Responsive Design: Clever Tips and Tricks

Vitaly Friedman
28/11/2012 @ FOWD, Prague
Vitaly Friedman, editor-in-chief and co-founder of SmashingMag
Designing The Well-Tempered Web
By Rob Flaherty

January 17th, 2012  Design, User Experience  28 Comments

As technology evolves, so does the art and craft of Web design. New technology creates new challenges, which require new solutions. Often we’re working in uncharted territory, where the solutions demanded really are new. Other times, we’re faced with problems of a more universal nature, problems that have a history.

Given the limited history of Web design, we have to look beyond our immediate domain for answers to the more challenging questions. We do this all the time when we draw on the rich history of graphic design and visual arts. But we’re not limited to sibling disciplines. If we can identify the abstractions and patterns that constitute our challenges, we can look to any source for guidance. We can look to a seemingly unrelated field, such as psychology or music. We can even look to an episode from the early 18th century about Johann Sebastian Bach.

In this article we’ll look at what Bach has to do with modern Web challenges — particularly the challenge of designing for devices with diverse attributes and capabilities.

Bach And “The Well-Tempered Clavier”

In 1722, Bach put together a book of solo keyboard works intended as a collection of educational pieces for young musicians. The book contained 48 pieces — a prelude and fugue in every major and minor key. Now a staple of the Western canon, it’s regarded as one of the most important works in the history of Western music. He named the book The Well-Tempered Clavier.
This talk is about **RWD techniques**.
This talk is about **RWD techniques**.
And simple, clever **tricks and ideas**.
This talk is about RWD techniques. And simple, clever tricks and ideas. And (a bit) about our 2012 redesign.
Responsive Web Design (Extended)
The dangerous thing is not that machines might start thinking like humans, but that humans might start thinking like machines.

— Sydney J. Harris
We are blinded by chrome. When it comes to RWD, we think about layouts, and often we should, but we have to keep in mind that we are not rectangle artists. We explore solutions to problems. Browsers think in boxes, but humans shouldn’t.
When it comes to responsive design, we think about layouts, and sometimes we should, but we have to keep in mind that we aren't rectangle artists. We explore solutions to problems.
A fluid, unpredictable, chaotic, interconnected environment with plenty of right and wrong solutions. I always feel weird about blog posts on why RWD is a wrong technique, or HTML5/native is a wrong solution. The Web isn’t black and white, it’s rich, extremely diverse and it requires pragmatic thinking.
Responsive Web Design is an appropriate tool for this “fluid” Web.
It’s a new mindset that requires us to rethink and extend our practices.
Content
Choreography
Media queries can be used to do more than patch broken layouts: with proper planning, we can begin to *choreograph* content proportional to screen size, serving the best possible experience at any width.

— Trent Walton
A bare-bones example of content choreography in Responsive Web Design.

Using flexbox and box-ordinal-group CSS properties, **we can change the order** of elements on a page at different viewport sizes. In this very basic example I initiate flexbox via a media query in my global styles stylesheet so that only browsers that can a) read media queries and b) have smaller viewports than 33em (i.e. smartphones and i.e not IE<9) see the re-ordered items. The circled labels represent the order they appear in the code - resize your browser to see them change if you are on a desktop.

Support for this technique is **widespread among smartphones**, it is safe to use to reorder page contents for mobile and tablet devices.

---

A [content choreography](#) demo using flexbox and box-ordinal-group by [Jordan Moore](#).
From the technical standpoint, arrangement of boxes is often implemented using **Flexbox**.

```css
@media screen and (max-width: 33.236em) {
  #main {
    display: flex;
  }
  #main > nav, #main > aside {
    flex: 1;
  }
  #main > article {
    flex: 2;
  }
  #main > nav {
    order: 0;
  }
  #main > article {
    order: 1;
  }
  #main > aside {
    order: 2;
  }
}
```
## Flexible Box Layout Module - Candidate Recommendation

Method of positioning elements in horizontal or vertical stacks.

**Resources:** Introduction with demos, Mozilla hacks article, Flexbox playground

<table>
<thead>
<tr>
<th></th>
<th>IE</th>
<th>Firefox</th>
<th>Chrome</th>
<th>Safari</th>
<th>Opera</th>
<th>IOS Safari</th>
<th>Opera Mini</th>
<th>Android Browser</th>
<th>Blackberry Browser</th>
<th>Opera Mobile</th>
<th>Chrome for Android</th>
<th>Firefox for Android</th>
</tr>
</thead>
<tbody>
<tr>
<td>18 versions back</td>
<td></td>
<td>4.0</td>
<td>-webkit</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17 versions back</td>
<td></td>
<td>5.0</td>
<td>-webkit</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16 versions back</td>
<td></td>
<td>2.0</td>
<td>moz</td>
<td>6.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15 versions back</td>
<td></td>
<td>3.0</td>
<td>moz</td>
<td>7.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14 versions back</td>
<td></td>
<td>3.5</td>
<td>moz</td>
<td>8.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13 versions back</td>
<td></td>
<td>3.6</td>
<td>moz</td>
<td>9.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12 versions back</td>
<td></td>
<td>4.0</td>
<td>moz</td>
<td>10.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11 versions back</td>
<td></td>
<td>5.0</td>
<td>moz</td>
<td>11.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 versions back</td>
<td></td>
<td>6.0</td>
<td>moz</td>
<td>12.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9 versions back</td>
<td></td>
<td>7.0</td>
<td>moz</td>
<td>13.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8 versions back</td>
<td></td>
<td>8.0</td>
<td>moz</td>
<td>14.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7 versions back</td>
<td></td>
<td>9.0</td>
<td>moz</td>
<td>15.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6 versions back</td>
<td></td>
<td>10.0</td>
<td>moz</td>
<td>16.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 versions back</td>
<td></td>
<td>11.0</td>
<td>moz</td>
<td>17.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 versions back</td>
<td></td>
<td>5.5</td>
<td>moz</td>
<td>18.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 versions back</td>
<td></td>
<td>6.0</td>
<td>moz</td>
<td>19.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 versions back</td>
<td></td>
<td>7.0</td>
<td>moz</td>
<td>20.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Previous version</td>
<td></td>
<td>8.0</td>
<td>moz</td>
<td>15.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Current</td>
<td></td>
<td>9.0</td>
<td>moz</td>
<td>22.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Near future</td>
<td></td>
<td>10.0</td>
<td>moz</td>
<td>23.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Farther future</td>
<td></td>
<td>18.0</td>
<td>moz</td>
<td>24.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note:** Partial support refers to supporting an older version of the specification or an older syntax.
We can manipulate experiences to make them genuine across different viewports — be it desktop, mobile or anything else.
Latest News

Du Toit signs off with a silver

Highlights
We're an independent digital design, development and branding studio. We create experiences for many.
We're an independent digital design, development and branding studio. We create experiences for many.

WORK
SELECTED PROJECTS

- Google Wallet
- Twitter Political Index
- Rdio
- Lookout
- Panjiva
It seems like you're trying to upload a document. Please upload the document, and I'll be happy to help you with it! 😊
Weightshift

We're an independent digital design, development and branding studio. We create experiences for many.

WORK
SELECTED PROJECTS
Thu, July 14th, 2011

The concept of permanently placing content on a web page for a single browsing width or resolution is becoming a thing of the past.

Media-queried responsive & adaptive sites afford us the ability to re-architect content on a page to fit its container, but with this exciting new
Resolution
Independence
Independence
Resolution Independence

• High pixel density displays prompt us to create future-proof solutions for graphics.

• Creating multiple assets for the same graphics (not photos) isn’t future-proof.

• Two options: SVG and Icon Fonts.
• **HTML:**

```
<ul class="actions">
<li><a class="a-share" href="#">Share</a></li>
<li><a class="a-print" href="#">Print</a></li>
</ul>
```

• **CSS:**

```
.actions a { font-size: 1em; /* Sprite: 30x160px */
    background-image: url('sprite.png'); }
.actions .a-share {
    background-position: 10px 0; }
.actions .a-print {
    background-position: 10px -40px; }
```
• **HTML:**

```html
<ul class="actions">
  <li><a class="a-share" href="#">Share</a></li>
</ul>
```

• **CSS:**

```css
body { font-size: 100%; } /* = 16px by default */
.actions a { font-size: 1em;
  background-image: url('sprite.svg');
  background-size: 1.875em 10em; }
.actions .a-share {
  background-position: 0.625em 0; }
```
Resolution Independence (SVG)

- SVG files are usually larger and browsers need more time to rasterize and display them.

- **Good SVG support:** Chrome 4+, Safari 4+, FF4+, Opera 9.5+, IE9+, mobile browsers.

- For legacy browsers (and Android 2.3) we need PNG-fallback with Conditional Comments (IE<9) or Modernizr.
Icon Fonts

**HTML:**

```html
<a class="icon share" href="#">Share</a>
```

**CSS:**

```css
@font-face { font-family: 'Icon Font';
    src: url('icon-font.eot');
    src: local('☺');
    url('icon-font.woff') format('woff'),
    url('icon-font.ttf') format('truetype'),
    url('icon-font.svg') format('svg'); }

.icon { font-family: 'Icon Font'; font-size: 20px; }
.share:before { content: "s"; }
```
• HTML:
  <a class="icon" data-icon="s" href="#">Share</a>
  <a class="icon" data-icon="h" href="#">History</a>

• CSS:
  .icon:before {
    content: attr(data-icon);
    /* Optional color definition */
  }
Resolution Independence (Web Fonts)

- There are many comprehensive Web fonts: *Entypo* and *FontAwesome* are free.

- **Excellent support**: everywhere but Opera Mini and Android 2.1.

- Build custom, small “bundles” with *Fontello* (combines popular open-source fonts).
Compressive Images
Compressive Images

• To display photos properly on high pixel density displays, we don’t need hi-res images.

• If a JPG image has relatively small dimensions, we can use a workaround to keep its size small.

• Solution: given a “normal” image resolution, double it and use minimal JPEG compression.
...Given two identical images that are displayed at the same size on a website, one can be dramatically smaller than the other in file size if it’s highly compressed and dramatically larger in dimensions than it is displayed.

— Daan Jobsis
600×400px file, 0% JPEG compression, displayed in 600×400 (file size 7 Kb)
600×400px file, 0% JPEG compression, displayed in 300×200 (file size 7 Kb)
<table>
<thead>
<tr>
<th>300×200px file (21 Kb)</th>
<th>600×400px file (7 Kb)</th>
</tr>
</thead>
<tbody>
<tr>
<td>80% JPEG compression</td>
<td>0% JPEG compression</td>
</tr>
<tr>
<td>displayed in 300×200</td>
<td>displayed in 300×200</td>
</tr>
<tr>
<td>Device</td>
<td>PPI</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----</td>
</tr>
<tr>
<td>Apple iPad 3</td>
<td>264</td>
</tr>
<tr>
<td>Apple iPhone 4 / 4S</td>
<td>326</td>
</tr>
<tr>
<td>Apple MacBook Pro 15 &quot;Retina Display&quot;</td>
<td>220</td>
</tr>
<tr>
<td>Archos 10.1 G9</td>
<td>149</td>
</tr>
<tr>
<td>HTC ChaCha</td>
<td>222</td>
</tr>
<tr>
<td>HTC Desire S</td>
<td>252</td>
</tr>
<tr>
<td>HTC One V</td>
<td>252</td>
</tr>
<tr>
<td>Nokia Lumia 800</td>
<td>252</td>
</tr>
<tr>
<td>Nokia Lumia 900</td>
<td>217</td>
</tr>
<tr>
<td>Samsung Galaxy Ace</td>
<td>164</td>
</tr>
<tr>
<td>Samsung Galaxy Nexus</td>
<td>316</td>
</tr>
<tr>
<td>Samsung Galaxy S Advance</td>
<td>233</td>
</tr>
<tr>
<td>Samsung Galaxy S III</td>
<td>306</td>
</tr>
<tr>
<td>Samsung Galaxy Tab 2 10.1</td>
<td>149</td>
</tr>
<tr>
<td>Samsung Galaxy Xcover</td>
<td>158</td>
</tr>
<tr>
<td>Samsung Note</td>
<td>285</td>
</tr>
</tbody>
</table>
Retina revolution

Geplaatst op 27 juli 2012 door Daan Jobsis

The devil is in the details

Detail is probably one of the most important values for a designer, an eye for detail should be in our DNA. As a perfectionist I like my designs to be pixel perfect. I am allergic for “jaggies” and ugly compressed artifacts in icons and images on websites. Apple’s Retina revolution is an interesting evolution that is turning the design world upside down. The Retina display has a high enough pixel density to prevent pixelation to be noticeable to the human eye. Therefore a Retina display is a lot sharper and more pleasant to look at. Apple has doubled the amount of horizontal and vertical pixels on the iPhone, The New iPad, and now also on the new MacBookPro. The Retina revolution is irreversible, and other companies have already started or will also start implementing this new Retina technology.

Nowadays pixel perfection can be obtained with techniques like @font-face and CSS3. Making fonts, borders, shadows, and gradients sparkle on your screen. These elements are based on vectors or mathematical expressions which allows them to be scaled to enormous sizes without creating distortion. This does not count for rasterized images which consist of pixels. An image that looks good on a normal display will appear blurry on a Retina display. The Retina display blows up the image, it doubles the amount of pixels. There is not enough data for the image to be displayed...
Conditional Loading
If you [...] had to choose between employing media queries to make the design look good on a mobile device or optimizing the site for performance, you would be better served by making the desktop site **blazingly fast**.

— Jason Grigsby
Conditional CSS

• We ask browsers to load assets progressively — and only when they *can* be displayed.

• Idea: if a CSS media query was fired, catch it with JavaScript and load additional assets.

• CSS:

```css
@media all and (min-width: 45em) {
  body::after {
    content: 'desktop';
    display: none;
  }
}
```
Conditional CSS

• **CSS:**

```css
@media all and (min-width: 45em) {
  body::after {
    content: 'desktop';
    display: none;
  }
}
```

• **JS:**

```javascript
var size = window.getComputedStyle(document.body, ':after').getPropertyValue('content');
if (size == 'desktop') {
  // Load some more content.
}
```
Conditional CSS (Example)

- **HTML:**
  
  ```html
  <a href="desktop-friendly-version.jpg">
    <img src="mobile-version.jpg"
        data-medium="tablet-friendly-version.jpg"
        data-large="desktop-friendly-version.jpg" />
  </a>
  ```

- **CSS:**
  
  ```css
  @media all and (min-width: 45em) {
    body:after {
      content: 'desktop';
      display: none;
    }
  }
  ```
Conditional CSS (Example)

- **JavaScript**:

```javascript
var size =
window.getComputedStyle(document.body,:after).getPropertyValue('content');

if (size == 'desktop') {
    $('img').each(function(index) {
        var large = $(this).data('large');
        $(this).attr('src',large);
    })
}

if (size == 'tablet') {...
```
Initial State:

39° Cloudy
WEATHER | TRAFFIC

Expanded, Ajax-enhanced state:

39° Cloudy
WEATHER | TRAFFIC

The Bi

<table>
<thead>
<tr>
<th>NEWS</th>
<th>METRO</th>
<th>ARTS</th>
<th>BUSINESS</th>
<th>SPORTS</th>
<th>OPIN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Weather
Five-day forecast
MORE WEATHER ON BOSTON.COM →

- Today
  - MARCH 28
  - A shower
  - 54° 41°

- Thursday
  - MARCH 29
  - A few showers
  - 50° 33°

With this pattern in place, we found a nice balance between loading performance and accessibility: the content begins lightweight, and accessible without Javascript, and it is enhanced into a richer experience on capable devices—all without holding up the initial page load.

From a technical standpoint, the approach was simple: any element in the page could reference an external HTML fragment by URL through a small set of standard HTML and CSS APIs. Loading the external content—whether using XHR at page load, or later, through an event—then proceeds to enhance the page in a way that's not visible until it’s needed.
An Ajax-Include Pattern for Modular Content

Posted by Scott on 03/28/2012

Topics: ajax javascript jQuery mobile php progressive enhancement

While developing the front-end of the BostonGlobe.com site last fall, one of the toughest challenges we faced was delivering roughly the same content to all devices (and connection speeds) while ensuring the most important content on a page was usable as soon as possible. We approached this challenge with a variety of techniques, such as only loading the most essential JavaScript up-front (weighing roughly 4-6kb tops) and lazy loading the rest, dynamically injecting advertisements, and loading "nice-to-have" content via JavaScript—all after the initial content was delivered and usable.
Conditional CSS

I got some great comments on my post about conditionally loading content.

Just to recap, I was looking for a way of detecting from JavaScript whether media queries have been executed in CSS without duplicating my breakpoints. That bit is important: I’m not looking for MatchMedia, which involves making media queries in JavaScript. Instead I’m looking for some otherwise-useless CSS property that I can use to pass information to JavaScript.

Tantek initially suggested using good ol’ voice-family, which he has used for hacks in the past. But, alas, that unsupported property isn’t readable from JavaScript.

Then Tantek suggested that, whatever property I end up using, I could apply it to an element that’s never rendered: meta or perhaps head. I like that idea.

A number of people suggested using font-family, citing Foresight.js as prior art. I tried combining that idea with Tantek’s suggestion of using an invisible element:

```css
@media screen and (min-width: 45em) {
  head {
    font-family: widescreen;
  }
}
```
Lazy Loading
JS, Social Buttons
Gmail’s Lazy Loading

- *Latency* is the time between when a browser requests a resource from a server and when it starts to receive the server’s response.

- On mobile, latency is a major UX killer. For a 1Mb page with 85 requests per page, it is 4.5s!

- JavaScript is *expensive*; parsing takes time and blocks the rendering of the page. Usually you don’t need *all* JavaScript right away.
Gmail’s Lazy Loading

- Idea: let browsers download all of the JS right away, but evaluate it “on demand”, i.e. when users need a particular feature.
- Much of the downloaded JS is commented out, and when needed uncommented and eval-ed.
- Gmail’s case:
  - 200 Kb of JS -> 2600 ms page load
  - 200 Kb of JS (lazy loaded) -> 240 ms page load
Gmail’s Lazy Loading

- `<script id="lazy">`  
  // Make sure you strip out (or replace) comment blocks in your JavaScript first.  
  /* JavaScript of lazy module */  
  </script>

- `<script>`  
  function lazyLoad() {  
    var lazyElement = document.getElementById('lazy');  
    var lazyElementBody = lazyElement.innerHTML;  
    var jsCode = stripOutCommentBlock(lazyElementBody);  
    eval(jsCode); }  
  </script>

- `<div onclick=lazyLoad()>Lazy Load</div>`
Thursday, September 03, 2009

Gmail for Mobile HTML5 Series: Reducing Startup Latency

On April 7th, Google launched a new version of Gmail for mobile for iPhone and Android-powered devices. We shared the behind-the-scenes story through this blog and decided to share more of what we’ve learned in a brief series of follow-up blog posts. This week, I’ll talk about how modularization can be used to greatly reduce the startup latency of a web app.

To a user, the startup latency of an HTML 5 based application is critical. It is their first impression of the application’s performance. If it's really slow, they might not even bother to wait for the app to load before navigating away. Even if your application is blazing fast after it loads, the user may never get the chance to experience it.

There are several aspects of an HTML 5 based application that contribute to startup latency:

1. Network time to fetch the application (JavaScript + HTML)
2. JavaScript parse time
3. Code execution time to fetch the data and render the home page of your application

The third issue is up to you! The first two issues, however, are directly correlated with the size of the application. This is a tricky problem since as your application matures, it will have more features and the code size will get bigger. So, what to do? Modularize your application! Split up your code into independent, standalone modules. Consider splitting each view/screen of your application and implement each new feature as its own module. This is only half the story. Now that you have your code modularized, you need to decide which subset of these modules are critical to load your application’s home page. All the non-core modules should be downloaded and parsed at a later time. With a consistent code size for your startup code, you can maintain a consistent startup time. Now, let’s go into some nitty gritty details of how we built an application with lazy-loaded modules.

How to Split Your Code into Modules
The Two-Click Social Widget

• Load social widgets only when user *explicitly* chooses to take that action to share content.

• Idea: load small social icons by default, and load the FB, Twitter and G+ widgets on click.

• Cuts down on bandwidth and on latency. (FB button alone weighs 120 Kb + 4 requests).

So hebt auch Samsung in einer ersten Reaktion darauf ab, dass Ericsson viel höhere Lizenzkosten als zuvor verlangt habe. Diese seien so hoch gewesen, dass sie einen neuen Lizenzvertrag für Samsung unmöglich gemacht hätten. Samsung habe in den vergangenen zwei Jahren Verhandlungen mit Ericsson auf der Basis geführt, eine FRAND-Lizenzierung zu erreichen, und werde sich nun gegen die überhöhten Forderungen Ericssons zur Wehr setzen. (jk)

Kommentare lesen (48 Beiträge)

Permalink: http://heise.de/-1757452

Auch auf heise online:
- US-Prozess um Lizenzierung von Standardpatenten beginnt
- Apple vs. Google: Zweiter Patentprozess geplatzt
- Apple will Motorola 1 Dollar pro iPhone als Lizenzgebühr zahlen
- Patentkriege: Google sucht Kompromise bei Standard-Patenten
- EU will faire Lizenzierung von Standardpatenten durchsetzen
- Patentstreit: Apple fordert faire Regeln

Mehr zum Thema Patente Mobilfunk Ericsson Samsung
jQuery Plug-In socialshareprivacy – Dokumentation

Download des jQuery-Plug-Ins:

jquery.socialshareprivacy.zip
jquery.socialshareprivacy.tar.gz

Navigation

1. Change-Log
2. Dateien
3. Voraussetzungen und Einschränkungen
4. Ausmaße
5. Einfache Einbindung
   1. Quelltext
   2. Erklärung des Codes
6. Optionen
   1. Allgemein
   2. Facebook
   3. Twitter
   4. Google+
7. Beispiel-Einbindungen
   1. Nur Facebook einbinden
   2. Keine Option zum dauerhaften Aktivieren anbieten
8. URL
9. Einstellung merken
10. Lizenz
Protecting image/video aspect ratios
Protecting Image Aspect Ratios

- When `max-width: 100%;` is applied to an image with `width` and `height` attributes defined in HTML, image rescales incorrectly.

- Solution: add `height: auto;` for images to which `max-width: 100%` is applied.

- CSS:

```
img, video { max-width: 100%; height: auto; }
```
**Intrinsic Ratio For Videos**

- To ensure the intrinsic 4:3 or 16:9 ratios for videos, we create a box with the proper ratio, then stretch the video inside to fit the dimensions of the box.

- **HTML:**

  ```html
  <div class="wrapper-with-intrinsic-ratio">
    <div class="element-to-stretch"></div>
  </div>
  ```
Intrinsic Ratio For Videos

**CSS:**

```css
.wrapper-with-intrinsic-ratio {
  position: relative;
  padding-top: 25px; /* player chrome */
  padding-bottom: 56.25%; /* 9:16 = 0.5625 */
  height: 0; }

.element-to-stretch {
  position: absolute;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background: teal; }
```
Creating Intrinsic Ratios for Video

by THIERRY KOBLENTZ

MAY 26, 2009

Published in: CSS, HTML and XHTML, Scripting

Discuss this article »  Share this article »

Did you ever want to resize a video on the fly, scaling it as you would an image? Using intrinsic ratios for video and some padding property magic, you can. Thierry Koblenz shows us how.

The concept:-

Ad via The Deck

JOB BOARD

Oyster is looking for a Lead iOS Engineer at Oyster.
A lightweight, easy-to-use jQuery plugin
Fluid aspect ratio

We can take this a step further. Suppose we have a widescreen image that looks great on a desktop computer. On a mobile device we don’t want to use the same aspect ratio or the image will become too small. We also don’t want to use the exact same height or the image becomes too tall. Instead the height should decrease gradually when the width is reduced. We call this a fluid aspect ratio.

![Diagram showing fluid aspect ratio between desktop and mobile devices](image)

The effect can be accomplished by decreasing the percentual padding and setting a height on the element. Suppose the large image is 800 by 200 pixels and we decide the image should be only 150 pixels high when the width is decreased to 300 pixels. Now we need to calculate the `height` and `padding-top` attributes.
Responsive Videos

- We can serve different video files to different devices by using `media` attribute on the `video <source>` attribute.

- Supported in the latest versions of Chrome, Opera, Safari, FF 15+, IE9+, mobile browsers.
Responsive Videos

• **HTML:**

```html
<video controls preload="none">
<source type="video/mp4" src="video_small.mp4"
    media="all and (max-width: 480px),
            all and (max-device-width: 480px)">
<source type="video/webm" src="video_small.webm"
    media="all and (max-width: 480px),
            all and (max-device-width: 480px)">
<source type="video/mp4" src="video.mp4">
<source type="video/webm" src="video.webm">
<!-- proper fallback content goes here -->
</video>
```
Vertical media queries & splitting
Vertical Media Queries

- `min-height` and `max-height` are useful for adjusting the font-size, padding, margin and cropping images.

- Beware of **h/v-media queries collisions** when resizing the browser. Things might easily get out of control.

```javascript
var size = window.getComputedStyle(document.body,':after').getPropertyValue('content');
if (size == 'desktop') {
  // Load some more content.
}
```
Vertical Media Queries & Wide Sites

I've been tinkering with the font-size and the media queries that trigger width-based changes for this site regularly over the past month. I thought that maxing out at 20px for body copy at widths above 900px would be my final answer, but I decided to drop in one more increase last week:

```css
@media screen and (min-width: 1234px) and (min-height:780px) {
  body {font-size: 137.5%};
}
```

This bumps body copy up to 22px for those bold enough to view the site at least 1234px wide. Why 1234px? Just because that's where, with all % values for widths & container padding, my site hits its max-width.

What's with the vertical media query? I used min-height to target...
Media Queries Splitting

• In development, we can use a breakpoint-based organization for CSS ("min-width"):
  0-up.css, 450-up.css, 720-up.css etc.

• We can also set breakpoints 1px apart and split styles instead of overriding from one media query to the next ("min/max-width"):
  base.css, 0-449.css, 450-719.css etc.

JavaScript:
```javascript
var size = window.getComputedStyle(document.body,':after').getPropertyValue('content');
if (size == 'desktop') {
  // Load some more content.
```
Media Queries Splitting

• In practice, it’s often a good starting point to work with em media queries right away. 0-up.css, 25em-up.css, 35em-up.css etc.

• If it’s not an option, it’s a good idea to convert px to em for production code to improve maintenance and avoid zooming issues.
The EMs have it: Proportional Media Queries FTW!

Lyza Gardner | March 28, 2012

If we’re going to be proportional...

A core tenet of Responsive Web Design (RWD) is fluidity and proportion. Instead of using fixed-width layouts, we enlightened web devs and designers use percentages in our CSS. Font units aren't pixels or points anymore, they're percentages (typically for top-level baseline reset) or, more often, ems. And yet the vast majority of us still write width-based media queries in pixels, e.g.:

```css
@media all and (min-width: 500px) {}
@media screen and (max-width: 800px) {}
```

It's a natural thought process: for windows or screens of particular width ranges, we want to flow our content in a particular way. This naturally seems like a pixel...
We’re Stuff & Nonsense, a small, British web design studio with a big reputation around the world because we design flexible websites that look fabulous on every type of device.

Find out more or Start a project with us
Debugging Media Queries
Designing for the Web is like visualizing a tesseract. We build experiences by manipulating their shadows.

— Tim Brown
Debugging Media Queries

• Due to lack of convenient tools, debugging RWD often feels like groping in the dark. There are some popular techniques though.

• Setting the body bg color to different colors for each breakpoint. Also box-sizing: border-box.

• The * technique for testing for optimal measure in the browser.
"Viewable" CSS example
"Viewable" CSS example 500-600
People like to test a number of metrics to see why people are not staying on a site. I think sometimes we spend so much time focusing on analytics that we have no...
People like to test a number of metrics to see why people are not staying on a site. I think sometimes we spend so much time focusing on analytics that we have no...
Character count


Aliquam sapien nisi, pharetra nec tempor at, semper sed lectus. Nullam
Debugging CSS Media Queries

MAY 11, 2011 · PERMALINK · READ MORE POSTS ABOUT DESIGN AND CSS

IN RESPONSIVE WEB DESIGN we're working with different states, widths, and viewport sizes. Fluidity and adaptive behavior is a hot subject nowadays, and it's perfectly justified when looking at today's mobile browser landscape. We achieve this with CSS's Media Queries. But sometimes it can be messy – I'm gonna share a quick tip for indicating (with pure CSS) which media query that has actually kicked in.

The problem

When I'm in the early markup stage of a site project I usually lay out a solid, fluid grid foundation to build upon. I'm carving out the different sections of the design in rough code and am also playing around with some media queries – deciding how many to use and how. Some debug stuff is used; boxes that change colour when different media queries fires, and so on. But I sort of wanted a more slick way of visually showing how the media queries fired and how conditions had changed.
Typography-Out Approach is an option for building responsive websites.
Typography-Out
Approach
We prepared a series of moodboards to further explore the idea we had, especially to find elements that a new, content-heavy layout could use.
We focused specifically on macro- and microtypography as
Dinosaurs & Science

As anyone who saw movies in the last decade is all too well aware, the delicate relationship between Thoreau and modern science is one that is fragile. Thoreau himself might try to bring the past back rather than their larger and more awesome peers.

For example, one of the smallest known dinosaurs (a tiny pterosaur, to be more precise) had a wingspan of just ten inches. Now, no offense to the little guy, but if my tax dollars are going to go towards dinosaur regeneration, I want them to contribute to something a big more imposing. Perhaps a full-size pteranodon or even the stately brontosaurus?

With regard to the contentious question of herbivores versus carnivores, I believe that we should not be too picky. Let us not give in to our passions on this issue and become distracted from the critical issue of body mass. After all, if regenerated dinosaurs can make the ground tremble beneath our feet, will...
Very wide window (2000px+, Cinema Display, etc.)

Average desktop size — comments drop below article

Small desktop / landscape iPad — comments and ads drop below article
Average desktop size — comments drop below article

Small desktop / landscape iPad — comments and ads drop below article

Portrait iPad and below — top-down, vertical approach
Our design process was defined by one major constraint: **perfect measure.**
If we could adequately typeset an article and thus establish the general context of the design, everything else would follow. […] The key attribute for achieving perfect typesetting was perfect measure: a good average between 45 and 90 characters per line—on all screen resolutions.

— Elliot Jay Stocks, “Smashing Book 3”
So we started looking for typefaces that would best express our new vision, values and our design persona... and oh boy were we excited. We felt like a kid in a candy store... until we had to be thrown into cold shower.

With the abundance of rich typefaces on the Web, we excitedly jumped into the myriad of possibilities. We experimented with literally dozens of typefaces from several type foundries in various pairings: we considered Centro Sans and Centro Serif, Meta and Meta Serif, Adelle, Ronnia, LFT Etica, FF Tisa, just to name a few. We observed how they looked at different font sizes and how well they worked together. We examined how well they appeared in bold and italic and in headings and body copy, as well as how they worked in less obvious contexts such as image captions.
Typefaces should be optimized for long reading on (many) screens and fit various contexts.
The Dark Side of Usability →

People using the more difficult interfaces tended to perform better, were less fazed by distractions and were found more likely to transfer their skills to new interfaces or tasks.

EXPLORE THE ARCHIVES

LEVEL-BASED LEARNING
- Beginner Foundations
- Intermediate Ideas
- Advanced Concepts
<table>
<thead>
<tr>
<th>OS</th>
<th>Visitors</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows</td>
<td>15,022,901</td>
<td>60.4%</td>
</tr>
<tr>
<td>Windows 7</td>
<td>9,818,101</td>
<td>39.5%</td>
</tr>
<tr>
<td>Windows XP</td>
<td>4,006,209</td>
<td>16.1%</td>
</tr>
<tr>
<td>Windows Vista</td>
<td>1,137,177</td>
<td>4.6%</td>
</tr>
<tr>
<td>Windows 8</td>
<td>39,325</td>
<td>0.2%</td>
</tr>
<tr>
<td>Windows</td>
<td>15,746</td>
<td>0.1%</td>
</tr>
<tr>
<td>Windows 2000</td>
<td>5,596</td>
<td></td>
</tr>
<tr>
<td>Windows 98</td>
<td>406</td>
<td></td>
</tr>
</tbody>
</table>

More...

<table>
<thead>
<tr>
<th>OS</th>
<th>Visitors</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mac</td>
<td>7,863,213</td>
<td>31.6%</td>
</tr>
<tr>
<td>Mac OS X</td>
<td>7,863,213</td>
<td>31.6%</td>
</tr>
<tr>
<td>Mobile</td>
<td>1,468,924</td>
<td>5.9%</td>
</tr>
<tr>
<td>iPhone</td>
<td>1,091,114</td>
<td>4.4%</td>
</tr>
</tbody>
</table>
We started out with setting up a couple of demo pages for typography, including links, italics, bold. With all design distractions removed, we could pay a great deal of technical attention to the type once we set up our basic styles. How legible is this typeface at a particular size? How well does it perform on Windows? Is there a superior version from an alternative font delivery network that perhaps uses PostScript outlines for display sizes? Focusing on these fine details is much easier when you’re looking purely at the type and nothing else.

Because there’s a lot going on visually on Smashing Magazine—screenshots, buttons and noisy ads—a sans-serif felt like a more sensible, uncluttered route for body copy. In fact, it was difficult to imagine a serif typeface being used for code-heavy articles in the coding section.
Proxima Nova, rendered by Typekit

*By Mark Simonson.* This would be body copy. Far far away, behind the word mountains, far from the countries Vokalia and Consonantia, there live the blind texts. Separated they live in Bookmarks groove right at the coast of the Semantics, a large language ocean.

Secondary level heading

Further body copy. A small river named Duden flows by their place and supplies it with the necessary regelia. It is a paradisematic country, in which roasted parts of sentences fly into your mouth.

---

Sorry, Proxima Nova rendering is incorrect on this screenshot, but we lost the original files. :-(
New Approaches To Designing Log-In Forms

For many of us, logging into websites is a part of our daily routine. In fact, we probably do it so often that we’ve stopped having to think about how it’s done... that is, until something goes wrong: we forget our password, our user name, the email address we signed up with, how we signed up, or even if we ever signed up at all.

These experiences are not just frustrating for us, but are bad for businesses as well. How bad? User Interface Engineering’s analysis of a major online retailer found that 45% of all customers had multiple registrations in the system, 160,000 people requested their password every day, and 75% of these people never completed the purchase they started once they requested their password.
For many of us, logging into websites is a part of our daily routine. In fact, we probably do it so often that we’ve stopped having to think about how it’s done... that is, until something goes wrong: we forget our password, our user name, the email address we signed up with, how we signed up, or even if we ever signed up at all.
New Approaches To Designing Log-In Forms

By Luke Wroblewski

August 22nd 2011  Forms, UI  62 Comments
New Approaches To Designing Log-In Forms

By Luke Wroblewski

For many of us, logging into websites is a part of our daily routine. In fact, we probably do it so often that we’ve stopped having to think about how it’s done... that is, until something goes wrong: we forget our password, our user name, the email address we signed up with, how we signed up, or even if we ever signed up at all.

These experiences are not just frustrating for us, but are bad for businesses as well. How bad? User Interface Engineering’s analysis of a major online retailer found that 45% of all customers had multiple registrations in the system, 160,000 people requested their password every day, and 75% of these people never completed the purchase they started once they requested their password.

To top it off, visitors who are not logged in do not see a personalized view of a website’s content and recommendations, which reduces conversion rates and engagement. So, log-in is a big deal—big enough that some websites have started exploring new designs solutions for the old problem.
New Approaches To Designing Log-In Forms

By Luke Wroblewski

For many of us, logging into websites is a part of our daily routine. In fact, we probably do it so often that we’ve stopped having to think about how it’s done... that is, until something goes wrong: we forget our password, our user name, the email address we signed up with, how we signed up, or even if we ever signed up at all.

These experiences are not just frustrating for us, but are bad for businesses as well. How bad? User Interface Engineering’s analysis of a major online retailer found that 45% of all customers had multiple registrations in the system, 160,000 people requested their password every day, and 75% of these people never completed the purchase they started once they requested their password.

To top it off, visitors who are not logged in do not see a personalized view of a website’s content and recommendations, which reduces conversion rates and engagement. So, log-in is a big deal—big enough that some websites have started exploring new designs solutions for the old problem.
We started from mobile, and worked our way upwards to desktop views.
Designing The Well-Tempered Web

By Rob Flaherty

January 17th, 2012

Design, User Experience

28 Comments

As technology evolves, so does the art and craft of Web design. New technology creates new challenges, which require new solutions. Often we’re working in uncharted territory, where the solutions demanded really may not. Others...
Designing The Well-Tempered Web

By Rob Flaherty

January 17th, 2012  Design, User Experience  28 Comments

As technology evolves, so does the art and craft of Web design. New technology creates new challenges, which require new solutions. Often we’re working in uncharted territory, where the solutions demanded really are new. Other times, we’re faced with problems of a more universal nature, problems that have a history.

Given the limited history of Web design, we have to look beyond our...
Designing The Well-Tempered Web

By Rob Flaherty

January 17th, 2012  Design, User Experience  28 Comments

As technology evolves, so does the art and craft of Web design. New technology creates new challenges, which require new solutions. Often we’re working in uncharted territory, where the solutions demanded really are new. Other times, we’re faced with problems of a more universal nature, problems that have a history.

Given the limited history of Web design, we have to look beyond our immediate domain for answers to the more challenging questions. We do this all the time when we draw on the rich history of graphic design and visual arts. But we’re not limited to sibling disciplines. If we can identify the abstractions and patterns that constitute our challenges, we can look to any source for guidance. We can look to a seemingly unrelated field, such as psychology or music. We can even look to an episode from the early 18th century about Johann Sebastian Bach.
Designing The Well-Tempered Web

By Rob Flaherty

January 17th, 2012
Design, User Experience
28 Comments

As technology evolves, so does the art and craft of Web design. New technology creates new challenges, which require new solutions. Often we’re working in uncharted territory, where the solutions demanded really are new. Other times, we’re faced with problems of a more universal nature, problems that have a history.

Given the limited history of Web design, we have to look beyond our immediate domain for answers to the more challenging questions. We do this all the time when we draw on the rich history of graphic design and visual arts. But we’re not limited to sibling disciplines. If we can identify the abstractions and patterns that constitute our challenges, we can look to any source for guidance. We can look to a seemingly unrelated field, such as psychology or music. We can even look to an episode from the early 18th century about Johann Sebastian Bach.
Designing The Well-Tempered Web

By Rob Flaherty

January 17th, 2012

Design, User Experience

28 Comments

As technology evolves, so does the art and craft of Web design. New technology creates new challenges, which require new solutions. Often we’re working in uncharted territory, where the solutions demanded really are new. Other times, we’re faced with problems of a more universal nature, problems that have a history.

Given the limited history of Web design, we have to look beyond our immediate domain for answers to the more challenging questions. We do this all the time when we draw on the rich history of graphic design and visual arts. But we’re not limited to sibling disciplines. If we can identify the abstractions and patterns that constitute our challenges, we can look to any source for guidance. We can look to a seemingly unrelated field, such as psychology or music. We can even look to an episode from the early 18th century about Johann Sebastian Bach.

In this article we'll look at what Bach has to do with modern Web challenges.
Designing The Well-Tempered Web

By Rob Flaherty

January 17th, 2012

Design, User Experience
28 Comments

As technology evolves, so does the art and craft of Web design. New technology creates new challenges, which require new solutions. Often we’re working in uncharted territory, where the solutions demanded really are new. Other times, we’re faced with problems of a more universal nature, problems that have a history.

Given the limited history of Web design, we have to look beyond our immediate domain for answers to the more challenging questions. We do this all the time when we draw on the rich history of graphic design and visual arts. But we’re not limited to sibling disciplines. If we can identify the abstractions and patterns that constitute our challenges, we can look to any source for guidance. We can look to a seemingly unrelated field, such as psychology or music. We can even look to an episode from the early 18th century about Johann Sebastian Bach.

In this article we’ll look at what Bach has to do with modern Web challenges — Particularly the challenge of designing for devices with diverse attributes and capabilities.

Bach And “The Well-Tempered Clavier”

In 1722, Bach put together a book of solo keyboard works intended as a collection of educational pieces for young musicians. The book contained 48 pieces—a prelude and fugue in every major and minor key. Now a staple of the Western canon, it’s regarded as one of the most important works in the history of Western music. He named the book The Well-Tempered Clavier.
Designing The Well-Tempered Web

By Rob Flaherty

January 17th, 2012  Design, User Experience  28 Comments

As technology evolves, so does the art and craft of Web design. New technology creates new challenges, which require new solutions. Often we’re working in uncharted territory, where the solutions demanded really are new. Other times, we’re faced with problems of a more universal nature, problems that have a history.

Given the limited history of Web design, we have to look beyond our immediate domain for answers to the more challenging questions. We do this all the time when we draw on the rich history of graphic design and visual arts. But we’re not limited to sibling disciplines. If we can identify the abstractions and patterns that constitute our challenges, we can look to any source for guidance. We can look to a seemingly unrelated field, such as psychology or music. We can even look to an episode from the early 18th century about Johann Sebastian Bach.

In this article we’ll look at what Bach has to do with modern Web challenges — Particularly the challenge of designing for devices with diverse attributes and capabilities.

Bach And “The Well-Tempered Clavier”

In 1722, Bach put together a book of solo keyboard works intended as a collection of educational pieces for young musicians. The book contained 48 pieces — a prelude and fugue in every major and minor key. Now a staple of the Western canon, it’s regarded as one of the most important works in the history of Western music. He named the book The Well-Tempered Clavier.

To appreciate the historical significance of the work, you have to understand that in Bach’s day the notion that one might play keyboard music in all keys was unorthodox. It was a matter not of philosophy, but of physics: a fixed-pitch keyboard instrument could be in tune only with a selection of keys at a time. In the tuning systems of the era, playing in tune in all 12 major keys was simply not possible.
We never targeted specific devices—and introduced media queries whenever it felt natural to do so.
Responsive design affects viewports and media, including **print**.
Beercamp: An Experiment With CSS 3D

By Tom Giannattasio

I recently had the pleasure of organizing this year’s Beercamp website. If you’re unfamiliar, Beercamp is a party for designers and developers. It’s also a playground for front-end experimentation. Each year we abandon browser support and throw a “Pshaw” in the face of semantics so that we can play with some emerging features of modern browsers.

This year’s experiment: a 3D pop-up book à la Dr. Seuss. If you’ve not seen it, hop over and take a look. The website was a test to see how far SVG and CSS 3D transforms could be pushed. I learned a lot in the process and wanted to share some of what I’ve learned.
I recently had the pleasure of organizing this year’s Beercamp website. If you’re unfamiliar, Beercamp is a party for designers and developers. It’s also a playground for front-end experimentation. Each year we abandon browser support and throw a “Pshaw” in the face of semantics so that we can play with some emerging features of modern browsers.

This year’s experiment: a 3D pop-up book à la Dr. Seuss. If you’ve not seen it, hop over and take a look. The website was a test to see how far SVG and CSS 3D transforms could be pushed. I learned a lot in the process and wanted to share some of the techniques that I found helpful when working in 3D space.
FOOTNOTES:

1 this year’s Beercamp website - http://2012.beercamp.com/

2 http://2012.beercamp.com/

3 Sylvester - http://sylvester.jcoglan.com/

4 nclud - http://nclud.com

5 Sylvester - http://sylvester.jcoglan.com/


Tom Giannattasio happily makes things at nclud. He works as an Editor for Smashing Magazine and teaches at Boston University Center for Digital Imaging Arts. He loves to experiment and share his work on his personal site: attasi.

With a commitment to quality content for the design community.
Smashing Media GmbH. Made in Germany. 2006-2012.
http://www.smashingmagazine.com
Responsive Design Patterns
Responsive Design Patterns

- Responsive design affects **all** design assets: layout, images, type, navigation, tables, calendars, galleries, forms, maps, ads...

- Offline access and **mobile UX enhancements** complement RWD very well (e.g. HTML5 localStorage, GeoLocation, Telephone links).
Pull Down for Navigation Demo

See the tutorial here


<table>
<thead>
<tr>
<th>First Name</th>
<th>Last Name</th>
<th>Job Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>James</td>
<td>Matman</td>
<td>Chief S. Eater</td>
</tr>
<tr>
<td>The</td>
<td>Tick</td>
<td>Crime Sorta</td>
</tr>
<tr>
<td>Jokey</td>
<td>Smurf</td>
<td>Giving Present</td>
</tr>
<tr>
<td>Cindy</td>
<td>Beyler</td>
<td>Sales Repres</td>
</tr>
</tbody>
</table>
Selectively displaying data

The number of columns displayed in the table below depends on the available screen space, by default: a smartphone will display 2 columns, for example, while an expanded desktop browser displays the full set. This is accomplished by assigning semantic classes to the column headings that indicate which data values take precedence (essential vs optional), in combination with media queries to display them at different screen widths (a.k.a., responsive design).

We added a bit of JavaScript logic so you can control which data is displayed by checking column names in the “Display” menu on the right. Once an option is checked, the associated data will persist and display at all screen widths until the option is unchecked.

You can also set a column to always persist by assigning a class in the markup, in which case it has no menu option. Here, the “Company” column is persistent.

<table>
<thead>
<tr>
<th>Company</th>
<th>Last Trade</th>
<th>Trade Time</th>
<th>Change</th>
<th>Prev Close</th>
<th>Open</th>
<th>Bid</th>
<th>Ask</th>
<th>1y Target Est</th>
</tr>
</thead>
<tbody>
<tr>
<td>GOOG (Google Inc.)</td>
<td>597.74</td>
<td>12:12PM</td>
<td>14.81 (2.54%)</td>
<td>582.93</td>
<td>597.95</td>
<td>597.73 x 100</td>
<td>597.91 x 300</td>
<td>731.10</td>
</tr>
<tr>
<td>AAPL (Apple Inc.)</td>
<td>378.94</td>
<td>12:22PM</td>
<td>5.74 (1.54%)</td>
<td>373.20</td>
<td>381.02</td>
<td>378.92 x 300</td>
<td>378.99 x 100</td>
<td>505.94</td>
</tr>
<tr>
<td>AMZN (Amazon.com Inc.)</td>
<td>191.55</td>
<td>12:23PM</td>
<td>3.16 (1.68%)</td>
<td>188.39</td>
<td>194.99</td>
<td>191.52 x 300</td>
<td>191.58 x 100</td>
<td>240.32</td>
</tr>
<tr>
<td>ORCL (Oracle Corporation)</td>
<td>31.15</td>
<td>12:44PM</td>
<td>1.41 (4.72%)</td>
<td>29.74</td>
<td>30.67</td>
<td>31.14 x 6500</td>
<td>31.15 x 3200</td>
<td>36.11</td>
</tr>
<tr>
<td>MSFT (Microsoft Corporation)</td>
<td>25.50</td>
<td>12:27PM</td>
<td>0.66 (2.87%)</td>
<td>24.84</td>
<td>25.37</td>
<td>25.50 x 71100</td>
<td>25.51 x 17800</td>
<td>31.50</td>
</tr>
<tr>
<td>CSCO (Cisco Systems, Inc.)</td>
<td>18.65</td>
<td>12:45PM</td>
<td>0.97 (5.49%)</td>
<td>17.68</td>
<td>18.23</td>
<td>18.65 x 10300</td>
<td>18.66 x 24000</td>
<td>21.12</td>
</tr>
<tr>
<td>YHOO (Yahoo Inc.)</td>
<td>15.81</td>
<td>12:25PM</td>
<td>0.11 (0.67%)</td>
<td>15.70</td>
<td>15.94</td>
<td>15.79 x 6100</td>
<td>15.80 x 17000</td>
<td>18.18</td>
</tr>
</tbody>
</table>
Responsive Calendar

This is Demo #1 of the Responsive Calendar Experiment — 3rd January, 2012.

January 2012

<table>
<thead>
<tr>
<th>Monday</th>
<th>Tuesday</th>
<th>Wednesday</th>
<th>Thursday</th>
<th>Friday</th>
<th>Saturday</th>
<th>Sunday</th>
</tr>
</thead>
<tbody>
<tr>
<td>26</td>
<td>27</td>
<td>28</td>
<td>29</td>
<td>30</td>
<td>31</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Event One</td>
<td>Event Two</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>Event Four</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
</tr>
<tr>
<td>16</td>
<td>17</td>
<td>Event Five</td>
<td>18</td>
<td>Event Six</td>
<td>19</td>
<td>20</td>
</tr>
<tr>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
<td>Event Seven</td>
<td>28</td>
</tr>
<tr>
<td>30</td>
<td>31</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>MONDAY</td>
<td>TUESDAY</td>
<td>WEDNESDAY</td>
<td>THURSDAY</td>
<td>FRIDAY</td>
<td>SATURDAY</td>
<td>SUNDAY</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
<td>-------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>GAMMA GALLER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
</tr>
<tr>
<td>MOBILE DESIGN</td>
<td>BARAJA: A PLUGI</td>
<td>CREAT</td>
<td>19</td>
<td>20</td>
<td>21</td>
<td>22</td>
</tr>
</tbody>
</table>
FLEXIBLE CALENDAR DEMO 1 | DEMO 2

NOVEMBER 2012

1 THU

2 FRI  HEADING SET STYLING WITH CSS

3 SAT

4 SUN

5 MON

6 TUE  GAMMA GALLERY: A RESPONSIVE IMAGE GALLERY EXPERIMENT

7 WED

8 THU

9 FRI  RESPONSIVE WORDPRESS THEME GIVEAWAY

10 SAT
<table>
<thead>
<tr>
<th>Date</th>
<th>Day</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Thu</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Fri</td>
<td>HEADING SET STYLING WITH CSS</td>
</tr>
<tr>
<td>3</td>
<td>Sat</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Sun</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Mon</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Tue</td>
<td>GAMMA GALLERY: A RESPONSIV...</td>
</tr>
<tr>
<td>7</td>
<td>Wed</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Thu</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Typeform re-invents the form

because answering questions online doesn't have to be boring.

Find out how press ENTER
Twitter Inspires With Unique Responsive Design

As the number of email opens on mobile devices continues to increase (123% in 18 months!), so does the talk about responsive design. And while I frequently hear about responsive design, I rarely see it in action. In fact, while we definitely make all of our emails “mobile-friendly,” only one of our templates here at Litmus is responsive. Since responsive design involves complex coding, requires more design time, and doesn’t work on all mobile devices, many organizations haven’t started creating mobile-friendly templates — which makes it that much more exciting when I see a responsive email!

After tuning into Yesmail Interactive’s webinar, “Mobile Email Design: Scalable & Responsive,” I was really impressed with (and excited by!) Twitter’s use of responsive design. Here is the example from that presentation:
Email in Apple Mail 6

Justine Jordan,
We have something for you...

New Twitter profiles
Make your profile beautiful with a header image. Browse your new photo reel. Check out what other people are doing with their profiles.

Try it now

Have an iPhone, iPad, Android, Windows Phone 7 or Blackberry? Get the Twitter app at Twitter.com/download

Forgot your Twitter password? Get instructions on how to reset it. You can also unsubscribe from these emails or change your notification settings. Need help? If you received this message in error and did not sign up for Twitter, click not my account.

Twitter, Inc. 1355 Market St., Suite 900 San Francisco, CA 94103

RELEVANT, DIRECT CONTENT & SIMPLE IMAGERY

Before even realizing that it was a responsive design, I was impressed with the email’s content and simplicity. As we’ve stated in so many of our blog posts, relevant content is key to getting your subscriber’s attention and, in this email, serves as a feature update email for Twitter users, the content is concise and to the point.
The responsive design of this email is amazing! CSS media queries (responsive design) can auto-adjust the layout, content, and text size of an email based on the screen size it is being read on; specific rules can be assigned for various screen sizes. When I compared the email on my desktop to the email on my iPhone, I was really impressed. Not only does the layout change for better viewing on a mobile device (changes into a single column for easy viewing on the small screen of a smartphone), the call to action changes as well! The main CTA on the desktop version is “Try it now,” while it’s “Get the app and try it now” on mobile devices.
Responsive Design Patterns

- Media queries aren’t supported in Android 2.1 native client, Gmail app on all platforms, Win Mobile 6.1 & Phone 7, BlackBerry OS 5.

- Also, Webmail services tend to ignore media queries and overwrite them with their custom CSS code.
INTRODUCTION

We're going to walk through how to create an adaptive web experience that's designed mobile-first. This article and demo will go over the following:

- Why we need to create mobile-first, responsive, adaptive experiences
- How to structure HTML for an adaptive site in order to optimize performance and prioritize flexibility
- How to write CSS that defines shared styles first, builds up styles for larger screens with media queries, and uses relative units
- How to write unobtrusive JavaScript to conditionally load in content fragments, take advantage of touch events and geolocation
- What we could do to further enhance our adaptive experience

THE NEED FOR ADAPTIVITY...
Super Ffly T-shirt

$23 ★★★★☆ 8 Reviews

QTY 1 SIZE S

ADD TO CART

SHARE  Ⓟ  Ⓡ  Ⓢ  FIND NEARBY

Behold, your new favorite shirt! This built-to-last tee adapts to your size and capabilities, starting off XS then stretches and scales to keep you comfortable as you grow taller and wider. The future is here, so look sharp.

http://bradfrostweb.com/blog/mobile/anatomy-of-a-mobile-first-responsive-web-design/
This Is Responsive.

Patterns, resources and news for creating responsive web experiences.
There are still many unsolved problems.
There are still many unsolved problems.

Web forms
Images
Performance

Consistency
Debugging
Maintenance
The Very Final Conclusion
Smashing Magazine’s Redesign: A Case Study

Written by Vitaly Friedman and Elliot Jay Stocks
Thank You
For Your Attention!

@smashingmag
Image credits

- **Front cover**: *Geometric Wallpapers*
  by Simon C Page ([http://simoncpage.co.uk/blog/2012/03/ipad-hd-retina-wallpaper/](http://simoncpage.co.uk/blog/2012/03/ipad-hd-retina-wallpaper/))