Responsive Web Design Tutorial
(fundamental concepts, without Bootstrap)

Responsive Web Design

Viewport

Design Mobile First then Apply Media Queries

Responsive Two-Up Desktop Layout versus One-Up Mobile Layout

First Page Hero and Response Form Styling

Responsive Image Styling

Responsive Tabular Data

Written by: Sally Kyvernitis, Temple University, 2-11-2016

1 Tutorial source https://developers.google.com/web/fundamentals/getting-started/your-first-multi-screen-site/responsive?hl=en (reworked and simplified)
Responsive Web Design

Responsive web design (RWD)² means designing sites to provide for optimal viewing and user interaction (easy to read and navigate with a minimum of resizing, panning, and scrolling) across a wide range of devices (from desktop computer monitors to mobile phones).

A site designed with RWD adapts the layout to the viewing environment by using fluid, proportion-based grids, flexible images, and CSS3 media queries, in the following ways:

- The fluid grid concept prevents horizontal scrolling by sizing elements in relative units like percentages, rather than absolute units like pixels.
- Flexible images are also sized in relative units, so as to prevent them from displaying outside their containing element.
- Media queries allow the page to use different CSS style rules based on characteristics of the device, like browser width.

Since mobile traffic now accounts for more than half of total internet traffic, Google boosts the ratings of mobile friendly sites if the search is made from a mobile device. This has the net effect of penalizing sites that are not mobile friendly.

JavaScript frameworks like Modernizr, jQuery, and jQuery Mobile can directly test browser support for HTML/CSS features (or identify the device or user agent) are popular.

Viewport

All pages must include a viewport meta tag. The viewport indicates to the browser that the page needs to be scaled to fit the screen. Although there are various options, we recommend this as a default for you to use:

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

Design Mobile First then Apply Media Queries

The recommended approach to responsive web design is "mobile first", which means you begin by styling your page for your smallest width, then you add a "media query" to redefine those CSS rules that need to be modified for the next wider screen and then the next wider, until you are done.

A media query first specifies media (like "screen" or "print), then a device width range, and possibly more factors (like device pixel ratio, orientation), and finally a set of braces inside which you place the CSS rules you want to apply to those devices. To implement your own responsive design, start out by deciding how you want to group the devices of users viewing your site. Of course, you would want to create a small number (2-4?) of groups, but you might want to look at detailed specifications of particular devices to help you decide where to set your "media query breakpoints". Here is an example of a media query (which is way too specific, but it shows the syntax of a media query and it also shows the level of detail that is available about current devices, if you are interested).

```css
/* From https://css-tricks.com/snippets/css/media-queries-for-standard-devices/ */
/* iPhone 4 and 4S Portrait */
@media only screen
  and (min-device-width: 320px)
  and (max-device-width: 480px)
  and (-webkit-min-device-pixel-ratio: 2)
  and (orientation: portrait)
{
  /* styles */
}
```

² Definition from https://en.wikipedia.org/wiki/Responsive_web_design (abbreviated)
To keep this tutorial simple, we will design for just two categories:

- Mobile (anything less than 600px) – no media query will be used since we are using the "mobile first" design approach.
- For Desktop, we will use this media query: `@media screen and (min-width: 600px) {
  /* styles */
}

For this tutorial, we will work on a "single page layout" that has several (4) "virtual pages" when shown in desktop layout (see below). As you can see, the desktop version has a two-up layout while the mobile version has a one-up layout.

Mobile

Desktop

...more images...

followed by rearranged table data
Responsive Two-Up Desktop Layout versus One-Up Mobile Layout

To keep things clear (and so that I could place the two style sheets side by side while I worked), I created two style sheets:

- one for mobile (has no media queries - contains initial styling, and any styling that is in common between the 2 layouts) and
- one for desktop (all styles are inside of a single media query specifies desktop widths).

My HTML code references the mobile/common style sheet then the desktop style sheet which redefine rules only where needed.

```
<head>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link href="mobile.css" rel="stylesheet" type="text/css" />
  <link href="desktop.css" rel="stylesheet" type="text/css" />
</head>
```

Terminology: this is a CSS Rule Set, with two CSS Rules in it.

To accomplish a one-up to two-up transformation, these are my suggestions (after studying and simplifying the Google code\(^3\)).

1. Put your most common CSS rules inside of a body selector within your mobile/common style sheet. By placing CSS rules at the highest level, you will have less code to copy/paste, and more consistency in your layout.

2. Create a container class that will hold either the one-up or two-up content for each "virtual" page. For mobile, you'll set its width is 100%, but in desktop, you'll prevent it from getting too large and make sure that it is centered on the page (margin:auto).

3. Create a class named "left" to style the first half of each two-up virtual page and a class named "right" to style the second half. Notice that we float the "left" class left, make it 49% wide and provide a 1% right margin. For the "right" class, we do the opposite (float it right with width 49% and left margin of 1%). Just as we saw with our first layout examples (where the "City Gallery" title was floated left and the Nav bar floated right, both inside of a titleNav element), we need a "clear:both" rule to stop the floating at the end of every virtual page. Since the desktop layout is the only version that needs to float elements (left/right), the mobile style sheet will not define the left, right, and clear styles.

```
body {
  margin: 0; padding: 0; /* avoid white line just inside browser window */
  /* set up default text properties - most commonly used */
  font-family: Tahoma, 'Arial Black', sans-serif;
  font-size: 18px;
}
.container { width: 100%; }
/* more rule sets */
```

<table>
<thead>
<tr>
<th>mobile.css</th>
<th>desktop.css</th>
</tr>
</thead>
<tbody>
<tr>
<td>body {</td>
<td>@media screen and (min-width: 600px) {</td>
</tr>
<tr>
<td>margin: 0; padding: 0; /* avoid white line just inside browser window */</td>
<td></td>
</tr>
<tr>
<td>/* set up default text properties - most commonly used */</td>
<td></td>
</tr>
<tr>
<td>font-family: Tahoma, 'Arial Black', sans-serif;</td>
<td>.container { margin: auto; max-width: 1000px; }</td>
</tr>
<tr>
<td>font-size: 18px;</td>
<td>.left { float: left; width: 49%; margin-right: 1%; }</td>
</tr>
<tr>
<td>}</td>
<td>.right { float: right; width: 49%; margin-left: 1%; }</td>
</tr>
<tr>
<td>.container { width: 100%; }</td>
<td>.clear { clear: both; }</td>
</tr>
<tr>
<td>/* more rule sets */</td>
<td>/* more rule sets */</td>
</tr>
</tbody>
</table>

This HTML code changes from one-up (mobile) to two-up (desktop) by using the left, right, and clear styles.

```
<div class="container">
  <h2>What will I learn?</h2>
  <p>After completing this class, you'll have built a web application with a first-class mobile experience.</p>
  <div class="left">
    <p>You'll understand what it takes to ...</p>
  </div>
  <div class="right">
    <video controls="">
      <source src="http://media.w3.org/2010/05/sintel/trailer.mp4" type="video/mp4"/>
      <source src="http://media.w3.org/2010/05/sintel/trailer.webm" type="video/webm"/>
    </video>
  </div>
  <div class="clear"></div>
</div>
```

In my html, I wrapped each "virtual page" (container class) inside of a div where I applied a "section" class (to allow for common styling of all the sections) but also a unique id to allow for styling anything special about that particular page (like different background color or different text alignment).

```html
<div class="section" id="section1">
  <div class="container">... </div>
</div>

<div class="section" id="section2">
  <div class="container">... </div>
</div>

<div class="section" id="section3">
  <div class="container">... </div>
</div>
```

Take a look at the "version 1" code example (v1.html, v1_mobile.css and v1_desktop.css). This version only shows the following page (and excludes styling for the other pages so it is easier to follow). To make this page change from one-up to two-up when the browser gets wider, its container class has a div styled "right", a div styled "left", followed by one styled "clear".

![Mobile](image1)

![Desktop](image2)

Getting the video to look nice only required this one rule set in mobile.css (see below). The video looks good in mobile width because it is 100% wide. When the desktop layout kicks in, the video still looks good, filling up 100% of its right/left container which is now only about 50% of the width of the screen.

```css
video {
  width:100%;
  height: auto;
  border-radius:16px;
  border:2px solid gray;
}
```

First Page Hero and Response Form Styling

If you look at the final version of code (final.html, mobile.css, desktop.css), you will see that the first page is styled with class "headSection" instead of just "section". The only difference between the two classes is that the "headSection" class has its headers styled larger (h1, h2, …) – there is nothing responsive about the "headSection" class. The registration form (id="register") has some nice styling, but it also does not have any responsive behavior. The responsiveness of the first page is achieved by nesting the opening text and registration form into divs that are classed "left" and "right" (respectively), followed by a a "clear" div, just as it was done for the "What will I learn?" page explained above.
Responsive Image Styling

The "Who will teach me?" page translates nicely from mobile to desktop (as shown below), going from one-up to three-up layout. The completed code for this is provided in the final code example (final.html, mobile.css, desktop.css).

Mobile

![Mobile layout]

... more images...

Desktop

![Desktop layout]

The approach was very similar to what we used for the one-up to two-up transformation, as you can see from the code below. In the HTML code, each image is surrounded by an imageContainer class which holds the image and the caption below it. In mobile layout, the image is 100% wide inside of its imageContainer. The imageContainer class has a little styling for padding but is also 100% wide inside of its container, so the images fill up the width of the mobile device. In desktop layout, the only change is that the imageContainer floats left and is 33% wide (to get the three-up effect). We add the "clear" div (after all the imageContainers) to stop the floating behavior. There is also a stylistic change from square images in mobile to round in desktop.

<table>
<thead>
<tr>
<th>HTML Code</th>
<th>mobile.css</th>
<th>desktop.css</th>
</tr>
</thead>
</table>
| <div class="section" id="section2">
  <div class="container">
    <h2>Who will teach me?</h2>
    <p>The worlds leading mobile web developers.</p>
    <div class="imageGroup">
      <div class="imageContainer">
        <img src="img/chriswilson.png">
        <br>Chris Wilson
      </div>
      <div class="imageContainer">
        <img src="img/peterlubbers.png">
        <br>Peter Lubbers
      </div>
      <div class="imageContainer">
        <img src="img/seanbennett.png">
        <br>Sean Bennett
      </div>
    </div>
    <div class="clear"></div>
  </div>
</div> <!-- image group -->
<br>
<div class="clear"></div>
</div> <!-- end container -->
</div> <!-- end section --> |
| .imageContainer {
  width: 100%;
  padding-left: 4%;
  padding-right: 4%;
  padding-bottom: 24px;
/* border-box includes padding & border in the total width & height of box */
  box-sizing: border-box;
}
| .imageContainer img {
  width: 100%;
  box-sizing: border-box;
  border-radius: 50%;
  box-shadow: black 0px 0px 5px;
  margin: 1%;
  margin-bottom:8px;
} |
| @media screen and (min-width: 600px) {
  .imageContainer {
    width: 33%;
    float:left;
  }
  .imageContainer img {
    border-radius: 50%;
  }
} |
Responsive Tabular Data

Tabular data can be hard to read in mobile formats. The Google Tutorial\textsuperscript{4} showed a technique to convert the tabular data (below right) to a format (below left) where the column headings become row headings that are repeated for each of the original data rows. Google's code for that is provided below.

<table>
<thead>
<tr>
<th>Mobile</th>
<th>Desktop</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Mobile Table" /></td>
<td><img src="image" alt="Desktop Table" /></td>
</tr>
</tbody>
</table>

### Google's HTML Code

```html
<table>
  <caption>Data from StateCounter</caption>
  <thead>
    <tr>
      <th>Country</th>
      <th>Desktop share</th>
      <th>Tablet share</th>
      <th>Mobile share</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td data-th="Country">India</td>
      <td data-th="Desktop share">32%</td>
      <td data-th="Tablet share">1%</td>
      <td data-th="Mobile share">67%</td>
    </tr>
    <tr>
      <td data-th="Country">GB</td>
      <td data-th="Desktop share">69%</td>
      <td data-th="Tablet share">13%</td>
      <td data-th="Mobile share">18%</td>
    </tr>
  </tbody>
</table>
```

### mobile.css (nothing in desktop.css)

```css
@media screen and (max-width: 600px) { /* mobile only */
  table thead {
    display: none;
  }
  table td {
    display: block;
    position: relative;
    padding-left: 13px;
    padding-top: 13px;
    padding-bottom: 13px;
    text-align: left;
    background: #e9e9e9;
    content: attr(data-th) " :";
    display: inline-block;
    color: #000000;
    border-right: 2px solid transparent;
    position: absolute;
    top: 0;
    left: 0;
    bottom: 0;
    width: 13%;
    max-height: 100%;
    font-size: 16px;
    font-weight: 300;
    padding-left: 13px;
    padding-top: 13px;
  }
  table td:before {
    display: none;
  }

  table { margin-top: 26px; width: 100%; }
  thead { font-weight: bold; }
  tbody { text-align: center; }
}
```