HTML5 Authoring with Mark Lassoff

Section 15: Page Layout

After completing this section, you will:

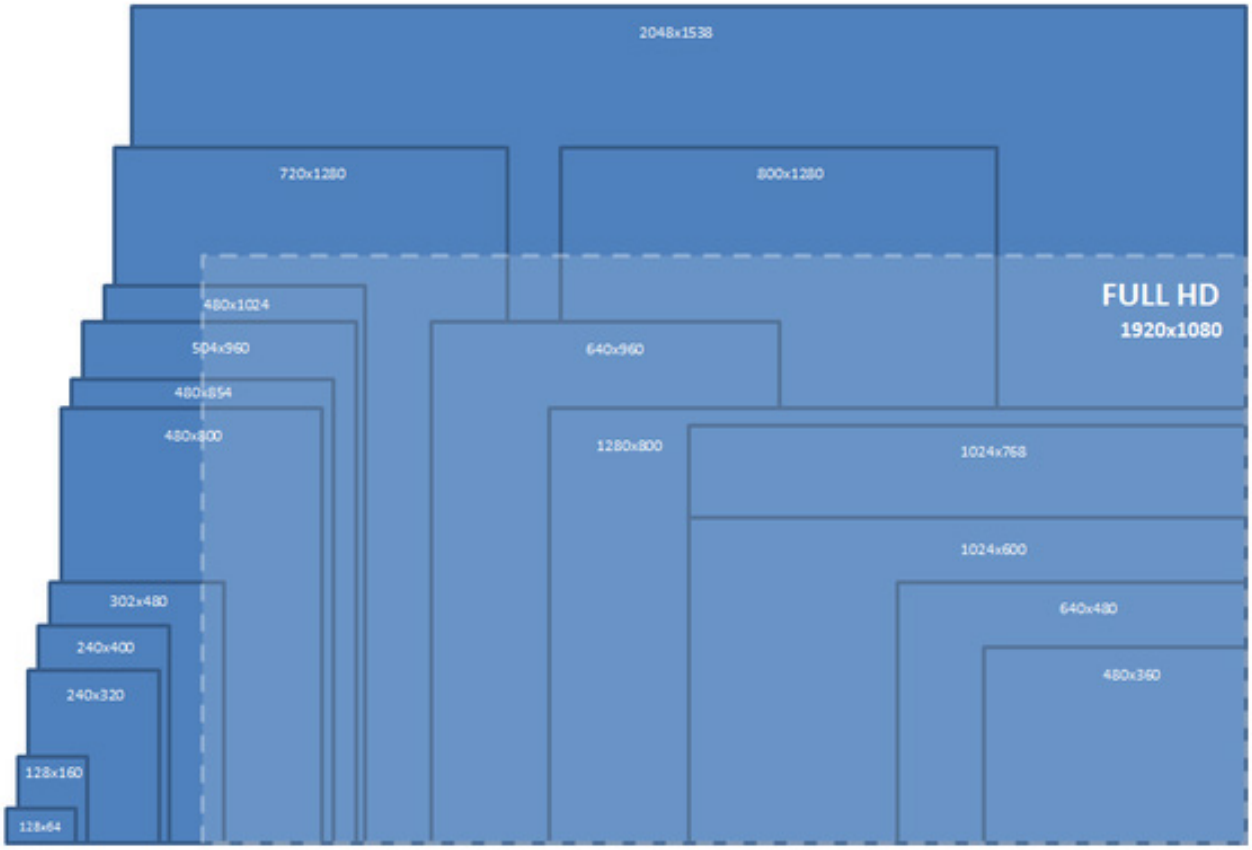
- ☐ Identify the purpose of normalize.css documents
- ☐ Be able to use a container and center your page
- ☐ Create Headers and Footers
- ☐ Use Float to Create Columns
- ☐ Use Media Queries to create a more responsive design

Introduction

Page layout with CSS is complicated by the fact that the destination screen size and resolution are most often unknown. One viewer could view your content on a large screen monitor, while others might be using a mobile device screen.

Despite the diversity of screen sizes, it's essential that digital content remains both readable and attractive regardless of the environment in which it's viewed. The image below illustrates the diversity of screen sizes and the challenge that digital designers often face.

In this section, we're going to look more closely at a couple of common page layout techniques.



Normalize.css

We'll begin with the HTML5 basic document structure with one addition. Key in the following code while making special note of the URL:

```
<!DOCTYPE html>
<html>
<head>
<link href="https://cdnjs.cloudflare.com/ajax/libs/
normalize/8.0.1/
normalize.min.css" rel="stylesheet" />
<title>Page Layout</title>
</head>
<body>
</body>
</html>
```

You, of course, noticed the `link` tag added to the `head` section of the document. The CSS style sheet we're linking to is known as a "reset" or "normalize" style sheet. These style sheets are designed to ameliorate the small difference in the default style sheet in various browser. Reset style sheets are also designed to remove some of the unpredictabliity from design as most unpredictable margin and padding values are set to zero.

The Page Container

When doing more complex page layout, it's a good idea to use a master container `div` to layout your page. This is a box that surrounds all of your other content. The page container div allows you to attach styles to all content on the page without using the `body` tag as a CSS selector.

The page container makes a better parent for elements on the page than the `body` tag. We'll implement the page container inside the body tag.

```
<!DOCTYPE html>
<html>
<head>
<link href="https://cdnjs.cloudflare.com/ajax/libs/
normalize/8.0.1/
 normalize.min.css" rel="stylesheet" />
 <title>Page Layout</title>
 </head>
 <body>
      <div id="container">
      </div> <!-- container -->
</body>
```

Now that we have our container div we'll build everything inside this element.

Centering the Container

I want to set up this layout so that the entire design is centered in the browser window. We'll have the layout occupy 90% of the width of the window. We'll also color the background a very light gray, add some padding and set the default typography.

Since the container is our root element, it makes sense to do this here. Centering the content is accomplished by setting the left and right margin to `auto`. (We set the top and bottom margins to 0 in this document.)

Add the following code to the `head` section of your document.

```
<style>
        #container
        {
           width: 90%;
           background-color: #eee;
           margin: 0 auto;
           padding-left: 3px;
           padding-right: 3px;
           font-family: Verdana;
           font-size: 1.1em;
        }
  </style>
```

CSS Set Up

We're going to divide our document into several divisions. The body of the document contains the sections and dummy text.

```
<body>
     <div id="container">
```

```
<header><h1>Connecticut News</h1></header>
<div id="colA">
<h2>Connecticut Cold in the Winter</h2>
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec auctor risus lorem. Nulla consequat ultrices arcu a elementum. Mauris mollis tincidunt nibh, vel tincidunt enim luctus nec. Donec efficitur mollis odio et laoreet. Aenean ac justo efficitur, euismod nisi non, sagittis enim. Integer bibendum pretium purus sed aliquam. Proin pretium aliquet mi. Curabitur eu sapien massa. Fusce vel leo quis dui lobortis ultrices non nec ipsum. Donec et ullamcorper nisl. Ut consequat sapien et dolor accumsan, eu volutpat neque vulputate. Nam id tempus velit. Etiam at tortor eget dolor viverra molestie. Donec porta neque eros, et vehicula justo lacinia quis.</p>

  <p>Nam lacinia feugiat pulvinar. Aliquam erat volutpat. Sed cursus volutpat justo, ac convallis nunc luctus vehicula. Proin ultrices imperdiet bibendum. Curabitur vitae pharetra leo, ut fermentum arcu. Sed ullamcorper lorem et pulvinar interdum. Proin volutpat imperdiet augue nec iaculis. Interdum et malesuada fames ac ante ipsum primis in faucibus. Quisque tempus consectetur commodo.</p>

</div>
<div id="colB">
<h2>Connecticut Sun:  Pro Basketball at Mohegan Sun</h2>
  <p>In tincidunt tellus metus, vel condimentum diam rhoncus ut. Nullam eu turpis ligula. Nulla vehicula nisl augue, posuere commodo nulla feugiat eget. Sed mattis ultricies purus eu auctor. Donec aliquam ante leo, sed ultrices libero egestas at. Donec quis magna aliquam, auctor turpis sed, tincidunt orci. Mauris suscipit eros nec nunc vulputate rhoncus at dictum ipsum. Aenean pretium nulla eget gravida convallis. Donec non dolor a diam egestas commodo. Sed et lectus sit amet sapien iaculis consectetur. Phasellus imperdiet purus sit amet suscipit placerat. Integer ac lorem eu lectus eleifend iaculis eget sit amet ante. Maecenas sed ex ipsum.</p>

  <p>Nulla sed mauris auctor, sagittis ipsum quis, aliquam mauris. Pellentesque lobortis faucibus hendrerit. Sed
```

```
ullamcorper vestibulum neque nec molestie. Suspendisse
potenti. Suspendisse gravida vehicula eros. Nam consequat nec
massa et tincidunt. Morbi euismod mollis iaculis. Donec nec est
ex. Aenean sit amet ante vestibulum, ultrices tellus vel, rutrum
nibh. Orci varius natoque penatibus et magnis dis parturient
montes, nascetur ridiculus mus. Suspendisse est quam, ultrices
vel rhoncus condimentum, convallis vel libero.</p>
      </div>
      <div id="colC">
      <h2>Mysteries of Connecticut</h2>
        <p>Donec eget euismod orci, a ultrices tortor. Ut et
pretium arcu. Ut ut est id elit dapibus aliquam ut eu massa.
Maecenas varius ligula a nunc tempor ullamcorper. Sed luctus
finibus nulla, vestibulum commodo sem. Nulla sed ligula varius,
ullamcorper magna non, hendrerit nulla. In hac habitasse platea
dictumst. Sed ut nulla vel metus fringilla iaculis ut eget mauris.
Curabitur sit amet urna euismod, rutrum velit vitae, porttitor
massa. Nam facilisis nisi vel nisl vestibulum, eget lobortis massa
tincidunt. Etiam cursus sem sit amet tortor posuere, eu ultricies
lacus venenatis. Vestibulum interdum sollicitudin porttitor. Fusce
nec sollicitudin orci.</p>
      </div>
      <footer>Copyright &copy;2018 Framework Television</
footer>
      </div> <!-- container -->
  </body>
```

If you want to use dummy text for this exercise (or for anything else) you can copy it from **www.lipsum.com**.

You'll notice that we use the id's `colA`, `colB`, and `colC` to label our individual columns. We also have header and footer tags that contain (obviously) the page header and footer content.

We'll create entries in our `<style>` element to hold the CSS we'll need to create for each of the content areas.

```
<style>
        #container
        {
           width: 90%;
           background-color: #eee;
           margin: 0 auto;
           padding-left: 3px;
           padding-right: 3px;
           font-family: Verdana;
           font-size: 1.1em;
        }

        header
        {

        }

        footer
        {

        }

        #colA
        {

        }

        #colB
        {

        }

        #colC
        {

        }
 </style>
```

Even though we haven't placed the style rules in the document yet, we can see the structure starting to take shape. Creating placeholders for style rules is especially helpful when you have complex documents with many divisions and content areas.

> Note that the `#container` rules come first in the style sheet. This is by design. Any individual rules in the content areas that differ from the container override the original container rules.

Header and Footer

Most websites have a header and footer content. (Sometimes header and footer content can be extensive)

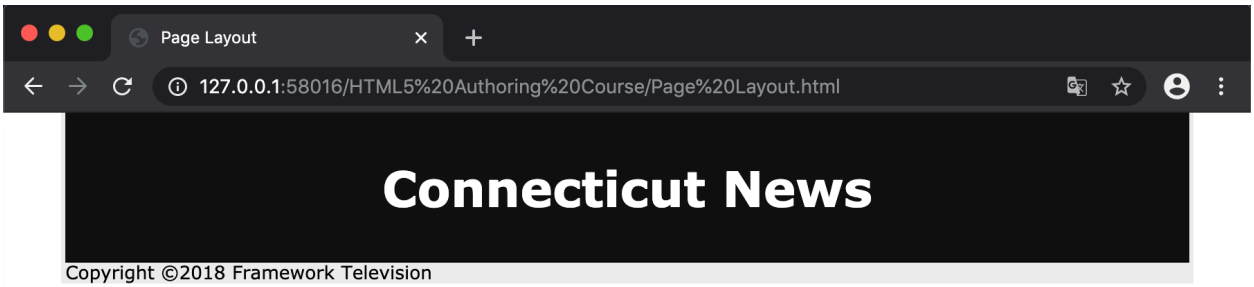
For the `header` element, let's center the content, reverse the background and text colors and add some padding.

```
header
        {
           text-align: center;
           background-color: #121212;
           color: white;
           padding-top: 10px;
           padding-bottom: 10px;
        }
```

For the `footer` element, let's add a line above it and reduce the size of the type.

```
footer
        {
           border-top: 1px solid black;
           font-size: .75em;
        }
```

With the few styles listed above, we can already get a sense of what our header and footer will look like in the browser. (The line above the footer isn't visible because it's right up against the header section.)



Column Floats

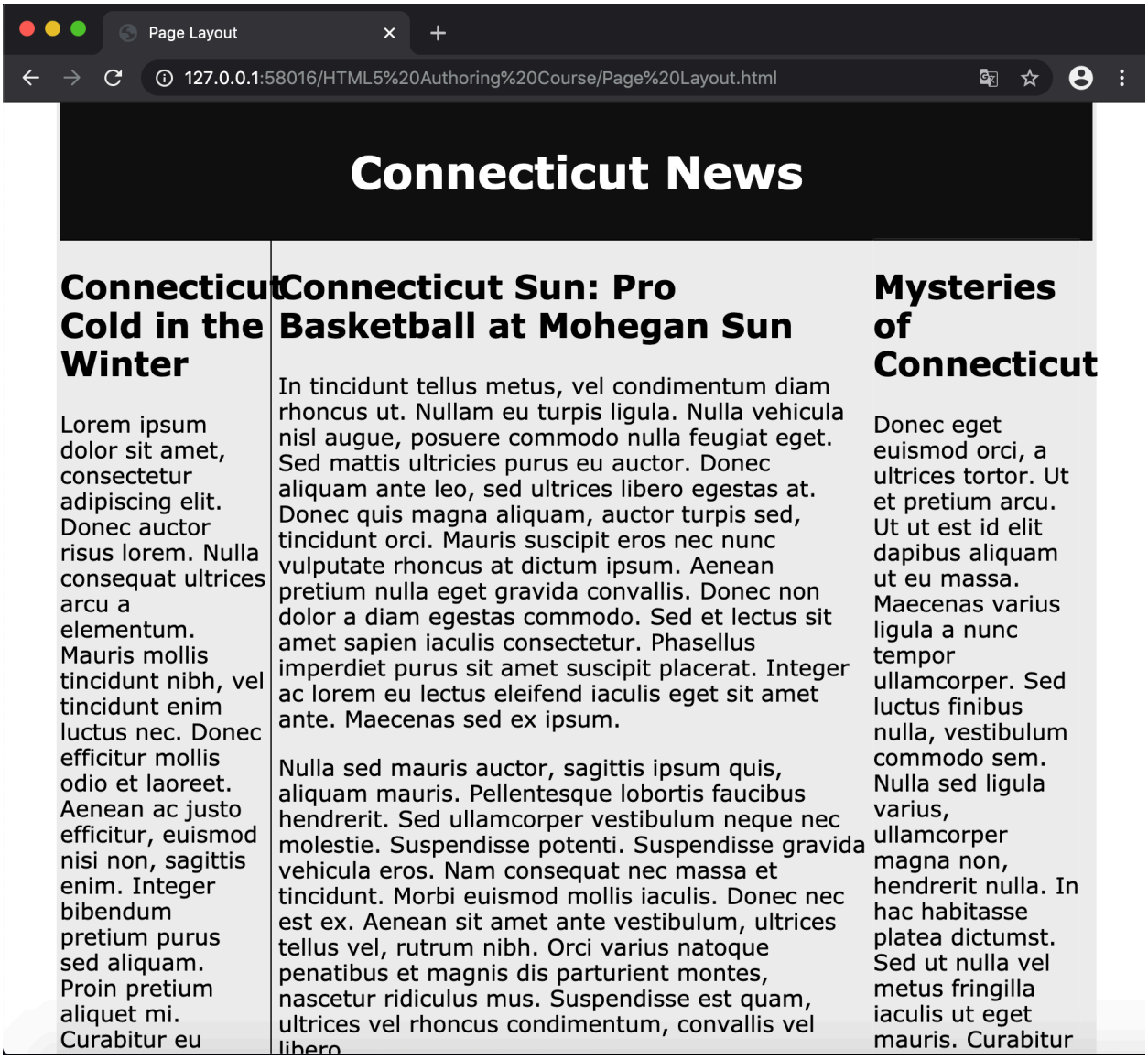
The design we're working on is a three-column layout. We'll accomplish by setting appropriate column widths, float rules, and borders. We'll also add some padding so that the text content from different columns don't crowd and become hard to read.

Let's add the following rules to our `style` element:

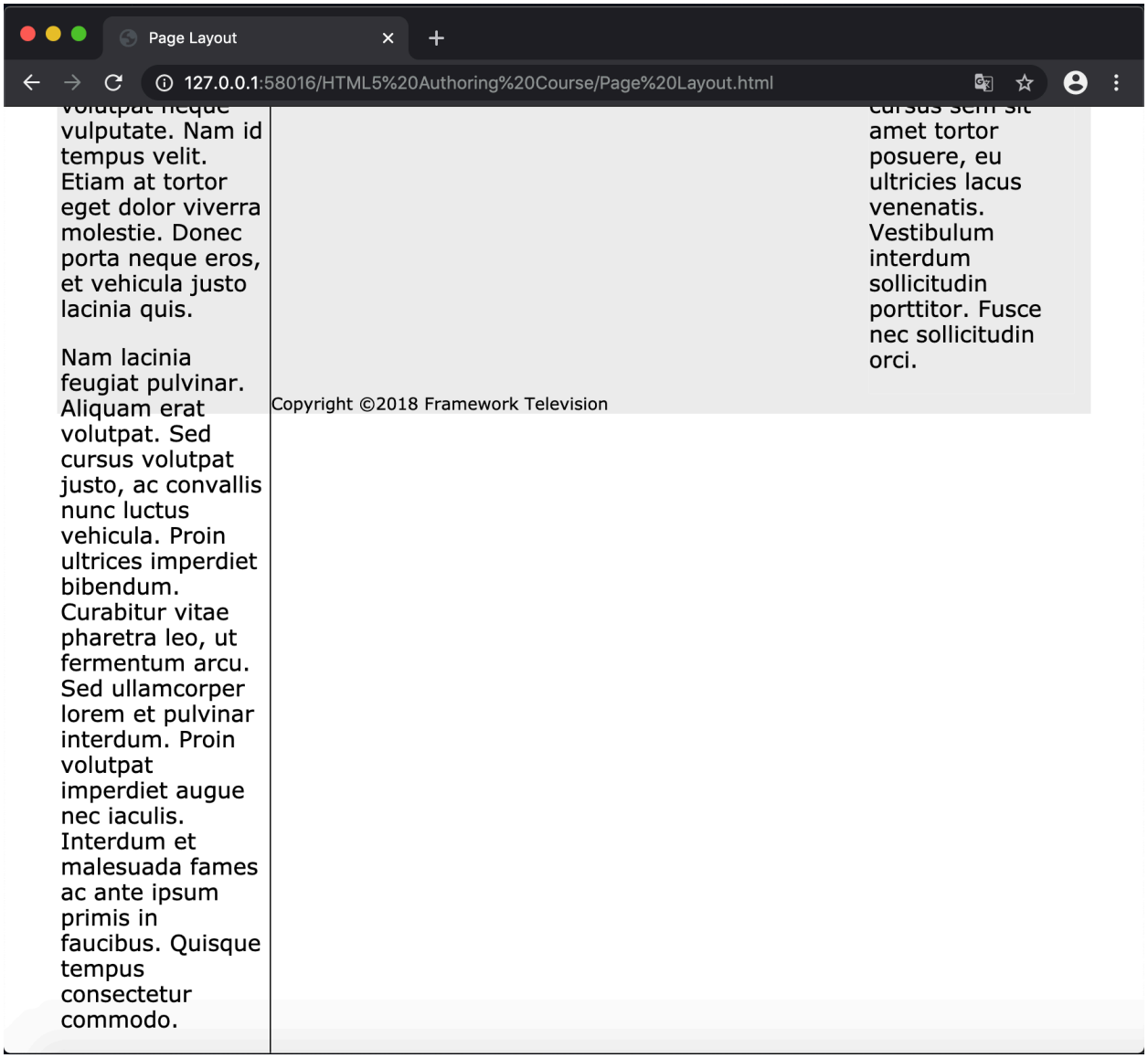
```
#colA
        {
           width: 20%;
           float: left;
           border-right: 1px solid black;
           padding-right: 3px;
        }

        #colB
        {
           width: 57%;
           float: left;
           padding-left: 5px;
           padding-right: 5px;
        }
```

```
#colC
{
   width: 20%;
   float: left;
}
```

If you test in the browser, you should see your content laid out in three columns.



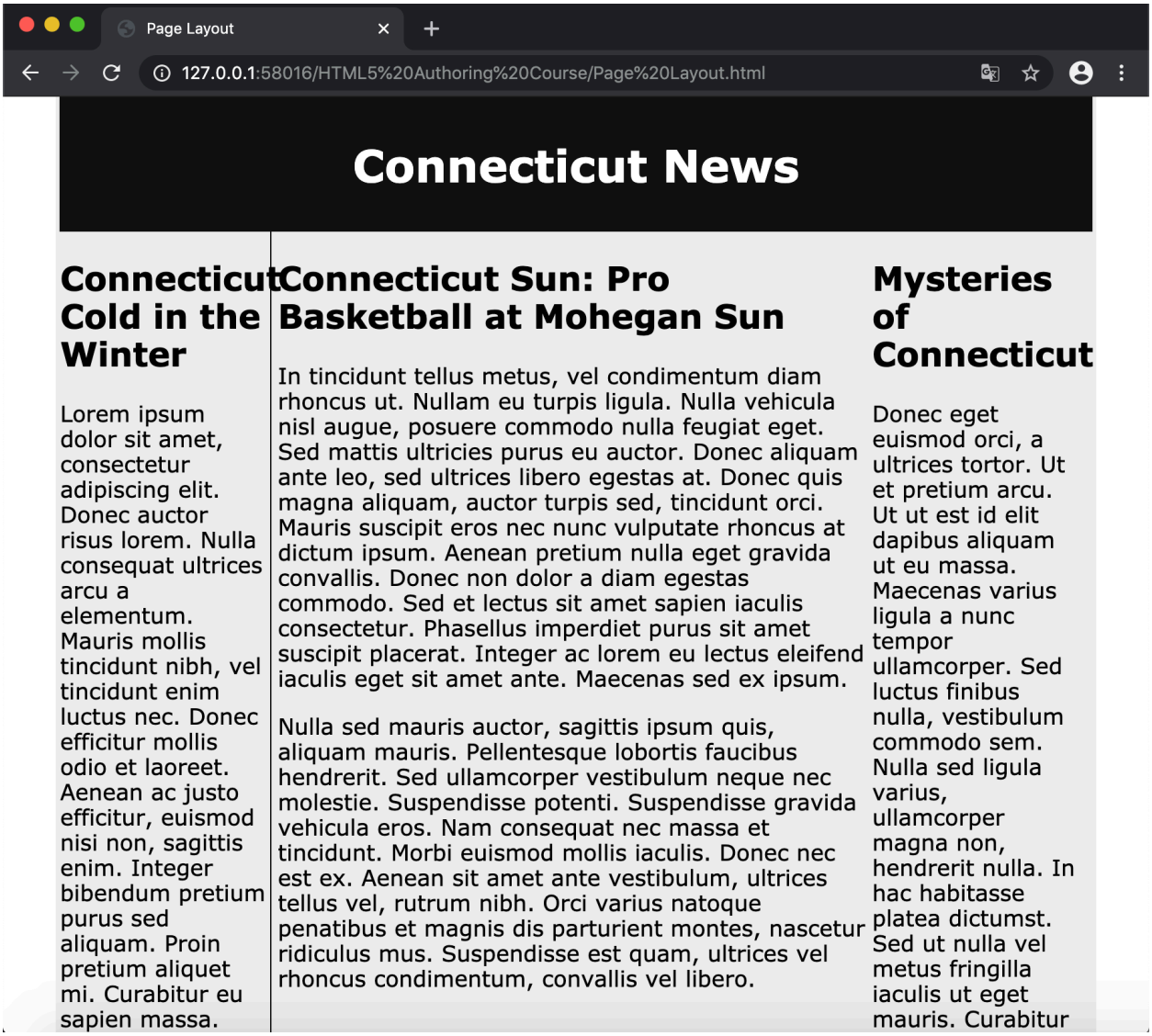
However, if you scroll down, you may notice a problem with the footer.



When you float an element in the browser you “break” the natural stacking of elements. The result is that any elements below the floated elements are laid out somewhat unpredictably. We can fix this by clearing the floats in the footer element:

```
footer
        {
           border-top: 1px solid black;
           font-size: .75em;
           clear: both;
        }
```

Once you add the `clear` rule to the footer, the problem will rectify itself. Now, our layout looks pretty good on a laptop screen. However, we have some problems on smaller screens. If the screen isn't wide enough to accommodate the three columns, and content, things get crowded quickly.



When the screen isn't wide enough to accommodate the content, the result is unacceptable. Generally, we'll undo the floats and stack our layers on smaller screens to make our content readable.

We accomplish this with media queries.

Media Queries

Media queries allow us to create breakpoints at which the design changes based on specific rules, usually on document width. Let's add the following media query to the bottom of our `style` element.

```
@media only screen and (max-width: 600px)
        {
          #colA, #colB, #colC
          {
            float: none;
            width: 100%;
          }

          #colA
          {
            border-right: none;
          }
        }
```

This media query states that the included rules will apply up to a maximum width of 600px. The float is removed from the three columns, and the border on the right side of `#colA` is removed as well. Any browser that is wider than 600px will not apply these rules.

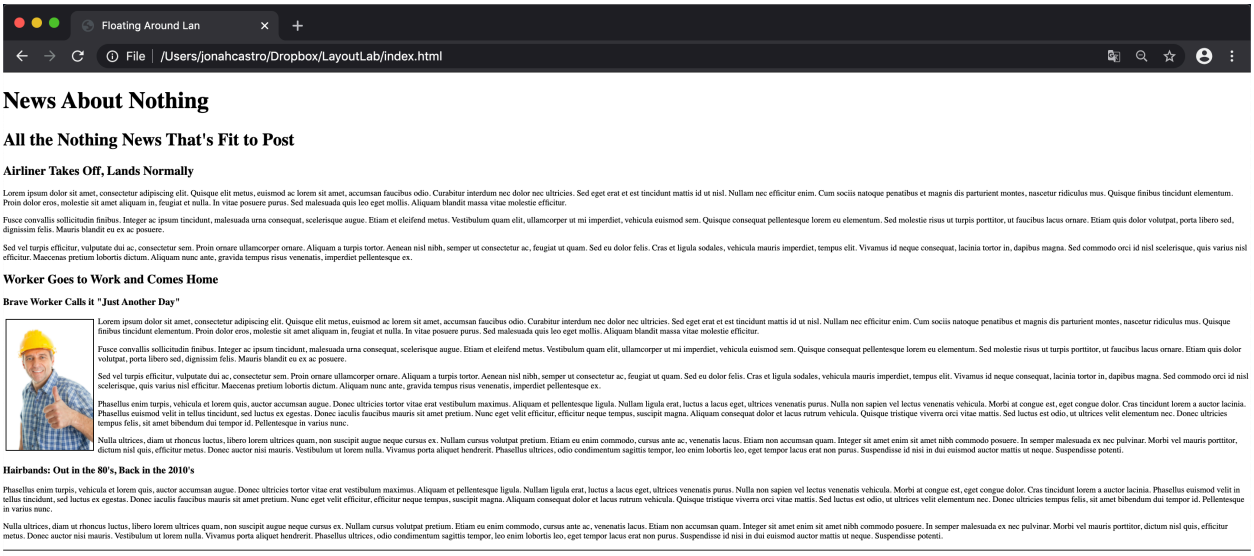
Many developers create different media queries for phones, tablets, and large screens. Others even target specific models of phones and tablets with their media queries. Going overboard is easy! In general, you want your media queries to ensure your content is accessible across a wide array of screen sizes.

Do This

Using document we've created redo the media Queries based on typical iPhone and iPad width. You can find information about Apple display sizes at https://developer.apple.com/library/archive/documentation/DeviceInformation/Reference/iOSDeviceCompatibility/Displays/Displays.html. Please note that it's easy to overdo it with media queries. Just add one or two queries to address a few Apple screen sizes.

Submit This

Download the lab starter at https://s3.amazonaws.com/coursewareframework/html5authoring/LayoutLab.zip. View the content in your browser. It should appear similar to the image below.



Create a traditional three column layout with header and footer using the HTML document provided. Using media queries, ensure that your document works across a variety of browser widths.

Remember, when submitting the work please use the following naming convention for your file: `HTMLAUTHORING_LastName_SectionName.html`. So if your last name is Smith and you are submitting this section. Your file name should be `HTMLAUTHORING_Smith_Layout.html`. Since you have two files for this exercise, please zip them together before uploading.

For this course go to https://www.dropbox.com/request/RhW9kBDXtisq2Fsvg3hY to submit your assignments.