BUKU AJAR

Interface USER EXPERIENCE

DISUSUN OLEH
HIDAYATULAH HIMAWAN
MANGARAS YANU F

Lembaga Penelitian & Pengabdian Kepada Masyarakat
Universitas Pembangunan Nasional "Veteran" Yogyakarta

BUKU AJAR

Interface USER EXPERIENCE

DISUSUN OLEH
HIDAYATULAH HIMAWAN
MANGARAS YANU F

Lembaga Penelitian & Pengabdian Kepada Masyarakat
Universitas Pembangunan Nasional "Veteran" Yogyakarta

Hidayatulah Himawan
Mangaras Yanu F.

Cetakan Pertama, 2020
ISBN: 978-623-7594-55-0

Diterbitkan oleh:
Lembaga Penelitian dan Pengabdian kepada Masyarakat
UPN Veteran Yogyakarta
Jl. SWK 104 (Lingkar Utara), Condongcatur , Yogyakarta, 55283
Telp. (0274) 486188,486733, Fax. (0274) 486400

PRAKATA

Puji syukur kehadirat Allah SWT atas limpahan rahmat dan karunia-Nya sehingga Buku Ajar dengan topik bahasan "User Interface dan User Experience" dengan judul "**Interface USER EXPERIENCE**" ini telah selesai disusun. Buku ajar ini merupakan dokumentasi hasil penelitian yang telah dilakukan oleh penyusun, yang diharapkan dapat dijadikan sebagai pedoman untuk mengajar mata kuliah dengan topik bahasan "User Interface dan User Experience".

Terimakasih kami sampaikan kepada Universitas Pembangunan Nasional "Veteran" Yogyakarta khususnya pada Lembaga Penelitian dan Pengabdian Kepada Masyarakat atas dukungan pendanaan sehingga kami dapat menyelesaikan penelitian yang kami jadikan sebagai dasar dalam penyusunan buku ini. Terimakasih juga kami sampaikan kepada semua pihak yang tidak bisa kami sebutkan satu-persatu yang telah berkontribusi baik dalam penelitian yang telah kami lakukan maupun dalam penyusunan buku ini.

Kami menyadari masih terdapat kekurangan dalam buku ini, untuk itu kritik dan saran terhadap penyempurnaan buku ini sangat diharapkan. Semoga buku ini dapat memberi manfaat bagi Universitas Pembangunan Nasional "Veteran" Yogyakarta khususnya dan bagi semua pihak yang membutuhkan.

Yogyakarta, Januari 2020

Penyusun

DAFTAR ISI

BAB I DEFINISI UI UX

Sasaran Pembelajaran

Mahasiswa mampu menguraikan dan menjelaskan definisi User Interface dan User Experience (UI/UX)

Kemampuan mahasiswa yang menjadi prasyarat

Mahasiswa sudah menguasai materi Interaksi Manusia Komputer dan menguasai tools untuk mendesain

Keterkaitan bahan pembelajaran dengan pokok bahasan lainnya

Materi ini sebagai dasar untuk mempelajari pokok bahasan lainnya

Manfaat atau pentingnya bahan pembelajaran ini

Memberi pemahaman pengertian UI UX

1.1 Definisi User Interface

UI atau User Interface merupakan mekanisme komunikasi antara pengguna (user) dengan sistem pada sebuah program, baik itu aplikasi website, mobile, ataupun software. Mekanisme itu disesuaikan dengan kebutuhan pengguna terhadap program yang tengah dikembangkan. Cakupan UI itu meliputi tampilan fisik, penggunaan warna, tampilan animasi, hingga pola komunikasi suatu program dengan penggunanya.

Biasanya, seorang desainer UI akan membuat desain yang kiranya memudahkan pengguna programnya. Adapun, desain itu disesuaikan dengan tingkat kebutuhan dasar pengguna terhadap program aplikasi web ataupun mobile tersebut. Output dari hasil desainer UI ialah program dengan segala fitur yang kiranya sesuai dengan kebutuhan pengguna dalam menggunakan program tersebut.

1.2 Definisi User Experience

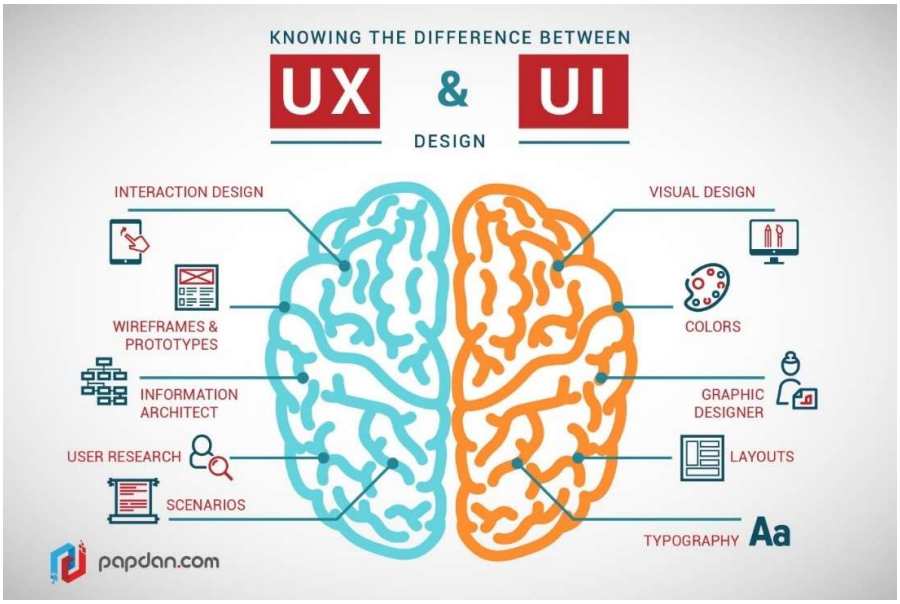
Pengertian UX atau User Experience memang tidak terlalu jauh berbeda dengan UI. Perbedaannya terletak pada fokus utama hubungan komunikasi antara pengguna dengan programnya, yakni berfokus pada pengalaman penggunanya.

Seorang desainer UX akan merancang program aplikasi web atau mobile berdasarkan pengalaman dari pengguna atau user setelah menggunakan aplikasi web atau mobile tersebut. Dengan begitu, program yang dirancang menjadi lebih mudah digunakan oleh penggunanya.

User experience (UX) sesuai artinya dalam bahasa indonesia “pengalaman pengguna” adalah pengalaman yang diberikan website atau software kepada penggunanya agar interaksi yang dilakukan menarik dan menyenangkan. Kalau dulu aplikasi mempunyai usability yang bagus saja sudah cukup. Sekarang sebuah aplikasi juga harus memiliki user experience yang bagus.

Seperti apa user experience itu? Saat membuka Instagram sampai berjam-jam tanpa bosan, saat chatting menggunaan WhatApp tanpa henti, saat berlama-lama mencari barang-barang jualan di toko online, berarti sudah menikmati user experience yang sudah diberikan oleh Instagram, WhatsApp dan juga toko online. Kenapa juga bisa berjam-jam sibuk dengan smartphone? Itu semua karena penerapan user experience dalam smartphone sudah sangat baik.

1.3 Perbedaan UI dan UX



Sebelumnya sudah disinggung bahwa perbedaan antara UI dan UX berada pada fokus utamanya. Bila UI fokus pada interaksi pengguna dengan programnya, maka UX fokusnya pada pengalaman pengguna dalam menggunakan suatu aplikasi web atau mobile.

Seorang desainer UI akan mendesain program aplikasi web atau mobile sesuai dengan kebutuhan pengguna. Sehingga, ketika menggunakan program tersebut pengguna lebih mudah dan tidak kesulitan.

Sedangkan, desainer UX membuat program berdasarkan pengalaman dari penggunanya. Apa saja yang dirasakan dan kesulitan apa saja yang dihadapi ketika menggunakan program tersebut.

Sebenarnya, keduanya memiliki tujuan yang sama dalam mendesain program aplikasi web ataupun mobile, yakni memudahkan penggunannya. Oleh sebab itu, seringkali dalam proses perancangan sebuah program, desainer UI dan UX selalu berada dalam satu tim. Sebab, dengan

perpaduan keduanya, sebuah program aplikasi web ataupun mobile menjadi sangat mudah digunakan oleh pengguna tanpa harus membaca panduan. Seringkali, desainer UI dan UX bertukar data analisis untuk menyempurnakan program yang tengah dibuatnya.

Jadi, UI dan UX sebenarnya berbeda. Perbedaannya pada fokus utama. UI fokus pada kebutuhan pengguna terhadap program aplikasi aplikasi web atau mobile, sedangkan UX fokus pada pengalaman pengguna.

Pada dasarnya, User Experience adalah tentang "memahami pengguna". Tujuan UX adalah mencari tahu siapa mereka, apa yang mereka capai dan apa cara terbaik bagi mereka untuk melakukan "sesuatu".

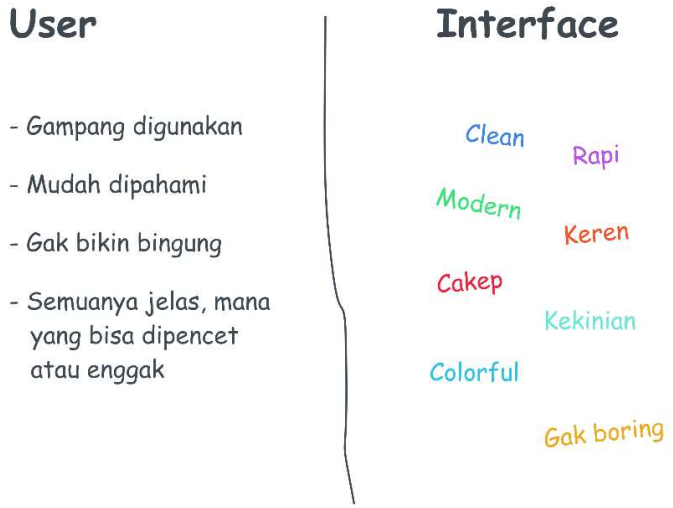
UX berkonsentrasi pada bagaimana sebuah produk terasa dan apakah itu memecahkan masalah bagi pengguna.

Sedangkan User Interface adalah bagaimana suatu website atau aplikasi yang dibuat terlihat dan berbentuk seperti apa. Hal tersebut mencakup Layout (tata letak), Visual Design (desain visual) dan Branding.

Mengerti perbedaan antara UI Design dan UX Design, bukan sekedar untuk teori, tapi akan berpengaruh pada proses design. Beberapa orang (kalau bukan kebanyakan) menganggap design itu hanya terkait warna, pemilihan font, gambar/foto, dan icon. Padahal UX Design itu jauh melebihi warna dan sebagainya.

1.4 UI Design

Seperti namanya, (UI) User Interface. Maka ada 2 hal yang harus diperhatikan oleh seorang designer saat membuat UI, yaitu User dan Interface. Berikut gambarannya:



Untuk menghasilkan UI (User Interface) yang baik, Anda harus memperhatikan 2 hal diatas. Kata User ditempatkan di depan kata Interface, karena UI yang baik selalu memperhatikan dan mengutamakan user. UI yang baik akan membantu user. Dan, UI yang baik akan membuat user nyaman menggunakannya.

1.5 Memulai Menjadi UI Designer

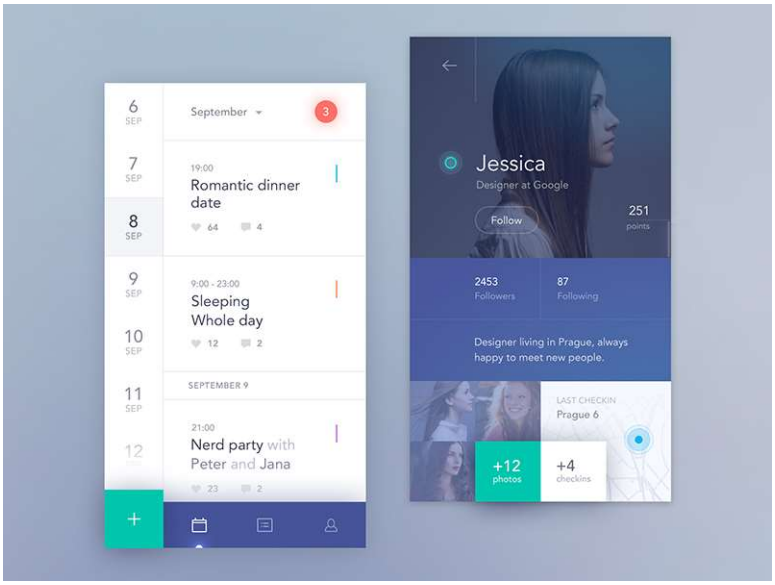
Untuk memulai menjadi UI Designer Anda boleh men-skip bagian User-nya dan fokus di Interface-nya dulu. Dalam pembuatan UI, Anda harus memikirkan tentang bagaimana nanti user memakainya. Contohnya: Sebuah Tombol.

Saat men-desain sebuah tombol, Anda harus memikirkan "Apakah user tahu kalau ini tombol yang bisa di-klik?". Tapi, hal yang tidak kalah penting adala Anda harus tahu bagaimana bentuk tombol itu. Maka dari itu, sebagai permulaan UI untuk fokus di bagian Interface-nya terlebih dahulu agar Anda familiar dengan bentuk-bentuk UI.

Apakah Anda familiar dengan Photoshop? GIMP? atau software sejenis lainnya?. Jika belum, maka sebaiknya Anda mulai belajar menggunakan software tersebut.

Langkah pertama untuk menjadi UI Designer adalah mencoba untuk membuat UI. Tidak perlu bingung mau mulai dari mana atau mau mendesain apa. Anda cukup memilih desain-desain interface yang ada di internet. Pilih yang Anda suka lalu jiplaklah, buat semirip mungkin.

Anda bisa browsing di situs seperti dribbble.com, behance.net atau kreavi.com. Banyak sekali desain UI disana.



Kenapa harus menjiplak desain?

Hal ini untuk membuat Anda familiar dengan UI. Bagaimana bentuk tombol, ukuran teks, jarak antar baris kalimat, jarak tiap elemennya.

Jadi, Semakin sering Anda menjiplak sebuah desain, maka Anda akan semakin familiar dengan ukuran setiap elemen UI. Tapi ingat, jiplak menjiplak ini hanya untuk kepentingan belajar. BUKAN untuk diupload di sosial media ataupun situs portfolio.

Jika Anda telah menjiplak UI berkali-kali, maka Anda sudah terbiasa membuat tombol, terbiasa membuat dropdown, input text, dan elemen-elemen UI lainnya. Kini, saatnya Anda untuk membuat UI Anda sendiri.

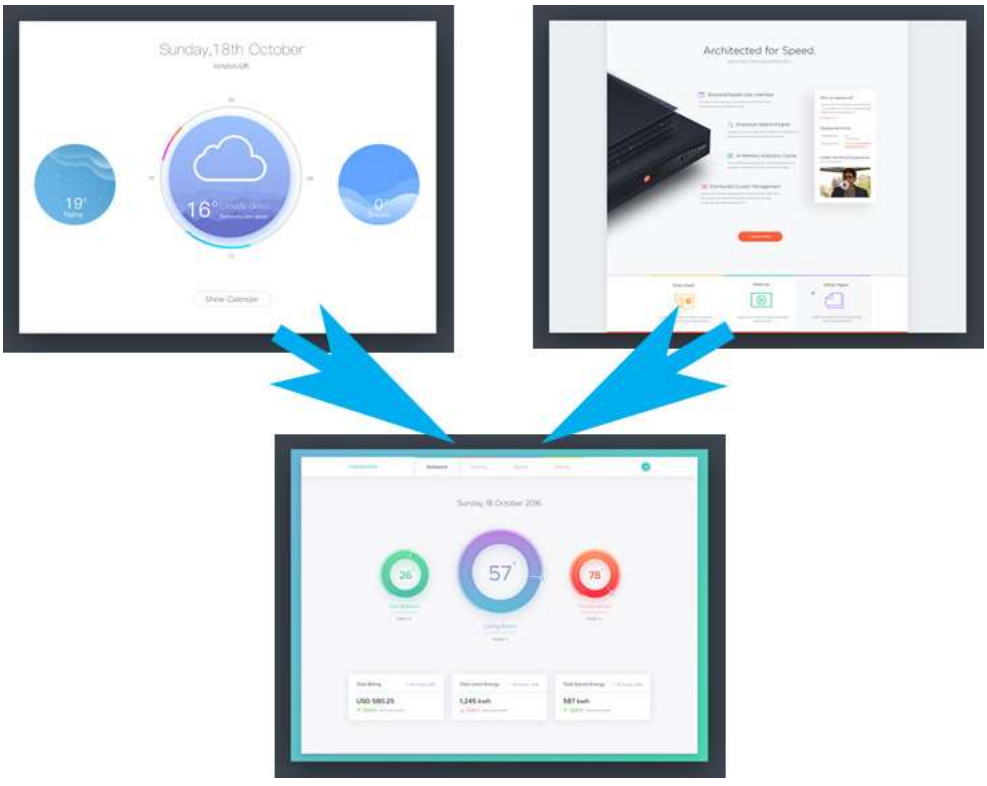
Caranya gampang, Pertama, tentukan platform terlebih dahulu. Anda ingin membuat apa? design untuk mobile app (Android/iOS), design untuk web, atau bahkan design untuk smartwatch app?

Setelah itu, tentukan tema design-nya. Anda ingin membuat tampilan untuk aplikasi kesehatan? website artikel teknologi? atau tampilan untuk aplikasi pemesanan makanan?

Setelah platform dan tema sudah dipilih, kini saatnya Anda browsing lagi design-design yang Anda sukai. Tapi kali ini, carilah design yang sesuai dengan platform dan tema yang Anda pilih tadi.

Lalu pilih 2 design yang paling Anda sukai, dan coba buat sesuatu yang baru dari hasil kombinasi dua design tersebut.

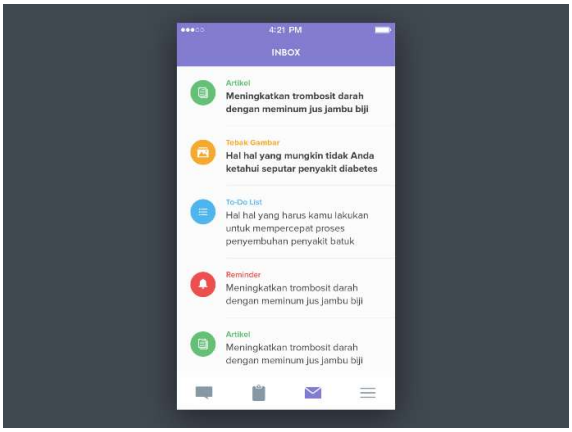
Contohnya seperti ini:



Saat mendesain sebuah UI, pikirkanlah bagaimana nanti user memakainya. Untuk melatih hal itu, biasakanlah mendesain dengan sebuah alasan. Apa artinya? Design yang Anda buat,

komponen yang Anda buat, sebaiknya memiliki alasan yang kuat kenapa diletakkan seperti itu atau berbentuk seperti itu.

Contoh 1:



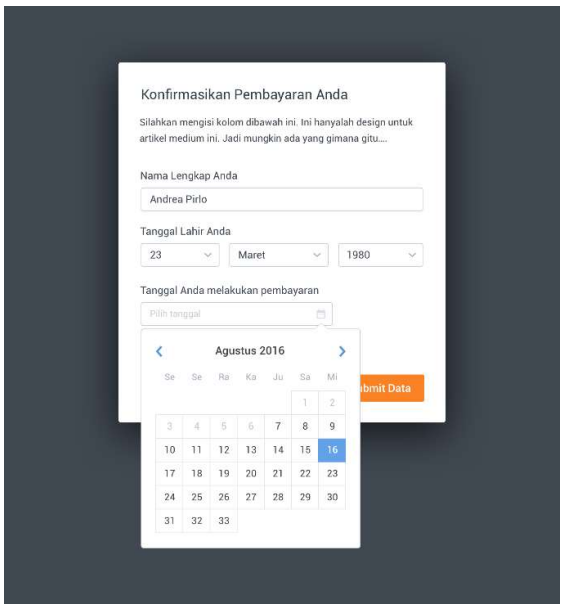
Kenapa menu nya ada dibawah?

Karena app ini memiliki 3 menu utama. Dan dalam penggunaan app ini, user cenderung berpindah-pindah dari menu satu ke menu utama lain-nya dalam selang waktu relatif singkat. Maka dari itu, untuk mempermudah user, 3 menu utama ditaruh di bawah dan sisanya ditaruh di menu more yang terletak di paling kanan.

Kenapa ada tulisan yang tebal dan ada yang tidak tebal?

Ini adalah menu Inbox, tulisan yang tebal sebagai penanda belum dibaca. Dan yang tidak tebal sebagai penanda sudah pernah dibaca. Sehingga memudahkan user untuk membedakannya.

Contoh 2:



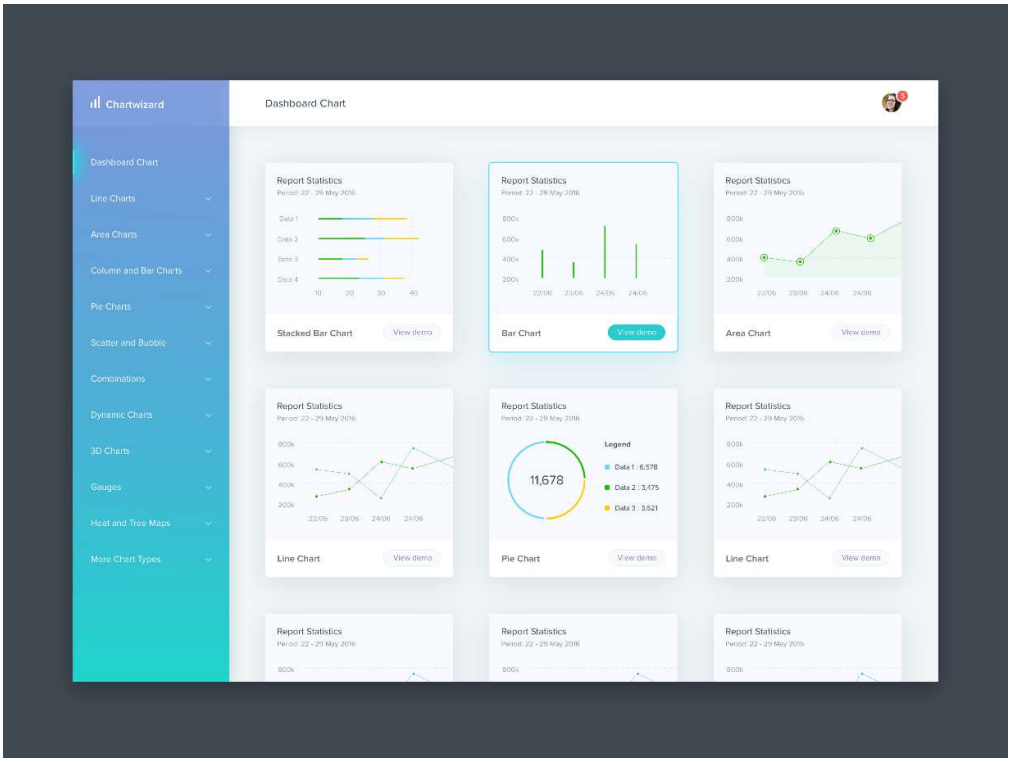
Ada dua inputan tanggal, tapi kenapa beda? Padahal kan sama-sama memilih tanggal.

Coba bayangkan jika Anda menjadi user. Untuk mengisi tanggal lahir, lebih mudah memakaiformat pengisian yang atas atau yang bawah? Tentu lebih mudah memakai format yang atas. Karena, Anda ingat tanggal lahir Anda. Dan dengan cepat akan mengisi hari, bulan dan tahunnya. Tapi, coba bayangkan jika menggunakan format pengisian yang bawah, berapa kali Anda harus mengklik tombol back untuk menuju bulan Maret 1980.

Lalu untuk inputan tanggal pembayaran, kenapa format pengisian-nya harus seperti itu?

Ini adalah form konfirmasi pembayaran, setelah user melakukan pembayaran, biasanya user akan langsung mengkonfirmasikan pembayaran-nya. Coba bayangkan, jika user harus mengisi dengan format pengisian seperti tanggal lahir. Cukup merepotkan user, karena dia harus mengingat hari ini tanggal berapa. Tapi dengan format pengisian seperti di design, ketika user memilih tanggal, maka akan terbuka kalender bulan ini, dengan tanggal hari ini ditandai biru. Sehingga user dengan mudah memilih tanggal pembayarannya.

Contoh 3:



Kenapa menu nya ada di kiri dan ke bawah? apakah karena ini dashboard yang tren-nya selalu menaruh menu di sebelah kiri?

Menaruh menu di kiri dan vertical ke bawah, pertimbangannya adalah skalabilitas. Sebelum merancang design ini, kami menganalisa bahwa kedepannya, menu menu di dashboard ini akan bertambah dan pertambahannya bisa cukup banyak. Jika membuat design menu-nya secara horizontal, maka kelak jika menunya sangat banyak, akan terjadi masalah pada layout-nya.

Tentu Anda tidak ingin melihat 2 baris menu terletak di bagian atas. Maka dari itu, diputuskan untuk membuat design menu-nya secara vertical ke bawah. Jadi, jika ada pertambahan menu, tidak akan terjadi masalah pada layout-nya.

1.6 Mengapa User Experience Itu Penting?

Memudahkan pengguna

Penerapan user experience akan memudahkan pengguna dalam menggunakan aplikasi. Karena didalamnya sudah ada penilaian aspek usability. Setiap aplikasi pastilah dibuat agar para pengguna mudah untuk menggunakannya.

Menarik minat pengguna

Selain faktor kemudahan penggunaan, penerapan user experience juga untuk menarik minat pengguna. Tujuan aplikasi dibuat sudah pasti ingin penggunanya selalu menggunakan aplikasinya. Jika aplikasi tidak menarik untuk pengguna, sudah pasti bisa dengan mudah akan ditinggalkan.

Berdampak pada faktor kesuksesan

Seperti pada poin diatas, pengguna akan mudah meninggalkan aplikasi yang tidak memberikan pengalaman menarik. Contoh aplikasi chatting WhatsApp. Aplikasi ini makin hari makin banyak penggunanya karena memberikan pengalaman pengguna yang menarik. Kita bisa bandingan dengan aplikasi sejenis yang penggunanya semakin menurun setiap hari. Oleh karena itu, user experience penting diterapkan pada sebuah aplikasi untuk meningkatkan kesuksesan atau minimal mempertahankan kesuksesan.

Menghasilkan UI yang bagus

User interface itu merupakan keluaran dari penerapan user experience. Jika user experience pada sebuah aplikasi benar-benar diperhatikan penerpannya, maka akan menghasilkan desain UI yang bagus. Bagus disini tidak harus warna-warni dan bling-bling, namun secara tampilan akan elegan dan menarik.

Untuk memenangkan persaingan

Apakah Anda mengetahui mengapa toko ritel seperti indomaret dan alfamart bisa memenangkan persainan dari toko di sekelilingnya? Itu karena mereka menerapkan user experience yang bagus. Padahal barang yang dijual sama, secara harga bisa lebih mahal, tapi secara pengalaman yang mereka berikan itu tidak ada di toko biasa.

Contoh lain lagi yaitu antara Android dan IOS. Mengapa orang mau bayar mahal smartphone ini? Karena user experience yang mereka berikan memang terbaik. Berikut beberapa hasil penelitian mengenai pentingnya UX:

1. Menurut penelitian dari Imaginovation, sebuah lembaga penelitian berbasis di Amerika: Jika konten Anda tidak dioptimalkan dengan baik, sebanyak 79% pengunjung akan keluar dari website Anda dan mencari konten/produk lainnya.
2. Menurut penelitian dari lembaga riset HubSpot: pengguna ponsel 5X lebih punya kecenderungan untuk meninggalkan website Anda jika website tidak dioptimalkan agar sesuai dengan perangkat yang mereka punya. (Gawat kalau setidaknya ada 2/3 pelanggan yang akses website Anda dari ponsel mereka sebenarnya ingin melakukan pembelian pada hari itu juga)
3. Menurut penelitian dari lembaga riset MindTouch: Ini kasus nyata, pendapatan dari website ESPN.com melonjak 35% setelah mereka mendengarkan keluhan pengguna mereka dan mendesain ulang homepage mereka.
4. Menurut Adobe: 39% orang akan berhenti mengakses website jika gambar tidak dimuat-muat atau terlalu lama loading-nya.

Jadi intinya, website Anda sekarang ini fungsinya mirip seperti toko. Bayangkan user sebagai seorang calon pembeli yang masuk ke toko Anda. Jika mereka mengalami pengalaman yang buruk, contohnya si calon pembeli ini tidak bisa menemukan apa yang mereka butuhkan. Tak hanya itu, bisa juga pihak toko tidak berhasil menjangkau calon pembelinya, maka si calon pembeli pasti akan pergi dan tidak akan kembali lagi.

BAB II Dasar-Dasar UX

Sasaran Pembelajaran

Mahasiswa mampu memahami dasar-dasar User Experience (UX)

Kemampuan mahasiswa yang menjadi prasyarat

Mahasiswa sudah menguasai materi Interaksi Manusia Komputer

Keterkaitan bahan pembelajaran dengan pokok bahasan lainnya

Materi ini memperkenalkan alur kerja yang bisa membantu tim, produk, startup dan perusahaan membuat proses yang kuat dan valuable untuk mengembangkan UX yang lebih baik bagi para pelanggan. Anda bisa menggunakan bagian proses yang berbeda secara terpisah, namun proses ini idealnya bekerja sangat baik bila dilakukan dalam serangkaian langkah

Manfaat atau pentingnya bahan pembelajaran ini

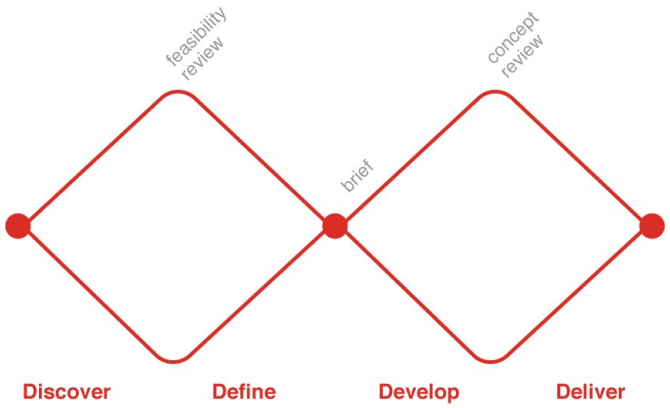
Memberi pemahaman dasar-dasar UX

Petunjuk belajar

Materi ini banyak mengadopsi metodologi Design Sprint yang digunakan beberapa tim di Google untuk memecahkan masalah dan tantangan seperti Self Driving Car dan Project Loon

Double Diamond

Alur kerja ini didasarkan pada apa yang kita sebut dalam lingkaran UX sebagai double diamond, dipopulerkan oleh British Design Council, dengan tim Anda terbagi untuk memahami ide melalui penelitian dan kemudian berkumpul untuk mendefinisikan tantangan, membaginya untuk membuat sketsa secara individual, berbagi ide, memutuskan apa yang terbaik ke depannya, pengujian dan validasi.



Model proses desain "double diamond' yang dipelopori oleh British Design Council, langkah-langkahnya melibatkan tahapan proyek berikut; *Memahami*, *Mendefinisikan*, *Membagi*, *Memutuskan*, *Prototipe* dan *Validasi*.

Menyiapkan langkah

Hal pertama adalah memulai dengan tantangan mendasar dan menulisnya seperti proposal, tanyakan pada diri Anda sendiri, "apa masalah yang sesungguhnya coba saya pecahkan?". Pernyataan tantangan adalah keterangan singkat yang ditetapkan ke proyek yang berisi tujuan Anda.

Tantangan ini bisa fitur produk saat ini yang perlu disaring atau produk yang sama sekali baru. Apapun tugas Anda, cukup sesuaikan bahasa agar sesuai dengan tujuan yang ingin Anda capai. Pernyataan harus dikaitkan dengan tujuan tim Anda, berfokus pada pengguna, memberikan inspirasi dan ringkas.

Berikut adalah beberapa contoh produk nyata yang telah dikerjakan:

- Merancang sebuah sistem untuk mengelola pengobatan dan perawatan lanjutan pasien penderita clubfoot.
- Membuat sebuah aplikasi yang menyederhanakan sistem keuangan kompleks dan menguranginya ke hal-hal penting saja.
- Merancang aplikasi seluler yang konsisten di seluruh platform yang berbeda tanpa mengorbankan merek.

Memperbarui pernyataan tantangan

Setelah Anda menulis beberapa variasi tujuan, presentasikan ke tim Anda untuk mendapatkan sebuah konsensus. Anda mungkin perlu memasukkan batas waktu karena ini akan membantu tim berfokus pada masalah. Jadi dengan penambahan tersebut, penyesuaian untuk daftar di atas bisa menjadi:

- Merancang sebuah sistem untuk mengelola pengobatan dan perawatan lanjutan anak-anak di bawah usia 2 tahun penderita clubfoot diluncurkan pada Q1 tahun ini.
- Membuat aplikasi keuangan sederhana yang memungkinkan Anda membeli dan menjual saham cukup dengan mengetuk tombol tanpa membutuhkan pengetahuan dasar dunia keuangan, dengan peluncuran awal Juli 2017.
- Menghasilkan panduan desain yang fleksibel di beberapa platform dan memosisikan merek perusahaan secara efektif pada setiap platform hingga akhir tahun ini.

Ketika pernyataan tantangan selesai, tampilkan dalam tempat yang menonjol sehingga Anda bisa melihatnya saat bekerja. Anda harus memeriksanya kembali secara konstan, bahkan mungkin memperbarui atau memodifikasinya selama proyek Anda berjalan.

Memvalidasi masalah

Langkah berikutnya adalah meneliti tantangan dan mempelajari masalah tersebut. Apa yang perlu Anda ketahui adalah apakah pemahaman tim Anda tentang masalah adalah valid. Cukup sering kita melihat masalah dari sudut pandang kita sendiri, yang berbahaya karena kebanyakan dari kita di dunia teknologi sebenarnya adalah power user dan pada kenyataannya merupakan pengguna minoritas. Kita adalah minoritas vokal dan bisa tertipu saat berpikir sesuatu dapat menjadi masalah padahal tidak.

Ada berbagai metode pengumpulan data untuk memvalidasi tantangan. Masing-masing bergantung pada tim dan jika Anda memiliki akses ke pengguna. Tujuannya adalah untuk mendapatkan pemahaman yang lebih baik dari masalah yang dihadapi.

Wawancara internal dengan para pemangku kepentingan

Wawancara dengan para pemangku kepentingan bisa informatif untuk menemukan wawasan dalam sebuah perusahaan atau tim. Proses wawancara termasuk melakukan wawancara kepada setiap anggota tim dan pemangku kepentingan di perusahaan Anda, dari pemasaran hingga keuangan. Ini akan membantu Anda menemukan apa yang mereka pikir tantangan nyata dan

apa solusi potensial yang bisa mereka pikirkan. Ketika saya mengatakan solusi, saya tidak berbicara tentang solusi teknis di sini, melainkan apa yang bisa menjadi skenario terbaik dan tujuan akhir bagi perusahaan atau produk. Misalnya menggunakan tantangan di atas "memiliki software clubfoot di 80% fasilitas medis hingga akhir tahun ini" dapat menjadi tujuan besar yang menjadi target.

Ada sebuah peringatan. Metode validasi adalah yang paling tidak disukai karena menghambat diskusi dan kolaborasi tim, berpotensi menciptakan suasana tertutup dalam sebuah organisasi. Meskipun demikian, ini bisa menghasilkan beberapa informasi bagus tentang klien dan tantangan desain yang bisa saja Anda lewatkan.

Presentasi kilat

Presentasi kilat adalah presentasi sangat singkat yang hanya berlangsung beberapa menit. Mirip dengan wawancara internal, namun kali ini Anda menghadirkan setiap pemangku kepentingan dalam satu ruangan. Kemudian Anda Memilih lima atau enam orang pemangku kepentingan (pemasaran, penjualan, desain, keuangan, penelitian dll.) untuk berbicara, masing-masing berfokus pada tantangan dari perspektif mereka selama maksimal 10 menit. Topiknya harus mencakup presentasi mereka:

- Tujuan bisnis
- Tantangan proyek dari sudut pandang mereka (ini bisa faktor teknis, pengumpulan penelitian, pembuatan desain dll..)
- Penelitian pengguna yang Anda miliki saat ini

Berikan waktu 5 menit di akhir untuk sesi pertanyaan, dengan orang yang dipilih mencatat semuanya. Setelah selesai, Anda mungkin ingin memperbarui tantangan untuk merefleksikan pembelajaran yang baru. Tujuannya adalah untuk mengumpulkan daftar poin-poin utama yang bisa mendorong fitur atau alur yang membantu Anda mencapai tujuan produk.

Wawancara pengguna

Wawancara pengguna adalah cara yang bagus untuk mempelajari tentang titik derita orang di setiap tugas yang diberikan.

Ini mungkin adalah cara terbaik untuk belajar tentang pengalaman pengguna, titik derita, dan alur. Aturlah setidaknya lima wawancara pengguna, lebih banyak lagi jika Anda memiliki akses kepada mereka. Jenis pertanyaan yang Anda tanyakan kepada mereka harus mencakup:

- Bagaimana mereka menyelesaikan tugas yang ada? Misalnya, Anda ingin menyelesaikan tantangan untuk aplikasi keuangan di atas, Anda bisa bertanya kepada mereka "bagaimana Anda membeli saham dan efek saat ini?"
- Apa yang mereka suka tentang alur ini?
- Apa yang tidak mereka suka tentang alur ini?
- Apa produk sejenis yang saat ini digunakan pengguna?
- Apa yang mereka suka?
- Apa yang tidak mereka suka?
- Jika mereka memiliki tongkat ajaib dan bisa mengubah satu hal tentang proses ini hal apakah itu?

Ide melakukan wawancara adalah agar pengguna berbicara tentang tantangan yang mereka alami. Ini bukanlah poin diskusi untuk Anda, itulah mengapa Anda harus tetap diam. Hal ini semakin benar ketika pengguna berhenti berbicara, selalu berikan waktu sebentar karena mereka bisa saja sedang mengumpulkan pemikirannya. Anda akan terkejut melihat betapa banyak orang yang akan terus berbicara setelah berhenti sejenak selama beberapa detik.

Catat seluruhnya dan jika mungkin rekam percakapan tersebut untuk membantu Anda merekam apa pun yang mungkin Anda lewatkan. Tujuannya adalah membandingkan tantangan terhadap wawasan pengguna yang Anda kumpulkan. Apakah mereka selaras? Apakah Anda mempelajari sesuatu yang membantu memperbarui pernyataan tantangan?

Penelitian bidang etnografi

Melihat pengguna dalam lingkungan alami mereka adalah cara yang bagus untuk memahami bagaimana pengguna mengatasi tantangan mereka sendiri.

Ini adalah bidang tempat Anda mengamati pengguna, dalam konteks saat melakukan sesuatu seperti bagaimana mereka berbelanja, bagaimana mereka melakukan perjalanan ke tempat kerja, bagaimana mereka mengirim pesan SMS dll.. Alasannya adalah karena dalam beberapa kasus orang akan memberi tahu apa yang mereka pikir ingin Anda dengarkan. Namun jika Anda menyaksikan sendiri pengguna melakukan tindakan dan tugasnya, ini bisa menjadi penuh wawasan. Pada dasarnya Anda mengamati tanpa mengganggu, mencatat hal-hal yang mereka rasa mudah atau sulit dan hal-hal yang mungkin mereka lewatkan. Tujuannya adalah untuk melibatkan diri Anda dalam lingkungan pengguna agar lebih berempati dengan titik derita mereka.

Teknik ini biasanya melibatkan beberapa pekerjaan yang dilakukan selama periode waktu yang lebih lama dan membutuhkan peneliti untuk memimpin bagian proyek ini. Namun inilah yang mungkin paling berwawasan karena Anda bisa melihat sekelompok orang yang Anda pelajari di lingkungan alami mereka.

Mengumpulkan semuanya

Setelah Anda menyelesaikan tahap pembelajaran proyek, Anda harus mengambil satu pemeriksaan terakhir pada tantangan Anda. Apakah Anda di jalur yang benar? Apakah ada

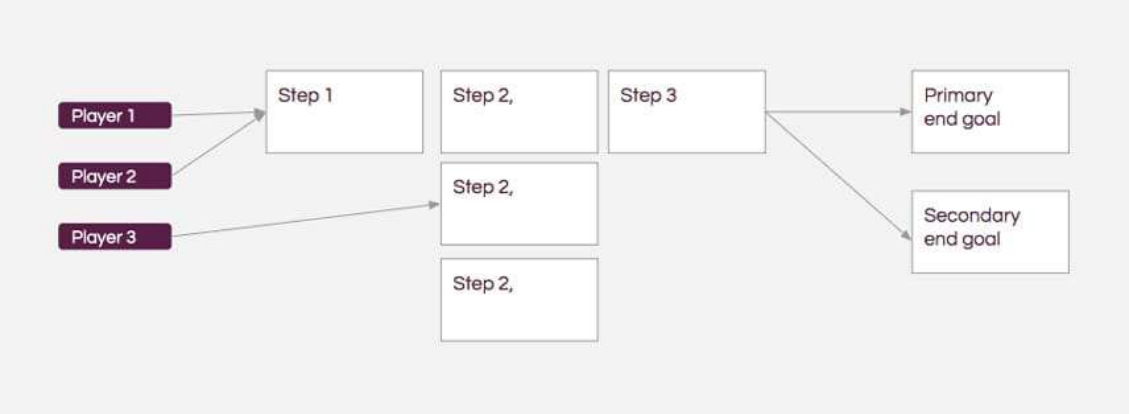
sesuatu yang perlu Anda sesuaikan? Tuliskan semua hal yang telah Anda pelajari dan kelompokkan mereka ke dalam kategori. Ini bisa menjadi dasar dari fitur atau alur, bergantung pada masalah yang Anda selesaikan. Juga bisa digunakan untuk memperbarui dan merevisi tantangan.

Setelah Anda memiliki masukan dan wawasan yang cukup, saatnya untuk menerapkan pengetahuan itu untuk membuat pemetaan proyek.

Pemetaan proyek

Masalah yang coba Anda selesaikan biasanya terdiri dari berbagai tipe orang (atau pemain), masing-masing dengan andil di alur proyek. Berdasarkan pembelajaran, Anda perlu mendaftar para pemain. Ini bisa jadi tipe pengguna atau pemangku kepentingan, misalnya, "dokter yang merawat clubfoot", "pasien yang menderita clubfoot", "perawat yang merawat pasien", dll.. Tuliskan masing-masing pemain di sisi kiri selembar kertas atau tulis pada papan tulis jika Anda memilikinya. Di sisi sebelah kanan, tuliskan tujuan masing-masing pemain.

Yang terakhir untuk setiap pemain, tuliskan jumlah langkah yang diperlukan untuk mencapai tujuan mereka. Misalnya untuk "dokter yang merawat clubfoot" tujuannya adalah "menyembuhkan pasien yang menderita clubfoot", sehingga langkah-langkahnya adalah "mendaftarkan pasien dalam sistem", "memulai rencana kesehatan", "membuat siklus ulasan kesehatan medis" dan "melakukan prosedur medis".



Pemetaan proyek merencanakan langkah-langkah utama untuk setiap pengguna atau pemain dalam alur.

Hasilnya adalah pemetaan proyek dengan langkah-langkah utama dalam prosesnya. Anggap saja itu sebagai ringkasan proyek tanpa terlalu banyak detail. Ini juga memungkinkan anggota tim menilai apakah pemetaan cocok dengan pernyataan tantangan. Kemudian, ketika Anda memecah setiap langkahnya, akan ada detail lebih lanjut. Namun untuk saat ini, pemetaan proyek memberikan Anda rincian tingkat tinggi dari langkah yang perlu diambil pengguna untuk menyelesaikan tujuan akhir mereka.

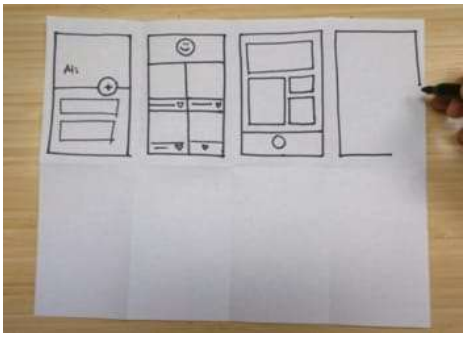
Wireframing dan storyboarding

Crazy 8s

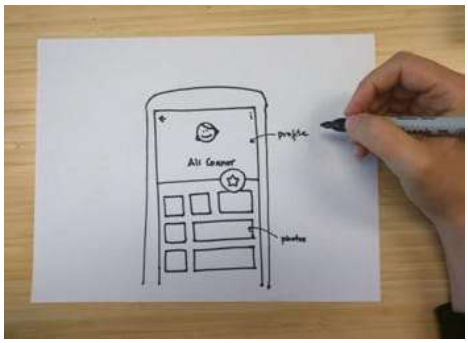
Untuk ini, disarankan metode yang disebut crazy 8s yang meliputi pelipatan kertas dua kali lebih banyak sehingga Anda memiliki delapan panel. Kemudian pada setiap panelnya Anda menggambar sebuah ide berdasarkan semua yang telah Anda pelajari sejauh ini. Berikan diri Anda sepuluh menit agar muncul dengan ide-ide untuk mengisi semua panel yang berjumlah delapan. Jika Anda memberikan diri Anda waktu lebih dari 20 menit, Anda bisa mulai menunda-nunda, pergi membuat kopi, memeriksa email, mengobrol dengan tim Anda dan pada dasarnya menghindari melakukan pekerjaan. Anda ingin menciptakan rasa urgensi dalam langkah ini karena memaksa Anda untuk bekerja dengan cepat dan lebih efektif.

Jika Anda bekerja dengan tim, suruh mereka semua untuk melakukan hal ini juga. Proses ini akan menyentak otak Anda dan membuat Anda berpikir tentang tantangan. Biasanya sketsa akan menjadi wireframe desain antarmuka.

Setelahnya, Anda dan semua orang di tim menyajikan ide-idenya ke kelompok. Setiap orang harus menjelaskan masing-masing delapan ide mereka secara rinci dan mengapa mereka memilih untuk mengambil jalur tersebut. Ingatkan setiap anggota tim untuk menggunakan pembelajaran untuk pembenaran ide-ide mereka. Setelah semua orang mengemukakan idenya, saatnya memilih ide-ide tersebut. Setiap orang mendapat dua titik tempelan dan bisa memberikan suara pada ide mana pun. Mereka bisa memberikan kedua suaranya untuk sebuah ide jika mereka benar-benar menyukainya.



Crazy 8s adalah cara yang bagus untuk memasukkan semua ide Anda ke dalam laman.

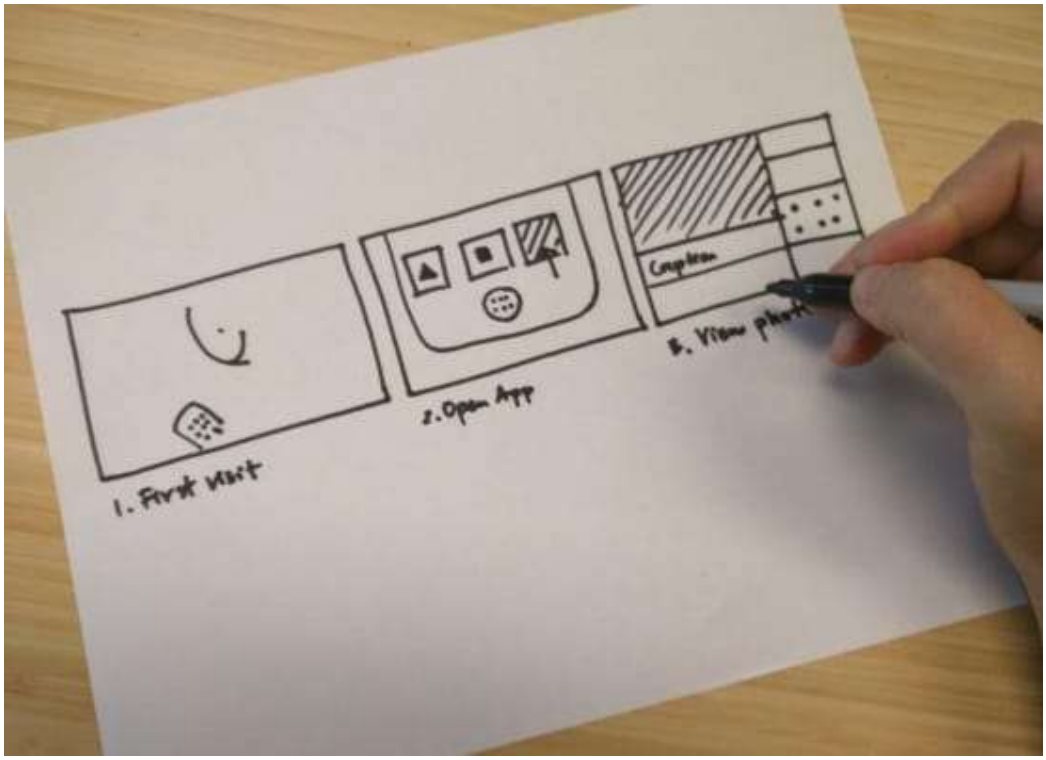


Sekarang Anda harus melakukan desain terperinci berdasarkan apa yang telah Anda pelajari.

Menyaring desain Anda

Setelah pemungutan suara mengambil ide dengan suara terbanyak dan membuat sketsa ide akhir. Anda juga bisa meminjam ide lain yang Anda dengar dari rekan kerja. Berikan diri Anda waktu sepuluh menit untuk menyelesaikan tugas ini. Setelah selesai, presentasikan kembali ide ini ke tim Anda dan lakukan pemungutan suara seperti sebelumnya.

Membuat storyboard ide



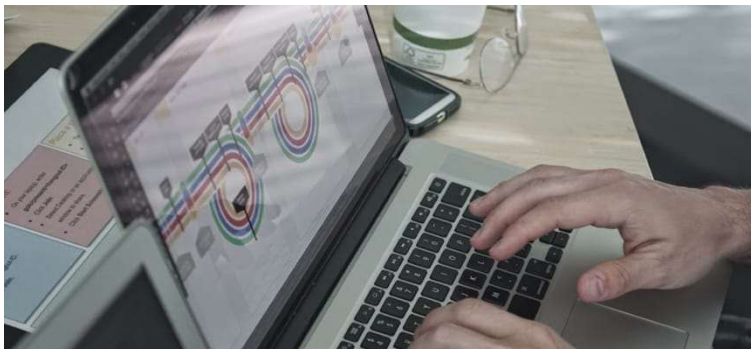
Storyboard melibatkan perpaduan sketsa dan ide Anda ke dalam alur komprehensif.

Dengan desain di tangan, saatnya untuk membuat storyboard interaksi dengan pengguna. Di titik ini Anda sebaiknya sudah berpikir tentang langkah berbeda yang diambil pengguna. Cukup biasa untuk menggabungkan salah satu dari desain rekan Anda ke dalam alur. Anda memerlukan proses langkah demi langkah yang jelas dengan beberapa titik di mana pengguna mungkin berbeda. Lihat kembali pemetaan proyek untuk memvalidasi desain terhadap tujuan Anda.

Membuat prototipe

Membuat prototipe bukan tentang menciptakan potongan kode yang sempurna, namun untuk membuat sesuatu yang bisa dipercaya bila digunakan oleh seseorang. Alat yang digunakan untuk membuat prototipe berbeda dari orang ke orang. Beberapa alat seperti Keynote atau Powerpoint karena memaksa Anda untuk memikirkan alur dan tidak merancang detail. Anda mungkin ingin meluangkan waktu untuk mempelajari alat seperti Balsamiq, Marvel atau Framer yang bisa memberikan kontrol perilaku yang lebih banyak. Apapun alat (bantu) yang

Anda gunakan pastikan itu adalah alat yang membuat Anda fokus pada alur dan terlihat nyata. Anda harus menguji prototipe pada pengguna yang nyata sehingga sebisa mungkin dapat dipercaya tapi pada saat yang bersamaan tidak memerlukan berminggu-minggu jam kerja untuk dibuat.



Prototipe harus cukup nyata untuk bisa dipercaya.

Membuat prototipe adalah keseimbangan antara waktu dan realitas, jadi berhati-hatilah agar tidak melenceng ke salah satu sisi secara ekstrim. Bila tidak, waktu Anda bisa saja terbuang percuma.

Pengujian-kegunaan desain Anda

Akan bagus sekali jika Anda memiliki lab pengujian. Bila tidak, membuat lab tidak sulit asalkan Anda memperhatikan pembuatan lingkungan yang nyaman bagi pengguna serta tidak mengganggu mereka. Pengujian biasanya melibatkan pengguna dan dua orang dari tim Anda, satu mencatat dan lainnya mengajukan pertanyaan. Persiapan yang baik adalah dengan menggunakan aplikasi seperti Hangouts dan merekam tindakannya, ini juga berguna jika Anda menginginkan seluruh tim untuk mengamati dari ruangan yang berbeda. Hal ini cukup menakutkan bagi kami sebagai pembuat aplikasi untuk melakukannya saat kami melihat desain kami keluar di alam liar. Ini bisa menjadi pengalaman yang menyegarkan dan menenangkan.

Storyboard termasuk menempatkan semua sketsa dan ide bersama-sama ke dalam alur yang komprehensif.

Pertanyaan untuk ditanyakan

Saat menguji desain Anda, minta pengguna untuk melakukan tugas di aplikasi dan minta mereka agar berbicara dengan suara keras serta mengungkapkan apa yang mereka lakukan dan mengapa. Ini mungkin terdengar aneh dilakukan, namun hal ini membantu Anda mengetahui apa yang mereka pikirkan. Cobalah untuk tidak mengganggu atau memberi tahu mereka apa yang harus dilakukan saat mereka terhenti. Cukup tanyakan kepada pengguna mengapa mereka mengambil alur tertentu setelah mereka menyelesaikan (atau TIDAK menyelesaikan).

Apa yang perlu Anda ketahui:

- Apa yang mereka suka dari prototipe?
- Apa yang mereka tidak suka dari prototipe?
- Apa saja titik deritanya?
- Mengapa alur bekerja
- Mengapa alur tidak bekerja
- Apa yang ingin mereka tingkatkan?
- Apakah keseluruhan desain/alur memenuhi kebutuhan mereka?

Mengunjungi kembali desain dan rentetan pengujian lagi

Anda memiliki prototipe yang bekerja dengan masukan. Sekarang saatnya merevisi desain Anda, dan menganalisis apa yang berhasil dan apa yang tidak. Jangan takut untuk membuat storyboard wireframe yang benar-benar baru dan membuat prototipe baru. Memulai lagi dari awal bisa membuat alur yang lebih baik dibandingkan mencoba untuk memindahkan sesuatu pada prototipe Anda sebelumnya. Cobalah agar jangan terlalu sayang karena itu hanyalah prototipe.

Setelah puas dengan desain, Anda bisa mengujinya lagi dan menyempurnakannya lagi. Dalam kasus di mana prototipe sama sekali tidak mencapai target, Anda mungkin berpikir proyek itu gagal. Nyatanya, tidak. Anda mungkin menghabiskan waktu development lebih sedikit dibandingkan jika Anda telah membangun desain dan mengetahui lebih banyak tentang apa yang benar-benar disukai pengguna. Dengan design sprints, kami memiliki filosofi yaitu Anda menang atau Anda belajar, jadi jangan terlalu menyalahkan diri sendiri jika ide tersebut tidak bekerja seperti yang direncanakan.

Buatlah!

Anda telah menguji ide. Pengguna menyukainya. Pemangku kepentingan berinvestasi karena mereka telah terlibat sejak awal. Sekarang saat yang tepat untuk membuatnya. Sekarang, Anda harus memiliki gagasan yang jelas tentang apa yang perlu dilakukan dan apa prioritas dari pengalaman ini. Pada setiap tonggak bersejarah proyek, Anda mungkin ingin memperkenalkan pengujian kegunaan untuk membantu memvalidasi pekerjaan dan menjaga Anda tetap di jalur.

Saya tidak bisa menekankan betapa pentingnya mencari tahu sebanyak mungkin informasi sebelum Anda berkomitmen untuk banyak pekerjaan, waktu dan energi pada sesuatu yang mungkin saja tidak menjadi solusi yang tepat.

Artikel ini seharusnya bisa memberikan landasan dasar bagi Anda tentang UX dan arti pentingnya. UX bukanlah sesuatu yang harus dipandang sebagai peran seorang desainer atau peneliti. Ini sebenarnya adalah tanggung jawab semua orang yang terlibat dalam proyek sehingga saya selalu merekomendasikan keterlibatan dalam setiap kesempatan.

BAB III STUDI KASUS

Sasaran Pembelajaran

Mahasiswa mampu secara khusus menjelaskan dan memahami obyek yang ditelitinya secara khusus sebagai suatu 'kasus'

Kemampuan mahasiswa yang menjadi prasyarat

Mahasiswa sudah menguasai materi Interaksi Manusia Komputer

Keterkaitan bahan pembelajaran dengan pokok bahasan lainnya

Materi ini memperkenalkan alur kerja yang bisa membantu tim, produk, startup dan perusahaan membuat proses yang kuat dan valuable untuk mengembangkan UX yang lebih baik bagi para pelanggan.

Manfaat atau pentingnya bahan pembelajaran ini

Mahasiswa dapat mencapai penyesuaian diri yang lebih baik

Apa Yang Membuat Sebuah Situs Seluler Bagus?

Pengguna seluler sangat berorientasi pada tujuan. Mereka berharap bisa mendapatkan apa yang mereka butuhkan, dengan segera, dan dengan cara mereka sendiri.

Penelitian ini berlangsung selama 119 jam per-orang dengan partisipan dari AS. Para partisipan diminta untuk melakukan tugas-tugas kunci di berbagai situs seluler. Termasuk pengguna iOS dan Android, dan pengguna menguji situs tersebut di ponsel mereka sendiri. Untuk setiap situs, para partisipan diminta untuk menyuarakan pemikiran mereka dengan keras karena mereka menyelesaikan tugas-tugas yang berfokus pada konversi seperti melakukan pembelian atau pemesanan reservasi.

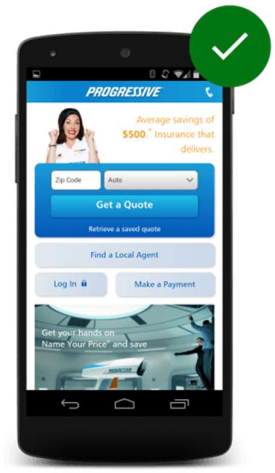
Penelitian ini menemukan 25 prinsip desain situs seluler, dikelompokkan ke dalam lima kategori.

Navigasi situs dan laman Beranda

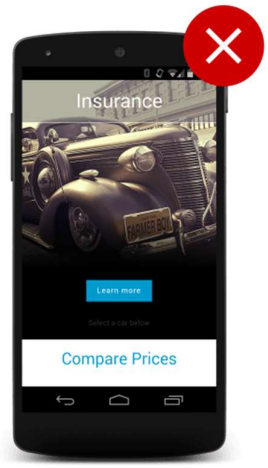
Indikator berhasil: Fokuskan beranda seluler Anda pada cara menghubungkan pengguna ke materi yang mereka cari.

Pertahankan panggilan untuk aksi di depan dan tengah

Menyediakan tugas sekunder melalui menu atau "paro bawah" (bagian dari laman web yang tidak bisa dilihat tanpa gulir ke bawah).

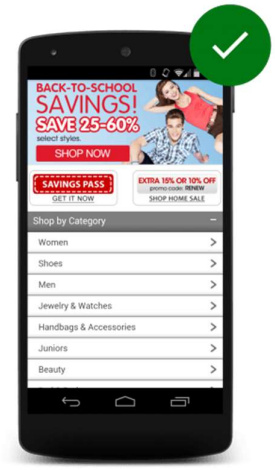


LAKUKAN: Memudahkan ketersediaan semua tugas yang paling sering dipakai pengguna.



JANGAN: Membuang ruang berharga paro-atas dengan panggil-untuk-aksi tidak jelas seperti "ketahui selengkapnya".

Pertahankan menu yang singkat dan manis



LAKUKAN: Pertahankan menu yang singkat dan manis.

Pengguna seluler tidak memiliki kesabaran untuk menggulir melalui daftar panjang opsi untuk menemukan apa yang mereka inginkan. Tata ulang menu Anda agar menggunakan item sesedikit mungkin, tanpa harus mengorbankan kegunaan.

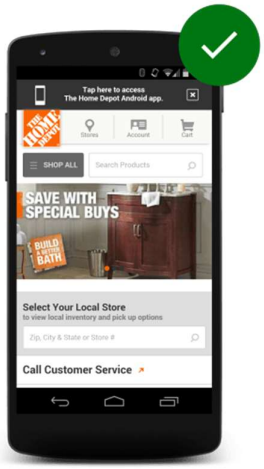
Mempermudah cara kembali ke laman beranda

LAKUKAN: Mempermudah cara kembali ke laman beranda.

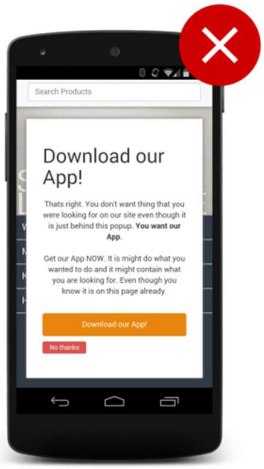
Pengguna ingin kembali ke beranda ketika mereka mengetuk logo di sudut kiri atas laman seluler, dan mereka bisa frustrasi bila tidak tersedia atau tidak bekerja.

Jangan biarkan promosi mencuri perhatian

Pengantara pemasangan aplikasi besar (mis. promosi selayar-penuh yang menyembunyikan materi dan meminta pengguna untuk memasang aplikasi) menjengkelkan pengguna dan mempersulit saat melakukan tugas. Selain menjengkelkan pengguna, situs yang menggunakan pengantara pemasangan aplikasi tidak akan lolos Uji Ramah Google Seluler, yang bisa berdampak negatif terhadap peringkat penelusuran mereka.



LAKUKAN: Promosi harus mudah ditutup dan tidak mengganggu pengalaman pengguna.



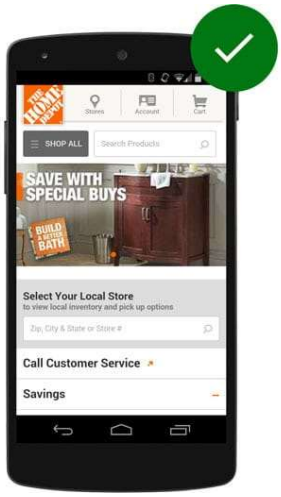
JANGAN: Pengantara (kadang-kadang disebut membanting pintu) sering menjengkelkan pengguna dan membuat menggunakan situs adalah sebuah penderitaan.

Penelusuran situs

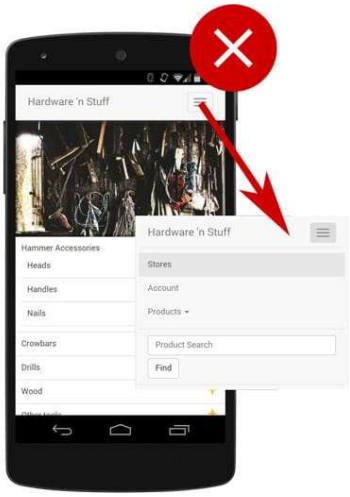
Indikator berhasil: Membantu pengguna seluler menemukan apa yang mereka cari dengan sangat cepat.

Membuat penelusuran situs terlihat

Pengguna yang mencari informasi biasanya membuka penelusuran, sehingga bidang penelusuran harus menjadi salah satu item utama yang mereka lihat di laman Anda. Jangan menyembunyikan kotak telusur di menu.



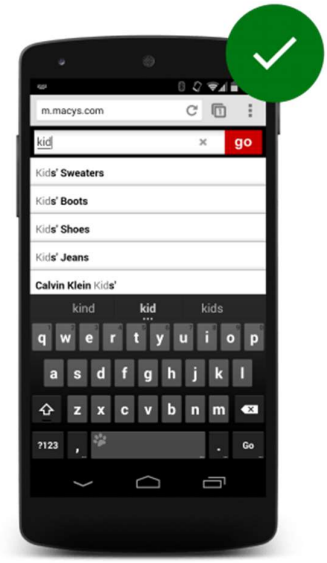
LAKUKAN: Membuat penelusuran terlihat



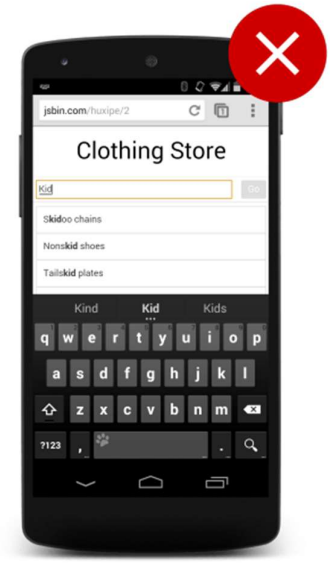
JANGAN: Menyembunyikan penelusuran di menu luapan

Pastikan hasil penelusuran situs relevan

Pengguna tidak memindai beberapa laman dari hasil penelusuran untuk menemukan apa yang mereka cari. Permudah pengguna dengan menyelesaikan-otomatis kueri, mengoreksi kesalahan eja, dan menyarankan kueri terkait. Daripada menciptakan kembali sesuatu yang sudah ada, pertimbangkan produk yang kuat seperti Google Penelusuran Khusus.



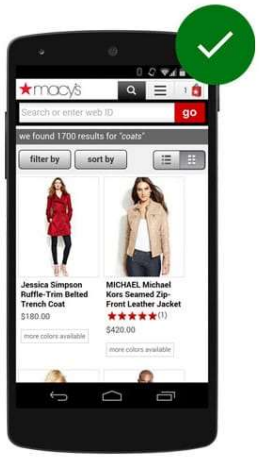
LAKUKAN: Macy hanya mengembalikan barang anak-anak (kids).



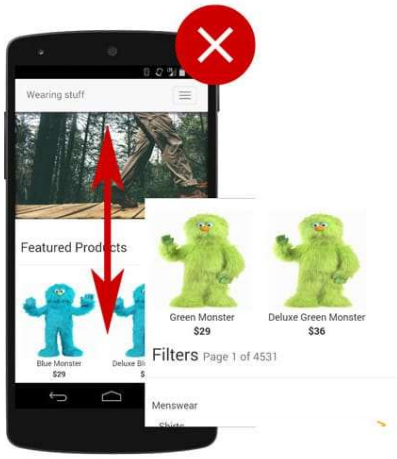
JANGAN: Mengembalikan hasil dengan kata anak (kid) di dalamnya.

Mengimplementasikan filter untuk mempersempit hasil

Partisipan penelitian mengandalkan filter untuk menemukan apa yang mereka cari, dan meninggalkan situs yang tidak memiliki filter yang efektif. Menempatkan filter di atas hasil penelusuran, dan membantu pengguna dengan menampilkan berapa banyak hasil yang dikembalikan ketika filter tertentu diterapkan.

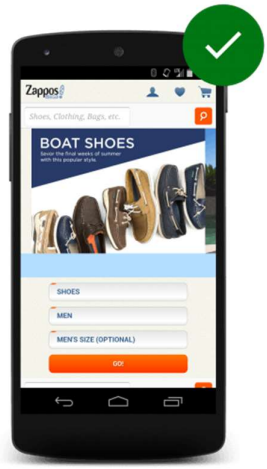


LAKUKAN: Permudah untuk memberi filter.



JANGAN: Menyembunyikan fungsionalitas filter.

Memandu pengguna agar hasil penelusuran situs lebih baik



LAKUKAN: Membantu pengguna untuk menemukan apa yang mereka cari dengan memandu mereka ke arah yang tepat.

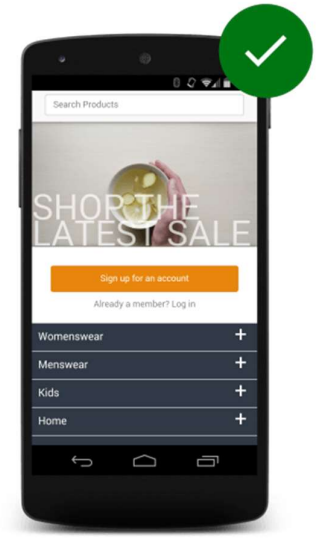
Untuk situs dengan segmen pengguna yang beragam, ajukan beberapa pertanyaan sebelum menyajikan kotak telusur, dan menggunakan respons pengguna sebagai filter kueri penelusuran untuk memastikan bahwa pengguna mendapatkan hasil dari segmen yang paling relevan.

Niaga dan konversi

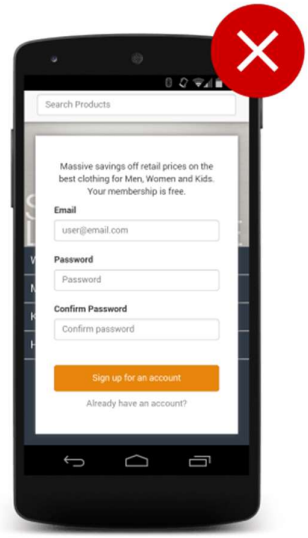
Indikator berhasil: Memahami perjalanan pelanggan Anda dan membiarkan pengguna melakukan konversi dengan cara mereka sendiri.

Biarkan pengguna menjelajahi sebelum mereka berkomitmen

Partisipan penelitian frustrasi oleh situs yang mengharuskan pendaftaran di awal untuk melihat situs, terutama ketika merek tersebut masih terdengar asing. Meskipun informasi pelanggan mungkin integral untuk bisnis Anda, memintanya terlalu dini bisa menyebabkan pendaftaran yang lebih sedikit.

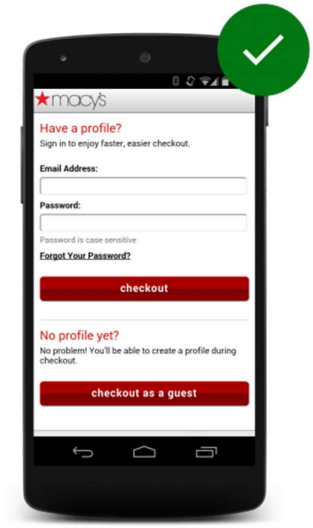


LAKUKAN: Izinkan pengguna menjelajahi situs tanpa harus mendaftar masuk.



JANGAN: Menempatkan login atau registrasi terlalu awal dalam sebuah situs.

Izinkan pengguna membeli sebagai tamu



LAKUKAN: Izinkan pengguna membeli dengan akun tamu.

Partisipan penelitian menganggap checkout tamu "nyaman", "sederhana", "mudah", dan "cepat". Pengguna kesal oleh situs yang memaksa mereka mendaftarkan sebuah akun saat melakukan pembelian, terutama ketika manfaat dari akun tidak jelas.

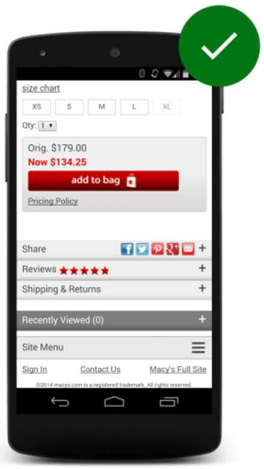
Menggunakan informasi yang ada untuk memaksimalkan kemudahan

Ingat dan pra-isi preferensi untuk pengguna terdaftar. Tawarkan layanan checkout pihak ketiga yang familier untuk pengguna baru.

Menggunakan tombol click-to-call untuk tugas yang kompleks

Pada perangkat dengan kemampuan menelepon, tautan click-to-call memungkinkan pengguna untuk melakukan panggilan telepon hanya dengan mengetuk tautan. Pada kebanyakan perangkat seluler, pengguna menerima konfirmasi sebelum nomor dihubungi, atau tampil menu yang akan menanyakan pengguna bagaimana sebaiknya nomor ditangani.

Buatlah mudah untuk diselesaikan pada perangkat lain



LAKUKAN: Berikan cara mudah bagi pengguna untuk melanjutkan browsing atau berbelanja di perangkat lain.

Pengguna sering kali ingin menyelesaikan tugas pada perangkat lain. Misalnya, mereka mungkin ingin menampilkan item pada layar yang lebih besar. Atau mereka mungkin sedang sibuk dan harus menyelesaikannya nanti. Dukung perjalanan pelanggan ini dengan memungkinkan pengguna untuk berbagi item di jaringan sosial, atau dengan memperbolehkan pengguna meng-email sendiri tautan secara langsung dari dalam situs.

Entri formulir

Indikator berhasil: Sediakan pengalaman konversi yang mulus dan tanpa friksi dengan formulir yang bisa dipakai.

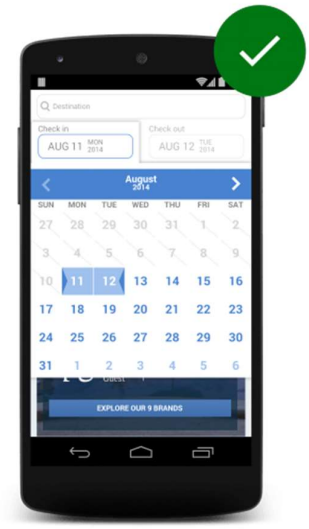
Merampingkan entri informasi

Secara otomatis maju ke bidang berikutnya ketika pengguna menekan Return. Secara umum, semakin sedikit pengguna melakukan ketukan, semakin baik.

Memilih masukan yang paling sederhana

Gunakan tipe masukan yang paling tepat untuk setiap skenario. Gunakan elemen seperti datalist untuk menyediakan nilai yang disarankan untuk bidang.

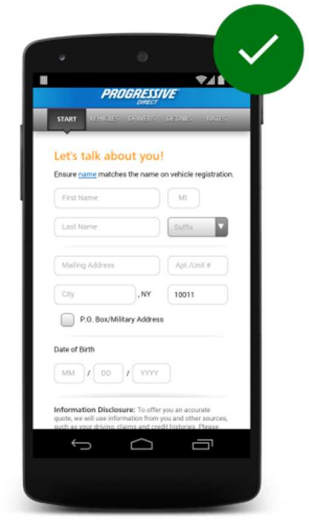
Menyediakan kalender visual untuk pemilihan tanggal



LAKUKAN: gunakan widget kalender jika memungkinkan.

Beri label tanggal awal dan akhir dengan jelas. Pengguna sebaiknya tidak perlu meninggalkan situs dan memeriksa aplikasi kalender hanya untuk menjadwalkan suatu tanggal.

Meminimalkan kesalahan formulir dengan label dan validasi real-time



LAKUKAN: Pra-isi materi apabila memungkinkan.

Labeli input dengan benar dan validasi input secara real-time.

Mendesain formulir efisien

Manfaatkan isiotomatis sehingga pengguna bisa dengan mudah melengkapi formulir dengan data pra-isi. Pra-isi bidang dengan informasi yang sudah Anda tahu. Misalnya, ketika mengambil alamat pengiriman dan penagihan, cobalah untuk menggunakan requestAutocomplete atau memperbolehkan pengguna menyalin alamat pengiriman ke alamat penagihan mereka (atau sebaliknya).

Kegunaan dan faktor form

Indikator berhasil: Menyenangkan pengguna seluler dengan hal-hal kecil yang meningkatkan pengalaman mereka.

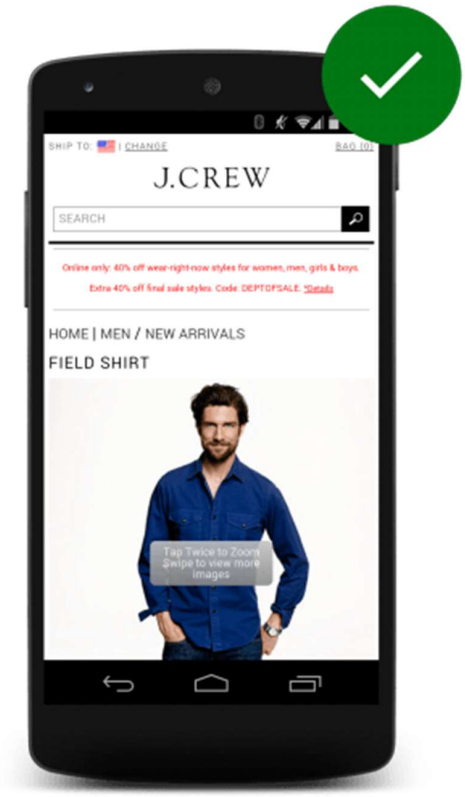
Mengoptimalkan seluruh situs Anda untuk perangkat seluler

Menggunakan layout responsif yang bisa berubah berdasarkan ukuran dan kemampuan perangkat pengguna. Partisipan penelitian menemukan bahwa situs dengan campuran laman yang dioptimalkan untuk seluler dan desktop, lebih sulit digunakan dibandingkan situs khusus untuk desktop.

Jangan buat pengguna melakukan pinch-to-zoom

Pengguna merasa nyaman dengan pengguliran situs secara vertikal, namun tidak secara horizontal. Hindari elemen dengan lebar tetap dan besar. Gunakan kueri media CSS untuk menerapkan penataan gaya yang berbeda untuk layar berbeda. Jangan membuat materi yang hanya terlihat baik pada lebar tampilan yang terlihat tertentu. Situs yang memaksa pengguna untuk gulir secara horizontal tidak akan lolos Uji Ramah-Google Seluler, yang bisa berdampak negatif terhadap peringkat penelusuran.

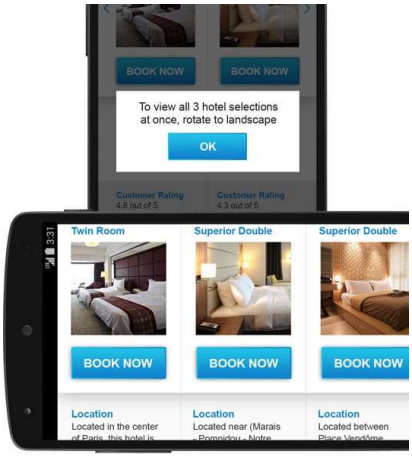
Membuat gambar produk yang bisa diperbesar



LAKUKAN: Membuat gambar produk yang bisa diperbesar sehingga mudah untuk melihat detailnya.

Pelanggan retail berharap situs mengizinkan mereka melihat tampilan dekat resolusi tinggi dari produk. Partisipan penelitian merasa kecewa ketika mereka tidak bisa melihat apa yang mereka beli.

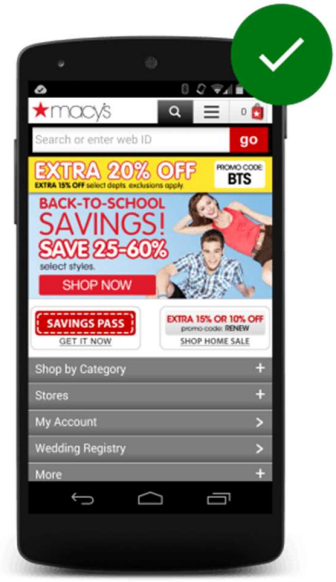
Memberi tahu pengguna orientasi yang terbaik



LAKUKAN: Memberi tahu pengguna orientasi yang terbaik.

Partisipan penelitian biasanya tetap menggunakan orientasi layar yang sama sampai sesuatu mendorong mereka untuk beralih. Desain untuk mode lanskap dan potret, atau dorong pengguna agar beralih ke orientasi optimal. Pastikan bahwa panggil-untuk-aksi yang penting bisa diselesaikan bahkan jika pengguna mengabaikan saran untuk beralih orientasi.

Menjaga pengguna tetap di satu jendela browser



LAKUKAN: Macy mempertahankan pengguna di situs mereka dengan memberikan kupon di situs.

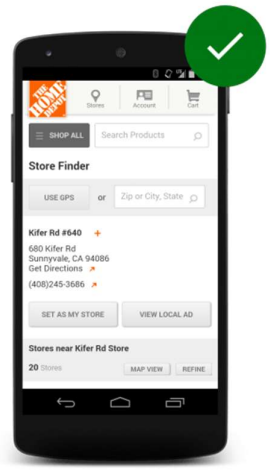
Pengguna mungkin mengalami kesulitan saat beralih antar jendela dan mungkin tidak dapat menemukan jalan kembali ke situs. Hindari panggil-untuk-aksi yang membuka jendela baru. Identifikasi setiap proses yang mungkin menyebabkan pengguna mencari di luar situs dan menyediakan fitur agar mereka tetap berada di situs Anda. Misalnya, jika Anda menerima kupon, langsung tawarkan kepada mereka di situs, bukannya memaksa pengguna mencari penawaran di situs lainnya.

Hindari pelabelan "situs lengkap"

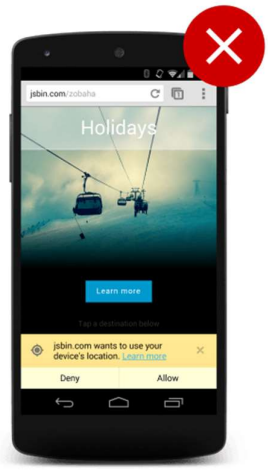
Ketika partisipan penelitian melihat opsi untuk "situs lengkap" (situs desktop) versus "situs seluler", mereka berpikir bahwa situs seluler kekurangan materi dan memilih "lengkap", yang mengarahkan mereka ke situs desktop.

Jelaskan mengapa Anda memerlukan lokasi pengguna

Pengguna harus selalu mengerti alasan Anda meminta lokasi mereka. Partisipan penelitian yang mencoba untuk memesan hotel di kota lain menjadi bingung ketika sebuah situs perjalanan mendeteksi lokasi mereka dan menawarkan hotel di kota mereka yang sekarang ini. Biarkan bidang lokasi kosong secara default, dan izinkan pengguna untuk mengisinya melalui panggil-untuk-aksi yang jelas seperti "Find Near Me".



LAKUKAN: Selalu minta akses ke lokasi berdasarkan isyarat pengguna.



JANGAN: Memintanya secara langsung di beranda saat situs sedang memuat akan menjadikan pengalaman pengguna yang buruk.

BAB IV VARIABEL FONT

Sasaran Pembelajaran

Mahasiswa mampu mengimplementasikan variable font dalam membuat program maupun aplikasi web dan mobile

Kemampuan mahasiswa yang menjadi prasyarat

Mahasiswa sudah menguasai materi Interaksi Manusia Komputer

Keterkaitan bahan pembelajaran dengan pokok bahasan lainnya

Pada materi ini, kita akan melihat seperti apa yang disebut font, bagaimana kita dapat menggunakannya dalam pekerjaan kita, dan potensi yang dimiliki font

Manfaat atau pentingnya bahan pembelajaran ini

Memberi pemahaman dasar tentang font dalam pembuatan program maupun aplikasi web dan mobile

Petunjuk belajar

Untuk memahami penggunaan dan potensi yang dimiliki font, pertama-tama harus mengeksplorasi bagaimana tipografi dan pemuatan font bekerja dalam laman web

Sebagaimana diketahui bahwa font memberikan visualisasi dari huruf sehingga kita bisa mengenali sebuah huruf untuk dirangkai menjadi sebuah kata dan kalimat. Dalam UI UX penggunaan font yang tepat, jelas dan mudah dibaca dapat membuat pengguna merasa nyaman membaca informasi yang disajikan, sehingga dapat mendukung dalam pemberian pengalaman pengguna.

Istilah Font dan jenis huruf digunakan secara bergantian di industri pengembang. Namun ada perbedaan antara jenis huruf dan font itu sendiri: jenis huruf termasuk seluruh keluarga desain, seperti jenis huruf Roboto. Sementara font adalah salah satu file digital dari keluarga itu, seperti "Roboto Bold" atau "Roboto Italic." Dengan kata lain, jenis huruf adalah apa yang Anda lihat, dan font adalah apa yang Anda gunakan.

Sebagai contoh font adalah Roboto, font ini dirancang dan dikembangkan oleh Christian Robertson. Font yang termasuk dalam keluarga Roboto adalah sebagai berikut:

Roboto

SUNGLASSES

Self-driving robot ice cream truck

Fudgesicles only 25¢

ICE CREAM

Marshmallows & almonds

#9876543210

Music around the block

Summer heat rising up from the boardwalk

Thin

Thin Italic

Light

Light Italic

Regular

Regular Italic

Medium

Medium Italic

Bold

Bold Italic

Black

Black Italic

Tantangan untuk Desainer dan Pengembang

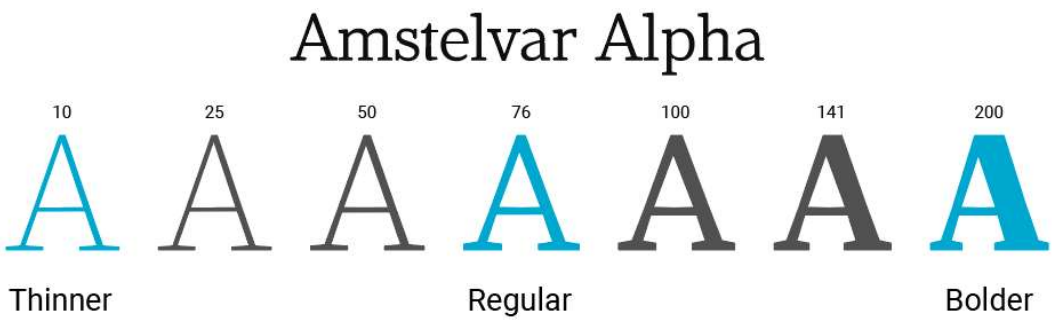
Ketika seorang desainer grafis menyiapkan karya mereka, mereka biasanya mengekspor karya seni terakhir dengan cara yang menanamkan semua font yang digunakan, atau mereka menyediakan arsip ekspor yang berisi semua aset yang digunakan secara terpisah, sedikit seperti situs web dasar. Biaya untuk perancang tergantung pada kesesuaian jenis huruf untuk konteks render (misalnya, dapat dibaca pada ukuran kecil) dan lisensi untuk menggunakan font.

Di web, kami harus mempertimbangkan kedua aspek tersebut, ditambah biaya bandwidth yang terkait. Untuk setiap anggota keluarga jenis huruf yang digunakan dalam desain kami, kami harus meminta file font lain untuk diunduh oleh pengguna kami sebelum mereka dapat melihat teks itu. Hanya termasuk gaya Reguler dan Tebal, ditambah rekan italic-nya, dapat berjumlah hingga 500rb atau lebih data. Ini telah menjadi poin penting bagi para desainer dan pengembang web, karena pengalaman tipografi yang lebih kaya akan dikenakan biaya. Ini bahkan sebelum kita membahas bagaimana font ditampilkan, dan pola mundur atau pemuatan tertunda yang akan kita gunakan (seperti "FOIT" dan "FOUT").)

Anatomi variable font

Font variabel adalah kumpulan gaya master, dengan satu master 'default' sentral (biasanya gaya font Reguler) dan beberapa "sumbu" terdaftar yang mengikat master pusat ke master lainnya. Sebagai contoh, sumbu Bobot mungkin menghubungkan master gaya Ringan ke gaya default dan melalui ke master gaya Bold. Gaya individu yang dapat ditemukan di sepanjang sumbu ini disebut instance.

Sebagai contoh, font variabel Amstelvar memiliki tiga master untuk sumbu Bobotnya: Master reguler berada di tengah, dan dua master, lebih tipis dan lebih tebal, berada di ujung yang berlawanan dari sumbu. Di antara ini ada 200 contoh yang berpotensi yang dapat dipilih oleh perancang atau pengembang:



Typeface Amstelvar, dirancang oleh David Berlow, tipe designer dan typographer di Font Bureau.

Spesifikasi OpenType menentukan sumbu lain, seperti Lebar, Ukuran Optik, Miring dan Miring. Semua sumbu ini berbagi master default yang sama, dan kita dapat menggabungkan mereka untuk menjelajahi sejumlah gaya tipografi eksponensial, seperti kekuatan 2, 3 dan 4 lakukan untuk angka.

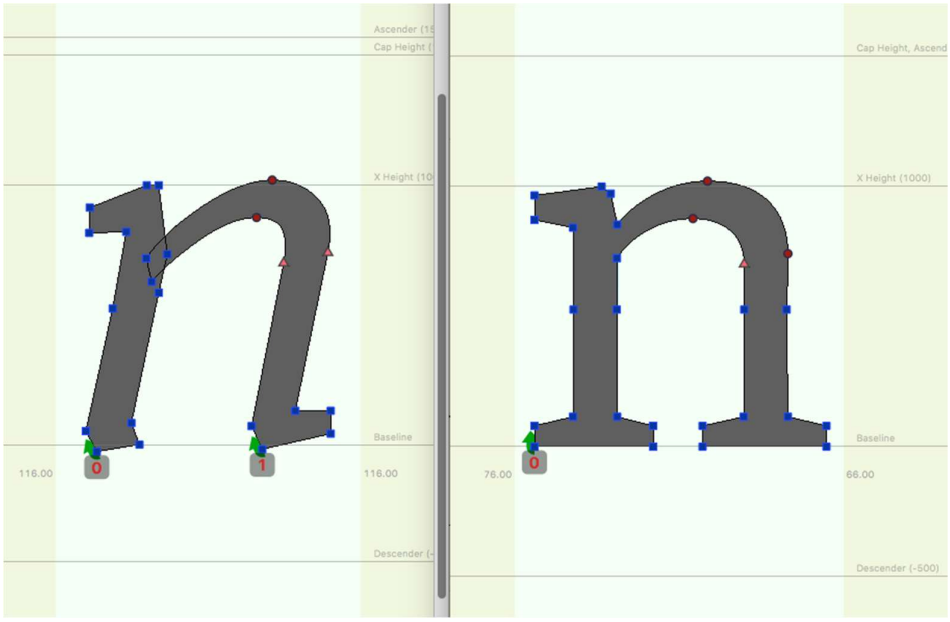
Amstelvar juga memiliki tiga master dalam sumbu Lebar: Sekali lagi Regular berada di tengah sumbu, dan dua master, sempit dan lebih lebar, berada di ujung yang berlawanan dari sumbu ini. Ini tidak hanya menyediakan semua lebar gaya Reguler, tetapi juga semua lebar dan bobot digabungkan.

Di antara kapak terdaftar Amstelvar (lebar, berat, dan ukuran optik) ada ribuan gaya. Ini mungkin tampak seperti pembunuhan besar-besaran, tetapi pertimbangkan bahwa Amstelvar hanya mendukung sistem penulisan Latin. Mempertimbangkan kebutuhan semua skrip dunia, dan banyak aplikasi tipografi saat ini, kualitas pengalaman membaca dapat sangat ditingkatkan oleh keragaman jenis gaya dalam font. Dan, jika tanpa penalti performa, pengguna dapat menggunakan beberapa atau sebanyak gaya yang mereka inginkan - terserah desain mereka.

Italics sedikit berbeda

Cara orang Italia menangani dalam font variabel menarik, karena ada dua pendekatan perbedaan. Jenis-jenis seperti Helvetica atau Roboto memiliki kontur yang kompatibel dengan interpolasi, sehingga gaya Romawi dan Miringnya dapat diinterpolasi antara dan sumbu Miring dapat digunakan untuk beralih dari Romawi ke Miring.

Tipografi lainnya (seperti Garamond, Baskerville, atau Bodoni) memiliki kontur mesin terbang Romawi dan Italia yang tidak kompatibel dengan interpolasi. Misalnya, kontur yang biasanya mendefinisikan huruf kecil Romawi "n" tidak cocok dengan kontur yang digunakan untuk mendefinisikan huruf kecil Italia "n". Alih-alih menginterpolasi satu kontur ke kontur lainnya, sumbu Italic beralih dari kontur Romawi ke kontur Italic.



Kontur “n” Amstelvar dalam bahasa Italia (12 poin, berat reguler, lebar normal), dan dalam bahasa Romawi. Gambar dipasok oleh David Berlow, tipe designer dan typographer di Font Bureau.

Setelah beralih ke Italic, sumbu yang tersedia untuk pengguna harus sama dengan yang untuk Romawi, sama seperti karakter yang ditetapkan harus sama.

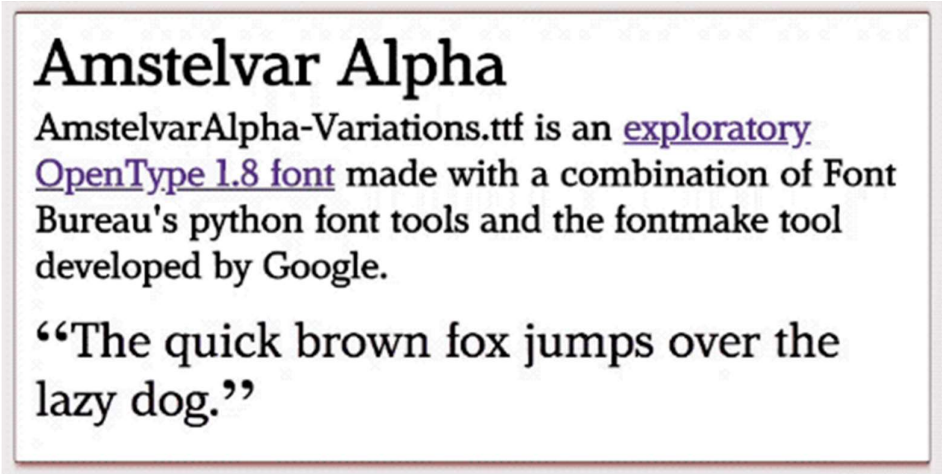
Kemampuan substitusi mesin terbang juga dapat dilihat untuk mesin terbang individu, dan digunakan di mana saja di ruang desain font variabel. Misalnya, desain tanda dolar dengan dua bilah vertikal berfungsi paling baik pada ukuran titik yang lebih besar, tetapi pada ukuran titik yang lebih kecil, desain dengan hanya satu bilah lebih baik. Ketika kami memiliki piksel lebih sedikit untuk rendering mesin terbang, desain dua batang dapat menjadi tidak terbaca. Untuk mengatasi hal ini, seperti halnya sumbu Italic, substitusi mesin terbang dari satu mesin terbang ke mesin terbang lain dapat terjadi di sepanjang sumbu Ukuran Optik pada titik yang ditentukan oleh perancang tipe.

Singkatnya, di mana kontur memungkinkan untuk itu, desainer tipe dapat membuat font yang interpolasi antara berbagai master dalam ruang desain multi-dimensi. Ini memberi Anda kontrol granular atas tipografi Anda, dan banyak kekuatan.

Definisi axis

Karena pengembang font menentukan sumbu mana yang tersedia dalam font mereka, penting untuk memeriksa dokumentasi font untuk mengetahui apa yang tersedia. Misalnya, dalam font variabel Gingham yang dirancang oleh Christoph Koeberlin, ada dua sumbu yang tersedia, Lebar dan Berat. Font variabel Amstelvar tidak mengandung sumbu miring, tetapi memang memiliki sumbu yang disebut Grade, ditambah banyak sumbu lainnya.

Sumbu Grade menarik karena mengubah bobot font tanpa mengubah lebar, sehingga jeda baris tidak berubah. Dengan bermain dengan sumbu Grade, Anda dapat menghindari dipaksa untuk bermain-main dengan perubahan pada sumbu Bobot yang mempengaruhi lebar keseluruhan, dan kemudian berubah ke poros Lebar yang mempengaruhi bobot keseluruhan. Ini dimungkinkan karena gaya default Amstelvar telah didekonstruksi dalam 4 aspek dasar bentuk: bentuk hitam atau positif, bentuk putih atau negatif, dan dimensi x dan y. Keempat aspek ini dapat dicampur untuk membentuk gaya lain, seperti Lebar dan Berat, dengan cara warna primer dapat dicampur untuk membuat warna lain.



Anda dapat melihat contoh kerja dan kode sumber untuk contoh di atas di sini. Lima axis terdaftar plus Kelas memiliki tag 4 karakter yang digunakan untuk menetapkan nilainya dalam CSS:

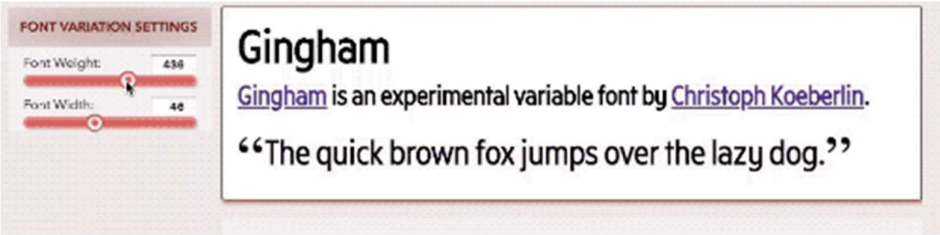
Nama-nama Axis dan CSS values	
Weight	wght
Width	wdth
Slant	slnt
Optical Size	opsz
Italics	ital
Grade	GRAD

Untuk menambahkan font variabel terlebih dahulu, kita harus menautkannya, seperti font kustom apa pun:

```
@font-face {
  font-family: 'AmstelvarAlpha';
  src: url('../fonts/AmstelvarAlpha-VF.ttf');
}
```

Cara kita mendefinisikan nilai sumbu adalah dengan menggunakan variasi font properti CSS yang memiliki serangkaian nilai yang memasangkan tag sumbu dengan lokasi contoh:

```
#font-amstelvar {
  font-family: 'AmstelvarAlpha';
  font-variation-settings: 'wdth' 400, 'wght' 98;
}
```

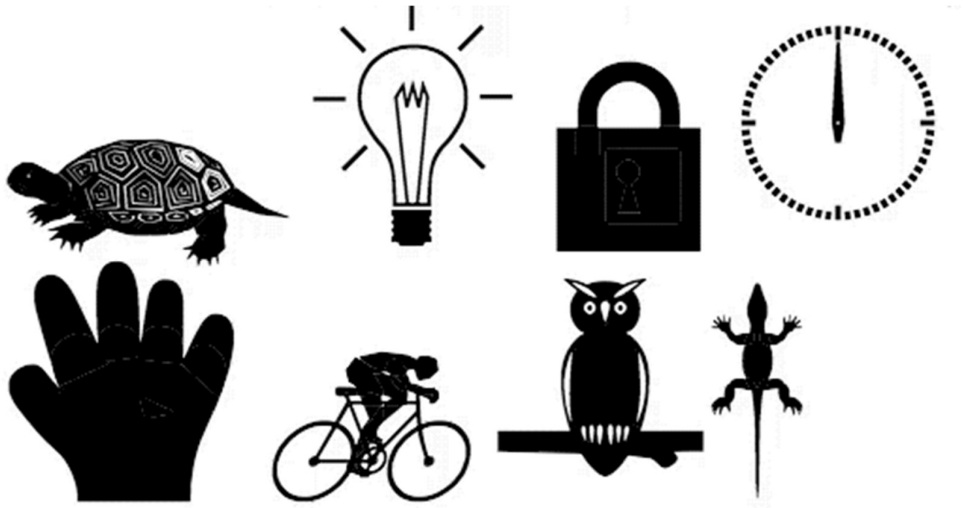


Dalam contoh ini Anda dapat melihat sumbu Bobot dan Lebar diubah dengan cepat.Anda dapat meliha t contoh kerja dan kode sumber untuk contoh di atas di sini.

Tanggung jawab penata letak

Menetapkan nilai sumbu diturunkan ke selera pribadi dan menerapkan praktik tipografi terbaik. Bahaya dengan teknologi baru apa pun adalah kemungkinan penyalahgunaan, dan pengaturan yang terlalu artistik atau eksploratif juga dapat mengurangi keterbacaan teks yang sebenarnya. Untuk judul, menjelajahi sumbu yang berbeda untuk membuat desain artistik yang hebat memang mengasyikkan, tetapi untuk tubuh menyalin risiko ini membuat teks menjadi tidak terbaca.

Salah satu contoh yang bagus dari ekspresi artistik ditunjukkan di atas, sebuah eksplorasi dari jenis huruf Decovar oleh Mandy Michael. Anda dapat melihat contoh kerja dan kode sumber untuk contoh di atas di sini.



Ada juga kemungkinan untuk mengeksplorasi karakter animasi dengan font variabel. Di atas adalah contoh dari berbagai sumbu yang digunakan dengan jenis huruf Zycon. Lihat contoh animasi langsung di Axis Praxis.

Variable fonts performance gains

Font variabel OpenType memungkinkan kita untuk menyimpan beberapa variasi tipe keluarga ke dalam file font tunggal. Monotipe menjalankan eksperimen dengan menggabungkan 12 font input untuk menghasilkan delapan bobot, melintasi tiga lebar, melintasi gaya Italia dan

Romawi. Menyimpan 48 font individual dalam file font variabel tunggal berarti pengurangan 88% dalam ukuran file.

Di sisi lain, jika Anda menghidupkan font di antara pengaturan, ini dapat menyebabkan masalah kinerja browser. Pelajari lebih lanjut tentang ini di Surma's Supercharged.

Dengan fonta variabel, pembuat aplikasi dan situs web dapat menawarkan pengalaman tipografi yang sangat kaya yang mengekspresikan setiap merek, tanpa bandwidth dan biaya latensi sebelumnya. Namun, jika Anda menggunakan font tunggal seperti Roboto Regular dan tidak ada yang lain, Anda mungkin melihat keuntungan bersih dalam ukuran font jika Anda beralih ke font variabel dengan banyak sumbu. Seperti biasa, itu tergantung pada kasus penggunaan Anda.

Fallbacks dan dukunan browser

Dukungan saat ini terbatas, tetapi font variabel akan berfungsi hari ini di luar kotak di Chrome dan Safari, dengan dukungan segera hadir untuk Edge 17 dan Firefox. Lihat caniuse.com untuk lebih jelasnya. Dimungkinkan untuk menggunakan @support di CSS Anda untuk membuat fallback yang layak:

```
@supports (font-variation-settings: 'wdth' 200) {
  @font-face {
    /* https://github.com/TypeNetwork/Amstelvar */
    font-family: AmstelvarAlpha;
    src: url('../fonts/AmstelvarAlpha-VF.ttf');
    font-weight: normal;
    font-style: normal;
  }

  #font-amstelvar {
    font-family: AmstelvarAlpha;
    font-variation-settings: 'wdth' 400, 'wght' 98;
  }
}
```

BAB V AKSESIBILITAS

Sasaran Pembelajaran

Mahasiswa mengetahui apa yang dimaksud aksesibilitas dan cara menerapkannya pada web-development, mengetahui cara membuat situs web yang bisa diakses dan bisa digunakan oleh setiap orang, mengetahui cara menyertakan aksesibilitas dasar dengan dampak minimal pada development, mengetahui apa saja fitur HTML yang tersedia dan cara menggunakannya untuk meningkatkan aksesibilitas, mengetahui tentang berbagai teknik aksesibilitas modern untuk membuat pengalaman aksesibilitas yang brilian

Kemampuan mahasiswa yang menjadi prasyarat

Mahasiswa sudah menguasai materi Interaksi Manusia Komputer

Manfaat atau pentingnya bahan pembelajaran ini

Memberi pemahaman mengenai aksesibilitas

Dengan memahami aksesibilitas, cakupannya, dan dampaknya bisa membuat Anda menjadi web developer yang lebih baik. Panduan ini dimaksudkan untuk membantu Anda memahami cara membuat situs web bisa diakses dan bisa digunakan oleh setiap orang.

"Aksesibilitas" bisa jadi sulit dieja, namun tidak harus sulit dicapai. Dalam panduan ini, Anda akan melihat beberapa cara mudah untuk membantu meningkatkan aksesibilitas dengan upaya minimal, cara menggunakan apa yang ditanamkan ke HTML untuk membuat antarmuka yang lebih bisa diakses dan sempurna, dan cara memanfaatkan sebagian teknik modern untuk memoles pengalaman yang bisa diakses.

Anda juga akan menemukan banyak dari teknik ini yang akan membantu membuat antarmuka yang lebih menyenangkan dan mudah digunakan oleh *semua* pengguna, tidak cuma bagi penyandang cacat.

Tentu saja, banyak developer yang hanya memiliki pemahaman kabur mengenai apa yang dimaksud dengan aksesibilitas — sesuatu yang harus dilakukan pada kontrak pemerintah, daftar periksa, dan pembaca layar, ya? — dan banyak juga yang gagal paham. Misalnya, banyak developer merasa bahwa mengurus aksesibilitas akan memaksa mereka harus memilih antara membuat pengalaman yang menyenangkan serta menarik, dan pengalaman bisa diakses yang ganjil dan jelek.

Tentu saja, sama sekali bukan itu masalahnya, jadi mari kita jelaskan sebelum melangkah lebih jauh. Apa yang dimaksud aksesibilitas, dan apa yang akan kita pelajari di sini?

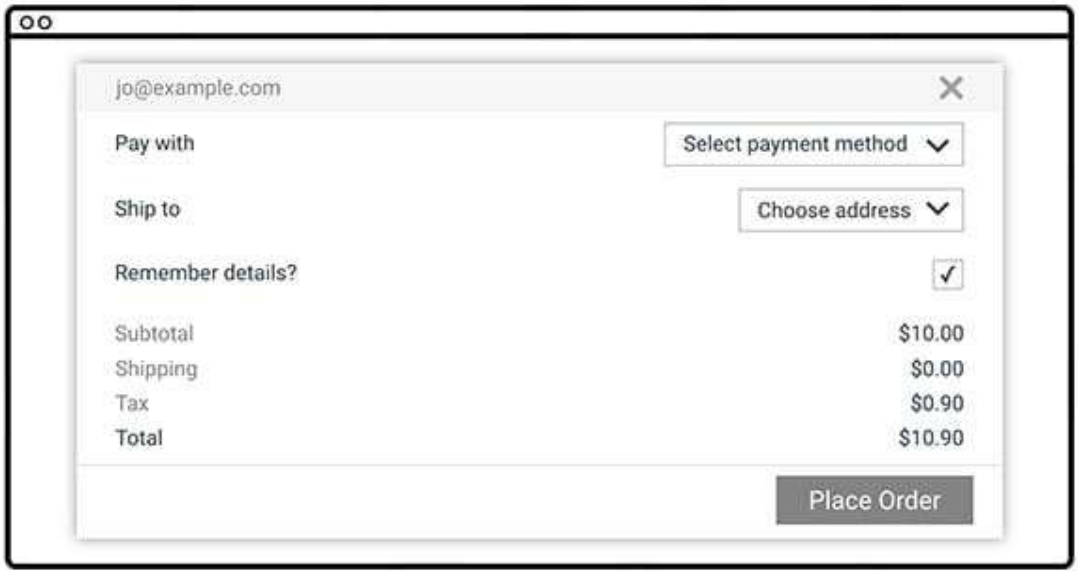
Apa yang dimaksud dengan aksesibilitas?

Umumnya, bila kita menyebut sebuah situs bisa diakses, maksud kami adalah materi situs yang tersedia, dan fungsionalitasnya bisa dioperasikan, secara harfiah, oleh *siapa saja*. Sebagai developer, mudah menganggap bahwa semua pengguna bisa melihat dan menggunakan keyboard, mouse, atau layar sentuh, dan bisa berinteraksi dengan materi laman Anda sama seperti yang Anda lakukan. Hal ini bisa menghasilkan pengalaman yang berfungsi dengan baik untuk sebagian orang, namun menimbulkan beragam masalah, dari gangguan biasa hingga penghenti-tampilan pada pengguna lainnya.

Aksesibilitas merujuk pada pengalaman pengguna yang mungkin berada di luar lingkup sempit pengguna "pada umumnya", yang mungkin mengakses atau berinteraksi dengan berbagai hal secara berbeda dari yang Anda perkirakan. Khususnya, ini menyangkut pengguna yang mengalami semacam kelemahan atau cacat — dan ingatlah bahwa pengalaman bisa bersifat non-fisik atau sementara.

Misalnya, walaupun kita cenderung memusatkan diskusi aksesibilitas pada pengguna yang memiliki kelemahan, kita bisa mengaitkan dengan pengalaman menggunakan antarmuka yang tidak bisa diakses oleh kita karena alasan lain. Pernahkah Anda mengalami masalah menggunakan situs versi desktop di ponsel, atau melihat pesan "This content is not available in your area", atau tidak bisa menemukan menu yang familier di tablet? Semua itu adalah masalah aksesibilitas.

Setelah mengetahui selengkapnya, Anda akan tahu bahwa menangani masalah aksesibilitas dalam pengertian yang lebih luas dan lebih umum ini biasanya akan selalu memperbaiki pengalaman pengguna bagi siapa saja. Mari kita amati sebuah contoh:

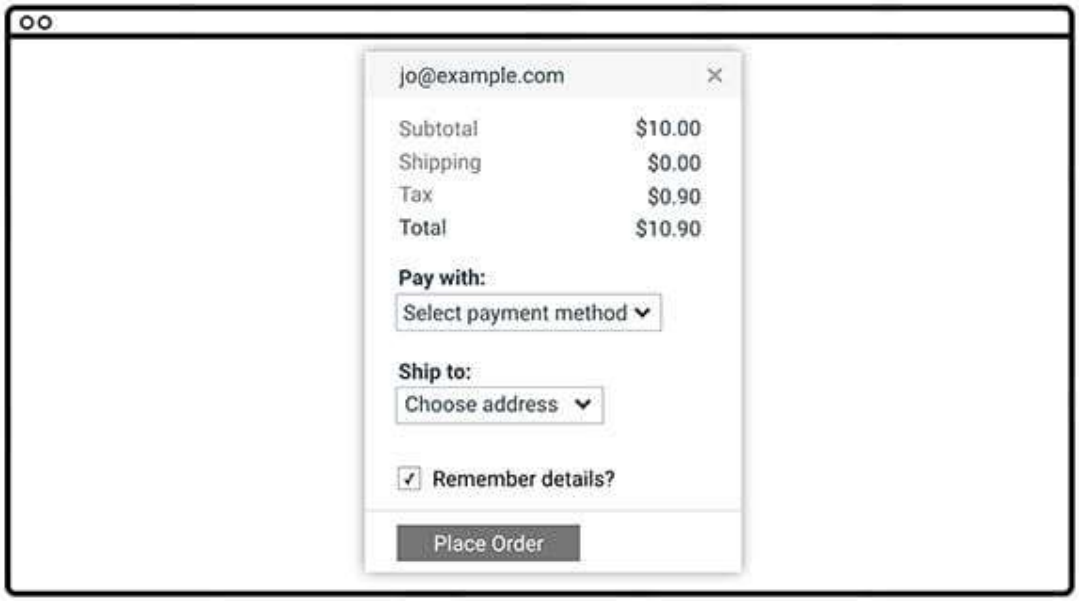


Formulir ini memiliki sejumlah masalah aksesibilitas sebagai berikut:

- Kontras teks rendah, sehingga sulit dibaca oleh pengguna yang memiliki penglihatan lemah.

- Adanya label di sebelah kiri dan bidang-bidang di sebelah kanan menyulitkan banyak orang untuk mengaitkannya, dan hampir mustahil bagi orang yang perlu memperbesar tampilan untuk menggunakan laman tersebut; bayangkan melihatnya di ponsel dan harus menggeser untuk mengetahui apa yang terjadi.
- Label "Remember details?" tidak dikaitkan dengan kotak centang, jadi Anda harus mengetuk atau mengeklik hanya pada segi empat kecil, bukan cuma mengeklik label; selain itu, orang yang menggunakan pembaca layar akan kesulitan mengetahui asosiasinya.

Sekarang, mari kita goyangkan tongkat sihir aksesibilitas dan lihat formulir yang telah diperbaiki masalahnya. Kita akan membuat teksnya lebih gelap, memodifikasi desainnya agar label dekat dengan apa yang dilabeli, dan memperbaiki label untuk dikaitkan dengan kotak centang sehingga Anda juga bisa beralih dengan mengeklik labelnya.



Mana yang lebih suka Anda gunakan? Jika Anda bilang "versi yang bisa diakses", berarti Anda sudah memahami premis utama panduan ini. Sering kali, sesuatu yang menjadi pemblokir penuh bagi segelintir pengguna juga menjadi hal yang menyakitkan bagi banyak pengguna lainnya, jadi dengan memperbaiki masalah aksesibilitas berarti Anda memperbaiki pengalaman bagi siapa saja.

Pedoman Aksesibilitas Konten Web

Sepanjang panduan ini kita akan merujuk Web Content Accessibility Guidelines (WCAG) 2.0, yaitu serangkaian panduan dan praktik terbaik yang dikumpulkan oleh pakar aksesibilitas untuk menjawab apa yang dimaksud dengan "aksesibilitas" secara metodis. Sejumlah negara sebenarnya mewajibkan penggunaan panduan ini dalam persyaratan legal aksesibilitas web mereka.

WCAG disusun oleh empat prinsip yang sering kali disebut dengan singkatan *POUR*:

- **Perceivable**: Bisakah pengguna mempersepsi materi? Ini membantu mengingatkan kita bahwa hanya karena sesuatu bisa dipersepsi orang dengan satu indera, misalnya penglihatan, tidak berarti semua pengguna bisa mempersepsinya.
- **Operable**: Bisakah pengguna menggunakan komponen UI dan menyusuri materi? Misalnya, sesuatu yang mengharuskan interaksi mengarahkan ke atas tidak bisa dioperasikan oleh orang yang tidak bisa menggunakan mouse atau layar sentuh.
- **Understandable**: Bisakah pengguna memahami materi? Bisakah pengguna memahami antarmuka dan apakah cukup konsisten untuk menghindari kebingungan?
- **Robust**: Bisakah materi dimanfaatkan oleh beragam agen-pengguna (browser)? Bisakah digunakan bersama teknologi pendukung?

Walaupun WCAG menyediakan ringkasan yang komprehensif mengenai apa yang dimaksudnya dengan materi yang bisa diakses, ini juga bisa sedikit membebani. Untuk membantu meminimkannya, grup WebAIM (Web Accessibility in Mind) telah meringkas panduan WCAG ke dalam sebuah daftar periksa yang mudah diikuti, yang ditargetkan secara khusus untuk materi web.

Daftar periksa WebAIM bisa memberi Anda rangkuman tingkat tinggi mengenai hal-hal yang perlu diimplementasikan, sekaligus menautkan ke spesifikasi WCAG yang mendasarinya jika Anda membutuhkan definisi yang diluaskan.

Dengan alat (bantu) ini, Anda bisa memetakan arah pekerjaan aksesibilitas dan menjadi percaya diri karena, asalkan proyek memenuhi kriteria yang dijelaskan, pengguna akan memiliki pengalaman positif saat mengakses materi Anda.

Memahami keberagaman pengguna

Saat mempelajari tentang aksesibilitas, akan membantu bila memiliki pemahaman mengenai keberagaman di antara para pengguna di seluruh dunia dan aneka topik aksesibilitas yang memengaruhi mereka. Untuk menjelaskan lebih jauh, inilah sesi tanya-jawab informatif dengan Victor Tsaran, Technical Program Manager di Google, yang buta total.

Apa yang Anda kerjakan di Google?

Di Google, pekerjaan saya adalah membantu memastikan produk kami berfungsi untuk semua macam pengguna, tanpa memandang kemampuan atau pun kelemahan.

Kelemahan macam apa yang dimiliki pengguna?

Bila kita memikirkan tentang kelemahan, yang akan menyulitkan seseorang untuk mengakses materi kita, banyak orang akan langsung membayangkan pengguna tuna netra seperti saya. Memang ada benarnya, kelemahan ini bisa sangat membuat frustrasi atau bahkan mustahil untuk menggunakan banyak situs web.

Banyak teknik web modern memiliki efek samping tidak menguntungkan pada pembuatan situs yang tidak berfungsi dengan baik pada alat yang digunakan oleh pengguna tuna netra untuk

mengakses web. Akan tetapi, sebenarnya aksesibilitas lebih dari itu. Kami rasa ada gunanya membayangkan kelemahan ke dalam empat kelompok besar: visual, motor, pendengaran, dan kognitif.

Mari kita bahas satu per satu. Bisakah Anda berikan beberapa contoh kelemahan visual?

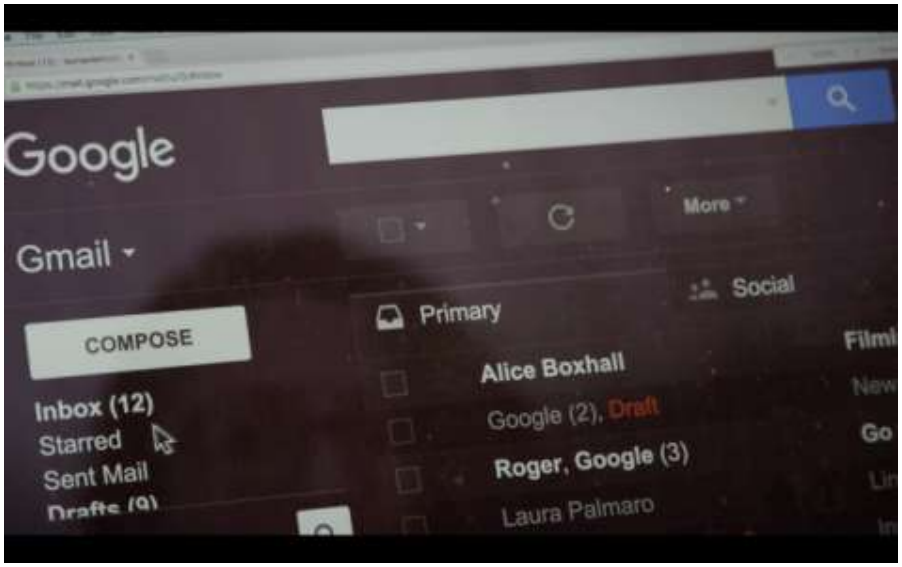
Kelemahan visual bisa dibagi ke dalam beberapa kategori: Pengguna yang tidak memiliki penglihatan, seperti saya, dapat menggunakan pembaca layar, braille, atau kombinasi keduanya.

Pembaca braille

Sekarang, sebenarnya jarang ada orang yang tidak bisa melihat sama sekali, namun tetap ada, mungkin saja Anda mengenal atau berjumpa dengan setidaknya satu orang yang tidak bisa melihat sama sekali. Akan tetapi juga jauh lebih banyak orang yang kita sebut pengguna yang berpenglihatan lemah.

Inilah kelompok terbesar, mulai dari orang-orang yang seperti istri saya, yang tidak memiliki kornea — jadi walaupun pada dasarnya ia bisa melihat sesuatu, ia kesulitan membaca barang cetak dan secara legal dianggap buta — hingga orang yang memiliki penglihatan buruk dan perlu mengenakan kaca mata resep yang sangat tinggi.

Ada banyak sekali macamnya, dan dengan sendirinya juga ada banyak macam akomodasi yang digunakan oleh orang-orang yang ada dalam kategori ini: ada yang menggunakan pembaca layar atau tampilan braille (saya bahkan pernah mendengar ada seorang wanita yang membaca braille yang ditampilkan di layar karena lebih mudah dilihat daripada teks tercetak), atau mereka mungkin menggunakan teknologi teks-ke-ucapan tanpa fungsionalitas lengkap pembaca layar, atau mereka dapat menggunakan pembesar tampilan layar yang memperbesar bagian layar, atau mereka mungkin cuma menggunakan zoom di browser untuk membuat semua font tampak lebih besar. Mereka mungkin juga menggunakan opsi kontras tingi seperti mode kontras tinggi di sistem operasi, ekstensi browser kontras tinggi atau tema kontras tinggi untuk situs web.



Mode kontras tinggi

Banyak pengguna bahkan menggunakan kombinasi dari semua ini, seperti teman saya Laura yang menggunakan kombinasi mode kontras tinggi, zoom di browser, teks-ke-ucapan.

Penglihatan rendah adalah sesuatu yang bisa dikaitkan dengan banyak orang. Sebagai awal, kita semua mengalami penurunan penglihatan seiring usia, jadi sekalipun Anda belum mengalaminya mungkin Anda pernah mendengar orang tua Anda mengeluhkan hal ini. Namun banyak orang mengalami frustrasi saat membawa laptopnya keluar ruangan saat matahari cerah dan tiba-tiba tidak bisa membaca apa-apa! Atau orang yang telah menjalani bedah laser atau mungkin cuma harus membaca sesuatu dari seluruh ruangan mungkin telah menggunakan salah satu akomodasi yang tadi saya sebutkan. Jadi menurut saya agak mudah bagi developer untuk memiliki empati bagi pengguna yang berpenglihatan rendah.

Oh, jangan lupa dengan orang-orang yang memiliki penglihatan warna yang buruk — sekitar 9% pria memiliki semacam defisiensi penglihatan warna! Plus sekitar 1% wanita. Mereka mungkin kesulitan membedakan merah dan hijau, atau kuning dan biru. Pertimbangkan tentang hal itu bila Anda nanti mendesain validasi formulir.

Bagaimana dengan kelemahan motorik?

Ya, kelemahan motorik, atau kelemahan ketangkasan fisik. Kelompok ini meliputi orang-orang yang lebih suka tidak menggunakan mouse, mungkin karena mengalami semacam RSI atau sesuatu dan merasa nyeri bila menggunakannya, hingga orang yang mungkin lumpuh fisik dan memiliki jangkauan gerak terbatas untuk anggota tubuh tertentu.

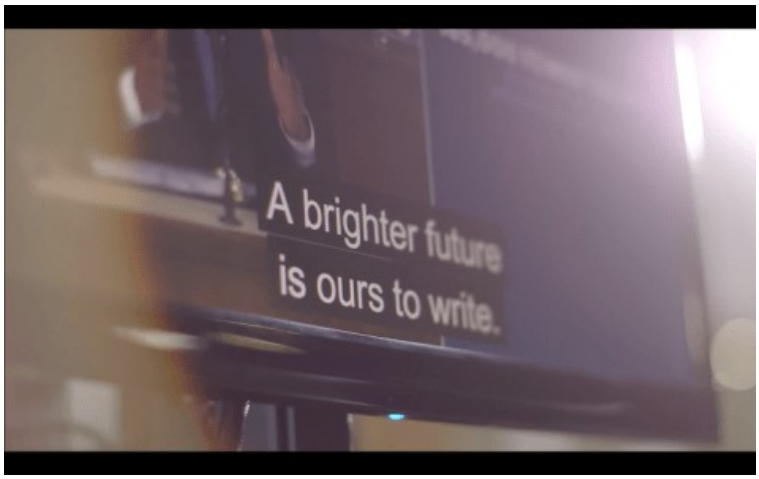
Pengguna yang mengalami kelemahan motorik mungkin menggunakan keyboard, perangkat switch, kontrol suara, atau bahkan perangkat pelacak mata untuk berinteraksi dengan komputer mereka.

Perangkat pelacak mata

Mirip dengan kelemahan penglihatan, mobilitas juga bisa menjadi masalah sementara atau situasional: Mungkin Anda mengalami patah pada tangan yang biasa memegang mouse. Mungkin trackpad rusak di laptop, atau Anda sedang naik kereta yang bergoncang-goncang. Bisa jadi ada banyak situasi yang menghalangi mobilitas pengguna, dan dengan memastikan kita mempedulikan mereka berarti kita memperbaiki pengalaman keseluruhan, untuk siapa saja yang memiliki kelemahan permanen maupun mereka yang untuk sementara tidak bisa menggunakan UI berbasis pointer.

Bagus, mari kita bicara tentang kelemahan pendengaran.

Kelompok ini bisa meliputi orang yang sepenuhnya tuli hingga orang yang sulit-mendengar. Dan mirip sekali dengan penglihatan, pendengaran kita cenderung menurun bersama usia. Banyak dari kita menggunakan kemampuan umum seperti alat bantu dengar.



Teks layar

Bagi pengguna yang lemah pendengaran, perlu dipastikan bahwa kita tidak mengandalkan suara, jadi pastikan menggunakan sesuatu seperti teks video dan transkrip, serta menyediakan semacam alternatif, jika suara merupakan bagian dari antarmuka.

Dan seperti yang kita ketahui pada kelemahan motorik dan penglihatan, mudah sekali kita bayangkan bahwa orang yang pendengarannya bekerja dengan baik akan turut merasakan manfaat akomodasi ini. Banyak teman saya bilang mereka suka bila video dilengkapi teks dan transkrip karena dengan begitu jika mereka berada di kantor yang tidak bersekat dan mereka tidak membawa headphone, mereka tetap bisa menonton video!

Baiklah, bisa jelaskan sedikit tentang kelemahan kognitif?

Ada banyak macam kondisi kognitif seperti ADD, Disleksia, dan Autis, yang berarti orang-orang demikian ingin atau perlu mengakses sesuatu dengan cara berbeda. Akomodasi untuk kelompok ini dengan sendirinya sangat beragam sekali, namun pasti menemukan semacam tumpang tindih dengan area lain, seperti penggunaan fungsionalitas zoom untuk mempermudah membaca atau berkonsentrasi. Selain itu, para pengguna ini mungkin merasa bahwa desain yang benar-benar minimal adalah yang paling bagus karena meminimalkan distraksi dan beban kognitif.

Saya kira siapa saja bisa mengaitkan dengan stres kelebihan beban kognitif, jadi jelas bahwa jika kita membuat sesuatu yang berfungsi dengan baik untuk orang yang memiliki kelemahan kognitif, maka kita akan membuat sesuatu yang akan menjadi pengalaman menyenangkan bagi siapa saja.

Jadi, bagaimana kesimpulan pendapat Anda tentang aksesibilitas?

Bila Anda mengamati berbagai macam kemampuan dan ketidakmampuan yang dimiliki orang, Anda bisa melihat bahwa mendesain dan membangun produk hanya bagi orang yang memiliki penglihatan, pendengaran, ketangkasan, dan kognisi sempurna, nampaknya itu benar-benar sempit. Ini berlawanan dengan tujuan semula, karena kita akan membuat pengalaman yang lebih menimbulkan stres dan kurang berguna bagi siapa saja, dan bagi sebagian pengguna akan menghasilkan pengalaman yang sebenarnya akan mengecualikan mereka sama sekali.

Dalam wawancara ini, Victor mengidentifikasi serangkaian kelemahan, dan memasukkannya ke dalam empat kategori besar: *visual*, *motorik*, *pendengaran*, dan *kognitif*. Ia juga menunjukkan bahwa setiap kelemahan bisa bersifat *situasional*, *sementara*, atau *permanen*.

Mari kita amati beberapa contoh sungguhan dari kelemahan akses dan melihat ke dalam kategori serta tipe apa memasukkannya. Perhatikan, beberapa kelemahan mungkin dimasukkan dalam lebih dari satu kategori atau tipe.

	Situasional	Sementara	Permanen
Visual		goncangan	kebutaan
Motor	memegang bayi	lengan patah, RSI*	RSI*
Pendengaran	kantor yang berisik		
Kognitif		goncangan	

*Repetitive Strain Injury - Cedera Akibat Tegang Berulang: mis., sindrom saluran karpal, radang siku, jari pemicu

Langkah berikutnya

Kita sudah banyak membahas hal-hal pokok! Anda telah membaca tentang

- apa yang dimaksud aksesibilitas dan mengapa hal ini penting bagi siapa saja
- daftar periksa aksesibilitas WCAG dan WebAIM
- berbagai tipe kelemahan yang harus Anda pertimbangkan

Untuk panduan selebihnya, kita akan mendalami beberapa aspek praktis dalam pembuatan situs web yang bisa diakses. Kita akan menyusun upaya ini dalam tiga bidang bahasan utama:

- **Fokus**: Kita akan mengamati cara membangu sesuatu yang bisa dioperasikan dengan keyboard sebagai ganti mouse. Tentunya hal ini penting bagi pengguna yang memiliki kelemahan motorik, namun juga memastikan bahwa UI Anda cocok untuk semua pengguna.
- **Semantik**: Kita akan memastikan bahwa kita mengekspresikan antarmuka pengguna dengan cara sempurna yang bisa digunakan pada berbagai macam teknologi pendukung.
- **Penataan gaya**: Kita akan mempertimbangkan desain visual dan mengamati beberapa teknik untuk membuat elemen visual antarmuka seluwes dan seberguna mungkin.

Masing-masing pokok bahasan bisa mengisi keseluruhan kursus, jadi kita tidak akan membahas setiap aspek pembuatan situs web yang bisa diakses. Akan tetapi, kita akan memberi Anda informasi yang cukup untuk memulai, dan menunjukkan beberapa tempat yang bagus untuk mempelajari tentang setiap topik tersebut.

5.1 Pengantar Fokus

Dalam pelajaran ini, kita akan membicarakan tentang *fokus* dan cara mengelolanya dalam aplikasi Anda. Fokus merujuk pada kontrol mana pada layar (item masukan seperti bidang, kotak centang, tombol, atau tautan) yang saat ini menerima masukan dari keyboard, dan dari clipboard bila Anda menempelkan materi.

Inilah tempat yang bagus untuk mulai mempelajari tentang aksesibilitas karena kita semua tahu cara menggunakan keyboard, mudah untuk menghubungkan dan menguji, serta menguntungkan hampir semua pengguna.

Pengguna yang memiliki gangguan motorik, yang boleh jadi berupa paralisis permanen hingga pergelangan yang keseleo, mungkin mengandalkan keyboard atau perangkat switch untuk menyusuri laman Anda, jadi strategi fokus yang baik adalah penting untuk menyediakan pengalaman yang bagus bagi mereka.

Dan bagi pengguna berpengalaman yang mengetahui setiap pintasan keyboard di mesin mereka, karena bisa dengan cepat menyusuri situs Anda menggunakan keyboard saja, tentu akan membuat mereka menjadi lebih produktif.

Karena itu, strategi fokus yang diimplementasikan dengan baik akan memastikan setiap orang yang menggunakan aplikasi Anda akan memiliki pengalaman yang lebih baik. Nanti akan kita lihat dalam pelajaran berikutnya bahwa upaya yang Anda berikan pada fokus adalah basis penting untuk mendukung pengguna teknologi pendukung, juga tentunya, semua pengguna.

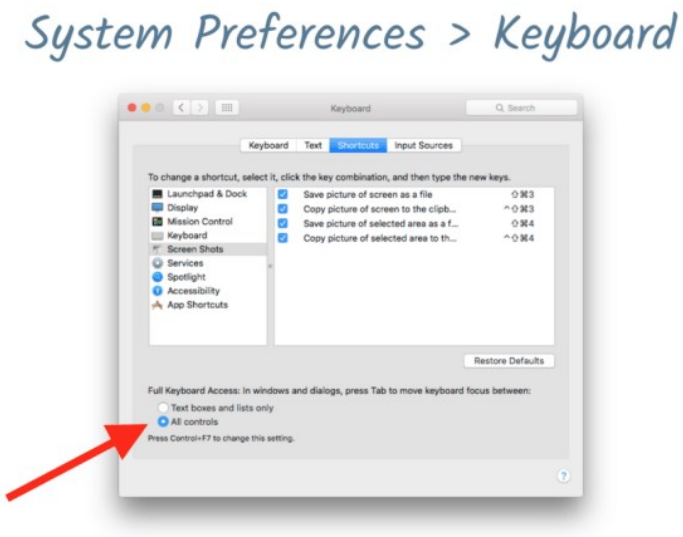
Apa yang dimaksud dengan fokus?

Fokus menentukan tempat kejadian keyboard di laman pada saat tertentu. Misalnya, jika Anda memfokus bidang masukan teks dan mulai mengetik, bidang masukan akan menerima kejadian keyboard dan menampilkan karakter yang Anda ketikkan. Walaupun memiliki fokus, bidang masukan juga akan menerima masukan yang ditempelkan dari clipboard.

Item yang saat ini difokus sering kali ditunjukkan oleh *lingkaran fokus*, gaya yang bergantung pada browser maupun segala penataan gaya yang diterapkan oleh penulis laman tersebut. Chrome, misalnya, biasanya menyorot elemen yang difokus dengan border biru, sedangkan Firefox menggunakan border putus-putus.

Sebagian pengguna mengoperasikan komputer mereka hampirnya sepenuhnya dengan keyboard atau perangkat masukan lainnya. Bagi para pengguna tersebut, fokus sangatlah penting; inilah sarana utama mereka untuk melakukan apa saja di layar. Karena alasan itu, daftar periksa Web AIM menyatakan di bagian 2.1.1 bahwa semua fungsionalitas laman harus tersedia menggunakan keyboard, kecuali jika ada yang tidak bisa Anda lakukan dengan keyboard, misalnya menggambar bebas.

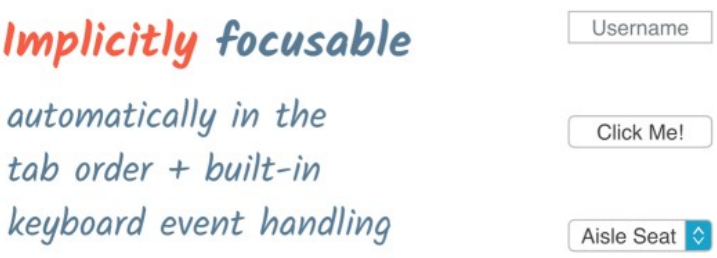
Sebagai pengguna, Anda bisa mengontrol elemen mana yang saat ini difokus dengan menggunakan Tab, Shift+Tab, atau tombol panah. Di Mac OSX, cara kerjanya sedikit berbeda: walaupun Chrome selalu memungkinkan Anda melakukan navigasi dengan Tab, Anda perlu menekan Option+Tab untuk mengubah fokus di browser lain seperti Safari. (Anda bisa mengubah setelan ini di bagian Keyboard pada System Preferences.)



Urutan pemrosesan fokus maju dan mundur pada elemen interaktif lewat Tab disebut *urutan tab*. Langkah penting untuk memastikan bahwa Anda mendesain laman dengan urutan tab yang logis akan kita bahas nanti.

Apa yang dapat difokus?

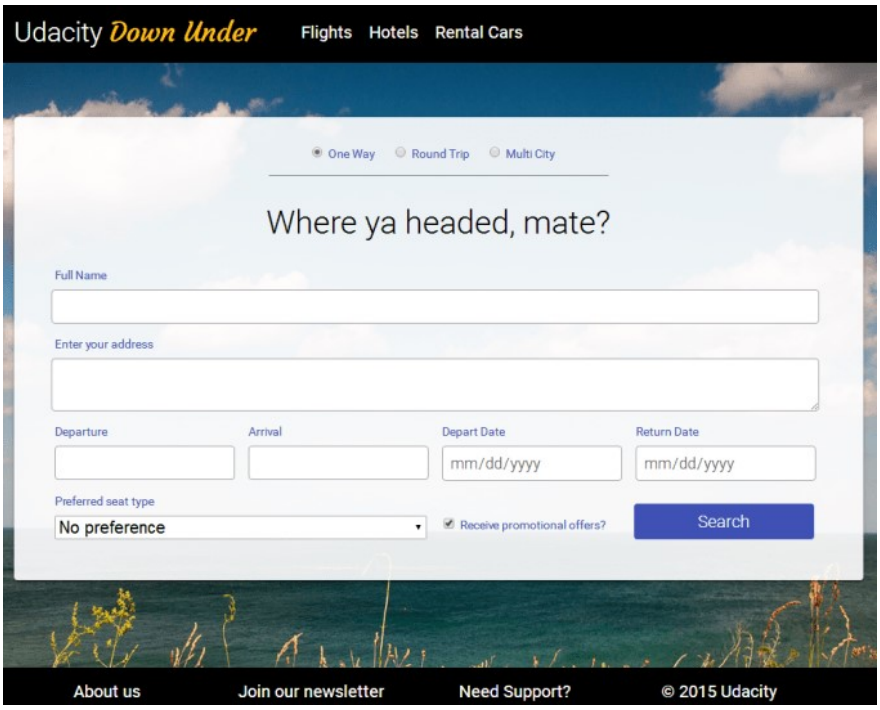
Elemen HTML interaktif bawaan seperti bidang teks, tombol, dan daftar pilihan *secara implisit dapat difokus*, maksudnya, secara otomatis elemen itu akan disisipkan ke dalam urutan tab dan memiliki penanganan kejadian keyboard bawaan tanpa intervensi developer.



Namun tidak semua elemen dapat difokus; paragraf, div, dan beragam elemen laman lainnya tidak difokus saat Anda berpindah tab di laman, dan memang didesain demikian. Umumnya tidak perlu memfokus sesuatu jika pengguna tidak bisa berinteraksi dengannya.

Mencoba fokus

Mari kita coba beberapa teknik fokus yang baru saja kita diskusikan. Dengan menggunakan Chrome, masuklah ke laman tiruan situs maskapai ini dan cari tiker tertentu **hanya dengan menggunakan masukan keyboard**. Laman tersebut tidak akan menerima masukan mouse, jadi Anda tidak bisa memalsukan latihan (bukannya kami tidak percaya Anda ;-).



Parameter tiket yang harus Anda tetapkan adalah:

- sekali jalan
- ke Melbourne
- berangkat tanggal 12 Oktober 2017 (10/12/2017)

- kembali tanggal 23 Oktober 2017 (10/23/2017)
- kursi dekat jendela
- tidak ingin menerima penawaran promosi

Bila Anda berhasil menyelesaikan formulir tanpa kesalahan masukan dan mengaktifkan tombol Search, formulir akan dikosongkan dan disetel ulang begitu saja. Lanjutkan dan selesaikan formulir, kemudian kembalilah.

Mari kita periksa bagaimana formulir menggunakan masukan keyboard Anda. Mulai dengan beberapa penekanan Tab pertama, browser menyorot item navigasi untuk Flights, Hotels, dan Rental Cars. Karena terus menekan Tab Anda melanjutkan ke grup radiobutton untuk memilih dari Round Trip, One Way, atau Multi City dengan menggunakan tombol panah.

Lanjutkan sampai bidang nama dan alamat, dengan mengisi informasi yang diperlukan. Bila sampai di elemen pemilihan tujuan, Anda bisa menggunakan tombol panah untuk memilih kota, atau bisa mulai mengetikkan untuk mengisi bidang pelengkapan otomatis. Begitu pula, dalam bidang tanggal, Anda bisa menggunakan tombol panah atau cuma mengetikkan tanggal.

Memilih tipe kursi juga akan menggunakan tombol panah, atau Anda bisa mengetikkan "w", "a", atau "n" untuk lompat ke opsi kursi. Kemudian Anda bisa menonaktifkan default penawaran promosi dengan menekan spasi saat kotak centang difokus. Terakhir, fokus tombol Search dan tekan Enter untuk mengirim formulir.

Sangat praktis berinteraksi dengan formulir cuma dengan menggunakan keyboard dan tidak perlu beralih ke mouse dan kembali menyelesaikan tugas. Karena semua elemen yang digunakan dalam formulir adalah tag HTML asli dengan fokus implisit, formulir bekerja dengan baik bersama keyboard, dan Anda tidak perlu menulis kode untuk menambahkan atau mengelola perilaku fokus.

5.2. Pentingnya Urutan DOM

Menggunakan elemen asli merupakan cara bagus untuk mempelajari tentang perilaku fokus karena elemen asli secara otomatis disisipkan ke dalam urutan tab berdasarkan posisinya di DOM.

Misalnya, Anda mungkin memiliki tiga elemen tombol yang berurutan dalam DOM. Menekan Tab akan memfokus setiap tombol secara berurutan. Cobalah mengeklik blok kode di bawah ini untuk memindah titik mulai navigasi fokus, kemudian tekan Tab untuk memindah fokus melewati tombol-tombol.

```
<button>I Should</button>
<button>Be Focused</button>
<button>Last!</button>
```

Akan tetapi, penting untuk diingat bahwa, dengan menggunakan CSS, dimungkinkan menempatkan sesuatu dalam satu urutan di DOM namun muncul dalam urutan berbeda di layar. Misalnya, jika Anda menggunakan properti CSS seperti `float` untuk memindah satu tombol ke kanan, tombol-tombol itu akan muncul dalam urutan berbeda di layar. Namun, karena urutannya dalam DOM tetap sama, maka begitu pula urutan tabnya. Bila pengguna berpindah tab di laman, tombol akan mendapat fokus dalam urutan yang tidak intuitif. Cobalah mengeklik blok kode di bawah ini untuk memindah titik mulai navigasi fokus, kemudian tekan `Tab` untuk memindah fokus melewati tombol-tombol.

```
<button style="float: right">I Should</button>
<button>Be Focused</button>
<button>Last!</button>
```

Berhati-hatilah saat mengubah posisi visual elemen di layar dengan menggunakan CSS. Hal ini bisa menyebabkan urutan tab melompat-lompat, seolah acak, sehingga membingungkan pengguna yang mengandalkan keyboard. Karena alasan ini, daftar periksa Web AIM menyatakan di bagian 1.3.2 bahwa urutan navigasi dan pembacaan, seperti yang ditentukan oleh urutan kode, harus logis dan intuitif.

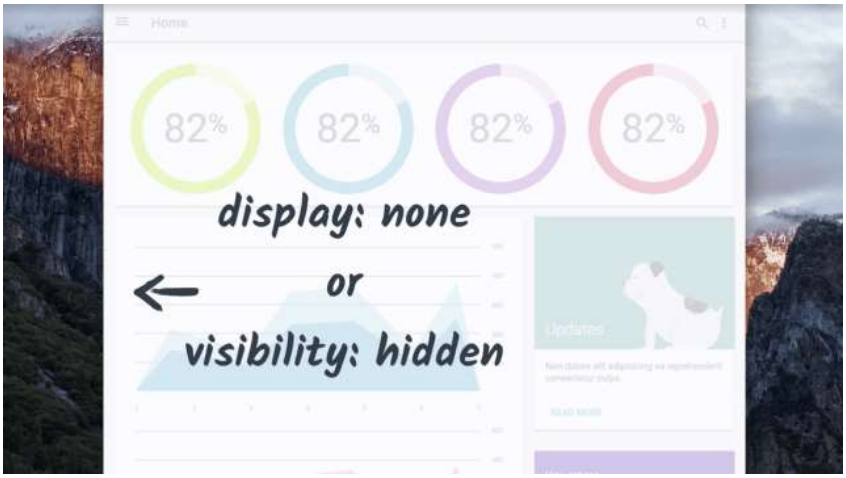
Sebagai aturan, cobalah berpindah-pindah tab melewati berbagai laman sesering mungkin sekadar untuk memastikan Anda bukan secara tidak sengaja mengacaukan urutan tab. Ini adalah kebiasaan yang baik untuk diterapkan, dan hal ini tidak memerlukan banyak usaha.

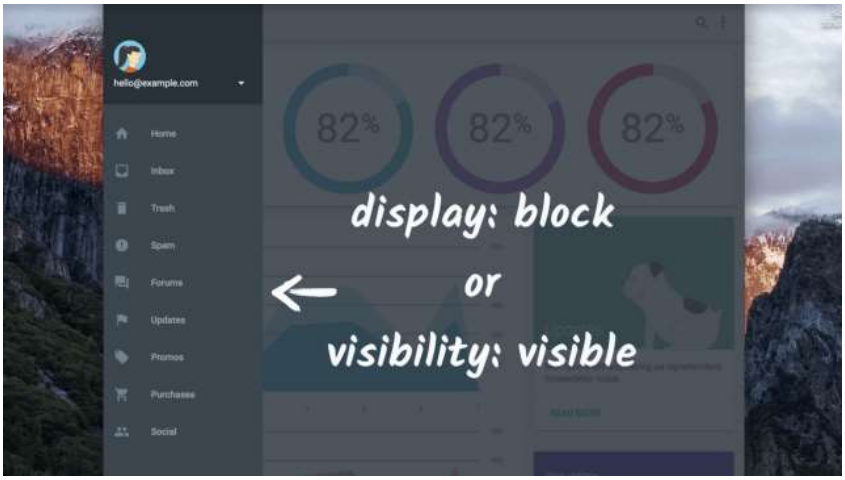
Materi tak terlihat

Bagaimana jika Anda memiliki materi yang saat ini tidak ditampilkan, namun tetap perlu ada di DOM, misalnya navigasi samping yang responsif? Bila Anda memiliki elemen seperti ini yang menerima fokus bila sedang tidak tampak di layar, elemen ini bisa terlihat seakan fokusnya menghilang dan muncul kembali saat pengguna berpindah tab di laman — ini jelas sebuah efek yang tidak diinginkan. Idealnya, kita harus mencegah agar panel tidak mendapat fokus bila sedang di luar layar, dan hanya bisa difokus bila pengguna bisa berinteraksi dengannya.

Kadang-kadang Anda perlu melakukan semacam pekerjaan detektif untuk mengetahui ke mana larinya fokus. Anda bisa menggunakan `document.activeElement` dari konsol untuk mengetahui elemen mana yang saat ini difokus.

Setelah mengetahui elemen di luar layar mana yang sedang difokus, Anda bisa menyetelnya ke `display: none` atau `visibility: hidden`, kemudian menyetelnya kembali ke `display: block` atau `visibility: visible` sebelum menampilkannya kepada pengguna.





Secara umum, kami mendorong developer untuk berpindah-pindah tab dalam situs mereka sebelum mempublikasikan untuk mengetahui apakah urutan tab tidak menghilang atau melompat dari urutan logis. Jika ternyata menghilang atau melompat, pastikan Anda telah menyembunyikan materi di luar layar dengan benar melalui `display: none` atau `visibility: hidden`, atau apakah Anda telah menyusun ulang posisi fisik elemen dalam DOM sehingga berada dalam urutan logis.

5.3 Menggunakan tabindex

Urutan tab default yang disediakan oleh posisi DOM elemen asli sudah praktis, namun ada kalanya Anda ingin memodifikasi urutan tab, dan secara fisik memindahkan elemen di HTML tidak selalu merupakan solusi yang optimal, apalagi layak. Untuk kasus-kasus ini, Anda bisa menggunakan atribut HTML tabindex untuk menyetel posisi tab elemen secara eksplisit.

tabindex bisa diterapkan ke elemen apa saja — walaupun tidak harus berguna pada setiap elemen — dan menggunakan serangkaian nilai integer. Dengan menggunakan tabindex, Anda bisa menetapkan urutan eksplisit untuk elemen laman yang bisa difokus, menyisipkan elemen yang tidak bisa difokus ke dalam urutan tab, dan membuang elemen dari urutan tab. Misalnya:

tabindex="0": Menyisipkan sebuah elemen ke dalam urutan tab alami. Elemen bisa difokus dengan menekan tombol Tab, dan elemen bisa difokus dengan memanggil metode focus() -nya

```
<custom-button tabindex="0">Press Tab to Focus Me!</custom-button>
```

tabindex="-1": Membuang elemen dari urutan tab alami, namun elemen tetap bisa difokus dengan memanggil metode focus() -nya

```
<button id="foo" tabindex="-1">I'm not keyboard focusable</button>
<button onclick="foo.focus();">Focus my sibling</button>
```

tabindex="5": Semua tabindex yang lebih besar dari 0 akan melompatkan elemen ke depan urutan tab alami. Jika ada beberapa elemen dengan tabindex lebih besar dari 0, urutan tab akan mulai dari nilai terendah yang lebih besar dari nol dan terus ke atas. Menggunakan tabindex yang lebih besar dari 0 dianggap sebuah **anti-pola**.

```
<button>I should be first</button>
<button>And I should be second</button>
<button tabindex="5">But I jumped to the front!</button>
```

Hal ini khususnya berlaku untuk elemen non-masukan seperti header, gambar, atau judul artikel. Penambahan tabindex ke elemen semacam itu adalah kontra-produktif. Jika memungkinkan, sebaiknya susun kode sumber Anda agar urutan DOM menyediakan urutan tab yang logis. Jika Anda menggunakan tabindex, batasi pada tombol kontrol interaktif khusus seperti tombol, tab, tarik-turun, dan bidang teks; yakni, elemen yang mungkin dikira pengguna untuk menyediakan masukan.

Jangan khawatir pengguna pembaca layar akan melewatkan materi penting karena tidak memiliki tabindex. Sekalipun materi sangat penting, seperti gambar, jika itu bukanlah hal yang akan berinteraksi dengan pengguna, maka tidak ada alasan membuatnya bisa difokus. Pengguna

pembaca layar tetap bisa memahami materi gambar asalkan Anda menyediakan dukungan atribut alt yang sesuai, yang nanti akan kita bahas sebentar lagi.

5.4 Mengelola fokus pada level laman

Inilah skenario di mana tabindex tidak hanya berguna, melainkan diperlukan. Anda mungkin membangun laman tunggal yang sempurna dengan beragam bagian materi, yang tidak semuanya terlihat sekaligus. Di laman semacam ini, mengeklik tautan navigasi mungkin mengubah materi yang terlihat tanpa melakukan penyegaran laman.

Bila ini terjadi, Anda mungkin akan mengidentifikasi area materi yang dipilih, memberinya sebuah tabindex berupa -1 sehingga tidak muncul dalam urutan tab alami, dan memanggil metode focus-nya. Teknik ini, yang disebut *mengelola fokus*, akan membuat konteks yang dipersepsi pengguna tetap sinkron dengan materi visual situs.

5.5 Mengelola fokus di komponen

Mengelola fokus saat Anda mengubah sesuatu di laman adalah hal penting, namun kadang-kadang Anda perlu mengelola fokus pada level kontrol — misalnya, jika sedang membangun komponen khusus.

Pertimbangkan elemen select asli. Elemen ini bisa menerima fokus dasar, bila ada, Anda bisa menggunakan tombol panah untuk mengekspos fungsionalitas tambahan (opsi yang bisa dipilih). Jika sedang membangun elemen select khusus, Anda tentu ingin mengekspos perilaku semacam ini sehingga pengguna yang terutama mengandalkan keyboard tetap bisa berinteraksi dengan kontrol Anda.

```
<!-- Focus the element using Tab and use the up/down arrow keys to navigate -->
<select>
  <option>Aisle seat</option>
  <option>Window seat</option>
  <option>No preference</option>
</select>
```

Boleh jadi sulit mengetahui perilaku keyboard mana yang akan diimplementasikan, namun ada dokumen berguna yang bisa Anda rujuk. Panduan Accessible Rich Internet Applications (ARIA) Authoring Practices mencantumkan daftar tipe komponen dan macam tindakan keyboard yang didukungnya. Kita akan membahas ARIA secara lebih detail nanti, namun untuk saat ini mari kita gunakan panduan tersebut untuk membantu menambahkan dukungan keyboard ke sebuah komponen baru.

Mungkin Anda sedang mengerjakan beberapa Elemen Khusus baru yang menirukan serangkaian tombol radio, namun dengan penampilan dan perilaku tiruan yang unik.

```
<radio-group>
  <radio-button>Water</radio-button>
  <radio-button>Coffee</radio-button>
  <radio-button>Tea</radio-button>
  <radio-button>Cola</radio-button>
  <radio-button>Ginger Ale</radio-button>
</radio-group>
```

Untuk menentukan dukungan keyboard macam apa yang dibutuhkan, Anda perlu memeriksa panduan ARIA Authoring Practices. Bagian 2 berisi daftar pola desain, dan dalam daftar itu adalah tabel karakteristik untuk grup tombol radio, komponen yang ada yang paling cocok dengan elemen baru Anda.

Seperti yang bisa Anda lihat dalam tabel, salah satu perilaku keyboard umum yang harus didukung adalah tombol panah naik/turun/kiri/kanan. Untuk menambahkan perilaku ini ke komponen baru, kita akan menggunakan teknik yang disebut *roving tabindex*.

W3C Working Draft

Radio Button (widget) #

Characteristics:

Description:	An option in single-select list
Keyboard Interaction:	• Tab key will enter the radio group. ◦ When Tab or Shift+Tab into a radio group, focus goes to the selected radio button. If none is selected, focus goes to the first radio button if Tab was pressed, or the last radio bottom if Shift+Tab was pressed. ◦ When focus is on any radio button, Tab or Shift+Tab will exit the radio group. • Up Arrow and Left Arrow moves focus to the previous radio button in the group, and selects that button. If focus is on the first item, then focus wraps to last item. • Down Arrow and Right Arrow moves focus to the next radio button in the group, and selects that button. If focus is on the last item, then focus wraps to first item. • Control+Arrow moves through the options without updating content or selecting the button. • Space selects the radio button with focus and de-selects other radio buttons in the group.
WAI-ARIA Roles, States, and Properties:	The individual option uses the role list radio. It is a member of a group of radio controls, radiogroup.

Roving tabindex bekerja dengan menyetel `tabindex` ke -1 untuk semua anak selain yang saat ini aktif.

```
<radio-group>
  <radio-button tabindex="0">Water</radio-button>
  <radio-button tabindex="-1">Coffee</radio-button>
  <radio-button tabindex="-1">Tea</radio-button>
  <radio-button tabindex="-1">Cola</radio-button>
  <radio-button tabindex="-1">Ginger Ale</radio-button>
</radio-group>
```

Selanjutnya komponen menggunakan event listener keyboard untuk menentukan tombol mana yang ditekan pengguna; bila terjadi, maka ini akan menyetel tabindex anak yang difokus sebelumnya ke -1, menyetel tabindex anak yang akan difokus ke 0, dan memanggil metode padanya.

```
<radio-group>
  // Assuming the user pressed the down arrow, we'll focus the next available child
  <radio-button tabindex="-1">Water</radio-button>
  <radio-button tabindex="0">Coffee</radio-button> // call .focus() on this element
  <radio-button tabindex="-1">Tea</radio-button>
  <radio-button tabindex="-1">Cola</radio-button>
  <radio-button tabindex="-1">Ginger Ale</radio-button>
</radio-group>
```

Bila pengguna mencapai anak yang terakhir (atau pertama, bergantung pada arahnya menggerakkan fokus), Anda akan membuat loop dan memfokus lagi anak yang pertama (atau terakhir).

Anda bisa mencoba contoh yang telah selesai di bawah ini. Periksa elemen di DevTools untuk mengamati pergerakan tabindex dari satu tombol radio ke tombol berikutnya.

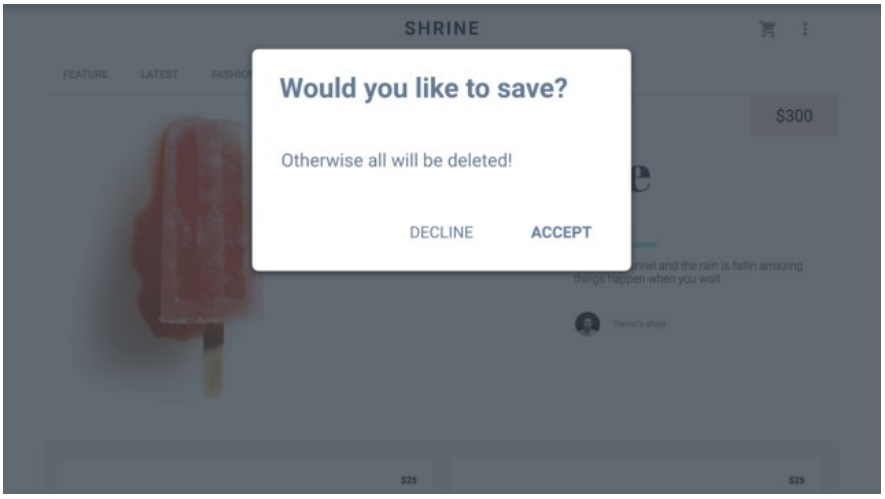
Anda bisa menampilkan sumber lengkap elemen ini melalui GitHub.

5.6 Modal dan jebakan keyboard

Kadang-kadang saat mengelola fokus, Anda bisa masuk ke suatu situasi dan tidak bisa keluar lagi. Perhatikan sebuah widget pelengkapan otomatis yang mencoba mengelola fokus serta merekam perilaku tab, namun mencegah pengguna meninggalkannya bila belum selesai. Ini disebut *jebakan keyboard*, dan hal ini bisa sangat mengesalkan pengguna. Bagian 2.1.2 pada daftar periksa Web AIM menangani masalah ini, yang menyatakan bahwa fokus keyboard tidak boleh dikunci atau terjebak pada satu elemen laman tertentu. Pengguna harus bisa mengarah ke dan dari semua elemen laman hanya dengan menggunakan keyboard.

Anehnya, ada kalanya perilaku ini malah sebenarnya yang diinginkan, seperti jendela modal. Biasanya, bila modal ditampilkan, Anda ingin pengguna mengakses materi di baliknya. Anda mungkin menambahkan overlay untuk menutupi laman secara visual, namun itu tidak menghentikan pergerakan fokus keyboard keluar dari modal secara tidak sengaja.

Dalam instance seperti ini Anda bisa mengimplementasikan jebakan keyboard sementara untuk memastikan bahwa Anda menjebak fokus hanya saat modal ditampilkan kemudian memulihkan fokus item yang sebelumnya difokus bila modal ditutup.



Ada beberapa proposal mengenai cara untuk mempermudah hal ini bagi developer, termasuk elemen <dialog>, namun belum mendapatkan dukungan browser yang luas.

Lihat artikel MDN ini untuk informasi selengkapnya mengenai <dialog>, dan contoh modal untuk informasi selengkapnya mengenai jendela modal.

Perhatikan dialog modal yang dinyatakan oleh div yang berisi beberapa elemen, dan div yang menyatakan overlay latar belakang. Mari kita bahas langkah-langkah dasar yang dibutuhkan untuk mengimplementasikan jebakan keyboard sementara di situasi ini.

1. Dengan menggunakan document.querySelector, pilih div modal dan overlay serta simpan referensinya.
2. Saat modal dibuka, simpan referensi ke elemen yang telah difokus ketika modal dibuka sehingga Anda bisa mengembalikan fokus ke elemen itu.
3. Gunakan *keydown listener* untuk menangkap tombol-tombol yang ditekan saat modal dibuka. Anda juga mendengarkan klik di overlay latar belakang, dan menutup modal jika pengguna mengekliknya.
4. Berikutnya, ambil kumpulan elemen yang bisa difokus dalam modal. Elemen yang bisa difokus pertama dan terakhir akan berfungsi sebagai "sentinel" untuk memberi tahu Anda kapan membuat loop fokus ke depan atau ke belakang agar tetap berada dalam modal.
5. Tampilkan jendela modal dan fokus elemen yang bisa difokus pertama.
6. Saat pengguna menekan Tab atau Shift+Tab, pindahkan fokus ke depan atau ke belakang, dengan melakukan loop pada elemen terakhir atau pertama.
7. Jika pengguna menekan Esc, tutuplah modal. Hal ini sangat membantu karena memungkinkan pengguna menutup modal tanpa menelusuri tombol tutup tertentu, dan hal ini menguntungkan bahkan untuk pengguna yang menggunakan mouse.

8. Bila tombol modal ditutup, sembunyikan ini dan overlay latar belakang, serta pulihkan fokus ke elemen yang telah difokus sebelumnya yang telah disimpan.

Prosedur ini memberi Anda jendela modal yang berguna dan tidak mengesalkan, yang bisa digunakan siapa saja secara efektif.

Untuk detail selengkapnya, Anda bisa memeriksa kode contoh ini, dan menampilkan contoh langsung dari laman yang telah selesai.

BAB VI PENGANTAR SEMANTIK

Sasaran Pembelajaran

Mahasiswa mengetahui apa yang dimaksud aksesibilitas dan cara menerapkannya serta mengetahui tentang berbagai teknik aksesibilitas modern untuk membuat pengalaman aksesibilitas yang brilian

Kemampuan mahasiswa yang menjadi prasyarat

Mahasiswa sudah menguasai materi Interaksi Manusia Komputer

Manfaat atau pentingnya bahan pembelajaran ini

Memberi pemahaman mengenai aksesibilitas

Anda telah melihat cara membuat situs bisa diakses oleh pengguna yang tidak bisa menggunakan mouse atau perangkat penunjuk — baik karena cacat fisik, masalah teknologi, atau preferensi pribadi — dengan menangani penggunaan keyboard-saja. Walaupun memerlukan kehati-hatian dan pertimbangan, ini bukanlah pekerjaan yang sangat berat jika Anda merencanakannya dari awal. Setelah pekerjaan dasar selesai, Anda tinggal menuju ke situs yang bisa diakses penuh dan lebih cemerlang.

Dalam pelajaran ini, kita akan mendasarkan pada pekerjaan itu dan membuat Anda memikirkan tentang faktor aksesibilitas lainnya, misalnya cara membangun situs web untuk mendukung pengguna seperti Victor Tsaran, yang tidak bisa melihat layar.

Terlebih dahulu, kita akan mendapatkan latar belakang mengenai *teknologi pendukung*, yakni istilah umum untuk alat seperti pembaca layar untuk membantu pengguna yang cacat agar tetap bisa mengakses informasi.

Berikutnya, kita akan mengamati beberapa konsep umum pengalaman pengguna, dan menjadikannya sebagai dasar untuk mendalami pengalaman pengguna teknologi pendukung.

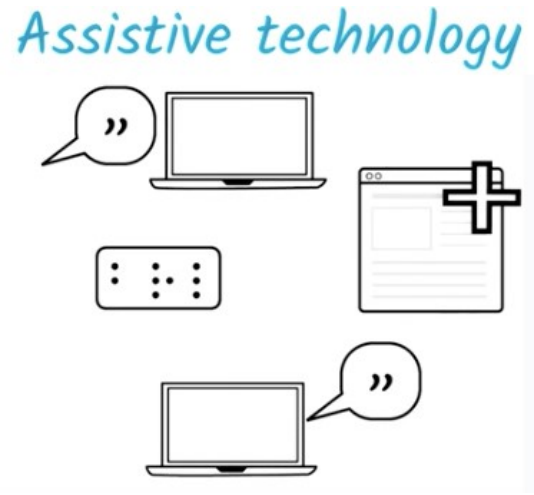
Terakhir, kita akan melihat cara menggunakan HTML secara efektif untuk membuat pengalaman yang baik bagi para pengguna ini, dan bagaimana hal itu sedikit tumpang tindih dengan cara kita menangani fokus sebelumnya.

6.1 Teknologi pendukung

Teknologi pendukung adalah istilah umum untuk perangkat, perangkat lunak, dan alat yang membantu penyandang cacat melakukan suatu tugas. Dalam pengertian paling luas, teknologi ini bisa berupa teknologi rendah seperti kruk untuk berjalan atau kaca pembesar untuk

membaca, atau teknologi tinggi seperti lengan robotik atau perangkat lunak pengenalan gambar di ponsel cerdas.

Teknologi pendukung bisa meliputi sesuatu yang umum seperti zoom di browser zoom, atau spesifik seperti pengontrol game yang didesain khusus. Teknologi ini berupa perangkat fisik terpisah seperti tampilan braille, atau diimplementasikan lengkap dalam perangkat lunak seperti kontrol suara. Teknologi pendukung bisa ditanamkan dalam sistem operasi seperti pembaca layar, atau berupa pengaya seperti ekstensi Chrome.



Garis antara teknologi pendukung dan teknologi secara umum adalah kabur; semua teknologi dimaksudkan untuk membantu orang melakukan tugas atau hal lain. Dan teknologi sering kali bisa masuk ke dan keluar dari kategori "pendukung".

Misalnya, salah satu produk sintesis ucapan komersial paling awal adalah kalkulator bicara untuk tuna netra. Kini sintesis ucapan ada di mana-mana, dari arah mengemudi hingga asisten virtual. Sebaliknya, teknologi yang pada mulanya serba guna sering kali digunakan sebagai pendukung. Misalnya, orang yang lemah penglihatan mungkin menggunakan zoom kamera di ponsel cerdas untuk melihat lebih baik benda kecil di dunia nyata.

Dalam konteks development web, kita harus mempertimbangkan berbagai macam teknologi. Orang mungkin berinteraksi dengan situs web menggunakan pembaca layar atau tampilan braille, dengan pembesar layar, melalui kontrol suara, menggunakan perangkat switch, atau dengan bentuk teknologi pendukung lainnya yang mengadaptasikan antarmuka default laman untuk membuat antarmuka yang lebih spesifik yang bisa mereka gunakan.

Banyak dari teknologi pendukung ini yang mengandalkan *semantik yang dinyatakan lewat program* untuk membuat pengalaman pengguna yang bisa diakses, dan itulah yang sebagian besar akan dibahas pelajaran ini. Namun sebelum bisa menjelaskan semantik yang dijelaskan lewat program, kita perlu membicarakan sedikit tentang *kemampuan*.

6.2 Kemampuan

Bila kita menggunakan alat (bantu) atau perangkat atau buatan manusia, biasanya kita melihat bentuk dan desainnya untuk memberikan gambaran mengenai apa manfaatnya dan cara kerjanya. *Kemampuan* adalah objek yang menawarkan, atau memberi, penggunanya kesempatan untuk melakukan suatu aksi. Semakin baik kemampuan tersebut didesain, semakin nyata atau intuitif penggunaannya.

Contoh klasik adalah cerek atau teko teh. Anda bisa dengan mudah mengetahui bahwa Anda harus mengambilnya melalui pegangannya, bukan di lehernya, sekalipun Anda belum pernah melihat teko itu sebelumnya.

Itu sebabnya kemampuan di sini mirip dengan kemampuan yang Anda lihat pada benda lainnya -- ceret penyiram, teko minuman, mug kopi, dan seterusnya. Anda barangkali *bisa* mengangkat teko pada lehernya, namun pengalaman Anda dengan kemampuan serupa akan memberi tahu pegangannya adalah opsi yang lebih baik.

Dalam Graphic User Interface, kemampuan menyatakan tindakan yang bisa kita ambil, namun hal itu bisa jadi meragukan karena tidak berinteraksi dengan objek fisik. Kemampuan GUI didesain khusus agar tidak meragukan: tombol, kotak centang, dan bilah gulir dimaksudkan untuk memberitahukan penggunaannya dengan pelatihan yang sesedikit mungkin.

Misalnya, Anda mungkin menafsirkan penggunaan beberapa elemen bentuk umum (kemampuan) seperti ini:

- Tombol radio — "Saya bisa memilih salah satu opsi ini."
- Kotak centang — "Saya bisa memilih 'ya' atau 'tidak' terhadap opsi ini."
- Bidang teks — "Saya bisa mengetikkan sesuatu ke dalam area ini."
- Menu tarik-turun — "Saya bisa membuka elemen ini untuk menampilkan opsi saya."

Anda bisa menarik kesimpulan tentang elemen ini *hanya karena Anda bisa melihatnya*. Secara alami, orang yang tidak bisa melihat petunjuk visual yang disediakan oleh sebuah elemen tidak bisa memahami maknanya atau secara intuitif menangkap nilai kemampuan. Jadi kita harus memastikan informasi diekspresikan cukup fleksibel untuk diakses oleh teknologi pendukung yang bisa membentuk suatu antarmuka alternatif agar cocok dengan kebutuhan penggunanya.

Paparan non-visual atas penggunaan kemampuan ini disebut *semantik*.

6.3 Pembaca layar

Satu tipe teknologi pendukung yang populer adalah *pembaca layar*, yaitu sebuah program yang memungkinkan orang yang memiliki masalah penglihatan untuk menggunakan komputer dengan membacakan teks layar dengan suara buatan. Pengguna bisa mengontrol apa yang dibaca dengan menggerakkan kursor ke area yang relevan dengan keyboard.

Kami meminta Victor Tsaran untuk menjelaskan bagaimana, sebagai orang buta, ia bisa mengakses web dengan menggunakan pembaca layar bawaan di OS X, yang disebut VoiceOver. Lihat video ini tentang Victor yang menggunakan VoiceOver.

Kini, giliran Anda mencoba menggunakan pembaca layar. Inilah laman dengan *ChromeVox Lite*, pembaca layar minimal namun fungsional yang ditulis dalam JavaScript. Layar sengaja dikaburkan untuk menyimulasikan pengalaman penglihatan-minim dan memaksa pengguna untuk melakukan tugas dengan pembaca layar. Tentu saja, Anda perlu menggunakan browser Chrome untuk latihan ini.

Laman demo ChromeVox lite

Anda bisa menggunakan panel kontrol di bagian bawah layar untuk mengontrol pembaca layar. Pembaca layar ini memiliki fungsionalitas sangat minim, namun Anda bisa menyusuri materi dengan menggunakan tombol Previous dan Next, dan bisa mengeklik sesuatu dengan menggunakan tombol Click.

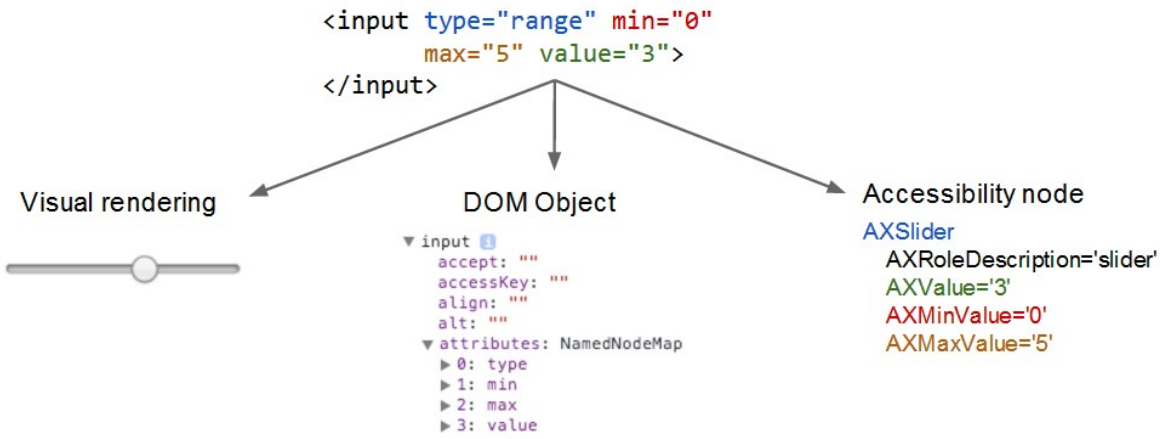
Cobalah menggunakan laman ini dengan ChromeVox lite yang telah diaktifkan untuk merasakan penggunaan pembaca layar. Ingatlah sebenarnya pembaca layar (atau teknologi pendukung lainnya) membuat suatu pengalaman pengguna alternatif lengkap bagi pengguna

berdasarkan pada semantik yang diekspresikan lewat program. Sebagai ganti antarmuka visual, pembaca layar menyediakan antarmuka yang terdengar.

Perhatikan bagaimana pembaca layar memberi tahu Anda informasi tentang setiap elemen antarmuka. Anda tentunya mengharapkan pembaca yang didesain dengan baik untuk memberi tahu Anda semua, atau setidaknya, informasi tentang elemen yang ditemukannya berikut ini.

- *Peran* atau tipe elemen, jika telah ditetapkan (seharusnya sudah).
- *Nama* elemen, jika memiliki (seharusnya sudah).
- *Nilai* elemen, jika memilikinya (mungkin atau mungkin tidak).
- *Keadaan* elemen, mis., apakah diaktifkan atau dinonaktifkan (jika berlaku).

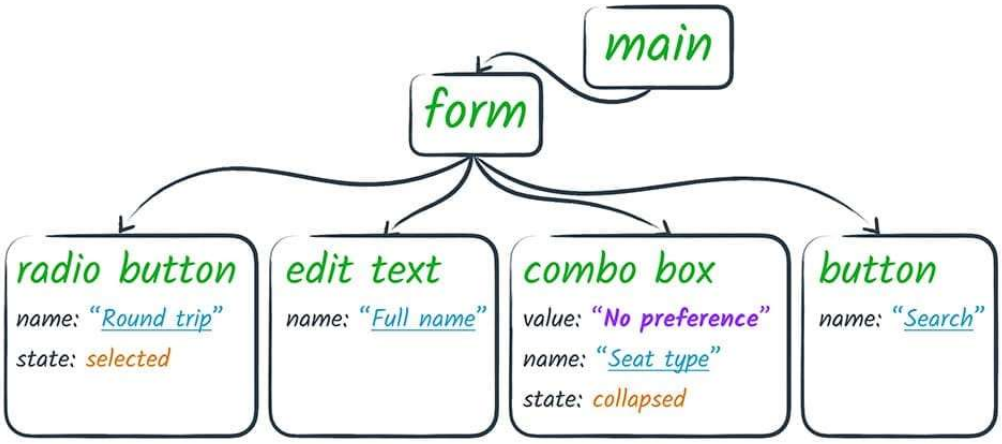
Pembaca layar mampu membentuk UI alternatif ini karena elemen asli berisi metadata aksesibilitas bawaan. Karena mesin rendering menggunakan kode asli untuk membentuk antarmuka visual, pembaca layar akan menggunakan metadata dalam simpul DOM untuk membentuk versi yang bisa diakses, seperti yang satu ini.



6.4 Pohon Aksesibilitas

Bayangkan Anda sedang membangun antarmuka pengguna *khusus untuk pengguna pembaca layar*. Di sini, Anda tidak perlu membuat UI visual sama sekali, melainkan cuma menyediakan informasi yang cukup untuk digunakan pembaca layar.

Yang akan Anda buat adalah semacam API yang menjelaskan struktur laman, mirip dengan DOM API, namun Anda bisa menghindar dengan sedikit informasi dan simpul lebih sedikit, karena banyak dari informasi itu yang hanya berguna bagi presentasi visual. Penampilannya mungkin seperti ini.



Pada dasarnya inilah yang sesungguhnya akan disajikan ke pembaca layar. Browser mengambil pohon DOM dan memodifikasinya menjadi suatu bentuk yang berguna untuk teknologi pendukung. Kami menyebut pohon yang telah dimodifikasi ini dengan *Pohon Aksesibilitas*.

Anda mungkin membayangkan pohon aksesibilitas ini seperti mirip dengan laman web tua dari tahun 90-an: sedikit gambar, banyak tautan, mungkin dengan satu bidang dan tombol.

Dengan memindai laman secara visual seperti ini akan memberi Anda pengalaman yang mirip dengan apa yang akan didapat oleh pengguna pembaca layar. Antarmuka memang ada, namun sederhana dan langsung, mirip sekali dengan antarmuka pohon aksesibilitas.

Kebanyakan teknologi pendukung berinteraksi dengan pohon aksesibilitas. Alurnya berjalan seperti ini.

1. Sebuah aplikasi (browser atau aplikasi lainnya) mengekspos versi semantik UI-nya kepada teknologi pendukung melalui API.
2. Teknologi pendukung dapat menggunakan informasi yang dibacanya melalui API untuk membuat presentasi antarmuka pengguna alternatif bagi pengguna. Misalnya, pembaca

layar akan membuat sebuah antarmuka yang digunakan pengguna untuk mendengarkan representasi aplikasi yang dibacakan.

3. Teknologi pendukung bisa juga memungkinkan pengguna berinteraksi dengan aplikasi dalam cara berbeda. Misalnya, kebanyakan pembaca layar menyediakan kait yang memungkinkan pengguna dengan mudah menyimulasikan klik mouse atau ketukan jari.
4. Teknologi pendukung yang menyampaikan maksud pengguna (misalnya "klik") kembali ke aplikasi melalui API aksesibilitas. Selanjutnya aplikasi bertanggung jawab untuk menafsirkan aksi sebagaimana mestinya dalam konteks UI asal.

Untuk browser web, ada langkah ekstra di setiap arah, karena browser sebenarnya adalah platform untuk menjalankan aplikasi web. Jadi browser perlu menerjemahkan aplikasi web menjadi pohon aksesibilitas, dan harus memastikan bahwa kejadian yang sesuai akan dipicu di JavaScript berdasarkan tindakan pengguna yang berasal dari teknologi pendukung.

Namun itu semua adalah tanggung jawab browser. Pekerjaan kita sebagai web developer sekadar mengetahui bahwa ini terjadi, dan mengembangkan laman web yang memanfaatkan proses ini untuk membuat suatu pengalaman yang bisa diakses oleh pengguna kita.

Kita melakukannya dengan memastikan bahwa kita mengekspresikan semantik laman dengan benar: dengan memastikan elemen penting di laman memiliki peran, keadaan, dan properti yang bisa diakses dengan benar, dan bahwa kita menetapkan nama dan keterangan yang bisa diakses. Selanjutnya browser bisa memungkinkan teknologi pendukung mengakses informasi itu untuk membuat pengalaman yang disesuaikan.

6.5 Semantik di HTML asli

Browser bisa mengubah pohon DOM menjadi sebuah pohon aksesibilitas karena kebanyakan DOM memiliki makna semantik *implisit*. Yakni, DOM menggunakan elemen HTML asli yang dikenali oleh browser dan berfungsi dengan cara yang bisa diprediksi pada berbagai platform. Aksesibilitas untuk elemen HTML asli seperti tautan atau tombol dengan demikian ditangani secara otomatis. Kita bisa memanfaatkan aksesibilitas bawaan itu dengan menulis HTML yang mengekspresikan semantik elemen laman kita.

Akan tetapi, kadang-kadang kita menggunakan elemen yang tampak seperti elemen asli padahal bukan. Misalnya, "tombol" bukanlah tombol sama sekali.

Ini dapat dibuat di HTML dengan banyak cara; salah satu caranya ditampilkan di bawah ini.

```
<div class="button-ish">Give me tacos</div>
```

Bila kita tidak menggunakan elemen tombol sesungguhnya, pembaca layar tidak memiliki cara untuk mengetahui telah sampai di mana. Selain itu, kita nanti harus melakukan pekerjaan ekstra berupa penambahan tabindex untuk membuatnya bisa digunakan oleh pengguna

keyboard-saja karena, berhubung sekarang telah dibuat kodenya, maka hanya bisa digunakan dengan mouse.

Kita bisa memperbaikinya secara mudah dengan menggunakan elemen button biasa sebagai ganti div. Penggunaan elemen asli juga berguna untuk menjagakan interaksi keyboard buat kita. Ingatlah bahwa Anda tidak harus kehilangan efek visual yang menyenangkan hanya lantaran menggunakan elemen asli; Anda bisa menata gaya elemen asli untuk membuatnya terlihat seperti yang Anda inginkan dan tetap mempertahankan semantik implisit dan perilakunya.

Sebelumnya kita telah memperhatikan bahwa pembaca layar akan membacakan peran, nama, keadaan, dan nilai elemen. Dengan menggunakan semantik yang tepat elemen, peran, keadaan, dan nilai telah tercakup, namun kita juga harus memastikan bahwa kita membuat nama elemen yang dapat ditemukan.

Secara umum, ada dua tipe nama:

- *Label yang terlihat*, yang digunakan oleh semua pengguna untuk mengaitkan makna dengan elemen, dan
- *Alternatif berupa teks*, yang hanya digunakan bila tidak memerlukan label visual.

Untuk elemen level-teks, kita tidak perlu melakukan apa-apa, karena menurut definisi, elemen akan berisi beberapa teks. Akan tetapi, untuk elemen masukan atau elemen kontrol, serta materi visual seperti gambar, kita perlu memastikan bahwa kita menetapkan sebuah nama. Sebenarnya, menyediakan alternatif berupa teks bagi materi non-teks adalah item paling pertama pada daftar periksa WebAIM.

Salah satu cara melakukannya adalah mengikuti saran bahwa "Masukan formulir memiliki label teks terkait." Ada dua cara untuk mengaitkan label dengan elemen formulir, misalnya kotak centang. Salah satu dari metode ini menyebabkan teks label juga menjadi target klik untuk kotak centang, sehingga juga berguna bagi pengguna mouse atau layar sentuh. Untuk mengaitkan label dengan elemen, bisa dengan

Menempatkan elemen masukan di dalam elemen label

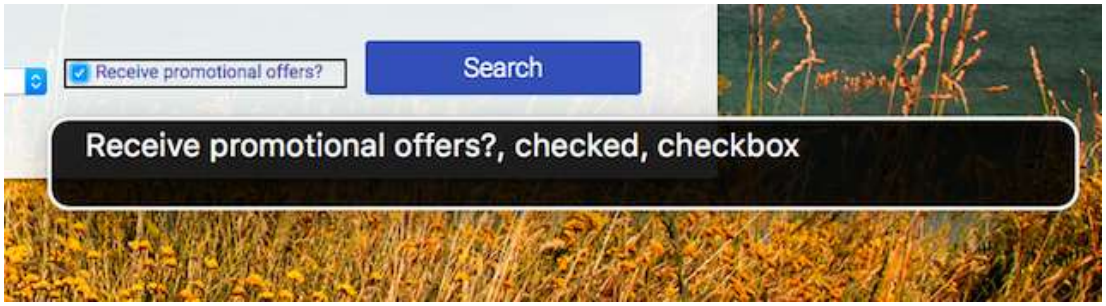
```
<label>
  <input type="checkbox">Receive promotional offers?</input>
</label>
```

atau

Mengunakan atribut for label dan merujuk id elemen

```
<input id="promo" type="checkbox"></input>
<label for="promo">Receive promotional offers?</label>
```

Bila kotak centang telah diberi label dengan benar, pembaca layar bisa melaporkan bahwa elemen memiliki peran kotak centang, dalam keadaan dicentang, dan dinamai "Receive promotional offers?".



Berhasil: Sebenarnya Anda bisa menggunakan pembaca layar untuk menemukan label yang tidak dikaitkan dengan benar dengan berpindah-pindah tab di laman dan memverifikasi peran, keadaan, dan nama yang dibacakan.

6.6 Alternatif Berupa Teks untuk Gambar

Gambar adalah komponen penting pada sebagian besar laman web, dan tentu saja menjadi titik-lekat khusus bagi pengguna yang lemah penglihatannya. Kita harus mempertimbangkan peran yang dimainkan gambar di laman untuk merencanakan tipe alternatif berupa teks yang harus digunakan. Perhatikan gambar ini.

```
<article>
  <h2>Study shows 9 out of 10 cats quietly judging their owners as they sleep</h2>
  <img src="imgs/160204193356-01-cat-500.jpg">
</article>
```

Studi menunjukkan 9 dari 10 kucing dengan tenang menilai pemiliknya saat mereka tidur

Di sini, kita memiliki gambar seekor kucing, yang mengilustrasikan perilaku kecenderungan menilai yang sudah dikenali pada kucing. Pembaca layar akan membacakan gambar ini dengan menggunakan nama literalnya, "/160204193356-01-cat-500.jpg". Itu memang akurat, namun tidak berguna sama sekali.

Anda bisa menggunakan atribut alt untuk menyediakan alternatif berupa teks yang berguna bagi gambar — misalnya, "A cat staring menacingly off into space."

```
<img src="/160204193356-01-cat-500.jpg" alt="A cat staring menacingly off into space">
```

Kemudian pembaca layar bisa membacakan keterangan singkat mengenai gambar tersebut (terlihat di bilah hitam VoiceOver) dan pengguna bisa memilih apakah akan berpindah ke artikel tersebut.

Sepasang komentar tentang alt:

- alt memungkinkan Anda menetapkan string sederhana untuk digunakan bila gambar tidak tersedia, misalnya bila gambar gagal dimuat, atau diakses melalui bot perayapan web, atau kebetulan ditemukan oleh pembaca layar.
- alt berbeda dengan title, atau tipe teks, yang *hanya* akan digunakan jika gambar tidak tersedia.

Menulis teks alternatif yang berguna ada seninya. Agar string bisa digunakan, alternatif berupa teks perlu menyampaikan konsep yang sama dengan gambarnya, dalam konteks yang sama.

Perhatikan gambar logo yang ditautkan di masthead laman seperti yang ditampilkan di atas. Kita bisa menjelaskan gambar tersebut dengan sangat akurat sebagai "logo The Funion".

```
<img class="logo" src="logo.jpg" alt="Logo The Funion">
```

Mungkin kita tergoda untuk memberikan alternatif berupa teks yang lebih sederhana berupa "beranda" atau "laman utama", namun itu tidak menguntungkan bagi pengguna yang penglihatannya lemah maupun tajam.

Namun bayangkan seorang pengguna pembaca layar yang ingin mencari logo masthead di laman; memberinya nilai alt berupa "beranda" sesungguhnya malah akan menambah bingung. Pengguna yang berpenglihatan tajam pun akan menghadapi kendala yang sama — yang mengetahui logo situs dengan mengekliknya — seperti halnya pengguna pembaca layar.

Di lain pihak, menjelaskan gambar tidak selalu berguna. Misalnya, perhatikan gambar kaca pembesar di dalam tombol telusur yang berisi teks "Telusur". Jika teks itu tidak ada, Anda pasti akan memberi gambar itu nilai alternatif berupa "telusur". Namun karena kita memiliki teks yang terlihat, pembaca layar akan mengambil dan membacakan kata "telusur"; sehingga, nilai `alt` yang identik pada gambar menjadi berlebihan.

Akan tetapi, kita tahu bahwa jika membiarkan teks `alt`, mungkin kita akan mendengar nama file gambar sebagai gantinya, yang tidak ada gunanya dan mungkin akan membingungkan. Dalam hal ini, Anda bisa menggunakan atribut `alt` kosong saja, dan pembaca layar akan melewati gambar sama sekali.

```
<img src="magnifying-glass.jpg" alt="">
```

Singkatnya, semua gambar harus memiliki atribut `alt`, namun tidak semuanya harus memiliki teks. Gambar penting harus memiliki teks alternatif penjelas yang menjelaskan secara singkat mengenai gambar itu, sedangkan gambar dekoratif harus memiliki atribut alt yang kosong — yakni, `alt=""`.

6.7 Semantik dan Menyusuri Materi

Anda telah mempelajari tentang kemampuan, semantik, dan bagaimana teknologi pendukung menggunakan pohon aksesibilitas untuk membuat pengalaman pengguna alternatif bagi pengguna mereka. Anda bisa melihat bahwa menulis HTML semantik yang ekspresif akan memberi banyak aksesibilitas dengan upaya sangat kecil, karena banyak elemen standar memiliki semantik dan perilaku pendukung bawaan.

Dalam pelajaran ini, kita akan membahas beberapa semantik yang kurang dimengerti namun sangat penting bagi pengguna pembaca layar, terutama berkenaan dengan navigasi. Di laman sederhana yang berisi banyak kontrol namun tidak banyak materinya, akan mudah memindai laman untuk menemukan apa yang Anda butuhkan. Namun pada laman yang sarat materi, seperti entri Wikipedia atau agregator berita, tidak praktis membaca tuntas semua hal dari atas ke bawah; Anda perlu cara untuk menyusuri materinya secara efisien.

Developer sering kali salah memahami bahwa pembaca layar membosankan dan lambat digunakan, atau bahwa segala sesuatu di layar harus bisa difokus agar bisa ditemukan oleh pembaca layar. Sering kali bukan itu masalahnya.

Pengguna pembaca layar sering kali mengandalkan daftar heading untuk menemukan informasi. Kebanyakan pembaca layar memiliki cara mudah untuk mengisolasi dan memindai daftar heading laman, yakni sebuah fitur penting yang disebut *rotor*. Mari kita lihat cara menggunakan heading HTML secara efektif untuk mendukung fitur ini.

6.8 Menggunakan heading secara efektif

Pertama, mari kita ulangi kembali poin sebelumnya: *urutan DOM itu penting*, bukan hanya untuk urutan fokus melainkan untuk urutan pembaca layar. Saat Anda bereksperimen dengan pembaca layar seperti VoiceOver, NVDA, JAWS, dan ChromeVox, Anda akan menemukan daftar heading mengikuti urutan DOM, bukan urutan visual.

Hal ini berlaku untuk pembaca layar pada umumnya. Karena pembaca layar berinteraksi dengan pohon aksesibilitas, dan pohon aksesibilitas berdasarkan pada pohon DOM, urutan yang dipahami oleh pembaca dengan demikian berdasarkan pada urutan DOM. Ini berarti struktur heading yang tepat menjadi kian penting.

Di kebanyakan laman yang terstruktur dengan baik, level heading disarangkan untuk menunjukkan hubungan induk-anak di antara blok materi. Daftar Periksa WebAIM berulang kali merujuk teknik ini.

- 1.3.1 menyebutkan "Markup semantik digunakan untuk menunjukkan heading"
- 2.4.1 menyebutkan struktur heading sebagai teknik untuk melangkahi blok materi
- 2.4.6 mendiskusikan beberapa detail untuk penulisan heading yang berguna
- 2.4.10 menyebutkan "masing-masing bagian materi ditetapkan menggunakan heading, bila memang sesuai"

Tidak semua heading harus terlihat di layar. Wikipedia, misalnya, menggunakan teknik yang sengaja menempatkan sebagian heading di luar layar untuk membuatnya *hanya* bisa diakses oleh pembaca layar dan teknologi pendukung lainnya.

```
<style>
 .sr-only {
   position:absolute;
   left:-10000px;
   top:auto;
   width:1px;
   height:1px;
   overflow:hidden;
 }
</style>

<h2 class="sr-only">This heading is
offscreen.</h2>
```

Note: Situs WebAIM mendiskusikan teknik ini panjang lebar dalam artikel ini di materi di luar layar.

Bagi aplikasi yang kompleks, ini bisa menjadi cara yang bagus untuk mengakomodasi heading bila desain visual tidak memerlukan atau memiliki ruang bagi heading yang terlihat.

Perhatian: Perlu kiranya kita tidak bertindak lebih jauh dengan teknik ini. Ingatlah bahwa pengguna teknologi pendukung mungkin juga dapat melihat layar bagi dirinya sendiri, jadi melangkah terlalu jauh ke pembuatan materi untuk "pembaca layar saja" sebenarnya mungkin akan menurunkan kualitas pengalaman pengguna bagi sebagian orang. Hal ini juga bisa membuat Anda kerepotan dalam pemeliharaannya nanti.

Opsi navigasi lainnya

Walaupun laman dengan heading yang baik akan membantu navigasi pengguna pembaca layar, ada beberapa elemen lainnya yang bisa mereka gunakan untuk menyusuri laman, termasuk *tautan*, *kontrol formulir*, dan *landmark*.

Pembaca bisa menggunakan fitur rotor milik pembaca layar (cara mudah untuk mengisolasi dan memindai daftar heading laman) untuk mengakses *daftar tautan* di laman tersebut. Kadang-kadang, seperti di wiki, ada banyak tautan, jadi pembaca dapat menelusuri suatu istilah dalam tautan tersebut. Ini akan membatasi hit pada tautan yang sebenarnya berisi istilah tersebut, bukannya setiap temuan istilah pada laman.

Fitur ini hanya berguna jika pembaca layar bisa menemukan tautan tersebut dan teks tautan itu bermakna. Misalnya, ini beberapa pola umum yang membuat tautan sulit ditemukan.

- Tag jangkar tanpa atribut href. Sering kali digunakan dalam aplikasi laman-tunggal, target tautan ini menyebabkan masalah bagi pembaca layar. Anda bisa membaca selengkapnya di artikel mengenai aplikasi laman-tunggal ini.
- Tombol yang diimplementasikan bersama tautan. Hal ini menyebabkan pembaca layar menafsirkan materi sebagai tautan, dan fungsionalitas tombol akan hilang. Untuk kasus-kasus ini, ganti tag jangkar dengan tombol sungguhan dan beri gaya dengan semestinya.
- Gambar digunakan sebagai materi tautan. Kadang-kadang diperlukan, gambar bertautan bisa menjadi tidak berguna bagi pembaca layar. Untuk menjamin bahwa tautan diekspos dengan benar ke teknologi pendukung, pastikan gambar tersebut memliki teks atribut alt.

Masalah lain adalah tautan yang buruk. Teks yang bisa diklik seperti "ketahui selengkapnya" atau "klik di sini" tidak menyediakan informasi semantik tentang akan ke mana tautan tersebut. Sebagai gantinya, gunakan teks deskriptif seperti "ketahui selengkapnya tentang desain responsif" atau "lihat tutorial kanvas ini" untuk membantu pembaca layar menyediakan konteks yang bermakna tentang tautan.

Rotor bisa juga mengambil *daftar kontrol formulir*. Dengan daftar ini, pembaca bisa menelusuri item tertentu dan langsung menuju ke sana.

Kesalahan umum yang dibuat pembaca layar adalah pengucapan. Misalnya, pembaca layar mungkin mengucapkan "Udacity" sebagai "oo-dacity", atau membacakan nomor ponsel sebagai integer besar, atau membaca teks berhuruf besar seakan berupa akronim. Menariknya, pengguna pembaca layar sudah sangat terbiasa dengan keganjilan ini dan telah memperhitungkannya.

Sebagian developer mencoba memperbaiki situasi ini dengan menyediakan teks khusus pembaca layar yang dieja secara fonetik. Inilah aturan sederhana untuk ejaan fonetik: **jangan lakukan**; ini hanya akan memperburuk masalah! Jika, misalnya, pengguna menggunakan tampilan braille, kata akan salah eja, sehingga menyebabkan semakin bingung. Pembaca layar memungkinkan kata dieja nyaring, sehingga membiarkan pembaca mengontrol pengalaman mereka dan memutuskan kapan memerlukannya.

Pembaca bisa menggunakan rotor untuk melihat *daftar landmark*. Ini membantu pembaca menemukan materi utama dan serangkaian landmark navigasi yang disediakan oleh elemen landmark HTML.

HTML5 memperkenalkan beberapa elemen baru yang membantu mendefinisikan struktur semantik laman, termasuk header, footer, nav, article, section, main, dan aside. Semua elemen ini secara spesifik menyediakan petunjuk struktural di laman tanpa memaksakan penataan gaya bawaan (yang bagaimana pun juga harus Anda lakukan dengan CSS).

Elemen struktural semantik menggantikan beberapa blok div repetitif, dan menyediakan cara yang lebih jelas dan lebih deskriptif untuk menyatakan struktur laman secara intuitif baik bagi penulis maupun pembaca.

BAB VII Pengantar ARIA

Sejauh ini, kita didorong untuk menggunakan elemen HTML asli karena memberi Anda fokus, dukungan keyboard, dan semantik bawaan, namun ada kalanya layout sederhana dan HTML asli tidak dapat menjalankan tugasnya. Misalnya, saat ini tidak ada elemen HTML terstandardisasi untuk bentuk UI umum, menu munculan. Atau tidak ada elemen HTML yang menyediakan karakteristik semantik, misalnya "pengguna perlu mengetahui tentang hal ini secepatnya".

Dalam pelajaran ini, nanti, kita akan mendalami cara mengekspresikan semantik yang tidak bisa dinyatakan dalam HTML.

Spesifikasi Web Accessibility Initiative's Accessible Rich Internet Applications (WAI-ARIA, atau cukup ARIA) bagus untuk menjembatani bidang-bidang masalah aksesibilitas yang tidak bisa ditangani dengan HTML asli. Hal ini bisa dilakukan dengan memungkinkan Anda menetapkan atribut yang memodifikasi cara elemen diterjemahkan ke dalam pohon aksesibilitas. Mari kita amati sebuah contoh.

Dalam cuplikan kode berikut, kita menggunakan item daftar sebagai kotak centang khusus. Kelas "checkbox" CSS memberi elemen karakteristik visual yang diperlukan.

```
<li tabindex="0" class="checkbox" checked>
  Receive promotional offers
</li>
```

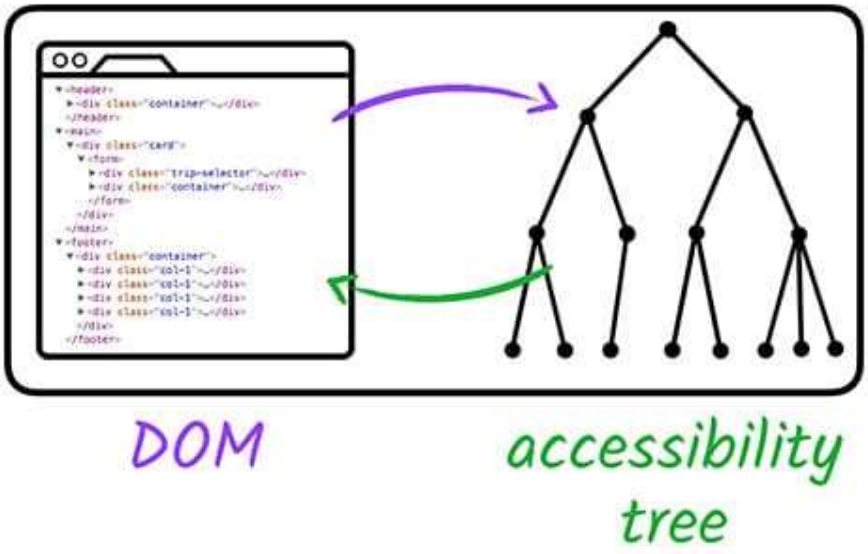
Walaupun cocok bagi pengguna yang berpenglihatan normal, pembaca layar tidak akan memberikan indikasi bahwa elemen dimaksudkan sebagai kotak centang, sehingga pengguna yang berpenglihatan lemah mungkin melewatkan elemen itu sama sekali.

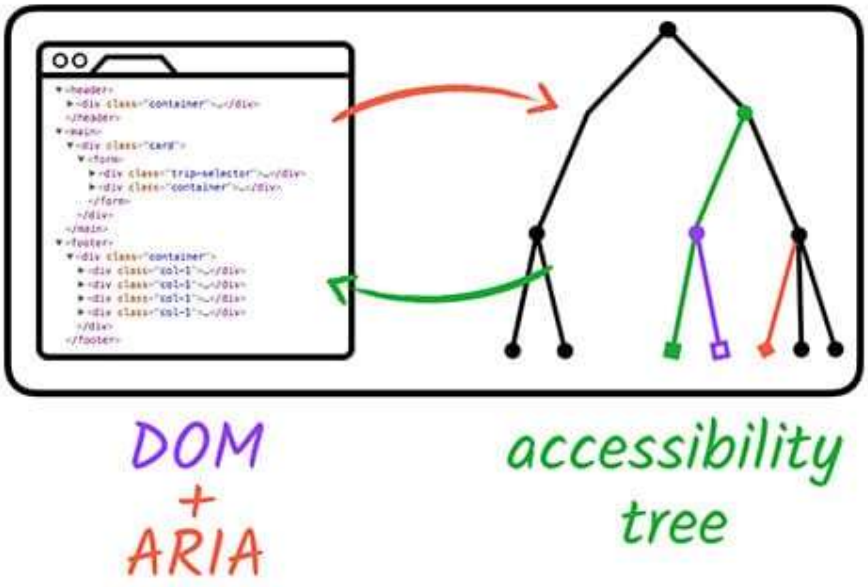
Akan tetapi, dengan menggunakan atribut ARIA, kita bisa memberi elemen tersebut informasi yang terlewat sehingga pembaca layar bisa menafsirkannya dengan benar. Di sini, kita telah menambahkan atribut role dan aria-checked untuk mengidentifikasi secara eksplisit elemen tersebut sebagai kotak centang dan menetapkan bahwa elemen itu telah dicentang secara default. Kini item daftar akan ditambahkan ke pohon aksesibilitas dan pembaca layar akan melaporkannya dengan benar sebagai kotak centang.

```
<li tabindex="0" class="checkbox" role="checkbox" checked aria-checked="true">
  Receive promotional offers
</li>
```

Note: Kita akan membahas daftar atribut ARIA dan kapan menggunakannya nanti.

ARIA bekerja dengan mengubah dan menambah pohon aksesibilitas DOM standar.





Walaupun ARIA memungkinkan kita secara halus (atau bahkan secara radikal) memodifikasi pohon aksesibilitas untuk elemen apa saja di laman, namun hanya elemen itu yang berubah. **ARIA tidak menambah perilaku inheren elemen**; ia tidak akan membuat elemen

dapat difokus atau memberinya event listener untuk keyboard. Itu tetap merupakan bagian dari tugas development kami.

Penting dipahami bahwa semantik default tidak perlu didefinisikan ulang. Apa pun penggunaannya, elemen `<input type="checkbox">` HTML standar tidak membutuhkan atribut ARIA tambahan `role="checkbox"` untuk diumumkan dengan benar.

Juga perlu diperhatikan bahwa elemen HTML tertentu memiliki batasan atas peran dan atribut ARIA apa saja yang bisa digunakan padanya. Misalnya, mungkin tidak ada peran/atribut tambahan yang diterapkan pada elemen `<input type="text">`.

Lihat ARIA di spesifikasi HTML untuk informasi selengkapnya.

Mari kita lihat kemampuan lain yang ditawarkan ARIA.

Apa yang bisa dilakukan ARIA?

Seperti yang Anda lihat pada contoh kotak centang, ARIA bisa memodifikasi semantik elemen yang ada atau menambahkan pada elemen bila tidak ada semantik asli. Pola semantik yang tidak ada sama sekali di HTML juga bisa dinyatakannya, seperti menu atau panel tab. Sering kali, ARIA memungkinkan kita membuat elemen bertipe widget yang tidak akan memungkinkan bila dengan HTML biasa.

Misalnya, ARIA bisa menambahkan label ekstra dan teks keterangan yang hanya diekspos kepada API teknologi pendukung.

```
<button aria-label="screen reader only label"></button>
```

ARIA bisa menyatakan hubungan semantik antar elemen yang memperluas hubungan induk/anak standar, misalnya bilah gulir khusus yang mengontrol region tertentu.

```
<div role="scrollbar" aria-controls="main"></div>
<div id="main">
. . .
</div>
```

ARIA bisa "menghidupkan" bagian laman, agar segera memberi tahu teknologi pendukung bila bagian itu berubah.

```
<div aria-live="true">
  <span>GOOG: $400</span>
</div>
```

Salah satu dari aspek inti sistem ARIA adalah kumpulan *peran*-nya. Peran dalam istilah aksesibilitas setara dengan indikator singkatan untuk pola UI tertentu. ARIA menyediakan kosakata pola yang bisa kita gunakan lewat atribut `role` pada suatu elemen HTML.

Bila kita menerapkan `role="checkbox"` dalam contoh sebelumnya, kita memberi tahu teknologi pendukung bahwa elemen harus mengikuti pola "kotak centang". Yakni, kami menjamin bahwa keadaannya akan dicentang (baik telah dicentang atau belum dicentang), dan bahwa keadaan itu dapat diubah-ubah menggunakan mouse atau tombol spasi, persis seperti elemen kotak centang HTML standar.

Kenyataannya, karena fitur interaksi keyboard begitu kentara dalam penggunaan pembaca layar, maka penting sekali memastikan bahwa, saat membuat widget khusus, atribut `role` selalu diterapkan di tempat yang sama seperti atribut `tabindex` ; ini memastikan bahwa kejadian keyboard pindah ke tempat yang tepat dan bahwa bila fokus jatuh pada suatu elemen, perannya akan dinyatakan dengan akurat.

Spesifikasi ARIA menjelaskan taksonomi nilai-nilai yang memungkinkan untuk atribut `role` dan atribut ARIA terkait yang boleh digunakan bersama-sama peran itu. Inilah sumber informasi definitif terbaik tentang cara kerja sama peran dan atribut ARIA dan bagaimana keduanya bisa digunakan dalam cara yang didukung oleh browser dan teknologi pendukung.

Akan tetapi, spesifikasinya sangat padat; tempat yang lebih mudah dicapai untuk memulai adalah dokumen ARIA Authoring Practices , yang mendalami berbagai praktik terbaik untuk menggunakan peran dan properti ARIA yang tersedia.

ARIA juga menawarkan peran landmark yang memperluas opsi yang tersedia di HTML5. Lihat spesifikasi Landmark Roles Design Patterns untuk informasi selengkapnya.

7.1 Hubungan dan Label ARIA

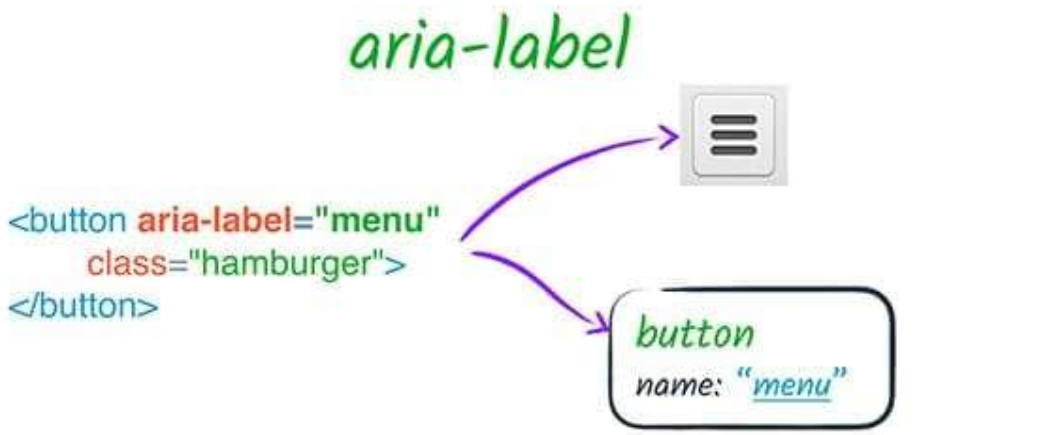
Label

ARIA menyediakan sejumlah mekanisme untuk menambahkan label dan keterangan ke elemen. Sebenarnya, ARIA adalah satu-satunya cara untuk menambahkan bantuan atau teks keterangan yang bisa diakses. Mari kita amati berbagai properti yang digunakan ARIA untuk membuat label yang bisa diakses.

aria-label

aria-label memungkinkan kita untuk menetapkan string yang akan digunakan sebagai label yang bisa diakses. Ini akan menggantikan mekanisme pelabelan asli lainnya, seperti elemen label — misalnya, jika button memiliki materi teks dan sebuah aria-label, maka hanya nilai aria-label yang akan digunakan.

Anda dapat menggunakan atribut aria-label bila memiliki semacam indikasi visual kegunaan elemen, misalnya tombol yang menggunakan grafik sebagai ganti teks, namun tetap perlu mengklarifikasi kegunaan itu untuk siapa saja yang tidak bisa mengakses indikasi visual, misalnya tombol yang hanya menggunakan gambar untuk menunjukkan kegunaannya.



aria-labelledby

aria-labelledby memungkinkan kita menetapkan ID elemen lain dalam DOM sebagai label elemen.

Ini mirip sekali dengan penggunaan elemen label, dengan beberapa perbedaan penting.

1. aria-labelledby dapat digunakan pada sembarang elemen, tidak cuma elemen yang bisa diberi label.

2. Walaupun elemen label merujuk pada sesuatu yang dilabelinya, hubungannya terbalik untuk aria-labelledby — sesuatu yang diberi label merujuk pada sesuatu yang melabelinya.

3. Hanya satu elemen label yang dapat dikaitkan dengan elemen yang bisa diberi label, namun aria-labelledby bisa menggunakan daftar IDREF untuk membuat label dari beberapa elemen sekaligus. Label akan digabung sesuai urutan IDREF yang diberikan.

4. Anda bisa menggunakan aria-labelledby untuk merujuk elemen yang disembunyikan dan bila tidak demikian tidak akan ada dalam pohon aksesibilitas. Misalnya, Anda bisa menambahkan span tersembunyi di sebelah elemen yang ingin Anda beri label, dan merujuknya dengan aria-labelledby.

5. Akan tetapi, karena ARIA hanya memengaruhi pohon aksesibilitas, aria-labelledby tidak memberi Anda perilaku pengeklikan label yang familier yang Anda dapat dari penggunaan elemen label.

Yang penting, aria-labelledby menggantikan **semua** sumber nama lainnya untuk elemen. Jadi, misalnya, jika sebuah elemen memiliki aria-labelledby dan aria-label, atau sebuah aria-labelledby dan label HTML asli, label aria-labelledby akan selalu didahulukan.

Hubungan

aria-labelledby adalah contoh sebuah *atribut hubungan*. Atribut hubungan membuat hubungan semantik antar elemen pada laman apa pun hubungan DOM-nya. Untuk aria-labelledby, hubungan itu adalah "elemen ini diberi label oleh elemen itu".

Spesifikasi ARIA mencantumkan delapan atribut hubungan. Enam di antaranya, aria-activedescendant, aria-controls, aria-describedby, aria-labelledby, dan aria-owns, mengambil referensi ke satu atau beberapa elemen untuk membuat tautan baru antar elemen pada laman.

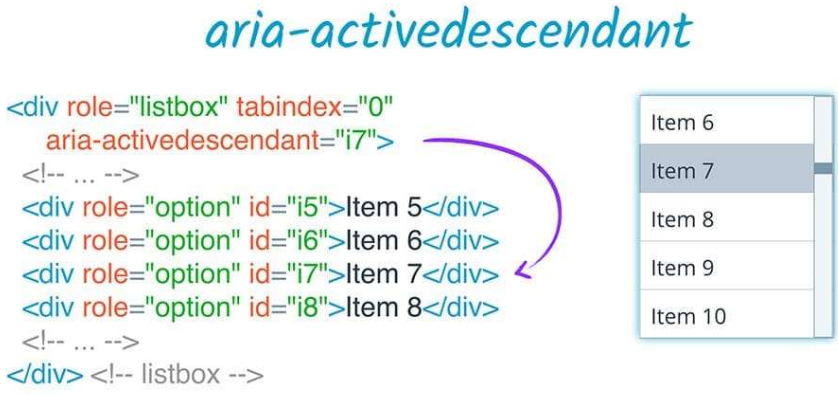
Perbedaan di setiap kasus adalah apa arti tautan itu dan bagaimana menyajikannya kepada pengguna.

aria-owns

aria-owns adalah salah satu hubungan ARIA yang paling banyak digunakan. Atribut ini memungkinkan kita memberi tahu teknologi pendukung bahwa elemen yang terpisah di DOM harus diperlakukan sebagai anak dari elemen saat ini, atau untuk menyusun ulang elemen anak yang ada ke dalam urutan yang berbeda. Misalnya, jika sebuah sub-menu munculan secara visual diposisikan di dekat menu induknya, namun tidak bisa berupa anak DOM dari induknya karena akan memengaruhi presentasi visual, Anda bisa menggunakan aria-owns untuk menyajikan sub-menu tersebut sebagai anak dari menu induk ke pembaca layar.

aria-activedescendant

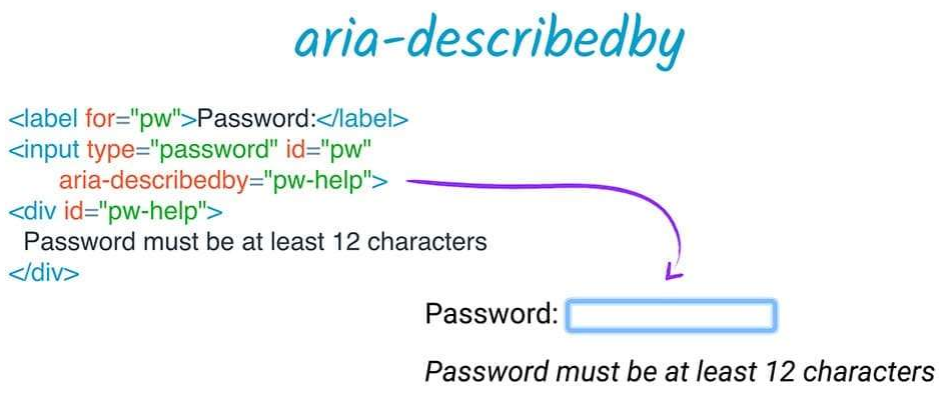
aria-activedescendant memainkan peranan terkait. Mirip dengan elemen aktif pada laman yang merupakan elemen berfokus, menyetel turunan aktif elemen memungkinkan kita untuk memberi tahu teknologi pendukung bahwa suatu elemen harus disajikan kepada pengguna sebagai elemen berfokus bila induknya memiliki fokus. Misalnya, dalam listbox, Anda mungkin ingin membiarkan fokus laman pada kontainer listbox, namun tetap memperbarui atribut aria-activedescendant-nya ke item daftar yang dipilih saat ini. Hal ini membuat item yang dipilih saat ini tampak oleh teknologi pendukung seakan item yang difokus.



aria-describedby

aria-describedby menyediakan keterangan yang bisa diakses dengan cara yang sama dengan label yang disediakan oleh aria-labelledby. Seperti halnya aria-labelledby, aria-describedby dapat mereferensikan elemen yang dalam keadaan lain tidak akan terlihat, baik yang disembunyikan dari DOM, atau yang disembunyikan dari pengguna teknologi pendukung. Ini merupakan teknik berguna bila ada beberapa teks penjelasan tambahan yang mungkin dibutuhkan pengguna, baik yang hanya berlaku pada pengguna teknologi pendukung atau pun semua pengguna.

Contoh umum adalah bidang masukan sandi dengan sejumlah teks deskriptif yang menjelaskan persyaratan sandi minimum. Tidak seperti label, keterangan ini mungkin atau mungkin tidak akan pernah disajikan kepada pengguna; mereka mungkin memiliki pilihan apakah akan mengaksesnya, atau keterangan tersebut mungkin ditampilkan setelah semua informasi lainnya, atau mungkin lebih dahulu ditempati oleh sesuatu yang lain. Misalnya, jika pengguna memasukkan informasi, masukan mereka akan dipantulkan kembali dan mungkin akan menginterupsi keterangan elemen. Sehingga, keterangan adalah cara yang bagus untuk mengomunikasikan informasi pelengkap, namun tidak esensial; ia tidak akan menghalangi informasi yang lebih penting seperti peran elemen.



aria-posinset & aria-setsize

Atribut hubungan selebihnya sedikit berbeda, dan bekerja bersama-sama. aria-posinset ("position in set") dan aria-setsize ("size of set") adalah tentang mendefinisikan hubungan antar elemen seinduk dalam suatu rangkaian, misalnya sebuah daftar.

Bila ukuran suatu rangkaian tidak bisa ditentukan melalui elemen yang ada dalam DOM — misalnya bila menggunakan lazy-rendering agar tidak semua daftar besar ada dalam DOM sekaligus — aria-setsize bisa menetapkan ukuran rangkaian sesungguhnya, dan aria-posinset bisa menetapkan posisi elemen dalam rangkaian tersebut. Misalnya dalam rangkaian yang bisa berisi 1000 elemen, anggaplah sebuah elemen tertentu memiliki aria-posinset sebesar 857 walaupun muncul lebih dahulu dalam DOM, kemudian menggunakan teknik HTML dinamis untuk memastikan pengguna bisa menyusuri daftar lengkap sesuai kebutuhan.

7.2 Menyembunyikan dan Memperbarui Materi

aria-hidden

Teknik penting lainnya dalam penyempurnaan pengalaman untuk pengguna teknologi pendukung antara lain memastikan bahwa hanya bagian laman yang relevan yang diekspos ke teknologi pendukung. Ada sejumlah cara untuk memastikan bagian DOM tidak diekspos ke API aksesibilitas.

Pertama, apa saja yang secara eksplisit disembunyikan dari DOM juga tidak akan disertakan di pohon aksesibilitas. Sehingga apa saja yang memiliki gaya CSS dengan atribut `visibility:hidden` atau `display: none` atau menggunakan `hidden` HTML5 juga akan disembunyikan dari pengguna teknologi pendukung.

Akan tetapi, elemen yang secara visual tidak dirender namun tidak secara eksplisit disembunyikan akan tetap disertakan dalam pohon aksesibilitas. Salah satu teknik umum adalah menyertakan "teks khusus pembaca layar" dalam elemen yang diposisikan di luar layar secara mutlak.

```
.sr-only {
  position: absolute;
  left: -10000px;
  width: 1px;
  height: 1px;
  overflow: hidden;
}
```

Selain itu, seperti yang telah kita lihat, bisa saja menyediakan teks khusus pembaca layar lewat atribut `aria-label`, `aria-labelledby`, atau `aria-describedby` yang mereferensikan elemen yang dalam keadaan lain disembunyikan.

Lihat artikel WebAIM ini di Teknik untuk menyembunyikan teks untuk informasi selengkapnya mengenai pembuatan teks "khusus pembaca layar".

Terakhir, ARIA menyediakan mekanisme untuk mengecualikan materi dari teknologi pendukung tidak secara visual disembunyikan, dengan menggunakan atribut aria-hidden. Menerapkan atribut ini ke elemen secara efektif akan membuangnya *dan semua turunannya* dari pohon aksesibilitas. Pengecualian satu-satunya adalah elemen yang dirujuk melalui atribut aria-labelledby atau aria-describedby.

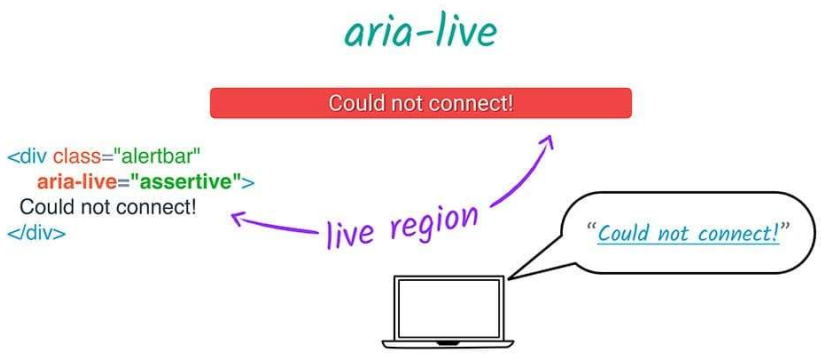
```
<div class="deck">
  <div class="slide" aria-hidden="true">
    Sales Targets
  </div>
  <div class="slide">
    Quarterly Sales
  </div>
  <div class="slide" aria-hidden="true">
    Action Items
  </div>
</div>
```

Misalnya, Anda dapat menggunakan aria-hidden jika membuat beberapa UI modal yang memblokir akses ke laman utama. Dalam hal ini, pengguna yang berpenglihatan normal mungkin akan melihat semacam overlay semi-transparan yang menunjukkan bahwa sebagian besar laman saat ini tidak bisa digunakan, namun pengguna pembaca layar mungkin tetap bisa menjelajahi bagian lain di laman tersebut. Dalam hal ini, seperti halnya membuat jebakan keyboard telah dijelaskan sebelumnya, Anda perlu memastikan bahwa bagian-bagian laman yang saat ini berada di luar cakupan berupa aria-hidden juga.

Karena kini Anda telah memahami dasar-dasar ARIA, cara memainkannya dengan semantik HTML asli, dan cara menggunakannya untuk melakukan pembedahan yang cukup besar pada pohon aksesibilitas serta pengubahan semantik elemen tunggal, mari kita amati cara menggunakannya untuk menyampaikan informasi yang sensitif terhadap waktu.

aria-live

aria-live memungkinkan developer menandai bagian laman sebagai "live" dalam artian bahwa pembaruan harus segera dikomunikasikan dengan pengguna, tanpa memperhatikan posisi laman, daripada cuma terjadi saat menjelajahi bagian laman tersebut. Bila elemen memiliki atribut aria-live, bagian laman yang berisi elemen tersebut dan turunannya disebut *live region*.



Misalnya, live region dapat berupa pesan status yang muncul sebagai hasil aksi pengguna. Jika pesan tersebut cukup penting untuk menarik perhatian pengguna yang berpenglihatan normal, berarti cukup penting untuk mengarahkan perhatian pengguna teknologi pendukung ke pesan tersebut dengan menyetel atribut aria-live-nya. Bandingkan div biasa ini

```
<div class="status">Your message has been sent.</div>
```

dengan pasangan "live"-nya.

```
<div class="status" aria-live="polite">Your message has been sent.</div>
```

aria-live memiliki tiga nilai yang diperbolehkan: polite, assertive, dan off.

- aria-live="polite" memberi tahu teknologi pendukung untuk memperingatkan pengguna pada perubahan ini bila telah menyelesaikan pekerjaan apa pun yang saat ini dilakukannya. Cocok sekali menggunakannya jika ada sesuatu yang penting namun tidak mendesak, dan inilah alasan untuk mayoritas penggunaan aria-live.
- aria-live="assertive" memberi tahu teknologi pendukung untuk menginterupsi apa pun yang sedang dilakukannya dan segera memperingatkan pengguna mengenai perubahan ini. Ini hanya untuk pemberitahuan penting dan mendesak, misalnya pesan status seperti "Ada kesalahan server dan perubahan Anda tidak disimpan; segarkan laman", atau pemberitahuan untuk bidang masukan sebagai akibat langsung dari aksi pengguna, misalnya tombol-tombol di stepper-widget.
- aria-live="off" memberi tahu teknologi pendukung untuk menangguhkan sementara interupsi aria-live.

Ada beberapa trik untuk memastikan live-region Anda bekerja dengan benar.

Pertama, region aria-live Anda harus telah disetel dalam pemuatan laman pertama. Ini bukan aturan mutlak, melainkan jika Anda mengalami kesulitan dengan region aria-live, maka hal ini bisa menjadi masalah.

Kedua, pembaca layar yang berbeda akan bereaksi berbeda terhadap tipe perubahan yang berbeda. Misalnya, bisa saja memicu peringatan pada beberapa pembaca layar dengan mengubah gaya hidden elemen turunan dari true ke false.

Atribut lain yang bisa digunakan bersama aria-live akan membantu Anda menyempurnakan apa yang dikomunikasikan kepada pengguna bila live-region berubah.

aria-atomic menunjukkan apakah keseluruhan region harus dianggap sebagai satu kesatuan saat mengomunikasikan pembaruan. Misalnya, jika widget tanggal yang terdiri dari hari, bulan, dan tahun memiliki aria-live=true dan aria-atomic=true, dan pengguna menggunakan kontrol stepper untuk mengubah nilai bulan saja, maka isi lengkap dari widget tanggal akan dibaca lagi. Nilai aria-atomic mungkin menjadi true atau false (default-nya).

aria-relevant menunjukkan tipe perubahan yang harus disajikan kepada pengguna. Ada beberapa opsi yang dapat digunakan secara terpisah atau sebagai daftar token.

- *additions*, berarti elemen yang telah ditambahkan ke live-region adalah signifikan. Misalnya, penambahan bentang atau span ke log pesan status yang ada menunjukkan bahwa bentang itu akan diumumkan kepada pengguna (dengan anggapan aria-atomic adalah false).
- *text*, berarti isi teks yang akan ditambahkan ke simpul turunan adalah relevan. Misalnya, memodifikasi properti textContent bidang teks khusus akan membacakan teks yang dimodifikasi kepada pengguna.
- *removals*, berarti pembuangan suatu teks atau simpul turunan akan disampaikan kepada pengguna.
- *all*, berarti semua perubahan adalah relevan. Akan tetapi, nilai default aria-relevant adalah additions text, yang berarti jika Anda tidak menetapkan aria-relevant, ia akan memberi tahu pengguna mengenai penambahan ke elemen, yang kemungkinan besar merupakan hal yang memang Anda inginkan.

Terakhir, aria-busy memungkinkan Anda memberi tahu teknologi pendukung agar untuk sementara mengabaikan perubahan pada elemen, misalnya bila ada sesuatu yang sedang dimuat. Setelah semua berada pada tempatnya, aria-busy harus disetel ke false untuk menormalkan operasi pembaca.

7.3 Gaya yang Dapat Diakses

Kita telah mendalami dua pilar aksesibilitas, fokus, dan semantik yang sangat penting. Sekarang mari kita bahas yang ketiga, penataan gaya. Ini adalah topik luas yang dapat kita masukkan ke dalam tiga bagian.

- Memastikan bahwa elemen-elemen ditata gayanya untuk mendukung upaya aksesibilitas dengan menambahkan gaya untuk fokus dan beragam keadaan ARIA.

- Penataan gaya UI kami untuk fleksibilitas sehingga UI dapat diperbesar atau diatur skalanya guna mengakomodasi pengguna yang mungkin memiliki masalah dengan teks berukuran kecil.
- Memilih warna dan kontras yang tepat guna menghindari penyampaian informasi dengan warna saja.

Penataan gaya fokus

Umumnya, setiap kali kita memfokuskan elemen, kita mengandalkan lingkaran fokus browser bawaan (properti outline CSS) untuk menata gaya elemen. Lingkaran fokus ini berguna karena, tanpanya, mustahil pengguna keyboard dapat memberi tahu elemen mana yang memiliki fokus. Daftar periksa WebAIM menunjukkan pentingnya hal ini, yang mengharuskan bahwa "Tampak nyata secara visual elemen laman mana yang memiliki fokus keyboard saat ini (yakni, saat melakukan navigasi pada laman tersebut, Anda bisa melihat tempat Anda berada)."

Akan tetapi, kadang-kadang lingkaran fokus bisa tampak terdistorsi atau mungkin hanya tidak pas dengan desain laman Anda. Beberapa developer membuang gaya ini sama sekali dengan menyetel outline elemen ke 0 atau none. Namun tanpa indikator fokus, bagaimana pengguna keyboard dapat mengetahui dengan item mana ia berinteraksi?

Caution: Jangan menyetel outline ke 0 atau tidak ada tanpa memberikan alternatif fokus!

Anda mungkin familier dengan penambahan kondisi mengambang ke kontrol dengan menggunakan *kelas semu* CSS :hover. Misalnya, Anda mungkin menggunakan :hover pada elemen tautan untuk mengubah warna atau latar belakangnya saat mouse berada di atasnya. Serupa dengan :hover, Anda bisa menggunakan kelas semu :focus untuk menargetkan elemen bila memiliki fokus.

```
/* At a minimum you can add a focus style that matches your hover style */
:hover, :focus {
  background: #c0ffee;
}
```

Solusi alternatif untuk masalah menghapus lingkaran fokus adalah memberi elemen Anda gaya mengambang dan fokus yang sama, yang mengatasi masalah "di mana fokusnya?" untuk pengguna keyboard. Seperti biasa, meningkatkan pengalaman aksesibilitas berarti meningkatkan pengalaman seseorang.

Modalitas Input

Untuk elemen bawaan seperti `button`, browser dapat mendeteksi apakah interaksi pengguna terjadi melalui mouse atau tekanan keyboard, dan biasanya hanya menampilkan lingkaran fokus untuk interaksi keyboard. Misalnya, bila Anda mengeklik `button` bawaan dengan mouse tidak ada lingkaran fokus, namun saat Anda menandainya dengan keyboard, lingkaran fokus akan muncul.

Logikanya di sini adalah bahwa pengguna mouse cenderung kurang memerlukan lingkaran fokus karena mereka tahu elemen apa yang mereka klik. Sayangnya saat ini tidak ada satu pun solusi lintas-browser yang menghasilkan perilaku yang sama ini. Sebagai hasilnya, jika Anda memberikan gaya `:focus` pada elemen apa pun, gaya tersebut akan ditampilkan *baik* bila pengguna mengeklik elemen atau memfokusnya dengan keyboard. Cobalah mengeklik tombol palsu ini dan perhatikan gaya `:focus` selalu diterapkan.

```
<style>
 fake-button {
  display: inline-block;
  padding: 10px;
  border: 1px solid black;
  cursor: pointer;
  user-select: none;
 }

 fake-button:focus {
  outline: none;
  background: pink;
 }
</style>
<fake-button tabindex="0">Click Me!</fake-button>
```

Ini bisa sedikit merepotkan, dan sering kali developer akan mengambil jalan untuk menggunakan JavaScript dengan kontrol khusus guna membedakan antara fokus mouse dan keyboard.

Di Firefox, CSS kelas semu `:-moz-focusring` memungkinkan Anda untuk menulis gaya fokus yang hanya berlaku jika elemen difokus melalui keyboard, sebuah fitur yang cukup bermanfaat. Sementara kelas semu saat ini hanya didukung di Firefox, ada upaya yang sedang dilakukan untuk menjadikannya standar.

Ada juga artikel sangat menarik yang ditulis oleh Alice Boxhall dan Brian Kardell bahwa mengeksplorasi topik modalitas dan berisi kode prototipe untuk membedakan antara masukan

mouse dan keyboard. Anda bisa menggunakan solusinya sekarang, kemudian menyertakan kelas semu lingkaran fokus nanti bila sudah didukung secara luas.

Kondisi penataan gaya dengan ARIA

Saat Anda membangun komponen, praktik yang umum adalah mencerminkan keadaannya, dan penampilannya, dengan menggunakan kelas CSS, yang dikontrol dengan JavaScript.

Misalnya, perhatikan tombol toggle yang masuk ke dalam kondisi visual "ditekan" saat diklik dan mempertahankan kondisi tersebut sampai tombol diklik kembali. Untuk menata gaya kondisi ini, JavaScript Anda mungkin menambahkan kelas `pressed` ke tombol tersebut. Dan, karena Anda menginginkan semantik yang baik pada semua kontrol, Anda juga perlu menyetel keadaan `aria-pressed` untuk tombol ke `true`.

Teknik yang berguna untuk digunakan di sini adalah benar-benar membuang kelas, dan cukup gunakan atribut ARIA untuk menata gaya elemen. Kini Anda bisa memperbarui pemilih CSS untuk keadaan tombol yang ditekan dari

```
.toggle.pressed { ... }
```

ke ini.

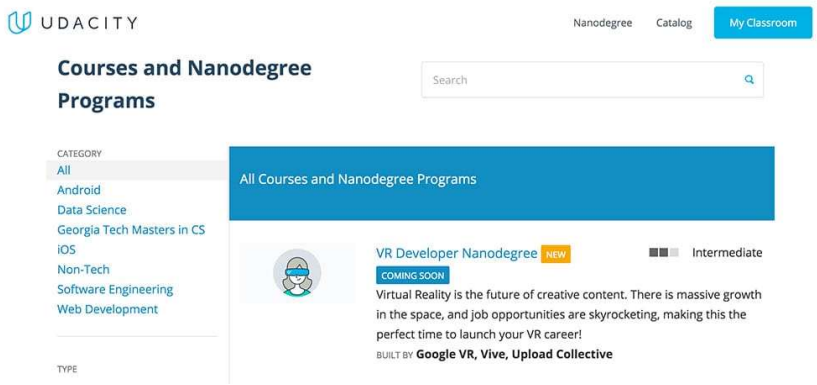
```
.toggle[aria-pressed="true"] { ... }
```

Hal ini menciptakan hubungan yang logis maupun semantik antara kondisi ARIA dan penampilan elemen, juga memangkas kode ekstra.

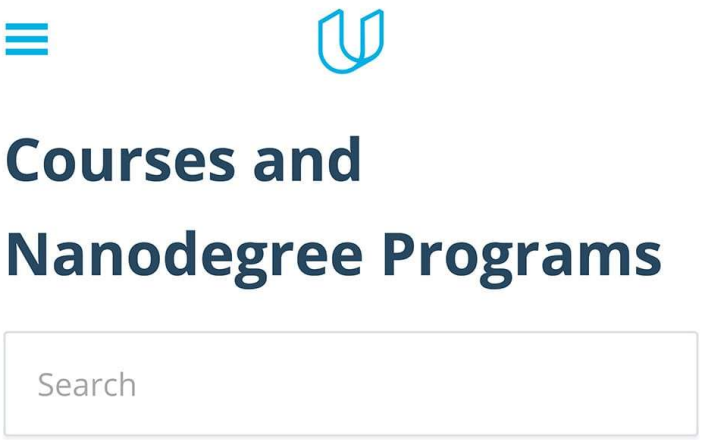
Desain responsif multi-perangkat

Kita mengetahui bahwa mendesain secara responsif guna menyediakan pengalaman multi-perangkat terbaik adalah ide bagus, namun desain responsif juga memiliki kelebihan dalam hal aksesibilitas.

Perhatikan situs seperti Udacity.com:



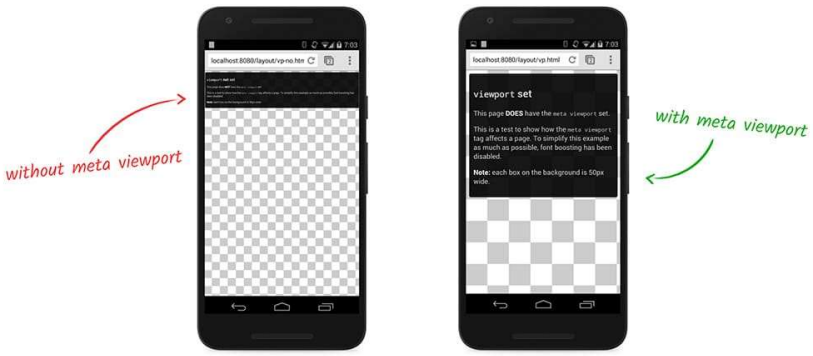
Pengguna dengan kemampuan penglihatan rendah yang memiliki kesulitan membaca tulisan kecil mungkin memperbesar laman, mungkin sebesar 400%. Karena situs didesain secara responsif, UI akan mengatur ulang sendiri untuk "tampilan yang terlihat yang lebih kecil" (sebenarnya untuk laman yang lebih besar), yang sangat bagus untuk pengguna desktop yang memerlukan perbesaran layar dan untuk pengguna pembaca layar seluler juga. Ini saling menguntungkan. Ini adalah laman yang sama yang diperbesar hingga 400%:



Kenyataannya, hanya dengan mendesain secara responsif, kita memenuhi aturan 1.4.4 dari daftar periksa WebAIM , yang menyatakan bahwa suatu laman "...harus dapat dibaca dan fungsional saat ukuran teksnya digandakan."

Memeriksa semua desain responsif adalah di luar cakupan panduan ini, tetapi ini adalah beberapa hal yang perlu diingat dan penting yang akan bermanfaat bagi pengalaman responsif Anda dan memberi pengguna Anda akses yang lebih baik ke materi.

- Pertama, pastikan Anda selalu menggunakan tag meta viewport yang tepat. <meta name="viewport" content="width=device-width, initial-scale=1.0"> Setelan width=device-width akan cocok dengan lebar layar dalam piksel yang tidak tergantung perangkat, dan setelan initial-scale=1 menetapkan hubungan 1:1 antara piksel CSS dan piksel yang tidak tergantung perangkat. Melakukan hal ini akan memerintahkan browser untuk mengepaskan materi dengan ukuran layar, sehingga pengguna tidak hanya melihat sekumpulan teks.



Caution: Saat menggunakan tag meta viewport, pastikan Anda tidak menyetel maximum-scale=1 atau menyetel user-scaleable=no. Biarkan pengguna memperbesar jika mereka perlu!

- Teknik lain yang perlu diingat adalah mendesain dengan grid responsif. Saat Anda melihat dengan situs Udacity, mendesain dengan grid berarti materi Anda akan mengubah posisi/geometri saat laman mengubah ukuran. Layout ini sering kali dibuat menggunakan unit-unit relatif seperti persen, ems, atau rems sebagai ganti nilai piksel hard-code. Keuntungan melakukannya dengan cara ini adalah teks dan materi dapat memperbesar dan memaksa ke bawah laman. Jadi urutan DOM dan urutan pembacaan tetap sama, bahkan jika ada perubahan layout karena perbesaran.

- Selain itu, perhatikan agar menggunakan unit-unit relatif seperti `em` atau `rem` untuk hal-hal seperti ukuran teks, sebagai ganti nilai piksel. Beberapa browser mendukung pengubahan ukuran teks hanya dalam preferensi pengguna, dan jika Anda menggunakan nilai piksel untuk teks, setelan ini tidak akan memengaruhi salinan Anda. Namun jika Anda telah menggunakan unit-unit relatif seluruhnya, maka salinan situs akan diperbarui untuk mencerminkan preferensi pengguna.

- Terakhir, saat desain Anda ditampilkan di perangkat seluler, Anda harus memastikan bahwa elemen interaktif seperti tombol atau tautan cukup besar, dan memiliki cukup ruang di sekitarnya, sehingga mudah untuk ditekan tanpa secara tidak sengaja tumpang-tindih dengan elemen yang lain. Hal ini bermanfaat bagi semua pengguna, namun khususnya berguna bagi siapa saja yang memiliki penurunan motorik.

Ukuran target sentuh minimum yang disarankan adalah sekitar 48 piksel yang tidak tergantung perangkat pada situs dengan tampilan yang terlihat untuk seluler telah disetel dengan benar. Misalnya, sementara suatu ikon mungkin hanya memiliki lebar dan tinggi 24 piksel, Anda bisa menggunakan pengisi tambahan untuk menambahkan ukuran target ketuk hingga 48 piksel. Area piksel 48x48 sesuai dengan sekitar 9 mm yaitu sekitar ukuran area isi jari seseorang.

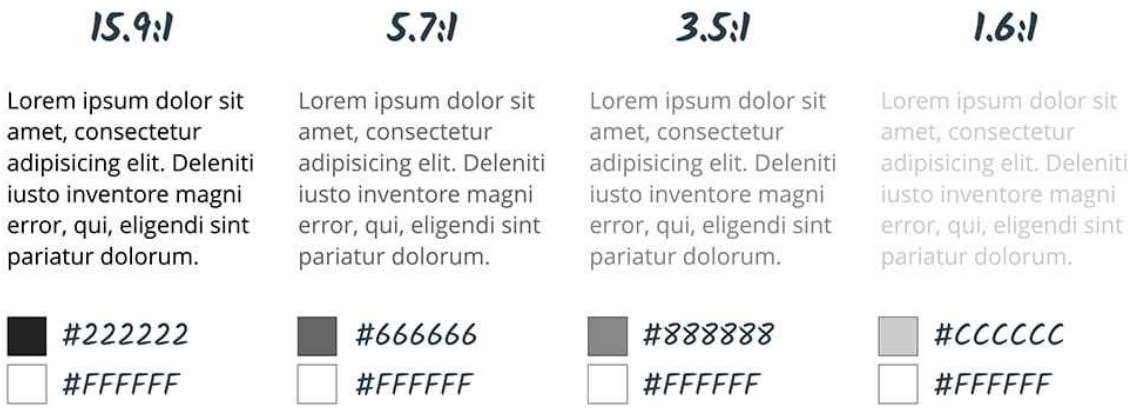
Target sentuh juga harus diberi ruang sekitar 8 piksel terpisah, baik horizontal maupun vertikal, sehingga jari pengguna yang menekan satu target ketuk tidak akan menyentuh target sentuh yang lain tanpa sengaja.

Warna dan kontras

Jika Anda memiliki penglihatan yang bagus, mudah untuk menganggap bahwa semua orang dapat melihat warna atau keterbacaan teks, dengan cara yang sama dengan Anda — namun tentu saja bukan itu masalahnya. Mari kita rangkum hal ini dengan melihat bagaimana kita secara efektif dapat menggunakan warna dan kontras untuk menciptakan desain menyenangkan yang dapat diakses bagi setiap orang.

Saat Anda dapat membayangkan, beberapa kombinasi warna yang mudah, bagi beberapa orang untuk dibaca ternyata sulit atau mustahil bagi orang lain. Ini biasanya bermuara pada *kontras warna*, hubungan antara *luminansi* warna latar belakang dan latar depan. Saat warna serupa, rasio kontras rendah; saat warna berbeda, rasio kontras pun tinggi.

Panduan WebAIM menyarankan rasio kontras AA (minimum) sebesar 4,5:1 untuk semua teks. Pengecualian akan dibuat untuk teks yang sangat besar (120-150% lebih besar dari teks isi default), yang rasionya dapat turun menjadi 3:1. Perhatikan perbedaan dalam rasio kontras yang ditampilkan di bawah ini.



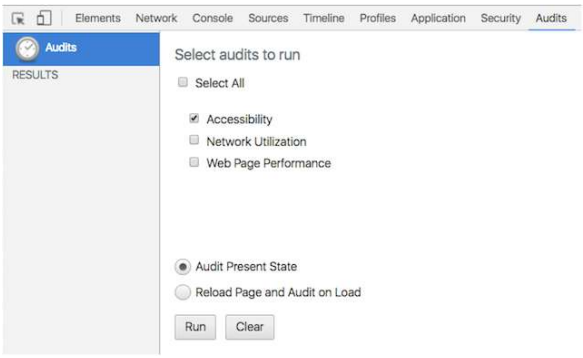
Rasio kontras 4,5:1 dipilih untuk level AA karena rasio ini mengganti hilangnya sensitivitas kontras yang biasanya dialami oleh pengguna yang kehilangan penglihatan setara dengan penglihatan sekitar 20/40. 20/40 umumnya dilaporkan sebagai ketajaman penglihatan biasa dari orang yang berusia sekitar 80 tahun. Untuk pengguna dengan gangguan penglihatan rendah atau defisiensi warna, kita dapat meningkatkan kontras hingga 7:1 untuk teks isi.

Anda bisa menggunakan ekstensi Accessibility DevTools untuk Chrome guna mengidentifikasi rasio kontras. Salah satu manfaat menggunakan Chrome Devtools adalah bahwa perangkat ini akan menyarankan alternatif AA dan AAA (disempurnakan) untuk warna Anda saat ini, dan Anda bisa mengeklik nilai untuk melakukan pratinjau di aplikasi.

Untuk menjalankan audit warna/kontras, ikuti langkah-langkah dasar ini.

1. Setelah memasang ekstensi, klik Audits
2. Hapus centang semuanya kecuali Accessibility
3. Klik Audit Present State

4. Perhatikan setiap peringatan kontras



WebAIM sendiri menyediakan pemeriksa kontras warna praktis yang bisa Anda gunakan untuk memeriksa kontras setiap pasangan warna.

Jangan menyampaikan informasi dengan warna saja

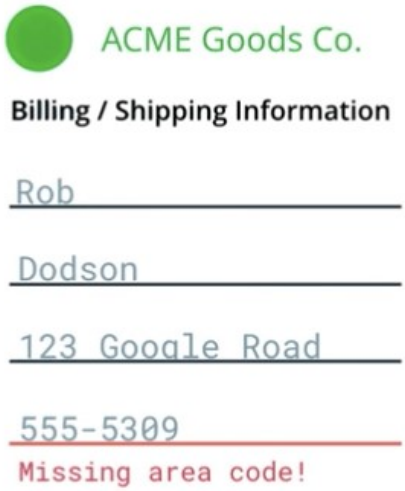
Ada sekitar 320 juta pengguna dengan defisiensi kemampuan melihat warna. Sekitar 1 dari 12 pria dan 1 dari 200 wanita memiliki beberapa bentuk "buta warna"; yang berarti sekitar 1/20 atau 5% pengguna Anda tidak akan merasakan pengalaman situs Anda sebagaimana yang Anda inginkan. Saat kita mengandalkan warna untuk menyampaikan informasi, kita menekan angka tersebut ke level yang tidak dapat diterima.

Note: Istilah "buta warna" sering kali digunakan untuk menjelaskan kondisi visual pada orang yang memiliki kesulitan dalam membedakan warna, namun sebenarnya hanya sedikit sekali orang yang benar-benar buta warna. Sebagian besar orang dengan defisiensi warna dapat melihat beberapa atau sebagian besar warna, namun memiliki kesulitan membedakan antara warna-warna tertentu seperti merah dengan hijau (yang paling umum), coklat dengan jingga, dan biru dengan ungu.

Misalnya, dalam sebuah formulir masukan, nomor telepon mungkin digarisbawahi dengan warna merah untuk menunjukkan bahwa nomor tersebut tidak valid. Namun bagi pengguna yang memiliki defisiensi warna atau pembaca layar, informasi tersebut tidak disampaikan dengan baik, bahkan benar-benar tidak baik. Maka, Anda harus selalu berupaya untuk menyediakan beberapa cara bagi pengguna untuk mengakses informasi penting.

Daftar periksa WebAIM menyatakan di bagian 1.4.1 bahwa "warna tidak boleh digunakan sebagai satu-satunya metode untuk menyampaikan materi atau membedakan elemen visual." Daftar periksa tersebut juga menyebutkan bahwa "warna saja tidak boleh digunakan untuk membedakan tautan dari teks sekelilingnya" kecuali jika warna tersebut memenuhi persyaratan kontras tertentu. Sebagai gantinya, daftar periksa menyarankan penambahan indikator tambahan seperti setrip bawah (menggunakan properti `text-decoration` CSS) untuk menunjukkan kapan tautan aktif.

Cara yang mudah untuk memperbaiki contoh sebelumnya adalah menambahkan pesan tambahan ke bidang tersebut, dengan mengumumkan bahwa ini tidak valid dan alasannya.



Saat Anda membangun sebuah aplikasi, tetap perhatikan hal-hal ini dan berhati-hati dengan area yang Anda mungkin terlalu mengandalkan warna untuk menyampaikan informasi penting.

Jika Anda penasaran tentang bagaimana tampilan situs Anda bagi beberapa orang, atau jika Anda sangat mengandalkan penggunaan warna pada UI Anda, Anda bisa menggunakan ekstensi NoCoffee Chrome untuk menstimulasi beragam bentuk gangguan visual, termasuk berbagai jenis buta warna.

Mode kontras tinggi

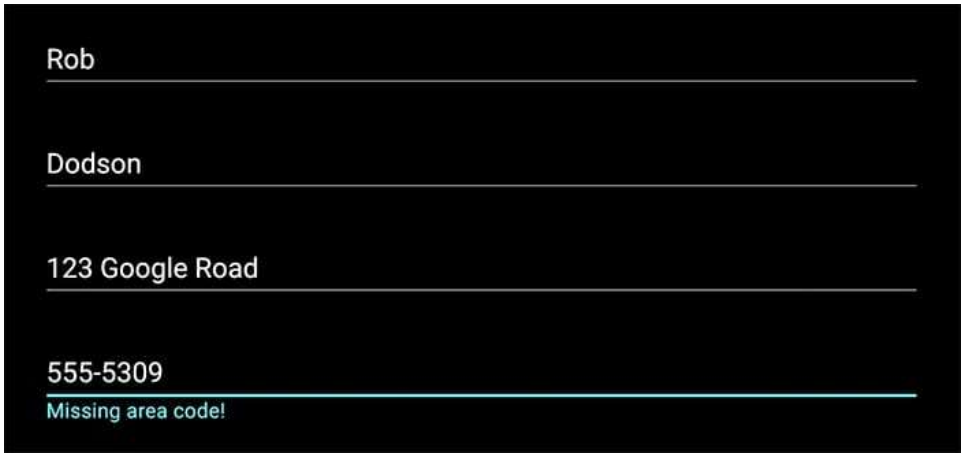
Mode kontras tinggi memungkinkan pengguna untuk membalik warna latar belakang dan latar depan, yang sering kali membantu teks tampak lebih baik. Bagi mereka dengan gangguan penglihatan rendah, mode kontras tinggi dapat memudahkan mereka untuk menavigasi materi pada laman. Ada beberapa cara untuk memperoleh penyiapan kontras tinggi di mesin Anda.

Sistem operasi seperti Mac OSX dan Windows menawarkan mode kontras tinggi yang dapat diaktifkan bagi siapa saja pada level sistem. Atau pengguna dapat memasang ekstensi, seperti ekstensi Chrome High Contrast untuk mengaktifkan kontras tinggi hanya di aplikasi khusus tersebut.

Latihan yang berguna dihidupkan pada setelan kontras tinggi dan memverifikasi bahwa semua UI di aplikasi Anda masih dapat dilihat dan dapat digunakan.

Misalnya, bilah navigasi dapat menggunakan warna latar belakang yang lembut untuk menunjukkan laman mana yang saat ini dipilih. Jika Anda menampilkannya dalam ekstensi kontras tinggi, yang perlahan seluruhnya menghilang, dan dengan cara ini pembaca mengerti laman mana yang aktif.

Demikian pula, jika Anda memperhatikan contoh dari pelajaran sebelumnya, garis bawah merah di bidang nomor ponsel yang tidak valid mungkin ditampilkan dalam warna biru-hijau yang sulit untuk dibedakan.



Jika Anda menemukan rasio kontras yang dicakup dalam pelajaran sebelumnya, Anda tidak akan mengalami masalah dalam hal mendukung mode kontras tinggi. Namun untuk menambah ketenangan, pertimbangkan untuk memasang ekstensi Chrome High Contrast dan periksa laman Anda untuk memeriksa apakah semuanya berfungsi, dan terlihat sebagaimana yang diharapkan.

7.4 Cara Melakukan Tinjauan Aksesibilitas

Menentukan apakah situs web atau aplikasi Anda dapat diakses sepertinya merupakan tugas yang berat. Jika Anda mendekati aksesibilitas untuk pertama kalinya, luasnya topik dapat membuat Anda bertanya-tanya di mana harus memulai - setelah semua, bekerja untuk mengakomodasi beragam kemampuan berarti ada beragam masalah yang perlu dipertimbangkan. Dalam posting ini, saya akan memecah masalah ini menjadi logis, langkah demi langkah proses untuk meninjau situs yang ada untuk aksesibilitas.

Mulai dengan keyboard

Untuk pengguna yang tidak bisa atau memilih untuk tidak menggunakan mouse, navigasi keyboard adalah sarana utama mereka untuk mencapai semua yang ada di layar. Audiens ini termasuk pengguna dengan gangguan motorik, seperti Repetitive Stress Injury (RSI) atau kelumpuhan, serta pengguna pembaca layar. Untuk pengalaman keyboard yang bagus, bertujuan untuk memiliki urutan tab yang logis dan gaya fokus yang mudah dilihat.

Key points

- Mulailah dengan membedah situs Anda. Urutan di mana elemen difokuskan harus bertujuan untuk mengikuti urutan DOM. Jika Anda tidak yakin elemen mana yang harus menerima fokus, lihat Dasar-Dasar Fokus untuk penyegaran. Aturan umum yang umum adalah bahwa kontrol apa pun yang dapat berinteraksi dengan pengguna atau memberikan input agar bertujuan untuk menjadi fokus dan menampilkan indikator fokus (mis., Cincin fokus). Merupakan praktik umum untuk menonaktifkan gaya fokus tanpa memberikan alternatif dengan menggunakan garis besar: tidak ada dalam CSS, tetapi ini merupakan anti-pola. Jika pengguna keyboard tidak dapat melihat apa yang difokuskan, mereka tidak memiliki cara untuk berinteraksi dengan halaman. Jika Anda perlu membedakan antara fokus mouse dan keyboard untuk penataan, pertimbangkan untuk menambahkan perpustakaan seperti input apa.

- Kontrol interaktif khusus harus bertujuan untuk menjadi fokus. Jika Anda menggunakan JavaScript untuk mengubah menjadi dropdown mewah, itu tidak akan secara otomatis dimasukkan ke dalam urutan tab. Untuk membuat kontrol khusus menjadi fokus, berikan tabindex = ”0”.

- Hindari kontrol dengan tabindex> 0. Kontrol ini akan melompati semua yang lain dalam urutan tab, terlepas dari posisi mereka di DOM. Ini dapat membingungkan bagi pengguna pembaca layar karena mereka cenderung menavigasi DOM secara linear.

- Konten non-interaktif (mis., Judul) harus menghindari fokus. Terkadang pengembang menambahkan tabindex ke pos karena mereka menganggapnya penting. Ini juga merupakan pola anti-karena membuat pengguna keyboard yang dapat melihat layar kurang efisien. Untuk pengguna pembaca layar, pembaca layar sudah akan mengumumkan judul-judul ini, jadi tidak perlu membuatnya fokus.

- Jika konten baru ditambahkan ke halaman, cobalah untuk memastikan bahwa fokus pengguna diarahkan ke konten itu sehingga mereka dapat mengambil tindakan terhadapnya. Lihat Mengelola Fokus di Level Halaman untuk contoh.

- Waspadalah terhadap fokus yang sepenuhnya terperangkap di titik mana pun. Perhatikan widget pelengkapan otomatis, di mana fokus keyboard mungkin macet. Fokus dapat secara sementara terperangkap dalam situasi tertentu, seperti menampilkan modal, saat Anda tidak ingin pengguna berinteraksi dengan bagian halaman lainnya - tetapi Anda harus bertujuan untuk menyediakan metode yang dapat diakses dengan

keyboard untuk keluar dari modal juga. Lihat panduan tentang Modals dan Traps Keyboard untuk contoh.

Hanya karena sesuatu itu bisa fokus, bukan berarti itu bisa digunakan

Jika Anda telah membangun kontrol khusus, maka bertujuan agar pengguna dapat mencapai semua fungsinya hanya menggunakan keyboard. Lihat Mengelola Komponen Fokus Dalam untuk teknik meningkatkan akses keyboard.

Jangan lupa konten di luar layar

Banyak situs memiliki konten di luar layar yang ada dalam DOM tetapi tidak terlihat, misalnya, tautan di dalam menu laci responsif atau tombol di dalam jendela modal yang belum ditampilkan. Meninggalkan elemen-elemen ini di DOM dapat menyebabkan pengalaman papan ketik yang membingungkan, terutama bagi pembaca layar yang akan mengumumkan konten di luar layar seolah-olah itu adalah bagian dari halaman. Lihat Menangani Konten di Layar Terbuka untuk kiat tentang cara menangani elemen-elemen ini.

Cobalah dengan pembaca layar

Meningkatkan dukungan keyboard umum memberikan beberapa dasar untuk langkah selanjutnya, yaitu memeriksa halaman untuk pelabelan dan semantik yang tepat dan segala penghalang untuk navigasi pembaca layar. Jika Anda tidak terbiasa dengan bagaimana markup semantik ditafsirkan oleh teknologi bantuan, lihat Pengantar Semantik untuk penyegaran.

Key points

- Periksa semua gambar untuk teks alt yang benar. Pengecualian untuk praktik ini adalah ketika gambar terutama untuk tujuan presentasi dan bukan bagian penting dari konten. Untuk menandakan bahwa suatu gambar harus dilewati oleh pembaca layar, setel nilai atribut alt ke string kosong, mis., Alt = "".

- Periksa semua kontrol untuk label. Untuk kontrol khusus, ini mungkin memerlukan penggunaan aria-label atau aria-labelled oleh. Lihat Label dan Hubungan ARIA untuk contoh.

- Periksa semua kontrol khusus untuk peran yang sesuai dan atribut ARIA yang diperlukan yang memberikan status mereka. Misalnya, kotak centang khusus akan memerlukan role = "checkbox" dan aria-checked = "true | false" untuk menyampaikan statusnya dengan benar. Lihat Pengantar ARIA untuk ikhtisar umum tentang bagaimana ARIA dapat menyediakan semantik yang hilang untuk kontrol khusus.

- Aliran informasi harus masuk akal. Karena pembaca layar menavigasi halaman dalam urutan DOM, jika Anda menggunakan CSS untuk memposisikan ulang elemen secara visual, mereka dapat diumumkan dalam urutan yang tidak masuk akal. Jika Anda memerlukan sesuatu untuk muncul lebih awal di halaman, cobalah untuk memindahkannya secara fisik sebelumnya di DOM.

- Bertujuan untuk mendukung navigasi pembaca layar ke semua konten di halaman. Hindari membiarkan bagian mana pun dari situs disembunyikan secara permanen atau diblokir dari akses pembaca layar.

- Jika konten harus disembunyikan dari pembaca layar, misalnya, jika itu layar atau hanya presentasi, pastikan bahwa konten diatur ke aria-hidden = "true". Lihatlah panduan tentang Menyembunyikan konten untuk penjelasan lebih lanjut.

Familiar dengan “even one screen reader goes a long way”

Meskipun mungkin sulit untuk mempelajari pembaca layar, sebenarnya cukup mudah untuk mengambilnya. Secara umum, sebagian besar pengembang dapat bertahan hanya dengan beberapa perintah kunci sederhana.

Jika Anda menggunakan Mac, periksa video ini tentang menggunakan VoiceOver, pembaca layar yang menyertai Mac OS. Jika Anda menggunakan PC untuk melihat video ini menggunakan NVDA, donasi yang didukung, pembaca layar sumber terbuka untuk Windows.

aria-hidden tidak mencegah fokus keyboard

Penting untuk dipahami bahwa ARIA hanya dapat memengaruhi semantik elemen; itu tidak berpengaruh pada perilaku elemen. Meskipun Anda dapat membuat elemen tersembunyi ke pembaca layar dengan aria-hidden = ”true”, itu tidak mengubah perilaku fokus untuk elemen itu. Untuk konten interaktif di luar layar, Anda sering harus menggabungkan aria-hidden = "true" dan tabindex = "- 1" untuk memastikan itu benar-benar dihapus dari aliran keyboard. Atribut inert yang diusulkan bertujuan untuk mempermudah ini dengan menggabungkan perilaku kedua atribut.

Elemen interaktif seperti tautan dan tombol harus menunjukkan tujuan dan statusnya

Memberikan petunjuk visual tentang apa yang akan dilakukan kontrol membantu orang mengoperasikan dan menavigasi situs Anda. Petunjuk ini disebut keterjangkauan. Menyediakan biaya memungkinkan orang menggunakan situs Anda di berbagai perangkat.

Key points

- Elemen interaktif, seperti tautan dan tombol, harus dapat dibedakan dari elemen non-interaktif. Sulit bagi pengguna untuk menavigasi situs atau aplikasi ketika mereka tidak tahu apakah suatu elemen dapat diklik. Ada banyak metode yang valid untuk mencapai tujuan ini. Salah satu praktik umum adalah tautan mendasar untuk membedakannya dari teks di sekitarnya.

- Mirip dengan persyaratan fokus, elemen-elemen interaktif seperti tautan dan tombol memerlukan keadaan mengambang untuk pengguna mouse sehingga mereka tahu jika mereka melayang di atas sesuatu yang dapat diklik. Namun, elemen interaktif masih harus dibedakan sendiri. Mengandalkan kondisi mengambang saja untuk menunjukkan elemen yang dapat diklik tidak membantu perangkat layar sentuh.

Manfaatkan headings dan landmarks

Heading dan elemen landmark mengilhami halaman Anda dengan struktur semantik, dan sangat meningkatkan efisiensi navigasi pengguna teknologi bantu. Banyak pengguna pembaca layar melaporkan bahwa, ketika mereka pertama kali mendarat di halaman yang tidak dikenal, mereka biasanya mencoba menavigasi dengan judul. Demikian pula, pembaca layar juga menawarkan kemampuan untuk melompat ke landmark penting seperti <main> dan <nav>. Untuk alasan ini, penting untuk mempertimbangkan bagaimana struktur halaman Anda dapat digunakan untuk memandu pengalaman pengguna.

Key points

- Manfaatkan hierarki h1-h6 dengan semestinya. Pikirkan pos sebagai alat untuk membuat garis besar halaman Anda. Jangan mengandalkan gaya tajuk bawaan; alih-alih, pertimbangkan semua judul seolah-olah ukurannya sama dan gunakan level semantik yang sesuai untuk konten primer, sekunder, dan tersier. Kemudian gunakan CSS untuk membuat judul sesuai dengan desain Anda.
- Gunakan elemen dan peran tengara sehingga pengguna dapat mem-bypass konten berulang. Banyak teknologi bantuan menyediakan pintasan untuk melompat ke bagian halaman tertentu, seperti yang ditentukan oleh elemen <main> atau <nav>. Elemen-elemen ini memiliki peran tengara yang tersirat. Anda juga dapat menggunakan atribut peran ARIA untuk secara eksplisit mendefinisikan kawasan pada halaman, mis., <Div role = ”search”>. Lihat panduan pada judul dan landmark untuk contoh lebih lanjut.
- Hindari role = ”aplikasi” kecuali Anda memiliki pengalaman sebelumnya bekerja dengannya. Peran tengara aplikasi akan memberi tahu teknologi bantu untuk menonaktifkan pintasannya dan melewati semua penekanan tombol ke halaman. Ini berarti bahwa tombol yang biasanya digunakan pembaca layar untuk bergerak di sekitar halaman tidak akan berfungsi lagi, dan Anda harus menerapkan sendiri semua keyboard yang menangani.

Tinjau headings dan landmarks dengan cepat dengan pembaca layar

Pembaca layar seperti VoiceOver dan NVDA menyediakan menu konteks untuk melompat-lompat ke wilayah penting pada halaman. Jika Anda melakukan pemeriksaan aksesibilitas, Anda dapat menggunakan menu ini untuk mendapatkan ikhtisar cepat laman dan menentukan apakah tingkat tajuk sesuai dan tengara mana yang digunakan. Untuk mempelajari lebih lanjut, lihat video instruksional ini tentang dasar-dasar VoiceOver dan NVDA.

Otomatisasi proses

Menguji aksesibilitas suatu situs secara manual bisa membosankan dan rawan kesalahan. Akhirnya, Anda akan ingin mengotomatiskan proses sebanyak mungkin. Ini dapat dilakukan melalui penggunaan ekstensi browser, dan suite tes aksesibilitas baris perintah.

Key points

- Does the page pass all the tests from either the aXe or WAVE browser extensions? These extensions are just two available options and can be a useful addition to any manual test process as they can quickly pick up on subtle issues like failing contrast ratios and missing ARIA attributes. If you prefer to do things from the command line, axe-cli provides the same functionality as the aXe browser extension, but can be easily run from your terminal.
- To avoid regressions, especially in a continuous integration environment, incorporate a library like axe-core into your automated test suite. axe-core is the same engine that powers the aXe chrome extension, but in an easy-to-run command line utility.
- If you're using a framework or library, does it provide its own accessibility tools? Some examples include protractor-accessibility-plugin for Angular, and a11ysuite for Polymer and Web Components. Take advantage of available tools whenever possible to avoid reinventing the wheel.

Jika Anda membuat Aplikasi Web Progresif, pertimbangkan untuk mencoba Lighthouse Jika Anda membuat Aplikasi Web Progresif, pertimbangkan untuk mencoba Lighthouse

Lighthouse adalah alat untuk membantu mengukur kinerja aplikasi web progresif Anda, tetapi juga menggunakan pustaka inti untuk memberi daya pada serangkaian uji aksesibilitas. Jika Anda sudah menggunakan Mercusuar, awasi kegagalan tes aksesibilitas dalam laporan Anda. Memperbaiki ini akan membantu meningkatkan pengalaman pengguna keseluruhan situs Anda.

Wrapping Up

Membuat ulasan aksesibilitas menjadi bagian rutin dari proses tim Anda, dan melakukan pemeriksaan ini lebih awal dan sering, dapat membantu meningkatkan pengalaman keseluruhan menggunakan situs Anda. Ingat, aksesibilitas yang baik sama dengan UX yang baik!

7.5 Aksesibilitas untuk tim

Membuat situs Anda lebih mudah diakses bisa menjadi tugas yang menakutkan. Jika Anda mendekati aksesibilitas untuk pertama kalinya, luasnya topik dapat membuat Anda bertanya-tanya harus mulai dari mana. Bagaimanapun, bekerja untuk mengakomodasi beragam kemampuan berarti ada berbagai isu yang perlu dipertimbangkan.

Ingat, aksesibilitas adalah upaya tim. Setiap orang memiliki peran untuk dimainkan. Artikel ini menguraikan kriteria untuk masing-masing disiplin ilmu utama (manajer proyek, perancang UX, dan pengembang) sehingga mereka dapat bekerja untuk memasukkan praktik terbaik aksesibilitas ke dalam proses mereka.

Project manager

Tujuan utama untuk setiap manajer proyek adalah mencoba memasukkan pekerjaan aksesibilitas dalam setiap tonggak sejarah; memastikan itu sama prioritasnya dengan topik lain seperti kinerja, dan pengalaman pengguna. Di bawah ini adalah beberapa item daftar periksa yang perlu diingat saat mengerjakan proses Anda.

- Jadikan pelatihan aksesibilitas tersedia bagi tim.
- Identifikasi perjalanan pengguna yang kritis di situs atau aplikasi.
- Cobalah untuk memasukkan daftar periksa aksesibilitas ke dalam proses tim.
- Jika memungkinkan, evaluasi situs atau aplikasi dengan studi pengguna.

Pelatihan aksesibilitas

Ada sejumlah sumber daya gratis yang bagus untuk belajar tentang aksesibilitas web. Menyisihkan waktu untuk tim Anda untuk mempelajari topik dapat membuatnya lebih mudah untuk memasukkan aksesibilitas di awal proses.

Beberapa sumber daya yang disediakan oleh Google meliputi:

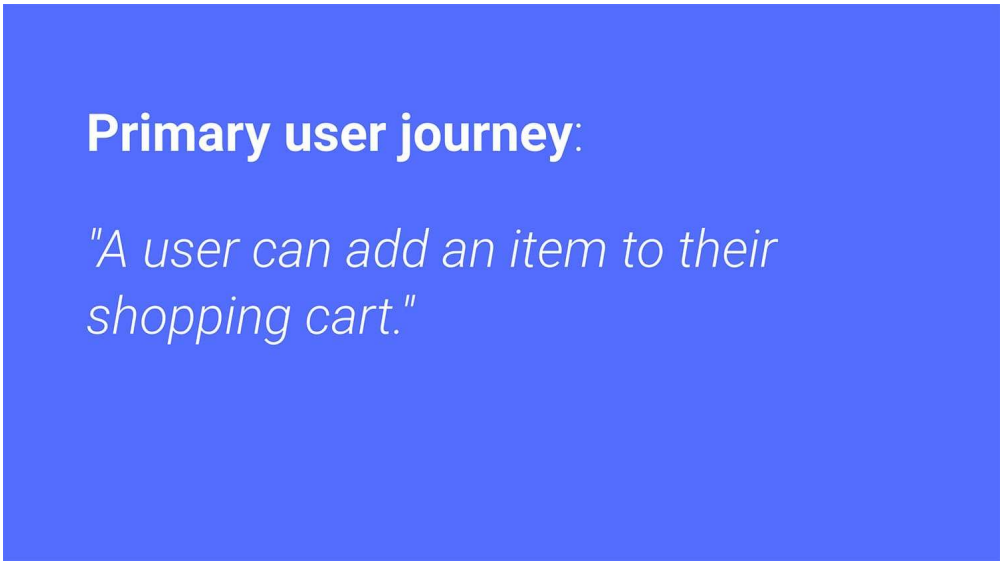
Aksesibilitas Web oleh Google - kursus pelatihan interaktif multi-minggu.

Fundamental Aksesibilitas - panduan aksesibilitas tertulis dan praktik terbaik.

Pedoman Bahan: Aksesibilitas - seperangkat praktik terbaik UX untuk desain inklusif.

Mengidentifikasi perjalanan pengguna yang kritis

Setiap aplikasi memiliki beberapa tindakan utama yang perlu dilakukan pengguna. Misalnya, jika Anda membuat aplikasi e-commerce, maka setiap pengguna harus dapat menambahkan item ke keranjang belanja mereka.



Beberapa tindakan mungkin memiliki kepentingan sekunder, dan mungkin hanya dilakukan sesekali. Misalnya, mengubah foto avatar Anda adalah fitur yang bagus, tetapi mungkin tidak penting untuk setiap pengalaman.

Mengidentifikasi tindakan utama dan sekunder dalam aplikasi Anda akan membantu Anda memprioritaskan pekerjaan aksesibilitas di depan. Kemudian, Anda dapat menggabungkan tindakan ini dengan daftar periksa aksesibilitas untuk melacak kemajuan Anda dan menghindari regresi.

Memasukkan daftar periksa aksesibilitas

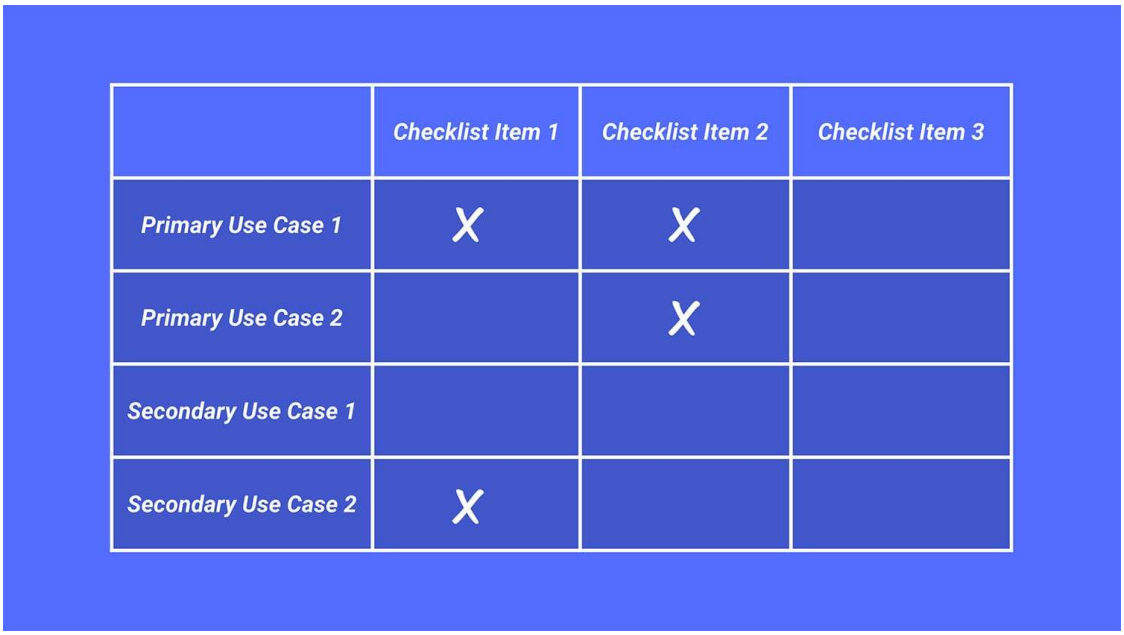
Topik aksesibilitas cukup luas, sehingga memiliki daftar periksa bidang-bidang penting untuk dipertimbangkan dapat membantu Anda memastikan Anda memenuhi semua basis Anda.

Ada sejumlah daftar aksesibilitas di luar sana, beberapa contoh industri termasuk:

Daftar Periksa WebAIM WCAG

Pedoman Aksesibilitas Vox

Dengan daftar periksa di tangan, Anda dapat melihat tindakan primer dan sekunder Anda untuk mulai menentukan apa pekerjaan yang masih perlu dilakukan. Anda bisa menjadi sangat taktis tentang proses ini dan bahkan membangun matriks tindakan primer dan sekunder dan menentukan untuk setiap langkah dalam proses tersebut, apakah ada bit aksesibilitas yang hilang.



Mengevaluasi dengan studi pengguna

Tidak ada yang mengalahkan saat duduk dengan pengguna sebenarnya dan mengamatinya saat mereka mencoba menggunakan aplikasi Anda. Jika Anda menyesuaikan aksesibilitas menjadi pengalaman yang ada, proses ini dapat membantu Anda dengan cepat mengidentifikasi area

yang perlu diperbaiki. Dan jika Anda memulai proyek baru, studi pengguna awal dapat membantu Anda menghindari menghabiskan terlalu banyak waktu mengembangkan fitur yang sulit digunakan.

Bertujuan untuk memasukkan umpan balik dari beragam populasi pengguna sebanyak mungkin. Pertimbangkan pengguna yang terutama bernavigasi dengan keyboard, atau mengandalkan teknologi bantu seperti pembaca layar atau kaca pembesar layar.

UX designer

Karena orang cenderung mendesain menggunakan bias mereka sendiri, jika Anda tidak memiliki cacat dan tidak memiliki kolega dengan cacat, Anda mungkin secara tidak sengaja merancang hanya untuk beberapa pengguna Anda. Saat Anda bekerja, tanyakan pada diri sendiri "apa saja tipe pengguna yang mungkin mengandalkan desain ini?" Berikut adalah beberapa teknik yang dapat Anda coba untuk membuat proses Anda lebih inklusif.

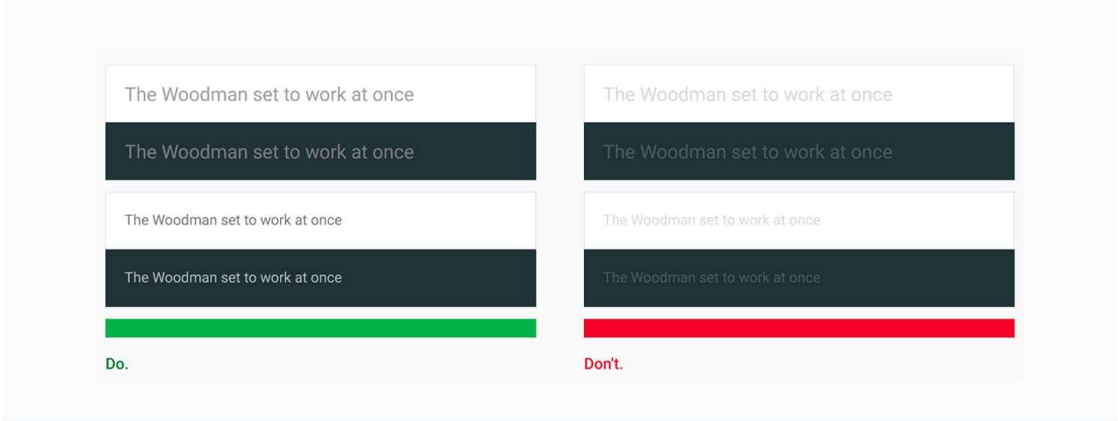
- Konten memiliki kontras warna yang cukup.
- Urutan tab ditentukan.
- Kontrol memiliki label yang dapat diakses.
- Ada beberapa cara untuk berinteraksi dengan UI.

Konten memiliki kontras warna yang baik

Tujuan utama sebagian besar situs adalah untuk menyampaikan beberapa informasi kepada pengguna, baik melalui teks tertulis atau gambar. Namun, jika konten ini kontras rendah, mungkin sulit bagi sebagian pengguna (terutama mereka yang memiliki gangguan penglihatan) untuk membaca. Ini dapat memengaruhi pengalaman pengguna mereka secara negatif. Untuk mengatasi masalah ini, bertujuan agar semua teks dan gambar memiliki kontras warna yang cukup.

Kontras diukur dengan membandingkan pencahayaan warna latar depan dan latar belakang. Untuk teks yang lebih kecil (apa pun di bawah 18pt atau 14pt tebal) disarankan rasio minimum 4,5: 1. Untuk teks yang lebih besar, rasio ini dapat disesuaikan menjadi 3: 1.

Pada gambar di bawah ini, teks di sisi kiri memenuhi minimum kontras ini, sedangkan teks di sisi kanan adalah kontras rendah.

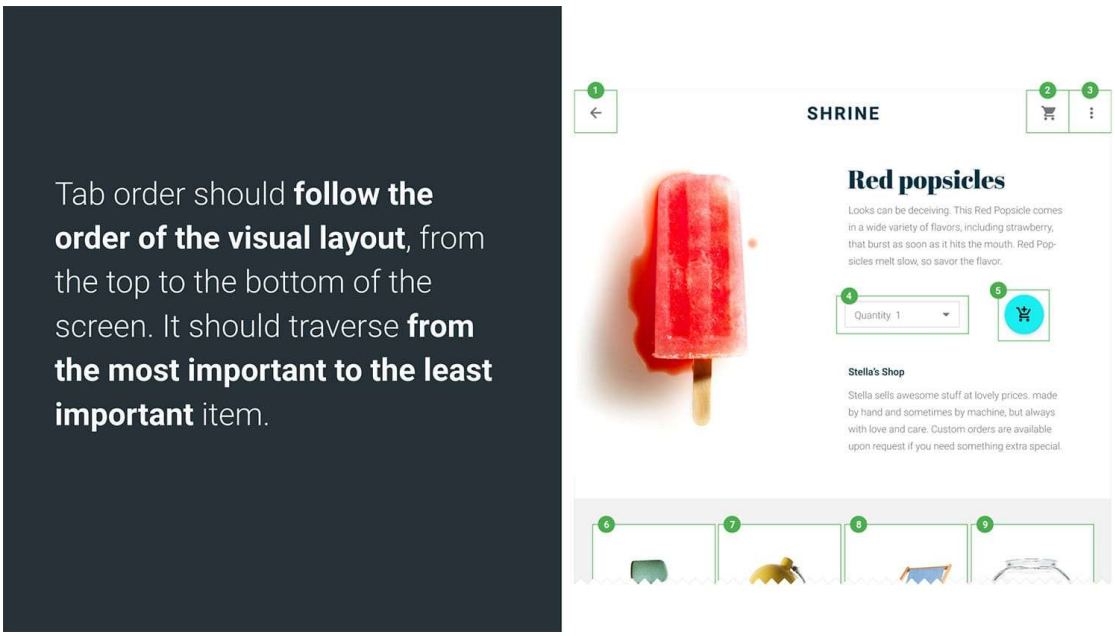


Ada sejumlah alat untuk mengukur kontras warna, seperti Alat Warna Bahan Google, aplikasi Rasio Kontras Lea Verou, dan kapak Deque.

Urutan tab ditentukan

Urutan tab adalah urutan di mana elemen menerima fokus saat pengguna menekan tombol tab. Untuk pengguna yang bernavigasi terutama dengan keyboard, tombol tab adalah sarana utama mereka untuk mencapai semua yang ada di layar. Anggap saja seperti kursor mouse mereka.

Idealnya urutan tab harus mengikuti urutan bacaan dan mengalir dari atas halaman ke bawah, dengan item yang lebih penting muncul lebih tinggi dalam urutan. Ini membuatnya lebih efisien bagi siapa saja yang menggunakan keyboard untuk dengan cepat mencapai item-item ini.



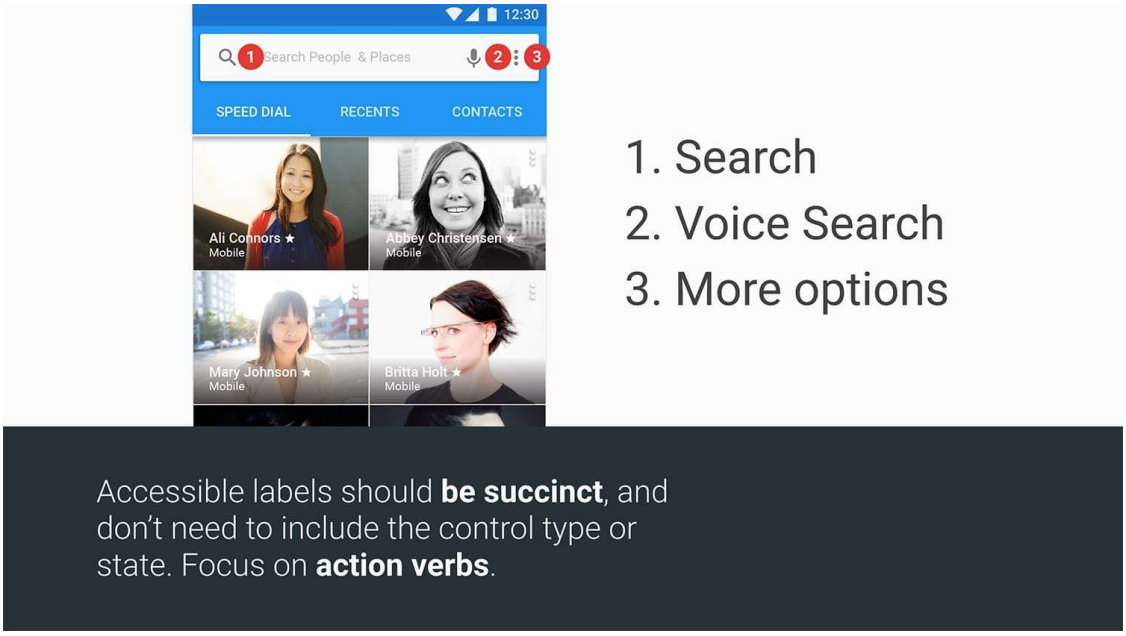
Antarmuka tiruan di atas diberi nomor untuk menampilkan urutan tab. Membuat tiruan seperti ini dapat membantu dengan mengidentifikasi urutan tab yang dimaksud. Ini kemudian dapat dibagikan dengan pengembang dan penguji QA untuk memastikan itu diterapkan dan diuji dengan benar.

Kontrol memiliki label yang dapat diakses

Untuk pengguna teknologi bantuan seperti pembaca layar, label memberikan informasi yang seharusnya hanya visual. Misalnya tombol pencarian yang hanya berupa ikon kaca pembesar dapat memiliki label "Cari" yang dapat diakses untuk membantu mengisi kekurangan visual yang hilang.

Berikut adalah beberapa saran sederhana untuk diikuti ketika mendesain label yang dapat diakses:

- Bersikap singkat - Membosankan mendengarkan deskripsi panjang bisa membosankan.
- Cobalah untuk tidak memasukkan tipe atau status kontrol - Jika kontrol dikodekan dengan benar maka pembaca layar akan mengumumkan ini secara otomatis.
- Fokus pada kata kerja tindakan - Gunakan "cari" bukan "kaca pembesar".



Anda mungkin mempertimbangkan untuk membuat tiruan dengan semua kontrol Anda berlabel. Ini dapat dibagikan dengan tim pengembangan Anda dan tim QA untuk implementasi dan pengujian.

Berbagai cara untuk berinteraksi dan memahami UI

Sangat mudah untuk mengasumsikan bahwa semua pengguna berinteraksi dengan halaman terutama menggunakan mouse. Saat mendesain, pertimbangkan bagaimana seseorang akan berinteraksi dengan kontrol menggunakan keyboard sebagai gantinya.

Rencanakan status fokus Anda! Ini berarti menentukan seperti apa sebuah kontrol ketika pengguna memfokuskannya dengan tab atau menekan tombol panah. Ini berguna untuk membuat negara-negara ini merencanakan lebih awal, daripada mencoba menyemir mereka ke dalam desain nanti.

Akhirnya, untuk setiap titik interaksi, Anda ingin memastikan bahwa pengguna memiliki banyak cara untuk memahami konten. Cobalah untuk tidak menggunakan warna sendirian untuk menyampaikan informasi, karena isyarat halus ini mungkin terlewatkan oleh pengguna dengan kekurangan penglihatan warna. Contoh klasik adalah bidang teks yang tidak valid. Alih-alih hanya menggarisbawahi merah untuk menandakan masalah, pertimbangkan juga menambahkan beberapa teks pembantu. Dengan begitu Anda mencakup lebih banyak pangkalan dan meningkatkan kemungkinan bahwa pengguna akan melihat masalah.

Developer

Peran pengembang adalah tempat manajemen fokus dan semantik bergabung untuk membentuk pengalaman pengguna yang kuat. Berikut adalah beberapa item yang dapat diingat pengembang saat mereka bekerja di situs atau aplikasi mereka:

- Urutan tab logis.
- Fokus dikelola dengan baik dan terlihat.
- Elemen interaktif memiliki dukungan keyboard.
- Peran dan atribut ARIA diterapkan sesuai kebutuhan.
- Elemen diberi label dengan benar.
- Pengujian otomatis.

Urutan tab logis

Elemen asli seperti input, tombol, dan pilih bisa ikut serta dalam urutan tab secara gratis dan secara otomatis dapat fokus dengan keyboard. Tetapi tidak semua elemen menerima perilaku yang sama ini! Secara khusus, elemen generik seperti div, dan span, tidak diikutkan dalam urutan tab. Ini berarti jika Anda menggunakan div untuk membuat kontrol interaktif, Anda harus melakukan pekerjaan tambahan agar keyboard dapat diakses.

Dua opsi adalah:

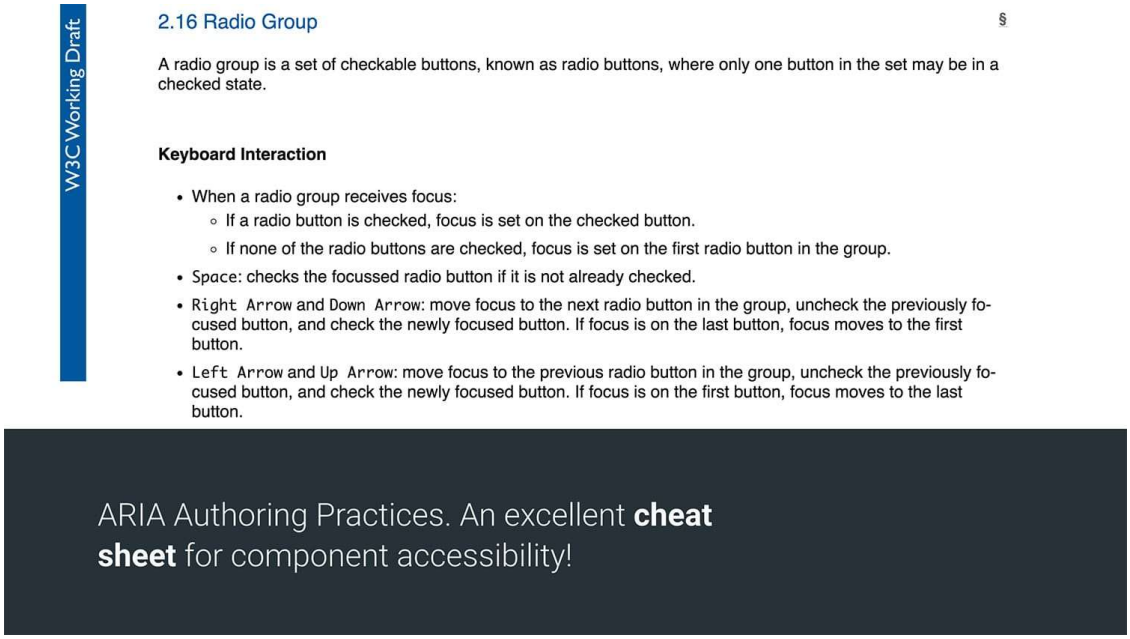
- Berikan kontrol tabindex = "0". Ini setidaknya akan membuatnya menjadi fokus, meskipun Anda mungkin perlu melakukan pekerjaan tambahan untuk menambahkan dukungan untuk penekanan tombol.
- Jika memungkinkan, pertimbangkan untuk menggunakan tombol, bukan div atau rentang untuk kontrol seperti tombol. Elemen tombol asli sangat mudah untuk ditata dan mendapat dukungan keyboard secara gratis.

Mengelola fokus

Saat Anda mengubah konten halaman, penting untuk mengarahkan perhatian pengguna dengan memindahkan fokus. Contoh klasik kapan teknik ini berguna adalah ketika membuka dialog modal. Jika pengguna yang mengandalkan keyboard menekan tombol untuk membuka dialog dan fokus mereka tidak dipindahkan ke elemen dialog, maka satu-satunya tindakan mereka adalah menabrak seluruh situs sampai mereka akhirnya menemukan kontrol baru. Dengan memindahkan fokus ke konten baru segera setelah muncul, Anda dapat meningkatkan efisiensi pengalaman pengguna ini.

Dukungan keyboard untuk elemen interaktif

Jika Anda membuat kontrol khusus seperti carousel atau dropdown, maka Anda perlu melakukan beberapa pekerjaan tambahan untuk menambahkan dukungan keyboard. Panduan Praktek Penulisan ARIA adalah sumber daya yang berguna yang mengidentifikasi berbagai pola UI dan jenis tindakan papan ketik yang diharapkan akan mereka dukung.



Untuk mempelajari lebih lanjut tentang menambahkan dukungan keyboard ke suatu elemen, lihat bagian tabindex keliling di dokumen Fundamentals Aksesibilitas Google.

Peran dan atribut ARIA diterapkan sesuai kebutuhan

Kontrol kustom tidak hanya membutuhkan dukungan keyboard yang tepat, tetapi juga memerlukan semantik yang baik. Bagaimanapun, div, secara semantik, hanyalah wadah pengelompokan generik. Jika Anda menggunakan div sebagai dasar untuk menu dropdown Anda, Anda harus bergantung pada ARIA untuk melapisi semantik tambahan sehingga jenis kontrol dapat disampaikan ke teknologi bantu. Di sini, sekali lagi, Panduan Praktik Penulisan ARIA dapat membantu dengan mengidentifikasi peran, status, dan properti yang harus Anda

gunakan. Sebagai bonus tambahan, banyak penjelasan dalam panduan ARIA juga disertai dengan kode sampel!

Labeling elements

Untuk memberi label pada input asli, Anda dapat menggunakan elemen <label> bawaan seperti yang dijelaskan pada MDN. Ini tidak hanya akan membantu Anda membuat kemampuan visual pada layar, tetapi juga memberi input nama yang dapat diakses di pohon aksesibilitas. Nama ini kemudian diambil oleh teknologi bantu (seperti pembaca layar) dan diumumkan kepada pengguna.

Sayangnya <label> tidak mendukung pemberian nama yang dapat diakses ke kontrol khusus (seperti yang dibuat menggunakan Elemen Kustom atau dari div dan bentang sederhana). Untuk kontrol semacam ini Anda harus menggunakan atribut aria-label dan aria-labelled oleh.

Pengujian otomatis

Menjadi malas bisa menjadi hal yang baik, terutama dalam hal pengujian. Sedapat mungkin berusaha mengotomatiskan tes aksesibilitas Anda sehingga Anda tidak perlu melakukan semuanya secara manual. Ada sejumlah alat pengujian industri hebat yang ada saat ini untuk memudahkan dan cepat memeriksa masalah aksesibilitas umum:

kapak, dibuat oleh sistem Deque, tersedia sebagai ekstensi Chrome dan modul Node (baik untuk lingkungan integrasi berkelanjutan). A11ycast singkat ini menjelaskan beberapa cara berbeda untuk memasukkan kapak ke dalam proses pengembangan Anda.

Lighthouse adalah proyek sumber terbuka Google untuk mengaudit kinerja Aplikasi Web Progresif Anda. Selain memeriksa apakah PWA Anda memiliki dukungan untuk hal-hal seperti Pekerja Layanan dan Manifes Aplikasi Web, Lighthouse juga akan menjalankan serangkaian tes praktik terbaik, termasuk tes untuk masalah aksesibilitas.

BAB VIII ANIMASI

Sasaran Pembelajaran

Mahasiswa mampu membuat antarmuka pengguna yang sangat responsif dan interaktif

Kemampuan mahasiswa yang menjadi prasyarat

Mahasiswa sudah menguasai materi Interaksi Manusia Komputer, menguasai tools untuk mendesain, Javascript dan CSS

Keterkaitan bahan pembelajaran dengan pokok bahasan lainnya

Materi ini sebagai dasar untuk mempelajari pokok bahasan lainnya

Manfaat atau pentingnya bahan pembelajaran ini

Membuat aplikasi web/situs dan mobile yang menarik

Animasi adalah bagian penting saat membuat aplikasi web dan situs yang menarik. Pengguna mengharapkan antarmuka pengguna yang sangat responsif dan interaktif. Namun, menganimasikan antarmuka Anda bukanlah hal yang mudah. Apa yang harus dianimasikan, kapan, dan apa jenis nuansa yang dimiliki animasi tersebut?

TL;DR

- Gunakan animasi sebagai cara untuk menambahkan jiwa ke proyek Anda.
- Animasi harus mendukung interaksi pengguna.
- Hati-hati dengan properti yang Anda animasikan; beberapa lebih berat dibanding yang lain.

Memilih hal yang tepat untuk dianimasikan

Animasi yang bagus menambahkan selapis kegembiraan dan rasa keterlibatan pengguna untuk proyek Anda. Anda bisa menganimasikan hampir semuanya, apakah itu lebar, ketinggian, posisi, warna, atau latar belakang, namun Anda harus menyadari potensi bottleneck kinerja dan bagaimana animasi memengaruhi personalitas aplikasi Anda. Animasi yang tersendat atau salah pilih bisa berdampak negatif terhadap pengalaman pengguna, sehingga animasi harus berkinerja baik dan tepat.

Gunakan animasi untuk mendukung interaksi

Jangan menganimasikan sesuatu hanya karena Anda bisa; itu hanya akan membuat pengguna jengkel dan mengganggunya. Namun, gunakan animasi yang ditempatkan secara tepat untuk memperkuat interaksi pengguna. Jika mereka mengetuk ikon menu, menggesek untuk menampilkan panel samping navigasi, atau mengetuk tombol, gunakan cahaya lembut atau efek

memantul untuk menerima interaksi. Hindari animasi yang mengganggu atau menghalangi aktivitas pengguna secara tidak perlu.

Hindari menganimasikan properti yang berat

Satu-satunya hal yang lebih buruk daripada animasi yang salah tempat adalah yang menyebabkan laman menjadi tersendat. Tipe animasi ini membuat pengguna merasa frustrasi dan tak senang, dan berharap Anda tidak menganimasikannya sama sekali.

Beberapa properti lebih sulit untuk diubah dibanding yang lain, dan hal inilah yang lebih mungkin membuat animasi tersendat. Jadi, misalnya, mengubah box-shadow sebuah elemen memerlukan operasi pewarnaan yang jauh lebih sulit dibandingkan mengubah, katakan, warna teksnya. Demikian pula, mengubah width sebuah elemen cenderung lebih sulit dibandingkan mengubah transform.

Anda bisa membaca selengkapnya mengenai perhitungan kinerja animasi dalam panduan Animasi dan Kinerja, namun jika Anda menginginkan TL;DR, tetaplah konsisten pada ubahan transform dan opacity, serta menggunakan will-change. Jika Anda ingin mengetahui secara pasti pekerjaan mana yang dipicu karena menganimasikan properti yang diberikan, lihat Pemicu CSS.

8.1 Animasi CSS Versus JavaScript

Ada dua cara utama untuk membuat animasi di web: dengan CSS dan dengan JavaScript. Mana yang Anda pilih sangat tergantung pada dependensi lain dari proyek, dan jenis efek yang ingin coba dicapai.

TL;DR

- Gunakan animasi CSS untuk transisi "satu-kali" yang lebih sederhana, seperti mengubah status elemen UI.
- Gunakan animasi JavaScript ketika Anda ingin menggunakan efek lanjutan seperti memantul, berhenti, jeda, memutar mundur atau memperlambat.
- Jika Anda memilih untuk membuat animasi dengan JavaScript, gunakan Web Animations API atau kerangka kerja modern yang sesuai bagi Anda.

Kebanyakan animasi dasar bisa dibuat dengan CSS atau JavaScript, namun besaran tenaga dan waktu bisa berbeda (lihat juga Kinerja CSS vs JavaScript). Masing-masing memiliki kelebihan dan kekurangan, tetapi ini bisa dijadikan panduan yang bagus:

- **Gunakan CSS ketika Anda memiliki status mandiri yang lebih kecil untuk elemen UI.** Transisi dan animasi CSS ideal untuk menampilkan menu navigasi dari samping, atau menampilkan keterangan alat. Anda mungkin akhirnya menggunakan JavaScript untuk mengontrol status, namun animasinya akan ada di CSS.

- **Gunakan JavaScript ketika Anda membutuhkan kontrol yang signifikan atas animasi Anda.** Web Animations API adalah pendekatan berbasis standar, saat ini tersedia di Chrome dan Opera. Ini memberikan objek nyata, ideal untuk aplikasi berorientasi objek yang kompleks. JavaScript juga berguna ketika Anda perlu berhenti, jeda, melambat atau membalikkan.
- **Gunakan `requestAnimationFrame` secara langsung ketika Anda ingin mengatur seluruh kejadian dengan tangan.** Ini adalah pendekatan JavaScript lanjutan, namun bisa bermanfaat jika Anda membangun sebuah game atau menggambar pada kanvas HTML.

Atau, jika Anda sudah menggunakan kerangka kerja JavaScript yang dilengkapi fungsionalitas animasi, seperti melalui metode jQuery `.animate()` atau GreenSock TweenMax, maka Anda mungkin merasa lebih nyaman dengan tetap menggunakannya untuk animasi Anda.

Menganimasikan dengan CSS

Menganimasikan dengan CSS adalah cara paling sederhana untuk menggerakkan sesuatu di layar. Pendekatan ini digambarkan sebagai *deklaratif*, karena Anda menentukan apa yang akan terjadi.

Di bawah ini adalah beberapa CSS yang memindahkan elemen 100 px pada sumbu X dan Y. Hal ini dilakukan dengan menggunakan transisi CSS yang disetel 500 md. Ketika kelas `move` ditambahkan, nilai `transform` berubah dan transisi dimulai.

```
.box {
  -webkit-transform: translate(0, 0);
  -webkit-transition: -webkit-transform 500ms;

  transform: translate(0, 0);
  transition: transform 500ms;
}

.box.move {
  -webkit-transform: translate(100px, 100px);
  transform: translate(100px, 100px);
}
```

Cobalah

Selain durasi transisi, ada beberapa pilihan untuk *easing*, yaitu bagaimana animasi terasa. Untuk informasi selengkapnya tentang easing, lihat panduan Dasar-Dasar Easing.

Jika, seperti dalam cuplikan di atas, Anda membuat kelas CSS yang terpisah untuk mengelola animasi, maka Anda bisa menggunakan JavaScript untuk menghidupkan dan mematikan setiap animasi:

```
box.classList.add('move');
```

Melakukan hal ini memberikan keseimbangan yang baik bagi aplikasi Anda. Anda bisa fokus dalam mengelola status dengan JavaScript, dan hanya menyetel kelas yang sesuai pada elemen sasaran, mempercayakan browser untuk menangani animasi. Jika menggunakan cara ini, Anda bisa mendengarkan kejadian `transitionend` pada elemen, namun hanya jika Anda mampu melepaskan dukungan pada versi Internet Explorer lama; versi 10 adalah versi pertama yang mendukung kejadian ini. Semua browser lain telah lama mendukung kejadian ini.

JavaScript yang diperlukan untuk mendengarkan akhir transisi terlihat seperti ini:

```
var box = document.querySelector('.box');
box.addEventListener('transitionend', onTransitionEnd, false);

function onTransitionEnd() {
  // Handle the transition finishing.
}
```

Selain menggunakan transisi CSS, Anda juga bisa menggunakan animasi CSS, yang memberikan Anda kontrol lebih banyak terhadap keyframe, durasi dan iterasi animasi individu.

Note: Jika Anda baru dalam dunia animasi, keyframe adalah istilah lama dari animasi yang digambar tangan. Animator akan membuat bingkai khusus untuk bagian dari tindakan, disebut bingkai kunci, yang akan merekam beberapa hal seperti bagian paling ekstrem dari beberapa gerakan, dan kemudian mereka akan menggambar semua bingkai secara tersendiri di antara keyframe. Kita menjalankan proses yang sama saat ini dengan animasi CSS, kita menginstruksikan browser tentang nilai yang harus dimiliki properti CSS pada titik tertentu, dan keyframe mengisi celahnya.

Misalnya, Anda bisa menganimasikan kotak dengan cara yang sama seperti transisi, namun menganimasikannya tanpa interaksi pengguna seperti klik, dan dengan pengulangan tak terbatas. Anda juga bisa mengubah beberapa properti sekaligus:

```
/**
 * This is a simplified version without
 * vendor prefixes. With them included
 * (which you will need), things get far
 * more verbose!
 */
.box {
  /* Choose the animation */
  animation-name: movingBox;

  /* The animation's duration */
```

```
  animation-duration: 1300ms;

  /* The number of times we want
     the animation to run */
  animation-iteration-count: infinite;

  /* Causes the animation to reverse
     on every odd iteration */
  animation-direction: alternate;
}

@keyframes movingBox {
  0% {
    transform: translate(0, 0);
    opacity: 0.3;
  }

  25% {
    opacity: 0.9;
  }

  50% {
    transform: translate(100px, 100px);
    opacity: 0.2;
  }

  100% {
    transform: translate(30px, 30px);
    opacity: 0.8;
  }
}
```

Cobalah

Dengan animasi CSS, Anda menentukan animasi secara independen dari elemen target, dan menggunakan properti nama animasi untuk memilih animasi yang diperlukan.

Animasi CSS tetap diberi awalan oleh vendor, dengan awalan `-webkit-` yang digunakan di Safari, Safari Mobile, dan Android. Chrome, Opera, Internet Explorer dan Firefox semua diluncurkan tanpa awalan. Banyak alat yang bisa membantu Anda membuat versi awalan dari CSS yang Anda butuhkan, sehingga Anda bisa menulis versi tanpa awalan dalam file sumber Anda.

Menganimasikan dengan JavaScript dan Web Animations API

Membuat animasi dengan JavaScript, jika dibandingkan, lebih kompleks daripada menulis transisi atau animasi CSS, tapi memberikan lebih banyak kendali kepada developer. Anda bisa menggunakan Web Animations API untuk menganimasikan properti CSS tertentu atau membangun efek objek yang bisa disusun.

Animasi JavaScript *sangat penting*, ketika Anda menulisnya secara inline sebagai bagian dari kode Anda. Anda juga bisa membungkusnya dalam objek lain. Berikut adalah JavaScript yang harus Anda tulis untuk membuat ulang transisi CSS yang sudah dijelaskan sebelumnya:

```
var target = document.querySelector('.box');
var player = target.animate([
  {transform: 'translate(0)'},
  {transform: 'translate(100px, 100px)'}
], 500);
player.addEventListener('finish', function() {
  target.style.transform = 'translate(100px, 100px)';
});
```

Secara default, Web Animasi hanya memodifikasi presentasi dari elemen. Jika Anda ingin agar objek tetap berada di lokasi pindahnya, maka Anda harus mengubah gaya dasarnya ketika animasi selesai, seperti contoh kami.

Cobalah

Web Animations API adalah standar baru dari W3C. Didukung secara native di Chrome dan Opera, dan dalam proses development aktif untuk Firefox. Untuk browser modern lainnya, tersedia polyfill.

Dengan animasi JavaScript, Anda memiliki kontrol penuh dari gaya elemen ini dalam setiap langkahnya. Ini berarti Anda bisa memperlambat animasi, melakukan jeda, menghentikan, membalikkan, dan memanipulasi elemen animasi sesuai keinginan Anda. Hal ini sangat berguna jika membangun aplikasi berorientasi objek yang kompleks, karena Anda bisa membungkus perilaku dengan baik.

8.2 Dasar-Dasar Easing

Di dunia ini tidak ada yang bergerak secara linear dari satu titik ke titik lainnya. Dalam kenyataannya, sesuatu cenderung mengalami percepatan atau perlambatan ketika mereka bergerak. Otak kita sudah diprogram untuk mengharapkan jenis gerakan ini, sehingga ketika membuat animasi, Anda harus menggunakannya untuk keuntungan Anda. Gerakan alami membuat pengguna merasa lebih nyaman dengan aplikasi Anda, yang pada akhirnya memberikan pengalaman yang lebih baik secara keseluruhan.

TL;DR

- Easing membuat animasi Anda terasa lebih alami.
- Pilih animasi ease-out untuk elemen UI.
- Hindari animasi ease-in atau ease-in-out kecuali Anda bisa membuatnya berdurasi pendek; mereka terasa lambat bagi pengguna akhir.

Dalam animasi klasik, istilah untuk gerak yang dimulai perlahan kemudian dipercepat adalah "slow in," dan gerakan yang dimulai dengan cepat kemudian berkurang kecepatannya adalah "slow out." Istilah yang paling umum digunakan di web untuk animasi ini adalah “ease in” dan “ease out”. Kadang-kadang keduanya digabungkan, yang disebut “ease in out.” Kemudian, easing adalah proses untuk membuat animasi lebih gampang untuk diucapkan.

Kata kunci easing

Transisi dan animasi CSS memperbolehkan Anda memilih jenis easing yang ingin Anda gunakan untuk animasi Anda. Anda bisa menggunakan kata kunci yang memengaruhi easing (atau kadang-kadang disebut timing) dari animasi tersebut. Anda juga bisa sepenuhnya menyesuaikan easing Anda, yang memberi Anda lebih banyak kebebasan untuk mengekspresikan kepribadian aplikasi.

Berikut adalah beberapa kata kunci yang bisa Anda gunakan dalam CSS:

- linear
- ease-in
- ease-out
- ease-in-out

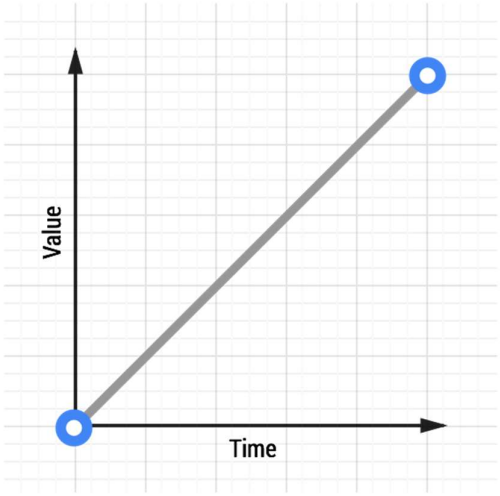
Sumber: Transisi CSS, W3C

Anda juga bisa menggunakan kata kunci steps, yang memungkinkan Anda untuk membuat transisi yang memiliki langkah-langkah terpisah, namun kata kunci yang tercantum di atas adalah yang paling berguna untuk membuat animasi yang terasa alami.

Animasi linear

Animasi tanpa menggunakan easing disebut sebagai **linear**. Ketika waktu berjalan, nilai juga mengalami pertambahan dalam jumlah yang setara. Dengan gerakan linear, sesuatu cenderung terasa kaku dan tidak wajar, dan hal ini bisa membuat janggal bagi pengguna. Pada intinya, Anda harus menghindari gerakan linear.

Apakah Anda sedang melakukan pengkodean animasi menggunakan CSS atau JavaScript, Anda akan mendapati bahwa selalu ada pilihan untuk gerakan linear. Grafik transisi linear terlihat sebagai berikut:

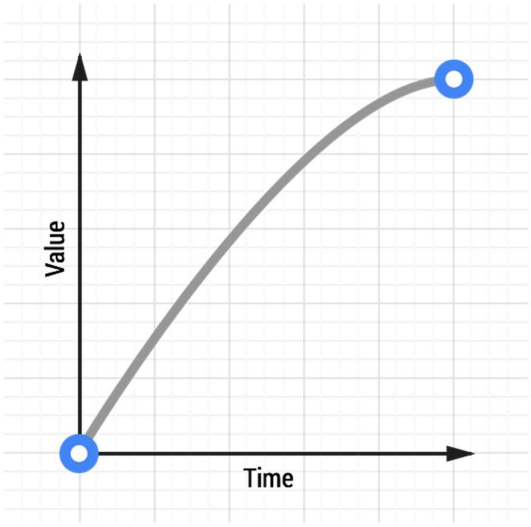


Lihat animasi linear

Untuk mendapatkan efek di atas dengan CSS, kodenya akan terlihat seperti berikut ini:

```
transition: transform 500ms linear;
```

Animasi ease-out



Easing out menyebabkan animasi dimulai lebih cepat dari yang linear, dan juga melambat di akhir.

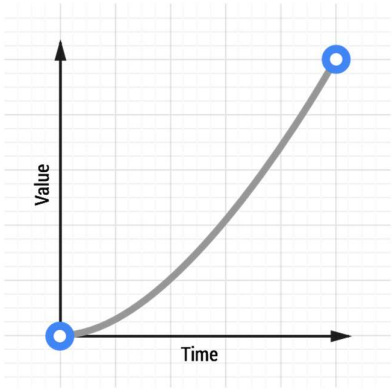
Easing out biasanya paling cocok digunakan untuk pekerjaan antarmuka pengguna, karena awal yang cepat dari animasi Anda memberikan kesan responsif, dan masih ada ruang untuk perlambatan alami di bagian akhir.

Lihat animasi ease-out

Ada banyak cara untuk mendapatkan efek ease out, tapi yang paling sederhana adalah kata kunci ease-out di CSS:

```
transition: transform 500ms ease-out;
```

Animasi ease-in



Animasi ease-in mulai dengan perlahan dan berakhir cepat, yang merupakan kebalikan dari animasi ease-out.

Jenis animasi ini seperti sebuah batu berat yang jatuh, batu itu awalnya jatuh perlahan-lahan lalu menghunjam tanah secara cepat dengan dentuman mematikan.

Namun, dari sudut pandang interaksi, ease-in terasa sedikit janggal karena akhir yang tiba-tiba; sesuatu yang bergerak di dunia nyata mengalami perlambatan, bukannya langsung berhenti tiba-tiba. Ease-in juga memiliki efek yang merugikan yaitu terasa lambat ketika dimulai, yang berdampak negatif terhadap persepsi daya respons dalam situs atau aplikasi Anda.

Lihat animasi ease-in

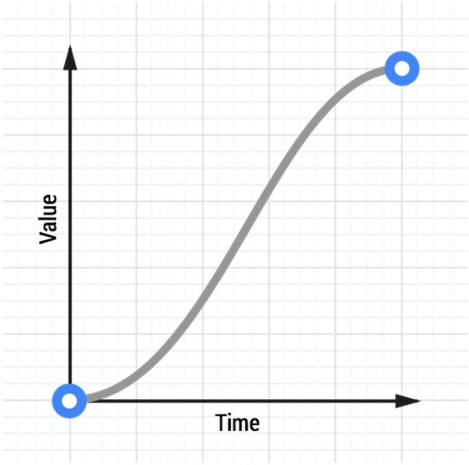
Untuk menggunakan animasi ease-in, sama seperti animasi ease-out dan linear, Anda bisa menggunakan kata kuncinya:

```
transition: transform 500ms ease-in;
```

Animasi ease-in-out

Easing-in dan easing-out mirip dengan mobil yang berakselerasi lalu melambat, dan jika digunakan dengan bijak, bisa memberikan efek yang lebih dramatis dari sekadar easing-out.

Jangan menggunakan durasi animasi yang terlalu lama, karena awal animasi ease-in agak lambat. Sesuatu di kisaran 300-500 md biasanya cocok, namun jumlah yang tepat sangat bergantung pada nuansa proyek Anda. Jadi, karena awal yang lambat, cepat di tengah, dan lambat di akhir, akan ada peningkatan kontras di animasi, yang bisa cukup memuaskan bagi pengguna.



Lihat animasi ease-in-out

Untuk mendapatkan animasi ease-in-out, Anda bisa menggunakan kata kunci CSS `ease-in-out`:

```
transition: transform 500ms ease-in-out;
```

8.3 Easing Khusus

Terkadang Anda tidak ingin menggunakan kata kunci easing yang disertakan dengan CSS, atau Anda akan menggunakan Animasi Web atau kerangka kerja JavaScript. Dalam hal ini, Anda biasanya bisa menentukan kurva (atau persamaan), dan ini memberikan Anda banyak kontrol terhadap nuansa animasi proyek.

TL;DR

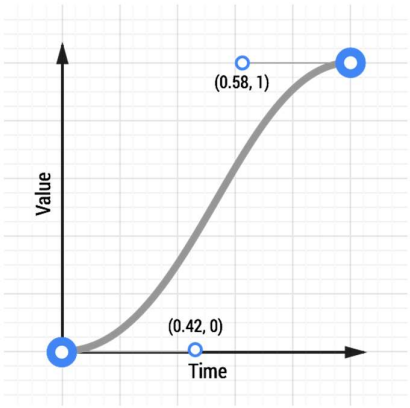
- Easing khusus memungkinkan Anda untuk memberikan lebih banyak kepribadian dalam proyek Anda.
- Anda bisa membuat kurva cubic Bézier yang menyerupai kurva animasi default (ease-out, ease-in, dll.) tapi dengan penekanan pada tempat yang berbeda.
- Gunakan JavaScript ketika Anda membutuhkan lebih banyak kontrol atas pengaturan waktu dan perilaku animasi, misalnya, animasi elastis atau memantul.

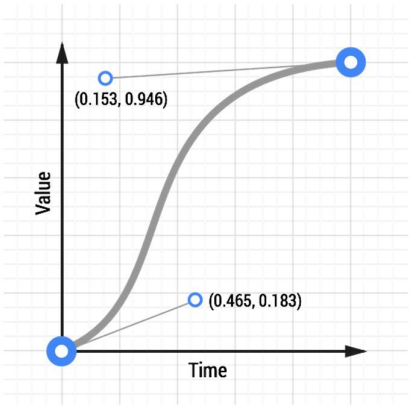
Jika membuat animasi dengan CSS, Anda akan mendapati bahwa Anda bisa menentukan kurva cubic Bézier untuk menetapkan waktunya. Faktanya, kata kunci ease, ease-in, ease-out dan linear memetakan ke kurva Bézier yang sudah ditetapkan, yang dijelaskan terperinci dalam spesifikasi transisi CSS dan spesifikasi Animasi Web.

Kurva Bézier ini mengambil empat nilai, atau dua pasang angka, dengan setiap pasangan menggambarkan koordinat X dan Y dari titik kontrol kurva cubic Bézier. Titik awal dari kurva Bézier memiliki koordinat (0, 0) dan titik akhir koordinat adalah (1, 1); Anda bisa menyetel nilai-nilai X dan Y dari dua titik kontrol. Nilai X untuk dua titik kontrol harus antara 0 dan 1, dan nilai Y setiap titik kontrol bisa melebihi batas [0, 1], meskipun spesifikasinya tidak menyebutkan seberapa banyak.

Mengubah setiap nilai X dan Y dari titik kontrol memberikan kurva yang sangat berbeda, dan karena itu memberikan nuansa sangat berbeda terhadap animasi Anda. Misalnya, jika titik kontrol pertama ada di daerah kanan bawah, maka animasinya akan dimulai dengan lambat. Jika ada di sudut kiri atas, animasi akan dimulai dengan cepat. Sebaliknya, jika titik kontrol kedua ada di daerah kanan bawah grid, animasi akan cepat di bagian akhir; jika di kiri atas, animasi akan lambat di bagian akhir.

Sebagai perbandingan, di sini ada dua kurva: kurva ease-in-out biasa dan kurva khusus:





Melihat animasi dengan easing khusus

CSS untuk kurva khusus adalah:

```
transition: transform 500ms cubic-bezier(0.465, 0.183, 0.153, 0.946);
```

Dua angka pertama adalah koordinat X dan Y dari titik kontrol pertama, dan dua angka kedua adalah koordinat X dan Y dari titik kontrol kedua.

Membuat kurva khusus sangat menyenangkan, dan memberikan Anda kontrol yang banyak atas nuansa animasi. Misalnya, pada kurva di atas, Anda bisa melihat bahwa kurva menyerupai kurva ease-in-out klasik, namun dengan ease-in dipersingkat, atau bagian "memulai," dan perlambatan panjang di bagian akhir.

Lakukan eksperimen dengan alat (bantu) kurva animasi dan lihat bagaimana kurva memengaruhi nuansa animasi.

Gunakan kerangka kerja JavaScript untuk lebih banyak kontrol

Terkadang Anda membutuhkan lebih banyak kontrol daripada yang disediakan oleh kurva cubic Bézier. Jika Anda ingin efek memantul elastis, Anda bisa mempertimbangkan menggunakan kerangka kerja JavaScript, karena ini adalah efek yang sulit dicapai dengan CSS atau Web Animations.

TweenMax

Salah satu kerangka kerja yang efektif adalah TweenMax dari GreenSock (atau TweenLite jika ingin membuat segalanya sangat ringan), karena Anda mendapatkan banyak kontrol dalam pustaka JavaScript yang ringan, dan basis kode yang sangat matang.

Lihat animasi ease elastis

Untuk menggunakan TweenMax, sertakan skrip berikut di laman Anda:

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/gsap/latest/TweenMax.min.js"></script>
```

Setelah skrip ditempatkan, Anda bisa memanggil TweenMax terhadap elemen Anda dan memberitahukan properti yang diinginkan, bersama tiap easing yang Anda inginkan. Ada banyak pilihan easing yang bisa Anda gunakan; kode di bawah menggunakan ease-out elastis:

```
var box = document.getElementById('my-box');
var animationDurationInSeconds = 1.5;

TweenMax.to(box, animationDurationInSeconds, {
  x: '100%',
  ease: 'Elastic.easeOut'
});
```

Dokumentasi TweenMax menyoroti semua pilihan yang Anda miliki, sehingga layak dibaca.

8.4 Menganimasikan Antar Tampilan

Sering kali, Anda ingin memindahkan pengguna di antara tampilan dalam aplikasi, apakah itu dari daftar untuk tampilan terperinci, atau menampilkan navigasi bilah sisi. Animasi antara tampilan-tampilan ini menjaga pengguna tetap terlibat dan menambahkan kesan lebih hidup untuk proyek Anda.

TL;DR

- Gunakan transisi untuk berpindah antar tampilan; hindari menggunakan `left`, `top`, atau properti lainnya yang memicu layout.
- Pastikan bahwa setiap animasi yang Anda gunakan cepat dan berdurasi pendek.
- Pertimbangkan bagaimana animasi dan layout Anda berubah ketika ukuran layar semakin besar; apa yang bekerja dengan baik pada layar yang lebih kecil, mungkin akan terlihat aneh ketika digunakan pada desktop.

Bagaimana transisi tampilan ini terlihat dan berperilaku akan bergantung pada tipe tampilan yang Anda hadapi. Misalnya, menganimasikan overlay modal di atas tampilan sebaiknya terlihat berbeda dibandingkan transisi antara tampilan daftar dan tampilan detail.

Berhasil: Cobalah untuk mempertahankan 60fps bagi semua animasi Anda. Dengan begitu, pengguna tidak akan melihat animasi yang tersendat yang mengganggu pengalaman mereka. Pastikan bahwa setiap elemen animasi memiliki `will-change` yang disetel untuk apa pun yang telah Anda rencanakan untuk diubah sebelum animasi dimulai. Untuk melihat transisi, kami sarankan agar Anda menggunakan `will-change: transform`.

Gunakan terjemahan untuk berpindah di antara tampilan

Untuk mempermudah, asumsikan bahwa ada dua tampilan: tampilan daftar dan tampilan detail. Saat pengguna mengetuk daftar item dalam tampilan daftar, tampilan detail bergeser ke dalam, dan tampilan daftar bergeser ke luar.

Untuk memperoleh efek ini, Anda membutuhkan kontainer bagi kedua tampilan tersebut dengan `overflow: hidden` yang telah disetel di situ. Dengan begitu dua tampilan tersebut bisa

berdampingan di dalam kontainer tanpa menampilkan bilah gulir horizontal, dan setiap tampilan bisa bergeser dari sisi-ke-sisi ketika diperlukan.

CSS untuk kontainer adalah:

```
.container {
  width: 100%;
  height: 100%;
  overflow: hidden;
  position: relative;
}
```

Posisi kontainer ditetapkan sebagai relative. Ini berarti bahwa setiap tampilan di dalam kontainer bisa diposisikan ke sudut kiri atas dan kemudian diubah dengan transformasi. Pendekatan ini lebih menguntungkan untuk kinerja daripada menggunakan properti left (karena hal itu memicu layout dan paint), dan biasanya lebih mudah untuk dirasionalisasi.

```
.view {
  width: 100%;
  height: 100%;
  position: absolute;
  left: 0;
  top: 0;

  /* let the browser know we plan to animate
     each view in and out */
  will-change: transform;
}
```

Menambahkan transition pada properti transform memberikan efek geser yang bagus. Untuk memberikan nuansa yang bagus, gunakan kurva cubic-bezier khusus, yang kita bahas dalam Panduan Easing Khusus.

```
.view {
  /* Prefixes are needed for Safari and other WebKit-based browsers */
  transition: -webkit-transform 0.3s cubic-bezier(0.465, 0.183, 0.153, 0.946);
  transition: transform 0.3s cubic-bezier(0.465, 0.183, 0.153, 0.946);
}
```

Tampilan di luar layar harus diterjemahkan ke kanan, sehingga tampilan detail harus dipindahkan:

```
.details-view {
  -webkit-transform: translateX(100%);
```

```
  transform: translateX(100%);
}
```

Sekarang sedikit JavaScript diperlukan untuk menangani kelas. Ini mengalihkan kelas-kelas yang sesuai pada tampilan.

```
var container = document.querySelector('.container');
var backButton = document.querySelector('.back-button');
var listItems = document.querySelectorAll('.list-item');

/**
 * Toggles the class on the container so that
 * we choose the correct view.
 */
function onViewChange(evt) {
  container.classList.toggle('view-change');
}

// When you click a list item, bring on the details view.
for (var i = 0; i < listItems.length; i++) {
  listItems[i].addEventListener('click', onViewChange, false);
}

// And switch it back again when you click the back button
backButton.addEventListener('click', onViewChange);
```

Akhirnya, kita tambahkan deklarasi CSS untuk kelas-kelas tersebut.

```
.view-change .list-view {
  -webkit-transform: translateX(-100%);
  transform: translateX(-100%);
}

.view-change .details-view {
  -webkit-transform: translateX(0);
  transform: translateX(0);
}
```

Cobalah

Anda bisa meluaskan ini untuk menutupi beberapa tampilan, namun konsep dasarnya harus tetap sama; setiap tampilan tak-terlihat harus di luar layar dan ditampilkan ketika diperlukan, dan tampilan aktif pada layar harus dialihkan keluar.

Perhatian: Membuat hierarki semacam ini dalam lintas-browser bisa sangat menantang. Misalnya, iOS membutuhkan properti CSS tambahan, -webkit-overflow-scrolling: touch, untuk "mengaktifkan kembali" fling scrolling, tetapi Anda tidak bisa mengontrol sumbunya, seperti yang bisa dilakukan dengan properti overflow standar. Pastikan untuk menguji implementasinya pada berbagai perangkat!

Selain transisi antar tampilan, teknik ini juga bisa diterapkan ke elemen geser-masuk lainnya, seperti elemen navigasi bilah sisi. Satu-satunya perbedaan adalah bahwa Anda tidak perlu memindahkan tampilan lainnya.

Pastikan bahwa animasi Anda berjalan pada layar yang lebih besar

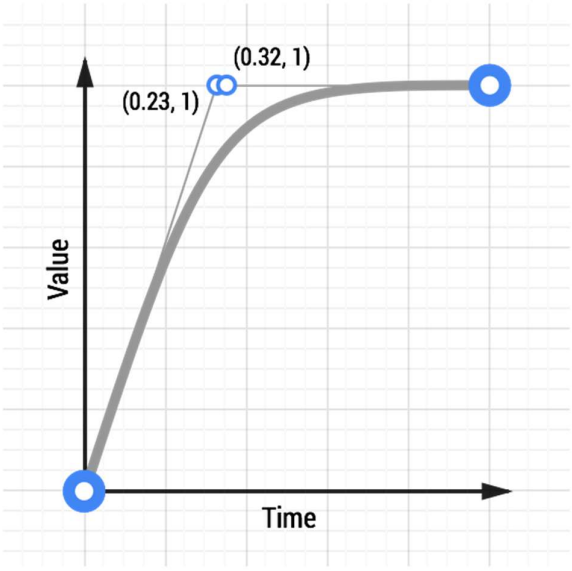
Pada layar yang lebih besar, Anda harus selalu menampilkan tampilan daftar bukan menghilangkannya, dan menggeser pada tampilan detail dari sisi sebelah kanan. Ini hampir sama dengan menangani tampilan navigasi.

8.5 Memilih Easing yang Tepat

Setelah membahas berbagai pilihan yang tersedia untuk efek easing di animasi, apa yang harus Anda pakai dalam proyek, dan berapa durasi yang sebaiknya digunakan dalam animasi Anda?

TL;DR

- Gunakan animasi ease-out untuk elemen UI; ease-out Quintic adalah ease yang sangat bagus, meskipun cepat.
- Pastikan untuk menggunakan durasi animasi; ease-out dan ease-in harus 200 md-500 md, sedangkan easing memantul dan elastis harus memiliki durasi yang lebih lama sekitar 800 md-1200 md.



Secara umum, **ease-out** adalah keputusan tepat, dan tentu saja standar yang baik. Efek ini cepat untuk dimulai, animasi Anda memberikan kesan responsif yang disukai, tetapi dengan perlambatan yang bagus di akhir.

Ada grup persamaan ease-out terkemuka selain yang ditetapkan dengan kata kunci `ease-out` di CSS, yang terentang dalam "agresivitas"-nya. Untuk efek ease-out yang cepat, pertimbangkan ease-out Quintic.

Lihat animasi ease-out Quintic

Efek easing lainnya, terutama easing memantul atau elastis tidak boleh sering digunakan, dan hanya dipakai ketika cocok untuk proyek Anda. Ada beberapa hal yang bisa menurunkan kualitas pengalaman pengguna, seperti animasi yang mengagetkan. Jika proyek Anda tidak ditujukan untuk kegembiraan, maka jangan gunakan elemen yang memantul di sekitar UI. Sebaliknya, jika Anda membuat sebuah situs yang seharusnya menggembirakan, maka tentu saja gunakan pantulan!

Coba beberapa easing, lihat mana yang sesuai dengan sifat proyek Anda, dan kembangkan dari situ. Untuk daftar lengkap tipe easing, bersama dengan demo-nya, lihat easings.net.

Pilih durasi animasi yang cocok

Setiap animasi yang ditambahkan ke proyek Anda harus memiliki durasi yang tepat. Animasi yang terlalu pendek akan terasa agresif dan tajam; jika terlalu lama, akan mengganggu dan menjengkelkan.

- **Ease-out: sekitar 200 md-500 md**. Ini memberikan kesempatan bagi pengguna untuk melihat animasi, tetapi tidak merasa terganggu.
- **Ease-in: sekitar 200 md-500 md**. Ingat bahwa efek akan tersentak di akhir dan perubahan waktu tidak akan melembutkan dampak itu.
- **Efek memantul atau elastis: sekitar 800 md-1200 md**. Anda harus memberikan waktu untuk efek memantul atau elastis agar "selesai." Tanpa waktu tambahan ini bagian memantul elastis dari animasi akan terlihat agresif dan tidak nyaman dilihat.

Tentu saja, ini hanya sekadar panduan. Lakukan percobaan dengan easing Anda sendiri dan pilih apa yang paling cocok untuk proyek Anda.

8.6 Menganimasikan Tampilan Modal

Tampilan modal hanya untuk pesan penting, dan Anda memiliki alasan yang sangat baik kenapa memblokir antarmuka pengguna. Gunakan dengan hati-hati, karena tampilan modal mengganggu dan bisa dengan mudah merusak pengalaman pengguna jika terlalu sering digunakan. Namun, dalam beberapa keadaan, tampilan modal adalah pilihan tampilan yang tepat, dan menambahkan beberapa animasi akan membuatnya semakin hidup.

Cobalah

TL;DR

- Gunakan tampilan modal secukupnya; pengguna akan merasa frustrasi jika Anda mengganggu pengalaman mereka degan hal-hal yang tidak penting.
- Menambahkan skala ke animasi memberikan efek "drop on" yang bagus.
- Singkirkan tampilan modal dengan cepat bila pengguna menutupnya. Namun, munculkan tampilan modal sedikit lebih lambat ke layar sehingga tidak mengejutkan pengguna.

Overlay modal harus selaras dengan tampilan yang terlihat, jadi setel `position`-nya ke `fixed`:

```
.modal {
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;

  pointer-events: none;
  opacity: 0;

  will-change: transform, opacity;
}
```

Contoh ini memiliki `opacity` awal 0 sehingga tersembunyi dari tampilan, dan `pointer-events` harus disetel ke `none` sehingga aktivitas klik dan sentuhan akan melewatinya. Tanpanya, semua interaksi akan diblokir, yang membuat seluruh laman menjadi tidak responsif. Yang

terakhir, karena itu menganimasikan opacity dan transform, harus ditandai sebagai berubah dengan will-change (lihat juga Menggunakan properti will-change).

Ketika terlihat, tampilan harus menerima interaksi dan memiliki opacity bernilai 1:

```
.modal.visible {
  pointer-events: auto;
  opacity: 1;
}
```

Sekarang, setiap kali tampilan modal diperlukan, Anda bisa menggunakan JavaScript untuk mengaktifkan kelas "visible":

```
modal.classList.add('visible');
```

Pada titik ini, tampilan modal muncul tanpa animasi apa pun, jadi sekarang Anda bisa menambahkannya dalam (lihat juga Easing Khusus):

```
.modal {
  -webkit-transform: scale(1.15);
  transform: scale(1.15);

  -webkit-transition:
    -webkit-transform 0.1s cubic-bezier(0.465, 0.183, 0.153, 0.946),
    opacity 0.1s cubic-bezier(0.465, 0.183, 0.153, 0.946);

  transition:
    transform 0.1s cubic-bezier(0.465, 0.183, 0.153, 0.946),
    opacity 0.1s cubic-bezier(0.465, 0.183, 0.153, 0.946);

}
```

Menambahkan scale dalam transformasi membuat tampilan tampak turun sedikit ke dalam layar, yang merupakan efek bagus. Transisi default diterapkan untuk properti transform dan opacity dengan kurva khusus dan durasi 0,1 detik.

Durasi ini cukup singkat, tetapi merupakan durasi yang ideal saat pengguna menutup tampilan dan ingin kembali ke aplikasi Anda. Kekurangannya adalah bahwa ketika tampilan modal muncul mungkin terlalu agresif. Untuk memperbaiki ini, ganti nilai-nilai transisi untuk kelas visible:

```
.modal.visible {
```

```
  -webkit-transform: scale(1);
  transform: scale(1);

  -webkit-transition:
    -webkit-transform 0.3s cubic-bezier(0.465, 0.183, 0.153, 0.946),
    opacity 0.3s cubic-bezier(0.465, 0.183, 0.153, 0.946);

  transition:
    transform 0.3s cubic-bezier(0.465, 0.183, 0.153, 0.946),
    opacity 0.3s cubic-bezier(0.465, 0.183, 0.153, 0.946);

}
```

Sekarang tampilan modal membutuhkan waktu 0,3 detik untuk ditampilkan ke layar, sedikit kurang agresif, tetapi bisa ditutup dengan cepat, yang akan lebih disukai pengguna.

8.7 Pengaturan waktu animasi asimetris

Pengaturan waktu animasi asimetris meningkatkan pengalaman pengguna dengan memungkinkan Anda untuk mengekspresikan kepribadian dan pada saat yang bersamaan merespons dengan cepat setiap interaksi pengguna. Hal ini juga memberikan kesan berbeda, yang membuat antarmuka lebih menarik secara visual.

TL;DR

- Gunakan pengaturan waktu animasi asimetris untuk menambahkan kepribadian dan perbedaan ke pekerjaan Anda.
- Selalu utamakan interaksi pengguna; gunakan durasi lebih pendek ketika merespons ketukan atau klik, dan simpan durasi yang lebih lama untuk hal lainnya.

Seperti kebanyakan "aturan" animasi, Anda harus bereksperimen untuk mencari tahu apa yang paling cocok bagi aplikasi Anda, namun ketika berurusan dengan pengalaman pengguna, pengguna itu terkenal tidak sabar. Aturan mudahnya adalah **selalu menanggapi interaksi pengguna dengan cepat**. Jadi karena, sebagian besar aksi pengguna adalah asimetris, maka animasi juga begitu.

Misalnya, ketika pengguna mengetuk untuk menampilkan navigasi bilah sisi, Anda harus menampilkannya ke layar secepat mungkin, dengan durasi sekitar 100 md. Namun ketika pengguna menutup menu, Anda bisa menganimasikan tampilan keluar dengan sedikit lebih lambat, misalnya dengan durasi sekitar 300 md.

Sebaliknya, ketika Anda menyajikan tampilan modal, ini biasanya untuk menampilkan pesan kesalahan atau beberapa pesan penting lainnya. Dalam keadaan seperti ini, Anda menyajikan tampilan dengan sedikit lebih lambat, sekitar 300 md, namun saat ditutup, yang dipicu oleh pengguna, harus terjadi sangat cepat.

Maka aturan mudahnya adalah sebagai berikut:

- Untuk animasi UI yang dipicu oleh interaksi pengguna, seperti transisi tampilan atau menampilkan elemen, gunakan proses masuk cepat (durasi singkat), tapi proses keluar lambat (durasi lebih lama).
- Untuk animasi UI yang dipicu oleh kode, seperti kesalahan atau tampilan modal, gunakan proses masuk lebih lambat (durasi lebih lama), tapi proses keluar cepat (durasi singkat).

8.8 Animasi dan Kinerja

Pertahankan 60 fps setiap kali Anda melakukan animasi, karena bila kurang bisa mengakibatkan ketidaklancaran atau perlambatan yang akan terlihat oleh pengguna dan berdampak negatif terhadap pengalaman pengguna.

TL;DR

- Jagalah agar animasi tidak menyebabkan masalah kinerja; pastikan Anda tahu dampak menganimasikan properti CSS yang diberikan.
- Menganimasikan properti yang bisa mengubah geometri laman (layout) atau menyebabkan painting berdampak sangat merugikan.
- Sebisa mungkin, tetap konsisten pada ubahan transform dan opacity.
- Gunakan will-change untuk memastikan bahwa browser tahu yang Anda rencanakan untuk dianimasikan.

Menganimasikan properti ini bukannya tanpa risiko, dan beberapa properti lebih mudah dianimasikan dibandingkan yang lainnya. Misalnya, menganimasikan width dan height dari sebuah elemen akan mengubah geometrinya dan bisa menyebabkan elemen lain pada laman tersebut berpindah atau berubah ukurannya. Proses ini disebut *layout* (atau *mengubah posisi/geometri* di browser berbasis Gecko seperti Firefox), dan bisa sangat merugikan jika laman Anda memiliki banyak elemen. Setiap kali layout terpicu, laman atau bagian darinya biasanya perlu digambar, yang biasanya lebih mahal daripada operasi layout itu sendiri.

Sebisa mungkin, Anda harus menghindari melakukan animasi properti yang memicu layout atau paint. Untuk kebanyakan browser modern, ini berarti membatasi animasi ke opacity atau transform, yang keduanya bisa dioptimalkan oleh browser; tidak masalah jika animasi ditangani oleh JavaScript atau CSS.

Untuk daftar lengkap pekerjaan yang dipicu oleh properti CSS individual, lihat Pemicu CSS. Anda bisa menemukan panduan lengkap tentang membuat Animasi Berkinerja Tinggi pada HTML5 Rocks.

Menggunakan properti will-change

Gunakan will-change agar browser mengetahui bahwa Anda bermaksud mengubah suatu properti elemen. Hal ini memungkinkan browser untuk menetapkan optimalisasi yang paling tepat sebelum Anda membuat perubahan. Namun, jangan terlalu sering menggunakan will-change, karena hal itu bisa menyebabkan browser membuang sumber daya, yang pada akhirnya dapat menyebabkan lebih banyak masalah kinerja.

Aturan mudahnya adalah bahwa jika animasi mungkin dipicu dalam 200 md berikutnya, baik oleh interaksi pengguna atau karena status aplikasi, maka menggunakan will-change pada elemen animasi adalah ide yang baik. Pada kebanyakan kejadian, setiap elemen dalam tampilan aktif aplikasi yang Anda ingin animasikan harus mengaktifkan will-change untuk properti apa pun yang ingin Anda ubah. Pada kejadian contoh boks yang telah digunakan dalam panduan sebelumnya, menambahkan will-change untuk transform dan opacity terlihat seperti ini:

```
.box {
  will-change: transform, opacity;
}
```

Sekarang browser yang mendukungnya, saat ini Chrome, Firefox dan Opera, akan membuat optimalisasi yang sesuai dalam pengaturannya untuk mendukung perubahan atau menganimasikan properti tersebut.

Kinerja CSS vs JavaScript

Ada banyak thread laman dan komentar di web yang membahas manfaat relatif dari animasi CSS dan JavaScript dari perspektif kinerja. Berikut adalah beberapa poin yang perlu diperhatikan:

- Animasi berbasis CSS, dan Animasi Web yang aslinya sudah didukung, biasanya ditangani di thread yang dikenal sebagai "thread compositor." Hal ini berbeda dengan "thread utama" browser, dengan penataan gaya, layout, painting, dan JavaScript dieksekusi. Ini berarti bahwa jika browser sedang menjalankan beberapa tugas penting di thread utama, animasi ini bisa terus bekerja tanpa terganggu.
- Perubahan lain untuk mengubah transform dan opacity, dalam beberapa kasus, juga ditangani oleh thread compositor.
- Jika animasi memicu paint, layout, atau keduanya, "thread utama" akan diharuskan untuk bekerja. Hal ini berlaku untuk animasi berbasis CSS dan JavaScript, dan overhead layout atau paint akan mengerdilkan setiap pekerjaan yang terkait dengan eksekusi CSS atau JavaScript, membuat persoalan yang tidak pasti.

Untuk informasi selengkapnya tentang pekerjaan apa yang dipicu karena menganimasikan properti tertentu, lihat Pemicu CSS.

BAB IX Dasar-Dasar Desain Web Responsif

Sasaran Pembelajaran

Mahasiswa mampu membuat desain web responsif

Kemampuan mahasiswa yang menjadi prasyarat

Mahasiswa sudah menguasai materi Interaksi Manusia Komputer, menguasai tools untuk mendesain, Javascript, CSS dan animasi

Keterkaitan bahan pembelajaran dengan pokok bahasan lainnya

Materi ini sebagai muara materi-materi yang dipelajari sebelumnya

Manfaat atau pentingnya bahan pembelajaran ini

Membuat aplikasi web/situs dan mobile yang menarik

Petunjuk belajar

Dalam materi ini mahasiswa akan mempelajari dasar-dasar desain web responsif dengan Google LePage! mahasiswa akan membuat halaman web responsif sendiri yang berfungsi baik di perangkat apa pun - ponsel, tablet, desktop, atau apa pun di antaranya.

Mahasiswa akan mulai dengan mengeksplorasi apa yang membuat situs responsif dan bagaimana beberapa pola desain responsif umum bekerja di berbagai perangkat. Dari sana, mahasiswa akan belajar cara membuat tata letak responsif sendiri menggunakan tag viewport dan kueri media CSS. Saat mahasiswa melanjutkan, akan bereksperimen dengan breakpoint besar dan kecil, dan mengoptimalkan teks untuk dibaca.

Penggunaan perangkat seluler untuk menjelajahi web tumbuh dengan kecepatan fantastis, tapi sayangnya banyak web tidak dioptimalkan untuk perangkat seluler. Perangkat seluler sering terkendala dengan ukuran layar dan memerlukan pendekatan berbeda dalam bagaimana materi ditampilkan di layar.

Ada banyak ukuran layar yang berbeda untuk ponsel, "phablet," tablet, desktop, konsol game, TV, dan bahkan perangkat yang dapat dikenakan. Ukuran layar selalu berubah, jadi sangatlah penting agar situs Anda bisa beradaptasi dengan tiap ukuran layar, sekarang atau di masa mendatang.

Desain web responsif, awalnya didefinisikan oleh Ethan Marcotte di A List Apart, sebagai jawaban atas kebutuhan pengguna dan perangkat yang mereka gunakan. Perubahan layout berdasarkan ukuran dan kemampuan perangkat. Misalnya, di ponsel, pengguna melihat materi

yang ditampilkan dalam tampilan satu kolom; tablet mungkin menampilkan materi yang sama dalam dua kolom.

9.1 Responsive Web Design

Menyetel tampilan yang terlihat

Laman yang dioptimalkan untuk berbagai perangkat harus menyertakan tag meta viewport di kepala dokumen. Sebuah tag meta viewport memberikan petunjuk ke browser tentang cara mengontrol ukuran laman dan penskalaan.

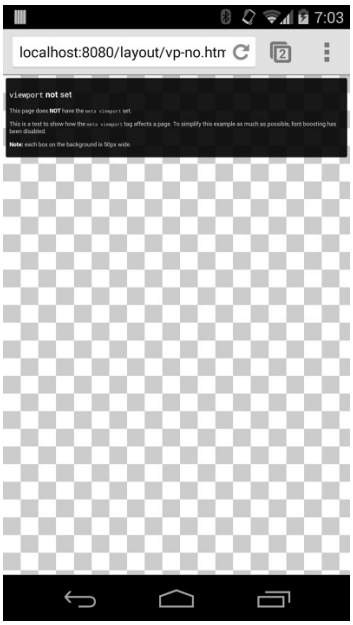
TL;DR

- Gunakan tag meta viewport untuk mengontrol lebar dan penskalaan tampilan yang terlihat di browser.
- Sertakan `width=device-width` untuk mencocokkan lebar layar dalam piksel yang tidak bergantung perangkat.
- Sertakan `initial-scale=1` untuk membentuk hubungan 1:1 antara piksel CSS dan piksel yang tidak bergantung perangkat.
- Pastikan laman Anda bisa diakses dengan tidak menonaktifkan penskalaan pengguna.

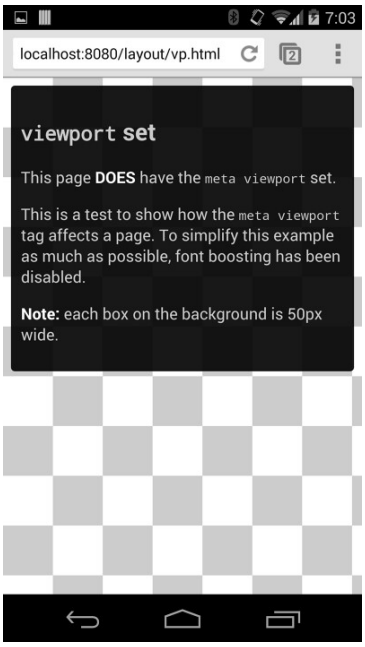
Dalam upaya menyediakan pengalaman terbaik, browser seluler merender laman pada lebar layar desktop (biasanya sekitar 980 px, meskipun ini bisa berbeda antar perangkat), dan kemudian mencoba membuat materi terlihat lebih baik dengan memperbesar ukuran font dan mengubah ukuran materi agar sesuai dengan layar. Ini berarti bahwa ukuran font mungkin tampil tidak konsisten bagi pengguna, yang mungkin harus ketuk dua kali atau cubit-untuk-zoom agar bisa melihat dan berinteraksi dengan materi.

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

Menggunakan nilai meta viewport `width=device-width` menginstruksikan laman untuk mencocokkan lebar layar dalam piksel yang tidak tergantung perangkat. Hal ini memungkinkan laman untuk meng-ubah posisi/geometri materi agar sesuai dengan ukuran layar yang berbeda, apakah di-render pada telepon seluler kecil atau monitor desktop yang besar.



Laman tanpa penyetelan tampilan yang terlihat



Laman dengan penyetelan tampilan yang terlihat

Beberapa browser menjaga lebar laman konstan ketika memutar ke mode lanskap, dan melakukan zoom bukannya meng-ubah posisi/geometri untuk mengisi layar. Menambahkan atribut `initial-scale=1` menginstruksikan browser untuk membangun hubungan 1:1 antara piksel

CSS dan piksel yang tidak tergantung perangkat terlepas dari orientasi perangkat, dan memungkinkan laman untuk memanfaatkan lebar lanskap penuh.

Note: Untuk memastikan browser lama bisa dengan benar parse atribut, gunakan tanda koma untuk memisahkan atribut.

Memastikan tampilan yang terlihat bisa diakses

Selain menetapkan initial-scale, Anda juga bisa mengatur atribut berikut pada tampilan yang terlihat:

- minimum-scale
- maximum-scale
- user-scalable

Bila diatur, ini bisa menonaktifkan kemampuan pengguna untuk melakukan zoom tampilan yang terlihat, berpotensi menyebabkan masalah aksesibilitas.

Menyesuaikan ukuran materi dengan tampilan yang terlihat

Pada desktop dan perangkat seluler, pengguna terbiasa menggulir situs web secara vertikal, tidak secara horizontal; memaksa pengguna menggulir secara horizontal atau harus memperkecil tampilan agar bisa melihat seluruh laman akan menyebabkan pengalaman pengguna yang buruk.

TL;DR

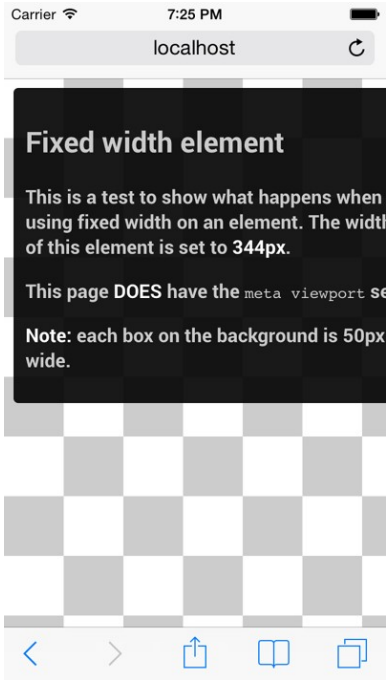
- Jangan menggunakan elemen berlebar tetap yang besar.
- Materi tidak boleh bergantung pada lebar tampilan yang terlihat tertentu untuk merender dengan baik.
- Gunakan kueri media CSS untuk menerapkan penataan gaya yang berbeda untuk layar kecil dan besar.

Ketika mengembangkan sebuah situs seluler dengan tag meta viewport, terkadang kita secara tidak sengaja membuat materi laman yang tidak muat dalam tampilan yang terlihat yang ditetapkan. Misalnya, gambar yang ditampilkan mempunyai lebar yang lebih lebar dari tampilan yang terlihat bisa menyebabkan tampilan yang terlihat untuk menggulir secara horizontal. Anda harus menyesuaikan materi ini agar muat ke dalam lebar tampilan yang terlihat, sehingga pengguna tidak perlu menggulir secara horizontal.

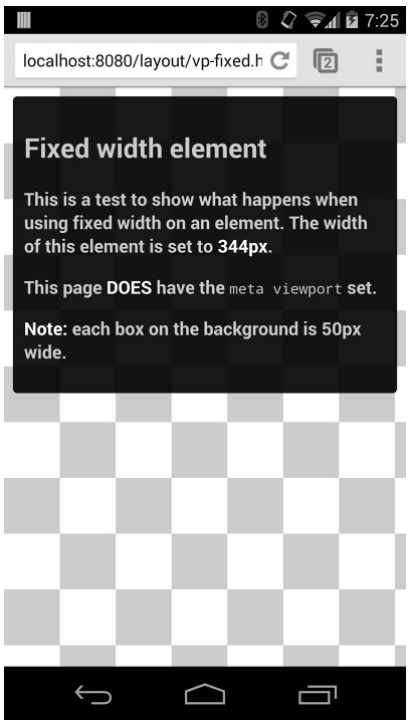
Oleh karena ukuran dan lebar layar dalam piksel CSS bervariasi antar perangkat (misalnya, antara ponsel dan tablet, dan bahkan antara ponsel yang berbeda), materi tidak boleh bergantung pada lebar tampilan yang terlihat tertentu untuk dirender dengan baik.

Menyetel lebar CSS mutlak besar untuk elemen laman (seperti contoh di bawah), menyebabkan div menjadi terlalu lebar untuk tampilan yang terlihat pada perangkat yang lebih

sempit (misalnya, perangkat dengan lebar piksel CSS 320, seperti iPhone). Sebaiknya, pertimbangkan untuk menggunakan nilai lebar relatif, seperti width: 100%. Demikian juga, berhati-hatilah menggunakan nilai pemosisian absolut besar yang bisa menyebabkan elemen berada di luar tampilan yang terlihat pada layar kecil.



Laman dengan elemen lebar tetap 344 px pada iPhone



Laman dengan elemen lebar tetap 344 px pada Nexus 5

Menggunakan kueri media CSS agar responsif

Kueri media adalah filter sederhana yang bisa diterapkan pada gaya CSS. Kueri media memudahkan kita untuk mengubah gaya berdasarkan karakteristik dari perangkat yang merender materi, termasuk tipe tampilan, lebar, tinggi, orientasi dan bahkan resolusi.

TL;DR

- Gunakan kueri media untuk menerapkan gaya berdasarkan karakteristik perangkat.
- Gunakan `min-width` di atas `min-device-width` untuk memastikan pengalaman yang paling luas.
- Gunakan ukuran relatif untuk elemen agar tidak merusak layout.

Misalnya, Anda bisa menempatkan semua gaya yang diperlukan untuk pencetakan dalam kueri media cetak:

```
<link rel="stylesheet" href="print.css" media="print">
```

Selain menggunakan atribut `media` di tautan style sheet, ada dua cara lain untuk menerapkan kueri media yang bisa disematkan dalam file CSS: `@media` dan `@import`. Karena alasan kinerja, salah satu dari dua metode pertama tersebut direkomendasikan pada sintaks `@import` (lihat Hindari pengimporan CSS).

```
@media print {
  /* print style sheets go here */
}

@import url(print.css) print;
```

Logika yang berlaku untuk kueri media adalah tidak saling eksklusif, dan untuk setiap filter yang memenuhi kriteria tersebut maka blok CSS yang dihasilkan akan diterapkan dengan menggunakan aturan standar prioritas sesuai CSS.

Menggunakan kueri media berdasarkan ukuran tampilan yang terlihat

Kueri media memungkinkan kita untuk menciptakan pengalaman responsif ketika gaya tertentu diaplikasikan ke layar kecil, layar besar, dan semua layar. Sintaks kueri media memungkinkan untuk pembuatan aturan yang bisa diterapkan tergantung pada karakteristik perangkat.

```
@media (query) {
  /* CSS Rules used when query matches */
}
```

Meskipun ada beberapa item berbeda yang bisa kita kueri, yang paling sering digunakan untuk desain web responsif adalah min-width, max-width, min-height, dan max-height.

Parameter	
min-width	Aturan diterapkan untuk setiap browser yang mempunyai lebar lebih besar dari nilai yang didefinisikan dalam kueri.
max-width	Aturan diterapkan untuk setiap browser yang mempunyai lebar lebih kecil dari nilai yang didefinisikan dalam kueri.
min-height	Aturan diterapkan untuk setiap browser yang mempunyai tinggi lebih besar dari nilai yang didefinisikan dalam kueri.
max-height	Aturan diterapkan untuk setiap browser yang mempunyai tinggi lebih kecil dari nilai yang didefinisikan dalam kueri.
orientation=portrait	Aturan diterapkan untuk setiap browser ketika tingginya lebih besar dari atau sama dengan lebarnya.
orientation=landscape	Aturan untuk setiap browser ketika lebarnya lebih besar dari tingginya.

Mari kita lihat contoh berikut:

Pratinjau laman menggunakan kueri media untuk mengubah properti ketika diubah ukurannya.

```
<link rel="stylesheet" media="(max-width: 640px)" href="max-640px.css">
<link rel="stylesheet" media="(min-width: 640px)" href="min-640px.css">
<link rel="stylesheet" media="(orientation: portrait)" href="portrait.css">
<link rel="stylesheet" media="(orientation: landscape)" href="landscape.css">
<style>
  @media (min-width: 500px) and (max-width: 600px) {
    h1 {
      color: fuchsia;
    }

    .desc:after {
      content:" In fact, it's between 500px and 600px wide.";
    }
  }
</style>
```

Cobalah

- Ketika browser lebarnya antara **0 px** dan **640 px**, diterapkan max-640px.css.
- Ketika browser lebarnya antara **500 px** dan **600 px**, gaya dalam @media akan diterapkan.
- Ketika browser lebarnya **640 px atau lebih**, diterapkan min-640px.css.
- Ketika browser **lebarnya lebih besar daripada tingginya**, diterapkan landscape.css.
- Ketika browser **tingginya lebih besar daripada lebarnya**, diterapkan portrait.css.

Catatan tentang min-device-width

Adalah mungkin juga membuat kueri berdasarkan min-device-width, meskipun praktik ini **sangat tidak dianjurkan**.

Perbedaannya sangat kecil namun sangat penting: min-width didasarkan pada ukuran jendela browser sedangkan min-device-width didasarkan pada ukuran layar. Sayangnya beberapa browser, termasuk browser Android lama, tidak melaporkan lebar perangkat dengan benar; browser tersebut melaporkan ukuran layar dalam satuan piksel perangkat, bukan dalam lebar tampilan yang terlihat yang diharapkan.

Selain itu, menggunakan min-device-width bisa mencegah materi diadaptasikan pada desktop atau perangkat lain yang memperbolehkan jendela diubah ukurannya karena kueri didasarkan pada ukuran perangkat yang sebenarnya, bukan ukuran jendela browser.

Gunakan any-pointer dan any-hover untuk interaksi yang fleksibel

Dimulai dengan Chrome 39, style sheet Anda bisa menulis selektor yang mencakup beberapa tipe pointer dan perilaku arahkan ke atas. Fitur media any-pointer dan any-hover mirip dengan pointer dan hover dalam mengizinkan Anda untuk melakukan kueri kemampuan pointer pengguna. Namun, tidak seperti yang terakhir, any-pointer dan any-hover beroperasi pada gabungan dari semua perangkat pointer dan bukan hanya perangkat pointer utama.

Menggunakan unit relatif

Konsep penting di balik desain responsif adalah fluiditas dan proporsionalitas yang bertentangan dengan konsep layout lebar tetap. Menggunakan unit relatif untuk pengukuran bisa membantu menyederhanakan layout dan mencegah kita secara tidak sengaja membuat komponen yang terlalu besar untuk tampilan yang terlihat.

Misalnya, setelan lebar: 100% pada div tingkat atas, memastikan bahwa itu membentang meliputi lebar tampilan yang terlihat dan tidak terlalu besar atau terlalu kecil untuk tampilan yang terlihat. div akan cocok, tidak peduli apakah itu iPhone berlebar 320 px, Blackberry Z10 berlebar 342 px, atau sebuah Nexus 5 yang berlebar 360 px.

Selain itu, menggunakan unit relatif memungkinkan browser untuk merender materi berdasarkan tingkat zoom pengguna tanpa perlu menambahkan bilah gulir horizontal ke laman.

Tidak disarankan—lebar tetap

```
div.fullWidth {
  width: 320px;
  margin-left: auto;
  margin-right: auto;
}
```

Disarankan—lebar responsif

```
div.fullWidth {
  width: 100%;
}
```

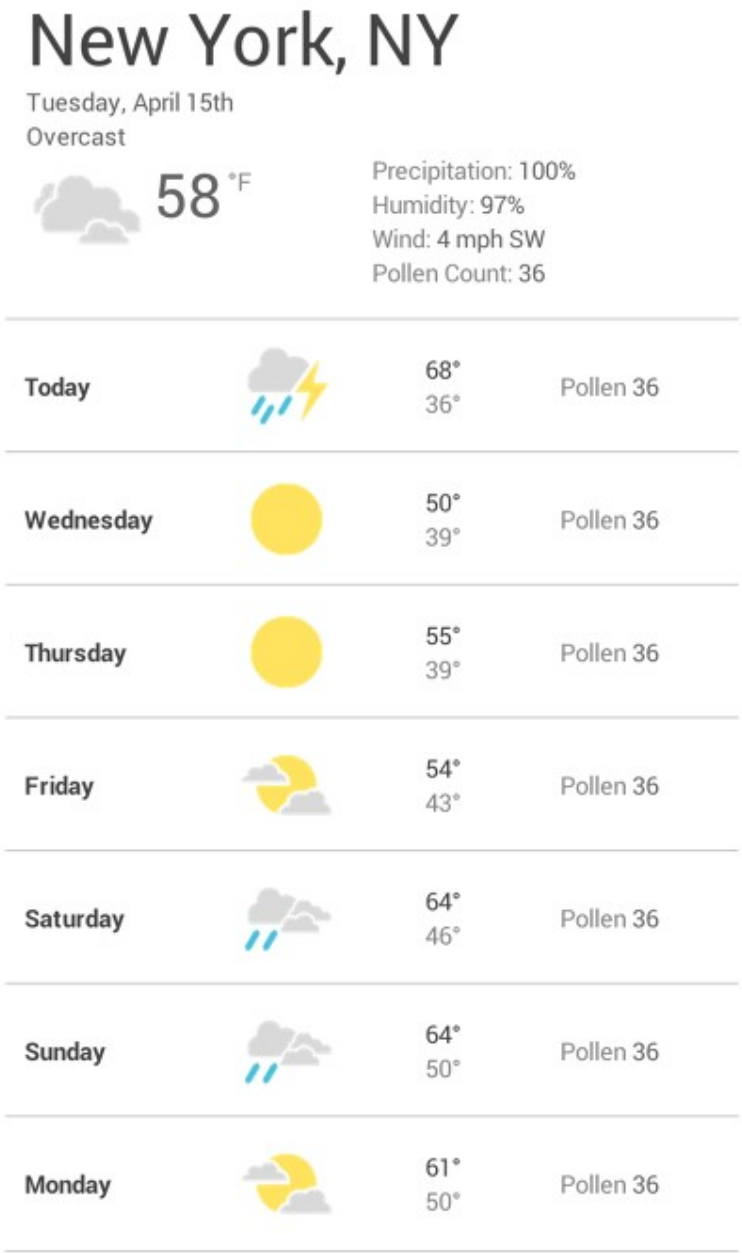
Cara memilih breakpoint

Jangan mendefinisikan breakpoint berdasarkan kelas perangkat. Mendefinisikan breakpoint berdasarkan perangkat, produk, nama merek, atau sistem operasi tertentu yang digunakan saat ini bisa mengakibatkan mimpi buruk dalam pemeliharaan. Malahan, materi itu sendiri yang harus menentukan bagaimana layout menyesuaikan dengan kontainer.

TL;DR

- Buat breakpoint berdasarkan materi, jangan pernah berdasarkan perangkat, produk, atau merek tertentu.
- Desainlah untuk perangkat seluler terkecil lebih dahulu; kemudian secara progresif meningkatkan pengalaman pengguna seiring bertambahnya properti layar.
- Jaga jumlah maksimum baris teks sekitar 70 atau 80 karakter.

Memilih breakpoint utama dengan secara bertahap mulai dari layar kecil hingga ke besar.



Pratinjau prakiraan cuaca yang ditampilkan di layar kecil.

Desain materi agar pas dengan ukuran layar kecil terlebih dahulu, kemudian perluas layar sampai breakpoint menjadi diperlukan. Ini memungkinkan Anda untuk mengoptimalkan breakpoint berdasarkan materi dan mempertahankan jumlah breakpoint sesedikit mungkin.

Mari kita bekerja melalui contoh yang kita lihat di awal: prakiraan cuaca. Langkah pertama adalah membuat prakiraan terlihat bagus di layar kecil.

Pratinjau prakiraan cuaca ketika laman semakin luas.

Berikutnya, ubah ukuran browser sampai ada terlalu banyak ruang putih antara elemen, dan tampilan prakiraan cuaca terlihat tidak bagus. Keputusan ini sedikit subjektif, namun di atas 600px pasti terlalu lebar.

Untuk memasukkan breakpoint pada 600 px, buat dua style sheet baru, satu untuk digunakan saat browser 600 px dan kurang dari itu, dan satu ketika luasnya lebih dari 600 px.

```
<link rel="stylesheet" href="weather.css">
<link rel="stylesheet" media="(max-width:600px)" href="weather-2-small.css">
<link rel="stylesheet" media="(min-width:601px)" href="weather-2-large.css">
```

Cobalah

Pratinjau dari prakiraan cuaca yang dirancang untuk layar yang lebih lebar.

Yang terakhir, optimalisasi CSS. Dalam contoh ini, kami telah menempatkan gaya umum seperti font, ikon, pemosisian dasar, dan warna di weather.css. Layout tertentu untuk layar kecil kemudian ditempatkan di weather-small.css, dan model layar besar ditempatkan di weather-large.css.

Memilih breakpoint kecil bila diperlukan

Selain memilih breakpoint besar ketika layout berubah secara signifikan, ini juga membantu untuk menyesuaikan perubahan kecil. Misalnya, antara breakpoint utama mungkin ada gunanya mengatur margin atau padding pada elemen, atau memperbesar ukuran font agar terlihat lebih natural dalam layout.

Mari kita mulai dengan mengoptimalkan layout layar kecil. Pada kasus ini, mari kita memperbesar font ketika luas tampilan yang terlihat lebih besar dari 360px. Kedua, ketika ada cukup ruang, kita bisa memisahkan suhu tinggi dan rendah sehingga semua berada di baris yang sama dan tidak tumpang tindih. Dan juga mari kita buat ikon cuaca sedikit lebih besar.

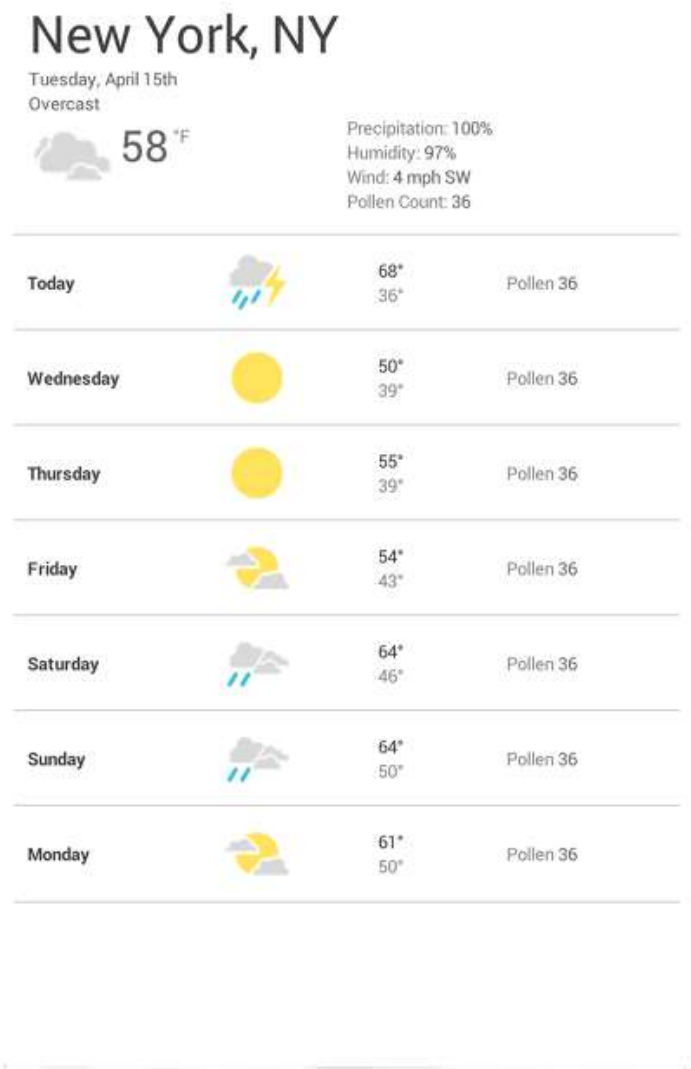
```
@media (min-width: 360px) {
 body {
  font-size: 1.0em;
 }
}

@media (min-width: 500px) {
 .seven-day-fc .temp-low,
 .seven-day-fc .temp-high {
```

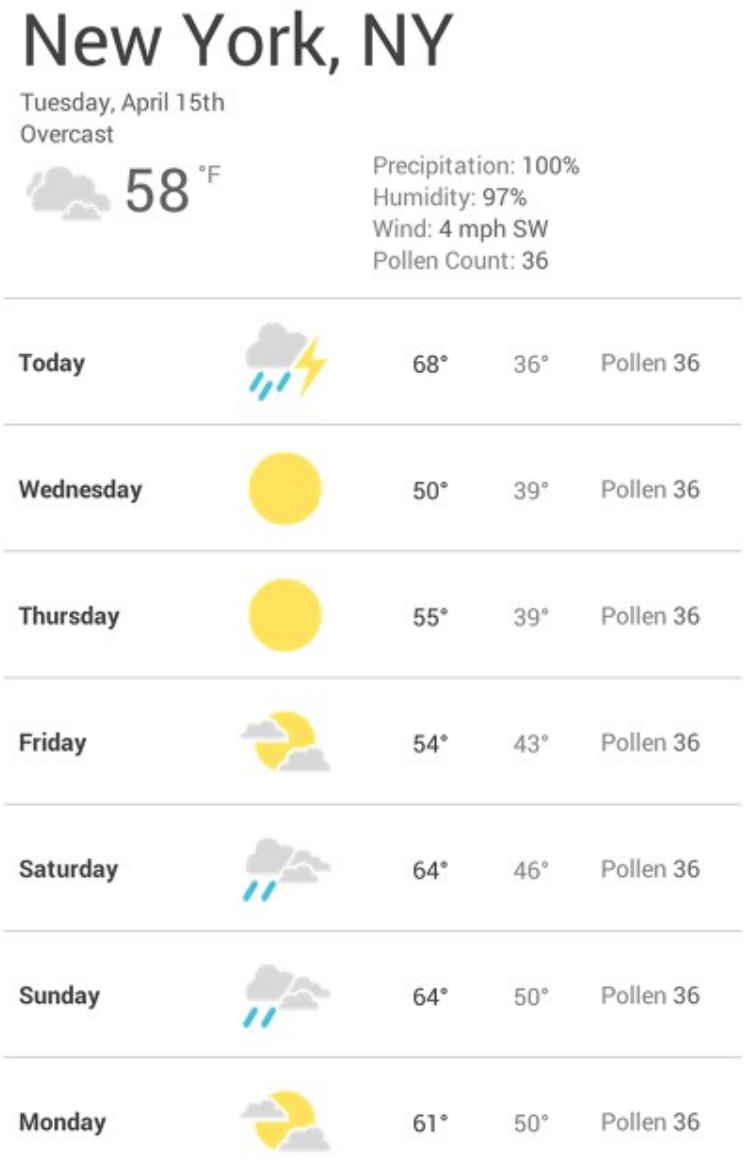
```
  display: inline-block;
  width: 45%;
 }

 .seven-day-fc .seven-day-temp {
  margin-left: 5%;
 }

 .seven-day-fc .icon {
  width: 64px;
  height: 64px;
 }
}
```



Sebelum menambahkan breakpoint kecil.



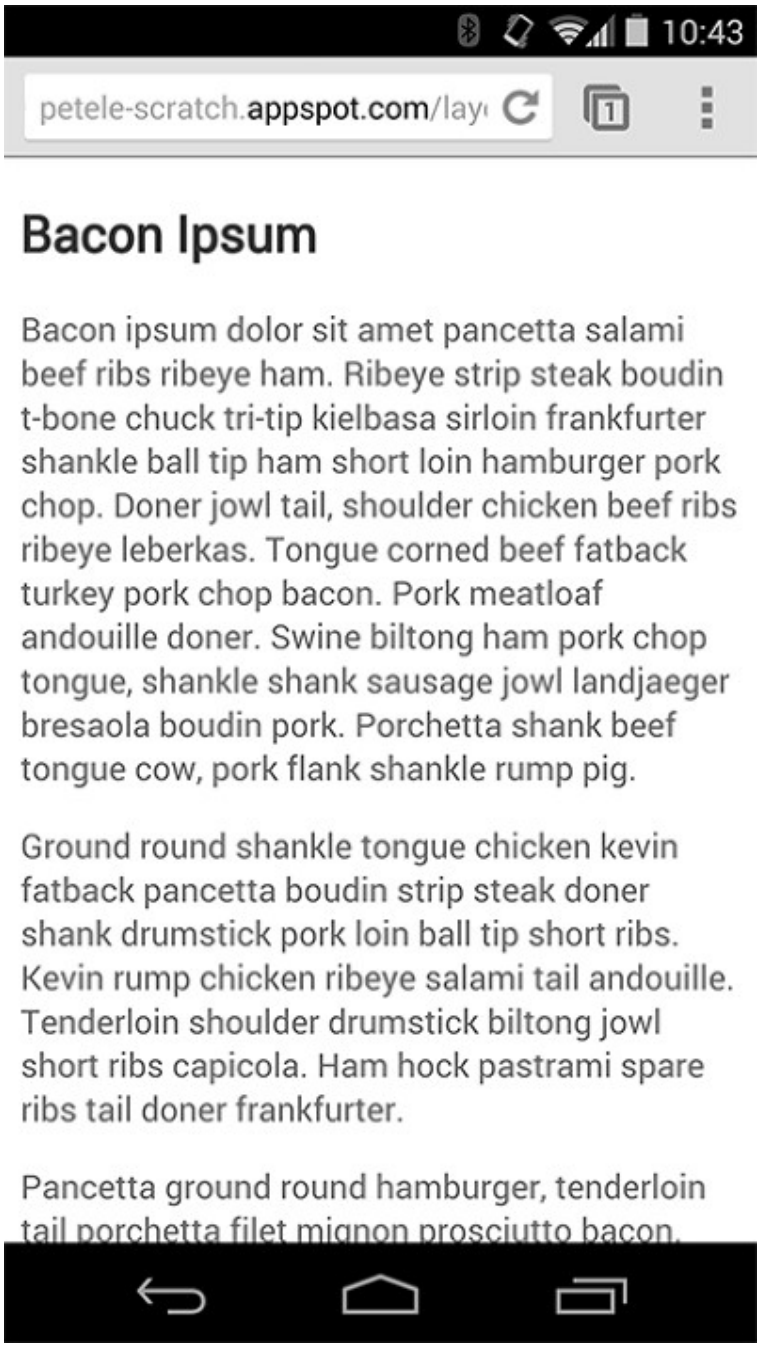
Setelah menambahkan breakpoint kecil.

Demikian pula, untuk layar besar akan sangat baik membatasi lebar maksimum panel prakiraan cuaca sehingga tidak memakai seluruh lebar layar.

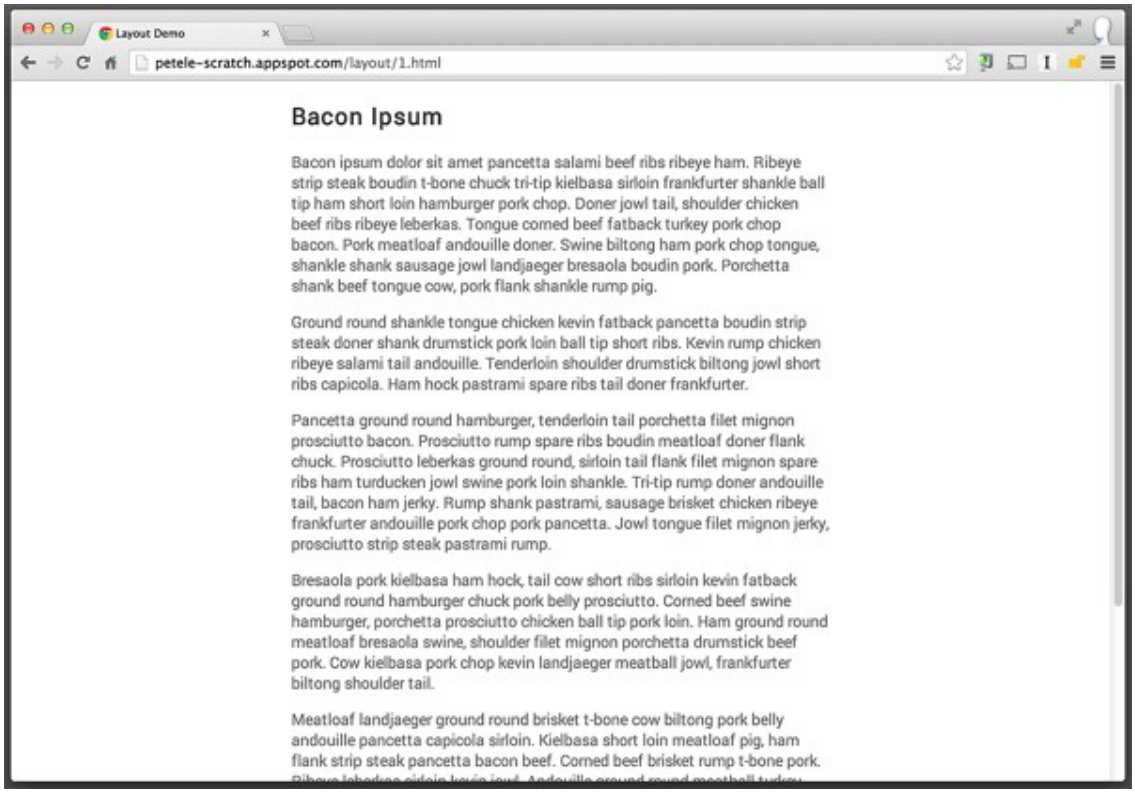
```
@media (min-width: 700px) {
  .weather-forecast {
    width: 700px;
  }
}
```

Mengoptimalkan teks untuk bacaan

Teori pembacaan klasik menyarankan bahwa kolom yang ideal harus berisi 70 sampai 80 karakter per baris (sekitar 8 sampai 10 kata dalam bahasa Inggris). Jadi setiap kali lebar blok teks bertambah melewati 10 kata, pertimbangkan menambahkan breakpoint.



Sebelum menambahkan breakpoint kecil.



Setelah menambahkan breakpoint kecil.

Mari kita lihat secara lebih mendalam contoh entri blog di atas. Pada layar yang lebih kecil, font Roboto dengan ukuran 1 em bekerja secara sempurna memberikan 10 kata per baris, namun layar yang lebih besar membutuhkan breakpoint. Pada kasus ini, jika lebar browser lebih besar dari 575px, lebar ideal materi adalah 550px.

```
@media (min-width: 575px) {
 article {
  width: 550px;
  margin-left: auto;
  margin-right: auto;
 }
}
```

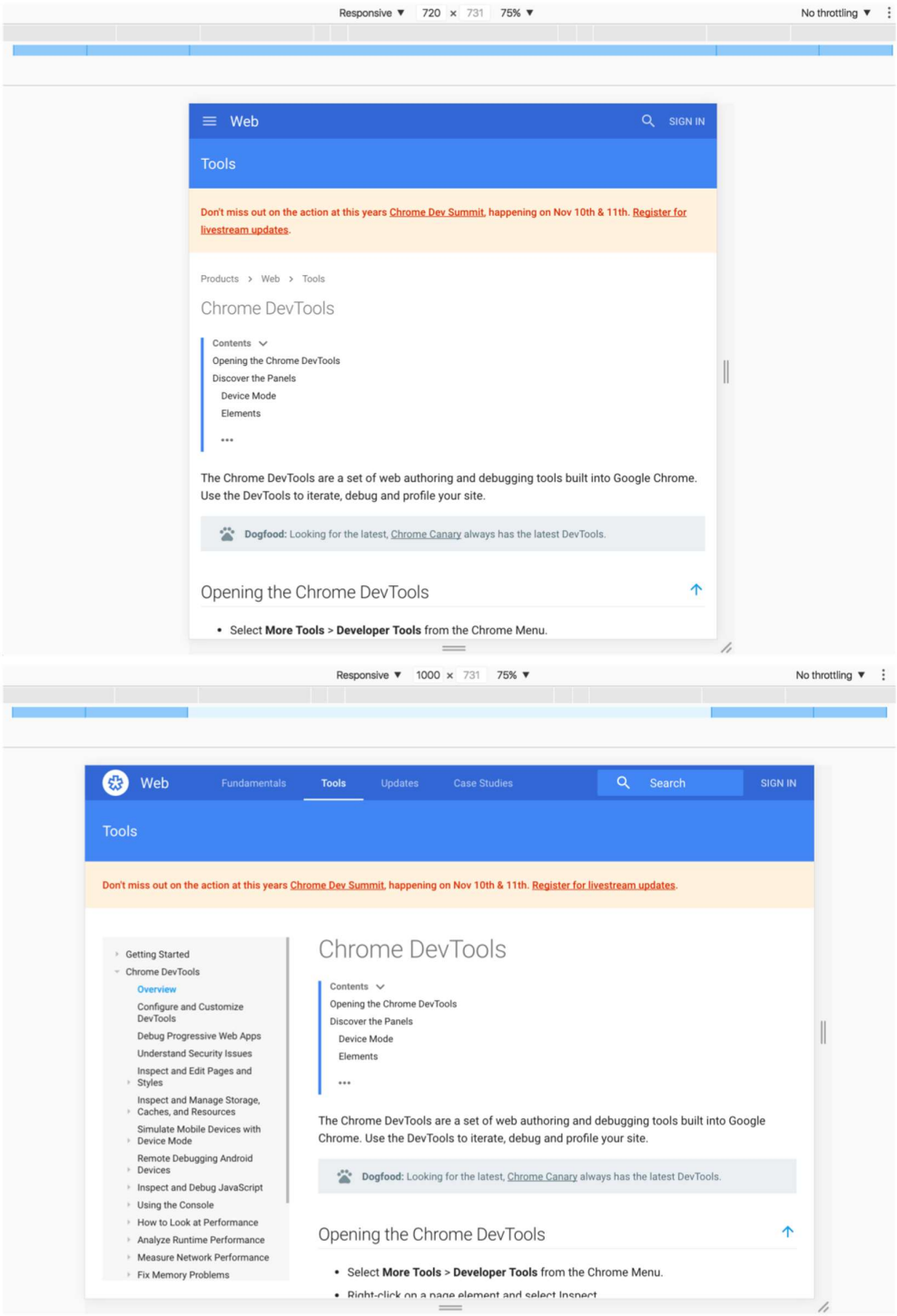
Cobalah

Jangan pernah benar-benar menyembunyikan materi

Berhati-hatilah saat memilih materi yang disembunyikan atau ditampilkan menurut ukuran layar. Jangan sembunyikan materi hanya karena Anda tidak bisa memuatnya di layar. Ukuran layar bukanlah indikasi pasti mengenai apa yang mungkin diinginkan pengguna. Misalnya, menghilangkan hitungan serbuk sari dari prakiraan cuaca bisa menjadi masalah serius bagi penderita alergi musim-semi yang membutuhkan informasi untuk menentukan apakah mereka bisa pergi ke luar atau tidak.

Menampilkan breakpoint kueri media di Chrome DevTools

Setelah Anda menyiapkan breakpoint kueri media, Anda pasti ingin melihat bagaimana situs akan terlihat. Anda *bisa* mengubah ukuran jendela browser untuk memicu breakpoint, namun ada cara yang lebih baik: Chrome DevTools. Dua tangkapan layar di bawah menunjukkan penggunaan DevTools untuk menampilkan bagaimana laman terlihat di bawah breakpoint yang berbeda.



Untuk menampilkan laman Anda di bawah breakpoint yang berbeda:

Buka DevTools kemudian hidupkan Device Mode.

Gunakan kontrol tampilan untuk memilih **Responsive**, yang menempatkan DevTools ke mode responsif.

Terakhir, buka menu Device Mode dan pilih **Show media queries** untuk menampilkan breakpoint sebagai bilah berwarna di atas laman Anda.

Klik pada salah satu bilah untuk menampilkan laman Anda saat kueri media aktif. Klik kanan pada bilah untuk melompat ke definisi kueri media. Lihat Kueri media untuk bantuan lebih lanjut.

9.2 Pola Desain Web Responsif

Pola desain web responsif berkembang dengan pesat, namun ada beberapa pola yang sudah terbukti bekerja dengan baik di desktop dan perangkat seluler.

Kebanyakan layout yang digunakan oleh laman web responsif bisa dikategorikan ke dalam salah satu dari lima pola ini: mostly fluid, column drop, layout shifter, tiny tweaks dan off canvas. Pada beberapa kejadian, laman mungkin menggunakan kombinasi pola, misalnya column drop dan off canvas. Pola-pola ini, yang awalnya diidentifikasi oleh Luke Wroblewski, memberikan titik awal yang solid untuk setiap laman responsif.

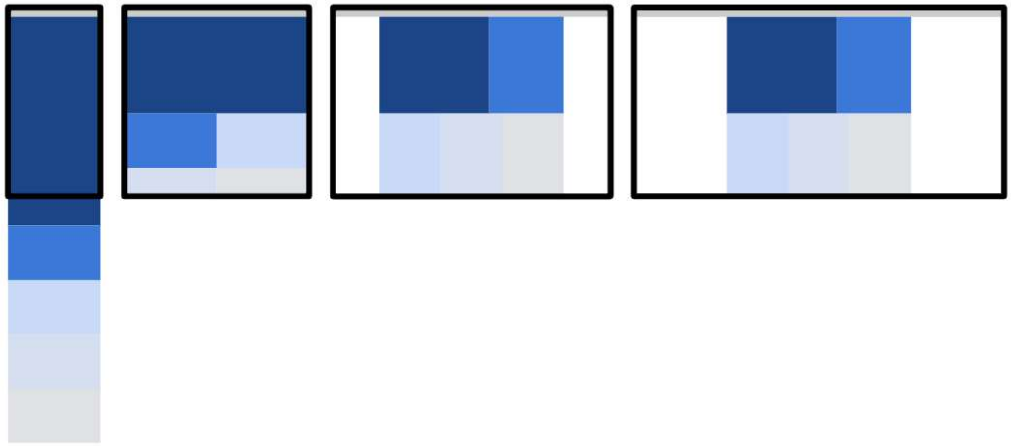
Pola

Agar sederhana serta mudah dipahami, masing-masing contoh di bawah ini dibuat dengan markup sungguhan menggunakan flexbox, biasanya dengan tiga materi div yang ditempatkan dalam kontainer primer div. Setiap contoh tersebut ditulis dimulai dari tampilan terkecil terlebih dahulu, dan ditambahkan breakpoint bila diperlukan. Mode layout flexbox didukung dengan baik untuk browser modern, meskipun mungkin masih memerlukan awalan vendor untuk dukungan optimal.

Mostly Fluid

Pola mostly fluid utamanya terdiri dari grid yang cair. Pada layar besar atau medium, biasanya ukurannya tetap sama, hanya menyesuaikan margin pada layar yang lebih lebar.

Pada layar yang lebih kecil, grid yang cair menyebabkan materi utama untuk meng-ubah posisi/geometri, seiring kolom ditumpuk secara vertikal. Salah satu keuntungan utama dari pola ini adalah bahwa pola ini biasanya hanya membutuhkan satu breakpoint antara layar kecil dan layar besar.



Cobalah

Pada tampilan terkecil, masing-masing div materi ditumpuk secara vertikal. Saat lebar layar menyentuh 600 px, materi div utama tetap berukuran width: 100%, sedangkan div sekunder ditampilkan sebagai dua kolom di bawah div utama. Di atas 800px, lebar kontainer div menjadi konstan dan di tengah layar.

Situs yang menggunakan pola ini antara lain:

- A List Apart
- Media Queries
- SimpleBits

```
.container {
  display: -webkit-flex;
  display: flex;
  -webkit-flex-flow: row wrap;
  flex-flow: row wrap;
}

.c1, .c2, .c3, .c4, .c5 {
  width: 100%;
}

@media (min-width: 600px) {
  .c2, .c3, .c4, .c5 {
    width: 50%;
  }
}

@media (min-width: 800px) {
```

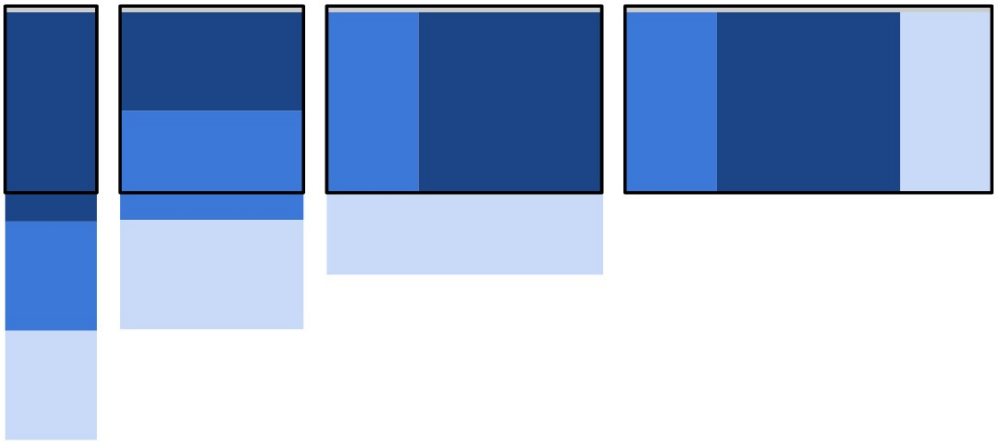
```
 .c1 {
   width: 60%;
 }
 .c2 {
   width: 40%;
 }
 /* Using 33.33%, doesn't always work right due to rounding */
 .c3, .c4 {
   width: 33%;
 }
 .c5 {
   width: 34%;
 }
}

@media (min-width: 800px) {
 .container {
   width: 800px;
   margin-left: auto;
   margin-right: auto;
 }
}
```

Kolom drop

Untuk layout multi-kolom lebar-penuh, column drop hanya menumpuk kolom secara vertikal saat lebar jendela terlalu sempit untuk materi.

Pada akhirnya proses ini mengakibatkan semua kolom ditumpuk secara vertikal. Memilih breakpoint untuk pola layout ini bergantung pada materi dan berubah untuk setiap desain.



Cobalah

Seperti kebanyakan contoh fluid, materi ditumpuk secara vertikal pada tampilan terkecil, namun ketika layar diluaskan melebihi 600px, materi `div` primer dan sekunder akan menggunakan lebar maksimal layar. Urutan `div` diatur menggunakan properti urutan CSS. Pada 800px ketiga materi `div` ditampilkan, menggunakan lebar layar penuh.

Situs yang menggunakan pola ini antara lain:

- Modernizr
- Wee Nudge

```
.container {
  display: -webkit-flex;
  display: flex;
  -webkit-flex-flow: row wrap;
  flex-flow: row wrap;
}

.c1, .c2, .c3 {
  width: 100%;
}

@media (min-width: 600px) {
  .c1 {
    width: 60%;
    -webkit-order: 2;
    order: 2;
  }

  .c2 {
    width: 40%;
    -webkit-order: 1;
    order: 1;
  }

  .c3 {
    width: 100%;
    -webkit-order: 3;
    order: 3;
  }
}


@media (min-width: 800px) {
  .c2 {
```

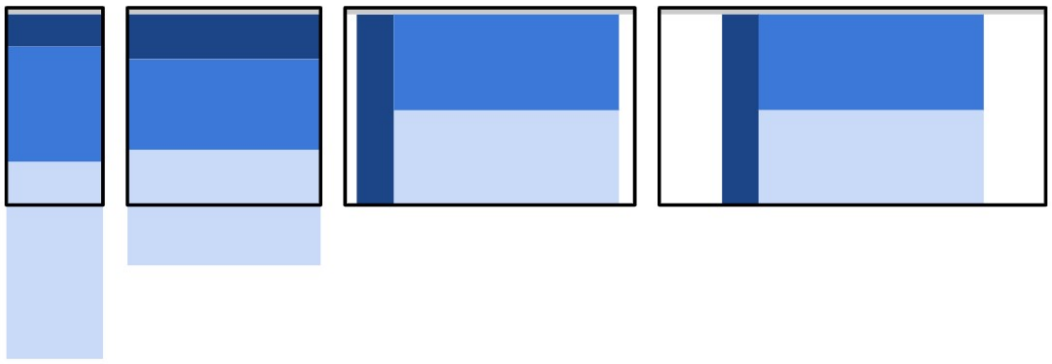
```
  width: 20%;
 }

 .c3 {
  width: 20%;
 }
}
```

Layout shifter

Pola layout shifter adalah pola yang paling responsif, dengan beberapa breakpoint melintasi beberapa lebar layar.

Kunci layout ini adalah tentang cara materi bergerak, bukan meng-ubah posisi/geometri dan menjatuhkannya di bawah kolom lainnya. Oleh karena perbedaan signifikan antara masing-masing breakpoint utama, itu lebih kompleks untuk mempertahankan dan mungkin melibatkan perubahan dalam elemen, bukan hanya layout materi secara keseluruhan.



Cobalah

Contoh yang disederhanakan ini menunjukkan pola layout shifter, pada layar yang lebih kecil materi ditumpuk secara vertikal, namun berubah secara signifikan ketika layar semakin besar, dengan `div` kiri dan dua `div` yang ditumpuk di sebelah kanan.

Situs yang menggunakan pola ini antara lain:

- Food Sense
- Contoh Desain Responsif Seminal
- Andersson-Wise Architects

```
.container {
 display: -webkit-flex;
 display: flex;
 -webkit-flex-flow: row wrap;
```

```
  flex-flow: row wrap;
}

.c1, .c2, .c3, .c4 {
  width: 100%;
}

@media (min-width: 600px) {
  .c1 {
    width: 25%;
  }

  .c4 {
    width: 75%;
  }

}

@media (min-width: 800px) {
  .container {
    width: 800px;
    margin-left: auto;
    margin-right: auto;
  }
}
```

Tiny tweaks

Tiny tweaks hanya melakukan perubahan kecil ke layout, seperti menyesuaikan ukuran font, mengubah ukuran gambar atau memindahkan materi dengan sangat kecil.

Ini bekerja dengan baik pada layout kolom tunggal seperti situs web linear laman tunggal dan artikel yang mengandung banyak teks.

Cobalah

Sesuai dengan namanya, tidak banyak perubahan yang dilakukan dengan contoh ini ketika ukuran layar berubah. Ketika lebar layar bertambah besar, begitu juga ukuran font dan pengisi.

Situs yang menggunakan pola ini antara lain:

- Ginger Whale
- Future Friendly

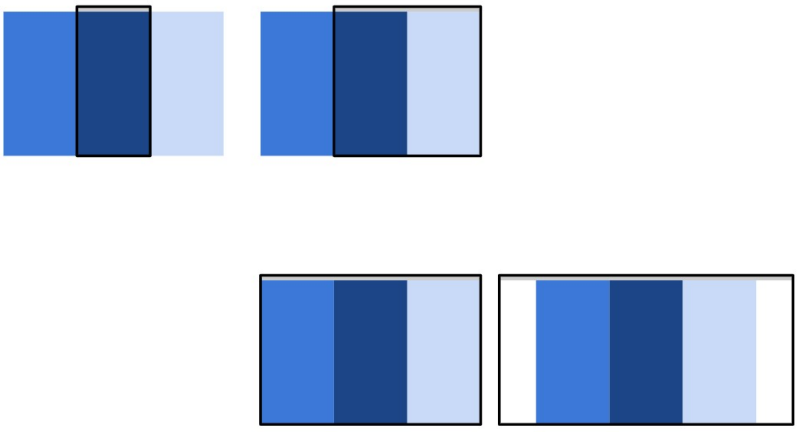
```
.c1 {
  padding: 10px;
  width: 100%;
}

@media (min-width: 500px) {
  .c1 {
    padding: 20px;
    font-size: 1.5em;
  }
}

@media (min-width: 800px) {
  .c1 {
    padding: 40px;
    font-size: 2em;
  }
}
```

Off canvas

Bukannya menumpuk materi secara vertikal, pola off canvas menempatkan materi yang lebih jarang digunakan—mungkin navigasi atau menu aplikasi—yang tidak terlihat di layar, dan hanya menampilkannya ketika ukuran layar cukup besar, pada layar yang lebih kecil, materi hanya satu klik jauhnya.



Cobalah

Bukannya menumpuk materi secara vertikal, contoh ini menggunakan deklarasi transform: translate(-250px, 0) untuk menyembunyikan dua div materi dari layar. JavaScript digunakan untuk menampilkan div dengan menambahkan kelas terbuka ke elemen untuk membuatnya terlihat. Ketika layar semakin lebar, posisi off-screen akan dihapus dari elemen dan mereka ditampilkan dalam tampilan yang terlihat.

Perhatikan dalam contoh ini, Safari untuk iOS 6 dan Browser Android tidak mendukung fitur flex-flow: row nowrap dari flexbox, jadi kami terpaksa melakukan fallback ke pemosisian absolut.

Situs yang menggunakan pola ini antara lain:

- Artikel HTML5Rocks
- Google Nexus
- Situs Seluler Facebook

```
body {
  overflow-x: hidden;
}

.container {
  display: block;
}

.c1, .c3 {
  position: absolute;
  width: 250px;
  height: 100%;

  /*
    This is a trick to improve performance on newer versions of Chrome
    #perfmatters
  */
  -webkit-backface-visibility: hidden;
  backface-visibility: hidden;

  -webkit-transition: -webkit-transform 0.4s ease-out;
  transition: transform 0.4s ease-out;

  z-index: 1;
}

.c1 {
  /*
```

```
  Using translate3d as a trick to improve performance on older versions of Chrome
  See: http://aerotwist.com/blog/on-translate3d-and-layer-creation-hacks/
  #perfmatters
  */
  -webkit-transform: translate(-250px,0);
  transform: translate(-250px,0);
}

.c2 {
  width: 100%;
  position: absolute;
}

.c3 {
  left: 100%;
}

.c1.open {
  -webkit-transform: translate(0,0);
  transform: translate(0,0);
}

.c3.open {
  -webkit-transform: translate(-250px,0);
  transform: translate(-250px,0);
}

@media (min-width: 500px) {
  /* If the screen is wider then 500px, use Flexbox */
  .container {
    display: -webkit-flex;
    display: flex;
    -webkit-flex-flow: row nowrap;
    flex-flow: row nowrap;
  }
  .c1 {
    position: relative;
    -webkit-transition: none 0s ease-out;
    transition: none 0s ease-out;
    -webkit-transform: translate(0,0);
    transform: translate(0,0);
  }
  .c2 {
    position: static;
```

```
  }
}

@media (min-width: 800px) {
  body {
    overflow-x: auto;
  }
  .c3 {
    position: relative;
    left: auto;
    -webkit-transition: none 0s ease-out;
    transition: none 0s ease-out;
    -webkit-transform: translate(0,0);
    transform: translate(0,0);
  }
}
```

9.3. Gambar

Desain web responsif berarti bahwa tidak hanya layout yang bisa berubah berdasarkan karakteristik perangkat, namun materi juga dapat berubah. Misalnya, pada tampilan (2x) resolusi tinggi, grafis resolusi tinggi memastikan ketajaman. Gambar dengan lebar 50% mungkin bekerja dengan baik ketika browser lebarnya 800 px, namun menggunakan terlalu banyak properti pada layar ponsel yang sempit, dan memerlukan overhead bandwidth yang sama ketika diperkecil agar muat pada layar yang lebih kecil.

Tujuan seni

Pada saat lainnya, gambar mungkin harus diubah lebih drastis: dengan mengubah ukuran, memotong dan bahkan mengganti seluruh gambar. Dalam hal ini, mengubah gambar biasanya disebut sebagai tujuan seni. Lihat responsiveimages.org/demos/ untuk contoh selengkapnya.

Responsive Images

Tahukah Anda bahwa gambar mewakili lebih dari 60% byte yang dibutuhkan rata-rata untuk memuat halaman web?

Dalam materi ini Anda akan belajar cara bekerja dengan gambar di web modern, sehingga gambar Anda terlihat bagus dan memuat dengan cepat di perangkat apa pun.

Sepanjang jalan, Anda akan mengambil berbagai keterampilan dan teknik untuk memadukan gambar responsif ke dalam alur kerja pengembangan Anda dengan lancar. Pada akhir kursus, Anda akan berkembang dengan gambar yang beradaptasi dan merespons berbagai ukuran viewport dan skenario penggunaan.

Gambar di markup

Elemen img adalah kuat—itu mengunduh, mengonversi dan merender materi—dan browser modern mendukung berbagai format gambar. Memasukkan gambar yang bekerja di seluruh perangkat tidak berbeda pada desktop, dan hanya membutuhkan beberapa ubahan kecil untuk menciptakan pengalaman pengguna yang baik.

TL;DR

- Gunakan ukuran relatif bagi gambar untuk mencegah mereka secara tanpa sengaja meluap dari kontainer.
- Gunakan elemen picture ketika Anda ingin menetapkan gambar yang berbeda bergantung pada karakteristik perangkat (alias tujuan seni).
- Gunakan srcset dan deskriptor x dalam elemen img untuk memberikan petunjuk ke browser tentang gambar terbaik yang digunakan saat memilih dari kepadatan yang berbeda.

- Jika laman Anda hanya memiliki satu atau dua gambar dan itu tidak digunakan di tempat lain pada situs Anda, pertimbangkan menggunakan gambar inline untuk mengurangi permintaan file.

Gunakan ukuran relatif untuk gambar

Ingatlah untuk menggunakan unit relatif ketika menetapkan lebar gambar untuk mencegah gambar tanpa sengaja meluap dari tampilan yang terlihat. Misalnya, width: 50%; menyebabkan lebar gambar menjadi 50% dari elemen yang terkandung (bukan 50% dari tampilan yang terlihat atau 50% dari ukuran piksel yang sebenarnya).

Karena CSS memungkinkan materi meluap dari kontainernya, Anda mungkin perlu menggunakan max-width: 100% untuk mencegah gambar dan materi lainnya meluap. Misalnya :

```
img, embed, object, video {
  max-width: 100%;
}
```

Pastikan untuk memberikan keterangan penuh arti melalui atribut alt pada elemen img ; ini akan membantu membuat situs Anda lebih mudah diakses dengan memberikan konteks untuk pembaca layar dan teknologi pendukung lainnya.

Tingkatkan img dengan srcset untuk perangkat DPI tinggi

Atribut srcset meningkatkan perilaku elemen img, sehingga memudahkan saat memberikan beberapa file gambar untuk karakteristik perangkat yang berbeda. Serupa dengan image-set fungsi CSS bawaan dari CSS, srcset memungkinkan browser untuk memilih gambar terbaik bergantung pada karakteristik perangkat, misalnya menggunakan gambar 2x pada tampilan 2x, dan berpotensi di masa mendatang, gambar 1x pada perangkat 2x saat berada di jaringan bandwidth yang terbatas.

```
<img src="photo.png" srcset="photo@2x.png 2x" ...>
```

Pada browser yang tidak mendukung srcset, browser hanya menggunakan file gambar default yang ditentukan oleh atribut src. Inilah sebabnya mengapa penting untuk selalu menyertakan gambar 1x yang bisa ditampilkan pada perangkat apa pun, terlepas dari kemampuannya. Ketika srcset didukung, daftar yang dipisahkan koma dari gambar/kondisi di-parse sebelum membuat permintaan, dan hanya gambar paling sesuai yang diunduh dan ditampilkan.

Meskipun ketentuan bisa berisi segala sesuatu dari kepadatan piksel hingga lebar dan tinggi, hanya kepadatan piksel yang didukung dengan baik pada saat ini. Untuk menyeimbangkan perilaku saat ini dengan fitur masa mendatang, bertahanlah dengan hanya menyediakan gambar 2x di atribut.

Tujuan seni dalam gambar responsif dengan picture

Untuk mengubah gambar berdasarkan karakteristik perangkat, juga dikenal sebagai tujuan seni, gunakan elemen picture. Elemen picture mendefinisikan solusi deklaratif untuk menyediakan beberapa versi dari sebuah gambar berdasarkan karakteristik yang berbeda, seperti ukuran perangkat, resolusi perangkat, orientasi, dan lainnya.

Dogfood: Elemen **picture** mulai mendarat di browser. Meskipun belum tersedia di semua browser, kami merekomendasikan penggunaannya karena kompatibilitas mundur yang kuat dan potensi penggunaan Picturefill polyfill. Lihat situs ResponsiveImages.org untuk lebih jelasnya.

Gunakan elemen picture ketika sumber gambar terdapat di beberapa kepadatan, atau ketika desain responsif menentukan gambar yang agak berbeda pada beberapa jenis layar. Serupa dengan elemen video, beberapa elemen source bisa dimasukkan, sehingga memungkinkan untuk menentukan file gambar yang berbeda bergantung pada kueri media atau format gambar.

```
<picture>
  <source media="(min-width: 800px)" srcset="head.jpg, head-2x.jpg 2x">
  <source media="(min-width: 450px)" srcset="head-small.jpg, head-small-2x.jpg 2x">
  <img src="head-fb.jpg" srcset="head-fb-2x.jpg 2x" alt="a head carved out of wood">
</picture>
```

Cobalah

Pada contoh di atas, jika lebar browser setidaknya 800 px, maka salah satu dari head.jpg atau head-2x.jpg akan digunakan, bergantung pada resolusi perangkat. Jika lebar browser antara 450 px dan 800 px, maka head-small.jpg atau head-small-2x.jpg akan digunakan, bergantung pada resolusi perangkat. Untuk lebar layar kurang dari 450 px dan

kompatibilitas mundur dengan elemen picture tidak didukung, browser merender elemen img sebagai gantinya, dan harus selalu disertakan.

Gambar ukuran relatif

Ketika ukuran akhir gambar tidak diketahui, bisa sulit untuk menentukan deskriptor kepadatan bagi sumber gambar. Hal ini terutama berlaku untuk gambar yang membentang seimbang dengan lebar browser dan bisa berubah-ubah, tergantung pada ukuran browser.

Alih-alih menyediakan ukuran gambar dan kepadatan tetap, Anda bisa menetapkan ukuran masing-masing gambar yang disediakan dengan menambahkan deskriptor lebar bersama dengan ukuran elemen gambar, yang memungkinkan browser untuk secara otomatis menghitung kepadatan piksel efektif dan memilih gambar terbaik untuk diunduh.

```
<img src="lighthouse-200.jpg" sizes="50vw"
    srcset="lighthouse-100.jpg 100w, lighthouse-200.jpg 200w,
            lighthouse-400.jpg 400w, lighthouse-800.jpg 800w,
            lighthouse-1000.jpg 1000w, lighthouse-1400.jpg 1400w,
            lighthouse-1800.jpg 1800w" alt="a lighthouse">
```

Cobalah

Contoh di atas merender sebuah gambar yang berukuran setengah lebar tampilan yang terlihat (sizes="50vw"), dan bergantung pada lebar browser dan rasio piksel perangkat, yang memungkinkan browser memilih gambar yang tepat terlepas dari seberapa besar jendela browsernya. Misalnya, tabel di bawah ini menunjukkan gambar mana yang akan dipilih browser:

Lebar browser	Rasio piksel perangkat	Gambar yang digunakan	Resolusi efektif
400 px	1	200.png	1x
400 px	2	400.png	2x
320 px	2	400.png	2,5x
600 px	2	800.png	2,67x

Lebar browser	Rasio piksel perangkat	Gambar yang digunakan	Resolusi efektif
640 px	3	1000.png	3,125x
1100 px	1	1400.png	1,27x

Memperhitungkan breakpoint dalam gambar responsif

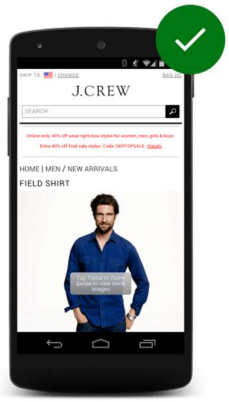
Dalam banyak kasus, ukuran gambar bisa berubah bergantung pada breakpoint layout situs. Misalnya, pada layar kecil, Anda mungkin menginginkan agar gambar membentang penuh pada tampilan yang terlihat, sementara di layar yang lebih besar, itu hanya menggunakan sebagian kecilnya.

```
<img src="400.png"
    sizes="(min-width: 600px) 25vw, (min-width: 500px) 50vw, 100vw"
    srcset="100.png 100w, 200.png 200w, 400.png 400w,
          800.png 800w, 1600.png 1600w, 2000.png 2000w" alt="an example image">
```

Cobalah

Atribut `sizes` pada contoh di atas, menggunakan beberapa kueri media untuk menentukan ukuran gambar. Ketika lebar browser lebih besar dari 600 px, gambar berukuran 25% dari lebar tampilan yang terlihat, saat lebarnya antara 500 px dan 600 px, gambar berukuran 50% dari lebar tampilan yang terlihat, dan saat lebarnya di bawah 500 px, lebarnya maksimal.

Membuat gambar produk yang bisa diperbesar



Situs web J. Crews dengan gambar produk yang diperluas.

Konsumen ingin melihat apa yang mereka beli. Pada situs ritel, pengguna berharap untuk bisa melihat tampilan-dekat resolusi tinggi dari produk agar bisa melihatnya secara lebih detail, dan partisipan penelitian merasa frustrasi jika mereka tidak dapat melakukannya.

Sebuah contoh bagus dari gambar yang bisa diketuk dan diperbesar disediakan oleh situs J. Crew. Sebuah overlay yang menghilang menunjukkan bahwa gambar bisa diketuk, yang memunculkan gambar yang diperbesar dengan detail halus terlihat.

Teknik gambar lainnya

Gambar kompresif

Teknik gambar kompresif menyajikan gambar 2x kompresi sangat tinggi untuk semua perangkat, tidak peduli kemampuan sebenarnya dari perangkat tersebut. Tergantung pada tipe gambar dan tingkat kompresi, kualitas gambar mungkin tidak terlihat berubah, namun ukuran file turun secara signifikan.

Cobalah

Perhatian: Gunakan teknik kompresi dengan hati-hati karena peningkatan biaya decoding dan memori yang diperlukan. Mengubah ukuran gambar besar agar muat pada layar yang lebih kecil tidak mudah dilakukan dan bisa sangat merugikan pada perangkat low-end karena memori dan kemampuan prosesor terbatas.

Pengganti gambar JavaScript

Pengganti gambar JavaScript memeriksa kemampuan perangkat dan "melakukan hal yang benar." Anda bisa menentukan rasio piksel perangkat melalui window.devicePixelRatio, memperoleh lebar dan tinggi layar, dan bahkan berpotensi melakukan beberapa sniffing koneksi jaringan melalui navigator.connection atau mengeluarkan permintaan palsu. Bila telah mengumpulkan semua informasi ini, Anda bisa memutuskan gambar mana yang akan dimuat.

Salah satu kelemahan besar dalam pendekatan ini adalah bahwa menggunakan JavaScript berarti bahwa Anda akan menunda pemuatan gambar sampai setidaknya parser lihat-depan telah diselesaikan. Bahkan ini berarti bahwa gambar tidak akan mulai diunduh sampai kejadian pageload sudah diaktifkan. Selain itu, browser kemungkinan besar akan mengunduh gambar 1x dan 2x, yang mengakibatkan peningkatan ukuran laman.

Menyisipkan gambar: bitmap dan vektor

Ada dua cara yang berbeda secara mendasar untuk membuat dan menyimpan gambar—dan ini memengaruhi bagaimana Anda menerapkan gambar secara responsif.

Gambar bitmap—seperti foto dan gambar lainnya—direpresentasikan sebagai grid dari titik-titik individu warna. Gambar bitmap mungkin berasal dari kamera atau pemindai, atau dibuat dengan elemen kanvas HTML. Format seperti PNG, JPEG dan WebP digunakan untuk menyimpan gambar bitmap.

Gambar vektor— seperti logo dan seni garis—didefinisikan sebagai serangkaian rangkaian kurva, garis, bentuk, warna isian dan gradien. Gambar vektor bisa dibuat dengan program seperti Adobe Illustrator atau Inkscape, atau ditulis tangan dalam kode menggunakan format vektor seperti SVG.

SVG

SVG memungkinkan untuk memasukkan grafis vektor responsif dalam laman web. Keuntungan dari format file vektor dibandingkan format file bitmap adalah bahwa browser bisa merender gambar vektor dalam ukuran apa saja. Format vektor menggambarkan geometri gambar—bagaimana itu dibuat dari garis, kurva, warna dan sebagainya. Format bitmap, di sisi lain, hanya memiliki informasi tentang titik-titik individu warna, sehingga browser harus menebak cara mengisi kekosongan saat penskalaan.

Di bawah ini adalah dua versi dari gambar yang sama: gambar PNG di sebelah kiri dan SVG di sebelah kanan. SVG terlihat bagus pada berbagai ukuran, sedangkan PNG di sebelahnya mulai terlihat buram pada ukuran layar yang lebih besar.

Jika Anda ingin mengurangi jumlah permintaan file yang dibuat laman, Anda bisa mengkodekan gambar menjadi inline menggunakan format Data URI atau SVG. Jika Anda melihat sumber dari laman ini, Anda akan melihat bahwa kedua logo berikut dideklarasikan inline: URI Data dan SVG.

SVG memiliki dukungan yang luar biasa pada seluler dan desktop, dan alat optimalisasi bisa secara signifikan mengurangi ukuran SVG. Dua logo SVG inline berikut terlihat sama, namun yang satu berukuran sekitar 3 KB dan lainnya hanya 2KB:

Data URI

Data URI menyediakan cara untuk menyertakan file, seperti gambar, inline dengan menetapkan src dari elemen img sebagai string mengenkode Base64 menggunakan format berikut:

```
<img src="[data]">
```

Awal kode untuk logo HTML5 di atas terlihat seperti ini:

```
<img
src="
LTgiPz4NCjwhLS0gR2VuZXJhdG9yOiB
BZG9iZSBJbGx1c3RyYXRvciAxNi4wLjAsIFNWRyBFeHBvcnQgUGx1Zy1JbiAuIFNWRy
BWZXJzaW ...">
```

(Versi lengkap lebih dari 5000 karakter panjangnya!)

Alat (bantu) seret dan lepas seperti jpillora.com/base64-encoder tersedia untuk mengonversi file biner seperti gambar ke Data URI. Sama seperti SVG, Data URI didukung dengan baik pada browser seluler dan desktop.

Penyisipan di CSS

URI Data dan SVG juga bisa disisipkan dalam CSS—dan ini didukung pada perangkat seluler dan desktop. Berikut adalah dua gambar serupa yang diimplementasikan sebagai gambar latar di CSS; sebuah URI Data, sebuah SVG:

Menyisipkan kelebihan & kekurangan

Kode inline untuk gambar bisa bertele-tele—terutama URI Data—jadi mengapa Anda mau menggunakannya? Untuk mengurangi permintaan HTTP! SVG dan Data URI bisa mengaktifkan seluruh laman web, termasuk gambar, CSS dan JavaScript, yang dapat diambil dengan satu permintaan.

Sisi negatifnya:

- Pada perangkat seluler, Data URI bisa secara signifikan lebih lambat untuk ditampilkan pada perangkat seluler daripada gambar dari `src` eksternal.
- Data URI bisa cukup banyak meningkatkan ukuran permintaan HTML.
- Mereka menambahkan kompleksitas pada markup dan alur kerja Anda.
- Format Data URI jauh lebih besar dari biner (hingga 30%) dan oleh karena itu tidak mengurangi jumlah ukuran unduhan.
- Data URI tidak bisa di-cache, sehingga harus diunduh untuk setiap laman yang menggunakannya.
- Mereka tidak didukung dalam IE 6 dan 7, dukungan tidak lengkap dalam IE8.
- Dengan HTTP/2, mengurangi jumlah permintaan aset akan menjadi berkurang prioritasnya.

Seperti pada segala sesuatu yang responsif, Anda harus menguji apa yang terbaik. Gunakan alat developer untuk mengukur ukuran file unduhan, jumlah permintaan, dan jumlah latensi. URI Data kadang-kadang bisa berguna untuk gambar bitmap—misalnya, pada beranda yang hanya memiliki satu atau dua foto yang tidak digunakan di tempat lain. Jika Anda membutuhkan gambar vektor inline, SVG adalah pilihan yang jauh lebih baik.

Gambar di CSS

Properti `background` CSS adalah alat (bantu) yang efektif untuk menambahkan gambar kompleks ke elemen, memudahkan ketika ingin menambahkan beberapa gambar, membuat pengulangan, dan banyak lagi. Ketika dikombinasikan dengan kueri media, properti latar belakang menjadi lebih efektif lagi, memungkinkan pemuatan gambar bersyarat berdasarkan resolusi layar, ukuran tampilan yang terlihat, dan lainnya.

TL;DR

- Gunakan gambar terbaik untuk karakteristik tampilan, pertimbangkan ukuran layar, resolusi perangkat dan layout laman.
- Ubah properti `background-image` dalam CSS untuk tampilan DPI tinggi menggunakan kueri media dengan `min-resolution` dan `-webkit-min-device-pixel-ratio`.
- Gunakan srcset untuk memberikan gambar resolusi tinggi selain gambar 1x dalam markup.
- Pertimbangkan biaya kinerja ketika menggunakan teknik pengganti gambar JavaScript atau ketika menyajikan gambar resolusi tinggi sangat terkompresi untuk perangkat resolusi rendah.

Gunakan kueri media untuk pemuatan gambar bersyarat atau tujuan seni

Kueri media tidak hanya memengaruhi layout laman; Anda juga bisa menggunakannya untuk memuat gambar secara bersyarat atau memberikan tujuan seni bergantung pada lebar tampilan yang terlihat.

Misalnya dalam contoh di bawah ini, pada layar yang lebih kecil, hanya `small.png` yang diunduh dan diaplikasikan ke materi `div`, sementara di layar yang lebih besar `background-image: url(body.png)` diaplikasikan ke tubuh dan `background-image: url(large.png)` is applied to the content `div`.

```
.example {
  height: 400px;
  background-image: url(small.png);
  background-repeat: no-repeat;
  background-size: contain;
  background-position-x: center;
}
```

```
@media (min-width: 500px) {
  body {
    background-image: url(body.png);
  }
  .example {
    background-image: url(large.png);
  }
}
```

Cobalah

Gunakan image-set untuk memberikan gambar resolusi tinggi

Fungsi image-set() dalam CSS meningkatkan perilaku properti background, sehingga memudahkan saat memberikan beberapa file gambar untuk karakteristik perangkat yang berbeda. Hal ini memungkinkan browser untuk memilih gambar terbaik tergantung pada karakteristik perangkat, misalnya menggunakan gambar 2x pada tampilan 2x, atau gambar 1x pada perangkat 2x ketika berada pada jaringan bandwidth terbatas.

```
background-image: image-set(
  url(icon1x.jpg) 1x,
  url(icon2x.jpg) 2x
);
```

Selain memuat gambar yang benar, browser juga mengubah ukurannya secara sesuai. Dengan kata lain, browser berasumsi bahwa gambar 2x berukuran dua kali lebih besar dari gambar 1x, lalu menurunkan ukuran gambar 2x dengan faktor 2, sehingga gambar yang muncul mempunyai ukuran yang sama pada laman.

Dukungan untuk image-set() masih baru dan hanya didukung di Chrome dan Safari dengan awalan vendor -webkit. Berhati-hatilah saat menyertakan gambar fallback ketika image-set() tidak didukung; misalnya:

```
.sample {
  width: 128px;
  height: 128px;
  background-image: url(icon1x.png);
  background-image: -webkit-image-set(
    url(icon1x.png) 1x,
    url(icon2x.png) 2x
  );
  background-image: image-set(
    url(icon1x.png) 1x,
```

```
    url(icon2x.png) 2x
  );
}
```

Cobalah

Hal di atas memuat aset yang tepat di browser yang mendukung image-set; jika tidak akan kembali ke aset 1x. Kekurangan yang nyata adalah bahwa meskipun dukungan browser image-set() rendah, kebanyakan browser mendapatkan aset 1x.

Menggunakan kueri media untuk memberikan gambar resolusi tinggi atau tujuan seni

Kueri media bisa membuat aturan berdasarkan rasio piksel perangkat, sehingga bisa menentukan gambar yang berbeda untuk tampilan 2x versus 1x.

```
@media (min-resolution: 2dppx),
(-webkit-min-device-pixel-ratio: 2)
{
  /* High dpi styles & resources here */
}
```

Chrome, Firefox dan Opera semua mendukung (min-resolution: 2dppx) standar, sementara Safari dan browser Android keduanya membutuhkan sintaks berawalan vendor yang lebih lama tanpa unit dppx. Ingatlah, gaya ini hanya dimuat jika perangkat cocok dengan kueri media, dan Anda harus menetapkan gaya untuk kejadian dasar. Ini juga memberikan manfaat untuk memastikan sesuatu dirender jika browser tidak mendukung kueri media resolusi spesifik.

```
.sample {
  width: 128px;
  height: 128px;
  background-image: url(icon1x.png);
}

@media (min-resolution: 2dppx), /* Standard syntax */
(-webkit-min-device-pixel-ratio: 2)  /* Safari & Android Browser */
{
  .sample {
    background-size: contain;
    background-image: url(icon2x.png);
  }
}
```

Cobalah

Anda juga bisa menggunakan sintaks min-width untuk menampilkan gambar alternatif tergantung pada ukuran tampilan yang terlihat. Teknik ini memiliki keuntungan bahwa gambar tidak diunduh jika kueri media tidak sesuai. Misalnya, bg.png hanya diunduh dan diaplikasikan ke body jika lebar browser 500px atau lebih besar:

```
@media (min-width: 500px) {
  body {
    background-image: url(bg.png);
  }
}
```

Menggunakan SVG untuk ikon

Ketika menambahkan ikon ke laman Anda, gunakan ikon SVG jika memungkinkan atau dalam kasus tertentu, karakter unicode.

TL;DR

- Menggunakan SVG atau unicode sebagai ikon bukannya gambar bitmap.

Ganti ikon sederhana dengan unicode

Banyak font mengikutsertakan dukungan untuk berbagai karakter unicode, yang bisa digunakan sebagai pengganti gambar. Tidak seperti gambar, font unicode akan diskalakan dengan baik dan terlihat baik tidak peduli seberapa kecil atau besar mereka ditampilkan di layar.

Selain himpunan karakter normal, unicode mungkin memasukkan simbol untuk panah (←), operator matematika (√), bentuk geometris (★), gambar kontrol (▶), notasi musik (♫), huruf Yunani (Ω), bahkan bidak catur (♞).

Memasukkan karakter unicode dilakukan secara sama dengan memberi nama entitas: &#XXXX, dengan XXXX merepresentasikan angka karakter unicode. Misalnya:

```
You're a super &#9733;
```

You're a super ★

Ganti ikon kompleks dengan SVG

Untuk persyaratan ikon kompleks lainnya, ikon SVG biasanya ringan, mudah digunakan, dan bisa diberi gaya dengan CSS. SVG memiliki sejumlah keunggulan dibandingkan gambar bitmap:

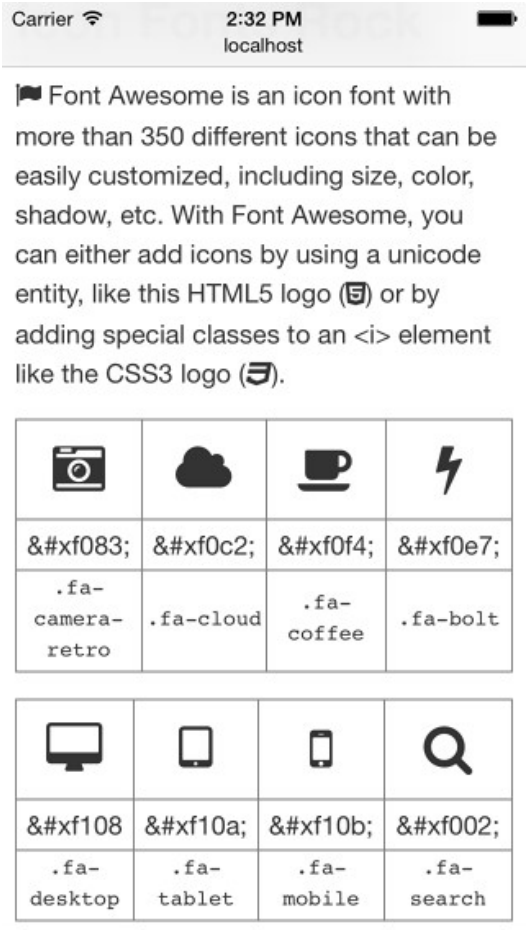
- Mereka adalah grafis vektor yang bisa diskalakan secara tak terbatas.
- Efek CSS seperti warna, bayangan, transparansi, dan animasi bisa dibuat dengan mudah.

- Gambar SVG bisa disisipkan langsung dalam dokumen.
- Mereka semantik.
- Ikon SVG menyediakan aksesibilitas yang lebih baik dengan atribut yang sesuai.

```
With SVG icons, you can either add icons using inline SVG, like
this checkmark:
 <svg version="1.1" xmlns="http://www.w3.org/2000/svg"
     xmlns:xlink="http://www.w3.org/1999/xlink"
     width="32" height="32" viewBox="0 0 32 32">
   <path d="M27 4l-15 15-7-7-5 5 12 12 20-20z" fill="#000000"></path>
 </svg>
or by using an image tag, like this credit card icon:
<img src="credit.svg">.
```

Cobalah

Gunakan font ikon dengan hati-hati



Contoh laman yang menggunakan FontAwesome untuk ikon font.

Font ikon memang populer, dan mudah digunakan, namun memiliki beberapa kekurangan jika dibandingkan dengan ikon SVG:

- Mereka grafis vektor yang bisa secara tak terbatas diskalakan, namun anti-alias mungkin menghasilkan ikon tidak setajam yang diharapkan.
- Penataan gaya terbatas dengan CSS.
- Pemosisian sempurna hingga tingkat piksel bisa sulit dilakukan, tergantung pada tinggi-baris, pengaturan jarak huruf, dll.
- Font ikon bukan semantik, dan sulit digunakan dengan pembaca layar atau teknologi bantu lainnya.
- Kecuali dengan benar tercakup, Font ikon bisa menghasilkan ukuran file yang besar hanya karena menggunakan subset kecil dari ikon yang tersedia.

```
With Font Awesome, you can either add icons by using a unicode
entity, like this HTML5 logo (<span class="awesome">&#xf13b;</span>)
or by adding special classes to an &lt;i&gt; element like the CSS3
logo (<i class="fa fa-css3"></i>).
```

Cobalah

Ada ratusan font ikon gratis dan berbayar yang tersedia termasuk Font Awesome, Pictos, dan Glyphicons.

Pastikan untuk menyeimbangkan bobot permintaan HTTP tambahan dan ukuran file dengan kebutuhan ikon. Misalnya, jika Anda hanya membutuhkan beberapa ikon, mungkin lebih baik untuk menggunakan gambar atau sprite gambar.

Mengoptimalkan gambar untuk kinerja

Gambar sering menjadi sumber besarnya byte yang diunduh dan juga sering kali menempati sejumlah besar ruang visual pada laman. Akibatnya, mengoptimalkan gambar bisa menghasilkan beberapa penghematan byte terbesar dan meningkatkan kinerja situs web Anda: semakin sedikit byte yang harus diunduh browser, semakin sedikit persaingan untuk mendapatkan bandwidth klien dan semakin cepat browser mengunduh dan menampilkan semua aset.

TL;DR

- Jangan hanya secara acak memilih format gambar—pahami format berbeda yang tersedia dan gunakan format yang paling cocok.
- Sertakan alat kompresi dan optimalisasi gambar ke dalam alur kerja Anda untuk mengurangi ukuran file.

- Kurangi jumlah permintaan http dengan menempatkan gambar yang sering digunakan ke dalam image sprites.
- Untuk mempercepat waktu muat laman awal dan mengurangi ukurannya, pertimbangkan memuat gambar hanya setelah mereka bergulir ke dalam tampilan.

Memilih format yang tepat

Ada dua tipe gambar yang bisa dipertimbangkan: gambar vektor dan gambar_bitmap. Untuk gambar bitmap, Anda juga harus memilih format kompresi yang tepat, misalnya: GIF, PNG, JPG.

Gambar bitmap, seperti foto dan gambar lainnya, direpresentasikan sebagai sebuah grid dari titik atau piksel individual. Gambar bitmap biasanya diperoleh dari kamera atau pemindai, atau bisa dibuat di browser dengan elemen canvas. Saat ukuran gambar semakin besar, begitu juga ukuran filenya. Ketika diskalakan dengan ukuran lebih besar dari aslinya, gambar bitmap menjadi buram karena browser harus menebak cara mengisi piksel yang hilang.

Gambar vektor, seperti logo dan seni garis, didefinisikan oleh rangkaian kurva, garis, bentuk dan warna isi. Gambar vektor dibuat dengan program seperti Adobe Illustrator atau Inkscape dan disimpan ke format vektor seperti SVG. Karena gambar vektor dibangun di atas konsep primitif sederhana, gambar tersebut bisa diskalakan tanpa penurunan kualitas atau perubahan ukuran file.

Ketika memilih format yang tepat, kita harus mempertimbangkan asal gambar (bitmap atau vektor), dan materi (warna, animasi, teks, dll). Tidak ada satu format yang bisa cocok untuk semua jenis gambar, dan masing-masing memiliki kelebihan dan kekurangan tersendiri.

Mulailah dengan panduan ini ketika memilih format yang tepat:

- Gunakan JPG untuk gambar fotografis.
- Gunakan SVG untuk seni vektor dan grafis warna solid seperti logo dan seni garis. Jika seni vektor tidak tersedia, coba gunakan WebP atau PNG.
- Gunakan PNG daripada GIF karena memberikan warna yang lebih banyak dan menawarkan rasio kompresi lebih baik.
- Untuk animasi yang lebih panjang pertimbangkan untuk menggunakan <video>, yang memberikan kualitas gambar lebih baik dan memberikan pengguna kontrol saat pemutaran.

Mengurangi ukuran file

Anda bisa memperkecil ukuran file gambar secara signifikan dengan melakukan "pascapemrosesan" gambar setelah penyimpanan. Ada beberapa alat untuk kompresi gambar—lossy dan lossless, online, GUI, baris perintah. Jika memungkinkan, akan lebih baik

mengotomatiskan optimalisasi gambar sehingga itu mendapat prioritas utama dalam alur kerja Anda.

Ada beberapa alat yang bisa melakukan kompresi lossless lebih baik pada file JPG dan PNG, tanpa memengaruhi kualitas gambar. Untuk JPG, cobalah jpegtran atau jpegoptim (hanya tersedia di Linux; jalankan dengan --menghilangkan-semua opsi). Untuk PNG, cobalah OptiPNG atau PNGOUT.

Gunakan image sprites

CSS spriting adalah sebuah teknik dengan sejumlah gambar digabungkan dalam satu gambar "sprite sheet". Anda kemudian bisa menggunakan setiap gambar tersebut dengan menetapkan gambar latar untuk elemen (sprite sheet) ditambah offset untuk menampilkan bagian yang benar.

```
.sprite-sheet {
  background-image: url(sprite-sheet.png);
  width: 40px;
  height: 25px;
}

.google-logo {
  width: 125px;
  height: 45px;
  background-position: -190px -170px;
}

.gmail {
```

```
  background-position: -150px -210px;
}

.maps {
  height: 40px;
  background-position: -120px -165px;
}
```

Cobalah

Spriting memiliki keuntungan yaitu mengurangi jumlah unduhan yang diperlukan untuk mendapatkan beberapa gambar, sambil tetap mengaktifkan caching.

Pertimbangkan lazy loading

Lazy loading bisa secara signifikan mempercepat pemuatan pada laman panjang yang memasukkan banyak gambar di paro bawah dengan memuatnya saat diperlukan atau ketika materi utama selesai dimuat dan dirender. Selain peningkatan kinerja, menggunakan lazy loading bisa menciptakan pengalaman gulir tak terbatas.

Hati-hati saat membuat laman gulir tak terbatas—karena materi dimuat ketika terlihat, mesin telusur mungkin tidak akan pernah melihat materi tersebut. Selain itu, pengguna yang mencari informasi yang mereka harapkan bisa dilihat di footer, tidak akan pernah melihat footer karena materi baru selalu dimuat.

Jangan gunakan gambar sama sekali

Terkadang, gambar terbaik sama sekali bukanlah sebuah gambar. Bila memungkinkan, gunakan kemampuan bawaan browser untuk menyediakan fungsionalitas yang sama atau serupa. Browser menghasilkan visual gambar yang diperlukan sebelumnya. Ini berarti bahwa browser tidak perlu lagi mengunduh file gambar yang terpisah sehingga mencegah gambar diskalakan dengan canggung. Anda bisa menggunakan unicode atau font ikon khusus untuk merender ikon.

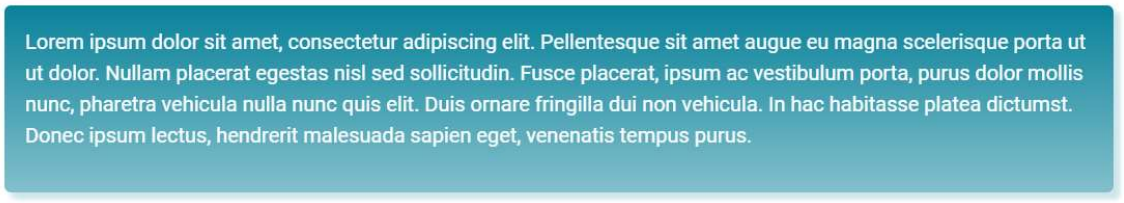
Tempatkan teks dalam markup bukannya disematkan pada gambar

Bila memungkinkan, teks harus berupa teks dan tidak disematkan ke dalam gambar. Misalnya, menggunakan gambar sebagai judul atau menempatkan informasi kontak—seperti nomor telepon atau alamat—secara langsung pada gambar untuk mencegah pengguna menyalin dan menempelkan informasi; hal ini membuat informasi tidak bisa diakses pembaca layar, dan tidak responsif. Sebagai gantinya, letakkan teks dalam markup Anda dan jika perlu gunakan webfonts untuk memperoleh gaya yang Anda butuhkan.

Gunakan CSS sebagai pengganti gambar

Browser modern bisa menggunakan fitur CSS untuk menciptakan gaya yang sebelumnya memerlukan gambar. Misalnya: gradien kompleks bisa dibuat dengan menggunakan

properti background, bayangan dapat dibuat dengan menggunakan box-shadow, dan sudut membulat bisa ditambahkan dengan properti border-radius.




```
<style>
 div#noImage {
  color: white;
  border-radius: 5px;
  box-shadow: 5px 5px 4px 0 rgba(9,130,154,0.2);
  background: linear-gradient(rgba(9, 130, 154, 1), rgba(9, 130, 154, 0.5));
 }
</style>
```

Harap diingat bahwa menggunakan teknik ini tidak memerlukan siklus rendering, yang bisa cukup signifikan pada perangkat seluler. Jika digunakan berlebihan, Anda akan kehilangan manfaat yang mungkin telah didapatkan dan mungkin menghambat kinerja.

9.4 Materi Multi-Perangkat

Bagaimana orang membaca di web

Panduan penulisan pemerintah AS merangkum apa yang orang inginkan dari menulis di web:

> Ketika menulis untuk web, menggunakan bahasa sederhana yang memungkinkan pengguna untuk menemukan apa yang mereka butuhkan, memahami apa yang mereka temukan, dan kemudian menggunakannya sesuai dengan kebutuhan.
>
> Ini juga harus bisa ditindaklanjuti, bisa ditemukan, dan bisa dibagikan.

Penelitian menunjukkan bahwa orang tidak membaca laman web, tetapi mereka memindainya. Rata-rata, orang hanya membaca 20-28% dari materi laman web. Membaca di layar jauh lebih lambat dibandingkan membaca di kertas. Orang akan bosan dan meninggalkan situs Anda kecuali informasi mudah untuk diakses dan dipahami.

Cara menulis untuk perangkat seluler

Fokus pada subjek inti dan langsung ceritakan. Agar tulisan bekerja optimal di berbagai perangkat dan tampilan yang terlihat, pastikan untuk menempatkan inti tulisan di awal: sebagai pedoman, idealnya dalam empat paragraf pertama, sekitar 70 kata.

Tanyakan pada diri sendiri apa yang orang inginkan dari situs Anda. Apakah mereka mencoba untuk menemukan sesuatu? Jika orang mengunjungi situs Anda untuk suatu informasi, pastikan bahwa semua teks berorientasi untuk membantu mereka mencapai tujuan. Menulis dalam aktif voice, menawarkan tindakan dan solusi.

Hanya mempublikasikan apa yang diinginkan pengunjung, tidak lebih.

Penelitian pemerintah UK juga menunjukkan bahwa:

> 80% orang lebih suka kalimat yang ditulis dalam bahasa Inggris yang jelas — dan semakin kompleks masalahnya, semakin besar preferensi-nya (mis., 97% memilih "among other things" dibandingkan bahasa Latin "inter alia").
>
> Semakin tinggi tingkat pendidikan orang dan lebih spesialis pengetahuannya, semakin besar preferensi mereka untuk bahasa Inggris sederhana.

Dengan kata lain: gunakan bahasa sederhana, kata-kata pendek dan struktur kalimat sederhana — bahkan untuk pengguna teknis dan terpelajar. Jagalah nada suara agar selalu enak didengarkan, kecuali ada alasan untuk tidak melakukannya. Aturan lawas jurnalisme adalah menulis seolah-olah Anda sedang berbicara dengan seorang anak yang cerdas berusia 11 tahun.

Miliaran pengguna berikutnya

Pendekatan pared-down untuk penulisan sangat penting bagi pembaca di perangkat seluler, serta sangat penting ketika membuat materi untuk ponsel murah dengan tampilan yang terlihat yang kecil dan memerlukan lebih banyak tindakan gulir dan mungkin memiliki kualitas layar yang lebih rendah dan kurang responsif.

Sebagian besar miliaran pengguna berikutnya akan online menggunakan perangkat murah. Mereka tidak ingin menghabiskan kuota data dengan menavigasi materi yang bertele-tele, dan mungkin tidak membacanya dalam bahasa utama mereka. Pangkas teks Anda: gunakan kalimat pendek, tanda baca minimal, paragraf lima baris atau kurang, dan judul satu baris. Pertimbangkan teks responsif (misalnya, menggunakan kepala berita pendek untuk tampilan yang terlihat yang lebih kecil) tapi waspadalah terhadap kerugiannya.

Perilaku minimalis untuk teks juga akan membuat materi Anda lebih mudah dilokalkan dan internasionalisasi — dan semakin besar kemungkinan materi Anda akan dikutip di media sosial.

Intinya:

- Buatlah tetap sederhana
- Kurangi kesemrawutan
- Langsung ke intinya

Menghapus materi yang tidak perlu

Dari segi ukuran byte, laman web menjadi besar dan semakin besar.

Teknik desain responsif memungkinkan untuk menyajikan materi berbeda dalam tampilan yang terlihat yang lebih kecil, tapi akan lebih bijak jika memulai dengan merampingkan teks, gambar dan materi lainnya.

> Pengguna web sering kali berorientasi tindakan, cenderung aktif memburu jawaban atas pertanyaan mereka saat ini, daripada diam mempelajari buku yang bagus.
>
> — Jakob Nielsen

Tanyakan pada diri sendiri: apa yang orang ingin dapatkan ketika mereka mengunjungi situs saya?

Apakah setiap komponen laman membantu pengguna mencapai tujuan mereka?

Menghapus elemen laman yang berlebihan

File HTML terbentuk dari hampir 70k dan lebih dari sembilan permintaan untuk rata-rata laman web, menurut Arsip HTTP.

Banyak situs populer menggunakan beberapa ribu elemen HTML per laman, dan beberapa ribu baris kode, bahkan di perangkat seluler. Ukuran file HTML yang terlalu besar mungkin tidak memperlambat pemuatan laman, namun payload HTML yang berat bisa menandakan materi yang gembung: file .html yang lebih besar berarti lebih banyak elemen, materi teks, atau keduanya.

Mengurangi kompleksitas HTML juga akan mengurangi besar laman, membantu pelokalan serta internasionalisasi dan membuat desain responsif agar lebih mudah untuk direncanakan dan di-debug. Untuk informasi tentang cara menulis HTML yang lebih efisien, lihat HTML berkinerja tinggi.

> Setiap langkah tambahan yang dilakukan pengguna sebelum mereka mendapatkan nilai dari aplikasi, akan membuat Anda dikenai penalti 20% dari pengguna
>
> — Gabor Cselle, Twitter

Hal yang sama berlaku untuk materi: bantu pengguna mendapatkan apa yang mereka inginkan secepat mungkin.

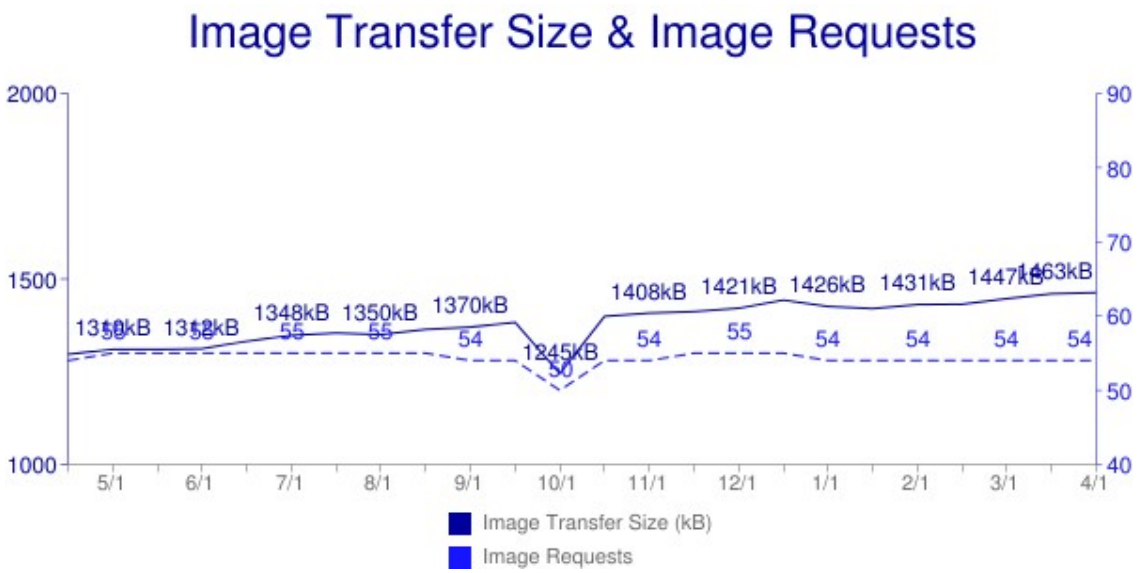
Jangan sekadar menyembunyikan materi dari pengguna seluler. Arahkan ke paritas materi, karena Anda tidak akan bisa menebak apa fitur seluler yang diinginkan pengguna. Jika Anda memiliki sumber daya, buat versi alternatif dari materi yang sama untuk ukuran tampilan yang terlihat berbeda — bahkan jika hanya bagi elemen laman dengan prioritas tinggi.

Perhatikan pengelolaan materi dan alur kerja: sistem lawas menghasilkan materi lawas?

Menyederhanakan teks

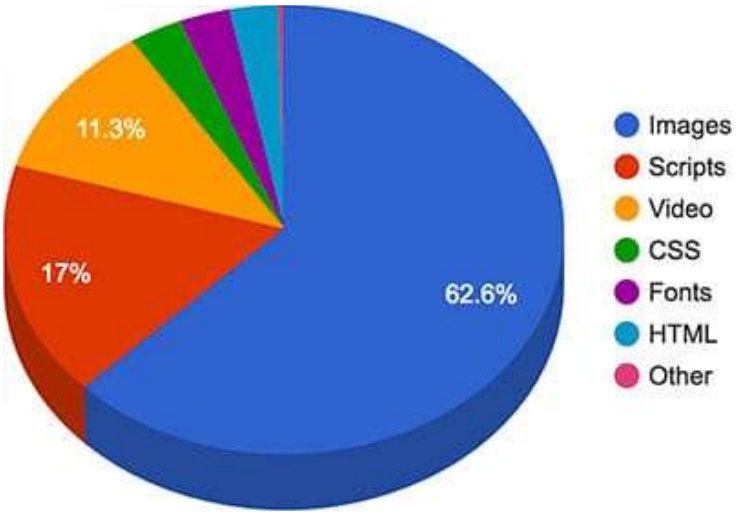
Karena web semakin populer di perangkat seluler, Anda harus mengubah cara Anda menulis. Buatlah tetap sederhana, kurangi kesemrawutan dan langsung ke intinya.

Membuang gambar yang tidak penting



Menurut data Arsip HTTP, laman web rata-rata membuat 54 permintaan gambar.

Gambar bisa indah, menyenangkan dan informatif — namun gambar juga menggunakan properti laman, menambah ukuran laman, dan meningkatkan jumlah permintaan file. Latensi semakin buruk ketika konektivitas memburuk, yang berarti bahwa permintaan file gambar yang berlebihan adalah masalah yang semakin meningkat ketika web digunakan di perangkat seluler.



Gambar membentuk lebih dari 60% berat laman.

Gambar juga mengonsumsi daya. Setelah layar, radio adalah fitur terbesar kedua yang paling menguras baterai. Semakin banyak permintaan gambar, semakin sering penggunaan radio, maka baterai akan semakin boros. Bahkan untuk merender gambar saja membutuhkan daya – dan ini sesuai dengan ukuran dan jumlahnya. Lihat laporan Stanford Apa yang Menghabiskan Baterai Saya?

Jika bisa, singkirkan gambarnya!

Berikut adalah beberapa sarannya:

- Pertimbangkan desain yang tidak menggunakan gambar sama sekali, atau menggunakan gambar secukupnya. Teks-saja bisa menjadi indah! Tanyakan pada diri sendiri, "Apa yang ingin dicapai pengunjung ke situs saya? Apakah gambar membantu proses tersebut? "
- Di masa lalu, merupakan hal yang biasa untuk menyimpan judul dan teks lain dalam bentuk grafis. Pendekatan tersebut tidak merespons dengan baik untuk perubahan ukuran tampilan yang terlihat, serta menambah latensi dan ukuran laman. Menggunakan teks dalam bentuk grafis juga berarti teks tidak bisa ditemukan oleh mesin telusur, dan tidak dapat diakses oleh alat pembaca layar dan teknologi pendukung lainnya. Gunakan teks "sungguhan" apabila memungkinkan - Web Fonts dan CSS dapat memberikan tipografi yang indah.
- Daripada gambar, gunakanlah CSS untuk gradien, bayangan, sudut lengkung, dan tekstur latar belakang, fitur didukung oleh semua browser modern. Namun harap diingat, bahwa CSS mungkin lebih baik dari gambar, tapi tetap ada penalti render dan pemrosesan, yang signifikan terutama di perangkat seluler.
- Gambar latar jarang sekali bekerja dengan baik di perangkat seluler. Anda bisa menggunakan kueri media untuk menghindari gambar latar pada tampilan yang terlihat yang kecil.
- Hindari gambar layar pembuka.
- Gunakan CSS untuk animasi UI.
- Kenali glyph Anda; gunakan ikon dan simbol Unicode bukan gambar, dengan Web Fonts bila perlu.
- Pertimbangkan font ikon; mereka adalah grafis vektor yang bisa diskalakan tanpa batas, dan seluruh kumpulan gambar bisa di unduh dalam satu font. (Namun, ketahui persoalan ini.)
- Elemen `<canvas>` bisa digunakan untuk membuat gambar di JavaScript dari garis, kurva, teks, dan gambar lainnya.
- Gambar URI Data atau SVG inline tidak akan mengurangi ukuran laman, namun mereka bisa mengurangi latensi dengan mengurangi jumlah permintaan sumber daya. SVG inline memiliki dukungan yang baik pada browser seluler dan desktop, dan alat optimalisasi bisa secara signifikan mengurangi ukuran SVG. Demikian juga, URI Data didukung dengan baik. Keduanya bisa disisipkan di CSS.
- Pertimbangkan menggunakan `<video>` bukannya animasi GIF. Elemen video ini didukung oleh semua browser di perangkat seluler (kecuali Opera Mini).

Untuk informasi selengkapnya, silakan lihat Optimalisasi Gambar serta Menghilangkan dan mengganti gambar.

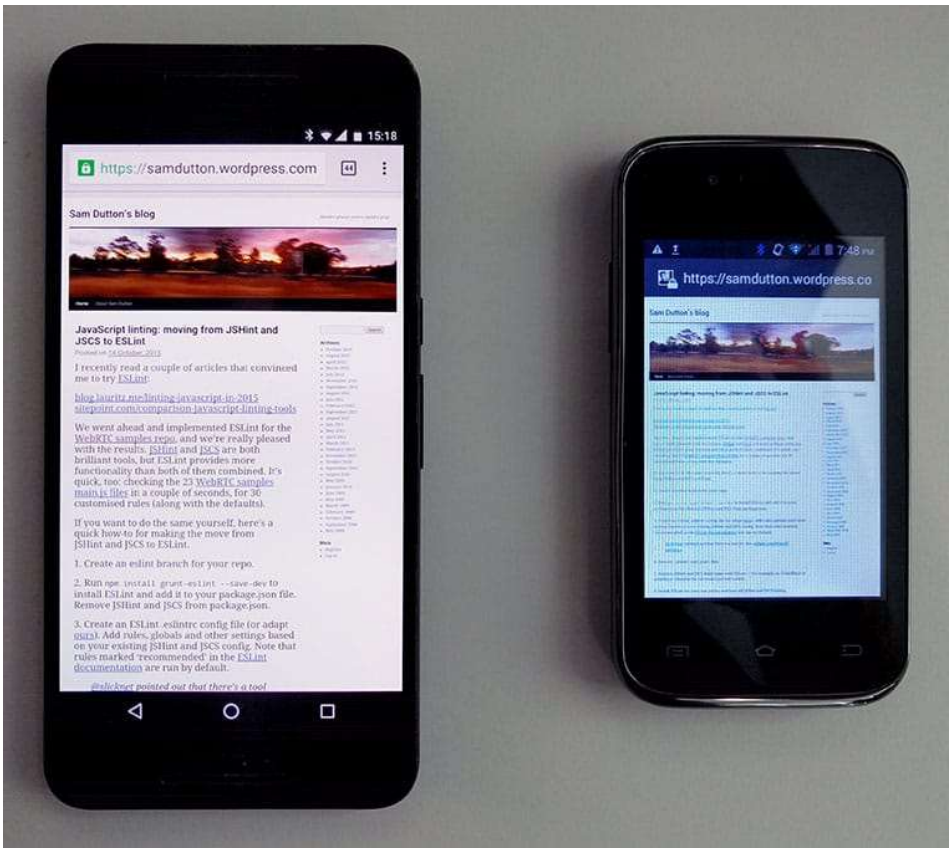
Merancang materi agar bekerja dengan baik pada ukuran tampilan yang terlihat berbeda

> "Create a product, don't re-imagine one for small screens. Great mobile products are created, never ported."
>
> — Mobile Design and Development, Brian Fling

"Perancang hebat tidak hanya "mengoptimalkan untuk perangkat seluler" — mereka berpikir tanggap untuk membangun situs yang bekerja di berbagai perangkat. Struktur teks dan materi laman lainnya sangat penting untuk keberhasilan lintas-perangkat.

Banyak dari semiliar pengguna berikutnya yang datang online menggunakan perangkat murah dengan tampilan yang terlihat yang kecil. Membaca dengan resolusi rendah di layar 3,5" atau 4" bukanlah pekerjaan yang mudah.

Berikut adalah foto dari keduanya:



Pada layar yang lebih besar, teks berukuran kecil tapi terbaca.

Pada layar yang lebih kecil, browser merender layout dengan benar, namun teks tidak terbaca, bahkan ketika diperbesar. Layar terlihat buram dan memiliki 'color cast' — warna putih tidak terlihat putih — sehingga materi kurang terbaca.

Mendesain materi untuk perangkat seluler

Ketika membangun untuk berbagai tampilan yang terlihat, pertimbangkan materi serta layout dan desain grafis, rancang dengan teks dan gambar nyata, jangan hanya materi dummy.

> "Materi mengawali desain. Desain tanpa materi bukanlah desain, tetapi hanya dekorasi."
>
> — Jeffrey Zeldman

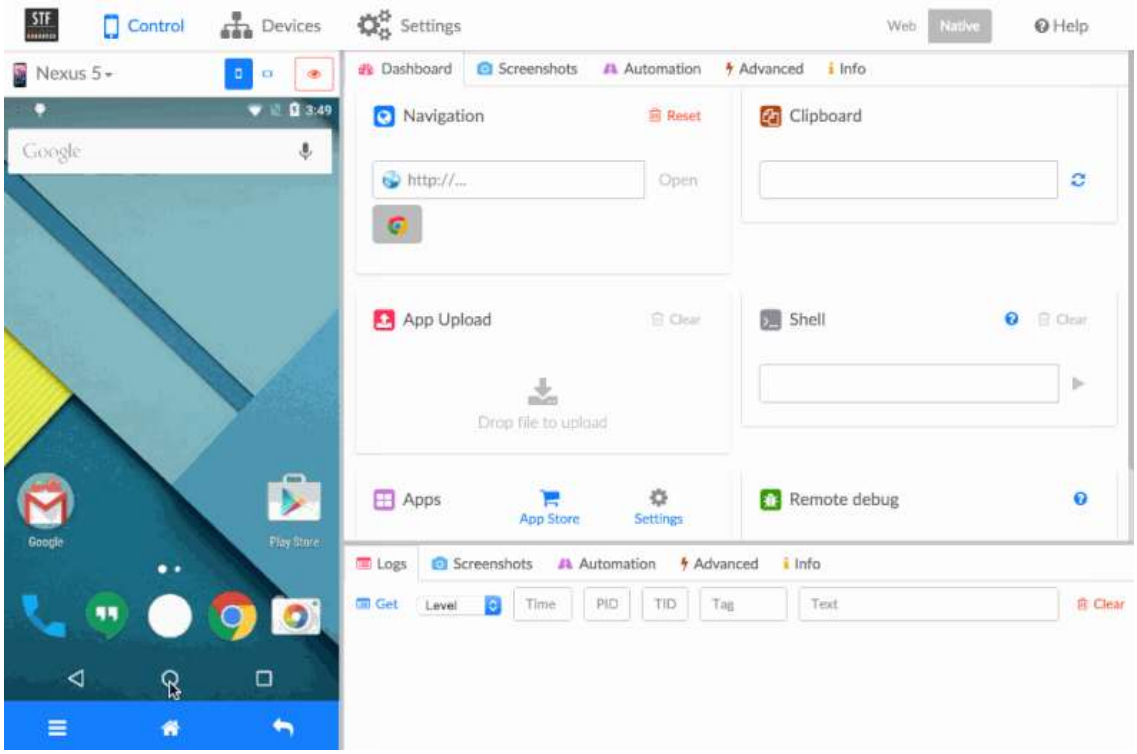
- Tempatkan materi Anda yang paling penting di atas, karena pengguna cenderung membaca laman web dalam pola berbentuk-F.
- Pengguna mengunjungi situs Anda untuk mencapai sebuah tujuan. Tanyakan pada diri sendiri apa yang mereka butuhkan untuk mencapai tujuan tersebut dan buang segala sesuatu yang lain. Lakukan tindakan tegas pada hiasan visual dan tekstual, materi lawas, tautan terlalu banyak, dan kesemrawutan lainnya.
- Hati-hati dengan ikon berbagi sosial; mereka bisa mengacaukan layout, dan kodenya dapat memperlambat pemuatan laman.
- Desain layout responsif untuk materi, bukan ukuran perangkat tetap.

Menguji materi

Berhasil: Apa pun yang Anda lakukan — **uji**!

- Periksa keterbacaan pada tampilan yang terlihat yang lebih kecil menggunakan Chrome DevTools dan alat emulasi lainnya.
- Uji materi di bawah kondisi bandwidth rendah dan latensi tinggi; cobalah materi dalam berbagai skenario konektivitas.
- Cobalah baca dan berinteraksi dengan materi Anda pada ponsel murah.
- Ajak teman dan kolega untuk mencoba aplikasi atau situs Anda.
- Membangun lab uji perangkat sederhana. GitHub repo untuk Mini Mobile Device Lab Google memiliki petunjuk tentang cara membangun lab sendiri. OpenSTF adalah aplikasi web sederhana untuk menguji situs web di beberapa perangkat Android.

Berikut adalah OpenSTF sedang beraksi:



Perangkat seluler semakin banyak dipakai untuk menggunakan materi dan memperoleh informasi — bukan hanya sebagai perangkat komunikasi, game dan media.

Hal ini membuat perencanaan materi semakin penting agar bisa bekerja dengan baik pada berbagai tampilan yang terlihat, dan untuk memprioritaskan materi ketika mempertimbangkan layout, antarmuka dan desain interaksi lintas-perangkat.

Memahami biaya data

Laman web semakin besar. Menurut Arsip HTTP, ukuran rata-rata laman untuk sejuta situs teratas sekarang melampaui 2 MB.

Pengguna menghindari situs atau aplikasi yang dianggap lambat atau mahal biaya datanya, jadi penting untuk memahami biaya pemuatan laman dan komponen aplikasi.

Mengurangi ukuran laman juga menguntungkan. Chris Zacharias dari YouTube menemukan hal tersebut ketika mereka mengurangi ukuran laman-tontonan dari 1,2 MB ke 250 KB:

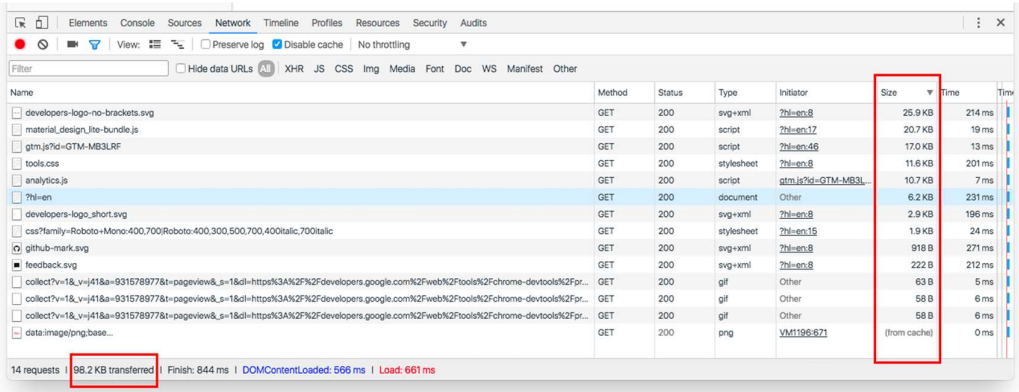
> Banyak orang yang sebelumnya tidak menggunakan YouTube, tiba-tiba memakainya.

Dengan kata lain, mengurangi ukuran laman **bisa membuka pasar baru**.

Menghitung ukuran laman

Ada beberapa alat untuk menghitung ukuran laman. Panel Network pada Chrome DevTools menunjukkan ukuran byte total untuk semua sumber daya, dan bisa digunakan untuk

memastikan ukuran untuk tipe aset individual. Anda juga bisa memeriksa item mana yang telah diambil dari cache browser.



Firefox dan browser lainnya menawarkan alat serupa.

WebPagetest menyediakan kemampuan untuk menguji pemuatan laman pertama dan berikutnya. Anda bahkan bisa mengotomatiskan pengujian dengan menggunakan skrip (misalnya, untuk masuk ke situs) atau menggunakan RESTful API. Contoh berikut (memuat developers.google.com/web) menunjukkan bahwa proses cache telah berhasil dan pemuatan laman berikutnya tidak membutuhkan sumber daya tambahan.

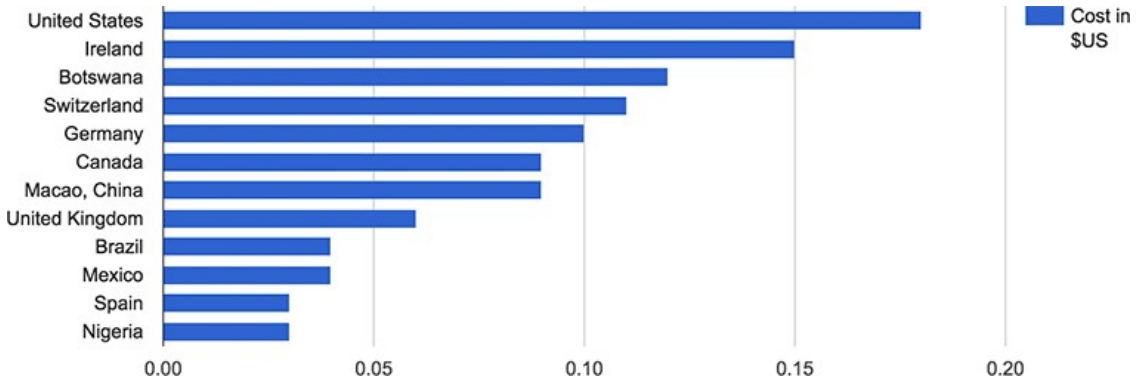
						Document Complete			Fully Loaded			
	Load Time	First Byte	Start Render	Speed Index	DOM Elements	Time	Requests	Bytes In	Time	Requests	Bytes In	Cost
First View	11.950s	7.932s	1.149s	7867	404	11.950s	31	495 KB	12.023s	33	501 KB	$$---
Repeat View	3.580s	3.233s	0.829s	895	404	3.580s	3	0 KB	3.501s	3	0 KB	

WebPagetest juga memberikan ukuran dan meminta rincian berdasarkan tipe MIME.

Menghitung biaya laman

Bagi banyak pengguna, biaya data tidak hanya berupa byte dan kinerja — namun juga uang.

Situs What Does My Site Cost? memungkinkan Anda untuk memperkirakan biaya finansial yang sesungguhnya untuk memuat situs Anda. Histogram di bawah ini menunjukkan berapa biaya (menggunakan paket data prabayar) untuk memuat amazon.com.



Ingatlah bahwa ini tidak memperhitungkan keterjangkauan harga relatif terhadap pendapatan. Data dari blog.jana.com menunjukkan biaya data.

	Biaya paket data 500 MB (USD)	Upah minimum per jam (USD)	Jam kerja untuk membayar paket data 500 MB
India	$3,38	$0,20	17 jam
Indonesia	$2,39	$0,43	6 jam
Brasil	$13,77	$1,04	13 jam

Ukuran laman bukan hanya masalah bagi pasar negara berkembang. Di banyak negara, orang menggunakan paket data seluler dengan data terbatas, dan akan menghindari situs atau aplikasi Anda jika mereka menganggap hal itu berat dan mahal. Bahkan paket data seluler dan wifi "tak terbatas" biasanya memiliki batas data yang apabila terlewati, data akan diblokir atau dibatasi kecepatannya.

Intinya: ukuran laman memengaruhi kinerja dan biaya. Mengoptimalkan efisiensi materi menunjukkan cara mengurangi biaya tersebut.

DAFTAR PUSTAKA

Bringhurst, Robert. *The Elements of Typographic Style*. Hartley & Marks, 2019.

Developers, Google. "Basics of UX | Web Fundamentals | Google Developers." *Google*, Google, 2018, developers.google.com/web/fundamentals/design-and-ux/ux-basics.

Gothelf, Jeff, and Josh Seiden. *Lean UX: Applying Lean Principles to Improve User Experience*. O'Reilly, 2016.

Itten, Johannes, and Ernst van. Haagen. *The Art of Color: the Subjective Experience and the Objective Rationale of Color*. John Wiley And Sons Ltd, 1997.

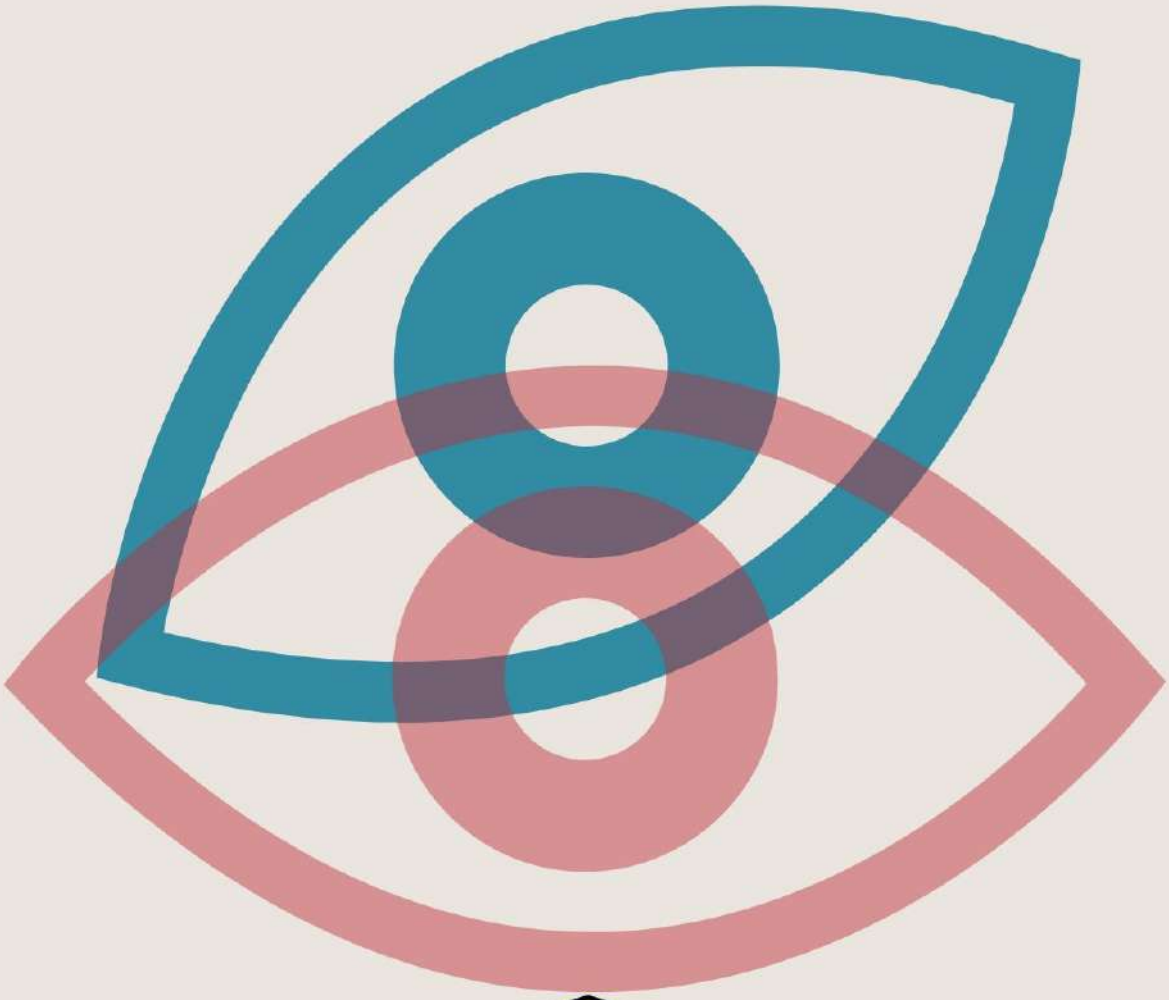
Krug, Steve. *Don't Make Me Think!: Web & Mobile Usability - Das Intuitive Web*. Mitp, 2014.

Leon, Donna. *About Face*. Grove Press, 2018.

Lidwell, William. *The Pocket Universal Principles of Design*. Rockport Publishers, 2015.

Norman, Donald A. *The Design of Everyday Things: Psychologie Und Design Der alltäglichen Dinge*. Vahlen, 2016.

Weinschenk, Susan M. *100 More Things Every Designer Needs to Know about People*. New Riders, 2016.



Penerbit
Lembaga Penelitian & Pengabdian Kepada Masyarakat
Universitas Pembangunan Nasional "Veteran" Yogyakarta

ISBN 978-623-7594-55-0