus selalu memperbaruinya setiap kali browser baru dirilis. Karena hal ini terjadi setiap enam minggu untuk Chrome dan FirefoX, memperbarui kode Anda akan menjadi tugas yang berat dan berulang sehingga menghalangi Anda melakukan hal-hal yang lebih penting.

Anda tidak menambah nilai pada bisnis atau aplikasi Anda dengan terus-menerus menulis ulang utilitas tingkat rendah. Menggunakan komponen pihak ketiga membebaskan Anda untuk fokus pada nilai sebenarnya yang dapat Anda berikan kepada pengguna.

Cara Memilih Komponen Pihak Ketiga

Ada banyak jenis komponen pihak ketiga yang berbeda. Ada perpustakaan JavaScript untuk hampir semua hal, kerangka kerja dan toolkit CSS, gambar dan font, dan jenis komponen lainnya yang akan terus berkembang seiring dengan teknologi Web. Memilih komponen pihak ketiga ini adalah keputusan yang sangat penting karena mereka mewakili bahan dan alat yang akan digunakan untuk membangun aplikasi Web Anda. Ibarat membangun rumah, Anda juga ingin memastikan bahannya kokoh dan peralatannya bisa dipercaya.

Pengembangan web memiliki komunitas sumber terbuka yang dinamis dengan banyak komponen pihak ketiga yang tersedia secara gratis. Itu kabar baiknya. Kabar buruknya adalah memilah-milah komponen open source membuat sulit untuk menemukan kualitas. Bahkan katalog komponen, seperti jQuery Plugin Registry20 dan NPM Registry21, menyulitkan untuk

menemukan komponen berkualitas. Masing-masing komponen ditempatkan sejajar dengan komponen lainnya, dan terkadang sistem penilaian sewenang-wenang, seperti bintang atau popularitas, atau ketika komponen baru saja diperbarui, tidak menceritakan kisah lengkapnya.

Inti dari penggunaan komponen pihak ketiga adalah untuk membebaskan diri Anda dari pemeliharaan beberapa kode. Untuk melakukan hal tersebut, Anda memerlukan tingkat kepastian yang wajar bahwa kode tersebut tidak ditinggalkan. Jika Anda akhirnya menyertakan komponen pihak ketiga yang tidak lagi diperbarui, pada akhirnya Anda akan menyimpannya sendiri. Demikian pula, jika komponen dipertahankan tetapi pengembang memerlukan waktu lama untuk menjawab pertanyaan Anda, pada akhirnya Anda akan memodifikasinya sendiri karena tidak sabar menunggu rilis resmi.

Jadi, bagaimana Anda dapat mengetahui bahwa komponen pihak ketiga dapat dipercaya? Berikut beberapa hal yang perlu dievaluasi. Kapan terakhir kali diperbarui? Jika komponen baru saja diperbarui, kemungkinan besar komponen tersebut akan terus diperbarui di masa mendatang. Jika komponen tersebut belum diperbarui, maka komponen tersebut mungkin akan ditinggalkan. Secara umum, carilah hal-hal yang telah berubah dalam sebulan terakhir. Ini merupakan indikator yang cukup bagus bahwa mereka masih dalam pengembangan aktif.

Siapa pengembangnya? Jika komponen disediakan oleh perusahaan atau organisasi, ini merupakan pilihan yang lebih aman dibandingkan komponen yang didukung oleh satu orang. Ada banyak proyek sumber terbuka yang dirilis oleh pembuatnya dan kemudian dibuang. Hindari komponen tersebut kapan pun Anda bisa.

Jika suatu komponen diperbarui oleh satu orang dan orang tersebut didukung oleh perusahaan atau organisasi (yaitu, pengembang ini dibayar untuk memelihara komponen tersebut), maka komponen tersebut mungkin juga aman untuk digunakan. Komponen apa pun yang tampak seperti hobi seseorang harus dihindari, meskipun komponen tersebut tampaknya melakukan semua yang Anda inginkan. Kecuali Anda bersedia melanjutkan proyek itu di masa depan, sebaiknya hindari.

Seberapa responsifkah pengembangnya? Pada titik tertentu, Anda akan menemukan kesalahan pada komponen yang Anda gunakan. Kesalahan tersebut mungkin menyebabkan masalah fungsionalitas pada aplikasi Web Anda sehingga Anda ingin memperbaikinya secepat mungkin. Kecepatan pengembang dalam mengatasi kekhawatiran Anda sangatlah penting. Anda dapat memahami hal ini dengan menelusuri pelacak masalah publik untuk melihat berapa lama masalah tertentu terbuka sebelum diselesaikan.

Perlu diingat bahwa diselesaikan tidak berarti harus ada rilis resmi, bisa jadi pengelola telah memeriksa perbaikan sehingga pelapor dapat menambal salinannya sendiri sambil menunggu rilis. Waktu penyelesaian yang baik untuk masalah yang signifikan diukur dalam hitungan hari, bukan minggu. Jika Anda tidak dapat mengandalkan perputaran cepat dari pengembang komponen pihak ketiga, maka komponen tersebut mungkin sebaiknya tidak digunakan.

Seberapa stabil API-nya? Anda mungkin tergoda untuk menggunakan komponen pihak ketiga yang dianggap sedang naik daun. Berhati-hatilah dalam mengandalkan apa pun yang

belum mencapai versi 1.0. Sebelum rilis 1.0, komponen memiliki kecenderungan perubahan yang cukup drastis. API biasanya tidak dikunci hingga versi 1.0 dan hal ini menimbulkan tantangan saat Anda mencoba mengupgrade komponen. Mengandalkan jejak API yang selalu berubah berarti Anda akan terus-menerus mengubah kode agar dapat berfungsi dengan komponen. Bantulah diri Anda sendiri dan tunggu hingga komponen tersebut mencapai kematangan sebelum mengandalkannya.

Siapa lagi yang menggunakannya? Lakukan riset untuk mengetahui siapa lagi yang menggunakan komponen ini. Idealnya Anda menginginkan sesuatu yang digunakan oleh dua atau lebih aplikasi Web besar. Ketika perusahaan atau organisasi besar mengandalkan komponen tersebut, terdapat lebih banyak insentif bagi pengembangnya untuk terus memperbarui komponen dan memperbaiki bug.

Itu tidak berarti Anda tidak boleh menerapkan apa pun kecuali itu digunakan oleh Facebook atau Google, itu hanya berarti bahwa aplikasi Web Anda tidak boleh menjadi yang pertama mengandalkan komponen tersebut. Terdapat keamanan dalam jumlah dan semakin banyak pengguna suatu komponen, semakin besar kemungkinan komponen tersebut akan terus berkembang dan masalah akan teratasi.

Jadi pastikan untuk diingat bahwa tidak semua plugin jQuery dibuat dengan cara yang sama. Tidak semua modul NPM dibuat dengan cara yang sama. Dan tentunya tidak semua proyek sumber terbuka dibuat dengan cara yang sama. Siapa pun dapat membuat komponen sumber terbuka, tetapi tidak semua orang berdedikasi untuk melanjutkannya. Pastikan untuk melakukan uji tuntas terhadap komponen pihak ketiga mana pun sebelum memasukkannya ke dalam produk Anda. Satu keputusan buruk dapat menimbulkan keretakan pada fondasi Anda yang mungkin sulit diperbaiki nantinya.

Bagian dari uji tuntas Anda adalah melihat kode pihak ketiga itu sendiri. Anda harus yakin tidak ada benda berbahaya yang terkubur di dalamnya sebelum menerapkannya. Gagal melakukan pemeriksaan sekilas terhadap kode dapat menyebabkan masalah. Pada tahun 2011, situs web yang sekarang sudah tidak berfungsi menghilang dari hasil pencarian Google dalam semalam. Setelah beberapa eksplorasi yang panjang, diputuskan bahwa plugin WordPress adalah penyebabnya. Setiap kali Googlebot mengunjungi situs, plugin akan mengalihkan ke tempat lain. Orang yang menulis plugin telah membuat kesalahan dalam kode karena dia hanya bermaksud memblokir Googlebot dari direktori plugin (lihat penjelasannya23). Sayangnya, kerusakan telah terjadi dan menyebabkan banyak ketidaknyamanan bagi pengguna plugin.

Disengaja atau tidak, potensi kerusakan komponen pihak ketiga pada aplikasi Anda sangatlah besar. Jangan berasumsi bahwa semua komponen pihak ketiga dibuat sama. Luangkan waktu untuk menyelidikinya sebelum memasukkannya ke dalam aplikasi Anda.

2.5 KOMPONEN PIHAK KETIGA

Setelah Anda memilih beberapa komponen pihak ketiga untuk aplikasi Web Anda, Anda mungkin menemukan bahwa komponen tersebut tidak melakukan semua yang Anda perlukan. Pada saat itu, Anda harus mengambil keputusan. Anda dapat memulai pencarian

lagi untuk komponen baru atau mencoba membuat komponen ini berfungsi sesuai kebutuhan Anda. Sayangnya, banyak pengembang cenderung mengambil pendekatan terakhir yang berarti mem-forking kode aslinya. Forking adalah aktivitas apa pun yang komponen pihak ketiganya dimodifikasi oleh orang lain selain pengelolanya. Tidak masalah jika Anda menambahkan fungsi baru, menghapus fungsi yang sudah ada, atau mengubah fungsi yang sudah ada, semua ini dianggap forking.

Membagi komponen pihak ketiga adalah ide yang buruk. Dengan melakukan hal ini, Anda memastikan jalur peningkatan non-linier untuk komponen tersebut. Memutakhirkan komponen semudah menghapus file Anda saat ini dan memasukkan file baru. Terkadang API komponen akan sedikit berubah sehingga Anda perlu menyesuaikan kode Anda. Namun, komponen yang baik dapat meminimalkan kejadian seperti ini. Peningkatan drop-in dianggap linier.

Jalur pemutakhiran non-linier adalah jalur di mana penghentian versi baru hanyalah awal dari pemutakhiran. Anda kemudian harus menjaring kode Anda dan membuat perbaikan yang sesuai untuk versi baru. Jika Anda telah melakukan fork pada komponen tersebut, maka Anda secara dramatis meningkatkan kemungkinan jalur peningkatan non-linier. Ada berbagai macam masalah berbeda yang dapat muncul. Misalnya, Anda mungkin telah menambahkan metode baru pada komponen hanya untuk menemukan bahwa versi resmi berikutnya dari komponen tersebut memiliki fungsi dengan nama tersebut yang melakukan sesuatu yang berbeda. Itu berarti Anda tidak hanya perlu mengubah nama fungsi Anda, tetapi Anda juga harus mengubah semua tempat penggunaannya dalam kode Anda.

Saat Anda memutuskan untuk menggunakan komponen pihak ketiga, sebaiknya hindari mem-forkingnya dengan cara apa pun. Mempertahankan jalur peningkatan linier sangat penting untuk komponen ini. Ingat, inti penggunaan komponen pihak ketiga adalah untuk menghilangkan biaya pemeliharaan. Segera setelah Anda membagi sebuah komponen, Anda akan menghadapi masalah pemeliharaan di masa depan.

Untuk memperjelas pemisahan antara kode Anda dan kode pihak ketiga, pastikan untuk menyimpannya di direktori terpisah. Misalnya, jika Anda memiliki direktori /js tempat semua kode JavaScript berada, tempatkan semua kode JavaScript pihak ketiga di /js/external atau /js/3rdparty. Hal yang sama juga berlaku untuk CSS. Memisahkan kode pihak ketiga adalah pengingat yang baik bahwa mengedit file-file ini dengan cara apa pun tidak pantas.

Kemungkinan besar komponen pihak ketiga tidak akan melakukan semua yang Anda perlukan. Pada titik ini, Anda semakin tergoda untuk mengedit komponen. Tolak dorongan ini karena hal ini akan merusak jalur peningkatan linear komponen. Sebaliknya, carilah cara untuk menambahkan fungsionalitas tambahan yang tidak mengharuskan Anda mengedit kode pihak ketiga secara langsung.

Komponen yang dirancang dengan baik biasanya memiliki sistem plugin atau ekstensi. Hal ini disediakan agar pengembang dapat meningkatkan fungsionalitas yang ada dengan cara yang dapat diprediksi dan dikendalikan. jQuery, YUI dan Dojo semuanya memiliki sistem plugin yang memungkinkan Anda menambahkan fungsionalitas baru tanpa mengedit file utama. Jika Anda menemukan bahwa komponen pihak ketiga tidak menyediakan semua fungsi yang Anda

perlukan, Anda sebaiknya mencoba menulis plugin atau ekstensi untuk komponen tersebut terlebih dahulu.

Saat melakukan ini, kode Anda tidak boleh berada dalam file yang sama dengan kode komponen. Untuk mempertahankan jalur peningkatan linier, Anda ingin memastikan bahwa Anda dapat memasukkan versi baru dari file komponen tanpa perlu memodifikasinya dengan cara apa pun. Butuh lebih banyak fungsi jQuery? Buat plugin jQuery dan letakkan di /js/plugins (bukan di /js/external atau /js/3rdparty).

Jika komponen yang Anda andalkan tidak memiliki sistem plugin, maka pendekatan terbaik adalah membuat komponen pembungkus yang mengabstraksi kode pihak ketiga. Komponen wrapper adalah komponen yang Anda buat dan kelola, sehingga Anda dapat menyediakan fungsionalitas apa pun yang Anda inginkan. Anda dapat membuat fungsi yang diteruskan langsung ke komponen pihak ketiga, serta fungsi baru di mana Anda telah mengimplementasikan fungsi khusus.

Mem-forking komponen pihak ketiga selalu merupakan ide yang buruk, jadi pastikan Anda memisahkan kode komponen pihak ketiga dan kode Anda. Pertahankan jalur pemutakhiran linier untuk komponen pihak ketiga dengan menyimpannya di direktori terpisah dari kode yang Anda tulis sendiri.

Mempertaruhkan Rumah

Membawa komponen pihak ketiga ke dalam aplikasi Web Anda adalah cara terbaik untuk memulai dan menjalankannya dengan cepat. Namun, dengan melakukan itu Anda memasang taruhan yang sangat signifikan pada komponen-komponen tersebut. Anda yakin waktu yang dihemat dengan menggunakan komponen ini lebih besar daripada waktu yang Anda habiskan untuk membuat sendiri komponen serupa dan merawatnya. Itu sebabnya memilih komponen pihak ketiga yang tepat dan mengelolanya dengan tepat sangatlah penting. Setelah sebuah komponen digunakan dalam aplikasi Web, sangat sulit untuk mengekstrak komponen tersebut dan menggantinya dengan komponen lain.

Ini adalah hal terpenting yang perlu diingat ketika menggunakan komponen pihak ketiga: sekali Anda berkomitmen, sulit untuk mengubah pikiran Anda. Kode pihak ketiga mulai direferensikan di banyak tempat di basis kode Anda, sehingga mengubah ke komponen lain berarti menelusuri seluruh basis kode Anda dan membuat perubahan. Itu adalah sesuatu yang biasanya tidak Anda sukai untuk dilakukan terus-menerus. Ada fitur-fitur baru yang harus dibangun dan bug yang perlu diperbaiki, dan hal terakhir yang ingin Anda lakukan adalah mengganti fondasi yang goyah ketika Anda bisa memberikan nilai lebih kepada pengguna Anda.

Memiliki segalanya untuk basis kode yang dapat dipelihara hanya dapat dijamin ketika memulai dari awal. Namun apa jadinya jika Anda tidak mampu melakukan itu? Bagaimana Anda mulai bekerja dengan kode yang tidak terorganisir dengan baik atau memiliki masalah mencolok lainnya?

Berurusan dengan Kode Warisan

Definisi kode lama bergantung pada siapa Anda bertanya. Beberapa orang menggambarkan kode lama sebagai kode yang berkaitan dengan fungsi yang tidak lagi

didukung. Yang lain menggambarkannya sebagai kode yang ditulis oleh seseorang yang tidak lagi memeliharanya. Saya menggambarkan kode warisan sebagai kode apa pun yang Anda warisi dari orang lain tanpa penyerahan formal. Hal ini sering terjadi karena kode tersebut ditulis oleh seseorang yang tidak lagi bekerja pada proyek tersebut sehingga pengelolaan kode tersebut menjadi sebuah ekspedisi melalui wilayah asing. Anda diharapkan dapat memperbaiki bug dan memperluas pekerjaan sebelumnya tanpa memahami sepenuhnya apa fungsinya atau mengapa pekerjaan itu dibuat dengan cara tertentu. Singkatnya, saya menggambarkan kode lama sebagai kode yang ditulis orang lain dan harus Anda pertahankan.

Saya belum pernah bertemu pengembang yang senang bekerja dengan kode lama. Merupakan pengalaman yang membuat frustrasi karena harus membiarkan kode orang lain tetap berjalan tanpa memahaminya sepenuhnya. Ada kecenderungan alami untuk percaya bahwa segala sesuatu dalam kode lama dilakukan secara tidak benar dan oleh karena itu harus dibuang sepenuhnya untuk memulai yang baru. Praktisnya, Anda jarang mendapatkan kesempatan untuk memulai proyek sepenuhnya dari awal, sehingga penting untuk memahami cara menangani kode lama ini.

Namun sebelum Anda dapat mempelajari cara bekerja secara efektif dengan kode lama, Anda harus memahami satu fakta sederhana: kode yang Anda tulis hari ini adalah kode lama di masa depan. Setiap baris kode yang Anda tulis pada akhirnya akan dikelola oleh orang lain. Mengingat hal itu membantu memandu keputusan yang Anda buat hari ini tentang kode apa pun yang sedang Anda kerjakan. Tanyakan pada diri Anda, apa yang perlu diketahui oleh orang yang mengelola file ini agar dapat bekerja secara efektif?

Tuliskan

Saya sering menggambarkan bekerja dengan kode lama mirip dengan spelunking, hobi menjelajahi gua. Dalam spelunking, Anda tidak selalu tahu apa yang ada di sekitar. Gua gelap, basah, dan tidak selalu stabil. Menggali kode lama memiliki kendala serupa karena Anda mungkin tidak sepenuhnya yakin dengan fungsi setiap bagian kode. Akibatnya, Anda adalah seorang penjelajah yang mencoba memastikan bahwa gua tersebut tidak menimpa Anda. Membuatnya lebih jauh ke dalam kode adalah sebuah pencapaian, dan untuk memudahkan orang lain, ada baiknya menggambar peta.

Salah satu tugas utama pengelola kode adalah menghilangkan keajaiban dari basis kode. Saat Anda bekerja dengan kode, memperbaiki bug, atau menambahkan fungsionalitas baru, pastikan Anda mendokumentasikan apa yang Anda lakukan. Setiap kali Anda menyentuh kode lama, berusahalah untuk mendapatkan pemahaman yang lebih baik tentang apa yang dilakukannya. Saat Anda menemukan sesuatu, tulislah. Menuliskannya semudah meninggalkan satu atau dua komentar di kode sehingga orang berikutnya yang melihatnya tidak perlu bertanya-tanya mengapa sesuatu dilakukan.

Sebaiknya tinggalkan jejak komentar seperti ini sehingga Anda perlahan mulai membangun dasar pengetahuan tentang cara kerja kode. Ini berguna tidak hanya bagi orang lain, namun bagi Anda jika Anda mendapati diri Anda mengerjakan kode ini lagi berbulan-bulan atau bertahun-tahun ke depan.

2.6 TAMBAHKAN UJI

Michael C. Feathers, penulis Working Effectively with Legacy Code (Pearson, 2004), menjelaskan kode warisan sebagai kode apa pun yang tidak memiliki pengujian. Dalam praktiknya, kelemahan kode lama sering kali dikaitkan dengan kurangnya pengujian, itulah sebabnya kode tersebut mulai tampak seperti sihir. Tanpa apa pun yang mendokumentasikan perilaku yang diharapkan, baik dalam bentuk komentar, dokumentasi formal, atau pengujian yang baik, sangat sulit untuk merasa percaya diri saat membuat perubahan pada kode lama. Sekali lagi, kecil kemungkinan Anda memiliki cukup waktu untuk berhenti dan menulis pengujian untuk semua kode lama sekaligus. Itu tidak berarti Anda harus menyerah sama sekali dalam ujian.

Pendekatan untuk menambahkan pengujian pada kode lama sangat mirip dengan menambahkan komentar pada kode lama. Setiap kali Anda bekerja dengan kode, pastikan untuk menambahkan setidaknya satu pengujian. Jika Anda memperbaiki bug, tulis tes yang mereproduksi masalah tersebut sebelum mencoba memperbaikinya. Ini disebut pengembangan yang didorong oleh pengujian. Anda mulai dengan menulis tes yang gagal (karena bug) dan kemudian menulis kode yang membuat tes tersebut lulus. Setelah memperbaiki bug, Anda dapat menambahkan komentar ke kode dan memeriksa pengujian baru. Sekarang ada satu bagian kode yang tampaknya ajaib.

Kombinasi komentar kode dan pengujian membantu mengendalikan kode lama. Meskipun kemajuan akan tampak lambat dengan menambahkan satu tes pada satu waktu, melakukan satu tes lebih baik daripada tidak melakukan tes sama sekali. Memiliki dua tes lebih baik daripada memiliki satu tes. Saat Anda terus bekerja dengan kode dan terus menambahkan pengujian, Anda akan menampilkan lebih banyak fungsi. Bagian dari kode lama akan mulai masuk akal karena Anda memiliki cara sederhana untuk menentukan apakah kode tersebut berfungsi dengan baik. Penting untuk memikirkan lari maraton daripada lari cepat. Anda tidak dapat menulis semua tes sekaligus, tetapi selama Anda terus menulis tes, pada akhirnya Anda akan mencapai pemahaman yang lebih baik tentang kode lama.

Pembuatan Ulang Dan Penulisan Ulang

Jika beruntung, Anda mungkin punya waktu untuk memfaktorkan ulang atau menulis ulang kode lama. Pemfaktoran ulang melibatkan membiarkan antarmuka publik API tidak berubah sambil mengubah implementasi yang mendasarinya. Di sinilah pengujian unit yang efektif sangat penting. Pengujian unit yang baik menggunakan antarmuka publik tanpa memperhatikan detail implementasi. Singkatnya, pengujian unit dirancang untuk menguji antarmuka dengan menyatakan bahwa masukan tertentu menghasilkan keluaran tertentu. Output dapat berupa nilai yang dikembalikan, perubahan pada objek yang ada, kesalahan yang terjadi, atau efek lain yang terlihat dari eksekusi beberapa kode. Tidak mungkin memfaktorkan ulang kode secara efektif kecuali Anda memiliki pengujian unit untuk memverifikasi bahwa perubahan Anda menghasilkan hasil yang sama seperti kode lama.

Menulis ulang kode adalah saat Anda membuat perubahan signifikan pada antarmuka publik dan implementasinya. Ini adalah impian seorang pengembang: Anda memiliki kebebasan untuk memulai dari awal dan membuat sesuatu yang lebih baik dari apa yang

selama ini Anda kerjakan. Saat menulis ulang kode, tidak masalah apa yang Anda gantikan karena Anda sedang membuat sesuatu yang baru. Sayangnya, ini biasanya merupakan waktu ketika kode yang paling tidak ramah masa depan ditulis.

Sesuatu yang menarik terjadi di benak pengembang ketika diberi kertas kosong. Rasanya tidak ada aturan lagi, karena tidak lagi terkendala dengan pekerjaan sebelumnya. Anda mulai bermimpi tentang betapa indahnya hal-hal dan bagaimana pekerjaan Anda jauh lebih baik daripada sebelumnya. Apa yang tidak Anda sadari adalah bahwa membuat sesuatu dari awal berarti Anda sedang menulis kode lama di masa depan. Kode Anda akan diberikan kepada orang lain untuk dijaga, jadi sekarang Andalah yang harus mencegah pembuatan kode yang lebih buruk lagi.

Salah satu mentor saya mengatakan kepada saya bahwa hanya ada sedikit pengembang yang dapat secara efektif memulai dari awal dan membangun sebuah proyek. Dalam kegembiraan berkreasi, Anda cenderung melupakan dasar-dasarnya dan mengambil jalan pintas untuk menyelesaikan pekerjaan. Faktanya, banyak penulisan ulang yang mewakili awal dari spiral kode lama yang sudah dikenal:

1. Kode ditulis ulang.
2. Penulis kode yang ditulis ulang hadir untuk menjawab pertanyaan.
3. Tim bekerja lebih efektif dengan kode baru.
4. Penulis kode baru mulai keluar.
5. Muncul pertanyaan tentang cara kerja bagian-bagian kode.
6. Semakin banyak bagian kode yang menjadi ajaib.
7. Pembangunan melambat karena kurangnya pemahaman.
8. Pengembang menjadi frustrasi dan menyatakan bahwa mereka tidak dapat bekerja dengan kode ini lebih lama lagi.
9. Lanjut ke #1.

Meskipun penulisan ulang adalah sesuatu yang suka dilakukan oleh pengembang, pemfaktoran ulang sering kali mencapai hasil positif dengan lebih cepat dan tanpa mendorong tim kembali ke siklus penulisan ulang kode lama.

Anda tentu saja dapat menulis ulang kode secara efektif jika Anda melakukannya dengan perspektif yang tepat. Setiap kali Anda menulis kode baru, Anda harus memikirkan lima tahun ke depan dan bagaimana orang lain akan melanjutkan apa yang Anda tinggalkan. Pikirkan tentang masalah yang Anda alami saat bekerja dengan kode lama: pengujian yang tidak cukup, dokumentasi yang terlalu sedikit, pola yang tidak masuk akal. Kemudian, selesaikan masalah tersebut saat Anda menulis kode baru.

Kesimpulan

Seluruh bab ini mencakup langkah-langkah spesifik yang dapat Anda ambil untuk memastikan kode yang Anda tulis dapat dipelihara dan ramah masa depan. Cara terbaik untuk menghindari kode spageti adalah dengan mengedepankan pekerjaan dalam mengatur berbagai hal. Konvensi kode memastikan bahwa setiap orang berbicara dalam bahasa yang sama. Arsitektur memberikan landasan yang kokoh untuk membangun. Dokumentasi yang baik adalah penting agar orang lain dapat memperluas kode Anda nanti. Memilih dan

mengelola komponen pihak ketiga dengan tepat memastikan Anda memiliki jalur peningkatan linier saat perbaikan tersedia.

Tentu saja, tidak setiap hari Anda memulai dengan kertas kosong. Anda akan cukup sering menangani kode orang lain. Jika kode tersebut termasuk dalam kategori kode lama, langkah selanjutnya adalah membuat bagian kecil dari kode tersebut lebih mudah dipahami setiap hari. Tambahkan dokumentasi. Tambahkan tes. Jangan percaya pada sihir.

Basis kode memiliki kehidupannya sendiri seiring berjalannya waktu. Beberapa tumbuh menjadi anggun dan cantik sementara yang lain menjadi degil dan bau. Pertumbuhan basis kode Anda bergantung pada cara Anda memeliharanya, struktur yang Anda berikan, dan seberapa nyaman orang-orang dalam melakukan modifikasi.

BAB 3
PRODUK WEB

Hype tentang HTML5 dan anggapan kemenangannya atas Flash sudah sedikit surut dan sekarang adalah saat yang tepat untuk menganalisis apa yang kami lakukan dan mengubah pendekatan kami agar siap menghadapi sekelompok orang baru yang datang ke Web. Dengan berlakunya Hukum Moore1 sepenuhnya, kita semua bekerja dengan sangat kuat, perangkat resolusi tinggi dengan koneksi cepat saat kami berkembang, dan banyak klien kami memiliki hal yang sama. Hal ini akan menghasilkan situs web dan etalase yang indah dan mengesankan yang mencatat ratusan permintaan server dan beberapa megabyte data.

Namun, ketika kita pergi keluar dan menggunakan perangkat seluler, atau koneksi nirkabel yang tersedia di kafe dan hotel, segalanya tampak berbeda. Sebagian besar waktu kita dihabiskan untuk menonton animasi berputar yang menunjukkan ada sesuatu yang sedang dimuat, dan sering kali kita diberitahu bahwa koneksinya tidak bagus dan kita harus mencoba lagi. Masalah yang sama akan muncul pada generasi pengguna online berikutnya. Ingat betapa frustasinya dial-up? Kita tidak boleh mengulangi kesalahan yang sama dengan menambahkan banyak konten dan fungsi yang tampak apik selagi kita mengembangkannya hanya karena kita bisa. Kita harus menjadi lebih ramping dan menghilangkan sebagian lemak.

Seperti halnya penurunan berat badan lainnya, hanya melenturkan otot dan pergi ke gym saja tidak cukup kita juga perlu menganalisis dan mengubah apa yang kita masukkan ke dalam tubuh kita. Waktu gym kami saat ini berkonsentrasi pada menghasilkan alur kerja baru yang lebih profesional untuk Web. Daripada menulis CSS, HTML, dan JavaScript, kami menggunakan proses pembuatan dan skrip seperti Grunt, kami melakukan praproses CSS kami dengan Sass dan LESS, dan kami secara otomatis menghasilkan sprite gambar dari banyak gambar dalam satu folder. Banyak orang juga menyatakan bahwa satu-satunya cara agar kita cukup fleksibel untuk membangun 'aplikasi nyata' adalah dengan beralih ke bahasa lain yang baru seperti Dart dan TypeScript dan menerapkan lebih banyak rekayasa perangkat lunak, ilmu komputer, dan pendekatan berbasis pola pada Web. Hal ini mengingatkan saya pada masa ketika pustaka JavaScript sedang populer dan membuat browser berperilaku sesuai dengan prediksi. Dari ratusan perpustakaan yang diciptakan pada masa itu, hanya sedikit yang tersisa saat ini dan kita dapat dengan mudah memperkirakan bahwa banyak dari apa yang disebut sebagai “alat penting” yang kita buat dan andalkan saat ini akan menjadi ketinggalan jaman dan ketinggalan jaman.

Mungkin kita juga harus berhenti sejenak dari upaya kita untuk menjadi keren, baru, dan inovatif sepanjang waktu, sekadar untuk menjadi keren, dan memeriksa apa yang kita lakukan - menganalisis kebiasaan makan kita, bisa dikatakan begitu. Saya menyebutnya Vanilla Web Diet, sama seperti orang-orang mulai menyebutnya menggunakan JavaScript tanpa perpustakaan vanilla JavaScript. Saya ingin berbagi beberapa ide dan pemikiran yang dapat Anda ingat untuk membantu memperkecil situs web atau aplikasi Anda berikutnya.

Tidak ada satu pun bagian dalam diet vanilla Web yang baru, namun sangat perlu untuk diulangi. Inilah yang akan kami bahas:

- Kembangkan apa yang berhasil
- Kurangnya dukungan adalah sebuah peluang
- Kode khusus browser tidak dapat dipercaya
- Gunakan gabungan teknologi, masing-masing untuk memberikan manfaat terbaiknya
- Mengajukan pertanyaan
- Tulislah sebanyak yang diperlukan, tidak sedikit pun
- Ini bukan tentang apa yang bisa Anda tambahkan: ini tentang apa yang tidak bisa kita hilangkan
- Kegunaan mengalahkan konsistensi di seluruh browser
- Muat hanya yang diperlukan saja
- Menganalisis efek mengalahkan penambahan FX
- Fondasi yang kuat mengalahkan kemungkinan penambahan di masa depan

Aturan pertama dari diet vanilla Web adalah memulai dari awal. Sangat menggoda untuk memulai dengan banyak boilerplate HTML5 atau solusi penyetelan ulang CSS yang ada, tetapi jangan langsung melakukannya.

Janji dari boilerplate HTML atau kerangka CSS adalah untuk meratakan perbedaan antar browser dan memungkinkan Anda memulai dari dasar yang tampaknya berfungsi di seluruh browser. Dalam banyak kasus, hal ini berarti beberapa browser desktop banyak di antaranya bahkan tidak digunakan lagi atau tidak akan pernah tersedia di lingkungan seluler. Itulah sebabnya pakar kinerja mulai menentang boilerplate: karena dianggap terlalu berat. Membuatnya lebih ringan adalah salah satu cara untuk mengatasi masalah tersebut, namun itu juga berarti kita secara efektif membuat garpu kode yang perlu dipelihara dan peningkatan jika ada perubahan yang diperlukan pada versi aslinya. Ada satu boilerplate yang sebenarnya disebut HTML5 dan didukung dengan cara yang dapat diprediksi oleh semua browser saat ini dan, khususnya, browser seluler. Ide di balik HTML5 adalah untuk memiliki parser yang sama di setiap browser, jadi mari kita berkonsentrasi pada penggunaannya, daripada padding asing untuk lingkungan penjelajahan yang sudah tidak ada lagi. Tanpa basa-basi lagi, mari selami bagian pertama dari produk Web yang sehat: titik awal yang masuk akal.

Kembangkan Apa yang Berhasil

Lapisan dasar kita harus berupa HTML biasa dan sederhana yang berfungsi sesuai tujuan produk. Sesuatu yang tampak seperti tombol tetapi tidak melakukan apa pun tidak membantu pengguna kami. Ketika semuanya gagal, HTML adalah apa yang pengguna dapatkan: jangan sampai mereka kehilangan hal tersebut. Kesalahan terjadi, kesalahan bisa muncul dari banyak sumber, banyak di antaranya mungkin di luar kendali kita. Seperti yang dikatakan Jake Archibald: *"Semua pengguna kami menonaktifkan JavaScript hingga skrip pertama Anda dimuat dan dieksekusi."* Itu sebabnya waktu yang dihabiskan untuk memikirkan lapisan dasar dari apa yang kita lakukan tidak pernah terbuang percuma.

Ini harus menjadi prinsip utama di balik apa yang kita bangun – dan ini bukanlah hal baru. Kami menyebutnya "pelapisan semantik, peningkatan progresif"; kami bahkan

membuat akronim lucu seperti POSH (untuk HTML semantik biasa). Namun kita terus menerus melupakan prinsip ini. Tampaknya upaya untuk membuat HTML yang masuk akal dan menjelaskan dirinya sendiri membingungkan orang-orang yang berasal dari dunia pemrograman berorientasi objek, di mana segala sesuatunya perlu dipakai dan diubah sebelum diterapkan. Dalam HTML, tidak terlalu banyak. Masuk akal untuk melihat HTML sebagai landasan untuk membangun. Anda tidak dapat membangun sesuatu yang berat dan besar di atas fondasi yang reyot, jadi mari gunakan HTML sebanyak yang diperlukan — namun jangan lebih banyak dan saat kita menggunakan HTML, gunakan apa yang berhasil.

Misalnya, Anda ingin memiliki kontrol tab. Tidak jarang widget JavaScript menghasilkan sesuatu seperti berikut, baik sebagai HTML yang harus Anda buat agar dapat berfungsi, atau dihasilkan oleh JavaScript:

```
<div class="tabcontrol">
 <div class="tab">One</div>
 <div class="tab">Two</div>
 <div class="tab">Three</div>
 <div class="panel">Panel One</div>
 <div class="panel">Panel Two</div>
 <div class="panel">Panel Three</div>
</div>
```

Makna struktural dari hal ini sebenarnya nol. Tidak, nama kelas semantik tidak cukup nama tersebut tidak berarti apa-apa bagi browser dan tidak memicu fungsi rendering atau interaksi apa pun yang sangat baik dilakukan oleh browser. Jadi mengapa tidak menggunakan apa yang kita miliki dan telah kita miliki sejak browser pertama?

```
<section class="tabcontrol">
   <nav id="nav">
     <ul>
        <li><a href="#one">One</a></li>
        <li><a href="#two">Two</a></li>
        <li><a href="#three">Three</a></li>
    </ul>
   </nav>
   <article id="one">
     <header><h1>One</h1></header>
     <section>
     <!-- fill me with content -->
   </section>
    <footer>
      <p class="back"><a href="#nav">Back to menu</a></p>
    </footer>
   </article>
 <article id="two">
    <header><h1>Two</h1></header>
```

```
      <section><!-- fill me with content --></section>
      <footer>
       <p class="back"><a href="#nav">Back to menu</a></p>
      </footer>
 </article>
 <article id="three">
    <header><h1>Three</h1></header>
    <section>
      <!-- fill me with content -->
 </section>
<footer>
       <p class="back"><a href="#nav">Back to menu</a></p>
    </footer>
  </article>
</section>
```

Memang benar, ini lebih banyak HTML, tetapi ada satu hal yang menarik: Anda mendapatkan banyak sekali manfaat dari struktur ini.

- Ia berfungsi tanpa JavaScript atau CSS apa pun. Pengunjung mendapatkan daftar link yang mengarah ke bagian dokumen, cara yang teruji dan benar untuk melompat ke bagian menarik dalam dokumen besar. Tautan kembali ke menu memungkinkan Anda kembali dengan mudah.
- Dengan menggunakan target nyata dalam dokumen dalam bentuk ID pada elemen (hari-hari bernama jangkar benar-benar sudah berakhir sekarang), panel kita dapat di-bookmark secara otomatis dan menyimpan riwayat browser tanpa harus menggunakan pushState atau peretasan yang mengerikan dari hashbang.
- Kami memiliki banyak kaitan untuk gaya dan JavaScript. Kita bahkan dapat menggunakan pemilih :target di CSS dan tidak memerlukan JavaScript sama sekali.
- Ketika browser pada akhirnya melakukan sesuatu yang berguna dengan algoritma garis besar bagian, artikel, dan navigasi (misalnya, mereka dapat membuat daftar isi otomatis, seperti yang dilakukan Word, atau membuatnya dapat ditemukan di pembaca layar seperti judul dan daftar dan tautannya sekarang), kita sudah memiliki HTML yang tepat kita dapat membuat hal-hal yang akan datang alih-alih melakukan simulasi.
- Teknologi pendukung menyukai hal ini. Pengguna diberitahu bahwa navigasi terdiri dari daftar enam item, atau mereka dapat menghindari semuanya dan melompat dari satu judul ke judul lainnya.

Tentu saja ada kekurangannya. Menggunakan lebih dari satu kontrol tab ini berarti Anda perlu membuat ID unik untuk semua target yang berbeda. Namun, harus kita akui — seseorang yang tidak memahami bahwa sebuah ID itu unik akan mengalami kesulitan dalam coding sama sekali. Dalam dokumentasi widget, Anda akan menjelaskan bahwa setiap ID juga menjadi bagian dari URL dan dengan demikian membuat URL yang dapat dibaca sebagai bonus.

Ide tentang HTML harus menjadi pemikiran utama kita sebelum kita tergila-gila pada fitur tambahan dan efek khusus. Ketika semuanya rusak, inilah yang didapat pengguna kami. Browser sangat pemaaf saat ini; parser HTML5 telah dibuat agar kompatibel ke belakang. Hal ini pasti terjadi, karena dosa markup yang sangat buruk yang kami dan alat kami lakukan di masa lalu dalam upaya mendukung browser buruk yang memerlukan peretasan demi peretasan agar antarmuka dapat digunakan. Namun, kompatibilitas ke belakang ini tidak memberikan kita kebebasan untuk melakukan apa pun yang kita inginkan dan mengandalkan browser untuk memperbaikinya untuk kita. HTML yang bersih seperti tata bahasa yang baik. Ini tidak mutlak penting tetapi membuat apa yang Anda lakukan lebih mudah dipahami dan diterjemahkan.

Menulis HTML yang semantik, bersih, dan mudah dipahami sudah ketinggalan zaman. Hal ini menyedihkan karena ini adalah lingua franca Web dan sepertinya kita kehilangan banyak kosa kata ketika kita terlalu tertarik dengan pustaka widget dan solusi CMS. Sekarang browser telah berpindah dan banyak masalah rendering “browser yang tidak akan disebutkan namanya lagi” tidak lagi menjadi masalah, kita harus melihat semua hal menakjubkan yang diberikan HTML kepada kita dan berhenti menggunakan kata-kata acak. konstruksi karena dapat dengan mudah ditata.

Contoh paling umum dari penyalahgunaan HTML adalah menambahkan tautan yang tidak mengarah ke mana pun. A # sebagai atribut href atau, lebih buruk lagi, javascript:void(0), berarti Anda akhirnya menulis HTML untuk sebuah skrip, dan bukan untuk dirender oleh browser atau pengguna untuk berinteraksi. Jika Anda perlu memanggil skrip, gunakan elemen tombol. Mereka dapat ditata dengan indah, memiliki dukungan keyboard bawaan dan berbagai status, dan bahkan dapat dinonaktifkan menggunakan atribut. Dengan menggunakan pemilih atribut di CSS Anda, Anda dapat mengatur gayanya. Tautan lebih dari sekadar pemrakarsa fungsi skrip: tautan memungkinkan pengguna mengeklik kanan tautan tersebut untuk mendapatkan menu konteks yang penuh dengan fungsi berguna seperti “Bookmark”, “Simpan sebagai”, atau “Buka di tab baru”. Semua ini tidak masuk akal jika URL-nya #, bukan?

Bentuknya juga sudah berkembang pesat. Atribut sederhana yang diperlukan pada elemen input berarti pengguna tidak dapat mengirimkan formulir sampai data tersebut dimasukkan. Menambahkan atribut pola memungkinkan Anda menentukan aturan dalam bentuk ekspresi reguler. Di masa lalu, semua ini memerlukan banyak JavaScript — sekarang sudah ada di browser secara default.

Kita akan membahas lebih banyak contoh seperti ini seiring berjalannya bab ini. Untuk saat ini, buka mata Anda dan lihat apa yang ditawarkan HTML5 kepada Anda, tanpa reaksi spontan yang mengatakan bahwa fitur tertentu tidak berfungsi di browser favorit Anda. Kami membangun untuk serangkaian kondisi baru dan untuk orang-orang yang tidak mengalami kesulitan yang sama dengan pengembang kami. Memiliki kosakata yang kaya adalah hal yang luar biasa. Kita tidak hanya harus melek web, kita juga harus menjadi penyair mark-up.

Kurangnya Dukungan adalah Peluang

Jika browser lama tidak dapat melakukan sesuatu, kami memiliki kesempatan untuk mengujinya dan tidak menyediakan fungsionalitas tersebut. Dalam kebanyakan kasus, fungsionalitasnya hanya sekedar bagus untuk dimiliki dan tidak diperlukan. Satu kesalahan besar yang kami buat di masa lalu, dan terus kami lakukan, adalah memberikan kode yang terstandarisasi dan sangat menuntut pada teknologi yang tidak standar dan ketinggalan jaman. Ya, ini tentang OldIE, begitu kami menyebutnya, tetapi juga menyangkut masalah baru browser anak: browser bawaan versi Android dan iOS yang lebih lama.

Sebagai pencipta Web, kita mempunyai dua tugas. Pertama, kita harus menghadirkan antarmuka kerja yang menarik dan menyenangkan untuk digunakan. Kenikmatan adalah tujuan yang berharga, tetapi bagian terpenting adalah kerja keras. Tugas kedua kita adalah menggunakan apa yang diberikan oleh pembuat browser untuk membuat hasil pemenuhan tugas pertama sehebat mungkin, dan dapat dipertahankan semaksimal mungkin bagi orang yang kita beri kode.

Apa yang kita lakukan untuk mencapai hal itu? Setiap kali teknologi baru keluar, kami melakukan polyfill dan menambal serta menambahkan perpustakaan untuk memberikan teknologi tersebut ke browser yang tidak boleh digunakan lagi. Kami mendasarkan hal ini pada kesalahpahaman bahwa premis Web adalah memberikan pengalaman yang sama kepada semua orang. Seringkali, ini bukan ide kami, melainkan ide yang didiktekan oleh manajer, klien, atau rencana proyek kami. Kami mencoba untuk membuat pengalaman mereka yang menggunakan teknologi ketinggalan jaman sehebat mungkin, karena kami menganggap salah jika memberikan mereka pengalaman yang lebih sedikit dibandingkan pengguna yang memiliki lingkungan terkini dan terus diperbarui. Ini bukan tentang Web. Tidak ada yang memaksa kami untuk mendukung teknologi lama dengan fitur-fitur yang tersedia di teknologi baru.

Premis Web adalah untuk menyampaikan konten kepada semua orang tanpa memandang kemampuan, kemampuan teknis dan pengetahuan, atau lokasi geografis. Teknologi web memungkinkan kita melakukan hal tersebut, namun hanya jika kita menggunakannya dengan bijak dan tidak mencoba memberikan pengalaman yang sama kepada semua orang, hal ini akan membuat kita semua kecewa karena kita berlomba untuk memenuhi kebutuhan yang paling rendah.

Kita sepertinya terobsesi dengan pertanyaan, “Apakah ini akan berfungsi di browser X?” dan gunakan segala macam trik dan solusi untuk membuatnya berhasil. Kita menyia-nyiakan banyak upaya untuk mendapatkan hasil yang tidak memuaskan dan ini meningkatkan rasa frustrasi kita. Jika kami memberikan fungsionalitas pada browser melalui peretasan atau solusi, kami juga akan terus mendukung dan menguji browser ini.

Yang lebih bermanfaat adalah bertanya, “Apa yang tidak berfungsi di browser ini?” lalu gunakan jawabannya untuk menentukan teknologi pendukung mana yang akan kami terapkan. Saat Anda menyertakan style sheet saat ini, Anda tidak perlu khawatir sama sekali tentang browser yang tidak mendukung CSS, Anda tahu bahwa browser hanya akan menerapkan apa yang mereka pahami, jadi tidak ada kemungkinan hal itu menimbulkan

masalah. Jarum kompas yang menunjuk ke selatan sama bergunanya dengan jarum kompas yang menunjuk ke arah yang benar Anda hanya perlu menggunakannya dengan benar.

Contoh yang bagus dari hal ini adalah situs web Smashing Magazine. Ini menggunakan desain responsif menggunakan media queries CSS. Ini tidak didukung oleh OldIE, itulah sebabnya, pada awalnya, situs web ini menggunakan patch JavaScript yang disebut respond.js untuk membuat versi IE yang lebih lama juga mengubah desain ketika ukuran layar berubah.

Namun, hal ini ternyata berlebihan, karena yang dibutuhkan hanyalah desain dengan lebar tetap untuk IE lama, membiarkan browser memutuskan apakah desain responsif sesuai kemampuannya atau tidak. Bagaimana kalau kita tidak meninggalkan siapa pun selain membangun teknologi generasi berikutnya? Berikan hanya apa yang dapat diterima oleh browser lama: sedikit CSS dan HTML yang benar-benar berfungsi. HTML semantik yang memiliki makna dan memicu fungsionalitas yang ditawarkan setiap browser sejak awal; tautan yang mengarah ke suatu tempat; URL yang dapat di-bookmark; navigasi yang menjadi benih sejarah browser; formulir yang divalidasi dan diproses di server dan dikirim kembali ke browser.

Hanya dengan begitu kita harus menambahkan lapisan demi lapisan kehebatan untuk browser yang dapat menanganinya. Misalnya, ambil addEventListener() yang bagus dan terstandarisasi. OldIE tidak memahaminya, jadi kami menulis pengisi untuk menimpa lampiranEvent(). Rencana yang buruk. Ini adalah perangkat lunak pemberat yang akan kami bawa selama bertahun-tahun ke depan dan melayani subkelompok kecil pengguna yang akan semakin kecil. Mengapa tidak menggabungkan semua JavaScript kita ke dalam if (window.addEventListener) {} dan tidak pernah mengganggu OldIE dengan JavaScript menuntut yang kita tulis saat ini?

Logika yang sama berlaku untuk CSS. Jika kita menggabungkan bagian-bagian CSS yang menakjubkan dan indah dalam kueri media, OldIE dan browser bawaan akan menjadi lebih bijaksana dan tidak akan mencoba menguraikan apa yang kita berikan kepada mereka. Browser yang ketinggalan jaman sudah tidak digunakan lagi; kita tidak boleh mengganggu mereka dengan tuntutan yang semakin banyak untuk mengeksekusi kode yang tidak diperuntukkan bagi mereka dan tidak akan pernah berjalan lancar. Jika Anda menggunakan pustaka pengisi untuk mendukung browser lama, Anda juga membebani diri Anda dengan pengujian di dalamnya untuk memastikan pengguna browser tersebut menerima pengalaman yang lancar dan indah. Menguji browser usang di lingkungan pengembangan yang kita miliki saat ini adalah sebuah tugas, dan hanya menambah banyak waktu yang tidak menyenangkan pada beban kerja kita. Mengapa melakukan itu?

Mulailah dengan kode dasar yang berfungsi di semua browser, lalu tambahkan ke dalamnya dan pastikan bahwa browser yang tidak boleh digunakan sama sekali tidak mendapatkan kode yang mungkin membuat mereka tersedak Anda akan membuat semua orang senang. Hal-hal yang belum pernah saya lihat, tidak boleh saya lewatkan. Hal-hal yang saya lihat tidak berhasil, membuat saya frustrasi. Kurangnya dukungan adalah peluang besar untuk tidak menjanjikan hal-hal yang tidak dapat dan tidak boleh Anda tawarkan.

3.1 KODE KHUSUS BROWSER TIDAK DAPAT DIPERCAYA

Setiap baris kode yang Anda tulis dengan awalan browser hanya menambah kumpulan kode yang akan segera rusak. Peluncuran iPhone dan kehebohan berikutnya dalam komunitas pengembang membuat bulu kuduk saya merinding. Saya teringat dengan jelas saat orang-orang mengatakan kepada saya bahwa segala sesuatu yang hanya berfungsi di IE6 dan tidak ada browser lain adalah hal yang harus diikuti oleh setiap pengembang, karena tidak akan ada browser lain di masa depan. Ternyata itu sampah, begitu pula semua kode di Web yang sekarang hanya berfungsi di iPhone generasi pertama, atau bahkan memblokir browser lain agar tidak mengakses konten halaman sama sekali.

Jika Anda pernah menemukan tombol putih dengan teks putih, ada kemungkinan besar bahwa pengembang menggunakan webkit linear gradient dan tidak menggunakan apa pun pada definisi latar belakang CSS tombol tersebut. Ini tidak cerdas, modern, dan pragmatis. Ini adalah kode rusak yang hanya berfungsi untuk waktu yang sangat singkat dalam lingkungan yang cepat berlalu. Menulis kode khusus browser seperti merilis film hanya di VHS. Tampaknya ini merupakan solusi yang murah dan cepat pada saat itu dan menyelesaikan pekerjaan, namun hal ini menyisakan ribuan salinan yang tidak terjual yang tidak dapat dilihat oleh siapa pun karena perangkat kerasnya sudah usang.

Jika Anda menggunakan kode awalan, setidaknya bungkus dalam kondisi pengujian. Namun, cara terbaik adalah mengawalinya dengan fallback yang berfungsi di mana saja dan mengikutinya dengan kode standar. Dalam contoh tombol dengan teks putih dan latar belakang putih, semua akan baik-baik saja jika pengembang menentukan latar belakang terlebih dahulu dan gradien linier khusus browser setelahnya.

Kode khusus browser adalah hal yang harus dilakukan. Jika Anda tidak dapat mengunjungi kembali dan memperbaiki kode setelah fungsinya distandarisasi maka Anda menulis kode yang rusak dan tidak dapat diandalkan. Lingkungan berubah. Itu sebabnya kami punya standar. Gunakan Campuran Teknologi, Masing-masing Teknologi untuk Kegunaannya yang Terbaik. Sangat menggoda untuk melakukan semuanya dalam JavaScript, tetapi Anda sebaiknya tidak melakukannya.

Saat Anda memiliki palu baru yang mengilap, semuanya tampak seperti paku dan berubah menjadi jempol begitu Anda mulai memalu. Kami memiliki teknologi yang sangat kuat di Web stack, yang paling kuat mungkin adalah JavaScript. CSS adalah pilihan kedua, mengingat penambahan terbaru. Secara umum, dimungkinkan untuk melakukan segalanya di browser dengan JavaScript. Anda dapat membuat seluruh aplikasi hanya dengan tag body di HTML Anda. Anda dapat membuat browser usang berperilaku seperti versi modernnya, Anda dapat berinteraksi dengan windows dan DOM, serta memindahkan data ke dan dari server. Mempertahankan kendali membuatnya tergoda untuk melakukan segala sesuatu dengan JavaScript.

Seluruh kerangka kerja telah dibangun berdasarkan premis tersebut (qooXdoo2, misalnya) dengan kinerja luar biasa dan prinsip pemrograman yang hebat. Namun, cepat atau lambat, mereka akan ketinggalan jaman dan melakukan hal-hal dalam JavaScript yang tidak perlu lagi dilakukan dengan mengorbankan komputasi prosesor. Ingatkah saat kita membuat

banyak gambar kecil untuk menambahkan sudut membulat pada sesuatu? Dan betapa kerennya sekarang memiliki radius batas CSS dan gradien latar belakang untuk mengubah tampilan dan nuansa, tanpa perlu membuat ulang semuanya atau memastikan pengguna kami tidak menyimpan cache yang lama?

Hal yang sama sedang terjadi pada JavaScript saat ini. Banyak hal yang kami gunakan dalam JavaScript (dan terutama perpustakaan seperti jQuery) sekarang ditangani oleh CSS: transisi, animasi, kueri media. Hal ini memungkinkan kami menciptakan pengalaman yang sangat mulus dan memanfaatkan kerja browser untuk memastikan pengalaman tersebut tetap lancar. Jika kami menggunakan JavaScript, kami mendapatkan kontrol lebih besar namun tanggung jawab kami yang lebih besar adalah memastikan semuanya berjalan lancar. Dan tanggung jawab ini sangat bergantung pada browser dan konteks teknologi — konteks yang berubah dari bulan ke bulan.

Satu hal yang sangat memperlambat aplikasi dan situs web adalah akses DOM. Namun kesederhanaan API akses DOM jQuery melahirkan seluruh generasi pengembang yang pelajaran pertamanya adalah bagaimana menulis sebuah loop (tersembunyi dalam pemilih $() yang dirantai ke suatu metode) untuk mengakses elemen dalam dokumen untuk mengubah tampilan dan merasa. Dalam banyak kasus, hal itu tidak diperlukan. Anda dapat menggunakan penanganan peristiwa, atau pemeriksaan kondisi dalam JavaScript dan menambahkan kelas ke badan atau elemen induk dari apa yang ingin Anda ubah dan biarkan CSS melakukan sisanya. Itu adalah sesuatu yang dilakukan browser saat melakukan reflow dan rendering, jadi mengapa tidak mendukungnya?

Jika Anda memeriksa apa yang perlu Anda capai sebelum tertarik dengan kesederhanaan rangkaian dan pemilihan elemen dalam JavaScript, Anda akan menemukan bahwa sebagian besar hal yang perlu Anda lakukan sekarang melibatkan penambahan dan penghapusan kelas, dan memuat konten. JavaScript bagus untuk perubahan seketika yang dipicu oleh interaksi; CSS sangat bagus untuk membuat perubahan muncul. Pembuat browser melakukan pekerjaan yang baik dengan memungkinkan kami, di alat pengembang, memanfaatkan rendering oleh browser dan melihat apa yang terjadi.

RequestAnimationFrame() memungkinkan Anda mengubah berbagai hal dan hanya menampilkannya saat hasilnya muncul di layar. Selain itu, ketika tab browser tempat skrip Anda dijalankan tidak aktif (saat pengguna berada di tab atau jendela lain), animasi tidak berjalan, sehingga tidak membuang waktu komputasi dan memperpendek masa pakai baterai. Sebaliknya, setTimeout() berharap browser siap menggambar dan berjalan baik pengguna melihat animasi Anda atau tidak.

Animasi dan transisi dalam CSS dipercepat perangkat keras; Animasi JavaScript tidak. Begitu juga dengan transformasi dalam CSS, yang berarti bahwa transform: Translate(x,y) mengalahkan position: absolute; atas: x ; kiri: kamu; ketika menyangkut kinerja. Saya mendorong siapa pun yang bekerja di Web untuk selalu mengikuti perkembangan teknologi browser dan dukungan standar. Hanya ketika pengembang menggunakan apa

browser menawarkan kita dapat membuat Web lebih baik. Jika fungsionalitas yang ditentukan dalam standar tidak digunakan, pembuat browser cenderung tidak

mendukungnya. Mengapa mendukung input type="range" ketika sebagian besar pengembang menggunakan div dan plugin jQuery untuk mengubahnya menjadi slider? Anda mungkin tidak mempunyai waktu untuk memperbarui kode Anda terus-menerus, jadi bermalas-malasanlah dalam hal ini — gunakan apa yang cocok untuk pekerjaan itu dan tingkatkan sesuai kebutuhan dengan JavaScript. Kami hanya dapat menyelesaikan masalah ayam dan telur dalam dukungan standar dengan meminta pembuat browser menambahkan dukungan tersebut ketika kami dapat menunjukkan bahwa produk kami siap untuk itu.

Setiap solusi Web yang baik bekerja dengan menyerahkan tugas yang tepat kepada teknologi yang tepat. Kita sering kali terlalu bersemangat dengan spesialisasi kita dan ingin segalanya menjadi mungkin sesuai dengan apa yang kita sukai. Sebuah band menciptakan musik ketika anggotanya memainkan alat musik yang ingin mereka mainkan. Sangat sedikit orang yang bisa bermain drum, gitar dan trombon, serta menyanyi — tidak ada yang bisa bermain pada saat yang bersamaan. Pilih dan pilih, jangan coba-coba mengganti.

3.2 AJUKAN PERTANYAAN “IF”.

Apa pun yang Anda lakukan harus dibungkus dengan “jika”, jadi hanya lingkungan yang dapat menerapkan apa yang Anda inginkan yang melakukannya. Menggunakan “jika” adalah alat yang ampuh, baik dalam percakapan maupun coding. Pengkodean yang baik bersifat defensif ia menguji kedalaman air sebelum menyelaminya. Solusi kami tidak boleh hanya berasumsi bahwa pengguna tertentu memiliki apa yang diperlukan untuk mengonsumsi konten yang kami berikan kepada mereka. Menggabungkan pemuatan konten dalam kasus pengujian adalah cara yang sangat ampuh untuk mencapai pengalaman paling menyenangkan bagi pengguna yang berbeda kita akan membahasnya lagi nanti.

Namun, “jika” dapat digunakan untuk lebih banyak hal. Anda dapat menanyakan lingkungan saat ini tentang kemampuannya sebelum menerapkan fungsi tertentu. Anda dapat menghentikan browser untuk mencoba menjalankan seluruh blok JavaScript dengan terlebih dahulu menanyakan apakah browser dapat melakukan hal paling sederhana sekalipun. Semakin sedikit kode yang kita paksa untuk diurai oleh browser, semakin baik kinerjanya. Mengapa membacakan seluruh buku dalam bahasa Islandia kepada teman tanpa menanyakan apakah mereka memahami bahasa tersebut?

Terkadang perlu menerapkan beberapa tipu daya untuk menghindari pengalaman yang sangat buruk. Video HTML5 adalah salah satu contohnya. Anda mungkin pernah melihat kode demo berikut di beberapa tempat.

Pertama, kalimat ini bukan konten cadangan untuk browser yang tidak dapat memutar video HTML5: ini menjadikan masalah Anda sebagai masalah pengguna, yang sangat membuat frustrasi jika mereka tidak paham teknologi. Bayangkan mencoba melakukan sesuatu dan orang-orang terus mengatakan kepada Anda, “Dapur Anda tidak dapat mengatasi boodiloo.” Oke, apa maksudnya? Cara yang jauh lebih baik adalah dengan memberi pengguna tautan ke video sebagai penggantinya:

```
<video src="kittens.mp4" controls>
```

```
<a href="kittens.mp4">Check the video of kittens</a>.</video>
```

Dengan begitu, pengguna browser lama dapat menonton video di pemutar media sistem operasi mereka hanya dengan mengikuti tautan. Anda dapat menjadikannya lebih baik lagi dengan menambahkan tangkapan layar. Hal ini tidak hanya memberikan bantuan visual tetapi juga memungkinkan situs media sosial seperti Facebook menampilkan pratinjau thumbnail semua orang menang. Hampir saja.

Browser yang memahami elemen video HTML5 tetapi tidak memahami format file MP4 (karena ini bukan format terbuka) tidak akan menampilkan konten cadangan. Sebaliknya, mereka menampilkan kotak abu-abu dengan pesan tentang video yang tidak dapat diputar, dan tidak menawarkan tautan untuk diikuti.

Ini menjengkelkan, tapi sesuai dengan definisi standar. Membacanya, ada cara untuk memeriksa apakah suatu video dapat diputar.

Katakanlah kita memiliki HTML ini:

```
<video controls>
       <source src="dynamicsearch.mp4" type="video/mp4"></source>
       <a href="dynamicsearch.mp4">
   <img src="dynamicsearch.jpg"
       alt="Dynamic app search in Firefox OS">
 </a>
 <p>Click image to play a video demo of
   dynamic app search</p>
</video>
```

JavaScript berikut menunjukkan cara untuk membuat browser (yang mendukung elemen video tetapi tidak dapat memutarnya menampilkan konten cadangan:

```
if (window.addEventListener && document.querySelector) {
      var v = document.querySelector('video'),
            sources = v.querySelectorAll('source'),
            lastsource = sources[sources.length-1];
lastsource.addEventListener('error', function(ev) {
      var d = document.createElement('div');
      d.innerHTML = v.innerHTML;
      v.parentNode.replaceChild(d, v);
 }, false);
}
```

Apa yang terjadi di sini? Saat browser gagal memutar video, browser akan mengaktifkan pengendali kesalahan pada elemen sumber terakhir di elemen video. Jadi, kami menguji apakah browser memahami addEventListener() dan querySelector() (yang sebenarnya merupakan cara standar $() jQuery) dan kemudian mendapatkan elemen sumber

terakhir di elemen video. Jika ada peristiwa kesalahan yang terjadi, kami membuat elemen div dan mengganti video dengan yang itu.

Dengan bertanya pada browser apa yang dapat dilakukannya, kita dapat memperbaiki pengalaman fallback yang tidak memuaskan dan menjadikannya berfungsi untuk semua orang. Bukan pengalaman yang sama, namun tetap berhasil. Tindakan tersebut memerlukan pemikiran dan penelitian (misalnya, saya yakin bahwa elemen video memicu peristiwa kesalahan, bukan sumber terakhir), namun hal ini bermanfaat karena kami tidak meneruskan masalah kami kepada pengguna. “Jika” adalah konstruksi yang hebat dan membuat kode Anda lebih baik dan tidak bergantung pada lingkungannya.

Tulislah Sebanyak Kebutuhan, Bukan Sedikit Mungkin

Mari pikirkan tentang apa yang kita tulis sebelum kita menulisnya, daripada menambahkan banyak solusi kecil dan cepat. Munculnya jQuery memulai sesuatu yang menjadi fetish di komunitas pengembang Web: semakin sedikit kode yang Anda tulis, semakin efektif Anda dianggap; dan semakin banyak kode yang Anda tulis yang melakukan sesuatu dalam waktu singkat, semakin hebat yang Anda dapatkan. Hal ini belum tentu benar.

Ya, mengetik lebih sedikit berarti kita bisa melakukan hal-hal lain, tapi jika harus mengabstraksikan apa yang kita lakukan ke dalam perpustakaan dan plugin, hal ini bisa menjadi rumit. Apa yang tidak pernah kami ketahui di Web adalah apa yang dilakukan atau dimiliki pengguna kami. Jadi mengandalkan abstraksi yang tidak kita kendalikan dan yang prosesnya tidak kita pahami tampaknya berbahaya. Kami sudah merasakan hal ini. Banyak solusi kecil yang bergantung pada jQuery atau perpustakaan lain memiliki kinerja yang sangat buruk di perangkat seluler.

Plugin mengesankan yang memberi tahu kami bahwa ukurannya hanya beberapa KB atau hanya memerlukan tiga baris kode untuk diterapkan cenderung mengirimkan jumlah byte yang jauh lebih besar kepada pengguna kami, terbungkus dalam banyak permintaan HTTP dan menambah jumlah keseluruhan JavaScript di aplikasi kami. Bukan hal yang aneh melihat orang menggunakan beberapa perpustakaan sekaligus karena mereka menyukai widget yang dibangun di masing-masing perpustakaan. Ini bukan tujuan perpustakaan browser. Mereka seharusnya menyelesaikan masalah dan membuat pekerjaan kita lebih mudah, bukan lebih sulit. Hal ini sangat berbahaya ketika perpustakaan berhenti dipelihara atau diperbarui dan ternyata menyebabkan masalah kinerja. Tentu saja, skrip pembangunan atau penerapan yang baik akan mengatasi masalah ini — satu hal lagi yang perlu dipelajari dan dipelihara. Mengapa menambahkan lebih banyak untuk membatalkan apa yang telah dilakukan sebelumnya, padahal kita harus menghindari membebani solusi kita dengan bantuan yang kecil dan tampaknya berguna?

Seperti yang ditunjukkan sebelumnya, penggunaan HTML yang lebih semantik dapat memicu manfaat fungsionalitas browser bawaan. Hal yang sama berlaku untuk semua kode kita. Jauh lebih penting untuk membangun dasar yang kokoh pada produk kita, inti yang mudah dibaca dan dipahami yang dapat ditambahkan ke dalamnya. Kami melakukan ini dengan menulis kode yang bersih dan mudah dipahami yang memiliki nama variabel dan metode yang masuk akal yang melakukan satu hal dengan baik. Bukan dengan menulis API

abstraksi agar orang lain dapat mencapai banyak hal dengan cepat tanpa mengetahui apa yang mereka lakukan dan menyembunyikan kerumitannya.

Anjuran untuk kode pendek terkadang hanya menjadi alasan untuk bermalas-malasan. Hal ini dapat menyebabkan thread konyol di forum dan papan pesan di mana orang menggunakan elemen <b> dan <i> alih-alih dengan nama kelas yang masuk akal untuk menghemat waktu, atau mereka menghilangkan tanda kutip di sekitar atribut untuk mencegah penekanan tombol lainnya agar tidak ditambahkan ke beban kerja dalam dokumen panjang. Latihan mana pun berarti Anda menghemat waktu sekarang, tetapi keduanya menyebabkan kebingungan di kemudian hari ketika Anda perlu memperluas kode Anda. Menghilangkan tanda kutip di sekitar atribut, misalnya, hanya berfungsi sampai Anda perlu menambahkan nilai lain yang dipisahkan spasi (misalnya dengan atribut class), jadi mengapa tidak menambahkannya sekarang juga agar lebih banyak nilai yang bisa ditambahkan?

Menggunakan kode yang tidak dapat dibaca, singkat, dan terburu-buru untuk menyelesaikan masalah dengan cepat tidak berarti Anda lebih efektif. Artinya, Anda akan mendapat masalah di kemudian hari. Pemeliharaan kode Anda adalah bagian terpenting untuk dipikirkan dan Anda hanya dapat melakukannya saat Anda menulisnya. "*Keluarkan ini sekarang, kita bisa membereskannya nanti*" adalah kebohongan yang sama dengan menyetujui bahwa "*Saya telah membaca syarat dan ketentuan secara lengkap."* Bangun untuk orang yang mengambil alih dari Anda, bukan untuk kondisi browser saat ini dan Anda akan menjadi orang yang hebat untuk diajak bekerja sama.

Ini Bukan Tentang Apa Yang Bisa Anda Tambahkan, Ini Tentang Apa Yang Tidak Bisa Kita Hilangkan

Fungsionalitas dasar harus selalu ada. Hal yang aneh tentang perangkat lunak adalah bahwa ia bisa salah.

Kita yang terlalu bersemangat cenderung melupakan hal itu. Menggunakan kerangka kerja MVC (Model-View-Controller) yang mewah untuk membangun aplikasi satu halaman dengan pemisahan tugas back-end yang sangat bersih adalah hal yang menggoda untuk dilakukan. Bagaimanapun, inilah yang dilakukan oleh anak-anak keren dan digunakan oleh Google dan perusahaan lain, yang harus berkembang hingga tak terbatas dan lebih jauh lagi.

Kami sangat bersemangat untuk menambahkan fitur-fitur baru ke produk kami karena kami melihatnya digunakan di tempat lain dan kami bosan dengan solusi yang sama dari tahun ke tahun. Rasanya seperti kita tidak mengalami kemajuan dan, yang lebih penting, rasanya aneh karena hampir tidak ada pembelajaran ilmu komputer di universitas yang dapat diterapkan begitu kita mengambil pekerjaan di bidang pengembangan Web. Pasti ada yang salah dan pengembangan Web tidak cukup berkembang untuk tugas pemrograman yang tepat, bukan? Salah.

Sebenarnya tidak ada komputasi yang sebanding dengan pengembangan Web, karena kami tidak mengkompilasi apa yang kami tulis ke dalam bytecode yang dioptimalkan untuk lingkungan tertentu. Kode kami keluar dan diubah di komputer, perangkat genggam, mobil, lemari es, jam tangan, dan apa pun yang akan dimiliki pengguna di masa depan yang mendukung Web (mungkin kacamata?). Itulah sebabnya banyak praktik terbaik dari masa lalu

yang dimasukkan ke dalam Web, dan meskipun merupakan ide bagus, namun belum tentu memberikan hasil yang diinginkan.

Pada akhirnya, kita harus selalu menawarkan fungsionalitas dasar kepada pengguna atau kita akan menciptakan banyak lingkungan berdinding kecil di Web. Tidak, Anda tidak bisa mengharapkan pengguna memiliki browser tertentu. Tidak, orang tidak dapat meningkatkan resolusi ponselnya agar sesuai dengan kebutuhan Anda. Tidak, kecepatan koneksi yang Anda miliki di mesin pengembangan Anda tidak sama dengan yang dialami setiap pengguna tidak dalam jangka panjang.

Apa fungsi dasar yang diperlukan? Sederhananya, ini memungkinkan pengguna untuk melakukan apa yang mereka inginkan terlepas dari kegagalan teknologi apa pun. Contoh buku teks tentang hal ini baru-baru ini terjadi pada salah satu perusahaan Web terbesar di luar sana: Google. Selama setengah hari penuh, pada tanggal 4 Februari 2013, halaman download Chrome tidak tersedia. Anda dapat mengklik tombol unduh tetapi tidak terjadi apa-apa.

Membuka alat pengembang Chrome mengungkapkan dua hal: pertama, HTML tombolnya adalah sebagai berikut:

```
<a class="button eula-download-button"
 href="javascript:void(0)"
 data-g-label="download chrome"
 data-g-event="cta"> … </a>
```

Kedua, konsol kesalahan menyambut Anda dengan "`TypeError: chrm.download tidak terdefinisi.`"

Apa yang telah terjadi? Ya, ada yang tidak beres di JavaScript dan menghilangkan tombol itu. javascript:void(0) bukan URL yang valid dan tidak ada urusan dengan atribut href. Ini adalah tanda peringatan bahwa di suatu tempat dalam proses pengembangan, Google menyerah dan menciptakan segala sesuatu dalam JavaScript. Saat diselidiki, saya menemukan bahwa tujuan kode tersebut adalah untuk menunjukkan perjanjian lisensi pengguna akhir sebelum mengunduh (seperti yang ditunjukkan dalam nama kelas), dan JavaScript secara otomatis mendeteksi sistem operasi untuk menyediakan paket pemasangan yang sesuai untuk Chrome.

Keduanya merupakan kegunaan yang sangat baik untuk laman ini, namun cara penerapannya membuat Google kehilangan setengah hari unduhan Chrome. Solusinya mudah: alih-alih mengarahkan tautan ke JavaScript sebaris kecil yang tidak melakukan apa pun, tautan tersebut harus mengarah ke laman unduhan yang mencantumkan semua versi Chrome yang dapat diunduh. Bagaimanapun, ini bisa menjadi sumber daya yang bagus, karena terkadang saya ingin mengunduh versi yang bukan untuk OS saya (misalnya, jika koneksi saya cepat di suatu tempat). Anda masih dapat menambahkan event handler ke tautan untuk melakukan semua hal penting lainnya yang dilakukan dalam JavaScript yang tidak pernah dipanggil.

Mengarahkan ulang ke halaman EULA di server sangatlah mudah, begitu juga dengan mengirimkan informasi tentang penginstal OS mana yang diperlukan. Ini adalah kasus klasik

penggunaan alat yang salah hanya karena pengembang bisa melakukannya. Intinya, penting untuk tidak merusak Web dengan apa yang kita lakukan: formulir harus dikirim ke kontrol sisi server; tautan harus mengarah ke sumber daya nyata; media harus ditautkan dan disematkan, bukan hanya disematkan dengan harapan browser akan melakukan segalanya dengan benar.

Setelah kita memilikinya, kita dapat menambahkan apa pun yang kita inginkan. Tapi serius, mencoba mengganti mekanisme transportasi dasar Web dengan konstruksi kita sendiri mungkin tampak lebih cepat namun akan selalu sangat rentan terhadap kesalahan. Kita mempunyai infrastruktur yang berfungsi kita harus menggunakannya.

3.3 KEGUNAAN MENGALAHKAN KONSISTENSI DI SELURUH BROWSER

Daripada mencoba memberikan pengalaman yang sama kepada semua orang, kita harus selalu menemukan cara terbaik untuk memastikan orang-orang dapat menggunakan apa yang kita bangun. Sesuatu yang menambah kegemukan Web hingga tingkat yang sangat besar adalah gagasan salah arah yang memberikan pengalaman yang sama kepada setiap browser dan setiap lingkungan. Sekali lagi, sebagai pembuat Web, kita tahu bahwa hal ini adalah hal yang bodoh, namun sering kali kita diminta untuk tunduk pada tekanan yang diberikan oleh perencana proyek atau manajer kita. Kita perlu mengambil sikap. Ini adalah ide yang ketinggalan jaman dan sangat picik yang tidak akan pernah mati. Ketika pengembangan Web dimulai, kami sering diberi desain cetak dan diminta untuk membuatnya berfungsi di Web dengan cara apa pun yang diperlukan. Inilah yang akhirnya membuat teks menjadi grafis dan, kemudian, Flash digunakan untuk segala hal yang diperlukan “untuk memenuhi spesifikasi desain.”

Saat ini, kami mencoba menciptakan pengalaman multimedia yang indah, interaktif, dan imersif dengan alat yang ditawarkan HTML5, dan kemudian membuatnya kompatibel dengan lingkungan yang bahkan tidak memiliki sarana dasar untuk menampilkan video. Dan kami membuat situs besar seperti Flash dan menyembunyikan elemen yang tidak dapat ditampilkan di layar ponsel. Kami masih memuat semua konten, dan berpindah dari besar ke kecil. Ketika tampilan resolusi tinggi keluar, kami mulai mengirimkan gambar besar beresolusi tinggi melalui kabel yang tidak akan pernah dapat ditampilkan pada perangkat keras beresolusi rendah yang menerimanya.

Ini tidak masuk akal. Jika kita ingin Web berhasil — dan karena ini adalah platform distribusi paling sederhana di seluruh dunia, kita mempunyai tujuan yang baik — maka kita harus memikirkan kembali pendekatan kita dan membangun antarmuka yang tidak hanya diubah ukurannya namun juga disesuaikan dengan konteksnya. ditampilkan di. Kita perlu mulai melihat desain dan UX sebagai tugas yang bergantung pada konteks, dan tidak mengambil pendekatan satu ukuran untuk semua. Daripada hanya menampilkan sebagian desain pada perangkat layar kecil dan lebih banyak lagi pada layar besar, kita perlu memikirkan apa yang orang ingin lakukan dan apa yang bisa mereka lakukan dengan baik di lingkungan tertentu.

Misalnya saja, saya menelusuri dan memilah-milah feed berita saya di ponsel dan menandainya untuk dibaca nanti. Ketika saya membuka laptop saya, saya membaca secara

detail apa yang saya tandai dan membagikannya kepada dunia. Namun, saya harus menggunakan antarmuka yang kurang lebih sama di kedua perangkat dan saya ingin memiliki antarmuka yang lebih sederhana di ponsel, disesuaikan dengan kebutuhan saya untuk melakukan triase, dan yang membuat berbagi lebih mudah di laptop.

Dengan cara-cara baru untuk berinteraksi dengan konten yang selalu dirancang gerak tubuh, sentuhan, kacamata, tombol di roda kemudi mobil kita harus siap untuk menciptakan pengalaman yang dapat dipesan lebih dahulu dengan cepat dan mudah. Kita tidak dapat melakukan hal tersebut jika kita mencoba menemukan inti dari sebuah desain yang dapat diterapkan di mana saja, atau memulai dengan kerangka kerja yang menjanjikan hal tersebut. Itu tidak akan ada, kecuali Anda tetap sangat sederhana dan tidak menyentuh banyak peluang yang ditawarkan oleh browser dan perangkat keras modern.

Kami harus melepaskan dan menyerahkan beberapa keputusan kepada pengguna kami. Lihatlah pembaca e-book. Saya suka bagaimana Google Play Buku berubah dari lanskap ke potret, dan bagaimana Kindle memungkinkan saya mengubah dari hitam di atas putih menjadi abu-abu di atas hitam ketika saya ingin membaca dalam gelap. Mari kita pikirkan tentang apa yang dilakukan pengguna kita dalam konteks yang berbeda, daripada menyalahkan mereka dan berharap menyembunyikan sebagian saja sudah cukup. Kami belum melakukan cukup banyak hal saat ini, itulah sebabnya browser seluler memiliki mode “Pembaca” atau memungkinkan pengguna untuk memaksakan “versi Desktop” ke layar kecil mereka karena antarmuka seluler mengecewakan dan membuat frustrasi.

Muat Hanya Apa Yang Diperlukan

Mengapa browser harus mendapatkan perpustakaan atau file data jika tidak dapat melakukan apa pun dengannya? Salah satu teknik paling menarik yang kami temukan setelah era DHTML dan sekarang didukung oleh AJAX adalah teknik pemuatan lambat. Intinya, ini berarti kita hanya memuat sumber daya saat kita membutuhkannya. Mengapa pengguna perangkat kecil harus mendapatkan gambar berukuran besar yang tidak pernah dapat ditampilkan dengan memuaskan? Mengapa Anda menyertakan perpustakaan JavaScript yang ditargetkan untuk iOS pada perangkat Android atau bahkan desktop?

JavaScript bagus dalam menguji dukungan dan kemudian memuat sumber daya sesuai permintaan. Kami hanya tidak cukup menggunakannya untuk itu. Saat ini, kami menemukan semakin banyak solusi yang memuat sumber daya kelas atas dalam jumlah besar terlebih dahulu karena caching akan meningkatkan pengalaman saat pengguna menelusuri situs. Ini sia-sia karena tidak membantu pengguna yang tidak akan pernah mendapatkan manfaat dari pengalaman kelas atas tersebut. Ini mungkin tidak menjadi masalah sama sekali bagi kami, dengan koneksi cepat, layar besar, dan prosesor bertenaga.

Tapi ini bukan tujuan kami membangun (kecuali jika Anda membuat alat pengembang) dan kami harus memastikan bahwa pekerjaan kami diuji pada mesin dan ponsel berdaya rendah serta koneksi yang tidak stabil. Semakin kami menunda pemuatan konten yang tidak diperlukan atau kemudian menyimpannya di perangkat pengguna, alih-alih memuatnya berulang kali, solusi kami akan semakin baik. Secara pribadi, menurut saya bilah kemajuan dan animasi merupakan tanda kegagalan. Tidak ada yang menikmatinya di intro Flash dan itu

membuat kami gila di YouTube, jadi mengapa saya harus menunggu beberapa menit karena Anda ingin memuat semuanya terlebih dahulu, daripada menganalisis cara saya berinteraksi dengan kode Anda?

Jadi mari kita berpikir sebelum menambahkan 12 font yang tidak perlu di halaman pertama, kerangka CSS yang kita gunakan untuk membuat tata letak dua kolom, dan pustaka JavaScript wastafel dapur yang kita gunakan untuk menambahkan satu pengendali peristiwa ke sebuah tombol.

Mari kita periksa ruang layar yang tersedia sebelum memuat konten yang hanya perlu disembunyikan karena tidak ada ruang untuk itu. Jangan memuat gambar sebelum pengguna benar-benar menggulirkannya hingga terlihat (untuk menghindari perubahan konten yang tidak sedap dipandang, simpan kotak dengan ukuran yang sama di tempatnya, lalu ganti dengan gambar). Jangan menambahkan musik latar hanya karena menurut kami pengguna menginginkannya — tunggu saja hingga mereka benar-benar senang mendengarnya.

Ada waktu henti alami dalam interaksi dengan aplikasi kami. Misalnya, orang akan meluangkan waktu untuk memasukkan data ke dalam formulir. Jadi mengapa tidak menggunakan waktu itu untuk memuat sumber daya tambahan? Penangan fokus pada bidang teks pertama dapat memicu pengunduhan sumber daya yang bagus. Jika pengguna tidak pernah memasukkan formulir, tidak ada yang perlu terjadi. Ini adalah dunia di mana konektivitas lebih sempit dibandingkan dengan konektivitas desktop. Jangan menyumbat pipa dengan barang-barang yang tidak akan pernah dikonsumsi oleh siapa pun.

3.4 MENGANALISIS EFEK BEATS MENAMBAHKAN FX

Jika Anda ingin menambahkan sesuatu yang berkilau, pastikan benda tersebut tampil berkilau. Tidak ada seorang pun yang menyukai browsernya melambat karena efeknya yang lambat. Secara keseluruhan, kami tampaknya terlalu fokus pada efek visual dalam desain kami. Hal ini terlihat dari prosesi mode desain yang datang silih berganti. Ketertarikan saat ini terhadap situs web gulir ke paralaks akan segera terlihat kuno dan menjengkelkan seperti tampilan pembagi pelangi atau halaman terowongan Flash sekarang.

Pertanyaan yang harus kita tanyakan pada diri kita sendiri adalah berapa banyak efek yang dapat kita tambahkan ke lingkungan tertentu sebelum kita membebani pengguna secara berlebihan. Apakah drop-shadow, sudut membulat, dan gradien benar-benar diperlukan dalam segala hal, atau apakah masuk akal untuk membuatnya dengan CSS hanya untuk lingkungan yang mendukungnya, alih-alih menyimulasikannya dengan gambar dan menambah waktu muat? Beralih dari desain skeuomorfik ke desain datar berarti hanya mengubah CSS tidak perlu mencari dan menghapus gambar yatim piatu di server atau, lebih mungkin, mengabaikannya di sana untuk menambah bobot proyek secara keseluruhan.

Apakah masuk akal jika dua status widget memudar satu sama lain jika itu berarti menambahkan pustaka JavaScript lain dan menghitung fade dalam JavaScript? Transisi CSS dilakukan oleh browser, dan pembuat browser dapat mengubah dan meningkatkan perilakunya. Perangkat keras video dapat menghitungnya, bukan prosesor utama. Jika Anda melakukan transisi dengan tangan di JavaScript, terserah pada Anda untuk membuatnya

berjalan lancar di mana saja dan Anda tidak dapat mengandalkan browser untuk melakukan pekerjaan kotor tersebut untuk Anda. Akankah pengguna benar-benar menyukai drop-shadow yang halus ketika baterai mereka lebih cepat habis dibandingkan tanpa baterai?

Hanya karena sesuatu berfungsi dengan baik di aplikasi asli tidak berarti hal itu sesuai di aplikasi Web atau situs web. Kami hanya akan menyimulasikan lingkungan yang tidak pernah dapat kami tandingi dalam hal kinerja, sehingga menciptakan pengguna yang tidak senang dan kecewa. Tak seorang pun ingin mendapatkan sepotong kue yang enak hanya untuk menyadari pada gigitan pertama bahwa kue tersebut tidak memiliki rasa apa pun. Efeknya bagus untuk dimiliki tetapi tidak boleh menjadi tujuan akhir kita. Mereka bagus untuk mengesankan pengembang Web dan pihak berkepentingan lainnya, namun kita harus meluangkan lebih banyak waktu untuk bertanya kepada pengguna sebenarnya apa tujuan mereka ketika mengunjungi situs kita atau menggunakan aplikasi kita. Terkadang solusi paling sederhana adalah solusi terindah.

Pondasi Yang Kuat Mengalahkan Kemungkinan Penambahan Di Masa Depan

Seringkali kita menambahkan banyak kode atau antarmuka untuk memungkinkan masa depan yang kita pikir akan membutuhkannya. Masa depan ini hampir tidak pernah terjadi. Mari kita membangun untuk saat ini. Mungkin penyebab terbesar dari solusi yang membengkak adalah kita melampaui batas ketika merencanakan arsitektur solusi Web kita. Web sedang populer saat ini dan tidak terdapat cukup pengembang Web untuk memenuhi permintaan produk yang akan dibuat. Dibutuhkan banyak pengembang dan banyak dari mereka berasal dari lingkungan perangkat lunak yang lebih tradisional di mana sistem dibangun dengan satu cara dan hanya satu cara untuk meningkatkan skala dan menjadi kuat. Seringkali kita sekarang memasukkan pendekatan ini ke dalam Web, dan mengklaim secara kultus bahwa apa pun yang tidak dibangun berdasarkan prinsip OO (Berorientasi Objek) dan dengan pendekatan MVC tidak akan pernah berskala dan dapat dioperasikan.

Namun, produk yang dibuat melalui web selama bertahun-tahun menunjukkan bahwa hal ini tidak terjadi. Tentu saja, pendekatan yang lebih bersih akan menghasilkan kode yang lebih baik dan mudah dipelihara, namun hal ini tidak akan mengorbankan antarmuka front-end yang bergantung pada JavaScript untuk bekerja, dan membuat kode yang berbeda untuk browser yang berbeda dalam banyak kasus, untuk browser yang berbeda. versi browser yang sama. Apa yang banyak dilupakan oleh pendekatan pengembangan adalah bahwa aplikasi satu halaman mengharuskan Anda untuk tidak hanya menulis aplikasi Anda, namun juga mengganti banyak fungsi yang dilakukan untuk Web dengan HTTP dan browser, seperti penyimpanan riwayat dan pembatalan fungsi. Intinya, Anda menjalankan aplikasi klien tebal di lingkungan klien tipis yang tidak dapat Anda kendalikan.

Ide yang bagus adalah memikirkan apa yang ingin Anda capai sebelum Anda menggunakan solusi terbaik dari kerangka kerja MVC dan beralih ke bahasa yang disebut-sebut sebagai bahasa yang lebih baik daripada JavaScript hanya untuk menghasilkan JavaScript. Banyak hal di Web berubah. Dan dalam banyak kasus, ini berarti mengganti seluruh basis kode tetapi menduplikasi fungsinya. Banyak aplikasi dan situs web yang kami hasilkan, pertama-tama, bertujuan untuk membuat orang memasukkan konten ke dalamnya dan

membuat konten tersebut tersedia bagi orang lain. Kode aplikasi itu sendiri harus memainkan peran sekunder.

Setiap kali sebuah perusahaan Web dibeli oleh perusahaan yang lebih besar dan perlu memperluas jangkauannya hingga jutaan pengguna, seluruh basis kodenya diganti atau diubah agar sesuai dengan kerangka kerja yang sudah ada dan telah terbukti mampu digunakan. Anda tidak perlu membuat Google Mail atau Facebook. Anda harus memikirkan untuk menciptakan apa yang Anda perlukan saat itu, dan mempersiapkan diri untuk menggantinya dalam waktu dekat.

Hal ini tampaknya berlawanan dengan diktum lama kami: bahwa pemisahan perhatian menjadi HTML, JavaScript, dan CSS berarti Anda dapat mengulang seluruh proyek hanya dengan mengubah CSS. Web telah menjadi lebih dari sekadar komoditas dibandingkan masa lalu, jadi kita harus tangkas. Jika Anda membuat burn down menggunakan Bootstrap dan melakukan segalanya dengannya, tidak apa-apa. Namun jangan menggunakan banyak perpustakaan, kerangka kerja, dan back-end sisi server yang menghasilkan aplikasi secara keseluruhan sambil mengklaim bahwa aplikasi tersebut dapat diskalakan. Mereka tidak akan melakukan hal itu secara ajaib: Anda masih perlu mengenal mereka dengan baik untuk memanfaatkan kekuatan mereka, daripada hanya menambahkan banyak kode untuk digunakan nanti suatu hal yang tidak akan pernah datang lagi.

Ringkasan

Tidak ada sesuatu pun di sini yang benar-benar baru, namun sepertinya kita selalu melupakan trik dan praktik yang pertama kali membuat Web sukses. Fleksibilitas mendefinisikan Web dan kami terus menciptakan ceruk untuk semua hal yang dapat dilakukan Web dan mempromosikan kualitas ini untuk menjadi tujuan utama setiap pengembang di luar sana. Ini bukan tentang membangun banyak klien atau aplikasi asli di Web, ini tentang menggunakan Web dan teknologinya untuk menyampaikan konten dan media, memungkinkan orang untuk dengan mudah menambahkan hal-hal baru ke rangkaian tabung yang menakjubkan ini.

Saya pikir ini saatnya berhenti mencari solusi yang mencoba menjinakkan Web dan menjadikannya lebih seperti lingkungan lain. Sebaliknya, kita harus menerima gagasan bahwa masa depan hanya akan menghasilkan taman bermain yang lebih beragam, dan kita tidak bisa memenuhi kebutuhan tersebut dengan bersikap membatasi. Jadi, daripada mengabstraksikan tugas berkomunikasi dengan browser ke dalam dunia perpustakaan yang tampak lebih pendek namun pada akhirnya lebih kompleks dan solusi sementara yang cepat, saya harap Anda menemukan beberapa ide di sini tentang cara menulis beberapa baris kode yang disesuaikan dengan apa yang sudah dimiliki browser.

BAB 4
BUDAYA PERTUNJUKAN

Orang tua saya adalah penggemar berat waktu keluarga. Mereka senang mengumpulkan saya dan keempat saudara saya untuk melakukan sesuatu. Kami melakukan banyak perjalanan keluarga. Setiap hari Minggu adalah malam menonton film keluarga. Mereka mengambil kesempatan apa pun agar keluarga melakukan sesuatu bersama. Contoh bagusnya adalah memilih pohon Natal. Sekarang, beberapa orang tua akan baik-baik saja jika mengambil pohon dari pohon terdekat. Tapi tidak dengan orang tuaku. Bagi mereka, ini adalah kesempatan lain untuk melakukan sesuatu sebagai sebuah keluarga untuk menciptakan kenangan keluarga.

Jadi setiap bulan Desember, kami berkumpul di dalam van dan berkendara ke bagian hutan yang tenang dan berhutan lebat yang penuh dengan kandidat utama untuk pohon Natal yang sempurna.Kami meluangkan waktu memilih pohon Natal bukanlah sesuatu yang terburu-buru. Kami berjalan berjam-jam di tengah salju tebal mencoba menemukan pohon yang tepat. Kami tidak pernah khawatir mengenai tindakan pencegahan untuk memastikan kami dapat menemukan jalan kembali ke van itu adalah sesuatu yang baru saja terjadi. Ayah saya hebat dalam menentukan arah dan sepertinya selalu tahu ke mana harus pergi.

Kami menemukan pohon yang sempurna setelah tiga atau empat jam. Setelah ayah saya menebangnya, kami mulai kembali ke arah yang kami pikir adalah arah van. Kami berjalan jauh, melihat banyak hutan yang tidak kami kenali, dan menyadari bahwa kami tersesat. Perasaan ayah saya yang tampaknya tidak bisa salah dalam menentukan arah telah mengecewakannya. Tentu saja hal ini membuat semua orang menjadi kesal. Ayah saya bangga dengan kemampuannya menemukan jalan kembali, tetapi kali ini dia gagal. Ibuku tidak begitu senang karena kelima anaknya tersesat di tengah hutan yang dipenuhi salju. Dan kami, anak-anak, bukanlah kelompok yang paling sabar.

Di kepalaku, aku sudah memutuskan bahwa kami pastinya tidak akan keluar dari hutan malam itu. Menganggap diri remaja saya adalah orang yang suka beraktivitas di luar ruangan, saya mulai membuat rencana darurat. Kita bisa membangun tempat berlindung kecil, menyalakan api dengan batu (mereka melakukannya di film seberapa sulitnya?) dan mencari makanan. Memang dingin, tapi kami akan mencari jalan keluarnya. Syukurlah bagi kami semua, kami tidak pernah menguji kemampuan bertahan hidup saya. Ketika malam begitu gelap sehingga kami hanya dapat melihat sekitar satu kaki di depan kami, kami akhirnya tersandung di jalan. Dari sana, ayah saya dengan mudah dapat memandu kami kembali ke mobil.

Tahun berikutnya, kami merencanakan lebih baik. Kami mengikatkan tali pada pepohonan saat kami berjalan, membawa kompas, dan sering berhenti untuk memeriksa arah. Tersesat bukanlah sesuatu yang ingin kami lakukan lagi. Daripada memandang kembali ke van sebagai sesuatu yang baru saja terjadi setelah segalanya selesai, kami akan lebih berhati-hati dalam memastikan perjalanan kami sukses.

4.1 TENTANG KINERJA

Perjuangan serupa juga merugikan situs web kita saat ini. Situs web menjadi semakin bertambah dengan kecepatan yang mengkhawatirkan. Dari bulan Maret 2012 hingga Maret 2013, rata-rata berat halaman melonjak sebesar 24% menurut data dari HTTP Archive. Kita menuju ke arah yang salah dengan cepat. Biasanya, kita bahkan tidak menyadarinya saat hal itu terjadi dan jika kita menyadarinya, sering kali sudah terlambat.

Saya sedang berbicara dengan seorang pengembang tentang situs web baru yang sedang dia kerjakan. Situs webnya tentu saja dibuat dengan baik. Tapi itu agak berat. Tidak banyak, tapi cukup sehingga dia merasa perlu menjelaskan sedikit situasinya. Apa yang dia katakan menyoroti masalah kinerja Web secara umum. “Saya ragu ada orang yang benar-benar ingin merilis situs yang kinerjanya tidak bagus,” jelasnya. “Ini hanyalah akibat dari tidak diberikan kemewahan waktu dan tekanan dari atas ke bawah.” Merasa seperti Anda pernah ke sana sebelumnya? Saya yakin sebagian besar dari kita dapat dengan mudah memahami hal ini. Saya tahu saya bisa.

Saya sedang mengerjakan sebuah proyek di mana saya berkolaborasi dengan tim pengembang internal. Sejak awal, semua orang menyatakan bahwa, antara lain, mereka ingin situs ini berjalan sangat cepat. Pengembang tempat saya bekerja sangat baik dalam pekerjaan mereka, dan mengingat tingginya tingkat kepentingan yang diberikan pada kinerja, ada alasan bagus untuk percaya bahwa hal ini akan berjalan dengan baik. Namun ternyata tidak.

Maket awal diperlihatkan kepada manajemen tingkat atas sebelum diperlihatkan kepada pengembang. Pada saat pengembang melihat maketnya, mereka telah disetujui oleh pihak yang berwenang. Tidak ada peluang untuk mencoba mengubahnya berdasarkan potensi risiko kinerja (dan ada beberapa).

Kemudian tenggat waktu internal ditetapkan berdasarkan kebutuhan bisnis yang secara signifikan mengubah agresivitas linimasa. Kombinasi dari rentang waktu yang ketat dan mock-up ambisius yang disetujui terlalu dini menimbulkan masalah lain. “Lakukan dengan cepat” dengan cepat berubah menjadi “Buatlah berhasil. Kami selalu bisa membuatnya lebih cepat nanti.” Tentu saja nanti tidak pernah datang. Jadi, setelah banyak kerja keras, kami hampir mencapai peluncurannya dan kinerja situs ini sangat buruk. Prosesnya lambat, lamban semua hal yang tidak diinginkan oleh kami semua.

Tapi ini adalah tim yang berkualitas, dan tim yang sangat bangga dengan pekerjaannya. Jadi untuk sisa pertandingan kandang, kami semua berusaha sekuat tenaga. Kami bekerja sangat larut malam. Kami bekerja selama akhir pekan. Segalanya menjadi stres bagi semua orang emosi dan air mata bercampur dengan dedikasi mutlak untuk memperbaiki situasi.

Saat situs ini diluncurkan, keadaannya lebih baik. Tidak hebat, tapi bukan lagi monster seperti dulu. Namun, dibutuhkan kerja keras yang luar biasa untuk mencapainya. Bahkan dengan adanya perbaikan, kami harus mengunjungi kembali laman utama situs tersebut beberapa bulan kemudian, meninggalkan semua pekerjaan yang telah kami lakukan dan memulai dari awal. Sama seperti ketika keluarga saya tersesat di hutan sebelum Natal, waktu yang lama, terlambat (dan akhirnya terbuang percuma) dalam proyek ini dapat dihindari seandainya kami lebih berhati-hati. Seandainya kinerja dimasukkan ke dalam proses

pembuatan situs, dan bukan sesuatu yang ditambahkan, hasilnya adalah situs yang lebih cepat dan tim yang jauh lebih bahagia.

Masalahnya bukan pada kompetensi pengembang. Seperti yang saya katakan, ini adalah tim yang sangat berbakat. Masalahnya juga bukan hanya masalah teknologi: banyak optimasi cerdas yang diterapkan untuk menurunkan beban. Sebaliknya, masalahnya adalah kurangnya budaya kinerja yang mapan komitmen total terhadap kinerja seluruh tim yang akan mendorong dan mempengaruhi keputusan selama siklus proyek. Sederhananya, kami tidak sengaja memastikan situs kami berkinerja baik. Itu diperlakukan sebagai sesuatu yang akan terjadi pada akhirnya. Tidak ada ikatan apa pun, tidak ada pengecekan arah untuk memastikan kami tidak menyimpang dari jalur. Akibatnya, ketika segala sesuatunya menjadi sulit, kinerja adalah salah satu hal pertama yang diabaikan. Mengingat betapa pentingnya peran kinerja dalam pengalaman pengguna, hal ini merupakan kesalahan besar yang harus dilakukan.

Dampak Kinerja

Sebutkan sesuatu yang menjadi perhatian bisnis Anda dan saya berani bertaruh bahwa kinerjanya telah menjadi faktor penting. Penelitian demi penelitian menunjukkan bahwa kinerja situs Anda berdampak langsung pada cara pengguna berinteraksi dengannya.

- Amazon menemukan bahwa untuk setiap 100 md peningkatan waktu buka halaman, terjadi peningkatan pendapatan sebesar 1%.
- Bing mencoba eksperimen yang sengaja membuat kueri penelusuran memakan waktu dua detik lebih lama. Hasilnya adalah penurunan pendapatan per pengguna sebesar 4,3%.
- Mozilla meningkatkan kinerja halaman arahannya sebesar 2,2 detik dan mengalami peningkatan konversi unduhan sebesar 15,4%. Ini secara kasar berarti 60 juta unduhan lebih banyak setiap tahunnya.
- Shopzilla memangkas waktu buka halaman dari 6 detik menjadi 1,2 detik. Selain peningkatan pendapatan sebesar 12%, lalu lintas halaman meningkat sebesar 25%.

Kita bisa melanjutkannya, tapi menurut saya gambarannya mulai menjadi cukup jelas: peningkatan kinerja mempengaruhi tampilan halaman, lalu lintas — dan laba. Kinerja adalah tentang menghormati pengunjung Anda, dan mereka akan memperhatikan jika Anda tidak melakukannya.

- 39% pengguna mengatakan bahwa kinerja lebih penting bagi mereka dibandingkan fungsionalitas situs.
- 57% pengguna akan meninggalkan situs setelah menunggu tiga detik hingga halaman dimuat.

Kinerja tidak hanya memengaruhi pengguna yang sudah ada, namun sebenarnya dapat membantu Anda menjangkau audiens baru juga. Pertimbangkan pengalaman YouTube ketika meningkatkan kinerja (lihat bab Mat Marquis). Pasar baru, peningkatan pendapatan, peningkatan metrik bisnis, kepuasan pengguna yang lebih baik dampak dari peningkatan kinerja Web bukanlah hal yang kecil.

Semua ini seharusnya tidak mengejutkan. Web adalah media interaktif. Klik tombol, gulir halaman ke bawah, kirimkan formulir: interaksi adalah inti dari penggunaan situs web.

Kinerja adalah komponen mendasar dari pengalaman pengguna. Kita dapat memperbaiki jalannya, dan kita harus melakukannya jika kita ingin benar-benar memanfaatkan sifat Web yang ada di mana-mana dan interaktif. Untuk melakukan hal ini, kita harus berhenti menganggap kinerja sebagai hal yang menyenangkan, atau tugas pengembang, dan mulai memasukkannya ke dalam alur kerja kita.

Mendapatkan Dukungan

Tidak peduli betapa berharganya kinerja yang Anda ketahui, Anda perlu mendapatkan dukungan. Jika orang yang berhubungan langsung dengan anggaran dan jadwal tidak peduli, Anda akan kesulitan memastikan kinerja diprioritaskan selama proses berlangsung.

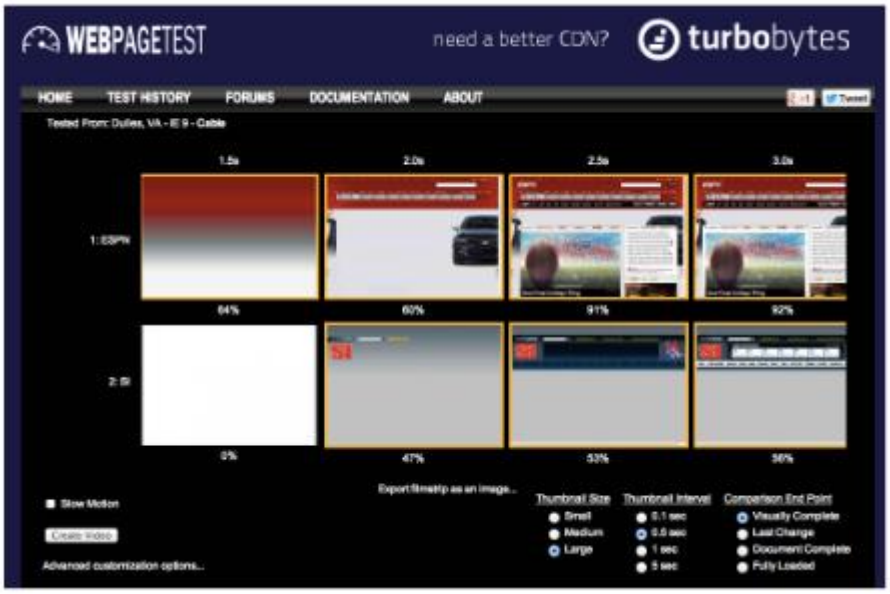
Jadikan Pribadi

Sangat mudah untuk merasa bersemangat dalam mengurangi metrik seperti waktu muat dan bobot halaman, namun hal tersebut mungkin bukan hal yang penting bagi orang-orang yang membutuhkan dukungan dari Anda. Mereka ingin mendengar manfaatnya bagi hal-hal yang mereka pedulikan. Beberapa orang ingin melihat pengaruhnya terhadap keuntungan. Orang lain mungkin lebih peduli tentang pengaruhnya terhadap tampilan halaman dan rasio pentalan. Pelajari apa yang menjadi perhatian orang lain dan fokuslah untuk menekankan bagaimana kinerja meningkatkan faktor-faktor tersebut. Anda akan lebih berhasil meyakinkan mereka tentang pentingnya kinerja jika Anda menghubungkannya dengan sesuatu yang langsung berarti bagi mereka.

BUAT VISUAL

Jika Anda mencoba meyakinkan klien atau atasan tentang pentingnya kinerja, tambahkan metrik dengan visual. Salah satu metode yang sangat efektif adalah dengan menunjukkan kepada mereka bahwa situs mereka dimuat di samping situs pesaing yang lebih cepat. Jika klien Anda adalah McDonald's dan situs Burger King memuat lebih cepat, pertama-tama tunjukkan kepada mereka apa yang dapat mereka peroleh dengan meningkatkan kinerja, dan kemudian tunjukkan kepada mereka bagaimana Burger King mengalahkan mereka. Tidak ada seorang pun yang suka kalah dari pesaing.

Anda dapat dengan mudah melakukan ini dengan WebPageTest.org menggunakan salah satu dari dua metode. Yang pertama adalah menjalankan pengujian untuk setiap situs satu per satu, memilih “Capture Video” di bawah pengaturan lanjutan. Simpan kedua video dan antrikan secara berdampingan.



Gambar 4.1 Opsi perbandingan visual di WebPageTest.org memungkinkan Anda melihat dengan mudah seberapa cepat situs dimuat jika dibandingkan satu sama lain.

WebPageTest juga menyediakan opsi "Perbandingan Visual". Dengan metode ini, Anda menambahkan URL untuk setiap halaman untuk dibandingkan dan kemudian menjalankan pengujian. WebPageTest akan menangkap tangkapan layar gambar statis selama proses pemuatan sehingga Anda dapat melihat dengan tepat kapan segala sesuatu mulai dimuat di setiap halaman, dan kapan halaman selesai. Kadang-kadang ini bahkan bisa menjadi sedikit lebih dramatis daripada sebuah video.

Memasukkan Kinerja ke dalam Proses

Ketika saya berusia awal dua puluhan, saya bekerja di RadioShack. Ini bukanlah pertunjukan yang bagus bagi saya. Tentu saja, RadioShack punya banyak gadget untuk dimainkan bagian itu menyenangkan. Namun saya tidak pernah menjadi tenaga penjualan yang hebat, dan peran inilah yang saya perlukan. Saya bekerja dengan seseorang yang ahli dalam hal itu. Bahkan dibuat untuk itu. Dia benar-benar akan mendorong saya ke samping kadang-kadang untuk menemui seseorang yang sedang mencari ponsel cara tercepat untuk mendapatkan beberapa dolar tambahan dari cek kami.

Anda tahu, kami mendapat upah minimum, namun satu-satunya cara Anda mendapatkan uang adalah melalui komisi dan apa yang mereka sebut SPIF (dana insentif kinerja khusus). SPIF adalah insentif yang Anda terima karena menjual barang tertentu. Jual hp, dapat SPIF. Anda juga mendapatkan SPIF untuk menjual paket layanan. Menurut pendapat saya, paket layanan kami sering kali bernilai baik khususnya untuk item tertentu.

Paket layanan juga merupakan satu-satunya hal yang dapat saya jual dan jual dengan baik. Saya secara konsisten berada di peringkat lima atau enam teratas di distrik tersebut untuk penjualan rencana layanan. Alasannya sederhana. Saya pikir ini penting, dan sebagai hasilnya, saya menjadikannya bagian dari proses. Kebanyakan orang tidak menyebutkan paket layanannya. Mereka akan fokus pada telepon itu sendiri – fitur apa yang dimilikinya dan, yang

lebih penting, apa yang tidak dimilikinya yang akan membuat Anda membeli model berikutnya. Mereka akan mendatangi kasir dan setelah memindai barang tersebut, sistem tempat penjualan akan memunculkan peringatan kecil yang menanyakan apakah pelanggan menginginkan paket layanan tersebut.

Mereka akan melihat dari layar dan bertanya, "Apakah Anda ingin membeli paket layanan untuk itu? Biayanya Rp. 80.000 untuk dua tahun dan mencakup masalah apa pun." Pelanggan tidak mau membeli. Mengapa mereka harus melakukannya? Rencana ini tidak disebutkan sama sekali selama percakapan penjualan dan sebagian besar pelanggan menyadari bahwa tenaga penjualan tersebut bertanya hanya karena perintah dari kasir. Bagi pelanggan, hal ini jelas bukan hal penting yang perlu dikhawatirkan. Dengan tidak mendiskusikan rencana tersebut secara menyeluruh, tenaga penjualan meremehkan pentingnya hal tersebut.

Proses saya berbeda. Seorang pelanggan akan masuk dan meminta telepon nirkabel. Saya akan berkata, "Tentu, kami memiliki banyak telepon nirkabel dan kami juga menawarkan paket layanan yang sangat baik untuk melindunginya. Mari kita lihat dan lihat apakah kami dapat menemukan satu yang cocok untuk Anda." Sambil menunjukkan kepada mereka ponsel dan mendiskusikan fitur-fiturnya, saya kembali menyebutkan rencananya. "Ponsel itu sendiri seharusnya bisa bertahan cukup lama, tapi baterainya biasanya akan habis setelah sekitar satu setengah tahun. Paket layanannya sudah termasuk baterai gratis setiap tahunnya, jadi Anda akan aman di sana."

Di konter, saat saya memindai telepon yang mereka pilih dan sebelum sistem sempat memberi tahu saya saya akan bertanya apakah mereka ingin mendapatkan paket layanan. Ini berhasil untuk 85–90% orang yang membeli ponsel dari saya. Tidak ada trik sulap yang terlibat, tidak ada dalih, tidak ada upaya untuk menipu siapa pun. Saya yakin rencana layanan tersebut adalah ide yang bagus dan menurut saya menawarkan nilai. Saya memastikan bahwa pelanggan mengetahui perasaan saya, sebelum mereka membeli barang tersebut. Hasilnya adalah mereka memahami hal ini. Mereka percaya bahwa saya menyarankan rencana tersebut bukan karena suatu sistem yang mendorong saya atau karena saya diberitahu untuk melakukan hal ini oleh orang lain – saya melakukannya karena saya merasa hal ini penting.

Ketika kita meninggalkan diskusi mengenai kinerja sampai akhir pembicaraan, ketika kita menyebutkannya secara sepintas, kita meremehkan pentingnya hal tersebut bagi proyek. Dengan tidak mengungkitnya selama proses berlangsung, kami mengatakan bahwa menurut kami hal ini tidak cukup penting untuk dibahas lebih lanjut. Kami mengatakan itu adalah sesuatu yang tidak terlalu bernilai. Jika kita ingin mulai memperbaiki jalannya kinerja di Web, kita harus menjadikan kinerja sebagai bagian dari diskusi sejak awal proses, dan kita harus konkret dalam hal ini. Salah satu cara terbaik untuk melakukannya adalah dengan menetapkan anggaran kinerja.

Menetapkan Anggaran Kinerja

Brad Frost menulis postingan blog tentang pentingnya mendiskusikan kinerja sejak awal siklus hidup suatu proyek9 dan dia menyarankan untuk menyebutkan kinerja dalam dokumen proyek: Pernyataan kerja, proposal proyek, dan ringkasan desain harus secara

eksplisit dan berulang kali menyebutkan kinerja sebagai tujuan utama. “Tujuan dari proyek ini adalah untuk menciptakan pengalaman yang menakjubkan, fleksibel, dan secepat kilat…”

Tentu saja dia benar

Namun sering kali, ungkapan seperti “secepat kilat” terbukti tidak cukup konkrit. Saya pernah mendengar banyak orang di tahap awal sebuah proyek mengatakan bahwa mereka ingin situs mereka cepat, hanya untuk melihatnya berubah menjadi salah satu hal yang pada akhirnya akan menyenangkan untuk diperbaiki. Satu hal yang menurut saya berhasil dengan baik adalah menetapkan anggaran kinerja. Anggaran kinerja persis seperti yang terlihat. Anda menetapkan anggaran untuk halaman Anda dan tidak mengizinkan halaman melebihi jumlah tersebut.

Merupakan ide bagus untuk memulai dengan waktu buka, namun anggaran yang Anda tetapkan dan rujuk akan lebih berpengaruh jika Anda dapat menentukan bobot halaman sebenarnya atau jumlah permintaan. Mereferensikan bobot halaman atau jumlah permintaan tertentu, bukan hanya waktu buka tertentu, membuat percakapan lebih mudah. Misalnya, jika anggaran Anda menyatakan bahwa situs harus dimuat dalam waktu kurang dari lima detik di jaringan 3G dan Anda mencoba memutuskan apakah akan menambahkan carousel ke laman atau tidak, Anda harus terlebih dahulu menerjemahkan lima detik tersebut ke dalam bobot atau jumlah permintaan untuk dapat membuat penentuan itu di muka. Jumlah permintaan dan bobot halaman juga merupakan hal yang relatif mudah untuk diperkuat dalam proses pembangunan Anda, sehingga Anda dapat menerapkan anggaran secara ketat jika Anda menginginkannya.

Tiba di Anggaran

Mengetahui bahwa kinerja memengaruhi hampir semua metrik bisnis yang penting, skenario idealnya adalah membuat situs Anda secepat mungkin. Target waktu respons yang paling terkenal telah ada sejak tahun 1968, dan dipopulerkan oleh Jakob Neilsen pada tahun 199310:

a. 0,1 detik: Batas bagi pengguna untuk merasakan bahwa sistem bereaksi seketika.
b. 1,0 detik: Batas aliran pikiran yang tidak terputus. Pengguna menyadari adanya penundaan, namun mereka tetap merasakan interaksi langsung.
c. 10 detik: Batas untuk menjaga perhatian pengguna. Jika lebih lama dari ini, pengguna akan menyerah atau melakukan hal lain sambil menunggu.

Idealnya, situs Anda mendobrak batasan kedua kalinya. Namun, terkadang hal tersebut tidak realistis baik karena jenis situs yang Anda bangun, anggaran, atau kendala eksternal lainnya. Berbekal pengetahuan tentang pentingnya kinerja, ada dua kriteria tambahan yang perlu dipertimbangkan ketika mencapai anggaran ideal Anda:

1. **Kinerja situs Anda saat ini**

   Pertama, audit situs Anda yang ada untuk melihat kinerjanya saat ini dalam kondisi jaringan yang berbeda. Catat waktu muat untuk tolok ukur ini, serta jumlah permintaan HTTP dan bobot halaman keseluruhan.

2. **Kinerja situs pesaing Anda saat ini**

Selanjutnya, lihat kinerja situs lain di industri Anda. Untuk pesaing yang sangat penting, lakukan analisis yang sama seperti yang Anda lakukan untuk situs Anda sendiri. Anda juga bisa mendapatkan gambaran umum yang bagus tentang kinerja situs dalam industri Anda dengan mengacu pada indeks kinerja Keynote yang mengelompokkan analisis berdasarkan jenis industri.

Sekarang setelah Anda mengetahui bagaimana kinerja situs Anda, serta bagaimana kinerja pesaing Anda, Anda dapat membuat keputusan yang tepat tentang berapa anggaran yang harus diselesaikan dengan menggunakan hukum Weber dan aturan 20%.

Aturan 20%.

Dokter Jerman E.H. Weber mengamati bahwa perbedaan mencolok antara dua properti bervariasi sesuai dengan ukuran propertinya. Misalnya, lebih mudah membedakan satu jam dan dua jam daripada membedakan satu menit dan dua menit. Menerapkan ini pada interaksi komputer, Steven C. Seow mengemukakan aturan 20% dalam bukunya, Designing and Engineering Time: The Psychology of Time Perception. Sederhananya, untuk menciptakan peningkatan kinerja yang nyata seperti yang dirasakan pengunjung, Anda perlu meningkatkan kinerja setidaknya sebesar 20%.

Jika Anda menerapkan aturan ini pada metrik yang Anda temukan untuk situs Anda saat ini, Anda hanya dapat memperoleh peningkatan minimal. Jika halaman Anda dimuat dalam 10 detik, memuatnya dalam 8 detik atau kurang akan memberikan peningkatan yang nyata; 9 detik, apalagi. Menerapkan aturan 20% pada metrik dari pesaing Anda dapat membantu Anda menentukan pada titik mana Anda akan memberikan peningkatan kinerja yang cukup besar untuk benar-benar membedakan situs Anda dengan lebih cepat. Misalnya, jika situs pesaing Anda dimuat dalam 5 detik, Anda ingin situs Anda paling tidak dalam waktu kurang dari 4 detik.

Tujuan utamanya adalah membuat pengalaman pengunjung Anda berlangsung secepat mungkin, namun aturan 20% dapat memberikan titik awal yang baik.

4.2 DAMPAK ANGGARAN KINERJA

Tanpa adanya anggaran, Anda tidak akan mempunyai kerangka kerja untuk diskusi ini. Karena tidak ada basis kinerja yang ditetapkan, sulit untuk menyatakan bahwa penggeser konten menambah terlalu banyak bobot pada halaman. Jika Anda sudah menetapkan garis dasar ini, diskusi akan disederhanakan. Clearleft, sebuah agen desain Web di Brighton, Inggris, menulis tentang pengalaman mereka menggunakan anggaran kinerja dan sampai pada kesimpulan yang sama:

Hal yang penting adalah melihat setiap keputusan, mulai dari proses desain/pembangunan, sebagai sesuatu yang mempunyai konsekuensi. Memiliki ‘anggaran’ yang telah ditentukan sebelumnya merupakan cara yang jelas dan nyata untuk menyusun keputusan mengenai apa saja yang boleh dan tidak boleh dimasukkan, dan pada tahap awal proyek yang sesuai. Hal ini juga berpotensi memberikan beberapa pembenaran kepada klien tentang mengapa hal-hal tertentu dihilangkan (atau lebih tepatnya, ditukar dengan hal lain). Penetapan anggaran memberikan pembenaran dan kerangka diskusi, dan hal ini terus

berlanjut sepanjang siklus hidup situs. Terus terapkan anggaran setelah peluncuran sebagai cara untuk menghindari pembengkakan perlahan yang cenderung muncul dengan sendirinya.

Jadilah realistik

Inti dari anggaran kinerja adalah untuk memberikan titik perbandingan yang sangat nyata, jadi jelaskan secara eksplisit. Sesuatu seperti “ukuran maksimum 500KB dan tidak lebih dari 15 permintaan HTTP” adalah tujuan Anda. Dan bersikaplah realistis tentang hal itu. Menetapkan anggaran yang terlalu tinggi (“Tidak lebih dari 5MB!”) atau terlalu rendah (“Tidak lebih dari 10KB!”) sama sekali tidak ada gunanya bagi Anda. Bersikaplah tegas, tetapi pahami kenyataan.

Perlu diperhatikan juga bahwa layanan pihak ketiga seperti iklan, meskipun penting bagi bisnis, dapat menghancurkan anggaran kinerja sendirian. Untuk skenario tersebut, masuk akal untuk mengkategorikan aset pada halaman. Pengembang di surat kabar Guardian membuat tiga kategori:

- Isi
- Peningkatan
- Sisa makanan

Konten adalah daging dan kentang: itulah alasan pengguna mengunjungi situs Anda. Peningkatannya adalah pada balutan: JavaScript dan gaya yang membuat pengalaman lebih menyenangkan. Dan sisa makanannya, itu adalah sisa-sisa yang Anda berikan kepada anjing di bawah meja, hal-hal yang tidak dipedulikan oleh pengguna. Itulah hal-hal yang harus dilakukan terakhir dalam proses pemuatan; berikan apa yang diinginkan pengguna terlebih dahulu, lalu muat kelebihannya setelahnya.

Hingga jaringan iklan dan layanan lainnya mengikuti program ini dan mulai membuat layanan yang lebih cepat, anggaran Anda mungkin harus diterapkan hanya untuk konten dan penyempurnaan saja. Tentu saja itu tidak berarti Anda tidak bisa membatasi penurunan kinerja pihak ketiga tersebut. Artinya, terkadang kami harus mengakui bahwa bagian tertentu dari halaman tersebut tetap berada di luar kendali kami.

Beberapa Kata Perhatian

Anggaran kinerja adalah cara terbaik untuk memastikan kinerja tetap menjadi bagian dari diskusi. Namun seperti yang kita lihat sebelumnya, anggaran harus ditetapkan sejak dini. Jika Anda menyelesaikan setengah proyek sebelum menetapkan anggaran, Anda akan kesulitan meyakinkan siapa pun bahwa hal itu cukup penting untuk diperhatikan. Belum lagi pada saat itu, mungkin sudah ada visual atau fitur yang disetujui yang langsung menghabiskan anggaran berapa pun yang perlu Anda tetapkan.

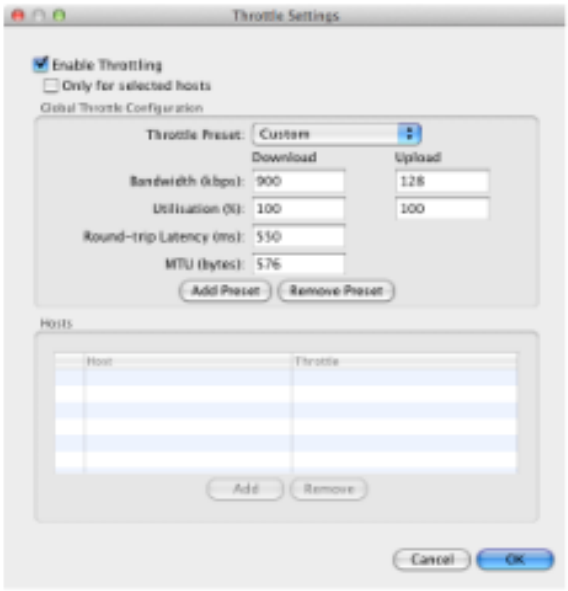
Hal penting lainnya yang perlu diperhatikan: saat Anda menetapkan anggaran, dan saat Anda menerapkannya, Anda harus sudah mengetahui konten apa yang perlu ada di halaman tersebut. Anggaran kinerja dimaksudkan untuk membantu Anda memutuskan bagaimana menampilkan konten Anda, bukan konten apa yang akan ditampilkan. Menghapus konten penting dari suatu halaman bukanlah strategi kinerja.

Rangkulah Sakitnya

Mereka mengatakan Anda tidak dapat memahami seseorang sampai Anda berjalan satu mil dengan posisi mereka. Sampai Anda mengalami apa yang mereka alami suka dan duka mungkin sulit untuk benar-benar berempati. Hal ini terutama terjadi pada kinerja. Baru-baru ini, sebuah perusahaan besar meluncurkan situs yang didesain ulang. Itu cantik. Citra yang indah, interaksi yang apik, tipografi yang luar biasa; tidak diragukan lagi, ini adalah situs yang indah. Namun, seperti banyak situs web lainnya, situs ini menyembunyikan sebuah rahasia. Di bawah hasil akhir yang indah dan mengilap itu ada situs yang sangat besar.

Mengaktifkan situs melalui koneksi kabel yang cepat, tampaknya baik-baik saja. Itu dimuat cukup cepat, tentu saja tidak cukup lambat untuk menimbulkan kekhawatiran. Namun situs yang sama, yang dimuat di jaringan 3G tiba-tiba membutuhkan waktu lebih dari 90 detik untuk dimuat. Satu setengah menit menatap layar putih dan bilah pemuatan. Tidak mungkin situs ini bisa aktif jika orang yang merancang dan membangunnya mengalami hal ini selama pengujian.

Selain memasukkan kode nyata sesegera mungkin, kita perlu mengatasi kesulitan jaringan yang lambat selama pengujian. Ada sejumlah alat luar biasa yang tersedia untuk membantu mensimulasikan jaringan yang berbeda dengan batasan latensi dan bandwidth yang berbeda. Jika Anda menggunakan Mac yang menjalankan OS X10.7 atau lebih baru, Anda dapat menggunakan Network Conditioner. Ada juga Slowy, aplikasi murah yang ada di bilah tugas Anda, membuatnya mudah untuk beralih di antara simulasi jaringan yang berbeda.



Gambar 4.2 Alat seperti Charles menawarkan kemampuan untuk membatasi koneksi Anda untuk melihat bagaimana rasanya menavigasi situs Anda melalui jaringan yang buruk.

Anda juga dapat memilih alat proXy debugging Web yang lengkap, seperti Charles atau Fiddler. Keduanya tersedia untuk Mac OS X dan Windows dan menawarkan berbagai fitur tambahan di luar simulasi jaringan.

Sering Sakit

Saya menyebutkan simulasi jaringan kepada seorang insinyur di sebuah perusahaan yang sangat memperhatikan kinerja, dan dia bercanda bahwa dia mungkin akan kembali dan memaksa timnya untuk menulis kode untuk minggu depan karena koneksi yang buruk.

Seminggu penuh mungkin sedikit berlebihan, namun pertimbangkan untuk menjadikan simulasi jaringan sebagai bagian dari proses mingguan, atau bahkan harian Anda. Selama beberapa jam sehari dalam seminggu, mintalah semua orang di tim Anda menggunakan koneksi jaringan simulasi. Tidak akan memakan waktu lama sebelum semua orang menyadari adanya hambatan kinerja dan mulai berupaya memperbaikinya.

Sadarlah

Saya tidak akan menulis permohonan panjang lebar tentang betapa cacatnya mendesain situs web dengan perangkat lunak seperti Photoshop. Saya pikir semakin banyak dibahas, semakin kita menyadari bahwa, seperti halnya alat apa pun, Photoshop memiliki fungsi yang baik dan fungsi yang buruk. Kami tidak akan pernah sepenuhnya menghilangkan perangkat lunak manipulasi gambar dalam proses desain dan, sejujurnya, hal itu bukanlah tujuannya. Namun penting untuk mempertimbangkan kekurangan apa yang dimilikinya sehingga proses kami dapat mempertimbangkannya.

Salah satu kekhawatiran yang sahih dengan menghabiskan terlalu banyak waktu di Photoshop dibandingkan browser adalah Anda melihat gambar situs web dalam situasi ideal dan pada ukuran tertentu yang terkontrol. Hal ini sering disebut sebagai masalah desain responsif, namun juga merupakan masalah kinerja. Seperti yang dikatakan Brad Frost, "*Anda tidak dapat meniru kinerja di Photoshop.*"

Menguasai browser sejak dini dapat membantu Anda mengetahui potensi peningkatan kinerja sebelum gangguan tersebut benar-benar lepas kendali. Maket yang setiap elemennya memiliki bayangan semi-transparan mungkin terlihat indah, tetapi jalankan di perangkat seluler dan Anda mungkin menyadari bahwa menggulir adalah tugas yang sulit. Mengetahui hal itu sejak dini memungkinkan Anda mempertimbangkan solusi lain.

Untuk lebih jelasnya, solusi-solusi lain tersebut tidak harus tanpa hiasan-hiasan semacam itu sama sekali. Situs yang berkinerja baik tidak harus terlihat polos atau membosankan secara visual. Ada serangkaian trade-off yang harus dilakukan dengan mempertimbangkan manfaat dan biayanya. Performa dan estetika visual keduanya penting situs Anda perlu menyeimbangkan keduanya. Menjadi nyata kode nyata pada perangkat nyata sedini mungkin akan membantu Anda menjaga keseimbangan tersebut.

Salah satu cara terbaik untuk memungkinkan Anda masuk ke browser lebih awal adalah dengan memikirkan situs Anda dalam kaitannya dengan komponen yang dapat digunakan kembali. Punya pertanyaan tentang bagaimana kinerja hiasan navigasi mewah itu di perangkat yang berbeda? Masuk ke browser, kembangkan komponen itu dan lakukan uji coba.

Ada beberapa cara untuk melakukan ini. Favorit saya saat ini adalah pendekatan Atomic Web Design dari Brad Frost, yang memecah komponen situs web menjadi bentuk terkecil yang memungkinkan Anda membuat, misalnya, catatan kaki, tanpa harus membuat

halaman selanjutnya juga. Hal ini memungkinkan Anda dengan cepat melihat bagaimana bagian yang berbeda dapat bekerja pada resolusi yang berbeda dan dengan metode masukan yang berbeda. Alat spesifik yang Anda gunakan kurang penting dibandingkan hasil akhirnya: kemampuan untuk menguji sedikit demi sedikit dengan cepat untuk mengetahui masalah kinerja sebelum Anda terlalu jauh untuk kembali lagi.

Buatnya

Cara lain untuk mendorong pemikiran tentang pengoptimalan kinerja di seluruh proyek adalah dengan membuat metrik kinerja utama situs (seperti waktu buka) terlihat pada setiap pemuatan halaman. Etsy, yang sudah lama dikenal karena dedikasinya yang luar biasa terhadap kinerja, menggunakan pendekatan ini, sebuah ide yang awalnya dibahas oleh Jeff Atwood.

Etsy menampilkan waktu render (dihitung oleh server) pada halaman, tetapi dengan diperkenalkannya API waktu navigasi, Anda dapat dengan mudah memasukkan waktu buka halaman dengan beberapa baris JavaScript sederhana. Cuplikan di bawah ini akan menampilkan waktu muat yang dirasakan ke konsol, namun Anda dapat dengan mudah memodifikasinya menjadi output ke elemen dalam dokumen Anda agar mudah terlihat.

```
function getLoadTime() {
          var now = new Date().getTime();

          // Get the performance object
          window.performance     =     window.performance     ||
window.mozPerformance
|| window.msPerformance || window.webkitPerformance || {};
          var timing = performance.timing || {};
          if (timing) {
               var load_time = now - timing.navigationStart;
               console.log('Load time: ' + load_time + 'ms');
          }
}
window.onload = function() {
     getLoadTime();
}
```

Ini mungkin tampak seperti hal kecil, namun menampilkan waktu buka untuk setiap halaman di sudut kanan atas dapat berdampak besar pada cara tim Anda memandang pengoptimalan kinerja. Waktu muat menjadi titik perbandingan dan percakapan dan jika halaman dimuat dengan lambat, fakta tersebut akan langsung terlihat setiap kali mereka mengerjakannya.

Dasar Yang Cahaya

"Dipersiapkan." Itulah semboyan Pramuka. Ketika saya tumbuh dewasa, saya mengingat hal itu. Setiap kali saya melakukan perjalanan, saya akan mengemas semua yang

saya bisa. Status default saya adalah memastikan saya membawa semuanya, kalau-kalau saya membutuhkannya.

Kami sering mengambil pendekatan serupa secara online. Bahkan sebelum sesuatu dibuat, banyak proyek yang menyertakan kerangka JavaScript, untuk berjaga-jaga. Sebelum analisis apa pun dilakukan untuk menentukan apakah analisis diperlukan, kerangka kerja CSS mungkin dimuat. Pelat ketel, yang dimaksudkan sebagai titik awal untuk memangkas, dibiarkan apa adanya. Korsel gambar ditambahkan secara acak sebagai cara untuk memasukkan lebih banyak konten ke dalam halaman. Proyek-proyek ini seperti pisau saku semuanya ada di sana, kalau-kalau kita membutuhkannya.

Yang terjadi justru sebaliknya: segala sesuatu yang ditambahkan ke halaman setiap skrip, setiap gambar, setiap baris kode harus memiliki justifikasi. Itu perlu memiliki tujuan. Kami mengetahui hal ini dari perspektif desain visual; desainer mana pun akan memberi tahu Anda mengapa keputusan dibuat untuk menyertakan gambar ini atau ikon itu. Kinerja juga harus menginformasikan diskusi itu. Sebuah gambar tidak hanya harus memiliki tujuan, tetapi nilainya juga harus melebihi biayanya. Tidak ada makan siang gratis dan sebagainya.

Mengenai kerangka kerja dan boilerplate: tidak ada yang salah dengan keduanya. Ini adalah alat yang sangat berharga bila diterapkan dengan hati-hati dan digunakan dengan tepat. Namun alat-alat tersebut bukannya tanpa kesalahan dan ada kekhawatiran jika alat-alat tersebut menjadi dasar kita memulai, dan bukan alat yang kita tambahkan secara hati-hati bila diperlukan. Jika kita ingin membalikkan tren situs web yang semakin membengkak, kita perlu memulai dengan standar yang lebih baik dan berhati-hati dalam segala hal yang kita masukkan ke dalam situs kita.

Telah ditunjukkan berkali-kali bahwa orang suka tetap menggunakan opsi default. Salah satu penelitian yang sering dikutip berkaitan dengan pilihan default untuk donor organ (PDF). Di Amerika Serikat, yang pilihan utamanya adalah tidak menjadi donor organ, 28% masyarakatnya menyetujui menjadi donor organ. Bandingkan dengan Belgia, yang pilihan utamanya adalah menyetujui menjadi donor organ. Di sana, 98% orangnya adalah donor organ. Jared Spool melakukan eksperimen untuk melihat berapa banyak orang yang mengubah 150+ pengaturan (pada saat itu) yang tersedia bagi mereka di Microsoft Word.

Apa yang kami temukan sungguh menarik. Kurang dari 5% pengguna yang kami survei telah mengubah pengaturan apa pun. Lebih dari 95% telah menyimpan pengaturan dalam konfigurasi yang sama persis dengan program yang diinstal. Mungkin yang lebih menarik adalah alasan mereka tidak mengubah pengaturan ini. Salah satu pengaturan yang dinonaktifkan secara default adalah fungsi penyimpanan otomatis Word. Jadi dalam pengaturan default, opsi untuk menyimpan dokumen saat orang sedang mengerjakannya dinonaktifkan. Ketika orang ditanya mengapa mereka tidak mengubah pengaturan, mereka mengungkapkan bahwa mereka berasumsi ada alasan mengapa pengaturan tersebut tidak aktif.

Tentu saja, ini berarti 95% pengguna menjalankan penyimpanan otomatis dimatikan. Ketika kami mewawancarai sampel mereka, mereka semua mengatakan hal yang sama kepada kami. Mereka berasumsi Microsoft telah mematikannya karena suatu alasan, oleh

karena itu siapakah mereka yang mengaturnya sebaliknya. “Microsoft harus tahu apa yang mereka lakukan,” kata beberapa peserta kepada kami.

Pemikiran tentang default ini juga mempengaruhi cara kami menggunakan alat pengembangan kami. Sebagian besar bobot alat ini adalah hasil dari penyelesaian masalah yang sangat spesifik yang mungkin Anda hadapi atau tidak. Seperti yang ditemukan oleh Spool dan banyak orang sebelumnya, ketika opsi tersebut dimasukkan secara default, kecil kemungkinan sebagian besar pengembang akan meluangkan waktu untuk membenarkan keberadaan opsi berdasarkan kebutuhan proyek bahkan ketika alat penyesuaian tersedia. Sebaliknya, kami memuat bagasi tambahan ini ke situs kami secara default. Untuk berjaga-jaga.

Korsel responsif Grup Filament adalah salah satu contoh kecil alat yang mengambil pendekatan sebaliknya. Modular, dengan default cerdas. Mereka memiliki skrip default yang menyelesaikan pekerjaan, dan kemudian skrip terpisah yang memperluas perilaku carousel dengan cara berbeda. Opsi default Anda adalah hanya menyertakan apa yang benar-benar diperlukan anda menentukan apa lagi, jika ada, yang perlu disertakan. Christian Heilmann membahas hal ini dengan sangat baik di bab sebelumnya, “The Vanilla Web Diet.”

Gambar 4.3 ImageOptim dan ImageAlpha dapat bekerja sama untuk memperkecil ukuran gambar Anda.

Kita perlu menerapkan pemikiran ini pada cara kita mendekati proyek secara umum. Buatlah templat dasar Anda seramping dan seseram mungkin. Miliki pilihan untuk perilaku atau perbaikan tertentu hal-hal yang Anda tahu telah teruji dengan baik dan ringan. Namun

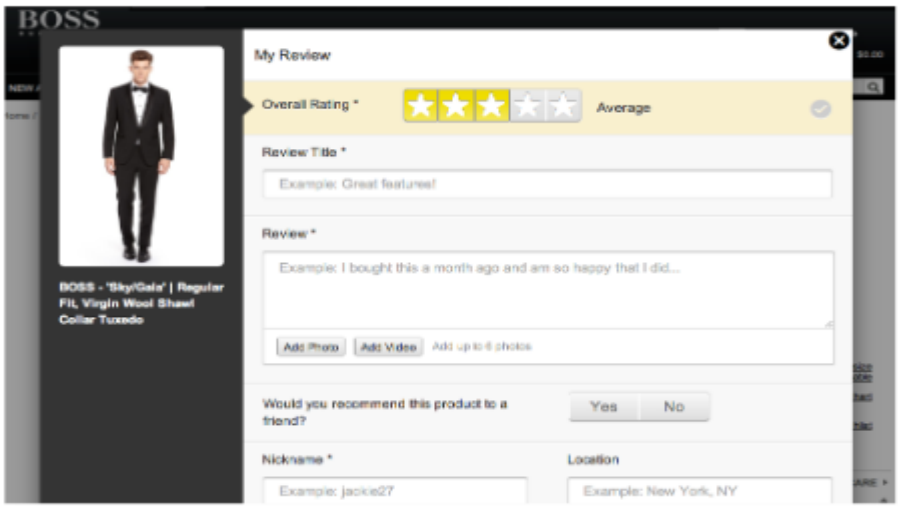
jangan menjadikannya bagian dari default Anda. Gulung hanya sesuai kebutuhan, jika diperlukan.

Tersangka Biasa dan Tidak Biasa

Untuk item apa pun yang disertakan di halaman Anda, lakukan apa pun yang Anda bisa untuk meminimalkan dampaknya terhadap bobot halaman dan waktu buka. Salah satu penyebab terbesar adalah gambar yang tidak dikompresi. Untungnya, ini juga salah satu yang paling mudah untuk diperbaiki. Pengoptimalan gambar dapat diotomatiskan ke dalam proses pembuatan, menggunakan alat canggih seperti ImageOptim Command Line Interface. Jika Anda tidak memiliki proses pembuatan yang formal, atau hanya dengan melihat jendela terminal saja sudah membuat Anda merinding, Anda dapat dengan mudah menggunakan alat berbasis GUI seperti ImageOptim dan ImageAlpha untuk secara drastis mengurangi ukuran file gambar Anda. dengan menyeret dan melepaskan.

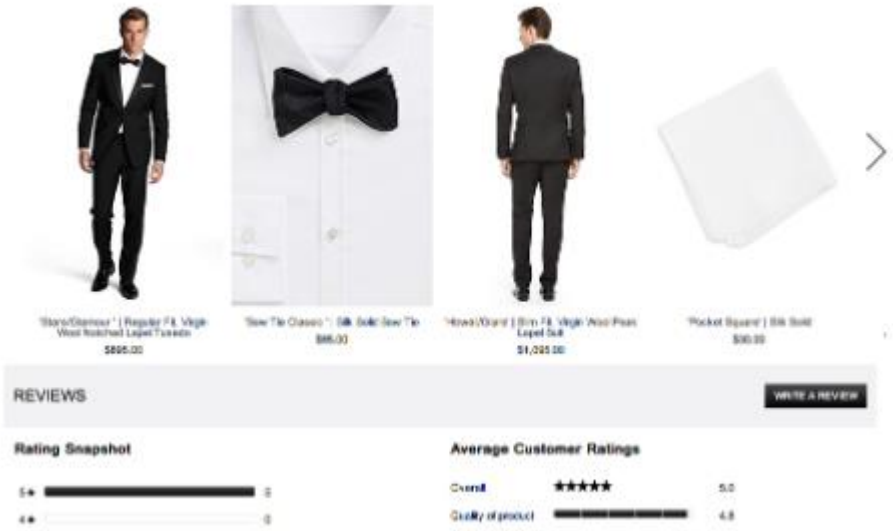
Tombol berbagi jaringan sosial adalah sumber umum lainnya dari pembengkakan. Untuk menghindari bobotnya yang berlebihan secara default, Anda dapat memuat kode tombol secara lambat hanya setelah pengguna menjelaskan bahwa mereka ingin menggunakannya. SocialCount dari Filament Group adalah contoh bagus dari teknik ini. Secara default, tombol-tombolnya hanyalah CSS dengan gambar kecil. Saat pengguna mengambil tindakan (misalnya, mengarahkan kursor ke tombol), skrip berbagi sebenarnya untuk jaringan tersebut dimuat secara dinamis dan ditambahkan ke tombol.

Meskipun berbasis jQuery, fungsionalitas yang dijalankannya cukup sederhana sehingga Anda dapat dengan mudah mengadopsinya untuk kerangka kerja Anda sendiri, atau bahkan untuk JavaScript vanilla. Anda dapat menerapkan pemikiran yang sama pada komponen lain di halaman Anda yang mungkin tidak terlihat jelas pada pandangan pertama. Dalam artikelnya yang luar biasa, Menyebarkan Aplikasi JavaScript, Alex Sexton berbicara tentang menunda pemuatan seluruh bagian kode hingga pengguna mengambil tindakan yang memerlukan kode tersebut.



Gambar 4.4 Tombol ulasan di situs Hugo Boss berada jauh di bawah halaman, dan mungkin jarang digunakan.

Contoh yang diberikannya adalah situs Hugo Boss. Jika pelanggan mengklik tombol “Tulis Ulasan”, kotak modal akan muncul bagi mereka untuk menilai produk. Tombol ini tidak hanya terletak jauh di bagian bawah halaman, tetapi kemungkinan besar tidak terlalu sering digunakan. Dengan tidak memuat kode dan aset yang diperlukan untuk kotak modal tersebut hingga beberapa saat setelah pemuatan halaman — bahkan mungkin menunggu hingga kursor berada dalam beberapa ratus piksel atau tombol telah diklik — ada sedikit peningkatan pada waktu buka halaman awal.



Gambar 4.5 Menunda pemuatan kode yang terkait dengan kotak modal tersebut hingga pemuatan halaman awal setelahnya dapat berdampak nyata pada waktu buka.

Untuk teknik yang lebih spesifik, pastikan untuk membaca bab yang ditulis oleh Aaron Gustafson dan Mat Marquis (selain bab yang disebutkan sebelumnya oleh Christian Heilmann). Intinya: ada banyak hal yang dapat Anda lakukan untuk menghindari kembung berlebih dengan cepat dan mudah.

Jika Anda Tidak Bisa Melakukannya, Palsu

Para eksekutif di bandara Houston punya masalah: mereka mendapat banyak sekali keluhan mengenai berapa lama waktu yang dibutuhkan penumpang untuk mengambil tas mereka di pengambilan bagasi. Tentu saja, hal pertama yang mereka putuskan adalah menyewa lebih banyak bagasi. penangan. Waktu tunggu turun menjadi hanya delapan menit, namun keluhan tetap ada.

Menggali lebih dalam, seperti yang sering dilakukan, mengungkap sesuatu yang jauh lebih menarik. Ternyata, perjalanan dari pesawat menuju area pengambilan bagasi memakan waktu sekitar satu menit. Artinya, kebanyakan orang menghabiskan tujuh menit berdiri dan menunggu di pengambilan bagasi. Jika Anda pernah ke bandara, Anda pasti paham betapa membosankannya hal itu.

Jadi, alih-alih mencoba mempersingkat waktu tunggu, mereka malah membuat perjalanan ke pengambilan bagasi menjadi lebih lama. Penumpang kini harus berjalan enam kali lebih jauh untuk mendapatkan tas mereka. Di permukaan, hal ini terdengar seperti cara

jitu untuk membuat marah orang. Namun kenyataannya jumlah pengaduan menurun drastis. Penumpang sibuk selama delapan menit menunggu, jadi meskipun mereka membutuhkan waktu yang sama lamanya untuk mengambil tas, namun terasa lebih cepat.

Ada banyak cerita tentang cara menarik orang memandang perjalanan waktu. Di New York, hampir semua tombol yang dapat Anda tekan di penyeberangan pejalan kaki tidak berfungsi dan belum berfungsi sejak tahun 1980an. Namun banyak orang yang masih memaksakannya, percaya bahwa ini akan menghemat beberapa detik. Disney, seperti banyak atraksi lainnya, menyembunyikan panjang antrean dengan memutarnya dalam pola berkelok-kelok dan sering kali menutupi sebagian antrean dari pandangan dengan membungkusnya secara strategis di sekitar bangunan. Mengapa?

Karena jika Anda tidak dapat melihat seluruh garisnya, dan Anda tampak bergerak, kecil kemungkinannya Anda akan merasa frustrasi. Faktanya, ternyata orang lebih bahagia berada dalam antrean panjang yang bergerak cepat dibandingkan antrean pendek yang bergerak lambat meskipun total waktu menunggunya sama. Lebih dari sekedar angka, yang terpenting adalah bagaimana pengguna memandang situs Anda. Apakah rasanya situs dimuat dengan cepat? Apakah interaksinya terasa cepat dan cepat, atau tertunda dan lamban? Kita dapat menerapkan pemikiran yang sama pada situs dan aplikasi yang kita bangun. Faktanya, beberapa aplikasi sukses telah melakukan hal tersebut.

Di penghujung tahun 2011, Mike Krieger, salah satu pendiri Instagram, memberikan presentasi bertajuk Rahasia Desain Seluler Secepat Kilat. Selama ceramahnya, dia fokus pada tiga rahasia:

1. Lakukan tindakan dengan optimis
2. Memuat konten secara adaptif
3. Pindahkan bit saat tidak ada yang melihat

Ada baiknya untuk mengeksplorasi masing-masing dengan lebih detail.

Lakukan Tindakan dengan Optimis

Katakanlah pengunjung situs Anda akan memberikan komentar. Untuk melakukannya, mereka mengklik tombol yang mengirimkan formulir. Ketika mereka melakukannya, ada dua hal yang terjadi. Formulir tersebut, menggunakan AJAX, mengirimkan permintaan ke server dan grafik pemuatan muncul untuk memberi tahu pengguna bahwa pengajuan mereka sedang dalam proses. Ketika skrip mendengar kabar dari server bahwa tugas berhasil diselesaikan, skrip memperbarui halaman yang memperingatkan pengunjung.

Ini adalah cara yang biasa dilakukan, tapi mungkin ini bukan cara terbaik. Permintaan tersebut, khususnya pada jaringan berlatensi tinggi, dapat memakan waktu beberapa ratus milidetik penundaan yang sangat nyata bagi orang yang mencoba mengirimkan komentarnya. Instagram telah mengambil pendekatan berbeda untuk menghindari penundaan tersebut. Segera setelah orang tersebut mengirimkan komentar, komentar itu akan muncul di halaman. Permintaan tersebut terjadi di latar belakang. Bagi orang yang mengirimkan komentar, sepertinya hal itu terjadi secara instan. Pada kenyataannya, proses ini memerlukan waktu yang sama lamanya dengan proses online lainnya namun persepsinya kini jauh lebih baik.

Mike menyebutnya sebagai “melakukan tindakan dengan optimis”, namun ada pula yang menyebutnya sebagai UI asinkron. Idenya sama: hilangkan status pemuatan dan biarkan pengguna merasakan segala sesuatunya bergerak lebih cepat. Jika tugasnya gagal, beri tahu mereka secara perlahan setelah kejadian tersebut dan biarkan mereka mengirimkannya kembali dengan mudah.

Contoh bagus lainnya adalah Polar, aplikasi polling yang populer. Saat Anda membuat jajak pendapat, jajak pendapat tersebut langsung muncul di feed Anda. Sekali lagi, ada beberapa UI asinkron yang cerdas yang sedang bekerja. Apa yang sebenarnya terjadi saat Anda membuat jajak pendapat baru adalah Polar membuat salinan lokal sementara dari jajak pendapat tersebut dan memindahkannya ke bagian atas feed Anda. Salinan sementara berfungsi penuh Anda dapat memilih dan mengomentarinya dan suara serta komentar tersebut akan dimasukkan ke jajak pendapat sebenarnya setelah diunggah.

Di latar belakang, Polar mengunggah salinan sementara ke servernya. Jika gagal, mereka mencoba lagi beberapa kali sebelum akhirnya mengakui kekalahan kepada pengguna. Hasilnya, sekali lagi, prosesnya terasa luar biasa cepat. Penting untuk dicatat bahwa baik untuk Instagram maupun Polar, solusi ini tidak sepenuhnya ideal dari sudut pandang teknik: ada lebih banyak kerumitan yang terlibat. Namun keuntungannya adalah pengguna mendapatkan sistem yang terasa instan.

4.3 PRAMUAT KONTEN SECARA ADAPTIF

Rahasia selanjutnya adalah melakukan pramuat konten secara adaptif. Ini tidak berarti memuat apa pun secara membabi buta dan segala sesuatu yang Anda bisa sebelum dibutuhkan. Sebaliknya, Anda perlu mempertimbangkan apa yang Anda ketahui tentang perilaku pengguna dan memuat secara selektif berdasarkan hal tersebut. Instagram menggunakan teknik ini untuk feed foto mereka.

Pada awalnya, mereka memuat semuanya sesuai urutan yang muncul di halaman, seperti yang dilakukan browser. Namun hal ini tidak selalu merupakan pendekatan terbaik karena hal tersebut belum tentu merupakan peringkat kepentingan yang tepat bagi pengguna. Sebaliknya, Instagram memilih untuk memprioritaskan ulang pemuatan berdasarkan ke mana pengguna akhirnya menggulir, menggunakan minat mereka untuk memprioritaskan apa yang dimuat pertama kali.

Pindahkan Bit Saat Tidak Ada Yang Menonton

Pernahkah Anda melihat pesulap tampil? Trik yang mereka lakukan didasarkan pada ilusi dan gangguan. Gerakan besar dan efek mencolok menyesatkan Anda dari sulap yang terjadi tepat di depan Anda. Anda dapat melakukan trik yang sama untuk memberikan pengalaman yang lebih baik bagi pengunjung Anda. Dalam presentasinya, Mike mendemonstrasikan salah satu cara untuk menerapkan pendekatan semacam ini.

Saat pengguna mendaftar ke Instagram, mereka diminta mengisi beberapa detail dasar. Saat hal ini terjadi, di latar belakang Instagram mulai mencari rekomendasi siapa yang harus diikuti. Hasilnya adalah saat pengguna mengirimkan formulir beserta informasi akunnya, rekomendasi langsung disajikan.

Instagram menggunakan trik yang sama pada unggahan gambar. Setelah Anda memilih filter untuk gambar Anda, Anda dapat memilih opsi seperti tempat untuk membagikan gambar, atau memberi geotag pada gambar tersebut dengan lokasi. Sementara itu, Instagram sudah mengunggah gambar di latar belakang untuk mengurangi waktu tunggu pengguna di akhir proses.

Perlu dicatat bahwa dalam kedua kasus di atas, ada kemungkinan besar orang tersebut akan melanjutkan ke halaman berikutnya. Ini adalah sebuah jurang yang licin antara bit-bit yang bergerak ketika tidak ada orang yang melihat dan menghabiskan bandwidth dan data semua orang untuk halaman-halaman yang mungkin tidak pernah mereka lihat dan aset-aset yang mungkin tidak pernah mereka perlukan. Namun jika ada kemungkinan besar pengunjung Anda akan membutuhkan aset ini suatu saat nanti, masuk akal untuk melakukan sedikit pra-cache agar tetap selangkah lebih maju.

Melampaui Teknis

Ini semua adalah hal-hal yang belum tentu jelas dan bukan merupakan solusi teknis semata. Hal ini memerlukan pertimbangan yang cermat tentang pengalaman pengguna dan pemahaman yang kuat tentang cara pengguna berinteraksi dengan aplikasi. Mereka juga membutuhkan sedikit kerja keras, tetapi hasil akhirnya sangat berharga. Luke Wroblewski (salah satu otak di balik Polar)33 menjelaskannya dengan baik:

Jika membuat jajak pendapat versi sementara berfungsi penuh dan menggunakan beberapa proses latar belakang untuk memastikan pengunggahan berhasil terdengar seperti banyak upaya ekstra untuk membuat segalanya terasa cepat—itu memang benar. Namun hasil akhirnya sepadan, ketika orang membuat sesuatu di Polar, hal itu tampak seketika. Dan dalam hal ini, persepsi mengalahkan kenyataan.

4.4 PENYELESAIAN TUGAS

Masing-masing prinsip di atas pada akhirnya bertujuan untuk mengurangi jumlah waktu yang dibutuhkan pengguna untuk menyelesaikan tugas tertentu. Pentingnya penyelesaian tugas tidak dapat diabaikan. Ada studi klasik yang dilakukan oleh UIE pada tahun 200134 mengenai dampak waktu yang dibutuhkan untuk menyelesaikan tugas terhadap persepsi kinerja pengunjung. Para peneliti mendudukkan orang-orang di depan sepuluh situs berbeda menggunakan modem 56kpbs dan memberi mereka tugas untuk diselesaikan.

Kejutan muncul ketika orang menilai situs paling lambat (Amazon.com) sebagai salah satu yang tercepat ketika ditanya. Alasannya adalah Amazon.com memungkinkan orang menyelesaikan tugas mereka dalam langkah yang lebih sedikit. Pada akhirnya, inilah yang menentukan: seberapa cepat pengguna merasakan situs tersebut. Anda bisa mencapai kemajuan dengan menerapkan teknik kinerja yang sering dikutip oleh pengembang, namun untuk memengaruhi perasaan pengguna tentang kinerja situs Anda, pengoptimalan kinerja harus melibatkan perancang.

Jika Anda seorang desainer, anggaplah diri Anda sebagai garis pertahanan pertama. Ya, pada akhirnya pengembang harus melakukan banyak optimasi spesifik, namun Andalah yang harus mengatur tahapannya. Anda harus membuat keputusan sejak dini yang akan

mendorong situs menjadi secepat dan indah, atau mendorongnya menjadi indah, namun membengkak.

Kesimpulan

Kinerja yang baik tidak terjadi secara kebetulan: kinerja yang baik harus direncanakan dan dirancang dengan cermat. Untuk menghindari kontribusi terhadap meningkatnya obesitas di Web, kinerja harus dimasukkan ke dalam proses sejak awal proses. Mengambil langkah-langkah seperti menetapkan anggaran kinerja, menyimulasikan kecepatan koneksi yang buruk, dan menggunakan browser dan perangkat nyata sedini mungkin akan membantu membuat semua orang yang terlibat lebih sadar akan dampak buruk pada kinerja.

Ini bukan lagi sesuatu yang dapat kami serahkan kepada pengembang. Desainer dapat menentukan tahap awal bagaimana kinerja situs. Melalui pertimbangan yang cermat, mereka dapat memastikan bahwa situs tersebut akan terlihat dan terasa cepat. Web tidak harus menjadi lebih berat. Kami dapat memperbaikinya dengan memprioritaskan kinerja dan menghormati orang-orang yang menggunakan situs kami. Dengan sedikit perhatian, situs bisa menjadi indah dan cepat. Mari berikan pengalaman cepat yang sangat mereka inginkan kepada orang-orang yang mengunjungi situs kami.

BAB 5
DESAIN WEB YANG KUAT, BERTANGGUNG JAWAB, DAN RESPONSIF

Desain web responsif mulai mendapatkan reputasi dan bukan reputasi yang pantas diterimanya. Ketika saya mengatakan bahwa saya di sini untuk menunjukkan kepada Anda beberapa cara yang dapat kita lakukan untuk membangun situs web responsif yang ramping, kaya fitur, dan sangat mudah diakses yang terlihat dan berfungsi dengan baik untuk pengguna mana pun dalam konteks penjelajahan apa pun, saya yakin saya' telah menarik perhatian saya dan saya memahaminya. Sejak Ethan Marcotte menciptakan istilah ini beberapa tahun yang lalu, kita semua telah dibombardir oleh postingan blog yang sensasional tentang kegagalan desain Web responsif sebagai sebuah metodologi di masa lalu, sekarang, atau di masa depan: ia tidak dapat melakukan hal ini dan tidak cukup melakukan hal tersebut. tentang itu; tidak ada gunanya tanpa skrip atau komponen sisi server tertentu. Desain Web yang Responsif perlu menghemat lebih banyak waktu dibandingkan Natal, jika Anda bisa mempercayai blog dan acara spesial liburan televisi.

Lonceng kematian paling umum yang pernah kita dengar akhir-akhir ini adalah bahwa setiap halaman situs responsif ditakdirkan untuk berbobot beberapa lusin megabita dan tidak ada satu pun dari kita yang dapat melakukan apa pun di antara kita untuk mengatasinya. . Tampaknya ada banyak bukti yang mendukung hal ini juga. Oakley baru-baru ini meluncurkan situs web responsif dengan halaman indeks 85 MB, lengkap dengan layar pemuatan di muka. Halaman arahan untuk item navigasi utama di time.com masing-masing berukuran rata-rata 3 MB.

Saya senang melihat begitu banyak orang yang merasa tidak nyaman dengan potensi masalah bandwidth pada situs web responsif, namun kita harus berhati-hati dalam mengarahkan kesalahan kita: kita tidak dapat meminta pertanggungjawaban desain Web responsif atas kesalahan yang telah kita buat.

Palu Kami yang Rusak

Di kehidupan lain, bertahun-tahun yang lalu, saya adalah seorang tukang kayu. Saat saya baru memulai, saya bekerja dengan dua orang yang masing-masing sudah berkecimpung di bidang ini selama lebih dari dua puluh tahun. Seperti banyak blogger lainnya, saya juga memiliki kekhawatiran mengenai alat yang kami gunakan — anehnya, kekhawatiran yang tampaknya tidak dimiliki oleh blogger lain. Semua peralatan mereka yang penyok dan penuh bekas luka adalah hal yang baru bagi saya: berat, tidak nyaman, dan mustahil untuk dikelola. Palu yang diberikan kepadaku, khususnya. Saya yakin itu cacat, karena beratnya sama besarnya.

Kini, meskipun alatnya sederhana, palu memungkinkan pengrajin yang terampil melakukan banyak pekerjaan: membingkai seluruh rumah, memasang atap, membuat furnitur, dan tentu saja membuka botol bir. Namun, ketika saya memegang milik saya, itu

hanya baik untuk memukul ibu jari saya, meninggalkan lubang di dinding, dan membuat paku terlepas dari atap dan mengenai mobil baru pemilik rumah. Membuka botol bir sudah cukup baik, seperti yang saya ketahui segera setelah insiden Lexus yang terkelupas.

Upaya awal kami dalam desain Web responsif tidak jauh berbeda dengan hari-hari awal saya di bidang pertukangan. RWD adalah alat sederhana dan merupakan hal baru bagi kami, meskipun teknik yang dicakupnya kisi berbasis persentase, kueri media, dan media fleksibel bukanlah hal baru. Di tangan yang tepat, ia dapat menghasilkan sesuatu yang luar biasa: tata letak yang seolah-olah dibuat khusus agar sesuai dengan tampilan apa pun di perangkat apa pun, termasuk yang tidak dapat kami prediksi. Memang benar, kita juga bisa melakukan kesalahan dalam hal ini. Situs web besar dan haus sumber daya tersebut bukanlah kesalahan desain web yang responsif, karena beberapa lusin paku yang bengkok dapat disebabkan oleh palu yang rusak. Tren ini ada pada kita, dan Anda tidak melihat ada tukang kayu yang menulis postingan blog tentang bagaimana palu adalah metodologi yang gagal karena mereka sering menjatuhkan palu. Kita bisa melakukan lebih baik daripada menyalahkan alat kita atas kesalahan kita.

5.1 PENINGKATAN PRESUMTIF

Kami tidak bisa terlalu disalahkan. Anda dan saya mungkin melakukannya dengan cukup mudah, dari sudut pandang penelusuran. Kami adalah pengembang kami memiliki komputer yang cepat dan bandwidth yang tersedia. Saya tidak yakin bisa menghirup udara yang tidak memiliki Wi-Fi di dalamnya.

Namun, itulah konteks istimewa kami; itulah yang biasa kami lakukan. Itu nyaman. Kami berasumsi bandwidth tinggi dan jaringan stabil, karena hal itu sudah pasti bagi kami. Kita dapat dengan aman berasumsi bahwa mengirimkan permintaan akan menghasilkan sesuatu yang dikirim kembali, di luar terowongan kereta bawah tanah sesekali. Bagi kita yang membangun Web, hal ini merupakan hal yang paling mudah dilakukan di Web, dan mungkin sebagai hasilnya, rata-rata halaman Web sekarang berukuran sekitar 1,4 MB2.

Jika Anda hanya pernah mengakses Web melalui koneksi seluler yang tidak dapat diandalkan, halaman seperti ini lebih dari sekadar ketidaknyamanan kecil: halaman tersebut mungkin tidak dapat Anda gunakan sama sekali. Halaman-halaman ini adalah bagian dari Web yang bukan untuk Anda. Halaman seperti ini adalah bukti bahwa kami membangun dari tempat yang memiliki hak istimewa. Meskipun saya yakin ini bukan niat siapa pun, yang kami lakukan akhir-akhir ini adalah membangun Web untuk kami.

[...] dalam jalinan ruang-waktu web, konteks tidak lagi hanya tentang saat ini dan di sini. Sebaliknya, konteks mengacu pada struktur fisik, digital, dan sosial yang mengelilingi titik penggunaan.

— Cennydd Bowles,
"Mendesain dengan konteks3", 16 Februari 2013

Akses universal adalah kebenaran mendasar yang mendasari Web.

Membangun situs yang besar dan banyak sumber daya berarti mengecualikan jutaan pengguna di negara-negara berkembang yang hanya mengetahui Web melalui ponsel menengah atau lebih baik lagi pengguna membayar untuk setiap kilobyte yang mereka konsumsi; pengguna yang sudah harus mengawasi situs mana yang harus mereka hindari setiap hari karena biaya mengunjunginya. Bukan biaya bandwidth yang samar-samar, namun biaya ekonomi yang sebenarnya.

Pengecualian para pengguna ini telah menjadi sebuah ramalan yang terwujud dengan sendirinya, dengan beberapa pengembang mengklaim bahwa kurangnya lalu lintas dari daerah dengan bandwidth terbatas membuat optimasi tidak diperlukan. Situs yang tidak dioptimalkan kemungkinan besar tidak akan menerima lalu lintas dari area dengan bandwidth terbatas karena situs tersebut tidak dioptimalkan.

Akhir tahun lalu, Chris Zacharias mulai mengurangi ukuran halaman pemutar YouTube sebesar 1,2 megabyte, dengan tujuan menjadikannya di bawah 100KB, dalam sebuah proyek yang ia beri nama kode "Feather." Dengan mengurangi halaman pemutar menjadi 98KB dan hanya 12 permintaan, dia sebenarnya melihat peningkatan tajam dalam latensi rata-rata halaman. Ketika peningkatan latensi ini diplot secara geografis, terungkap bahwa seluruh populasi kini dapat menggunakan layanan ini meskipun koneksinya lebih lambat. Halaman yang sebelumnya terlalu mahal untuk dirender dan memakan waktu rata-rata dua puluh menit kini dapat dimuat hanya dalam beberapa menit.

[...] seluruh populasi tidak dapat menggunakan YouTube karena terlalu lama untuk melihat apa pun. Di bawah Feather, meskipun membutuhkan waktu lebih dari dua menit untuk sampai ke frame pertama video, menonton video sebenarnya menjadi sebuah kemungkinan yang nyata. Selama seminggu, berita tentang Feather telah menyebar di wilayah ini dan sebagai hasilnya, jumlah kami benar-benar tidak seimbang. Sejumlah besar orang yang sebelumnya tidak dapat menggunakan YouTube tiba-tiba bisa.

— Chris Zacharias,
"Berat Halaman Penting", 21 Des 2012

5.2 PENINGKATAN PROGRESIF

Desain Web Responsif, dalam arti tertentu, merupakan perpanjangan alami dari peningkatan progresif. Desain Web yang responsif dan peningkatan progresif keduanya bertujuan untuk memastikan pengalaman yang dapat digunakan bagi semua pengguna, apa pun konteks penjelajahan mereka. RWD lebih mementingkan penyediaan tata letak dan presentasi yang terasa alami bagi pengguna untuk mengakses konten, dan peningkatan progresif bertujuan untuk memastikan akses ke konten tersebut terlepas dari fitur dan kemampuan browser. Kita sering melihat kedua konsep tersebut digabungkan sebuah kebingungan yang jarang terjadi dalam pengembangan Web yang akhirnya menguntungkan semua orang. Bagi banyak orang, desain Web responsif lebih dari sekedar tata letak fleksibel yang disesuaikan dengan permintaan media, dan mencakup permasalahan seperti lanskap fitur browser seluler yang selalu berfluktuasi, pengalaman offline, dan kinerja jaringan.

Kekhawatiran-kekhawatiran ini tentu saja merupakan peningkatan yang progresif, namun saya senang melihatnya menjadi sesuatu yang kabur kekhawatiran yang hanya berfungsi untuk menyoroti potensi masalah yang mungkin menghambat akses pengguna ke situs yang kami buat, dan menjaga masalah tersebut. di depan pikiran kita.

Landasan peningkatan progresif adalah memikirkan makna yang melekat pada suatu elemen: apa cara terbaik untuk mengekspresikan widget ini hanya dalam markup? Bagaimana kita dapat mewakili niat suatu elemen dengan cara yang dapat diterapkan oleh semua orang, dan kemudian membangunnya dari sana? Inti dari setiap widget jQuery Mobile adalah markup yang bermakna terlepas dari JavaScript atau CSS. Penggeser rentang5, misalnya, didukung oleh masukan angka, yang diturunkan menjadi masukan teks di browser yang tidak terbiasa dengan masukan angka yang relatif baru. Dalam lingkungan berkemampuan JavaScript, masukan tersebut ditingkatkan menjadi kontrol geser. Baik pengguna menerima pengalaman yang disempurnakan atau dasar, mereka akan dapat memasukkan datanya. Saat Anda menerapkan prinsip ini ke seluruh situs, Anda membangun situs yang masuk akal situs yang dapat digunakan sebelum menambahkan penyempurnaan JavaScript Anda ke dalamnya.

Tidak sulit untuk menemukan angka tentang jumlah pengguna yang telah mengubur tombol pemicu di preferensi browser mereka dan menonaktifkan JavaScript sepenuhnya. Seperti yang mungkin sudah Anda duga, hal ini cukup jarang terjadi dan angka-angka tersebut biasanya disertai dengan seruan untuk mengabaikan peningkatan progresif dan hanya memenuhi denominator umum tertinggi. Namun dukungan di Web jarang sekali bersifat biner bahkan jika kami berasumsi bahwa persentase pengguna kami yang sangat rendah akan menonaktifkan JavaScript secara massal, mengandalkan JavaScript untuk fungsionalitas inti situs akan menimbulkan titik kegagalan yang sangat besar.

Dalam bab tujuh, Aaron Gustafson akan membahas beberapa situasi ketika JavaScript mungkin tidak tersedia bagi pengguna Anda, namun poin ini perlu diulangi. Saya tidak perlu memberi tahu Anda betapa buruknya kode tersebut di sebagian besar widget atau iklan pihak ketiga. Apa yang terjadi jika pengguna Anda mengklik sesuatu sambil menunggu satu megabyte tombol “Suka” untuk diunduh? Apa yang terjadi jika salah satu tag iklan Anda rusak parah sehingga browser tidak lagi menggunakan JavaScript? Apakah situs Anda akan berfungsi?

Mengatakan bahwa setiap orang mengaktifkan JavaScript tidak lebih dari sekedar alasan untuk kenyamanan pengembang hal ini tidak meningkatkan kualitas pengguna. pengalaman, hanya milik kita. Meskipun kami dapat berasumsi bahwa situs yang sepenuhnya bergantung pada JavaScript dapat digunakan oleh setiap pengguna, kami tetap tidak memberikan manfaat apa pun kepada mereka. Kami masih memperkenalkan potensi bagi pengguna untuk melihat GIF yang dimuat tanpa batas waktu atau halaman kosong hal-hal yang membuat frustrasi yang sering kita lihat, bahkan di sini, dengan koneksi internet yang cepat dan andal. Saya pribadi telah melihat sejumlah situs yang dibangun dengan cemerlang mengalami time out atau menampilkan halaman kosong akibat plugin browser pemblokiran iklan: satu titik kegagalan yang menghentikan eksekusi setiap skrip lain pada halaman tersebut, menjadikan keseluruhan situs tidak dapat digunakan.

Memang benar, ini hanyalah ketidaknyamanan bagi Anda dan saya. Kita dapat dengan aman menentukan dari mana masalah seperti itu berasal dan mengatasinya. Bagi pengguna yang kurang paham teknologi, ketidaknyamanan yang mungkin terjadi bagi kami bisa berarti hilangnya akses sepenuhnya ke halaman atau keseluruhan situs.

Akses itu Penting

Pengeboman Boston Marathon terjadi pada tanggal 15 April 2013, dan seperti kebanyakan orang, saya membuka situs Boston Globe sepanjang hari untuk mencari tahu apa yang terjadi di sekitar saya alasan di balik sirene yang terus-menerus dan untuk meyakinkan bahwa ledakan tersebut Saya mendengar sepanjang hari ada ledakan terkendali yang dilakukan oleh polisi. Bisa dikatakan, akses yang dapat diandalkan ke BostonGlobe.com menjadi sangat, sangat penting bagi banyak orang.

Mengingat lonjakan awal yang luar biasa dan lalu lintas yang berkelanjutan sepanjang hari, dan meskipun tim pengembang dan operasi Globe telah berupaya keras, mudah untuk memahami bagaimana server mereka mulai mengalami beberapa masalah. Pada puncak lalu lintas, jaringan pengiriman konten mereka kehabisan tenaga, sehingga menghasilkan situs tanpa aset eksternal: tanpa gambar, tanpa CSS, dan tanpa JavaScript.

Saya yakin banyak di antara Anda yang meringis, tapi itu bukanlah bencana yang mungkin Anda bayangkan. Benar, situsnya tidak secantik yang seharusnya, tapi berfungsi dengan baik. Seandainya Globe bergantung sepenuhnya pada JavaScript untuk rendering atau navigasi, atau jika skrip dan gaya tidak dibangun di atas landasan penyempurnaan progresif, puluhan ribu pengguna yang merasa tidak nyaman akan bergantung pada mesin pencari berita mereka.

Kita akan membahas tentang peningkatan progresif yang berkaitan dengan desain Web responsif di sisa bab ini, namun tidak sepenuhnya seperti yang mungkin pernah Anda baca sebelumnya. Bagi kami, ini bukan sekadar menambahkan fungsionalitas dengan cara yang tidak mencolok, namun menerapkan filosofi yang sama pada cara kami memuat aset situs (JavaScript, CSS, gambar, dan bahkan markup kami) dengan cara yang paling sesuai untuk pengguna. konteks pengguna. Kami akan berupaya membangun situs web responsif yang tidak hanya terlihat dibuat khusus untuk tampilan apa pun, namun juga memberikan aset yang sesuai dengan perangkat dan konteks.

Gambar 5.1 Saat mengarahkan kursor ke tautan di navigasi utama, BostonGlobe.com menampilkan tarik-turun yang berisi artikel unggulan, artikel terbaru di bagian tersebut, dan subbagian terkait.

Menandai

Dalam hal markup kami saja fondasi untuk semua hal lain di situs kami — ada banyak ruang untuk pengoptimalan. Bagaimana kami memberikan pengalaman yang kaya sambil memastikan bahwa bagian terpenting dari sebuah halaman, yaitu konten itu sendiri, tersedia sesegera mungkin?

Cara paling sederhana (walaupun jauh dari ideal) untuk menangani menampilkan dan menyembunyikan konten ini adalah dengan menggunakan display: none; dan menampilkan menu tarik-turun saat tautan navigasi diarahkan. Namun kami dapat berasumsi dengan aman bahwa pengguna yang ingin mengetahui skor pertandingan Bruins kemarin tidak akan mengarahkan kursor ke setiap tautan dalam navigasi di setiap halaman situs. Sebagian besar konten ini tidak akan pernah terlihat oleh pengguna, yang masih akan meminta semua markup ini termasuk gambar, yang akan tetap dimuat di sebagian besar browser, meskipun tidak pernah ditampilkan di setiap halaman. situs tersebut.

Meskipun kami tidak ingin menyertakan semua konten tersebut untuk berjaga-jaga, kami juga tidak ingin membatasi pengguna pada titik henti sementara tertentu, atau membuat asumsi berdasarkan konteks pengguna. Drop-down adalah tambahan yang berguna untuk situs ini, dan layak untuk disimpan. Kami membutuhkan dasar, sarana markup yang ketat agar pengguna dapat ikut serta dalam konten di situs, yang lebih sederhana dari kedengarannya. Seperti yang diharapkan, semua link di navigasi utama membawa pengguna ke halaman arahan yang berisi semua konten yang sama: artikel unggulan, artikel terbaru, dan navigasi untuk setiap subbagian. Menu tarik-turun merupakan kemudahan yang menyenangkan, namun tidak penting untuk menavigasi situs. Masuk akal untuk memperlakukan menu drop-down ini sebagai penyempurnaan, dan memuatnya melalui JavaScript hanya jika diperlukan.

Untuk dengan malas memuat konten tidak penting lainnya di situs Globe tanpa menimbulkan hambatan akses apa pun, dan terinspirasi oleh pendekatan yang kami ambil pada menu drop-down, kami mengembangkan pola JavaScript berbasis markup sederhana

yang diberi nama AjaxInclude6. AjaxInclude meningkatkan tautan dengan menggunakannya untuk mengambil bagian dari konten yang ditautkan. Tag jangkar itu sendiri berfungsi sebagai cadangan, memastikan bahwa pengguna akan dapat mengakses konten yang mendasarinya terlepas dari apakah JavaScript tersedia atau tidak, tetapi juga menyediakan semua informasi yang dibutuhkan skrip kita untuk mengambil konten terkait dan menerapkannya ke halaman: tag lokasi bagian dokumen yang akan digunakan sebagai pengganti tautan; dan bagaimana konten tersebut harus dimasukkan ke halaman saat ini relatif terhadap posisi link.

Untuk mengganti tautan dengan markup yang dimasukkan:

```
<a href="/sports" data-replace="articles/sports/fragment">Sports</a>
```

Untuk menyisipkan konten sebelum posisi tautan di sumber:

```
<a href="/sports" data-before="articles/sports/fragment">Sports</a>
```

Untuk menyisipkan konten setelah posisi link di sumber:

```
<a href="/sports" data-after="articles/sports/fragment">Sports</a>
```

Dari sudut pandang desain responsif, skrip yang sama ini dapat membantu kita menghindari penayangan semua konten dan menyembunyikannya secara selektif, alih-alih memungkinkan kita menyertakan konten hanya jika sesuai dengan tata letak kita. AjaxInclude juga memungkinkan kita menggunakan atribut kedua sehingga kita dapat memenuhi syarat penyertaan konten di atas atau di bawah breakpoint tertentu.

Di sini, kami akan menyertakan fragmen yang direferensikan hanya pada lebar area pandang 30em ke atas:

```
<a href="/sports" data-append="articles/sports/fragment"
data-media="(min-width: 30em)">Sports</a>
```

Memulai dengan konten inti situs sebagai fondasi memungkinkan kami membuat keputusan yang lebih besar tentang bagaimana dan kapan kami dapat memuat aset tambahan tanpa mengambil risiko meninggalkan pengguna mana pun. Jika semuanya gagal, tujuan situs akan tetap utuh: tidak ada pengguna yang tidak dapat menggunakan situs tersebut. Hal ini menetapkan dasar yang berfungsi untuk semua orang, sekaligus memberi kami kebebasan untuk memberikan skrip dan gaya yang sesuai konteks dengan cara yang sama: situs web yang berfungsi untuk semua, dengan penyempurnaan bersyarat jika diperlukan.

Alasan yang sama memperluas proyek terbaru, sebuah situs yang menampilkan carousel gambar di setiap halaman, menampilkan daftar apartemen. Ini sering kali berarti menyertakan lusinan gambar beresolusi tinggi per halaman.

Namun jika carousel ini muncul bersamaan dengan sejumlah informasi penting lainnya yang mungkin menjadi perhatian utama pengguna, tentu saja lebih penting daripada foto-foto dalam listingan tidak ada jaminan bahwa pengguna akan mengeklik galeri saat membuka setiap halaman. Seandainya kami menyertakan semua tag img terkait di markup kami dan menyembunyikannya dengan CSS, itu tidak akan mencegah permintaan gambar tersebut. Kami berpotensi membebani pengguna kami sebesar megabita sekaligus, hanya untuk membaca beberapa cuplikan teks.

Dalam contoh ini kami menerapkan filosofi "sesuai kebutuhan" yang sama seperti yang kami lakukan dengan pola navigasi AJAX: foto awal disertakan dengan payload pertama halaman, dan skrip galeri diinisialisasi. Meskipun menggunakan AJAX untuk mengambil masing-masing slide saat pengguna bernavigasi melalui galeri akan menambah penundaan yang merepotkan dan merugikan pengalaman keseluruhan, kami ingin memastikan bahwa kami masih merespons maksud pengguna.

Dalam contoh ini, ketika pengguna memicu tautan berikutnya, kami akan mengambil dan menambahkan semua slide lainnya sekaligus. Hal ini masih mengakibatkan sedikit penundaan; pengguna disajikan dengan indikator pemuatan setelah menekan item kedua dalam tayangan slide sementara item lainnya dimuat. Setelah beberapa kali pengulangan, kami akhirnya memuat foto pertama dan kedua di bagian depan. Saat pengguna mengklik item berikutnya di galeri yang sudah ada di markup item ketiga dan seterusnya akan dimuat secara diam-diam di belakang layar dan ditambahkan ke tayangan slide. Pada saat pengguna berinteraksi lagi dengan pemicu berikutnya beberapa saat kemudian, slide ketiga sudah siap dan menunggu mereka. Hasilnya adalah pengurangan biaya bandwidth secara besar-besaran untuk setiap pengguna di situs, namun pada saat yang sama memberikan pengalaman yang sepenuhnya lancar.

Pengoptimalan semacam ini merupakan masalah kemahiran dan penyempurnaan konsep sederhana langkah yang tampaknya kecil namun memiliki potensi manfaat yang sangat besar bagi pengguna kami.

5.3 CSS

Setelah kami memiliki markup yang berarti sebagai landasan dan kami tahu bahwa kami memastikan pengalaman yang bermanfaat bagi semua orang, kami dapat membuat beberapa keputusan penting tentang cara kami memberikan pengalaman tersebut tanpa mengabaikan siapa pun.

Saat mengerjakan proyek jQuery Mobile, kami segera menyadari bahwa menyajikan CSS inti kami tanpa pandang bulu dapat menimbulkan potensi pengalaman yang rusak. Siapa pun yang menggunakan feature phone atau ponsel pintar versi awal tidak akan memiliki perlengkapan yang memadai untuk menghadapi gaya-gaya canggih, bahkan jika kita berasumsi bahwa gaya-gaya itu sendiri akan menurun dengan baik. Hanya dengan mencoba mengurai style sheet itu sendiri dapat menimbulkan masalah pada platform yang umum seperti perangkat Blackberry yang baru berumur beberapa tahun.

Kami menetapkan gagasan untuk menyajikan dua tingkat peningkatan yang berbeda semacam titik henti pengalaman setelah menentukan apakah browser mampu menangani tata letak yang rumit dan fungsionalitas yang ditingkatkan, atau apakah browser tersebut mendekati ponsel menengah. Meski terdengar menakutkan, proses ini ternyata lebih sederhana dari yang Anda kira. Langkah pertama adalah membagi style sheet kita.

Lembar gaya awal adalah kumpulan gaya berkaliber ponsel: ukuran font, elemen blok versus sebaris, beberapa latar belakang warna solid. Style sheet ini dikirimkan ke setiap pengguna. Karena ini disajikan untuk semua orang, kami baru-baru ini mulai menggunakan Normalisasi CSS reset sebagai dasar untuk style sheet awal kami. Daripada bertindak sebagai penyetelan ulang yang menghilangkan gaya default browser, ini memberikan serangkaian standar yang dinormalisasi dan masuk akal: berguna untuk menggabungkan gaya kami yang disempurnakan untuk pengguna yang memenuhi syarat, sekaligus berfungsi sebagai dasar yang masuk akal untuk gaya dasar kami.

Style sheet yang disempurnakan hampir semuanya tata letak tingkat lanjut, animasi, font Web, dan sebagainya tentu saja diperkecil dan di-gzip. Apa yang kita dapatkan pada akhirnya adalah dua pengalaman yang tampak sangat berbeda, memang ada, tetapi bukan pengalaman inti yang berbeda. Kami tidak sekadar menyembunyikan apa pun, bahkan dari pengguna biasa. Tidak ada seorang pun yang tertinggal dalam kedinginan. Pengguna dengan perangkat yang lebih lama atau kurang bertenaga akan diberikan tampilan situs yang jauh lebih bermanfaat. Dalam hal ini, mereka mungkin tidak berharap banyak pada hal-hal menarik: pertanyaan kuno “Apakah situs web saya harus terlihat sama di setiap browser?”.

Dalam versi asli Enhance.js8 dari Filament Group dan versi awal jQuery Mobile, kami menggunakan serangkaian pengujian fitur yang sebagian besar tidak terkait untuk menentukan apakah suatu perangkat memenuhi syarat untuk pengalaman dasar atau pengalaman yang ditingkatkan. Kami akan menetapkan hasil pengujian ini dalam cookie, dan menggunakan variabel tersebut untuk mengirimkan aset sepanjang waktu pengguna di situs.

Akhirnya kami menyadari bahwa pengujian ini sejalan dengan dukungan untuk kueri media, yang merupakan komponen kunci dalam tata letak kami yang kompleks dan pengujian yang jauh lebih relevan. Selain itu, ini memberi kami metode asli untuk mengirimkan style sheet ini secara kondisional, menghilangkan ketergantungan pada JavaScript.

```
<link rel="stylesheet" href="/css/basic.css">
<link rel="stylesheet" href="/css/enhanced.css" media="only all">
```

Di sini, style sheet dasar ditautkan seperti biasa. Semua orang mengerti itu. Atribut `media="only all"` pada style sheet yang disempurnakan memastikan bahwa style sheet hanya diterapkan oleh browser yang memahami kueri media.

Tentu saja, ini berarti mengecualikan versi IE yang lebih lama tetapi kami telah memberikan beberapa opsi di sana. Kami masih dapat mengirimkan style sheet kami yang telah disempurnakan ke Internet EXplorer melalui komentar bersyarat, sambil memastikan bahwa versi IE sebelum versi minimum yang kami tentukan mendapatkan pengalaman dasar yang dapat digunakan dengan sempurna. Daripada memilih versi minimum IE untuk didukung

dan membiarkan situs rusak di versi sebelumnya, kami hanya memberikan pengalaman dasar kepada versi IE sebelumnya.

```
<link rel="stylesheet" href="basic.css" id="basic">
<!--[ if ( gte IE 6 ) & ( lte IE8 ) ]>
<link rel="stylesheet" href="enhanced.css">
<![endif]-->
<!--[ if ( !IE ) | ( gte IE 9 ) ]><!-->
<link rel="stylesheet" href="enhanced.css" media="only all">
<!--<![endif]-->
```

Kami menyajikan CSS kami yang disempurnakan dengan cara lama untuk IE8 dan yang lebih baru, sementara browser lain akan tetap menggunakan tautan media yang memenuhi syarat. Benar, IE8 masih belum tahu apa yang harus dilakukan dengan kueri media, jadi kami mungkin menyertakan Respond.js9 dari Filament, skrip ringan yang mem-parsing dan menerjemahkan dukungan kueri media lebar minimum dan lebar maksimal untuk IE 6–8.

Sekarang, kami telah bersusah payah memastikan bahwa situs kami tetap berguna, jika tidak ideal, di browser yang tidak menerima gaya kami yang disempurnakan. Dengan cara ini, kami telah memberi ruang untuk bernapas: jika kami memilih untuk tidak mengabaikan dukungan kueri media untuk versi Internet EXplorer yang sudah ketinggalan zaman, kami dapat mengubah komentar bersyarat kami untuk hanya memberikan pengalaman yang ditingkatkan ke IE8 dan yang lebih baru, atau melayani mereka lembar gaya lebar statis yang terpisah. Dukungan untuk Internet EXplorer versi lama tidak lagi menjadi masalah hitam-putih; kita tidak lagi menyudutkan diri kita sendiri. Kami mendukung versi IE yang lebih lama, tentu saja, hanya saja dengan cara yang berbeda. Kebetulan kami melakukannya dengan cara yang paling mudah bagi kami, dan dengan cara yang memastikan pengguna yang bergantung pada browser kuno tidak disajikan dengan lukisan Jackson Pollock dari situs kami. Sulit untuk membantahnya, dari sudut mana pun.

Ada satu hal lain selain dukungan IE lama, dan sayangnya hal ini mempengaruhi semua browser utama lainnya: ketika kita memenuhi syarat style sheet kita dengan atribut media, atribut tersebut tidak diterapkan, namun akan diminta. Hal ini masuk akal karena kita tidak pernah tahu apakah suatu lingkungan akan berubah. Jika monitor eksternal dicolokkan atau jendela diubah ukurannya, kami tidak ingin menunggu style sheet baru tersebut diminta. Seperti yang bisa Anda lihat dari gambar di halaman berikutnya, browser cenderung terlalu berlebihan dalam hal ini. Sayangnya, hal ini memblokir permintaan. Bahkan jika style sheet tidak mungkin diterapkan pada konteks pengguna, halaman akan tetap dicegah untuk dimuat saat style sheet diminta, diunduh, dan kemudian diabaikan oleh browser.

Kami melakukan beberapa percobaan dengan memuat style sheet yang berlaku secara asinkron dalam sebuah proyek bernama eCSSential10, namun menemukan bahwa menggunakan JavaScript untuk meminta style sheet kami berarti menghindari sejumlah optimasi tingkat browser. Secara umum, eCSSential secara kasar mencapai titik impas dengan

memuat CSS dengan cara lama lebih baik di beberapa browser dan lebih buruk di browser lain namun hal ini menimbulkan ketergantungan pada JavaScript.

Tabel 5.1 Tabel atribut media yang diminta: Menariknya, antara versi 11 dan 12, Opera memutuskan bahwa saya harus menghubungkan laptop saya ke brachiosaurus eksternal.

	IOS6, Android 4.0, Chrome 24, Firefox, YAITU 9, opera 12	Opera 11
Hanya semua	Diunduh	Diunduh
(lebar minimum: 999 piksel)	Diunduh	Diunduh
(lebar perangkat minimum: 9999 piksel)	Diunduh	Diunduh
(min-perangkat rasio piksel: 7)	Diunduh	Diunduh
Televisi	Diunduh	Diunduh
Handheld	Diunduh	Diunduh
Dinosaur	Diunduh	Tidak ada permintaan

Meskipun eCSSential tidak memberi kita kemajuan pasti yang kita harapkan, hal ini menyebabkan sejumlah bug yang dilaporkan pada browser, dan diskusi sedang berlangsung saat ini tentang bagaimana browser dapat mengunduh style sheet yang tidak dapat diterapkan secara asinkron dalam cara yang dioptimalkan, cara yang tidak menghalangi.

5.4 JAVASCRIPT

Pendekatan yang sama yang kami ambil dengan CSS juga berlaku untuk JavaScript khusus kami: kami memulai dengan serangkaian skrip dasar untuk semua orang, dan menggunakan pengujian yang sama untuk menentukan apakah mereka mendapatkan pengalaman JavaScript yang ditingkatkan: dukungan kueri media.

Muatan JavaScript awal meliputi: Enhance.js dari Filament untuk meminta skrip dan style sheet secara kondisional; Modernizr sebagai kerangka pengujian fitur kami; dan Respond.js jika kami memilih untuk memberikan pengalaman yang lebih baik kepada IE8. Skrip ini dimuat di bagian atas halaman, karena sensitif terhadap waktu (Respond.js) atau hal-hal yang harus segera kami siapkan jika kami mereferensikannya di skrip kami yang lain (Modernizr).

```
function(win, undefined) {
    var mqSupport = "matchMedia" in win && win.matchMedia( "only all"
).matches;
    if( !mqSupport && !respond.mediaQueriesSupported ) {
        return;
    }
})( this );
```

Skrip ini memeriksa apakah browser pengguna mendukung metode matchMedia (metode asli JavaScript untuk mengurai kueri media) dan kemudian, sebagai ukuran yang baik, skrip ini memastikan bahwa hanya semua pengujian yang sama yang kami gunakan di pass CSS kami. Jika metode asli tidak ada, metode tersebut akan memeriksa dukungan kueri media Respond.js. Jika Anda menargetkan versi minimum IE tertentu untuk pengalaman yang lebih baik, Respons ini. Tes js dapat ditukar untuk memeriksa keberadaan kelas bersyarat IE.

```
(function(win, undefined){
      /* This script assumes a conditional comment scheme along the
lines of the following:
      <!--[if (lt IE 8) ]> <html class="old-ie"> <![endif]-->
      <!--[if (IE 8) ]> <html class="ie8"> <![endif]-->
      <!--[if (gt IE 8)|!(IE)]><!--> <html> <!--<![endif]-->
      */
      var mqSupport = "matchMedia" in win && win.matchMedia( "only
all" ).matches,
           htmlClass = document.getElementsByTagName( "html" )[ 0
].getAttribute( "class" ),
           ie8 = htmlClass && htmlClass.indexOf( "ie8" ) > -1;
   if( !enhanced && !ie8 ){
         return;
}
})( this );
```

Masalahnya adalah, antara Enhance.js, Respond.js, build Modernizr kami, dan pengujian peningkatan baru kami, kami baru saja menambahkan empat permintaan pemblokiran ke bagian atas halaman. Kita bisa membuang semua ini ke dalam satu file, tapi hal itu mungkin akan membuat kita pusing ketika tiba waktunya untuk memperbarui salah satu perpustakaan ini.

Untuk menghindari ketidaknyamanan atau membebani pengguna dengan permintaan tambahan, kami baru-baru ini memperkenalkan kerangka kerja tugas Grunt ke dalam proses pengembangan kami. Grunt dapat diatur untuk melihat direktori dan menggabungkan file Anda setiap kali ada perubahan, artinya Anda dapat menyimpan semua perpustakaan dan JavaScript khusus Anda dalam file terpisah dan mengerjakannya seperti biasa, namun tautkan template Anda ke satu file “dist” yang dihasilkan secara otomatis. siap untuk produksi. Ini akan melakukan hal yang sama untuk style sheet Anda, memungkinkan Anda untuk membagi CSS yang disempurnakan dan mengatur lingkungan pengembangan sesuai keinginan Anda, tetapi menghasilkan satu file gabungan. Selanjutnya, Grunt akan mengecilkan semua JavaScript dan CSS Anda, menyaring kesalahan pada kode Anda, menjalankan pengujian unit Anda, atau menjalankan tugas khusus apa pun yang dapat Anda bayangkan semua dilakukan secara otomatis, melalui baris perintah. Grunt dengan cepat menjadi bagian tak terpisahkan dari proses pengembangan kami.

Dengan adanya file JavaScript awal yang digabungkan, kami sekarang memiliki kerangka kerja untuk memuat file secara kondisional sesuai kebutuhan, berdasarkan konteks

pengguna. Kami dapat memuat pustaka JavaScript, plugin, dan skrip khusus yang lebih besar secara asinkron yang berlaku di seluruh situs tanpa menunda pemuatan halaman. Jika kami memiliki skrip khusus untuk menambahkan interaksi gesek pada perangkat sentuh, kami dapat mendeteksi fitur untuk peristiwa sentuh dan menyertakan skrip tersebut, serta gaya yang sesuai, hanya jika diperlukan. Jika ada bagian unik dari situs dengan CSS atau JS yang sangat spesifik, kami menambahkan kelas ke tag body dan memuat skrip dan style sheet khusus halaman tersebut hanya ketika halaman tersebut dimuat.

Ingatlah bahwa akan ada sedikit penundaan saat memuat style sheet menggunakan metode ini. Pastikan untuk membatasi pendekatan ini pada gaya untuk komponen tertentu dalam tata letak, bukan seluruh halaman, atau Anda berisiko menampilkan sekilas konten tanpa gaya kepada pengguna.

Sedikit Bantuan dari Server

Permintaan bersyarat ini dapat bertambah dengan cepat pada proyek besar: kerangka kerja JavaScript, beberapa pustaka plugin, skrip untuk menambahkan penyempurnaan khusus perangkat seperti penyimpanan offline atau dukungan sentuh. Meskipun mereka tidak lagi mengambil risiko menunda pemuatan konten inti halaman, permintaan asinkron yang cukup masih berpotensi membuat situs terasa lamban, dan mencegah penyempurnaan kami tersedia bagi pengguna secepat yang mereka harapkan.

Untuk menyiasatinya, kami menggunakan pola penggabungan sisi server yang disebut QuickConcat, dibuat untuk bekerja dengan Enhance.js, untuk menggabungkan semua skrip kondisional dan style sheet kami ke dalam satu permintaan.

QuickConcat adalah file PHP kecil yang memotong dan mem-parsing permintaan untuk kumpulan skrip atau style sheet yang dipisahkan koma, dan merakitnya menjadi satu file untuk dikirimkan ke pengguna. Misalnya:

```
<script src="quickconcat.php?files=js/file1.js, js/file2.js,
js/file3.js"></script>
```

Atau:

```
<link href="quickconcat.php?files=css/file1.css,css/file2.css,
css/file3.css" rel="stylesheet">
```

Dengan sedikit pembersihan melalui file .htaccess (atau yang setara untuk lingkungan server Anda):

```
<script src="js/file1.js, js/file2.js, js/file3.js"></script>
<link href="css/file1.css, css/file2.css, css/file3.css"
rel="stylesheet">
```

Kami masih akan menggunakan Grunt untuk menggabungkan file JavaScript awal dan CSS global kami, karena keduanya tidak akan pernah bervariasi di sisi klien keunggulan

QuickConcat adalah permintaan asinkron kami. Daripada menulis beberapa skrip dan tag link ke dalam halaman dan mengirimkan beberapa permintaan, kita dapat menggunakan Enhance.js untuk menyiapkan daftar skrip dan style sheet yang sesuai dengan konteks pengguna dan meminta semuanya sekaligus:

```
(function(win, undefined){

   var mqSupport = "matchMedia" in win && win.matchMedia( "only all"
).matches;
   if( !mqSupport && !respond.mediaQueriesSupported ){
            return;
   }

   ejs.addFile.jsToLoad( "js/lib/jQuery.js" );
   ejs.addFile.jsToLoad( "js/lib/konami-code.js" );

   // Load custom fonts > 600px
   if( window.screen.width > 600 ){
      ejs.addFile.cssToLoad( "css/fonts.css" );
   }

   if( Modernizr.touch ) {
      ejs.addFile.jsToLoad( "js/swipe.js" );
      ejs.addFile.cssToLoad( "css/swipe.css" );
   }
   ejs.enhance();
})( this );
```

Saat `Enhance.js` dipanggil, semua file yang diantri dengan `ejs.addFile.cssToLoad` dan `ejs.addFile.jsToLoad` dikirim sebagai satu permintaan, melalui QuickConcat.

Saya biasanya menyebut QuickConcat sebagai sebuah pola karena jarang ada sesuatu yang dimasukkan ke dalam lingkungan produksi sebagaimana adanya. Biasanya ini adalah sesuatu yang akan kami serahkan kepada klien untuk diimplementasikan dalam bahasa back-end pilihan mereka. Berkat QuickConcat, meskipun kami memuat beberapa skrip dan gaya setelah halaman dimuat, kami hanya menambahkan dua permintaan: satu untuk semua skrip tambahan, dan satu lagi untuk gaya tambahan.

Gambar dan Video

Topik tentang bobot aset media memiliki tempat khusus di hati saya. Pada saat artikel ini ditulis, gambar menyumbang sekitar 64% dari rata-rata berat situs web, dan hal ini menjadi lebih buruk karena semakin banyak orang yang melapisi situs mereka dengan gambar-gambar beresolusi tinggi yang sangat besar agar sesuai dengan tampilan terbaru dengan kepadatan tinggi. Rata-rata bobot gambar situs web saja telah meningkat lebih dari 13% sejak 1 Januari tahun ini.

Anda mungkin sudah hafal dasar-dasar gambar responsif: untuk membuat gambar fleksibel, pertama-tama kami menghapus atribut lebar dan tinggi. Dengan menetapkan lebar maksimum 100% di CSS, kami mencegah gambar ini meluap dari wadah induknya. Saat wadah fleksibel kami berubah ukuran, gambar pun ikut berubah.

Tentu saja, pendekatan ini mengharuskan kita untuk menggunakan aset yang setidaknya sama besar dengan ukuran terbesar yang akan menampilkannya: jika gambar ditujukan untuk bagian tata letak yang lebarnya bisa berkisar antara 300pX hingga lebar 2000pX, Anda masih harus menyajikan gambar dengan lebar bawaan minimal 2000pX. Ini merupakan jumlah besar bandwidth dan daya pemrosesan yang terbuang pada perangkat seluler, tanpa manfaat yang dapat dirasakan oleh pengguna.

Biaya bandwidth ini berlipat empat kali lipat seiring kami memperbarui aset agar sesuai dengan tampilan dengan kepadatan tinggi. Gambar Retina tidak hanya dua kali lebih besar namun juga dua kali lebih besar di kedua dimensi: gambar Retina yang sebenarnya berukuran empat kali lebih besar. Kita tidak boleh menyajikannya kepada semua orang tanpa pandang bulu ketika sebagian besar pengguna tidak merasakan manfaat apa pun, terutama jika dikaitkan dengan biaya bandwidth gambar yang sudah tinggi sehingga perlu ditingkatkan skalanya agar sesuai dengan tata letak desktop dan tata letak seluler.

Paling banter, itu akan sangat sia-sia. Kemungkinan terburuknya, browser seluler lama mungkin melihat semua data ini menempel padanya dan menyerah sepenuhnya, sehingga laman tidak dirender. Apa pun kasusnya, kami menyebabkan kerusakan besar pada paket data pengguna, dan membebani pengguna dengan kerugian yang sangat besar.

Elemen video HTML5 membuatnya sangat mudah untuk menyesuaikan aset agar paling sesuai dengan konteks pengguna. Meskipun kami belum dapat memperhitungkan faktor spesifik seperti kecepatan koneksi, setidaknya kami dapat memastikan bahwa aset yang kami kirimkan sesuai dengan tampilan pengguna. Elemen sumber HTML5 memungkinkan kita menentukan sumber mana yang harus diterapkan berdasarkan kueri media yang sama yang kita gunakan dalam tata letak, dalam atribut media.

```
<video>
       <source src="vid-large.webm" media="(min-width: 600px)"
          type="video/webm">
<source      src="vid-large.ogg"      media="(min-width:      600px)"
type="video/ogg">
       <source src="vid-large.mp4" media="(min-width: 600px)"
          type="video/mp4">

       <source src="vid-small.webm" type="video/webm">
       <source src="vid-small.ogg" type="video/ogg">
       <source src="vid-small.mp4" type="video/mp4">
       <!-- Fallback for browsers that don’t support 'video': -->
       <a href="vid.mpg">Watch Video</a>
</video>
```

Memang benar, ini sedikit bertele-tele ketika kita perlu menentukan beberapa format untuk setiap sumber; dukungan codec masih merupakan ladang ranjau. Dalam contoh di atas, sumber video yang lebih kecil dalam format apa pun yang didukung oleh browser disajikan kepada pengguna mana pun dengan tampilan yang lebih sempit dari 600pX. Atribut ini ternyata didukung dengan baik, meskipun merupakan fitur yang kurang dikenal: sintaksis ini akan berfungsi di versi terbaru Chrome, FirefoX, Opera, Safari, Internet EXplorer, iOS, Windows Phone, BlackBerry dan Android.

Kami mengembangkan cara serupa untuk memberikan gambar yang sesuai dengan layar saat kami bekerja di situs Globe, dimulai dengan filosofi bahwa teknik ini sebaiknya dilakukan pada perangkat seluler: mulai dengan gambar berukuran dan berformat seluler, lalu tukar dengan gambar yang berukuran dan diformat untuk perangkat seluler. versi yang lebih besar tergantung pada ukuran layar pengguna.

Kuncinya adalah mendapatkan lebar layar dalam JavaScript dan menyampaikan informasi tersebut ke server tepat waktu untuk menunda permintaan sumber yang ditentukan dalam src gambar jika tidak, kami membuat dua permintaan per gambar pada layar yang lebih besar. Kami dapat mencegah hal ini dengan menghindari penggunaan img src dan memasukkan gambar sesuai kebutuhan, namun kami tidak ingin membuat ketergantungan pada JavaScript agar pengguna dapat melihat konten situs. Sebaliknya, kami akhirnya membuat peretasan kecil yang cerdas yang mengandalkan JavaScript untuk mengatur lebar layar dalam cookie yang akan dikirim bersama dengan permintaan src asli gambar, memungkinkan kami memilih sumber gambar yang paling sesuai di server. Jika ada aspek skrip yang gagal, sumber asli yang ditentukan dalam src akan diminta seperti biasa. Ini bekerja dengan baik setidaknya, untuk sementara.

Sayangnya, pendekatan ini tidak bertahan lama. Berkat semakin agresifnya pengambilan awal di versi terbaru beberapa browser desktop utama, src gambar akan diminta sebelum skrip khusus kami diterapkan menghasilkan dua permintaan untuk satu gambar yang dilihat pengguna. Yang terjadi selanjutnya adalah kisah kotor tentang elemen noscript dan tag dasar yang disuntikkan secara dinamis, document.write dan eval - pengembang Web yang setara dengan cerita api unggun yang menakutkan. Itu tidak bagus dan, yang lebih penting, tidak ada yang berhasil.

Pada saat itulah kami membentuk Grup Komunitas Gambar Responsif18, dan melibatkan lebih banyak orang. Meskipun melibatkan banyak pengembang dalam brainstorming kami, dengan cepat menjadi jelas bahwa gambar responsif bukanlah sesuatu yang dapat diselesaikan untuk selamanya dengan sedikit JavaScript yang cerdas. Jadi, kami mulai mencari ide untuk solusi asli: jika HTML5 menawarkan cara untuk menyelesaikannya, seperti apa solusinya?

Bruce Lawson awalnya mengusulkan pola markup untuk memberikan gambar yang sesuai konteks dan sejalan dengan sintaksis untuk video dan elemen sumber:

```
<picture>
     <source src="fullsize.jpg" media="(min-width: 60em)" />
```

```
    <source src="small.jpg" />
    <!-- Fallback for browsers that don’t support 'video': -->
    <img src="fallback.jpg" alt="..." />
</picture>
```

Sekitar waktu yang sama ketika kami mempresentasikan pola ini kepada WHATWG, mereka mengajukan ide mereka sendiri untuk sarana berbasis markup dalam menyajikan sumber gambar yang sesuai konteks: atribut srcset.

```
<img src="fallback.jpg" srcset="small.jpg 320w 1x, small-hd.jpg 320w
2x, medium.jpg 640w 1x, medium-hd.jpg 640w 2x, large.jpg 1x, large-
hd.jpg
2x">
```

Meskipun sintaksis atribut srcset hampir tidak dapat dipahami, ia menangani salah satu bagian persamaan dengan cara yang sangat efisien: peralihan resolusi, menggunakan nilai 1x, 2x, dan seterusnya. Selain itu, ia juga menangani pertanyaan tentang penyelesaian di luar pertanyaan media, yang segera kami sadari akan memberikan manfaat yang lebih besar bagi pengguna.

Pertanyaan media adalah hal yang mutlak, setidaknya di atas kertas. Sulit membayangkan keadaan di mana pengguna mungkin ingin memilih tata letak yang tidak sesuai untuk tampilan mereka. Demikian pula, dibutuhkan banyak imajinasi untuk menghasilkan skenario di mana pengguna lebih memilih gambar yang terlalu kecil atau terlalu besar untuk tata letak yang mereka lihat. Resolusi gambar, di sisi lain, adalah cerita yang berbeda: jika saya menggunakan Retina MacBook tetapi saya tertambat ke koneksi 3G yang lemah, saya mungkin lebih memilih untuk tidak menggunakan gambar beresolusi tinggi yang besar.

Tidak seperti kueri media, atribut srcset didokumentasikan sebagai serangkaian saran. Saat membuat keputusan aset berdasarkan ukuran layar atau kepadatan piksel pengguna melalui kueri media, kami dapat memastikan bahwa kami tidak pernah memberikan aset yang lebih besar daripada yang dapat memberikan manfaat bagi pengguna. Yang tidak dapat kami lakukan adalah menggunakan informasi tersebut untuk membuat asumsi tentang kondisi bandwidth atau preferensi pengguna.

Dengan bertindak sebagai saran, srcset akan memungkinkan browser untuk memperkenalkan pengaturan pengguna seperti “selalu beri saya gambar beresolusi tinggi”, “selalu beri saya gambar beresolusi rendah,” atau “berikan saya gambar beresolusi tinggi jika bandwidth memungkinkan.” Browser memiliki informasi bandwidth tersebut, dan di situlah keputusan harus dibuat bukan untuk pengguna, namun bersama mereka.

Pada cuplikan berikut, kami masih mengandalkan atribut media untuk memilih elemen sumber yang sesuai. Hal ini masuk akal kami akan menentukan titik henti sementara berdasarkan kombinasi beberapa faktor: kueri media tata letak kami; berat gambar; dan

melakukan cropping dan zoom secara bergantian sehingga kami dapat merepresentasikan fokus gambar dengan lebih baik pada tampilan yang lebih kecil.

```
<picture>
     <source media="(min-width: 40em)" src="big.jpg">
     <source src="small.jpg">
     <img src="fallback.jpg" alt="...">
</picture>
```

Setelah kami menetapkan elemen sumber, kami kemudian menyajikan opsi resolusi. Di sinilah kami menggunakan sebagian dari atribut srcset yang diusulkan WHATWG untuk menentukan sumber resolusi mana yang paling tepat.

```
<picture>
     <source media="(min-width: 40em)" srcset="big-sd.jpg 1x,
big-hd.jpg 2x">
     <source srcset="small-sd.jpg 1x, small-hd.jpg 2x">
     <img src="small-sd.jpg" alt="">
</picture>
```

Jika Anda tidak perlu menentukan beberapa resolusi untuk sumber gambar tertentu, namun ingin menentukan sumber berdasarkan tata letaknya, Anda dapat menggunakan elemen gambar yang tidak bergantung pada srcset:

```
<picture><source media="(min-width: 30em)" src="big.jpg">
         <source src="small.jpg">
         <img src="small.jpg" alt=""></picture>
```

Jika Anda hanya memerlukan aspek pengalihan resolusi, Anda dapat menggunakan srcset yang tidak bergantung pada elemen gambar:

```
<img src="image-sd.jpg" srcset="image-hd.jpg 2x" alt="...">
```

Kedua usulan tersebut tidak hanya dapat hidup berdampingan secara harmonis dan saling melengkapi, namun keduanya juga dapat menyelesaikan permasalahannya sendiri-sendiri. Kami mengusulkan versi elemen gambar ini ke HTML WG, dan beberapa bulan yang lalu kami mencapai First Public Working Draft, yang berarti sudah waktunya bagi para pelaksana untuk mulai menggali lebih dalam dan mengajukan pertanyaan kepada kami. Meskipun kami mengalami kemajuan yang stabil, mungkin perlu waktu sebelum kami dapat menggunakan salah satu dari pola markup yang tepat ini dalam pekerjaan kami.

Namun, semua ini tidak memberikan banyak manfaat bagi kita saat ini maksud saya, ini mungkin memakan waktu cukup lama, dan ada pekerjaan yang harus kita selesaikan.

Untungnya, kita dapat mulai menggunakan markup ini hari ini: Scott Jehl membuat polyfill untuk gambar saat kami menulis spesifikasinya. Picturefill mengemulasikan perilaku yang diusulkan elemen gambar menggunakan div dan atribut data semua markup yang memenuhi standar, dengan semua perilaku yang kita inginkan dari gambar.

```
<div data-picture>
        <div data-source data-media="( min-width: 30em )" data-
src="big.jpg">
        <div data-source data-src="small.jpg">
        <noscript>
         <img src="fallback.jpg" alt="...">
        </noscript>
</picture>
```

Jika JavaScript tidak didukung, pengguna akan menerima elemen img standar. Jika JavaScript tersedia, Picturefill akan menguraikan atribut yang ditentukan pada elemen sumber data, menentukan mana yang paling sesuai dengan tampilan pengguna, dan memasukkan tag img dengan sumber yang sesuai ke dalam halaman. Kami telah menggunakan Picturefill dalam pekerjaan klien kami di Filament Group selama hampir satu tahun, dan kami sangat beruntung dengan itu. Manfaat dari upaya standar yang dipimpin oleh pengembang adalah kami dapat memulai lebih awal dalam melakukan polyfill pada markup baru: lusinan polyfill untuk gambar dan srcset sudah tersedia di GitHub.

Di SouthStreet

Filament Group telah menerapkan semua pelajaran yang telah kami pelajari tentang mengoptimalkan pengiriman HTML, CSS, JavaScript, dan gambar ke dalam proyek yang kami sebut SouthStreet, yang diberi nama sesuai dengan lokasi kantor FG.

SouthStreet memberi Anda seperangkat alat yang dapat Anda gunakan untuk memastikan bahwa perangkat mendapatkan jumlah kode seefisien mungkin, sambil tetap mempertahankan aksesibilitas dan dukungan yang luas. Kami terus menyempurnakan SouthStreet seiring dengan munculnya pendekatan dan teknik baru, dan semua proyek yang disebutkan dalam bab ini sepenuhnya bersifat open source: umpan balik, saran, dan ide-ide baru selalu diterima.

Pada hari peluncuran BostonGlobe.com, kami membuka situs tersebut pada sejumlah perangkat yang belum pernah kami uji dan tentunya tidak direncanakan sebelumnya: Amazon Kindle generasi pertama, Nintendo DS, browser bawaan Playstation 3, dan bahkan Apple Newton. Kami tidak pernah dihadapkan pada layar kosong: apa pun konteksnya, kami dapat menggunakan situs web sebaik mungkin yang dimungkinkan oleh perangkat.

Setiap kali perangkat baru muncul di lab pengujian jQuery Mobile, kami melihat situs Globe dan sejauh ini, kami telah menghasilkan ribuan perangkat. Tidak ada email panik tentang memperbarui string UA; tidak perlu khawatir tentang fitur baru yang tidak cocok, atau ukuran tampilan yang tidak direncanakan. Tentu saja ada peningkatan baru yang dapat kami lakukan seiring dengan peluncuran fitur browser dan API baru, namun fondasi kami kuat dan

kami tidak pernah dibatasi oleh hal tersebut. Ia bekerja di mana saja, untuk semua orang. Dengan mengikuti beberapa prinsip dalam melayani aset secara bertanggung jawab, Anda dapat melakukan hal yang sama.

Membangun situs web adalah bisnis yang rumit dan tidak mudah. Itulah sifat permainannya, bukan kesalahan alat atau teknik apa pun yang kami gunakan. Bagian tersulit dalam pengembangan Web adalah penyederhanaan: menghilangkan hal-hal yang tidak penting. Desain Web yang responsif dapat memastikan bahwa kita tidak menghalangi fleksibilitas yang melekat pada Web; peningkatan progresif dapat memastikan bahwa kita tidak menghalangi kegunaan yang melekat pada Web. Halaman pertama Web, hingga hari ini, berfungsi untuk pengguna dalam konteks penjelajahan apa pun.

Kita tidak bisa berharap untuk selalu melakukan segalanya dengan benar; kami masih akan membengkokkan satu atau dua paku di sana-sini. Alat kami juga tidak sempurna. Namun kita bisa berbuat lebih baik kita selalu bisa berbuat lebih baik dan kita bisa menggunakan alat yang kita punya untuk membangun hal-hal menakjubkan.

BAB 6
MENEMUKAN DAN MEMPERBAIKI MASALAH RENDERING WEB SELULER

Percakapan tentang kinerja mulai berubah. Tidak lagi penting hanya untuk mempertimbangkan seberapa cepat sebuah situs dimuat tetapi juga seberapa cepat situs tersebut dirender dan dijalankan hal ini sangat penting pada perangkat seluler di mana kinerja halaman Web sering dibandingkan dengan aplikasi asli. Browser saat ini dapat berjalan dengan kecepatan refresh perangkat kita dan ketika berjalan, halaman kita terasa jauh lebih lancar. Mereka bermentega, renyah, dan enak untuk digunakan. Jika tidak, dan pengguna melihat gangguan visual di halaman mereka, mereka tidak menyukainya. Ada bau kinerja dari sesuatu yang salah dengan halaman dan itu adalah sesuatu yang perlu kami perbaiki.

Hal ini dapat menjadi tantangan karena perangkat seluler kurang bertenaga jika dibandingkan dengan sistem desktop. Merender halaman di layar membutuhkan waktu lebih lama; memuat sumber daya jaringan membutuhkan waktu lebih lama; decoding gambar membutuhkan waktu lebih lama; dan mengeksekusi skrip membutuhkan waktu lebih lama. Performa di perangkat seluler hampir tidak pernah setara dengan performa di desktop. Dengan CPU yang lebih lambat dan GPU yang bertenaga lebih rendah, platform seluler saat ini lima kali lebih lambat dibandingkan desktop. Meskipun demikian, grafik dan JavaScript seluler menjadi lebih baik pada perangkat tersebut dan 30 frame per detik (fps) dikatakan dapat dicapai untuk sejumlah kasus penggunaan.

Meskipun kinerja jaringan penting dan eksekusi JavaScript biasanya cepat, banyak yang mendapati bahwa rendering (melukis piksel ke layar) adalah hambatannya. Situs-situs besar, termasuk Facebook, Flickr dan Pinterest, mulai lebih memperhatikan sisi kinerja ini. Mereka menemukan bahwa hal ini tidak hanya memengaruhi pengalaman pengguna tetapi juga keterlibatan pengguna. Pengukuran adalah bagian terpenting dalam pembuatan profil kinerja dan, jika memungkinkan, selalu periksa situs dan aplikasi Anda menggunakan alat di browser lain untuk memeriksa ulang apakah perlambatan yang Anda alami terjadi pada browser tertentu.

Tiga Pilar Kinerja

Jadi, kami sudah mengatakan bahwa performa rendering itu penting, tapi apa hubungannya dengan performa keseluruhan? Tiga faktor kunci kinerja di Web adalah: jaringan, komputasi, dan render. Mari kita tinjau secara singkat.

6.1 JARINGAN

Selalu perhatikan jumlah permintaan jaringan yang dibuat situs Anda. Seringkali, setiap file akan diminta secara terpisah dan permintaan ini mungkin memiliki latensi yang cukup besar; Artinya, server akan membutuhkan waktu lama untuk menerima dan memproses permintaan sebuah halaman. Di perangkat seluler, mereka juga akan membuat radio tetap

hidup di perangkat Anda, yang merupakan pengurasan daya terbesar setelah layar Anda. Menahan permintaan tersebut berarti kami dapat meminimalkan berapa lama radio tetap menyala. Perlu diingat juga bahwa bandwidth dan latensi adalah hal yang berbeda sehingga meskipun Anda menggunakan koneksi 3G atau 4G, latensi Anda mungkin tidak meningkat. Inilah salah satu alasan mengapa praktik terbaik seperti menggabungkan skrip, memasukkan CSS, dan menggunakan sprite gambar sangat penting.

Sebagian besar lalu lintas Web adalah gambar lebih dari setengahnya menurut Arsip HTTP. Di banyak belahan dunia, pengguna mempunyai batas data tetap di perangkat seluler, artinya jika batas ini (misalnya 1GB per bulan) terlampaui, pelanggan harus membayar lebih. Inilah salah satu alasan pentingnya gambar dioptimalkan semaksimal mungkin. Saat ini, format baru seperti WebP menawarkan penghematan ukuran file yang cukup besar dibandingkan dengan kualitas setara JPEG atau PNG. Saya mengatakan kualitas karena Anda biasanya dapat mengalahkan codec jika Anda menurunkan kualitasnya dalam format lain. Bagi mereka yang cukup beruntung untuk menjalankan server mereka sendiri, beberapa rekan saya di Google merekomendasikan untuk mencoba mod_pagespeed sebuah alat yang secara otomatis dapat mengoptimalkan situs Anda, menangani minifikasi dan optimasi gambar tanpa usaha apa pun. Saya juga dengan senang hati merekomendasikan ImageOptim dan JPEGMini untuk pengoptimalan gambar.

Menghitung

Kami menyebut pemrosesan JavaScript sebagai “komputasi”. Semuanya berjalan di dalam mesin khusus di browser (misalnya V8 di Chrome, JavaScriptCore di Safari, OdinMonkey di FirefoX) dan dalam banyak kasus, mesin ini sangat cepat. Salah satu alasan mengapa mereka begitu cepat adalah karena mesin mengawasi kode Anda dan menukarnya dengan kode tingkat rendah yang dioptimalkan jika memungkinkan.

Pengembang JavaScript sering khawatir tentang kebocoran memori yang disebabkan oleh kode mereka. Karena kita tidak menangani sendiri penyimpanan dan pelepasan memori dan menyerahkannya kepada pemulung, kita harus berhati-hati untuk tidak melakukan hal-hal konyol seperti meninggalkan referensi ke objek yang tidak lagi kita perlukan. Penggunaan memori dari semua objek gantung ini akan bertambah seiring berjalannya waktu dan itulah yang pada dasarnya menyebabkan kebocoran memori pengumpul sampah tidak dapat melepaskan sesuatu karena mengira Anda mungkin masih memerlukannya. Alat pengembang browser di Chrome, Opera, dan FirefoX dapat menunjukkan tempat pengumpulan sampah terjadi sehingga Anda dapat mengetahui di mana dan mengapa Anda menghasilkan sampah dan berupaya mengatasinya.

Hal terakhir yang perlu diingat adalah sesuatu yang kami sebut deoptimisasi. Hal ini terjadi ketika beberapa desain yang Anda buat dalam kode menyebabkan mesin harus keluar dari jalur yang dioptimalkan untuk kode yang lebih lambat. Ada banyak sekali alasan mengapa hal ini bisa terjadi dan bervariasi dari satu mesin ke mesin lainnya. Di Chrome, Anda bisa mendapatkan sedikit manfaat menggunakan versi V8 mandiri yang disebut d8. Ini dapat memberi tahu Anda JavaScript apa yang sedang dinonaktifkan, sehingga memberi Anda kesempatan untuk mempertimbangkan kembali cara Anda menulis beberapa kode.

Memberikan

Kinerja rendering baru-baru ini menjadi sorotan banyak pengembang Web dan kami akan mencurahkan sisa bab ini untuk memahaminya. Setiap halaman Anda berisi pohon DOM (mewakili konten dan strukturnya). Melukis DOM menjadi piksel di layar bisa menjadi salah satu operasi termahal dalam siklus hidup laman Anda. Upaya ekstra apa pun yang dilakukan saat pengguna berinteraksi dengan halaman Anda dapat mengakibatkan perlambatan visual. Banyak hal yang dapat memicu hal ini menggulir, memasukkan konten baru ke dalam halaman, perubahan tata letak (perubahan apa pun yang mengubah tata letak halaman Anda), berinteraksi dengan UI hampir semua perubahan yang perlu dilakukan.

Seperti yang akan segera kita bahas lebih mendalam, melukis bukan hanya tentang interaksi pengguna. Ini juga mencakup upaya yang harus dilakukan browser untuk mendekode gambar (jika Anda memberikannya dalam format JPEG, ini harus didekodekan menjadi Bitmap), serta mengubah ukurannya. Jika Anda memberi browser gambar lebar 1.024pX yang Anda ubah ukurannya menjadi 360pX menggunakan CSS, itu akan menjadi jauh lebih efisien daripada sekadar menyediakan gambar lebar 360pX yang telah diubah ukurannya. DevTools Chrome dapat memberi Anda lebih banyak wawasan tentang waktu dekode gambar di Timeline.

Merender Jank dan Mencapai 60fps

Mata manusia merasakan aliran informasi yang terus menerus. Ia tidak secara alami melihat gerak sebagai serangkaian bingkai. Dalam dunia animasi, film, dan game, penggunaan serangkaian bingkai foto untuk menyimulasikan gerakan akan menciptakan beberapa artefak persepsi yang menarik terutama jika bingkai diputar terlalu lambat. Ketika kecepatan bingkai bervariasi, mata kita merasakan gerakan yang tersentak-sentak dan bergetar, bukan kehalusan, dan apa yang kita lihat tampak berkedip-kedip. Untuk pengalaman pengguna yang optimal di Web, animasi harus halus, pengguliran harus mulus, dan halaman Anda harus mengandung sedikit atau tanpa jank, sebuah istilah yang berarti gangguan pada kecepatan bingkai konsisten yang muncul secara visual.

Anda mungkin pernah mengalami jank sebelumnya. Pernahkah Anda mengunjungi situs yang penelusurannya terasa sangat lamban? Atau mungkin ada banyak animasi kompleks atau UI baru yang diperkenalkan secara dinamis sehingga menghalangi Anda untuk melakukan apa pun. Pengalaman seperti inilah yang ingin kami hindari.

Dalam kehidupan sebuah halaman Web, kita biasanya melakukan tiga tugas inti: mengambil sumber daya; penguraian dan tokenisasi sumber daya ini (HTML/CSS/JS); dan akhirnya menggambar sesuatu untuk disaring. Selama interaksi pengguna dengan suatu halaman, hanya sebagian saja yang akan diubah. Misalnya, mereka mungkin melakukan tindakan mengubah visibilitas atau menambahkan garis besar ke suatu elemen. Proses sebenarnya memperbarui layar dikenal sebagai cat.

Perubahan pada halaman Anda (misalnya ketika JavaScript memodifikasi gaya CSS) membuat persegi panjang yang Anda lihat di layar tidak valid dan menyebabkan browser Anda menganggapnya rusak.

Pengecatan adalah operasi yang mahal tetapi juga sulit untuk dihindari. Anda selalu perlu menggambar sesuatu ke layar. Kuncinya adalah memastikan area yang Anda pengecatan (atau pengecatan ulang) sekecil mungkin, jika tidak, Anda mungkin akan mengalami jank. Di Chrome, kami mengawasi apa yang perlu diubah di layar, membuat kotak kerusakan dengan koordinat bagian halaman yang memerlukan pengecatan ulang. Kami menyimpan persegi panjang lama, sebelum perubahan Anda, sebagai bitmap dan kemudian hanya mengecat delta antara persegi panjang baru dan yang lama. Jika Anda melihat ada area tertentu pada halaman yang memerlukan banyak pengecatan ulang, ada gunanya menyelidiki apa yang dapat dilakukan untuk mengurangi biaya pengecatan.

Di Web, frame rate yang rendah (dan pengalaman yang tidak stabil) berarti bahwa setiap frame yang dirender oleh browser dapat dilihat oleh mata manusia. Memberikan pengalaman bebas jank kepada pengguna sering kali berarti menawarkan pengalaman yang dapat berjalan pada 60fps di situs dan aplikasi Web, bukan hanya game dan animasi. Pada 60fps, Anda memiliki waktu 16,66 md untuk menyelesaikan segalanya agar Chrome dapat menampilkan bingkai halaman Anda yaitu pemrosesan logika, pengecatan, tata letak, pengodean gambar, pengomposisian semuanya. Setelah Anda memperhitungkan berbagai proses browser, angka ini akan terlihat seperti 8–10 md dan menghabiskan anggaran ini dapat berarti pengguna Anda lebih cenderung melihat jank di halaman mereka.

Apa yang ajaib dari angka 60? Kami mengatakan 60fps karena ini cocok dengan kecepatan refresh perangkat yang kami gunakan saat ini. Animasi harus sesuai dengan kecepatan refresh perangkat yang digunakan. Ponsel biasanya memiliki frekuensi 55–60Hz, laptop 58–60Hz (walaupun 50Hz dalam mode daya rendah), dan sebagian besar monitor memiliki frekuensi 50–62Hz.

Untuk mencapai 60fps, terkadang kami perlu melampaui JavaScript sebagai satu-satunya penghambat kinerja laman kami dan menghabiskan lebih banyak waktu untuk menyelidiki masalah cat dan tata letak. Beberapa penyebab utama jank pada situs dan aplikasi antara lain:

- Waktu pengecatan yang berat untuk elemen DOM Anda.
- Pengubahan ukuran gambar yang tidak perlu, karena Anda belum melakukan pra-skala ke ukuran yang diperlukan.
- Decode gambar panjang (misalnya decoding PNG atau JPEG).
- Pembatalan lapisan yang tidak terduga.
- Pengumpul sampah berjalan.
- Permintaan jaringan (misalnya memproses XHR).
- Animasi berat atau pemrosesan data.
- Input handler dengan JavaScript dalam jumlah besar. Salah satu kesalahan umum adalah menambahkan banyak JavaScript untuk mengatur ulang halaman dalam pengendali onscroll yang memengaruhi waktu pengecatan.

Animasi Lebih Cepat, Minta Bingkai Animasi

setInterval dan setTimeout secara teratur digunakan untuk membuat animasi setiap 16 md. Hal ini mempunyai tantangan tersendiri, namun ada dua hal yang perlu diperhatikan:

kecepatan refresh berbeda dari satu perangkat ke perangkat lainnya (misalnya, kecepatan refresh di ponsel Anda belum tentu sama dengan kecepatan refresh di desktop Anda); dan resolusi pengatur waktu dari JavaScript hanya dalam hitungan beberapa milidetik.

Agar penyegaran layar berikutnya terjadi, Anda memerlukan bingkai animasi lengkap dengan semua JavaScript, manipulasi DOM, pengecatan, dan tata letak agar siap. Akan sangat sulit untuk menyelesaikan bingkai animasi sebelum penyegaran berikutnya ketika Anda bekerja dengan resolusi pengatur waktu rendah dan variasi kecepatan penyegaran layar membuat hal ini hampir tidak mungkin dilakukan dengan pengatur waktu tetap. Terlepas dari berapa interval pengatur waktu Anda, pada akhirnya Anda akan keluar dari jendela pengaturan waktu untuk sebuah frame dan akan menghapusnya, yang berarti pengguna mungkin melihat penurunan visual dalam kehalusan. Anda juga akan melakukan banyak pekerjaan untuk menghasilkan bingkai yang tidak pernah ditampilkan, sehingga membuang-buang baterai dan waktu CPU yang penting.

Anda mungkin telah memperhatikan bahwa sejauh ini kita selalu memperhatikan kecepatan bingkai ketika berbicara tentang kinerja rendering varians berpotensi menjadi masalah yang lebih besar karena, seperti yang saya sebutkan, mata kita memperhatikan gangguan kecil dalam gerakan dan hal ini cenderung terjadi bersamaan dengan animasi dengan waktu yang buruk. Cara terbaik untuk mendapatkan bingkai animasi dengan waktu yang akurat adalah dengan menggunakan API requestAnimationFrame, yang saat ini didukung di semua browser modern.

Saat Anda menggunakannya, Anda meminta browser untuk memberi Anda bingkai animasi dan panggilan balik Anda dipanggil saat akan menghasilkan bingkai baru. Hal ini terjadi terlepas dari kecepatan refresh perangkat dan ini luar biasa.

Tom Wiltzius dan Paul Lewis telah menulis di HTML5Rocks tentang pengoptimalan animasi dengan requestAnimationFrame dengan lebih ringkas daripada yang saya bisa, dan mereka juga sebelumnya telah menunjukkan beberapa hal bagus lainnya yang diberikan kepada Anda yang cukup relevan dengan seluler. Misalnya, animasi di tab latar belakang dijeda, yang dapat menghemat masa pakai baterai Anda, dan jika perangkat tidak dapat merender pada kecepatan refresh layar, perangkat sebenarnya dapat membatasi animasi dan hanya menghasilkan panggilan balik yang lebih jarang (mis. 30 kali satu detik, bukan 60). Meskipun ini mungkin berarti Anda mengurangi separuh kecepatan bingkai, itu berarti animasi Anda tetap konsisten. Kecepatan bingkai yang lebih rendah secara konstan lebih baik daripada variasi 60Hz yang menurunkan beberapa bingkainya.

6.2 ANIMASI CSS

Kita telah membicarakan tentang requestAnimationFrame, tetapi tahukah Anda bahwa yang lebih efisien daripada animasi JavaScript yang lebih ringan di callback Anda bukanlah JavaScript sama sekali? Tidak ada solusi sempurna untuk menghindari interupsi dalam callback requestAnimationFrame, namun Anda bisa mendapatkan keuntungan menggunakan animasi CSS untuk menghilangkan kebutuhan akan hal tersebut. Di browser

seperti Opera Mobile dan Chrome untuk Android, animasi CSS dapat dijalankan oleh browser saat JavaScript berjalan berkat multi-threading.

Animasi CSS memaparkan beberapa teknik berbeda untuk menganimasikan. Ini termasuk: transisi, yang secara otomatis bernyawa jika properti CSS tertentu berubah; transforms, yang menyediakan metode untuk mengubah cara suatu elemen ditampilkan di layar (misalnya penskalaan, terjemahan); dan animasi @keyframebased untuk mendefinisikan animasi yang lebih kompleks yang berubah seiring waktu. Anda harus menggunakan animasi atau transisi keyframe CSS jika memungkinkan, karena animasi atau transisi tersebut sangat dioptimalkan (sering kali dipercepat dengan GPU) dan kinerjanya hampir secara umum bagus.

Seperti yang direkomendasikan oleh Paul Irish sebelumnya, jika Anda benar-benar perlu menggunakan animasi berbasis JavaScript, gunakan requestAnimationFrame. set-Timeout dan setInterval harus dihindari seperti wabah. Transformasi 2-D biasanya memberikan pengalaman yang lebih halus dibandingkan mengandalkan posisi absolut dan akan menghasilkan waktu pengecatan yang lebih cepat dan keseluruhan animasi yang lebih halus.

Akselerasi Perangkat Keras (GPU).

Hal berikutnya yang akan kita lihat adalah akselerasi GPU. Di masa lalu, browser sangat bergantung pada CPU untuk merender halaman. Ini melibatkan dua hal: pertama, melukis elemen menjadi sekumpulan tekstur, yang disebut lapisan; dan kedua, menggabungkan semua lapisan tersebut menjadi gambar akhir yang terlihat di layar. Namun, selama beberapa tahun terakhir, kami menemukan bahwa melibatkan GPU dalam proses pengomposisian dapat menghasilkan percepatan yang signifikan. Premisnya adalah, meskipun tekstur masih dicat di CPU, tekstur tersebut dapat diunggah ke GPU untuk dikomposisi.

Dengan asumsi bahwa semua yang kita lakukan pada frame masa depan adalah memindahkan elemen (menggunakan transisi atau animasi CSS) atau mengubah opacitynya, kita cukup memberikan perubahan ini ke GPU dan sisanya akan ditangani. Kami pada dasarnya menghindari keharusan memberikan grafis baru kepada GPU; sebaliknya, kami hanya memintanya untuk memindahkan yang sudah ada. Ini adalah sesuatu yang dilakukan GPU dengan sangat cepat, sehingga meningkatkan kinerja secara keseluruhan.

Tidak ada jaminan bahwa pengomposisian perangkat keras ini akan tersedia dan diaktifkan pada platform tertentu, namun jika tersedia saat pertama kali Anda menggunakan, katakanlah, transformasi 3-D pada suatu elemen, maka pengomposisian tersebut akan diaktifkan di Chrome. Saat ini, versi terbaru FirefoX, Safari, IE9+, dan Opera versi terbaru semuanya juga dikirimkan dengan akselerasi perangkat keras. Banyak pengembang menggunakan peretasan TranslateZ untuk melakukan hal itu. Efek samping lain dari penggunaan peretasan ini adalah elemen yang dimaksud mendapatkan lapisannya sendiri, yang mungkin sesuai atau tidak Anda inginkan. Akan sangat berguna untuk mengisolasi suatu elemen secara efektif sehingga tidak mempengaruhi elemen lain saat dicat ulang. Perlu diingat bahwa mengunggah tekstur ini dari memori sistem ke memori video belum tentu cepat. Semakin banyak lapisan yang Anda miliki, semakin banyak tekstur yang perlu diunggah dan semakin banyak lapisan yang perlu dikelola, jadi sebaiknya jangan berlebihan.

Berhati-hatilah saat mempromosikan lapisan untuk seluler secara manual karena dapat dengan mudah merugikan diri Anda sendiri. Jangan menerapkannya pada semua hal seolah-olah hal ini dapat meningkatkan kinerja Anda di desktop, biaya untuk melakukan hal ini tidak akan sama dengan di perangkat seluler tempat Anda bekerja dengan GPU yang lebih terbatas.

Menghindari Kompleksitas yang Tidak Perlu

Cara terbaik untuk menghindari masalah kinerja rendering adalah dengan menjaga segala sesuatunya tetap sederhana. Saran ini sangat penting terutama pada perangkat seluler. Satu kesalahan yang sering dilakukan pengembang saat mengembangkan Web adalah memilih untuk menciptakan pengalaman visual yang kompleks (seperti efek paralaks).

Ini melibatkan pembuatan pembaruan visual pada halaman saat Anda mendapatkan acara gulir. Masalah besarnya di sini adalah peristiwa gulir tidak diatur waktunya untuk pembaruan visual browser (yaitu dalam panggilan balik requestAnimationFrame). Dengan demikian, Anda berisiko membuat banyak pembaruan dalam satu bingkai render yang dapat menimbulkan jank pada desktop dan sangat memperlambat kinerja di perangkat seluler.

Sekarang, jika pembaruan yang Anda lakukan sangat mahal (hal ini dapat terjadi pada animasi yang kaya visual dan situs paralaks), mungkin ada banyak area yang memerlukan pengecatan dan pengomposisian. Melakukan ini lebih dari yang Anda perlukan adalah ide yang buruk. Dalam kasus pengguliran, Anda dapat mengatasinya dengan membatalkan pantulan peristiwa pengguliran Anda. Hal ini dilakukan dengan menyimpan nilai gulir terakhir yang diketahui dalam sebuah variabel setiap kali Anda mendapatkan peristiwa gulir dan kemudian membuat pembaruan visual dalam requestAnimationFrame menggunakan nilai terakhir yang diketahui. Ini akan meminimalkan kerusakan tata letak.

Hal ini memungkinkan browser menjadwalkan pembaruan visual pada waktu yang tepat dan tidak melakukan pekerjaan lebih dari yang diperlukan di setiap frame. Untuk saran lebih lanjut tentang mengoptimalkan pengguliran dan paralaks, pastikan untuk membaca artikel Paul Lewis.

Mendiagnosis Waktu Cat Lambat

Seperti yang telah kita bahas, browser harus melakukan banyak pekerjaan untuk menampilkan sesuatu ke layar. Apa pun yang Anda lakukan untuk meningkatkan kompleksitas tugas tersebut (seperti memaksa seluruh tata letak halaman dihitung ulang) berpotensi menimbulkan jank pada halaman Anda. Anda ingin menghindari ini. Jadi, mari kita bahas beberapa alat yang dapat membantu Anda mengukur perlambatan ini.

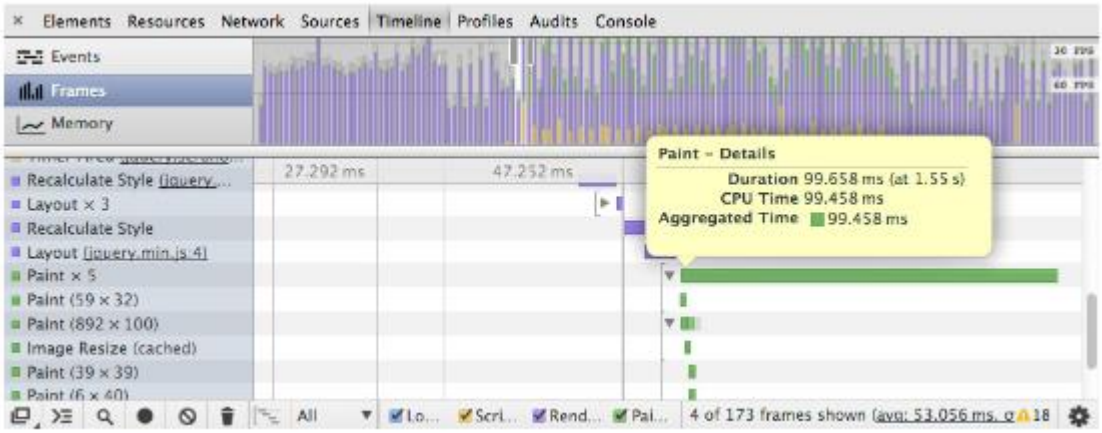
Catatan: Pada saat penulisan ini, Opera menggunakan alat pengembang front-end yang sama dengan Chrome DevTools, sehingga banyak alat yang disebutkan dalam bab ini juga akan tersedia di sana. Hal ini mungkin berubah seiring berjalannya waktu.

6.3 ALAT KINERJA DEVTOOLS

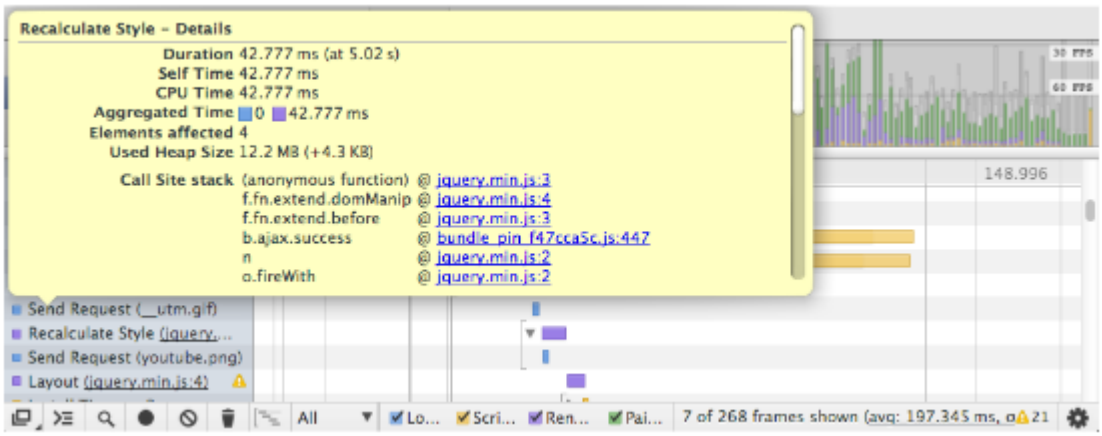
Di Chrome, panel Timeline DevTools memberikan ikhtisar tentang waktu yang dihabiskan untuk memuat aplikasi Web Anda, seperti berapa lama waktu yang dibutuhkan untuk memproses peristiwa DOM, merender tata letak halaman, atau melukis elemen ke

layar. Hal ini memungkinkan Anda menelusuri tiga aspek terpisah yang dapat membantu Anda mengetahui mengapa aplikasi Anda lambat: peristiwa; bingkai; dan penggunaan memori sebenarnya. Saat ini, kami tertarik pada mode bingkai, yang memberi Anda wawasan tentang tugas yang harus dilakukan browser untuk menghasilkan satu bingkai (pembaruan) aplikasi Anda untuk presentasi di layar.

Timeline tidak akan menampilkan data apa pun secara default tetapi Anda dapat memulai sesi perekaman dengan membuka aplikasi Anda dan mengklik lingkaran abu-abu di bagian bawah panel anda juga dapat menggunakan pintasan Command / Control+E . Tombol rekam ini akan berubah dari abu-abu menjadi merah dan Timeline akan mulai mencatat timeline halaman Anda. Selesaikan beberapa tindakan di dalam aplikasi Anda (atau yang disarankan, seperti menggulir) dan setelah beberapa detik, klik tombol lagi untuk berhenti merekam.

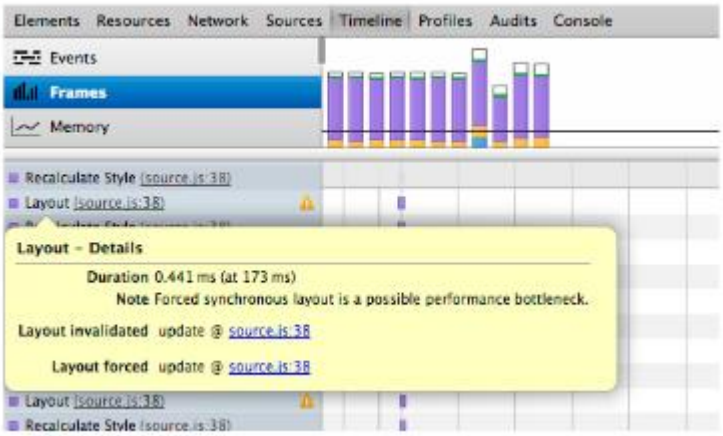


Gambar 6.1 Garis Waktu Chrome DevTools, memperbesar kumpulan rekaman dengan cat yang cukup tebal.



Gambar 6.2 Mengarahkan kursor ke atas rekaman akan menampilkan keterangan alat yang diperluas dengan detail tentang waktu yang dibutuhkan untuk menyelesaikannya. Ini memiliki begitu banyak informasi berguna di sana, jadi perhatikanlah mereka, terutama tumpukan panggilan.

Tampilan ringkasan (di bagian atas Timeline) menampilkan bilah horizontal yang mewakili jaringan dan penguraian HTML (biru), JavaScript (kuning), penghitungan ulang gaya dan tata letak (ungu), serta peristiwa pengecatan dan pengomposisian (hijau) untuk laman Anda. Pengecatan ulang adalah peristiwa browser yang dipicu oleh respons terhadap perubahan visual seperti pengubahan ukuran jendela atau pengguliran. Penghitungan ulang terjadi ketika properti CSS diubah, sedangkan peristiwa tata letak (atau perubahan posisi) disebabkan oleh perubahan posisi elemen.



Gambar 6.3 Timeline juga mengidentifikasi kapan aplikasi Anda menyebabkan tata letak asinkron yang dipaksakan dan menandai rekaman ini dengan ikon peringatan berwarna kuning.

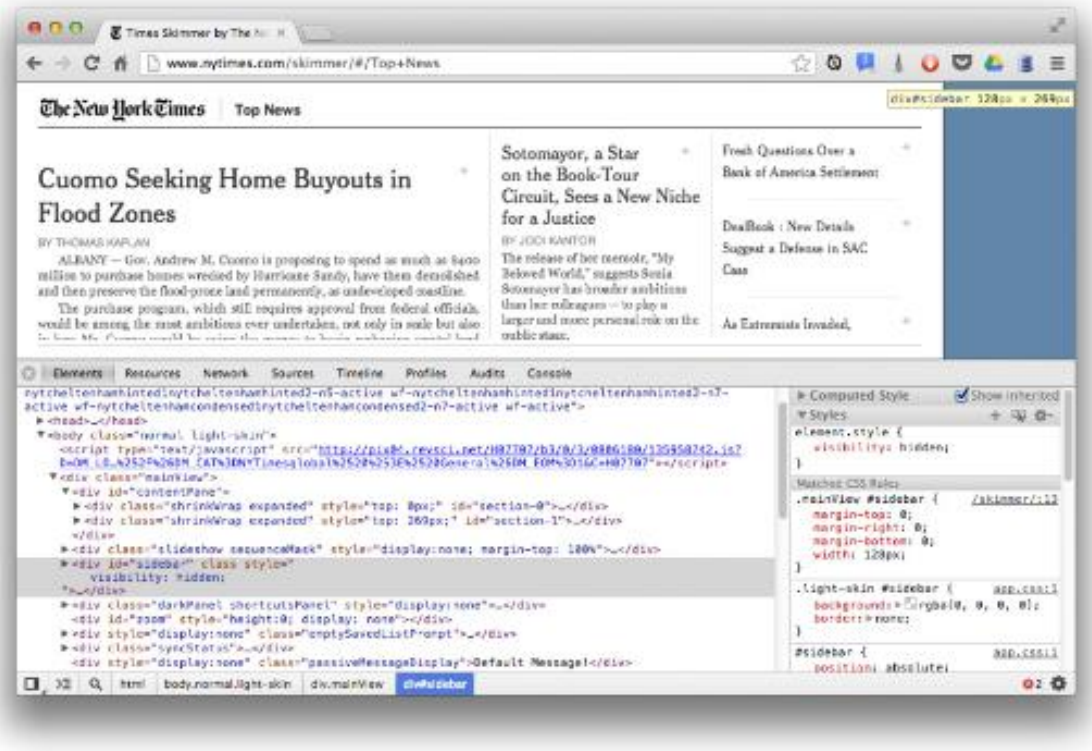
Ketahui Alat Anda

Sebelum kita menyelami alur kerja yang efisien untuk menemukan dan memperbaiki masalah kinerja rendering, ada beberapa alat lain yang tersedia di Chrome yang perlu diperhatikan.

6.4 PINTASAN UNTUK MENYEMBUNYIKAN ELEMEN DOM DENGAN CEPAT

DevTools memiliki pintasan berguna yang memungkinkan Anda dengan mudah mengubah pengaturan visibilitas: tersembunyi pada suatu elemen. Saat gaya ini diterapkan pada sebuah elemen, gaya ini tidak dicat namun mempertahankan tata letak halaman dalam keadaan tidak berubah.

Untuk menggunakan pintasan, pilih elemen DOM di panel Elemen lalu tekan tombol H. Saat dipasangkan dengan cat persegi panjang dan Timeline, Anda dapat dengan mudah mengevaluasi elemen DOM mana yang menghabiskan waktu lama dalam pengecatan.

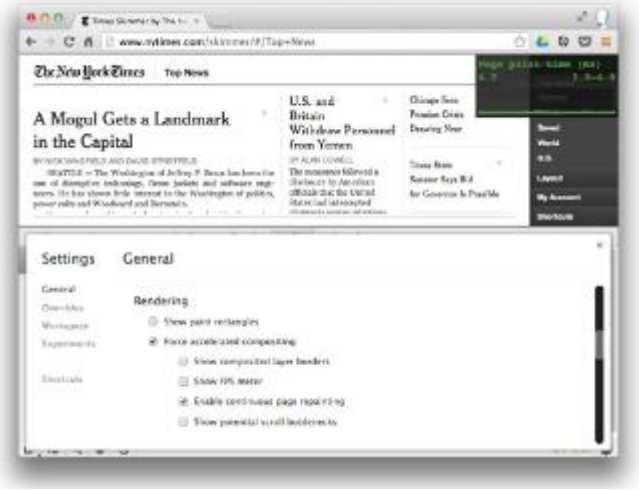


Gambar 6.4 Menelusuri pohon DOM di panel Elemen, tombol pintas H membantu mengidentifikasi elemen dengan cat tebal.

Mode Lukisan Terus Menerus Untuk Diagnosis Gaya Lambat

Beberapa alasan Chrome mengecat ulang area laman meliputi: interaksi pengguna yang menyebabkan perubahan gaya pada elemen DOM; Node DOM diubah (memaksa perhitungan ulang tata letak); dan operasi lainnya yang menyebabkan perubahan tata letak halaman.

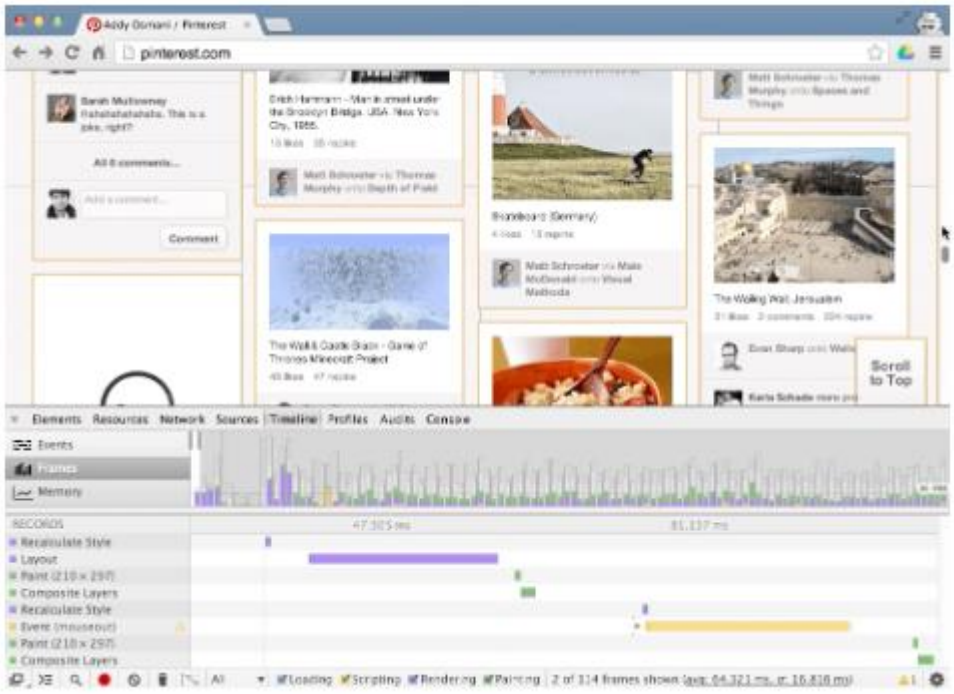
Akan berguna untuk memahami mengapa pengecatan ulang terjadi pada halaman Anda. “Pengecatan ulang halaman berkelanjutan” adalah fitur di panel Pengaturan yang membantu mengidentifikasi elemen yang memiliki biaya pengecatan tinggi pada halaman. Ini memaksa halaman untuk terus-menerus mengecat ulang, memberikan penghitung yang menunjukkan seberapa banyak pekerjaan pengecatan yang sedang dilakukan. Anda dapat menggunakan pintasan H yang disebutkan di atas untuk mengganti gaya yang berbeda (perhatikan penghitungnya!) untuk mendiagnosis penyebab perlambatan.



Gambar 6.5 Perhatikan penghitung hitam dan hijau di sudut kanan atas untuk mendapatkan wawasan tentang pengecatan ulang.

Tampilkan Batas Lapisan Kompositasi

Pengaturan bagus lainnya di Alat Pengembang yang dapat membantu di sini adalah "Tampilkan batas lapisan gabungan." Fitur ini akan memberi Anda wawasan tentang elemen DOM yang dimanipulasi di tingkat GPU.



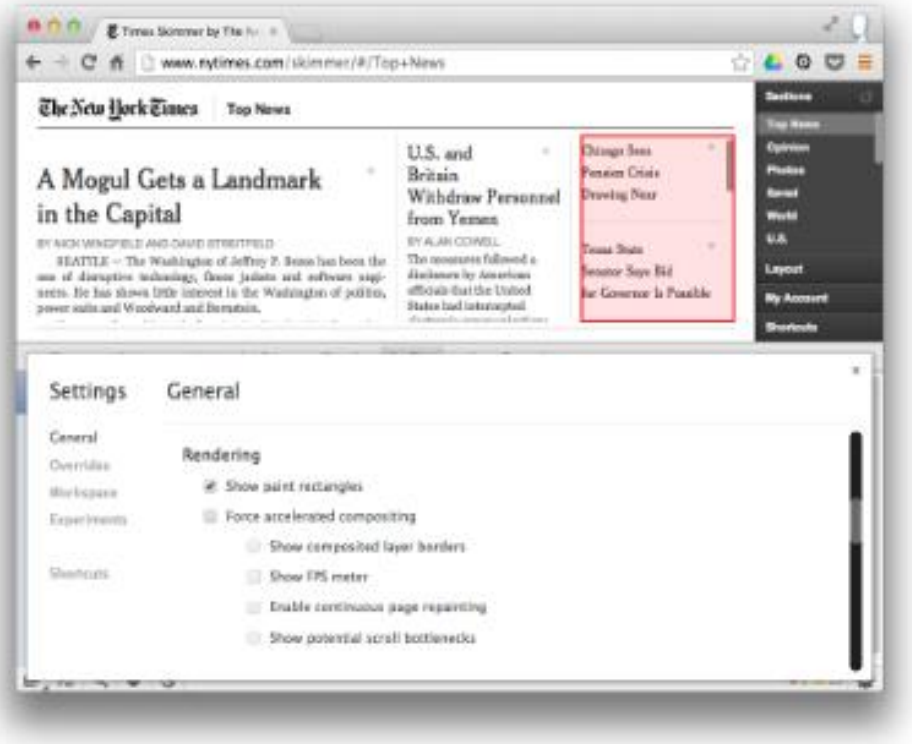
Gambar 6.6 Dengan mengaktifkan "Tampilkan batas lapisan gabungan", elemen yang dipromosikan ke lapisan baru akan disorot dengan batas berwarna.

Jika suatu elemen memanfaatkan akselerasi GPU, Anda akan melihat batas oranye di sekelilingnya jika elemen ini aktif. Sekarang ketika kita menelusurinya, kita tidak benar-benar

melihat penggunaan lapisan gabungan pada halaman ini — tidak ketika kita mengklik “Gulir ke atas” atau sebaliknya. Chrome menjadi lebih baik dalam menangani promosi lapisan secara otomatis di latar belakang, namun, seperti disebutkan, pengembang terkadang menggunakan peretasan TranslateZ untuk membuat lapisan gabungan. Di bawah ini adalah beranda salah satu situs dengan terjemahanZ(0) diterapkan ke semua pin. Ini tidak mencapai 60fps, tetapi semakin mendekati 30fps yang konsisten di desktop, yang sebenarnya tidak buruk.

Tampilkan Cat Persegi Panjang

Di bawah “Rendering” dalam roda “Pengaturan”, Anda dapat mengaktifkan fitur yang disebut “Tampilkan persegi panjang cat” untuk membantu Anda melihat secara visual area yang dicat ulang di setiap bingkai. Dengan mengaktifkan fitur ini, memvisualisasikan apa yang memperlambat halaman akan menjadi mudah. Anda ingin menjaga area yang dicat ulang sekecil mungkin.



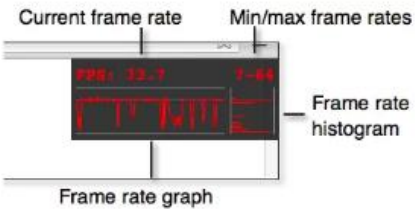
Gambar 6.7 Dalam tangkapan layar ini, sebuah persegi panjang cat digambar di atas wilayah tempat div dengan overflow:scroll digambar. Ini bagus karena ini adalah bagian layar yang relatif kecil.

6.5 PENGHITUNG FPS

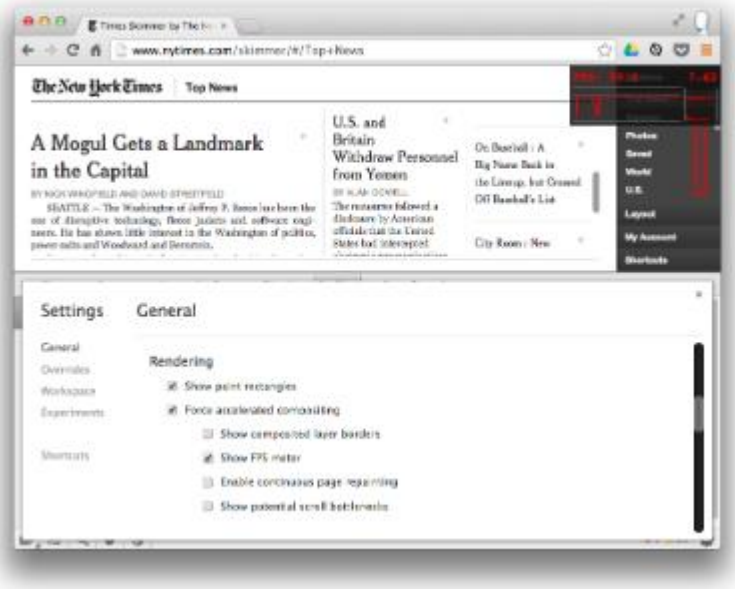
Alat yang lebih tua namun sama bergunanya untuk memvisualisasikan frame rate dan jank adalah penghitung frame per detik secara real-time. Ini dapat diaktifkan di DevTools dengan masuk ke menu Pengaturan dan mencentang “Tampilkan FPS meter.”

Saat diaktifkan, Anda akan melihat kotak gelap di sudut kanan atas halaman Anda dengan statistik bingkai. Penghitung dapat digunakan selama pengeditan langsung untuk

mendiagnosis apa yang menyebabkan penurunan kecepatan bingkai di halaman Anda tanpa harus beralih bolak-balik dengan tampilan Timeline.



Ingatlah bahwa hanya dengan melacak penghitung FPS dapat menyebabkan Anda tidak melihat bingkai dengan jank yang terputus-putus. Berhati-hatilah saat menggunakan konten. Perlu juga dicatat bahwa FPS di desktop tidak sama dengan FPS di perangkat lain dan perhatian khusus harus diberikan untuk membuat profil kinerja di sana juga.



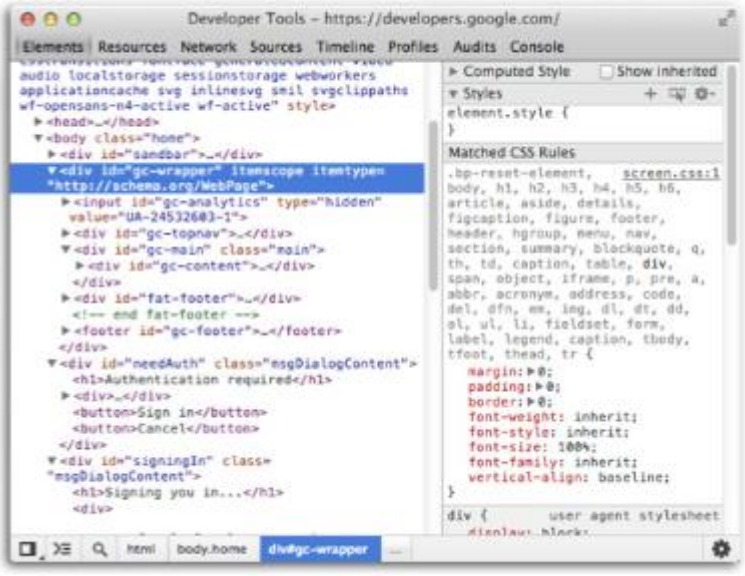
Gambar 6.8 Pengukur FPS yang menunjukkan kecepatan bingkai saat ini, minimum dan maksimum serta histogram varian kecepatan bingkai.

Alur Kerja "Temukan dan Perbaiki" untuk Seluler

Sulit untuk membangun pengalaman Web seluler yang bermakna tanpa melakukan pengujian pada perangkat sebenarnya yang Anda dukung. Untungnya, browser Web modern menampilkan alat yang dapat memprofilkan kinerja rendering Anda di desktop serta kinerja Anda di perangkat seluler yang terhubung. Hal ini dilakukan melalui proses debug jarak jauh, yang harus Anda siapkan sebelum Anda dapat membuat profil halaman Anda di perangkat seluler.

Siapkan Debugging Jarak Jauh

Biasanya Anda akan men-debug halaman Anda dari jarak jauh melalui USB. Selama perangkat seluler Anda terhubung ke mesin pengembangan, Anda akan dapat membuat profil halaman menggunakan Timeline, serta melihat dan mengedit HTML, skrip, dan gaya hingga Anda memiliki halaman yang dioptimalkan yang berperilaku sedikit lebih baik di semua halaman. perangkat target Anda.



Gambar 6.9 Men-debug Chrome untuk Android menggunakan Alat Pengembang Chrome.

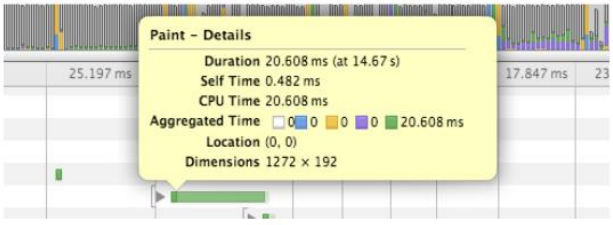
Untuk menyiapkan proses debug jarak jauh untuk versi Chrome atau Opera Anda, ikuti panduan proses debug jarak jauh di dokumentasi Chrome DevTools. Anda kemudian dapat mengikuti tutorial di bawah ini pada perangkat seluler asli atau desktop Anda.

Perhatikan bahwa dokumen yang ditautkan di atas juga akan memberi Anda dua opsi untuk mengakses halaman yang ingin Anda debug pada perangkat Anda. Anda dapat membuka halaman tersebut di browser perangkat Anda, atau menggunakan fitur baru yang disebut penerusan port terbalik untuk men-debug versi kode yang dihosting secara lokal di perangkat Anda.

6.6 ALUR KERJA OPTIMASI

Setelah Anda menyiapkan proses debug jarak jauh, berikut adalah alur kerja untuk mendiagnosis masalah cat dan jank:

1. Buka halaman di perangkat Anda, luncurkan Chrome DevTools dan alihkan ke panel “Timeline”. Tekan rekam dan berinteraksilah dengan halaman Anda dengan cara yang sama seperti yang dilakukan pengguna Anda.
2. Periksa Timeline untuk melihat frame apa pun yang melebihi anggaran (yaitu di bawah 60fps ideal). Jika anggaran Anda hampir habis, kemungkinan besar anggaran Anda di perangkat seluler jauh melebihi anggaran. Usahakan untuk menyelesaikan semua pekerjaan Anda dalam waktu 10 md untuk mendapatkan margin tertentu. Perhatikan bahwa margin ini ditujukan untuk perangkat yang lebih lambat dan Anda hampir pasti harus



menjalankan analisis ini pada perangkat seluler menggunakan debugging jarak jauh16 (jika membangun untuk perangkat seluler, Anda memang seharusnya melakukannya!).

3. Setelah Anda menyadari ada bingkai yang tersendat-sendat, periksa apa penyebabnya. Apakah itu cat yang sangat besar? Masalah tata letak CSS? JavaScript?
4. Perbaiki masalahnya. Jika ini masalah cat atau tata letak:
   - Buka Pengaturan dan aktifkan "Pengecatan ulang halaman berkelanjutan."
   - Telusuri pohon DOM, sembunyikan elemen yang tidak penting menggunakan pintasan sembunyikan (H). Anda mungkin menemukan bahwa menyembunyikan elemen tertentu akan membuat perbedaan besar pada waktu pengecatan dan kecepatan bingkai Anda.
   - aku aku aku. Kini kita mengetahui bahwa ada sesuatu pada suatu elemen yang memperlambat proses pengecatan. Hapus centang gaya yang dapat berdampak pada waktu pengecatan (misalnya bayangan kotak) untuk elemen dan periksa kembali kecepatan bingkai Anda.
   - Lanjutkan hingga Anda menemukan gaya yang bertanggung jawab atas perlambatan tersebut.
5. Bilas dan ulangi.

Khususnya pada situs yang sangat bergantung pada gulir, Anda mungkin menemukan bahwa konten utama Anda mengandalkan overflow:scroll. Ini merupakan tantangan nyata karena pengguliran ini tidak dipercepat oleh GPU dengan cara apa pun sehingga konten dicat ulang setiap kali pengguna Anda menggulir. Anda dapat mengatasi masalah tersebut menggunakan pengguliran halaman normal (meluap: terlihat) dan posisi: tetap.

Referensi Garis Waktu

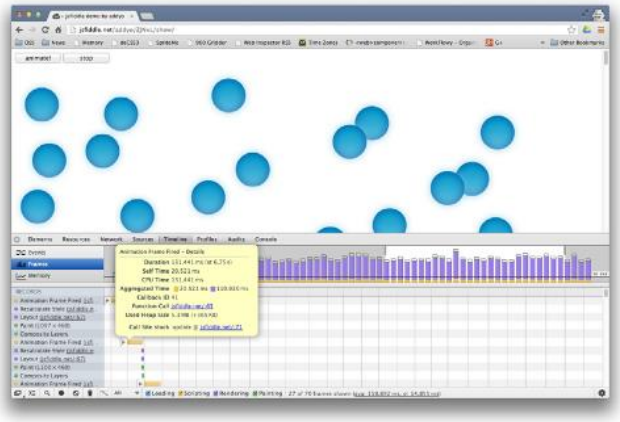
- Lapisan komposit: mesin rendering Chrome menggabungkan lapisan gambar.
- Dekode gambar: sumber daya gambar didekodekan.
- Pengubahan ukuran gambar: gambar diubah ukurannya dari dimensi aslinya.
- Cat: lapisan gabungan dicat pada suatu wilayah tampilan. Mengarahkan kursor ke rekaman Paint akan menyorot wilayah tampilan yang telah diperbarui.
- Tata letak tidak valid: tata letak halaman menjadi tidak valid karena perubahan DOM.
- Layout: tata letak halaman dijalankan.
- Hitung ulang gaya: Chrome menghitung ulang gaya elemen.
- Gulir: konten tampilan bertingkat digulir.

Namun, ingatlah untuk melakukan pengujian pada desktop dan seluler: karakteristik kinerjanya sangat bervariasi. Gunakan garis waktu di keduanya, dan lihat grafik waktu pengecatan Anda dalam mode Cat Berkelanjutan untuk mengevaluasi seberapa cepat Anda menghabiskan anggaran Anda. Sekali lagi, jangan gunakan peretasan ini pada setiap elemen di halaman – ini mungkin bisa dilakukan di desktop, tetapi tidak di seluler. Alasannya adalah peningkatan penggunaan memori video dan peningkatan biaya pengelolaan lapisan, yang keduanya dapat berdampak negatif pada kinerja. Sebaliknya, gunakan pengomposisian perangkat keras hanya untuk mengisolasi elemen yang biaya catnya cukup tinggi.

Tutorial: Mendapatkan Pengalaman Web Seluler Bebas Jank

Kita sudah membicarakannya, tapi mari kita lihat aplikasi sederhana dengan beberapa animasi yang tersendat-sendat dan lihat apakah kita dapat mengoptimalkannya agar bebas dari jank. Sekarang, ingatlah bahwa kinerja laman Anda di seluler sangat berbeda dengan kinerja desktop, jadi pastikan Anda telah menyiapkan proses debug jarak jauh untuk tutorial ini. Mari kita mulai.

1. Buka http://jsfiddle.net/AXEJY/.
2. Klik "hidupkan!" Seperti yang Anda lihat, ada jeda visual dalam gerakan, sehingga menghasilkan animasi yang kurang optimal. Kami dapat merekam sesi Timeline selama animasi halaman ini untuk mengonfirmasi bahwa kami mengalami masalah dalam mencapai frame rate optimal. Performa animasi jauh lebih buruk di seluler dibandingkan di desktop karena kami bekerja dengan GPU yang lebih terbatas.



3. Mari kita lihat apa yang menyebabkan segalanya melambat. Berikut JavaScript untuk animasi kami serta CSS:

JavaScript

```
// setup
var rAF = window.requestAnimationFrame;
var startBtn = document.querySelector('.animate');
var stopBtn = document.querySelector('.stop');

// state
var running = false;

// add listeners
// start
startBtn.addEventListener('click', function(e) {
 running = true; rAF(update); });

// stop
stopBtn.addEventListener('click', function(e) {
 running = false;
});

// Set the heights for all these
// movers in simple CSS style.top
var movers = document.querySelectorAll('.mover');
```

```
(function init() {
     for (var m = 0; m < movers.length; m++) {
     movers[m].style.top = (m * 20 + 50) + 'px';
}
})();

// animation loop
function update(timestamp) {
     for (var m = 0; m < movers.length; m++) {
       movers[m].style.left = ((Math.sin(movers[m].offsetTop +
timestamp / 1000) + 1) * 500) + 'px';
  }
  if (running){
  rAF(update);
 }
};
rAF(update);
mover {
      background:url(http://jankfree.org/velocityeurope/example
s/toomuch-layout/particle.png);
       height: 100px;
       width: 100px;
       position: absolute;
       z-index: 0;
}

input {
      z-index: 2;
      font-size: 25pt;
      height: 100px;
      width: 100px;
      display: inline-block;
}
```

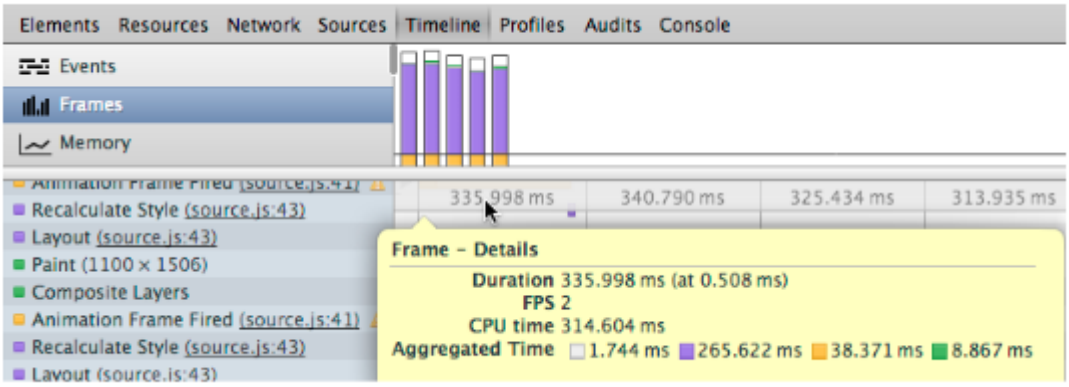
CSS

```
     .mover {
          background:url(http://jankfree.org/velocity-
europe/examples/toomuch-layout/particle.png);
          height: 100px;
          width: 100px;
          position: absolute;
          z-index: 0;
     }
     input {
```

```
        z-index: 2;
        font-size: 25pt;
        height: 100px;
        width: 100px;
        display: inline-block;
}
```

Analisis Rekaman

Melihat rekaman beberapa frame pertama, terlihat jelas bahwa setiap frame membutuhkan waktu lebih dari 300 ms untuk diselesaikan.

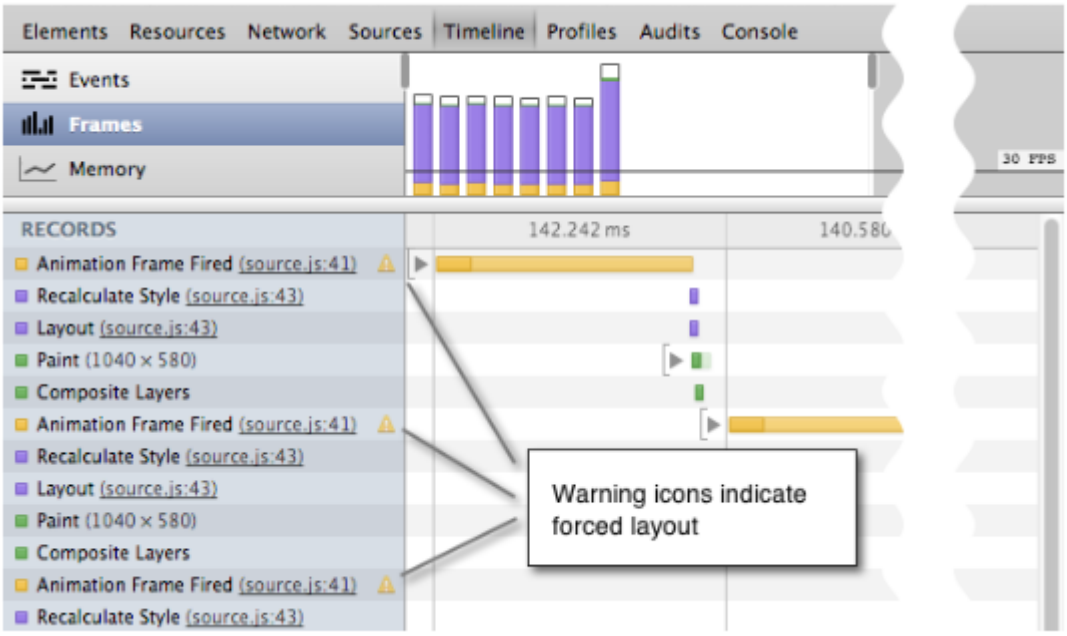


Gambar 6.10 Jika Anda mengarahkan mouse ke salah satu bingkai, akan muncul pop-up yang menunjukkan detail tambahan tentang bingkai tersebut.

Untuk meningkatkan kinerja rendering, Chrome biasanya mengelompokkan perubahan tata letak yang diminta oleh laman dan mencoba menjadwalkan penerusan tata letak untuk menghitung dan merender perubahan yang diminta secara asinkron. Namun, jika halaman meminta nilai properti tergantung pada tata letaknya (misalnya offsetWidth atau offsetHeight), browser dipaksa untuk segera dan serentak melakukan tata letak halaman. Ini disebut tata letak sinkron paksa dan dapat memberikan penurunan performa rendering yang cukup signifikan, terutama bila dilakukan berulang kali pada pohon DOM yang lebih besar. Kami menyebutnya tata letak skenario thrashing.

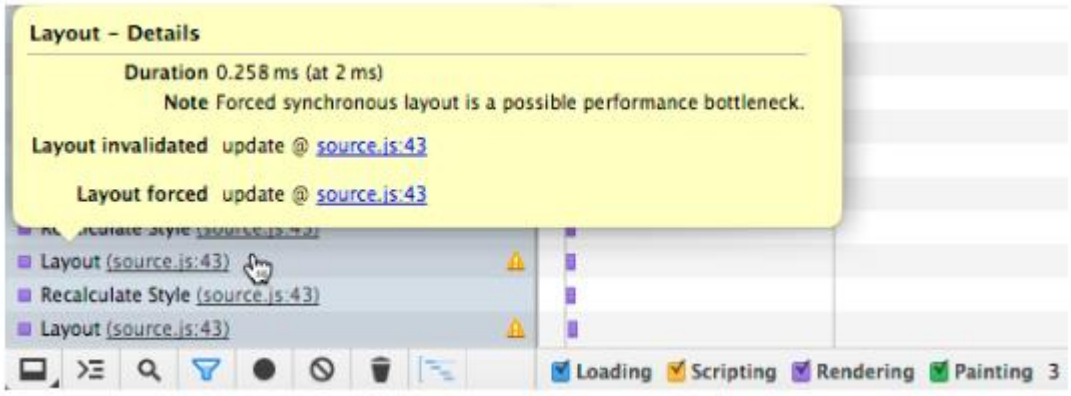
Timeline dapat memperingatkan Anda ketika menemukan tata letak sinkron yang dipaksakan dengan ikon peringatan kuning di sebelah catatan Timeline yang sesuai. Jika Anda mengarahkan kursor ke salah satu rekaman ini, jejak tumpukan untuk kode yang membuat tata letak menjadi tidak valid dan kode yang memaksanya akan ditampilkan.

Jadi, di Timeline kami, temukan rekaman "Bingkai Animasi Diaktifkan" dan temukan ikon peringatan kuning di sebelahnya yang menunjukkan tata letak sinkron yang dipaksakan. Ikonnya sedikit redup yang menunjukkan bahwa salah satu catatan turunannya berisi kode yang melanggar, bukan catatan itu sendiri.



Gambar 6.11 Perluas "Bingkai Animasi Diaktifkan" untuk melihat turunannya.

Rekaman anak memperlihatkan pola penghitungan ulang gaya dan rekaman tata letak yang panjang dan berulang. Setiap rekaman tata letak adalah hasil penghitungan ulang gaya yang, pada gilirannya, merupakan hasil dari penangan requestAnimationFrame() yang meminta nilai offsetTop untuk setiap gambar di laman.



Gambar 6.12 Arahkan mouse Anda ke salah satu rekaman Layout dan klik tautan untuk source.js di samping properti Layout Forced.

Panel Sumber terbuka pada baris 43 file sumber pada fungsi update(), yang merupakan pengendali panggilan balik `requestAnimationCallback()`. Penangan menghitung properti gaya CSS kiri gambar pada nilai offsetTop gambar. Hal ini memaksa Chrome untuk segera melakukan tata letak baru untuk memastikan tata letak tersebut memberikan nilai yang benar.

```
// animation loop
function update(timestamp) {
  for (var m = 0; m < movers.length; m++) {
    movers[m].style.left = ((Math.sin(movers[m].offsetTop +
    timestamp/1000) + 1) * 500) + 'px';
    }
  raf = window.requestAnimationFrame(update);
};
```

Kita tahu bahwa memaksakan tata letak halaman pada setiap frame animasi akan memperlambat segalanya. Sekarang kita dapat mencoba memperbaiki masalahnya langsung di DevTools.

Terapkan Fix Dalam Devtools

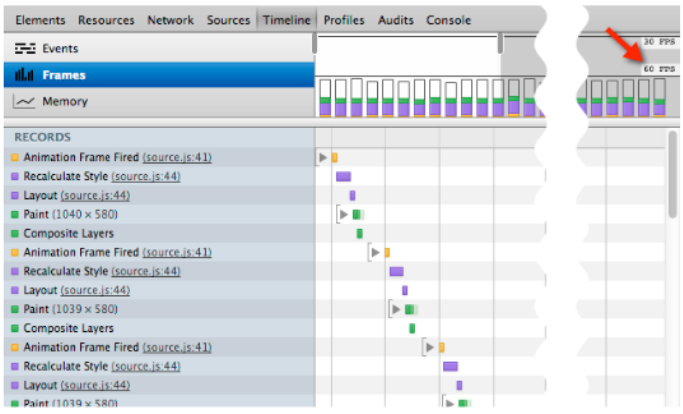
Sekarang setelah kita mengetahui penyebab masalah kinerja, kita dapat memodifikasi file JavaScript langsung di panel Sumber dan langsung menguji perubahan di desktop atau perangkat seluler.

Pada panel Sources yang dibuka sebelumnya, ganti baris 43 dengan kode berikut.

```
movers[m].style.left = ((Math.sin(m + timestamp/1000) + 1) *
500) + 'px';
```

1. Versi ini menghitung properti gaya kiri setiap gambar pada indeksnya dalam array penahannya, bukan pada properti yang bergantung pada tata letak (offsetWidth).
2. Simpan perubahan Anda dengan menekan Command+S atau Control+S.

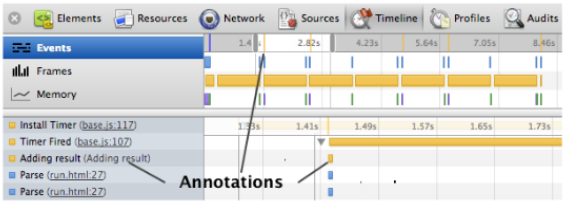
verifikasi dengan rekaman lain



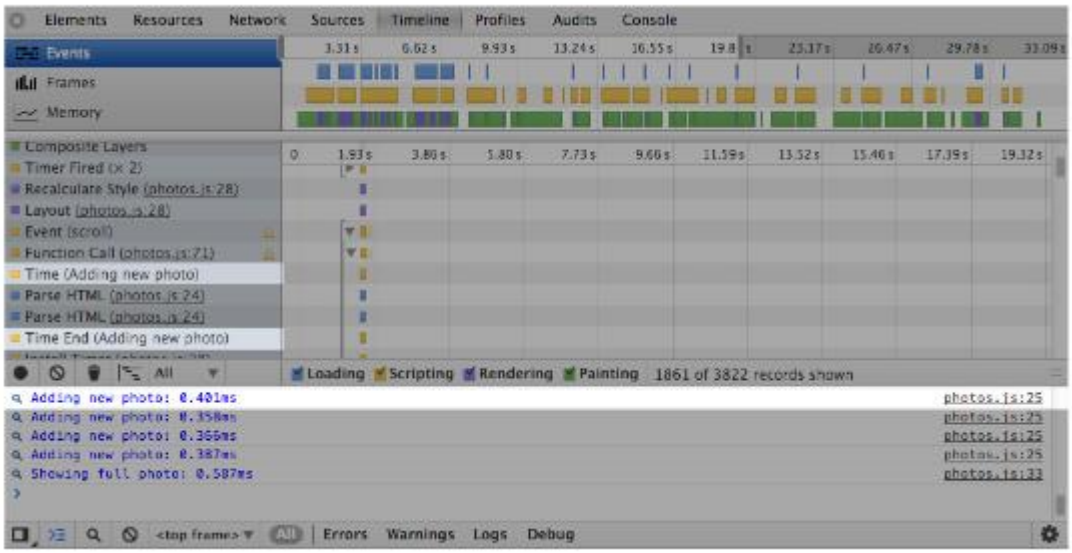
Animasinya jelas lebih cepat dan halus dibandingkan sebelumnya, namun selalu merupakan praktik yang baik untuk mengukur perbedaannya dengan rekaman lain. Seharusnya terlihat seperti rekaman di bawah ini.

Kiat Pro

JavaScript Anda dapat menganotasi rekaman Timeline DevTools menggunakan console.timeStamp() yang berfungsi baik Anda menggunakan proses debug jarak jauh atau tidak. Lihat di bawah untuk "Menambahkan hasil," sebuah anotasi yang ditambahkan oleh kode kami selama pembuatan profil

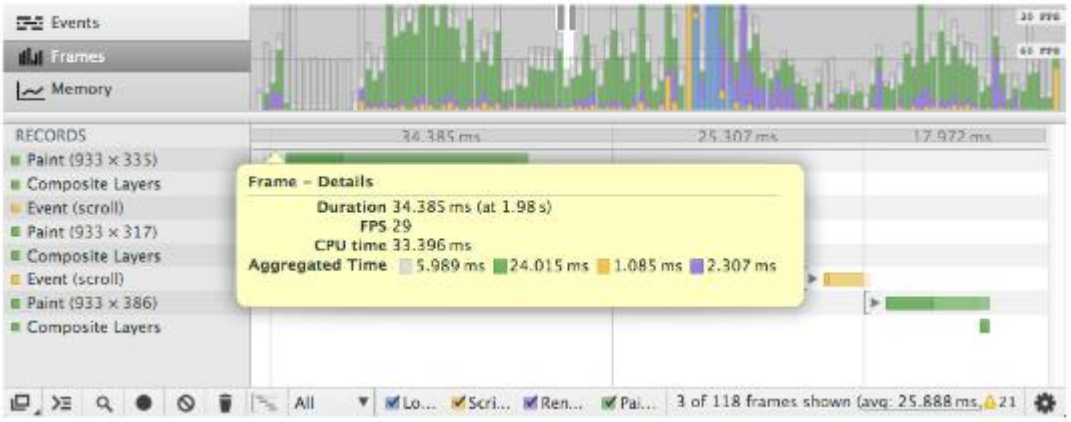


Kode Anda juga dapat menggunakan console.time() dan console.timeEnd() untuk menandai rentang dalam rekaman Timeline DevTools:



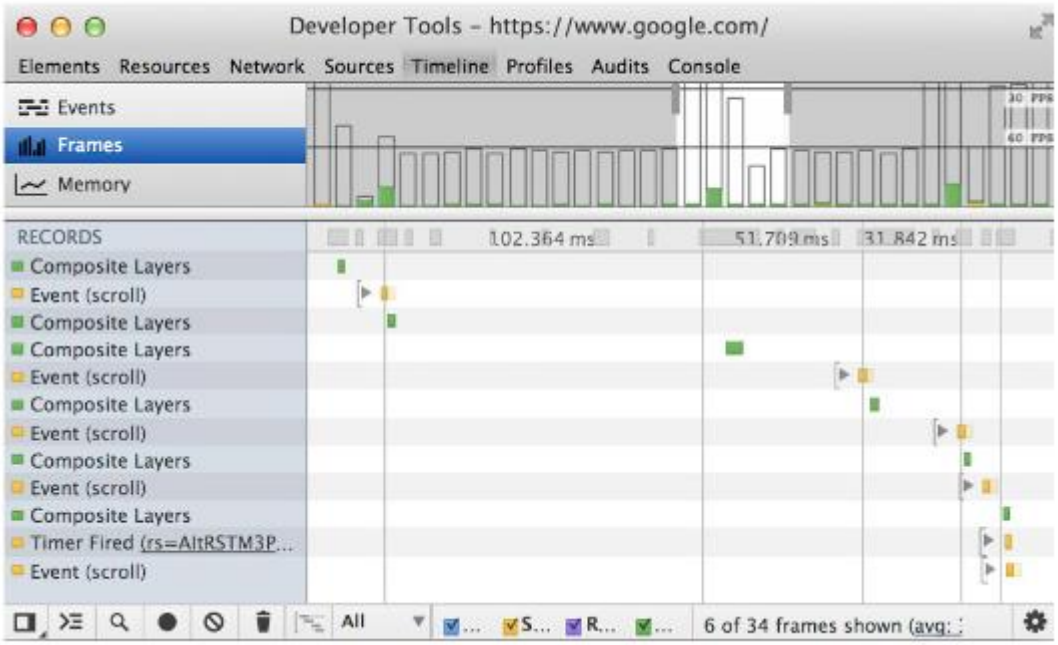
Gambar 6.13 Menambahkan Foto baru

Jika Anda mengaktifkan “Tampilkan aktivitas CPU pada penggaris”, Anda dapat melapisi aktivitas CPU di rekaman Timeline Anda. Jika ini menyala, bilah lampu menunjukkan CPU sedang sibuk. Jika Anda mengarahkan kursor ke bilah CPU, ini akan menyoroti wilayah di mana CPU aktif.



Gambar 6.14 aktivitas CPU pada penggaris

Jika Anda ingin menelusuri rekaman jenis tertentu, Anda dapat melakukannya dengan menggunakan pintasan Control+F (Command+F di Mac OS X) saat berada di Timeline. Cukup masukkan nama jenis catatan (misalnya “scroll”) dan Timeline hanya akan menampilkan catatan yang berisi istilah tersebut.



Gambar 6.14 Hasil Rendering

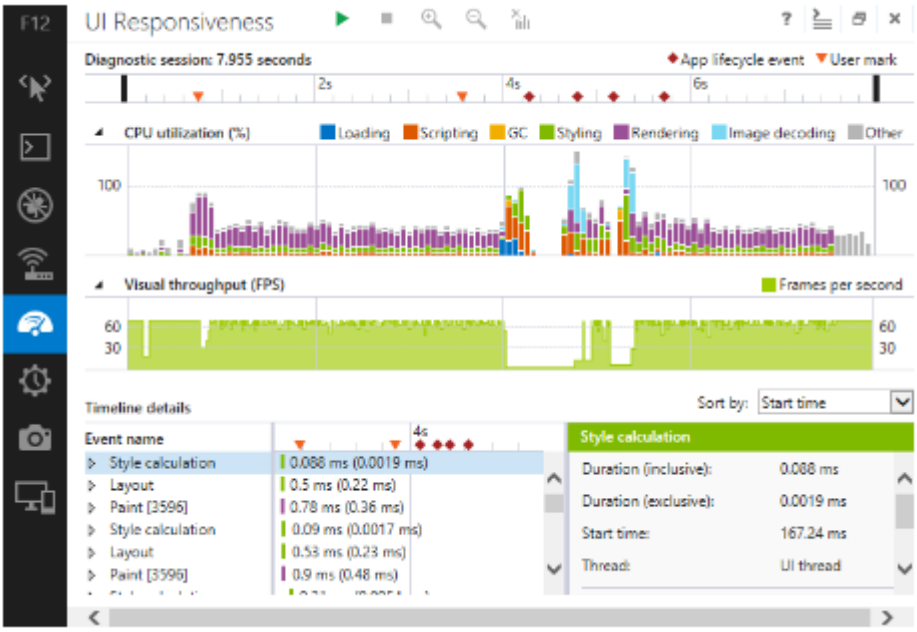
Jika Anda bertanya-tanya apa arti bilah transparan di Timeline tersebut, bingkai berongga ini berhubungan dengan salah satu dari dua hal: JavaScript Anda di thread utama sedang sibuk melakukan sesuatu yang lupa kami tampilkan (instrumen) di DevTools; atau Anda mengalami hambatan oleh GPU Anda.

Merender Alat Kinerja di IE, Firefox dan Safari

Sejauh ini kita telah membahas alat untuk menemukan dan memperbaiki masalah kinerja rendering di Chrome. Karena rendering masih merupakan area baru yang dapat dioptimalkan oleh banyak pengembang Web, alat yang tersedia untuk itu di browser lain masih terus berkembang namun saya gembira dengan arah yang telah mereka ambil.

6.7 ALAT PENGEMBANG IE11 F12

Banyak pengembang yang terkejut mendengar bahwa alat pengembang IE telah berkembang pesat akhir-akhir ini. Di IE11, alat pengembang F12 memperkenalkan fitur Responsif UI khusus untuk membuat profil masalah kinerja seputar jank, kelambatan, dan area masalah umum lainnya seperti penggunaan CPU dan memori.

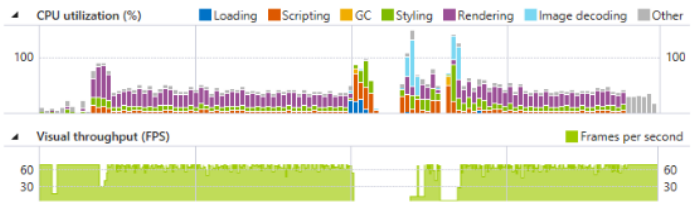


Gambar 6.15 Tampilan Responsih UI

Alur kerja

Dengan IE11 terinstal dan alat Responsif UI dimuat, Anda akan diminta untuk memulai sesi profil kinerja baru di panel utama.

- Pilih panah di bagian atas alat untuk memulai pembuatan profil. Pertahankan tindakan yang Anda lakukan seminimal mungkin untuk menangkap perlambatan yang ingin Anda selidiki di halaman Anda. Lebih dari itu akan berisiko menurunkan keterbacaan hasil Anda.



- Ketika Anda telah menyelesaikan interaksi yang ingin Anda tangkap, klik “Hentikan pembuatan profil” atau ikon berbentuk persegi di bagian bawah alat pengembang untuk membuat laporan dari hasil ini.

Alat Responsif UI hadir dengan Timeline kinerjanya sendiri yang akan Anda lihat selanjutnya. Ini serupa, tetapi sedikit berbeda dengan yang kita lihat di Chrome. Gunakan ini untuk memvisualisasikan kecepatan bingkai halaman (yang oleh IE disebut sebagai throughput visual) dan penurunan kecepatan bingkai ini, menunjukkan bahwa kami telah menurunkan bingkai atau telah terjadi perlambatan. Ini juga menangkap berapa banyak CPU yang digunakan, tapi ini kurang menarik untuk tujuan kita.

Selanjutnya, Anda dapat menelusuri rekaman tertentu yang sesuai dengan menggunakan tampilan Detail Garis Waktu. Dokumen resmi alat pengembang IE11 mencakup kategori yang tercantum secara lebih komprehensif, namun singkatnya mencakup: penguraian CSS dan HTML; permintaan jaringan; evaluasi naskah; menerjemahkan gambar;

panggilan balik bingkai animasi; dan yang menarik bagi kami: tata letak dan rendering. Tata letak dalam konteks ini mengacu pada perubahan pada DOM yang menyebabkan dimensi atau posisi elemen berubah dan rendering mengacu pada perubahan visual pada DOM yang menyebabkan wilayah halaman dicat ulang. Detail yang dirangkum mencakup dimensi dan koordinat lapisan render yang terpengaruh.

Untuk mempelajari lebih lanjut tentang alat pengembang F12, lihat dokumentasi resmi dan panduan mereka tentang alat Responsiveness UI.

FIREFOX

Nightlies FirefoX memiliki fitur yang disebut paint fiashing yang juga dapat digunakan untuk menentukan wilayah halaman mana yang sedang dicat ulang oleh browser. Dengan menyalakan cat berkedip, setiap wilayah diwarnai dengan warna acak sehingga sangat mudah untuk membedakan satu wilayah dengan wilayah lainnya. Daerah dengan cat yang sangat banyak berkedip adalah daerah yang akan merugikan Anda, jadi cobalah untuk meminimalkannya sebisa mungkin.

Gambar 6.17 Contoh Browsing menggunakan FireFox

Aktifkan flashing cat

1. Pastikan Anda telah menginstal FirefoX 11 atau lebih tinggi (Beta, Aurora, atau Nightly).
2. Buka tentang: konfigurasi.
3. Terima peringatan yang ditampilkan.
4. Klik kanan dan pilih Baru → Boolean.
5. Ketik ngalayout.debug.paint_flashing.
6. Atur pilihan ke Benar. Itu dia!

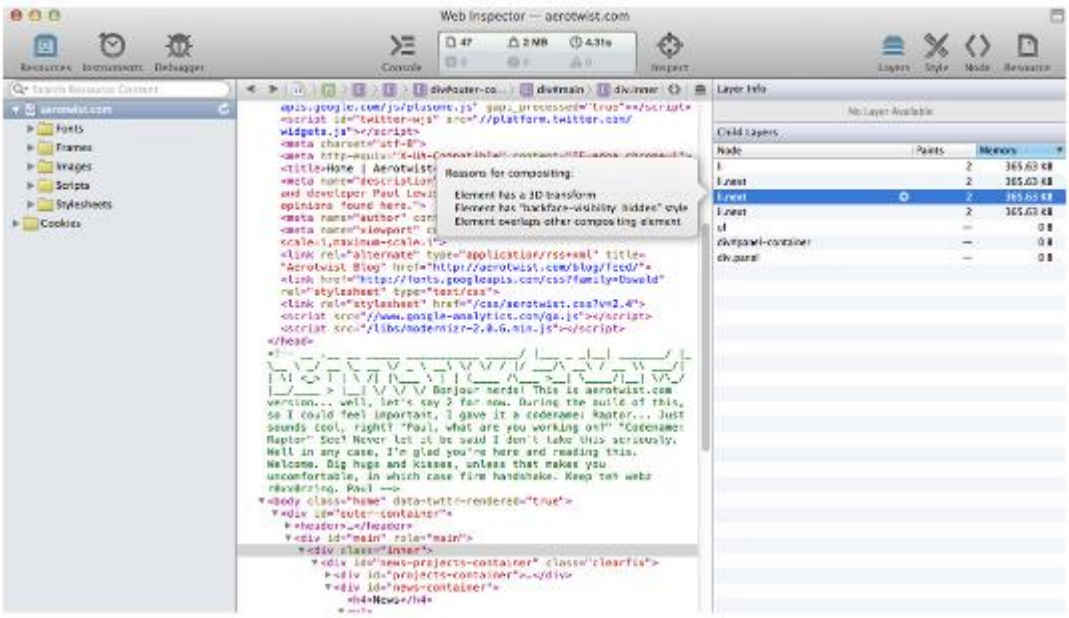
WEBKIT/SAFARI

Apple juga telah melakukan pekerjaan menarik di nightlies WebKit. Jika Anda mengambil salah satu rilis terbaru, Anda akan menemukan bahwa dua alat baru ditampilkan untuk membantu meningkatkan kinerja rendering. Yang pertama adalah Web Inspector baru-

baru ini memperkenalkan sidebar detail lapisan untuk mendapatkan wawasan tentang komposisi elemen WebKit. Ini menyoroti informasi lapisan untuk elemen DOM yang dipilih (jika telah dipromosikan menjadi lapisan) serta lapisan untuk elemen turunannya.

Saat Anda memilih sebuah lapisan di sidebar, ini akan menampilkan pop-over yang merangkum alasan mengapa lapisan tersebut dibuat (dipromosikan). Bergantung pada jenis halaman yang sedang Anda kerjakan, menghilangkan lapisan dapat menjadi cara yang baik untuk mengurangi overhead kinerja grafis halaman Anda.

Cara tercepat untuk melihat informasi lapisan suatu halaman adalah dengan memeriksa isi dokumen dan meninjau lapisan anak. Anda kemudian dapat mempersempit daftar dengan memeriksa keturunan yang lebih dalam. Mirip dengan Chrome, Anda juga dapat menampilkan pengomposisian batas lapisan, yang dilakukan di bilah navigasi pohon DOM, yang melapisi halaman Anda untuk memberikan visualisasi lapisan yang lebih jelas dan berapa kali lapisan tersebut dicat ulang.



Gambar 6.18 Tampilan Safari

Tunjukkan Alasan Pengomposisian

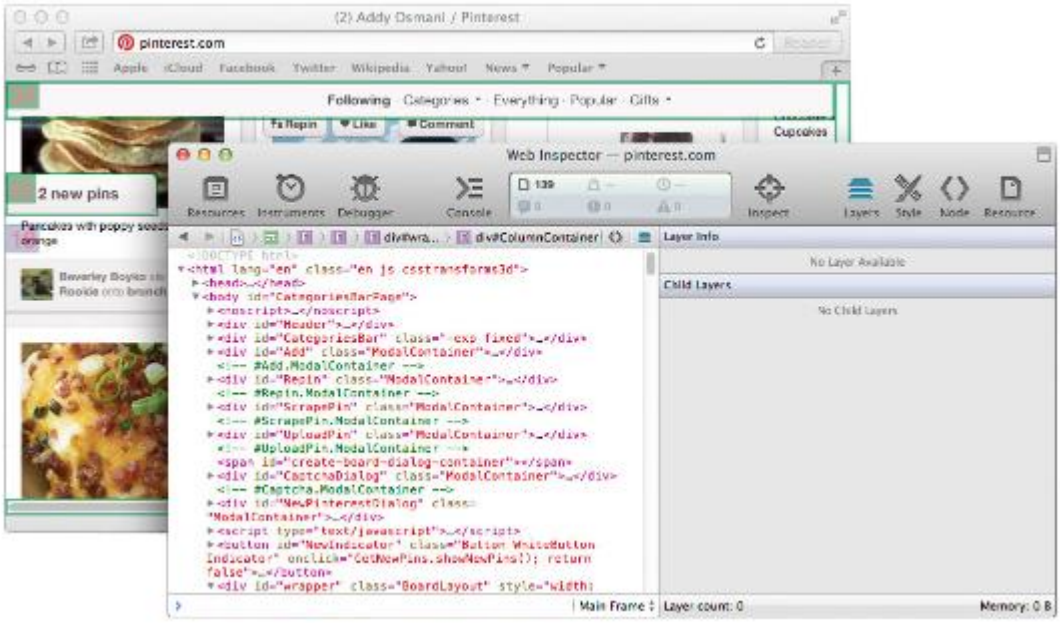
- Ambil nightlies WebKit.20
- Pastikan untuk menonaktifkan “Gunakan WebKit Web Inspector.”
- Tekan tombol “Lapisan” yang mengkilap. Ledakan!
- Alasan untuk promosi lapisan ditampilkan saat diarahkan.

Yang kedua adalah Anda dapat menampilkan berapa kali suatu lapisan dicat (ulang), berguna untuk memahami bagian mana dari halaman Anda yang mungkin dicat secara berlebihan karena perilaku dalam skrip Anda.

Tampilkan Jumlah Lapisan Cat

- Di bawah Lapisan → Tampilkan batas lapisan gabungan.

- Seperti Chrome, ini menampilkan lapisan yang dipromosikan menjadi lapisan komposit.
- Namun, ini juga menampilkan berapa kali sebuah layer dicat!



Gambar 6.19 Lampisan komposit

Kesimpulan

Pengalaman bebas jank sangat penting agar Web seluler dapat mendekati apa yang biasa digunakan pengguna dengan aplikasi asli. Setinggi itulah standarnya dan Anda bisa menghadapi tantangan untuk menyelesaikannya.

Namun membangun pengalaman luar biasa di Web seluler membutuhkan lebih dari sekadar desainer yang baik dan CSS yang canggih anda benar-benar harus peduli dengan kinerja. Pengguna mengharapkan nuansa asli, dan animasi halus yang tidak pernah menghilangkan bingkai dapat memberikannya kepada mereka. Saat halaman Anda di-scroll dengan lambat, animasi akan tersendat-sendat dan efeknya sangat lambat… hal ini sangat mengganggu dan dapat memengaruhi pengalaman dan keterlibatan pengguna Anda.

Ingatlah bahwa kinerja dapat sangat bervariasi antar browser dan bau kinerja di satu browser mungkin tidak ada di browser lain. Gunakan alat pengembang mereka untuk melihat apa yang sebenarnya terjadi. Pastikan perhitungan ulang gaya Anda tidak mengubah lebih banyak gaya daripada yang Anda harapkan. Usahakan area cat Anda tetap kecil. Jika Anda melihat banyak cat berukuran besar atau layar penuh, mungkin ada masalah. Kurangi pengubahan ukuran gambar yang tidak perlu karena bahkan satu pengubahan ukuran besar pun dapat membuat Anda kehilangan 60fps.

Jika semuanya berjalan dengan baik, Anda akan membuat pengguna Anda senang dengan pengalaman yang lebih lancar dan mulus, apa pun perangkat yang mereka gunakan. Anda telah melewati batas bebas jank dan dapat memberikan tepukan yang layak bagi diri

Anda sendiri. Untuk mempelajari lebih lanjut tentang mengoptimalkan kinerja cat halaman Anda, kunjungi Jankfree.org dan ingat, jika Anda merasa memiliki masalah kinerja pada perangkat seluler, jangan menebaknya, ujilah!

BAB 7
MERANCANG ANTARMUKA ADAPTIF

Desain, sebagai sebuah konsep, adalah hal yang rumit. Di satu sisi, objek yang dirancang dengan baik harus menarik, namun di sisi lain juga harus mudah dipahami dan bermanfaat. Agar suatu desain berhasil, kedua aspek ini harus seimbang. Ketika estetika mengalahkan kegunaan, karya yang dihasilkan melayani desainer dengan menjadi sarana ekspresi diri bukan konsumen. Demikian pula, ketika sebuah proyek mengorbankan estetika demi kegunaan, pekerjaan yang dihasilkan bisa jadi membosankan, bahkan hanya untuk pejalan kaki. Orang-orang akan menggunakannya, tetapi mereka tidak menyukainya.

Desain yang menyenangkan belum tentu bisa digunakan. Namun perlukah atribut-atribut ini bertentangan? Mengapa bukan kecantikan dan kecerdasan, kesenangan dan kegunaan?

— Don Norman,
Emosi & Desain: Hal-hal yang menarik bekerja lebih baik

Secara etimologis, “desain” berasal dari bahasa Latin abad pertengahan designare, untuk menandai. Secara klasik, ini identik dengan tindakan menunjukkan. Sejak awal, desain lebih dari sekedar estetika; ini tentang mencerahkan konten dan memudahkan konsumen untuk menyelesaikan tugas-tugas utama. Kami merancang untuk memecahkan masalah.

Untuk menciptakan desain yang benar-benar luar biasa, kita tidak hanya harus mengurangi hambatan yang ada dalam menyelesaikan suatu tugas, namun kita harus membuatnya (berani saya katakan) menyenangkan! Kita harus menyelaraskan estetika dengan kegunaan.

Kami Berempati

Desain tidak ada dalam ruang hampa. Ini bukan seni di dinding. Desain dimaksudkan untuk interaksi. Untuk digunakan. Dan siapa penggunanya? Kadang-kadang itu adalah diri kita sendiri, tetapi kecuali kita cukup beruntung untuk menghabiskan sepanjang hari membangun barang-barang yang hanya dimaksudkan untuk konsumsi kita sendiri, kita mungkin membangun untuk orang lain. Sulit mendesain untuk orang lain. Bagaimanapun, kita adalah makhluk kompleks dengan perspektif, tujuan, dan kebutuhan yang unik. Sangat sulit untuk mengesampingkan bias kita sendiri dan mendekati suatu masalah dari sudut pandang orang lain.

Namun, syukurlah, kami sudah terprogram dengan kapasitas untuk melakukan hal tersebut. Sepanjang tahun 1980an dan 90an, Giacomo Rizzolatti dan sekelompok ahli neurofisiologi di Parma, Italia, mempelajari neuron yang mengontrol tindakan tangan dan mulut. Untuk menguji neuron ini, para peneliti akan menempatkan elektroda di korteks premotor ventral monyet dan merekam penembakan neuron individu saat kera meraih kacang dari mangkuk.

Secara kebetulan, seekor kera masih terhubung ke alat perekam ketika seorang asisten peneliti masuk ke ruangan dan mengambil kacang. Yang mengejutkan semua orang, neuron yang sama terpicu ketika kera melihat kacang dipetik dari mangkuk seperti ketika monyet melakukan tindakan yang sama. Melalui apa yang kemudian dikenal sebagai neuron cermin, kera dapat berbagi pengalaman, meski tidak benar-benar mengambil bagian di dalamnya. Itulah akar dari empati.

"Empati" berasal dari bahasa Yunani empatheia yang berarti keadaan emosi dan didefinisikan oleh Merriam-Webster sebagai: Tindakan memahami, menyadari, peka terhadap, dan secara langsung mengalami perasaan, pikiran, dan pengalaman orang lain […] tanpa perasaan, pikiran, dan pengalaman tersebut sepenuhnya dikomunikasikan dengan cara yang obyektif dan eksplisit.

Kami sering mengandalkan riset pengguna untuk membangun dan mempertahankan empati terhadap pelanggan kami. Kami dapat melakukan penelitian ini dengan mengirimkan tim kecil ke lapangan untuk menemui pelanggan kami di tempat mereka tinggal dan bekerja. Atau kami dapat menugaskan satu orang untuk melihat data analitik situs kami untuk mendapatkan wawasan. Atau, baik atau buruk, kita mungkin hanya mengandalkan naluri kita dan berasumsi bahwa kita tahu apa yang diinginkan pengguna.

Terlepas dari bagaimana (atau apakah) kita melakukan penelitian, penting bagi kita untuk tidak pernah melupakan fakta bahwa orang-orang nyata akan menggunakan antarmuka kita. Data memang bagus, tetapi mustahil untuk berempati dengan data.

7.1 PENDAHULUAN

Apa yang Kita Ketahui?

Data, terutama data analitik, bisa memberikan pencerahan, namun juga bisa membawa kita pada asumsi yang salah.

- Skenario: Kami tidak melihat pengguna non-JavaScript di statistik kami.
- Asumsi: Semua pengguna kami memiliki JavaScript, jadi kami tidak perlu mendukung kasus penggunaan non-JavaScript.
- Pertanyaan Tindak Lanjut: Apakah kami memeriksa apakah perangkat lunak analitik kami dikonfigurasi untuk melacak pengguna tanpa mengaktifkan JavaScript? Apakah situs ini berfungsi tanpa JavaScript? Seperti apa pengalamannya saat pengguna menunggu JavaScript dibaca dan dieksekusi? Bagaimana Anda menangani SEO? (Petunjuk: spider pencarian tidak menjalankan JavaScript.)
- Skenario: Kami tidak melihat ada pengguna yang menjelajahi situs kami dengan Blackberry 5.
- Asumsi: Kami tidak memiliki pengguna Blackberry 5 dan dapat berhenti mendukungnya.
- Pertanyaan Lanjutan: Bagaimana tampilan dan fungsi situs pada Blackberry 5? Jika ini pengalaman buruk, apakah Anda akan mempertimbangkan untuk kembali lagi?

Seperti yang Anda lihat, statistik sederhana tidak memberi tahu kita segalanya. Selain itu, dalam kedua skenario ini, pilihan desain dan penerapan awal kita mungkin secara tidak

sengaja membentuk statistik: JavaScript untuk statistik + tanpa JavaScript = tanpa statistik; pengalaman Blackberry 5 yang buruk = Pengguna Blackberry 5 pergi ke tempat lain. Ketika keputusan (strategis atau tidak disengaja) membentuknya, statistik menjadi kurang berguna dan kita berisiko mengusir calon pelanggan. Sebaliknya, kita harus berusaha mengembangkan basis pelanggan kita dengan memperlakukan semua orang sebagaimana kita ingin diperlakukan dengan rasa hormat.

Faktanya adalah ketika pengguna mengunjungi situs kami, kami hanya mengetahui sedikit tentang mereka. Tentu saja, kita bisa membaca string agen pengguna (UA) yang dikirim dalam permintaan dari browser mereka, tapi itu hanya memberi tahu kita banyak hal (bahkan jika itu mengatakan yang sebenarnya. String UA mudah dipalsukan).

String UA tidak dapat memberi tahu kami apakah mereka mengunjungi situs kami dengan Blackberry 5, memiliki Rp.85.000 untuk dibelanjakan pada produk kami, atau Rp.5.000.000. String UA tidak dapat memberi tahu kami jika pengguna tidak memiliki pengetahuan domain atau tingkat pendidikan yang sama dengan kami. String UA tidak dapat memberi tahu kita apakah mereka memiliki penglihatan yang buruk dan perlu memperbesar teks. Senar UA tidak dapat memberi tahu kita apakah lengannya patah, sedang menggerakkan mouse dengan tangan yang tidak dominan, dan tidak seakurat yang seharusnya. String UA tidak dapat memberi tahu kami apakah pengguna telah menginstal plugin browser yang akan membuat kerangka JavaScript kami yang dibuat dengan cermat menjadi tidak berfungsi. String UA tidak dapat memberi tahu kami apakah perangkat dengan kepadatan piksel tinggi sedang digunakan terhubung melalui jaringan seluler dan pengguna mungkin tidak menginginkan foto berukuran 7 MB tersebut.

Ada begitu banyak kemungkinan untuk dipertimbangkan di luar apa yang kami ketahui (atau pikir kami ketahui) tentang pengguna ketika mereka mengunjungi situs kami. Daripada mencoba untuk memastikan dan mengontrol setiap detail, kita harus memanfaatkan sifat Web yang cair dan fleksibel. Kita harus fokus pada membangun pengalaman yang dapat diakses secara universal dan kemudian memanfaatkan peluang yang diberikan oleh berbagai perangkat, platform, dan teknologi untuk meningkatkan pengalaman tersebut. Anda tahu: peningkatan progresif. Peningkatan progresif membuat desain tetap terbuka terhadap kemungkinan keseksian dalam konteks yang tepat, dibandingkan memulai dengan pengalaman “keseluruhan” yang harus dikompromikan.

Bata demi Bata, Baris demi Baris

Kita tidak tahu apa yang akan terjadi di masa depan, tapi kita tahu bahwa apa yang berhasil di masa lalu akan berhasil di masa depan. Itulah janji Web: ramah terhadap masa depan. Merangkul masa lalu tautan nyata, formulir yang dikirimkan ke halaman tindakan, elemen isi yang penuh dengan konten aktual tidak menghalangi kami: hal ini memberi kami landasan yang kuat untuk membangun pengalaman yang lebih menakjubkan.

Ketika orang bersemangat memikirkan antarmuka Web, mereka sering kali tertarik dengan efek animasi mewah atau widget dinamis. Saya tidak punya masalah dengan itu. Sebagai perancang konten, data, piksel, interaksi, atau kode kita harus bersemangat tentang bagaimana kita membantu orang mencapai apa yang ingin mereka lakukan. Namun sering kali,

kita terjebak dalam tren, taktik, dan teknologi hal-hal yang kita anggap keren, menyenangkan, menarik, atau mungkin menarik perhatian industri dan kita melupakan pengguna dan kebutuhan mereka.

Saya kagum dengan betapa seringnya orang-orang di luar disiplin desain berasumsi bahwa apa yang dilakukan desainer hanyalah dekorasi kemungkinan besar karena banyak sekali desain yang buruk hanyalah dekorasi. Desain yang bagus tidak. Desain yang baik adalah pemecahan masalah.

— Jeffrey Veen, Seni dan Ilmu Desain Web

Tidak ada yang salah dengan antarmuka yang apik untuk orang-orang dengan browser terbaru dan tercanggih selama kami mempertimbangkan seperti apa pengalaman bagi seseorang yang tidak memiliki akses ke teknologi tersebut. Sangat menggoda untuk fokus pada pembuatan versi yang mencolok dan kemudian kembali menambalnya agar berfungsi dengan cukup baik di browser yang kurang mampu (degradasi yang baik). Dan jika kita membiarkan statistik menipu kita sehingga percaya bahwa kita tidak perlu khawatir tentang pengguna non-JavaScript atau browser lama tertentu, kita bahkan mungkin akan berhenti sejenak dan berasumsi bahwa siapa pun yang mengunjungi browser tersebut akan mendapatkan pengalaman buruk (atau tidak punya pengalaman sama sekali).

Tentu saja, ini bukan demonstrasi yang baik mengenai empati pengguna. Kita perlu membangun pengalaman yang solid dan universal yang akan bekerja tanpa perlu repot dan kemudian menambahkan bagian-bagian yang mencolok ketika kita tahu browser benar-benar dapat menggunakannya. Mendukung warna RGBa? Luar biasa, mari kita ubah sedikit tampilan dan nuansanya. Punya dukungan isyarat? Bagus, mari tingkatkan antarmuka agar berfungsi dengan gesekan.

Antarmuka yang dibangun dengan cerdas menawarkan pengalaman yang berkelanjutan. Jika Anda belum familiar dengan istilah ini, kontinum adalah kumpulan langkah dari titik A ke titik B yang setiap langkahnya bervariasi satu menit derajatnya. Sebagai contoh sederhana, pertimbangkan kontinum dari kacang tanah hingga kacang M&M:

1. Pertama, ada kacang tanah, camilan yang alami dan lezat.
2. Berikutnya adalah kacang berlapis coklat, yang merupakan peningkatan nyata dari aslinya. Lapisan coklat yang halus dan kaya melengkapi kacang dengan indahnya.
3. Akhirnya paketnya selesai: cangkang permen memberikan tekstur dan sentuhan manis yang melengkapi pengalaman dengan sempurna.

Setiap langkah dalam kontinum ini dari kacang tanah hingga Peanut M&M adalah pilihan camilan yang benar-benar valid, tetapi masing-masing langkah (setidaknya menurut saya) juga merupakan peningkatan yang signifikan dari langkah sebelumnya.

Kita harus berusaha untuk menciptakan antarmuka yang beroperasi seperti ini. Setiap langkah dalam proses membangun antarmuka harus menambah pengalaman. Pada akhirnya, pengguna independen mungkin memiliki pengalaman antarmuka yang berbeda, namun tidak ada yang ditolak aksesnya ke pengalaman yang baik.

Gambar 7.1 Kontinum kacang tanah hingga kacang M&M.

Kue Lapis, Ditinjau Kembali

Ketika pergerakan standar Web masih muda, kita didorong untuk memisahkan kode yang mengendalikan konten, presentasi, dan perilaku kita. Kami melakukan itu untuk membuat situs kami lebih mudah dikelola, halaman kami lebih kecil, dan markup kami lebih bersih. Dan itu berhasil. Faktanya, luar biasa baik. Peningkatan progresif meminta kita mengambil konsep itu selangkah lebih jauh dan mempertimbangkan tiga lapisan kue standar Web yang terdiri dari banyak lapisan pengalaman yang lebih kecil.

Mengapa? Seperti yang pasti Anda ketahui, ekosistem browser sangatlah beragam. Satu browser yang mengakses situs kami saat ini mungkin hanya mendukung sebagian tag HTML4, sementara browser lain mungkin mendukung semua HTML4 dan beberapa HTML5, dan browser ketiga mungkin hanya mendukung HTML4 dan beberapa mikroformat. Yang satu mungkin hanya mendukung segelintir properti CSS2, sementara yang lain mendukung sebagian besar warna CSS2 dan CSS3, dan yang ketiga mendukung semua itu selain animasi CSS3 dan kueri media. Dukungan JavaScript juga tersedia di seluruh peta. Sekilas matriks pengujian Acid 3 Peter Paul Koch untuk WebKit5 mengungkapkan bahwa tidak semua browser dibuat sama, bahkan ketika browser tersebut didasarkan pada mesin rendering dasar yang sama. Tentu saja, dukungan standar secara keseluruhan sekarang jauh lebih baik dibandingkan ketika saya membuat halaman Web pertama saya pada tahun 1996, namun itu tidak berarti semuanya mendukung standar yang sama.

Kami tidak pernah bisa berharap untuk menciptakan (dan menguji) pengalaman unik untuk setiap kombinasi kemampuan ini seperti halnya kami tidak dapat menguji situs web kami di setiap browser dan setiap perangkat yang pernah dibuat. Kami tidak akan pernah tidur dan tidak akan pernah meluncurkan apa pun. Namun kami dapat menganggap pengalaman sebagai sebuah kontinum dan merancang antarmuka kami untuk beradaptasi dengan kemampuan perangkat pengguna kami guna memastikan pengalaman yang positif — meskipun pengalaman tersebut tidak sama.

Seperti yang diingatkan oleh gerakan mobile first, kita perlu fokus pada kasus penggunaan inti untuk sebuah halaman atau antarmuka dan membangun pengalaman dari sana. Dan, saat kami menciptakan pengalaman tersebut, kami perlu memastikan tidak ada keputusan desain atau teknis yang melemahkan pengalaman tersebut. Lapisan-lapisan pengalaman Web pada umumnya dipecah seperti ini:

7.2 INTI

Konten teks dan elemen interaktif dasar yang membentuk landasan yang dapat digunakan secara universal (tautan, formulir, dll.). Copywriting kami (bahkan mikrokopi seperti label dan pesan kesalahan) harus jelas dan sesuai.

1. Semantik

Markup HTML yang menerangi konten pengalaman inti dan menyampaikan arti kata-katanya. Setiap elemen yang kita pilih harus meningkatkan makna semantik konten. Kita tidak boleh memilih elemen yang dapat mengaburkan makna konten kita atau membingungkan pembaca (misalnya menggunakan div, span, atau bahkan jangkar padahal seharusnya berupa tombol). Kami juga harus menerapkan peran penting ARIA agar lebih mudah memandu pengguna kami yang mengandalkan teknologi bantu di seluruh halaman.

2. Desain

CSS yang menetapkan hierarki visual, meningkatkan pengalaman membaca (melalui penggunaan ritme vertikal, panjang garis yang nyaman, dll.), dan memperkuat merek. Setiap keputusan desain yang kami buat harus memperkaya pengalaman dan tidak boleh merusak inti dengan mengurangi akses ke konten atau mengaburkannya melalui penggunaan teks atau visual dengan kontras rendah.

3. Peningkatan interaksi

JavaScript, bersama dengan elemen interaktif berbasis HTML tingkat lanjut (misalnya detail/ringkasan, tipe masukan tingkat lanjut) dan gaya CSS terkait, yang meningkatkan kegunaan antarmuka. Interaksi yang ditingkatkan harus memanusiakan antarmuka dan mengurangi segala hambatan yang melekat dalam menyelesaikan tugas tertentu. JavaScript yang kita tulis dan gunakan harus memanipulasi dokumen untuk memperkenalkan markup dan gaya yang diperlukan untuk menciptakan antarmuka yang disempurnakan, sehingga mengurangi ukuran halaman untuk pengguna non-JavaScript.

Keputusan gaya atau pengkodean apa pun yang kami buat tidak boleh membatasi akses teknologi pendukung ke pengalaman inti jika JavaScript dijalankan. JavaScript kami juga harus memperkenalkan dan memperbarui atribut ARIA yang diperlukan untuk memberikan teknologi bantu dengan detail berguna tentang antarmuka dan apa yang terjadi saat pengguna berinteraksi dengannya.

Setiap lapisan tidak hanya memiliki tujuan teknis, tetapi juga tujuan kemanusiaan. Setiap keputusan yang kami buat, setiap tombol yang kami tekan, memengaruhi pengalaman pengguna kami. Dengan empati sebagai panduan, kemungkinan besar Anda akan mengambil keputusan yang tepat.

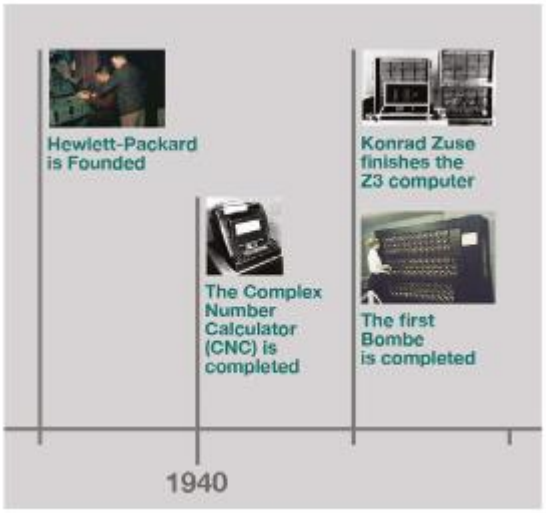
7.3 MEMPERTIMBANGKAN KENDALA

Dunia perangkat yang mendukung Web sangat beragam dan tampaknya semakin beragam setiap harinya. Dengan setiap platform, browser, dan mesin rendering baru yang memiliki matriks fitur dan batasan teknis yang kompleks, upaya untuk menciptakan pengalaman hebat secara universal bisa tampak seperti upaya yang menakutkan. Sekali lagi, memandang antarmuka sebagai tujuan inti membantu menjaga kewarasan kita.

BAGAIMANA CARA BACANYA?

Pertama dan terpenting, kita perlu mempertimbangkan bagaimana sebuah antarmuka dibaca. Pernahkah Anda menggunakan pembaca layar? Bagaimana dengan browser berbasis teks? Pernah membaca email sebagai teks biasa? Ketika kita menghilangkan desain, semantik, dan kemampuan interaktif sebuah antarmuka, yang tersisa hanyalah intinya: teks, yang disempurnakan dengan hyperlink. Jika antarmuka kami berfungsi dalam konteks ini, kami hampir dapat menjamin bahwa antarmuka tersebut dapat digunakan pada hampir semua hal.

Hanya teks mungkin tampak mustahil ketika kita berkonsentrasi pada tampilan konten tingkat lanjut, namun dengan sedikit pertimbangan mendalam pada antarmuka, kita dapat menemukan cara untuk mengeja secara harfiah apa yang disampaikan secara visual. Ambil garis waktu, seperti yang ada di sebelah kanan, misalnya: Apa yang dimaksud dengan garis waktu, namun merupakan daftar peristiwa yang dikelompokkan berdasarkan penanda waktu?



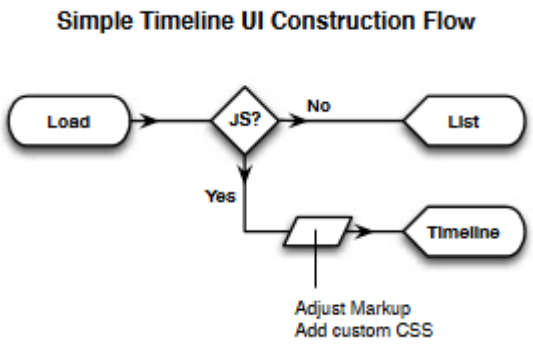
Gambar 7.2 Garis waktu dari Museum Sejarah Komputer.
http://www.computerhistory.org/timeline/

```
<h1>Timeline of Computer History</h1>
<ol>
   <li>
      <h2>1939</h2>
      <ul>
        <li>
          <figure>
             <img src="images/1939_HewlettPackard.jpg" alt=""/>
             <figcapti on>David Packard and Bill Hewlett in their
Palo Alto, California Garage</figcaption>
             </figure>
             <p>Hewlett-Packard is Founded. David Packard and
Bill Hewlett found Hewlett-Packard in a Palo Alto, California
```

```
garage. Their first product was the HP 200A Audio Oscillator,
which rapidly becomes a popular piece of test equipment for
engineers. Walt Disney Pictures ordered eight of the 200B model
to use as sound effects generators for the 1940 movie
"Fantasia."</p>
       </li>
     </ul>
</li>
<li>
      <h2>1940</h2>
       <ul>
          <li>
             <figure>
               <img src="images/1940_complex.jpg" alt=""/>
```

Dalam konteks teks saja, contoh ini cukup mudah dipahami. Selain itu, ia berfungsi cukup baik di layar kecil, di mana harga real estat sangat mahal. Namun, jika kami ingin menyempurnakan timeline dengan JavaScript agar menjadi pengalaman yang lebih menarik, kami dapat menguji untuk melihat apakah pengalaman yang ditingkatkan tersebut masuk akal mengingat keterbatasan browser dan perangkat yang digunakan untuk mengakses konten. Jika kondisinya benar, JavaScript dapat memasukkan markup tambahan, mengatur ulang elemen pada halaman, menyembunyikan konten yang secara visual berlebihan (dengan cara yang mudah diakses), dan menyisipkan serangkaian aturan gaya baru untuk mengatur tampilan dan nuansa antarmuka yang ditingkatkan tersebut.

Saat memikirkan berbagai cara untuk menikmati konten, saya suka membuat sketsa gambar konstruksi UI. Anda dapat menganggap alur konstruksi UI sebagai peta jalan untuk proses pembuatan halaman. Ini menguraikan berbagai potensi pengalaman di sepanjang kontinum dan membantu kita memvisualisasikan dengan lebih baik bagaimana semuanya cocok satu sama lain. Hal ini juga memberi kita kesempatan untuk mengeksplorasi kendala yang perlu kita pertimbangkan untuk antarmuka tersebut. Di sebelah kanan, Anda akan menemukan contoh diagram garis waktu:



Gambar 7.3 Alur Kontruksi UI Konstruksi

Kendala dalam menawarkan pengalaman hanya teks merupakan langkah pertama yang penting untuk memastikan semua orang dapat menggunakan situs web kami, namun ini hanyalah kendala pertama dari tiga kendala yang saya gunakan untuk memandu pekerjaan saya. Kendala kedua yang saya terima adalah jaringan.

Bandwidth. Latensi. Batas koneksi simultan. Masing-masing aspek jaringan ini berpengaruh pada pengalaman memuat dan berinteraksi dengan situs. Hal ini berdampak pada setiap koneksi, namun pada jaringan seluler, dampaknya bahkan lebih besar (kita akan membahasnya sebentar lagi).

Selama bertahun-tahun, pakar kinerja Web telah meminta kita untuk menggabungkan file, membuat peta sprite, mengompresi gambar, dan mengecilkan semuanya. Mengurangi jumlah permintaan ke server membuat perbedaan besar dalam kecepatan pengiriman konten ke browser. Dan tentu saja, file yang lebih kecil akan diunduh lebih cepat. Sebagian besar dari kita ikut-ikutan mengikuti perkembangan kinerja Web dengan cukup cepat. Itu masuk akal. Tapi bisakah kita berbuat lebih banyak?

Jaringan seluler menghadirkan tantangan yang menarik karena jaringan tersebut biasanya memiliki latensi yang tinggi, dan pengguna roaming dapat berpindah dari menara seluler ke menara seluler lainnya, merasakan beragam stabilitas dan ketersediaan layanan. Semakin cepat kami dapat memberikan pengalaman kami kepada pelanggan, semakin besar kemungkinan mereka dapat menggunakannya sebelum mereka kehilangan konektivitas di zona mati atau kehilangan kecepatan dari 4G ke Edge.

Lalu ada masalah uang. Hanya sedikit pelanggan data seluler yang cukup beruntung memiliki paket penggunaan tak terbatas. Sebagian besar pengguna membayar sedikit demi sedikit dan semakin besar situs kami, semakin besar biaya yang mereka keluarkan untuk melihat konten kami. Segala sesuatu yang tidak relevan yang kami kirimkan kepada pelanggan sama saja dengan pajak atas akses terhadap konten atau layanan kami, dan kami bahkan bukan pihak yang memperoleh keuntungan finansial.

Inilah sebabnya mengapa sangat penting bagi kita untuk fokus pada tujuan inti halaman Web atau antarmuka saat kita membangunnya. Tweet Mat Marquis yang sekarang terkenal (lihat di sebelah kanan) menjelaskan semuanya. Saya rasa belum ada orang yang mampu mengartikulasikan dengan lebih jelas ketegangan antara visi departemen pemasaran untuk sebuah halaman dan visi pelanggan. Tentu saja, kami ingin halaman kami terlihat indah, namun ada beberapa hal yang perlu kami waspadai. Pada akhirnya, kita harus berempati terhadap orang-orang yang mengunjungi situs kita sehingga mereka menjadi (dan tetap) pelanggan kita.

Untuk itu, mari kita pertimbangkan situs web surat kabar pada umumnya. Kebanyakan situs surat kabar berisi satu halaman atau lebih yang menampilkan sejumlah artikel secara agregat. Beranda adalah lokasi yang umum, namun halaman arahan bagian surat kabar sering kali berfungsi sama.

Gambar 7.3 Mat "Wilto" Marquis di twitter.com

Masing-masing konteks ini berisi beberapa contoh modul penggoda. Secara universal, modul teaser berisi headline, byline, dan lede. Namun terkadang juga berisi thumbnail kecil untuk menarik perhatian kita atau memberi petunjuk tentang apa yang mungkin kita temukan di halaman artikel lengkap.

Meskipun gambar mini ini menarik secara visual, gambar mini ini hanya melengkapi tujuan inti dari teaser: membuat kita mengeklik dan membaca artikel selengkapnya. Selain itu, gambar jempol-kuku seperti ini menimbulkan masalah tambahan:

1. Gambar yang direferensikan secara eksternal memerlukan pengunduhan tambahan (dan semua overhead terkait jaringan yang menyertainya), sehingga membuat laman membutuhkan waktu lebih lama untuk dirender, sekaligus membebani konsumen lebih banyak uang untuk melihat konten melalui jaringan terukur.
2. Gambar yang disematkan menggunakan URI data biasanya berukuran tiga kali lebih besar dibandingkan gambar binernya, sehingga menghasilkan muatan HTML yang lebih besar yang mungkin lebih cepat untuk diunduh, namun sebenarnya memerlukan biaya yang lebih besar bagi pelanggan jaringan terukur untuk melihatnya dibandingkan gambar eksternal.
3. Dalam tata letak yang lebih sempit, gambar-gambar ini sebenarnya dapat membuat pembacaan menjadi lebih sulit, sehingga menggagalkan seluruh tujuan konten pada awalnya.

Itu adalah beberapa argumen yang cukup kuat untuk menolak adanya thumbnail teaser, bukan?

Gambar 7.4 Bagian "Berita Terbaru" di Nooga.com

Penelitian telah menunjukkan bahwa beberapa gambar, gambar yang relevan dengan daya tarik cerita benar-benar dapat memikat pembaca, jadi kita tidak boleh membuangnya seluruhnya. Sebaliknya, kita harus meneliti konten setiap gambar untuk menentukan apakah gambar tersebut benar-benar menambahkan sesuatu ke teaser. Setelah kita memisahkan hal-hal yang bermanfaat dan yang tidak berguna, kita dapat berkonsentrasi untuk mengatasi masalah-masalah di atas.

Jika keterbacaan adalah perhatian utama kami, kami mungkin tergoda untuk menautkan ke gambar dengan cara tradisional dan menyembunyikannya dengan CSS saat browser berukuran kurang dari dua kali lebar thumbnail. Masalah dengan strategi tersebut adalah browser akan tetap mendownload gambar meskipun gambar tersebut tidak ditampilkan. Itu sangat tidak menghargai waktu dan uang pelanggan kami.

Strategi yang lebih baik untuk digunakan disebut pemuatan lambat. Anda mungkin akrab dengan pemuatan lambat sebagai sebuah konsep dari dunia JavaScript, di mana kode tambahan dimuat ke dalam halaman Web sesuai kebutuhan. Namun tidak ada alasan untuk menggunakan teknik itu untuk JavaScript. Kita dapat memuat secara lambat apa pun yang merupakan tambahan dari tujuan inti halaman: gambar, HTML, CSS, dll. Faktanya, inilah yang banyak situs berita (misalnya Guardian, Boston Globe, dan BBC) mulai melakukannya melakukan. Terkait dengan pemuatan gambar yang lambat, preferensi saya saat ini adalah mengambil paragraf kosong dan memberinya atribut data-* yang menunjuk ke gambar yang mungkin ingin saya muat jika kondisinya tampak benar:

```
<p data-image-src="/path/to/my.jpg"></p>
```

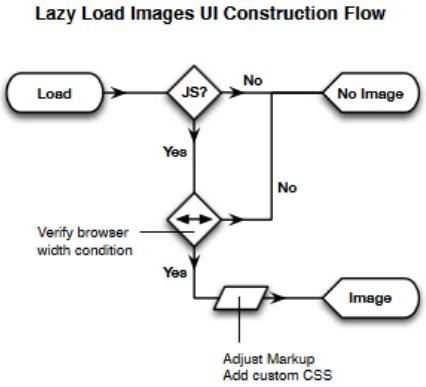
Saya menggunakan beberapa CSS sederhana untuk menyembunyikan paragraf ini secara default:

```
[data-image-src] {
```

```
            display: none;
    }
```

Lalu saya menggunakan JavaScript untuk menguji beberapa kondisi dan memutuskan apa yang harus dilakukan. Alur konstruksi UI untuk pendekatan ini terlihat seperti gambar di bawah:

Jelasnya, tanpa JavaScript, skrip tidak akan berjalan dan paragraf hanya diam di sana, tidak terlihat. Tentu saja, ini sedikit markup tambahan untuk diunduh, tetapi itu mengalahkan gambar tersembunyi atau (lebih buruk lagi) gambar tersemat yang tersembunyi.



Gambar 7.4 Arus Kontruksi UI Lazy Load Image

Jika JavaScript tersedia, skrip akan dijalankan dan dapat menguji hal-hal seperti lebar browser, kueri media yang saat ini digunakan, kecepatan jaringan, dan bahkan apakah pengguna menggunakan koneksi terukur atau tidak. Kegunaan dan keandalan pengujian terkait jaringan saat ini masih diragukan, namun saya berharap pengujian tersebut dapat mencapai tujuan tersebut di masa mendatang.

Demi kesederhanaan, katakanlah kita mengetahui gambar mini berukuran 200pX persegi dan kita ingin memuatnya jika jendela browser setidaknya dua kali lebih lebar (400pX). JavaScript untuk itu akan terlihat seperti ini (komentar dalam kode ada untuk membantu Anda memahami apa yang terjadi):

```
// self-executing function
(function(){
        var
        // set the threshold
        threshold = 400,
        // collect the window width
        browser_width = window.innerWidth ||
                        document.body.offsetWidth,
        // prototype image for reuse
        image = document.createElement('img'),
```

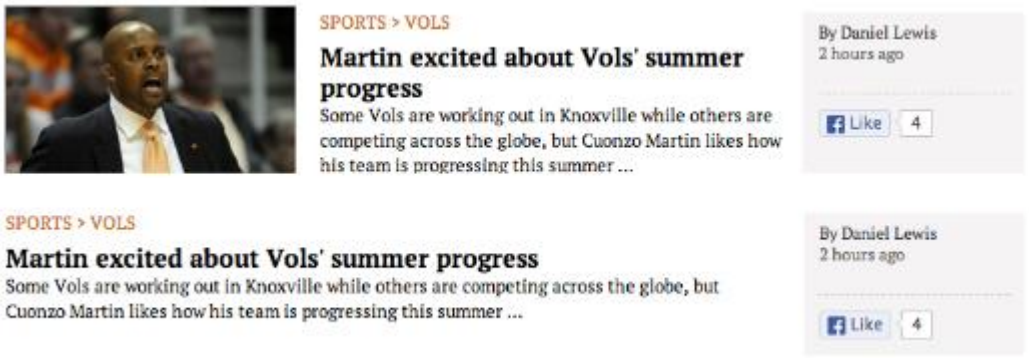
```
        // get all paragraphs
        paragraphs = document.getElementsByTagName('p'),
        // count the paragraphs
        i = paragraphs.length,
        // instantiate loop vars
        p, src, img;

        // are we over the threshold
        if ( browser_width >= threshold )
        {
                // make sure we have an empty alt for
                accessibility
                image.setAttribute('alt',");

                // reverse looping through the paragraphs is
                faster
                while ( i-- )
                {

                   // reference the paragraph
                   p = paragraphs[i];
                   // collect the image path
                   src = p.getAttribute('data-image-src');
                   // do we have a path?
                   if ( src != null )
                   {
                      // clone the prototype image
                      img = image.cloneNode(true);
                      // set the source
                      img.setAttribute('src',src);
                      // append it to the paragraph
                      p.appendChild( img );
                      // flag the paragraph as having an image
                      p.setAttribute('data-image-loaded',");
         }

     }

     // release DOM references
      image = paragraphs = p = img = null;
   }
}());
```

Itu menangani pemuatan awal halaman. Pada layar yang sempit (<400pX), thumbnail tidak akan dimuat, namun pada layar yang lebih lebar, thumbnail akan diminta secara dinamis dan ditambahkan ke halaman.

```
<p data-image-src="/path/to/my.jpg" data-image-loaded>
   <img src="/path/to/my.jpg" alt=""/>
</p>
```

Paragrafnya sendiri ditampilkan dengan memasukkan atribut data-image-loaded yang ditetapkan secara dinamis:

```
[data-image-src][data-image-loaded] {
   display: block;
}
```



Gambar 7.5 Membandingkan potensi pengalaman JavaScript dan pengalaman tanpa JavaScript pada teaser berita.

Tentu saja, perangkat seluler dapat dipegang dalam orientasi lanskap atau potret dan pengguna sering kali beralih untuk menyesuaikan tugas yang ingin mereka selesaikan. Untuk memberikan pengalaman membaca terbaik, sekaligus menunjukkan empati, kita harus benar-benar mengambil antarmuka satu langkah lebih jauh dan membiarkannya beradaptasi. Kita dapat melakukannya dengan mendengarkan perubahan ukuran jendela menggunakan suatu fungsi. Inilah yang saya gunakan saat ini:

```
window.watchResize = function(callback)
{
   // used to track the timer
   var resizing;
   // this runs when resizing has stopped
   function done()
   {
     // stop the timeout
```

```
      clearTimeout( resizing );
      resizing = null;
      // run the callback
      callback();
    }
    // track the resize event
    window.onresize = function(){
      // if we are currently resizing, clear the timeout
      if ( resizing )
      {
        clearTimeout( resizing );
        resizing = null;
      }
          // set the done function to execute when the resize
          completes
          resizing = setTimeout( done, 50 );
      };
      // run the callback once
      callback();
 };
```

Dengan fungsi seperti watchResize, pelacakan perubahan ukuran browser menjadi mudah sehingga kita dapat memperbarui antarmuka dengan cara yang tidak bisa dilakukan oleh CSS saja.

Untuk memperkenalkan fungsionalitas yang lebih dinamis ini ke skrip sebelumnya, kita akan mulai dengan memindahkan variabel browser_size ke cakupan global sehingga tersedia untuk skrip UI adaptif lainnya yang mungkin ingin kita tulis. Kami akan memperbaruinya secara real time menggunakan watchResize:

```
// watch browser width on resize
var browser_width = 0;
window.watchResize(function(){
   browser_width = window.innerWidth ||
   document.body.offsetWidth;
});
```

Dengan browser_width yang diperbarui secara langsung, kita dapat meninjau kembali skrip asli dan membuatnya lebih adaptif dengan memeriksa lebarnya saat orientasi berubah:

```
// lazy-load images
window.watchResize(function(){
   var
   threshold = 400,
   image = document.createElement('img'),
```

```
        paragraphs = document.getElementsByTagName('p'),
        i = paragraphs.length,
        p, loaded, src, img;
        if ( browser_width >= threshold )
        {
           image.setAttribute('alt',");
           while ( i-- )
           {
             p = paragraphs[i];
              src = p.getAttribute('data-image-src');
              // check to see if the image is already loaded
              loaded = p.getAttribute('data-image-loaded');
              // Do we have a path?
              // Is the image already loaded?
              if ( src != null &&
                   loaded == null )
              {
                  img = image.cloneNode(true);
                   img.setAttribute('src',src);
                   p.appendChild( img );
                   p.setAttribute('data-image-loaded',");
                   }
                   image = paragraphs = p = img = null;
            }
         });
```

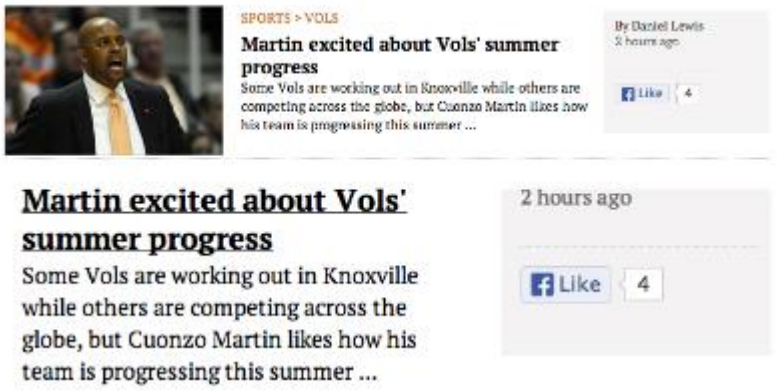
Perbedaan utama antara versi ini dan versi sebelumnya adalah:

1. Fungsi tersebut sekarang diteruskan ke watchResize sebagai callback.
2. Kami tidak lagi menilai browser_width dalam cakupan fungsinya.
3. Kami memeriksa atribut data-image-loaded untuk melihat apakah gambar telah dimuat untuk paragraf tertentu sehingga kami tidak memuat gambar dua kali jika pengguna terus-menerus mengubah orientasi layar.

Terakhir, kita dapat memindahkan kumpulan aturan CSS tampilan ke dalam kueri media untuk memastikan gambar apa pun yang dimuat dengan lambat saat perangkat berada dalam orientasi yang lebih luas (misalnya lanskap) tidak ditampilkan ketika layar terlalu sempit (misalnya potret):

```
@media only screen and (min-width:400px) {
   [data-img-src][data-image-loaded] {
      display: block;
   }
}
```

Seperti yang Anda lihat, dengan berempati terhadap pengguna kami dan menyadari harga yang mereka bayar untuk mengakses konten kami, kami dapat menciptakan pengalaman adaptif yang luar biasa bagi mereka tanpa banyak usaha ekstra.



Gambar 7.6 Membandingkan pengalaman yang sesuai dengan orientasi potensial dari penggoda berita.

Tanpa Javascript, Tanpa Masalah

JavaScript dapat melakukan beberapa hal bagus, namun merencanakan ketidakhadirannya juga sama pentingnya. Bagaimanapun, ketersediaan JavaScript tidak dijamin di setiap browser. Meskipun JavaScript didukung oleh browser, ada banyak keadaan di mana JavaScript yang kami buat dengan cermat mungkin tidak dapat dijalankan dan hal tersebut sepenuhnya berada di luar kendali kami. Misalnya:

- Pengiriman kode JavaScript mungkin diblokir oleh firewall yang berlebihan.
- JavaScript mungkin dinonaktifkan oleh pengguna baik secara manual atau melalui plugin dalam upaya menghentikan iklan yang mengganggu, pop-over, dan sejenisnya.
- Pustaka JavaScript pihak ketiga yang kami sertakan dan berfungsi kemarin telah ditingkatkan dan menimbulkan bug.
- Pengguna memasang add-on browser yang mengandung kesalahan atau menimbulkan konflik dengan kode kami.
- Pengguna masih menunggu aset laman selesai diunduh dan JavaScript kami tidak dapat dijalankan hingga mereka selesai mengunduhnya.

Saya menyukai JavaScript, tetapi saya juga menyadari bahwa JavaScript cukup rapuh. Tidak seperti penulisan kode yang akan dieksekusi di server kami, yang kami kendalikan, kode JavaScript dieksekusi di tempat dan pada sistem yang tidak dapat kami kendalikan dan kami tidak dapat menjamin keberhasilan eksekusinya. Jika kode JavaScript saya gagal dijalankan, saya tidak ingin menjadi orang yang menerima panggilan di tengah malam karena ada yang rusak. Saya ingin situs web, aplikasi Web, atau jenis Web lainnya yang saya buat harus kuat. Saya ingin tetap berjalan seperti Chrysler Imperial dalam derby pembongkaran13.

Seperti yang Anda lihat pada contoh pemuatan lambat sebelumnya, saya menyiapkan status istirahat (alias tanpa JavaScript) sehingga informasi yang diperlukan tersedia agar JavaScript dapat bertindak tanpa menimbulkan masalah bagi pengguna yang tidak mengaktifkan JavaScript. Namun ketika JavaScript tersedia dan kode saya dijalankan, halaman tersebut dimanipulasi untuk menciptakan pengalaman terbaik bagi pembaca. Pendekatan terhadap JavaScript ini sering disebut tidak mengganggu karena tidak menghalangi. Di satu sisi, JavaScript tidak diperlukan untuk mengakses pengalaman inti, namun di sisi lain, bit markup terkait JavaScript tidak dimasukkan ke dalam konten dengan cara yang mengganggu atau membingungkan.

Thumbnailnya adalah contoh yang cukup sederhana, tapi izinkan saya memberi Anda satu lagi: antarmuka bertab. Ini adalah gadget yang cukup umum yang terdiri dari serangkaian panel konten yang ditampilkan satu per satu saat tab terkait diaktifkan.

Gambar 7.7 Resep sederhana ditampilkan sebagai antarmuka tab.

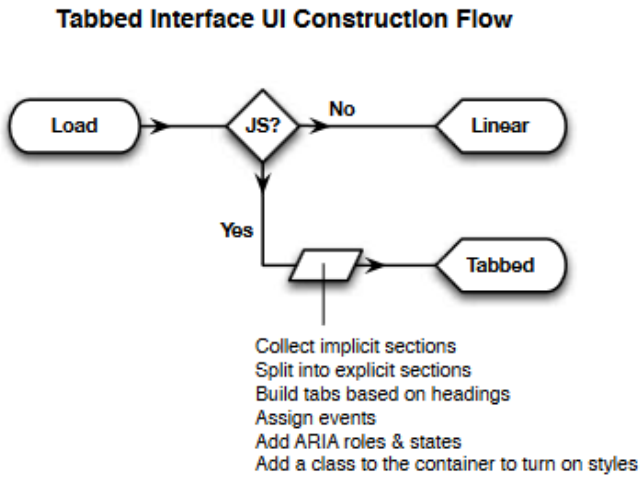
Gambar 7.8 Resep yang sama, dilinearisasi.

Jika dilihat dari sudut pandang semantik, panel konten dapat ditempatkan di elemen bagian terpisah dengan tab sebagai item daftar dalam daftar yang diurutkan. Cukup sederhana dan lugas, bukan?

Bagaimana jika tidak ada JavaScript yang membuatnya berfungsi sebagai antarmuka tab? Dalam hal ini, tampilannya tidak akan terlihat seperti antarmuka bertab, karena akan membuat sebagian besar konten tidak dapat diakses secara visual, dan ini tidak terlalu diperhatikan. Tanpa JavaScript, daftar tautan dan bagian-bagiannya juga tidak terlalu diperlukan; mereka tidak menambah banyak makna pada halaman dan hanya markup tambahan untuk diunduh. Akan lebih baik jika menyimpan bit-bit tersebut di tempat yang benar-benar berguna, untuk mikroformat dan sejenisnya.

Terlepas dari segala kekurangannya, yang kita miliki dalam contoh ini adalah resep yang terdiri dari beberapa bagian berjudul. Jika Anda seperti saya dan sangat kutu buku tentang semantik, Anda mungkin sudah menyadari bahwa judul tersebut harus berupa judul (katakanlah h2s) dan tingkat judul membuat kerangka dokumen alami. (Jika Anda tidak mengerti apa yang saya bicarakan, tekan “Lihat Garis Besar Dokumen” di Toolbar Pengembang Web15 untuk melihat garis besar halaman Web mana pun.)

Hal hebat tentang kerangka dokumen adalah ia menyiratkan bagian-bagian. Mengetahui hal tersebut, kita dapat menulis JavaScript sederhana untuk mengurai konten tersebut, secara dinamis membangun markup tambahan untuk antarmuka bertab, memicu penyertaan beberapa aturan dasar CSS untuk menatanya, dan menambahkan beberapa peningkatan aksesibilitas melalui ARIA ketika JavaScript digunakan. tersedia.



Gambar 7.9 Membuat diagram proses pembuatan antarmuka tab menggunakan alur konstruksi UI.

Saya tidak akan melelahkan Anda dengan kode untuk melakukan ini, tetapi Anda dapat melihat skrip yang telah saya kembangkan untuk tujuan ini di Github16 jika Anda tertarik.Jadi sekarang kami memiliki antarmuka yang beradaptasi dengan baik berdasarkan ketersediaan JavaScript; namun antarmuka bertab tidak selalu berfungsi dengan baik pada layar sempit,

terutama jika tabnya tidak pas. Untuk benar-benar menyebut antarmuka tab ini adaptif, kita perlu mempertimbangkan pengalaman dalam berbagai faktor bentuk.

Mengambil pelajaran dari contoh gambar yang dimuat dengan lambat yang baru saja kita diskusikan, kita dapat menyesuaikan kode JavaScript untuk menyertakan pengujian untuk melihat apakah header bagian akan pas secara horizontal ketika dirender sebagai tab. Jika cocok, skrip dapat melanjutkan dan membangun antarmuka lainnya. Jika tidak, skrip dapat berhenti di tempatnya dan membiarkan konten linier apa adanya. Pengujian ini bahkan dapat berupa pemeriksaan langsung (menggunakan fungsi seperti watchResize) yang memicu pembuatan dan penghancuran komponen antarmuka seiring dengan perubahan ukuran browser atau orientasi perangkat.

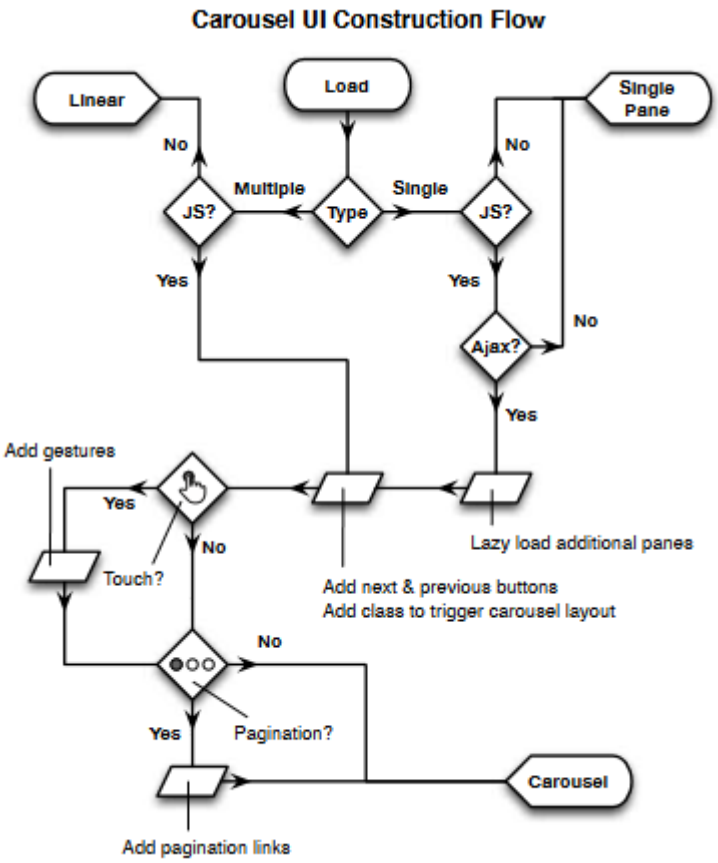
Sekali lagi, dengan menempatkan diri pada posisi pengguna, kami dapat menemukan bahwa ada banyak cara yang bermakna untuk berinteraksi dengan konten kami. Dan menempatkannya dalam satu kontinum dapat menciptakan antarmuka adaptif yang mengesankan.

7.4 BLOK BANGUNAN

Meskipun masalah tersebut tampak menakutkan pada awalnya, masalah yang kompleks selalu dapat dipecah menjadi masalah yang lebih sederhana. Hal yang sama berlaku untuk halaman Web yang kompleks dan antarmuka yang kompleks. Sangat mudah bagi saya untuk merasa kewalahan ketika pertama kali disajikan dengan antarmuka yang rumit. Namun, ketika saya mulai memilahnya menjadi komponen halaman yang lebih kecil dan interaksi terpisah, saya mulai bernapas sedikit lebih mudah.

Komponen sangat bagus karena dapat dibuat dan diuji secara individual, namun dapat dikombinasikan dengan berbagai cara untuk memenuhi tantangan persyaratan antarmuka yang paling rumit sekalipun. Seringkali tim akan mengatur komponen-komponen ini (atau pola desain) ke dalam katalog yang disebut perpustakaan pola. Setiap komponen dalam pustaka pola ada secara terpisah, dengan persyaratan, kemampuan, dan dokumentasinya sendiri. Hal ini memungkinkannya untuk diulang tanpa memengaruhi elemen UI lain yang digunakan di situs web atau di seluruh rangkaian properti Web.

Bagi saya, titik awal untuk setiap komponen adaptif adalah alur konstruksi UI yang telah saya sebutkan beberapa kali. Sebagian besar berupa sketsa sederhana pada selembar kertas atau papan tulis yang mengilustrasikan status konten yang berbeda dan dalam keadaan apa terjadi perubahan dalam metode presentasi atau interaksi.



Gambar 7.10 Alur konstruksi UI bisa menjadi sangat rumit, seperti yang ditunjukkan oleh contoh carousel ini.

Sederhananya, kami memiliki contoh seperti contoh gambar yang memuat lambat. Pada akhir yang kompleks, kita bisa mendapatkan aliran yang luas untuk sebuah carousel. Menurut saya alur konstruksi UI tidak hanya membantu saya mengatur pemikiran saya seputar berbagai cara untuk menyajikan dan berinteraksi dengan konten, namun juga membantu saya mengomunikasikan ide-ide saya kepada orang lain.

7.5 MEMBUAT KOMPONEN ADAPTIF

Menurut pengalaman saya, pengembangan berbasis komponen dapat dilakukan dengan baik jika dilakukan secara terpisah, namun akan lebih baik jika dilakukan dalam tim kecil, terintegrasi, dan kolaboratif. Anda tidak perlu mengikuti Lean UX atau Agile, tetapi memiliki keragaman perspektif pada antarmuka sangat membantu. Ini sangat membantu ketika anggota tim yang berbeda memiliki bidang keahlian yang berbeda. Manajer produk, ahli strategi konten, desainer UX, desainer visual, insinyur front-end, dan pengembang back-end semuanya menghadirkan ide-ide yang berbeda namun berharga.

Setiap orang tidak hanya akan mampu menjelaskan dampak positif atau negatif dari setiap keputusan terhadap bidang perhatian utamanya, namun masing-masing orang mungkin akan berempati dengan pengguna kami dengan cara yang sedikit berbeda. Di kantor saya, alur konstruksi UI umumnya merupakan langkah pertama dengan ketelitian rendah yang

kami gunakan untuk mengatur pemikiran kami dan memahami paradigma interaksi yang berbeda. Dari sana, kita beralih ke sketsa kasar.

Tim kami cenderung membuat sketsa di atas kertas atau papan tulis, namun Anda mungkin lebih nyaman menggunakan OmniGraffle, Photoshop, Illustrator, atau salah satu dari banyak alat wireframing dan prototyping yang tersedia. Gunakan apa pun yang membuat Anda nyaman dan efisien, tetapi berhati-hatilah agar tidak terpaku pada detail. Sketsa adalah cara berisiko rendah untuk menyempurnakan pengalaman dalam alur konstruksi UI.

Selanjutnya, kami mulai membuat prototipe dalam HTML, CSS, dan JavaScript, merujuk pada spesifikasi fidelitas rendah yang kami susun dalam alur konstruksi UI dan sketsa kami. Terkadang, kami akan menghentikan beberapa fungsi back-end jika kami perlu mereferensikan API atau semacamnya, namun sering kali kami memalsukannya dengan file JSON statis.

Kami menggunakan prototipe untuk melihat apakah ide yang kami hasilkan benar-benar masuk akal dan berjalan sesuai harapan. Jika ada sesuatu yang berantakan dalam konteks tertentu atau tidak berfungsi dalam faktor bentuk tertentu, kami berkumpul kembali dan mencoba sesuatu yang lain. Kami melakukan iterasi pada prototipe hingga kami secara umum puas dengan perilaku yang berbeda dan kemudian kami mulai menyempurnakan desainnya, menyesuaikan kode untuk kinerjanya, dan menghubungkannya ke fungsionalitas back-end jika diperlukan.Hasilnya adalah komponen adaptif yang hidup dan bernafas yang dapat ditambahkan ke perpustakaan pola dan dimasukkan ke dalam konteks apa pun yang kita perlukan.

Prasmanan Interaksi Yang Benar-Benar

Sebagai sebuah alat, perpustakaan pola telah bersama kami selama beberapa waktu. Sebagian besar perpustakaan JavaScript memiliki pola UI mereka sendiri (misalnya YUI, jQuery UI, Dojo Digits) dan beberapa kerangka UI mandiri seperti Bootstrap dan Foundation telah menangkap imajinasi populer akhir-akhir ini. Namun saya sangat yakin bahwa satu ukuran tidak cocok untuk semua. Jangan salah paham, menurut saya alat seperti Bootstrap bisa sangat membantu untuk membuat prototipe dan menguji interaksi, tapi saya benci gagasan menerapkan situs yang sepenuhnya dibangun di atasnya. Saya merasakan hal ini karena berbagai alasan, beberapa di antaranya akan saya rangkum di sini:

verbositas

Dalam upaya untuk membuat kerangka UI universal, kode di balik layar sering kali membengkak atau mengharuskan kita menggembungkan kode agar dapat menggunakan sistem grid atau widgetnya.

1. Kelebihan

Dalam kebanyakan kasus, kerangka UI (atau bahkan perpustakaan JavaScript) mencakup lebih banyak opsi dan komponen daripada yang sebenarnya kita perlukan untuk situs kita. Dan, jika kami tidak menghapus CSS dan JavaScript yang tidak digunakan, hal ini dapat menimbulkan masalah kinerja bagi pelanggan kami.

2. Filsafat

Membeli kerangka UI dengan sepenuh hati akan mengunci kita pada cara berpikir dan membangun yang sempit, yang mungkin tidak sejalan dengan cara berpikir kita atau cara tim kita beroperasi. Misalnya, Bootstrap dibuat agar responsif, namun hadir dari perspektif desktop-first, bukan mobile-first.

3. Estetika

Kerangka kerja UI memiliki estetika tertentu dan terkadang sulit untuk menghindarinya. Kegagalan untuk keluar dari cetakan UI stok berarti situs kami terlihat hampir sama dengan setiap situs lain di luar sana yang menggunakan kerangka tersebut.

Mungkin terdengar seperti saya tidak menyukai kerangka UI, namun bukan itu masalahnya. Meskipun saya tidak menganjurkan penggunaannya di situs publik, menurut saya hal tersebut memberikan peluang pembelajaran yang berharga. Hal yang sama berlaku untuk perpustakaan pola lain yang tersedia untuk umum.

Melihat apa yang telah dilakukan orang lain membantu memastikan kami tidak melewatkan apa pun saat kami mengerjakan perpustakaan pola kami sendiri perpustakaan yang secara langsung memenuhi kebutuhan perusahaan atau klien kami. Dalam diskusinya mengenai hasil yang responsif, Dave Rupert mengutarakan poin berikut: “Hasil yang responsif harus terlihat seperti sistem Twitter Bootstrap yang berfungsi penuh dan dirancang khusus untuk kebutuhan klien Anda.”

Gambar 7.11 PatternLab milik Brad Frost adalah alat luar biasa untuk membantu membuat dan memelihara pustaka pola dan kemudian menggabungkan pola-pola tersebut ke dalam templat halaman: http://demo.pattern-lab.info/

Pustaka pola adalah anugerah bagi semua orang di tim, mulai dari manajemen tingkat atas hingga halaman-halaman yang menggerutu. Mereka menyediakan katalog komponen yang konsisten yang dapat diambil dan dimasukkan ke dalam mock-up atau prototipe. Mereka

bertindak sebagai clearinghouse untuk cuplikan kode dan dokumentasi untuk membantu para insinyur front-end bekerja lebih cepat dan efisien. Dan mereka memungkinkan tim QA untuk menguji komponen-komponen terpisah daripada harus mendiagnosis masalah ketika komponen-komponen ini saling terkait.

Pustaka pola juga merupakan keuntungan bagi pengguna. Ketika sebuah situs web dibangun menggunakan perpustakaan pola, situs tersebut akan memiliki konsistensi dan predikabilitas yang membuatnya lebih mudah bagi mereka untuk melakukan apa yang perlu mereka lakukan. Mereka tidak perlu menghabiskan waktu untuk memikirkan cara mengisi formulir tertentu atau berinteraksi dengan widget tertentu karena setiap contoh sudah familiar bagi mereka. Ini meyakinkan, selain itu, pustaka pola mempermudah semua orang di tim pengembang untuk mengetahui informasi terbaru saat komponen ditingkatkan atau API diubah, karena ini adalah pusat penyimpanan informasi bagi semua orang yang bekerja di situs web. Bagaimanapun, perpustakaan pola harus berkembang seiring waktu. Ini harus menjadi dokumen hidup dari berbagai modul yang berkembang secara independen. Hal ini tidak boleh dibiarkan begitu saja seperti sebuah monolit, namun harus dibentuk dan dibentuk seiring dengan pemahaman kita yang lebih baik terhadap media kita, Web, dan interaksi pengguna kita dengannya.

Masa Depan Berantakan, Rangkullah

Kita hidup di masa yang sangat menarik. Tampaknya setiap hari perangkat baru muncul, browser baru, dan model interaksi baru. Sulit untuk melacak semuanya dan membingungkan membayangkan menciptakan pengalaman tersendiri untuk setiap hal baru yang muncul. Seperti mengejar ukuran layar: itu tugas yang bodoh. Pada titik tertentu kita perlu melepaskan dan fokus pada hal yang benar-benar penting: manusia.

Kami merancang, konten, alur pengguna, interaksi, antarmuka, API, dan pengalaman, untuk membantu orang melakukan apa yang perlu mereka lakukan secepat dan seefisien mungkin. Kami bekerja keras sehingga mereka tidak perlu melakukannya. Kami di sini untuk memecahkan masalah. Untuk melakukan itu, kita perlu menjadi satu dengan pelanggan kita. Kita perlu berempati terhadap mereka dan merasakan perjuangan mereka seolah-olah itu adalah perjuangan kita sendiri.

Dengan wawasan itu, kita bisa menciptakan pengalaman luar biasa bagi mereka. Kami akan mencapai keseimbangan sempurna antara estetika dan kegunaan. Kami akan membangun pengalaman dari inti, mengikuti filosofi peningkatan progresif. Kami akan membuat pekerjaan kami lebih konsisten dan fleksibel dengan memanfaatkan perpustakaan pola. Dan, apa pun perangkat yang diberikan perusahaan kepada kami, kami yakin pelanggan kami akan dilayani dengan baik.

BAB 8
MEMPERBAIKI TEKNIK BACK-END WEB DAN RAHASIA TERMINAL

Bayangkan Anda bangun di suatu pagi, meraih laptop Anda dengan grogi dan menyalakannya. Anda baru saja selesai mengembangkan situs web baru dan tadi malam Anda dengan bangga mengklik daftar produk. Jendela browser masih terbuka, Widget 3000 masih berkilau dalam kebaruan AJAXy-nya. Anda menyeringai seperti orang tua baru dan dengan penuh harap mengklik “Detail lebih lanjut”. Dan tidak ada yang terjadi. Anda klik lagi, tetap tidak ada. Anda menekan Refresh dan mendapatkan ikon berputar-putar yang mengganggu dan kemudian halaman menjadi kosong. Membantu! Internet hilang!

Kita akan membahas banyak hal dalam bab ini, mulai dari router hingga server, dari log kesalahan hingga peretasan PHP. Saya akan mulai dengan skenario terburuk dan mendalaminya, menjelajahi infrastruktur Internet dan pembuatan server Web, memberikan banyak tip dan perintah kecil sepanjang prosesnya, membuka perspektif baru tentang bagaimana situs web dapat berhenti beroperasi. bekerja dan diperbaiki.

Akhir dunia

Kecil kemungkinan peradaban akan runtuh dalam semalam, terutama jika Anda mudah terbangun. Anda dapat memverifikasi ini dengan radio bertenaga baterai. Sebuah kiamat pasti akan menjadi berita dan mungkin memenuhi syarat untuk peringatan penuh dari pemerintah. Semua stasiun harus melaporkannya, dengan asumsi masih ada yang tersisa; itu akan menjadi berita yang sangat buruk jika tidak.

Di AS, banyak lembaga penyiaran berpartisipasi dalam Sistem Peringatan Darurat, yang secara teori memungkinkan Presiden menyampaikan pidato nasional dalam waktu 10 menit, meskipun mungkin rentan terhadap peretas. Prancis menggunakan sirene serangan udara untuk Signal National d'Alerte dan J Jepang -Peringatan menggunakan pengeras suara. Semuanya merupakan bagian dari Program Peringatan Dini Internasional PBB. Hal ini penting, terutama saat ini setelah Dekade Internasional untuk Pengurangan Bencana Alam (tahun 1990an) telah berakhir.

Anda seharusnya dapat memastikan dengan cepat apakah masalah situs web Anda terkait dengan hal ini. Jika tidak, lanjutkan ke bagian berikutnya.

8.1 INFRASTRUKTUR

Internet bergantung pada listrik. Perusahaan hosting Anda mungkin memiliki catu daya tak terputus (UPS) yang akan langsung mengambil alih jika listrik padam. Ini dapat memberikan daya ke server Web Anda selama beberapa menit, cukup lama untuk memiliki generator diesel yang siap mengambil alih. Peralatan jaringan utama yang menghubungkan server Web Anda ke Internet mungkin juga dilindungi dengan UPS dan generator. Dan laptop Anda akan bertahan beberapa jam lagi jika terisi penuh.

Namun router nirkabel Anda akan berhenti berfungsi. Itu adalah mata rantai terlemah. Anda dapat menyiasatinya dengan mengecek Internet melalui ponsel cerdas Anda, yang seharusnya berfungsi selama menara terdekat Anda memiliki daya cadangan, dan ada jalur Internet melalui menara lain yang berfungsi. Namun hal ini mungkin berjalan sangat lambat, terutama jika semua orang di lingkungan Anda juga mempunyai gagasan yang sama. Jika situs web Anda masih hilang, maka masalahnya lebih bersifat pribadi.

Jaringan

Pemadaman listrik bukan satu-satunya hal yang mengganggu router broadband. Mereka mempunyai banyak cara untuk mengatasi kegagalan, seperti halnya semua peralatan jaringan lain antara Anda dan situs web Anda, dan semua kabel tembaga dan serat optik di antaranya.

Untuk menjelajahi masalah jaringan, Anda perlu menjalankan beberapa perintah. Sebagian besar informasi ini juga tersedia melalui berbagai menu, namun baris perintah memberi Anda lebih banyak data dengan lebih cepat. Di Mac OS X, buka Aplikasi → Utilitas dan jalankan Terminal. Di Ubuntu LinuX, terminal berada di bawah Aplikasi → Aksesori. Di Windows, buka Mulai → Semua Program → Aksesori dan pilih Command Prompt.

Alamat Ip Anda

Setiap komputer yang terhubung ke Internet memiliki alamat IP (Protokol Internet) numerik. Untuk mengetahuinya, jalankan perintah ifconfig di Mac dan LinuX dan ipconfig /all di Windows:

```
$ ifconfig
eth0            Link encap:Ethernet HWaddr 00:10:dc:75:d9:5b
                inet      addr:192.168.0.11      Bcast:192.168.0.255
                Mask:255.255.255.0
                inet6 addr: fe80::210:dcff:fe75:d95b/64 Scope:Link
                UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1...
lo              Link   encap:Local   Loopback   inet   addr:127.0.0.1
                Mask:255.0.0.0  inet6  addr:  ::1/128  Scope:Host  UP
                LOOPBACK RUNNING MTU:16436 Metric:1...
```

Artinya komputer (LinuX dalam hal ini) memiliki dua antarmuka jaringan. Eth0 adalah yang berkomunikasi dengan Internet melalui kabel. Jika komputer ini memiliki nirkabel, juga akan ada antarmuka eth1 atau wlan0. Antarmuka loopback lo digunakan sebagai jalan pintas untuk berkomunikasi dengan dirinya sendiri. Setiap antarmuka memiliki alamat IP di jaringan lokal. Baris penting di sini adalah inet addr:192.168.0.11. Ini memberikan alamat IP komputer. Jika Anda memasang kabel dan mengaktifkan nirkabel, Anda mungkin memiliki dua antarmuka aktif dan dua alamat IP, namun biasanya hanya satu. Mac cenderung menyebut antarmuka ini en0 dan en1.

Windows lebih bertele-tele dan menggunakan nama seksi seperti "Koneksi Area Lokal adaptor Ethernet".

DHCP

Bagaimana komputer Anda mengetahui alamat IP-nya? Khususnya di jaringan rumah atau nirkabel, Anda tidak perlu memasukkan informasi ini sendiri. Saat komputer Anda pertama kali terhubung ke jaringan rumah, ia mengirimkan permintaan ke setiap perangkat lain di jaringan, seperti: “Bisakah seseorang memberi saya alamat IP?”

Router broadband Anda harus merespons dengan patuh dan memberikan alamat IP pada komputer Anda. Seperti yang mungkin Anda ketahui, router adalah perangkat yang menyatukan Internet. Tidak seperti laptop dan desktop, router memiliki lebih dari satu antarmuka jaringan, lebih dari satu kabel (atau titik nirkabel) yang terpasang padanya, dan lebih dari satu alamat IP. Di rumah atau kantor Anda, router adalah kotak kecil yang menyediakan koneksi Anda ke Internet melalui layanan broadband Anda.

Metode yang digunakan untuk menetapkan alamat IP adalah Dynamic Host Configuration Protocol (DHCP). Jika tidak ada alamat IP saat Anda menjalankan ifconfig atau ipconfig, Anda dapat memaksa komputer Anda untuk mengambil pengaturan DHCP baru. Di Windows, jalankan ipconfig /release diikuti dengan ipconfig /renew. Di Mac, jalankan sudo ipconfig set en0 DHCP, dan di LinuX gunakan sudo dhclient eth07. Di Mac dan LinuX, perintah sebenarnya diawali dengan sudo8 yang memaksa Anda memasukkan kata sandi root untuk komputer. Anda juga harus menentukan antarmuka mana yang akan diperbarui, biasanya eth0 di LinuX dan en0 untuk Mac. Jika ini berhasil, voila! Anda sedikit lebih dekat untuk kembali ke Internet. Jika tidak, periksa router broadband Anda. Mungkin mati, terputus atau rusak, atau mungkin hanya perlu diatur ulang.

8.2 GERBANG DEFAULT

Anda tahu router broadband Anda pernah hidup di masa lalu karena memberi Anda alamat IP. Tapi itu bisa saja terjadi hingga tiga hari yang lalu: apakah masih ada sampai sekarang?

Setiap komputer di Internet juga memiliki gateway default. Ini pada dasarnya adalah alamat IP peralatan jaringan di ujung lain kabel jaringan atau koneksi nirkabel Anda. Ini adalah pintu gerbang komputer Anda ke Internet. Setiap kali Anda meminta sesuatu dari Internet, permintaan tersebut dikirim melalui gateway ini. Untuk menemukan gateway default kami, jalankan netstat -nr di Mac dan LinuX, dan rutekan pencetakan (atau ipconfig lagi) di Windows. Anda akan mendapatkan sesuatu seperti:

```
Destination  Gateway          Genmask         Flags Metric   Ref   Use Iface
192.168.0.0  0.0.0.0          255.255.255.0   U        0     0     0 eth0
0.0.0.0      192.168.0.1      0.0.0.0         UG       100   0     0 eth0
```

Pada kolom Destination diatas, angka 0.0.0.0 (atau kata “default”) berarti dimana saja dan huruf G pada kolom Flags berarti “Gateway”. Oleh karena itu, gateway default komputer ini adalah 192.168.0.1. Untuk orang-orang di rumah atau di kantor kecil, ini mungkin adalah alamat IP internal router broadband.

Anda dapat menggunakan perintah ping untuk melihat apakah sudah aktif dan tersedia. Jenis

```
ping 192.168.0.1.
$ ping 192.168.0.1
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.
64 bytes from 192.168.0.1: icmp_seq=1 ttl=64 time=1.31 ms
64 bytes from 192.168.0.1: icmp_seq=2 ttl=64 time=0.561 ms
64 bytes from 192.168.0.1: icmp_seq=3 ttl=64 time=12.6 ms
```

“64 bytes dari 192.168.0.1” menunjukkan balasan yang berhasil. Jika Anda mendapat balasan, maka router broadband Anda dapat dijangkau. Jika tidak, periksa lagi. Di Mac dan LinuX, balasan akan berlangsung selamanya hingga Anda menekan Control + C. Di Windows, balasan akan berhenti setelah ping keempat.

Di Luar Router

Untuk melampaui router Anda, Anda memerlukan perintah traceroute di Mac dan LinuX, dan tracert di Windows. Perintah ini menelusuri rute melalui Internet, melaporkan setiap perangkat jaringan (router) yang ditemuinya. Alamat IP dibentuk dari empat angka dari 0 hingga 25510. Pilih alamat IP dan cobalah:

```
$ traceroute -q 1 -n 1.2.3.4
traceroute to 1.2.3.4 (1.2.3.4), 30 hops max, 60 byte packets
  1 *
  2 80.3.65.217 9.163 ms
  3 213.105.159.157 11.158 ms
  4 213.105.159.190 11.215 ms
  ...
  13 72.14.236.149 98.484 ms
  14 209.85.252.47 104.071 ms
  15 *
  16 *...
```

Opsi -q 1 di Mac dan LinuX memberitahukan perintah untuk mencoba setiap router hanya sekali. Kemudian memerintahkannya untuk tidak repot-repot mencari nama yang dapat dibaca manusia untuk setiap router, yang membuat perintah menjadi lebih lambat. Opsi ini adalah -d di Windows, jadi gunakan tracert -d 1.2.3.4.

Setiap langkah di atas dikenal sebagai lompatan dalam jargon jaringan. Lompatan pertama adalah router broadband. Ini mungkin dikonfigurasi untuk tidak memberikan informasi apa pun tentang dirinya sendiri, jadi traceroute hanya menampilkan tanda bintang. Hop kedua membawanya keluar dari jaringan lokal ke sisi lain dari router broadband. Pada setiap hop berikutnya terdapat router lain, mungkin dengan banyak antarmuka jaringan dan banyak alamat IP. Setiap router memiliki tabel routing seperti di atas. Tabelnya berisi aturan

seperti: jika tujuan dimulai dengan 0 hingga 91, maka kirimkan paket ke antarmuka eth1 (kolom Gunakan Iface); jika dimulai dengan 92 hingga 128, gunakan eth2.

Contoh ini berjalan 14 hop sebelum menemui jalan buntu, baik karena alamat IP diblokir atau tidak digunakan. Hanya sejauh itulah angka dapat membawa kita. Mudah-mudahan Anda telah menyadari bahwa peradaban masih berkembang, setidaknya beberapa lompatan jaringan di luar pintu depan Anda. Anda juga telah mempelajari cara menggunakan perintah jaringan yang berguna ifconfig, ping dan traceroute. Untuk menjelajah lebih jauh, Anda memerlukan DNS.

Sistem Nama Domain

Smashing Magazine bisa saja menggunakan http://80.72.139.101 sebagai alamat situs web utamanya daripada http://www.smashingmagazine.com. Ini mempunyai dua keuntungan: akan menggunakan lebih sedikit ruang pada kartu nama; dan itu akan tetap berfungsi ketika DNS sedang down. Namun, staf pemasaran Smashing mungkin keberatan, dan basis pelanggan mereka akan terbatas pada orang-orang dengan ingatan yang sangat baik.

Sistem nama domain membuat Internet lebih ramah manusia dengan menerjemahkan antara nama domain seperti www.smashingmagazine.com dan alamat IP seperti 80.72.139.101. DNS adalah database terdistribusi hierarki besar. Anda mungkin menyadari bahwa tidak ada satu komputer pun yang mengetahui semua terjemahan dan dapat berkonsultasi dengan orang lain. Sebaliknya, ini adalah jaringan komputer yang sangat besar yang masing-masing mengetahui beberapa terjemahan, dan mengetahui siapa lagi yang harus ditanyai jika tidak.

8.3 SERVER DNS LOKAL

Setiap komputer tahu tentang server DNS lokal. Ini adalah salah satu informasi penting yang disediakan router broadband Anda melalui DHCP: alamat IP Anda; alamat IP gateway default Anda; dan alamat IP server DNS lokal Anda.

Saat Anda mengetikkan alamat situs web ke browser Anda, komputer Anda terlebih dahulu meminta server DNS lokalnya untuk menerjemahkannya menjadi alamat IP. Untuk mengetahui server DNS Anda, jalankan perintah cat /etc/resolv.conf di Mac dan LinuX11, atau ipconfig /all di Windows. Di Mac dan LinuX, perintah cat menampilkan file, dan file /etc/resolv.conf berisi server nama domain Anda. Filenya terlihat seperti:

```
$ cat /etc/resolv.conf
nameserver 194.168.4.100
nameserver 194.168.8.100
```

NSLOOKUP

Untuk mendiagnosis masalah DNS, pertama-tama periksa apakah server DNS lokal Anda aktif dengan ping. Jika ya, Anda dapat menggunakan perintah nslookup untuk melihat

apakah responsnya benar. nslookup adalah singkatan dari pencarian server nama dan digunakan seperti ini:

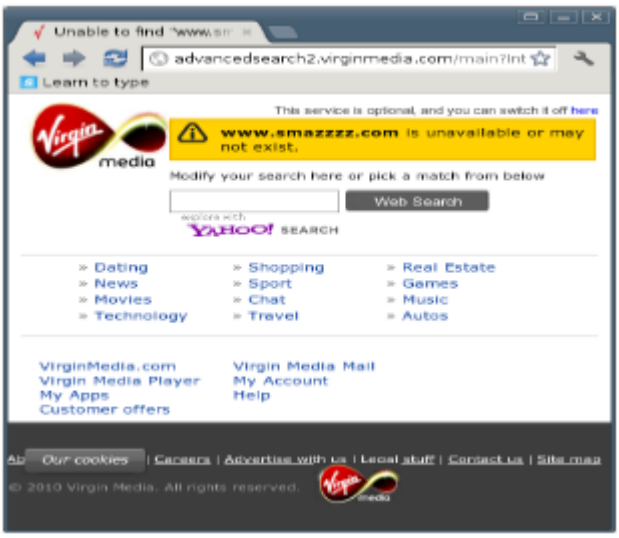
```
$ nslookup www.smashingmagazine.com
Server: 194.168.4.100
Address: 194.168.4.100#53
Non-authoritative answer:
Name: www.smashingmagazine.com
Address: 80.72.139.101
```

Perintah ini memberitahu Anda server DNS yang digunakan (194.168.4.100) dan alamat IP yang Anda cari (80.72.139.101). Jika nslookup tidak merespons, maka masalah Internet Anda terletak pada server DNS lokal Anda. Jika Anda berada di rumah atau di kantor kecil, server DNS Anda mungkin disediakan oleh perusahaan broadband Anda.

Mereka umumnya menyediakan dua atau lebih, jadi kecil kemungkinannya semuanya akan gagal. Jika Anda mengetahui alamat IP server nama yang berbeda, Anda dapat menanyakannya (nslookup www.smashingmagazine.com 194.168.8.100), tetapi server tersebut mungkin menolak untuk berbicara dengan orang asing. Anda mungkin perlu mengajukan keluhan kepada perusahaan broadband Anda tentang masalah tersebut.

Iklan Gratis

Pernahkah Anda salah mengetik alamat situs web dan mendapatkan halaman bermerek dari penyedia broadband Anda? Alih-alih mengakui “Saya tidak tahu”, server nama lokal Anda diam-diam membalas dengan alamat IP alternatif pilihannya, mempromosikan perusahaan broadband yang dimilikinya. Sangat menarik untuk melihat bahwa orang-orang pemasaran menggunakan DNS, dan sayang sekali rekan teknis mereka tidak menghentikan mereka, karena praktik semacam ini membuat beberapa proses jaringan otomatis menjadi lebih sulit.



Gambar 8.1 Perusahaan broadband menyadap situs web yang tidak ada.

PERJALANAN LENGKAP

Setelah mengonfirmasi bahwa server DNS lokal Anda berfungsi, Anda dapat mencoba menjalin kontak ramah manusia dengan Internet menggunakan traceroute dengan nama domain alih-alih alamat IP, dan mengabaikan opsi -n. Perintahnya adalah tracert di Windows. Ini akan melaporkan nama yang dapat dibaca untuk setiap router.

```
$ traceroute -q 1 www.smashingmagazine.com
traceroute to www.smashingmagazine.com (80.72.139.101), 30 hops max, 60
byte packets
 1 *
 2 brig-core-2b-ae6-725.network.virginmedia.net (80.3.65.177) 10.542 ms
 3 popl-bb-1b-ae14-0.network.virginmedia.net (213.105.159.157) 13.934 ms
 4 popl-bb-1c-ae1-0.network.virginmedia.net (213.105.159.190) 14.454 ms
 5 nrth-bb-1c-ae7-0.network.virginmedia.net (62.253.174.137) 15.982 ms
 6 nrth-tmr-1-ae1-0.network.virginmedia.net (213.105.159.30) 16.215 ms
 7 fran-ic-1-as0-0.network.virginmedia.net (62.253.185.81) 36.090 ms
 8 FFMGW4.arcor-ip.net (80.81.193.117) 39.064 ms
 9 92.79.213.133 (92.79.213.133) 47.404 ms
10 92.79.200.190 (92.79.200.190) 45.385 ms
11 kar-145-254-15-178.arcor-ip.net (145.254.15.178) 40.421 ms
12 145.253.159.106 (145.253.159.106) 46.436 ms
13 1-3-frr02.continum.net (80.72.130.170) 49.321 ms
14 1-6-frr04.continum.net (80.72.130.158) 47.194 ms
15 www.smashingmagazine.com (80.72.139.101) 48.081 ms
```

Ini mengungkapkan lebih banyak tentang perjalanan yang diambil paket data. Setelah meninggalkan jaringan lokal Anda, beberapa lompatan pertama di traceroute mana pun mungkin dimiliki oleh perusahaan broadband lokal, dalam hal ini Virgin Media. Jika traceroute berhenti di sini, maka itu akan menjadi masalah yang harus mereka selesaikan. Anda dapat menelepon mereka untuk informasi lebih lanjut.

Setelah traceroute meninggalkan penyedia broadband Anda, ia memasuki wilayah tak berpenghuni dengan perangkat jaringan besar yang tidak dapat dipahami. Dalam hal ini dimiliki oleh Arcor, anak perusahaan Vodafone. Jika traceroute gagal di sini, ini mungkin menunjukkan masalah jaringan yang cukup besar dan tidak banyak yang dapat Anda lakukan.

Pada akhirnya, itu akan mencapai perusahaan hosting untuk situs web yang dimaksud (dalam hal ini Continum.net). Jika gagal di sana, kesalahannya mungkin terletak pada perusahaan hosting Anda. Atau mungkin saja traceroute diblokir oleh firewall. Atau situs web Anda telah dipindahkan.

Situs Web Pindah

Kecil kemungkinan situs web Anda berpindah ke server lain tanpa Anda sadari, terutama karena Anda baru mengerjakannya tadi malam, namun Anda dapat memeriksanya kembali. Setiap server DNS menyimpan cache dari setiap nama domain yang diminta. Ini menghemat penyumbatan Internet dengan permintaan akan hal-hal yang jarang berubah. Kelemahannya adalah jika seseorang mengubah alamat IP untuk domain seperti

www.smashingmagazine.com, diperlukan waktu 24 hingga 48 jam untuk menghapus semua cache sehingga semua orang di dunia mengetahui alamat IP baru.

Untuk memastikan bahwa Anda memiliki informasi terbaru, pertama-tama Anda perlu mencari tahu server nama lokal untuk nama domain yang Anda tanyakan. Untuk melakukan ini, berikan nslookup opsi -type=ns, seperti ini di Mac, LinuX dan Windows:

Asal (atau terkadang server nama utama) adalah server DNS lokal untuk www.smashingmagazine.com. Anda dapat menggunakan nslookup lagi untuk menanyakan server ini secara langsung:

```
$ nslookup -type=ns www.smashingmagazine.com
Server: 194.168.4.100
Address: 194.168.4.100#53
Authoritative answers can be found from:
smashingmagazine.com
    origin = a.regfish-ns.net
    mail addr = postmaster.regfish.com...
```

Bandingkan ini dengan nslookup di server DNS lokal Anda. Ia tidak lagi mengatakan “tidak berwibawa”. Inilah jawaban resminya. Itu alamat IP yang sama, jadi kita tahu bahwa www.smashingmagazine.com tidak berpindah secara tiba-tiba tadi malam.

Di Mac dan LinuX, Anda dapat menggunakan perintah dig untuk mengetahui dengan tepat berapa lama server DNS lokal Anda menyimpan terjemahan ini dalam cache. Itu singkatan dari penggerek informasi domain. Pengguna Windows perlu mencari alat penggalian online karena Windows tidak mendukung perintah ini:

```
$ dig www.smashingmagzine.com
...
;; ANSWER SECTION:
www.smashingmagzine.com. 246 IN A 80.72.139.101...
```

246 adalah jumlah detik sebelum cache server DNS lokal kedaluwarsa dan harus menemukan kembali alamat IP untuk smashingmagazine.com.

8.4 ROUTER BROADBAND DIKEMBALIKAN

Sekarang DNS berfungsi, Anda dapat mengetahui pendapat dunia tentang Anda. Anda telah menemukan alamat IP komputer Anda di atas. Tapi itu mungkin bukan yang digunakannya di Internet. Jika diawali dengan 192.168 atau 10, maka sudah pasti itu bukan alamat publik. Rentang alamat IP tersebut menandakan alamat IP lokal yang hanya digunakan dalam jaringan internal Anda.

Saat komputer Anda mengirimkan permintaan ke Internet, pertama-tama permintaan tersebut masuk ke gateway default Anda (router broadband Anda), yang juga memiliki alamat IP internal lokal seperti 192.168.0.1. Namun router Anda juga memiliki alamat IP publik lain.

Router Anda menyelesaikan permintaan Anda dan mengirimkannya ulang dari alamat IP publik ini. Oleh karena itu, alamat IP publik router broadband Anda pada dasarnya adalah alamat IP publik Anda. Beginilah cara orang lain di Internet melihat Anda saat Anda menjelajah.

Ini adalah alamat IP yang akan muncul di file log di situs web mana pun yang Anda kunjungi. Akibatnya, siapa pun yang menggunakan router broadband yang sama akan memiliki alamat IP publik yang sama dengan Anda. Router Anda menangani semua ini menggunakan proses yang disebut terjemahan alamat jaringan, memastikan permintaan dan informasi keluar dan kembali ke alamat IP lokal yang benar.

Anda dapat mengetahui alamat IP publik router broadband Anda dengan mengunjungi situs web seperti whatismyipaddress.com. Alternatifnya, Anda dapat menjalankan perintah curl ipinfo.io/ip di Mac atau LinuX, atau wget -O- -q ipinfo.io/ ip di LinuX. Curl dan wget mengambil halaman Web (http://ipinfo.io/ip) dari Internet. Opsi -O- (yaitu huruf O, bukan nol) memerintahkan wget untuk menampilkan hasilnya ke layar (ditandai dengan satu tanda hubung) daripada menyimpannya ke file, dan -q memerintahkannya untuk melakukannya secara diam-diam. curl secara otomatis ditampilkan ke layar. Untuk menggunakan curl di Windows Anda harus mendownload dan menginstalnya terlebih dahulu. Semua metode ini keluar dari jaringan lokal Anda dan lihat ke belakang. Tidak ada cara untuk mengetahui alamat IP publik router Anda dari dalam. Outputnya cukup sederhana:

```
$ curl ipinfo.io/ip
85.106.118.199
```

Di Mana Mereka?

Situs web seperti whatismyipaddress.com dan ipinfo.io melakukan lebih dari sekadar memberi tahu Anda alamat IP publik Anda. Mereka juga menyediakan layanan geolokasi. Sangat menarik untuk mengambil alamat IP dari traceroute yang disebutkan sebelumnya di halaman 187 dan menyalin dan menempelkannya. Geolokasi menebak lokasi fisik router, dan juga dapat memberi tahu Anda siapa pemilik alamat IP tersebut. Alamat 62.253.185.81 berada di Inggris selatan tetapi alamat berikutnya melintasi Selat ke 80.81.193.117 di Frankfurt, Jerman. Jenis geolokasi ini bergantung pada database kepemilikan alamat IP.

Faktanya, ada situs web yang dapat memetakan semua ini untuk Anda, seperti DNStools.ch dan YouGetSignal. Titik awal untuk traceroute ini adalah server Web yang menghosting alat tersebut, bukan komputer Anda sendiri.

Gambar 8.2 Traceroute visual dari Los Angeles ke London menempuh jarak 7.904 mil dalam 4,8 detik. Di atas adalah contoh jarak dan waktu yang dibutuhkan antara server Web di Los Angeles dan situs BBC di London.

Menghubungkan ke Server Anda

Jadi, peradaban dan Internetnya sudah aktif dan berjalan. Apa yang salah? Situs web Anda ada di komputer di suatu tempat di luar sana, mungkin di ruangan besar ber-AC yang penuh dengan komputer lain, dengan banyak pintu pemadam kebakaran dan banyak sekali kabel berwarna-warni. Komputer ini dalam bahasa sehari-hari dikenal sebagai server Web.

Bayangkan sejenak bahwa server Web Anda adalah negara Perancis. Jika Anda ingin mengirim perabot berukuran besar ke suatu tempat di Prancis, perabot tersebut akan dibungkus rapat di kapal kontainer dan dikirim ke seberang laut. Kapal tersebut akan tiba di salah satu pelabuhan utama Perancis, mungkin Marseille atau BordeauX atau Le Havre. Tidak masalah bagi Anda pelabuhan mana yang dilewatinya, tetapi penting bagi perusahaan pelayaran. Komputer serupa, hanya saja ukurannya sedikit lebih kecil dan memiliki 65.535 port.

Di komputer, beberapa port diberi fungsi tertentu. Di server Web, port 80 menerima dan membalas permintaan penelusuran Web. Port 25 dan 110 berhubungan dengan email. Permintaan Web pada umumnya akan melibatkan port bernomor tinggi (biasanya acak) di komputer Anda yang mengirimkan permintaan ke port 80 di 80.72.139.101, seperti: “Hai, kirimkan saya halaman Web /index.html.”

8.5 TELNET DAN NETCAT

Perintah telnet memungkinkan Anda meniru kapal kontainer dan terhubung ke port tertentu di server. Windows tidak memiliki telnet secara default, tetapi Anda dapat mengaktifkannya di Windows 7 dengan membuka Mulai → Panel Kontrol → Program → Mengaktifkan atau menonaktifkan fitur Windows → Klien Telnet.

Karena kita sedang menghadapi masalah situs web, dan karena server Web hampir selalu berada pada port 80, cobalah melakukan telnet ke port 80:

```
$ telnet www.smashingmagazine.com 80
Trying 80.72.139.101...
telnet: Unable to connect to remote host: Connection refused
```

Mac dan LinuX mendukung perintah alternatif: netcat. Ini lebih cocok untuk tugas jaringan dan mendukung fitur tambahan seperti proXies. Bab ini akan fokus pada telnet, karena ia juga berfungsi pada Windows. Tambahkan -v ke netcat untuk membuatnya memberi tahu Anda secara jelas apa yang dilakukannya.

```
$ netcat -v www.smashingmagazine.com 80
netcat: connect to www.smashingmagazine.com port 80 (tcp) failed:
Connection refused
```

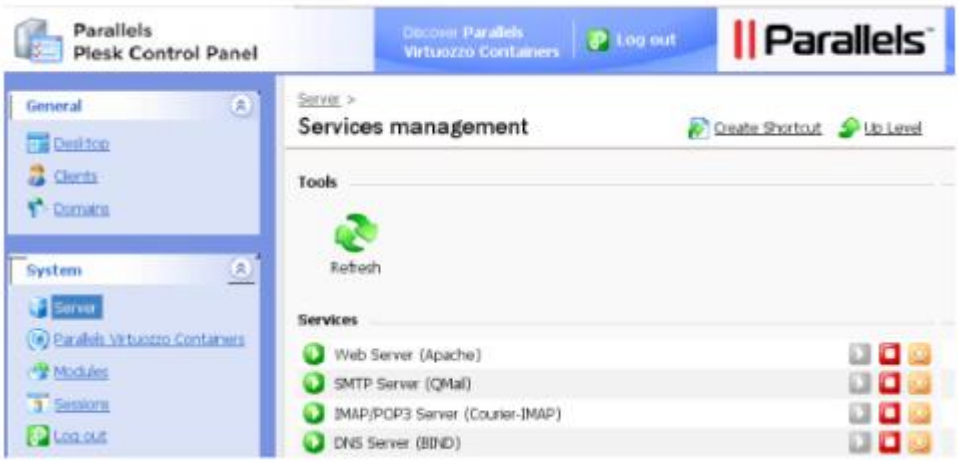
Uh oh.

Kecuali, tidak juga. Saya memalsukan masalah di atas. Smashing Magazine tidak terlalu down. Tapi saya akan menggunakan www.smashingmagazine.com sebagai contoh domain di seluruh bab ini. Tangguhkan ketidakpercayaan Anda dan anggaplah Smashing telah pindah ke pasar Widget 3000 dan secara berturut-turut menjadi korban dari hampir semua masalah jaringan dan situs web yang bisa dibayangkan, dan kemudian mengatasinya.

Panel Kendali

Setiap kali server Web Anda menerima data pada port 80, ia mengirimkannya ke perangkat lunak untuk diproses. Yang membingungkan, perangkat lunak itu juga disebut server Web. Sejauh ini jenis perangkat lunak server Web yang paling umum adalah Apache. Menurut W3Techs pada bulan Juni 2013, perusahaan ini memiliki pangsa pasar sebesar 65,2%.15 Server Informasi Internet (IIS) Microsoft berada di urutan kedua dengan 15,7%, tepat di depan nginx sebesar 14,3%.

Perangkat lunak server web tidak boleh berhenti bekerja. Tapi kalau bisa, mudah-mudahan restart saja. Cara tercepat untuk melakukannya adalah dengan menggunakan panel kontrol yang disediakan oleh server Anda. Server Windows (pangsa pasar 34,3% pada Juni 201316) sering kali dikelola oleh Remote Desktop atau VNC yang memungkinkan Anda mengontrol layar, keyboard, dan mouse server, serta mengubah pengaturan langsung di server.



Gambar 8.3 Memulai ulang server Web dengan Plesk.

Namun, sisa bab ini akan fokus pada server LinuX dan UNIX (65,7%), yang biasanya dikelola melalui antarmuka Web seperti Plesk, CPanel atau Webmin. Alat manajemen ini sebenarnya hanya situs web, tetapi berjalan pada port yang berbeda. Plesk misalnya, biasanya berjalan pada port 8443, CPanel pada 2082 atau 2083 dan Webmin pada 10000.

Gali lebih dalam folder email Anda dan cari URL, nama pengguna, dan kata sandi untuk panel kontrol Anda. Masuk dan temukan layar yang memungkinkan Anda memulai ulang perangkat lunak server Web Anda. Di Plesk, cari “Manajemen layanan” (di Server atau Alat & Pengaturan) dan tekan tombol Putar di sebelah “Server Web (Apache)”.

SSH

Jika port 80 tidak aktif, kemungkinan besar panel kontrol juga tidak tersedia. Anda harus masuk ke server dan mengeluarkan perintah secara langsung. Untuk ini ada Secure Shell (SSH). SSH seperti Remote Desktop versi teks saja. Ini memungkinkan Anda untuk membuka jendela terminal di server. Dari desktop atau laptop LinuX atau Mac, gunakan perintah ssh. Dari komputer Windows, unduh dan jalankan PuTTY.

Anda memerlukan nama pengguna dan kata sandi untuk server Anda, yang terdapat dalam email yang sama seperti di atas. Pada server LinuX, root adalah pengguna administratif yang paling kuat. Untuk alasan keamanan, pengguna SSH dari email Anda sering kali kurang diistimewakan. Saat Anda menjalankan SSH, Anda harus memberikan nama pengguna sebagai bagian dari perintah. Ini akan meminta Anda untuk menerima sidik jari keamanan dan memasukkan kata sandi:

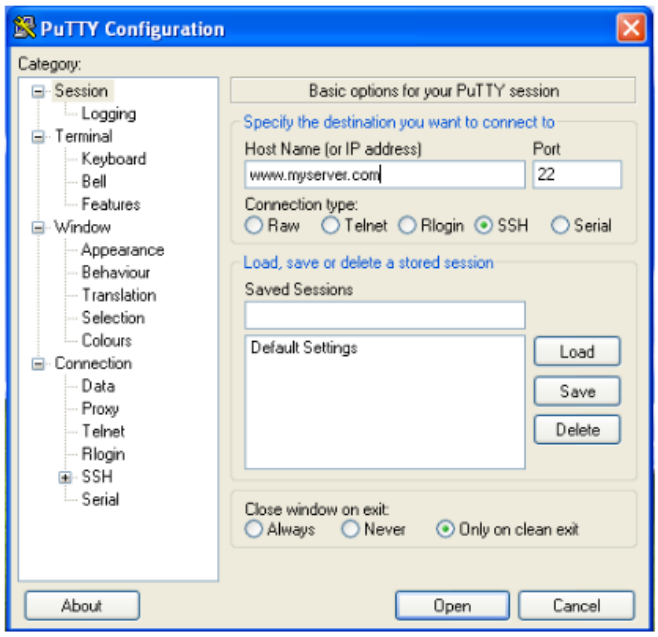
```
$ ssh root@www.smashingmagazine.com
The authenticity of host 'www.smashingmagazine.com (80.72.139.101)'
can't
be established.
RSA                 key                 fingerprint                 is
00:5a:cf:51:83:46:0b:91:29:ef:2e:1d:c9:59:e9:ab.
Are you sure you want to continue connecting (yes/no)? yes
```

```
Warning: Permanently added 'www.smashingmagazine.com,80.72.139.101'
(RSA)
to the list of known hosts.
root@www.smashingmagazine.com's password: ...
```

Jika berhasil, Anda akan mendapatkan pesan selamat datang dan terminal prompt:

```
Linux dedivps-13236 2.6.10-091stab048.3 #1 SMP Fri Dec 7 17:06:14 GMT
2012 x86_64
Last login: Thu May 2 07:20:11 2013 from cpc1-brig18-2-0-cust123.3-3.
cable.virginmedia.com
root@dedivps-13236:~#
```

Perhatikan bahwa ini hanya akan bekerja pada server LinuX atau UNIX yang memiliki server SSH yang menerima koneksi, dan server Windows langka yang memilih untuk menginstalnya. Untuk sebagian besar server Windows lainnya, Anda memerlukan Remote Desktop. Jika Anda tidak bisa masuk ke server Anda melalui panel kontrol atau SSH, pilihan Anda sangat terbatas. Ada kemungkinan kecil bahwa firewall yang terlalu aktif menghalangi, atau Anda mengalami serangan penolakan layanan. Atau Anda harus menghubungi perusahaan hosting Anda dan meminta bantuan.



Gambar 8.4 Menggunakan PuTTY untuk SSH dari komputer Windows.

firewall

Firewall adalah bagian perangkat keras atau perangkat lunak yang menyaring data masuk dan keluar. Filter ini diterapkan sesuai dengan alamat IP dan port sumber dan tujuan. Jadi, misalnya, firewall harus mengizinkan semua permintaan masuk ke port server 80 atau tidak ada yang bisa mengakses situs web. Tapi itu mungkin memblokir semua permintaan ke

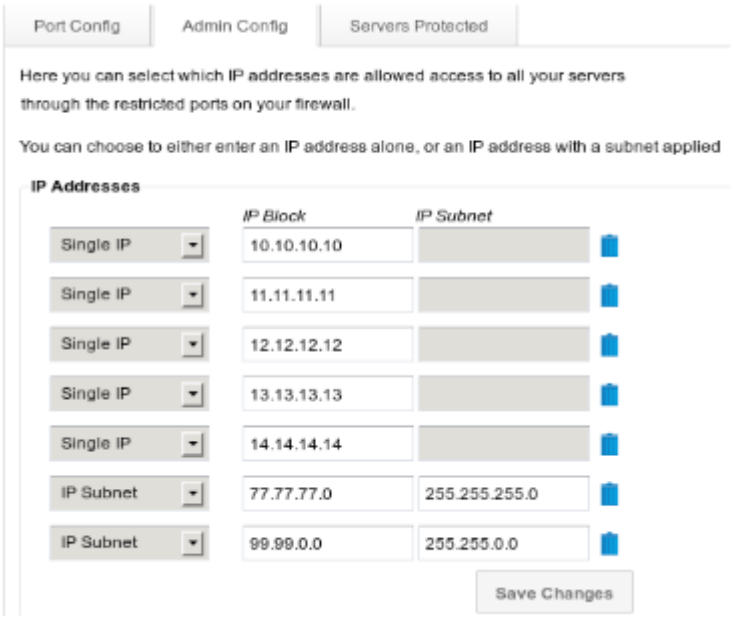
port 8443 (Plesk), port 22 (SSH) atau port 3389 (desktop jarak jauh) kecuali dari beberapa alamat IP yang dikenal dan tepercaya.

Anda dapat mengetahui apakah ada firewall yang menghalangi Anda tergantung pada bagaimana koneksi gagal. Untuk menguji apakah SSH diblokir, Anda dapat menjalankan perintah ssh atau menggunakan telnet seperti di atas untuk port 22:

```
$ telnet www.smashingmagazine.com 22
Trying 80.72.139.101...
telnet: Unable to connect to remote host: Connection refused
```

“Koneksi ditolak” berarti data Anda mencapai server tetapi mungkin ditolak masuk karena alasan non-firewall. Misalnya, SSH mungkin dimatikan atau dijalankan pada port berbeda. Pesan “Connection timed out” atau tidak ada pesan yang lebih kuat menunjukkan adanya blok firewall. Jika terhubung, tekan Control +] untuk membuka prompt telnet> lalu ketik quit untuk keluar.

Jadi, jika Anda memiliki firewall (email tersebut akan memberi tahu Anda), Anda perlu memastikan bahwa SSH diizinkan dan alamat IP publik Anda ada dalam daftar yang tepercaya. Meskipun Anda mengetahui alamat IP Anda ada dalam daftar kemarin, alamat tersebut mungkin telah berubah hari ini, terutama jika Anda mengalami masalah broadband baru-baru ini. Alamat IP publik yang ditetapkan ke router rumah bisa tetap sama selama berbulan-bulan, dan kemudian tiba-tiba berubah. Bagaimana, mengapa, dan kapan bergantung pada perusahaan broadband Anda dan tetangga Anda. Satu-satunya cara untuk menghindari hal ini adalah dengan membayar ekstra untuk alamat IP statis atau tetap.



Gambar 8.5 Banyak firewall menyimpan daftar alamat IP tepercaya.

8.6 KEGAGALAN LAYANAN

Bayangkan Widget 3000 tiba-tiba menjadi viral. Ratu Inggris difilmkan melemparkan satu ke arah pemenang X Factor dan tiba-tiba semua orang di dunia ingin memilikinya. Anda mungkin berpikir “Fantastis!” Namun kecuali infrastruktur server Anda siap untuk meningkatkan dari 100 kunjungan per jam menjadi 100 juta, Anda mungkin tidak akan menjual terlalu banyak. Semua pengunjung yang mengakses situs web Anda sekaligus akan menghentikan jaringan dan server Anda. Beberapa ribu pengunjung pertama mungkin menerima setengah halaman Web, sisanya akan menatap browser kosong.

Dan ketika Anda mencoba melakukan telnet ke server Anda seperti di atas, server tersebut juga akan menunggu di sana tidak ada penolakan tetapi juga tidak ada entri. Hal ini kira-kira terjadi pada serangan penolakan layanan terdistribusi (DDoS). Semua peretas yang telah menghabiskan 15 tahun terakhir menemukan lubang di Internet EXplorer tidak bekerja dengan sia-sia. Mereka telah berhasil memasang kuda Troya di jutaan komputer di seluruh dunia. Ketika mereka mengeluarkan perintah, semua komputer tersebut tiba-tiba mencoba mengirim data dan meminta data dari server Web Anda, membuatnya kewalahan dan membuatnya tidak dapat dijangkau.

Kecuali Anda menjalankan bank atau operasi pengiriman spam, atau telah berhasil membuat musuh yang cerdik dan gigih, hal ini tidak mungkin terjadi pada Anda. Anggaplah telnet telah berhasil terhubung.

Memeriksa Server Anda

Sekarang Anda berada dalam bisnis. Anda membuka jendela terminal di server Anda menunggu setiap perintah Anda. Mulai sekarang, semua perintah dikeluarkan di server LinuX Anda, bukan laptop Anda. Jenis server UNIX lainnya, termasuk Mac, mungkin memiliki perintah berbeda atau perintah yang sama dengan opsi berbeda.

Mendengarkan Port 80

Langkah pertama adalah mencari tahu perangkat lunak mana yang seharusnya merespons ketika Anda mencoba melakukan telnet ke port 80. Untuk itu, Anda dapat menggunakan perintah netstat untuk menampilkan semua informasi jaringan tentang server Anda. Tambahkan -tlpn ke perintah untuk membuatnya hanya menampilkan koneksi TCP yang didengarkannya, bersama dengan port numerik dan program yang dimilikinya. Pada beberapa sistem, opsi p memerlukan akses pengguna super, sehingga perintah di bawah ini juga melakukan sudo.

Ini hanya menampilkan keluaran yang disingkat. Biasanya menampilkan semua port yang didengarkan server, dan jumlahnya bisa antara 10 dan 100. Saat menjalankan perintah apa pun, Anda dapat mengurangi outputnya menggunakan perintah grep. grep memfilter output dan hanya menampilkan baris yang berisi kata atau frasa yang Anda tentukan. Bilah vertikal | membuat pipa, menyalurkan output dari satu perintah (netstat) ke input perintah lain (grep). Coba ini sebagai gantinya:

```
netstat -tlpn | grep :80
```

```
tcp6 0 0 127.0.0.1:8005 :::* LISTEN 22885/java
```

Ini menjalankan netstat dan hanya menampilkan hasil yang mengandung :80. Server ini memiliki proses java yang mendengarkan port 8005 tetapi tidak ada server Web yang berjalan.

Server Web yang mana

Ketika server LinuX dijalankan, ia mencari di direktori /etc/init.d dan menjalankan skrip di sana untuk meluncurkan perangkat lunaknya. Ini bervariasi di antara distribusi LinuX, dan pada varian UNIX lain seperti BSD, ini mungkin /etc/rc.d atau /etc/rc.d/init.d. Ini mirip dengan folder menu Startup di Windows.

Anda dapat melihat server Anda mulai menggunakan perintah ls yang mencantumkan file dalam direktori. -l menunjukkan format panjang dengan izin, pemilik, ukuran dan tanggal pembuatan.

```
$ ls -l /etc/init.d
total 368
-rwxr-xr-x 1 root root 7621 Sep 13 2012 apache2
-rwxr-xr-x 1 root root 3281 Oct 5 2012 bind9
-rwxr-xr-x 1 root root 2444 Jan 1 2011 bootlogd
-rwxr-xr-x 1 root root 5364 Nov 1 2011 courier-imap
-rwxr-xr-x 1 root root 3753 Dec 19 2010 cron...
```

Anda sedang mencari paket perangkat lunak server Web seperti apache2, httpd (nama lama Apache) atau nginx. Anda dapat menggunakan grep lagi dengan opsi-e yang memerintahkannya untuk menggunakan ekspresi reguler. Dalam ekspresi reguler, bilah vertikal berarti “atau” sehingga ini menampilkan file apa pun yang berisi “apache” atau “http” atau “nginx”. Bilah harus di-escape dengan garis miring terbalik:

```
$ ls -l /etc/init.d | grep -e "apache\|http\|nginx"
-rwxr-xr-x 1 root root 7621 Sep 13 2012 apache2
```

MEMULAI ULANG PERANGKAT LUNAK SERVER WEB

File di /etc/init.d disebut skrip shell. Perintah yang Anda gunakan di Mac dan LinuX hingga saat ini merupakan bagian dari bahasa tipe C yang juga mencakup variabel pengaturan dan loop yang berjalan. Seperti dalam JavaScript, titik koma di akhir setiap baris bersifat opsional, tetapi jika Anda menggunakannya, Anda dapat menjejalkan beberapa perintah ke dalam satu baris. Berikut adalah perulangan for sederhana pada baris perintah:

```
$ for i in 1 2 3; do echo Line $i; done
Line 1
Line 2
Line 3
```

Untuk melihat skrip shell yang kompleks, lihat beberapa skrip startup di /etc/init.d. Gunakan perintah less untuk melihat file. Tekan spasi untuk melihat halaman berikutnya atau q untuk berhenti dari halaman berikutnya.

Alasan kami sebenarnya ada di sini adalah untuk memulai ulang perangkat lunak server Web. Jika Anda masuk ke ssh sebagai root pengguna administratif, Anda dapat menjalankan perintah restart secara langsung. Dalam hal ini, Anda akan mengetikkan perintah setelah #, bukan $. Jika tidak, Anda harus mengawalinya dengan perintah sudo, yang mengatakan lakukan beberapa perintah sebagai pengguna super. Anda memerlukan kata sandi root untuk sudo. Untuk memberitahukan perangkat lunak server Web untuk memulai, jalankan:

```
$ sudo /etc/init.d/apache2 start
Password:
Starting apache2...
```

Semoga ini gagal dengan pesan kesalahan yang berguna. Mudah-mudahan, karena Anda akan tahu mengapa crash tersebut terjadi, dan Anda dapat memasukkan pesan kesalahan tersebut ke Google untuk mengetahui cara memperbaikinya.

Jika kurang beruntung untuk bekerja, maka server Web Anda sekarang sedang berjalan. Semua skrip di /etc/init.d dijalankan sebagai proses latar belakang, atau daemon dalam bahasa UNIX. Ini berarti Anda dapat memulainya dan mereka akan tetap berjalan bahkan setelah Anda keluar dari ssh, matikan komputer Anda dan pergi ke pantai. Ini tidak seperti perintah seperti traceroute dan ls yang melakukan tugasnya dan menyelesaikannya. Anda dapat menjalankan netstat lagi untuk memeriksa ulang apakah server Web sedang berjalan. Perhatikan d di akhir apached. Itu singkatan dari daemon.

```
netstat -tlpn | grep :80
tcp 0 0 :::80 :::* LISTEN 18201/apached
tcp6 0 0 127.0.0.1:8005 :::* LISTEN 22885/java
```

8.7 LOG KESALAHAN SERVER WEB

Jika gagal memulai dan tidak memberikan pesan kesalahan yang ramah, tempat berikutnya yang harus dicari adalah log kesalahan. Ini biasanya berada di direktori /var/log yang diberi nama sesuai dengan perangkat lunaknya. Jalankan ls /var/log untuk memeriksa ulang. Misalnya, log Apache ada di /var/log/apache2.

```
$ ls -l /var/log/apache2/*log*
-rw-r----- 1 root adm 1944899 May 5 09:59 /var/log/nginx/access.log
-rw-r----- 1 root adm 538152 May 4 02:40 /var/log/nginx/access.
log.2.gz
-rw-r----- 1 root adm 28647 May 5 08:18 /var/log/nginx/error.log
-rw-r----- 1 root adm 5701 May 4 02:35 /var/log/nginx/error.log.2.gz
```

Ini menunjukkan bahwa Apache memiliki log akses dan log kesalahan. Kedua log tersebut di-zip sekitar pukul 02:30 setiap pagi. Perintah selanjutnya pertama-tama berubah ke direktori /var/log/apache2 dengan perintah cd, dan kemudian menggunakan tail untuk melihat beberapa entri terakhir dalam file log.

```
$ cd /var/log/apache2; tail error.log
```

Untuk melihat file log yang di-gzip, gunakan perintah zcat untuk mengeluarkannya dan menyalurkannya melalui tail. -20 menunjukkan 20 baris terakhir.

```
$ zcat error.log.2.gz | tail -20
```

Atau lebih baik lagi, cari saja kesalahannya menggunakan grep. Gunakan zcat dengan opsi -f untuk menampilkan file log normal dan zip. Kemudian menyalurkan output melalui grep untuk mencari kata "kesalahan" secara tidak sensitif:

```
$ zcat -f error.log* | grep -i error
```

Perintah ini mungkin menghasilkan banyak output. Jika penggemar Matrix kebetulan melewati komputer Anda saat ini, mereka akan terkesan melihat semua data mentah lewat. Namun, ini tidak akan banyak membantu Anda, jadi kurangi saja:

```
$ zcat -f error.log* | grep -i error | less
```

kurang itu kuat. Anda dapat menekan tombol panah atau j untuk turun, k untuk naik, /sesuatu untuk mencari “sesuatu” dan h untuk melihat daftar semua perintah yang berguna. Jika Anda dapat mempersempit momen kegagalan server Web Anda menjadi beberapa jam, Anda dapat menggunakan lebih sedikit untuk menavigasi ke bagian file log tersebut.

LOG SISTEM

Ada beberapa file log berguna lainnya di /var/log seperti syslog (pencatat sistem) dan dmesg (pesan bootup). Semuanya menggunakan format tanggal yang serupa sehingga jika Anda dapat mempersempit waktu ketika Anda mencurigai ada yang tidak beres, Anda dapat mencari semuanya sekaligus. Perintah ini berubah ke direktori /var/log dan kemudian menampilkan semua file menggunakan zcat -f. [234] di grep dipinjam dari ekspresi reguler dan cocok dengan angka 2 atau 3 atau 4. Jadi ini akan menampilkan pesan kesalahan apa pun di log mana pun yang terjadi pada tanggal 5 Mei antara pukul 02:00 dan 04:00 pagi :

```
$ cd /var/log; zcat -f * | grep "May 5 0[234]:" | less
```

Keluar Dari Ruang

Jika perangkat lunak server Web Anda masih tidak dapat dijalankan dan kesalahan tetap sulit dipahami, ada beberapa masalah umum yang dapat Anda periksa secara eksplisit.

Server Anda mungkin kehabisan ruang hard drive. Gunakan perintah df untuk menunjukkan penggunaan file disk. -h menunjukkan hal-hal dalam bentuk yang ramah manusia (dengan M untuk megabyte dan G untuk gigabyte, bukan angka yang sangat besar):

```
$ df -h
Filesystem  1M-blocks   Used Available    Use% Mounted on
/dev/simfs  20.4G       9.8G             10.6G 49% /
tmpfs       1.6G        0                1.6G 0% /lib/init/rw
tmpfs       1.6G        0                1.6G 0% /dev/shm
```

Jika itu masalahnya, maka solusi cepatnya adalah mencari dan menghapus file yang sangat besar. Perintah find sangat kuat. Opsi -size memerintahkannya untuk mencari file setidaknya dengan ukuran tertentu, dan opsi -printf memerintahkannya untuk mencetak ukuran (%12s, di mana 12 mengarahkan area cetak menjadi lebar 12 karakter untuk membantu menyelaraskan) , waktu modifikasi terakhir (%t), direktori (%h), nama file (%f) dan kemudian baris baru (\n). Untuk melihat semua file yang berukuran lebih dari 10 megabyte, coba:

```
$ find / -size +10M -printf "%12s %t %h/%f\n"
445631888 Mon Mar 18 13:38:07.0380913017 2013 /var/www/huge-file.zip
```

Jika Anda benar-benar yakin bahwa file ini berlebihan, Anda dapat menggunakan perintah berikut untuk menghapus file dan mengosongkan ruang dengan cepat. Berhati-hatilah karena tidak ada jalan untuk kembali. Perintah rm tidak memiliki folder sampah tempat Anda dapat memulihkannya nanti: `$rm /var/www/huge-file.zip`

Keluar Dari Memori

Untuk memeriksa penggunaan RAM server Anda, jalankan perintah free, sekali lagi dengan -m untuk menampilkan dalam megabyte:

```
$ free -m
     total       used       free       shared     buffers    cached
Mem:  3067       2673       393              0          0     819
-/+ buffers/cache:1853      1213
Swap: 0          0          0
```

Server ini memiliki total memori 3GB dan ruang kosong 393MB. Ini bagus karena LinuX suka menggunakan banyak memorinya. Ini adalah baris buffer/cache yang harus Anda fokuskan. Jika ini hampir semuanya digunakan, maka sistem Anda mungkin kesulitan mengatasinya.

Untuk mengetahui apa yang memonopoli semua memori, gunakan perintah top. Ini menunjukkan tampilan real-time dari penggunaan CPU dan memori sistem. Sayangnya ini juga akan berjalan sangat lambat jika server Anda sedang tegang, namun ini mungkin memberi tahu Anda apa yang menyebabkan masalahnya.

```
$ top
 PID USER     PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
22885 tomcat6 20 0 2061m 159m 2644 S 1 5.2 780:41.85 java
 1 root       20 0 8360 568 536 S 0 0.0 0:52.30 init
 2 root       20 0 0 0 0 S 0 0.0 0:00.00 kthreadd/10086
 3 root       20 0 0 0 0 S 0 0.0 0:00.00 khelper/10086
14579 root    20 0 40900 3124 1668 S 0 0.1 1:30.27 nginx...
```

Jika ada sesuatu yang memakan memori, Anda dapat memulai ulang atau mematikannya. Namun, pastikan Anda mengetahui apa itu terlebih dahulu. Anda mungkin membuat server crash sepenuhnya, atau mengunci diri Anda sendiri, atau menghentikan penggabungan basis data penting yang dimulai oleh rekan Anda yang tidak mengetahui efisiensi tiga hari yang lalu.

Pertama-tama, cobalah mencari cara yang ramah untuk memulai kembali proses yang salah. Banyak layanan yang dapat di-restart dengan baik dengan mengeluarkan perintah restart ke skrip shell yang memulainya, seperti:

```
$sudo /etc/init.d/apache2 restart
```

Jika itu tidak tersedia, cari online untuk mengetahui cara terbaik memulai ulang atau mematikannya dengan baik. Jika Anda masih belum beruntung, maka untuk menghentikan suatu proses, tekan k dan ketikkan nomor dari kolom ID proses (PID). Ini akan meminta konfirmasi dan kemudian mencoba menghentikan prosesnya. Jika Anda bukan root, mungkin tertulis “Operasi tidak diizinkan”, dalam hal ini Anda harus menjalankan sudo top saja. PID digunakan untuk mengidentifikasi perangkat lunak yang berjalan di komputer. Setiap contoh aplikasi, program, atau perangkat lunak memiliki PID unik.

Jika proses tidak mau dihentikan, Anda dapat menekan q untuk keluar dari atas dan mencoba perintah kill. Berikan opsi -9 yang lebih ekstrim. top mengirimkan sinyal ramah 15 (terminasi). Signal 9 langsung membunuh.

```
$ sudo kill -9 22885
```

Jalankan ke atas lagi. Jika beberapa proses serupa lainnya telah mengambil alih penghargaan pemakan memori, maka Anda hanya membunuh satu proses anak. Anda perlu mencari tahu orang tua yang melahirkan anak nakal tersebut, karena membunuh orang tua juga akan menghentikan semua anak. Perintah ps dapat digunakan untuk menampilkan informasi tentang proses tertentu. Biasanya ini tidak menampilkan ID proses induk, jadi Anda perlu menambahkan opsi -o dan menentukan bahwa output harus menampilkan ID proses induk ppid dan perintah lengkap yang memulainya:

```
$ ps -o ppid,command 14579 PPID COMMAND 6950 nginx: worker process
```

Proses nginx ini bukan yang utama.

```
$ ps -o ppid,command 6950
  PPID COMMAND
     1 nginx: master process /usr/sbin/nginx
```

PID induk yang sangat rendah berarti proses ini adalah daddy17. Membunuh proses 6950 akan mematikan proses nginx utama dan semua turunannya.

Ada cara yang lebih mudah untuk melakukan ini. Anda dapat mencari proses menggunakan pgrep dan mematikannya dengan pkill. -l memberitahu pgrep untuk mencantumkan nama proses juga. Misalnya:

```
$ pgrep -l nginx
6950 nginx
14579 nginx...
```

Dan kemudian lakukan pembunuhan dengan Sudo pkill nginx. Cara selanjutnya untuk mencari proses adalah menggunakan ps dengan opsi aux seperti pada ps aux | terima nginx. Lebih mudah, tetapi Anda tidak akan belajar tentang keajaiban top.

Berbicara HTTP

Pada tahap ini, perangkat lunak server Web Anda diharapkan sudah aktif dan berjalan. Jika crash terjadi, Anda telah memulai ulang, mencari tahu alasannya, dan mengambil langkah untuk mencegah kejadian serupa terjadi lagi.

Anda sekarang dapat memeriksa ulang apakah server Web Anda aktif dan berjalan dengan melakukan telnet ke port 80 dari laptop Anda lagi. Kali ini seharusnya tertulis “Tersambung” dan kemudian menunggu permintaan Anda. Server web memahami HTTP (protokol transfer hiperteks). Setelah koneksi dibuat, ketik GET / HTTP/1.1 untuk memberi tahu server bahwa Anda ingin GET (bukan POST) halaman beranda / dan bahwa Anda menggunakan protokol versi 1.1.

Tekan Enter lalu ketik Host: diikuti dengan nama host. Baris ini hanya diperlukan pada server yang menampung lebih dari satu situs web. HTTP tidak mengetahui bahwa Anda melakukan telnet ke www.smashingmagazine.com. Sejauh menyangkut hal ini, Anda melakukan telnet ke 80.72.139.101 dan ia perlu mengetahui situs web mana yang Anda minati. Tekan Enter dua kali untuk membuat permintaan. Anda akan mendapatkan kembali aliran teks dan HTML yang panjang:

```
$ telnet www.smashingmagazine.com 80
Trying 80.72.139.101...
Connected to www.smashingmagazine.com.
Escape character is '^]'.
GET / HTTP/1.1
Host: www.smashingmagazine.com
```

```
HTTP/1.1 200 OK
Date: Thu, 09 May 2013 13:25:52 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
X-Powered-By: PHP/5.2.17
Content-Length: 25023
Content-Type: text/html

<html>
<head><...
```

Baris-baris ini sebagian besar merupakan header HTTP. HTTP/1.1 200 OK mengatakan bahwa server juga menggunakan HTTP versi 1.1 dan memberikan kode respons HTTP yang berhasil 200. Respons umum lainnya adalah 500 untuk Kesalahan Server Internal dan 404 untuk File Tidak Ditemukan. Kemudian dilanjutkan dengan HTML. Jika header Koneksi menentukan “tetap hidup” maka telnet akan menunggu permintaan Anda berikutnya dan Anda harus mengetikkan Control +] lalu “keluar” untuk keluar. Jika header Koneksi mengatakan "dekat" maka itu akan selesai dengan sendirinya dan mengatakan "Koneksi ditutup oleh host asing" di bagian bawah.

Menemukan Situs Web Anda

Kode 200 berarti beranda Anda baik-baik saja, dan Anda seharusnya dapat mengunjunginya di browser Anda. Namun, ini mungkin tidak menampilkan apa yang Anda harapkan, dan halaman Widget 3000 Anda yang luar biasa mungkin masih belum ada.

Host Dan Stream Virtual

Seperti disebutkan di atas, banyak server menghosting banyak situs web. Salah satunya adalah website bawaan. Ini adalah situs web yang Anda dapatkan ketika Anda mengunjungi server berdasarkan alamat IP http://80.72.139.101/ alih-alih berdasarkan nama, atau ketika Anda meninggalkan baris Host: dalam permintaan HTTP saat melakukan telnet. Situs web lainnya dikenal sebagai host virtual. Setiap situs web ini memiliki lokasi fisik di server yang dikenal sebagai root dokumennya. Untuk menyelidiki lebih lanjut masalah situs web Anda, Anda perlu menemukan akar dokumennya.

Untungnya dan masuk akal, sebagian besar paket manajemen server seperti Plesk menyimpan situs web yang dihosting secara virtual sesuai dengan nama domainnya, sehingga Anda biasanya dapat menemukannya langsung di nama domainnya. / pada perintah di bawah ini memberitahu find untuk mencari seluruh sistem file, -type d hanya mencari direktori, dan bagian -name mencari direktori mana pun yang berisi "smashingmagazine". Tanda bintang adalah kartu liar. Anda harus menghindarinya *smashingmagazine* atau memasukkannya ke dalam tanda kutip “*smashingmagazine*”:

```
$ find / -type d -name "*smashingmagazine*"
find: '/var/run/cups/certs': Permission denied
```

```
find: '/var/run/PolicyKit': Permission denied
/var/www/vhosts/smashingmagazine.com
/var/www/vhosts/smashingmagazine.com/httpdocs...
```

Jika Anda menjalankan perintah ini sebagai pengguna biasa yang tidak memiliki hak istimewa, Anda mungkin akan melihat banyak "Izin ditolak" saat find mencoba menjelajahi tempat terlarang. Anda sebenarnya melihat dua jenis keluaran di sini: stdout untuk keluaran standar dan stderr untuk kesalahan standar. Mereka disebut aliran keluaran dan dicampur secara membingungkan.

Anda telah menemukan simbol pipa | untuk menyalurkan aliran keluaran (stdout) dari satu perintah ke aliran masukan (stdin) perintah lainnya. Simbol > dapat mengarahkan output tersebut ke dalam file. Coba perintah ini untuk mengirim semua kecocokan ke dalam file bernama match.txt:

```
$ find / -type d -name ^"*smashingmagazine*" > matches.txt
find: '/var/run/cups/certs': Permission denied
find: '/var/run/PolicyKit': Permission denied...
```

Dalam hal ini, semua stdout dialihkan ke file match.tXt dan hanya aliran keluaran kesalahan stderr yang ditampilkan di layar. Dengan menambahkan nomor 2 Anda dapat mengarahkan ulang stderr ke dalam file dan hanya menampilkan stdout:

```
$ find / -type d -name ^"*smashingmagazine*" 2> matcherrors.txt
/var/www/vhosts/smashingmagazine.com
/var/www/vhosts/smashingmagazine.com/httpdocs...
```

Ada file khusus di komputer LinuX, UNIX dan Mac yang pada dasarnya adalah sebuah lubang hitam tempat barang dikirim dan menghilang. Ini disebut /dev/null, jadi untuk hanya melihat stdout dan mengabaikan semua kesalahan:

```
$ find / -type d -name ^"*smashingmagazine*" 2> /dev/null
/var/www/vhosts/smashingmagazine.com
/var/www/vhosts/smashingmagazine.com/httpdocs...
```

Hasil akhirnya adalah perintah find ini memberi tahu Anda secara kasar di mana akar dokumen Anda berada. Di Plesk, semua host virtual umumnya disimpan dalam direktori /var/www/vhosts, dengan akar dokumen di /var/www/vhosts/domain.com/httpdocs.

Jalan Panjang

Anda dapat menemukan root dokumen secara lebih akurat dengan melihat file konfigurasi. Untuk server Apache, Anda dapat menemukan root dokumen situs web default

dengan melihat file konfigurasi utama yang biasanya /etc/apache2/apache2.conf atau /etc/httpd/conf/httpd.conf.

```
$ grep DocumentRoot /etc/httpd/conf/httpd.conf
DocumentRoot "/var/www/html"
```

Di suatu tempat di dalam file conf ini juga akan ada baris Sertakan yang mereferensikan file conf lainnya, yang mungkin menyertakan file conf lebih lanjut. Untuk menemukan DocumentRoot untuk host virtual Anda, Anda harus mencari semuanya. Anda dapat melakukan ini menggunakan grep dan find tetapi perintahnya panjang, jadi kami akan membangunnya secara bertahap.

Pertama, kita akan menemukan semua file (karena -type f) di seluruh server (/) yang namanya diakhiri dengan “conf” atau “include”. -type f hanya menemukan file dan -o memungkinkan kita mencari file yang diakhiri dengan “conf” atau “include”, dengan tanda kurung di sekelilingnya. Seperti di atas, kesalahan dibuang begitu saja:

```
$ find / -type f \( -name \*conf -o -name \*include \) 2>
/dev/null
/var/spool/postfix/etc/resolv.conf
/var/some file with spaces.conf
/var/www/vhosts/myserv.com/conf/last_httpd.include...
```

Ini tidak cukup lengkap karena file apa pun dengan spasi akan membingungkan perintah grep yang akan kita coba. Untuk memperbaikinya Anda dapat menyalurkan output dari perintah find melalui perintah sed yang memungkinkan Anda menentukan ekspresi reguler. Ekspresi reguler adalah topik yang sangat besar. Pada perintah di bawah ini, s/ /\\ /g akan mengganti semua spasi dengan garis miring diikuti spasi:

```
$ find / -type f \( -name \*conf -o -name \*include \)
2>/dev/null | sed
's/ /\\ /g'
/var/spool/postfix/etc/resolv.conf
/var/some\ file\ with\ spaces.conf
/var/www/vhosts/myserv.com/conf/last_httpd.include...
```

Sekarang Anda dapat menggunakan backtick untuk menyematkan hasil perintah find tersebut ke dalam perintah grep. Menggunakan ` berbeda dengan | karena ini sebenarnya membantu membangun sebuah perintah, bukan hanya memanipulasi inputnya. Opsi -H untuk grep memberitahukannya jadi tampilkan nama file juga. Jadi, sekarang kita akan mencari referensi ke "smashingmagazine" di file conf mana pun.

```
$ grep -H smashingmagazine `find / -type f \( -name \*conf -o -
name \*include \) 2> /dev/null | sed 's/ /\\ /g'`
/var/www/vhosts/smashingmagazine.com/conf/last_httpd.include:
ServerName
"smashingmagazine.com"...
```

Ini mungkin memerlukan waktu beberapa detik untuk dijalankan. Ia menemukan setiap file conf di server dan mencari "smashingmagazine" di semua file tersebut. Ini mungkin mengungkapkan DocumentRoot secara langsung. Jika tidak, setidaknya itu akan menampilkan file di mana Nama Server atau VirtualHost ditentukan. Anda kemudian dapat menggunakan grep atau less untuk memeriksa file tersebut untuk DocumentRoot.

Anda juga dapat menggunakan perintah xargs untuk melakukan hal yang sama. Ini juga memungkinkan keluaran dari satu perintah untuk disematkan ke perintah lain:

```
$ find / -type f \( -name \*conf -o -name \*include \) 2> /dev/null | sed
's/ /\\ /g' | xargs grep -H smashingmagazine
/var/www/vhosts/smashingmagazine.com/conf/last_httpd.include: ServerName
"smashingmagazine.com"...
$grepDocumentRoot/var/www/vhosts/smashingmagazine.com/conf/last_httpd.inclu
de
DocumentRoot "/var/www/vhosts/smashingmagazine.com/httpdocs"
```

Mudah-mudahan, hasil akhirnya adalah Anda telah menemukan akar dokumen untuk situs web Anda secara pasti. Anda dapat menggunakan teknik serupa untuk nginx. Ia juga memiliki file conf utama, biasanya di /etc/nginx/nginx.conf, dan dapat juga menyertakan file conf lainnya. Namun root dokumennya hanya disebut "root".

7.8 ANTARMUKA KONTROL APACHE

Dengan Apache, ada cara lain untuk menemukan file conf yang tepat, menggunakan perintah apachectl atau apache2ctl yang lebih baru dengan opsi -S.

```
$ apachectl -S
VirtualHost configuration:
80.72.139.101:80 is a NameVirtualHost
     defaultserverdefault(/usr/local/psa/admin/conf/generated/
13656495120.10089200_server.include:87)
     port80namevhostdefault(/usr/local/psa/admin/conf/generate
d/13656495120.10089200_server.include:87)
     port      80      namevhost      www.smashingmagazine.com
(/var/www/vhosts/
smashingmagazine.com/conf/last_httpd.include:10)...
```

Jika ini berjalan terlalu cepat, Anda dapat mencoba menyalurkan hasilnya melalui grep. Namun itu tidak akan berhasil, karena grep hanya beroperasi di stdout dan untuk beberapa

alasan apachectl mengeluarkan informasinya ke stderr. Jadi, Anda harus mengarahkan stderr terlebih dahulu ke stdout lalu mengirimkannya melalui grep. Hal ini dilakukan dengan mengarahkan aliran kesalahan 2 ke aliran keluaran 1 dengan 2>&1, seperti ini:

```
$ apachectl -S 2>&1 | grep smashingmagazine
     port          80          namevhost          smashingmagazine.com
(/var/www/vhosts/smashingmagazine.com/conf/13656495330.08077300_http
d.include:10)
```

Ini juga menampilkan file conf yang berisi DocumentRoot untuk situs web ini. Seperti di atas, grep lebih lanjut atau kurang akan mengungkapkan DocumentRoot.

Memeriksa Akar Dokumen

Sekarang setelah Anda menemukan root dokumen, Anda dapat mengintip untuk memastikan semuanya baik-baik saja. Ubah ke direktori dengan cd:

```
$ cd /var/www/vhosts/smashingmagazine.com/httpdocs
bash: cd: /var/www/vhosts/smashingmagazine.com/httpdocs: No
such file or
directory
```

Jika Anda mendapatkan pesan kesalahan “Tidak ada file atau direktori”, itu adalah berita buruk. Entah DocumentRoot salah disetel atau seluruh situs web Anda telah dihapus. Jika ada, Anda dapat membuat daftar file dengan ls. -a juga menampilkan file tersembunyi yang dimulai dengan titik, dan -l menampilkannya dalam format panjang dengan izin dan tanggal:

```
$ ls -al
drwxrwxrwx 8 nobody nogroup 4096 May 9 14:03 .
drwxr-xr-x 14 root root 4096 Oct 13 2012 ..
```

Setiap folder setidaknya akan menampilkan dua entri ini. Tunggal "." adalah untuk direktori saat ini dan “..” untuk direktori induk. Jika hanya itu yang ada, maka direktori tersebut kosong. Saat Anda berada di sana, Anda dapat memeriksa ulang apakah Anda berada di tempat yang benar. Buat file baru menggunakan echo dan gunakan lagi simbol > untuk mengirim output ke file.

```
$ echo “<h1>My test file</h1>” > testfile.html
```

Ini akan membuat file bernama testfile.html yang berisi sedikit HTML. Anda dapat menggunakan browser atau telnet atau curl atau wget untuk melihat apakah file tersebut berada di tempat yang seharusnya.

```
$ curl http://www.smashingmagazine.com/testfile.html
```

```
<h1>My test file</h1>
```

Jika berhasil, selamat, Anda telah menemukan situs web Anda! Hapus file tes itu untuk membersihkannya sendiri dengan rm testfile.html dan lanjutkan.

Cadangan Dan Kembalikan

Perintah tar dan zip dapat digunakan untuk membuat cadangan dan memulihkan. Jika situs web Anda hilang, pemulihan tidak akan banyak membantu Anda kecuali Anda telah membuat cadangan sebelumnya. Jadi kembali ke masa lalu dan buat cadangan data Anda dengan salah satu perintah di bawah ini. Untuk kembali sepanjang hari:

```
$ gobackintime 86400
It is now Sat May 10 20:30:57 BST 2013
```

Hanya bercanda — tapi itu akan menyenangkan! Perintah tar adalah singkatan dari tape archive dan berasal dari masa ketika data dicadangkan pada pita magnetik. Untuk membuat arsip direktori, berikan opsi cfz ke tar yang akan membuat arsip baru dalam sebuah file dan kemudian zip dalam format gzip.

```
$ tar cfz backupfile.tgz /var/www/vhosts/smashingmagazine.com/httpdocs
tar: Removing leading `/' from member names
```

Semua komputer Mac dan LinuX mendukung perintah tar dan sebagian besar juga memiliki zip. Untuk melakukan hal yang sama dengan zip:

```
$ zip -r backupfile.zip /directory/to/backup
```

Untuk melihat isi arsip, jalankan:

```
tar tfz backupfile.tgz
var/www/vhosts/smashingmagazine.com/httpdocs/
var/www/vhosts/smashingmagazine.com/httpdocs/.htaccess...
```

Atau untuk format zip:

```
unzip -l backupfile.zip
Archive: test.zip
 Length Date Time Name
--------- ---------- ----- ----
 0 2012-05-28 00:33 var/www/vhosts/smashingmagazine.com/httpdocs
 234                        2012-05-28                          00:33
var/www/vhosts/smashingmagazine.com/httpdocs/.htaccess
```

Baik tar maupun zip menghapus garis miring di depan saat dicadangkan. Jadi ketika Anda memulihkan file, file tersebut akan dikembalikan ke direktori saat ini. Untuk memulihkannya di lokasi yang sama tempat mereka membuat cadangan, pertama-tama pulihkan mereka di direktori saat ini dan kemudian pindahkan ke tempatnya dengan mv.

```
$ tar xfzv backupfile.tgz
var/www/vhosts/smashingmagazine.com/httpdocs/...
```

Huruf “v” di atas adalah singkatan dari verbose dan menyebabkan tar menunjukkan apa yang dilakukannya. zip memiliki opsi serupa:

```
$ unzip -v backupfile.zip
Archive: backupfile.zip
 Length Method Size Cmpr Date Time CRC-32 Name
-------- ------ ------- ---- ---------- ----- -------- ----
 0 Stored 0 0% 2012-05-28 00:33 00000000 var/www/vhosts/
smashingmagazine.com/httpdocs/...
```

7.9 KESALAHAN SITUS WEB

Anggaplah situs web Anda belum benar-benar hilang. Tempat berikutnya untuk mencari adalah file log kesalahan.

Menemukan File Log

Saat menggunakan paket manajemen server seperti Plesk, setiap situs web mungkin memiliki file lognya sendiri. Anda dapat menemukannya dengan mencari kata "log" di file conf yang Anda identifikasi di atas. -i berarti tidak peka huruf besar-kecil.

```
$ grep -i log /var/www/vhosts/smashingmagazine.com/conf/last_httpd.include
 CustomLog   /var/www/vhosts/smashingmagazine.com/statistics/logs/access_log
plesklog
 ErrorLog
"/var/www/vhosts/smashingmagazine.com/statistics/logs/error_log"...
```

Ada juga log seluruh server tempat terjadinya kesalahan yang tidak spesifik pada situs web. Anda dapat menemukannya di file conf utama:

```
$ grep -i log /etc/apache2/apache2.conf
ErrorLog /var/log/apache2/error.log...
```

Kesalahan Htaccess

Sangat mudah untuk mengacaukan situs web. Anda dapat dengan mudah menjatuhkan situs web yang sangat besar dengan menghapus satu karakter dari file .htaccess. Apache menggunakan file .htaccess untuk menyediakan opsi konfigurasi menit-menit terakhir untuk sebuah situs web. Ini paling sering digunakan untuk aturan penulisan ulang URL yang terlihat seperti ini:

```
RewriteRule ^products/.*/([0-9]+)$ products/view.php?id=$1
[L,QSA]
```

Aturan ini menyatakan untuk menulis ulang URL apa pun dalam bentuk "products/widget-3000/123" menjadi URL sebenarnya "products/view.php?id=123". L berarti ini adalah aturan terakhir yang diterapkan dan QSA berarti Apache harus melampirkan string kueri apa pun ke URL baru.

Penulisan ulang URL sering digunakan untuk optimasi mesin pencari sehingga pengelola Web dapat memasukkan nama produk ke dalam URL tanpa harus membuat direktori yang disebut "widget-3000". Namun, jika salah ketik saja, seluruh situs web Anda akan menampilkan 500 Internal Server Error.

Perintah tail akan menampilkan 10 baris terakhir file log. Berikan -1 untuk menampilkan satu baris terakhir. Masalah .htaccess akan terlihat seperti ini:

```
$ tail -1 /var/www/vhosts/smashingmagazine.com/statistics/logs/error_log
[Thu May 06 11:04:00 2013] [alert] [client 81.106.118.59] /var/www/
vhosts/smashingmagazine.com/httpdocs/.htaccess: Invalid command
'RewiteRule', perhaps misspelled or defined by a module not included in the
server configuration.
```

Atau berikan opsi -f untuk mengikuti file log, menampilkan entri log tambahan apa pun saat terjadi:

```
$ tail -f /var/www/vhosts/smashingmagazine.com/statistics/logs/error_log...
```

Anda dapat memahami semua jenis kesalahan berikut:

```
$ grep alert /var/www/vhosts/smashingmagazine.com/statistics/logs/error_log
[Thu May 06 11:04:00 2013] [alert] [client 81.106.118.59]...
```

KESALAHAN PARSE DAN Runtime PHP

Banyak situs web yang menggunakan kombinasi LAMP: LinuX, Apache, MySQL dan PHP. Alasan umum halaman Web tidak muncul adalah karena terdapat kesalahan PHP. Untungnya, hal ini cukup mudah ditemukan dan ditentukan.

Ada dua kelas besar kesalahan PHP: kesalahan parse dan kesalahan runtime. Kesalahan parse adalah kesalahan sintaksis dan termasuk menghilangkan titik koma atau melupakan $ di depan nama variabel. Kesalahan runtime mencakup fungsi yang tidak ditentukan atau objek referensi yang tidak ada.

Seperti kesalahan .htaccess, kesalahan parse akan menyebabkan kode respons HTML 500 untuk Kesalahan Server Internal, seringkali dengan halaman HTML yang benar-benar kosong. Kesalahan runtime akan memberikan respons HTML sukses sebesar 200 dan akan menampilkan HTML sebanyak yang telah diproses (dan dihapus) sebelum kesalahan terjadi.

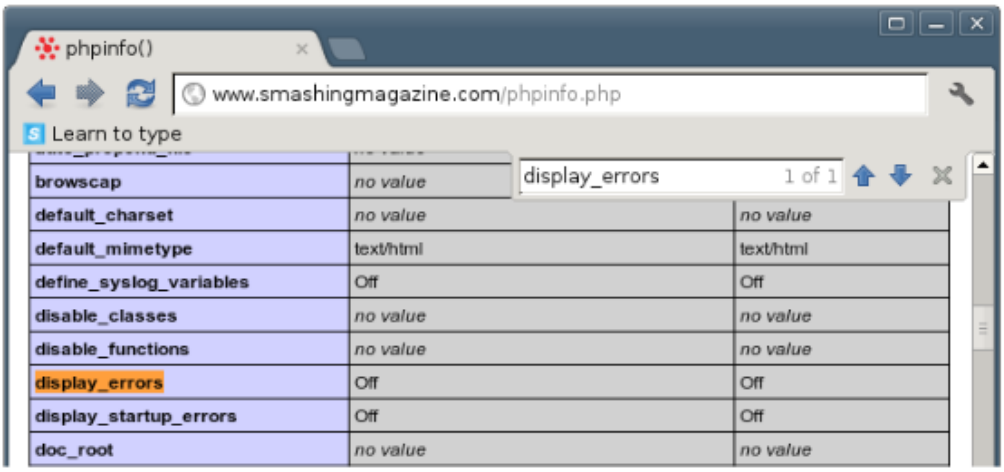
Anda dapat menggunakan telnet atau wget -S atau curl -i untuk mendapatkan header dari URL saja. Jadi sekarang, salin dan tempel halaman Anda yang salah ke dalam perintah:

```
$ curl -i http://www.smashingmagazine.com/products/widget-3000/123
HTTP/1.0 500 Internal Server Error
Date: Sun, 12 May 2013 17:44:49 GMT
Server: Apache
Vary: Accept-Encoding
Content-Length: 0
Connection: close
Content-Type: text/html
```

Pengaturan Kesalahan Php

Untuk menemukan kesalahan sebenarnya, Anda perlu memastikan kesalahan dilaporkan dalam file log.

Ada beberapa pengaturan PHP yang menutupi kesalahan. display_errors menentukan apakah kesalahan ditampilkan kepada pengunjung situs web atau tidak, dan log_errors menentukan apakah kesalahan tersebut akan muncul di file log. error_reporting menentukan jenis kesalahan yang dilaporkan: hanya kesalahan fatal, misalnya, atau peringatan dan pemberitahuan juga. Semua ini dapat diatur dalam file konfigurasi, di .htaccess atau di dalam skrip PHP itu sendiri.



Gambar 8.5 fungsi phpinfo menampilkan pengaturan konfigurasi.

Anda dapat mengetahui pengaturan Anda saat ini dengan menjalankan fungsi PHP phpinfo. Buat file PHP yang memanggil fungsi tersebut dan kunjungi di browser Anda:

```
$ echo "<?php phpinfo()?>" > /var/www/vhosts/smashingmagazine.com/httpdocs/phpinfo.php
```

Kedua kolom menunjukkan situs web dan pengaturan seluruh server. Ini menunjukkan bahwa display_errors tidak aktif, dan ini bagus, karena seharusnya tidak aktif di situs web yang aktif. Artinya tidak ada kesalahan PHP yang akan dilihat oleh pengunjung biasa. log_errors di sisi lain harus aktif. Ini sangat berguna untuk men-debug masalah PHP.

Nilai error_reporting adalah 30719. Angka ini mewakili bendera bit atau bidang bit. Ini adalah teknik untuk menyimpan beberapa nilai ya/tidak dalam satu nomor. Dalam PHP terdapat serangkaian konstanta yang mewakili berbagai jenis kesalahan.18 Misalnya, konstanta E_ERROR untuk kesalahan fatal dan memiliki nilai 1; E_WARNING untuk peringatan dan sama dengan 2; E_PARSE adalah untuk parsing atau kesalahan sintaksis dan memiliki nilai 4. Semua nilai ini merupakan pangkat dua dan dapat dijumlahkan dengan aman. Jadi angka 7 berarti ketiga jenis kesalahan tersebut harus dilaporkan, karena E_ERROR + E_WARNING + E_PARSE = 7. Nilai 5 hanya akan melaporkan E_ERROR + E_PARSE.

Faktanya, ada 16 jenis kesalahan dari 1 untuk E_ERROR hingga 16384 untuk E_USER_DEPRECATED. Anda dapat mengetikkan “30719 dalam biner” ke Google dan itu akan memberi Anda setara biner: 0b111011111111111. Artinya semua error diaktifkan kecuali yang kedua belas, yaitu E_STRICT. Setup khusus ini juga telah diberikan konstanta E_ALL = E_ERROR + E_ WARNING + E_PARSE +etc = 30719. Dari PHP versi 5.4.0, E_ALL sebenarnya adalah 32767 yang mencakup semua kesalahan termasuk E_STRICT.

Jika pengaturan error_reporting Anda adalah 0, maka tidak ada kesalahan yang akan muncul di file log. Anda dapat mengubah pengaturan ini di file php.ini, tetapi kemudian Anda harus me-restart Apache agar berpengaruh. Cara yang lebih mudah untuk mengubah pengaturan ini di Apache adalah dengan menambahkan baris dalam file bernama .htaccess di root dokumen Anda: php_value error_reporting 30719.

Atau Anda dapat melakukannya pada baris perintah, menggunakan panah ganda yang menambahkan file yang sudah ada atau membuat file jika tidak ada:

```
$ echo "php_value error_reporting 30719" >> .htaccess
$ echo "php_value log_errors On” >> .htaccess
```

Segarkan halaman Web Anda yang salah. Jika ada kesalahan PHP di halaman Anda, kesalahan tersebut seharusnya muncul di log kesalahan. Anda dapat mengambil log untuk semua kesalahan PHP:

```
Grep                                                                PHP
/var/www/vhosts/smashingmagazine.com/statistics/logs/error_log
[Sun May 12 18:19:09 2013] [error] [client 81.106.118.59] PHP Notice:
Undefined              variable:              total              in
/var/www/vhosts/smashingmagazine.com/httpdocs/products/view.php    on
line 10...
```

Jika Anda telah mereferensikan variabel atau indeks array sebelum menetapkan nilainya, Anda mungkin melihat ribuan pemberitahuan PHP seperti di atas. Itu terjadi ketika

Anda melakukan hal-hal seperti <? $total = $total + 1 ?> tanpa menyetel $total ke 0 terlebih dahulu. Mereka berguna untuk menemukan potensi bug, namun bukan merupakan penghenti pertunjukan. Situs web Anda seharusnya tetap berfungsi.

Anda mungkin mendapat begitu banyak pemberitahuan dan peringatan seperti ini sehingga kesalahan sebenarnya bisa hilang. Anda dapat mengubah error_reporting menjadi 5 untuk hanya menampilkan E_ ERROR dan E_PARSE atau Anda dapat menerima secara khusus jenis kesalahan tersebut. Sangat umum untuk menyatukan perintah grep seperti ini ketika Anda ingin memfilter berdasarkan banyak hal. Opsi -e di bawah ini memberitahu grep kedua untuk menggunakan ekspresi reguler. Perintah ini menemukan semua entri log yang berisi "PHP" dan "Parse" atau "Fatal".

```
$ grep PHP /var/www/vhosts/smashingmagazine.com/statistics/logs/error_log
| grep -e "Parse\|Fatal"
[Thu Jul 19 12:26:23 2012] [error] [client 81.106.118.59] PHP Fatal error:
Class 'Product' not found in /var/www/vhosts/smashingmagazine.com/
httpdocs/library/class.product.php on line 698
[Sun May 12 18:16:21 2013] [error] [client 81.106.118.59] PHP Parse error:
syntax error, unexpected T_STRING in
/var/www/vhosts/smashingmagazine.com/httpdocs/products/view.php on line
100...
```

MELIHAT KESALAHAN DI BROWSER

Jika Anda menelusuri kesalahan runtime dan bukan kesalahan parse, Anda juga dapat mengubah pengaturan error_reporting secara langsung di PHP. Dan Anda dapat dengan cepat mengaktifkan display_errors sehingga Anda akan melihat kesalahan langsung di browser Anda. Hal ini membuat proses debug menjadi lebih cepat, namun berarti semua orang juga dapat melihat kesalahan tersebut. Tambahkan baris ini ke bagian atas halaman PHP Anda:

```
<? ini_set ('display_errors', 1); error_reporting (E_ERROR |
E_WARNING); ?>
```

Kedua fungsi ini mengubah dua pengaturan PHP. | dalam panggilan pelaporan error_ adalah operator bit OR. Ini secara efektif melakukan hal yang sama seperti + di atas tetapi beroperasi pada bit, begitu pula operator yang tepat untuk digunakan dengan flag bit.

Kesalahan atau peringatan fatal apa pun nanti di halaman PHP kini akan ditampilkan langsung di browser. Teknik ini tidak akan berfungsi untuk kesalahan parse karena tidak ada halaman yang akan berjalan jika ada kesalahan parse.

Bendera Bit

Menggunakan tanda bit untuk error_reporting menghindari 15 argumen terpisah pada fungsi untuk setiap jenis kesalahan. Bendera bit juga dapat berguna dalam kode Anda sendiri. Untuk menggunakannya, Anda perlu mendefinisikan beberapa konstanta, gunakan operator bit OR | saat memanggil fungsi dan bit AND operator & di dalam fungsi. Berikut adalah contoh

PHP sederhana yang menggunakan bit flag untuk memberi tahu fungsi bernama showproduct properti produk mana yang akan ditampilkan:

```
<?
define ('PRODUCT_NAME', 1);
define ('PRODUCT_PRICE', 2);
function showproduct ($product, $flags) {
 if ($flags & PRODUCT_NAME) echo $product['name'];
 if ($flags & PRODUCT_PRICE) echo ': $' . $product['price'];
}
$product = array ('name'=>'Widget 3000', 'price'=>10);
showproduct ($product, PRODUCT_NAME | PRODUCT_PRICE);
?>
```

Ini akan menampilkan “Widget 3000: $10” di browser. Ini adalah contoh bit flag yang agak dangkal. Biasanya terdapat proses sistem yang jauh lebih dalam dan lebih konstan.

Loop Tak Terbatas

Pelaporan kesalahan PHP mungkin bermasalah dengan satu jenis kesalahan: loop tak terbatas. Sebuah loop mungkin terus dijalankan hingga mencapai batas waktu PHP, yang biasanya 30 detik (pengaturan max_execution_time PHP), menyebabkan kesalahan fatal. Atau jika loop mengalokasikan variabel baru atau memanggil fungsi, loop mungkin terus berjalan hingga PHP kehabisan memori yang bisa diterapkan (pengaturan memory_limit PHP).

Namun, hal ini mungkin menyebabkan proses anak Apache terhenti, yang berarti tidak ada yang dilaporkan, dan Anda hanya akan melihat halaman kosong atau sebagian. Jenis kesalahan ini semakin jarang terjadi, karena PHP dan Apache sekarang sudah sangat matang dan dapat mendeteksi serta menangani masalah yang tidak dapat diatasi seperti ini. Namun jika Anda akan membenturkan kepala ke dinding karena frustrasi karena tidak ada satu pun cara di atas yang berhasil, pertimbangkanlah. Jauh di dalam kode Anda, Anda mungkin memiliki fungsi yang memanggil beberapa fungsi lain, yang memanggil fungsi asli dalam rekursi tak terbatas.

Debugger

Jika Anda sudah sampai sejauh ini, dan halaman Anda masih belum muncul, Anda memasuki wilayah yang lebih sulit. PHP Anda mungkin dijalankan secara valid dan melakukan segala sesuatu yang seharusnya, namun ada beberapa kesalahan logis dalam pemrograman Anda. Untuk debugging cepat, Anda dapat memasukkan variabel var_dump ke browser, mungkin membungkusnya dalam pernyataan if sehingga hanya alamat IP Anda yang melihatnya:

```
<? if ($_SERVER['REMOTE_ADDR'] == '85.106.118.199') var_dump
($product); ?>
```

Metode ini akan mempersempit kesalahan namun tidak dapat dipahami dan rawan kesalahan, jadi Anda mungkin mempertimbangkan alat debugging seperti Xdebug atau FirePHP. Mereka dapat memberikan banyak informasi, dan juga dapat berjalan tanpa terlihat oleh pengguna, menyimpan keluarannya ke file log. Xdebug dapat digunakan seperti ini:

```
<?
ini_set ('xdebug.collect_params', 1);
xdebug_start_trace ('/tmp/xdebugtrace');
echo "This will get traced.";
xdebug_stop_trace();
?>
```

Sedikit kode ini mencatat semua panggilan fungsi dan argumen ke file /tmp/xdebugtrace.txt. Ini menampilkan lebih banyak informasi ketika ada pemberitahuan atau kesalahan PHP. Namun, overhead mungkin tidak cocok untuk lingkungan live, dan perlu diinstal di server, sehingga mungkin tidak tersedia di sebagian besar lingkungan hosting.

FirePHP, di sisi lain, adalah perpustakaan PHP yang berinteraksi dengan add-on pada Firebug, sebuah plugin untuk FirefoX. Anda dapat mengeluarkan informasi debug dan menumpuk jejak dari PHP ke konsol Firebug.

Masalah Keamanan

Pada titik ini, Anda seharusnya sudah memiliki beberapa HTML yang menjangkau browser Anda. Jika tidak seperti yang Anda harapkan, ada kemungkinan situs web Anda telah disusupi. Jangan tersinggung (pada awalnya). Ada banyak jenis peretasan dan sebagian besar dilakukan secara otomatis. Seseorang yang pintar namun tidak bermoral telah menulis sebuah program yang mendeteksi kerentanan dan mengeksploitasinya.

Tujuan eksploitasi mungkin hanya untuk mengirim spam, atau menggunakan server Anda sebagai bagian dari serangan yang lebih besar terhadap target yang lebih spesifik (DDoS).

Peretasan Server

Sistem operasi adalah perangkat lunak yang sangat kompleks. Mereka mungkin dibangun dari jutaan baris kode pemrograman. Mereka kemungkinan besar memiliki celah di mana mengirimkan pesan yang salah pada waktu yang salah akan menyebabkan semacam kesalahan yang memungkinkan seseorang atau sesuatu untuk masuk. Itu sebabnya Microsoft, Apple, Ubuntu, dan lainnya terus merilis pembaruan.

Demikian pula, Apache, nginx, IIS dan semua perangkat lunak lain di server biasa sangatlah rumit. Hal terbaik yang dapat Anda lakukan adalah selalu memperbarui patch terbaru. Kebanyakan tuan rumah yang baik akan melakukan ini untuk Anda. Seorang peretas dapat menggunakan kelemahan ini untuk masuk ke server Anda dan merekayasa sendiri sesi terminal. Mereka mungkin awalnya mendapatkan akses sebagai pengguna yang tidak memiliki hak istimewa dan kemudian mencoba peretasan lebih lanjut untuk menjadi pengguna root. Anda harus membuat ini sesulit mungkin dengan menggunakan kata sandi yang baik, izin yang terbatas, dan berhati-hati dalam menjalankan perangkat lunak (seperti Apache) sebagai pengguna yang tidak memiliki hak istimewa.

Jika seseorang mendapatkan akses, mereka mungkin meninggalkan sedikit perangkat lunak yang nantinya dapat mereka gunakan untuk mengambil kendali server Anda. Ini mungkin dapat dideteksi oleh pemindai anti-virus atau sesuatu seperti Rootkit Hunter, yang mencari anomali seperti file tersembunyi yang tidak terduga. Namun ada juga beberapa hal yang dapat Anda lakukan jika Anda mencurigai adanya gangguan.

Perintah w menunjukkan siapa yang sedang login ke server dan apa yang mereka lakukan:

```
$ w
 20:44:32 up 44 days, 7:51, 2 users, load average: 0.07, 0.03,
0.05
USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT
root pts/0 cpc1-brig17-2-0- 17:54 1:02m 0.15s 0.13s -bash
root pts/1 cpc1-brig17-2-0- 20:44 0.00s 0.02s 0.00s w...
```

Perintah terakhir menunjukkan siapa yang baru saja login sesuai urutan tanggal. Masukkan melalui head untuk hanya menampilkan 10 baris pertama.

```
$ last
paul pts/0 :0.0 Sun May 12 17:21 still logged in
paul tty7 :0 Sun May 12 17:20 still logged in
reboot system boot 2.6.32-41-386 Sun May 12 17:18 - 20:48 (03:29)
fred tty7 :0 Sat May 11 10:10 - down (01:12)
```

Ini memberi tahu Anda siapa yang telah login dan berapa lama, ditambah sesi terminal apa pun yang mereka buka. down artinya sampai server dimatikan. Cari entri yang tidak terduga dan konsultasikan dengan host Anda atau pakar keamanan jika Anda ragu.

Peretasan Php

Yang lebih umum adalah peretas yang mendapatkan akses melalui kerentanan dalam skrip PHP, terutama sistem manajemen konten populer seperti WordPress. Siapa pun dapat menulis plugin untuk WordPress dan, jika berguna, orang akan menginstalnya. Saat menulis plugin, sebagian besar pengembang hanya memikirkan fungsionalitas dan sedikit memikirkan keamanan. Dan karena WordPress mengizinkan pengunggahan file, peretas yang menemukan kerentanan dapat menggunakannya untuk mengunggah skrip PHP mereka sendiri dan kemudian mengambil kendali komputer.

Skrip PHP ini dapat menggunakan fungsi email PHP untuk mengirimkan spam sesuai permintaan, namun skrip tersebut juga dapat mencoba menjalankan perintah dengan cara yang sama seperti yang Anda bisa lakukan melalui sesi terminal. PHP dapat menjalankan perintah dengan fungsi exec atau sistemnya. Jika Anda tidak perlu menggunakan fungsi-fungsi ini, disarankan untuk menonaktifkannya. Anda dapat melakukan ini dengan menambahkan direktifdisable_functions ke file php.ini server Anda (atau php5.ini untuk PHP 5) atau ke file

php.ini di dalam root dokumen Anda. Jika Anda mencari “fungsi nonaktifkan php” di Google, Anda akan menemukan daftar lengkap fungsi yang harus dinonaktifkan dengan cara ini:

```
disable_functions=fpassthru,crack_check,crack_close...
```

Pemeriksaan cepat yang dapat Anda lakukan untuk jenis peretasan ini adalah dengan mencari semua file PHP yang baru saja dimodifikasi dan memastikan tidak ada anomali. Opsi -mtime -1 memberitahu find untuk hanya mempertimbangkan file yang diubah dalam hari terakhir. Ada juga -mmin untuk beberapa menit. Perintah ini mencari semua situs web di /var/www/vhosts untuk file yang baru saja dimodifikasi yang diakhiri dengan “php” atau “inc”:

```
$ find /var/www/vhosts -mtime -1 \( -name \*php -o -name \*inc \) -
printf
"%t %h/%f\n"
Sun          May          12          21:20:17.0000000000          2013
/var/www/vhosts/smashingmagazine.com/
httpdocs/products/view.php
```

Peretasan PHP sulit dideteksi karena dirancang agar tidak menonjol. Salah satu metode yang digunakan peretas adalah dengan melakukan gzip PHP mereka dan kemudian menyandikannya sebagai base64. Dalam hal ini, Anda mungkin memiliki file PHP di sistem Anda dengan sesuatu seperti ini di dalamnya:

```
eval(gzinflate(base64_decode('HJ3HkqNQEkU/ZzqCBd4t8V4YAQI2E3jvPV8...
```

Metode lain adalah dengan menyandikan teks di dalam variabel lalu menggabungkannya dan mengevaluasinya:

```
$unywlbxc = " uwzsebpgi840hk2a jf";
$hivjytmne = " jqs9m4y 1znp0 ";
eval ( "m"."i". "croti"...
```

Kedua metode ini menggunakan fungsi PHP eval, sehingga Anda dapat menggunakan grep untuk mencari eval. Menggunakan ekspresi reguler dengan \beval\b berarti kata “eval” harus memiliki batas kata sebelum dan sesudahnya, sehingga tidak dapat ditemukan di tengah kata. Anda dapat menggabungkan ini dengan perintah find di atas dan menyalurkan lebih sedikit agar mudah dibaca:

```
$ find /var/www/vhosts -mtime -1 \( -name \*php -o -name \*inc \) | sed
's/ /\\ /g' | xargs grep -H -e "\beval\b" | less
/var/www/vhosts/config.php:eval(gzinflate(base64_decode('HJ3HkqNQE...
```

Jika Anda menemukan jenis peretasan ini di situs web Anda, cobalah mencari tahu bagaimana mereka bisa masuk sebelum menghapus seluruh file yang tercemar.

8.10 LOG AKSES

Selain log kesalahan, Apache juga menyimpan log akses. Anda dapat menelusurinya untuk mencari aktivitas mencurigakan. Misalnya, jika Anda menemukan peretasan PHP di dalam file yang tampak tidak berbahaya bernama test.php, Anda dapat mencari semua aktivitas yang terkait dengan file tersebut. Log akses biasanya berada di samping log kesalahan dan ditentukan dengan arahan CustomLog di file konfigurasi Apache. Ini berisi alamat IP, tanggal dan file yang diminta. Telusuri dengan grep:

```
$ grep -e "\(GET\|POST\) /test.php" /var/www/vhosts/smashingmagazine.com/
statistics/logs/error_log
70.1.5.12 - - [12/May/2013:20:10:49 +0100] "GET /test.php HTTP/1.1" 200
1707 "-" "Mozilla/5.0 (X11; Ubuntu; Linux i686;...
```

Ini mencari permintaan GET dan POST untuk file test.php. Ini memberi Anda alamat IP, sehingga Anda sekarang dapat mencari semua akses lain berdasarkan alamat ini, dan juga mencari tanggal tertentu:

```
$ grep 70.1.5.12
/var/www/vhosts/smashingmagazine.com/statistics/logs/error_log | grep
"12/May/2013"
70.1.5.12 - - [12/May/2013:20:10:49 +0100] "GET /products/view.php?something
HTTP/1.1" 200 1707 "-"...
70.1.5.12 - - [12/May/2013:20:10:49 +0100] "GET /test.php HTTP/1.1" 200
1707 "-" "Mozilla/5.0 (X11; Ubuntu; Linux i686;...
```

Proses debug semacam ini juga bisa sangat berguna untuk kesalahan situs web normal. Jika Anda memiliki formulir umpan balik di situs web Anda, tambahkan alamat IP pengguna ke pesan tersebut. Jika seseorang melaporkan kesalahan, nanti Anda dapat melihat log untuk melihat apa yang telah mereka lakukan. Hal ini jauh lebih baik daripada mengandalkan informasi yang tidak jelas mengenai permasalahan yang dilaporkan.

Ini juga berguna untuk mendeteksi serangan injeksi SQL, di mana peretas mencoba mengekstrak detail dari database Anda dengan mengelabui fungsi pengambilan database Anda. Hal ini sering kali melibatkan banyak trial and error. Anda dapat mengirim email kepada diri Anda sendiri setiap kali permintaan database mengalami kesalahan dan menyertakan alamat IP pengguna. Anda kemudian dapat melakukan referensi silang dengan log tersebut untuk melihat apa lagi yang telah mereka coba.

Resor Terakhir

William Edward Hickson dipuji karena mempopulerkan pepatah: “Jika pada awalnya Anda tidak berhasil, cobalah, coba, coba lagi.”19 Hickson adalah seorang penulis pendidikan Inggris yang hidup pada awal zaman Victoria. Nasihatnya tidak sesuai untuk pengembang Web

modern, yang berbaring di tempat tidur pada Sabtu pagi, tenggelam dalam rasa frustrasi, menatap halaman Web yang kosong, bersiap melemparkan laptop mahal ke dinding bata.

Anda sekarang telah melalui semua saran di atas. Anda telah memeriksa bahwa dunia belum berakhir, memverifikasi kotak broadband Anda, menguji Internet dan mencapai server Anda. Anda telah mencari masalah perangkat keras dan masalah perangkat lunak, dan mempelajari kode PHP. Tapi entah kenapa, Widget 3000 Anda masih belum ada. Hal berikutnya yang harus dilakukan adalah...

Sarapan

Bangunlah dari tempat tidur dan alihkan pikiran Anda dari masalah untuk sementara waktu. Makanlah roti panggang, semangkuk sereal, minuman. Bahkan mungkin mandi. Cobalah sampo lavender dan jeruk baru yang Anda beli secara tidak sengaja. Saat Anda melakukan semua ini, alam bawah sadar Anda sedang sibuk menangani masalah situs web, dan mungkin secara tidak terduga memunculkan solusi dalam pikiran Anda. Jika demikian, cobalah. Jika tidak…

Meminta Bantuan

Periksa tingkat dukungan yang berhak Anda dapatkan dari perusahaan hosting Anda. Jika Anda membayar $10 per bulan, itu mungkin tidak banyak. Anda mungkin bisa membuat mereka melirik sekilas ke arah Anda dalam 72 jam ke depan. Jika jumlahnya lebih banyak, mereka dapat masuk dan melihatnya dalam beberapa menit ke depan. Mereka harus dapat membantu masalah perangkat keras atau perangkat lunak. Mereka tidak akan membantu masalah pemrograman Web. Alternatifnya, hubungi kolega atau pekerja lepas. Jika Anda masih terjebak…

Mempersiapkan

…untuk melepaskan energi gugup. Temukan salah satu bola cumi yang bisa Anda remas tanpa ampun di tangan Anda, atau beberapa pensil untuk digunakan sebagai stik drum, atau sebungkus rokok dan sepoci penuh kopi. Dan kemudian coba upaya terakhir untuk masalah komputasi apa pun…

Menyalakan Ulang

Saat laptop atau desktop Anda bermasalah, solusi umum adalah dengan melakukan reboot. Anda dapat mencoba trik yang sama di server Web Anda. Hal ini merupakan hal yang cukup beresiko. Pertama, hal ini mungkin tidak menyelesaikan masalah. Jika itu kesalahan PHP, maka tidak ada yang berubah. Namun, jika masalah Anda disebabkan oleh perangkat lunak tidak jelas yang menjadi tidak responsif, hal ini mungkin dapat membantu, meskipun mungkin tidak memperbaiki masalah secara permanen. Hal yang sama mungkin terjadi minggu depan.

Kedua, jika reboot gagal maka Anda akan benar-benar stuck. Jika server dimatikan tetapi gagal untuk memulai kembali, maka seseorang mungkin harus pergi dan menekan tombol daya pada mesin fisik. Seseorang itu adalah karyawan perusahaan hosting Anda, dan mereka mungkin juga menikmati sarapannya, di kantor yang nyaman dan nyaman di suatu tempat. Mereka mungkin meninggalkan jaketnya di rumah. Mereka mungkin tidak ingin

memasuki bunker ber-AC tempat semua server disimpan. Anda akan sangat bergantung pada waktu respons mereka. Mengingat semua risikonya, perintahnya adalah:

```
$ sudo /sbin/reboot
Broadcast message from admin@thisserver.com (/dev/pts/1) at
13:21 ...
The system is going down for reboot now.
```

Perintah reboot menyebabkan server dimatikan dan kemudian restart. Ini mungkin memakan waktu beberapa menit. Segera setelah mengeluarkan perintah di atas, sesi SSH Anda akan tiba-tiba berakhir. Anda kemudian akan dibiarkan selama beberapa menit dengan gugup bertanya-tanya apakah itu akan muncul kembali. Gunakan alat yang Anda siapkan di atas.

Sementara Anda menunggu, Anda dapat mengeluarkan ping untuk melihat apakah dan kapan server Anda kembali. Di Windows gunakan ping -t untuk ping tidak terbatas:

```
$ ping www.smashingmagazine.com
PING www.smashingmagazine.com (80.72.139.101) 56(84) bytes of data.
Request timeout for icmp_seq 0
Request timeout for icmp_seq 0
Request timeout for icmp_seq 0...
64 bytes from www.smashingmagazine.com (80.72.139.101): icmp_seq=1
ttl=52
time=39.4 ms
64 bytes from www.smashingmagazine.com (80.72.139.101): icmp_seq=1
ttl=52
time=32.4 ms...
```

Anda bisa bernapas lega ketika ping akhirnya merespons. Tunggu beberapa menit lagi dan Anda akan dapat menggunakan SSH lagi lalu coba lihat Widget 3000 di browser Web Anda.

Kesimpulan

Ini merupakan perjalanan epik, dari akhir dunia hingga satu karakter yang salah tempat dalam sebuah file. Semoga dapat membantu Anda melewati beberapa menit awal kepanikan ketika Anda bangun di suatu pagi dan halaman produk indah yang Anda buat tadi malam hilang.

Beberapa alasan dan solusi di atas sangat jarang terjadi. Penyebab yang paling mungkin hanyalah kerusakan kecil pada kotak broadband Anda. Kehabisan ruang disk atau diretas adalah satu-satunya hal lain yang mungkin terjadi di tengah malam ketika tidak ada orang lain yang bekerja di situs web. Namun libatkan pengembang lain, administrator server, dan klien yang antusias — dan segalanya mungkin terjadi.

BAB 9
LANGKAH SELANJUTNYA UNTUK TIPOGRAFI WEB

Ini klise, tapi tetap saja itu benar bagi banyak desainer Web, ini adalah saat yang penuh kegembiraan dan penghargaan pribadi, baik melalui perintisan metode dan teknik baru, penggunaan teknologi. Untuk mengotomatiskan proses yang sebelumnya manual (jika memungkinkan), atau menetapkan aturan dan standar baru. Dalam beberapa hal, ini adalah petualangan menuju hal yang tidak diketahui. Dan tahukah kamu? Kita bahkan belum sampai setengah jalan! Mampu merancang dan mengoptimalkan setiap sudut dan perspektif yang dikenal saat ini hampir mustahil. Banyak sekali penemuan-penemuan baru mengenai diri kita baik secara individu maupun kolektif dalam bidang biologi, psikologi, sosiologi dan evolusi, sehingga kalimat "Banyak sekali hal yang harus dilakukan" mempunyai arti yang benar-benar baru. Tidak ada proyek desain Web yang selesai dan selalu ada ruang untuk perbaikan, baik itu memperbaiki masalah kegunaan kecil, penyesuaian konversi, atau optimalisasi kinerja.

Tipografi tidak terkecuali. Memang benar, kami mewarisi banyak hal dari bidang saudara kami, desain grafis, namun ada juga banyak sekali pilihan yang belum mungkin dilakukan hingga saat ini. Maafkan antusiasme saya, tetapi bagi saya baik sebagai pengguna Web maupun produser Web saat-saat ini cukup menyenangkan!

Berikut ini ikhtisar singkat tentang apa yang akan kami bahas. Pertama, gambaran besarnya, hal-hal yang saya harap saya ketahui lebih awal. Tidak banyak contoh kode di sini:

- Model konteks yang tidak terlalu menakutkan,
- Daftar aktor yang berpartisipasi dalam pembuatan dan penerbitan konten.

Kemudian detail praktisnya, dengan banyak contoh kode dan tip:

- Persiapan, tipografi dan desain tipografi secara umum
- Organisasi dan kinerja
- Penyusunan huruf
- Teknik tingkat lanjut

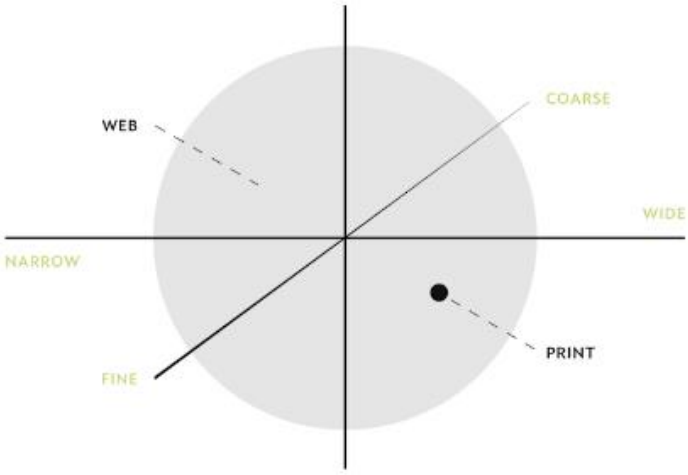
Sebelum kita memulai, satu hal yang perlu diingat adalah bahwa mempraktikkan tipografi sebagai suatu disiplin ilmu yang terisolasi bisa sangat membingungkan, terutama ketika menyelami langsung kekhususan tipografi terkecil dan anekdot sejarah. Karena Web dalam konteksnya yang paling murni mempunyai banyak segi, mari kita mulai dengan mengamati tipografi sebagai bagian integral dari gambaran yang lebih besar.

9.1 GAMBARAN BESAR: TIPOGRAFI UNIVERSAL

Pada tahun 2012, Tim Brown, manajer tipe di Typekit mempresentasikan ceramahnya "Tipografi Universal" di mana dia menjelaskan sifat dan tantangan proses desain Web.

Dia berpendapat bahwa dalam desain cetak, di mana variabel media dan fisik pada dasarnya tetap, dengan menyelidiki opsi yang terbuka bagi Anda, Anda dapat menentukan pengaturan tipografi terbaik; yaitu jenis huruf, tingkatan font, berat font, lebar font, ukuran

huruf, panjang garis, spasi baris, hierarki, dan tata letak. Dan begitu Anda telah memutuskan, Anda kurang lebih sudah selesai. Pada akhirnya, ada satu desain yang tidak bisa diubah. Itu permanen. Karya tersebut kemudian dikirim ke percetakan dan hasil akhirnya adalah objek fisik yang — secara absolut — terlihat konsisten bagi semua orang.



Sebuah buku, majalah, atau selebaran memiliki dimensi yang konsisten. Lebar, tinggi, tebal dan berat sama di seluruh dunia. Yang lebih penting lagi, kita tahu cara menggunakan benda-benda cetakan, meskipun jika dibandingkan dengan lingkungan digital, pilihan kita tetap atau lebih terbatas. Misalnya, kita tidak dapat membaca teks di atas kertas dalam kondisi pencahayaan yang kurang optimal, dan secara realistis kita hanya dapat membawa konten cetakan dalam jumlah terbatas pada satu waktu.

Dengan tipografi Web saat ini, ada banyak sekali kombinasi. Itu sebabnya Brown mengusulkan agar kita menyelidiki pilihan-pilihan kita dengan memikirkan rentang yang dapat diterima, bukan titik tetap.

Namun, hal ini lebih mudah diucapkan daripada dilakukan. Juru ketik Web perlu memahami semua aspek konteks untuk melakukan penelitian ekstensif dan menemukan solusi yang tepat. Sistem yang berlaku di wilayah jelajah kita bisa sangat menakutkan dan terus berkembang, terutama seiring dengan kemajuan teknologi dan ditemukannya pola perilaku baru. Tapi jangan khawatir. Sebagian besar variabel dapat diatasi dan diatasi dengan sedikit kesabaran, sesekali melihat kembali sejarah, dan sedikit keberanian untuk mempertanyakan status quo.

Pikirkan tentang sifat responsif dari pengalaman web tertentu sebagai sebuah kontinum keberadaan. Sepanjang kontinum ini, pada satu sumbu, pengalaman bisa meluas atau menyempit. Sepanjang sumbu yang berbeda, bisa dekat atau jauh. Sepanjang sumbu yang masih berbeda, bisa kasar atau halus. Ada banyak sumbu.

— Tim Brown,
“Breakpoint dan aturan jangkauan”

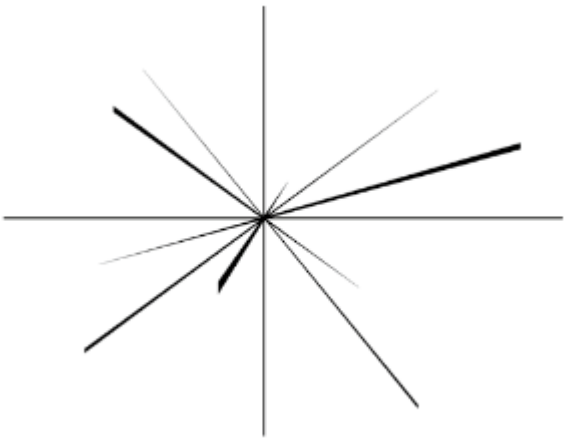
Model Beraneka Ragam Sederhana

Jika kita hanya mementingkan media dan konten, maka kita dapat mengeset sepanjang tujuh sumbu dasar yang menggambarkan perangkat itu sendiri dan hubungan dasar antara perangkat dan pembaca. Sumbu-sumbu tersebut adalah:

- jarak membaca (jarak pandang)
- dimensi layar (lebar dan tinggi area pandang)
- hierarki konten

- kepadatan informasi
- kepadatan piksel (resolusi)
- orientasi perangkat
- rasio aspek layar

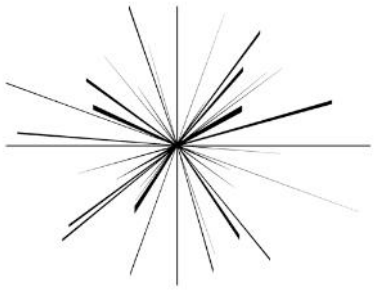
Jika menurut Anda ini sudah keterlaluan, jangan khawatir itu adalah reaksi normal dan Anda adalah orang normal. Namun, jangan biarkan hal ini membuat Anda putus asa, dan anggap itu sebagai tanda evolusi profesional. Kita seharusnya senang karena ada mainan baru di kotak pasir kita setiap pagi! Dimungkinkan untuk mengeset untuk Web dalam model ini. Itu bisa dilakukan. Kami dapat memperhitungkan kondisi ekstrem dan memberikan solusi yang tepat untuk menciptakan hasil yang memuaskan. Untuk tujuan perjalanan ini, kita dapat memanggil serangkaian metode ini:



Gambar 9.2 Tipografi Web Responsif.

Perkecil Sedikit Lagi Melanjutkan tema pembuatan daftar, mari kita perkecil lebih jauh lagi. Pada bulan Februari 2013, Cennydd Bowles menulis artikel “Mendesain dengan Konteks” tentang berbagai konteks.

- ☞ Perangkat
- ☞ Lingkungan
- ☞ Waktu
- ☞ Aktivitas
- ☞ Individu
- ☞ Lokasi
- ☞ Sosial



Kenyamanan dapat diperoleh dari kombinasi kedua model ini meskipun terdapat perpaduan yang eksplosif. Apa lagi yang harus diucapkan selain “Selamat atas pekerjaan baru Anda!” Rasa ini sangat mudah diingat berkat singkatannya yang praktis: DETAIL. Mereka tidak mempengaruhi penyusunan huruf secara langsung, namun menjadi bahan pemikiran ketika kita membuat keputusan pada tingkat yang lebih strategis. Secara umum, metodologi UX dapat membantu kita memahami tujuan bisnis, kebutuhan audiens target, dan menemukan titik temu di mana universalitas bertemu orisinalitas.

ISI

Untuk memulai, mari gunakan pemetaan perjalanan (pelanggan), alat dari bidang UX untuk mengilustrasikan proses penerbitan konten dari perspektif desain. Dari konsepsinya di benak penulis hingga pemahaman dan pemahaman penuh di pembaca, konten sedang dalam sebuah perjalanan. Itu dimanipulasi, diubah, ditingkatkan, dan ditata untuk mengirim pesan dengan cara yang paling sukses. Hal ini dilakukan oleh sederet aktor yang mengambil bagian dalam proses dan mempengaruhi pengalaman membaca:

- ※ pengarang
- ※ editor
- ※ Desainer web
- ※ tipe desainer
- ※ Hosting font web
- ※ Mesin rendering OS
- ※ browser (atau pembaca eBook)
- ※ pembaca

Menyadari banyaknya bagian dan pihak yang terlibat saja sudah merupakan sebuah pencapaian tersendiri. Semakin cepat kita mulai mempertimbangkan semua faktor ini, semakin cepat kita dapat melepaskan ilusi bahwa kita dapat menciptakan desain yang universal. Tipografi web dan desain Web secara umum adalah tentang menciptakan kondisi terbaik bagi sebagian besar pengguna, selama kami dapat mempertahankan pengalaman yang memadai untuk kasus-kasus edge.

Hasil yang kurang sempurna dapat diterima oleh pengguna di kedua ujung kurva lonceng, selama informasi dapat diakses dan tipografi dapat dimakan. Pada bagian berikut, kita akan menyelidiki bagaimana tipografi dapat ditingkatkan pada setiap tahap dalam perjalanan konten.

Pertama, Susun Kontennya

Setelah melihat keseluruhan gambarannya, dengan kesadaran bahwa tipografi saja tidak dapat memperbaiki pola-pola rusak yang terlewatkan pada awal proses desain, maka akan lebih mudah untuk menyiapkan proyek dengan dasar yang kuat. Fondasi dari setiap proyek desain Web adalah HTML dan saya mungkin akan mengutip ulang banyak orang pintar di luar sana ketika saya mengatakan bahwa HTML tanpa CSS adalah breakpoint pertama. Dokumen HTML tanpa gaya yang terstruktur dengan benar dapat diakses secara default. Dengan menggunakan markup semantik yang tepat dan dengan membangun hubungan antara berbagai bagian konten dengan elemen seperti header, footer, section, article dan side, kita dapat menciptakan struktur dan makna. Bahkan dengan menerapkan semua markup tersebut, dokumen masih dapat dirender dan dapat dengan mudah beradaptasi dengan wadahnya, jendela browser.

Kita semua tahu cara menandai dokumen HTML dasar. Namun, masih banyak konten di Web yang tidak terstruktur dengan baik. Misalnya unsur singkatan, ruang tak putus, ruang tipis, ruang rambut, semuanya masih kurang dimanfaatkan. Spasi yang tepat sebelum dan sesudah tanda baca pada inisial, inisialisme, dan pemotongan, bahkan rasio atau ekspresi

tanggal dan waktu, dapat sangat meningkatkan tekstur, memberikan jumlah jeda yang tepat untuk pembacaan tanpa gangguan. E.R.Burroughs, 18.04.2013, mis. dan 2:3, serta E.R. Burroughs, 18.04.2013, e. G. dan 2 : 3 sama-sama salah. Sebaliknya, kita harus menggunakan ruang tipis dan ruang rambut antar karakter. Misalnya:

```
E. R. Burroughs
24. 2. 2013.
24. 1. 2013.
e. g.
2 : 3
```

E.R Borroughs	E.R Borroughs	E.R Borroughs
24.2.2013	24.2.2013	24.2.2013
24.1.2013	24.1.2013	24.1.2013
e.g	e.g	e.g
2:3	2:3	2:3

Beberapa contoh ruang yang diatur dengan benar.

Situasi dengan tanda kutip (“…” dari pada “…” dan '…' dari pada '…') dan jeda baris (tanda hubung) sekarang sedikit lebih baik — setidaknya dalam publikasi desain Web — namun kita jarang melihat rentang yang diterapkan dengan tepat, misalnya Zagreb–Split atau 09.00–17.00 (keduanya diberi spasi dengan tanda hubung), di situs web mainstream.

non-breaking space	** **	** **
narrow non-breaking		
en-quad		
em-quad		
em-quad	⁦	
em-quad		
three-per-em space		
four-per-em space		
six-per-em space		
figure space		
punctuation space		
thin space		
hair space		
medium mathematical		

Berikan kepada pembuat konten (termasuk milik Anda) alat untuk membantu menyempurnakan teks mereka dengan mendorong penggunaan gaya penulisan yang tepat.

Ada beberapa situs web referensi untuk penulis yang memberikan titik awal yang baik bagi kita semua, misalnya The Chicago Manual of Style Online5, The OXford Guide to Style, atau prinsip konten Layanan Digital Pemerintah Inggris yang lebih spesifik. Dan setiap juru ketik Web harus menulis esai atau artikel pada suatu saat dalam karier mereka setidaknya untuk mengubah perspektif dan mempelajari cara kerja konsepsi konten.

Saya juga mendorong Anda untuk menulis. Anda mungkin bukan seorang Tolstoy, namun pemikiran bersama dapat menginspirasi teman-teman yang belum Anda temui.

— Jeffrey Zeldman,
Di Luar Layar #3, 2012.

Bekerja dengan Editor dan Direktur Seni

Dalam beberapa tahun terakhir kita telah menyaksikan peningkatan penggunaan arahan seni Web. Tentu saja, ini pertama kali dimulai di blog pribadi para desainer Web, tetapi sekarang banyak majalah online memiliki seseorang yang memeriksa ulang bagaimana konten terlihat dan apakah masih ada anak yatim atau janda yang tersisa, apakah mereka berperan sebagai pengarah seni, pengoreksian atau apa pun. Anda akan berpikir, “Mengapa saya harus peduli? Itu adalah Web!” Saya tahu, saya tahu kita harus menerima kelancaran Web. Namun memperbaiki kata janda di akhir paragraf mudah dilakukan baik secara manual atau dengan preprocessor8, dengan memasukkan tanda   atau   di antara dua kata terakhir, jadi tidak ada alasan untuk tidak melakukannya. Dalam lingkungan editorial, pemimpin redaksi dan direktur seni bekerja sama untuk menyusun ulang, memecah, dan merombak konten agar mudah dibaca. Proses yang sama dapat diterapkan pada setiap situs web yang keluar. Bagaimanapun, pesan perlu diedit baik isi maupun bentuknya agar sampai ke penerima.

Bekerja lebih erat dengan pembuat konten dan pengembang CMS untuk memastikan markup teks yang tepat diterapkan, dan memberikan pedoman untuk menetapkan ritme dan keseimbangan, terutama jika terdapat jenis konten yang berbeda seperti teks, foto, video, grafik, atau tabel campuran dan cocok bersama. Kita dapat melangkah lebih jauh dan merancang templat antipeluru untuk markup apa pun yang dihasilkan oleh CMS; misalnya, membatasi gambar mengambang, mengabaikan gaya font dan poin khusus. Membantu mengembangkan CMS yang mendukung objek konten, selain mendukung skema metadata yang luas, juga mendukung pengurutan dan hierarki konten. Jika hal ini tidak memungkinkan, paling tidak, perluas objek konten asli dengan sub-objek khusus, sebuah fitur yang kini tersedia selama beberapa tahun di beberapa CMS yang tangguh, misalnya di eZ Publish.

Minta pengembang Anda membangun jaring pengaman dengan formulir pengeditan kaku yang membatasi eksplorasi desain dan menjaga semantik di seluruh CMS. Konsultasikan dengan editor dan buat sistem placeholder untuk perlengkapan artikel, seperti pendahuluan, fakta penting (misalnya nomor paling penting), kutipan tarik atau catatan samping, dan perhitungkan kombinasi yang berbeda. Ini mungkin tampak berlebihan, namun jika kebersihan konten dan konsistensi gaya itu penting, inilah cara yang harus dilakukan. Jika Radio Publik Nasional mengembangkan API untuk kontennya, kita semua juga bisa.

(Responsif) Juru Tulis Web

Seperti yang pernah ditulis oleh Andrew Clarke, “Desain Web Responsif adalah desain web yang dilakukan dengan benar.” Demikian pula, tipografi Web responsif adalah tipografi Web yang dilakukan dengan benar. Kami menggunakan kata sifat responsif untuk sementara, sama seperti kami menggunakan frasa tata letak berbasis CSS sampai semua orang dan anjing tetangganya menyadari bahwa tabel tidak bagus untuk menata halaman. Bisa dikatakan, saat ini semua orang menyebut tata letak sebagai tata letak.

Peran utama tipografi adalah menyediakan antarmuka untuk pesan yang ingin disampaikan kepada penerima. Cara Anda menyampaikannya sering kali lebih penting daripada informasi itu sendiri. Namun, tipografi Web bukan hanya tentang menelusuri katalog font dan mencium spesimen tipe yang baru dipanggang (jangan bilang Anda tidak pernah melakukan itu!?). Tipografi web terutama berkaitan dengan membuat informasi dapat diakses, dibaca, dan dibaca. Itulah mengapa memahami tipografi mikro sangatlah penting. Jika konten tidak dapat dibaca dengan nyaman, gaya tidak menjadi masalah.

Ada kalanya kita bisa menempatkan gaya di atas substansi. Jika satu-satunya tujuan konten adalah untuk mencapai daya tarik jangka pendek dan ajakan bertindak satu kali, maka gaya dapat menjadi hal pertama yang harus kita pertimbangkan, namun lebih sering daripada tidak dan terutama di Web, konten harus dapat diakses dan dibaca tidak hanya sekarang, tapi juga di masa depan. Setidaknya, itulah idenya.

Sebagai pencari jodoh profesional antara konten dan pembaca, kita harus memahami apa yang harus dicakup dalam merancang pengalaman membaca. Saat kita mengetahui konten, konteks, dan pengguna, pertimbangkan semua variabel dan pahami batasan, batasan, dan cita rasa dari semua bahan dalam sebuah proyek, karena pada akhirnya pilihan kita diinformasikan sebagai hasil dari mencakup semua dasar dan dasar ini. menghindari jebakan. Dengan kata lain, pengaturan huruf adalah praktik yang sangat rasional.

Erik Spiekermann suka mengatakan bahwa Anda hanya memerlukan warna dan jenis huruf dan Anda memiliki merek. Sekali kita menghilangkan semua hiasan dan dekorasi, tipenya tetap ada, sehingga membentuk esensi komunikasi visual. Hal ini jauh lebih jelas saat ini karena kami merancang untuk layar seluler yang sangat sempit dan lingkungan yang adaptif. Tidak ada ruang untuk perawatan visual dan tambahan yang intensif, karena tidak mungkin membawa semua beban itu ke media yang berbeda.

Selain itu, kita perlu mempelajari kembali cara mendesain Web dengan lebih sedikit perangkat visual dan lebih sedikit bahan bangunan. Setelah kita menghapus semua garis, latar belakang, tekstur dan tata letak, yang tersisa hanyalah konten. Itu sebabnya tipografi sangat penting. Karena tipografi adalah wajah dari konten, segala hal lainnya dapat dihilangkan, namun tipografi tetap ada.

Untungnya, tipografi adalah disiplin ilmu yang sudah lama dan mapan serta ada banyak sumber daya untuk dipelajari. Di bawah ini adalah pilihan buku yang direkomendasikan, diurutkan kira-kira agar mudah dibaca.

- Paragraf Dalam oleh Cyrus Highsmith
- Berpikir dengan Type oleh Ellen Lupton

- Berhenti Mencuri Domba dan Pelajari Cara Kerja Type
- oleh Erik Spiekermann dan E.M. Ginger
- Detail dalam Tipografi oleh Jost Hochuli
- Elemen Gaya Tipografi oleh Robert Bringhurst
- Stroke oleh Gerrit Noordzij
- U&lc: Mempengaruhi Desain dan Tipografi

Temui Tipe Desainer

Pada bulan Mei 2013 Jessica Hische menulis esai yang sangat mudah diikuti "Upping Your Type Game" tentang pemilihan tipografi, menekankan pentingnya memilih beberapa desainer tipe dan tetap berpegang pada tipografi mereka. Saya sangat setuju. Tidak hanya tipografi yang berbeda dari desainer yang sama biasanya bekerja sama dengan baik, namun berbicara dengan desainer tipe favorit Anda dapat membawa perspektif lain ke dalam proses Anda sendiri. Desainer tipe memiliki banyak pengujian dan umpan balik dan kami berdua sama-sama mengalami kesulitan yang sama dalam hal publikasi sekali, bekerja di mana saja. Misalnya, kami menggunakan kueri @media untuk menyesuaikan desain kami ke layar yang berbeda; mereka menggunakan petunjuk untuk menerjemahkan kurva bezier ke dalam kotak piksel. Kami frustrasi dengan fragmentasi dan semakin banyaknya titik henti sementara (breakpoint) dan titik penyesuaian (tweakpoint) yang sulit dipertahankan; mereka frustrasi dengan petunjuk manual pada setiap ukuran huruf. Kami memiliki browser yang berbeda; mereka memiliki rasterizer yang berbeda. Singkatnya, kami memiliki banyak kesamaan dan mereka bisa menjadi teman yang sempurna jika Anda membutuhkan bahu untuk menangis.

Desainer tipe dapat memberi saran kepada Anda mengenai ukuran huruf terbaik untuk desain Anda, bagaimana mengkompensasi kurangnya kontrol spasi halus di CSS, jenis huruf apa yang paling cocok dalam situasi tertentu. Mereka akan memiliki pandangan yang berbeda mengenai berbagai hal, karena mereka telah melihat tipografi mereka digunakan dalam lebih banyak skenario daripada yang kita miliki. Mereka tahu siapa pengaruhnya sebelum mulai membuat jenis huruf. Dan pada akhirnya, mereka mengetahui sejarahnya, karena mereka pernah menjadi bagian di dalamnya.

Pabrik pengecoran logam kecil sangat mudah dijangkau, begitu pula dengan pabrik pengecoran logam besar yang perwakilannya dapat dihubungi melalui jejaring sosial dan pada konferensi di sebelahnya. Jadikan misi Anda untuk berbicara dengan desainer tipe atau manajer tipe saat Anda menghadiri konferensi lagi. Jika Anda tidak memiliki kesempatan untuk bertemu langsung dengan seorang desainer tipe, forum Typophile18 adalah tempat yang bagus untuk mengajukan pertanyaan dan mendapatkan banyak jawaban. Majalah 8 Faces dari Elliot Jay Stocks19 menampilkan wawancara dengan desainer tipe yang selain memberikan wawasan tentang cara berpikir mereka memberikan kepada pembaca pilihan font favorit mereka yang terperinci.

Masih belum yakin? Berikut kutipan refleksi diri yang menghibur dari Chris Schwartz dari Commercial Type20:

Kembali ke pertanyaan saya sebelumnya: "Menurut saya, bagaimana karya saya akan dipengaruhi oleh media 'baru' Web?"

— Chris Schwartz,
konferensi Ampersand, 2013.

Pengiriman dan Distribusi Font

Kelompok lain yang harus diajak bicara oleh desainer Web adalah distributor dan layanan hosting font Web. Dengan melakukan hal ini, desainer Web mempunyai kesempatan untuk mempengaruhi kelompok-kelompok ini untuk menyesuaikan layanan mereka guna memenuhi kebutuhan pasar, sehingga sebagai imbalannya mereka dapat memberikan hasil yang lebih baik kepada klien mereka melalui penggunaan tipe. Kami sudah tidak perlu lagi menjelaskan pentingnya pengalaman merek lintas saluran yang konsisten kepada klien kami.

Merek terkenal di seluruh dunia, seperti Red Bull atau The Guardian adalah contoh yang baik untuk ditampilkan jika Anda memerlukan referensi untuk memperkuat pesan Anda. Layanan hosting font web juga memainkan peran penting dalam pengiriman font ke seluruh perangkat yang mendukung Internet. Mereka berada di ujung bisnis kemajuan teknologi terkini, karena mereka bekerja sama dengan pabrik font untuk memberikan hasil terbaik.

Mereka akan sering mendengarkan jika Anda memiliki permintaan khusus. Misalnya, sub-pengaturan (menghapus semua mesin terbang yang tidak diperlukan) adalah salah satu metode paling mudah untuk mengurangi ukuran file font dan mempercepat kinerja jenis Web. Jika Anda memerlukan subset tertentu untuk sebuah proyek (misalnya, subset huruf kecil atau gambar lapisan), perusahaan hosting font Web mungkin dapat membuat dan mengirimkan file font terpisah dan membuat situs web Anda terasa secepat kilat. Meskipun layanan tersebut mungkin tidak tersedia secara gratis, biayanya mungkin juga tidak terlalu mahal terutama jika Anda dapat meningkatkan kinerja situs web secara signifikan.

Dengan adopsi font Web menjadi lebih umum dan tersebar luas, banyak pabrik font mulai menghargai dan memahami pentingnya kinerja, sehingga banyak pabrik yang menawarkan layanan font Web melalui CDN mereka yang dapat diandalkan. Pabrik pengecoran seperti Hoefler & Frere-Jones, Just Another Foundry, Monotype Imaging, Typonine dan Typotheque semuanya baru-baru ini memasuki persaingan dan bergabung dengan tersangka biasa seperti Adobe Edge Web Fonts, Fontdeck, Google Fonts, Typekit, Webink, dan Webtype yang semuanya terutama mempromosikan font Web pengecoran lain.

9.2 MESIN RENDERING OS

Tidak banyak yang dapat Anda lakukan untuk memengaruhi produsen OS, namun ada gunanya mengetahui bagaimana berbagai mesin rendering memengaruhi tipografi dan apa yang dapat kami lakukan untuk meningkatkan keterbacaan. Saat ini ada tiga OS utama yang tipe rendernya berbeda: Windows, Mac OS X, dan iOS. Pada OS Apple baik desainer tipe maupun web mempunyai sedikit kendali, karena mesin rendering sama sekali mengabaikan petunjuk. Hal ini menghasilkan bentuk huruf yang agak lebih tebal, namun hal ini dapat diterima karena renderingnya dapat diandalkan dan konsisten.

Di Windows, kita dapat meningkatkan tampilan jenis huruf di layar dengan memilih format font yang sesuai. Untuk IE6–8 yang menggunakan rendering subpiksel ClearType GDI (Windows Graphic Device Interface) satu-satunya pilihan yang layak pada ukuran yang lebih kecil adalah font TrueType Web yang telah terbukti dengan baik. Namun, ada satu masalah. GDI ClearType meningkatkan rendering hanya sepanjang sumbu horizontal, karena subpiksel merah, hijau, dan biru hanya tersedia pada arah tersebut. Untuk lebih memahami efeknya, perbesar cuplikan layar teks dan amati lingkaran cahaya oranye dan biru di sekitar huruf melengkung, seperti O. Lingkaran cahaya hanya muncul dalam arah horizontal dan merupakan hasil rendering subpiksel.

Pada sumbu vertikal ClearType menggunakan rendering hitam-putih paling primitif (bukan rendering skala abu-abu, yang merupakan solusi anti-aliasing perantara). Efeknya mudah terlihat di bagian atas dan bawah kurva pada beberapa huruf, seperti o, p, dan b, dengan undakan dan jag yang merusak bentuknya, terutama saat jenis tampilan diatur pada ukuran besar. Karena GDI ClearType bergantung pada petunjuk yang baik, dan font yang diberi petunjuk dengan baik sulit ditemukan, menggunakan font yang telah dicoba dan diuji seperti Georgia atau Verdana untuk body copy dikombinasikan dengan font Web untuk judul dan subjudul dengan ukuran lebih besar adalah hal yang masuk akal.

Font PostScript yang menggunakan tipe petunjuk berbeda lebih baik untuk ukuran besar dan bentuk oval di IE6–8. Berbeda dengan font TrueType, font Post-Script tidak dirender dengan GDI ClearType, namun dikembalikan ke rendering skala abu-abu, yang menghasilkan hasil lebih baik dalam arah vertikal, sehingga meningkatkan tampilan bentuk secara keseluruhan. Beberapa pengecoran dan layanan, seperti JAF dan Typekit, menyediakan format berbeda tergantung pada penggunaan jenis huruf.

Font Web yang dihosting sendiri dapat disematkan dengan dua format berbeda untuk IE6–8, menggunakan file font TrueType untuk elemen konten yang disetel pada ukuran lebih kecil (sekitar 14pX ke bawah) dan file font PostScript untuk elemen yang disetel pada ukuran lebih besar (sekitar 18pX ke atas) . Prosedur lengkap untuk mengkonversi file untuk kegunaan yang berbeda dijelaskan dengan baik dalam artikel blog wax-o “Masalah rendering font-face di chrome dan firefoX: gunakan garis besar postscript”. Peringatan: periksa kembali EULA font untuk memastikan Anda diizinkan melakukan perubahan tersebut.

Untuk anti-aliasing subpiksel yang tepat, mesin rendering mempertimbangkan informasi tentang latar belakang, sehingga teks yang berada pada lapisan transparan atau semi-transparan perlu dibuat anti-alias terhadap warna yang terlihat dari lapisan di latar belakang. Apple bahkan menonaktifkannya di bilah menu transparan Mac OS X. Jadi, jika rendering tampak sedikit salah, opacity yang kurang dari 100% dikombinasikan dengan latar belakang yang sulit mungkin menjadi alasannya.

Terakhir, dalam penelitian saya, saya menemukan beberapa sumber tidak resmi yang mengklaim iOS membuang rendering subpiksel demi anti-aliasing standar, karena rendering subpiksel hanya dapat dilakukan dalam arah asli. Mendukung anti-aliasing subpiksel hanya dalam satu arah jelas akan merusak konsistensi dalam orientasi perangkat yang berbeda. Memang benar, jika Anda mengambil tangkapan layar dari dalam perangkat iOS dan

memperbesarnya, Anda akan melihat piksel abu-abu muda menggantikan piksel oranye dan biru muda.

Namun, jika Anda memperbesar foto yang diambil dengan kamera ponsel pintar sederhana, pinggiran berwarna oranye dan biru akan terlihat jelas. Misalnya, pada iPhone Retina, urutan subpiksel horizontal BGR (biru, hijau, merah) terlihat jelas bila dilihat dalam potret, sedangkan pada iPad urutan yang sama muncul bila dilihat dalam lanskap (dengan tombol kontrol volume menghadap ke atas). Saya mungkin harus berhenti di titik ini, karena saya hanya bisa berspekulasi bahwa aplikasi tangkapan layar bawaan meratakan piksel tepi ke warna abu-abu yang sesuai. Namun apa pun masalahnya, ketika menyangkut tipografi, abaikan spesifikasi perangkat dan percayalah pada penilaian baik Anda.

Peramban

Browser yang berbeda menggunakan default yang berbeda untuk menangani tipografi. Aku tahu, beritahu aku sesuatu yang baru. Namun, situasinya tidak seburuk dulu di awal tahun 2000an, karena vendor browser kini mendengarkan dan berkolaborasi dengan desainer dan pengembang. Bahkan tim pengembangan IE yang pernah ditutup merilis beberapa demo tipografi bagus yang dapat ditemukan di Use The Whole Font.

Sama pentingnya dengan tetap mengikuti pembaruan terkini, penting juga untuk berpartisipasi dengan permintaan fitur, laporan bug, studi kasus, dan uji kompatibilitas. Banyak konferensi dan pertemuan untuk pengembang Web disponsori oleh vendor browser dan mereka biasanya mengirimkan penginjil teknis untuk menyajikan perkembangan terkini. Bicaralah dengan mereka saat istirahat, melalui email atau jejaring sosial. Bahkan jika mereka mungkin tidak memiliki jawaban atas pertanyaan khusus Anda, mereka pasti akan meneruskan Anda ke seseorang dalam organisasi mereka yang memiliki jawaban tersebut. Pembuka percakapan terbaik dengan penginjil teknis adalah memiliki halaman pengujian siap pakai dengan contoh langsung dan tangkapan layar. Mudah!

Perbedaan utama dalam default tipografi di seluruh browser terletak pada format font yang didukung, perilaku pemuatan font, dan opsi keterbacaan. Pembahasan tentang format font akan lebih baik daripada bab yang sudah intensif (saya sudah memperingatkan Anda bahwa saya sangat antusias dengan topik ini), namun saya akan menjelaskan cara mengelola perbedaan dengan perilaku pemuatan nanti. Mari kita fokus sejenak pada keterbacaan dan rendering.

Properti rendering teks

Jika Anda belum pernah tidur di bawah batu selama beberapa tahun terakhir, Anda mungkin akrab dengan properti CSS rendering teks yang dapat diatur untuk mengaktifkan kerning dan ligatur. Meskipun Internet EXplorer dan Opera tidak mendukung rendering teks pada saat penulisan, ini diterapkan secara default di FirefoX. Nilai otomatis di FirefoX diperlakukan sebagai optimizeLegibility (yang mengaktifkan kerning dan ligatur umum, namun memperlambat pemuatan dan pengguliran halaman), sedangkan di Chrome dan Safari, nilai otomatis diperlakukan sebagai optimizeSpeed (yang menonaktifkan kerning dan ligatur umum demi yang lebih baik pertunjukan).

Namun, properti ini dilaporkan menyebabkan sejumlah masalah. rendering teks dapat memperlambat pemuatan halaman secara drastis jika diterapkan pada teks yang lebih panjang. Masukkan varian font: huruf kecil; dan beberapa spasi huruf khusus dan terkadang menghasilkan hasil yang tidak terduga. Aturan praktis saya adalah menerapkan rendering teks hanya pada judul dan subjudul, atau sekadar memfilter perangkat yang kurang mampu melalui kueri media. Seorang standardista ortodoX dalam diri saya mempunyai masalah kecil lainnya - properti ini sebenarnya bukan bagian dari standar CSS, melainkan standar SVG. Interrobang!

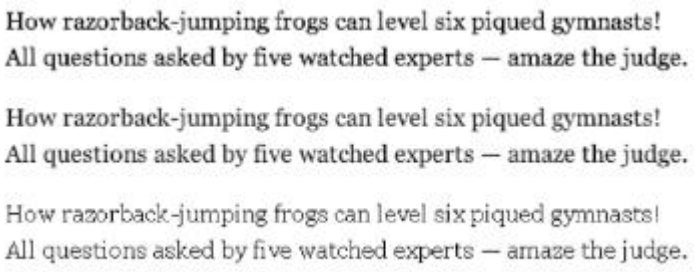
Efek yang sama dapat dicapai dengan menggunakan properti font-kerning dan font-feature-settings yang lebih cocok dan ramah masa depan:

```
body {
        -webkit-font-kerning: normal;
        -moz-font-kerning: normal;font-kerning: normal;
        -webkit-font-feature-settings: "liga";
        -moz-font-feature-settings: "liga", "kern";
        font-feature-settings: "liga", "kern";
       /* IE 10 supports the standard property name */
}
```

Peningkatan Keterbacaan khusus browser

Untuk pahlawan tanpa tanda jasa di antara kita yang merasa nyaman mengelola aturan CSS yang berbeda untuk browser yang berbeda, ada beberapa trik yang bisa kita terapkan untuk meningkatkan tampilan font.

Cara termudah untuk mengubah rendering di Safari dan Chrome di Mac OS X adalah dengan properti -webkit-font-smoothing dan nilainya none, sub-pixel-antialiased (default), dan antialiased. Nilai antialias menghasilkan mesin terbang yang lebih tipis, namun dengan rendering garis diagonal yang kurang optimal, jadi sebaiknya hindari menggunakan huruf miring:



Hasil yang berbeda berdampingan. Lihat halaman pengujian `-webkit-font-smoothing` oleh Christoph Zillgens32.

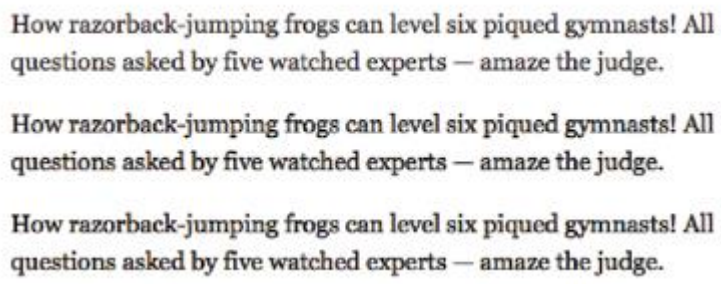
```
body { -webkit-font-smoothing: antialiased; }
```

text-shadow meningkatkan antialiasing teks di Chrome pada Windows. Bayangan tidak berpengaruh pada Mac:

```
body { text-shadow: 1px 1px #fff; }
/* Replace the color with the background color */
```

Keterbacaan di FirefoX dapat ditingkatkan dengan menambahkan kombinasi bayangan teks33:

```
body { text-shadow: 0px 0px 0px #777, 0px 0px 1px #ddd; }
```



Render berbeda dengan text-shadow di FirefoX.

Properti penyesuaian ukuran teks

Sebagian besar browser modern menerapkan algoritme pemompaan teks untuk memperbaiki pengalaman seluler untuk situs web yang menggunakan tata letak tetap (Aduh!). Kita dapat memilih keluar dari perilaku ini dengan properti text-size-adjust. Namun, jika nilainya disetel ke none, ini akan mencegah zoom in, jadi berhati-hatilah saat menggunakannya.

```
body {
     -webkit-text-size-adjust: 100%;
     -moz-text-size-adjust: 100%;
     -ms-text-size-adjust: 100%;
     text-size-adjust: 100%;
}
```

Pembaca Dunia

Saat mendesain untuk dicetak, sebesar apa pun edisinya, tetap saja terbatas. Mendesain untuk Web berarti mendesain untuk semua orang, di mana saja dan secara teori kapan saja di masa depan. Berbeda dengan edisi cetak, kami tidak dapat merencanakan dan mengontrol distribusi website yang kami bangun. Mendesain untuk audiens internasional berarti memahami adanya perbedaan yang signifikan.

Pekerjaan yang dilakukan untuk memelihara situs web multi-bahasa, seperti BBC World Service sungguh menakjubkan. Tidak hanya perbedaan arah penulisannya, tetapi juga

kekhasan budayanya, seperti larangan penggambaran makhluk hidup untuk tujuan ibadah dalam Islam. Mempertimbangkan faktor-faktor tersebut tentu saja lebih menekankan pada tipografi.

Berbeda dengan tulisan kursif yang berdasarkan abjad Latin, gaya bahasa Arab standar memiliki bentuk yang sangat berbeda tergantung apakah akan dihubungkan dengan huruf sebelumnya dan/atau berikutnya, sehingga semua huruf utama memiliki bentuk kondisional (alograf), bergantung pada baik di awal, tengah, atau akhir kata, sehingga dapat menunjukkan empat bentuk berbeda (awal, tengah, akhir, atau terisolasi).

Sumber: http://en.wikipedia.org/wiki/Arabic_alphabet

Sekalipun hanya khalayak lokal yang dijadikan sasaran dan perancang memahami bahasanya, penting bagi perancang untuk keluar dan mengamati masyarakat. Anda tidak perlu dilatih untuk melakukan wawancara pengguna yang komprehensif atau pertanyaan kontekstual. Pendekatan yang kurang formal dapat diterima sepenuhnya. Luangkan waktu saja di antara masyarakat umum dan Anda akan mendapatkan wawasan berharga tentang perilaku masyarakat. Berpura-puralah Anda adalah mata-mata. Berdirilah di dekat kios untuk melihat surat kabar apa yang dibaca dan bagaimana halaman sampul memengaruhi pilihan pembelian. Amati berapa banyak waktu yang dihabiskan orang untuk membaca buku di taman. Pelajari jenis topik apa yang dikonsumsi dan dalam konteks apa. Transportasi umum sangat ideal untuk ini, seperti biasanya mengasingkan diri sambil mendalami konten, baik itu teks, musik, atau video. Kunjungi perpustakaan atau lihat orang-orang di ruang tunggu dokter atau penata rambut. Ini semua adalah metode penelitian untuk setiap proyek desain.

Saat kita mendesain dengan tipografi, tes yang baik adalah dengan meminta seseorang membacakan teks dengan suara keras. Jika mereka kesulitan membaca teks yang mudah dibaca dan kecepatan membaca tidak merata, itu pertanda pasti ada sesuatu yang menghalangi. Solusinya mungkin sederhana seperti mengubah jarak huruf, memperpendek panjang garis, atau memperlebar alur di antara dua kolom.

Ujian hebat lainnya adalah memberikan sebuah dokumen kepada seseorang dan meminta mereka menemukan beberapa informasi yang terkubur di dalamnya secepat mungkin. Atau Anda dapat meminta mereka membaca teks dalam jangka waktu terbatas dan kemudian mengajukan serangkaian pertanyaan untuk menguji apakah mereka mampu menemukan dan memahami semua fakta penting. Meskipun tidak perlu melakukan penelitian semacam ini pada setiap proyek, mengamati cara orang membaca dan berbagi konten biasanya akan meningkatkan perspektif dan pemahaman Anda.

Jadi, bisakah kita mengukur keberhasilan suatu desain? Kevin Larson dari Microsoft dan Dr. Rosalind Picard dari MIT melakukan penelitian untuk mengeksplorasi pengaruh tipografi yang baik. Dalam studi “The Aesthetics of Reading” (PDF) mereka membagi 20 peserta menjadi dua kelompok. Setiap kelompok harus membaca sebuah dokumen, dengan satu kelompok menerima versi tata letak yang baik, dan kelompok lainnya menerima versi

tata letak yang buruk. Mereka melakukan dua penelitian untuk mengkonfirmasi hasilnya. Adakah ide tentang hasilnya?

Nah, inilah sedikit kejutan. Waktu membaca dan pemahaman sama pada kedua kelompok. Namun, kelompok yang membaca versi yang diketik dengan baik memiliki kinerja yang lebih baik dalam hal durasi relatif subjektif mereka lebih meremehkan waktu membaca sehingga lebih mampu menyelesaikan tugas-tugas kreatif, seperti tugas lilin37 dan tugas jarak jauh. mengutip tugas38 (walaupun hasil tes selanjutnya tidak dapat diandalkan secara statistik). Dalam bahasa Inggris yang sederhana, tipografi yang baik menghasilkan suasana hati yang baik.

Sekarang Kita Semua Berteman…

Anda seharusnya sudah yakin sekarang bahwa desain bukan sekadar aktivitas tunggal. Kita dapat mempelajari kebutuhan suatu proyek dengan baik jika kita keluar, bertukar ide, dan mengamati pengguna di kedua sisi perjalanan konten. Belajar melihat gambaran besarnya dan keputusan tentang teknologi, kinerja, atau pemilihan tipe akan menjadi lebih mudah dibuat. Buka mata Anda dan pola akan mulai muncul. Setelah Anda merasa nyaman dalam menemukannya, Anda akan mengetahui bahwa ada banyak detail yang dapat Anda liput untuk meningkatkan tipografi. Jadi sekarang mari kita lihat metode yang lebih praktis.

Rincian Praktis

Untuk menetapkan arah desain, pertama-tama kita perlu mengevaluasi kontennya. Selain membaca teks, yang merupakan langkah pertama yang paling jelas, kita dapat mengukur teks dengan beberapa rumus praktis. Nilai numerik tersebut dapat digunakan untuk membandingkan teks dengan bagian serupa lainnya dan memberi kita pemahaman yang lebih baik tentang teks tersebut. Statistik keterbacaan artikel adalah skrip PHP yang menghitung berbagai properti teks. Ini dapat dengan mudah tertanam dalam CMS pilihan Anda dan oleh karena itu dapat secara otomatis menghitung waktu membaca, indeks keterbacaan otomatis, dan kemudahan membaca untuk setiap artikel di situs web.

Kecepatan membaca berkisar dari di bawah 100 kata per menit hingga beberapa ribu. Rata-rata orang dewasa membaca 250 kata per menit dengan pemahaman 70%, jadi waktu membaca dihitung dengan membagi jumlah kata dalam teks dengan 250. Cukup masukkan konten proyek Anda ke instalasi lokal CMS favorit Anda dan jalankan melalui skrip untuk mengamati hasil berdasarkan tingkat rata-rata itu. Tentu saja, jika tingkat membaca untuk proyek Anda berbeda dari 250, gunakan nilai yang sesuai dengan target audiens Anda untuk menilai teks.

Selanjutnya, jika teks tersebut tampaknya tidak sesuai dengan format waktu dalam konteks yang diharapkan (misalnya, perjalanan kereta api selama 30 menit), kami memiliki beberapa opsi:

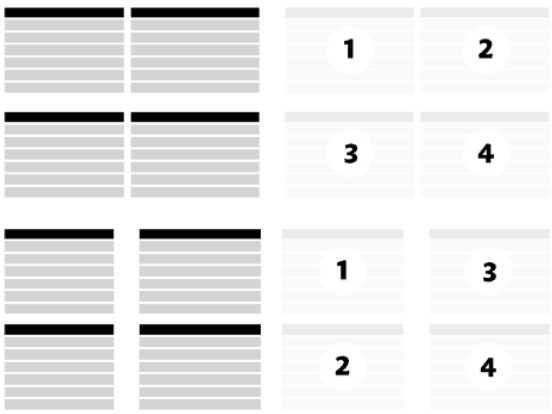
1. Bicaralah dengan penulis dan editor untuk mengatur ulang teks agar sesuai dengan format waktu yang diperlukan. Hal ini dapat dilakukan dengan memperpendek artikel agar sesuai dengan formatnya atau dengan memecah artikel menjadi sekuel atau artikel terpisah yang lebih kecil yang dapat berdiri sendiri.

2. Pisahkan teks utama dengan subjudul dan tanda kutip tarik. Hapus semua penjelasan dan contoh dari dalam teks dan letakkan di luar isi utama sebagai catatan samping. Dengan cara ini, pembaca yang familiar dengan topik tersebut dapat dengan cepat membaca artikel tersebut dan mereka yang membutuhkan penjelasan tambahan dapat menggunakan catatan samping dan kaki untuk mempelajari lebih lanjut.
3. Menyediakan alat bantu navigasi kepada pembaca seperti subjudul, metadata, dan struktur berbeda untuk elemen pelengkap (misalnya daftar, tabel data, dan grafik), sehingga mereka dapat memperoleh informasi penting dengan cepat.

Tipografi Makro

Setelah kami memiliki konten yang memenuhi syarat, bergantung pada temuannya, kami dapat mengambil pendekatan berbeda untuk menyiapkan makrotipografi. Jika teks mudah dibaca, yaitu mendapat skor tinggi pada Flesch-Kincaid Reading Ease atau rendah pada Automated Readability Index artinya teks dapat dibaca dengan cukup cepat. Dalam hal ini, ada ruang untuk variasi gaya. Anda dapat memilih jenis huruf dengan banyak karakter dan menata dokumen dengan cara yang tidak biasa untuk menambah daya tarik. Unsur-unsurnya bisa dipadupadankan agar pembaca bisa memahami ceritanya.

Salah satu contohnya adalah poster (atau situs web) untuk suatu acara. Kumpulan informasi pada poster pada umumnya sudah diketahui dan jelas. Misalnya, situs web konferensi pada umumnya menampilkan pembicara, peserta, sponsor, topik, jadwal, lokasi, dan formulir tiket. Namun, selain memberikan informasi kepada pengunjung, tujuannya adalah untuk menonjol di tengah-tengah acara serupa. Karena berfungsi sebagai alat komunikasi untuk beberapa konten lain (acara), dan diharapkan dapat diingat secara mendalam, kita dapat memilih jenis huruf yang berkarakter.



Di sisi lain, jika teks atau masalahnya rumit, tata letaknya harus baik dan stabil dan dalam hal ini setiap keanehan atau pemborosan tipografi yang tidak terduga dapat mengganggu alur pemikiran. Teks tersebut dapat dipecah menjadi beberapa komponen dengan penggunaan ilustrasi dan infografis yang ekstensif, keterangan gambar bernomor, dan kerangka dokumen yang jelas. Sebuah dokumen yang sulit dibaca atau dipahami mendapat manfaat dari struktur yang kuat.

Cara termudah untuk mengatur strukturnya adalah dengan paragraf blok teman lama kita, daftar bernomor, kisi yang jelas, dan selokan yang besar. Misalnya, petunjuk langkah demi langkah paling mudah diikuti jika setiap langkah disertai dengan foto atau ilustrasi yang jelas. Sangat mudah untuk mengacaukan segalanya di sini juga. Arah alami membaca bagi orang barat adalah horizontal, dari kiri ke kanan, jadi lebih baik merancang setiap langkah sebagai pasangan horizontal daripada menumpuk gambar dan teks di atas satu sama lain. Untuk kejelasan yang lebih baik, gambar dan penjelasan tekstual harus diletakkan

berdampingan, meskipun ini berarti menggunakan gambar yang lebih kecil agar sesuai dengan ruang horizontal yang tersedia. Ada perbedaan besar antara foto dengan keterangan dan foto sebagai alat bantu visual untuk teks. Dalam kasus pertama, foto adalah konten utama, daya tarik utama. Dalam kasus terakhir, gambar hanya bertindak sebagai pembantu, perpanjangan yang menambah kejelasan pada teks utama. Jika diperlukan, kami selalu dapat memberikan link ke versi yang lebih besar.

9.3 MENGGABUNGKAN TIPOGRAFI

Inilah bagian yang menyenangkan memilih tipografi. Seni memilih dan menggabungkan tipografi dibahas dengan baik dalam panduan saku Combining Typefaces oleh Tim Brown. Namun, seperti yang dikatakan Brown sendiri, hal itu membutuhkan latihan. Banyak latihan.

Setelah Anda dapat mengenali klasifikasi jenis huruf apa pun yang Anda lihat, akan sangat mudah untuk menggabungkan jenis huruf, karena pada saat Anda dapat melihat perbedaannya, Anda telah mempelajari banyak hal lain tentang jenis huruf dan tipografi secara umum. Maaf atas kabar buruknya, tetapi jalan pintas untuk berhasil memilih dan menggabungkan tipografi adalah 10.000 pengulangan.

Serius, seni memilih tipografi adalah seni mempersempit pilihan Anda. Kebutuhan proyek Anda adalah kriteria Anda. Semakin baik Anda mengetahui proyek tersebut, Anda akan semakin pemilih. Setelah membuang semua tipografi yang tidak memiliki kumpulan karakter lengkap, yang kehilangan beberapa gaya dan bobot, yang tidak akan terlihat bagus dalam berbagai ukuran, dan yang umumnya tidak sopan jika digabungkan dengan tipografi lain anda tinggal dengan hanya beberapa pilihan yang masuk akal dan pada akhirnya Anda hanya perlu memilih dari kelompok yang terdiri dari dua atau tiga orang.

Untuk memilih jenis huruf, kita harus mengetahui seberapa besar kolamnya. Jika kita hanya mengamati dan mengenali segelintir tipografi sepanjang waktu, kita tidak dapat menciptakan sesuatu yang benar-benar baru dan kita akan berakhir dengan desain konvensional yang sama berulang kali. Langkah pertama dalam mempelajari tipografi adalah mengenal klasifikasi tipografi dan sejarah tipografi. Sangat penting untuk mempelajari dan menyelidiki tipografi dari semua klasifikasi, bahkan yang Anda pikir tidak akan pernah Anda gunakan. Sebagai permulaan, fokuslah pada tipografi profesional yang sudah mapan. Ini populer karena suatu alasan.

Mereka bekerja di lingkungan yang menuntut dan menolak penuaan.Kesalahan umum yang dilakukan adalah memilih jenis huruf yang indah, yang terlihat menarik sehingga lebih mengutamakan bentuk daripada fungsi. Ini berarti meletakkan kereta di atas kuda. Betapapun anehnya kedengarannya, 'tampilan' jenis huruf seharusnya tidak menjadi perhatian Anda.

— Alessandro Cattaneo, Yves Peters, Jon Tan,
Buku Smashing #1

Tempat yang baik untuk memulai adalah mengunjungi berbagai daftar:

- Bagian JELAJAHI Typedia mencantumkan semua perancang dan pengecoran tipe. Mulailah dengan desainer paling populer terlebih dahulu, lalu pelajari tipografi dari masing-masing desainer: http://typedia.com/explore
- Orang-orang di Typekit membuat sejumlah daftar yang akan membantu Anda menemukan alternatif selain tipografi umum: https://typekit.com/lists
- FontBook untuk iPad adalah sumber bagus lainnya. Font dikelompokkan berdasarkan genre, yang sangat berguna jika Anda mencari jenis huruf yang memenuhi harapan audiens target Anda: http://www.fontshop.com/blog/newsletters/fontbookipad/

Kembangkan Perpustakaan Tipe Anda

Bagi juru ketik berpengalaman, membeli tipografi adalah sebuah komitmen besar. Hal ini sebanding dengan membeli pakaian baru karena idealnya pakaian tersebut harus sesuai dengan barang yang sudah kita miliki di lemari, dan memiliki daya tahan tertentu. Belilah set yang melengkapi perpustakaan Anda yang sudah ada, jika tidak, Anda mungkin menemukan diri Anda memiliki jenis huruf yang mungkin tidak akan pernah digunakan, yang mengakibatkan barang dagangan bernilai beberapa ratus dolar dibuang ke luar jendela.

Sangat jarang ada desainer bunglon yang sukses dan kenyataannya kita semua memiliki gaya yang khas. Beberapa dari kita lebih baik dalam merancang satu halaman, sementara yang lain lebih baik dalam merancang sistem informasi multifaset. Beberapa desainer mengkhususkan diri pada toko Web, yang lain mengkhususkan diri pada aplikasi Web. Itulah mengapa sangat penting untuk menemukan dan bekerja dengan serangkaian tipografi yang sesuai dengan gaya khusus Anda dan jenis proyek yang biasanya Anda kerjakan. Jangan berpikir bahwa desainer terkenal menggunakan banyak sekali tipografi yang berbeda. Sebaliknya, jika Anda membandingkan proyek mereka yang paling sukses, semuanya menggunakan pilihan font yang terbatas.

Ini bukanlah hal yang aneh bagi Anda atau desainer mana pun untuk akhirnya menggunakan font dari pengecoran atau perancang tipe yang sama berulang kali, karena struktur dasar yang sudah dikenal dari masing-masing mesin terbang bersama dengan DNA umum mungkin paling sesuai dengan gaya penyusunan huruf pribadi Anda.

Mengelola perpustakaan font akan merepotkan tanpa perangkat lunak manajemen font seperti Fontcase atau Linotype FontEXplorer X Pro. Setelah Anda mengklasifikasikan font di perpustakaan Anda, melihat pratinjau dan mengujinya seharusnya relatif mudah. Cukuplah untuk mengatakan, buat cadangan file preferensi Anda!

Temukan inspirasi

Latih diri Anda untuk mengenali dan mengenali pola tipografi. Ke mana pun kita pergi, kita selalu membawa ponsel pintar, jadi ambillah foto tipografi menarik apa pun yang Anda temukan di jalan, di perpustakaan, atau di toko buku. Sumber inspirasi berharga lainnya adalah surat kabar cetak, majalah mode, atau selebaran yang terdapat di lobi hotel. Kumpulkan setiap yang Anda lihat dan tulis komentar tentang mengapa Anda mengambilnya di catatan Post-it. Masih belum membawa satu blok Post-it di perjalanan Anda? Inilah insentif untuk mulai melakukannya.

Yang terakhir, namun tidak kalah pentingnya, belilah sebanyak mungkin spesimen dan berlangganlah setiap buletin tipografi yang ada di luar sana. Seiring waktu, Anda akan memiliki referensi bagus yang berisi kombinasi jenis yang telah terbukti dan dapat digunakan kembali yang dibuat oleh desainer dan juru ketik berpengalaman. Salah satu nasihat otobiografi, suatu bentuk nostalgia jika Anda mau - berhati-hatilah agar tidak ketahuan meneteskan air liur ke kaki serif yang rasional atau menampilkan senyum berseri-seri seperti pencium pertama kali sambil menyentuh alat tulis yang diukir.

9.4 SEJARAH TIPOGRAFI

Nasihat yang sangat umum diberikan oleh juru ketik berpengalaman kepada pendatang baru adalah mempelajari sejarah jenis huruf. Maksudnya itu apa? Apa hubungan periode seperti Renaissance dengan proyek yang sedang dikerjakan? Mempelajari bagaimana tipografi muncul melalui sejarah juga dapat membantu kita memahami anatomi tipografi. Banyak desain tipe yang merupakan hasil teknologi dan kebutuhan praktis pada masa itu. Misalnya, jenis huruf dengan perangkap tinta terlihat konyol di layar, kecuali jika digunakan sebagai jenis tampilan untuk menyampaikan pesan tertentu. Jenis huruf yang direproduksi pada zaman pengaturan foto, ketika jenis huruf dirancang sedikit lebih lembut, dapat terlihat kurang memuaskan pada layar yang menggunakan kisi-kisi yang kaku.

Di dunia kreatif, semuanya neo-sesuatu-dari-masa lalu dan semuanya merupakan remix46. Wajar jika menggunakan kembali metafora dan desain lama di pundak para pendahulu kita yang mulia. Saat sebuah tren muncul, ada gunanya mempelajari asal usulnya. Pada bulan Mei 2013, Yves Peters, editor di Font Feed, memberikan ceramah tentang Trajan Pro sebagai tipografi poster film. Dia menjelaskan bagaimana tren ini muncul dan meramalkan tren baru penggunaan Gotham dalam poster film. Kedua tipografi tersebut menjadi populer berdasarkan kesuksesan mereka sebelumnya Trajan Pro menjadi font yang digunakan dalam film yang memenangkan Oscar dan Gotham menjadi font resmi kampanye pemilu Presiden Obama tahun 2008. Karena semuanya hanyalah iterasi dari beberapa ide sebelumnya, kita dapat menarik koneksi ke ide awal dan menggunakan jenis huruf yang cocok dengan periode tertentu.

Terkadang mengetahui sejarah juga dapat membantu Anda membangun hubungan yang lebih langsung. Misalnya, Arno adalah jenis huruf yang terinspirasi oleh tipografi humanis awal abad ke-15 dan ke-16, yang diberi nama berdasarkan nama sungai Florentine. Ini juga terbukti sangat cocok untuk situs web yang menawarkan vila mewah untuk disewa di

Dubrovnik, Kroasia49. Mengetahui bahwa Dubrovnik di masa lalu memiliki hubungan budaya dan ekonomi yang kuat dengan Venesia dan Florence yang memengaruhi perkembangan dan kemakmuran kota tersebut pada abad kelima belas dan keenam belas memudahkan kami memilih Arno sebagai jenis huruf utama untuk situs web.

Organisasi dan Kinerja CSS

Dokumen HTML yang terstruktur dengan baik dapat diakses dan memenuhi semua persyaratan dasar pembaca, bahkan tanpa CSS. Teks dapat diperbesar, dipilih, disalin, dan dibagikan. Ukuran huruf secara default cukup besar agar nyaman dibaca sejauh lengan dan tata letaknya fleksibel. Jadi fallback sudah ada dan kita hanya perlu memastikan penyempurnaan yang kita tambahkan tidak menghalangi, terutama jika browser kurang mumpuni atau kecepatan koneksinya buruk.

Meskipun demikian, cara terbaik untuk mengelola tipografi responsif dengan CSS adalah dengan menangani tipografi mikro dan default paling umum di file CSS utama, sambil menempatkan semua tipografi khusus breakpoint ke dalam file terpisah.

File CSS utama harus berisi aturan default untuk:

- tumpukan font
- hierarki dokumen
- ukuran huruf relatif untuk judul, subjudul, keterangan, dan catatan samping
- tebakan terbaik tinggi garis berdasarkan panjang garis optimal
- lebar maksimum untuk elemen konten

File CSS khusus breakpoint dapat berisi:

- nilai skala tipografi yang berbeda berdasarkan kepadatan informasi
- penggantian ketinggian garis berdasarkan penambahan (atau penurunan) panjang garis
- aturan khusus tata letak

Hal pertama yang selalu disarankan adalah mengatur ulang margin dan padding. Apakah Anda lebih suka menggunakan reset.css50 Eric Meyer atau reset asterisk yang lebih radikal, tujuannya adalah untuk mengambil kendali penuh atas dokumen. Ini mungkin tampak berlebihan pada awalnya, tetapi hal ini memaksa kita untuk meninjau kembali semua elemen dan memperhatikan detail terkecil.

```
* { margin: 0; padding: 0 }
html { font-size: 100%; line-height: 1.5em; }
```

Karena panjang garis yang ideal adalah sekitar 66 karakter51, kita dapat membatasi lebar maksimum blok teks dasar menjadi 33em, mengingat rata-rata lebar karakter adalah sekitar setengah em. Agar semuanya relatif aman dan sederhana, margin bawah pada elemen level blok dapat diatur ke nilai yang sama dengan tinggi garis, yang dalam hal ini adalah 1,5em. Dengan cara ini, kita mempertahankan ritme vertikal - pola berulang yang biasa dilakukan otak kita - membantu mata kita melompat dua kali lipat tinggi garis dan bukan panjangnya sembarangan, yang tidak berirama.

```
article { max-width: 33em; }
p, ul, ol, dl, table { margin-bottom: 1.5em; }
```

Kami juga dapat memperluas aturan lebar maksimal untuk pemirsa internasional. Berdasarkan artikel Vasilis van Gemert “Logical Breakpoints For Your Responsive Design”52, Jordan Moore mengemukakan gagasan tentang panjang garis berbasis bahasa53. Karena panjang garis ideal mungkin berbeda dari satu bahasa ke bahasa lain, kita dapat menerapkan nilai maksimum panjang garis khusus bahasa:

```
article { max-width: 33em; }
:lang(de) article { max-width: 40em; }
```

9.5 JARINGAN DASAR

Untuk membantu kita menyusun ritme vertikal, kita dapat membuat pola berulang di latar belakang dan menyesuaikan grid dasar sesuai kebutuhan kita. Kami dulu membuat GIF untuk ini, tapi hari ini kami dapat membuat gradien secara dinamis di CSS.

```
html {
     background-image:  -webkit-linear-gradient(top,  #fff  0,
#fff 95%, #f00
95%, #f00 100%);
     background-image: -moz-linear-gradient(top, #fff 0, #fff
95%, #f00
95%, #f00 100%);
     background-image: linear-gradient(top, #fff 0, #fff 95%,
#f00 95%,
#f00 100%);
     background-repeat: repeat-y; background-size: 100% 24px;
/* Background size height equals rendered line height at the
respective
breakpoint */
     }
```

Desainer web merasa sangat sulit untuk menyusun ritme vertikal dan saya setuju. Ini bukan keterampilan yang paling mudah untuk dikuasai, tetapi ini tidak berarti kita harus menyerah. Daniel Eden dan Matt WilcoX masing-masing mengembangkan plugin jQuery yang menghitung margin bawah gambar yang tidak sesuai ritme. Dengan semakin banyaknya solusi seperti ini, tidak ada alasan untuk tidak menyelaraskan semuanya ke grid dasar. Pada awalnya, ini adalah sebuah perjuangan, namun seiring berjalannya waktu, menyusun grid dasar menjadi kebiasaan.

Font sebagai Peningkatan Progresif

Dengan banyaknya browser yang masih belum mampu menangani font Web secara optimal (misalnya Android 2.2–2.4) atau kurang mendukung fitur yang lebih canggih (misalnya URI data di IE6–8), font Web harus dianggap sebagai penyempurnaan progresif. Mereka masih belum bekerja secara optimal dan untuk membuatnya bekerja dengan baik terkadang kita perlu menggunakan sintaksis yang rumit.

Secara default, browser tidak perlu mendownload font untuk merender halaman Web, browser akan menggunakan font yang sudah tersedia di sana. Font Web kustom kami merupakan override dan karenanya terkadang dapat memberikan beberapa efek yang tidak diinginkan, seperti kesalahan teks tanpa gaya (FOUT).

Google dan Typekit berkolaborasi untuk membuat Web Font loader56, perpustakaan JavaScript yang berfungsi dengan sebagian besar layanan font Web serta font yang dihosting sendiri. Ini sangat mudah digunakan dan, dalam kasus Typekit, ini sudah ada di dalam cuplikan standar yang dapat disematkan. Jika Anda menghosting sendiri atau menggunakan layanan lain, kode berikut harus ditempelkan ke <head> dokumen:

```
<script type=”text/javascript”>
     WebFontConfig = {
        custom: {families: 'Font Family Name', 'Another Font
Family'], urls: 'http://domain.com/fonts.css']}
};

(function() {
   var wf = document.createElement('script');
   wf.src = ('https:' == document.location.protocol ? 'https' :
  'http')                                                    +
'://ajax.googleapis.com/ajax/libs/webfont/1/webfont.js';
   wf.type = 'text/javascript';
   wf.async = 'true';
   var s = document.getElementsByTagName('script')[0];
   s.parentNode.insertBefore(wf, s);
  })();
</script>
```

Kemudian Anda memiliki tiga kelas yang dapat Anda gunakan: .wf-loading, .wf-active dan .wf-inactive yang diatur pada elemen <html>. Kelas yang paling menarik adalah .wf-loading yang digunakan untuk mengontrol apa yang terjadi hingga font Web diunduh dan diterapkan pada teks. Berikut ini contohnya:

```
.wf-loading h1 { visibility: hidden; }
/* — or — */
.wf-loading h1 { font-family: 'A fallback font'; }
```

Saat membuat tumpukan font, hasil terbaik dicapai jika: tipe wajah yang dipilih termasuk dalam klasifikasi yang sama; dan masing-masing karakter mencakup jumlah ruang fisik yang sama. Ini berarti font fallback yang sesuai akan memiliki kualitas berikut, diurutkan berdasarkan kepentingannya:

- mencocokkan tinggi x
- lebar karakter serupa
- spasi huruf serupa

Tumpukan font antipeluru harus berisi jenis huruf pilihan pertama, alternatif terdekat, alternatif spesifik platform, alternatif universal, dan terakhir alternatif umum. Jika Anda menghosting sendiri font Web, Font Squirrel Webfont Generator yang hebat, di antara opsi lainnya, memungkinkan Anda memperbaiki metrik vertikal dan mengubah ukuran mesin terbang agar sesuai dengan tinggi x font aman Web yang dipilih (Arial, Verdana, Trebuchet, Georgia, Times New Roman dan Kurir).

Sub-pengaturan

Kami dapat meningkatkan kinerja secara drastis dengan mengurangi ukuran file font Web dan dengan mengurangi jumlah permintaan HTTP (kita akan membahasnya sebentar lagi). Biasanya juga file font berisi banyak skrip yang tidak selalu diperlukan atau diwajibkan. Jika hanya sebagian mesin terbang dari file font yang diperlukan, mengapa mengunduh keseluruhan font?

Beberapa layanan font Web menawarkan subset, seperti huruf kecil atau gambar lapisan sebagai font terpisah. Yang lain memungkinkan Anda memilih rentang karakter yang Anda perlukan sebelum mengekspor. Meskipun ini merupakan langkah maju yang bagus, dalam banyak kasus, masih tidak selalu mungkin untuk membuat subset font berdasarkan mesin terbang demi mesin terbang.

Jika font Web dihosting sendiri, kita dapat mengelompokkannya dengan FF Subsetter atau Font Squirrel yang disebutkan di atas. Kedua alat tersebut memungkinkan Anda memilih dan melihat pratinjau karakter yang Anda perlukan dalam font, serta menghapus petunjuk, kerning, dan mesin terbang opsional, seperti angka gaya lama. Meski semua opsi tersebut tersedia, bukan berarti harus dimanfaatkan. Berhati-hatilah untuk tidak melumpuhkan font dengan menghapus tanda baca, tanda hubung, spasi, dan lainnya.

eh simbol non-abjad yang umum. Jika kerning bawaan dihilangkan, jenis huruf akan terlihat kurang memuaskan di Windows. Bahkan properti text-render atau font-kerning tidak akan mengetahui cara menangani pasangan kerning dengan benar. Jelasnya, kita tidak boleh mengorbankan kualitas rendering, jadi akan ada saatnya Anda harus menghadapi kenyataan dan hanya menggunakan font yang aman untuk Web.

Pengkodean Base64

Cara lain untuk mengoptimalkan kinerja adalah dengan mengurangi jumlah permintaan HTTP dengan menyertakan font Web sebagai string berkode Base64 di file CSS utama. URI data didukung di browser modern (IE9 dan yang lebih baru), sehingga Anda dapat menyajikan file font terpisah untuk IE8 dan pendahulunya, atau menghilangkan font Web di

browser lama. Beberapa situs web terkenal, seperti The Guardian dan GOV.UK menggunakan metode ini untuk menyajikan font Web mereka.

```
@font-face {
     font-family: "My Font";
     src:   url("data:font/opentype;base64,[base-encoded   font
here]");
}
```

Jika font Web dihosting sendiri, kita dapat mengelompokkannya dengan FF Subsetter atau Font Squirrel yang disebutkan di atas. Kedua alat tersebut memungkinkan Anda memilih dan melihat pratinjau karakter yang Anda perlukan dalam font, serta menghapus petunjuk, kerning, dan mesin terbang opsional, seperti angka gaya lama. Meski semua opsi tersebut tersedia, bukan berarti harus dimanfaatkan. Berhati-hatilah untuk tidak melumpuhkan font dengan menghapus tanda baca, tanda hubung, spasi, dan simbol non-abjad umum lainnya. Jika kerning bawaan dihilangkan, jenis huruf akan terlihat kurang memuaskan di Windows. Bahkan properti text-render atau font-kerning tidak akan mengetahui cara menangani pasangan kerning dengan benar. Jelasnya, kita tidak boleh mengorbankan kualitas rendering, jadi akan ada saatnya Anda harus menghadapi kenyataan dan hanya menggunakan font yang aman untuk Web.

Pengkodean Base64

Cara lain untuk mengoptimalkan kinerja adalah dengan mengurangi jumlah permintaan HTTP dengan menyertakan font Web sebagai string berkode Base64 di file CSS utama. URI data didukung di browser modern (IE9 dan yang lebih baru), sehingga Anda dapat menyajikan file font terpisah untuk IE8 dan pendahulunya, atau menghilangkan font Web di browser lama. Beberapa situs web terkenal, seperti The Guardian dan GOV.UK menggunakan metode ini untuk menyajikan font Web mereka.

Breakpoint Kueri Media Berbasis Konten

Untuk menentukan di mana memperkenalkan breakpoint, kita dapat merujuk pada kutipan Jeremy Keith, salah satu pendiri dan direktur teknis di Clearleft: Breakpoint tidak boleh ditentukan oleh perangkat, namun oleh konten. Biarkan konten memutuskan kapan akan memperluas dan kemudian menyesuaikan desain Anda.

Dalam praktiknya, ini berarti kita dapat memulai dengan kolom teks default tanpa kueri media, lebar 100%, dengan atau tanpa batas lebar maksimal yang disebutkan di atas. Dengan cara ini konten akan memenuhi layar pada ukuran yang lebih kecil dari batas lebar maksimal dan memberikan lebih banyak ruang untuk kolom tambahan pada layar yang lebih lebar dari nilai lebar maksimal. Ketika mencapai tanda lebar maksimal, kita memiliki dua opsi. Pertama, kita bisa mulai menambah spasi (misalnya, dengan margin: auto; rule) hingga ada cukup ruang untuk muat di kolom baru. Kedua, kita dapat segera memperkenalkan breakpoint baru, mengecilkan kolom asli hingga batas minimum yang dapat diterima, sambil memasang kolom baru ke dalam ruang yang tersisa.

Ada trade-off yang penting ketika memilih di antara kedua opsi ini. Dengan tata letak yang lebih terkontrol, kepadatan informasi jauh lebih rendah, yang mungkin menjadi kontra-efektif, terutama dalam aplikasi Web atau apa pun yang seharusnya menyediakan banyak informasi dengan cepat. Di sisi lain, jauh lebih sulit untuk menjaga panjang garis dalam batas optimal dengan tata letak yang sepenuhnya fleksibel, dan biasanya menghasilkan lebih banyak titik henti sementara dan titik penyesuaian yang ditetapkan. Seperti biasa, kebutuhan proyek Anda harus memengaruhi keputusan ini.

Menghitung panjang garis ideal bukanlah hal yang paling mudah untuk dilakukan. Itu sebabnya Trent Walton mengusulkan penambahan tanda bintang setelah karakter ke-45 dan ke-75 dalam paragraf, yang membuat hidup lebih mudah ketika mengamati konten yang dialirkan ulang pada titik henti sementara yang berbeda.

Beberapa pengguna memperbesar situs web. Untuk memungkinkan penskalaan proporsional dengan teks yang diperbesar, kita dapat menggunakan ems sebagai pengganti piksel seperti yang diusulkan oleh Lyza Gardner dalam artikelnya “EMs has it: Proportional Media Queries FTW!”. Efeknya terjadi ketika teks diperbesar dan kondisi breakpoint bawah terpenuhi. Misalnya, breakpoint ideal untuk sebuah proyek adalah 600, 800, dan 1.000 piksel. Jika kita membagi nilai piksel dengan ukuran font akar, yang dalam contoh ini adalah 16pX, nilai kueri berbasis em akan masing-masing menjadi 37,5, 50, dan 62,5ems.

```
@media only screen and (min-width: 37.5em) { } /* 600px */
@media only screen and (min-width: 50em) { } /* 800px */
@media only screen and (min-width: 62.5em) { } /* 1000px */
```

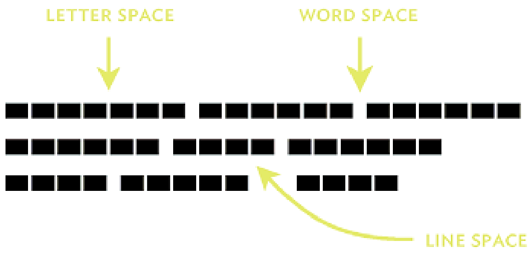
Terlepas dari kenyataan bahwa mereka telah mengubah cara kita merencanakan dan membangun situs web dari awal, pertanyaan media masih belum menjadi solusi akhir. Mereka sempurna untuk memperkenalkan tata letak yang berbeda, namun unit konten harus berperilaku dengan cara tertentu tergantung pada ruang yang tersedia, tidak hanya berdasarkan ukuran layar keseluruhan. Salah satu contoh tipikal adalah tabel data yang dapat dengan mudah diubah menjadi daftar definisi pada layar kecil dengan lebar 300pX, namun pada saat yang sama tidak dapat disusun ulang dengan cara yang sama ketika ditempatkan di sidebar sempit dengan lebar 300pX. di layar yang lebih besar. Bisakah Anda melihat masalahnya di sini? Bagaimana kita bisa menanyakan hal itu?

Andy Hume mengemukakan konsep container responsif yang ia uraikan di blognya, jadi saya tidak akan membahas detail pastinya di sini. Untuk saat ini, yakinlah bahwa masalahnya telah terdeteksi dan beberapa orang pintar sedang mencari solusinya. Semoga saja itu menjadi bagian dari standar CSS.

Hirarki Ruang Putih

Ada tiga spasi putih dalam paragraf yang saling bergantung spasi huruf, spasi kata, dan spasi baris. Jika salah satu dari ketiga hal tersebut diubah, maka semua hal lainnya juga harus ditinjau kembali. Dengan mempertahankan hierarki antar spasi, kita dapat memperoleh

pengalaman membaca terbaik untuk berbagai skenario (tempo bervariasi, panjang baris berbeda).

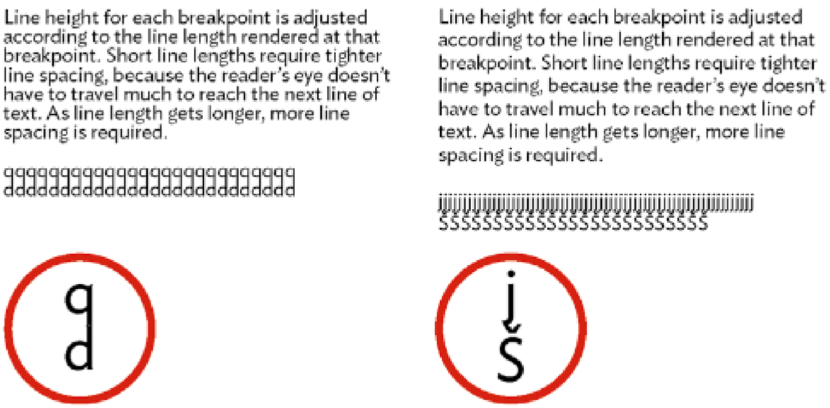


Gambar 9.3 Ilustrasi ruang putih

Ukuran Font, Leading dan Ukuran

Hubungan mendasar lainnya adalah keterkaitan antara ukuran font, tinggi garis, dan panjang garis. Panjang garis ditentukan oleh kombinasi ukuran font dan ruang horizontal yang tersedia dalam wadah. Semakin panjang garisnya, semakin banyak ruang yang kita perlukan antar garis yang berurutan dan sebaliknya. Sebagai akibat dari ketergantungan ini, jika salah satu dari ketiganya diubah, maka ketiga hal lainnya juga harus ditinjau ulang.

Seperti yang diketahui oleh setiap desainer Web, panjang baris yang memuaskan adalah antara 45 hingga 75 karakter per baris64. Merupakan hal yang umum di kalangan desainer Web untuk memberi jarak pada garis tersebut dengan tinggi garis 1,5ems, kurang lebih. Masalahnya, kita hampir tidak bisa memuat 40 karakter di layar ponsel. Ketinggian garis 1,5 dikombinasikan dengan panjang garis yang pendek menghasilkan jarak yang terlalu lebar. Jika paragraf terlihat seperti daftar, kurangi jarak antar baris.



Seberapa ketatkah itu terlalu ketat? Ujilah dengan deretan huruf q atau p diikuti dengan deretan bs atau ds. Untuk audiens internasional Anda mungkin ingin memasukkan huruf besar dengan diakritik, misalnya Š.

Jelasnya, ada batas alami untuk panjang garis dan tinggi garis. Meskipun kita perlu mengurangi tinggi garis pada panjang garis yang pendek, kita harus berhati-hati agar tidak bertabrakan dengan pemanjang garis pada baris yang berurutan.

Panjang garis dapat dengan mudah dikontrol dengan lebar minimum dan maksimum di CSS, namun saat ini hanya ada satu nilai tetap yang tersedia untuk tinggi garis di CSS.

Meskipun demikian, kita dapat mengubah ukuran font, tinggi garis, dan margin bawah dengan kueri media, bergantung pada panjang garis pada titik henti sementara tertentu. Kode paragraf pada breakpoint yang berbeda akan terlihat seperti ini:

```
@media only screen and (min-width: 37.5em) {
    p {
     font-size: 1rem; /* 16px */
     line-height: 1.4;
     margin-bottom: 1.4em; /* = line-height */
    }
}
@media only screen and (min-width: 50em) {
    p {
     font-size: 1.125rem; /* 18px */
     line-height: 1.45;
     margin-bottom: 1.45em; /* = line-height */
    }
}
@media only screen and (min-width: 62.5em) {
      p {
      font-size: 1.312rem; /* 21px */
      line-height: 1.5;
      margin-bottom: 1.5em; /* = line-height */
    }
 }
```

Namun, ini bukanlah solusi terbaik dan sangat bergantung pada jumlah breakpoint. Semakin banyak breakpoint yang kami perkenalkan, semakin baik pengalamannya. Tapi bukankah akan lebih mudah jika aturan matematis sederhana seperti itu diterapkan secara otomatis?

Pemimpin Cair

Dalam “Molten lead (or, fluid line-height)”, Tim Brown mengusulkan aturan rentang untuk tinggi garis dalam CSS, yang akan memungkinkan desainer menentukan batas tinggi garis dan membiarkan browser mengubah teks mengikuti perubahan lebar fluida wadah. Dia menyarankan sepasang properti CSS baru min- dan max-line-height, yang akan berperilaku serupa dengan min- dan max-width. Seperti yang akan kita lihat nanti di bab ini, Draf Kerja CSS3 memperkenalkan pendekatan lain untuk menangani nilai rentang.

Berdasarkan ide Brown, Mat Marquis membuat Molten Leading66, sebuah plugin jQuery yang memecahkan masalah tersebut. Sintaksnya semudah:

```
$('p').moltenLeading({
      minline: 1.4,
      maxline: 1.7
});
```

Gaya Paragraf

Jika teksnya lugas dan linier, serta pemikirannya terhubung erat, kita dapat menggunakan cara alternatif yang lebih intim untuk membagi paragraf. Hal ini sangat berguna pada ukuran layar kecil, di mana tidak banyak informasi yang bersaing untuk menarik perhatian pembaca.

```
p { margin-bottom: 0; }
p + p { text-indent: 1em; }
```

Contoh ini hanyalah trik sederhana untuk menjejalkan lebih banyak konten ke dalam layar tanpa mengorbankan keterbacaan. Untuk mempelajari lebih banyak opsi untuk menata paragraf di Web, bacalah " Contoh Tipografi Paragraf 12" karya Jon Tan dan bab tipografi Web yang ia tulis bersama di Smashing Book #1.

Spasi Huruf

Secara default, huruf-huruf di layar dapat tampak terlalu berdekatan satu sama lain, terutama pada layar dengan kepadatan piksel tinggi. Hal ini dapat diperbaiki dengan cukup mudah dengan menggunakan properti spasi huruf. Mobile Safari di iPhone menerima penambahan spasi huruf sebesar 0,01ems, yang berguna pada perangkat iOS.

Meskipun IE7+ dan versi FirefoX yang lebih baru mampu menghasilkan peningkatan sekecil itu, kekasaran layar 96dpi biasanya memberikan hasil yang kurang memuaskan, jadi sebaiknya simpan nuansa seperti itu hanya untuk layar dengan dpi tinggi.

```
     @media only screen and (-webkit-min-device-pixel-ratio: 1.5),
only screen and (min-device-pixel-ratio: 1.5) {
          p {
               font-size: 1rem;
               line-height: 1.5;
               letter-spacing: 0.01em;
               word-spacing: 0.01em;
           }
}
```

Jarak huruf dan kata yang benar seharusnya tidak terlihat oleh mata yang tidak terlatih, jadi jika rata-rata pembaca dapat melihat celah antar huruf, Anda mungkin sudah keterlaluan.

Tanda Hubung dan Pembenaran

Untuk menyamakan panjang baris individual, paragraf dapat diatur rata. Untuk mencegah kesenjangan antar kata dalam pengaturan yang dibenarkan, teks harus diberi tanda hubung. Tanda hubung dan pembenaran dasar dapat dilakukan di semua browser modern:

```
p {
     text-align: justify;
     -webkit-hyphens: auto;
     -moz-hyphens: auto;
```

```
     -ms-hyphens: auto;
     hyphens: auto;
}
```

Sayangnya, hal ini belum cukup dan hasilnya jauh dari dapat diterima. Selain tanda hubung, ritme baris dalam paragraf rata harus dipertahankan dengan kombinasi spasi kata yang bervariasi, spasi huruf, dan lebar karakter. Kita bisa lolos tanpa menggunakan lebar karakter variabel, namun spasi variabel adalah suatu keharusan untuk paragraf yang diratakan dengan benar. Ini adalah contoh lain di mana properti min- dan max- dapat berperan, misalnya:

```
p {
     text-align: justify;
     min-word-spacing: -0.05em;
     max-word-spacing: 0.1em;
     min-letter-spacing: -0.01em;
     max-letter-spacing: 0.02em;
}
/* A concept code */
```

Draf Kerja Modul Teks CSS3 Level 3 mengusulkan jenis nilai batas spasi baru, yang mewakili spasi optimal, minimum, dan maksimum dalam spasi kata dan spasi huruf. Sintaksnya dalam hal ini adalah:

```
p {
   text-align: justify;
   word-spacing: 0 -0.05em 1em;
   letter-spacing: 0 -0.01em 0.02em;
}

/* optimum, minimum, maximum respectively */
```

Masalah lain dengan tanda hubung adalah kurangnya kontrol menyeluruh terhadap tingkat keparahan dan terjadinya tanda hubung. Ini dianggap sebagai jenis kejahatan jika lebih dari tiga baris berturut-turut diberi tanda hubung. Selain itu, Robert Bringhurst, penulis The Elements of Typographic Style menyarankan bahwa setidaknya tiga karakter dari kata yang diberi tanda penghubung harus dibawa ke baris berikutnya.

Draf Kerja Media Halaman CSS3 mengusulkan beberapa properti menarik (masih banyak lagi, tapi menurut saya ini adalah yang paling penting):

- tanda hubung-sebelum menentukan jumlah minimum karakter dalam sebuah kata sebelum karakter tanda hubung

- tanda hubung-setelah menentukan jumlah minimum karakter dalam sebuah kata setelah karakter tanda hubung
- garis tanda hubung menentukan jumlah maksimum garis tanda hubung yang berurutan dalam sebuah elemen

Jika ini didukung oleh browser, rangkaian aturan CSS selanjutnya akan terlihat seperti ini:

```
{
     hyphens: auto;
     hyphenate-before: 2;
     hyphenate-after: 3;
     hyphenate-lines: 3;
}
```

Draf Editor Teks CSS Level 4 tahun 2013 mengusulkan perluasan ke tanda hubung-sebelum dan tanda hubung-sesudah dalam bentuk properti tanda hubung-batas-karakter yang akan mengambil tiga nilai: nilai minimum yang diperlukan untuk total karakter dalam sebuah kata; dan jumlah minimum karakter dalam sebuah kata sebelum dan sesudah tanda hubung. Draf yang sama mengganti nama properti garis hubung menjadi garis batas tanda hubung yang pada saat penulisan ini didukung di IE10 dan Safari.

Untuk saat ini, hasil terbaik bagi pengguna masih dapat dicapai dengan tanda hubung manual. Untuk memberi tanda hubung pada teks secara manual, cukup sisipkan tanda hubung lembut &#1; (atau) dimanapun yang nyaman. Jika perlu jeda kata, browser akan menyisipkan karakter tanda hubung. Ini berfungsi seperti yang diiklankan di semua browser modern.

```
oto&#173;rhino&#173;laryngo&#173;logy

p { hyphens: manual; }
```

9.6 MENGGABUNGKAN ELEMEN PSEUDO DENGAN PROPERTI KONTEN

Saya penggemar berat penggabungan elemen semu dengan properti konten. Meskipun elemen tersebut tidak didukung di browser lama, kita dapat menggunakannya untuk mendapatkan kontrol yang lebih baik terhadap simbol gantung, tanda kutip, dan penomoran bertingkat.

Simbol dan Angka Gantung

Daripada mengandalkan properti gaya daftar yang tidak dapat kita kendalikan, kita dapat menggunakan elemen semu :before untuk menambahkan poin atau tanda hubung sebelum teks dalam item daftar dan mendapatkan kembali kendali penuh atas simbol tersebut. Pilihannya sekarang tidak terbatas dan salah satu yang langsung terlintas dalam pikiran adalah menggunakan warna abu-abu yang berbeda untuk peluru gantung.

```
{
     list-style: none;
}
ul li:before {
     content: "-"; /* en dash */
     float: left;
     margin-left: -1em;
     color: #999;
}
ol { counter-reset: item }
/* “item” is an arbitrary variable, by default equals zero */

ol li:before {
     content: counter(item) ". ";
     counter-increment: item; /* equals item++ */
     float: left;
     margin-left: -1em;
}
```

Tanda kutip

Mengikuti prinsip yang sama dari bagian sebelumnya, kita dapat menambahkan sedikit keanggunan pada blockquote.

```
blockquote:before {
      content: "“";
      font-size: 4em;
      color: #eee;
      float: left;
      margin: -.33em 0 0 -.5em;
}
/* — or – */
blockquote p:first-child: before {
 content: "“";
 font-size: 4em;
 color: #eee;
float: left;
 margin: -.33em 0 0 -.5em;
}
blockquote p:last-child: after { content: "”"; }
```

Untuk memberikan sepasang tanda kutip untuk masing-masing bahasa beserta alternatifnya, kita akan menggunakan pemilih :lang:

```
q:lang(de) { quotes: '„' '“' '‚' "‘" }
```

```
q:lang(en-gb) { quotes: "‘" "’" '‘' '’' }
q:lang(en-us) { quotes: '‘' '’' "‘" "’" }
q:lang(fr) { quotes: '« ' ' »' "“" "”" }
q:before { content: open-quote }
q:after { content: close-quote }
```

Setiap bahasa mempunyai aturannya masing-masing, jadi saya mendorong Anda untuk membaca “Penggunaan tanda kutip non-Inggris” di Wikipedia.

Penomoran Bertingkat Otomatis

Penomoran otomatis dalam konten bertingkat memberi pembaca garis besar yang mudah diikuti dan menerapkan hierarki. Dengan penomoran otomatis, penataan gaya item daftar yang diurutkan dalam “1., 1.1., 1.1.1.” fashion menjadi sangat mudah.

```
ol { counter-reset: item }
ol li { display: block }
ol li:before {
     content: counters(item, ".") ". ";
     counter-increment: item;
}
```

Vladimir Simović melangkah lebih jauh dan menjelaskan cara menerapkan penomoran otomatis pada heading75 yang sangat berguna untuk penerbitan eBook.

```
body { counter-reset: subhead1; }

h1:before {
     content: counter(subhead1) " ";
     counter-increment: subhead1;
}
h1 {
     counter-reset: subhead2;
}
h2:before {
     content: counter(subhead1) "." counter(subhead2) " ";
     counter-increment: subhead2;
}
h2 {
     counter-reset: subhead3;
}

h3:before {
     content: counter(subhead1) "." counter(subhead2) "."
          counter(subhead3) " ";
```

```
      counter-increment: subhead3;
}
```

Hirarki

Anda tidak akan pernah menduganya! Tipografi juga dapat membantu meresmikan struktur dokumen. Subjudul memberi pembaca petunjuk tentang konten berikut; catatan samping dan catatan kaki memberi kita penjelasan tambahan tentang istilah-istilah kunci di dalam teks; dan keterangan memberikan deskripsi untuk gambar, tabel data, dan grafik. Semua elemen ini merupakan bagian dari hierarki tipografi.

Timbangan Modular

Cara paling umum untuk menetapkan hierarki adalah dengan skala tipografi. Tim Brown menciptakan Modular Scale, sebuah kalkulator online yang mengembalikan serangkaian nilai terkait berdasarkan ukuran body copy tertentu dan beberapa skala matematika umum serta interval musik. Kalkulator ini dinamai konsep tipografi dengan nama yang sama. Seperti yang dikatakan Brown dalam artikelnya "Tipografi Lebih Bermakna:

Skala modular adalah rangkaian angka yang berhubungan satu sama lain secara bermakna. Untuk keperluan bab ini, saya memilih 16pX sebagai ukuran teks ideal, dan skala kelima sempurna 2:3, yang memberi saya serangkaian nilai berguna: 7.111, 10.667, 16 (nilai awal), 24, 36, dan 54. Diterjemahkan ke rem dan diterapkan ke subjudul dan keterangan di CSS, tampilannya seperti ini:

```
h4 { font-size: 1rem }
h3 { font-size: 1.5rem }
h2 { font-size: 2.25rem }
h1 { font-size: 3.375rem }
caption { font-size: 0.667rem }
small { font-size: 0.444rem }
```

Gaya Subjudul

Meskipun skala tipografi yang dikombinasikan dengan bobot font yang berbeda adalah cara yang bagus untuk membangun hierarki dan keseimbangan di layar besar, variasi tersebut dapat menghalangi di layar kecil. Dengan layar besar terdapat lebih banyak elemen informasi dan konten utama seringkali dapat bersaing dengan iklan dan konten terkait atau tidak terkait lainnya.

Pada layar yang lebih kecil, hanya ada beberapa elemen di layar secara bersamaan dan rasio signal-to-noise jauh lebih baik. Subjudul 36pX yang terlihat bagus di monitor desktop, mungkin sedikit berlebihan di ponsel cerdas. Jadi, temui variasi gaya. Daripada menambah ukuran font secara bertahap untuk setiap tingkat judul, kita dapat menggunakan huruf miring, huruf kecil, dan huruf besar semua dengan ukuran huruf yang sama untuk menetapkan hierarki teks. Tambahkan huruf tebal ke dalam campuran, dan kami memiliki palet yang terdiri dari setidaknya enam gaya.

Meskipun semua huruf besar dan miring dapat ditata dengan mudah menggunakan CSS, huruf kecil relatif baru dalam desain Web. Font web lebih jarang mengandung subset huruf kecil. Intinya, ada tiga alternatif penerapan topi kecil:

1. Terapkan aturan fitur OpenType CSS3.
2. Muat file font huruf kecil yang terpisah.
3. Buat topi kecil palsu.

```
.subhead-1 {
     -moz-font-feature-settings: "smcp";
     -ms-font-feature-settings: "smcp";
     -webkit-font-feature-settings: "smcp";
     -o-font-feature-settings: "smcp";
     font-feature-settings: "smcp";
     text-transform: lowercase;
}
.subhead-2 {
     font-family: small-caps-typeface;
     text-transform: lowercase;
}
.subhead-3 {
     font-weight: 500;
     font-size: 80%;
     text-transform: uppercase;
     letter-spacing: .125em;
}
```

Salah satu skenario yang mungkin terjadi dalam tipografi responsif adalah peralihan antara variasi gaya dan skala subjudul. Misalnya, kita dapat mengatur berbagai gaya dengan ukuran yang sama untuk layar kecil — di mana satu unit konten terlihat dan sinyalnya jelas — dan menciptakan lebih banyak kontras dengan skala tipografi untuk layar yang lebih besar di mana terdapat lebih banyak noise.

Subjudul adalah peluang bagus untuk menambah gaya pada komposisi. Untuk daftar opsi yang lengkap, lihat artikel saya “Mengatur subjudul dengan CSS” dan halaman demo yang menyertainya.

Gambar 9.4 Enam gaya khas dengan variasi gaya font

Tabel Data

Tabel menjadi elemen yang sangat umum di Web, berguna untuk membandingkan data numerik, serta membuat koneksi antar array informasi. Oleh karena itu, hal ini hampir tidak dapat dihindari dalam aplikasi Web.

Untuk tabel yang bersih, kita dapat menggunakan properti border-collapse dan atribut spasi sel. Pertahankan ritme vertikal dengan menambahkan aturan margin bawah dan, jika perlu, ganti margin bawah dalam file CSS khusus titik henti sementara.

```
<table cellspacing="0" cellpadding="0" />

table {
       border-collapse: collapse;
       border-spacing: 0;
       margin-bottom: 1.5em;
}
```

Batas horizontal mudah dilihat dan akan menambah kejelasan, sehingga kita dapat membuat batas tersebut menggunakan properti border-bottom atau border-top. Selanjutnya, kami menetapkan beberapa padding agar data dapat bernapas, dengan tetap memperhatikan harmoni ritme vertikal. Untuk kolom mana pun yang berisi data numerik, kita dapat meratakan datanya ke kanan sehingga nilainya dapat dengan mudah dibaca dan dicocokkan satu sama lain tunggal di bawah tunggal, puluhan di bawah puluhan, dan seterusnya.

```
th, td {
      border-top: .1em solid #ccc;
      padding: .65em 1em .75em; /* .75em = 1.5em/2 */
      vertical-align: top;
}

.numeric {
      text-align: right;
      font-feature-settings: 'tnum';
}
```

Beberapa tipografi populer di Web, misalnya ProXima Nova dan Georgia tidak mendukung angka garis lebar tetap. Dalam hal ini, Anda mungkin ingin mempertimbangkan alternatif lain, misalnya Anonymous Pro80 oleh Mark Simonson. Ini gratis dan tersedia di layanan host font Web di dekat Anda, atau font lama yang aman untuk Web seperti Courier New, Trebuchet MS, atau Verdana.

Mengatur gambar tabel dengan font OpenType adalah pilihan terbaik untuk data tabel. Aktifkan dengan properti font-feature-settings. Aturan CSS lain yang akan berguna untuk data tabular masih dalam rancangan kerja adalah perataan berbasis karakter dalam kolom tabel.

Ketika karakter perataan ditentukan, karakter tersebut dipusatkan di sepanjang sumbu paralel kolom tunggal dan teks digeser sesuai dengan itu. Hal ini khususnya berguna untuk menata susunan angka dengan pemisah, misalnya angka desimal. Pada saat penulisan, aturan ini masih belum didukung di browser.

```
td { text-align: "." center }
```

Ketika sejumlah besar data yang tidak terduga memenuhi tabel (misalnya URL yang panjang), data tersebut dapat keluar dari area yang ditentukan dan merusak tata letak. Agar tabel tidak berantakan, kita dapat menggunakan aturan CSS table-layout: fixed dan mengatur lebar maksimum tabel, misalnya 100%. Ini harus menjaga tabel tetap berada di dalam elemen induk.

```
table {
     border-collapse: collapse;
     border-spacing: 0;
     table-layout: fixed;
     max-width: 100%;
}
```

TEKNIK LANJUTAN: KEPADATAN DAN KELAS PIXEL

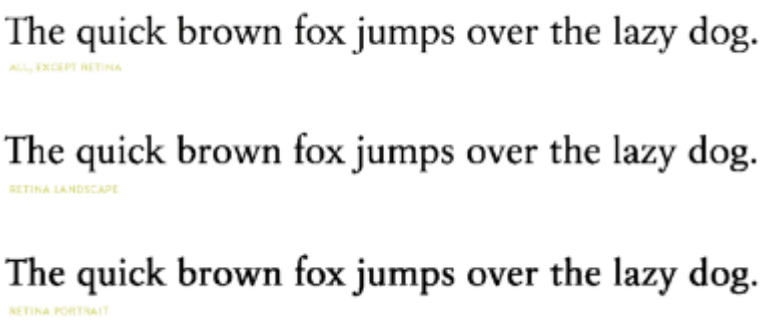
Kami telah melihat teknologi rendering yang berbeda di OS yang berbeda, namun produsen perangkat, yang dipimpin oleh Apple, memperkenalkan variabel lain kepada kami: resolusi layar. Kepadatan piksel mempengaruhi tampilan tipe pada perangkat tertentu. Ambil contoh, perbedaan nyata dalam tipe rendering pada iPhone dan iPad yang semuanya dibuat oleh pabrikan yang sama. Bayangkan apa yang terjadi bila Anda menambahkan lebih banyak kepadatan ke dalam campuran.

Sebagai masalah tambahan pada layar iOS Retina82 dan Windows ClearType83, ketika orientasi perangkat diubah, tumpukan subpiksel tidak lagi sejajar dengan baris teks, melainkan tegak lurus terhadapnya. Oleh karena itu, mesin rendering tidak dapat mengandalkan tata letak subpiksel untuk anti-aliasing subpiksel.

Saya telah memperkenalkan Anda pada topik samar anti-aliasing pada perangkat iOS, namun terlepas dari teori mana pun yang benar, masih ada masalah teks yang tampak kurang elegan dalam orientasi lanskap di iPhone dan iPad. Saya kira saat ini Anda mungkin ingin pergi ke balkon dan berteriak ke langit, jadi silakan lanjutkan saya tidak keberatan.

Merasa lebih baik? Bisakah kita melanjutkan?

Karena mendesain untuk layar sangat mirip dengan mendesain untuk surat kabar, banyak solusi terhadap permasalahan kita dapat ditemukan di dunia desain cetak. Dulu, perancang tipe menemukan font bertingkat, varian yang sedikit disesuaikan dari gaya tipe yang sama, yang membantu desainer buku dan surat kabar mengontrol kinerja tinta dengan lebih baik pada kualitas kertas yang berbeda.



Gambar 9.5 iA font bertingkat

Dalam desain cetak ada kertas kasar; dalam desain Web terdapat layar beresolusi rendah yang kasar, sehingga pendekatan yang sama dapat diterapkan. Desainer di Information Architects yang dipimpin oleh Oliver Reichenstein menggunakan font bertingkat85 untuk menormalkan tampilan teks pada resolusi yang berbeda. Mereka menyematkan tiga nilai berbeda:

- nilai default, non-Retina
- tingkat lanskap Retina
- dan tingkat potret Retina

Karena Anda melek CSS, kodenya cukup jelas:

```
@font-face {
     font-family: '-graded font';
     ...
}

@media only screen and (min-device-pixel-ratio: 1.5) {
@font-face {
     font-family: '+graded font';
     ...
  }
}
@media only screen and (min-device-pixel-ratio: 1.5)
and (orientation: portrait) {
      @font-face {
      font-family: '++graded font';
      ...
  }
}
```

Jarak Membaca, Ditinjau Kembali

Saat ini kami berasumsi bahwa jarak membaca didasarkan pada faktor bentuk perangkat. Produsen perangkat menentukan piksel referensi, membuat banyak asumsi

mengenai bagaimana perangkat akan digunakan pada jarak pembacaan yang seragam. Artinya, kita hanya membaca dari smartphone ketika digenggam dengan telapak tangan menghadap wajah, membaca dari tablet ketika digenggam dengan kedua tangan, dan selalu duduk tepat 70cm dari layar desktop kita.

Kita memerlukan sesuatu yang lebih baik, cara untuk mendeteksi hubungan fisik antara pengguna dan perangkat dan menyesuaikan tipografinya. Kita sudah mempunyai teknologi untuk mengukur jarak baca sehingga kita dapat dengan mudah mendeteksi dan menghitung seberapa jauh jarak benda dari perangkat dengan sensor yang sudah tersedia di perangkat.

FirefoX OS dan FirefoX untuk Android keduanya mendukung ProXimity API86, yang mendeteksi seberapa dekat perangkat dengan objek fisik lainnya, dengan mengakses sensor proXimity. Dalam implementasi FirefoX, saat ini ia bekerja pada jarak hingga 10cm, namun jika dapat diperluas untuk mengembalikan nilai akurat untuk jarak dalam jangkauan tangan, kita dapat menggunakan nilai tersebut untuk menambah atau mengurangi ukuran huruf dan menyesuaikan spasi paragraf. Bekerja dengan proXimity API sangatlah mudah. Antarmuka DeviceProximityEvent memberikan informasi tentang jarak antara perangkat dan objek terdekat, sedangkan UserProximityEvent saat ini mengembalikan nilai boolean jika pengguna dekat dengan perangkat.

```
window.ondeviceproximity = function (event) {
     var prox = "Proximity: " + event.value + " cm";
     prox += "Min. value supported: " + event.min + " cm";
     prox += "Max value supported: " + event.max + " cm";
     proximityDisplay.innerHTML = prox;
 };
window.onuserproximity = function (event) {
     var userProx = "User proximity - near: " + event.near;
     userProximityDisplay.innerHTML = userProx;
 };
```

Contoh lainnya adalah menangkap objek dengan kamera Web biasa. Dengan menggabungkan API getUserMedia dengan algoritma pengenalan wajah, kita dapat mendeteksi wajah pengguna dan menghitung jarak antara pengguna dan layar87.

Apa pun teknologi sensornya, setelah jarak baca tersedia, kita dapat menggunakannya sebagai kueri pelengkap untuk kueri @media ukuran layar untuk menghasilkan tipografi yang lebih baik. Beberapa kombinasi dan pengaturannya masing-masing terlintas dalam pikiran:

- Ponsel pintar dalam jarak telapak tangan (ukuran kecil, ukuran pendek, ujung depan rapat)
- Ponsel pintar dengan jarak pangkuan (ukuran kecil hingga sedang, ukuran pendek, terdepan sedang)
- Tablet pada jarak telapak tangan (ukuran kecil, ukuran sedang, depan normal)

- Tablet dengan jarak pangkuan (ukuran kecil hingga sedang, ukuran sedang, terdepan sedang)
- Tablet pada jarak meja (ukuran sedang, ukuran pendek, ujung depan rapat)
- Laptop dengan jarak pangkuan (ukuran kecil hingga sedang, ukuran sedang, terdepan sedang)
- Laptop di jarak meja (ukuran sedang, ukuran dan terdepan)
- Layar desktop pada jarak meja (ukuran sedang hingga besar, ukuran sedang, terdepan sedang)
- Layar desktop pada jarak dinding (ukuran besar, ukuran dan terdepan)

Kombinasi tipografi yang berbeda ini tidak boleh diterapkan secara real-time, karena dalam hal ini skala antarmuka akan naik dan turun tepat di depan pengguna. Jelas, itu sama sekali tidak berguna. Sebaliknya, perangkat dapat menerapkan pengaturan yang sedikit berbeda setiap kali pengguna memuat halaman, dan melakukan pengukuran terutama jika pengguna mencoba menyesuaikan kembali jarak membaca dengan menjauhkan atau mendekatkan perangkat ke wajahnya.

Semua data tersebut dapat dikumpulkan, disimpan, dan digunakan untuk menjalankan serangkaian pengujian multi-variasi di latar belakang, tanpa mengganggu pengguna. Statistik yang dikumpulkan dapat membantu kita menentukan dengan lebih baik pengaturan optimal pada setiap jarak pembacaan. Perangkat dapat mengenali setiap pengguna tertentu, mengkalibrasi ulang, dan menerapkan pengaturan tipografi optimal untuk mereka. Belum lagi raut wajah ahli kacamata saat Anda membuang semua statistik Anda ke mejanya. Siapa yang nerd sekarang, ya? Apakah kita mencapai puncaknya di Web? Saya kira tidak demikian!

Tidak Sesulit Itu

Anda akan setuju bahwa setiap teknik yang disajikan cukup sederhana jika diamati secara terpisah. Bagian rumit dalam tipografi Web adalah mengingat semua sumbu dan menciptakan pengaturan terbaik untuk situasi tertentu. Namun jika Anda memprioritaskan, Anda dapat mencapai hasil yang cukup baik dengan sedikit usaha.

Pertama, tandai teks dengan tanda baca, tanda hubung, tanda kutip, dan spasi yang benar. Kedua, jaga paragraf default dengan menyeimbangkan ukuran huruf, panjang garis, dan tinggi garis. Bertujuan untuk tekstur yang rata. Sekalipun terasa tidak menarik, penggunaan karakter dan spasi yang tepat akan menghasilkan keselarasan. Ketiga, buatlah hierarki yang tepat sehingga pembaca benar-benar dapat membaca sampai akhir. Terakhir, jika Anda punya cukup waktu, sesuaikan kehalusan gayanya anggap saja itu sebagai hiasan pada kue.

Bagaimana Anda merasa nyaman dalam mendesain tipografi? Sebagai latihan sehari-hari, gunakan satu font hitam atau putih dengan warna latar belakang dan cobalah untuk mengekspresikan suasana hati yang berbeda menggunakan ukuran variabel, ruang putih makro dan mikro, serta menyesuaikan susunan elemen. Latihan lainnya adalah mencoba strategi berbeda untuk menetapkan hierarki dan bereksperimen dengan furnitur tambahan, seperti keterangan atau tanda kutip hanya menggunakan satu jenis huruf. Dengan cara ini

Anda akan mempelajari kemampuan setiap keluarga tipe dan apa yang dapat dicapai dengan CSS.

Saya menantang Anda untuk belajar hidup tanpa perkembangan, kemewahan dan kecemerlangan, untuk merancang situs Web dan menjadikannya ringan dan mudah diakses baik dalam tampilan maupun kinerja, serta bersifat lintas platform. Tolak godaan penghitung dan duri yang indah, dan tunda penelusuran tipografi sampai Anda membangun fondasi yang kokoh.

Ingatlah selalu bahwa tipografi menyajikan konten dan kami melayani pengguna bukan sebaliknya. Isilah konten terlebih dahulu, karena kenyataan sebenarnya adalah bahwa tipografi Web bukan tentang memilih font, namun tentang membuat informasi dapat diakses, dibaca, dan dibaca. Kita semua memiliki kecenderungan bawaan untuk membuat segala sesuatunya menyenangkan mata, jadi setelah Anda belajar untuk mencakup hal-hal penting, secara naluriah Anda akan tahu cara menambahkan jumlah daya tarik yang tepat.

BAB 10
DUA SISI STRATEGI KONTEN

Pada tahun 2005, saya membuat blog wordpress saya sendiri. Itu tidak mudah. Hal ini terasa seperti pembelajaran yang sangat banyak pembelajaran yang saya lakukan pada waktu saya sendiri, mengandalkan siklus terus-menerus dalam mencoba dan gagal jalan menuju setiap kesuksesan kecil. Itu hampir tidak berhasil, tapi diluncurkan, dan saya mengambil tugas untuk menciptakan sesuatu yang menarik setiap hari.

Ini adalah pengalaman pertama saya dalam menerbitkan buku sendiri, di mana saya merebut kendali publikasi dari para penulis esai dan jurnalis yang saya kagumi, sehingga memaksa diri saya untuk tampil di hadapan publik. Saya melakukannya untuk berlatih menulis, mengasahnya seperti seseorang mengasah kapak, membelah semak belukar menuju ketenaran, kekayaan, dan mungkin Pulitzer, jika saya beruntung. Sebaliknya, saya akhirnya menemukan kembali diri saya sebagai seorang profesional Web, menggabungkan dua hal yang saya sukai: kata-kata dan teknologi.

Ini adalah pilihanku. Saya telah berpindah dari audiens ke editor. Saya sekarang berada di belakang mesin. Itu delapan tahun yang lalu. Sejak itu, kami melihat situs web menjadi lebih rumit. Kami telah meningkatkan metode kami. Kami berfokus pada arsitektur informasi yang berpusat pada pengguna, kemudian strategi konten yang berpusat pada pengguna, kemudian kembali fokus pada strategi multilayar yang berpusat pada pengguna, semuanya ada di mana saja.

Perhatian kami terhadap pengguna pada pemirsa dan pelanggan yang kami andalkan untuk membuat produk dan konten kami sukses membutuhkan waktu lama untuk menyatu. Tapi di sinilah kita! Zaman keemasan desain Web! Dimana konten dianggap serius! Dimana pengguna mendapatkan tempat duduk di meja! Dimana semuanya berupa pelangi dan kue coklat dan tidak ada kue kismis oatmeal yang terlihat!. Kecuali bagi mereka yang mungkin tidak memiliki keinginan yang sama untuk membongkar Web dan mendukungnya. Kecuali orang-orang yang mengambil barang-barang yang kita buat dan menjadikannya sebagai pekerjaan sehari-hari. Kecuali bagi para editor yang kesulitan dengan beban CMS baru, atau perubahan alur kerja yang sepele, atau gejolak politik yang datang dengan situs web baru.

Dalam perjalanannya, orang-orang ini melakukan gerakan yang sama seperti yang saya lakukan dari audiens hingga editor dan mereka semua mengambil bagian dalam pembuatan konten Web. Namun kami terlalu fokus untuk membuat situs web yang dapat digunakan oleh pelanggan sehingga kami lupa bahwa kami juga perlu memandu editor, atasan, dan rekan kerja melalui proses tersebut.

Ada dua strategi tatap muka konten: orang yang kami targetkan (pengguna kami), dan orang yang melakukan penargetan (editor kami). Kami bertanggung jawab untuk membuat situs web yang hebat. Namun kami juga bertanggung jawab untuk membuat situs web dapat digunakan dari sudut pandang editor. Kami adalah orang-orang yang membuat Web; kita jugalah yang bertanggung jawab untuk membantu mereka yang menopangnya.

Fokus Baru pada Manusia

Seperti yang saya temukan di blog WordPress pertama itu, orang-orang yang mengelola dan membuat konten dihadapkan pada lompatan kemajuan teknologi, sistem baru yang rumit, dan perubahan inheren dalam alur kerja mereka. Saat kami lebih memahami cara orang mengakses Web, kami menyadari bahwa metode yang kami gunakan saat ini tidak berfungsi sebaik yang kami harapkan sehingga kami terpaksa menyesuaikan cara kami melakukan sesuatu dan apa yang kami lakukan. Kami mengibarkan spanduk untuk konten terstruktur. Kami mengadopsi alat konten seperti GatherContent, atau alat editorial seperti Editorially. Kami melobi untuk lebih banyak ahli strategi konten yang berdedikasi dalam organisasi kami.

Kami mulai meminta perubahan. Dan melawan perubahan itu muncullah perlawanan. Ini bukanlah hal yang mudah untuk dilakukan. Ketika Chip dan Dan Heath, penulis Switch: How to Change Things When Change is Hard, mengatakan *"Perubahan bukanlah sebuah peristiwa; ini sebuah proses,"* mereka memperingatkan kita bahwa manusia tidak terprogram untuk berpindah arah dengan uang sepeser pun. Dalam pekerjaan kita, dalam tim kita, dalam situasi apa pun yang kita hadapi, nilai kita ditentukan oleh apa yang kita lakukan.

Bayangkan Anda bekerja dengan editor konten Web di sebuah universitas besar. Sebagai bagian dari proses desain Web, kami mendorong perubahan pada bagaimana halaman beranda ditata, atau bagaimana konten mengalir melalui sistem, atau CMS apa yang digunakan. Kami juga dapat menyarankan teknologi canggih seperti personalisasi, dan kami meminta editor konten untuk mempelajari taksonomi baru secara mendalam sehingga mereka dapat secara efisien menarik blok konten dari seluruh situs web.

Kami tidak hanya mendesain ulang situs web. Kami juga membangun kembali proses mereka, yang pada gilirannya mengubah peran mereka di universitas. Perubahan berarti lebih dari sekedar adaptasi sederhana tentang cara kita membuat spreadsheet atau apa artinya ketika kita mengatakan "*audit konten*". Ini adalah metamorfosis skala penuh dalam arti kita terhadap rekan kerja, atasan, dan mitra kita.

Jonathan Kahn, kepala sekolah di Together London, mengatakan "*Kami punya banyak ide, kami bekerja dengan orang-orang terampil, dan alat kami menjadi lebih baik setiap hari tetapi sampai kami mulai mengubah budaya organisasi, kami tidak akan mencapai tujuan kami.* " Hal ini berarti mengubah ekspektasi kami terhadap konten yang sebenarnya berfungsi dan siapa yang bertanggung jawab. Karena kita semua yang bertanggung jawab sekarang.

Empati

Konten yang baik membutuhkan orang-orang yang berdaya dan terlibat. Konten yang baik membutuhkan orang untuk membacanya, dan membutuhkan orang untuk membuatnya. Tanpa manusia, tidak ada konten. Strategi konten, pada intinya, adalah strategi manusia. Namun kami masih melihat pekerjaan konten dibuang ke back office, diserahkan ke TI atau copywriter atau departemen pemasaran (terima kasih kepada orang-orang yang sudah memiliki banyak hal untuk dilakukan). Pengalihan tanggung jawab seperti ini merupakan tanda pasti bahwa disiplin kita masih berjuang untuk mencapai kedewasaan. Kita masih menghabiskan terlalu banyak waktu untuk mencari metode dan alur kerja yang tepat, dan

tidak cukup waktu untuk memikirkan bagaimana perubahan besar dalam konten Web akan mempengaruhi tidak hanya pekerjaan mereka yang bertugas membuat konten, namun juga setiap pekerjaan lain di perusahaan.

Jadi, kita yang membantu membuat atau mendesain ulang situs web harus mempertimbangkan kebutuhan audiens situs web kita dan orang-orang yang akan membuat konten, semuanya artinya: semua orang. Untuk melakukan hal tersebut, kita harus menempatkan diri kita pada posisi mereka dan bekerja dengan empati terhadap situasi mereka.

Kami menyerahkan kebutuhan konten kami kepada editor dengan harapan semuanya akan sempurna, namun setiap editor bertanggung jawab atas tugas tambahan yang mungkin tidak melibatkan pembuatan atau pemeliharaan konten Web, tugas yang memerlukan waktu dan perhatian. Setiap kebutuhan konten baru setiap penambahan alur kerja konten, dan setiap sistem analitik baru, setiap platform CMS baru menghabiskan waktu dan perhatian tersebut.

Dua Jalan Menyatu

Sederhananya, orang-orang yang terlibat dengan situs web dapat dibagi menjadi dua kelompok: mereka yang akan menggunakan situs web; dan mereka yang akan memelihara situs web tersebut. Di satu sisi, Anda memiliki pengguna, audiens, dan pelanggan yang kami perlukan untuk membuat perusahaan atau konten kami sukses. Tujuan kami dengan grup ini sederhana: memberikan nilai dan menyampaikan pesan. Di sisi lain, Anda memiliki editor, orang-orang yang membuat konten ini, menciptakan pengalaman dan cerita yang sesuai dengan pengguna situs web. Pengguna meminta informasi dan produk. Redaksi mohon perhatian dan kesabarannya.

Berkat metode periklanan tradisional yang mengungkapkan secara besar-besaran, kami telah dilatih untuk berpikir bahwa ada semacam pergulatan terus-menerus antara editor dan pengguna; bahwa kepekaan artistik dan pandangan pribadi seorang editor terhambat oleh data, mesin pencari, dan umpan balik pengguna, sementara pengguna mengabaikan apa pun yang tidak memiliki relevansi langsung dan hanya merespons hal-hal besar, berani, dan bodoh.

Editor kami berhak mendapatkan rasa hormat yang lebih, begitu pula pengguna kami. Kenyataannya, kedua kelompok tersebut dipisahkan hanya oleh beberapa baris kode dan koneksi Internet. Mereka memainkan peran yang saling melengkapi dalam proses pembuatan Web penawaran dan permintaan, pembuatan dan konsumsi mewakili dua jalur yang, bukannya menyimpang, bergerak ke arah yang sama dengan pertukaran yang konstan.

Saat kami membuat website, kami melalui tahapan sebagai berikut:

- Discovery: Untuk siapa situs ini dan mengapa mereka mengunjunginya?
- Strategi: Bagaimana kami mengarahkan pengguna mencapai tujuan mereka?
- Eksekusi: Apa yang kami katakan kepada pengguna kami?
- Tata Kelola: Bagaimana kami menjaga situs ini tetap relevan?

Metodologi di bagian editorial mengikuti jalur yang sama:

- Discovery: Siapa yang membuat konten untuk situs web ini dan mengapa?

- Strategi: Bagaimana kami membantu perusahaan dan editor mencapai tujuan mereka?
- Eksekusi: Apa yang dikatakan editor kami dan bagaimana kami membantu mereka menyampaikannya?
- Tata Kelola: Bagaimana kita tetap berada pada jalur yang benar setelah peluncuran?

Beda tugas, beda tujuan, tapi arahnya sama. Tahap penemuan berkaitan dengan siapa dan mengapa. Tahap strategi berkaitan dengan bagaimana, sedangkan tahap eksekusi berkaitan dengan apa. Terakhir, tata kelola memberi kita lebih banyak wawasan tentang di mana dan kapan dengan memberikan seperangkat alat editorial dan pedoman yang harus kita patuhi setelah situs web pertama aktif.

Dengan dua jalur ini, kami dapat mulai mencari cara untuk menjangkau audiens kami dan memberikan apa yang mereka butuhkan, sambil tetap mengingat permasalahan yang mungkin muncul di sisi lain dari proses konten. Mari kita lihat di mana kedua jalur ini paling sering bertemu dalam beberapa langkah dasar strategi konten.

10.1 PENEMUAN: SIAPA DAN MENGAPA?

Konten yang sukses bergantung pada tingkat pemahaman yang mendalam terhadap komposisi audiens dan tujuan pengguna. Mengembangkan konten yang sukses memerlukan tingkat pemahaman tambahan tentang cara kerja organisasi, siapa yang dapat dianggap sebagai juara konten, dan bentuk apa yang nantinya akan diambil oleh jaringan editor konten kami. Hal ini dilakukan dengan mengajukan banyak pertanyaan dan melakukan banyak penyelidikan selama pertemuan penemuan awal.

Sebelum pekerjaan konten sebenarnya dapat dimulai, kami berdiskusi dengan semua pihak terkait dan menentukan audiens serta hasil yang dirasakan: siapa yang akan mengunjungi situs web kami dan apa yang mereka harapkan dari situs tersebut. Proses “audiens dan hasil” ini memberi kita kesempatan bagus untuk mengajukan beberapa pertanyaan editorial juga.

Proses penemuan adalah tahap paling penting dalam membangun hubungan empati dengan tim editorial Web. Ini penting untuk dukungan editorial. Hal ini diperlukan untuk keterlibatan yang efektif. Dan itu membantu membangun perasaan hangat dan tidak jelas sepanjang proyek. Ini semua tentang mencari tahu siapa yang akan menggunakan situs web di kedua sisi proses.

Audiens dan Hasil

Pertama-tama, mari kita selesaikan hal ini: kita bukanlah penontonnya. Kita mungkin adalah penumpang setia penerbangan, namun bukan berarti kita bisa mewakili setiap penumpang maskapai penerbangan. Kita mungkin punya anak, tapi bukan berarti kita bisa mewakili semua orang tua. Kita mungkin penggemar berat album terbaru Daft Punk, namun bukan berarti kita bisa berbicara dengan banyak orang yang juga menikmati album baru Daft Punk. Jadi kita harus mencari dan berbicara dengan orang-orang yang dapat berbicara mewakili khalayak ini: khalayak itu sendiri.

Kami mulai dengan mengumpulkan semua orang untuk pertemuan besar pengumpulan informasi. Rapatnya mungkin memakan waktu 30 menit, bisa juga memakan

waktu beberapa jam. Yang penting adalah hal ini terjadi dan melibatkan orang-orang yang tepat. Ini berarti Anda perlu mencari seseorang dari setiap bidang organisasi yang relevan. Daftar tersebut mungkin mencakup, namun tidak terbatas pada, perwakilan dari:

- Pemasaran
- Kreatif/Web
- DIA
- Pengembangan produk
- Layanan Garis Depan
- Dewan Eksekutif
- Pengembangan Bisnis

Kisaran peserta sengaja dibuat luas. Setiap area dalam perusahaan memiliki sewa konten Web yang berbeda, dan semuanya bekerja secara berbeda untuk mewujudkan konten Web.

Anggota TI dapat membicarakan kebutuhan dan keterbatasan mereka, terutama yang berkaitan dengan alur kerja konten, sementara seseorang dari bagian pengembangan produk atau penjualan dapat mendiskusikan gumaman yang mereka dengar di luar perusahaan. Layanan garis depan adalah kunci untuk menangkap pemikiran pengguna atau pelanggan yang tidak puas, sementara dewan eksekutif akan menjelaskan potensi strategi bisnis dan akan menghargai menjadi bagian dari proyek ini.

Suara yang berbeda menghasilkan solusi yang berbeda. Jika pertemuan penemuan awal Anda hanya melibatkan beberapa anggota dari bagian pemasaran, Anda akan mendapat masalah. Dengan adanya kelompok ini, inilah saatnya untuk memperkuat audiens kita dan hasil yang diharapkan oleh audiens tersebut. Buat dua daftar dan mulailah mengurutkan semua audiens dan semua hasil yang dirasakan ke dalam dua kolom. Kemudian, mulailah mengajukan pertanyaan. Misalnya:

- ❖ Menurut Anda, siapa audiens situs web Anda?
- ❖ Audiens manakah yang kurang terwakili dengan baik, dan apa yang mereka cari?
- ❖ Siapa lagi yang bersaing untuk mendapatkan perhatian audiens Anda?
- ❖ Apa yang mendorong bisnis Anda, dan bagaimana audiens Anda membantu mencapai hasil positif?
- ❖ Kelompok audiens manakah yang menjadi bagian dari inisiatif strategis? Apakah ada audiens sekunder yang ingin diinvestasikan oleh perusahaan?
- ❖ Bayangkan kita menjalani proses ini di sebuah universitas kecil.
- ❖ Saat Anda memberikan jawaban, daftar audiens Anda mungkin mulai terlihat seperti ini:
- ❖ Siswa potensial
- ❖ Siswa saat ini
- ❖ Orang tua
- ❖ Alumni
- ❖ Penggemar atletik

Dan hasilnya mungkin:

- Mendaftar untuk tur universitas

- Temukan informasi tentang cara mendaftar ke sekolah
- Pelajari lebih lanjut tentang asrama
- Ajukan permohonan bantuan keuangan
- Membaca berita tentang proyek universitas yang didanai oleh alumni
- Membaca skor tim sepak bola

Audiens ini akan semakin dipisahkan berdasarkan kebutuhan – lagipula, calon siswa bisa saja berasal dari non-tradisional, lulusan, online, atau pasca-sekolah menengah – dan hasilnya akan tumpang tindih dengan audiens tersebut. Anda akan mendapatkan daftar audiens yang panjang, daftar hasil yang panjang…dan Anda juga akan mengumpulkan banyak informasi yang tidak sesuai. Karena ketika sekelompok editor dan profesional pemasaran mulai berbicara tentang pelanggan dan konten mereka, mereka juga mulai berbicara tentang kegagalan sistem mereka saat ini, perjuangan yang mereka hadapi dengan politik internal, dan inisiatif yang mereka dorong untuk diterapkan. bertahun-tahun.

Saatnya berbicara tentang proses editorial. (Untuk melihat lebih dalam mengenai khalayak, hasil dan kepribadian, bacalah artikel saya untuk A List Apart, atau bacalah buku yang menginspirasi artikel ini: Online and On Mission, oleh C. David Gammel. Untuk informasi dasar yang bagus tentang bagaimana, kapan dan apa yang harus ditanyakan selama wawancara penemuan dan wawancara audiens, baca Mewawancarai Pengguna: Cara Mengungkap Wawasan Menarik oleh Steve Portigal.)

Ketika Editor Menjadi Audiens

Ingatkah saat saya menulis, “kita bukan penontonnya?” Saya mengambilnya kembali. Karena pada titik inilah ketika penelitian audiens pengguna berubah menjadi keluhan alur kerja hal-hal menyenangkan mulai muncul ke permukaan: audiens editorial.

Editor mendapatkan konten dari seluruh penjuru organisasi. Mereka mungkin diwakili oleh satu orang, atau mungkin staf suatu departemen. Mereka bisa jadi CEO, atau bisa jadi segelintir pekerja magang. Editor berbeda-beda tergantung pada organisasi, industri, dan situs web, yang menjadikan mereka sama pentingnya dengan audiens situs web.

Meskipun kita telah mengumpulkan kelompok untuk pertemuan penemuan awal, sekaranglah waktunya untuk mengajukan beberapa pertanyaan sulit. Bicara tentang siloing departemen, tentang alur kerja yang salah, dan tentang seseorang yang kesulitan memahami editor WYSIWYG. Bicarakan segalanya, karena ini adalah kesempatan terbaik Anda untuk membuat konten dan arsitektur yang ramah editor dan pengguna sebelum terlambat.

Percayalah kepadaku. Mereka akan berbicara, jika diberi kesempatan. Hal ini karena pekerjaan konten memerlukan tingkat ego tertentu; itu adalah menulis dan berkreasi, meskipun itu hanya FAQ dasar atau siaran pers. Ketika proses yang salah atau frustrasi staf menghasilkan konten yang kurang bagus, mereka yang bertanggung jawab untuk menempatkan konten tersebut di Web akan tersinggung.

Jadi mulailah menghilangkan gangguan dan selami lebih dalam. Apa yang menyebabkan gangguan ini? Apa yang dapat kita lakukan untuk mengatasi rasa frustrasi ini? Jika seseorang di bagian pemasaran menyebutkan bahwa mereka tidak dapat melakukan semua perubahan yang mereka inginkan karena jumlah editor yang ada tidak mencukupi,

tanyakan keterampilan apa yang mereka lewatkan dan coba cari tahu apakah ada cara untuk menyebarkan perubahan tersebut. bekerja di sekitar. Jika departemen tertentu kesulitan mendapatkan konten di situs web, tanyakan tentang proses departemen tersebut dan apakah masalahnya adalah masalah sumber daya atau sekadar sikap apatis.

Berbekal informasi ini, kita dapat membuat daftar audiens dan hasil serupa untuk alur kerja editorial. Misalnya, sebuah universitas mungkin melihat audiens editorial berikut:

- Direktur departemen
- Komite pengarah web
- Staf dan dosen
- Pemasaran
- Penerimaan sarjana
- Staf redaksi

Masing-masing kelompok internal memiliki peran berbeda dalam proses konten Web. Dengan mengidentifikasi semua pihak yang terlibat, kami bisa mendapatkan pemahaman yang lebih baik mengenai potensi kendala yang ada, dan cara mengatasinya sebelum menjadi konflik konten internal yang lebih besar.

Setelah kami menentukan semua audiens editorial, terserah kepada kami untuk memastikan masing-masing terwakili, baik dalam rencana konten strategis maupun sebagai bagian dari diskusi situs web secara keseluruhan. Kami tahu, kami tahu: komite sulit untuk diajak bekerja sama. Namun berurusan dengan komite lebih mudah daripada harus berjuang keras untuk membuat perubahan bagi departemen atau pemangku kepentingan yang sudah terlambat terlibat dalam proyek. Mike Monteiro, penulis Design is a Job, berbicara tentang perlunya merekrut orang-orang yang tepat:

Saat memasuki sebuah proyek, Anda perlu mengetahui siapa di sisi klien yang memberikan masukan, siapa yang memberikan umpan balik, dan siapa yang menyetujui. Anda mungkin memiliki gagasan yang lebih baik tentang siapa orang-orang ini seharusnya daripada klien Anda. Meskipun penting untuk memiliki kelompok kecil yang memberikan umpan balik untuk meninjau keputusan desain, yang lebih penting lagi adalah bahwa kelompok tersebut adalah kelompok yang tepat.

Masing-masing kelompok editorial Anda memainkan peran besar dalam menentukan alur kerja editorial dan keberlanjutan proyek. Dengan mencari tahu apa yang membuat masing-masing kelompok tergerak, kami dapat memberikan solusi yang sesuai dengan harapan proyek.

10.2 MENGAUDIT PROSES YANG ADA

Dengan mempertimbangkan audiens kita baik eksternal maupun editorial, inilah saatnya untuk mencari tahu apa yang harus kita kerjakan. Ini merupakan latihan dalam alokasi sumber daya, namun juga merupakan ujian besar dalam hal kesabaran.

Kami sudah terbiasa terjun ke dalam proyek yang dilengkapi dengan spreadsheet dan mendengarkan musik selama beberapa jam, siap menginventarisasi setiap halaman PDF dan kebijakan terakhir. Kami melakukan ini untuk mengetahui di mana kami berada. Area mana di

situs web yang kurang terwakili? Konten apa yang tidak digunakan dan mengacaukan hasil pencarian? Mengapa kami masih mengaitkannya dengan program Program Natal Tahunan 2002?

Tujuan dari inventarisasi ini adalah untuk:

- Menentukan arah website saat ini, serta mendapatkan pemahaman yang baik tentang ruang lingkup proyek.
- Dapatkan paparan terhadap suara dan nada konten situs web.
- Temukan hubungan antara halaman dan arsitektur informasi situs web saat ini.

Namun hal ini hanya berfokus pada khalayak eksternal: bagaimana orang menemukan sesuatu dan apakah hal tersebut relevan. Kita juga perlu mencari tahu bagaimana konten muncul di situs web, di mana konten tersebut muncul, kapan konten tersebut muncul, dan bagaimana konten tersebut muncul.

Artinya, selain kolom inventaris standar seperti “ID”, “Nama Halaman”, dan “Jenis Konten”, kita juga harus mulai menentukan “Pemilik Konten”, “Ahli di Tempat”, dan apakah suatu halaman bersifat statis atau tidak. berubah. Dengan melakukan hal ini, kita akan membuat alur kerja editorial de facto, sebuah alur kerja yang memerlukan elemen pengeditannya sendiri juga.

Dalam strategi konten, kita sering berbicara tentang menghilangkan konten ROT (berlebihan, ketinggalan jaman, sepele). Alur kerja juga memerlukan hal ini. Saat kita menelusuri konten situs web, kita juga harus melihat area di mana alur kerja menjadi mubazir, ketinggalan jaman, atau sepele. Beberapa contohnya mungkin:

- Mengirimkan salinan yang tidak mengikat secara hukum ke departemen hukum untuk disetujui,
- Menjalankan semua konten melalui supervisor departemen alih-alih memberdayakan tim konten untuk melakukan pengeditan sendiri,
- Fokus pada produksi konten berbasis waktu dibandingkan relevansi — dengan kata lain, memposting demi postingan,
- Kurangnya tata kelola editorial dalam meninjau konten lama atau mengarsipkan konten kedaluwarsa.

Audit alur kerja ini (seperti audit konten) akan menyingkirkan bagian-bagian buruk dan membantu kami membuat keputusan strategis yang baik.

Strategi: Bagaimana

Perencanaan strategis sering kali ditangani dari sudut pandang pengguna (Apa tujuan audiens kami? Bagaimana kami dapat membantu mencapainya? Konten apa yang kami perlukan?); itu baik-baik saja, hanya saja kita perlu berbuat lebih banyak untuk menjawab pertanyaan-pertanyaan tersebut. tions di bawah permukaan. Selain bagaimana mencapai tujuan, kita perlu menentukan siapa yang dapat membuat konten. Selain mencari tahu konten apa yang kami perlukan, kami juga perlu menentukan siapa yang dapat memelihara konten tersebut.

Melissa Rach dan Kristina Halvorson mendefinisikan strategi sebagai “sebuah ide yang menentukan arah masa depan.” Strategi bukanlah sebuah dokumen; itulah ide di balik

dokumen tersebut. Artinya, dokumen mencakup lebih dari sekedar dokumen – dokumen terdiri dari budaya, tenaga kerja, peralatan, dan yang paling penting, manusia. Oleh karena itu, rencana strategis kita harus mencakup kedua hal tersebut.

Di Blend, rencana konten strategis kami menetapkan konsep tingkat tinggi berikut:

- Pengulangan audiens dan tujuan situs web
- Analisis permasalahan konten terkini dan usulan perubahan
- Analisis kesenjangan kebutuhan konten dan usulan penambahan
- Rencana tindakan untuk pesan situs web yang diusulkan
- Sebuah rencana aksi untuk struktur situs web yang diusulkan
- Rencana aksi untuk usulan tata kelola situs web
- Pengukuran kesuksesan

Anda akan melihat bahwa selain rencana tata kelola situs web yang diusulkan, hampir semua item tersebut merupakan strategi yang dihadapi pengguna. Namun, ada tujuan tersembunyi dari semua analisis dan strategi ini: kami mengeluarkan ide-ide ini tidak hanya untuk melihat apakah ide-ide tersebut dapat digunakan oleh pengguna situs web, tetapi juga untuk melihat apakah ide-ide tersebut akan diterima oleh editor.

Setiap paket perpesanan dilengkapi dengan saran alur kerja: temukan seseorang untuk menjadi penjaga panduan gaya; atau menentukan satu orang yang dapat menjalankan pengujian A/B pada konten situs web. Saran struktural mencakup masalah alur kerja editorial dari seluruh departemen; misalnya, jika kita membuat feed berita, apakah ada yang akan memeliharanya?

Masalah tata kelola dan alur kerja inilah yang menimbulkan hambatan terbesar dalam proses ini. Ingat: orang tidak menyukai perubahan. Dan ini sering kali merupakan tahap proyek di mana kami mulai menyarankan perubahan pada pekerjaan mereka saat ini.

Tugas kita adalah memfasilitasi perubahan tersebut. Uraikan semua hasil yang diharapkan dan tunjukkan perubahan apa yang diperlukan. Temui secara pribadi orang-orang yang mungkin harus mengambil tugas tambahan dan jelaskan manfaat dari tugas tersebut. Bekerja samalah dengan para manajer untuk memastikan mereka memahami jumlah kerja ekstra atau perubahan keahlian yang diperlukan untuk setiap inisiatif baru. Namun yang terpenting, ajak orang-orang untuk bergabung lebih awal.

10.3 MENENTUKAN PERAN

Siapa yang harus terlibat dengan konten Web? Setiap orang. Siapa yang bisa terlibat? Beberapa orang terpilih yang memegang kata sandi dan memiliki waktu untuk mengerjakan situs web baru. Siapa yang akan terlibat? Itu adalah subjek yang bisa menjadi sederhana dan rumit pada saat yang bersamaan. Itulah inti dari tim konten interdisipliner.

Kami berbicara tentang tim konten interdisipliner karena, pada intinya, setiap kelompok pekerja konten yang dikumpulkan dalam beberapa hal bersifat interdisipliner. Mereka memanfaatkan pengalaman mereka sendiri, latar belakang mereka sendiri, dan pemikiran mereka sendiri untuk menciptakan sistem konten yang kompleks untuk situs web kami. Mereka tidak selalu akur, tetapi mereka menggabungkan upaya mereka untuk

menciptakan perpaduan yang lebih baik daripada yang bisa dilakukan oleh masing-masing bagian.

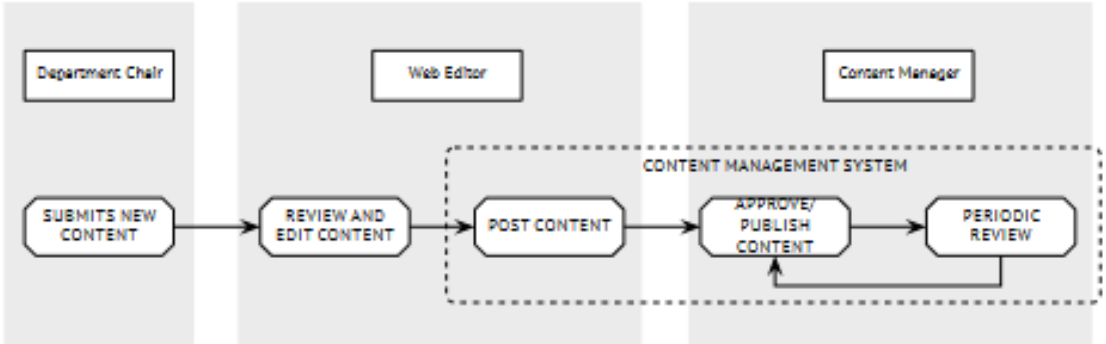
Ada konsep yang disebut kesederhanaan informasi yang berperan saat membangun dan menetapkan peran dalam tim konten. Matthew Frederick, dalam bukunya 101 Things I Learned in Architecture School, mengatakan kesederhanaan yang diinformasikan adalah "didasarkan pada kemampuan untuk membedakan atau menciptakan pola-pola yang memperjelas dalam campuran yang kompleks." Dia menyamakannya dengan pengenalan pola, namun dengan twist, dimana kita bergerak melewati banyaknya konflik kepentingan dan mencari tempat di mana mereka dapat bersatu secara harmonis.

Jika ini terdengar seperti kesulitan yang kami lalui dalam upaya untuk membuat semua pengguna situs web kami senang, Anda tidak salah. Sama seperti kita menghabiskan waktu berjam-jam mengatur dan menguji serta mengatur ulang konten di situs web kita untuk memberikan tampilan tercepat kepada semua audiens terkait, kita juga harus, melalui trial and error, menentukan siapa yang akan bekerja paling baik dalam tim konten organisasi.

Anda mulai dengan mengembalikan kelompok Anda dari sesi penemuan awal. Satu demi satu, pelajari setiap bagian dari rencana strategis yang diusulkan. Ajukan pertanyaan tentang kepemilikan dan kemampuan. Siapa yang sudah melakukan ini? Siapa lagi yang bisa melakukannya? Berapa banyak orang yang kita miliki yang mengerjakan proyek ini? Berapa jam yang bisa kita ambil dari departemen lain? Di manakah peluang bagi kita untuk menyerahkan konten kepada ahli materi pelajaran untuk menghemat waktu?

Dengan pertanyaan-pertanyaan ini, Anda dapat membuat rencana alur kerja awal. Ini bisa sesederhana dokumen Word yang berisi tanggung jawab setiap orang berdasarkan posisi, atau rumit seperti spreadsheet dengan jam dan tugas yang tepat. Namun, sederhana lebih baik. Step Two Designs, yang telah melakukan penelitian ekstensif dalam mengembangkan tim konten intranet dan mengevaluasi alur kerja, memperingatkan terhadap proses alur kerja yang terlalu rumit. James Robertson, direktur pelaksana Step Two Designs, menulis bahwa "alur kerja yang sederhana dapat bermanfaat, dengan satu atau dua langkah antara penulis dan halaman yang diterbitkan… Namun, lebih dari itu, alur kerja dapat terbukti tidak efektif atau bahkan bermasalah."

Kami biasanya mengembangkan bagan yang menunjukkan aliran konten relatif, seperti yang ditampilkan di bawah.



Gambar 10.1 Aliran konten relatif

Kemudian, kami bersiap untuk semuanya berubah. Alur kerja harus selalu berulang. Itu tergantung pada kemampuan dan kepegawaian saat ini. Ada kebutuhan untuk revisi terus-menerus, dan revisi ini harus dimasukkan ke dalam proses. Apakah konten dibuat menggunakan metode Agile atau Lean, atau proses lain yang sudah ada atau dibuat, konten tersebut harus ditinjau secara berkala. Periksa mesin apakah ada kebocoran. Sesuaikan. Gunakan kembali bagian-bagian yang tidak berfungsi. Dan ketika keadaan menjadi sulit, mulailah memprioritaskan tugas. Artinya, jika Anda tidak membuat prioritas sejak awal.

Jangan Coba Melakukan Semuanya

Mari kita perjelas dari awal: tidak semua klien atau tim proyek mampu menangani setiap bagian proyek. Menetapkan peran dan menentukan strategi mungkin memerlukan lebih dari sekedar pengetahuan tentang konten dan beberapa hasil pengguna. Ini membutuhkan pemahaman keahlian setiap pengguna. Dan hal ini membutuhkan pemahaman kapan harus mengatakan cukup sudah cukup.

Percaya atau tidak, kami tidak perlu menggunakan metodologi lengkap kami di setiap proyek. Pertama kali saya menyadari hal ini adalah ketika saya bekerja dengan sebuah organisasi nirlaba kecil yang didanai sepenuhnya dari sumbangan. Mereka tidak punya uang untuk sebuah proyek, namun saya masih langsung terjun dengan paket lengkap: inventaris konten lengkap, audit penuh dan rencana strategis, gambar rangka terperinci dan peta situs web, kalender editorial dengan pembaruan mingguan. Kami telah sepakat untuk menyumbangkan waktu kami, dan saya sangat yakin dengan organisasi ini, jadi mengapa saya tidak memberikan segalanya?

Inilah alasannya. Aku sudah melampaui batas mereka. Bukan karena mereka tidak memahami rencananya; mereka tidak mengerti mengapa saya menghabiskan begitu banyak waktu untuk itu. Situs web tersebut memiliki total 20 halaman, dan mereka memiliki sekelompok sukarelawan yang menjalankan organisasi tersebut. Mereka tidak memerlukan rencana konten yang mendetail mereka hanya membutuhkan wawasan dan arahan tentang cara menjangkau audiens mereka. Mereka tidak memerlukan kalender konten mingguan mereka hampir tidak memiliki cukup staf untuk menghadiri rapat mingguan. Mereka adalah kelompok nirlaba. Mereka adalah teman. Mereka menganggap kontribusi saya sebagai ocehan berlebihan dari seorang fanatik konten.

Tapi bagaimana jika ini adalah klien yang membayar? Bagaimana jika ini adalah seseorang yang setuju untuk menghabiskan 50% anggaran pemasarannya untuk situs web baru? Bagaimana jika Anda menyampaikan hasil yang tidak ada gunanya karena menurut Anda itulah yang mereka butuhkan?

Hal ini juga akan terjadi di kepala mereka segera setelah kemarahan mereda. Karena Anda tidak hanya membuang-buang waktu dan uang mereka, pada saat itu: Anda secara tidak sadar menyabotase tim internal mereka dengan menuntut lebih banyak dukungan proyek daripada yang bisa mereka tangani. Kita semua mempunyai metodologi, namun tidak ada satupun yang cocok untuk semua. Mereka dapat diperluas dan dikontrak, dapat dipindahkan dan dilewati. Mereka harus dirancang untuk bekerja sebagian, untuk diperkecil, dan disesuaikan dengan kebutuhan proyek saat ini.

Berikan seseorang setiap alat yang ada di kotak peralatannya, dan mereka akan menghabiskan sebagian besar waktunya untuk mencoba mencari tahu obeng mana yang harus digunakan. Peralatan kami akan berbeda-beda pada setiap proyek, dan itulah hal yang harus menjadi fokus kami perlu memahami dan berempati dengan situasi yang ada agar dapat memberikan lebih dari sekedar solusi cookie cutter dari metodologi kami.

Eksekusi: Apa

Dengan adanya rencana strategis, kini kami beralih ke pembuatan konten situs web aktual, sebuah tugas yang biasanya ditangani dalam waktu yang terasa seperti seribu hari dengan masukan dari 17 juta orang. Pembuatan konten (kata-kata, video, organisasi, hubungan, sidebar, tutorial apa pun itu) terbentuk kembali pada fase penemuan, ketika kita mulai berbicara tentang pesan situs web, standar merek, dan konten yang ada. Tapi sekarang kami siap untuk lebih spesifik dan mengubah pesan dan hasil kami menjadi kata-kata nyata di halaman. Kecuali, siapa yang akan membuat konten itu? Editor situs web Anda. Itu siapa. Dan jika Anda belum pernah cukup beruntung untuk bekerja dengan migrasi konten 10.000 halaman, atau jika Anda berhasil menghindari proyek perubahan konten yang lengkap, saya akan memberi tahu Anda sebuah rahasia kecil: eksekusi adalah berantakan, penuh tekanan, dan benar-benar membebani.

Halaman demi halaman. Kalimat demi kalimat. Setiap gambar memerlukan tag yang tepat, dan setiap video memerlukan penyematan yang tepat, dan haruskah judulnya ditempatkan di sini atau di sana?

Pembuatan konten bisa menjadi tindakan yang sulit seperti istilah itu sendiri, terperosok dalam politik dan revisi serta detail terkecil, berulang kali. Tapi masalahnya ada pada detailnya, seperti yang mereka katakan. Dan itulah yang perlu kita fokuskan pada proyek konten situs web apa pun: detailnya. Detailnya membantu kami menarik konten di seluruh situs web melalui metadata, dan memberikan informasi dalam mikrokopi dan keputusan interaksi kecil selama pengujian, dan bahkan mungkin membantu kami menyadari ejaan yang tidak menguntungkan dari “departemen hubungan masyarakat” kami.

Dan meskipun kami belum tentu bertanggung jawab atas setiap detail proyek bagaimanapun juga, kami melakukan hal ini untuk memberdayakan editor situs web agar dapat membuat situs web yang dapat digunakan lama setelah diluncurkan kami bertanggung jawab untuk menciptakan lingkungan yang mendukung kehati-hatian dan kegunaan. konten, dari konten halaman beranda penting hingga mikrokopi terkecil. Berikut beberapa cara yang dapat kami bantu:

Tempatkan seseorang sebagai penanggung jawab

Setiap proyek membutuhkan seorang pemimpin yang dapat mengatur dan mendelegasikan tugas. Pemimpin ini dapat mengambil dua bentuk: pemimpin alami yang terhubung dengan proyek yang memahaminya dan dapat bekerja dengan berbagai departemen (pikirkan: direktur pemasaran, pemimpin editor konten, dll.); atau seseorang dari luar proyek yang dapat menjembatani kesenjangan yang tercipta melalui silo dan politik (pikirkan: konsultan strategi konten).

Ini tampaknya sangat sederhana, bukan? Namun kami masih kesulitan menghadapi tim yang memiliki terlalu banyak juru masak di dapur dan terlalu banyak sumber konten yang mengklaim prioritas. Tidak ada yang lebih menggagalkan proses ini selain pekerjaan yang tidak perlu, jadi pastikan seseorang bertanggung jawab untuk menentukan dari mana konten berasal, ke mana perginya, dan (yang paling penting) apa yang penting.

Andalkan ahli materi pelajaran

Saya tidak tahu apa pun tentang mesin flywheel atau chip komputer atau kue mangkuk atau apa pun yang sedang kami buat situs webnya minggu ini. Tapi itu bukan tugas saya sebagai ahli strategi konten. Tugas saya adalah membantu mereka yang mengetahui tentang mesin roda gila, chip komputer, atau kue mangkuk membuat konten yang dapat digunakan.

Ingat: kami tidak bertanggung jawab atas pengetahuan dalam domain tersebut. Kami bertanggung jawab mengkomunikasikan pengetahuan tersebut dengan cara yang dapat digunakan dan bermanfaat bagi audiens domain tersebut. Kami melakukan hal ini dengan mengandalkan hubungan yang kami miliki dalam proyek, mengajukan pertanyaan yang baik dan mengakui keahlian kami, alih-alih hanya memikirkan semua jawaban yang sudah ditentukan sebelumnya.

Ketahui kapan harus menahannya, dan ketahui kapan harus melipatnya

Politik dalam negeri. Rekan kerja apatis. Ketergantungan pada metode-metode usang yang menarik bagi bagian tertentu dalam organisasi. Ini adalah masalah serius, dan masalah tersebut benar-benar muncul saat kita mulai mencari tahu detail konten situs web. Siapa yang bisa berada di halaman beranda? Siapa yang dapat memiliki biodata terbesar? Mengapa Anda belum membicarakan departemen ini?

Ketika kita mulai mengerjakan sebuah proyek, kita sering dibombardir dengan informasi. Namun informasinya bias. Itu semua terkait dengan tujuan dan departemen tertentu dalam perusahaan, dan semuanya bersaing untuk area yang sama di situs web. Bagian terburuknya? Kumpulan informasi yang bersaing ini semuanya dapat dibingkai ulang agar sesuai dengan hierarki pesan situs web kami. Contohnya, jika seseorang benar-benar menginginkan tautan perpustakaan universitas menjadi salah satu judul teratas di situs webnya, orang tersebut tentu dapat membuktikannya dengan menggunakan kriteria yang ada.

Tujuan kami dalam hal ini adalah untuk menyusun ulang pembicaraan, memanfaatkan tujuan orang tersebut dan memberikan kelonggaran. Misalnya, bayangkan sebuah situs web universitas tempat empat departemen berbeda bersaing untuk mendapatkan situs web real estat yang sama. Anda punya pilihan. Anda dapat menunjukkan kepada mereka data yang membantah klaim mereka, yang dapat menjadi bumerang jika firasat di masa depan tidak dapat diringkas dengan angka-angka yang relevan. Anda bisa menyerah dan menyelamatkan pertarungan untuk pertempuran yang lebih besar, yang bisa membuat marah dan mengasingkan departemen lain. Atau, Anda dapat menyaring beberapa suara dengan menemukan proyek yang menarik.

Pada akhirnya, setiap departemen atau pemangku kepentingan sangat fokus pada beberapa hal ideal. Berikan kelonggaran di sini atau di sana berdasarkan proyek-proyek yang

Anda minati, sambil menunjukkan banyaknya informasi yang harus Anda tangani. Berempati terhadap kebutuhan mereka dan berikan sedikit, sambil membantu mereka berempati dengan situasi Anda.

Gunakan alat editorial untuk membantu berkomunikasi

Seringkali, kesulitan terbesar yang kita hadapi bukanlah dalam mengatur orang dan menciptakan alur kerja kesulitan terbesar yang kita hadapi hanyalah menemukan cara untuk memberi setiap orang akses terhadap perubahan dan komentar dalam konten kita sendiri. Untungnya, masalah ini sedang ditangani melalui beberapa aplikasi Web online.

Ada solusi untuk perencanaan konten Web, seperti GatherContent, yang membantu Anda berkolaborasi pada peta situs, model konten, dan kebutuhan pra-pengembangan lainnya. Setelah situs tersebut diluncurkan, terdapat aplikasi editorial kolaboratif seperti Editorially atau Poetica yang menjanjikan untuk menciptakan satu sumber tunggal untuk umpan balik konten.

Dan, serius. Mengingat permasalahan yang kita hadapi saat mengumpulkan ribuan dokumen dari tujuh departemen berbeda dan mengorganisasikannya menjadi sesuatu yang dapat digunakan, kita hanya dapat berasumsi bahwa semakin banyak orang yang akan membuat lebih banyak solusi online. Kita berada di ambang ledakan alat konten. Kembangkan dengan CMS yang mudah digunakan oleh editor. Nah, sekarang kita hanya bersikap konyol, bukan? Mungkin tidak. Mari kita bicara tentang pengembang.

Bekerja Dengan Pengembang

Terlepas dari kenyataan bahwa hampir setiap ahli strategi konten mempunyai masalah dengan ketidakkonsistenan sistem manajemen konten, kita semua mengandalkan beberapa jenis solusi manajemen konten untuk membuat pekerjaan kita lebih mudah dan pekerjaan editor kita relatif tidak menimbulkan kesulitan.

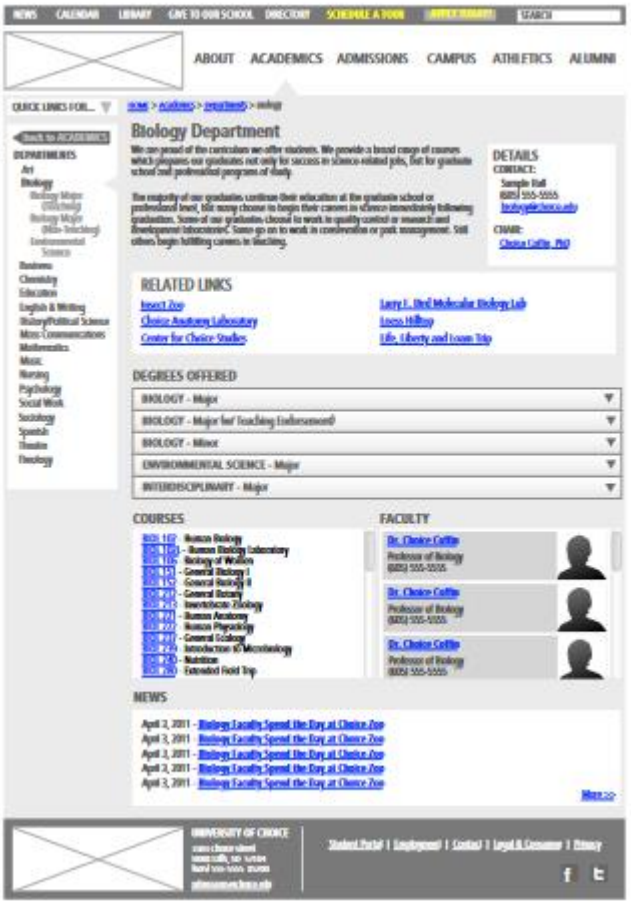
Saya katakan secara relatif, karena meskipun kita tidak pernah mengharapkan editor untuk kembali menandai setiap halaman HTML secara manual, itu tidak berarti solusi CMS saat ini sudah optimal. Mereka tidak. Tapi kita bisa membuatnya lebih baik. Selama kami memiliki hubungan baik dengan pengembang kami.

Keretakan ahli strategi konten/pengembang telah dibicarakan selama bertahun-tahun. “Pengembang tidak memahami kebutuhan editor dan selalu mencari solusi cepat, bukan solusi yang tepat,” kata ahli strategi konten. Para pengembang menjawab, “Para ahli strategi konten mengembangkan solusi yang mustahil dan rumit yang kemudian harus kami terapkan, seolah-olah mereka tidak memiliki pengetahuan tentang cara kerja database atau pemrograman.” Dan itu benar, dalam beberapa hal. Strategi konten kini mencapai titik kritis pada konten yang terstruktur dan berbasis basis data, sementara para pengembang mulai menyadari manfaat dan efisiensi dari pendekatan yang mengutamakan konten.

Namun, stereotip-stereotip tersebut hanya sebatas itu saja. Stereotip. Dalam hal ini, pengembang dan ahli strategi konten bekerja sama dengan sempurna. Pengembang ingin mengetahui jenis salinan apa yang akan ada di situs web, dan bagaimana mereka akan berinteraksi. Ahli strategi konten mendapat manfaat dari kerja sama yang erat dengan

pengembang untuk menciptakan pengalaman CMS yang dapat dipahami dan digunakan oleh editor.

Hal ini membawa kita pada seni pemodelan konten dan bagaimana kami dapat membantu menciptakan mekanisme CMS yang berpusat pada editor bahkan dalam paket CMS yang sudah ada hanya dengan mendengarkan editor kami dan mengumpulkan kolom konten apa yang mungkin mereka perlukan.



Gambar 10.2 Editor dalam paket CMS

Pertama, bayangkan blok konten dasar: halaman departemen universitas, misalnya. Halaman ini dirancang untuk memberikan akses ke semua konten dalam departemen itu. Halaman departemen biologi mungkin memiliki link ke penelitian biologi, daftar jurusan biologi, daftar staf, dan banyak lagi. Ini juga akan memiliki dasar-dasar seperti judul, paragraf pengantar, nomor telepon dan lokasi kantor.

Dengan mengambil contoh ini, kita bisa menyelami lebih dalam. Bidang apa saja yang dibutuhkan? Apa yang perlu diperbarui oleh editor, dan konten apa yang akan berfungsi lebih baik jika diisi melalui tag atau kategori? Dalam hal ini, kita perlu membuat field untuk atribut yang ada di halaman: judul, paragraf pengantar, nomor telepon dan lokasi kantor. Jadi redaksi tidak perlu memasukkan setiap jurusan atau proyek penelitian secara manual, kami juga perlu memasukkan jurusan, berita penelitian, staf, dan kelas yang dikategorikan dalam “biologi.”

Namun, CMS mungkin juga akan menambahkan kolom yang tidak relevan dengan pengalaman pengeditan: tanggal publikasi, ID halaman, bahasa halaman, indeks pengurutan, dan izin, misalnya. Bidang ini dapat membingungkan editor yang harus menyaring hal-hal yang tidak relevan hanya untuk mengubah satu blok salinan, namun bidang ini penting untuk membuat perubahan dalam CMS. Tugas kita adalah membantu menjembatani kesenjangan ini.

Jeff Eaton dari Lullabot berbicara panjang lebar tentang hubungan antara konten dan pengembangan, mengingatkan kita bahwa editor tidak akan senang dengan sistem CMS yang kaku dan khas. Dalam artikelnya, “Ketika Editor Merancang: Mengontrol Presentasi dalam Konten Terstruktur,” dia mengatakan bahwa “memberi editor dan penulis kontrol lebih besar atas presentasi konten mereka adalah hal yang menyebabkan kegagalan banyak model konten yang bermaksud baik.” Untuk ini Alasannya, kata Eaton, “beberapa tim melakukan hal yang sebaliknya. Mereka menumpuk lusinan bidang khusus ke setiap jenis konten untuk menangkap setiap kemungkinan opsi presentasi, atau mereka memberi editor menu templat visual yang dirancang dengan cermat untuk dipilih untuk setiap postingan.”

Di sinilah kolaborasi antara pengembang dan ahli strategi konten menjadi sangat penting. Di sinilah kolaborasi antara pengembang dan editor menjadi sangat penting. Ingat, kami orang-orang konten bukanlah perancang antarmuka, namun kami dapat bekerja di antara kedua kubu untuk membantu mengatur bidang bagi editor sambil tetap mengembangkan model konten terstruktur yang dapat dibangun oleh pengembang. Itulah yang kami inginkan: situs web yang terus digunakan oleh editor dan pengembang, terutama setelah peluncuran.

Tata Kelola: Kapan dan Dimana

Jadi situs webnya telah diluncurkan. Sekarang apa? Mudah. Mulailah dengan membuang asumsi bahwa website sudah selesai. Karena tidak ada situs web yang benar-benar selesai. Pernah. Tidak ada. Mustahil. Dalam pengertian strategi konten tradisional, di sinilah kita mulai membahas tentang analisis data dan pengujian konten serta pengujian A/B dan semua alat tata kelola pasca peluncuran yang telah mulai kami terapkan selama bertahun-tahun. Namun, alat-alat ini tidak berarti apa-apa tanpa pendidikan editor yang mendalam pendidikan yang sama pentingnya dengan model konten apa pun yang dapat kami rancang.

10.4 PENDIDIKAN DAN ITERATING

"Pendidikan?" Anda mungkin bertanya. “Ya,” jawabku. Pendidikan.

Bukan hanya pelatihan CMS. Bukan kursus menulis Web. Maksud saya adalah pemahaman bahwa kita sebagai ahli strategi konten dan pembuat Web secara keseluruhan mempunyai kewajiban untuk mengikuti perkembangan industri agar dapat memberikan manfaatnya kepada mereka yang menggunakan situs web yang kita buat.

Tata kelola adalah titik ketika pencipta dan editor bergabung, ketika pengujian coba-coba dan penggunaan terus-menerus mengungkap masalah yang tidak dapat kita rencanakan pada awal sebuah proyek. Ini juga menandai titik dalam proyek ketika kami menyerahkan keahlian kepada editor. Itu situs web mereka sekarang. Itu adalah konten mereka,

pengetahuan domain mereka. Ketika mereka menjadi lebih akrab dengan situs web tersebut, dan ketika kita menjadi kurang akrab dengan proses mereka, karena proses tersebut berubah secara organik sepanjang masa proyek yang sekarang berjalan, kita kehilangan kemampuan untuk membantu.

Kita tidak boleh kehilangan kemampuan itu. Kita tidak bisa menjadi orang asing. Kita harus check-in. Hal ini memerlukan biaya, baik waktu maupun uang. Dan ya, ini berarti kerja ekstra setelah situs web diluncurkan. Keterlibatan kita harus lebih dari sekedar "mengeluarkan dan mengeluarkannya."

Solusinya adalah berpikir lebih dari sekedar peluncuran. Baik itu proyek internal yang diperkirakan waktunya, atau kontrak klien yang diperkirakan biaya keseluruhannya, setiap kontrak proyek harus mencakup beberapa jam pemeriksaan bulanan. Periksa kembali skrip pertemuan penemuan, ajukan pertanyaan tentang alur kerja saat ini, apa yang berhasil, apa yang tidak. Apa yang dapat kita lakukan untuk mengatasi permasalahan tersebut? Pelatihan tambahan apa yang Anda perlukan sekarang setelah Anda terjun dan terbiasa dengan cara kerja?

Jangan berhemat pada bagian kontrak itu. Jelaskan manfaatnya dengan jelas, dan jangan sampai manfaatnya dihilangkan. Itu sangat penting. Perubahan perlu dilakukan. Jika tidak, tampaknya Anda adalah ahli strategi konten yang sempurna. Dan terakhir saya periksa, tidak ada orang yang sempurna.

Pikirkan Kembali Kalender Editorial: Pemicu Editorial

Kami sering menganggap kalender editorial sebagai alat untuk membantu menghadirkan konten baru kepada pengguna kami. Ini adalah alat tata kelola penting yang memerlukan sebagian besar perhatian editorial kami. Namun sejujurnya, tidak semua perusahaan membutuhkan kalender editorial. Sebagian besar konten kami tidak diluncurkan secara berkala, namun sesuai kebutuhan. Masukkan pemicu editorial. Lebih sedikit kalender editorial dan lebih banyak garis waktu editorial, pemicu editorial tidak berfokus pada memaksakan konten ke dalam jadwal harian atau mingguan, melainkan menciptakan jalur terpandu untuk menyelesaikan konten sesuai kebutuhan. Mereka banyak meminjam metodologi Getting Things Done karya David Allen, yang memecah setiap proyek menjadi serangkaian tugas dan menetapkan garis waktu bila relevan, dan mereka menghilangkan tekanan editorial dari perubahan berkala demi perubahan.

Misalnya, peluncuran produk baru untuk proyek Anda mungkin tidak terjadi pada interval yang ditentukan. Sebaliknya, hal itu terjadi saat produk siap diluncurkan terlepas dari apa yang tertulis di kalender editorial. Dalam hal ini, pemicu editorial untuk konten produk baru membawa kita keluar dari kekakuan kalender dan memasuki alur siklus hidup produk, dengan menggunakan langkah-langkah berikut:

Konten Produk Baru — Waktu maksimum: 5 hari. Aksi Cepat.

1. Mengumpulkan informasi tentang produk baru berdasarkan copy deck Produk Baru (Editor konten web, 2 hari)
2. Unggah konten awal ke situs web menggunakan gambar placeholder (1 hari)
3. Diperlukan persetujuan: Diperlukan persetujuan awal atas konten (manajer produk)

4. Ambil gambar akhir untuk produk dan buat untuk semua ukuran (desain, 150pX, 300pX, spanduk)
5. Persetujuan Akhir (manajer produk/editor konten web, 1 hari)

Kami mengetahui bahwa pemicuan editorial ini memerlukan waktu maksimal lima hari, dan kami mengetahui siapa saja yang terlibat dalam proses tersebut. Produk baru seperti ini memerlukan tindakan segera, jadi kami tahu ini menjadi prioritas dibandingkan pekerjaan yang sudah ada. Terlebih lagi, karena pemicu ini hanya ditarik bila diperlukan, kita tidak perlu menunggu tenggat waktu datang dan pergi karena kita berharap bisa mengerjakan konten produk baru meskipun tidak ada produk baru untuk dikerjakan.

Ini bukan berarti kalender editorial sudah mati; sebaliknya, kalender editorial diperlukan untuk memeriksa pemicu editorial. Alih-alih berfokus pada penerbitan pada tanggal tertentu, kalender ini dirancang ulang untuk menangani penjadwalan konten dan peninjauan konten, yang merupakan bagian yang paling sering diabaikan dalam alur kerja tata kelola.

Hal ini memberikan kita keseimbangan antara kebutuhan editorial dan sumber daya editorial. Kami dapat mengikuti perkembangan konten, dan merencanakan inisiatif editorial di masa mendatang jika hal tersebut muncul. Editor senang. Konten situs web sedang diperbarui. Dan sekarang kami menyelami alur kerja konten yang berpusat pada editor.

Strategi Konten untuk Rakyat, oleh Rakyat

Situs web terdiri dari kode, desain, dan pesan, ketiganya dibuat untuk mencapai suatu tujuan. Meskipun tujuan-tujuan tersebut berbeda dari satu perusahaan ke perusahaan lain dan dari satu proyek ke proyek lainnya, semua tujuan tersebut memiliki satu kesamaan: tujuan tersebut dibuat oleh manusia. Orang-orang di belakang layar, menarik tuas dan menulis salinan. Orang yang bertanggung jawab untuk menciptakan produk baru dan standar merek baru. Orang-orang yang membuat keputusan di tingkat perusahaan dan orang-orang yang menangani pekerjaan penting dalam perubahan sehari-hari.

Jadi, meskipun logis untuk fokus membuat pengguna situs web kami senang dengan mengetahui kebutuhan, konteks, dan tuntutan waktu mereka masuk akal juga untuk membuat editor situs web senang. Selalu beri tahu mereka, bantu mereka memahami proses konten, dan buat alur kerja Web lebih lancar untuk membuat situs web yang lebih baik.

Namun hal ini membutuhkan lebih dari sekedar empati, lebih dari sekedar menempatkan diri kita pada posisi mereka. Ini membutuhkan tindakan. Saat kami mengatakan empati, yang kami maksud adalah ketekunan; tidak hanya menempatkan diri kita pada posisi orang-orang yang kita ciptakan, tapi juga memastikan kita tidak menyerah pada mereka jika mereka tidak segera memahami proses kita, atau ketika mereka terlihat menolak perubahan.

Yang kami maksud adalah pemberdayaan, melakukan apa yang kami bisa untuk membuat tim, rekan kerja, atasan, dan semua orang yang terlibat merasa bahwa mereka tidak sekadar menerapkan perubahan acak, namun juga memiliki peran dalam hal ini. Mereka memiliki kekuatan untuk memfasilitasi perubahan sendiri. Website itu milik mereka, siap dibentuk menjadi sesuatu yang mengagumkan.

Yang kami maksud adalah semangat terhadap proyek dan industri, semangat untuk memecahkan masalah yang pasti akan muncul. Kita harus bisa mengatakan, “Ya, ini adalah pekerjaan yang berat saat ini, tapi semuanya akan baik-baik saja. Ini akan membuat pelanggan Anda senang. Ini akan membuat tim Anda bahagia.”

Dan yang terakhir, yang kami maksud adalah kerendahan hati, suatu sifat yang sering kita abaikan. Ada dua sisi strategi konten, sama seperti ada dua sisi pada setiap bagian proses Web. Ketika ada dua wajah, di situ ada dua pendapat, jadi kita harus sering mengakui bahwa dengan gemetar kita salah. Bahwa kami tidak sepenuhnya memahami hubungan antar departemen, atau kami terlalu bersemangat dengan perubahan yang kami usulkan.

Jika ada tugas yang harus kita lakukan, ingatlah perbedaan antara mengetahui cara melakukan sesuatu dan mengomunikasikan cara melakukan hal tersebut. Kami dapat membuat situs web terbaik yang kami bisa, dengan desain yang unik dan konten setingkat Pulitzer. Namun tidak ada gunanya jika kita tidak dapat mengkomunikasikan kepada klien dan tim kita bagaimana cara menggunakannya. Tidak ada gunanya jika kita membuat untuk pengguna, tapi lupa membuat untuk editor.

Strategi konten bukanlah suatu disiplin ilmu tunggal, dan Web bukanlah semacam klub eksklusif. Industri kami dibentuk dari berbagai disiplin ilmu, yang masing-masing telah beradaptasi dengan situasi saat ini untuk menciptakan situs web yang lebih baik, lebih mudah dibaca, lebih gesit, dan tentunya lebih efektif.

Saya seorang ahli strategi konten. Anda adalah ahli strategi konten. Setiap orang yang bekerja di situs web, yang bekerja dengan kata-kata, yang bekerja di bidang komunikasi atau jurnalisme atau hubungan masyarakat, pada dasarnya, juga merupakan ahli strategi konten pada suatu saat dalam karier mereka. Saya juga pengguna situs web dan audiens. Begitu juga kamu. Kami berada di kedua sisi mata uang. Oleh karena itu, terserah pada kami untuk membuat model konten dan kalender editorial yang lebih baik, belum lagi perangkat lunak pengeditan Web dan sistem manajemen konten yang lebih baik, untuk memastikan hal-hal hebat di Internet yang kami buat dapat ditangani, pada saat peluncuran dan seterusnya. Empati hanyalah sebuah kata kunci sampai dipadukan dengan tindakan yang sesuai. Jadi mari kita mulai mengambil tindakan.

BAB 11
MENDUKUNG PRODUK ANDA

Sebagian besar pesaing Anda menghabiskan hari-harinya menantikan momen langka ketika semuanya berjalan baik. Bayangkan seberapa besar pengaruh yang Anda miliki jika Anda menghabiskan waktu untuk memaksimalkan momen-momen umum yang sebenarnya tidak Anda miliki.

— Seth Godin

Saya tidak pernah bermaksud bahwa dukungan teknis akan menjadi bagian dari pekerjaan saya. Namun, sebagai salah satu pendiri Perch2 (sistem manajemen konten yang dihosting sendiri), dukungan teknis adalah salah satu bagian terpenting dalam pekerjaan saya sehari-hari. Ketika saya berbicara dengan pengembang lain tentang perpindahan kami dari bisnis jasa, mengembangkan proyek untuk klien, menjadi perusahaan produk, saya biasanya mendapat dua tanggapan. Yang pertama adalah untuk mengungkapkan betapa hebatnya hanya mengerjakan produk kita sendiri; yang kedua adalah menanyakan apakah dukungan tersebut merupakan mimpi buruk!

Perch dihosting sendiri: pelanggan kami mengunduh perangkat lunak dan menginstalnya di hosting mereka sendiri dan lingkungan pengembangan lokal. Berbeda dengan aplikasi perangkat lunak sebagai layanan, kami hanya mempunyai sedikit kendali atas lingkungan di mana perangkat lunak kami diinstal, dan hosting Web PHP dapat menjadi lingkungan yang tidak bersahabat.

Pelanggan umum kami adalah desainer Web, banyak di antaranya belum pernah menggunakan CMS, atau menginstal sesuatu yang mengandalkan database MySQL sebelumnya. Jadi meskipun produk kami sangat berguna dan bebas bug, kami akan selalu memiliki orang-orang dengan konfigurasi hosting yang gila atau tidak yakin bagaimana memulainya.

Sesuatu yang selalu mengejutkan kami adalah pentingnya dukungan dari pelanggan kami. Awalnya, kami menandai sebagai favorit di Twitter hal-hal baik yang dikatakan orang tentang kami, sehingga kami dapat menampilkannya di situs web. Perlu dicatat bahwa banyak dari tweet tersebut bukan tentang fitur produk, namun tentang dukungan yang kami berikan. Cara Anda memperlakukan pelanggan yang memberikan dukungan akan membuat perbedaan besar dalam perasaan mereka terhadap produk atau layanan Anda. Apakah pengalaman pelanggan membuat mereka merasa dihargai? Apakah mereka benar-benar merasa bahwa telepon, tiket, atau email mereka sangat penting bagi Anda?

Memberikan pengalaman pelanggan yang baik sangat penting dalam mendukung. Pelanggan mungkin melakukan kontak awal dengan perasaan kesal tentang produk Anda. Mereka punya masalah, dan mereka tidak melakukan apa yang mereka inginkan. Tujuan Anda

bukan hanya memecahkan masalah mereka, namun membuat mereka merasa lebih positif terhadap produk Anda dibandingkan jika mereka tidak mempunyai masalah yang membuat mereka mendapat dukungan.

11.1 NILAI TERSEMBUNYI DARI DUKUNGAN PELANGGAN

Saya harap siapa pun yang membaca buku ini memahami bahwa dukungan pelanggan dan fokus pada pengalaman pelanggan adalah hal yang penting. Oleh karena itu, sebagian besar bab ini akan membahas bagaimana kami sebenarnya memberikan dukungan.

Sebelum saya membahas inti dan inti dari pemberian dukungan, saya ingin memulai dengan membahas tentang manfaat tersembunyi yang dapat diberikan oleh dukungan pelanggan yang hebat terhadap produk atau layanan Anda. Selain membentuk pengalaman yang luar biasa bagi pelanggan, dukungan dapat membuat orang percaya diri untuk mencoba produk Anda; ini bisa menjadi alat pemasaran yang luar biasa; dan dapat menjadi sumber informasi riset pasar dan pengembangan produk.

Membangun Kepercayaan Diri Untuk Mencoba Produk Baru

Fakta bahwa Anda menawarkan dukungan bisa menjadi nilai jual yang besar. Dalam kasus kami, kami menjual produk komersial yang memiliki banyak pesaing gratis. Membeli lisensi memberi pelanggan hak untuk mendapatkan dukungan gratis dan tidak terbatas, sehingga pelanggan kami dapat mengandalkan seseorang yang ada untuk membantu mereka. Pelanggan kami menggunakan produk kami untuk memberikan layanan kepada pelanggan mereka sendiri dan perlu memiliki kepercayaan diri untuk mendapatkan bantuan dengan cepat jika mereka menemui kesulitan. Mereka tidak dapat menunggu berhari-hari untuk menerima bantuan ketika situs web mereka harus ditayangkan pada hari berikutnya, dan mereka perlu memastikan bahwa mereka tidak akan melewatkan tenggat waktu karena lambatnya atau tidak ada sama sekali dukungan.

Ada sejumlah pesaing sumber terbuka bagi Perch, salah satunya adalah WordPress. Namun, dukungan untuk produk-produk ini sering kali dilakukan melalui forum komunitas, tempat Anda bisa mendapatkan bantuan, namun karena Anda belum membayar uang apa pun, hal ini tergantung pada kesediaan seseorang untuk membantu Anda secara gratis. Oleh karena itu, dukungan inklusif yang kami tawarkan dapat menjadi nilai jual yang besar untuk membuat pelanggan tidak lagi bertanya-tanya apakah mereka harus membayar untuk solusi CMS, hingga harus mengeluarkan uang untuk biaya lisensi.

Dukungan Dapat Menjadi Alat Pemasaran Terbaik Anda

Salah satu alasan saya menulis dan berbicara tentang dukungan adalah karena besarnya peran dukungan dalam kesuksesan Perch. Ini adalah hal yang paling banyak dibicarakan orang, dan berbicara dengan pengembang produk, bukan hanya staf dukungan teknis adalah bagian besar dari hal tersebut. Dukungan benar-benar dapat menjadi alat pemasaran terbaik Anda, karena orang-orang membicarakan pengalaman yang mereka miliki.

Seperti yang telah saya uraikan, menawarkan dukungan dapat menjadi pembeda yang signifikan antara produk Anda dan alternatif gratis. Mendengar dari pelanggan yang sudah

ada bahwa janji dukungan benar-benar menggarisbawahi bahwa menjanjikan jauh lebih banyak daripada materi pemasaran Anda sendiri.

Dukungan Sebagai Alat Penelitian Pasar

Satu pelanggan yang dilayani dengan baik bisa lebih berharga daripada iklan senilai Rp.1.000.000.

—Jim Rohn

Agar berita tentang produk Anda tersebar, dukungan tidak hanya bertindak sebagai alat pemasaran dan bahan pembicaraan dukungan juga bisa menjadi tempat yang tepat untuk mengetahui apa yang diinginkan orang dari produk Anda. Anda dapat menghabiskan banyak waktu dan uang untuk melakukan riset pasar, mengirimkan dan menganalisis survei, ketika sumber informasi yang bagus tersedia di saluran dukungan Anda.

Semakin banyak Anda terlibat dengan pelanggan, semakin jelas segala sesuatunya dan semakin mudah untuk menentukan apa yang harus Anda lakukan.

— John Russell, mantan V.P., Harley-Davidson

Terkadang pelanggan ingin memberikan saran atau meminta fitur secara langsung, jadi penting untuk menunjukkan bahwa Anda terbuka terhadap komentar dan memberi orang cara untuk melakukan hal ini. Kami menyediakan forum permintaan fitur yang juga memberikan kesempatan kepada pelanggan lain untuk membaca permintaan dan mengomentarinya, mungkin menambahkan kasus penggunaan baru atau, paling tidak, menunjukkan bahwa permintaan ini relevan bagi lebih banyak orang daripada satu pengguna yang mengusulkan.

Beberapa perusahaan mengambil langkah lebih jauh dengan menggunakan sistem pendukung seperti UserVoice, yang memungkinkan pelanggan memilih permintaan fitur yang telah diposting oleh orang lain. Saya akan berbicara lebih banyak tentang menangani permintaan fitur nanti di bab ini.

Berdasarkan pengalaman kami, penting untuk memperhatikan permintaan fitur yang tidak eksplisit; saat-saat, misalnya, ketika Anda harus memberi tahu pelanggan bahwa produk Anda tidak melakukan hal tertentu. Mencatat hal-hal tersebut membantu Anda melihat apakah kekurangan fitur tertentu berulang kali terjadi.

Yang lebih tidak jelas lagi adalah poin di mana pelanggan harus melewati beberapa rintangan untuk membuat sesuatu berhasil pada produk Anda. Di Perch, ada banyak hal yang dapat Anda lakukan dengan menulis sedikit PHP, tetapi banyak pelanggan yang tidak mengetahui PHP. Kita dapat menyelamatkan orang-orang dari kesulitan memikirkan apa yang harus dilakukan dengan menambahkan fungsi sederhana untuk tugas-tugas yang sering dibutuhkan, memungkinkan pelanggan untuk mengambil kode dari contoh-contoh dalam dokumentasi. Inilah alasan mengapa menurut saya sangat penting bagi pengembang produk untuk juga membantu memberikan dukungan. Kecuali Anda melihat masalah seperti ini

muncul, dan berpikir seperti pengembang untuk menyelesaikannya, masalah tersebut mungkin terlewatkan.

Dukungan pelanggan dapat dianggap sebagai gangguan bagi pengembang, seperti yang dijelaskan oleh Josh Emerson dari Clearleft, “Dukungan pelanggan dapat sangat mengganggu. Hal ini memiliki kecenderungan untuk menjauhkan Anda dari pekerjaan dan membuat Anda keluar dari arus.” Namun, dia juga memahami manfaat melibatkan pengembang dalam mendukung pelanggan yang mereka tulis kodenya: “Manfaat utama jika pengembang menangani dukungan pelanggan adalah memungkinkan mereka melakukan perubahan secara langsung untuk meningkatkan produk berdasarkan masukan pelanggan.”

Saya sangat yakin bahwa pengembang harus dilibatkan dalam dukungan. Namun, di perusahaan yang lebih besar, Anda dapat mengatur waktu orang-orang sehingga mereka membantu memberikan dukungan tanpa mengganggu mereka saat menulis kode. Misalnya, di 37 signals, semua pengembang terlibat dalam dukungan, dan setiap orang bergiliran menghabiskan satu hari bekerja dengan pelanggan.

Nanti di bab ini saya akan membahas alat yang dapat Anda gunakan untuk melakukan dukungan teknis, namun saat menyiapkan sistem apa pun, saya sarankan mencari cara untuk mengumpulkan semua jenis permintaan fitur dan ide yang disebutkan pelanggan. Mereka mewakili tambang emas informasi ketika menentukan fitur apa yang akan Anda tambahkan ke versi produk Anda di masa depan.

Selain permintaan fitur untuk kemampuan masa depan, hati-hati terhadap permintaan fitur yang sudah ada tetapi pelanggan belum menemukannya. Ini merupakan tantangan besar bagi kami karena saat kami mendemonstrasikan kesederhanaan dan kecepatan Perch untuk kasus penggunaan sederhana, terkadang kami gagal melakukan beberapa tugas kompleks yang mampu dilakukannya. Pelanggan yang menanyakan fitur terkini menunjukkan di mana hal ini terjadi. Seringkali yang dibutuhkan adalah dokumentasi yang lebih baik, dan Anda mungkin ingin memeriksa informasi pemasaran dan penjualan Anda untuk memastikan bahwa fitur-fitur tersebut tercantum di tempat yang mungkin dicari orang.

11.2 MENGELOLA PERMINTAAN FITUR

Saya mendapatkan tiket dukungan yang merupakan pemerasan.

— Andrey Butov

Seperti yang telah saya jelaskan di atas, tidak semua tiket dukungan atau postingan di forum produk Anda merupakan masalah yang dapat Anda selesaikan, tutup sebelum melanjutkan dengan cepat. Banyak tiket yang pada dasarnya merupakan permintaan fitur, terutama di masa-masa awal produk Anda. Jika Anda meluncurkan dengan serangkaian fitur kecil untuk menguji air, Anda akan segera mendapatkan banyak permintaan untuk fitur dan tambahan.

Kutipan dari Andrey Butov di atas adalah sesuatu yang dia katakan dalam edisi podcast Bootstrap5 yang dia bawakan bersama Ian Landsman. Butov berbicara tentang dukungannya

terhadap produk selulernya dan bagaimana pelanggan mengancamnya dengan ulasan satu bintang di App Store jika fitur tertentu tidak ditambahkan. Saya senang saya tidak beroperasi di dunia tersebut, namun kami harus mengelola pelanggan yang menganggap satu fitur tertentu penting, meskipun hanya mereka yang memintanya. Bagaimana Anda bisa mengelola permintaan tersebut dan membuat pelanggan senang?

Saya telah menyebutkan tip utama saya untuk mengelola permintaan fitur: pastikan Anda mencatat siapa yang meminta apa, baik sebagai permintaan langsung atau ketika Anda harus memberi tahu pelanggan bahwa produk Anda tidak sesuai dengan keinginan mereka. Log ini penting karena memungkinkan Anda memeriksa apakah suatu fitur memang diminta oleh banyak orang yang berbeda, atau hanya tampak seperti itu karena satu atau dua orang berisik yang berulang kali memintanya. Setelah Anda memiliki daftar fitur yang diminta dan dapat melihat apa yang muncul di atas, Anda memiliki sesuatu untuk dikerjakan.

Melindungi Kasus Penggunaan Inti

Perch sangat didorong oleh permintaan pelanggan. Kami meluncurkan produk yang cakupannya sangat berbeda dengan yang ada saat ini. Saat peluncuran kami memiliki editor konten sederhana tidak ada kemampuan untuk menambahkan halaman baru atau mengubah ukuran gambar, dan tidak ada API, sehingga tidak ada add-on atau kemampuan bagi pengguna untuk mengembangkannya.

Dengan halaman baru, misalnya, kami berasumsi bahwa target pasar kami ingin melindungi arsitektur situs web seperti yang dirancang dan tidak ingin klien menambahkan halaman baru di semua tempat. Namun, pelanggan kami memikirkan kasus penggunaan lain dan kemampuan untuk menambahkan halaman baru menjadi permintaan fitur utama, sesuatu yang kami tambahkan ke produk sejak awal.

Sangat penting bagi Anda untuk mempertahankan kendali atas produk Anda dan melindunginya dari fitur yang membengkak saat Anda mencoba memenuhi permintaan pelanggan. Ingatlah bahwa sangat wajar jika produk Anda tidak cocok untuk semua orang.

Jika Anda mencoba memenuhi setiap kebutuhan yang ada, kemungkinan besar Anda akan mendapatkan produk yang sulit digunakan, dengan begitu banyak pilihan sehingga orang merasa sulit untuk memulainya. Dalam kasus kami, hal ini berarti berakhir seperti banyak produk CMS lain yang jenisnya kami coba berikan alternatifnya.

Kami mengatasi hal ini dengan sangat melindungi kasus penggunaan inti dan dasar kami kasus penggunaan yang tidak berubah dalam empat tahun sejak kami meluncurkannya. Setelah Perch diinstal, untuk mulai mengedit konten pada halaman, Anda perlu menyertakan runtime Perch dan kemudian menambahkan tag konten Perch ke halaman Anda.

Kemudian, yang perlu Anda lakukan hanyalah memuat ulang halaman tersebut di browser Anda dan wilayah tersebut akan muncul di area admin tempat Anda dapat memilih template dan mulai mengedit.

```
<?php include('perch/runtime.php'); ?>
<!doctype html>
<html lang="en">
```

```
<head>
     <meta charset="utf-8" /><title>Perch Example Page</title>
</head>
<body>
     <?php perch_content('Content'); ?>
</body>
</html>
```

Kami tidak ingin pengalaman memulai produk menjadi lebih kompleks dari itu. Namun, pelanggan kami melihat hampir setiap permintaan fitur sebagai tambahan pada perch_content tag yang menyatakan wilayah Perch di laman Anda. Mereka ingin meneruskan jumlah variabel yang semakin banyak saat membuat suatu wilayah, yang akan menghasilkan sesuatu seperti berikut ini di halamannya, dan pengguna harus membuat banyak pilihan hanya untuk memulai.

```
<?php include('perch/runtime.php'); ?>
<!doctype html>
<html lang="en">
<head>
     <meta charset="utf-8" />
     <title>Perch Example Page</title>
</head>
<body>
   <?phpperch_content('Content','index.php','article','html',true,'t
itle','ASC',`singlepage',true,'index.php.true,1,true,',','admin,hr',
'title,text'); ?>
</body>
</html>
```

Kami ingin orang-orang memulai dengan sederhana dan kemudian menemukan fitur yang lebih kompleks sesuai kebutuhan mereka. Jadi, saat kami mencatat permintaan ini, kami tidak menerapkannya sesuai permintaan pelanggan. Ini mungkin berhasil untuk pelanggan tertentu, tetapi akan merusak pengalaman banyak pelanggan lainnya.

Dalam kasus yang diuraikan di atas, ada cara orang dapat mendeklarasikan wilayah dalam kode dan menambahkan opsi tambahan ke wilayah menggunakan tag alternatif sehingga Anda sering dapat menggunakan permintaan ini untuk segera memikirkan solusinya. Jika kami dapat menyediakan fitur yang sering diminta dengan cara yang tidak merusak kasus penggunaan dasar kami, maka, dengan asumsi kami tidak menghabiskan banyak waktu pengembangan untuk kasus penggunaan yang sangat jarang terjadi, biasanya kami akan menyediakannya.

Namun, Anda perlu melakukan panggilan itu berdasarkan fitur demi fitur. Jika produk Anda memiliki API, seperti yang dimiliki Perch, Anda dapat menambahkan fitur tambahan melalui plugin untuk produk inti, dan membuat API tersedia sehingga orang dengan kebutuhan khusus dapat mengembangkan fungsinya sendiri.

Saya melihat kebutuhan untuk melindungi kasus penggunaan inti sebagai potensi masalah saat menggunakan sistem publik yang memungkinkan upvoting. Mungkin ada permintaan fitur populer yang tidak sesuai dengan kasus penggunaan inti yang ingin Anda lindungi. Tidak mengatasinya dapat menyebabkan orang percaya bahwa Anda tidak mendengarkan.

Saya berbicara dengan Ian Landsman dari UserScape6, yang mengembangkan aplikasi helpdesk HelpSpot, sistem yang kami gunakan untuk dukungan pelanggan kami. Saya bertanya kepadanya apakah fitur tertentu tidak disertakan karena berada di luar cakupan dukungan, atau ada masalah yang tidak boleh dipecahkan oleh perangkat lunak. Dia berkata:

Sangat. Faktanya, HelpSpot menangani hal ini dengan lebih serius daripada kebanyakan orang. Banyak pesaing kami yang memiliki banyak sekali fitur lain yang, bagi saya, berada di luar layanan pelanggan, seperti pemantauan jaringan, pelacakan aset, fungsi CRM, pengaturan ulang kata sandi jaringan, dan sebagainya. Tentu saja, sebagian orang menyukai solusi lengkap, namun kami lebih memilih membangun sesuatu yang sangat terfokus pada satu masalah tertentu dan menyelesaikannya dengan baik daripada menjadi segalanya bagi semua orang.

Ketika Anda harus menolak suatu fitur, mungkin karena fitur tersebut tidak sesuai dengan produk atau hanya berguna bagi sebagian kecil orang, Anda berisiko mengecewakan pelanggan. Hal ini terutama berlaku jika pemohon fitur tersebut sangat vokal. Kami pernah mengalami skenario tersebut satu atau dua kali, dan pelanggan kemudian menuduh kami tidak mendengarkan pelanggan kami. Saya merasa hal ini sangat sulit untuk diatasi karena kami menghabiskan sepanjang hari mendengarkan pelanggan kami dan berusaha membuat produk terbaik untuk mereka.

Yang harus saya ingat dalam kasus seperti ini adalah kami tidak menambahkan fitur karena kami mendengarkan dan memahami pelanggan kami; menyertakan fitur tersebut tidak akan menguntungkan sebagian besar orang, atau bahkan mungkin merugikan penggunaan produk oleh banyak orang.

Saya melihat ini sebagai kelemahan utama karena pengembang suatu produk juga mendukungnya. Sebagai pendiri perusahaan dan pengembang suatu produk, Anda pasti merasa terikat secara emosional dengannya. Ketika pelanggan menuduh Anda tidak membantu mereka, atau memberi tahu Anda bahwa produk Anda jelek karena tidak memberikan manfaat tertentu, hal ini mungkin sulit untuk ditangani. Inilah saatnya untuk menjauh, tarik napas dalam-dalam, dan lihatlah daftar hal-hal indah yang dikatakan orang-orang!

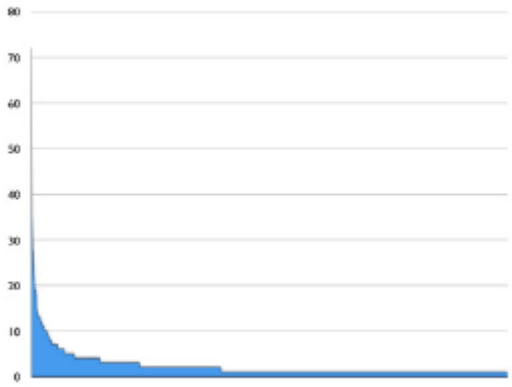
Saat menambahkan fitur berdasarkan permintaan pelanggan, Anda harus memastikan bahwa Anda selalu mengingat basis pelanggan Anda yang puas. Orang-orang yang menyukai produk Anda dan menggunakannya tanpa masalah tidak pernah memberikan dukungan. Ingatlah hal-hal tersebut saat merencanakan penambahan produk dan jangan biarkan pandangan dari kelompok minoritas yang ribut menjauhkan Anda dari produk yang dibeli dan digunakan oleh kelompok mayoritas yang diam.

Mengelola Pelanggan yang Sulit

Pelanggan sering kali "salah". Mereka juga manusia, jadi mereka berhak mendapatkan empati, rasa hormat, dan saran terbaik yang dapat kita berikan, meskipun itu berarti mengarahkan mereka ke pesaing yang lebih sesuai dengan kebutuhan mereka. Sangat mudah untuk terjebak dalam kata-kata yang digunakan seseorang, bukan pada maksudnya — mereka menghubungi karena membutuhkan bantuan.

— Jim Mackenzie, 37 sinyal

Meskipun beralih dari pekerjaan klien ke suatu produk adalah sesuatu yang dicita-citakan banyak dari kita, hal ini bukanlah utopia tanpa klien seperti yang Anda bayangkan. Sebagai pengembang yang meluncurkan produk, kami beralih dari menangani sangat sedikit klien selama satu tahun menjadi memiliki ribuan pelanggan yang harus ditangani. Mayoritas pelanggan dengan senang hati menggunakan produk kami dan kami jarang mendengar kabar dari mereka, meskipun ada beberapa nama yang kami kenal baik sebagai pendukung.



Gambar 11.1 Dukungan pelanggan yang panjang.

Saat saya berbicara dengan rekan-rekan saya tentang Perch dan dukungan, mereka cenderung bertanya tentang cara kami menangani pelanggan yang menyita seluruh waktu kami dan menanyakan pertanyaan demi pertanyaan. Ada hari-hari ketika kami merasa seolah-olah kami hanya membangun beberapa situs web satu per satu, namun menurut saya akan menarik untuk melihat statistik dukungan kami yang sebenarnya. Hanya 26% pelanggan kami yang pernah mengumpulkan tiket dukungan dan hanya 10% yang pernah mengumpulkan lebih dari satu tiket. Setelah kami mendapatkan pemohon yang lebih sering, kami menemukan bahwa 10% permintaan dibuat oleh 10 orang yang sama dan faktanya 2% permintaan berasal dari satu pelanggan.

Grafik di atas menunjukkan bahwa terdapat long tail dukungan pelanggan yang tercermin dari mayoritas pelanggan yang tidak pernah menghubungi dukungan atau hanya melakukannya sekali. Kami juga memiliki sejumlah kecil pelanggan yang sering menghubungi kami. Dalam pembelaan mereka, para pemohon dukungan utama tersebut juga memiliki banyak lisensi sehingga telah mengembangkan banyak proyek menggunakan perangkat lunak

kami. Gagasan bahwa kita akan dibanjiri oleh orang-orang yang membutuhkan bantuan dasar, atau bantuan terkait masalah seperti CSS, tidaklah benar. Ya, kami menghabiskan banyak waktu untuk membantu beberapa pelanggan dengan hosting PHP yang dikonfigurasi dengan buruk, atau menjawab pertanyaan yang lebih cocok untuk forum CSS. Namun, sebagian besar pelanggan kami tidak pernah kami dengar kabarnya sama sekali. Kami tahu mereka menggunakan Perch dan banyak di antara mereka yang memiliki banyak lisensi, namun lisensi tersebut pasti berfungsi untuk mereka.

Selama produk Anda telah dideskripsikan dan diiklankan secara akurat, dan materi pendukung Anda bagus, sebagian besar pelanggan tidak perlu menggunakan dukungan sama sekali. Namun, Anda akan melihat beberapa orang yang sering muncul di sistem tiket Anda.

11.3 MASALAH KLIEN AKHIR

Seiring dengan semakin matangnya produk Perch, sekumpulan pelanggan sulit mulai bermunculan, dan untuk produk seperti milik kami, jumlah tersebut mungkin akan terus meningkat. Karena perangkat lunak kami diinstal oleh pelanggan kami untuk klien mereka, kami mulai dihubungi oleh klien akhir tersebut, orang atau perusahaan yang menginstal CMS oleh seorang desainer Web. Cara kami menangani hal ini dapat dijadikan sebagai studi kasus tentang cara menangani situasi yang sangat spesifik yang muncul dalam mendukung produk tertentu.

Biasanya kami dapat mengidentifikasi seseorang sebagai klien akhir ketika kami melihat permintaan dukungan yang berbunyi seperti: “Bagaimana cara menambahkan gambar ke halaman 'Tentang' saya?” atau “Saya tidak bisa login ke Perch saya.” Kami biasanya bertanya kepada orang-orang yang kami curigai sebagai klien akhir, “Apakah Anda menginstal Perch sendiri?” Kami kemudian mendengar cerita tentang bagaimana situs web mereka dikembangkan oleh seorang desainer yang menghilang, atau dengan siapa kliennya berselisih. Terkadang klien tidak mau membayar untuk pembaruan dan berpikir mereka harus bisa mengelola semuanya sendiri mulai sekarang. Kami memiliki satu klien yang menghubungi kami yang mencoba mentransfer lisensi karena, sayangnya, desainer mereka telah meninggal. Semakin mapan Perch sebagai suatu produk, semakin banyak situasi yang harus kita hadapi.

Situasi klien akhir menimbulkan beberapa masalah. Pertama adalah pihak sah yang memiliki lisensi. Kontrak kami adalah dengan orang yang membeli lisensinya: jika itu adalah desainer Web maka kontrak kami adalah dengan desainer tersebut. Tidak peduli seberapa keras klien melakukan hal tersebut (atau berpikir demikian), kami tidak bisa begitu saja memberi mereka kendali atas lisensinya, jadi kami harus menjelaskan bahwa mereka perlu menghubungi desainer tersebut dan meminta desainer tersebut untuk mengajukan permintaan transfer lisensi kepada mereka. kita.

Masalah lainnya adalah kami berasumsi bahwa desainer Web akan mendukung klien mereka. Kami tidak disiapkan untuk mendukung orang-orang yang menggunakan Perch sebagai editor, karena kami tidak mengetahui apa pun tentang instalasi individual karena CMS dihosting sendiri. Kami harus memberi tahu klien akhir bahwa kecuali mereka memiliki

keterampilan dasar dalam HTML, CSS, dan pembuatan situs web umum serta mampu mengikuti materi dukungan kami, mereka perlu mencari desainer lain.

Kami telah menangani sumber desainer tersebut untuk klien akhir melalui program pengembang terdaftar kami. Pelanggan Perch yang mendaftar biasanya sudah berpengalaman dan menyukai produk tersebut. Kami kemudian dapat mengirimkan daftar tersebut ke klien akhir sebagai awal dalam menemukan seseorang untuk membantu mereka kami tahu bahwa pelanggan kami yang berpengalaman ini memiliki pemahaman tentang produk untuk membantu mereka. Mudah-mudahan, kita semua dapat melakukannya dengan baik: kita dapat membuat orang tersebut tetap menggunakan produk kita; seorang desainer mendapat klien baru yang berterima kasih; dan klien akhir mendapat bantuan dengan situs web mereka.

Situasi ini mungkin hanya relevan bagi Anda jika Anda menawarkan alat atau layanan yang digunakan orang untuk menyediakan layanan mereka sendiri. Jika, seperti kami, Anda berada dalam situasi tersebut, ada baiknya untuk mempertimbangkan cara menangani orang-orang yang tidak memiliki kontrak dengan Anda, namun masih menjadi pengguna produk Anda dan mungkin akan datang kepada Anda untuk meminta bantuan jika hubungannya dengan orang yang benar-benar menerapkan produk Anda rusak.

Pelanggan Yang Sulit, Atau Bentrokan Budaya?

Pelanggan adalah manusia dan manusia dapat melihat situasi dengan cara yang tidak terduga.

— Marilyn Suttle

Terlepas dari upaya terbaik kami, kami terkadang mengalami insiden di mana pelanggan merasa kami tidak membantu atau bahkan kasar, dan hal ini cenderung disebabkan oleh salah tafsir atas balasan kami berdasarkan fakta bahwa kami adalah orang Inggris dan berbicara bahasa Inggris, dan kami pelanggannya berasal dari seluruh dunia. Meskipun bahasa Inggris adalah bahasa pertama yang tidak berlaku bagi sebagian besar pelanggan kami masih ada ruang bagi perbedaan budaya untuk menciptakan kebingungan ketika membahas masalah dukungan.

Contoh paling ekstrem kami mengenai hal ini datang dari tiket dukungan yang ditanggapi oleh Drew. Pelanggan mengalami kesulitan dengan beberapa bagian perangkat lunak dan dalam salah satu balasannya, Drew menyebutkan memiliki model mental tentang cara kerja sesuatu. Tanggapan ini membuat pelanggan marah dan Drew tidak mengerti alasannya. Saya membaca thread tersebut dan juga tidak dapat memahami mengapa pelanggan menjadi begitu kesal, dan kemudian kami sadar: pelanggan tersebut mengira bahwa Drew menyebutnya sebagai model mental! Mungkin dia seorang model, kita tidak tahu. Namun kami tidak terbiasa memanggil nama pelanggan kami.

Sebagian besar kebingungan tidak terlalu ekstrem seperti itu, namun ada gunanya menggunakan bahasa yang sejelas dan sesederhana mungkin untuk mendukungnya. Orang yang Anda tanggapi mungkin menggunakan bahasa Inggris sebagai bahasa kedua, mungkin

merasa sulit mengikuti instruksi tertulis karena alasan apa pun, atau mungkin tidak memahami selera humor atau bahasa sehari-hari dari belahan dunia Anda.

Salah satu tantangan menjadi bisnis yang menawarkan produk digital adalah Anda pada dasarnya menjadi eksportir sejak hari pertama. Bisnis tradisional sering kali sudah cukup mapan sebelum mereka mulai mengekspor ke pasar baru. Saat menjual produk digital, Anda memiliki keuntungan karena dapat menjual ke seluruh dunia dengan sedikit pengeluaran tambahan, namun hal ini berarti Anda harus belajar dengan cepat bagaimana menghadapi pelanggan yang mungkin tidak bisa berbahasa Anda dengan baik, atau memiliki ekspektasi yang berbeda terhadap produk tersebut. Anda berdasarkan apa yang normal dalam budaya mereka.

Pelanggan Yang Sangat Membutuhkan Produk Yang Berbeda

Kami memiliki beberapa pelanggan yang bukan berasal dari target pasar kami, dan bukan tipe orang yang kami pikirkan saat mengambil keputusan. Perch selalu ditujukan untuk desainer dan pengembang Web profesional, jadi kami berasumsi pengguna kami mengetahui HTML dan CSS. Kami tidak berasumsi mereka memahami PHP, atau cara mengatur database atau bagian teknis lainnya dalam menggunakan sistem manajemen konten. Namun kami berharap mereka mengetahui cara mengembangkan situs web HTML statis.

Namun, kami telah merekrut sejumlah pelanggan yang menggunakan alat pengembangan visual seperti Dreamweaver dan Freeway Pro, dan yang tidak mengetahui HTML. Ketika orang tersebut menghubungi kami sebelum membeli produk kami, kami selalu menjelaskan bahwa Perch memerlukan pengetahuan HTML dan CSS. Namun, jika mereka kemudian terus membeli, akan sangat sulit untuk mendukung mereka.

Kami baru-baru ini menaikkan harga produk, sehingga mungkin sedikit kurang menarik bagi penghobi, dan lebih menargetkan pasar profesional. Kami perlu melakukan sejumlah kompromi agar Perch dapat digunakan oleh orang-orang yang tidak memiliki pengetahuan HTML apa pun. Kami tidak ingin mengkompromikan produk dengan cara seperti itu karena kami ingin produk tersebut menjadi alat untuk penggunaan profesional, dan untuk menarik desainer dan pengembang profesional ke dalamnya. Ada alat pembuat situs web lain di luar sana yang lebih cocok untuk non-coder.

Mendukung pelanggan yang tidak cocok dengan produk Anda adalah hal yang sulit, karena memenuhi permintaan dan kebutuhan mereka bisa berarti menjadi produk yang berbeda. Materi penjualan yang baik dapat membantu, serta indikasi yang jelas tentang untuk siapa produk tersebut, dengan kasus penggunaan yang umum. Namun, pengalaman kami menunjukkan bahwa meskipun kami memberi tahu calon pelanggan secara langsung bahwa menurut kami Perch tidak cocok untuk mereka, terkadang mereka tetap membeli lisensinya.

Orang Yang Sangat Sulit

Kami menemukan bahwa dengan bersikap ramah dan profesional, kami tidak menghadapi banyak masalah dengan pelanggan yang benar-benar sulit. Mungkin percakapan dukungan yang paling sulit adalah dengan orang yang berisik dengan permintaan fitur yang sangat spesifik yang telah kami jelaskan tidak akan kami penuhi.

Kami menawarkan dukungan gratis tanpa batas, yang menurut saya membebaskan kami dari beberapa percakapan rumit yang mungkin akan kami lakukan jika orang membayar per insiden. Tidak peduli apa masalahnya: jika kami dapat membantu, kami akan membantu, atau kami membantu pelanggan mendapatkan bantuan (misalnya, jika masalahnya ada pada host mereka).

Tentu saja ada beberapa orang yang mustahil untuk disenangkan. Selama hal tersebut jarang terjadi dan bukan hal yang biasa, kemungkinan besar masalahnya ada pada mereka — bukan Anda atau produk Anda. Saran saya adalah tetap bersikap objektif setiap saat, memperlakukan pelanggan yang sulit dengan cara yang sama seperti pelanggan lainnya, dan berusaha memecahkan masalah mereka.

Mendukung Model Penetapan Harga

Di bagian sebelumnya saya sebutkan bahwa kami memberikan dukungan gratis dan tidak terbatas di Perch. Dukungan selalu menimbulkan biaya dalam hal waktu Anda atau karyawan Anda. Cara Anda menutup biaya tersebut akan bergantung pada model penetapan harga yang Anda pilih. Anda akan menemukan model yang berkisar dari dukungan gratis yang disertakan dengan pembelian, langganan dukungan, dan model per insiden, di mana Anda membayar setiap kali Anda mengajukan permintaan dukungan.

Kami sering ditanya mengapa kami memutuskan untuk menawarkan dukungan gratis dan tidak terbatas, daripada berlangganan dukungan berbayar atau dukungan per insiden. Alasan penting untuk tidak memilih model bayar per insiden atau berlangganan adalah karena kami merasa bahwa kami tidak seharusnya mendapatkan manfaat dari orang-orang yang membutuhkan dukungan. Tidak setiap permintaan dukungan terkait dengan kegagalan produk atau dokumentasi, namun dengan menawarkan dukungan gratis, kami memberikan tanggung jawab kepada kami untuk memastikan orang-orang mendapatkan pengalaman yang luar biasa.

Seperti yang telah saya tunjukkan, sangat jarang ada pelanggan yang mengharapkan kami membantu membangun situs web mereka satu per satu. Meminta orang membayar untuk mendapatkan bantuan pada dasarnya akan menghukum semua orang, karena hanya sedikit orang yang memerlukan banyak bantuan.

Kami juga ingin produk kami dapat dengan mudah dimasukkan ke dalam proses penjualan untuk pekerjaan desain Web rata-rata. Biasanya, desainer mengutip berdasarkan proyek, termasuk biaya skrip dan layanan yang digunakan. Jika mereka kemudian membutuhkan bantuan dan harus membayar bantuan tersebut, mereka mungkin tidak dapat membebankan biaya tersebut kembali kepada klien mereka.

Kami selalu ingin harga kami adil dan mudah dimengerti. Jadi ketika kami memutuskan harga Perch, kami harus memikirkan berapa banyak yang perlu kami hasilkan agar mampu mendukung dan mengembangkannya. Dengan produk baru yang jelas merupakan sebuah tebakan, namun jika Anda ingin menawarkan dukungan inklusif, Anda perlu mempertimbangkan waktu saat menetapkan harga.

Untuk aplikasi SaaS, pelanggan biasanya membayar layanan bulanan, sehingga Anda sudah menerima pembayaran berulang yang umumnya mencakup dukungan. Dalam hal ini,

Anda perlu memastikan bahwa biaya bulanan mencakup waktu yang diperlukan untuk menawarkan dukungan tersebut.

Meskipun kami merasa bahwa menawarkan dukungan gratis dan inklusif dengan lisensi untuk Perch adalah solusi terbaik, perusahaan lain di bidang kami telah beralih dari model tersebut. CMS EXpressionEngine dulunya beroperasi dengan model yang mirip dengan Perch, namun baru-baru ini dipindahkan ke model berbayar. Pengguna EXpressionEngine mendapatkan dukungan selama tiga bulan yang disertakan saat mereka pertama kali membeli lisensi, namun kemudian harus membayar dukungan lanjutan dengan berlangganan di berbagai tingkat.

Paket dukungan ini merupakan campuran dari langganan dan dukungan per insiden. Sebuah paket mencakup sejumlah tiket mendesak, meskipun Anda juga dapat membeli tiket mendesak tambahan jika Anda membutuhkannya. Ellis Labs menerbitkan postingan blog10 yang merinci mengapa mereka pindah ke sistem ini, yang memberikan beberapa wawasan tentang pengambilan keputusan mereka. Tampaknya faktor pendorong utama adalah faktor ekonomi, dan mereka menyatakan:

Biaya untuk menyediakan layanan yang berkelanjutan tidak didukung dengan baik oleh pembelian satu kali, dan pemutusan hubungan semakin parah seiring dengan meningkatnya jumlah pembelian satu kali. Berdasarkan pengalaman kami, saya tidak setuju bahwa menawarkan dukungan inklusif adalah hal yang mustahil. Meskipun lisensi Perch merupakan pembelian satu kali, pelanggan kembali untuk membeli lisensi tambahan. Apa yang telah kami lihat adalah bahwa kami sering memberikan dukungan yang jauh lebih besar untuk pembelian lisensi awal daripada yang diharapkan untuk harga lisensi, namun karena pelanggan kami terus membeli lisensi untuk setiap situs web yang mereka buat, lama kelamaan hal tersebut menjadi seimbang. Saya menyertakan informasi Ellis Labs di sini karena menurut saya menarik untuk membandingkan pendekatan dua perusahaan yang cukup mirip dalam hal jenis produk yang dijual dan persyaratan dukungan.

Model lain yang digunakan oleh beberapa perusahaan yang menawarkan sistem yang dihosting sendiri atau produk desktop adalah dengan menggabungkan biaya satu kali untuk perangkat lunak dengan biaya dukungan tahunan yang juga mencakup pembaruan pada produk. HelpSpot menggunakan model penetapan harga ini untuk dukungan dan pembaruan.

Dengan model berlangganan, seringkali pembelian awal mencakup dukungan satu tahun; jika Anda ingin terus menerima dukungan dan peningkatan, maka Anda harus memperbarui bagian dukungan dari lisensi. Namun, dengan model ini, Anda dapat terus menggunakan perangkat lunak meskipun Anda tidak memerlukan dukungan berkelanjutan untuk perangkat lunak tersebut. Kami jarang menggunakan dukungan dari Help-Spot, meskipun kami ingin selalu memperbarui perangkat lunak dengan peningkatan apa pun. Saya sangat senang membayar langganan dukungan tahunan saya karena saya mengetahui pekerjaan yang dilakukan pada suatu produk dan saya senang bahwa produk tersebut terus dikembangkan dan didukung.

Baik Anda mengenakan biaya secara eksplisit untuk dukungan atau menggabungkannya dengan biaya satu kali atau bulanan, biaya dukungan harus ditanggung

secara keseluruhan. Saat Anda membuat keputusan penetapan harga untuk produk Anda, pastikan untuk memperhitungkan waktu yang diperlukan. Jika Anda merasa bahwa dukungan menguras sumber daya Anda, akan jauh lebih sulit bagi Anda untuk merasa senang karena telah melakukan upaya ekstra demi pelanggan.

Pelanggan dapat kecewa jika dukungan atau model penetapan harga berubah sehingga mereka merasa merugikan mereka. Hal ini tidak berarti bahwa Anda tidak boleh mengubahnya jika Anda menyadari bahwa model Anda saat ini tidak bekerja untuk Anda, namun ada baiknya Anda memikirkan secara serius dalam keputusan awal Anda karena jika Anda dapat melakukannya dengan benar pada saat pertama kali, hal ini akan menghemat potensi buruk. PR seputar perpindahan.

11.4 STRATEGI UNTUK MEMINIMALKAN DUKUNGAN

Secara kolektif Anda dapat menemukan pola yang sangat jelas [...] jika banyak orang bertanya di mana menemukan sesuatu maka Anda mungkin memiliki masalah desain di sana.
—David Goss

Memberikan sejumlah dukungan tidak bisa dihindari. Betapapun jelasnya dokumentasi Anda, seseorang akan memerlukan bantuan untuk memulai; betapapun hati-hatinya pengujian Anda, beberapa bug akan lolos. Jadi, strategi dukungan yang masuk akal juga harus mencakup upaya untuk meminimalkan jumlah permintaan yang masuk. Kami telah sangat berhasil dengan pendekatan ini di Perch; meskipun terjadi peningkatan besar dalam penjualan lisensi dan pelanggan baru selama setahun terakhir, jumlah tiket dukungan yang kami tangani tetap sama.

Bagaimana kita mengelolanya? Untuk Perch, kami berupaya merancang permintaan dukungan. Jika orang sering kali harus meminta dukungan untuk bagian tertentu dari produk Anda, apakah mungkin untuk menghapus atau mengubah hal yang menyebabkan permintaan tersebut, daripada berasumsi bahwa hal tersebut tidak dapat dihindari? Sebagai contoh, kami menerima sejumlah tiket setiap minggu saat pengguna pertama kali menginstal Perch, login, lalu melihat pesan berikut: “Maaf, kunci lisensi Anda tidak valid untuk domain ini.”

Para pengguna ini melewatkan bagian email yang dikirim dengan rincian lisensi mereka yang menjelaskan bahwa mereka perlu masuk ke akun mereka di situs web kami dan mengonfigurasi domain tempat mereka menggunakan Perch. Akan mudah bagi kami untuk duduk santai dan mengeluh tentang pengguna yang tidak membaca, dan terus menangani tiket saat mereka masuk.

Gambar 11.2 Layar yang diperbarui memberikan instruksi kepada pengguna.

Namun, masalah ini berarti bahwa pengguna pertama kali memulai lebih lambat dari yang kami inginkan, karena mereka harus memposting tiket dan menunggu kami menjawab, dan kami menghabiskan waktu untuk menanggapi permintaan yang sama berulang kali. Kami menghilangkan permintaan ini dengan perubahan sederhana pada pesan di layar itu. Daripada hanya memberi tahu pengguna bahwa mereka telah melakukan kesalahan, kami memberi tahu mereka cara memperbaikinya. Ini berarti pelanggan kami dapat dengan cepat mulai menggunakan produk kami tanpa harus menunggu dukungan.

Khususnya pada produk teknis, tidak selalu mungkin untuk mencegah orang terjebak. Apa yang tampak jelas bagi pengguna yang lebih teknis bisa jadi membingungkan bagi pemula. Jadi, jika Anda sering menjawab pertanyaan tentang bagian tertentu dari produk Anda, mungkin ada beberapa hal yang dapat Anda lakukan untuk membantu orang-orang yang memilikinya.

Meskipun banyak pelanggan kami yang sangat terbiasa menginstal dan menggunakan sistem manajemen konten, mungkin karena menggunakan WordPress, kami memiliki sejumlah besar pelanggan yang menganggap Perch sebagai CMS pertama yang mereka instal. Jika pengalaman Anda dalam pengembangan Web hingga saat ini hanya berupa file HTML datar, maka sistem apa pun, betapapun mudahnya, akan melibatkan pembelajaran konsep-konsep baru. Kami menyadari bahwa kami sedang membimbing orang-orang melalui tahap awal penggunaan Perch. Begitu mereka memahami konsepnya, kami sering kali tidak mendengarnya lagi mereka hanya memerlukan bantuan untuk memulai.

Kami mengurangi frekuensi permintaan ini dengan membuat serangkaian video tutorial. Ini membantu pendatang baru di Perch melalui pengaturan awal dan implementasi dasar di situs web. Kami telah menemukan bahwa hal ini sangat mengurangi jumlah pegangan

awal yang perlu kami lakukan. Memiliki beragam materi pendukung akan memastikan bahwa pelanggan Anda dapat belajar dengan cara yang sesuai untuk mereka.

Tidak Ada Kebijakan FAQ

Kami selalu berusaha menghindari halaman pertanyaan umum, yang cenderung hanya menjadi tempat yang tepat untuk mendorong pelanggan, dan cara untuk menghindari upaya memecahkan masalah agar tidak terjadi. Contoh halaman login kami sebelumnya adalah contoh yang bagus untuk ini. Akan sangat mudah bagi kami, jika kami memiliki area FAQ, untuk menaruh informasi tersebut di sana. Namun, informasi ini sudah dirinci di situs web kami dan di email pasca pembelian, namun masih banyak orang yang melewatkannya.

Menurut saya, menambahkannya ke halaman FAQ tidak akan menghasilkan apa pun selain menyediakan tempat lain bagi orang-orang untuk tidak mencari informasi, sehingga tidak akan mengurangi masalah sebenarnya. Kami percaya bahwa jika memungkinkan, masalah yang menyebabkan orang menghubungi dukungan harus diperbaiki, atau penjelasan yang masuk akal ditempatkan dalam dokumentasi. Memasukkan sesuatu ke dalam halaman FAQ bukanlah pendekatan terbaik.

11.5 ALAT UNTUK DUKUNGAN

Saya pikir siapa pun yang memulai bisnis yang memerlukan dukungan pelanggan harus menyiapkan sistem terukur yang tepat sejak awal - buatlah sistem ini berfungsi hanya dengan memiliki 1 "agen" dukungan, tetapi pastikan juga sistem tersebut akan tetap berfungsi jika Anda tiba-tiba memiliki 10. Cukup menunjuk support@perusahaansaya.com ke kotak masuk Anda sendiri adalah cara yang buruk untuk memulai. Ketika transisi tersebut terjadi, itu mungkin berarti Anda sedang berkembang sehingga akan ada banyak hal lain yang terjadi, jadi Anda tidak ingin berurusan dengan migrasi sistem pusat bantuan dan perubahan proses pada saat yang bersamaan.

—David Goss

Ada banyak pilihan yang tersedia untuk membantu Anda mengelola permintaan dukungan pelanggan. Dari sekedar menawarkan dukungan melalui email hingga aplikasi helpdesk SaaS, bagaimana Anda memilih metode terbaik untuk mendukung suatu produk? Saat meluncurkan produk atau layanan kecil, sering kali hal pertama yang dilakukan orang adalah mengirimkan alamat email support@ dan menangani masalahnya langsung di kotak masuk. Masalah dengan dukungan email yang tidak didukung oleh sistem lain adalah bahwa dukungan tersebut sangat sulit untuk ditingkatkan. Jika beberapa orang masuk ke kotak surat yang sama, Anda perlu menerapkan semacam sistem untuk memastikan bahwa permintaan tidak terlewatkan (karena semua orang mengira orang lain akan melihatnya), dan permintaan tidak dijawab dua kali (saat dua orang membuka email secara bersamaan).

Jika Anda memiliki tim yang sangat kecil, mungkin dengan satu orang yang bertanggung jawab menangani email, ini mungkin berhasil. Namun, jika Anda melihat adanya kebutuhan untuk meningkatkan dukungan Anda, maka saran saya adalah melihat sistem apa yang dapat diterapkan untuk memastikan bahwa Anda tidak bergantung pada satu kotak

masuk sebagai sistem Anda. Banyak produk pusat bantuan yang memungkinkan pelanggan menghubungi Anda melalui email, jadi beralih dari email sebagai metode mengelola permintaan tidak berarti pelanggan Anda harus masuk dan mengirim pesan ke forum dukungan. Sistem dapat sepenuhnya transparan kepada pengguna.

Statistik Dukungan

Sesuatu yang sering dilakukan oleh sistem helpdesk adalah memberikan statistik tentang tiket dukungan yang dikumpulkan: angka yang saya kutip sebelumnya datang langsung dari sistem dukungan kami. Dengan mengetahui tren dukungan, Anda dapat melihat apakah fitur di produk atau layanan Anda menyebabkan lonjakan atau penurunan jumlah tiket. Anda bisa mendapatkan gambaran tentang bagaimana kebutuhan dukungan Anda perlu ditingkatkan untuk memenuhi kebutuhan pelanggan yang jumlahnya semakin banyak. Hal ini dapat membantu Anda merencanakan perekrutan staf pendukung tambahan sebelumnya, dan memastikan bahwa pendapatan yang Anda peroleh pada saat itu akan cukup untuk menutupi perluasan tersebut.

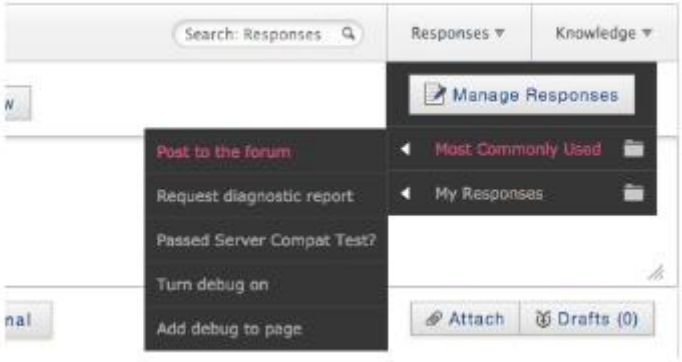
Selain memberi Anda dasar untuk perencanaan jangka panjang, statistik dapat membantu Anda merencanakan kebutuhan dukungan sehari-hari. Saat kami menghadirkan versi baru yang besar, kami melihat lonjakan besar dalam jumlah tiket seiring orang-orang melakukan upgrade dan mencoba fitur baru. Mengetahui hal ini terjadi berarti kami tidak berencana melakukan apa pun yang akan mempersulit pemberian dukungan dalam beberapa hari setelah rilis.

Respon Kaleng

Respons terekam adalah balasan standar terhadap pertanyaan standar, yang dapat dikirim dengan cepat saat pertanyaan tersebut diajukan. Kami merasa hal ini sangat berguna dalam situasi umum ketika pelanggan pada awalnya tidak memberikan informasi yang cukup kepada kami untuk dapat membantu mereka. Misalnya, Perch memiliki laporan diagnostik yang tersedia di perangkat lunaknya, dan kami meminta pelanggan yang meminta dukungan menempelkan laporan ini ke tiket mereka karena laporan ini memberi kami banyak informasi tentang lingkungan tertentu di mana produk kami dijalankan. Jika pelanggan belum menambahkan informasi itu kami dapat, dengan satu klik, mengirimkan respons yang memintanya.

Beberapa pemilik produk dan penyedia dukungan khawatir bahwa tanggapan terekam dapat menghilangkan sentuhan pribadi. Josh Emerson dari Clearleft menyuarakan keprihatinan ini dengan mengatakan:

Saya menghabiskan banyak waktu membalas pelanggan dengan solusi yang sangat mirip [...] Saya sering mempertimbangkan untuk mengotomatiskan respons pelanggan dengan cara tertentu, namun saya khawatir dengan menawarkan solusi yang lebih otomatis, kualitas respons kami akan menurun . Saya benci menghubungi perusahaan dan mendapatkan tanggapan umum yang tidak menjawab pertanyaan saya. Saya sangat benci jika email tersebut berpura-pura berasal dari orang sungguhan.



Gambar 11.3 Memilih tanggapan terekam di HelpSpot.

Dia melanjutkan dengan mengatakan, “Saya yakin ada solusi yang lebih baik, seperti memfilter dan menyortir email untuk mempercepat pekerjaan saya, berpotensi mempopulasi respons yang paling mungkin, namun dengan manusia yang memverifikasi bahwa respons tersebut benar sebelum mengirimkannya.”

Menurut saya, menggunakan alat untuk memastikan orang-orang dengan pertanyaan yang sama mendapatkan jawaban yang sama adalah hal yang bermanfaat. Selain menghemat waktu dan mencegah Anda menulis jawaban yang sama berulang kali, Anda juga dapat menyusun jawaban yang jelas dan bagus untuk pertanyaan umum. Idealnya, pertanyaan-pertanyaan semacam ini juga dijawab dalam dokumentasi Anda, tetapi seperti yang kita ketahui, tidak semua pengguna akan membaca dokumentasi tersebut, betapapun jelasnya dokumentasi tersebut.

Seperti Josh, saya tidak ingin mengirimkan tanggapan yang sepenuhnya otomatis kepada orang yang meminta bantuan, namun tanggapan terekam dapat menghemat banyak waktu dalam mengetik balasan yang sama berulang kali. Ketika balasannya memerlukan pencarian beberapa tautan dalam dokumentasi, waktu yang dapat Anda hemat sangat besar selama seminggu.

Sistem yang kami gunakan memungkinkan kami mengedit respons sebelum mengirimkannya, jadi terkadang saya menggunakan respons terekam sebagai titik awal dan juga menambahkan beberapa informasi yang menurut saya relevan dengan pelanggan tertentu. Sistem yang baik harus memungkinkan Anda menggunakan tanggapan template Anda sebagai titik awal, daripada bergantung sepenuhnya pada tanggapan standar untuk setiap pelanggan.

Memilih Produk Helpdesk

Solusi yang tepat bagi Anda harus didorong oleh jenis dukungan yang perlu Anda lakukan. Ada berbagai macam sistem helpdesk, masing-masing mencoba memecahkan masalah dalam memberikan dukungan dengan cara tertentu. Bab ini tidak mencoba untuk meninjau semua sistem ini, karena tinjauan semacam itu akan diproyeksikan melalui lensa yang menurut saya penting. Oleh karena itu, saya menyarankan sebelum Anda mulai mencoba sistem, Anda harus mempertimbangkan jenis dukungan apa yang ingin Anda tawarkan kepada pelanggan Anda; Anda kemudian dapat menilai setiap sistem berdasarkan

persyaratan tersebut. Misalnya, jika Anda berencana menawarkan dukungan melalui telepon, dapatkah tiket tersebut dicatat dalam sistem? Beberapa sistem berasumsi Anda hanya menawarkan dukungan berbasis web. Apakah dukungan media sosial dalam sistem bermanfaat bagi Anda? Apakah ada jenis tanggapan yang perlu Anda berikan yang harus Anda periksa apakah didukung? Sebelum menyiapkan uji coba berbagai sistem, pertimbangkan hal-hal berikut:

1. Metode apa saja (telepon, email, Twitter, Facebook, login ke sistem tiket) yang kita inginkan agar orang-orang menghubungi kita?
2. Apakah permintaan dukungan biasanya berupa pertanyaan dan jawaban singkat, atau apakah Anda mengharapkan rangkaian pesan yang lebih panjang membahas suatu masalah secara bolak-balik?
3. Apakah dukungan harus bersifat publik, swasta, atau gabungan keduanya?
4. Jika Anda menggunakan sistem publik, apakah pelanggan dapat saling berkomentar dan memberi saran?
5. Apakah Anda harus bisa mengumpulkan informasi spesifik saat pelanggan memposting tiket?
6. Apakah Anda harus bisa memverifikasi status pelanggan dengan cara apa pun? Ini penting jika Anda memiliki keterbatasan seputar dukungan.
7. Apakah Anda perlu mengintegrasikan dukungan dengan sistem lain?

Dalam kasus kami, salah satu fitur penting adalah sistem memungkinkan kami memposting contoh kode ke dalam tiket dan tanggapan forum. Sebagian besar dukungan kami melibatkan membantu seseorang dengan templat atau PHP khusus mereka. Beberapa sistem mempunyai dukungan yang sangat buruk untuk melakukan hal ini. Saat kami meminta saran tentang cara mengatasi masalah ini, mereka menyarankan agar semua pelanggan kami menggunakan alat pihak ketiga untuk menampilkan cuplikan kode. Di mata kami, hal ini justru menjadi penghalang tambahan bagi orang-orang yang membutuhkan bantuan. Apa yang mungkin tampak sebagai hal yang sangat kecil menjadi pemecah masalah bagi kami dengan beberapa sistem yang populer.

Kami juga menyarankan agar berhati-hati ketika memiliki sistem yang sepenuhnya publik di mana pelanggan dapat saling membantu. Orang sering kali memposting informasi aman seperti detail koneksi database ke domain publik. Pelanggan tidak selalu menyadari bahwa tiket mereka bersifat publik.

Masalah lain yang kami temukan adalah jika pelanggan membalas pelanggan lain dengan saran yang salah, pengirim pesan asli tidak selalu menyadari bahwa tanggapan tersebut bukanlah dukungan resmi. Saat kami mencoba menciptakan pengalaman dukungan yang baik bagi pelanggan kami, hal ini menjadi perhatian kami. Sistem kami saat ini memiliki forum dan area dukungan tiket terpisah. Kami mendorong permintaan implementasi, masalah CSS, dan permintaan fitur untuk diposting ke forum publik tempat kami dan Percher lainnya dapat merespons. Seringkali, jika permintaan benar-benar berkaitan dengan plugin jQuery pihak ketiga atau editor tertentu seperti Dreamweaver, pelanggan lain mengetahui lebih banyak tentang hal ini daripada kami!

Banyak sistem pendukung memiliki API sehingga Anda dapat menghubungkan sistem lain yang Anda gunakan atau mengembangkan fungsionalitas tambahan yang spesifik untuk kebutuhan Anda. Seperti yang dijelaskan oleh David Goss, yang mendukung situs web e-niaga untuk Thackerays:

Kami berhubungan baik dengan Zendesk. Ini cukup fleksibel sehingga kita dapat mengatur alur kerja dan proses sesuai keinginan kita, namun yang paling berguna adalah API, sehingga kita dapat melakukan banyak hal secara otomatis dari bagian belakang situs web. Sebagai contoh, saya sedang membuat perubahan pada laman "pembayaran gagal" saat ini sehingga jika pembayaran pelanggan gagal, ada tombol untuk mereka yang berlabel "Bantuan Saya tidak tahu mengapa pembayaran saya gagal" dan itu akan menghasilkan tiket dengan semua detail pesanan dan pembayaran mereka terlampir dan kirimkan email kepada mereka untuk memberi tahu mereka bahwa kami sedang memeriksanya untuk mereka.

Menjadi lebih proaktif juga merupakan sesuatu yang sedang saya kerjakan. Sebagian besar sistem helpdesk didasarkan pada premis bahwa pelanggan mempunyai masalah dan melaporkannya kepada perusahaan, namun jika kita melihat masalah sebelum pelanggan mengalaminya, seperti mungkin pengiriman mereka tertunda sehari atau barang yang mereka pesan kehabisan stok, kami ingin membuka tiket atas nama mereka, beri tahu mereka apa yang terjadi dan tanyakan apa yang ingin mereka lakukan, seotomatis mungkin. Dengan API, hal ini sebenarnya akan cukup mudah dilakukan.

Cobalah untuk membuat pengalaman menerima dukungan sesederhana mungkin. Semua sistem helpdesk yang besar melakukan pekerjaan yang hampir sama, dengan fitur dan fokus yang sedikit berbeda. Mereka dapat membantu Anda membuat sistem yang cukup kompleks untuk operasi dukungan Anda. Jika Anda memiliki kebutuhan dukungan yang sangat sederhana maka layanan bantuan lengkap mungkin tampak berlebihan. Saya yakin inilah sebabnya banyak perusahaan hanya mengelola dukungan email di kotak masuk mereka. Namun, terdapat solusi yang lebih kecil di luar sana karena perusahaan berupaya memecahkan masalah dukungan dengan cara baru. Salah satu solusi tersebut datang dari pengembang sistem HelpSpot yang kami gunakan, dan ini menarik karena ini adalah produk baru di ruang yang sama dengan HelpSpot, tetapi dirancang untuk memenuhi kebutuhan akan sistem yang lebih kecil dan sederhana.

Snappy adalah produk yang benar-benar baru, namun saya menyebutkannya di sini karena menurut saya produk ini memiliki banyak potensi bagi siapa pun yang membaca ini dan berpikir bahwa solusi besar tidak diperlukan tetapi ingin memiliki sesuatu untuk membantu mengelola permintaan email. Snappy hanya mengelola tiket email dan tanggapan dengan cara yang modern dan efisien. Pelanggan Anda dapat mengirimi Anda email dan Anda dapat membalas melalui email, tetapi Snappy memecahkan masalah langsung dalam melacak apa yang telah ditanggapi.

Merespon Tiket Di Snappy

Saya pikir solusi seperti Snappy memiliki banyak potensi bagi pengembang produk kecil. Saya melihat ini sebagai solusi yang bagus jika Anda mendukung aplikasi seluler, misalnya, di mana pertanyaannya cenderung cukup mudah. Ian menulis postingan blog

tentang beberapa pemikiran di balik Snappy, yang menarik datang dari seseorang yang juga telah mengembangkan solusi helpdesk yang lebih besar.

Karena tema buku ini menghadap ke masa depan, menurut saya menyertakan sistem pendukung generasi baru adalah hal yang penting dan saya akan sangat tertarik untuk melihat bagaimana Snappy dan sistem lain yang tidak menggunakan layanan bantuan tradisional berkembang. Menarik juga bahwa sistem baru kembali ke email pada intinya. Email jelas tidak mati bagi sebagian besar pelanggan, dan sering kali merupakan cara termudah bagi mereka untuk berbicara dengan kami.

Menggunakan Media Sosial Untuk Mendukung Produk Anda

Jika Anda membuat pelanggan tidak senang di dunia fisik, mereka masing-masing mungkin akan memberi tahu 6 temannya. Jika Anda membuat pelanggan tidak senang di Internet, mereka masing-masing dapat memberi tahu 6.000 teman.

—Jeff Bezos

Perusahaan-perusahaan besar telah memanfaatkan Twitter dan memiliki perwakilan dukungan yang memantau dan membantu orang-orang di sana adalah cara yang bagus untuk membalikkan pengalaman negatif pelanggan dengan cepat. Ketika perwakilan benar-benar mempunyai kekuatan untuk membantu di Twitter, maka mereka dapat, di hadapan pelanggan lain dan calon pelanggan, memecahkan masalah dan mengubah pelanggan yang tidak senang untuk melampiaskan rasa frustrasinya kepada pengikut Twitter menjadi pelanggan yang bahagia.

Twitter adalah tempat yang wajar bagi kami untuk membicarakan Perch dan berbicara dengan pelanggan serta calon pelanggan kami karena saya dan Drew sudah aktif di sana. Kami menggunakan akun @grabaperch untuk memberikan informasi terbaru kepada orang-orang tentang rilis perangkat lunak baru atau tutorial dan dokumentasi serta membalas pertanyaan. Kami juga memiliki halaman Facebook dan menemukan bahwa meskipun orang-orang sering menyukai dan membagikan postingan kami, mereka cenderung tidak mengobrol atau bertanya di sana seperti yang mereka lakukan di Twitter.

Kami melihat Twitter sebagai cara yang sangat penting untuk berkomunikasi dengan pelanggan. Namun, lembaga ini tidak selalu merupakan tempat terbaik untuk memberikan dukungan. Bagi kami, menanggapi tiket dukungan sering kali melibatkan pengeposan kode. Hal ini tidak mungkin dilakukan dengan 140 karakter yang diperbolehkan dalam pesan Twitter. Terkadang, yang dibutuhkan orang hanyalah tautan ke sesuatu di dokumentasi kami, dan itu bisa ditangani di Twitter. Jadi itu sangat tergantung pada jenis respons apa yang diperlukan.

Twitter bisa menjadi tempat yang ideal untuk jawaban cepat, seperti mengarahkan seseorang ke tempat yang tepat dalam dokumentasi. Memberikan dukungan teknis di Twitter atau Facebook mungkin tidak tepat. Dengan memastikan bahwa orang-orang yang membicarakan produk Anda atau membutuhkan dukungan mendapat tanggapan, Anda menunjukkan bahwa Anda ada untuk membantu. Selain itu, mendukung orang-orang dengan

cara yang nyata akan mempromosikan produk Anda dan dapat membantu mendorong rasa kebersamaan.

Beberapa produk helpdesk memiliki beberapa bentuk integrasi media sosial. Misalnya, Zendesk dapat mengubah tweet menjadi tiket dukungan, yang berguna untuk memantau jalur yang diambil oleh suatu permintaan. Kami belum merasakan perlunya integrasi yang erat antara layanan bantuan dan aktivitas media sosial kami, meskipun banyak perusahaan yang menganggap integrasi semacam ini bermanfaat, khususnya ketika menangani banyak dukungan yang berasal dari saluran media sosial.

37signals menggunakan Twitter secara ekstensif saat mendukung produknya, dan meskipun sistem layanan bantuan yang dipilihnya memiliki integrasi Twitter, tim mendapati bahwa sistem tersebut tidak berhasil bagi mereka:

Sejak kami mulai serius menggunakan Twitter, kami sebagian besar menggunakan fungsionalitas bawaan Twitter yang disediakan oleh alat dukungan kami (Desk.com). Ketika saya bertanya kepada tim bagaimana cara kerjanya beberapa bulan yang lalu, reaksi umumnya hangat. Konsensusnya adalah meskipun ia menyelesaikan pekerjaannya, penggunaannya agak lambat, dan banyaknya jumlah retweet dan tautan ke postingan SvN yang tercampur membuat sulit untuk mendapatkan jawaban segera bagi orang-orang yang memiliki pertanyaan mendesak. Sebagian besar tim menggunakannya, tapi tidak ada yang senang dengan hal itu.

— Noah Lorang, 37 sinyal

Serangkaian postingan tentang alat Twitter yang digunakan oleh 37signals memberikan gambaran menarik tentang bagaimana perusahaan dapat mengambil sebagian kecil dari operasi dukungan mereka dan benar-benar menyesuaikan pengalaman untuk staf dukungan dan pelanggan. Hal ini juga menunjukkan bahwa meskipun Anda telah memilih sistem dukungan tertentu, jika sebagian dari sistem tersebut tidak sesuai dengan kebutuhan Anda, Anda tidak perlu sepenuhnya beralih ke sistem lain. Sebaliknya, Anda dapat menggunakan pendekatan alternatif di bidang tersebut.

Cara lain untuk menggunakan Twitter adalah bersikap proaktif dalam memantau apa yang dikatakan orang tentang produk Anda dan melakukan tindakan jika diperlukan. Kami telah menyimpan penelusuran yang disiapkan untuk istilah seperti Perch CMS dan terkadang akan menghubungi seseorang yang membicarakan kami, meskipun mereka belum langsung membalas @grabaperch. Jika digunakan dengan hati-hati, hal ini bisa menjadi cara yang baik untuk mendorong seseorang agar memberikan dukungan, dibandingkan sekadar meminta nasihat dari teman dan rekannya.

Ini juga bekerja sangat baik bagi kami dalam hal pra-penjualan karena kami dapat memastikan pertanyaan mereka terjawab. Namun, Anda harus berhati-hati agar tidak dianggap sebagai spam saat melakukan ini; kita cenderung hanya mengatakan, “Beri tahu kami jika Anda memiliki pertanyaan” dan kemudian menyerahkan kepada mereka untuk menjawab kami jika mereka mau.

Sebuah Pemikiran Terakhir

Ingatlah selalu bahwa dukungan adalah tentang membantu orang-orang memiliki pengalaman hebat dengan produk atau layanan Anda. Dukungan adalah bagian penting dari produk yang sukses. Ini dapat bermanfaat bagi produk, bahkan menjadi sumber riset pelanggan dan cara beriklan. Tujuan akhir dari dukungan pelanggan, tentu saja, adalah untuk membantu pelanggan Anda mendapatkan hasil maksimal dari produk atau layanan Anda, memecahkan masalah mereka dengan cepat dan membuat mereka kembali melakukan apa pun yang mereka lakukan saat membeli produk Anda.

Anda harus menjadi orang baik untuk melakukan dukungan dengan baik. Anda harus mau membantu orang. Segala sesuatu yang lain cara menyusun email Anda, cara menggunakan perangkat lunak pusat bantuan, pengetahuan teknis semuanya dapat diajarkan. Anda hanya perlu cukup peduli untuk melakukannya dengan benar, dan tetap peduli, bahkan ketika suasana hati Anda sedang buruk, atau server sedang terbakar.

— Jim Mackenzie, 37 sinyal

Jika Anda mendekati dukungan dengan mempertimbangkan kebutuhan pelanggan, bahkan ketika Anda tidak memiliki jawaban yang tepat untuk mereka, Anda mungkin mendapati bahwa hal itu mulai menjadi bagian yang menyenangkan dari apa yang Anda lakukan, daripada tugas yang diyakini banyak orang. dia.

Tindakan Berikutnya untuk Dukungan Ramah Masa Depan

Saya telah menyusun daftar hal yang harus dilakukan dan beberapa bacaan lebih lanjut mengenai tema-tema dalam bab ini.

Untuk Produk atau Layanan Baru Hanya Menjelajahi Cara Melakukan Dukungan

1. Pikirkan jenis dukungan yang akan Anda tawarkan. Apakah sebagian besar akan berupa balasan cepat atau Anda perlu memberikan bantuan teknis yang panjang? Dari mana datangnya permintaan: email, telepon, media sosial, dan di mana saja?
2. Pikirkan siapa yang akan memberikan dukungan saat ini dan dalam jangka pendek hingga menengah. Apa yang perlu Anda lakukan jika produk atau layanan menjadi sukses dalam semalam? Dapatkah Anda memulai setidaknya sebagian dari infrastruktur tersebut?
3. Buat daftar persyaratan untuk dukungan menggunakan pemikiran saya di bagian 'Memilih Produk Helpdesk'.
4. Gunakan informasi yang dikumpulkan di atas untuk meneliti solusi dukungan dan membuat daftar solusi yang sesuai. Ini akan menjadi solusi yang dapat Anda demokan secara menyeluruh sebelum menentukan pilihan.
5. Tulis dokumen yang merinci nada apa yang ingin Anda gunakan dalam dukungan, dan pastikan semua orang yang melakukan dukungan dapat mengaksesnya.

Untuk Bisnis Mapan yang Ingin Meningkatkan Penawaran Dukungannya

1. Buatlah daftar lima sampai sepuluh isu teratas yang mendapat dukungan.
2. Dari daftar tersebut, dapatkah Anda mengidentifikasi sesuatu yang tidak lagi menjadi masalah dengan melakukan beberapa perubahan pada produk atau layanan Anda? Jika ya, lakukan perubahan itu.

3. Dalam daftar tersebut adakah yang bisa dijelaskan dengan lebih baik dalam dokumentasi atau materi pendukung Anda? Jika ya, lakukan perubahan itu.
4. Identifikasi lima permintaan dukungan yang benar-benar merupakan permintaan fitur, baik permintaan eksplisit atau area di mana Anda harus memberi tahu pelanggan bahwa produk Anda tidak memenuhi persyaratan tertentu. Untuk setiap permintaan, pertimbangkan apakah permintaan tersebut dapat: merusak kasus penggunaan inti Anda dan tidak boleh diterapkan; dapat diimplementasikan dengan cepat; atau dapat ditambahkan ke rencana jangka panjang produk tersebut. Jangan lupa untuk memberi tahu pelanggan ketika Anda telah menerapkan saran mereka.
5. Jika Anda memiliki halaman FAQ, apakah ada masalah yang dapat didokumentasikan atau ditangani dalam produk itu sendiri agar masalah tersebut tidak sering ditanyakan?
6. Tinjau kembali proses di atas secara rutin!
7. Apakah Anda menggunakan sistem pendukung Anda secara maksimal? Luangkan waktu untuk melihat fitur apa pun yang tidak Anda gunakan. Dapatkah Anda meningkatkan penggunaan sistem, mungkin dengan memanfaatkan API untuk menyesuaikan pengalaman pelanggan Anda?

BAB 12
DESAIN ORANG

Tujuan Bab ini

Kisah saya, meskipun sedikit ekstrem, bukanlah sesuatu yang unik. Kita semua pernah berada di sana dalam satu kapasitas atau lainnya: rekan kerja yang sulit berurusan dengan, bos yang tidak kompeten, klien yang mengelola mikro, kepemimpinan yang tampaknya gila, desain oleh komite dan masih banyak lagi.

Ada banyak sekali cara untuk mengiris dan menganalisis cerita saya. Beberapa orang akan berempati dan mengungkapkan kemarahan atas lingkungan kerja yang tidak berfungsi. Beberapa orang akan menuding kepemimpinan yang buruk. Beberapa pihak akan fokus pada dugaan tidak berharganya manajemen menengah. Beberapa orang akan mengaitkan kegagalan tersebut dengan proses mungkin Anda seharusnya menggunakan metodologi Agile. Dan beberapa orang akan berpendapat bahwa saya sebenarnya adalah pelaku sebenarnya dan seharusnya dipecat; lagi pula, sayalah yang memimpin proyek itu.

Tak satu pun dari analisis ini yang salah. Faktanya, mereka baik-baik saja. Dan masih banyak lagi yang belum saya sebutkan yang juga berlaku. Namun tidak peduli seberapa banyak kita menganalisis situasi seperti itu dalam postmortem kita dan bersumpah untuk menghindarinya dengan cara apa pun di masa depan, mau tidak mau kita sering kali mendapati diri kita berada dalam kesulitan serupa. Meskipun kita telah mengumpulkan banyak sekali taktik dan pola yang sering membantu kita menghindari situasi disfungsional, kita kurang memahami akar masalahnya dan, akibatnya, sangat sedikit kendali atas nasib kita.

Setelah bertahun-tahun berusaha melewati situasi-situasi disfungsional yang tidak ingin saya ingat, saya dengan tegas menetapkan sebuah keyakinan (sebenarnya, ini adalah satu-satunya keyakinan yang saya anggap suci lagi): hampir setiap masalah yang kita hadapi di tempat kerja (dan bermain) dimulai dan diakhiri dengan satu orang atau lebih.

Sementara kami menjalankan proses perbaikan, mempekerjakan ahli, meminta masukan dari pengguna, meningkatkan jumlah peninjauan kode per minggu, mengubah metodologi pemrograman, membuat lebih banyak mock-up, menjelaskan strategi kami kepada pemangku kepentingan, dan menerapkannya seribu perbaikan darurat lainnya, solusi sebenarnya terus luput dari perhatian kita. Kita adalah makhluk yang memiliki kebiasaan, kita hanya menutup mata dan berenang lebih keras ke hulu sampai kita mendapati diri kita kelelahan dan letih, siap untuk berhenti dari pekerjaan kita. Namun seperti yang pernah dikatakan oleh orang bijak, “Definisi kegilaan adalah melakukan hal yang sama berulang kali dan mengharapkan hasil yang berbeda.”

Tujuan bab ini sederhana: memperkenalkan Anda pada manusia sebagai pusat dari setiap keberhasilan atau kegagalan dalam hidup kita. Tapi tidak dengan cara yang melelahkan, kita semua bersalah karena bersimpati dan curhat di Twitter. Atau cara kami mempublikasikan postingan blog tentang beban birokrasi yang merupakan hambatan sebenarnya bagi inovasi. Atau bahkan kita menulis artikel, bab, dan buku yang menyebarkan praktik terbaik untuk

menangani beban mati tersebut. Kami telah melakukan semuanya sebelumnya, dan kami pasti akan melakukannya lagi. Namun saat ini, mari kita tolak kedok kegilaan. Mari kita berhenti mengoleskan lipstik pada babi, dan mencari tahu mengapa babi itu sangat jelek. Artinya, mari kita bicara tentang akar masalahnya, bukan gejalanya.

Terlalu Banyak Juru Masak tidak Merusak Kaldu

Beberapa tahun yang lalu saya kebetulan bertanggung jawab atas desain ulang dan konsolidasi serangkaian situs komunitas pengembang yang sangat populer, sebuah proyek yang kami sebut Project Unify. Kami menggabungkan lima situs berbeda, yang masing-masing melayani target audiens berbeda, dan dijalankan oleh tim internal berbeda. Bersama-sama, situs-situs tersebut menyajikan ribuan jenis media unik: mulai dari video HD hingga postingan blog pendek. Beberapa mengatur kontennya dalam bentuk pertunjukan; yang lain menampilkan konten streaming langsung. Media hadir dalam berbagai bentuk, format dan ukuran. Nada visual dan arsitektur informasi setiap situs web berbeda. Secara umum, ini merupakan desain ulang yang rumit.

Lebih buruk lagi, ada banyak sekali pemangku kepentingan di seluruh perusahaan yang akan terlibat dalam proyek ini: pembawa acara dari berbagai acara, pengembang yang bekerja di berbagai situs, pendiri situs, manajer, dan bahkan beberapa eksekutif. . Dan kebanyakan dari mereka tidak terlalu senang dengan adanya proyek ini, karena hal ini berarti kehidupan mereka sehari-hari akan segera berubah. Belum lagi, mereka menyukai keadaannya.

Saya telah mencoba membatasi diri pada proyek-proyek kecil dengan lebih sedikit pemangku kepentingan setelah pengalaman saya dengan Prakash dan tim, dan ini adalah proyek besar pertama saya sejak saat itu. Saya gugup - lagipula, berenang bersama hiu dan keluar hidup-hidup bukanlah indikator yang baik untuk mengetahui kemampuan Anda bertahan dalam ekspedisi berikutnya. Tapi saya juga bersemangat. Berkat rekomendasi buku awal dari seorang teman, saya akhirnya tenggelam dalam dunia perilaku manusia - psikologi kognitif, ekonomi perilaku, ilmu saraf, antropologi, biologi evolusi, dan sebagainya — dan telah mengumpulkan cukup pengetahuan untuk menjamin eksperimen dengan ukuran sampel yang lebih besar.

Project Unify adalah peluang bagus untuk karier saya, namun lebih baik lagi untuk menguji pengetahuan baru saya.

12.1 EFEK PSIKOLOGI SISTEM ADIL

Dalam Sway: The Irresistible Pull of Irrational Behavior, Ori dan Rom Brofman menulis, “Sekelompok peneliti meminta ratusan penjahat dari Baltimore, Detroit dan Phoenix untuk mengisi survei. Bagian pertama dari survei ini terdiri dari pertanyaan faktual, seperti sifat hukuman mereka dan lamanya hukuman penjara. Pada bagian kedua, survei beralih ke pertanyaan mengenai persepsi keadilan: Bagaimana Anda diperlakukan? Bagaimana Anda menyukai hakimnya? Apakah para pengacara itu baik padamu?” Para peneliti mencoba menyimpulkan faktor-faktor apa saja yang mempengaruhi persepsi narapidana terhadap keadilan sistem peradilan.

Para peneliti menemukan bahwa responden memberikan bobot yang sama besarnya pada proses hukum dan pada hasilnya. “Salah satu faktor yang paling membebani responden adalah berapa banyak waktu yang dihabiskan pengacara mereka bersama mereka. Semakin banyak waktu yang dia habiskan bersama mereka, semakin puas responden dengan hasil akhirnya,” tulis para penulis. Hal ini mengejutkan karena ada yang mengira bahwa narapidana yang dijatuhi hukuman lebih lama setelah menghabiskan banyak waktu dengan pengacaranya akan merasa lebih tidak puas. Namun yang terjadi justru sebaliknya. “Meskipun hasilnya mungkin sama, ketika kami tidak menyuarakan keprihatinan kami, kami merasakan keadilan keseluruhan dari pengalaman tersebut dengan cara yang berbeda,” para penulis menyimpulkan.

Desainer cenderung bersembunyi dari pemangku kepentingannya. Masing-masing dari kita mempunyai alasan masing-masing, namun sebagian besar didasarkan pada rasa takut diberi tahu cara merancang sesuatu. Namun para peneliti telah berkali-kali menemukan bahwa memenuhi kebutuhan seseorang untuk didengarkan mempunyai pengaruh yang lebih besar terhadap persepsi mereka terhadap hasil suatu situasi dibandingkan hasil sebenarnya.4 Dan jika Anda cerdas dalam hal ini, Anda dapat memengaruhi pemangku kepentingan agar bekerja sama dan berkreasi. desain yang Anda yakini tepat untuk proyek yang sedang dikerjakan. Di Project Unify, saya memaksakan diri untuk menyediakan waktu untuk percakapan empat mata dengan sekitar dua puluh pemangku kepentingan. Saya ingin mereka bertemu dan mengenal saya. Pertemuan tersebut berkisar dari makan siang bersama hingga membiarkan para pengembang membingkai visi mereka tentang situs web di papan tulis mereka. Saya selalu memulai rapat dengan, “Jadi, beri tahu saya menurut Anda apa yang harus kita lakukan untuk Project Unify?”

Ada yang curhat soal politik dan birokrasi (saya ikut). Beberapa orang sangat tidak setuju dengan keberadaan proyek ini (saya berempati dan mengingatkan mereka tentang tingkat gaji kami). Yang lain merasa sudah waktunya bagi seorang desainer profesional untuk mengambil tindakan ini (saya menyembunyikan gejala sindrom penipu5 saya). Beberapa fokus pada aspek tertentu yang benar-benar mereka pedulikan (saya mencatat). Dan ada juga yang senang keluar untuk minum kopi selama jam kerja (tipe orang saya). Itu adalah minggu yang intens.

Namun pada minggu berikutnya, warna suara Project Unify menjadi sangat positif. Para pemangku kepentingan sangat optimis, bahkan bersemangat untuk meluncurkan desain ulang yang kontroversial tersebut. Perubahan sikap jauh melampaui ekspektasi saya, sedemikian rupa sehingga pertemuan informal telah menjadi bagian tak terpisahkan dari repertoar desainer saya.

Banyak sekali tulisan tentang wawancara pemangku kepentingan di Internet. Saya yakin Anda akan dapat mencari artikel tersebut. Namun sebelum Anda mulai membuat daftar pertanyaan wawancara tentang strategi merek, kriteria keberhasilan, dan kepribadian pengguna, pertimbangkan sejenak bahwa meskipun kami memerlukan informasi dari pemangku kepentingan, mereka juga memerlukan keamanan dari kami. Mereka perlu didengarkan, dan kebutuhan ini, jika tidak dipenuhi, akan sangat membahayakan proyek yang

ada. Dan cara terburuk untuk memperlakukan seseorang yang berharap didengarkan adalah dengan membawa papan klip dan daftar periksa. Akan ada waktu untuk itu nanti. Pertama, fokuslah pada hal yang penting: hal-hal kecil.

12.2 MENYATAKAN PERILAKU YANG BAIK DALAM TINJAUAN DESAIN

Membuat pemangku kepentingan merasa optimis terhadap suatu proyek adalah satu hal, namun menerjemahkan optimisme tersebut menjadi umpan balik yang bermanfaat dan menguntungkan adalah hal lain.

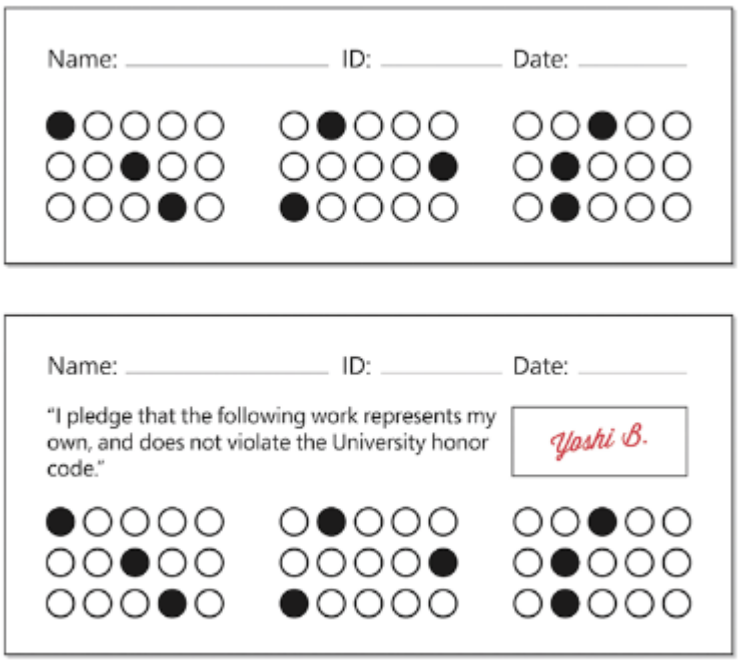
Ada konsep menarik dalam psikologi yang dikenal sebagai penahan. Anchoring adalah sebuah fenomena psikologis di mana manusia sangat bergantung pada informasi pertama yang diberikan kepada mereka (dikenal sebagai jangkar) dalam mengambil keputusan selanjutnya. Dan Ariely, ekonom perilaku dan penulis beberapa buku paling menarik tentang perilaku manusia, memberikan contoh mendasar tentang penahan dalam buku pertamanya yang Dapat Diprediksi Irasional: Kekuatan Tersembunyi yang Membentuk Keputusan Kita: “Beberapa dekade yang lalu, naturalis Konrad Lorenz menemukan bahwa anak angsa, setelah keluar dari telurnya, akan menempel pada benda bergerak pertama yang mereka ambil. bertemu.

Lorenz mengetahui hal ini karena dalam satu percobaan dia menjadi hal pertama yang mereka lihat, dan mereka mengikutinya dengan setia sejak saat itu hingga masa remaja.” Selama beberapa dekade terakhir, para peneliti telah mengkonfirmasi peran jangkar di semua lapisan masyarakat melalui temuan yang tampaknya aneh: kita lebih cenderung menikah dengan orang yang namanya dimulai dengan huruf pertama sebagai milik kita, memilih produk yang nama mereknya memiliki tiga huruf pertama. surat dengan milik kita sendiri, memberikan ulasan positif kepada orang-orang yang memiliki tanggal lahir yang sama dengan kita, dan masih banyak lagi.8 Faktanya, dampak dari penjangkaran bahkan meluas hingga ke ranah moral seperti yang ditunjukkan Ariely dalam buku terbarunya, The (Honest) Truth About Dishonesty: How We Berbohong kepada Semua Orang Terutama Diri Kita Sendiri. Ariely melakukan penelitian yang menunjukkan bahwa Anda hampir dapat menghilangkan kecurangan dalam ujian yaitu, Anda benar-benar dapat membuat orang lebih jujur hanya dengan meminta siswa menandatangani kode kehormatan sederhana tepat sebelum mereka mengikuti ujian.

Saat kembali ke Project Unify, saya bertanya-tanya dalam hati apakah saya dapat menggunakan kekuatan penahan untuk menghimpun kelompok pemangku kepentingan yang sangat besar dan beragam melalui proses peninjauan desain tanpa drama dan konflik seperti yang kita harapkan dalam situasi seperti itu. Apakah ada sesuatu yang dapat saya lakukan atau katakan yang akan membuat individu dalam kelompok fokus untuk memberikan umpan balik yang logis daripada bereaksi berdasarkan emosi yang secara alami akan mereka (dan kita semua) rasakan saat melihat warna, bentuk, dan pola baru yang akan membuat mereka merasa lebih baik? menjadi bagian dari desain baru? Saya pikir ini pantas untuk dicoba dan menghasilkan solusi sederhana.

Tepat sebelum tim desain saya mulai mengerjakan desain baru, saya mengirim email ke seluruh kelompok pemangku kepentingan. Email tersebut disusun sebagai korespondensi biasa yang menjelaskan proses desain, namun di dalamnya tersembunyi beberapa jangkar.

Mari kita telusuri email ini bersama-sama.



Gambar 12.1 Lembar gelembung ujian reguler (atas) dan lembar gelembung yang dimodifikasi dengan slot tanda tangan kode kehormatan (bawah).

Email: Anda Sudah Mengantuk, Sangat Mengantuk

Saya telah berusaha sebaik mungkin untuk mereproduksi email dalam bentuk aslinya, namun mau tidak mau, beberapa nama dan detail perlu diubah untuk melindungi privasi semua yang terlibat. Meskipun demikian, tidak ada satupun perubahan yang dapat mengalihkan perhatian dari diskusi yang sedang berlangsung.

From: Nishant Kothary
Sent: Friday, May 01, 5:48 PM
Subject: How to provide feedback

Folks –
Apologies in advance for the long email; if you think I've forgotten someone, please email me instead of adding them yourself. My intention is to keep the group limited to the core stakeholders. I will be soliciting feedback from others as needed.

Perhatikan bahwa email dikirim setelah jam kerja pada hari Jumat. Ini mungkin bukan pilihan untuk semua proyek, tapi menurut saya ini meningkatkan kemungkinan (tentu saja

secara tidak ilmiah) bahwa penerima Anda akan membaca email dalam kerangka berpikir positif (hal pertama pada hari Senin sebelum stres minggu ini dimulai. di dalam).

Saya juga mencatat di sini bahwa pemangku kepentingan didorong untuk tidak menunjuk orang lain di perusahaan sebagai pemangku kepentingan. Keputusan itu perlu dibuat oleh saya. Namun, sebagai imbalannya, mereka menjadi bagian dari kelompok pemangku kepentingan inti yang terbatas. Perdagangan yang adil.

> So, yes, hello again! We are entering the most critical design phase of the project — information architecture — and will be delivering wireframes soon. We're running a really tight schedule for the deliverables and this includes feedback loops, too. In this email, I'd like to get everyone on the same page about what's coming and how to remain involved. To that end, I've compiled a set of FAQ's, milestones and some instructions on how to provide feedback when the time is here.

Ada satu elemen kecil yang menjadi penahan di sini - "dan ini termasuk putaran umpan balik". Hal ini menetapkan harapan bahwa tindakan memberikan umpan balik akan terikat pada jadwal yang ketat (seperti tindakan merancang situs sebenarnya).

> **FAQ's**
>
> ***Will you be reviewing the wireframes with each of us?*** Nope. Doing reviews with individuals or even on a per team (John's team, Erics's team, etc.) basis is not feasible and also unnecessary. Instead, I'm going to share the wireframes with you electronically. They will also be posted on the Project Unify wall (I request that you stand in front of it to capture your feedback instead of printing the set of wireframes). I expect legible feedback through email. More on that towards the end.
>
> ***May I drop by or just schedule a call with you to provide the feedback?*** No and no. :) There are quite a few of you, so it's not very scalable. If you deem the feedback important enough, then I would expect that it'd be worth your time to take 30 mins to type up a clean bulleted list of comments.

Selanjutnya, bagian FAQ seperti yang ditunjukkan pada halaman sebelumnya: Sekali lagi, saya menekankan keadilan. Kesimpulannya adalah kita semua adalah bagian dari tim dan kita semua harus memberikan kontribusi yang setara.

> ***What if I don't understand some part of the wireframe?*** I think this is going to be unlikely; a wireframe, after all, is simply a skeletal representation of the site with the color, fancy graphics, and real typography pulled out. But, there are always exceptions, and in those cases, I suggest you send me a quick email and I'll try to get back to you.

Ini adalah pesan dukungan bagi siapa pun yang merasa cemas dengan proses tersebut. Ada jalan keluar, tetapi ada kendala (saya akan membantu Anda melalui email).

If I don't hear back from you, should I be offended? No. With the volume of email that I expect to receive, I won't be able to respond to everything and everyone. You are going to have to trust that I've heard you and will do the needful to make your feedback actionable (and in many cases, I will be discarding it, or tabling it for future releases so we can hit our ship dates).

What if I don't see a feature that I really wanted in the site (and I even told you about it)? Put it in your feedback email (discussed below), but bear in mind a couple of things, (a) it may have been tabled for future releases because this release is about getting a solid foundation in place and not all the bells and whistles, (b) it was a cool feature/idea but not necessarily worth investing in for the business.

Do I have to do this and will it reflect badly upon me if I have nothing to say to you? Heck no. You've heard of the saying, "Too many cooks spoil the broth", right? You will actually be increasing our chances of success by not chiming in (and don't take that to mean, "Your feedback is stupid anyway, so bugger off"; I am just alluding to the fact that there is always a higher probability of finding sane solutions when you reduce the number of inmates running the asylum even if it could come at the regrettable cost of some valuable features and interactions).

If I don't provide feedback, will this project fail? No. It's my (and the design team's) job to catch all the UX issues. If we're doing our job well, chances are that we will not be surprised by any of your emails (either we'll have already caught what you're pointing out and accounted for it, or we'll have discarded/tabled it with some objective-sounding justification).

Ada beberapa jangkar yang tersembunyi dalam bagian ini: meninjau desain satu per satu tidaklah mungkin; memberikan umpan balik tertulis itu penting; desain seringkali subjektif; gambaran besarnya lebih penting daripada ciri-ciri individual; "percayalah kepadaku"; dan banyak lagi. Selanjutnya, bagian pencapaian, tidak hanya untuk berbagai tahap desain, namun juga untuk memberikan umpan balik.

MILESTONES

Note that the above schedule is subject to change because of how tight things are right now. Also, I'll provide more milestones moving forward based on how this process goes. Mark your calendars!

- *Friday, May 8* — First round of wireframes will be distributed via email and posted on the Project Unify wall.
- *Monday, May 11 by noon* — Feedback due via email (described below)
- *Wed, May 13* — Second round of wireframes will be distributed via email and posted on the Project Unify wall.
- *Thurs, May 14 by end of day* — Feedback due via email

Mengarahkan pemangku kepentingan pada pencapaian dalam memberikan umpan balik bukanlah sebuah konsep baru, namun konsep ini sering kali kita abaikan karena pada umumnya kita tidak tahu bagaimana cara memberikan argumen mengenai hal tersebut.

Argumen terbaik umumnya adalah aritmatika sederhana. Garis waktu desainnya sendiri sangat ketat untuk Project Unify. Tim desain memiliki waktu sembilan hari (termasuk akhir pekan; ini jelas merupakan proyek minyak tengah malam) untuk memproduksi gambar rangka untuk lokasi baru. Sebagai perbandingan, para pemangku kepentingan mempunyai waktu tiga hari penuh untuk memberikan masukan. Secara keseluruhan, ini terbukti menjadi jangkar yang sangat baik, dan juga cara untuk mengontrol ruang lingkup.

Terakhir, bagian terpenting: bagaimana memberikan umpan balik. Meskipun sebagian besar bagian ini berorientasi pada proses, poin terakhirnya "Masukan Anda harus dapat ditindaklanjuti, rasional, dan masuk akal. Dengan kata lain, fokuslah pada hal-hal spesifik dari wireframe dan bukan hal-hal yang angan-angan/bagus untuk dimiliki" menonjol. Meskipun hal ini tidak setara dengan tanda tangan yang jelas (ingat, Ariely meminta subjek secara fisik menandatangani kode kehormatan sebelum mereka melanjutkan untuk mengikuti tes), saya berharap poin terakhir ini, digabungkan dengan ucapan terima kasih tertulis dan lisan dari semua pemangku kepentingan di bawah ini. hari akan mendorong para pemangku kepentingan untuk berperilaku lebih rasional.

Jika dipikir-pikir, solusi yang lebih baik adalah meminta para pemangku kepentingan untuk merespons secara individu dengan semacam tanda tangan digital untuk menjamin rasionalitas mereka; beberapa perangkat lunak email seperti Microsoft Outlook memungkinkan Anda menerapkan respons biner ("Saya menerima" atau "Saya tidak menerima") dari setiap penerima setelah menerima email. Selain itu, jika laporan Brafman bersaudara mengenai persepsi narapidana terhadap proses hukum dapat diterapkan secara lebih luas, maka email pembuka ini tidak hanya akan berhasil menginspirasi partisipasi yang bermanfaat dalam proses tersebut, namun para pemangku kepentingan juga akan merasa positif terhadap keseluruhan pengalaman tersebut, apa pun latar belakangnya. hasil.

Jadi pertanyaannya adalah: apakah itu berhasil?

Hasil

Sederhananya, ya. Namun yang membuat saya kagum adalah prosesnya dan bukan hasilnya. Tentu, kami berhasil mendesain website dalam tiga minggu. Dengan ukuran yang masuk akal, hal ini hampir mustahil mengingat cakupannya. Namun hal yang menurut saya paling menarik adalah para pemangku kepentingan tidak hanya memberikan umpan balik tepat waktu, namun juga menawarkan wawasan luar biasa dengan cara yang belum pernah saya saksikan sebelumnya dalam karier saya.

Banyak pemangku kepentingan melangkah lebih jauh dalam menjelaskan alasan mereka ketika mereka tidak setuju dengan pilihan tim desain. Beberapa memberikan informasi sejarah yang berguna untuk membantu kami membuat keputusan desain yang tepat. Yang lain menggunakan keahlian mendalam mereka untuk memberikan konteks bagi perubahan yang mereka sarankan. Pengembang secara khusus berfokus pada kelayakan penerapan aspek-aspek tertentu dari desain yang diusulkan. Dan sejumlah besar pemangku kepentingan secara sukarela menarik diri dari proses tersebut. Anda mendapat kesan bahwa setiap orang tidak hanya berinvestasi dalam proyek, namun juga fokus untuk memajukan proses. Terutama ketika masa-masa menjadi sedikit sulit.

Misalnya, pada tahap akhir proyek, arahan seni khususnya, warna teal yang merupakan bagian dari dewan seni menemui beberapa penolakan. Dalam kasus seperti ini, para pemangku kepentingan (bahkan desainer terbaik sekalipun) sering kali sangat berfokus pada perasaan pribadi mereka terhadap karya seni tersebut, seperti dalam kalimat "Saya benar-benar tidak menyukai ini!" Namun, tidak ada cara untuk benar-benar mempertimbangkan masukan tersebut, dan situasi seperti itu biasanya cenderung menimbulkan perdebatan sengit. Anggota grup Project Unify, setelah mengakui ketidaksukaan mereka terhadap warna tersebut, fokus untuk memberikan umpan balik yang dapat ditindaklanjuti dan rasional: mulai dari saran penyesuaian warna, hingga pemutusan hubungan dengan merek perusahaan. Para pemangku kepentingan bertindak seperti desainer ideal.

Sekarang, Anda mungkin bertanya-tanya apa inti dari email tersebut apa yang dapat Anda ambil dan terapkan pada email awal proyek berikutnya yang Anda kirim? Jelas sekali ada hal-hal yang telah kita pelajari dari buku-buku dasar manajemen proyek; misalnya, dengan menguraikan prosesnya dengan jelas, saya mengurangi ketidakpastian, memberikan langkah-langkah selanjutnya yang dapat ditindaklanjuti, dan pada gilirannya, meningkatkan peluang keberhasilan proyek ini. Meskipun tidak mungkin untuk mengetahui secara pasti apa yang benar-benar membantu dan apa yang memiliki efek netral pada proyek khusus ini, berikut adalah prinsip-prinsip inti yang saya ambil secara pribadi:

1. Penahan

Email tersebut menetapkan ekspektasi tentang proses desain dengan tujuan mendorong perilaku yang diinginkan di antara para peserta. Ia melakukannya dengan menggunakan kekuatan penahan berulang kali.

2. Keadilan

Proses desain yang dibingkai dalam email menggunakan apa yang kita ketahui tentang pengaruh sistem yang adil dalam menginspirasi perilaku positif manusia.

3. Kode Kehormatan

Ungkapan dan koreografi email memanfaatkan jangkar yang secara khusus mempengaruhi moralitas manusia dalam jangka pendek.

4. Koordinasi Sosial

Email tersebut menyatukan tim dengan membuat gaya umpan balik menjadi sesuatu yang sama-sama dimiliki oleh semua pemangku kepentingan.

5. Norma Sosial

Email tersebut menghapus saya dari puncak hierarki akuntabilitas dan malah menggantikan posisi saya dengan proyek. Jika ada satu orang yang mengabaikan petunjuk tersebut, merekalah yang akan merugikan proyek, bukan saya.

Perlu disebutkan bahwa perjalanan ini tidak sepenuhnya mulus. Ada beberapa pemangku kepentingan yang memberikan masukan jauh melampaui tenggat waktu mereka. Ketika saya menolak tanggapan mereka dengan alasan bahwa waktu telah berlalu, mereka meneruskannya ke sponsor eksekutif (dan pendiri asli situs tersebut). Untungnya, berkat pengalaman masa lalu, saya telah memberikan perlindungan udara (dukungan dari influencer

paling senior di proyek) sebelum proyek dimulai. Dia menolak eskalasi seperti itu, dan pada gilirannya menambahkan kekuatan psikologis yang kuat pada momentum ke depan Project Unify.

Pada titik ini, kita akan berhenti membicarakan orang lain dan mulai melihat akar masalah kita dari sudut pandang lain. Lagi pula, ada batasan tegas mengenai apa yang dapat Anda pelajari bahkan dari pengamatan dan diskusi terbaik tentang orang lain. Seperti kata pepatah lama, “Tidak ada yang menjadi nyata sampai ia dialami.” Tentu saja, kita telah menggunakan kata kita sepanjang diskusi ini, yang menyiratkan bahwa kita - Anda dan saya - cenderung menjadi korban jebakan psikologis yang sama. Tapi akui saja, kami tidak seburuk orang lain. Benar?

Mendesain untuk Desainer

Beberapa tahun yang lalu saya ikut mendirikan situs komunitas kecil untuk desainer dan pengembang Web. Komunitas kami menerbitkan artikel oleh tamu undangan (tidak seperti Smashing Magazine) tentang segala hal mulai dari strategi UX dan proses desain Web hingga manajemen dan filosofi. Kami juga membuat perangkat lunak sumber terbuka dan membagikannya di bagian situs komunitas kami.

Seiring waktu, situs web mendapatkan popularitas di komunitas Web. Lalu lintas masuk cukup bagi tim kami untuk mempertimbangkan mendesain ulang situs web agar dapat menangani rencana masa depan kami dengan lebih baik. Situs web aslinya adalah sesuatu yang saya rancang dan kodekan selama beberapa akhir pekan. Itu tidak terlalu mudah dikelola, dan pada tahun pertama menjalankan situs web ini, kami belajar lebih banyak tentang merek, tujuan, dan identitas kami sendiri. Jadi, kami menetapkan tenggat waktu untuk diri kami sendiri, dan saya setuju untuk memimpin proyek desain ulang. Kami akan menjulukinya Project Redo.

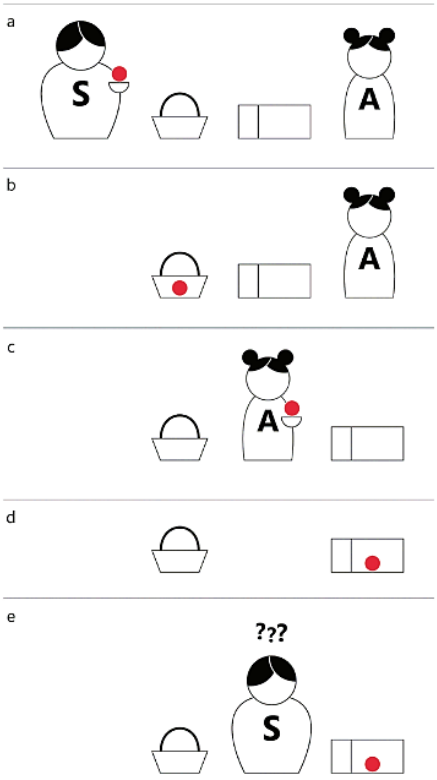
Saya segera memutuskan bahwa saya memerlukan bantuan untuk proyek ini. Saya mempunyai banyak tanggung jawab lain dalam pekerjaan saya; desain ulang situs web hanyalah salah satunya. Tapi, saya tidak akan menyerahkan bayi saya kepada sembarang orang. Aku membutuhkan seseorang yang dapat kupercaya, seseorang yang rela memberikan nyawanya sebelum membiarkan bayiku terluka.

Saya menghubungi seorang teman dan desainer lokal yang cocok dengan kebutuhan tersebut. Sebut saja dia Dave. Saya meneleponnya dan kami mengobrol tentang tujuan proyek ini. Taruhannya tinggi, kataku padanya. “Anda memiliki kebebasan yang luar biasa dalam proyek ini. Bagaimanapun, saya adalah satu-satunya pemangku kepentingan. Saya ingin Anda berpura-pura seperti sedang mendesain ulang situs web pribadi Anda. Saya ingin perhatian dan energi itu. Kamu bermain?” Dave mendaftar. Saat itu, saya tidak hanya menjadi teman dan rekan desainer, tetapi juga klien.

Kami membagi pekerjaan di antara kami sendiri. Saya akan bertanggung jawab atas arsitektur informasi. Dia akan bertanggung jawab atas desain visual dan markup front-end. Kami sepakat untuk berkolaborasi melalui semua itu. Dengan itu, saya pergi ke papan gambar untuk mengerjakan arsitektur informasi. Seminggu kemudian saya memposting wireframe ke Basecamp. Dave memeriksanya, kami melakukan panggilan telepon singkat, dan dia mulai

mengubah kerangka itu menjadi makhluk penuh warna. Beberapa hari kemudian dia memposting komposisi warna salah satu halaman ke proyek Basecamp kami untuk memberi saya gambaran sekilas tentang arah seninya. Tak perlu dikatakan lagi, dia sangat menyukai apa yang dia ciptakan. Wajar saja kan?9 Sayangnya, saya tidak bisa melihat apa yang dilihatnya. Saat dia melihat keindahan, saya melihat sebaliknya. Seolah-olah dia berfungsi dalam realitas yang sama sekali berbeda dengan realitas saya.

12.3 UJI SALLY–ANNE



Saya pertama kali menemukan permata ini dalam buku Kathryn Schulz yang luar biasa, Being Wrong. Tes Sally–Anne dilakukan oleh anak-anak berusia antara tiga dan empat tahun. Ini melibatkan pementasan pertunjukan boneka sederhana yang melibatkan dua karakter, Sally dan Anne (gambar a di halaman berikutnya). Sally menaruh kelereng ke dalam keranjang, menutup tutupnya dan meninggalkan ruangan (gambar b). Tak lama kemudian, Anne yang sangat nakal membuka tutup keranjang, mengeluarkan kelereng (c) dan meletakkannya di dalam kotak yang terletak di pojok (d). Kini, anak yang menyaksikan semua ini ditanyai pertanyaan sederhana: ketika Sally kembali, di mana dia akan mencari kelereng (e)?

Hampir setiap anak dalam kelompok usia ini berseru dengan percaya diri, “In the box!” Jawaban ini membingungkan orang dewasa karena alasan yang jelas: tidak mungkin Sally mengetahui bahwa kelereng itu telah dipindahkan secara nakal oleh Anne karena Sally tidak ada untuk menyaksikannya. Tapi anak-anak tidak peduli dengan detail ini. Bagi mereka, realitas dan representasi realitas dalam pikiran mereka adalah satu dan sama. Sally mengira kelereng itu ada di dalam kotak karena, ya, kelereng itu ada di dalam kotak.

Anak-anak memberikan jawaban yang salah karena mereka belum mengembangkan apa yang disebut teori pikiran (ToM), sebuah ciri pembeda manusia jika dibandingkan dengan kebanyakan mamalia lainnya. Kami mengembangkan ToM pada saat kami berusia lima tahun. Faktanya, jika Anda memberikan tes tersebut kepada anak berusia lima tahun, Anda akan terkejut karena telah membuang-buang waktu mereka sebelum mereka memberikan jawaban yang benar.

Teori pikiran memberi kita dua pengetahuan penting yang, bila digunakan dengan benar, memiliki kemampuan untuk mengeluarkan sisi terbaik dari diri kita:

1. Realitas versi pikiran kita bukanlah realitas yang sebenarnya: ia hanyalah salah satu penafsiran realitas.

2. Setiap orang mempunyai pemikirannya masing-masing, dan dengan demikian, penafsirannya sendiri terhadap realitas.

Namun seperti yang ditulis oleh David Eagleman, penulis Incognito: The Secret Lives of Brains, “Mungkin ada kesenjangan besar antara pengetahuan dan kesadaran”. Ketika saya kembali ke Project Redo, saya sama sekali tidak menyadari lingkaran setan irasionalitas yang akan saya masuki.

Ini Pesta Saya Dan Saya Akan Menangis Jika Ingin

Saya memberi Dave tanggapan jujur saya. “Menurutku ini bukan arah yang benar, Dave. Saya tidak tahu harus berkata apa lagi selain itu, itu terlalu diharapkan. Desainnya tidak berani.” Kepala berbandul Dan Ariely-ku sedang menatapku, dan aku mengakui kepada Dave bahwa aku mungkin berada di bawah pengaruh yang tidak rasional. Itu adalah seruannya tentang bagaimana melanjutkannya (tapi saya menyilangkan jari di belakang punggung). Berpengalaman saat dia berurusan dengan klien seperti saya, dia tidak bergeming. “Mari kita mencoba arah yang benar-benar baru. Ini akan menyenangkan.” Dia kembali ke papan gambar, namun desain pertamanya telah menimbulkan kerusakan yang diperlukan dalam bentuk menetapkan jangkar yang sangat negatif bagi saya.

Saat pertama kali membuka desain yang baru diposting, saya terkejut. Ini jelas berbeda. Namun setelah beberapa menit, sistem limbik saya yang bertanggung jawab atas emosi seperti ketakutan dan kemarahan, dan juga rumah amigdala yang saya sebutkan sebelumnya – mengambil alih. Saya menyimpulkan bahwa desain baru ini, meskipun lebih baik, hanyalah sepupu dari desain pertama, yang saya yakini, benar-benar mengerikan. Mengetahui dengan baik bahwa saya mungkin berada di bawah pengaruh bias yang tidak rasional, saya memutuskan untuk tidak memikirkannya. Keesokan harinya, saya menampilkan desainnya di monitor saya dan tidak ada yang berubah. Faktanya, keadaannya menjadi lebih buruk. Saya menunjukkannya kepada istri saya, rekan satu tim, dan beberapa orang lainnya. Tidak ada yang membagikan reaksi saya. Kebanyakan orang berkata, “Itu bagus. Tapi saya bukan seorang desainer.”

Saya berpikir pada diri sendiri, “Oke, saya seorang desainer. Saya adalah seseorang yang sangat sadar akan perilaku irasional. Saya tahu cara kerjanya. Saya telah membaca banyak tentang otak dan psikologi. Saya telah mengikuti saran dari banyak penulis favorit saya: mulai dari mendapatkan dukungan setan hingga membiarkan reaksi awal saya membara. Tapi saya masih merasa ini arah yang salah. Saya pasti benar. Benar?" Tentu saja saya setuju dengan diri saya sendiri.

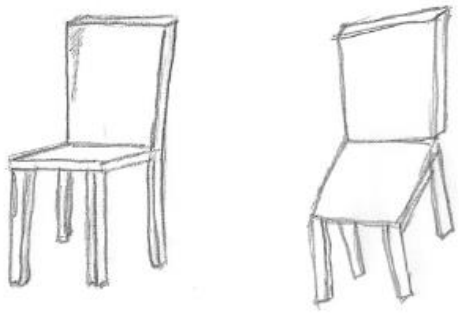
Saya kemudian dengan cepat memutuskan bahwa saya harus menyelamatkan hari itu, dan melakukan kecerobohan desainer utama: Saya sendiri yang mulai mengerjakan komposisi warna. Eagleman bergerak-gerak di suatu tempat di kejauhan, dan dalam sekejap, semua pengetahuanku luput dari kesadaranku. Namun yang benar-benar menakutkan adalah hal ini terjadi jauh lebih sering daripada yang kita ketahui, dan ini memang disengaja. Ini pada dasarnya adalah sifat manusiawi.

Dan tidak ada yang lebih mengungkap kekurangan elegan ini selain fakta yang tampaknya tidak berhubungan tentang diri kita sendiri: kebanyakan dari kita tidak bisa menggambar.

Untuk Belajar Menggambar, Kamu Harus Belajar Melihat

"Saya tidak bisa menggambar," kata istri saya dengan jelas. "Yah, aku bisa menggambar figur tongkat, tapi itu saja. Tapi tentu saja, saya akan mencobanya." Saya telah menyarankan agar kita mempelajari buku instruksi menggambar yang paling banyak digunakan di dunia, buku klasik modern Betty Edwards, Drawing on the Right Side of the Brain. Kebanyakan dari kita bisa memahami perkataan istri saya karena kebanyakan dari kita tidak bisa menggambar lebih baik dari anak tetangga. Dan kepercayaan umum mengapa kita tidak bisa menggambar adalah bahwa kemampuan menggambar adalah anugerah bawaan: sesuatu yang kita miliki sejak lahir. Namun Edwards sangat tidak setuju dengan teori ini.

Sebaliknya, dia percaya bahwa siapa pun bisa belajar menggambar secara realistis hanya dengan sedikit instruksi. Dia telah membuktikan hal ini berkali-kali melalui buku dan kursusnya. Edwards menulis, "Seorang siswa menggambar pemula yang diminta menggambar tampak depan sebuah kursi sering kali mengubah bayangan retina dari kursi tersebut menjadi bentuk bulat atau persegi sepenuhnya, bahkan melalui gambar retina dari kursi tersebut bentuknya sempit dan horizontal."



Gambar 12.2 Kursi yang digambar dengan sudut pandang yang tepat (kiri); sebuah kursi yang ditarik di bawah pengaruh keteguhan ukuran (kanan).

Dia mengidentifikasi penyebab di balik fenomena ini sebagai proses saraf inti yang dikenal sebagai keteguhan ukuran: sebuah proses yang memastikan bahwa persepsi kita terhadap objek relatif konstan terlepas dari jarak sebenarnya dari retina. Tanpa keteguhan ukuran, kita tidak akan bisa mengidentifikasi gajah di cakrawala dari siluetnya yang kecil. "Keteguhan ukuran dapat mengacaukan persepsi dengan mengesampingkan informasi langsung yang mengenai retina, menyebabkan kita 'melihat' gambar yang sesuai dengan pengetahuan yang sudah ada sebelumnya," lanjut Edwards.

Faktanya, ketika kita mencoba menggambar kursi, pengetahuan kita yang sudah ada sengaja menyabot tangan kita. Edwards melanjutkan, "Belahan kiri (otak) tidak sabar dengan persepsi mendetail ini dan mengatakan, pada dasarnya, ini adalah kursi, saya beritahu Anda. Itu cukup untuk diketahui. Bahkan, jangan repot-repot melihatnya, karena saya sudah punya

simbol siap pakai untuk Anda. Ini dia; tambahkan beberapa detail jika kamu mau, tapi jangan ganggu aku dengan urusan yang terlihat seperti ini."

Kunci dari metode pengajaran menggambar Edwards adalah dia berfokus pada mengajar siswanya cara mematikan sementara proses visual seperti keteguhan ukuran. Dengan kata lain, Edwards mengajari siswanya cara melihat apa yang ada, bukan apa yang dipikirkan otak mereka. Ada perbedaan yang halus namun mendalam di antara keduanya, perbedaan yang sering ditemui sebagian orang secara alami. Bagi kita semua, ada metode Edwards. Ini sangat efektif sehingga istri saya mampu meningkatkan potret dirinya secara signifikan setelah mengikuti beberapa nasihat dalam buku Edwards.

Gambar 12.3 Upaya pertama istri saya mengambil potret diri (kiri); upaya istri saya mengambil potret diri setelah 15 menit menerima instruksi dari buku Edwards (kanan).

Sekarang pertimbangkan bahwa keteguhan ukuran hanyalah satu dari ribuan proses neurologis dan fisiologis diam yang terlibat dalam berfungsinya manusia. Dan, tidak seperti keteguhan ukuran, kami tidak memiliki kemampuan untuk mengakses atau mematikannya. Faktanya, berfungsinya kita bergantung pada proses yang bekerja tanpa henti di belakang layar. Ketika dikombinasikan dengan pengalaman pribadi kita, proses-proses ini memainkan peran utama dalam membentuk realitas kita: sebuah realitas yang, bahkan jika kita lulus tes Sally-Anne pada saat kita berusia lima tahun, sering kali tetap terdistorsi dan, yang lebih penting, unik. milik kita sendiri.

Sayangnya, saat ini Project Redo masih berjalan lambat, dan tidak ada seorang pun di sekitar saya yang dapat membantu otak saya melakukan perubahan kognitif yang diperlukan untuk melihat kenyataan sebagaimana adanya.

Menyelamatkan Hari

Dalam Kebenaran Jujur Tentang Ketidakjujuran: Bagaimana Kita Berbohong kepada Semua Orang Terutama Diri Sendiri, Ariely menulis tentang model ekonomi yang berlaku untuk kecurangan dan kebohongan yang dikenal sebagai SMORC: model kejahatan rasional yang sederhana. Dia menulis, "Kita semua mencari keuntungan kita sendiri saat kita menjalani dunia ini. Apakah kita melakukan ini dengan merampok bank atau menulis buku, itu tidak penting bagi perhitungan rasional kita mengenai biaya dan manfaat." Tapi ini tidak cukup

menggambarkan keseluruhan gambarannya. “Bagaimana kita bisa mendapatkan manfaat dari berbuat curang dan pada saat yang sama tetap memandang diri kita sebagai orang yang jujur dan hebat?” dia menantang. “Di sinilah fleksibilitas kognitif kita yang luar biasa berperan. Berkat keterampilan manusia ini, selama kita berbuat curang sedikit saja, kita bisa mendapatkan manfaat dari berbuat curang dan tetap memandang diri kita sebagai manusia yang luar biasa. Tindakan penyeimbangan ini adalah proses rasionalisasi, dan ini adalah dasar dari apa yang kita sebut teori faktor fudge.”

Sejalan dengan itu, ketika saya memposting desain saya sendiri ke Project Redo Basecamp beberapa jam setelah saya mulai mengerjakannya, saya memberi judul pesan tersebut dengan pesan yang membawa keberuntungan dan agak menjengkelkan, “Arah yang Baru.” Di dalamnya saya menjadi puitis tentang kekuatan desain baru ini dan mengapa ini adalah cara yang tepat untuk maju (tentu saja, dengan mengakui bahwa saya bisa saja salah dalam penilaian saya). Saat ini, otakku sudah menyusun cerita yang rumit tentang keseluruhan situasi, dan berkat faktor fudge aku telah menjadikan diriku peran protagonis. Saya adalah pahlawan yang, setelah banyak perenungan dan kesengsaraan, akan menyelamatkan proyek ini. Aku harus melakukan apa yang harus kulakukan, tapi itu demi tujuan baik, pikirku.

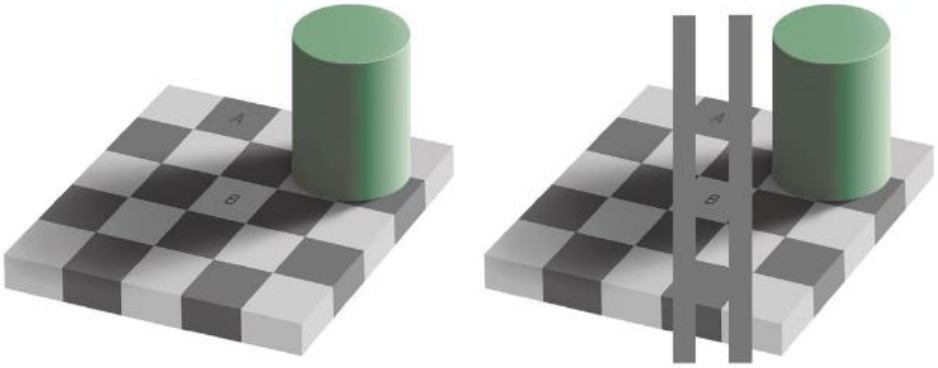
Kalau dipikir-pikir, enam jam yang saya habiskan untuk mengerjakan konsep saya tidak sebanding dengan kerja keras Dave selama seminggu penuh. Dave pasti langsung melihatnya ketika dia memegang desainku di sebelahnya, tapi memutuskan untuk tetap diam. Sebaliknya, dia melakukan apa yang sering kita lakukan saat menghadapi konfrontasi dalam proses desain: dia mencoba menyelamatkan situasi dengan menyarankan agar dia mencoba menggabungkan desain kami. Saya tidak terlalu menyukai gagasan itu, namun tidak terlalu menanggapinya. Saya juga menghindari konfrontasi. Neokorteks telah membantu kami berdua, tapi sudah sedikit terlambat. Saya berkata, “Baiklah, mari kita mencobanya.”

Kami tidak berhasil melewati dua versi desain Frankenstein. Pada saat itu, ada sedikit kecanggungan yang tidak dapat disangkal, dan kami berdua tahu bahwa proyek tersebut telah gagal. Kami memutuskan untuk mengobrol di telepon. Saya akhirnya mengakui, “Dave, menurut saya ini tidak akan berhasil. Tidak ada perasaan sakit hati, tapi saya rasa saya ingin mengambil alih desain visual.” Tapi kali ini Dave mengejutkanku dengan melawan. “Nishant, menurutku konsep keduaku adalah yang terbaik. Saya bahkan menunjukkan kepada desainer lain kedua desain tersebut secara berdampingan, dan dia setuju,” katanya. Dia menyebutkan nama desainer lainnya, seorang veteran terkenal yang saya hormati.

Aku merasa seperti angin telah menghempaskan diriku. Dalam momen kejelasan yang tidak disengaja, saya menjawab, "Saya pikir saya harus keluar dari proses pengambilan keputusan." Dave, kaget, setuju. Kami menutup telepon. Dave telah berusaha sekuat tenaga, dan saya menghormatinya. Tapi aku masih tidak bisa melihat apa yang dia lihat. Saya bahkan bisa mengakui bahwa mungkin desain saya tidak tepat. Tapi aku tidak bisa melihat bagaimana keadaannya. Ternyata, keinginan untuk melihat dunia secara berbeda mempunyai batas-batas yang tidak menguntungkan. Dan hal ini memberikan petunjuk terakhir mengenai sumber kesengsaraan kita.

Melihat Itu Percaya Bukan Melihat

Ilusi optik memberikan contoh yang bagus tentang keterbatasan kita.14 Favorit pribadi saya adalah ilusi checkershadow yang pertama kali dikembangkan oleh peneliti MIT Edward Adelson. Ilusi di sini melibatkan penguraian warna kotak bertanda A dan B pada gambar kiri gambar 7. Kapan ditanyakan, peserta (termasuk Anda) akan menunjukkan bahwa A berwarna abu-abu tua, sedangkan B berwarna abu-abu terang. Faktanya, kotak-kotak itu memiliki warna abu-abu yang sama. Dan ini menjadi jelas ketika Anda menghubungkan dua kotak dengan persegi panjang abu-abu solid seperti yang ditunjukkan pada gambar kanan. Namun bagian terbaik dari ilusi ini masih di depan.



Gambar 12.4 Gambar ilusi

Jika Anda menghilangkan persegi panjang abu-abu (lihat kembali gambar kiri), itu seolah-olah kita bodoh lagi. Tidak peduli seberapa keras kita mencoba, kita melihat kedua kotak itu sebagai warna abu-abu yang berbeda. Terlepas dari kenyataan bahwa sistem visual kita - lebih dari tiga puluh bagian otak berbeda yang didedikasikan untuk membantu kita melihat - adalah bagian tubuh manusia yang paling canggih secara anatomis, kita mungkin akan tertipu berulang kali oleh ilusi optik seperti bayangan Adelson. Dalam Predictably Irrational, Ariely bertanya-tanya seberapa sering kita harus dibodohi di area di mana otak tidak memberikan bantuan khusus? Suka membuat keputusan keuangan? Atau, sedikit lebih dekat dengan rumah, menganggap desain yang bagus sebagai sesuatu yang hebat?

Penelitian Ariely berkaitan dengan penemuan dan dokumentasi apa yang disebut ilusi pikiran, yang lebih dikenal sebagai bias kognitif. Penahan hanyalah salah satu dari ratusan bias kognitif yang ada pada manusia. Setelah Anda membaca temuan ini, hampir mustahil untuk melihat dunia, dan diri Anda sendiri, dengan cara yang sama lagi. Di satu sisi, sungguh melegakan jika kita bisa mendapatkan penjelasan ilmiah atas beberapa permasalahan hidup. Namun dalam arti yang berbeda, Anda dihadapkan pada betapa banyak yang tidak Anda ketahui tentang dunia ini, seberapa banyak Anda menganggap remeh, dan betapa salahnya kita sebagai manusia.

Kembali ke Project Redo, saya mulai yakin akan kesalahan saya dalam situasi tersebut. Saya mengirimkan kedua desain tersebut untuk pemungutan suara buta ke seluruh tim saya tanpa menambahkan konteks apa pun selain, “Berikut adalah dua opsi untuk desain tersebut.

Pilih salah satu yang paling kamu sukai." Kelima anggota tim saya memilih desain Dave. Keputusannya jelas bagi saya meskipun saya tidak dapat memahaminya. Aku adalah balas dendam Adelson yang menyeringai. Saya tidak dapat melihat kotak-kotak tersebut memiliki warna yang sama tidak peduli seberapa keras saya mencoba, namun saya tahu bahwa warnanya sama.

Jadi, saya menelepon. "Kami akan melanjutkan rencanamu, Dave. Terima kasih telah berusaha keras," saya memposting di Basecamp. Dave sangat gembira. Tiga minggu kemudian kami menyelesaikan situs web tersebut dan mendapat banyak sambutan meriah. Produk ini menjadi hit di komunitas, ditampilkan di beberapa pameran desain, dan bahkan memenangkan beberapa penghargaan bergengsi. Dave menyatakan bahwa itu tetap menjadi salah satu portofolio terbaiknya hingga saat ini. Dan bagi saya, hal itu tetap menjadi pengingat akan ketidakrasionalan saya sendiri.

Dan… Adegan

Saat saya mengingat kembali pertemuan pertama saya dengan disfungsi di tempat kerja episode saya dengan Prakash dan tim - mau tak mau saya bertanya-tanya apa yang akan saya lakukan secara berbeda jika saya memiliki pengetahuan yang saya miliki sekarang.

Selama beberapa tahun terakhir, saya merasa senang (dan sedih) karena menemukan sisi baru dari manusia. Dengan menerapkan karya orang-orang seperti Brafman bersaudara, Ramachandran, Eagleman, Gigerenzer, Edwards, Ariely, dan banyak lainnya ke dunia kolaborasi desain, saya telah mampu melewati banyak situasi rumit, dan juga mampu mengembangkan sebuah seperangkat aturan praktis baru yang tampaknya kontra-intuitif untuk membantu saya selama ini. Berikut beberapa di antaranya (dan sebagian besar berasal dari pengalaman yang belum saya sebutkan secara khusus di bab ini):

- Jangan mencoba mendidik klien dan pemangku kepentingan tentang desain. Sebaliknya, luangkan waktu untuk mempersiapkan mereka agar menyadari bahwa itu bukan domain mereka.
- Jangan pernah melakukan tinjauan desain di ruang konferensi yang dihadiri semua pemangku kepentingan. Studi kesesuaian sosial dengan tegas menetapkan bahwa pendekatan ini ditakdirkan untuk gagal. Ketika Anda berhasil, itu hanya karena statistik bisa memaafkan.
- Jika klien Anda meminta beberapa versi desain, Anda akan meningkatkan peluang Anda untuk mendapatkan persetujuan secara cepat dengan membuat tiga versi: dua versi serupa (versi yang Anda ingin klien pilih harus lebih baik), dan yang lain jelas merupakan yang terburuk dan tidak mirip dengan pasangan tersebut. Meskipun demikian, Anda mungkin lebih baik memecat klien tersebut.
- Mengajak para desainer hebat untuk berkolaborasi pada umumnya akan menghasilkan desain yang tidak sebaik yang dihasilkan oleh seorang desainer hebat (kecuali, para desainer tersebut benar-benar dapat berfungsi sebagai satu desain; jarang terjadi, namun tentu saja mungkin). Ini karena Anda mencoba menyatukan realitas-realitas independen dan merata-ratakan visi-visi besar secara independen.

- Mencoba meyakinkan seseorang tentang sesuatu yang bertentangan dengan keyakinannya biasanya akan memperburuk situasi. Anda harus mencari solusi yang lebih halus untuk menangani individu tersebut.
- Dalam sebagian besar situasi di tempat kerja, pujian dan dukungan positif merupakan motivator yang jauh lebih baik dibandingkan pembayaran dalam bentuk uang atau barang lainnya.
- Pemangku kepentingan pada umumnya ingin menjadi warga yang terhormat dalam proses perancangan meskipun perilaku mereka tampaknya menunjukkan sebaliknya. Belajar membaca yang tersirat untuk mempengaruhi kecenderungan alami manusia untuk ingin berbuat baik.
- Tujuan terburuk yang dapat Anda tetapkan untuk diri Anda sendiri adalah menjadi yang terbaik. Tujuan terbaik yang dapat Anda tetapkan untuk diri Anda sendiri adalah (selalu) menjadi lebih baik. Ada perbedaan yang halus namun mendalam.
- Tidak ada seorang pun yang kebal terhadap perilaku tidak rasional. Dan sekeras apa pun Anda mencoba, Anda tidak akan pernah menghilangkannya dalam diri Anda atau orang lain. Pendekatan terbaik untuk menghadapi irasionalitas adalah dengan menetapkan checks and balances untuk mendeteksi dan mengelolanya (terutama untuk diri Anda sendiri). Belajarlah untuk memantau diri sendiri dan sambil bertanya pada diri sendiri pertanyaan-pertanyaan seperti, “Apakah saya secara tidak sadar berada dalam ilusi optik saya sendiri saat ini?”
- Orang mengabaikan masa depan. Artinya, kita dirancang untuk menjadi pemikir jangka pendek. Inilah sebabnya mengapa janji manfaat jangka panjang dari suatu desain umumnya tidak pernah menjadi argumen yang meyakinkan.

Saya memiliki lebih banyak heuristik yang belum saya sebutkan. Dan perlu diperhatikan bahwa tujuan saya mencantumkan hal-hal di atas tentu saja bukan untuk membuat Anda merasa tertipu; lagi pula, saya belum memberikan pengalaman dan argumen yang relevan untuk lebih dari setengahnya. Sebaliknya, niat saya adalah untuk menyampaikan poin kritis: meskipun heuristik di atas telah memberikan manfaat yang baik bagi saya, namun heuristik tersebut bukanlah aturan yang tegas dan tegas, dan saya sering kali harus melanggarnya.

Dalam beberapa tahun terakhir saya menjadi sangat bersemangat untuk menyebarkan apa yang telah saya pelajari. Dengan cara saya sendiri, saya telah menulis tentang perspektif tertentu di blog saya, dan mempresentasikan beberapa konsep ini di konferensi. Selama ini, saya telah didekati oleh sejumlah orang, termasuk beberapa penerbit, yang mendesak saya untuk menulis artikel, bahkan buku tentang topik tersebut. Namun sebagian besar pembicaraan ini dengan cepat mengarah pada satu pertanyaan: praktik terbaik apa yang dapat menyelesaikan masalah ini untuk selamanya?

Jawaban saya selalu, “Tidak ada.” Memang benar, ini adalah jawaban yang mengecewakan. Dan itu mungkin mengecewakan Anda juga. Ini tentu saja mengecewakan saya saat pertama kali saya mengucapkan kata-kata itu dalam sesi tanya jawab setelah salah

satu ceramah saya. Namun kekecewaan ini khususnya mengingat apa yang telah kita pelajari tentang diri kita sendiri dalam bab ini adalah ironis sekaligus paradoks.

Kami tidak memiliki praktik terbaik statis yang selalu berhasil dalam membuat desain sempurna atau menulis kode sempurna. Sebaliknya, kami mengambil semua pengetahuan, keterampilan, dan pengalaman kami, dan menerapkannya secara kreatif pada proyek kami. Setiap situasi berbeda dari situasi berikutnya. Kenyataannya adalah bahwa bidang kita - sebenarnya, dunia kita - sangat tidak sempurna dan tidak ada apa-apanya gabungan kecerdasan dari semua ahlinya atau kekuatan seribu naga bernapas api, akan mengubah hal itu. Kami terus-menerus memperbarui praktik terbaik kami, mulai dari praktik yang berkaitan dengan penataan markup hingga merancang perangkat yang jumlahnya tak terbatas — seringkali menemukan diri kami tepat di titik awal kami memulainya. Meski terkadang membuat frustrasi, kita semua menghargainya sebagai salah satu hal yang membuat pekerjaan kita menarik.

Bahkan mungkin itu adalah sumber kebahagiaan kita. Seperti yang dikemukakan Mihaly Csikszentmihalyi dalam bukunya, Flow: The Psychology of Optimal Experience, menetapkan tujuan yang berada di luar jangkauan dan berupaya mencapainya adalah rahasia sebenarnya menuju kebahagiaan. Atau seperti yang dikatakan Gandhi, “Kemuliaan terletak pada upaya mencapai tujuan, bukan pada pencapaiannya.”

LANGKAH SELANJUTNYA

Saya yakin kebutuhan kita untuk menemukan solusi jitu bagi permasalahan manusia berasal dari kognisi yang melukiskan gambaran diri yang pemaaf dan membangun teori rumit tentang dunia. Namun yang bisa kita pelajari dalam bab ini adalah bahwa teori dan ideologi kita sering kali dibangun di atas fondasi yang lemah, hanya berdasarkan sedikit informasi nyata, dan disusun berdasarkan biologi di luar kendali kesadaran kita. Semakin cepat kita dapat menerima bahwa manusia sama seperti teknologi dan sistem kita yang tidak dapat diperbaiki lagi semakin cepat kita dapat mulai menangani permasalahan manusia dengan cara yang sama seperti kita menangani semua permasalahan dalam hidup: dengan memperoleh pengetahuan, mempraktikkan, menganalisis, dan mengulanginya.

BAB 13
TENTANG SEMANGAT KREATIF

Saat memulai proyek baru, kita sering kali dihadapkan pada tekanan kuat untuk memberikan sesuatu apa pun! dalam jangka waktu sesingkat mungkin. Seringkali klien membutuhkannya besok, atau lebih baik lagi kemarin, tentunya pada akhir minggu! (Dan coba tebak, hari ini Kamis.)

Dihadapkan dengan tenggat waktu yang keras dan tidak realistis, kita terbebani oleh ekspektasi dan mendapati diri kita mencari jalan pintas. Sebelum kita menyadarinya, kita mengambil jalur yang paling sedikit penolakannya terhadap ide-ide baru, yang, sebagai konsekuensinya, sering kali bukanlah ide-ide baru sama sekali, hanya ide-ide daur ulang dari proyek-proyek lama yang sudah ditinggalkan.

Ketika waktu terbatas dan anggaran terbatas, kita segera beralih ke komputer dalam upaya mewujudkan ide secepat mungkin. Namun komputer adalah tempat terakhir yang harus kita lihat. Sejarah menunjukkan kepada kita bahwa ide-ide ada di sekitar kita dan jika kita memilih untuk memperluas kerangka acuan dan memperluas bidang visi kita, meski hanya sedikit, kita akan menemukannya dengan cepat.

BERHENTI! Ambil napas dalam-dalam. Mari kita temukan kembali apa sebenarnya budaya ide dan, pada saat yang sama, ingatkan diri kita bahwa ini adalah tentang melangkah mundur dari dunia digital yang berisi satu dan nol, dan sebaliknya, beralih ke dunia analog berupa perpustakaan berdebu dan kopi yang dipenuhi aroma. toko.

Jadi, bagaimana Anda membangun budaya generasi ide? Bagaimana Anda membuka pintu air dan mengeluarkan ide-ide baru dan orisinal yang tiada habisnya? Apakah mungkin untuk mengadopsi strategi yang memungkinkan kita menghasilkan ide-ide bermakna yang dapat diandalkan, yang kemudian dapat kita laksanakan dengan terampil? Saya yakin, dengan mengadopsi kerangka mental sederhana dan menerapkan serangkaian strategi inti, hal tersebut akan berhasil. Saya yakin prosesnya melibatkan tiga faktor: pertama, mempersiapkan otak; kedua, memberdayakan seorang konduktor atau orkestrator gagasan; dan ketiga, mempertimbangkan ruang sebagai fasilitator.

Bagian pertama dari persamaan pembangkitan ide terletak pada memastikan otak dipelihara dan dipelihara secara teratur dengan pengetahuan, menjaga agar otak tetap terisi bahan bakar. Bagian kedua dari persamaan ini terletak pada menemukan konduktor yang memungkinkan otak yang prima bekerja dengan baik secara harmonis. Bagian ketiga dari persamaan ini terletak pada mendorong terjadinya benturan yang tidak disengaja, memfasilitasi benturan yang teratur antara otak yang sudah prima satu sama lain. Ketika bagian-bagian ini bersatu, ide-ide mengalir deras.

13.1 MENEMPATKAN TEORI KE DALAM PRAKTEK

Namun, jangan percaya begitu saja banyak perusahaan sukses yang menerapkan proses serupa. Seperti yang akan kita lihat melalui serangkaian contoh di bab ini, perusahaan

kelas dunia seperti Apple, Pixar, dan Google menerapkan strategi serupa untuk memastikan bahwa perusahaan mereka dipenuhi dengan ide-ide hebat. Mari kita lihat satu contoh dengan lebih detail.

Anda mungkin pernah atau mungkin belum pernah mendengar tentang 3M, namun Anda pasti pernah mendengar tentang banyak produk inovatifnya. Post-it note dan Scotch Tape hanyalah dua dari serangkaian ide yang sepertinya tak ada habisnya yang tumbuh dari pendekatan kreatif 3M dalam membangun budaya ide dan memungkinkan inovasi berkembang.

Kita semua pernah mendengar tentang "20 persen waktu" Google ini menghasilkan banyak ide menarik. Namun, ketika Google masih menjadi bintang di angkasa, 3M mulai menerima manfaat dari kerja fleksibel yang ditawarkan oleh 15 persen waktunya. Meyakini bahwa "kreativitas membutuhkan kebebasan", 3M telah menawarkan "hari impian1" ini sebagai waktu untuk bereksplorasi dan ruang untuk berefleksi, sejak tahun 1948 sebuah pendekatan yang sangat terbuka jika Anda mempertimbangkannya. Mendorong karyawan untuk menghabiskan 15% waktu kerja mereka pada proyek mereka sendiri, perusahaan juga menawarkan sumber daya kepada staf dan memberi mereka izin untuk mengatur diri sendiri dan membangun tim mereka sendiri. Pendekatan ini memungkinkan orang-orang 3M "untuk mengikuti wawasan mereka sendiri dalam upaya pemecahan masalah."

Seperti yang dikatakan perusahaan:2

Dalam iklim bisnis yang serba cepat dan penuh tekanan saat ini, banyak perusahaan mengambil pendekatan jangka pendek terhadap jalur pengembangan produk baru. Karena inovasi tidak terjadi dalam jangka waktu yang ditentukan, 3M mengambil jalur yang berbeda sebagian besar berkat prinsip-prinsip yang ditanamkan oleh mantan CEO, William L. McKnight ke dalam perusahaan. McKnight percaya akan pentingnya mempekerjakan orang yang tepat, menoleransi kesalahan, dan memberikan kebebasan kepada karyawan untuk bereksplorasi guna menumbuhkan budaya inovasi. 3M telah menerapkan Prinsip McKnight dengan mendorong karyawan untuk mendedikasikan sebagian besar waktu mereka untuk proyek dan penelitian yang melampaui tanggung jawab inti mereka. Meskipun mungkin memerlukan waktu bertahun-tahun agar inovasi "bermain-main" dapat membuahkan hasil, hasil dari 15 Persen Waktu 3M benar-benar luar biasa. Contohnya termasuk Pita Perekat Merek Scotch®, Post-it® Notes, Pelindung Kain Scotchgard™[…]

Jika Anda membagi Prinsip McKnight menjadi beberapa bagian, Anda mungkin melihat tiga faktor yang sama seperti yang diuraikan sebelumnya, faktor-faktor yang mengarah pada ide dan inovasi: memungkinkan orang untuk bermain-main dan mempelajari hal-hal baru (singkatnya, memberi mereka ruang untuk mengembangkan otak mereka)); memungkinkan orang untuk mengatur diri mereka sendiri dan mengelola diri mereka sendiri, memungkinkan mereka untuk membentuk tim ad hoc yang menyatukan potongan-potongan teka-teki (singkatnya, memungkinkan konduktor untuk mengatur tim untuk menciptakan hal-hal yang menakjubkan); dan yang terakhir, memungkinkan terjadinya pertemuan-pertemuan kebetulan yang berpotensi menghasilkan penemuan-penemuan baru (singkatnya, mendorong terjadinya benturan yang tidak disengaja).

Seperti yang dapat kita lihat dari contoh ini (dan mungkin Anda ketahui dari pengalaman Anda sendiri), ide-ide, berlawanan dengan opini umum, relatif mudah didapat; pelaksanaan ide melalui implementasi yang bermakna dan menyeluruh itulah bagian yang sulit. Luangkan waktu untuk menghasilkan ide dan Anda akan menemukan persediaan ide yang mengalir tanpa henti, terlalu banyak untuk diterapkan; bagian tersulitnya terletak pada memutuskan di mana akan menerapkan sumber daya untuk mengembangkan ide-ide ini. Mari kita mulai dan melihat tiga aspek pembangkitan ide, yang diuraikan di atas, dengan lebih detail.

Mempersiapkan Otak

Untuk menumbuhkan budaya menghasilkan ide, kita perlu meningkatkan rasa lapar akan informasi. Proses seperti ini tidak boleh hanya terjadi pada awal proyek, namun harus terjadi setiap saat. Pikiran yang secara alami ingin tahu akan selamanya dipenuhi dengan ide-ide. Maka, tugas utama yang harus kita selesaikan adalah menempatkan otak pada tempat yang tepat. Satu strategi sederhana untuk mendorong lebih banyak ide? Baca lebih lanjut, dan bacalah secara luas. Ada sejumlah buku bagus tentang ide dan dari mana asalnya. Dua karya yang harus berada di urutan teratas daftar desainer yang menghargai diri sendiri adalah The Myths of Innovation karya Scott Berkun dan A Technique for Producing Ideas karya James Webb Young. Keduanya bernilai emas dan akan membayar sendiri berkali-kali lipat. Mungkin tidak mengherankan jika keduanya mengikuti jalur yang sama, mengusulkan model yang – jika dipatuhi pasti akan menghasilkan ide yang tiada habisnya.

13.2 MITOS INOVASI

The Myths of Innovation karya Berkun yang diterbitkan pada tahun 2007 mengungkap mitos-mitos yang sering muncul untuk menjelaskan inovasi. Seperti yang dikatakan Berkun, “Ide tidak pernah berdiri sendiri”; ide-ide yang kita ingat selalu merupakan hasil ide dan penemuan lain. Hancurkan ide apa pun (misalnya, Twitter dan Instagram) dan Anda akan menemukan ide lain (masing-masing pesan teks SMS dan fotografi Polaroid). Kebijaksanaan yang diterima bahwa ide-ide ini adalah hasil inspirasi ilahi, yang muncul secara utuh entah dari mana, sangat jauh dari kenyataan. Faktanya, hal-hal tersebut merupakan hasil dari rasa ingin tahu yang melihat hubungan baru atau melihat pola lama terulang kembali dengan cara yang baru.

Berkun menggunakan gagasan tentang teka-teki untuk menjelaskan momen pencerahan, yang, bukan akibat campur tangan ilahi, sebenarnya merupakan momen di mana Anda melihat semua potongan-potongan dari teka-teki itu jatuh pada tempatnya dengan jelas. Seperti yang dia katakan:

Salah satu cara memikirkan pencerahan adalah dengan membayangkan mengerjakan sebuah teka-teki gambar. Saat Anda meletakkan potongan terakhir pada tempatnya, apakah ada yang istimewa dari potongan terakhir itu…? Satu-satunya alasan mengapa potongan terakhir penting adalah karena potongan-potongan lain sudah Anda letakkan pada tempatnya. Jika Anda mencampurkan kepingan-kepingan itu untuk kedua kalinya, salah satu dari kepingan-kepingan itu bisa menjadi kepingan ajaib terakhir.

Jadi, dari manakah potongan-potongan teka-teki itu berasal?

Singkatnya, Anda mengembangkan sendiri beberapa bagian dari teka-teki tersebut (melalui eksperimen dan pembuatan prototipe), namun bagian penting lainnya dari teka-teki tersebut sudah ada di luar sana. Anda belajar melihat bagian-bagian lain dari teka-teki itu dengan membaca secara luas, mengeksplorasi tema-tema yang asing dan menantang, dan mempertahankan pikiran yang ingin tahu. Menginterogasi dunia secara terus-menerus, baik offline maupun online, sangatlah penting. Potongan-potongan puzzle ada di mana-mana, ada di sekitar Anda; Anda hanya perlu belajar melihatnya.

Mari kita ambil gergaji ukir dan pecahkan.

iPhone. Sebuah teka-teki indah yang telah melalui serangkaian iterasi yang panjang dan inventif, di setiap kesempatan mengungkapkan ide-ide baru yang imajinatif dan menarik. Saat Steve Jobs meluncurkannya pertama kali pada 29 Juni 2007, kami terpesona. Kami kagum, tidak hanya pada kecakapan memainkan pertunjukan Jobs yang karismatik "Hanya ada satu hal lagi…." tetapi juga pada perangkat ajaib yang dia pegang di tangannya. Jobs, yang pernah menjadi impresario, menyatakan: Sesekali muncul produk revolusioner yang mengubah segalanya.

Tentu saja ini revolusioner dan tentu saja mengubah segalanya.

Namun, apakah itu benar-benar muncul, terbentuk sepenuhnya entah dari mana? Momen inspirasi ilahi? Tentu saja tidak. Itu adalah produk dari ide-ide lain, produk dari ide-ide lain, produk dari ide-ide lain…

Jika kita menafsirkan iPhone sebagai sebuah teka-teki menggunakan metafora Berkun, kita melihat sejumlah besar teknologi bersatu. Hal yang luar biasa mengenai "produk revolusioner" Jobs adalah cara ia mengambil begitu banyak ide yang sudah ada, masing-masing memiliki kompleksitas yang tak terhitung jumlahnya, dan merangkainya bersama-sama dengan mudah untuk menciptakan sebuah produk yang belum pernah kita lihat sebelumnya namun kita telah melihatnya, pada masanya. bagian penyusunnya berkali-kali sebelumnya. Di sinilah letak daya tarik penemuan. Kita melihat suatu produk yang benar-benar baru dan tampaknya tidak dapat dibayangkan, namun begitu benar dan jelas.

Mari kita periksa potongan-potongan puzzle iPhone dengan lebih detail, jelajahi bagaimana masing-masing ide yang bersatu untuk membentuk iPhone itu sendiri merupakan produk dari ide-ide lain. Pada tingkat yang paling abstrak, iPhone merangkai serangkaian ide inti: iPod asli, dan produknya sendiri dari banyak ide lainnya, tidak terkecuali Walkman dari Sony; telepon, yang berasal dari perangkat seluler pada saat itu (yang mengikuti serangkaian gagasan panjang sejak Alexander Graham Bell yang, saya yakin, akan menyukai iPhone, mungkin sebagai imbalan atas gagasan sebelumnya yang membantu mewujudkannya menjadi ada); kamera, dengan garis pendahulunya yang panjang; dan daftarnya terus bertambah. Singkatnya, semuanya terbuat dari benda lain.

13.3 TEKNIK MENGHASILKAN IDE

Lucunya, The Myths of Innovation karya Berkun juga merupakan produk gagasan lain. Dalam semangat akademis yang sebenarnya, yang lebih sering berdiri di pundak para raksasa

daripada yang Anda bayangkan, buku Berkun tahun 2007 menggemakan banyak tema dari volume yang jauh lebih ramping yang diterbitkan sekitar empat dekade sebelumnya, A Technique for Producing Ideas karya James Webb Young. dari tahun 1965. (Rekomendasi pribadi: jika Anda membeli hanya satu buku untuk meningkatkan kemampuan Anda dalam menghasilkan ide, jadikanlah buku itu milik Young.)

Dihormati sebagai 'Manusia Periklanan Tahun Ini' pada tahun 1946, Young (1886–1973) adalah seorang eksekutif periklanan pemenang penghargaan yang mungkin menulis salah satu buku paling berpengaruh dalam menghasilkan ide; tentu saja salah satu pendekatan yang paling ringkas dan tanpa basa-basi. Diambil dari pengalamannya yang kaya dan keinginannya yang besar akan pengetahuan, A Technique for Producing Ideas adalah sebuah risalah singkat, tajam dan sangat berharga mengenai proses menghasilkan ide.

Proses yang dilakukan Young sederhana saja, berpusat pada “melatih pikiran” dengan mengisinya dengan persediaan bahan mentah yang selalu siap pakai. Dari bahan bakar ini, seperti yang dia katakan, terbentuklah ide-ide, yang tidak mengejutkan, merupakan “kombinasi dari ide-ide lama.” Young mengutip sosiolog, ekonom, dan filsuf terkenal Vilfredo Pareto (1848–1923) sebagai pengaruh dalam pemikirannya (Anda mungkin pernah mendengarnya melalui Prinsip Pareto, yang juga dikenal sebagai aturan 80–20). Dalam bab berjudul “Teori Pareto”, Young mengacu pada pemikiran Pareto, menulis: Pareto berpendapat bahwa seluruh dunia dapat dibagi menjadi dua tipe orang utama. Jenis-jenis ini ia sebut, dalam bahasa Prancis yang ia tulis, “Spekulator” dan “Penyewa”. Dalam klasifikasi ini spekulan adalah istilah yang digunakan dalam arti kata 'spekulatif'. Spekulan adalah tipe orang yang spekulatif dan ciri khas tipe ini, menurut Pareto, adalah bahwa ia terus-menerus disibukkan dengan kemungkinan-kemungkinan kombinasi baru.

Young merangkum pemikiran tersebut melalui dua prinsip, sebagai berikut: pertama, “sebuah ide tidak lebih dan tidak kurang dari kombinasi baru dari elemen-elemen lama.” Kedua, “kapasitas untuk menggabungkan unsur-unsur lama ke dalam kombinasi baru sangat bergantung pada kemampuan melihat hubungan.” Pembaca yang cerdas akan melihat dengan jelas bahwa salah satu premis utama Berkun, bahwa gagasan tidak pernah berdiri sendiri, menggemakan salah satu prinsip utama Young, bahwa semua gagasan merupakan kombinasi dari gagasan-gagasan lama. Hal ini, pada gilirannya, menggemakan salah satu argumen utama Pareto, bahwa mereka yang memiliki kemampuan untuk memunculkan ide-ide “terus-menerus disibukkan dengan kemungkinan-kemungkinan kombinasi baru.”

Tiga pemikir cerdik, yang semua gagasannya merupakan campuran dari gagasan-gagasan lain. Kebetulan? Sangat tidak mirip. Meskipun Berkun, Young, dan Pareto memiliki bahasa yang sama dalam mengartikulasikan tesis utama ini – bahwa semua ide terbuat dari ide-ide lain bagi saya, Young-lah yang paling jelas mengartikulasikan model paling jelas untuk menerapkan pemikiran ini dalam cara yang kreatif. konteks dalam layanan penemuan ide yang dapat diandalkan. Mari kita lihat lebih detail teknik inti Young, yang menjadi tulang punggung proses lima tahap yang dapat digunakan untuk menghasilkan ide. Seperti yang dikatakan Young:

Teknik pikiran ini mengikuti lima langkah. Saya yakin Anda akan mengenalinya satu per satu, namun yang penting adalah mengenali hubungannya dan memahami fakta bahwa pikiran mengikuti lima langkah ini dalam urutan yang pasti - tidak ada kemungkinan salah satu langkah dapat diambil sebelum langkah sebelumnya selesai. , jika sebuah ide ingin dihasilkan. Lima tahapan Young pada dasarnya dapat diberi label sebagai berikut:

1. Kumpulkan bahan mentah.
2. Mengunyah.
3. Tinggalkan semuanya dan pergi.
4. Kagumi ide yang tiba-tiba terwujud.
5. Pertimbangkan ide tersebut keesokan paginya.

Tahap pertama, mengumpulkan bahan mentah, mudah; namun, seperti yang dikatakan Young dalam bukunya, hal itu sangat mudah namun juga sulit. Dengan bijak ia menyatakan, ”Mengumpulkan bahan mentah tidaklah semudah kedengarannya. Ini adalah tugas yang sangat buruk sehingga kami terus-menerus berusaha menghindarinya.” Menurut pengalaman saya, tahap ini adalah yang paling sulit; dibutuhkan disiplin untuk meluangkan waktu dari pekerjaan dan melatih pikiran. Sayangnya, hanya sedikit orang yang menikmati olahraga, dan bahkan lebih sedikit lagi yang menikmati latihan pikiran. Mengapa tidak melawan tren, mengambil risiko, dan melatih pikiran Anda? Percayalah, itu akan berterima kasih untuk itu.

Terlalu sering kita mengabaikan tahap pertama, tahap bahan mentah, karena kita terburu-buru menyelesaikan segala macam tugas yang ada. Gagal menyadari pentingnya mempersiapkan otak kita dengan pasokan bahan bakar yang selalu siap, kita membiarkan otak kita kekurangan gizi dan tidak mampu menciptakan koneksi baru (tidak ada bahan bakar baru, tidak ada pengunyahan). Singkatnya, kita mempunyai potongan-potongan puzzle yang terlalu sedikit, dan dengan persediaan potongan-potongan puzzle yang sama, kita akhirnya menciptakan potongan-potongan puzzle yang sama. Dengan persediaan bahan mentah yang melimpah, proses pengunyahan bisa dimulai.

Tahap kedua pengunyahan adalah tahap yang penting. Membalikkan ide-ide dalam pikiran Anda, melihatnya dari berbagai sudut pandang dan mencari hubungan di antara ide-ide tersebut, membantu Anda melihat dunia dan kemungkinan hubungannya dengan cara yang baru dan menarik. Seperti yang dikatakan Young: *“Apa yang Anda cari saat ini adalah hubungan, sebuah sintesis di mana segala sesuatunya akan bersatu dalam kombinasi yang rapi seperti teka-teki gambar.”*

Seperti yang dikatakan Young, tahap kedua sering kali disalahartikan sebagai ketidakhadiran pikiran, karena tenggelam dalam pikiran, ide-ide berputar-putar di benak Anda, dan Anda tertidur. Pada titik ini Anda akan menemukan bahwa fragmen atau, seperti yang dikatakan Young, “gagasan parsial” mulai muncul ke permukaan. Tuliskan semua ini di atas kertas, tidak peduli seberapa tidak penting atau absurdnya hal tersebut. Tindakan menuliskannya akan memperkuat otak Anda dan secara halus menabur benih untuk fase penting berikutnya.

Tahap ketiga meninggalkan segalanya dan menjauh sering kali merupakan tahap tersulit. Karena terdesak oleh tenggat waktu, kita terburu-buru mencari solusi terlalu dini;

pendekatan yang jauh lebih baik adalah dengan membiarkan pikiran kita bergerak, berfermentasi sedikit, seperti anggur berkualitas atau wiski yang sudah tua. Untuk menekankan pentingnya tahap ini, Young menggunakan contoh Sherlock Holmes yang memahami lebih dalam daripada asistennya Dr Watson bahwa agar sebuah ide atau solusi muncul, Anda harus menjauh, istirahat, dan membiarkan alam bawah sadar mengambil alih. Jika Watson lebih suka menangani kasus ini siang dan malam, Holmes tahu bahwa perjalanan ke teater hampir pasti akan memberikan jawaban yang sulit Anda dapatkan.

Tahap keempat kekaguman ketika sebuah ide muncul secara tiba-tiba adalah, diharapkan, sebuah pengalaman yang dimiliki oleh semua orang kreatif. Dengan memberi diri kita ruang untuk bernapas, kita akan menemukan solusi yang muncul secara alami, saat tahap ketiga (perjalanan ke teater) menghasilkan keajaiban. Inilah sebabnya kita sering mendengar orang-orang kreatif mempunyai tanda "Eureka!" saat di kamar mandi (atau, begitulah yang saya dengar, di kamar mandi).

Tahap kelima dan terakhir – menilai ide secara kritis setelah terburu-buru membayangkan – adalah momen kebenaran, atau apa yang disebut Linds Redding sebagai The Overnight Test3, titik di fajar pagi yang dingin dan kelabu saat Anda menjalani ujian baru. mengeluarkan ide dan mengujinya terhadap kenyataan. Seperti yang dikatakan Young dengan bijak: "*Pada tahap ini Anda harus mewujudkan ide kecil Anda yang baru lahir ke dunia nyata. Ketika Anda melakukannya, Anda biasanya mendapati bahwa anak tersebut tidak sehebat yang Anda bayangkan saat pertama kali Anda melahirkannya."*

Dalam fase terakhir ini, sangat penting untuk membagikan ide Anda secara luas. Jangan menahan diri ajukan pertanyaan tersebut untuk dicermati dan dorong kritik yang membangun. Jika ide tersebut tidak bertahan, lebih baik ulangi prosesnya (kembali ke tahap pertama) untuk mencari konsep yang lebih baru dan lebih kuat. Saya serahkan kata terakhir kepada Young, yang menyatakan: Jangan membuat kesalahan dengan menahan ide Anda pada tahap ini. Kirimkan ke kritik yang bijaksana. Ketika Anda melakukannya, hal mengejutkan akan terjadi. Anda akan menemukan bahwa ide yang bagus seolah-olah memiliki kualitas yang dapat berkembang dengan sendirinya. Ini merangsang mereka yang melihatnya untuk menambahkannya. Dengan demikian kemungkinan-kemungkinan yang selama ini Anda abaikan akan terungkap

Begini prosesnya: mudah, tapi seperti semua hal yang kelihatannya sederhana, sebenarnya cukup sulit. Pelajari prosesnya. Terapkan itu. Ulangi itu. Praktekkanlah. Saya yakin, ini akan bermanfaat bagi Anda; itu sangat membantu saya. Dengan banyaknya bukti (yang dikumpulkan oleh Pareto, Young dan Berkun) yang mengungkapkan bahwa bahan mentah adalah sumber munculnya ide-ide baru, menjadi jelas bahwa kita perlu menerapkan strategi dan teknik untuk mengumpulkan ide. Anda tidak akan terkejut saat mengetahui bahwa ada sejumlah pendekatan yang telah dicoba dan diuji, tidak terkecuali lembar memo atau buku sketsa (dan buku digital sejenisnya) untuk melakukan hal ini. Tekan mereka ke dalam layanan dan semuanya akan baik-baik saja.

Strategi

Saya ingin menyarankan tiga teknik yang dapat Anda terapkan untuk melatih otak dengan bahan mentah.

1. Perpustakaan
2. Buku sketsa dan lembar memo
3. Sabuk perkakas digital

Perpustakaan… Siapa yang Tahu?

Gunakan itu! Perpustakaan mengandung banyak sekali pengetahuan jauh lebih banyak daripada yang kita miliki melalui mesin pencari. Perpustakaan mendorong penemuan yang tidak disengaja, penemuan yang tidak disengaja ketika Anda kebetulan berada di rak buku yang kaya akan potensi. Mesin pencari, di sisi lain, memilih sendiri: dunia yang dapat diprediksi dan disaring oleh Google. Cari istilah dan temukan pilihan yang diberi peringkat secara algoritmik saja; hanya ada sedikit keberuntungan yang terlibat sama sekali. (Dan keberuntungan, dalam hal ini, membedakan Anda.)

Belajarlah untuk mencintai perpustakaan yang berdebu dan apak. Di sana Anda akan menemukan banyak sekali ide, yang membedakan Anda dari rekan-rekan Anda yang masih bergantung pada sumber-sumber digital yang sama. Jika Anda berada dalam situasi yang tidak menguntungkan karena tidak memiliki perpustakaan di lingkungan sekitar, jangan takut toko buku bisa menjadi alternatif yang baik dan jika itu adalah toko buku bekas, itu lebih baik!

Simpan Buku Sketsa atau Buku Tempel di Tangan

Buku sketsa dan lembar memo menjanjikan inspirasi yang tiada habisnya; Anda hanya perlu mengisinya. Selalu selalu! membawa buku sketsa dan pena. Meskipun tampaknya ide-ide terbaik dituliskan di belakang alas bir atau serbet, membawa peralatan berpikir Anda sudah menjadi kebiasaan. Semakin sering Anda menggunakan buku sketsa, Anda akan semakin tidak berharga. Membuat sketsa bukan tentang menjadi juru gambar yang hebat, ini tentang mensintesis dan memproses pemikiran dan ide Anda, seperti yang dirangkum dengan baik oleh Jason Santa Maria dalam artikelnya “Pretty Sketchy”: *“Buku sketsa bukanlah tentang menjadi seniman yang baik, melainkan tentang menjadi seorang seniman yang baik. pemikir yang baik.”*

Kenakan Sabuk Alat Digital

Manuver terakhir yang harus dilakukan (setelah Anda menggunakan alat offline) adalah menyesuaikan diri Anda dengan sabuk alat digital yang dipesan lebih dahulu. Kami diberkati dengan banyaknya alat yang kami miliki, seperti Gimme Bar5, Instapaper6, Instagram7, dan bahkan RSS teman lama kami, yang menawarkan metode siap pakai untuk mengumpulkan dan menghemat bahan mentah. Melengkapi sabuk alat digital Anda akan mengisi momen-momen tersebut saat bepergian dan memungkinkan Anda memanfaatkan waktu senggang (antrean bus, perjalanan singkat, rapat yang membosankan atau tanpa akhir), memungkinkan Anda menambahkan lebih banyak potongan puzzle ke EndlessJigsawPuzzleBoX™ Anda. Ingat, lebih banyak bahan mentah berarti lebih banyak kombinasi baru.

Pilih Konduktor

Dengan sekumpulan otak yang siap, kita dapat mulai mengaturnya, menghasilkan ide-ide yang lebih besar dari kesadaran kolektif mereka. Saat kita memanfaatkan peluang yang muncul saat memanfaatkan pikiran sarang, kita mulai menghasilkan ide-ide yang lebih besar daripada gabungan bagian-bagiannya. Gestalt sederhana dimana 1 + 1 + 1 = 5.

Untuk menghindari hiruk-pikuk, ada baiknya jika ada seorang konduktor yang siap mengatur segalanya, memastikan semua orang bekerja sama secara harmonis. Kondektur Anda mungkin adalah sosok yang mirip dengan Steve Jobs, memanipulasi tindakan berdasarkan visi yang agung dan tunggal; atau mereka bisa menjadi pemimpin tim tingkat rendah hingga menengah, atau serangkaian pemimpin yang menduduki peran kepemimpinan di antara mereka, yang dirancang untuk mendorong reaksi baru di antara otak Anda yang sudah prima. Bayangkan konduktor sebagai katalis, yang menyatukan unsur-unsur berbeda dan memfasilitasi reaksi di antara unsur-unsur tersebut untuk menciptakan sesuatu yang baru.

Kondektur tidak harus orang yang paling senior. Memang benar, terkadang menjungkirbalikkan hierarki organisasi dapat menghasilkan pemikiran baru yang radikal. Tempatkan sekretaris sebagai penanggung jawab selama satu hari dan Anda akan terkejut dengan kebijaksanaan yang dapat mereka bagikan melalui pandangan dunia mereka yang unik (pandangan dunia yang mungkin telah Anda lupakan dalam perjalanan Anda yang tiada henti menuju puncak).

Sejarah memberikan banyak contoh konduktor hebat, yang terbaru adalah Steve Jobs di Apple (dan NeXT); John Lasseter di Pixar; David Kelley di IDEO; dan masih banyak lagi. Dua yang perlu ditelusuri lebih detail, karena alasan berbeda, adalah Tony Fadell dan Thomas Edison, yang masing-masing dikenal sebagai 'The Podfather' dan 'The Wizard of Menlo Park'. Keduanya menceritakan kisah tentang berbagai peran yang dapat dimainkan oleh konduktor dalam membantu mengatur dan membentuk tim, untuk mendapatkan nilai maksimal dari semangat kreatif yang ada. Dalam kasus Fadell, kita belajar bahwa kadang-kadang konduktor yang tepat untuk tugas yang ada adalah orang luar, seseorang di luar tim yang ada. Dari Edison kita belajar bahwa kadang-kadang keterampilan konduktor terletak pada langkah mundur dan bertindak sebagai penuntun bagi orang lain, menempatkan tim yang tepat dan memungkinkan mereka untuk menemukan (dan mengagumi saat Anda menyaksikan, dan mematenkan, apa yang mereka lakukan).

Di bawah bimbingan seorang konduktor, kita dapat menghasilkan musik yang indah. Tanpanya, yang ada mungkin hanya suara dan kemarahan, yang tidak berarti apa-apa.

AYAH POD

Anda mungkin pernah atau mungkin belum pernah mendengar tentang Tony Fadell, tetapi Anda pasti pernah mendengar tentang produk ikonik yang ia perjuangkan dan bantu bentuk: iPod. Seorang konduktor yang ahli, dan juga seorang pria sejati, dia adalah kandidat ideal untuk mengelola tim pengembangan iPod Apple.

Dalam biografinya tentang Steve Jobs, Walter Isaacson menggarisbawahi pentingnya iPod bagi Jobs pada akhir tahun 2000. Frustrasi dengan pemutar musik portabel yang ada (dia

pikir semuanya “sangat jelek”), Jobs sangat tertarik pada Apple untuk menciptakan musik portabel. pemain lain tidak bisa. Bekerja sama dengan Jon Rubinstein, Jobs mulai mencari orang yang tepat untuk memimpin tim pengembangan.

Fadell, yang saat itu baru berusia 21 tahun, memiliki rekam jejak yang sempurna. Setelah memulai tiga perusahaan saat masih di Universitas Michigan, dia menghabiskan waktu bekerja di pembuat perangkat genggam General Magic sebelum menghabiskan beberapa waktu di Philips. Fadell mempunyai sejumlah ide untuk menciptakan pemutar musik digital yang lebih baik, namun tidak berhasil menyampaikannya ke RealNetworks, Sony dan Philips.

Suatu hari, saat bermain ski, dia mendapat telepon. Seperti yang dikatakan Isaacson: Rubinstein memberitahunya bahwa Apple sedang mencari seseorang yang bisa mengerjakan 'perangkat elektronik kecil'. Fadell, yang tidak kurang percaya diri, membual bahwa dia ahli dalam membuat perangkat semacam itu. Seorang penyihir dan konduktor sisanya adalah sejarah. Kisah Fadell berpusat pada bagaimana menempatkan otak yang tepat menyatukan orang-orang yang tepat. Hal ini menunjukkan bahwa mempersiapkan otak hanyalah bagian pertama dari persamaan.

Kisah Fadell menarik. (Jika Anda berkesempatan mendengarnya berbicara, hapus buku harian Anda karena dia adalah seorang presenter yang sangat menarik.) Selama periode awal mengerjakan pengembangan iPod, Fadell adalah seorang kontraktor independen; baru kemudian, di bawah tekanan, dia bergabung dengan Apple secara permanen. Hal ini menggarisbawahi fakta bahwa seorang konduktor tidak harus selalu menjadi CEO; mereka juga tidak perlu menjadi karyawan penuh waktu untuk membentuk visi dan mengoordinasikan tim. Di bawah tekanan kuat dari Jobs untuk mengirimkan iPod tepat pada hari Natal, Fadell melihat lebih jauh dari tim internal Apple untuk mengidentifikasi bakat eksternal. Kemampuan ini, untuk mengidentifikasi anggota tim terbaik - otak kanan yang prima - dan membawa mereka untuk membantu tugas yang ada adalah apa yang dimiliki oleh seorang konduktor hebat. Butuh drummer baru untuk menambah dimensi lain pada suara Anda? Ada banyak sekali musisi sesi berbakat; gunakan mereka.

Itu adalah pemikiran lateral Fadell - menyusun kembali anggota tim – itulah menarik. Pemikiran lateral, sebuah ungkapan yang diciptakan oleh Edward de Bono pada tahun 1967, adalah istilah lain yang harus dipahami oleh para pemburu ide: Dengan logika, Anda memulai dengan bahan-bahan tertentu, sama seperti dalam bermain catur Anda memulai dengan bidak-bidak tertentu […] Berpikir lateral tidak berkaitan dengan bermain dengan bidak-bidak yang ada, tetapi dengan upaya untuk mengubah bidak-bidak tersebut. Fadell bisa dibilang mengambil langkah ini lebih jauh, tidak hanya mengubah bidak-bidaknya, namun juga para pemain dalam permainan yang memainkan bidak-bidak tersebut.

Mengumpulkan sejumlah pemain yang dipilih dengan cermat dari berbagai tim dan kemudian menggunakan pikiran mereka untuk memunculkan ide-ide terbukti sangat ampuh dan merupakan inti dari perusahaan seperti Apple, Google, dan banyak raksasa inovasi lainnya. Memanfaatkan kumpulan gen yang kaya akan bakat dan mencari kombinasi baru dari ide-ide yang dikumpulkan dapat dengan cepat menghasilkan kekayaan yang berlimpah.

Kumpulkan sejumlah orang dalam satu ruangan, dilengkapi dengan Sharpies dan flip chart, berikan waktu terbatas pada diri Anda (setengah jam sudah terbukti lebih dari cukup), dan Anda akan terkejut dengan hasil yang diperoleh beberapa orang yang dipilih dengan baik. dapat menghasilkan. Kuncinya adalah bekerja cepat, mengejar pemikiran yang berbeda dan tidak menghambat kreativitas siapa pun. Dengan seorang konduktor yang memegang kendali dan memandu proses, ide-ide akan mengalir dalam waktu singkat.

Tentu saja, pilihan Anda terhadap konduktor akan sangat mempengaruhi banyak hal, jadi bagaimana Anda memilih konduktor yang tepat? Saya yakin penting untuk melepaskan sedikit dan menjelajah. Kondektur tidak harus selalu menjadi anggota paling senior yang hadir, tentu saja, jika anggota tersebut layak atas senioritasnya, mereka akan memahami bahwa memberdayakan orang lain dan membiarkan mereka mengambil alih tongkat estafet dapat memberikan hasil yang mengejutkan dan sering kali mengubah keadaan.

Cobalah memutar tongkat estafet dan biarkan setiap anggota tim memimpin suatu bagian, dalam periode singkat dan tajam dengan masukan yang berbeda-beda. Anda akan segera menemukan bahwa di sebagian besar anggota tim Anda jika Anda memberi mereka dorongan terdapat kemampuan untuk membangkitkan semangat. Lepaskan, dengarkan konduktornya, dan mulailah bermain.

Penyihir Taman Menlo

Seorang penemu, pengusaha dan putus sekolah, Thomas Edison (1847–1931), mungkin paling dikenal karena penemuannya yang luar biasa, termasuk: fonograf (agak mirip dengan iPod, hanya saja lebih tua dan – yah – berbeda); kamera film; dan tentu saja bola lampu listrik. Namun pendekatan Edison terhadap manajemen dan koordinasi tim mungkin merupakan pencapaiannya yang paling menarik, namun sering diabaikan.

Dijuluki 'The Wizard of Menlo Park', Edison adalah salah satu pengusaha pertama yang menerapkan prinsip produksi massal dan kerja sama tim skala besar dalam proses penemuan. Dia mungkin salah satu konduktor sejati pertama, memimpin tim ilmuwan dan peneliti di “pabrik penemuan” miliknya di Laboratorium Menlo Park, yang secara luas dianggap sebagai laboratorium penelitian industri pertama di dunia. Seperti yang dikatakan oleh asisten lama Edison, Francis Jehl: “Edison pada kenyataannya adalah kata benda kolektif dan berarti hasil karya banyak orang.”

Dengan menciptakan laboratorium yang berisi semua bahan yang diperlukan – seperti yang dikatakan Edison: “persediaan hampir semua bahan yang ada” untuk memicu ide-ide baru, Edison mampu mengumpulkan portofolio paten yang sangat besar. Menurut Rutgers, Universitas Negeri New Jersey, Edison berhasil memegang 1.093 paten AS dan lebih dari seribu paten di luar AS. Banyaknya jumlah paten yang didaftarkan Edison sungguh menakjubkan, dan merupakan bukti bahwa laboratorium penelitian terapannya tidak pernah kekurangan ide. Paten-patennya dapat dibaca dengan teliti di arsip Rutgers yang menarik8 dan paten-paten tersebut secara individual mewakili aspek-aspek proses penemuan dan merupakan jendela menuju dunia yang penuh dengan ide-ide menarik. Secara kolektif mereka merupakan panggilan sejati Edison, sebagai pemimpin dan orkestra dari individu-individu yang sangat cerdas.

Bakat Edison terletak pada mengelilingi dirinya dengan pikiran-pikiran yang sangat prima, yang dengannya ia dapat melakukan diskusi yang menarik dan luas, yang memicu ide-ide baru melalui dialog dan pertukaran. Edison mengajukan banyak paten untuk ide-ide eksplorasi terpisah yang secara kumulatif menghasilkan bola lampu yang menjadi penghargaannya. Dalam setiap kasus, tim peneliti bekerja untuk menciptakan dan menyempurnakan, yang diatur olehnya. Edison menunjukkan bahwa ide-ide sukses bersifat kolaboratif dan seorang konduktor yang baik dapat mewujudkan kesuksesan hanya dengan memahami dan memanfaatkannya.

Faktanya, dan bertentangan dengan pendapat umum, Edison bukanlah penemu bola lampu listrik pertama, melainkan penemu lampu pijar komersial pertama yang praktis. (Kata-kata "praktis secara komersial" dijamin akan menarik bagi setiap pelaku bisnis yang menghargai diri sendiri.) Banyak penemu terdahulu yang telah merancang lampu pijar, termasuk demonstrasi kawat pijar yang dilakukan Alessandro Volta pada tahun 1800 (lebih dari delapan dekade sebelum paten Edison).

Bohlam awal ini mempunyai berbagai kelemahan - masa pakai yang sangat pendek, biaya produksi yang tinggi, dan kebutuhan arus listrik yang tinggi - sehingga sulit untuk diterapkan secara komersial dalam skala besar. Setelah banyak percobaan dengan platina dan filamen logam lainnya, Edison kembali menggunakan filamen karbon. Baru beberapa bulan setelah hak paten diberikan, Edison dan timnya menemukan filamen bambu berkarbonisasi yang dapat bertahan lebih dari 1.200 jam.

Apa yang mendorong Edison mempertimbangkan bambu?

Ide penggunaan bambu berawal dari ingatan Edison saat mengamati beberapa benang dari alat pancing bambu, saat bersantai di perjalanan. Tentu saja, para pembaca yang cerdas akan melihat resonansi yang jelas mengenai pentingnya mempersiapkan otak. Bambu adalah bagian yang hilang dari teka-teki tersebut dan, seperti yang selalu terjadi dalam penemuan-penemuan tersebut, bambu bukanlah bagian dari teka-teki yang disertakan dalam kotak teka-teki 'Buat Bola Lampu'.

Strategi

Berikut adalah tiga teknik yang dapat Anda terapkan untuk memaksimalkan hasil ide dengan mengeksplorasi cara membentuk dan memikirkan kembali tim dengan cepat.

1. Temukan pasak persegi untuk lubang bundar
2. Membalikkan hierarki
3. Melarang budaya "Mereka"

Temukan Pasak Persegi untuk Lubang Bulat

Kadang-kadang membantu untuk menemukan seseorang yang tidak cocok dengan tempatnya. Orang luar dapat menawarkan cara-cara baru dalam melihat sesuatu dalam suatu disiplin ilmu. Jika Anda telah dilatih untuk selalu melakukan sesuatu dengan cara tertentu, akan sulit, bahkan tidak mungkin, untuk melihat sesuatu dari sudut pandang yang berbeda.

Salah satu cara untuk menghindari perilaku tersebut adalah dengan secara aktif merekrut anggota tim yang tidak cocok. Kadang-kadang karyawan yang paling berharga adalah mereka yang, di permukaan, tampak sangat tidak produktif, namun bertindak sebagai

katalisator perubahan yang berharga. Sangat mudah untuk menemukan orang yang dapat mengikuti instruksi dan menyelesaikan pekerjaan tepat waktu dan sesuai anggaran. Namun, katalis kreatif yang dapat mengganggu inovasi lebih sulit ditemukan. Jika Anda menemukan pasak yang sempurna, pertahankanlah mereka akan membalas Anda melalui wawasan dan perspektif unik mereka.

Membalikkan Hierarki

CEO memberi tahu CFO siapa yang menanyakan COO yang berkonsultasi dengan CAO… serius, siapa yang menciptakan judul-judul ini? Sementara percakapan ini berputar terus-menerus, seperti adegan di kantor Dilbert, PA hanya mengangkat bahu dan menyelesaikan pekerjaannya.

Jika Anda terjebak pada kebuntuan kreatif, salah satu ide untuk memaksakan perubahan adalah dengan membalikkan hierarki. Berikan kendali kepada mereka yang berada di urutan paling bawah. Tanyakan kepada mereka bagaimana mereka akan melakukan pendekatan terhadap tugas yang ada, Anda mungkin akan terkejut dengan wawasan yang Anda terima. Kebijaksanaan tidak selalu mengalir dari atas ke bawah dan para pemimpin yang hebat memahami hal itu. Bertukar peran dari waktu ke waktu dan rasakan hidup sebagai orang lain. Anda akan menemukan perubahan perspektif mengarah pada pemahaman baru.

Larang Budaya "Mereka"

Jika Anda telah mengadopsi pendekatan yang lancar untuk mengkonfigurasi ulang anggota tim Anda, menganut budaya tabrakan yang tidak disengaja, dan tim Anda bekerja dengan baik, kata "mereka" seharusnya tidak ada. Larang frasa seperti, "Oh, 'mereka' melakukan pekerjaan itu." Jika semua orang bersatu, tidak ada alasan bagi "mereka" untuk ada. Siapa mereka sebenarnya?

Dalam bukunya The Art of Innovation, Tom Kelley dari IDEO menggambarkan sebuah perusahaan transportasi besar yang memutuskan untuk "melonggarkan". Dia menulis: *'Mereka' berpendapat bahwa ada banyak kebingungan dan kecemasan tentang apa yang harus dikenakan pada 'hari Jumat santai'. Jadi 'mereka' membentuk satuan tugas. 'Mereka' mengeluarkan memo yang mengatakan, antara lain, bahwa jika Anda tidak yakin apakah sesuatu itu pantas [dipakai] atau tidak, mungkin itu tidak pantas.*

Dan begitulah yang terjadi… Usir "mereka" dan, apa pun yang Anda lakukan, jangan menjadi seperti mereka.

Pertemuan Singkat

Kita semua pernah mengalami momen ketika menabrak seseorang di koridor, atau berpapasan dengan mereka di kedai kopi, memicu percakapan yang menyatukan berbagai bagian yang mengarah pada sesuatu yang baru. Bentrokan yang tidak disengaja ini – ketika, tanpa Anda sadari, Anda sedang berbincang selama dua jam, memikirkan rencana untuk menguasai dunia adalah saat dimana keajaiban terjadi.

Perusahaan seperti Google, Pixar, dan IDEO memahami hal ini dengan sangat baik dan mengatur ruang mereka untuk mendorong interaksi yang tidak disengaja ini. Stopkontak di tangga, beanbag di ruang terbuka, meja-meja yang tampaknya tidak dimiliki berserakan… semuanya adalah fasilitator. Ide-ide bagus membutuhkan lingkungan yang kreatif dan kacau;

lingkungan yang tidak didasarkan pada hierarki yang ketat; pengaturan yang mencakup alur kerja yang fleksibel dan spontan.

Mendapatkan lingkungan fisik di mana tim bekerja dengan benar adalah sebuah tantangan. Namun, jika didekati dengan cerdas, hal ini dapat menghasilkan penggandaan dan penguatan ide. Walaupun orang-orang mempunyai potongan puzzle yang berbeda-beda, para pemikir yang benar-benar berwawasan luas menyadari bahwa hanya ketika mereka bersatu barulah gambarannya terbentuk. Cara terbaik untuk melakukan hal ini adalah dengan membentuk ruang untuk mendorong pertemuan singkat.

Dengan merancang ruang kerja untuk memanfaatkan pertemuan yang tidak disengaja dan memfasilitasi pola kerja baru, kita dapat memastikan otak kita yang prima bersatu untuk menciptakan ide-ide yang tidak terduga dan berpotensi. Anda tidak perlu memiliki anggaran Google untuk menerapkan hal-hal ini; Anda malah dapat memupuk budaya kantor yang lebih bebas atau mungkin sungguh luar biasa mengizinkan karyawan Anda untuk sesekali bekerja di luar kampus.

Tidak mengherankan jika kita mulai memahami potensi kreatif ruang untuk menggairahkan generasi ide. Bagaimanapun, saat ini kita tidak kekurangan perusahaan yang bisa dijadikan sumber inspirasi. Entah itu atrium raksasa Pixar, yang digunakan Steve Jobs untuk mendorong terjadinya tabrakan, atau tangga Google di kantornya di Kota New York, yang dirancang untuk mendorong para Googler untuk “bertabrakan dengan santai” sepanjang hari kerja (mungkin saat turun), kami memiliki banyak ruang untuk belajar.

Dua ruang yang lebih kecil namun tidak kalah kuatnya adalah d.school di Institute of Design di Stanford, dan model inovatif Erik Spiekermann untuk studio yang sempurna, “Kantor Terpusat”, yang menyalurkan seluruh stafnya dengan sengaja melewati satu sama lain setiap kali mereka bekerja. hari. D.school terdiri dari semacam desain ruang fleksibel yang perlahan tapi pasti mempengaruhi desain kantor kreatif di seluruh dunia. Kantor Terpusat saat ini hanyalah sebuah konsep, gagasan Spiekermann tentang ruang kerja yang sempurna – semoga saja ia menaruh antusiasme dan semangatnya untuk mewujudkannya. Saya, misalnya, ingin mengunjunginya.

Sekolah D

D.school di Institute of Design di Stanford memiliki manifesto sederhana, yang pas di bagian belakang serbet. Bunyinya, secara sederhana:

Niat kami:

- Menciptakan sekolah desain terbaik. Periode.
- Mempersiapkan inovator masa depan untuk menjadi pemikir dan pelaku terobosan
- Gunakan pemikiran desain untuk menginspirasi tim multidisiplin
- Menumbuhkan kolaborasi radikal antara mahasiswa, fakultas & industri
- Tangani proyek-proyek besar dan gunakan pembuatan prototipe untuk menemukan solusi baru

Didirikan pada tahun 2004 oleh David Kelley, (ketua dan mitra pengelola firma desain multidisiplin IDEO yang dihormati secara internasional), pendekatan d.school terhadap desain ruang kreatif telah memberikan informasi bagi ruang kreatif di seluruh dunia. D.school dimulai

dari “trailer satu kamar yang bobrok di pinggiran kampus Stanford.” Dalam setahun, perusahaan tersebut harus pindah ke ruang yang lebih besar seiring dengan meningkatnya permintaan untuk proyek dan kelasnya.

Sikap sekolah terhadap ruang sebagai alat yang cair, yang dirancang untuk membentuk kembali dan membentuk kerja kolaboratif terangkum dengan rapi dalam pemikiran sekolah tentang cara “Membuat Ruang untuk Inovasi”:

D.school[…] mendekati ruang sebagai alat untuk mempengaruhi perilaku siswa agar bias terhadap tindakan[…] ketika dimanipulasi dengan sengaja, ruang dapat digunakan sebagai alat untuk mendorong proses kreatif dengan mendorong dan mencegah perilaku/tindakan tertentu dan dengan menciptakan tempat untuk ekspresi emosional dan negosiasi fisik. Dengan disposisi ini, d.school[…] mengeksplorasi penggunaan artefak, penataan, dan ruang fisik aktual dari lingkungan yang dirancang, untuk mendukung peran ruang sebagai guru.

Sebuah ruang yang dimaksudkan untuk memicu kreativitas, pendekatan d.school (yang didasari oleh eksperimen pendirinya David Kelley sebelumnya dengan desain tata ruang di IDEO) telah menyebar ke lingkungan bisnis kreatif lain yang lebih besar, di mana ruang semakin dipandang sebagai alat yang sangat penting dalam menciptakan kreativitas. proses pembentukan ide. Berjiwa murah hati, sekolah telah menciptakan serangkaian panduan yang memungkinkan Anda membuat peralatan sendiri untuk membentuk ruang. Mendorong pihak lain untuk mengikuti jejak mereka dan menciptakan ruang yang mendorong eksplorasi terbuka dan memfasilitasi tabrakan yang tidak disengaja, d.school menawarkan saran berikut:

Menurut kami, faktor terpenting dalam menciptakan ruang inovasi adalah memulai: mulai dari hal kecil, dan mulai sekarang. Apa yang kami pelajari dari satu tahun di trailer membentuk dasar pendekatan kami terhadap luar angkasa; jika kita menunggu sampai kita memiliki gedung baru yang besar, maka proses pembelajaran akan memakan waktu lima tahun sebelum dimulai. Buatlah ruang Anda!

Pendekatan d.school yang berpikiran terbuka dan inventif terhadap ruang, dan bagaimana hal tersebut dapat digunakan sebagai alat lain dalam kotak alat budaya ide, terwujud melalui proyek siswanya dan kesuksesan mereka yang berkelanjutan. Dengan panduan DIY bersumber terbuka dan buku Make Space karya Scott Doorley dan Scott Witthoft, rahasianya sebenarnya bukanlah rahasia. Anda tidak punya alasan untuk tidak mengambil salinannya dan mulai memahami serta mengeluarkan potensi kreatif ruang jika digunakan dengan baik. Seperti yang dikatakan oleh para d.schoolers: “Buatlah ruangmu!”

Kantor Terpusat

Jika Anda pernah merasa senang mendengar Erik Spiekermann berbicara, Anda akan tahu bahwa ia bukan hanya seorang pembicara yang sangat menarik dan berpengetahuan luas (penghibur adalah istilah yang jauh lebih baik), namun ia juga memiliki visi untuk studio yang sempurna.

Menulis dalam Budaya Studio: Kehidupan Rahasia Studio Desain Grafis, kritikus desain Adrian Shaughnessy menyatakan: *“Tidak seperti biasanya di kalangan desainer kontemporer, Spiekermann memiliki serangkaian teori canggih yang berkaitan dengan tata letak, struktur,*

dan manajemen studio desain. Teori-teorinya telah diuji secara ekstensif di berbagai perusahaan kreatif yang ia dirikan dan jalankan selama kariernya yang panjang."

Selama tahun 1970-an Spiekermann bekerja sebagai desainer lepas di London sebelum kembali ke Berlin pada tahun 1979 di mana, bersama dua mitranya, ia mendirikan MetaDesign. Pada tahun 2001 ia meninggalkan MetaDesign dan memulai UDN (United Designers Network), dengan kantor di Berlin, London dan San Francisco. Sejak Januari 2009 ia menjadi direktur Edenspiekermann, yang mempekerjakan lebih dari 100 orang dan memiliki kantor di Berlin dan Amsterdam.

Kantor Terpusat Spiekermann mengumpulkan pengalaman seumur hidupnya menjalankan agensi desain baik besar maupun kecil, dan mengusulkan desain untuk studio yang sempurna. Desain ini memecahkan banyak masalah yang sering dihadapi oleh kantor, yaitu: tim yang berbeda sering kali terpisah; jarang sekali terjadi tabrakan; hierarki manajemen sering kali (secara sengaja atau tidak sengaja) menyebabkan pemisahan fisik berdasarkan pangkat; dan masih banyak lagi.

FormFiftyFive mempunyai video yang luar biasa "Erik Spiekermann on the Centralized Office" di mana Spiekermann, dengan sikap yang kuat, mengikuti alur pemikiran yang membawanya ke visi studionya, sambil membuat sketsa ide-idenya, memberikan bentuk visinya. Saya mendorong Anda meluangkan waktu sepuluh menit untuk menontonnya.

Jadi, seperti apa kantornya?

Kantor Terpusat berbentuk bulat, terdiri dari tiga atau empat lingkaran konsentris. Di tengahnya terdapat area resepsionis, tempat masuknya semua karyawan dan pengunjung. Dengan menyalurkan semua orang karyawan di semua tingkatan serta klien melalui area pusat ini, kemungkinan terjadinya tabrakan dapat terjadi. Yang lebih memicu terjadinya tabrakan ini adalah pusat kantor, jantungnya yang berdebar kencang, adalah tempat "semua mesin berada," seperti yang dikatakan Spiekermann. Yang paling utama di antara mesin ini adalah printer komputer dan mesin espresso. (Mesin espresso adalah sesuatu yang sangat disukai Spiekermann *"Selalu berinvestasi pada mesin espresso termahal yang dapat Anda beli!".* Kopi yang enak akan selalu membuat Anda bersemangat dari meja kerja Anda.)

Menempatkan printer dan kopi di jantung Kantor Terpusat merupakan sebuah pemikiran strategis yang hati-hati, mendorong aliran dari pinggiran kantor ke pusat dan kembali lagi. Seperti studio Pixar yang secara kreatif dirancang untuk "mempromosikan pertemuan dan kolaborasi yang tidak direncanakan," inti dari visi Spiekermann memiliki tujuan yang sama, untuk menyatukan semua orang.

Biografi Jobs karya Walter Isaacson menggemakan visi ini. Isaacson menyatakan: Jobs percaya bahwa, "*Jika sebuah bangunan tidak mendorong [kolaborasi], Anda akan kehilangan banyak inovasi dan keajaiban yang dipicu oleh kebetulan. Jadi kami merancang gedung ini untuk membuat orang keluar dari kantor mereka dan berbaur di atrium pusat dengan orang-orang yang mungkin tidak mereka lihat."*

Brad Bird, sutradara "The Incredibles", mengatakan: "*Atrium awalnya mungkin tampak seperti membuang-buang ruang, namun Steve menyadari bahwa ketika orang bertemu satu sama lain, ketika mereka melakukan kontak mata, banyak hal terjadi*."

Pemahaman tentang memfasilitasi pertemuan singkat ini semakin ditingkatkan di Kantor Terpusat melalui pembangunan tembok yang cermat. Meskipun strukturnya konsentris, dinding kantor tidak mencapai langit-langit. Seperti yang dikatakan Spiekermann: "Dindingnya hanya setinggi bahu. Jika seorang sekretaris ingin melihat apakah saya berada di lingkaran luar, dia dapat berdiri dan melihat ke seberang dan melihat apakah saya benar-benar ada di sana."

Bukan suatu kebetulan bahwa ruang-ruang ini, mulai dari ruang yang lebih kecil di d.school dan Kantor Terpusat, hingga ruang yang lebih besar di Apple dan Pixar, mendorong generasi ide dan pertukaran ide dalam struktur desain mereka. Ini adalah pabrik ide sejati, yang dimaksudkan untuk memupuk elemen paling ajaib yang sulit dipahami, yaitu percikan kreatif. Dengan membangun ruang-ruang yang mendorong terjadinya benturan-benturan yang tidak disengaja, kita pada akhirnya dapat menyatukan ketiga aspek proses pembangkitan ide, memupuk dan memupuk semangat kreatif, membuka ide-ide yang terpendam dalam diri kita. Membuka pintu air, jika Anda mau.

Strategi

Saya ingin menyarankan tiga teknik yang dapat Anda terapkan untuk memikirkan kembali bagaimana ruang diatur untuk berkontribusi pada budaya ide.

1. Budaya kafe
2. Kantor Lego
3. Ruang kosong

Budaya Kafe

Keluarlah, keluarlah sedikit lagi dan hilangkan kartu-kartu punch itu. Marissa Mayer mungkin telah melarang teleworking di Yahoo! (sambil merancang ulang branding perusahaan sendirian), namun bukan berarti Anda tidak bisa sedikit pun melonggarkan ikatan di tempat kerja.

Dengan mengatur pertemuan sesekali di luar kampus dengan rekan kerja, Anda menjauh dari meja kerja dan sedikit melepaskan pemikiran Anda. Ini tidak hanya merupakan latihan yang sangat dibutuhkan, tetapi juga memungkinkan Anda untuk berbicara jauh dari ruang dan sedikit melepaskan diri. Tidak mengherankan jika kita mendengar tentang startup yang tumbuh di kedai kopi; budaya kafe tidak hanya menyediakan Wi-Fi gratis, tetapi juga menyediakan lingkungan yang jauh dari para manajer mikro yang menghambat kreativitas.

Kantor Lego

Mengapa membatasi diri Anda pada kantor yang tidak sesuai rencana, dengan dinding putih pucat dan karpet biasa, jika Anda dapat membangun kantor sendiri, besar atau kecil, sesuai keinginan Anda? Google memahami bahwa karyawan tidak dapat memenuhi kebutuhan semua orang dengan menyediakan landasan atau Blox untuk menciptakan kantor yang sempurna atau ruang kerja tim ad hoc.

BloXes, kotak karton saling bertautan yang dirancang untuk membangun ruang kerja yang fleksibel, adalah penemuan Jef Raskin (yang memulai proyek Macintosh di Apple). Mereka mungkin tidak bisa mencegah hujan, tapi pasti bisa mencegah kebosanan.

Ruang bebas

Tahan keinginan untuk merencanakan setiap inci persegi kantor Anda. Biarkan sudut, atau ruang terbuka lainnya, tetap tidak terisi, dengan layar atau papan tulis yang mudah dipindahkan, sehingga tim ad hoc dapat keluar dari meja mereka dan berkolaborasi. Tidak ada yang menyukai peternakan bilik dan Anda tidak akan memberikan manfaat apa pun kepada karyawan Anda jika Anda membangunnya. Kosongkan ruang kantor dan gunakan dengan lebih fleksibel; budaya ide Anda akan meningkat dan tim Anda akan berterima kasih.

Kesimpulan

Ide tidak terwujud dalam ruang hampa. Tanpa masukan yang konstan, keluaran Anda pasti akan tetap sama. Oleh karena itu, penting untuk mempertahankan pikiran yang ingin tahu, memastikan aliran pemicu dan rangsangan baru yang memungkinkan pemikiran Anda berkembang. Perluas kumpulan gen ide dan Anda akan memperdalam sumber ide yang mampu Anda ciptakan.

Demikian pula, ide-ide terbaik seringkali merupakan hasil dari tim yang tidak takut untuk melakukan konfigurasi ulang secara berkala (baik dari proyek ke proyek atau, jika jalan buntu kreatif telah tercapai, di tengah proyek, untuk membantu menggoyahkan segalanya).). Pertimbangkan peran konduktor untuk mengatur tim, memastikan mereka memberikan lebih dari sekedar jumlah bagian mereka. Jika cara ini berhasil untuk perusahaan seperti IDEO, yang secara rutin membentuk ulang tim agar masukan tetap bervariasi, cara ini bisa berhasil untuk Anda.

Terakhir, pertimbangkan ruang kerja. Anda tidak perlu mengeluarkan anggaran multi-miliar dolar untuk kampus baru Apple di Cupertino, tetapi itu tidak berarti Anda tidak bisa memikirkan bagaimana ruang Anda memfasilitasi tabrakan yang tidak disengaja. Pada tingkat yang paling sederhana, hal ini dapat berupa perubahan pemandangan, bekerja di kedai kopi, atau bahkan di taman. Sebuah perubahan sama baiknya dengan istirahat dan kadang-kadang hanya melihat sesuatu dari sudut pandang yang berbeda secara harfiah dapat membuat perbedaan besar.

Menghasilkan ide tidaklah sulit, dan jika Anda mengikuti strategi yang diuraikan di atas, akan sangat mudah untuk merangsang budaya ide. Yang perlu Anda lakukan hanyalah menempatkan potongan-potongan itu dengan cara yang cerdas. Pabrik ide dan ruang ide kreatif mudah dibangun jika didekati secara strategis. Fakta bahwa mereka berhasil telah dibuktikan berulang kali: Thomas Edison membuktikannya di Laboratorium Menlo Park miliknya pada akhir abad ke-19; dan Steve Jobs dengan tegas menggarisbawahi pemikiran tersebut di awal abad ke-21.

PRESPEKTIF BARU DALAM
DESAIN WEB

Dr. Mars Caroline Wibowo. S.T., M.Mm.Tech

Bio Data Penulis

Penulis lahir di Semarang pada tanggal 1 Maret 1983. Penulis menempuh pendidikan Sarjana Teknik Elektro di Universitas Kristen Satya Wacana (UKSW), lulus tahun 2004, kemudian tahun 2005 melanjutkan studi pada Magister Desain pada Fakultas Seni Rupa dan Desain, Institut Teknologi Bandung (ITB), dan melanjutkan studi pada program studi Teknologi Multimedia pada Swinburne University of Technology Australia.

Penulis sejak tahun 2010, menjadi dosen pada program studi Desain Grafis Universitas Sains dan Teknologi Komputer (Universitas STEKOM), memiliki jabatan fungsional Lektor 300 dan sedang proses mengajukan kenaikan jabatan fungsional menjadi Lektor Kepala. Penulis juga seorang wirausaha di bidang toko online yang berhasil di kota Semarang dan juga aktif sebagai freelancer dalam bidang fotografi, web design dan multimedia.

YAYASAN PRIMA AGUS TEKNIK

PENERBIT :
YAYASAN PRIMA AGUS TEKNIK
Jl. Majapahit No. 605 Semarang
Telp. (024) 6723456. Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

ISBN 978-623-8642-06-9 (PDF)