GENERAL ASSEMBLY

HTML & CSS

DAY 2

Kim Goulbourne

AGENDA

- Review
- HTML5 Structural Elements
- Box Model
- Floating Elements with CSS
- Determining layout after looking at a design
- Lab Time

STYLING: BOX MODEL

REVIEW

STYLING: BOX MODEL

- inheritance vs specificity vs importance
- block vs inline
- relative vs absolute path
- nesting selectors in css (when does it make sense vs using classes or ids?)
  - it's always better to start by styling by tags first (especially when you will have a lot of repeated styles), when it comes to more custom elements or a deeply nested element (4 max deep), then you can start to add classes or ids

THINGS TO REMEMBER

STYLING: FLOATS

- Use lowercase letters for all your files (html, css, images)
  - Names should be alphanumeric and never contain spaces but can contain hyphens or underscores.
- Use lowercase letters for all your classes and ids.
- Include <meta> tags and the <title> tag in all your html files.
  - The title and description may be the only things that change.
- Practice indenting your code for better readability. It will help you properly nest your elements.
- Always place css in an external stylesheet unless otherwise noted.
- Remember to consider your “working directory” when writing file paths.

HTML STRUCTURAL ELEMENTS

HEADER

The <header> tag is used to group elements that make up the *header* of a webpage.

It is primarily used as a "container" tag similar to the <div> tag.

```
<header>
  <nav>Nav Links</nav>
</header>
```

* It was introduced in html 5 and is a more "semantic" tag to use than <div> tags.

NAV

The <nav> tag is used to group the elements that make up the *nav* of a webpage.

It is typically nested in the <header> tag and includes a list of links.

```
<nav>
 <ul>
   <li><a href="">…</a></li>
 </ul>
</nav>
```

* It was introduced in html 5 and is a more "semantic" tag to use than <div> tags.

ASIDE

The <aside> tag is used to group elements that make up the *sidebar* of a webpage.

It is primarily used as a “container” tag similar to the <div> tag.

```
<aside>
  <h2>Categories</h2>
</aside>
```

* It was introduced in html 5 and is a more “semantic” tag to use than <div> tags.

FOOTER

The <footer> tag is used to group elements that make up the *footer* of a webpage.

It is primarily used as a “container” tag similar to the <div> tag.

```
<footer>
  <p>Copyright</p>
</footer>
```

* It was introduced in html 5 and is a more “semantic” tag to use than <div> tags.

SECTION

The <section> tag is used to group elements that make up a “section” in a document.

It is primarily used as a “container” tag similar to the <div> tag.

```
<section>
  <h2>About Us</h2>
</section>
```

* It was introduced in html 5 and is a more “semantic” tag to use than <div> tags.

ARTICLE

The <article> tag is used to group elements that make up independent, self-contained content such as a Forum post, Blog post or News story.

It is primarily used as a "container" tag similar to the <div> tag.

```
<article>
  <h2>Blog Post Title</h2>
</article>
```

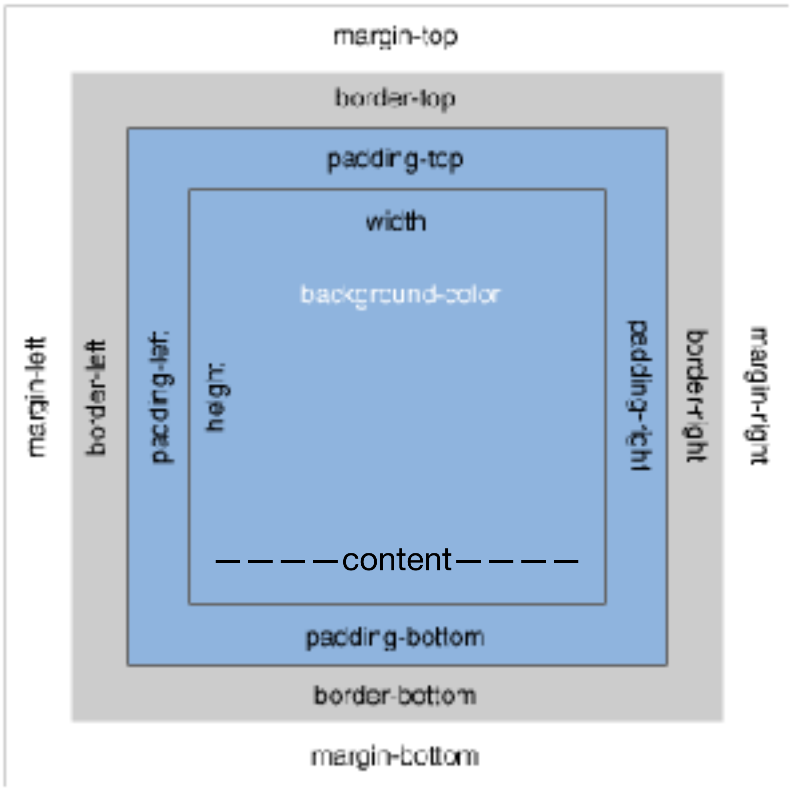
* It was introduced in html 5 and is a more "semantic" tag to use than <div> tags for articles/posts.

BOX MODEL

STYLING: BOX MODEL

All HTML elements can be considered as boxes.

The CSS box model is essentially a box that wraps around every HTML element. It consists of: margins, borders, padding, and the actual content.



STYLING: BOX MODEL

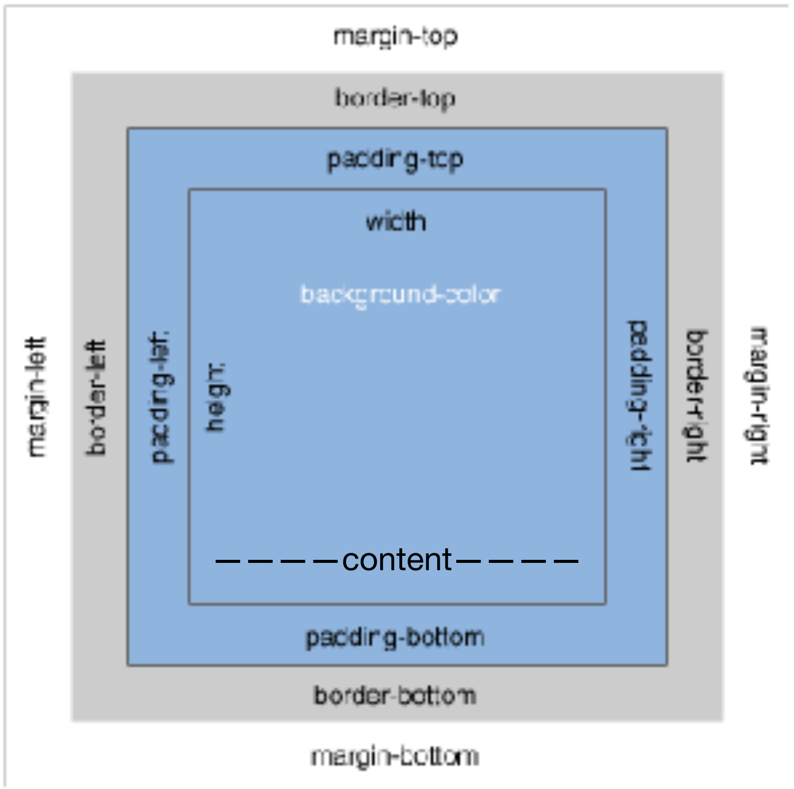
Content - The content of the box, where text and images appear.

Width & Height - Sets the size of the area of the content.

Padding - Clears an area around the content. The padding is transparent.

Border - The border goes around the padding and content.

Margin - Clears the area outside the border. The margin is transparent.



IMPORTANT NOTE ON WIDTH AND HEIGHT

When you set the width and height properties of an element with CSS, you just set the width and height of the content area. To calculate the full size of an element, you must also add padding, borders and margins.

* The padding, borders and margins contribute to the resulting width of the element once rendered in the browser.

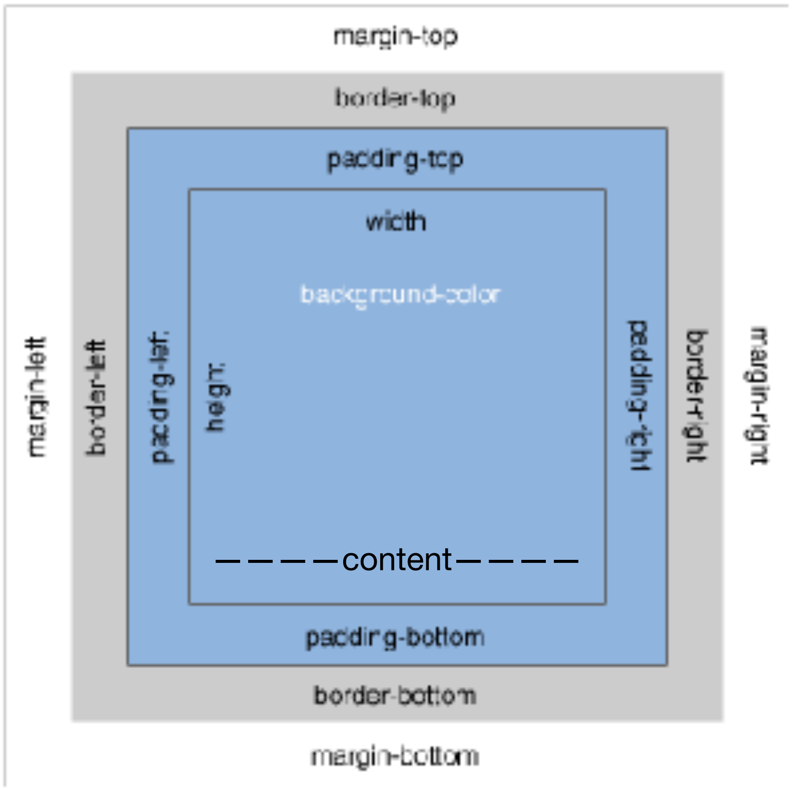
STYLING: BOX MODEL

RESULTING WIDTH

width + left padding + right padding + left border + right border + left margin + right margin

RESULTING HEIGHT

height + top padding + bottom padding + top border + bottom border + top margin + bottom margin



LET’S PRACTICE

```
div {
   width: 320px;
   padding: 10px;
   border: 5px solid gray;
   margin: 0;
}
```

What will be the total width of this div?

If I want a <div> with a class of “test” to have a total width of 560px with:

- Top and bottom padding of 10px and left and right padding of 25px.
- and a margin right of 10px.

How do I write this?

WHEN TO USE MARGIN AND PADDING

- Considering the box model
  - Padding is considered to be a apart of the element so if you imagine 2 elements are on top of each other with 10px padding, there will be no space between the elements.
  - Margin is considered to be outside the element so with the same 10px as margin, the resulting space will be 10px (not 20px), because the margins will overlap UNLESS they are beside each other using inline-block or floats which in that case the total space will add be 20px.
- Borders or backgrounds
  - Normally you will add padding to an element to give it space inside a border.
  - Using margin will add space on the outside of the border
- Click region
  - Padding is included in the click region, margin is not

BOX-SIZING (CSS PROPERTY)

Specify that elements should have padding and border included in the element's total width and height.

```
div {
   box-sizing: border-box;
}
```

ALTER THE BOX MODEL

```
div {
   box-sizing: content-box;
}
```

DEFAULT

STYLING: BOX MODEL

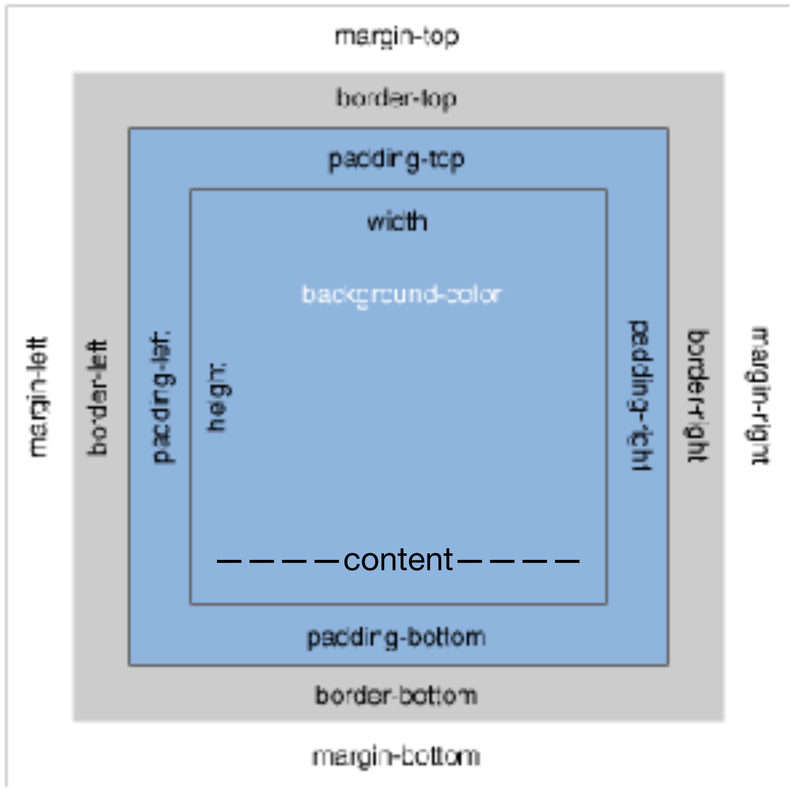
WITH BOX SIZING: BORDER-BOX

RESULTING WIDTH

width + left margin + right margin

RESULTING HEIGHT

height + top margin + bottom margin



LET’S PRACTICE

```
div {
   width: 320px;
   padding: 5px;
   margin: 10px;
  box-sizing: border-box;
}
```

What will be the total width of this div?

If I want a <div> with a class of “test” to have a total width of 420px with:

- Top and bottom padding of 10px.
- and a margin left and right of 20px.

How do I write this?

PRACTICE

STYLING THE BOX MODEL

STYLING: FLOATS

PSEUDO-ELEMENTS & PSEUDO-CLASSES

WHAT ARE PSEUDO-ELEMENTS?

Pseudo-elements allow you to style certain parts of a document. They are denoted by the double colon(CSS3).

A common example of these are the pseudo-elements used to insert content before or after an element.

- ::before (first child/node in the element)
- ::after (last child/node in the element)

```
div::before {
  content: “hello”;
}

// syntax for support in IE8
div:before  {
  content: “hello”;
}
```

WHAT ARE PSEUDO-CLASSES?

Pseudo-classes are used to define a special state of an element. They are denoted by a single colon.

A common example of these are the pseudo-classes used for links.

- :link (original state of a link)
- :visited (state of a link after being clicked on)
- :hover (state of a link on mouseover)
- :active (state of a link on mousedown)

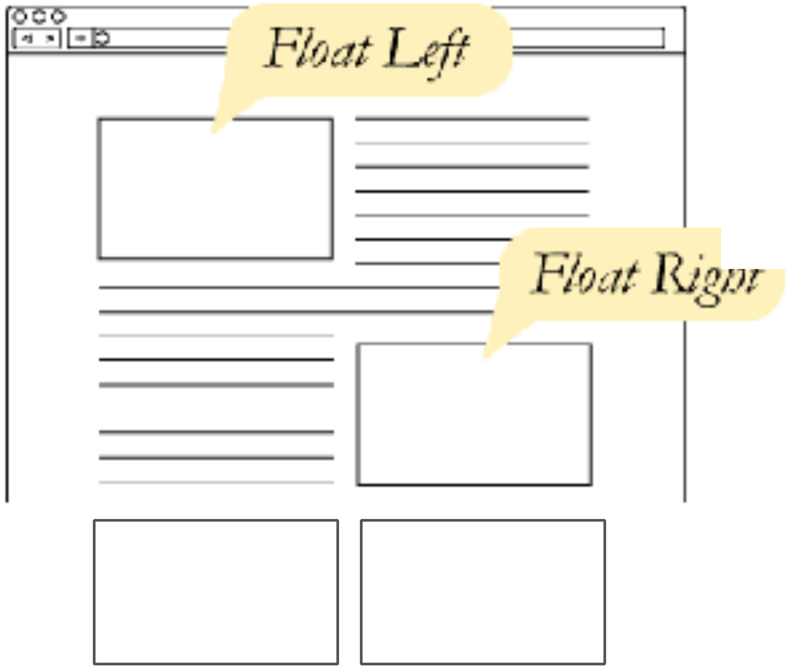
```
a:link {
  color: blue;
}

a:hover {
  color: yellow;
}
```

FLOATING ELEMENTS WITH CSS

WHAT IS FLOAT?

Float is a css property that can be used to place elements beside each other.

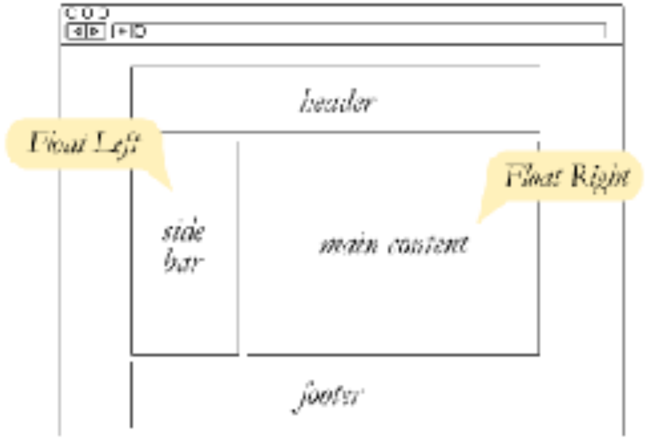
Floated elements are similar to inline elements except in the case of an image, the text flows around it. Eg. https://jsfiddle.net/kimgoulb/2z1Lktr1/



WHEN ARE FLOATS USED?

Floats are helpful in more complex layouts where elements of the page need to be “reflowed”.

- Article style layouts
- Grid based layouts
- Elements that need to sit beside each other.



HOW TO USE THE FLOAT PROPERTY

Available values are *left*, *right* and *none*(typically used to reset/unfloat an element at a smaller display size).

```
aside {
   float:left;
}
```

SIDEBAR

```
div {
   float:right;
}
```

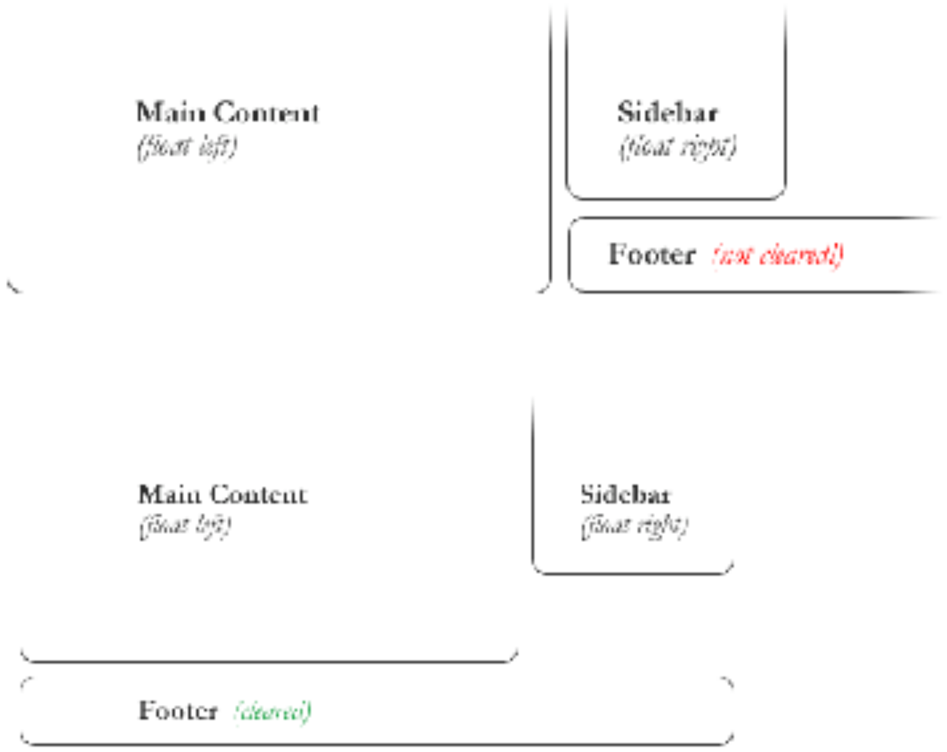
MAIN CONTENT

CLEARING THE FLOAT

An element that has the clear property set on it will not move up adjacent to the float like the float desires, but will move itself down past the float. Available values are *left*, *right*, *both*, *none* and *inherit*. Eg. https://jsfiddle.net/kimgoulb/g0yatgjw/

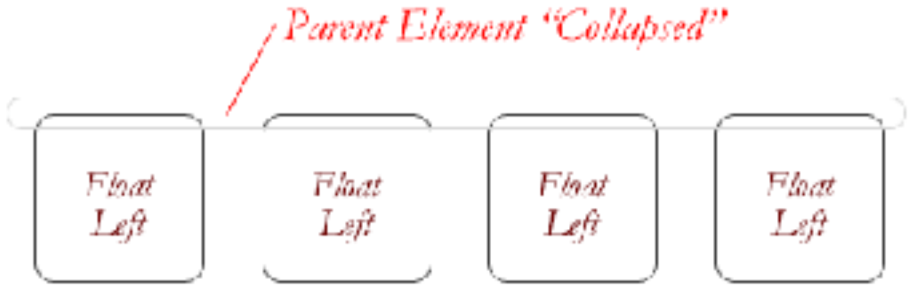
```
footer {
    clear: both;
}
```

Note: “clear” on an element only clears the floated elements before it in document order.



THE GREAT COLLAPSE

Floats always affect their “parent”element. If this parent element contains nothing but floated elements, the height of it would literally collapse to nothing. This odd behavior affects any element that comes after it. Eg. https://jsfiddle.net/kimgoulb/g0yatgjw/



TECHNIQUES FOR CLEARING FLOATS & FIXING THE COLLAPSE

In some cases, simply adding the clear property to the element after will fix this but in most cases, you will have to do more.

- *The empty <div> method:* This is an older less used method that simply puts an empty <div> container with the “clear:both” property at the end of the parent container.

```
<div class="parent">
  <aside></aside>
  <div class="main-content"></div>
  <div style="clear:both;"></div>
</div>
```

HTML

- *The overflow method:* This is a more common method which relies on setting “overflow:hidden” on the parent element. Not always the best choice if the container needs to hold it’s scroll.

```
.parent {
  overflow: hidden;
}
```

CSS

```
<div class=“parent”>
  <aside></aside>
  <div class=“main-content”></div>
</div>
```

HTML

MORE ON THE OVERFLOW PROPERTY

The overflow property specifies what happens if content overflows an element's box. It can be used to activate scrollbars on elements where the height is shorter than the content. Common values are auto, hidden and scroll. It can also be used to contain elements that are floated and properly close the container.

```
div {
  height: 200px;
  overflow: auto;
}
```

IF CONTENT IS LONGER, SCROLLBARS WILL APPEAR

```
.main-wrapper {
   overflow: hidden;
}
```

WRAPPED AROUND THE SIDEBAR AND MAIN CONTENT

- *The clearfix:* uses a clever “CSS pseudo-element(::after)” to clear floats. Rather than setting the overflow on the parent, you apply an additional class like "clearfix" to it and set the styles on that class.

```
.clearfix::after {
  content: ".";
  visibility: hidden;
  display: block;
  height: 0;
  clear: both;
}
```

CSS

```
<div class=“parent clearfix”>
  <aside></aside>
  <div class=“main-content”></div>
</div>
```

HTML

LAB TIME

LAYOUT CHALLENGE

INSTRUCTIONS:

- Use your new knowledge of floats to create these different layouts.
- Create a repo for this project called "*layout-challenge*" and submit when finished.

LUNCHTIME

RESET & NORMALIZE

DEFAULT BROWSER STYLES

RESET.CSS

Reset.css is a stylesheet that will remove all the default, pre defined styles applied by the browser to give you a blank canvas.

It removes things like extra padding and margin around elements that can affect the way your custom styles render.

* Tonight's lab will include a reset.css file that you can use on all your projects.

NORMALIZE.CSS

Normalize.css is a base stylesheet meaning its the starting point for your website styles and it styles the default elements to be consistent across the browsers.

* Tonight's lab will include a normalize.css file that you can use on all your projects.

STYLING: BOX MODEL

DESIGN, LAYOUT & BOXES

TIPS ON EVALUATING A DESIGN

- Every element on a webpage is considered a box. When looking at a design it can help determine structure and styles.
  - Before coding draw the layout as perceived in squares and rectangles then use the appropriate tags to define them.
- Some regions have their own pre-defined tags such as the header or the footer; use those first.
- Elements that are beside each other are typically styled using “floats” (we will cover that in the next class).

STYLING: BOX MODEL

LAB TIME

RELAXR LANDING PAGE

INSTRUCTIONS FOR IN CLASS:

- Let’s wireframe the structure of the page based on the design
- Create the project repo and publish to the cloud
- Create the index.html and style.css file
- Link all 3 css files to your index
- Get the Google Font Open Sans and link it to your index using the method provided by Google
- Start adding the content and styles

REQUIREMENTS:

- Include meta and title tags
- Use HTML5 structural elements (nav, header, footer)
- Include the CSS Reset and Normalize file in addition to your style.css
- Use IDs and Classes to to select and style elements on the page
- Style your text with the Google Fonts provided by your style guide
- Follow naming conventions, maintain consistency across .html and .css files and use best practices for naming IDs and Classes
- Indent nested elements to increase your code's readability

RESOURCES

A FEW LINKS

STYLING: BOX MODEL

- http://www.w3schools.com/css/css_boxmodel.asp
- https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Box_Model/Introduction_to_the_CSS_box_model
- http://learn.shayhowe.com/html-css/opening-the-box-model/
- http://cssreset.com/what-is-a-css-reset/
- http://thesassway.com/intermediate/avoid-nested-selectors-for-more-modular-css