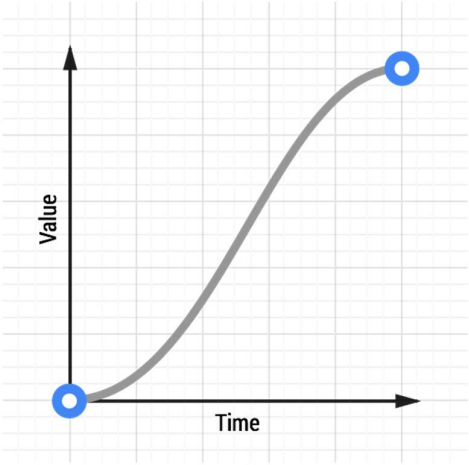
Jangan menggunakan durasi animasi yang terlalu lama, karena awal animasi ease-in agak lambat. Sesuatu di kisaran 300-500 md biasanya cocok, namun jumlah yang tepat sangat bergantung pada nuansa proyek Anda. Jadi, karena awal yang lambat, cepat di tengah, dan lambat di akhir, akan ada peningkatan kontras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

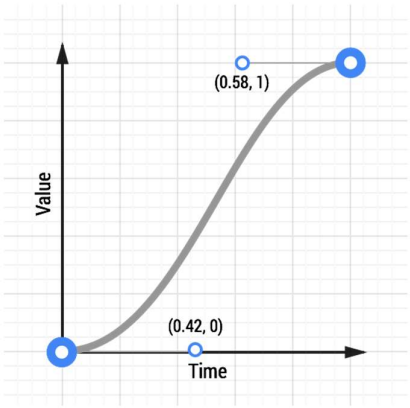
- Easing khusus memungkinkan Anda untuk memberikan lebih banyak kepribadian dalam proyek Anda.
- Anda bisa membuat kurva cubic Bézier yang menyerupai kurva animasi default (ease-out, ease-in, dll.) tapi dengan penekanan pada tempat yang berbeda.
- Gunakan JavaScript ketika Anda membutuhkan lebih banyak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

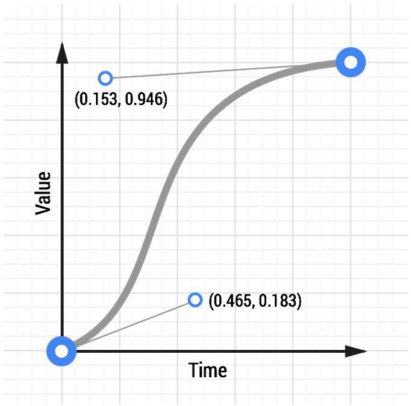
Jika membuat animasi dengan CSS, Anda akan mendapati bahwa Anda bisa menentukan kurva cubic Bézier untuk menetapkan waktunya. Faktanya, kata kunci ease, ease-in, ease-out dan linear memetakan ke kurva Bézier yang sudah ditetapkan, yang dijelaskan terperinci dalam spesifikasi transisi CSS dan spesifikasi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

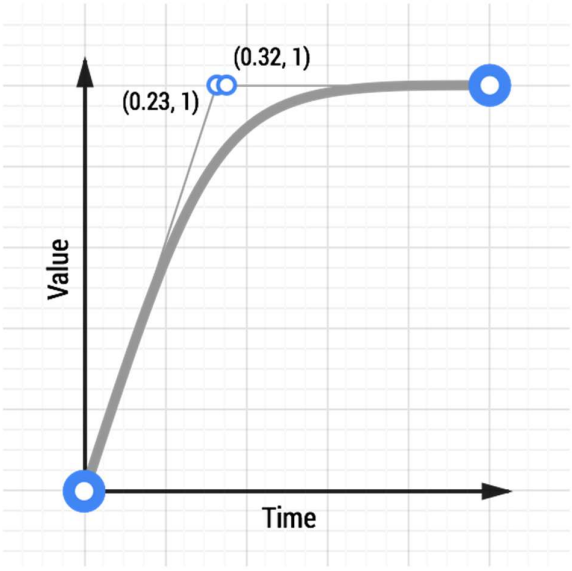
Pada layar yang lebih besar, Anda harus selalu menampilkan tampilan daftar bukan menghilangkannya, dan menggeser pada tampilan detail dari sisi sebelah kanan. Ini hampir sama dengan menangani tampilan navigasi.

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.

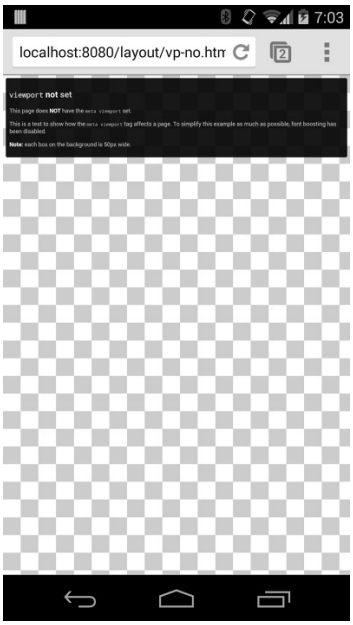
TL;DR

- 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.

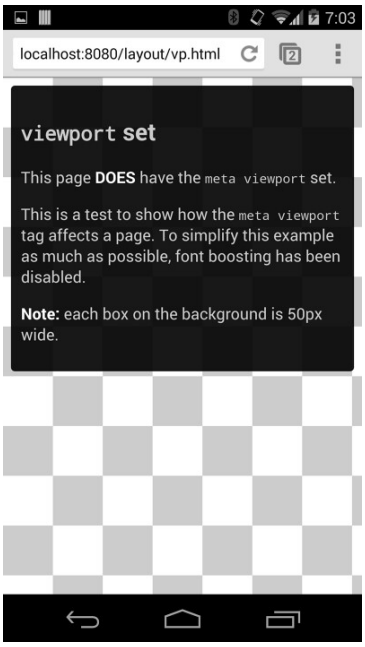
Dalam upaya menyediakan pengalaman terbaik, browser seluler merender laman pada lebar layar desktop (biasanya sekitar 980 px, meskipun ini bisa berbeda antar perangkat), dan kemudian mencoba membuat materi terlihat lebih baik dengan memperbesar ukuran font dan mengubah ukuran materi agar sesuai dengan 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

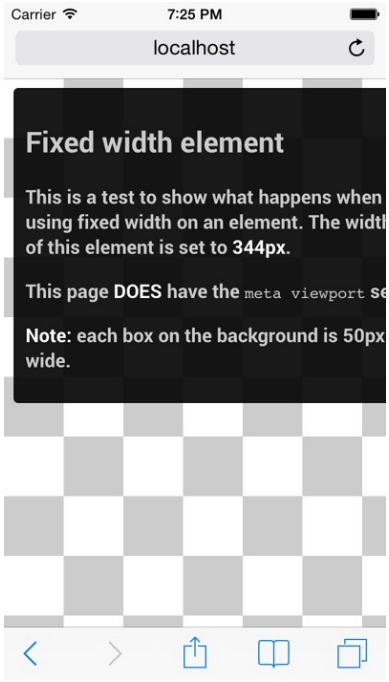
- Jangan menggunakan elemen berlebar tetap yang besar.
- Materi tidak boleh bergantung pada lebar tampilan yang terlihat tertentu untuk merender dengan baik.
- Gunakan kueri media CSS untuk menerapkan penataan gaya yang berbeda untuk layar kecil dan besar.

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.

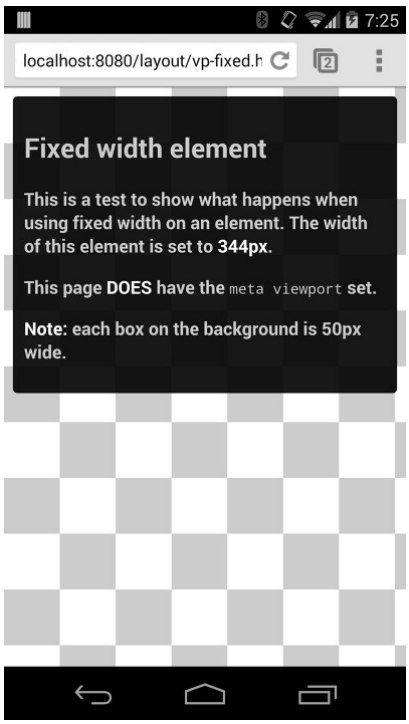
Oleh karena ukuran dan lebar layar dalam piksel CSS bervariasi antar perangkat (misalnya, antara ponsel dan tablet, dan bahkan antara ponsel yang berbeda), materi tidak boleh bergantung pada lebar tampilan yang terlihat tertentu untuk dirender dengan baik.

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%;
}
```

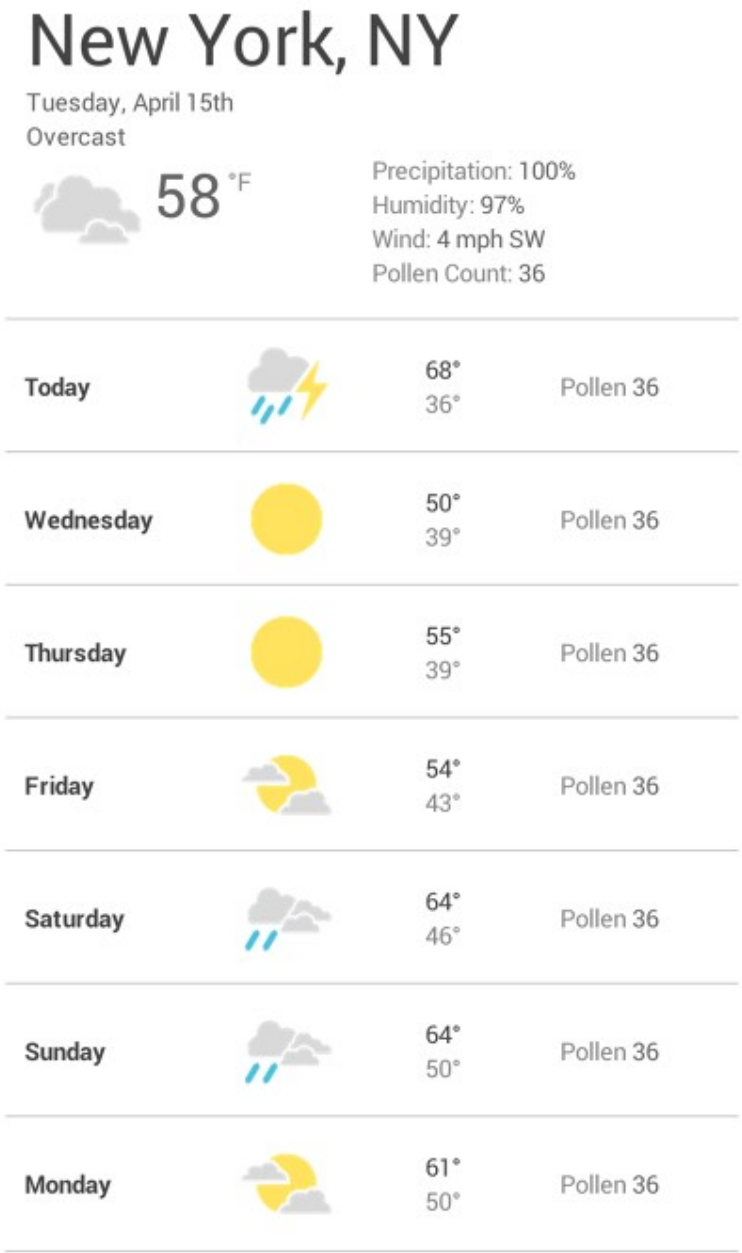
Cara memilih breakpoint

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.

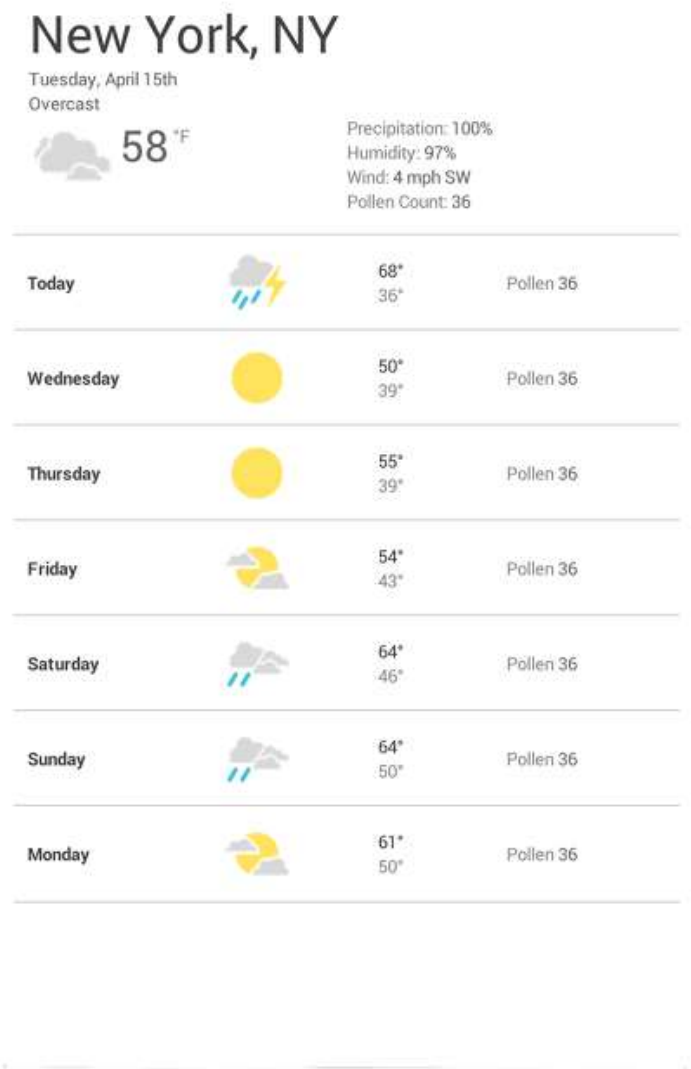
```
@media (min-width: 360px) {
 body {
  font-size: 1.0em;
 }
}

@media (min-width: 500px) {
 .seven-day-fc .temp-low,
 .seven-day-fc .temp-high {
```

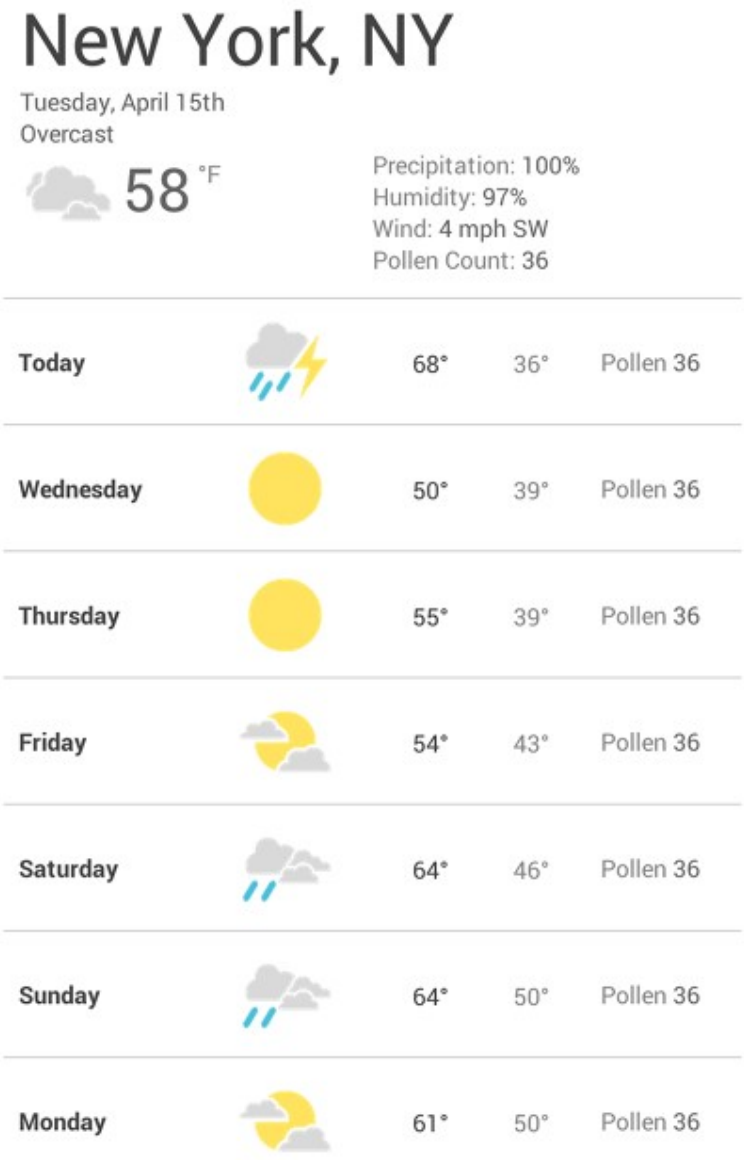
```
  display: inline-block;
  width: 45%;
 }

 .seven-day-fc .seven-day-temp {
  margin-left: 5%;
 }

 .seven-day-fc .icon {
  width: 64px;
  height: 64px;
 }
}
```



Sebelum menambahkan breakpoint kecil.



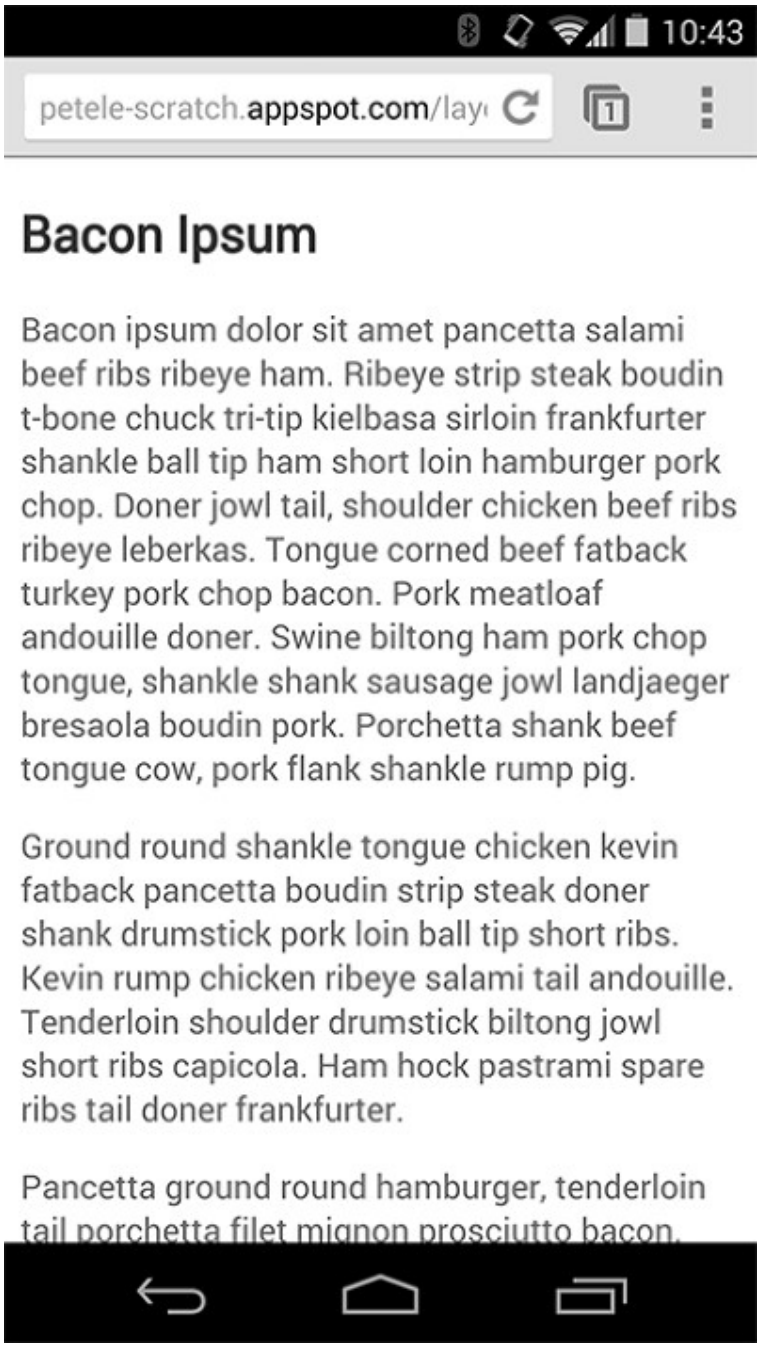
Setelah menambahkan breakpoint kecil.

Demikian pula, untuk layar besar akan sangat baik membatasi lebar maksimum panel prakiraan cuaca sehingga tidak memakai seluruh lebar layar.

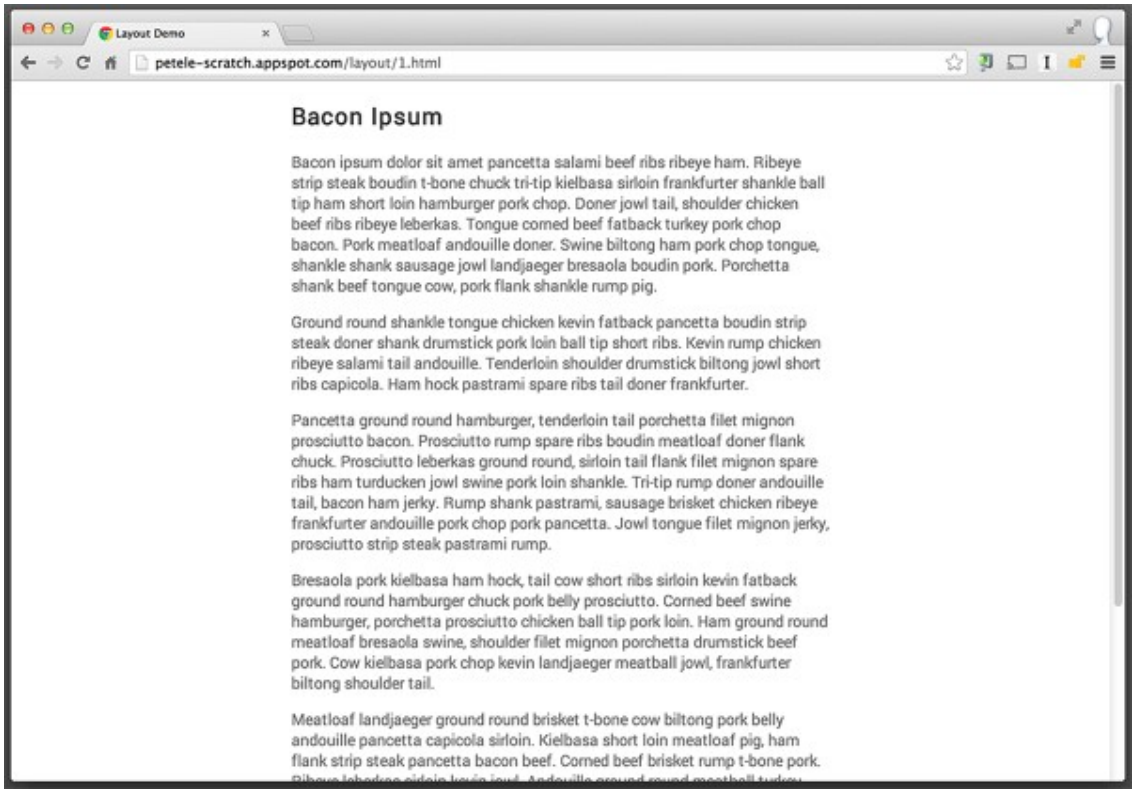
```
@media (min-width: 700px) {
  .weather-forecast {
    width: 700px;
  }
}
```

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;
 }
}
```

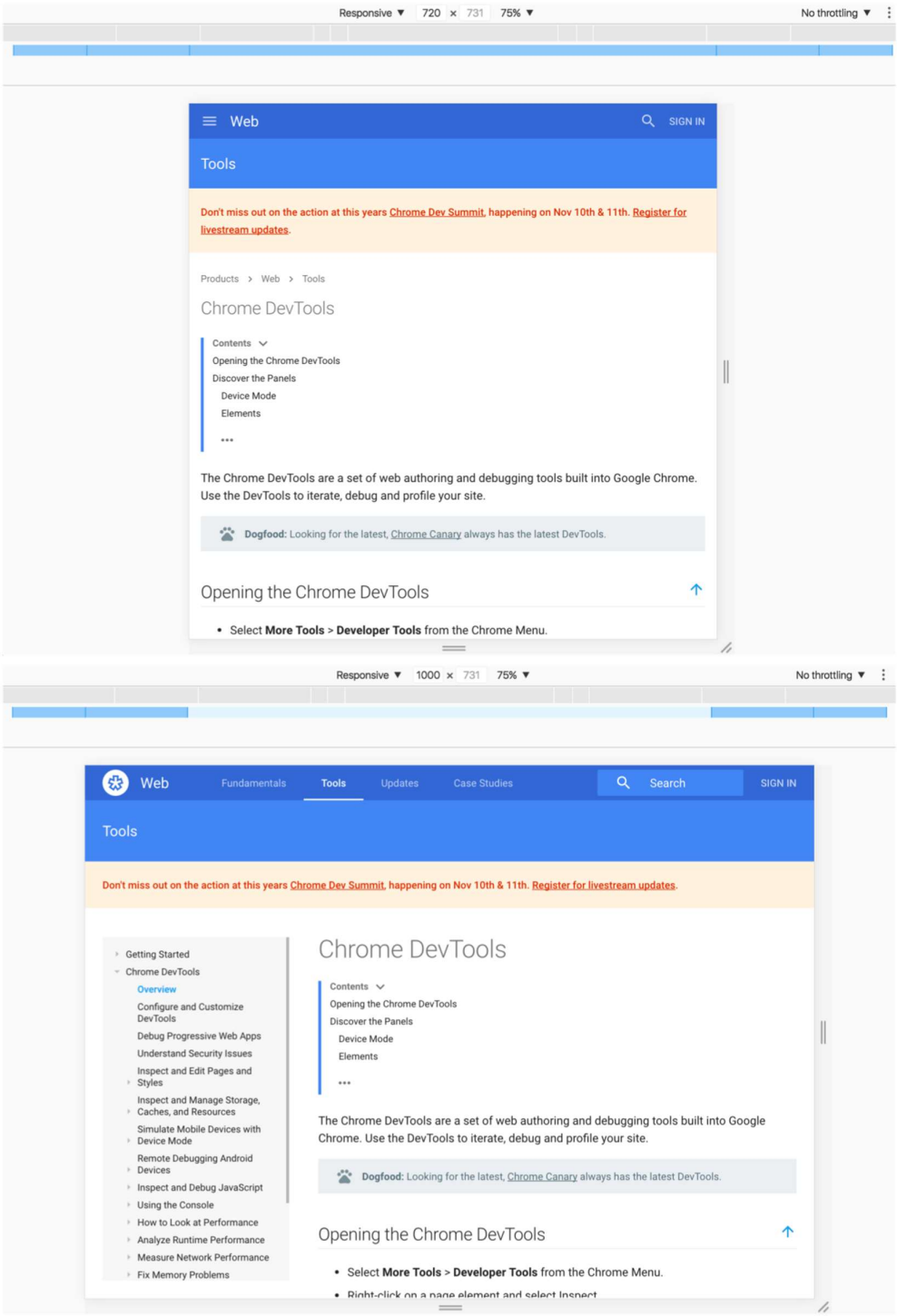
Cobalah

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.

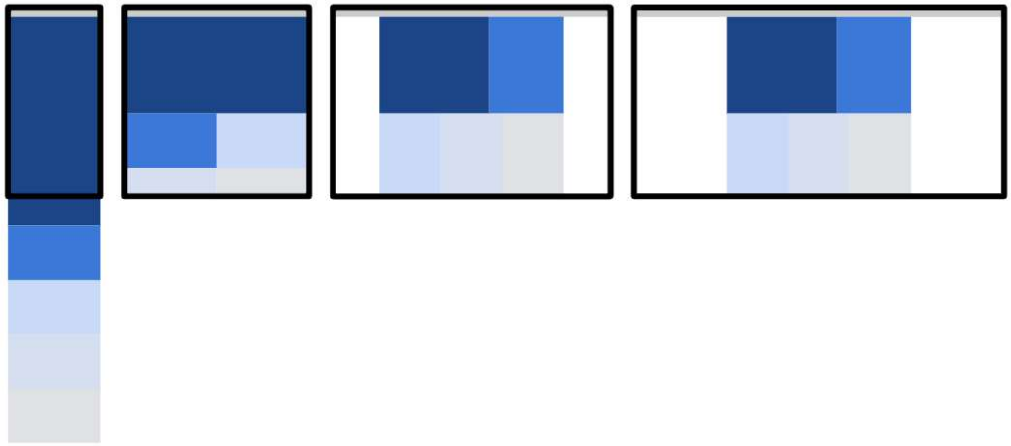
Pola

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) {
```

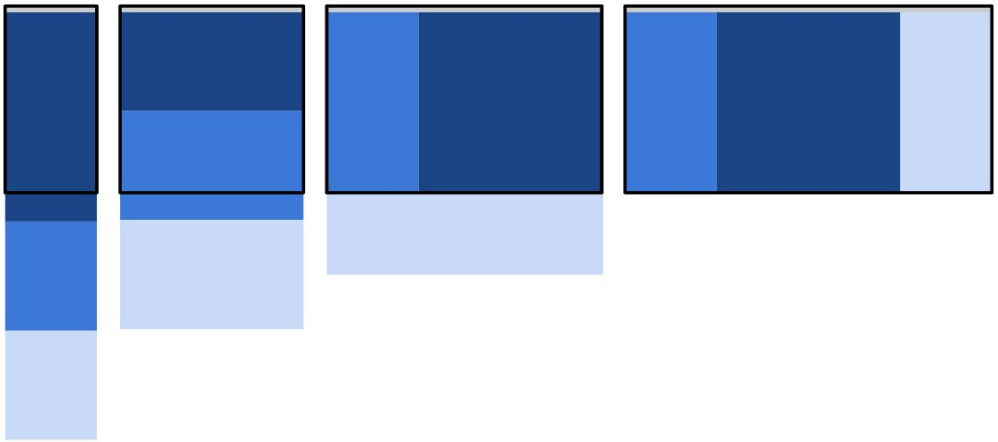
```
 .c1 {
   width: 60%;
 }
 .c2 {
   width: 40%;
 }
 /* Using 33.33%, doesn't always work right due to rounding */
 .c3, .c4 {
   width: 33%;
 }
 .c5 {
   width: 34%;
 }
}

@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 {
```

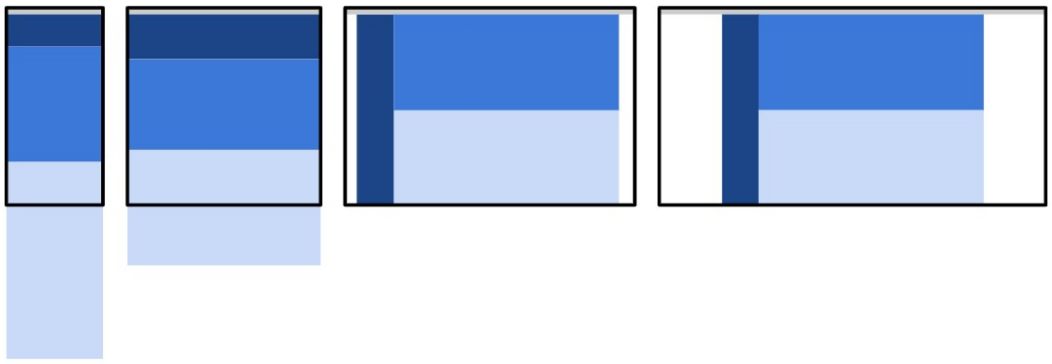
```
  width: 20%;
 }

 .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

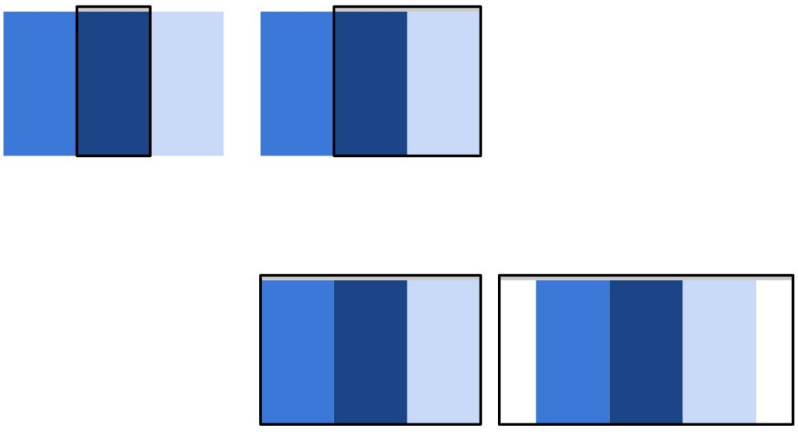
```
.c1 {
  padding: 10px;
  width: 100%;
}

@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

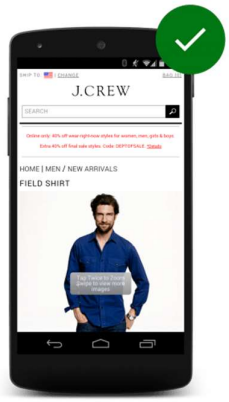
Dalam banyak kasus, ukuran gambar bisa berubah bergantung pada breakpoint layout situs. Misalnya, pada layar kecil, Anda mungkin menginginkan agar gambar membentang penuh pada tampilan yang terlihat, sementara di layar yang lebih besar, itu hanya menggunakan sebagian kecilnya.

```
<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:

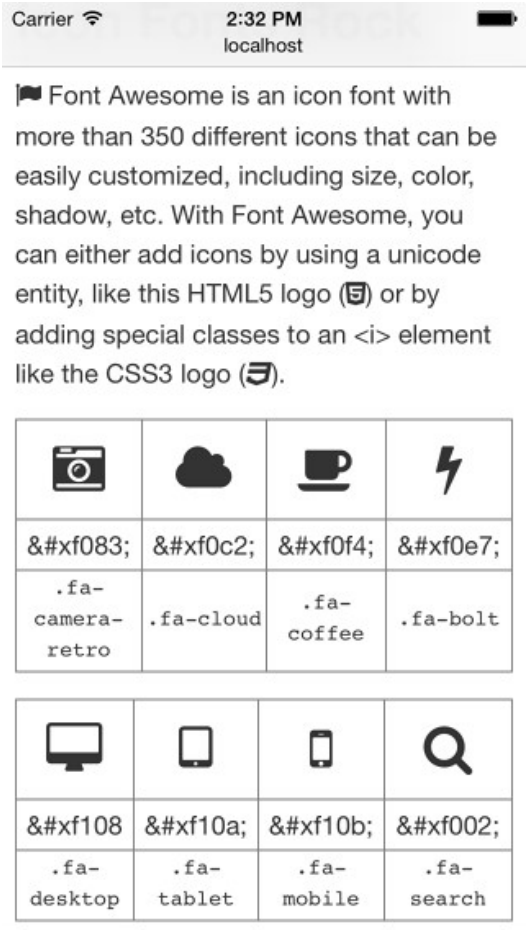
- Mereka adalah grafis vektor yang bisa diskalakan secara tak terbatas.
- Efek CSS seperti warna, bayangan, transparansi, dan animasi bisa dibuat dengan mudah.

- 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.

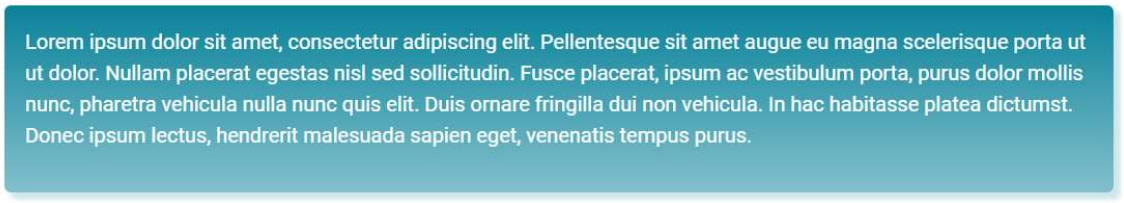
Tempatkan teks dalam markup bukannya disematkan pada gambar

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.

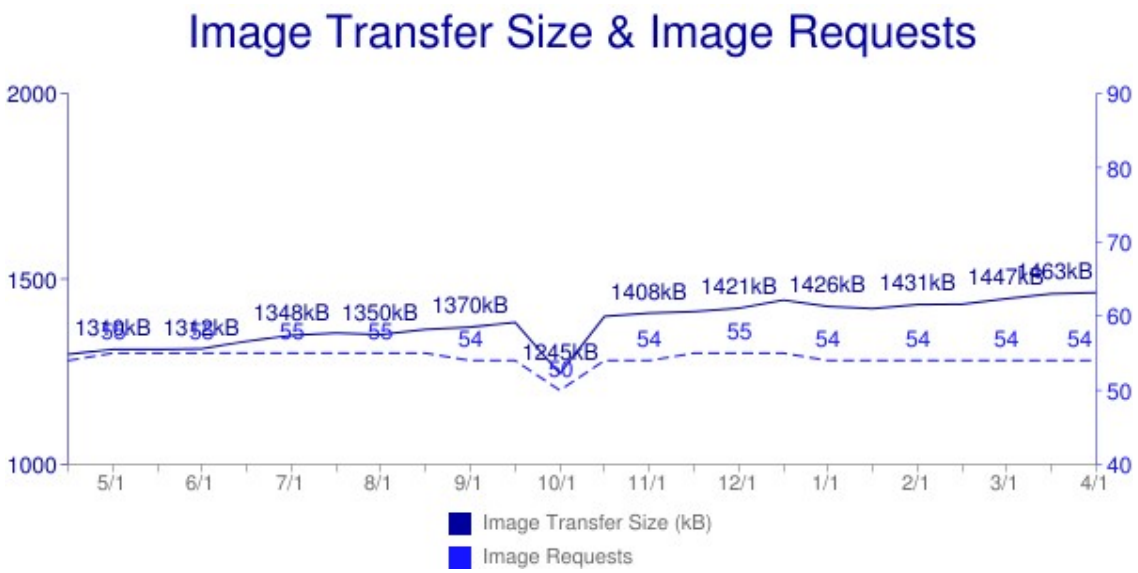
Jangan sekadar menyembunyikan materi dari pengguna seluler. Arahkan ke paritas materi, karena Anda tidak akan bisa menebak apa fitur seluler yang diinginkan pengguna. Jika Anda memiliki sumber daya, buat versi alternatif dari materi yang sama untuk ukuran tampilan yang terlihat berbeda — bahkan jika hanya bagi elemen laman dengan prioritas tinggi.

Perhatikan pengelolaan materi dan alur kerja: sistem lawas menghasilkan materi lawas?

Menyederhanakan teks

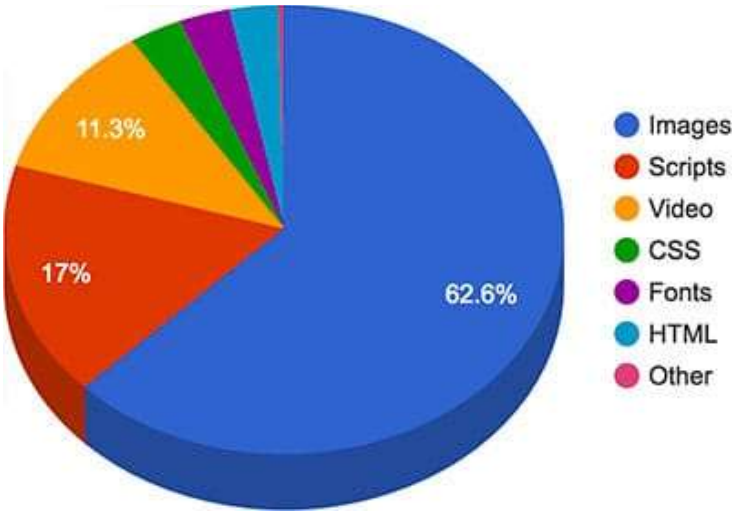
Karena web semakin populer di perangkat seluler, Anda harus mengubah cara Anda menulis. Buatlah tetap sederhana, kurangi kesemrawutan dan langsung ke intinya.

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.

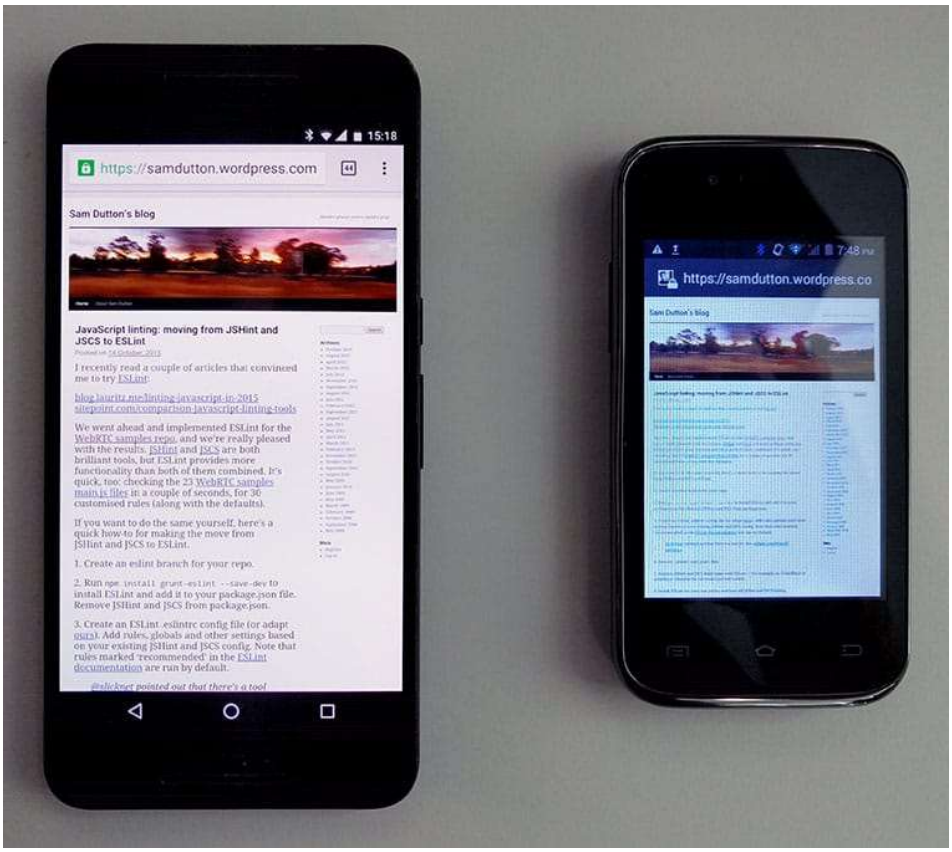
Merancang materi agar bekerja dengan baik pada ukuran tampilan yang terlihat berbeda

> "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

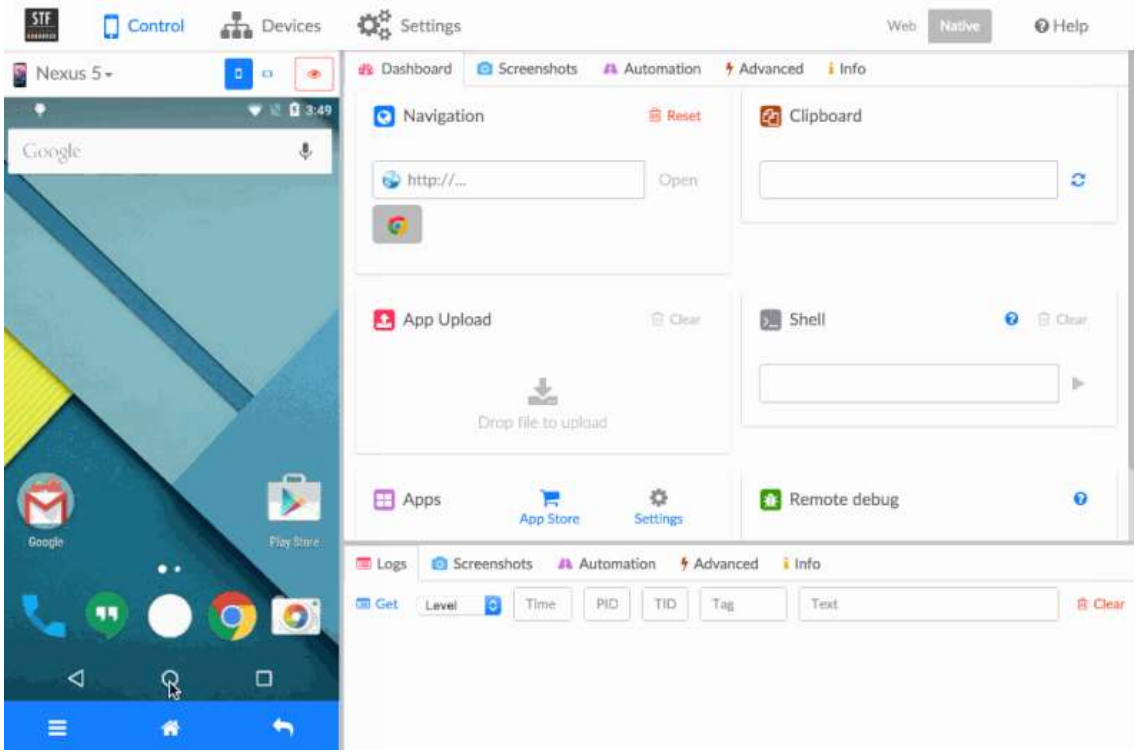
- Tempatkan materi Anda yang paling penting di atas, karena pengguna cenderung membaca laman web dalam pola berbentuk-F.
- Pengguna mengunjungi situs Anda untuk mencapai sebuah tujuan. Tanyakan pada diri sendiri apa yang mereka butuhkan untuk mencapai tujuan tersebut dan buang segala sesuatu yang lain. 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:

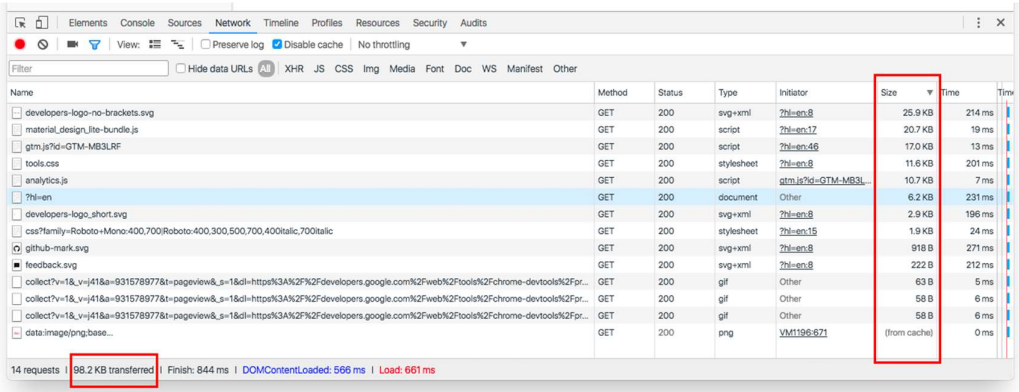
> Banyak orang yang sebelumnya tidak menggunakan YouTube, tiba-tiba memakainya.

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.

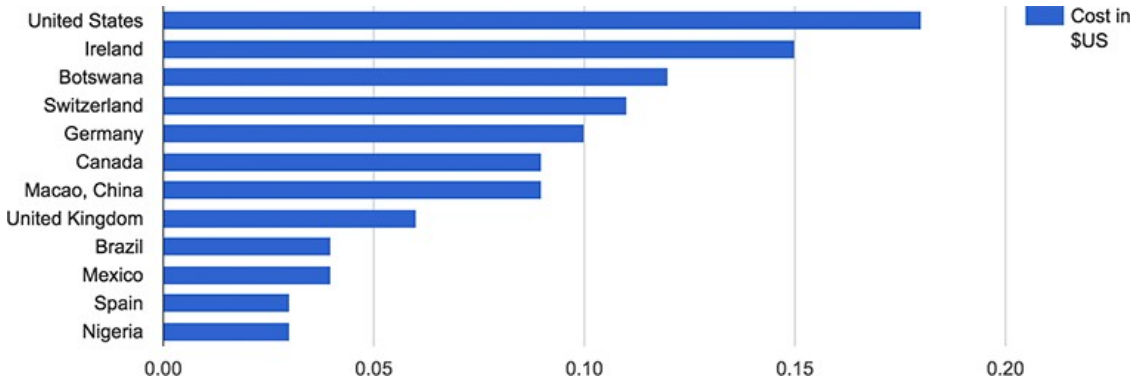
						Document Complete			Fully Loaded			
	Load Time	First Byte	Start Render	Speed Index	DOM Elements	Time	Requests	Bytes In	Time	Requests	Bytes In	Cost
First View	11.950s	7.932s	1.149s	7867	404	11.950s	31	495 KB	12.023s	33	501 KB	$$---
Repeat View	3.580s	3.233s	0.829s	895	404	3.580s	3	0 KB	3.501s	3	0 KB	

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.

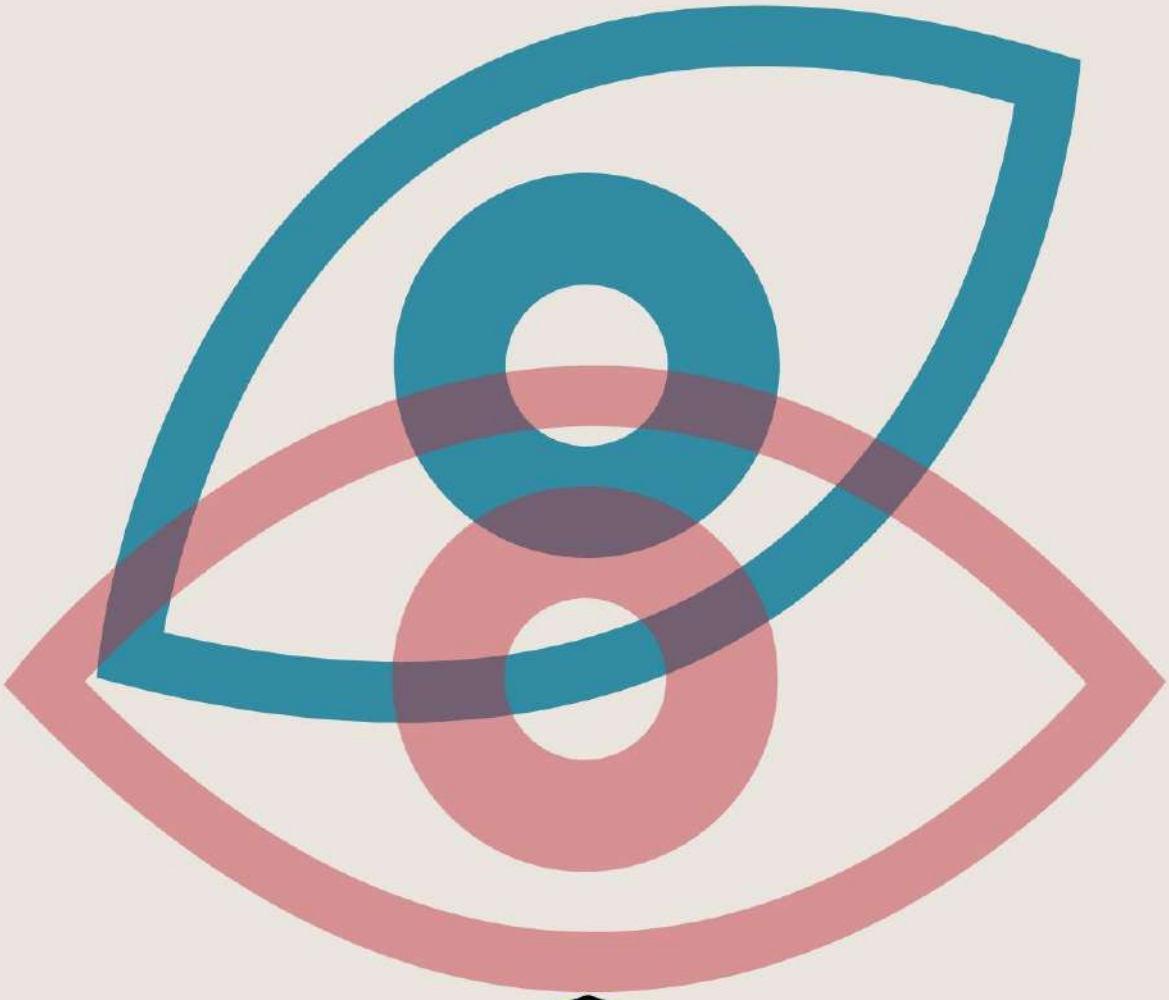
Krug, Steve. *Don't Make Me Think!: Web & Mobile Usability - Das Intuitive Web*. Mitp, 2014.

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