

Developing Mobile Websites

Designing for Mobile

Lesson 1, Activity 2: **Responsive Design: Good for All Devices**

Responsive web design is the process of building Web pages to work well for all viewing environments: whether users access our sites on an iPhone or Android smartphone, an iPad tablet, a small laptop, or a desktop computer with a huge monitor, we want the site to be optimally viewable. A few years ago, we might have crafted a "mobile version" of our site - sniffing the incoming http request to find out the type of browser and device asking for our page, and sending back wholly different sites in response. Nowadays, we create one site - built upon a flexible grid layout, employing CSS3 media queries, and often utilizing some JavaScript to enhance the user experience - that presents our pages in a manner best suited for the size and other properties of the screen being used.

Why It's Worth It?

Of course, creating a responsive site is more difficult than creating nonresponsive sites. We're designing more than one look for the site: a sidebar floating right for users on a desktop might slide to the bottom of the page for mobile users; a nav element laid out horizontally on large screens might stack vertically for small screens. We have to develop CSS stylesheets with media queries to render the different designs, with percentage-based (rather than pixel-based) widths, and with appropriate rules to shrink down images and other fixed-width elements.

So why go to the trouble? We all know that more and more users now access Web sites via mobile devices - and the numbers will only grow as

time moves on. Recent studies like [these](#) highlight the business case for taking the time to present users with mobile-optimized sites: phone-friendly sites generate more traffic, better user engagement, and more money.

Building Responsive Sites

A number of methodologies, technologies, and tools are involved in building responsive sites. Here's an overview of the key strategies:

Flexible Grid

A first step toward a responsive design is a fluid grid layout. Instead of setting widths for HTML elements like divs in pixels, we will instead use percentages. Instead of a 980-pixel, fixed-width, center-justified design - a page which remains always at 980 pixels wide, regardless of how we view it - we'll use a fluid layout, where the page scales to fill a set percentage of the screen, and elements within the design scale bigger and smaller (but remain in proportion) as the size of the device or browser changes.

Flexible Images

A flexible grid design, in which elements shrink to accommodate different browser or screen widths, would fail miserably if images and other fixed-width elements didn't scale, too. We'll use the `max-width:100%` CSS directive to effect this.

Media Queries

CSS3 media queries allow us to target various properties of the user's device - screen size, browser width, and more. We'll use media queries

extensively to craft different presentations of our sites - to change a two-column layout for desktop users to a one-column layout for mobile users, for instance.

Frameworks

We'll explore how [jQuery Mobile](#) - the JavaScript framework built on top of the popular jQuery framework - can help us "design a single highly branded Web site or application that will work on all popular smartphone, tablet, and desktop platforms." Similarly, we'll check out [Foundation](#), the grid-based responsive layout framework from Zurb.

Further Reading

[Ethan Marcotte's article on A List Apart](#) is a great overview of responsive design. Do also check out this article on [Think Vitamin](#) and this article from [Smashing Magazine](#).

Lesson 1, Activity 3: *The New York Times* and *Boston Globe* Web Sites

Duration: 10 to 20 minutes.

In this exercise, you will evaluate how the Web sites for two major American newspapers differ in responsiveness. Can you find the code behind the differences in the sites' behaviors?

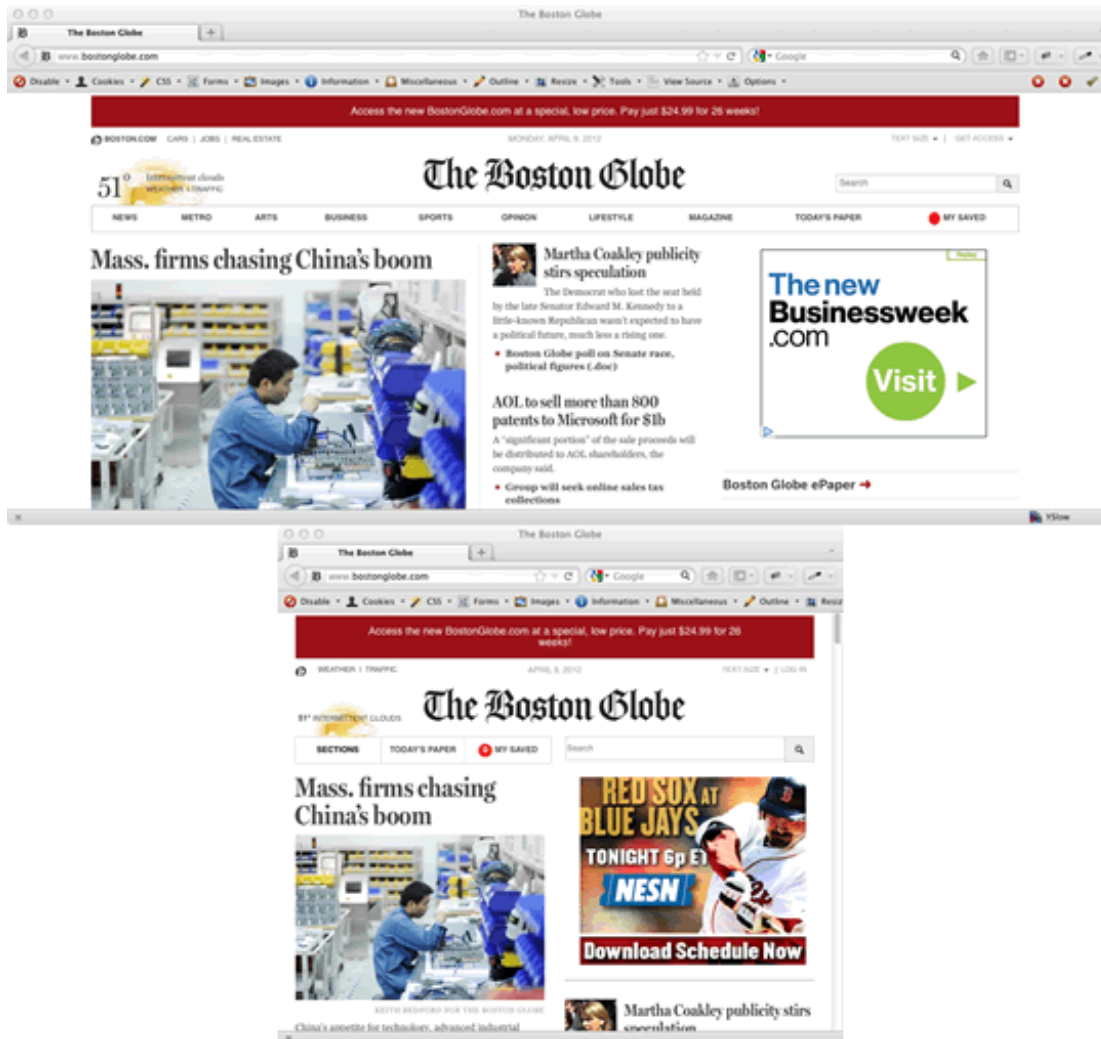
Your work on this exercise - and on work throughout the course - will be easier if you use a tool like [Firebug](#), the free Firefox plug-in that offers right-click inspection of markup and CSS elements. The Google Chrome browser has some of this functionality built in.

1. Open the [New York Times](#) and [Boston Globe](#) Web sites in a browser.
2. Drag to make your browser wider and narrower - check how each site behaves at different browser widths.
3. Inspect the HTML and CSS elements on each site; specifically:
 1. How does the outer shell of the *Boston Globe* site change width as the browser resizes?
 2. Why does the outside shell of the *New York Times* site stay a fixed width?
 3. How is the *Boston Globe* site's column-layout-change (from three to two to one column, with narrowing of the browser) accomplished?

Solution:

The first thing we notice is that the *Boston Globe* site changes as we make the browser narrower: the width of the entire design decreases, columns slide under each other, and the nav changes. The screen shots

below shows the *Boston Globe* home page on the same browser. The top browser is about 1500 pixels wide; the bottom is about 750 pixels wide:



The *New York Times* site stays the same width regardless of whether we change the width of the browser; right content gets cut off as we resize narrower and narrower. Here are two screen shots, again at 1500 and 750 pixel browser widths:



If we check under the hood, we see that the outermost element on the NY Times site gets a fixed CSS width:

```
#shell {
  margin: 0 auto;
  text-align: left;
  width: 972px;
}
```

The *Boston Globe's* outer container, on the other hand, is set to be a percentage of the browser's current width - about 94% of the total screen real estate available:

```
#contain {
```

```
margin: 0 auto 10px;  
width: 93.75%;  
}
```

On the *Boston Globe* site, it's not just that the columns get proportionately narrower as the screen resizes; rather, the design changes at some points - three columns become two columns become (at smallest widths) one column. How do they do this? With CSS3 media queries, setting different style rules for different media types, browser widths, and device properties. For instance, the following rule (from a stylesheet on the *Boston Globe* site) says "set the right and left padding on links contained within any list item inside of an element of class `sections` to have right and left padding of 8px":

```
@media screen and (min-width: 900px){  
  .sections li a {  
    padding-right: 8px;  
    padding-left: 8px;  
  }  
}
```

We will learn much more about media queries later in this course.