A LIST APART

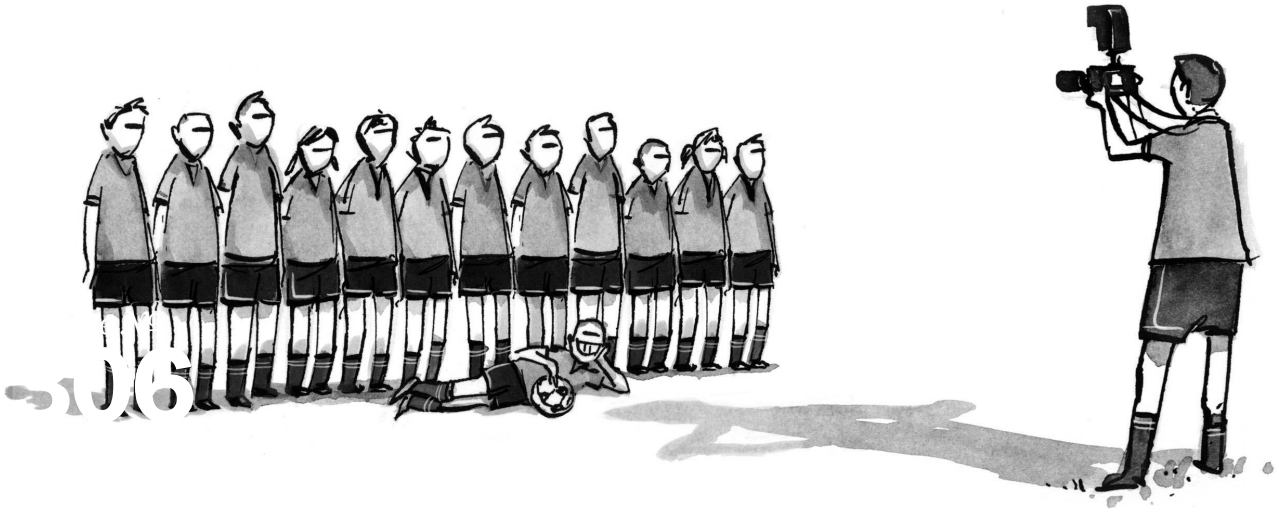


Illustration by Kevin Cornell

Responsive Web Design

by **Ethan Marcotte** · May 25, 2010

Published in CSS, Layout & Grids, Mobile/Multidevice, Responsive Design, Interaction Design

The English architect Christopher Wren once quipped that his chosen field “aims for Eternity,” and there’s something appealing about that formula: Unlike the web, which often feels like aiming for next week, architecture is a discipline very much defined by its permanence.

A building’s foundation defines its footprint, which defines its frame, which shapes the facade. Each phase of the architectural process is more immutable, more unchanging than the last. Creative decisions quite literally shape a physical space, defining the way in which people move through its confines for decades or even centuries.

Working on the web, however, is a wholly different matter. Our work is defined by its transience, often refined or replaced within a year or two. Inconsistent window widths, screen resolutions, user preferences, and our users’ installed fonts are but a few of the intangibles we negotiate when we publish our work, and over the years, we’ve become incredibly adept at doing so.

But the landscape is shifting, perhaps more quickly than we might like. Mobile browsing is expected to outpace desktop-based access within three *(http://www.mediapost.com/publications/?fa=Articles.showArticle&art_aid=120590)* to five years *(http://www.morganstanley.com/institutional/techresearch/mobile_internet_report122009.html)*. Two of the three dominant video game consoles have web browsers (and one of them *(http://www.nintendo.com/consumer/systems/wii/en_na/channelsInternet.jsp)* is quite excellent). We're designing for mice and keyboards, for T9 keypads, for handheld game controllers, for touch interfaces. In short, we're faced with a greater number of devices, input modes, and browsers than ever before.

In recent years, I've been meeting with more companies that request "an iPhone website" as part of their project. It's an interesting phrase: At face value, of course, it speaks to mobile WebKit's quality as a browser, as well as a powerful business case for thinking beyond the desktop. But as designers, I think we often take comfort in such explicit requirements, as they allow us to compartmentalize the problems before us. We can quarantine the mobile experience on separate *(http://webkit.dailykos.com/)* subdomains *(http://m.nytimes.com/)*, spaces distinct and separate from "the non-iPhone website." But what's next? An iPad website? An N90 website? Can we really continue to commit to supporting each new user agent with its own bespoke experience? At some point, this starts to feel like a zero sum game. But how can we—and our designs—adapt?

A flexible foundation

Let's consider an example design *(/d/responsive-web-design/ex/ex-site-flexible.html)*. I've built a simple page for a hypothetical magazine; it's a straightforward two-column layout built on a fluid grid *(http://www.alistapart.com/articles/fluidgrids/)*, with not a few flexible images *(http://unstoppablerobotninja.com/entry/fluid-images)* peppered throughout. As a long-time proponent of non-fixed layouts, I've long felt they were more "future proof" simply because they were layout agnostic. And to a certain extent, that's true: flexible designs make no assumptions about a browser window's width, and adapt beautifully to devices that have portrait and landscape modes.

Huge images are huge. Our layout, flexible though it is, doesn't respond well to changes in resolution or viewport size.

But no design, fixed or fluid, scales seamlessly beyond the context for which it was originally intended. The example design *(/d/responsive-web-design/ex/ex-site-flexible.html)* scales perfectly well as the browser window resizes, but stress points quickly appear at lower resolutions. When viewed at viewport smaller than 800×600, the illustration behind the logo quickly becomes cropped, navigation text can wrap in an unseemly manner, and the images along the bottom become too compact to appear legible. And it's not just the lower end of the resolution spectrum that's affected: when viewing the design on a widescreen display, the images quickly grow to unwieldy sizes, crowding out the surrounding context.

In short, our flexible design works well enough in the desktop-centric context for which it was designed, but isn't optimized to extend far beyond that.

Becoming responsive

Recently, an emergent discipline called "*responsive architecture*" has begun asking how physical spaces can *respond* to the presence of people passing through them. Through a combination of embedded robotics and tensile materials, architects are experimenting with art installations *(
 http://www.robotecture.com/bubbles-417/)* and wall structures *(http://vimeo.com/4661618)* that bend, flex, and expand as crowds approach them. Motion sensors can be paired with climate control systems to adjust a room's temperature and ambient lighting as it fills with people. Companies have already produced "smart glass technology" that can automatically become opaque *(http://www.smartglassinternational.com/)* when a room's occupants reach a certain density threshold, giving them an additional layer of privacy.

In their book *Interactive Architecture*, Michael Fox and Miles Kemp described this more adaptive approach as "a multiple-loop system in which one enters into a conversation; *a continual and constructive information exchange*." Emphasis mine, as I think that's a subtle

yet powerful distinction: rather than creating immutable, unchanging spaces that define a particular experience, they suggest inhabitant and structure can—and should—mutually influence each other.

This is our way forward. Rather than tailoring disconnected designs to each of an ever-increasing number of web devices, we can treat them as facets of the same experience. We can design for an optimal viewing experience, but embed standards-based technologies into our designs to make them not only more flexible, but more adaptive to the media that renders them. In short, we need to practice *responsive web design*. But how?

Meet the media query

Since the days of CSS 2.1, our style sheets have enjoyed some measure of device awareness through *media types (http://www.w3.org/TR/CSS21/media.html)*. If you've ever written a print style sheet *(http://www.alistapart.com/articles/goingtoprint/)*, you're already familiar with the concept:

```
<link rel="stylesheet" type="text/css"
href="core.css"
  media="screen" />
<link rel="stylesheet" type="text/css"
href="print.css"
  media="print" />
```

In the hopes that we'd be designing more than neatly formatted page printouts, the CSS specification supplied us with a bevy of acceptable media types *(http://www.w3.org/TR/CSS21/media.html#media-types)*, each designed to target a specific class of web-ready device. But most browsers and devices never really embraced the spirit of the specification, leaving many media types implemented imperfectly *(http://www.alistapart.com/articles/return-of-the-mobile-stylesheet)*, or altogether ignored.

Thankfully, the W3C created *media queries (http://www.w3.org/TR/css3-mediaqueries/)* as part of the CSS3 specification, improving upon the promise of media types. A media query allows us to target not only certain device classes, but to actually inspect the physical characteristics of the device rendering our work. For example, following the recent rise of mobile WebKit, media queries became a popular client-side technique for delivering a tailored style sheet to the iPhone, Android phones, and their ilk. To do so, we could incorporate a query into a `link`ed style sheet's `media` attribute:

```
<link rel="stylesheet" type="text/css"
  media="screen and (max-device-width: 480px)"
  href="shetland.css" />
```

The query contains two components:

1. a media type (`screen`), and
2. the actual query enclosed within parentheses, containing a particular *media feature* (`max-device-width`) to inspect, followed by the target value (`480px`).

In plain English, we're asking the device if its horizontal resolution (`max-device-width`) is equal to or less than `480px` . If the test passes—in other words, if we're viewing our work on a small-screen device like the iPhone—then the device will load `shetland.css` . Otherwise, the `link` is ignored altogether.

Designers have experimented with resolution-aware layouts in the past, mostly relying on JS-driven solutions like Cameron Adams' excellent script *(http://www.themaninblue.com/experiment/ResolutionLayout/)*. But the media query specification provides a host of media features *(http://www.w3.org/TR/css3-mediaqueries/#media1)* that extends far beyond screen resolution, vastly widening the scope of what we can test for with our queries. What's more, you can test multiple property values in a single query by chaining them together with the `and` keyword:

```
<link rel="stylesheet" type="text/css"
  media="screen and (max-device-width: 480px) and
(resolution: 163dpi)"
  href="shetland.css" />
```

Furthermore, we're not limited to incorporating media queries in our `link` s. We can include them in our CSS either as part of a `@media` rule:

```
@media screen and (max-device-width: 480px) {
  .column {
    float: none;
  }
}
```

Or as part of an `@import` directive:

```
@import url("shetland.css") screen and (max-
device-width: 480px);
```

But in each case, the effect is the same: If the device passes the test put forth by our media query, the relevant CSS is applied to our markup. Media queries are, in short, conditional comments *(http://msdn.microsoft.com/en-us/library/ms537512(VS.85).aspx)* for the rest of us. Rather than targeting a specific version of a specific browser, we can surgically correct issues in our layout as it scales beyond its initial, ideal resolution.

Adapt, respond, and overcome

Let's turn our attention to the images at the base of our page. In their default layout, the relevant CSS currently looks like this:

```
.figure {
  float: left;
  margin: 0 3.31753554502369682% 1.5em 0;   /*
21px / 633px */
  width: 31.121642969984202211%;            /*
197px / 633px */
}li#f-mycroft,
li#f-winter {
  margin-right: 0;
}
```

I've omitted a number of typographic properties to focus on the layout: Each `.figure` element is sized at roughly one third of the containing column, with the right-hand margin zeroed out for the two pictures at the end of each row (`li#f-mycroft, li#f-winter`). And this works fairly well, until the viewport is either noticeably smaller or wider than our original design. With media queries, we can apply resolution-specific spotfixes, adapting our design to better respond to changes in the display.

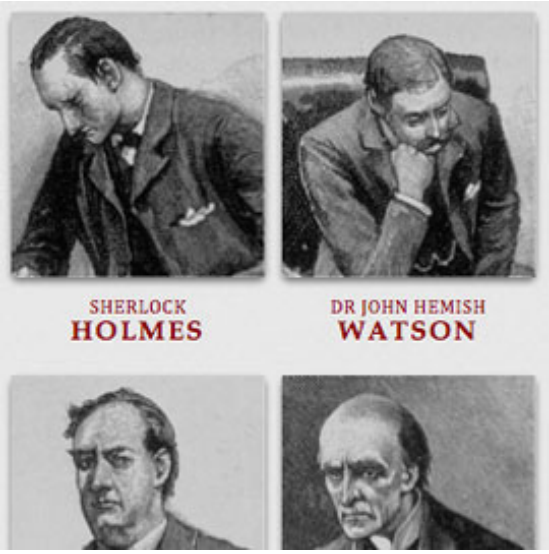
First of all, let's linearize our page once the viewport falls below a certain resolution threshold —say, `600px` . So at the bottom of our style sheet, let's create a new `@media` block, like so:

```
@media screen and (max-width: 600px) {
  .mast,
  .intro,
```

```
  .main,
  .footer {
    float: none;
    width: auto;
  }
}
```

If you view our updated page *(/d/responsive-web-design/ex/ex-site-linearize.html)* in a modern desktop browser and reduce the size of your window below `600px`, the media query will disable the floats on the design's major elements, stacking each block atop each other in the document flow. So our miniaturized design is shaping up nicely, but the images still don't scale down that intelligently. If we introduce another media query, we can alter their layout accordingly:

```
@media screen and (max-width: 400px) {
  .figure,
  li#f-mycroft {
    margin-right: 3.31753554502369682%;    /*
21px / 633px */
    width: 48.341232227488151658%;         /*
306px / 633px */
  }  li#f-watson,
  li#f-moriarty {
    margin-right: 0;
  }
}
```



Our figures can responsively change their layout to better suit smaller displays.

Don't mind the unsightly percentages; we're simply recalculating the widths of the fluid grid *(http://www.alistapart.com/articles/fluidgrids/)* to account for the newly linearized layout. In short, we're moving from a three-column layout to a two-column layout *(/d/responsive-web-design/ex/ex-site-mini.html)* when the viewport's width falls below `400px`, making the images more prominent.

We can actually take the same approach for widescreen displays, too. For larger resolutions, we could adopt a six-across treatment for our images, placing them all in the same row *(/d/responsive-web-design/ex/ex-site-larger.html)*:

```
@media screen and (min-width: 1300px) {
  .figure,
  li#f-mycroft {
    margin-right: 3.31753554502369668%;    /*
21px / 633px */
    width: 13.902053712480252764%;         /*
88px / 633px */
  }
}
```

Now our images are working beautifully at both ends of the resolution spectrum *(/d/responsive-web-design/ex/ex-site-larger.html)*, optimizing their layout to changes in window widths and device resolution alike.

By specifying a wider `min-width` in a new media query, we can shift our images into a single row layout.

But this is only the beginning. Working from the media queries we've embedded in our CSS, we can alter much more than the placement of a few images: we can introduce new, alternate layouts *(/d/responsive-web-design/ex/ex-site-FINAL.html)* tuned to each resolution range,

perhaps making the navigation more prominent in a widescreen view, or repositioning it above the logo on smaller displays.

By designing responsively, we can not only linearize our content on smaller devices, but also optimize its presentation across a range of displays.

But a responsive design isn't limited to layout changes. Media queries allow us to practice some incredibly precise fine-tuning as our pages reshape themselves: we can increase the target area on links for smaller screens, better complying with Fitts' Law *(http://en.wikipedia.org/wiki/Fitts'_law)* on touch devices; selectively show or hide elements that might enhance a page's navigation; we can even practice responsive typesetting *(/d/responsive-web-design/ex/ex-article.html)* to gradually alter the size and leading of our text, optimizing the reading experience for the display providing it.

A FEW TECHNICAL NOTES

It should be noted that media queries enjoy incredibly robust support among modern browsers. Desktop browsers such as Safari 3+, Chrome, Firefox 3.5+, and Opera 7+ all natively parse media queries, as do more recent mobile browsers such as Opera Mobile and mobile WebKit. Of course, older versions of those desktop browsers don't support media queries. And while Microsoft has committed to media query support in IE9 *(http://ie.microsoft.com/testdrive/HTML5/85CSS3_MediaQueries/Default.html)*, Internet Explorer currently doesn't offer a native implementation.

However, if you're interested in implementing legacy browser support for media queries, there's a JavaScript-tinted silver lining:

- A jQuery plugin *(http://plugins.jquery.com/project/MediaQueries)* from 2007 offers somewhat limited media query support, implementing only the `min-width` and `max-width` media properties when attached to separate `link` elements.
- More recently, css3-mediaqueries.js *(http://code.google.com/p/css3-mediaqueries-js/)* was released, a library that promises "to make IE 5+, Firefox 1+ and Safari 2 transparently parse, test, and apply CSS3 Media Queries" when included via `@media` blocks. While very much a 1.0 release, I've personally found it to be quite robust, and I plan to watch its development.

But if using JavaScript doesn't appeal, that's perfectly understandable. However, that strengthens the case for building your layout atop a flexible grid *(http://www.alistapart.com/articles/fluidgrids/)*, ensuring your design enjoys some measure of flexibility in media query-blind browsers and devices.

The way forward

Fluid grids, flexible images, and media queries are the three technical ingredients for responsive web design, but it also requires a different way of thinking. Rather than quarantining our content into disparate, device-specific experiences, we can use media queries to progressively enhance our work within different viewing contexts. That's not to say there isn't a business case for separate sites geared toward specific devices; for example, if the user goals for your mobile site are more limited in scope than its desktop equivalent, then serving different content to each might be the best approach.

But that kind of design thinking doesn't need to be our default. Now more than ever, we're designing work meant to be viewed along a gradient of different experiences. Responsive web design offers us a way forward, finally allowing us to "design for the ebb and flow of things."

About the Author

Ethan Marcotte

Ethan Marcotte is an independent web designer who cares deeply about beautiful design, elegant code, and the intersection of the two. Over the years, his clientele has included New York Magazine, the Sundance Film Festival, The Boston Globe, and People Magazine. Ethan can be found on Twitter or on his long-neglected blog, and his most recent book is Responsive Web Design.

MORE FROM THIS AUTHOR

Fluid Images *(fluid-images)*, Fluid Grids *(fluidgrids)*, Where Our Standards Went Wrong *(whereourstandardswentwrong)*, Frameworks *(frameworks)*

ISSN 1534-0295