MMP 350 Week 4

Table of contents

Goals for Week

Presentation(s)
Quiz Review
Possible Quiz
Review of media queries
Review of columns and grids
In class exercises
Confirmation of WordPress Setup in Cloud 9
Brief Lecture: Content Management Systems
Discussion of Readings: Agile versus Waterfall Project Management

Review of Homework

Correction

In Week 3 notes I incorrectly stated that increasing margins increases the size of the containing element. This is not quite true. Increasing the margins of a child element can cause it to become wider and / or longer than its containing element. The containing element remains the same size. The space taken up by the container and its child increases.

Media Query Review

Overview

A **media query** consists of an optional media type (for example screen, printer) and zero or more expressions that limit the style sheets' scope based on properties such as width, height, and color. Media queries allow designers to separate content from presentation.

Example 1

```
@media screen and (device-aspect-ratio: 16/9), screen and
(device-aspect-ratio: 16/10) {
     .container div {width:100%}
}
```

Q: What media type is used in example 1?

Example 2: Color

@media all and (min-color-index:256) { ... }

Q: What does the 256 refer to in example 2?

Hint: complicated answer found at https://www.w3.org/TR/mediaqueries-4/

Example 3: Orientation

Q: What does the following media query do?

```
@media all and (orientation: portrait) { ... }
```

Q: What does the following media query do?

```
@media all and (orientation: landscape) { ... }
```

Exercise 1

Step 1

Create a file called media_query_example.html and add the following code to it. Notice how the display changes which the class .main is changed from *block* to *inline-block*.

```
<!DOCTYPE html>
<html>
<head>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>MMP 350 Media Query Example</title>
</head>
<style>
      .main div{display:inline-block;}
</style>
<body>
<section class="main container-fluid">
  <div class="col-xs-12 col-md-4 col-sm-6 alexandra">Project 1</div>
  <div class="col-xs-12 col-md-4 col-sm-6 alexandra">Project 2</div>
  <div class="col-xs-12 col-md-4 col-sm-6 alexandra">Project 3</div>
  <div class="col-xs-12 col-md-4 col-sm-6 alexandra">Project 4</div>
  <div class="col-xs-12 col-md-4 col-sm-6 alexandra">Project 5</div>
  <div class="col-xs-12 col-md-4 col-sm-6 alexandra">Project 6</div>
  <div class="col-xs-12 col-md-4 col-sm-6 alexandra">Project 7</div>
  <div class="col-xs-12 col-md-4 col-sm-6 alexandra">Project 8</div>
  <div class="col-xs-12 col-md-4 col-sm-6 alexandra">Project 9</div>
  <div class="col-xs-12 col-md-4 col-sm-6 alexandra">Project 10</div>
  <div class="col-xs-12 col-md-4 col-sm-6 alexandra">Project 11</div>
  <div class="col-xs-12 col-md-4 col-sm-6 alexandra">Project 12</div>
</section>
</body>
</html>
```

View the results in a browser

Step 2

Add the following code to your <style> section:

```
<style>
.main div{display:inline-block;}

@media screen and (min-width: 1000px)  {
    .main div {
      border: 1px solid orange;
    }
}
@media screen and (min-width: 600px)  {
    .main div {
      border: 1px solid red;
    }
}
@media screen and (min-width: 300px)  {
    .main div {
```

```
      border: 1px solid green;
    }
}
</style>
```

Q: Notice how the divs always have a green border, even when you resize your screen. Why?

A: Every screen width of 300px and greater meets the condition for the last media query, so the styling for the last one is the one that displays. Inspect the webpage to see this.

Step 3

Replace the contents of your style section with the following and then save and refresh your page.

```
<style>
.main div{display:inline-block;}
@media screen and (min-width: 1000px)  {
    .main div {
      border: 1px solid orange;
    }
}
@media screen and (min-width: 600px) and (max-width:999px) {
    .main div {
      border: 1px solid red;
    }
}
@media screen and (min-width: 400px) and (max-width:599px) {
    .main div {
      border: 1px solid green;
    }
}
</style>
```

Q: Resize your page. Now the borders of the divs change color. What has happened?
A: Setting min-width and max-width specifically constrains the styles, so the expected ones appear.

Inspect one of your columns. Notice that only one media query has been called.

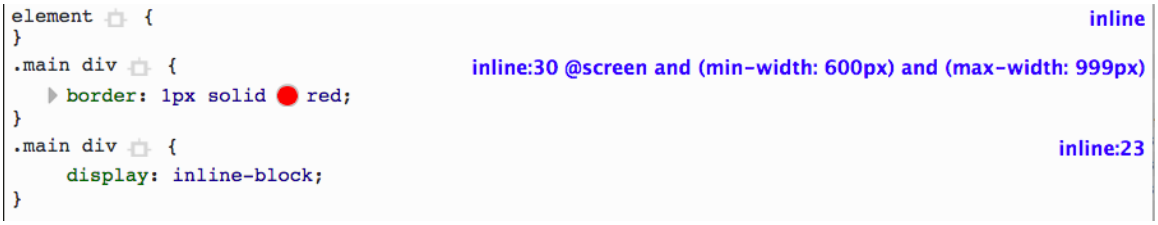

Figure: the styling in step three only activates one media query because it is very specific.

Step 4: Relying on document flow

Now replace the style section with the following code.

```
<style>
.main div{display:inline-block;}
@media screen and (min-width: 400px)  {
    .main div {
      border: 1px solid green;
    }
}
@media screen and (min-width: 600px)  {
    .main div {
      border: 1px solid red;
    }
}
@media screen and (min-width: 1000px)  {
    .main div {
      border: 1px solid orange;
    }
}
</style>
```

Q: Notice that the borders change color correctly. Why?
A: Because the min-width directives are organized from smallest to largest.

Q: How would you use max-width to implement the above logic?
A: Reverse the order of media queries with max-width

Inspect the columns to see how the browser interprets the <style>.

Grid Review

For details, please review the readings:
https://960.gs/
http://maxdesign.com.au/articles/liquid/

Grid Metrics

Traditional page width is 960 pixels divided into either 16 columns 40 pixels wide with 10 pixel gutters, or 12 columns 60 pixels wide with 10 pixel gutters. Gutter is

a design term for the space between columns, and are created using margin-left and margin-right settings. Another common format is 24 columns 30 pixels wide, with 10 pixel gutter and 5 pixel margins for containers

Note that these conventions are guidelines. Many webpages use 1,000 pixels as the default screen size. Liquid layouts replace specific pixel settings with percentages.

Please click on the following link for an excellent example of 12 and 16 columns and how they work:

https://960.gs/demo.html

Fixed, Liquid and Mixed Layouts

Please click on the following link, which uses a 1000 pixel grid.
http://maxdesign.com.au/articles/liquid/

Exercise 2: Using Bootstrap to Understand Both Responsive Design and Columns

Now lets use the bootstrap framework to do something similar.

Step 1

Create a document called `bootstrap_example.html` and add the following code to it:

```
<!DOCTYPE html>
<html>
<head>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>MMP 350 Bootstrap Example</title>
</head>
<style>
  .main div{display:inline-block;}
</style>
<body>
<header class="container">
  <h2>MMP 350 Week 4 Grid Example</h2>
  <p></p>
  <span class="dropdown">
    <button class="btn btn-default dropdown-toggle" type="button" data-
toggle="dropdown">Students
    <span class="caret"></span></button>
    <ul id="ul-coursework" class="dropdown-menu">
      <li><a href="#">Student One</a></li>
      <li><a href="#">Student Two</a></li>
      <li><a href="#">Student Three</a></li>
      <li><a href="#">Student Four</a></li>
      <li><a href="#">Student Five</a></li>
      <li><a href="#">Student Six</a></li>
      <li><a href="#">Student Seven</a></li>
      <li><a href="#">Student Eight</a></li>
```

```
      <li><a href="#">Student Nine</a></li>
</ul>
  </span>
  <span class="dropdown">
    <button class="btn btn-default dropdown-toggle" data-icon="carat-r" data-
role="button" type="button" data-toggle="dropdown">Projects
    <span class="caret"></span></button>
    <ul id="ul-projects" class="dropdown-menu">
      <li><a href="#">Menu Item </a></li>
      <li><a href="#">Menu Item </a></li>
      <li><a href="#">Menu Item </a></li>
    </ul>
  </span>
</header>
<section class="main container-fluid">
  <div class="col-sm-6">Project 1</div>
  <div class="col-sm-6">Project 2</div>
  <div class="col-sm-6">Project 3</div>
  <div class="col-sm-6">Project 4</div>
  <div class="col-sm-6">Project 5</div>
  <div class="col-sm-6">Project 6</div>
  <div class="col-sm-6">Project 7</div>
  <div class="col-sm-6">Project 8</div>
  <div class="col-sm-6">Project 9</div>
</section>

</body>
</html>
```

Resize the results in your browser. What you see if normal html document flow.

Step 2: Add references to bootstrap

Add references to bootstrap and jQuery to the head section of your document:

```
<head>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>MMP 350 Bootstrap Example</title>
  <meta charset="utf-8">
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
  <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
</head>
```

Q:When you save, refresh and resize your screen, what happens?
A: The lists have turned into drop down menus and the divs have adopted the bootstrap responsive design style for the sm class.

Exercise 3: The Bootstrap Grid in Detail

Step 1

Create a document called `bootstrap_grid.html` with the following code from the bootstrap examples. Save and view the document at different browser widths. Pay attention to the class settings and how they influence layout.

```
<!DOCTYPE html>
<html>
<head>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>MMP 350 Bootstrap Example</title>
  <meta charset="utf-8">
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
  <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
</head>
</head>

<style>
.main div{display:inline-block;}
.main div{border:1px solid red;}
.emphasis{color:red;font-weight:400;}
</style>
<body>
   <div class="container">
      <div class="page-header">
        <h1>Bootstrap grid examples</h1>
        <p class="lead">Basic grid layouts to get you familiar with building within the
Bootstrap grid system.</p>
      </div>
      <h3>Three equal columns</h3>
      <p>Get three equal-width columns <strong>starting at desktops and scaling to large
desktops</strong>. On mobile devices, tablets and below, the columns will automatically
stack.</p>
      <div class="row">
        <div class="col-md-4">.col-md-4</div>
        <div class="col-md-4">.col-md-4</div>
        <div class="col-md-4">.col-md-4</div>
      </div>
      <h3>Three unequal columns</h3>
      <p>Get three columns <strong>starting at desktops and scaling to large
desktops</strong> of various widths. Remember, grid columns should add up to twelve for a
single horizontal block. More than that, and columns start stacking no matter the
viewport.</p>
      <div class="row">
        <div class="col-md-3">.col-md-3</div>
        <div class="col-md-6">.col-md-6</div>
        <div class="col-md-3">.col-md-3</div>
      </div>
      <h3>Two columns</h3>
      <p>Get two columns <strong>starting at desktops and scaling to large
desktops</strong>.</p>
      <div class="row">
        <div class="col-md-8">.col-md-8</div>
        <div class="col-md-4">.col-md-4</div>
      </div>
      <h3>Full width, single column</h3>
      <p class="text-warning">No grid classes are necessary for full-width elements.</p>
      <hr>
      <h3>Two columns with two nested columns</h3>
      <p>Nesting is easy—just put a row of columns within an existing column. This gives
you two columns <strong>starting at desktops and scaling to large desktops</strong>, with
another two (equal widths) within the larger column.</p>
      <p>At mobile device sizes, tablets and down, these columns and their nested columns
will stack.</p>
      <div class="row">
        <div class="col-md-8">
          .col-md-8
          <div class="row">
            <div class="col-md-6">.col-md-6</div>
            <div class="col-md-6">.col-md-6</div>
          </div>
        </div>
```

```
        <div class="col-md-4">.col-md-4</div>
      </div>
      <hr>
      <h3>Mixed: mobile and desktop</h3>
      <p>The Bootstrap 3 grid system has four tiers of classes: xs (phones), sm
(tablets), md (desktops), and lg (larger desktops). You can use nearly any combination of
these classes to create more dynamic and flexible layouts.</p>
      <p class="emphasis">Each tier of classes scales up. If you plan on setting the same
widths for <i>xs</i> and <i>sm</i>, you only need to specify <i>xs</i>.</p>
      <div class="row">
        <div class="col-xs-12 col-md-8">.col-xs-12 .col-md-8</div>
        <div class="col-xs-6 col-md-4">.col-xs-6 .col-md-4</div>
      </div>
      <div class="row">
        <div class="col-xs-6 col-md-4">.col-xs-6 .col-md-4</div>
        <div class="col-xs-6 col-md-4">.col-xs-6 .col-md-4</div>
        <div class="col-xs-6 col-md-4">.col-xs-6 .col-md-4</div>
      </div>
      <div class="row">
        <div class="col-xs-6">.col-xs-6</div>
        <div class="col-xs-6">.col-xs-6</div>
      </div>
      <hr>
      <h3>Mixed: mobile, tablet, and desktop</h3>
      <p></p>
      <div class="row">
        <div class="col-xs-12 col-sm-6 col-lg-8">.col-xs-12 .col-sm-6 .col-lg-8</div>
        <div class="col-xs-6 col-lg-4">.col-xs-6 .col-lg-4</div>
      </div>
      <div class="row">
        <div class="col-xs-6 col-sm-4">.col-xs-6 .col-sm-4</div>
        <div class="col-xs-6 col-sm-4">.col-xs-6 .col-sm-4</div>
        <div class="col-xs-6 col-sm-4">.col-xs-6 .col-sm-4</div>
      </div>
      <hr>
      <h3>Column clearing</h3>
      <p><a href="http://getbootstrap.com/css/#grid-responsive-resets">Clear floats</a>
at specific breakpoints to prevent awkward wrapping with uneven content.</p>
      <div class="row">
        <div class="col-xs-6 col-sm-3">
          .col-xs-6 .col-sm-3
          <br>
          Resize your viewport or check it out on your phone for an example.
        </div>
        <div class="col-xs-6 col-sm-3">.col-xs-6 .col-sm-3</div>
        <!-- Add the extra clearfix for only the required viewport -->
        <div class="clearfix visible-xs"></div>
        <div class="col-xs-6 col-sm-3">.col-xs-6 .col-sm-3</div>
        <div class="col-xs-6 col-sm-3">.col-xs-6 .col-sm-3</div>
      </div>
      <hr>
      <h3>Offset, push, and pull resets</h3>
      <p>Reset offsets, pushes, and pulls at specific breakpoints.</p>
      <div class="row">
        <div class="col-sm-5 col-md-6">.col-sm-5 .col-md-6</div>
        <div class="col-sm-5 col-sm-offset-2 col-md-6 col-md-offset-0">.col-sm-5 .col-sm-
offset-2 .col-md-6 .col-md-offset-0</div>
      </div>
      <div class="row">
        <div class="col-sm-6 col-md-5 col-lg-6">.col-sm-6 .col-md-5 .col-lg-6</div>
        <div class="col-sm-6 col-md-5 col-md-offset-2 col-lg-6 col-lg-offset-0">.col-sm-6
.col-md-5 .col-md-offset-2 .col-lg-6 .col-lg-offset-0</div>
      </div>
    </div> <!-- /container -->
</body>
</html>
```

Discussion

Agile Versus Waterfall Development

A discussion of last week's readings. This discussion will only happen if there is time.

Review of SDLC, based on homework reading

https://www.tutorialspoint.com/sdlc/sdlc_quick_guide.htm

Review of Twelve Principles of Agile Development

http://agilemanifesto.org/principles.html

The Differences Between SDLC and Agile

https://dzone.com/articles/why-do-we-capture-requirements-differently-between

Homework

Read the following articles
https://developer.wordpress.org/themes/basics/template-hierarchy/
https://en.support.wordpress.com/start/
https://en.support.wordpress.com/five-step-blog-setup/

COMPLETE ASSIGNMENT ONE

Links relevant to today's class

Portfolios
https://www.format.com/magazine/resources/illustration/illustration-portfolio-examples-2016

Requirement Gathering
https://dzone.com/articles/why-do-we-capture-requirements-differently-between

Agile Manifesto
http://agilemanifesto.org/principles.html

Liquid Layouts the Easy Way
http://maxdesign.com.au/articles/liquid/

Error Handling Media Queries
https://www.w3.org/TR/mediaqueries-4/#error-handling

Grids – Very Important New Browser Feature

https://hacks.mozilla.org/2016/12/css-grid-and-grid-highlighter-now-in-firefox-developer-edition/

Key Concepts

960 pixel grid system has two main implementations, based on a 12 or 16 columns. The 12 column system has 60 pixel columns, the 16 column system has 40 pixel columns. Both use 10 pixel margins, to create 20 pixel "gutters" between columns.

Media queries are used to create styles specific to particular media, for example, *screen*, *printer*, *monochrome*.

The ordering of media queries in a style sheet can influence the rendering of your webpages. For example, the directives min-width:768px min-width:720px and min-width:480px can all apply to a "phablet" phone like and iPhone 7plus or a Galaxy Note 8, so you have to ensure that media queries are either specific (for example by setting both min-width and max-width) or properly ordered in your css files.

SDLC: Software Development Lifecyle. Steps are: Planning, Defining, Designing, Building, Testing, Deployment.

Agile Development: Reaction to the "top down" or "waterfall" approach to SDLC. More general than SDLC; less hierarchical; more flexible because it off-loads implementation details to developers and designers, and because it assumes that requirements will be changing continually.