Lesson 1: Course Overview

What We'll Be Building

Throughout this course we're going to learn HTML and CSS by building out the One Million Lines site. In the process of making the site you will learn everything you need to know about HTML and CSS so that you can create any site you want.

I think you are going to be surprised at how easy it is to build great looking sites fast and hopefully you have flashes of feeling like you have a new super power.

Download Resources

- Sublime Text
- Google Chrome Browser
- Bootstrap

Lesson 3: Tips For Course

Use these suggestions to crush this course and be building awesome sites in no time.

1. Go all the way through the course the first time following along **typing in the code yourself**.
2. After the first time, then go to create your own site - any type of site it doesn't have to be the same style and structure as the One Million Lines site - and use these videos, source codes, and other learning materials provided in the lessons as guides
3. **QUANTITY over quality**. What I mean by that is to not worry about creating the perfect site, with the perfect code and don't worry about messing up. Instead just focus on creating sites as fast as you can. In the process of pumping out sites you'll find yourself doing the same actions such as adding images or creating 3 columns as well as many others and they'll become second nature to you over the time. For those of you that liked the pottery story First 20 Hours is an amazing book that outlines a simple process of learning how to learn - in other words how to acquire new skills quickly. Its been amazing for me as a tool in my desire to continue learning new skills.
4. **Practice in shorts bursts.** I don't recommend you going through this course in its entirety in one sitting. Instead spread out the learning with hour long sessions to keep you excited and wanting more. During those hours sessions create breaks and break up the 1-2 hours into short bursts. Use e.ggtimer to set a timer and practice in 20 minute bursts with 5 minute breaks afterwards.
5. I can't stress this enough, but **repetition, repetition, repetition**. Keep creating, keep building sites and it will be second-nature in no time

6. Code snippets will be included in lessons so you can easily copy and paste or type the code. If you're ever having trouble getting the code to work or look the same in your browser as it does in the lesson you can always find the zipped source code for that file at the bottom of each lessons page. Click on the link to download, unzip the file, and you will have all the files as they should be at the end of that video.

Lesson 4: Hello World

Let's dive into bootstrap and get your first site up.

1. Download Bootstrap here. Make sure to choose Download Bootstrap option.

2. Unzip the bootstrap folder and rename "dist" folder as "first-site" folder

NOTE: IF you are a PC user when you zip/extract the Bootstrap zip file it will take you to a folder titled `bootstrap-3.1.1-dist` or with a number similar to that - 3.1.1 is the Bootstrap version number. Take that file and rename it `first-site`. Then copy the file and paste it on your desktop. Once you've done that open the `first-site` folder in Sublime as directed in the next step.

3. Open Bootstrap in sublime

On a mac you can grab the first-site folder and drag it over Sublime app icon (if the Sublime app icon is in your bottom app doc) to open. Otherwise open Sublime and click File at the top, then Open Folder or Open if on Mac. Locate the first-site folder, highlight it and then click open.

4. Create and save an index file

Right click on the "first-site" folder and choose new file. Highlight the code below (or on Bootstrap site) and insert it into the document. Save the file and name it "index.html".

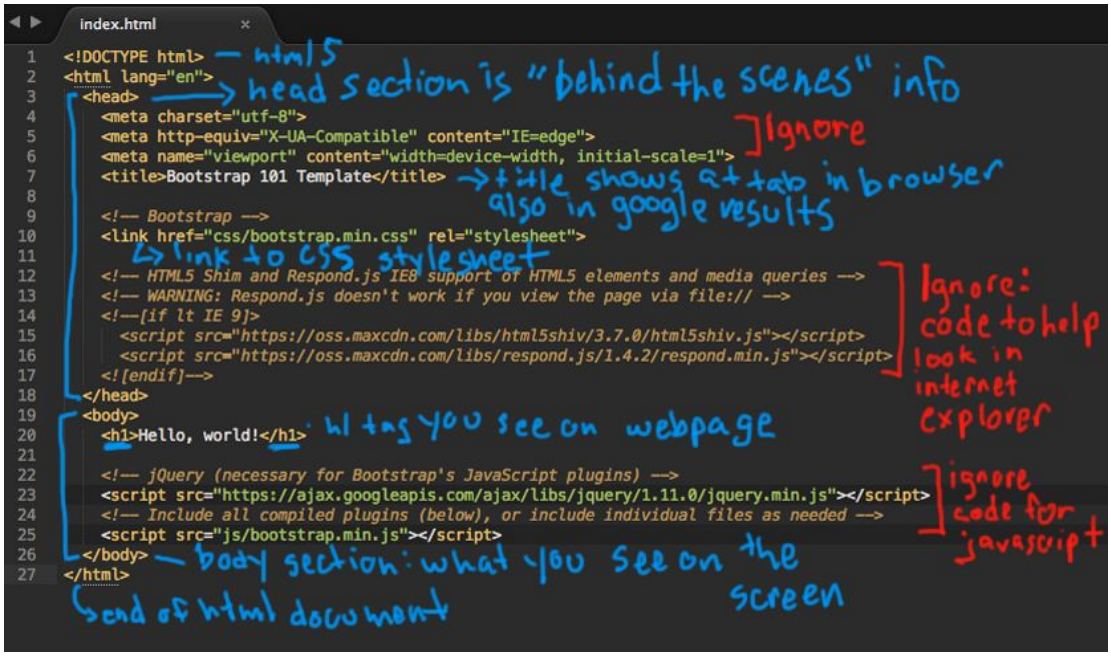
5. Open the index.html file in google chrome to view your first ever web site!

Find your file (in finder for mac, in your folders for pc) and right click on it selecting "open with" > "Google Chrome". Nerdy tidbit: Hello World is common practice among coders to do as their first program or output in any new coding language. Welcome to our world!

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Bootstrap 101 Template</title>

    <!-- Bootstrap -->
    <link href="css/bootstrap.min.css" rel="stylesheet">

    <!-- HTML5 Shim and Respond.js IE8 support of HTML5 elements and media queries -->
    <!-- WARNING: Respond.js doesn't work if you view the page via file:// -->
    <!--[if lt IE 9]>
      <script src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"></script>
      <script src="https://oss.maxcdn.com/libs/respond.js/1.4.2/respond.min.js"></script>
    <![endif]-->
  </head>
  <body>
    <h1>Hello, world!</h1>

    <!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.0/jquery.min.js"></script>
    <!-- Include all compiled plugins (below), or include individual files as needed -->
    <script src="js/bootstrap.min.js"></script>
  </body>
</html>
```

Lesson 5: An HTML Document

Become familiar with the different types of tags throughout this course and how to open and close each. Below is a one-pager for you to refer back to throughout the course. Also use W3Schools.com as a reference throughout the course.



Lesson 7: Let’s Bootstrap!

Bootstrap offers a handful of templates for you to jumpstart your site's creation process. We'll learn how to grab any of these templates we want.

1. Go to Bootstrap's getting started section and grab the Jumbotron template at 0:35

After you click to open the Jumbotron template then right click on the page and select "view page source". Copy the entire HTML document and replace all the code in your "index.html" file currently with the copied code. Save the new index.html file.

2. Edit the CSS path so that it is correct

index.html

change

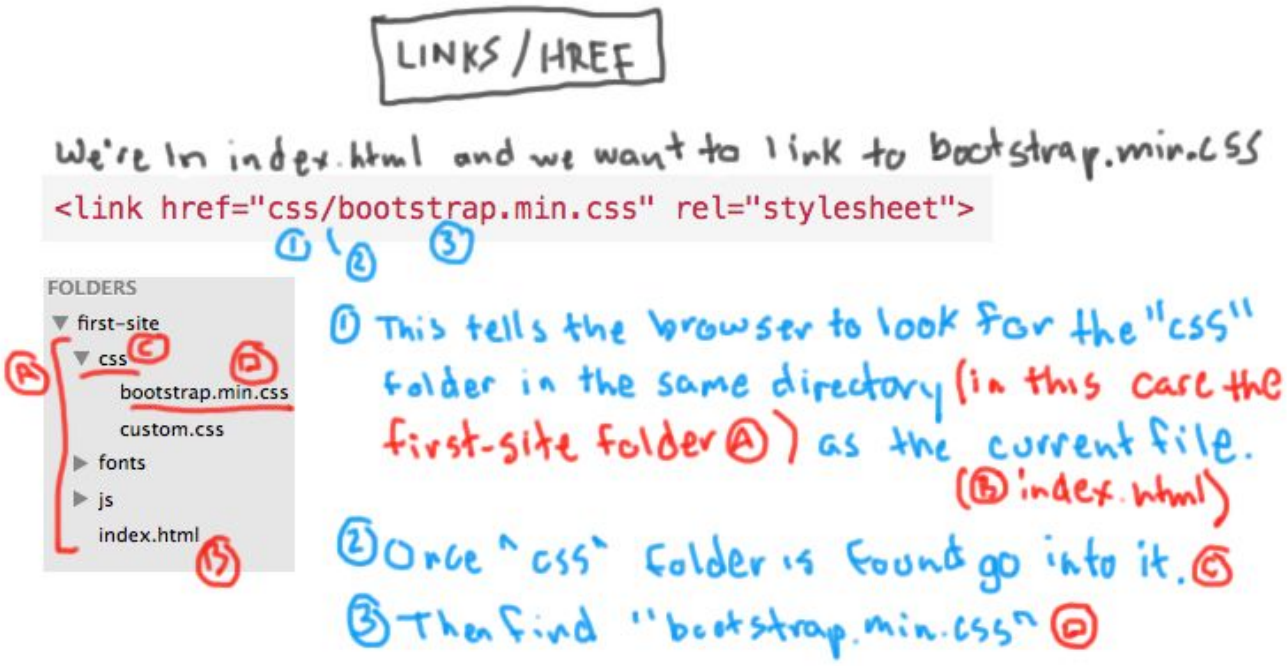
```
<link href="../../dist/css/bootstrap.min.css" rel="stylesheet">
```

to

```
<link href="css/bootstrap.min.css" rel="stylesheet">
```

Link Notes

We're creating the link so that it is relative to the current file. In other words we're telling the browser how to get to (or the path to) the correct CSS document from the index.html file. So "css/bootstrap.min.css" is telling the browser to look for the "css" folder that is in the current folder - which is "first-site" - then go into it and find the "bootstrap.min.css" file.



3. Include our own custom.css file

View page source on the jumbotron template page if you don't still have it open. Then click on the jumbotron.css link. Copy all of the text in the document. Go to sublime and right click on the CSS folder. Click on New File, paste all the text in the document and save it as"*custom.css*" in the CSS folder. Then change the current "*jumbotron.css*" CSS link in our index.html file to our newly created "*custom.css*"file.

CONGRATS! You just created your first CSS file

Extra Lesson Notes

1. In this lesson we are linking to the CSS file via a relative path from the file we are currently in (index.html). In a future lesson we'll discuss how to do absolute links like to http://www.google.com.
2. Comments in HTML documents (or any coding documents for that matter) are for the human eye only. The code that indicates the beginning and the ending of a comment tells the browser just to skip over that section and not worry about the code in there. Comments begin with "<!--" and end with "-->". For example: `<!-- This is where you put the comment -->`
3. You've also seen CSS comments in this video. Check out the first line of your custom.css file where it says: `/* Move down content because we have a fixed... */`. Comments in CSS start with "/*" and end with "*/".
4. Make sure to include the custom.css file lower in our "*index.html*" file. This way the browser reads the custom.css file second after bootstrap's CSS file. Doing it this way makes sure that the CSS rules we include in our "*custom.css*" file will take precedence over the same ones in bootstraps's "*bootstrap.min.css*" file. Don't worry if you don't understand this right now. You will see it in action soon and it will soon become second-nature.

5. **Bonus**** You don't need to know this, but is an interesting tidbit. A "min" file is a minimized version of a file. So instead of linking to a .css file in this lesson we are linking to a .min.css file. Browsers have to read the entire CSS file to interpret it. What a minimized version of a file does is take out all the unnecessary spaces and line breaks so that the browser can read the file faster and thus your site loads faster. Every extra line and space in a document makes it longer to for the browser to interpret it. Minimized files eliminate these unnecessary wastes.

Lesson 8: Editing the Navbar

In this lesson we'll explore how to combine templates, use Chrome's dev tools, and utilize Bootstrap's native components to start putting together the site fast.

1. Grab the Bootstrap's Starter Template navbar

Copy the navbar div (as seen in video) of the Starter Template and replace the current navbar div in our "*index.html*" with it.

index.html

change

```
<div class="navbar navbar-inverse navbar-fixed-top" role="navigation">
  <div class="container">
    <div class="navbar-header">
      <button type="button" class="navbar-toggle" data-toggle="collapse"
data-target=".navbar-collapse">
        <span class="sr-only">Toggle navigation</span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
      </button>
      <a class="navbar-brand" href="#">Project name</a>
    </div>
    <div class="navbar-collapse collapse">
      <form class="navbar-form navbar-right" role="form">
        <div class="form-group">
          <input type="text" placeholder="Email" class="form-control">
        </div>
```

```
        <div class="form-group">
          <input type="password" placeholder="Password" class="form-control">
        </div>
        <button type="submit" class="btn btn-success">Sign in</button>
      </form>
    </div><!--/.navbar-collapse -->
  </div>
</div>
```

to

```
<div class="navbar navbar-inverse navbar-fixed-top" role="navigation">
  <div class="container">
    <div class="navbar-header">
      <button type="button" class="navbar-toggle" data-toggle="collapse"
data-target=".navbar-collapse">
        <span class="sr-only">Toggle navigation</span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
      </button>
      <a class="navbar-brand" href="#">Project name</a>
    </div>
    <div class="collapse navbar-collapse">
      <ul class="nav navbar-nav">
        <li class="active"><a href="#">Home</a></li>
        <li><a href="#about">About</a></li>
        <li><a href="#contact">Contact</a></li>
      </ul>
    </div><!--/.nav-collapse -->
  </div>
</div>
```

2. Change the navbar-brand to be One Million Lines

index.html

change

```
<a class="navbar-brand" href="#">Project name</a>
```

to

```
<a class="navbar-brand" href="#">One Million Lines</a>
```

3. Move the links in the Navbar to the right

In the bootstrap components we find that there is a "navbar-right" class to do just that. Add the "navbar-right" class to our "ul" element.

index.html

```
<ul class="nav navbar-nav navbar-right">
```

4. Change the links in the navbar

index.html

change

```
<ul class="nav navbar-nav navbar-right">
  <li class="active"><a href="#">Home</a></li>
  <li><a href="#about">About</a></li>
  <li><a href="#contact">Contact</a></li>
</ul>
```

to

```
<ul class="nav navbar-nav navbar-right">
  <li class=”active”><a href="#about">WHO WE ARE</a></li>
  <li><a href="#contact">GET INVOLVED</a></li>
</ul>
```

NOTE: We'll worry about the href in the above code later in the course

Extra Lesson Notes

1. **Child and parent elements.** An HTML document’s structure can be looked like a family tree. Each element has a parent element. An element can have a child (or children) element if it contains other HTML elements. Indentation within a document does not determine whether an element is a child or a parent element. Instead the indentation just keeps the document organized so we can see it easier visually. In the example below the div with class navbar is the parent element of the div with class of container. The *ul* is a child element of the div with class collapse and navbar-collapse. At the same time the *ul* is the parent element of the two *li* HTML elements

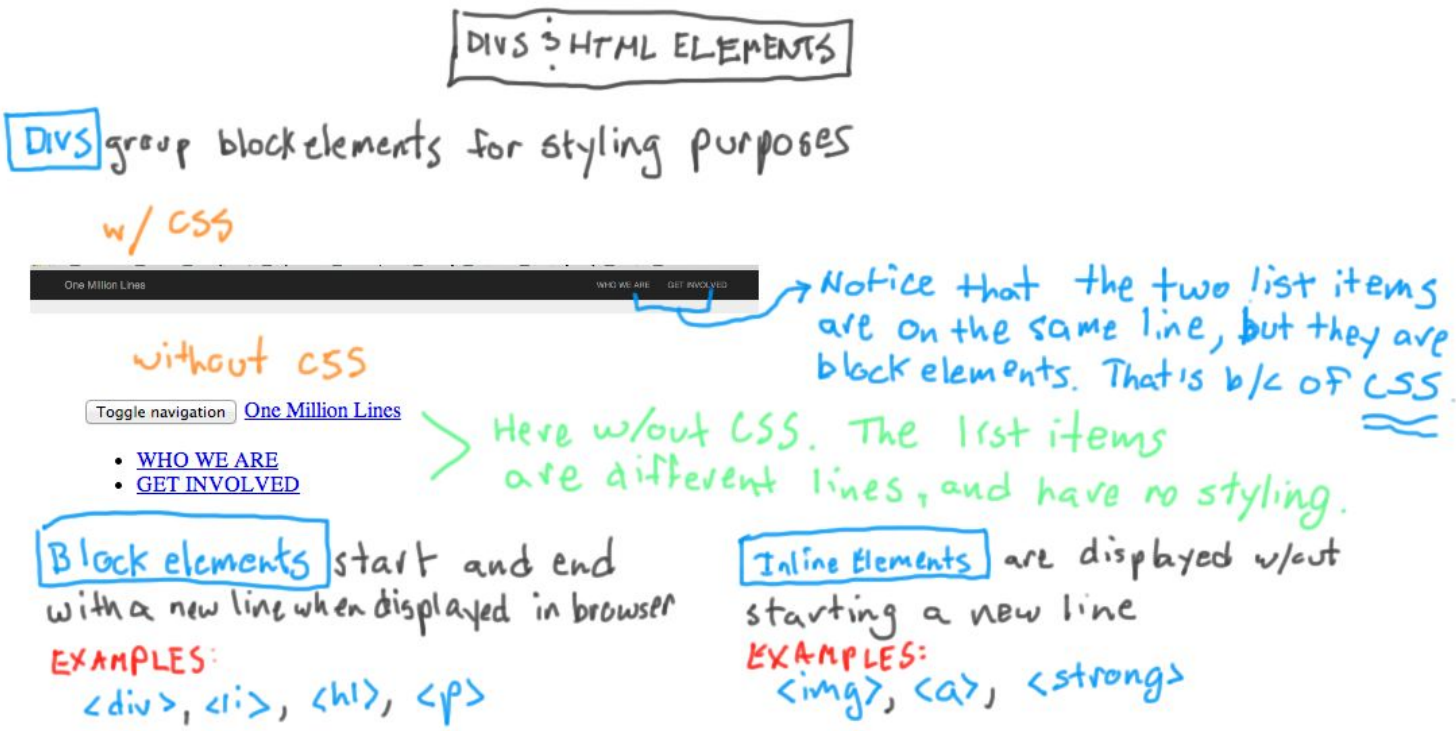
```
<div class="navbar navbar-inverse navbar-fixed-top" role="navigation">
  <div class="container">
    <div class="navbar-header">
      <button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">
        <span class="sr-only">Toggle navigation</span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
      </button>
      <a class="navbar-brand" href="#">One Million Lines</a>
    </div>
    <div class="collapse navbar-collapse">
      <ul class="nav navbar-nav navbar-right">
        <li class="active"><a href="#about">WHO WE ARE</a></li>
        <li><a href="#contact">GET INVOLVED</a></li>
      </ul>
    </div><!--/.nav-collapse -->
  </div>
</div>
```

2. The *Inspect Element* feature of Google Chrome is an awesome tool to experiment and use as your sandbox to try new things and learn about how changes in CSS and HTML will look live on your site. You'll become much more familiar with this tool throughout the course.

Lesson 9: What the Div?!

Divs are all over HTML documents. Understand divs and you'll be like an HTML magician. Feel the power!

Lesson Notes



- A div defines a division or section in an HTML document
- A div is used to group block elements so that you can format them with CSS. Their main benefits are organization and for styling purposes. As you can see in the image above of the difference between an unordered list with and without CSS
- The class *.container* keeps all of our site content lined up within a certain width
- Divs are also block elements, which means that there is a line-break before and after the element by default. You can change this through CSS as Bootstrap did for us in the Navbar.
- Unordered lists - `<ul>` - create bulleted lists, ordered lists - `<ol>` - create numbered lists. Use `<li>` for each list item in both types of lists. Learning to style these lists as Bootstrap does for us will be for a future lesson. `<ul>`, `<ol>`, and `<li>` are all block elements as you can see when we took away the CSS.
- Block elements start and end with a line-break. Inline elements are displayed without starting a new line - an example we've seen so far is the `<a>` tag. This is why you can have the `<li>` and `<a>` tags together in the code and they're on the same line. Also in the example you see the button and the link are on the same line when CSS is not present. This is because they are both inline elements.

Lesson 10: Your First CSS Rule

CSS controls the style of your site. Make your first CSS rule and start transforming the site from boring text to sexy startup quality.

1. Update the h1 tag

index.html

change

```
<h1>Hello, world!</h1>
```

to

```
<h1>Our goal is to inspire <br>Tallahassee to write 1,000,000<br> lines of code_</h1>
```

NOTE: `<br>` is a line break in HTML. Notice on the web page that after each `<br>` the text goes to a new line

2. Update the paragraph

index.html

change

```
<p>This is a template for a simple marketing or informational website. It includes a large callout called a jumbotron and three supporting pieces of content. Use it as a starting point to create something more unique.</p>
```

to

```
<p>All over the country people are taking the <strong>HOUR OF CODE</strong> challenge issued by <strong>CODE.org</strong>. Millions of lines of code are being written. In the capital of Florida, Tallahassee, the community is taking the challenge and our goal is to write 1,000,000 lines of code_</p>
```

NOTE: `<strong>` boldens the text that the opening and closing strong tags wrap.

3. Center all the text in the div with a class of "jumbotron"

css/custom.css

```
.jumbotron {
  text-align: center;
}
```

NOTE: The above code translates to "for any div with a class 'jumbotron' align all text in the center." This is your first CSS rule. Make sure you follow the syntax exactly.

About the inspect element feature: When you have the proper HTML element selected you can type CSS declarations for that HTML element on the right in the element.style section. For example, in this part of the video I have selected the div with a class of jumbotron so whatever CSS declarations I put in the element.style section will apply to that div only. To reset the style just reload the page. This feature is your sandbox to have fun!

4. Make the button red and change its text

Here we take advantage of Bootstrap's prewritten CSS classes for buttons

index.html

change

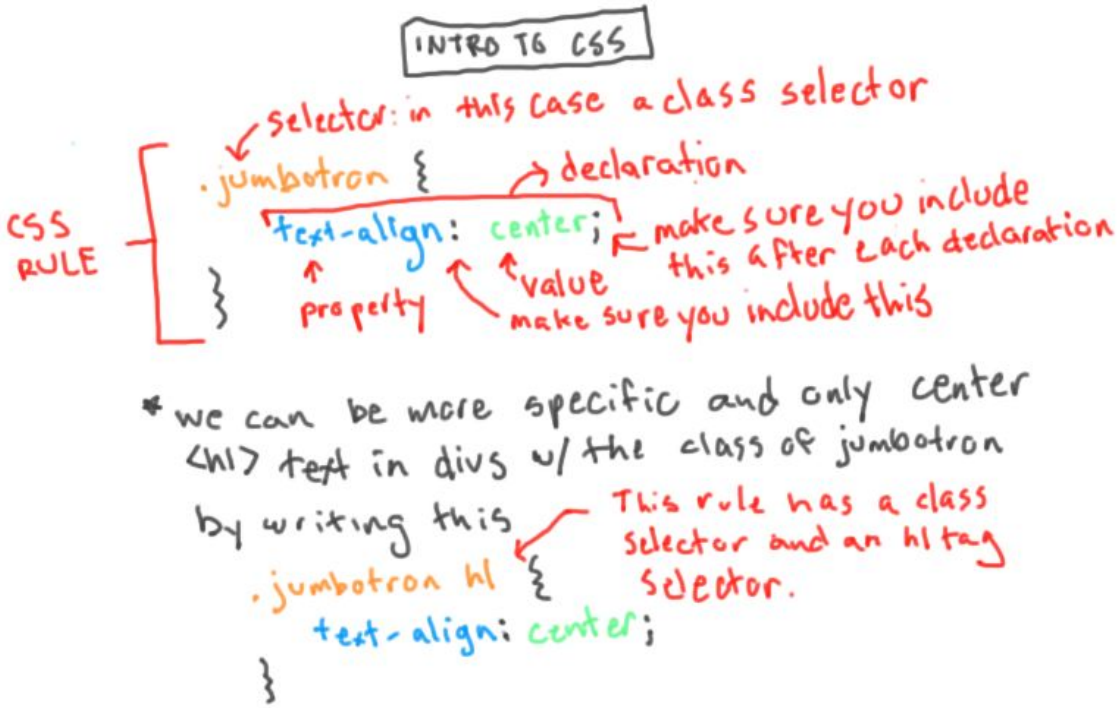
```
<a class="btn btn-primary btn-lg" role="button">Learn more »</a>
```

to

```
<a class="btn btn-danger btn-lg" role="button">Get Involved</a>
```

Lesson Notes

- In addition to `<br>` and `<strong>` an opening and closing `<em>` tag makes the text it surrounds display in italics.
- Giving an HTML element a *class* is just a naming convention so that we can know what to refer to it as and assign it a style in CSS. There is also an *id* naming convention which we will cover later including when to use which one (id or class).
- You denote a class in CSS by putting a period in front of the class name. For the example, in the video we defined styling for the jumbotron class in our CSS document by typing `.jumbotron`
- Look at the image below to learn more about CSS syntax and the different parts of a rule



Lesson 11: More CSS Fun

Lets take a quick moment to learn more about CSS

1. Make the button bigger

Here we take advantage of Bootstrap's prewritten CSS classes for buttons

css/custom.css

```
.btn-lg {
  font-size: 36px;
}
```

NOTE: After you change this above, inspect the button as I did in the video. Notice on the right how there is a `.btn-lg` CSS rule from the "jumbotron.min.css" document that makes a declaration `font-size: 18px`. Also notice how that declaration is crossed out and the CSS rule you made in "custom.css" is above it and the declaration `.btn-lg { font-size: 36px; }` is not crossed out. This is because we included the custom.css file lower in our HTML document than the bootstrap.min.css file. So now when any declarations or rules conflict between Bootstrap's and our custom.css file the one in our custom.css file takes priority.

Lesson Notes

- 3 ways to include CSS
    - External - how we've been doing with custom.css
    - Internal - in the header as shown in the video
    - Inline - using the style attribute within the element's opening tag
- External and internal style sheets have the same priority (assuming the rules have the exact same HTML selectors) so whichever is read last (in other words included in the head section further down in the document) will be the CSS rule that takes place. This is why we are able to overwrite Bootstrap's CSS with our own custom.css - because we include the custom.css lower in the HTML document.
- So the above bullet as to do with the *cascading* part of CSS. Now let's talk about *specificity.* If there are two competing style rules for an element the CSS rule that had the more specific selectors will take priority. In other words if one rule says all text in a div with a class of jumbotron should be red, but then there is another rule that says all text of any p that is in a div with a class of jumbotron should be blue. Which color will the p text be? Well which is more specific? In the first rule it applies to any HTML element in a div with that class, but in the second rule it is limiting it only p's in a div with that class. Thus the second rule is more specific so the p text will be blue and other html elements their text will be red.
- Inline styling is the most specific styling and will take highest priority so that the inline style rule will occur. This is because when you style an individual HTML element you can't get more specific than that because you a declaring a CSS rule for that HTML element alone and no other.
- Use External Stylesheets so that you can make small CSS changes across your site quickly
- In this lesson we made a CSS rule with the font-size property. You can learn more about the font-size property and other CSS font properties here. I like to use pixels when I refer to font-size

because it gives the browser an exact size, but some developers use *em* or % which are relative measurements.

Lesson 12: Get Your Font On

If you've ever heard of Steve Jobs you know that fonts are critical in the design of a product. Let's learn to put any font you want into your website.

1. Find and include the Arvo font

You can find the Arvo font at the Google Fonts page here.

index.html

include

```
<link href='http://fonts.googleapis.com/css?family=Arvo' rel='stylesheet' type='text/css'>
```

2. Make CSS rule for ul.nav and .jumbotron h1 to have Arvo font

css/custom.css

```
ul.nav, .jumbotron h1 {
  font-family: 'Arvo', courier, serif;
}
```

This will change the font-family of an ul with a class of nav and any h1 in a div with the class "jumbotron".

NOTE: You could include multiple font-family names after Arvo for fallbacks (as we did here) in case Arvo doesn't load correctly from Google. Learn more about how to do that here.

Lesson Notes

- So we've seen 2 new types of selectors in CSS in this lesson. Last lesson we did `.jumbotron` to define styling in any HTML element with a class of jumbotron.
    - However in this less we used `.jumbotron h1` to style only h1 tags in our div with the jumbotron class. So this is saying any HTML element with a class of jumbotron that has an h1 within it (another way to say would be that has an h1 child element) style those h1's this way.
    - Then we also used `ul.nav` in this lesson. This says any *ul* that has a class of *nav* style it this way. So you may be asking the question well we didn't put div before *.jumbotron*. We could

have. So we could have said `div.jumbotron` and it would be referring to only *divs* with a class of *jumbotron*, but since in our HTML we have only given a class of *jumbotron* to divs and we haven't given any *uls* or any other type of HTML element the class of *jumbotron* then it doesn't matter.

    - So notice the difference in the two naming conventions of the above examples with the selectors. A selector with a class and then an HTML tag (ex. `.jumbotron h1`) refers to the styling of the h1 child elements of *divs* (or any HTML element for that matter since div wasn't specified) with a class of jumbotron. In contrast to an HTML tag followed with a class in a selector refers to only HTML elements that are the tag mentioned and has the class mentioned, for example `p.lead` refers only to paragraphs with a class of lead (the HTML would look like this `<p class="lead">content</p>`).
- Also we've seen in this video that you can combine multiple selectors in one CSS rule. To do this all you have to do is separate the additional selectors with a comma and a space - for example: `ul.nav, .jumbotron h1 { CSS rule content here }`.

Lesson 14: The Box Model

Every html element is a rectangular box. Understand this along with what margin, padding, and border and you're on your way to being a website ninja

1. Change the list items `<li>` in the unordered list `<ul>` to be smaller

css/custom.css

```
ul.nav {
    font-size: 13px;
}
```

2. Include correct margin for the .jumbotron `<h1>` and `<p>`

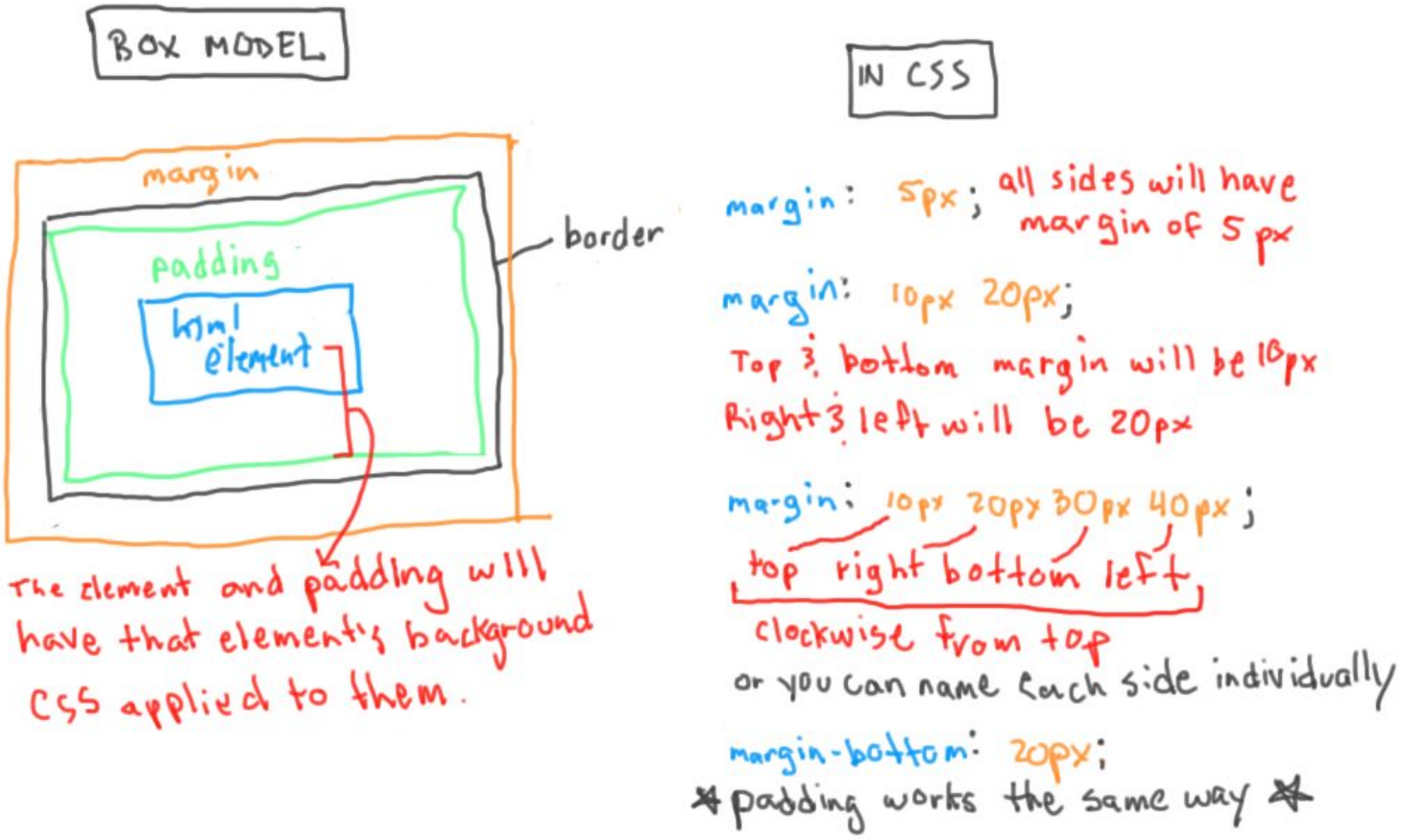
css/custom.css

```
.jumbotron h1 {
    margin-top: 0px;
    margin-bottom: 50px;
}

.jumbotron p {
```

```
  margin-bottom: 30px;
}
```

Lesson Notes

- Every HTML element is in the shape of a box and has the box model applied to it. Inspect elements on the page and highlight various HTML elements in order to see this in action.
- When inspecting HTML elements in Google Chrome blue is the HTML element itself, green is padding, and orange is the margin.
- The HTML element and the padding is within the border. Margin is outside the border. Use padding if you want to add spacing within the border; in other words if you want to expand the background color of that particular element. Use margin if you want to add spacing outside of the border. With margin the spacing will have the same background as that particular elements parent HTML element.
- You can visualize what an element's parent is by indentation in the inspect element area. In our video example the `<body>` element is the parent of the `<div class="jumbotron">` element. See how the `<div class="jumbotron>` is indented one tab in and under the`<body>` element. Then when you click on the arrow to the left of the `<div class="jumbotron>` to open it you'll see`<div class="container>` appear which is the child element of the `<div class="jumbotron>` element.
- There are multiple ways to declare the padding and margin for an HTML element. We went over them in the video and you can use the image below to as a reference.



Lesson 15: Bringing In The Images

Lets go through how to add images to your site so that it really pops, both through HTML and CSS. And now we'll be done with the top section. Pat yo' self on the back!

1. Include the Code.org image

You can find the images in the lesson description/download tab to the right. Download the images. Then create a new folder within your "first-site" folder titled "images". Save the images in that folder.

index.html

include

```
<img src="images/code.png" title="Code.org logo" alt="Code.org">
```

on the line after the button

The `alt` attribute is what will show if the image loads incorrectly and the `title` attribute is what will show if you hover over the image long enough with your mouse. These are optional, but can help your site's accessibility say if the image doesn't load correctly or if someone who is visually impaired comes to your site - the screen reader will read what image should be there since they can't see it. **Every HTML element has a set of HTML attributes that go along with it. You can always find them for each element at w3schools.com.**

2. Insert background image

You can get image in the download/lesson description tab. Save it as hero_image.jpg.

css/custom.css

```
.jumbotron {
  text-align: center;
  background-image: url(“../images/hero_image.jpg”);
}
```

- We were introduced to links earlier. The only thing new we're adding here is `../` which tells the browser to exit the current folder. In other words, we're in the custom.css file so we're in the CSS folder. We tell the browser to exit the CSS folder so then its in the first-site folder. From there we tell it to find the images folder `images`. To enter the images folder `/` and then to find the `hero_image.jpg` image.
- In this lesson we've use the background image. There are also other background properties you can learn about here.

3. Change font color

css/custom.css

```
.jumbotron {
  text-align: center;
  background-image: url("../images/hero_image.jpg");
  color: white;
}
```

3. Correct font weight, font size and spacing (padding and margin)

css/custom.css

```
.jumbotron h1 {
  margin-top: 0px;
  margin-bottom: 50px;
}
```

```
.jumbotron p {
  margin-bottom: 30px;
  font-weight: 100;
  padding: 0 50px;
}
```

4. Add One Million Lines image

You can get image in the download/lesson description tab. Save it as top_logo.png in the images folder.

index.html

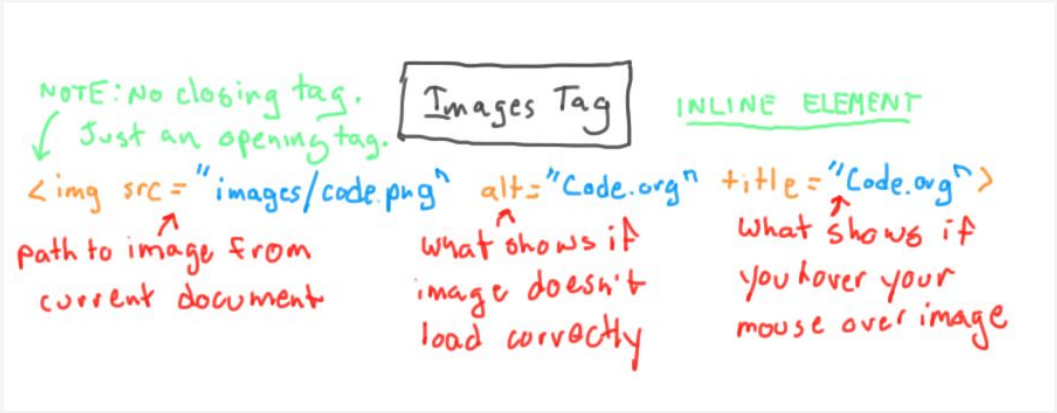
change

```
<a class="navbar-brand" href="#">One Million Lines</a>
```

to

```
<a class="navbar-brand" href="#"><img src="images/top_logo.png" alt="One Million Lines"></a>
```

Use the image below as a reference when creating new image tags.

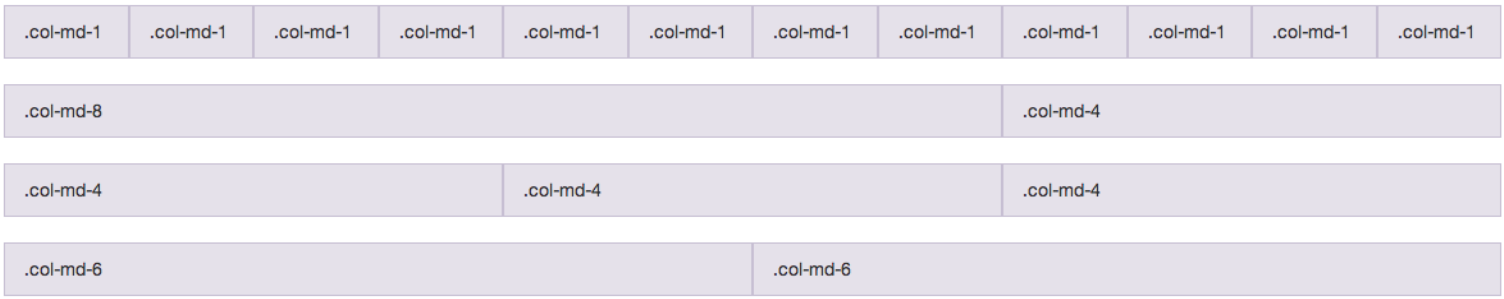


Lesson 16: The Grid System

Lets go into bootstrap's grid system so you can learn to make the site look exactly however you want it. With the added understanding of floats you can build **any** complex-looking site.

Lesson Notes

- In `.col-md-4` the number represents the width, more specifically how many columns wide of the 12-column layout. In other words that div will be 4 columns wide. Since it is out of 12 it will be 1/3 of the page's width. Look at the image below for a quick cheat sheet on the grid system and div widths.



- **Rules**
  - Rows are placed w/in a container div
  - Only columns may be immediate children of rows

Lesson 17: Linking It Up

Use links to control the flow of users from and through your site. In this lesson we cover the different types of links

1. Update content

index.html

change

```
<div class="row">
  <div class="col-md-4">
    <h2>Heading</h2>
    <p>Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Etiam porta sem malesuada magna mollis euismod. Donec sed odio dui. </p>
    <p><a class="btn btn-default" href="#" role="button">View details &raquo;</a></p>
  </div>
  <div class="col-md-4">
    <h2>Heading</h2>
```

```
    <p>Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Etiam porta sem malesuada magna mollis euismod. Donec sed odio dui. </p>
    <p><a class="btn btn-default" href="#" role="button">View details &raquo;</a></p>
 </div>
  <div class="col-md-4">
    <h2>Heading</h2>
    <p>Donec sed odio dui. Cras justo odio, dapibus ac facilisis in, egestas eget quam. Vestibulum id ligula porta felis euismod semper. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus.</p>
    <p><a class="btn btn-default" href="#" role="button">View details &raquo;</a></p>
  </div>
</div>
```

to

```
<div class="row">
  <div class="col-md-4">
    <h2>Students</h2>
    <p>Want to learn how to code? Want to help us get to 1,000,000 lines?  Click the button below and we'll let you know how to get involved.</p>
    <p><a class="btn btn-default" href="#" role="button">Start Learning &raquo;</a></p>
  </div>
  <div class="col-md-4">
    <h2>Educators</h2>
    <p>Want to bring this initiative to your school or institution?  Awesome!  Click the button below and we'll make it happen.</p>
    <p><a class="btn btn-default" href="#" role="button">Join The Iniative &raquo;</a></p>
 </div>
  <div class="col-md-4">
    <h2>Sponsors</h2>
    <p>We love all the support we get to help host more events, and empower more lives with the knowledge of coding.  Click to get involved.</p>
    <p><a class="btn btn-default" href="#" role="button">Give Support &raquo;</a></p>
```

```
  </div>
</div>
```

<a> Notes

- Use the `href` attribute to determine the URL (aka the address) of what page or website the browser will go to when the user clicks on a link
- **Absolute link/URL:** Starting a href with `http://` will direct the user to the exact site destination specified afterwards. In other words I used `http://google.com` as an example in the lesson. Users will be directed to google.com when they click on that link.
- **Relative link/URL:** Take away `http://` and you can link to other pages within your site. The destination page is relative from the current page the link is in and you declare the path the same way we did earlier with images. In other words if you are currently in the `index.html` file and you want to create and link to another page in the "first-site" folder titled `grid.html` you would do `href="grid.html"`. If the grid.html page was in a folder called "about" you would do `href="about/grid.html"`.
- Use `target="_blank"` to open link in a new tab.
- **Added Bonus**: you can use `href="mailto:youremailaddress@domainname.com"` so that if someone clicks they will be set up to email you.
- You can also see more special HTML character symbols here.

Lesson 18: BONUS: Google Forms

Learn how to bring in Google Forms. An awesome, free tool to get feedback from you site users and put a sense of interactivity in your site

NOTE: You can also embed the form into your site so that you can style it to look like your site. We will cover this when we discuss embedding iframes into your site.

Lesson 19: Font Awesome Is Awesome

Font-awesome allows you to include professional quality icons in your site free and easy to give it that extra pop.

1. Download Font-awesome here. Click download.

2. Unzip the font-awesome.zip and rename the resulting unzipped folder as "font-awesome".

3. Take the font-awesome folder and drag or paste it into the fonts folder within your first-site folder

On a mac you can grab the font-awesome folder and drag it into the "first-site/fonts" folder as in video. With PCs you may have to highlight the font-awesome folder and copy it. Then find the "fonts" folder (wihin your "first-site" folder), highlight it, and then click paste and the font-awesome folder should now be within the fonts folder.

4. Include font-awesome in your index.html file

index.html

add

```
<link rel="stylesheet" href="fonts/font-awesome/css/font-awesome.min.css">
```

5. Put an icon in your index.html file to see if we included font-awesome correctly

index.html

add

```
<i class="fa fa-camera-retro"></i>
```

above `<h2>Students</h2>`

NOTE: If you have any trouble including it correctly check the source files to see how to include it correctly.

6. Once you have font-awesome included correctly put in correct icons

index.html

change

```
<div class="row">
  <div class="col-md-4">
    <h2>Students</h2>
    <p>Want to learn how to code? Want to help us get to 1,000,000 lines?  Click the button below and we'll let you know how to get involved.</p>
    <p><a class="btn btn-default" href="#" role="button">Start Learning &raquo;</a></p>
  </div>
  <div class="col-md-4">
    <h2>Educators</h2>
    <p>Want to bring this initiative to your school or institution?  Awesome!  Click the button below and we'll make it happen.</p>
    <p><a class="btn btn-default" href="#" role="button">Join The Iniative &raquo;</a></p>
  </div>
  <div class="col-md-4">
    <h2>Sponsors</h2>
    <p>We love all the support we get to help host more events, and empower more lives with the knowledge of coding.  Click to get involved.</p>
    <p><a class="btn btn-default" href="#" role="button">Give Support &raquo;</a></p>
  </div>
</div>
```

to

```
<div class="row">
  <div class="col-md-4">
    <span class="fa-stack fa-4x">
      <i class="fa fa-circle fa-stack-2x"></i>
      <i class="fa fa-user fa-stack-1x fa-inverse"></i>
    </span>
    <h2>Students</h2>
    <p>Want to learn how to code? Want to help us get to 1,000,000 lines?  Click the button below and we'll let you know how to get involved.</p>
    <p><a class="btn btn-default" href="#" role="button">Start Learning &raquo;</a></p>
  </div>
```

```
  <div class="col-md-4">
    <span class="fa-stack fa-4x">
      <i class="fa fa-circle fa-stack-2x"></i>
      <i class="fa fa-pencil fa-stack-1x fa-inverse"></i>
    </span>
    <h2>Educators</h2>
    <p>Want to bring this initiative to your school or institution?  Awesome!  Click the button below and we'll make it happen.</p>
    <p><a class="btn btn-default" href="#" role="button">Join The Iniative &raquo;</a></p>
  </div>
  <div class="col-md-4">
    <span class="fa-stack fa-4x">
      <i class="fa fa-circle fa-stack-2x"></i>
      <i class="fa fa-money fa-stack-1x fa-inverse"></i>
    </span>
    <h2>Sponsors</h2>
    <p>We love all the support we get to help host more events, and empower more lives with the knowledge of coding.  Click to get involved.</p>
    <p><a class="btn btn-default" href="#" role="button">Give Support &raquo;</a></p>
  </div>
</div>
```

Lesson 20: Playing With Spans

Spans let you style inline-elements. Use it if you want to make certain words, phrases, or other inline elements stand out with a different color or style.

In this example we used span to change the font color and size, but you could use it to do a variety of other things with the font. Go ahead and play around with different font and text properties, combine them, and see what they do. You can find a list of different font properties here.

Lesson 21: Styling The Get Involved Section

Through working on the get-involved section we learn and revisit many text-based CSS properties as well as see how to give the same CSS rule to multiple different HTML elements.

1. Update Get Involved section's HTML content and give it a class

index.html

and change

```
<div class="container">
  <!-- Example row of columns -->
  <div class="row">
    <div class="col-md-4">
      <span class="fa-stack fa-4x">
        <i class="fa fa-circle fa-stack-2x"></i>
        <i class="fa fa-user fa-stack-1x fa-inverse"></i>
      </span>
      <h2>Students</h2>
      <p>Want to learn how to code? Want to help us get to 1,000,000 lines?  Click the button below and we'll let you know how to get involved.</p>
      <p><a class="btn btn-default" href="#" role="button">Start Learning »</a></p>
    </div>
    <div class="col-md-4">
      <span class="fa-stack fa-4x">
        <i class="fa fa-circle fa-stack-2x"></i>
        <i class="fa fa-pencil fa-stack-1x fa-inverse"></i>
      </span>
      <h2>Educators</h2>
      <p>Want to bring this initiative to your school or institution?  Awesome! Click the button below and we'll make it happen.</p>
      <p><a class="btn btn-default" href="#" role="button">Join The Iniative »</a></p>
    </div>
    <div class="col-md-4">
      <span class="fa-stack fa-4x">
        <i class="fa fa-circle fa-stack-2x"></i>
        <i class="fa fa-money fa-stack-1x fa-inverse"></i>
      </span>
```

```
      <h2>Sponsors</h2>
      <p>We love all the support we get to help host more events, and empower more lives with the knowledge of coding.  Click to get involved.</p>
      <p><a class="btn btn-default" href="#" role="button">Give Support »</a></p>
    </div>
  </div>

  <hr>

  <footer>
    <p>© Company 2014</p>
  </footer>
</div> <!-- /container -->
```

to

```
<div class="container homepage">
  <h2>Get Involved</h2>
  <!-- Example row of columns -->
  <div class="row">
    <div class="col-md-4">
      <span class="fa-stack fa-4x">
        <i class="fa fa-circle fa-stack-2x"></i>
        <i class="fa fa-user fa-stack-1x fa-inverse"></i>
      </span>
      <h3>Students</h3>
      <p>Want to learn how to code? Want to help us get to 1,000,000 lines?  Click the button below and we'll let you know how to get involved.</p>
      <p><a class="btn btn-default" href="#" role="button">Start Learning »</a></p>
    </div>
    <div class="col-md-4">
      <span class="fa-stack fa-4x">
        <i class="fa fa-circle fa-stack-2x"></i>
        <i class="fa fa-pencil fa-stack-1x fa-inverse"></i>
      </span>
      <h3>Educators</h3>
      <p>Want to bring this initiative to your school or institution?  Awesome! Click the button below and we'll make it happen.</p>
```

```
      <p><a class="btn btn-default" href="#" role="button">Join The Iniative »</a></p>
    </div>
    <div class="col-md-4">
      <span class="fa-stack fa-4x">
        <i class="fa fa-circle fa-stack-2x"></i>
        <i class="fa fa-money fa-stack-1x fa-inverse"></i>
      </span>
      <h3>Sponsors</h3>
      <p>We love all the support we get to help host more events, and empower more lives with the knowledge of coding.  Click to get involved.</p>
      <p><a class="btn btn-default" href="#" role="button">Give Support »</a></p>
    </div>
  </div>

  <hr>

  <footer>
    <p>© Company 2014</p>
  </footer>
</div> <!-- /container -->
```

2. Center align the font and style the h2 and h3 tags in the Get Involved section classes

css/custom.css

add

```
.homepage {
  text-align: center;
}

.homepage h2, .homepage h3 {
  font-family: 'Arvo', courier, serif;
  color: #e74c3c;
  text-transform: uppercase;
}

.homepage h2 {
```

```
  font-size: 30px;
}

.homepage h3 {
  font-size: 26px;
}

.homepage p {
  font-size: 21px;
  font-weight: 300;
}


.homepage a {
  color: #e74c3c;
}
```

Lesson Notes

- You can give HTML elements multiple classes by separating them with a space as we did in this video `<div class="container homepage">.` This div has 2 classes: homepage and container.
- Let's talk about **colors**. The color of your site is very important to the styling and feeling a user gets when visiting. In the past lessons I have been using word values for color; **I do not recommend this for final product sites**. The pre-assigned color words are harsh on user's eyes. Instead I recommend using hexadecimal values as in this video. Hexadecimal values give you millions of different color options with all types of shading. You can find different color schemes all over the internet and you can even grab different colors from sites you love now that you know how to use the ***inspect element*** feature. One of my favorite sites to grab colors from is flatuicolors.com. Go to the site, click on the color you want to use, and that color's hexadecimal value will be copied for you to paste into your code.

Lesson 22: Centering Block Elements

Let's have some fun learning how to center block elements

1. Include the hr

index.html

add

```
<hr>
```

underneath `<h2>Get Involved</p>`

css/custom.css

add

```
.homepage hr {
  border-top: 1px solid #e74c3c;
  width: 150px;
  margin-top: 15px;
}
```

Lesson Notes

- If you want to center a block element in the middle of its parent element make sure to give it a left and right margin value of auto for example `margin: 10px auto`. In this case that HTML element would have a top and bottom margin of 10 pixels and would be centered.
- Also in this lesson you saw two different types of values for width: **%** and **px**. Percentage (&) is relative to the parent element, in the case shown in the video the **<hr>** % width was relative to the container div. That is why it became smaller as decreased the width of the page's viewing area. A pixel width is absolute and stays the same length without regard of the page's viewing area or the parent element. As stated with font-size which value to use changes per the occasion and the intended result. If you are trying to style a div and you know you want that div to be half the size of that div's container div then give it a width of 50%. If you are styling a line like we are here and we want it to be an exact width then use pixels.

Lesson 23: Seeing The Benefits Of Using Classes

In this lesson you'll see the power of assigning classes to elements when you want multiple elements to have a similar style. It's a beautiful thing!

1. Put in Who We Are section

index.html

```
<div class="container homepage">
  <h2>Who We Are</h2>
  <hr>
  <p>We are <a href="http://www.massiveacademy.us" target="_blank">MASSIVE Academy</a>. We aim to improve education through both method - effective project-based learning - and material - by teaching skills that are applicable to improving your life today.</p>
</div>
```

2. Put in image

You can grab image from the lesson's download tab.

index.html

```
<img src="images/academy_brand_med.png" title="MASSIVE Academy" alt="MASSIVE Academy">
```

Lesson 24: Hello Hover

We're rockin' and rollin' now as we style links for when users hover over them and get an introduction to pseudo-classes.

1. Put in styling for links when a user hovers over them

css/custom.css

add

```
.homepage a:hover {
  color: #e74c3c;
  text-decoration: none;
  opacity: .8;
}
```

Lesson Notes

- The opacity property is the transparency of the HTML element. Its on scale from 0 to 1 with zero being completely transparent to 1 being not transparent at all. A value of .8 would be about 20% transparent. I like to use opacity with links when they're hovered to show a slight change in color as shown in the video - neat little trick.
- **Bonus**: This wasn't covered in the video, but you can combine opacity and color by giving an html element an rgba value. So you can convert hexadecimal values to rgb values here. By doing this our hexadecimal salmon color (`color: #e74c3c;`) can also be styled in CSS with the following code `color: rgb(247, 76, 60);`. Then to give it the slight change in color with the opacity when someone hovers you could use the following CSS code `color: rgba(247, 76, 60, .8);`. Some people like to use the rgba value as a background-color. Say for example you wanted your header to be slightly transparent for an effect you could use **rgba**.

Lesson 25: Beautification Through Background Colors

Lets learn how to style full-width background colors in order to give our some subtle design beauty

1. Change the background color

index.html

Wrap the div `<div class="container homepage">` with another div `<div id="get-involved">`. Also move the closing tag for`<div class="container homepage">` so that it doesn't wrap the `<hr>` and `<footer>`. Don't forget to indent all the lines in between the opening and closing tags of `<div class="container homepage">`.

css/custom.css

add

```
#get-involved {
  background-color: #f5f5f5;
}
```

2. Put in some padding to make the two sections look good

css/custom.css

add

```
.homepage {
  text-align: center;
  padding-bottom: 50px;
  padding-top: 40px
}
```

Lesson Notes

- Notice the naming convention for `id` versus `class`. Where we have been identifying classes by putting a `.` before the class name, to identify an id we put a `#` before the id name. Ex: `#get-involved`.
- There will also be an added benefit to using the `id` tag later when we create links to certain parts of our page from the navbar.

Lesson 26: IDs Versus Classes

In this lesson we'll learn about the benefits of IDs and when to use them versus classes

1. Link to the get-involved div

index.html

change

```
<li><a href="#">GET INVOLVED</a></li>
```

to

```
<li><a href="#get-involved">GET INVOLVED</a></li>
```

2. Wrap the WHO WE ARE section with a div with an id of who

index.html

Wrap the div `<div class="container homepage">` (of the WHO WE ARE section) with another div `<div id="who">`. Don't forget to indent all the lines in between the opening and closing tags of `<div class="container homepage">`.

3. Link to the get-involved div

index.html

change

`<li><a href="#">WHO WE ARE</a></li>`

to

`<li><a href="#who">WHO WE ARE</a></li>`

Lesson Notes

- When to use an ID vs a class? Use an ID when it is a unique element that will have its own type of styling that no other element on your site will have or you can use ID to name an element that you want to link to take a user to that section when a certain link is clicked. Use class when you have multiple HTML elements that you want to style in a similar manner.
- IDs are more specific than classes so they will take priority. In other words if we wanted the h2 in the get-involved section to have a different color than the h2 color we specified by coding `.homepage h2 { color: #ef4c3c; }` we could have put in the rule `#get-involved h2 { color: black; }`. In this case the h2 in the who we are section would still be that specified by the homepage class, but the h2 in the get-involved section would be black since IDs are more specific than classes. This is exactly like what we did with the "educator" ID example in the video.

Lesson 27: Finish Off With The Footer

A quick styling of the footer section and you have yourself your first website. Do a dance!

1. Put image in Footer section

You can grab image from this lesson's downloads tab.

index.html

```
<footer>
  <div class="container">
    <p><img src="images/one_million_bottom_brand.png" alt="One Community_ One Million Lines_"></p>
  </div>
</footer>
```

Also delete the `<hr>` above the footer section.

2. Style the footer section correctly

css/custom.css

```
footer {
  background-color: #e74c3c;
  padding-top: 150px;
  padding-bottom: 30px;
}
```

also change CSS for the body

```
body {
  padding-top: 50px;
}
```

Lesson 28: Making Your Site Look Good Across All Devices

We'll make the site look good across all devices with some responsive design and your bonus intro to the awesome world of Javascript.

1. Make yo' site responsive

css/custom.css

```
/* Responsive Styling */

@media (max-width: 1199px) {

  .jumbotron h1 {
    font-size: 56px;
    padding: 30px;
  }

}

@media (max-width: 991px) {

  .jumbotron h1 {
    font-size: 44px;
```

```
  }

  .jumbotron p {
    padding: 0 10px;
  }

  #get-involved .col-md-4 {
    padding-top: 20px;
    padding-bottom: 20px;
  }

}

@media (max-width: 767px) {

  .jumbotron h1 {
    font-size: 24px;
  }

  .jumbotron p {
    font-size: 16px;
    padding: 0;
  }

  .btn-lg {
    font-size: 18px;
  }

  .homepage p {
    font-size: 18px;
  }

  footer img {
    width: 80%;
  }

}
```

2. Correct the link to Bootstrap's built in Javascript document

index.html

change

```
<script src="../../dist/js/bootstrap.min.js"></script>
```

to

```
<script src="js/bootstrap.min.js"></script>
```

3. Change our page's title

index.html

change

```
<title>Jumbotron Template For Bootstrap</title>
```

to

```
<title>One Million Lines</title>
```

Lesson Notes

- In this lesson we fix the link to Bootstrap's Javascript. Javascript allows us to add functionality to our site. You can see this in the video once we correct the link to the .min.js file as the button now works - when clicked you see the drop-down menu appear. Pretty awesome right?! Through javascript we can make our sites do some extremely cool things and build in functionality our users will love. We'll talk more about javascript and its capabilities in future lessons.

Lesson 29: Getting Your Site Live Online With Dropbox

Let's get yo' site live and online with Dropbox for the world to see.

You can get dropbox here.

1. Make Google Font load over a secure server

index.html

change

```
<link href='http://fonts.googleapis.com/css?family=Arvo' rel='stylesheet' type='text/css'>
```

to

```
<link href='https://fonts.googleapis.com/css?family=Arvo' rel='stylesheet' type='text/css'>
```

Lesson Notes

- Dropbox is a good service to use if you are working on a site and want to have it live from the beginning for people to see - say if you have a client you are building a site for. Just transfer your site's folder into the public Dropbox folder when you start building the site and everytime you make changes in Sublime or any other text editor the changes will be shown live on the web for anyone to see.
- Push: putting the site live online for people to access.

Lesson 30: GitHub Pages

Dropbox allows you to put up sites free and fast. Lets look at another option Github that has the same benefits but also gives you a more human friendly URL

1. Sign in to GitHub or create an account and sign in

You can access Github pages instructions here.

2. Create a new repository

3. Click set up in desktop. Then download and install GitHub for your computer

4. Open Github and click on your username to the right.

5. Find your username and then click on “Clone to Computer”

Clone the repository to where you want. I chose to clone it to my desktop for easy access.

6. Click on the arrow to open the directory in GitHub

At this point it should say “master” and “no commits”

7. Copy all your files and folders from your “first-site” folder and paste them in your “username.github.io” folder

At this point you should see a bunch of changes to be committed when you switch back over to your GitHub app

8. Put in a summary and then click the “commit & sync” button in your GitHub app

Lesson Notes

- Dropbox and Github both allow you to host sites for free and update them almost instantly. The main benefit for Dropbox over Github is that you can put up multiple sites for free. While Github only allows you to put up one site for free per account (you can pay for multiple at a low rate), at the same time the main benefit for Github over Dropbox is that the URL that will come with your Github site is much more human friendly - for example rbonhardt.github.io. Plus, I’ll show you in the next video how to host your Github site for free with your own custom domain name!
- If you continue to work in your “username.github.io” folder each time you make a change in sublime you will see the uncommited changes in your GitHub app for that site. The changes won’t show on your actual site until you commit & sync the changes.

Lesson 31: Custom Domain Name

Let’s get you your first very own custom domain name!

You can buy a domain name at GoDaddy.

Lesson Notes

- No matter what site you are using to buy your domain names you will have to include the CNAME file, change the “www” CNAME in your DNS Zone File Settings, and forward the “domainyouchoose.com” to “www.domainyouchoose.com”. However, if you buy your domain through a site other than GoDaddy the last two will change in terms of how you do it. You can consult that site about the exact steps to do so. A method I’d recommend is to google it. For instance if you bought your site through dreamhost just google “dreamhost change CNAME”.

Lesson 32: What's Next

Now that we know the building blocks of CSS and HTML as well as how to get our sites live online what's next?

---

*Bonus * Lesson 33: Scrollspy Magic

Let's start putting magic into our site with scrollspy.

Intro to Javascript

We're going to add some "magical" javascript to our site in the next couple lessons. While going into and learning javascript is outside the scope of this particular course I wanted to introduce you to a couple easy javascript tricks that have big positive effects on your users' experience. Small changes for big wins that's what we're all about.

With that being said for the sake of this course all you need to know is that Javascript allows programmers (thats me and you :)) to create interactive elements within the site. Which means that as the user interacts with our site it changes, as you can see in this lesson that when the user scrolls down the different navbar list items highlight. The result is the feeling of the user of having a "power" and having a fun interaction with the web page. Ultimately a great experience for our site visitors. So don't worry if you don't understand everything going on in the next couple lessons involving javascript.

These are just meant as bonus lessons to give you a little taste of the power of javascript, equip you with a couple more fun tricks in your website building arsenal and wet your palette to want to learn more. Enjoy!

1. Add `data-spy` and `data-scroll` to body tag

index.html

change

```
<body>
```

to

```
<body data-spy="scroll" data-target=".navbar-collapse">
```

2. Add `data-offset`

index.html

change

```
<body data-spy=”scroll” data-target=”.navbar-collapse”>
```

to

```
<body data-spy=”scroll” data-target=”.navbar-collapse” data-offset=”50”>
```

BONUS Lesson 34: Smooth Scrolling

Let’s put some smooth transitions in our tower page with a smooth scrolling feature.

1. Create a new file, include the stack overflow javascript, and save it

js/smoothscroll.js

```
$(".navbar-collapse ul li a[href^='#']").on('click', function(e) {
   // prevent default anchor click behavior
   e.preventDefault();
   // store hash
   var hash = this.hash;
   // animate
   $('html, body').animate({
       scrollTop: $(this.hash).offset().top
     }, 300, function(){
       // when done, add hash to url
       // (default click behaviour)
       window.location.hash = hash;
     });
});
```

Note: `[href^='#']` is just saying that each `a` link has an href attribute. Nothing more complex than that. So for now just ignore that for the lesson.

2. Link to the smoothscroll.js document in our html document

index.html

add

```
<script src="js/smoothscroll.js"></script>
```

3. Change the speed of the scroll to slow it down

js/smoothscroll.js

change

```
$('html, body').animate({
        scrollTop: $(this.hash).offset().top
    }, 300, function(){
```

to

```
$('html, body').animate({
        scrollTop: $(this.hash).offset().top
    }, 700, function(){
```

4. Include a smooth scroll feature for our brand/logo to top of page

js/smoothscroll.js

add at bottom

```
$("a.navbar-brand[href^='#']").on('click', function(e) {
   // prevent default anchor click behavior
   e.preventDefault();
   // store hash
   var hash = this.hash;
   // animate
   $('html, body').animate({
        scrollTop: $(this.hash).offset().top
    }, 300, function(){
        // when done, add hash to url
        // (default click behaviour)
        window.location.hash = hash;
    });
```

```
});
```

5. Include the proper id linkage between our .navbar-brand link to the top of jumbotron in the index file

index.html

change

```
<a class="navbar-brand" href="#"><img src="images/top_logo.png" alt="One Million Lines" title="One Million Lines"></a>
```

to

```
<a class="navbar-brand" href="#jumbotron"><img src="images/top_logo.png" alt="One Million Lines" title="One Million Lines"></a>
```

and change

```
<div class="jumbotron">
```

to

```
<div id=”jumbotron” class="jumbotron">
```

6. Include an offset

js/smoothscroll.js

for both functions change

```
scrollTop: $(this.hash).offset().top
```

to

```
scrollTop: $(this.hash).offset().top -50
```

Lesson 35: Fixed Positioning

Lets explore a new type of positioning.

1. Add the chevron icon to our html

index.html

```
<div id="back-to-top">

      <i class="fa fa-chevron-circle-up fa-3x"></i>
```

```
 </div>
```

2. Add the CSS for the icon

css/custom.css

```
#back-to-top {
      position: fixed;
      bottom: 20px;
      right: 20px;
}
```

3. Add the link to the icon

index.html

change

```
<div id="back-to-top">
      <i class="fa fa-chevron-circle-up fa-3x"></i>
</div>
```

to

```
<div id="back-to-top">
   <a href="#jumbotron"><i class="fa fa-chevron-circle-up fa-3x"></i></a>
</div>
```

4. Fix the styling for the link

css/custom.css

add

```
#back-to-top a {
      color: #333;
}
```

5. Add the smooth scrolling to our back-to-top icon

js/smoothscroll.js

add

```
$("#back-to-top a[href^='#']").on('click', function(e) {
   // prevent default anchor click behavior
   e.preventDefault();
   // store hash
   var hash = this.hash;
   // animate
   $('html, body').animate({
       scrollTop: $(this.hash).offset().top -50
     }, 700, function(){
       // when done, add hash to url
       // (default click behaviour)
       window.location.hash = hash;
     });
});
```

Lesson Notes

- With fixed positioning the element's position is "fixed" to the screen

---

Lesson 36: Absolute And Relative Positioning

Now we'll round out our knowledge of types of positioning with absolute and relative positioning.

1. Move the icon inside the footer

index.html

move

```
<div id="back-to-top">
```

```
      <a href="#jumbotron"><i class="fa fa-chevron-circle-up fa-3x"></i></a>
</div>
```

to be in footer

```
    <footer>
      <div class="container">
        <p><img src="images/one_million_bottom_brand.png"></p>
      </div>
      <div id="back-to-top">
        <a href="#jumbotron"><i class="fa fa-chevron-circle-up fa-3x"></i></a>
      </div>
    </footer>
```

2. Fix the CSS positioning for the footer and the icon

css/custom.css

```
footer {
      background-color: #f74c3c;
      padding-top: 150px;
      padding-bottom: 30px;
      position: relative;
}
```

and

```
#back-to-top {
      position: absolute;
      bottom: 20px;
      right: 20px;
}
```

Lesson Notes:

- Notice how putting the “back-to-top” div within the footer did not affect any of the other elements. This is because an element with an absolute position is outside the normal flow of the page and other elements act like the absolute positioned element does not exist. A better way to understand this would be to think of your website as a piece of paper and the absolute positioned element is another cut out piece of paper that is laid on top. This element does not affect the original paper but is simply laid on top.
- To position an element relative to another you must put it inside a parent element, give the parent element relative positioning and then give the element you want to position absolute positioning.
- By giving a parent element relative positioning you are able to position absolutely positioned children elements inside of it.
- An absolute positioned element looks for the first ancestor (parent, grandparent, etc.) element that has a positioning other than static - it doesn’t have to be relative, could be fixed, but it is **highly common** to use relative - and will be positioned relative to itself. If there is no ancestor element with a position other than static the element will be positioned relative to the <html> - which is the initial viewing block. You will see this come in play in later in the course when we do a sticky footer.
- Having trouble understanding absolute and relative positioning or just want to hear about it in a different way? No reason for us to reinvent the wheel so click here to see how CSS Tricks explains it. Chris does a great job with his visual that shows you. **Note:** in Chris’s second drawing the far right block should actually be 30% from the top of the screen and only a little over half of the box should be showing as it is should be positioned right -25px of the screen, not the box. **Second note:** in his page he says “are positioning themselves in relation to the body element instead of…”. “the body element” should be replaced by “initial viewing block”. Once again you will see this come in play with the sticky footer.
- I know this is a lot to take in right now. If it seems like a lot don’t worry it will make sense when you see it in action with the sticky footer.

Lesson 37: Modal Magic

Bring in modals to add interactivity and give your user more information on a topic.

1. Link our “Learn more” button to a modal

index.html

change

```
<p><a class="btn btn-danger btn-lg" role="button">Learn more &raquo;</a></p>
```

to

```
<p><a class="btn btn-danger btn-lg" role="button" data-toggle="modal"
data-target="#myModal">Learn more &raquo;</a></p>
```

2. Include the modal

index.html

add

```
<div class="modal fade" id="myModal" tabindex="-1" role="dialog"
aria-labelledby="myModalLabel" aria-hidden="true">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <button type="button" class="close" data-dismiss="modal"
aria-hidden="true">&times;</button>
        <h4 class="modal-title" id="myModalLabel">Modal title</h4>
      </div>
      <div class="modal-body">
        ...
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-default"
data-dismiss="modal">Close</button>
        <button type="button" class="btn btn-primary">Save changes</button>
      </div>
    </div>
  </div>
</div>
```

Lesson 38: iframe, you frame, we all frame

Let's learn about iframes by bringing in a youTube video

1. Edit the modal so it only have the iframe in it

index.html

```
<!-- Modal -->

        <div class="modal fade" id="myModal" tabindex="-1" role="dialog"
aria-labelledby="myModalLabel" aria-hidden="true">

          <div class="modal-dialog">

            <div class="modal-content">

              <iframe width="560" height="315"
src="//www.youtube.com/embed/29714SBp098?showinfo=0" frameborder="0"
allowfullscreen></iframe>

            </div>

          </div>

        </div>
```

2. Style the iframe to give it a little spacing

css/custom.css

```
.modal-content iframe {

      margin-top: 15px;

      margin-bottom: 10px;

}
```

3. Include the "x-button"

index.html

add

```
<button type="button" class="close btn btn-danger" data-dismiss="modal"
aria-hidden="true">x</button>
```

4. Style the button correctly

css/custom.css

add

```
.modal-content button.close {
    position: absolute;
    top: 5px;
    right: 5px;
    background-color: white;
    font-size: 28px;
    width: 30px;
    opacity: 1;
    border-radius: 24px;
    -webkit-border-radius: 24px;
    -moz-border-radius: 24px;
}
```

Lesson Notes

- Let's talk about positioning. When you give something absolute positioning it is no longer in the flow of the page. So think of writing in Microsoft Word. You write words and include pictures and you don't want to overlap the two. If there is a picture on the page the words go around it. So the images affect the flow of the page. But however have you ever included an image and chosen the "image is on top of text" feature? Notice how the text then flows as if the image isn't even on the page and the image and text can overlap each other. It would be the same as printing out the document and then placing another cutout piece of paper on top of it. This is how absolute positioning and fixed positioning works. When you give an element one of these two types of positioning they no longer affect the flow of the page - its as if they are another layer laid on top of the page. You can see this here around the 7:05 mark in the video as you see the iframe move when we change the x button to have an absolute positioning. The button immediately gets placed on top of the frame. Just like a cutout piece of paper on top of another.
- border-radius curves the border of an html element

Lesson 39: YouTube Videos

A quick intro into youTube player parameters to give our videos a site-customized look

1. Fix the iframe so that the youTube video info doesn't show

index.html

change

```
<iframe width="560" height="315" src="//www.youtube.com/embed/29714SBp098" frameborder="0" allowfullscreen></iframe>
```

to

```
<iframe width="560" height="315" src="//www.youtube.com/embed/29714SBp098?showinfo=0" frameborder="0" allowfullscreen></iframe>
```

Lesson 40: Centering Absolute Elements

Move into the ranks of website ninjas as you learn how to center absolute elements. This is a crazy cool power to have.

1. Change the .modal-dialog styling to center it in the middle of the page

css/custom.css

```
.modal-dialog {
	position: absolute;
	left: 50%;
	margin-left: -300px;
	top: 50%;
	margin-top: -173px;
}
```

Lesson 41: Z-Index

Let's learn about stacking with the z-index.

Lesson Notes:

- Z-index is the stacking order of elements. Z-index only applies to elements that are either absolute, fixed, or relative positioning.
- To understand z-index think of a graph. The x-axis is left to right, the y-index is up and down, and the z-index would be depth or top to bottom. Think of 2 pieces of paper - one on top of the other. You could say the bottom piece of paper would have a z-index of 1 and the one on top would have a z-index of 2.
- Z-index ranges from -9999 to 9999. A higher number z-index is on top of a lower number z-index element.
- So if you ever can't find an element but you know it should be showing check the z-index. Maybe it has a lower z-index than element it should be showing on top of.

BONUS Lesson 42: Making Money with Gumroad

Want to test a business idea or sell products on your site? Gumroad is your answer

1. Go to gumroad, create an account, and create a product.

If you have any questions go here.

2. Put the gumroad link in your site.

index.html

```
<p><a class="btn btn-danger btn-lg" role="button"
href="https://gumroad.com/l/BYFWI1W">Get the course today &raquo;</a></p>
```

BONUS Lesson 43: Grabbing eMail Addresses With Mailchimp

A quick, free way to grab email addresses. Mailchimp will be your friend

1. Sign up for a free Mailchimp Account. Click Here.

2. Create a list

Lesson 44: All About Forms

Forms allow users to interact with your site. Lets learn how to include them in yours

Lesson Notes

- The <input> is the most important form element. Use the type attribute to define the type of input field. The most common types are text, email, password, radio butons, checkboxes, and submit buttons. Another common input is the <textarea> form element - this is what you type your facebook status update in.
- Within the form tag you will always have an action and method attribute. The action attribute tells the form what page to send the data to when the submit button is clicked. The method attribute tells the form how to send the data. The two options are “get” and “post”. For now don’t worry about the difference between the two. When dealing with forms at this point you will most likely be embedding them like we are in this lesson and the action attribute will already be filled out for you. Once you start learning how to create web applications and create your own forms you will at that point learn about get vs post more.
- The name attribute tells the form what attribute that input field is connected to. In other words the name attribute of our email input was “EMAIL”. When our mailchimp form is submitted the form updates that entry’s “EMAIL” attribute to have the value of whatever was in that input field.
- With radio buttons and checkboxes the value attribute is what gets assigned to the entry for that name depending on what the user clicks on. So the value is a predetermined value depending on what they click on versus a user-determined value of whatever they want to type with email and text inputs.
- The label tag is just assigns a label to an input element. The only added benefit that the label tag brings over simply putting text identifying the input before the input tag is that when the label element’s text is clicked on it automatically highlights the input element it is connected to and allows the user to start typing in that input field.

Lesson 45: Styling Forms

Now that we know all about the important parts of forms lets style them to look good on our site

1. Bring the mailchimp form into our index.html page and make it so that the email input field is ¾ the width of the container and the submit button is ¼ width

index.html

```
<!-- Begin MailChimp Signup Form -->
```

```
        <div id="mc_embed_signup" class="row">
          <div class="col-sm-9">
            <form
action="http://massiveacademy.us8.list-manage1.com/subscribe/post?u=636e49cc7c70ceee9
39312b50&amp;id=fe7a597b01" method="post" id="mc-embedded-subscribe-form"
name="mc-embedded-subscribe-form" class="validate" target="_blank" role="form"
novalidate>


            <div class="mc-field-group form-group">
              <input type="email" value="" name="EMAIL" class="required email
form-control" id="mce-EMAIL" placeholder="Enter email">
            </div>
          </div>
          <div class="col-sm-3">
            <div id="mce-responses" class="clear">
              <div class="response" id="mce-error-response"
style="display:none"></div>
              <div class="response" id="mce-success-response"
style="display:none"></div>
            </div>    <!-- real people should not fill this in and expect good things
- do not remove this or risk form bot signups-->
            <div style="position: absolute; left: -5000px;"><input type="text"
name="b_636e49cc7c70ceee939312b50_fe7a597b01" tabindex="-1" value=""></div>
            <div class="clear"><input type="submit" value="Join The Fun"
name="subscribe" id="mc-embedded-subscribe" class="btn btn-danger btn-lg"></div>
          </div>
          </form>
        </div>
        <!--End mc_embed_signup-->
```

2. Style the email input via form-control so that it is the same height and font-size as the button

```
.jumbotron .form-control {
      height: 69px;
      font-size: 36px;
}
```

If you wanted to be more specific with the selector you could change it to:

```
.jumbotron input[type="email"].form-control  {
      height: 69px;
      font-size: 36px;
}
```

All this is saying that for any input tag with type=”email” and a class of form-control in an html element (in our case div) will have the following styling. Its good to be specific, but since in this case the jumbotron div only has one field with the form-control class the less-specific selector we used in the video is fine as well.

Lesson Notes

- Class “form-group” just gives sufficient margin-bottom so the spacing looks good.
- Class “form-control” is what styles the input and gives it 100% width - of the parent element that is.
- The placeholder attribute is what shows in the input field until the user starts typing.

Lesson 46: Building A Blog Template

We’ll continue to round out our knowledge of HTML and CSS while we build a blog template

You know the drill when getting started...

1. Download bootstrap

2. Grab the Bootstrap blog template html and save it into an index.html making sure to fix the links in the <head> section

3. Grab the blog’s CSS and create and save a blog.css file in our site’s css folder

---

Lesson 47: Setting Up The Blog

We'll get some common styling set up with the body's font-family and bring in a navbar

1. Delete the div with class of "blog-header" in our index.html file

2. Wrap the `<div class="container">` with a `<div id="blog-page">`

3. Fix the linking to the Bootstrap javascript

index.html

change

```
<script src="../../dist/js/bootstrap.min.js"></script>
```

```
<script src'../../assets/js/docs.min.js"></script>
```

to

```
<script src="js/bootstrap.min.js"></script>
```

4. Give your blog page some padding at the top

css/blog.css

```
/* Blog page */
```

```
#blog-page {
```

```
  padding-top: 30px;
```

```
}
```

5. Include Open Sans and fix the body's font-family styling

index.html

add in the <head> section

```
<!-- Include Open Sans from Google Fonts -->
```

```
    <link
href='https://fonts.googleapis.com/css?family=Open+Sans:300italic,400italic,600italic
,700italic,800italic,400,300,600,700,800' rel='stylesheet' type='text/css'>
```

css/blog.css

change

```
body {
  font-family: Georgia, “Times New Roman”, Times, serif;
  color: #555;
}
```

to

```
body {
  font-family: 'Open Sans', "Helvetica Neue", Helvetica, Arial, sans-serif;
  color: #555;
  font-weight: 300;
}
```

6. Bring in the static navbar

index.html

add

```
<!-- Static navbar -->
    <div class="navbar navbar-default navbar-static-top" role="navigation">
      <div class="container">
        <div class="navbar-header">
          <button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">
            <span class="sr-only">Toggle navigation</span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
          </button>
          <a class="navbar-brand" href="#">Project name</a>
```

```
        </div>
        <div class="navbar-collapse collapse">
          <ul class="nav navbar-nav">
            <li class="active"><a href="#">Home</a></li>
            <li><a href="#about">About</a></li>
            <li><a href="#contact">Contact</a></li>
            <li class="dropdown">
              <a href="#" class="dropdown-toggle" data-toggle="dropdown">Dropdown <b
class="caret"></b></a>
              <ul class="dropdown-menu">
                <li><a href="#">Action</a></li>
                <li><a href="#">Another action</a></li>
                <li><a href="#">Something else here</a></li>
                <li class="divider"></li>
                <li class="dropdown-header">Nav header</li>
                <li><a href="#">Separated link</a></li>
                <li><a href="#">One more separated link</a></li>
              </ul>
            </li>
          </ul>
          <ul class="nav navbar-nav navbar-right">
            <li><a href="../navbar/">Default</a></li>
            <li class="active"><a href="./">Static top</a></li>
            <li><a href="../navbar-fixed-top/">Fixed top</a></li>
          </ul>
        </div><!--/.nav-collapse -->
```

```
      </div>

    </div>
```

Lesson Notes

- A word about the “font-family” attribute. When giving multiple options to font-family as we’ve done in this lesson what you’re doing is giving a list of different fonts to fall back on if the current font is not loaded properly or accessible by the browser. In other words, from our example, if we didn’t include ‘Open Sans’ correctly or if the browser for some reason had trouble accessing it from google it would then try and display the font as ‘Helvetica Neue’. If for some reason that didn’t display properly it would then try to display Helvetica. So on and so forth.

Lesson 48: Styling Fonts With CSS

With all the possibilities of text and font styling in CSS you can create your own unique logos and brands directly in the browser.

1. Delete the entire `<ul class=”nav navbar-nav”>`

2. Change the background color of the active state

css/blog.css

```
.navbar-default .navbar-nav>.active>a, .navbar-default .navbar-nav>.active>a:hover,
.navbar-default .navbar-nav>.active>a:focus {

  background-color: #f8f8f8;

}
```

3. Put in your own navbar-brand for your page’s header title

index.html

change

```
<a class="navbar-brand" href="#">Project name</a>
```

to

```
<a class="navbar-brand" href="#"><strong>Ryan</strong>Bonhardt</a>
```

4. Include the 'Lato' Google Font

index.html

add

```
<link
href='http://fonts.googleapis.com/css?family=Lato:100,300,400,700,900,100italic,300it
alic,400italic,700italic,900italic' rel='stylesheet' type='text/css'>
```

5. Style your .navbar-brand

css/blog.css

add

```
.navbar-brand {
  font-family: 'Lato', "Helvetica Neue", Helvetica, Arial, sans-serif;
  font-size: 1.9em;
  text-transform: lowercase;
  font-weight: 100;
  letter-spacing: -1px;
}
```

Lesson Notes

- Word wrap in Sublime just wraps the words to the next line within the editor so that they are all in view and you don't have to scroll to the right, out of sight, to see what's on the line. As you see in the video when I selected word wrap the long selector I was using in blog.css was now visible and just wrapped to multiple lines.
- I included all font-weights and styles for both the 'Lato' and 'Open Sans' fonts because I wanted to show you how to do it. However, when you go to production with your site and start hosting it online I suggest you only include the font-weights and styles you use in your site. This is because the more fonts that you page must download each time the longer it takes for you page to load. This delay will became more obvious once you start getting a lot of traffic to your site.
- Letter spacing is just the spacing between letters. The higher the number the greater the spacing. For you typography and design enthusiasts this is the same as "kerning".

- As you can tell from the video I like to play with the CSS within the browser to see which design values I like best. I think its important to find a sense of play when you're working and designing. Try new things, and experiment. A lot of times you'll be surprised at what comes of it.

Lesson 49: Child Pseudo-Elements

Lets "google"-ize our blog's look through child pseudo-elements

1. Switch up the bold in our navbar-brand

index.html

change

```
<a class="navbar-brand" href="#"><strong>Ryan</strong>Bonhardt</a>
```

to

```
<a class="navbar-brand" href="#">Ryan<strong>Bonhardt</strong></a>
```

2. Change the font color of our navbar-brand

css/blog.css

add

```
.navbar-default .navbar-brand {
  color: #555;
}
```

3. Add a fourth list item

index.html

add

```
<li><a href="../navbar-fixed-top/">Fourth li</a></li>
```

4. Change list items' color

css/blog.css

add

```
.navbar-default .navbar-brand>li>a {
```

```
  color: #555;
}
```

5. Color each list item

css/blog.css

change

```
.navbar-default .navbar-brand>li>a {
  color: #555;
}
```

to

```
.navbar-default .navbar-nav>li:first-child>a {
  color: #e74c3c;
}
.navbar-default .navbar-nav>li:nth-child(2)>a {
  color: #2ecc71;
}
.navbar-default .navbar-nav>li:nth-child(3)>a {
  color: #3498db;
}
.navbar-default .navbar-nav>li:nth-child(4)>a {
  color: #f39c12;
}
```

Lesson Notes

- As discussed in the video when you use “>” in a CSS rule it signifies a parent to child HTML element relationship. In other words the selector `ul.navbar-brand>li>a` will only select links that are immediate child elements of list items that are immediate child elements of an unordered list with a class of “.navbar-brand”. So when we wrapped the “a” tag in a paragraph tag that “a” tag was no longer an

immediate child element of the list item and thus was not styled the same as the other “a” tags in the list. However if you remove the “>” (greater than carrots) and replace them with spaces the rule is selecting all “a” tags that are inside list items that are inside an unordered list with a class of .navbar-brand.

- Put the child pseudo-selector directly after the HTML selector that has multiple child elements per parent element. In other words in our code (aka the index.html document) there are four list items that we want to style differently within the unordered list. So we’re putting the child pseudo-selector after each list item. We are not putting them after each “a” tag because there is only one “a” tag per list item. Thus each “a” tag would be the first child.
- So to keep hitting this home lets talk about when we style each blog-post’s header differently in the video’s example. The question to ask yourself is which is there multiple child elements of per parent element? In this case there is only one h2 with a class of “blog-post-title” so we wouldn’t put the child pseudo-selector after “h2.blog-post-title”. However there are multiple divs with a class of “blog-post” within the div with an id “blog-page”. In other words there are multiple blog posts within this blog page. Thus, we put the child pseudo-selector attached to “.blog-page”. So it’s not what you’re styling that you attach the child pseudo-selector to, but what has multiple child instances within a parent element.

Lesson 50: More Pseudo Selectors

Lets get specific and creative with our styling via pseudo-selectors

1. Delete the entire `<div class=”blog-masthead”>` the `.blog-masthead {...}` CSS rule

2. Give the nav list items a greater font-weight

css/blog.css

add

```
.navbar-default .navbar-nav>li>a {
  font-weight: 700;
}
```

3. Give our list items a border at the bottom when hovered over and when it has a class of active and fix the padding

css/blog.css

add

```
.navbar-default .navbar-nav>li>a:hover, .navbar-default .navbar-nav>li.active>a {
  border-bottom: 3px solid;
  padding-bottom: 12px;
}
```

4. Style the list items with the correct color when active, hovered, and/or focused

css/blog.css

add `.navbar-default .navbar-nav>li:first-child>a:hover, .navbar-default .navbar-nav>li:first-child>a:focus` to each list item's selector as shown in video

Lesson Notes

- Remember that when you attach the pseudo-selector :hover to an html selector the styling rules take place when that element is hovered over with the mouse. With focus its whenever the element is tabbed over in a browser as shown in the video.

Lesson 51: Styling Our Blog Posts

We'll revisit many CSS and HTML principles as we style out our blog posts.

1. Delete the entire blog name and description section within our blog.css file

2. Style our site's 'h2's

css/blog.css

add

```
h2 {
  font-family: 'Open Sans', "Helvetica Neue", Helvetica, Arial, sans-serif;
  font-size: 21px;
  line-height:  1.2em;
  font-weight: 400;
  color: #555;
}
```

3. Style our blog post titles

css/blog.css

change

```
.blog-post-title {
  margin-bottom: 5px;
  font-size: 40px;
}
```

to

```
.blog-post-title {
  margin-bottom: 5px;
  font-size: 30px;
  font-weight: 700;
  text-transform: capitalize;
}
```

4. Style blog post meta

css/blog.css

change

```
.blog-post-meta {
  margin-bottom: 20px;
  color: #999;
}
```

to

```
.blog-post-meta {
  margin-bottom: 20px;
  color: #888;
  letter-spacing: .04em;
```

```
  text-transform: uppercase;
  font-size: 14px;
  line-height: 18px;
}
```

5. Add a comments indicator

index.html

add within the “blog-post-meta” paragraph

```
<a type="button" class="btn btn-danger" href="blog-post-link">55 Comments</a>
```

6. Style the comments indicator button

css/blog.css

add

```
.blog-post-meta .btn-danger {
  background-color: #e74c3c;
  padding: 3px 5px;
  margin-right: 5px;
  text-transform: uppercase;
  font-weight: 300;
}
```

7. Add a read more link

index.html

add to the end of the blog post content

```
... <a href="blog-post-link">Read More</a>
```

8. Style blog posts

css/blog.css

change

```
.blog-post {
```

```
  margin-bottom: 60px;
}
```

to

```
.blog-post {
  margin-bottom: 40px;
  border-bottom: 1px solid #eee;
  padding-bottom: 20px;
}
```

9. Link blog post titles to the blog post pages themselves

index.html

change

```
<h2 class="blog-post-title">Sample blog post</h2>
```

to

```
<h2 class="blog-post-title"><a href="blog-post-link">Sample blog post</a></h2>
```

10. Style the blog post title

css/blog.css

add

```
.blog-post-title a {
  color: #555;
}
.blog-post-title a:hover {
  color: #2a6496;
}
```

11. Add in a featured image to your blog post

index.html

add below the meta section and above the post content

```
<img src="images/udemy-banner.png"></img>
```

12. Style the image so that it doesn't go outside the blog column

css/blog.css

add

```
.blog-main img {
  max-width: 100%;
  margin-bottom: 30px;
}
```

Lesson Notes

- In this video we chose to style the images under ".blog-main". We could have done them under ".blog-post" as well with the same effect. The only reason I chose the first way was because if you decided maybe to include an advertisement for one of your products between two of your blog posts and you included an image or banner image in that advertisement it would still be within the ".blog-main" div, but would be outside one of your ".blog-post" divs. And thus the styling would still apply because that image would be within the ".blog-main" div.

Lesson 52: Subscribe Box

Its important to have a way for your readers to subscribe to your blog. Let's revisit forms and create a way.

1. Fix the width of the blog-sidebar

index.html

change

```
<div class="col-sm-3 col-sm-offset-1 blog-sidebar">
```

to

```
<div class="col-sm-4 blog-sidebar">
```

css/blog.css

add

```
.blog-sidebar {
  padding-left: 30px;
}
```

2. Change the styling of the .sidebar-module-inset

css/blog.css

change

```
.sidebar-module-inset {
  padding: 15px;
  background-color: #f5f5f5;
  border-radius: 4px;
}
```

to

```
.sidebar-module-inset {
  padding: 15px;
  background-color: #f5f5f5;
  border-radius: 4px;
  border: 5px dashed #2ecc71;
}
```

3. Update subscribe box content

index.html

change

```
<div class="sidebar-module sidebar-module-inset">
<h4>About</h4>
<p>Etiam porta <em>sem malesuada magna</em> mollis euismod. Cras mattis consectetur purus sit amet fermentum. Aenean lacinia bibendum nulla sed consectetur.</p>
</div>
```

to

```
<div id="subscribe-box" class="sidebar-module sidebar-module-inset">
<h4>Like what you're reading? Go VIP.</h4>
<p>Join today to receive instant updates, free learning resources, and a bunch of other awesome stuff. Oh by the way, I hate spam so don't worry about receiving any of that from me.</p>
</div>
```

4. Change the font and fix the styling so that the h4 is all on one line

index.html

add

```
<link href='http://fonts.googleapis.com/css?family=Roboto+Condensed' rel='stylesheet' type='text/css'>
```

css/blog.css

add

```
#subscribe-box h4 {
  font-family: 'Roboto Condensed';
  font-size: 20px;
  color: #666;
  text-align: justify;
}
```

5. Style the subscribe box

css/blog.css

add

```
#subscribe-box {
  padding: 15px 20px;
  font-size: 15px;
}
```

```
#subscribe-box p {
  line-height: 1.5em;
  text-rendering: optimizeLegibility;
}
```

6. Add the form to our subscribe box and style it

index.html

add

```
<form role="form">
   <div class="form-group">
     <input type="email" class="form-control" id="exampleInputEmail"
placeholder="Enter email">
   </div>
     <p class=“center”><a type="submit" class="btn btn-warning">Join For Free</a></p>
</form>
```

css/blog.css

add

```
#subscribe-box .form-group {
  margin-top: 15px;
}

#subscribe-box .btn-warning {
  background-color: #f39c12;
  text-transform: uppercase;
  margin-top: 10px;
  margin-bottom: 10px;
}
```

```
#subscribe-box .btn-warning:hover {
  background-color: #f0ad4e;
}
```

and

```
.center {
  text-align: center;
}
```

7. Fix styling of comments button when it is hovered over

css/blog.css

add

```
.blog-post-meta .btn-danger:hover {
  background-color: #d2322d;
}
```

Lesson 53: Mailchimp Revisited

Take a walk down Mailchimp memory lane and give your subscribe form functionality

1. Login to Mailchimp and create a list

2. Grab your list's naked subscribe form and embed in your site

NOTE: Don't copy exactly from below. The action is linked to my blog site list. If you do and people enter their email address it will be added my email list on mailchimp.

index.html

change

```
<form role="form">
   <div class="form-group">
```

```
    <input type="email" class="form-control" id="exampleInputEmail"
placeholder="Enter email">
  </div>
    <p class="center"><a type="submit" class="btn btn-warning">Join For Free</a></p>
</form>
```

to

```
<form role="form"
action="http://massiveacademy.us8.list-manage.com/subscribe/post?u=636e49cc7c70ceee93
9312b50&amp;id=14d21b31a8" method="post" id="mc-embedded-subscribe-form"
name="mc-embedded-subscribe-form" class="validate" target="_blank" novalidate>
  <div class="form-group">
    <input type="email" value="" name="EMAIL" class="form-control" id="mce-EMAIL"
placeholder="Enter email">
  </div>
  <div id="mce-responses" class="clear">
    <div class="response" id="mce-error-response" style="display:none"></div>
    <div class="response" id="mce-success-response" style="display:none"></div>
  </div>    <!-- real people should not fill this in and expect good things - do not
remove this or risk form bot signups-->
  <div style="position: absolute; left: -5000px;"><input type="text"
name="b_636e49cc7c70ceee939312b50_14d21b31a8" tabindex="-1" value=""></div>
  <p class="center"><input type="submit" value="Join For Free" name="subscribe"
class="btn btn-warning"></p>
</form>
```

BONUS Lesson 54: Advanced Mailchimp Shenanigans

In this lesson we learn advanced tactics to keep users on our site as they sign up for our email list.

1. Move your site files over to dropbox and fix the google font includes

index.html

change

```
<!-- Include Google Fonts -->
```

```
    <link
href='http://fonts.googleapis.com/css?family=Open+Sans:300italic,400italic,600italic,
700italic,800italic,400,300,600,700,800' rel='stylesheet' type='text/css'>
```

```
    <link
href='http://fonts.googleapis.com/css?family=Lato:100,300,400,700,900,100italic,300it
alic,400italic,700italic,900italic' rel='stylesheet' type='text/css'>
```

```
    <link href='http://fonts.googleapis.com/css?family=Roboto+Condensed'
rel='stylesheet' type='text/css'>
```

to

```
<!-- Include Google Fonts -->
```

```
    <link
href='https://fonts.googleapis.com/css?family=Open+Sans:300italic,400italic,600italic
,700italic,800italic,400,300,600,700,800' rel='stylesheet' type='text/css'>
```

```
    <link
href='https://fonts.googleapis.com/css?family=Lato:100,300,400,700,900,100italic,300i
talic,400italic,700italic,900italic' rel='stylesheet' type='text/css'>
```

```
    <link href='https://fonts.googleapis.com/css?family=Roboto+Condensed'
rel='stylesheet' type='text/css'>
```

2. Create a new subscribe-confirm.html file and add content within the container

subscribe-confirm.html

```
<h2>Thank you for subscribing to our newsletter</h2>
```

```
<p>Before we can start sending our awesome content we have to make sure you are
human, and not a funny robot. A confirmation email has been sent to your inbox.
Please open it and confirm your email address to start receiving updates.</p>
```

3. Add your subscribe-confirm page URL in the Mailchimp “Signup thank you page” area that says “Instead of showing this thank you page, send subscribers to another URL”. Then click save

4. Create a new subscribe-success.html file and add content within the container

subscribe-success.html

```
<h2>Thanks for confirming your email address and welcome to the club</h2>

<p>For joining here are some goodies. Enjoy!</p>
```

5. Now go to the “Confirmation thank you page” in Mailchimp and add your subscribe-success page URL in the “Instead of showing this thank you page, send subscribers to another URL” field. Save it.

Lesson 55: Styling Lists

Let’s get into some list-styling skills

1. Quick fix on our subscribe-box border

css/blog.css

change

```
.sidebar-module-inset {
  padding: 15px;
  background-color: #f5f5f5;
  border-radius: 4px;
  border: 5px dashed #2ecc71;
}
```

to

```
.sidebar-module-inset {
  padding: 15px;
  background-color: #f5f5f5;
  border-radius: 4px;
  border: 5px dashed #e74c3c;
```

```
}
```

2. Delete the “elsewhere” sidebar module

3. Redo the archive

index.html

```
<div id="posts-box" class="sidebar-module">
  <h4>Most Popular Posts</h4>
  <ol>
    <li><a href="#">This is the most popular post</a></li>
    <li><a href="#">This is the second most popular post</a></li>
    <li><a href="#">This posts was many people's favorites</a></li>
    <li><a href="#">This is just a longer title to make sure it goes to second line</a></li>
  </ol>
</div>
```

4. Style the posts-box

css/blog.css

add

```
#posts-box {
  padding: 0;
  border: 3px solid #efefef;
  border-top: 0;
  border-radius: 3px;
  margin-top: 30px;
}
#posts-box h4 {
  font-size: 16px;
```

```
  text-transform: uppercase;
  background-color: #efefef;
  color: #555;
  padding: 12px;
  text-align: center;
  font-weight: 600;
  font-family: 'Open Sans';
  letter-spacing: 2px;
  margin: 0;
}
#posts-box ol {
  list-style: none;
  padding-left: 0;
  margin-bottom: 0;
}
#posts-box li {
  font-size: 16px;
  padding: 12px;
  border-bottom: 3px solid #efefef;
  min-height: 60px;
  line-height: 20px;
}
#posts-box li:last-child {
  border-bottom: 0;
}
#posts-box a {
```

```
  color: #555;
}
```

Lesson 56: Revisit Responsiveness And Edit Footer

Remember responsive design? Lets do a quick revisit as well as touch up our footer.

1. Change the footer section to whatever you want. I'm keeping it simple

index.html

```
<div class="blog-footer">
  <p>&#169; 2014 Ryan Bonhardt. Have any questions or just want to drop me a line? Email me at <a href="mailto:rbonhardt@gmail.com" target="_blank">rbonhardt@gmail.com</a></p>
</div>
```

2. Add some responsive design if you'd like. I've started you off below

css/blog.css

add

```
@media (max-width: 991px) {
  #subscribe-box h4 {
    text-align: center;
  }
}
```

BONUS Lesson 57: Jedi Tools For Increasing Signups and Shares

Let's visit our sumo friends to get some amazing tools to significantly increase our site's newsletter signups and social shares. Oila, Magic!

1. Add the SumoMe List Builder script to your site in the header section

index.html

```
<!-- SumoMe App Include -->
```

```
<script src="//load.sumome.com/"
data-sumo-site-id="036ff13b43694e9eaabea6e276c9bf0bf0c4e9a6b762a0daed431d37ae84464c"
async></script>
```

2. Signup and create an account

3. Go to the Sumo Store and add in the List Builder by clicking on the FREE button next to List Builder

4. Click on Settings, edit the appearance to your liking and then make sure to connect your Mailchimp Account by clicking on Email Services, clicking on Mailchimp, and then click the connect button, login to your Mailchimp account, select the list you want to connect and Oila! Save and you're good to go.

5. If you want to go back into sumo store and add Highlighter.

With Highlighter if anyone highlights text on your page it makes it instantly easy for them to tweet it or share it on Facebook.

Lesson 58: Share Buttons And Vertical Alignment

Let's add some share buttons to our posts and learn about how to vertical-align inline elements.

1. Link the the first blog posts to a blog post page template

index.html

```
<h2 class="blog-post-title"><a href="posts/first-post.html">Sample blog post</a></h2>
```

2. Create a blog post page template

Copy the index.html page and paste it into a new one. Then delete the extra blog posts so that there is only one blog post on the page. Create a new folder called "posts" and save it as "first-post.html".

3. Fix all links and delete the title's link

posts/first-post.html

CSS links

```
<!-- Bootstrap core CSS -->
```

```
    <link href="../css/bootstrap.min.css" rel="stylesheet">
<!-- Custom styles for this template -->
    <link href="../css/blog.css" rel="stylesheet">
```

navbar link

```
<li><a href="../index.html">Blog</a></li>
```

and image

```
<img src="../images/udemy-banner.png"></img>
```

and eliminate blog post title linkage

```
<h2 class="blog-post-title">Sample blog post</h2>
```

4. Get on over to simplesharebuttons.com and grab the CSS and HTML for the buttons

css/blog.css

add

```
/* Blog Post Page */
#share-buttons img {
  width: 35px;
  padding: 5px;
  border: 0;
  box-shadow: 0;
  display: inline;
}
```

posts/first-post.html

add right before blog post closing div

```
<div id="share-buttons">
  <p>Share: </p>
  <!-- Facebook -->
  <a href="http://www.facebook.com/sharer.php?u=http://www.simplesharebuttons.com"
target="_blank"><img
```

```
src="http://www.simplesharebuttons.com/images/somacro/facebook.png" alt="Facebook"
/></a>

  <!-- Twitter -->

  <a href="http://twitter.com/share?url=http://www.simplesharebuttons.com&text=Simple
Share Buttons" target="_blank"><img
src="http://www.simplesharebuttons.com/images/somacro/twitter.png" alt="Twitter"
/></a>

  <!-- Google+ -->

  <a href="https://plus.google.com/share?url=http://www.simplesharebuttons.com"
target="_blank"><img
src="http://www.simplesharebuttons.com/images/somacro/google.png" alt="Google" /></a>

</div>
```

5. Fix the styling for the share section

css/blog.css

change

```
#share-buttons img {

  width: 35px;

  border: 0;

  box-shadow: 0;

  display: inline;

}
```

to

```
#share-buttons p {

  font-size: 40px;

  font-weight: 700;

}

#share-buttons img {

  width: 55px;
```

```
  border: 0;
  box-shadow: 0;
  display: inline;
  vertical-align: top;
  margin-bottom: 0;
}
```

6. Save all 3 images to our own site's images folder

This is because if for some reason the HTML Share Buttons site goes down or changes the link to the images and we're linking to them then they won't properly show on our site. However if we save the images to our site we know they'll be displaying correctly until we decide to change them.

7. Properly link to our images and our blog post's URL in the share links.

posts/first-post.html

```
<div id="share-buttons">
  <p>Share:
  <!-- Facebook -->
  <a href="http://www.facebook.com/sharer.php?u=https://dl.dropboxusercontent.com/u/40999753/blog/posts/first-post.html" target="_blank"><img src="../images/facebook.png" alt="Facebook" /></a>
  <!-- Twitter -->
  <a href="http://twitter.com/share?url=https://dl.dropboxusercontent.com/u/40999753/blog/posts/first-post.html&text=Sample Blog Post" target="_blank"><img src="../images/twitter.png" alt="Twitter" /></a>
  <!-- Google+ -->
  <a href="https://plus.google.com/share?url=https://dl.dropboxusercontent.com/u/40999753/blog/posts/first-post.html" target="_blank"><img src="../images/google.png" alt="Google" /></a></p>
```

```
</div>
```

Lesson Notes

- The only thing new in this lesson in terms of CSS and HTML is vertical-align. Vertical-align only works on images that are displayed as inline elements. In this case since we wrapped the paragraph tag around the image tags we are able to use vertical-align. However if the images were wrapped in div tags and displayed as block elements it would not work.
- Now that we got the use case of vertical-align out of the way, what do the different possible values mean? The “top” value aligns the element, in our case the images, with the tallest element on the line. Since the images are wrapped in a paragraph tag the top of the image is aligned with the top of the paragraph. The “middle” value aligns the elements with the middle of the text, in our case “share” so when you change the value to middle you can see the middle of your images aligning with the middle (vertical-wise) of the text “share”. The final value I’ll discuss is the default value “baseline”, this aligns the images’ bottoms with the baseline of the text. For a more indepth coverage of vertical align go to CSS-Tricks’ article.

Lesson 59: Meta Tags

FB Meta tags allow us to determine the look of our shares and optimize them for increased sharing and SEO.

1. Add Facebook Meta tags

post/first-post.html

```
<meta property="og:title" content="Ryan Bonhardt Blog | Sample Blog Post">

<meta property="og:type" content="article">

<meta property="og:url"
content="https://dl.dropboxusercontent.com/u/40999753/blog/posts/first-post.html">

<meta property="og:image"
content="https://dl.dropboxusercontent.com/u/40999753/blog/images/udemy-banner.png">

<meta property="og:description" content="This is a sample post of the blog template
we have been designing.">
```

Lesson 60: SEO

Learning the key basics of SEO is a very important part of learning HTML. After all, you do want people to be able to find your site online.

1. Put in the meta description

posts/first-post.html

```
<meta name="description" content="This is a sample post of the blog template we have been designing.">
```

Lesson Notes

- The SEO and FB Meta work we've done in the last couple lessons is for that "blog post" page alone. You should do this for each page of our website. Don't worry if it sounds like a lot of work. It's really not and as your site traffic grows its worth the relatively small work up front to gain organic and social traffic.

Lesson 61: Offsetting Columns

Sometimes you want some simple spacing between divs or you don't want a div to span the whole width of the page.

1. Fix the link to the blog page from the first-post page

posts/first-post.html

```
<a class="navbar-brand" href="../index.html">Ryan<strong>Bonhardt</strong></a>
```

2. Fix the link from index.html to the about page

index.html

```
<li><a href="about.html">About</a></li>
```

3. Create an about page from index.html

about.html

change

```
<ul class="nav navbar-nav navbar-right">
  <li><a href="../navbar/">Home</a></li>
```

```
  <li class="active"><a href="./">Blog</a></li>
  <li><a href="about.html">About</a></li>
  <li><a href="../navbar-fixed-top/">Fourth li</a></li>
</ul>
```

to

```
<ul class="nav navbar-nav navbar-right">
  <li><a href="../navbar/">Home</a></li>
  <li><a href="index.html">Blog</a></li>
  <li class="active"><a href="about.html">About</a></li>
  <li><a href="../navbar-fixed-top/">Fourth li</a></li>
</ul>
```

4. Delete the sidebar div and change the blog-main div as well as delete the <ul class="pager">

about.html

blog-main div should now be:

```
<div id="about" class="col-md-10 col-md-offset-1 blog-main">


</div><!-- /.blog-main -->
```

Lesson Notes

- You could delete the "blog-main" class. I'm keeping it for the already done styling of font-size, line-height, and the img styling of max-width and margin-bottom. But it wouldn't hurt to delete the class and copy and paste that styling into the CSS rules with the "about" ID.
- Also you might be saying "Hey Bootstrap is based on a 12-column grid system and above 10 + 1 only equals 11. What's up with that?" By offsetting the div by one column it gives one column of space on the left (1/12th of the container's width). Then the div is 10 columns wide (10/12th of the container's width). So by default there is one-column of spacing on the right, but there is no need to declare anything about that since it is just space sitting out there with no content in it.

Lesson 62: Let's Float

Knowing float is the key to making complex website layouts.

1. Save your image into the images folder, link to it in the about document, and then style the border radius in your CSS

about.html

```
<img src="images/ryan.jpg" alt="Ryan">
```

css/blog.css

```
#about img {
  border-radius: 100px;
}
```

2. Add in some information about yourself

about.html

```
<h1>I want to make education more effective through both material - skills that matter today - and method - project-based learning. </h1>
```

3. Float the image to the left and give it some margin for proper spacing

css/blog.css

```
#about img {
  border-radius: 100px;
  float: left;
  margin-right: 30px;
  margin-bottom: 10px;
}
```

4. Add some more info and style the h1

about.html

```
<h1>I want to make education more effective through both material - skills that matter today - and method - project-based learning. </h1>

<p id="first-paragraph">I am the founder of MASSIVE Academy and Online Director for Miles 2 Give. As you can see above I have a strong passion for improving education,
```

```
specifically online education and have a dream that one day every one will have the chance to learn how to code and build things online. </p>
```

css/blog.css

```
#about h1 {
  color: #e74c3c;
  font-family: 'Open Sans';
  font-weight: 300;
}
```

5. Clear the float for the paragraph and give it proper spacing

css/blog.css

```
#first-paragraph {
  clear: left;
  padding-top:  20px;
}
```

Lesson Notes

- Floating images allows text to flow around them just as if you were in Microsoft Word and you placed an image on the left or right of the page and chose for the text to wrap (another word flow) around it.
- Floating allows you to display multiple block elements on the same line
- Floated elements don't act the same in terms of flow of the page and the box model as static positioned elements do. If you noticed when we gave the floated image certain margins it acted as we'd expect static positioned elements would having a margin-right of 30px and margin-bottom of 10px. The image had the proper spacing to the right and left as expected. However, once we cleared the paragraph and then tried to give it a margin-top it was as if the floated image was not in the normal flow of the page as you saw the margin-top just go right through the image and ignore it until the margin-top got so great in pixels that it hit the static-positioned h1 and then started pushing down as you'd expect it. So the takeaway is that when assigning margin to a floated element it acts as you've seen before. However when assigning margin to a static positioned element and it comes in contact with a float it will act as if the floated element is not even there, or for lack of a better term floating, not a part of the normal flow of the page. This is why we use padding in the video to give the paragraph proper spacing, since padding is within the border of the paragraph itself.

- A cleared object will not continue to wrap itself around the floated element, but instead will clear the float and move itself below the floated element. This is easier understood when seen in the video.
- While we will go deeper into floats in later lessons, for an even deeper dive into floats right now I suggest going over to CSS Tricks - an amazingly in-depth article for floats. No need for me to re-invent the wheel as Chris has already done an incredible job describing floats in full. For now just check it out and don't worry if you don't understand everything right now. We will talk more about floats and clear things up for you in the next lessons.

Lesson 63: Sticky Footer

Is your page content short and your footer half way up the screen? Let's fix that with a sticky footer.

1. Give our footer styling so that it stays at the bottom of our page

css/blog.css

```
.sticky-footer {
  position: absolute;
  width: 100%
  bottom: 0;
}
```

about.html

```
<div class="blog-footer sticky-footer">
```

Lesson Notes:

- We talked about this before in the lesson when we were first introduced to absolute and fixed positioning. If an absolute positioned element doesn't have an ancestral element (parent, grandparent, etc) that has a positioning other than static it will be positioned relative to the html - which is the initial viewing block. This is why if you reload your page with the inspect element feature open the "sticky" footer stays half way up the page when you close the inspect element feature. This is also why if your content grows and goes longer than the page the "sticky" footer will stay in its original position as you scroll down.

Lesson 64: Deep Dive Into Floats And Divs

In this lesson we learn how to create our own multi-column layout.

1. Copy the about page and create a portfolio.html page

2. Fix the linkage on both for the navbar

portfolio.html

```
<ul class="nav navbar-nav navbar-right">
  <li><a href="../navbar/">Home</a></li>
  <li><a href="index.html">Blog</a></li>
  <li><a href="about.html">About</a></li>
  <li class="active"><a href="portfolio.html">Portfolio</a></li>
</ul>
```

about.html

```
<ul class="nav navbar-nav navbar-right">
  <li><a href="../navbar/">Home</a></li>
  <li><a href="index.html">Blog</a></li>
  <li class="active"><a href="about.html">About</a></li>
  <li><a href="portfolio.html">Portfolio</a></li>
</ul>
```

3. Delete the enter and replace it with:

portfolio.html

```
<div class="example">
  <p>Div 1</p>
</div>
<div class="example">
  <p>Div 2</p>
</div>
```

```
<div class="example">
  <p>Div 3</p>
</div>
```

4. Style the example divs

css/blog.css

```
.example {
  background-color: #eee;
  float: left;
  width: 33.33333%;
  border: 1px solid;
  height: 300px;
}
```

5. Add a paragraph after the divs

portfolio.html

```
<p class="clear"> this is some extra text</p>
```

6. Add a clear class to your CSS

css/blog.css

```
.clear {
  clear: both;
}
```

Lesson Notes

- When you float a div without declaring its width it is only as wide and tall as its contents - including any styled margin and/or padding. This was evident in the film when we initially floated the first div to the left. Notice how its width was only as big as the writing “Div 1” and the second div was pushed right up against it.

Lesson 65: Styling Columns Of Divs

We add a powerful skill to our repertoire of being able to style rows of columned divs and learn the “trick of clear fixes”. After this lesson you are officially a website building beast!

1. Bring in a new set of divs

portfolio.html

```
<div class="example-2">
  <div class="inner-example">
    <img src="images/udemy-banner.png">
    <h3>This is some sample text</h3>
    <p>This is our paragraph</p>
  </div>
</div>
<div class="example-2">
  <div class="inner-example">
    <img src="images/udemy-banner.png">
    <h3>This is some sample text</h3>
    <p>This is our paragraph</p>
  </div>
</div>
<div class="example-2">
  <div class="inner-example">
    <img src="images/udemy-banner.png">
    <h3>This is some sample text</h3>
    <p>This is our paragraph</p>
  </div>
</div>
```

2. Take away the sticky footer

portfolio.html

```
<div class="blog-footer">
```

3. Style the example divs and its contents correctly

portfolio.html

```
.example-2 {
  width: 33.3333%;
  float: left;
}
.inner-example {
  background-color: #eee;
  margin: 5px;
  border-radius: 5px;
  box-shadow: 0 1px 1px #999;
  -moz-box-shadow: 0 1px 1px #999;
  -webkit-box-shadow: 0 1px 1px #999;
}
```

****NOTE:** I forgot to include the last two lines. “-moz” and “-webkit” make it so that the box-shadow shows properly on earlier versions of Chrome, Firefox, and Safari for your users with outdated web browsers. You can do the same prefix for border-radius as well.

```
.inner-example img {
  max-width: 98%;
  margin: 1%;
}
.inner-example h3 {
  margin: 5px;
  font-size: 20px;
```

```
  border-bottom: 1px solid black;
}
.inner-example p {
  padding: 5px;
}
```

4. Inact the clear fix on your row of divs

portfolio.html

wrap your 3 example divs in

```
<div class="row-example">
```

and put in the paragraph afterwards for demonstration purposes

```
<p> this is some extra text</p>
```

5. Put in the correct styling for the clear fix

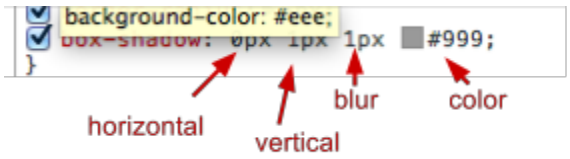
css/blog.css

```
.row-example:after {
  clear: both;
  content: " ";
  display: table;
}
```

Lesson Notes

- We declared the width of the “example-2” divs and didn’t declare the width of the “inner-example” divs, but we did declare their margin. What that does is push the inner divs in 5 pixels from the edge of the “example-2” divs. In other words say the “example-2” divs are 300px wide. Well we don’t declare the width of the “inner-example” divs but we did give them a margin of 5px. Thus the width of the “inner-example” divs will decrease to be able to include a margin of 5px on each side between them and the “example-2” div. Which would result in the “inner-example” divs having a width of 290px and a margin of 5px on each side - 290 + 5 +5 = 300.
- With the box shadow property the first value is the position of the horizontal shadow. Negative numbers moves it to the left, positive to the right. The second value is the position of the verticala shadow. Negative numbers moves it up, positive numbers down. The third value is the blur distance.

It does not allow negative numbers but the higher the number the greater the blur. I keep it low as higher numbers tend to look fuzzy and off-putting. Then the final number is the color of the shadow.



Lesson 66: Nested Columns

Lets learn even more ways to make complex layouts with nested columns.

1. Redo the portfolio page so that we use Bootstrap nested columns instead of floating the image

about.html

```
<div class="row">

  <div class="col-sm-3 xs-center">

    <img src="images/ryan.jpg" alt="Ryan">

  </div>

  <div class="col-sm-9">

    <h1>I want to make education more effective through both material - skills that matter today - and method - project-based learning. </h1>

  </div>

</div>
```

2. Style the two nested columns so that they are centered on extra small screens and fix sticky footer so that it is positioned static when on extra small screens

about.html

```
<div class="col-sm-3 xs-center">
```

css/blog.css

```
@media (max-width: 767px) {

  .xs-center, #about h1 {
```

```
    text-align: center;
  }
  .sticky-footer {
    position: static;
  }
}
```

Lesson 67: Setting Up A Landing Page

Lets start setting up a nice landing page to welcome our site visitors

1. Rename index.html as blog.html

2. Create a new index.html page from our about page

3. Fix the nav links on each page with the correct active class per page

portfolio.html

for example

```
<li><a href="index.html">Home</a></li>
<li><a href="blog.html">Blog</a></li>
<li><a href="about.html">About</a></li>
<li class="active"><a href="portfolio.html">Portfolio</a></li>
```

4. Delete the navbar in index.html

5. Insert the form from blog.html and give it the correct HTML and CSS for our homepage

```
<div class="col-sm-9">
  <h1>I send out learning hacks, business tips, and all things cool and valuable.</h1>
  <div class="row">
    <form role="form" action="http://massiveacademy.us8.list-manage.com/subscribe/post?u=636e49cc7c70ceee93
```

```
9312b50&amp;id=14d21b31a8" method="post" id="mc-embedded-subscribe-form"
name="mc-embedded-subscribe-form" class="validate" target="_blank" novalidate>
    <div class="col-sm-9">
      <div class="form-group">
        <input type="email" value="" name="EMAIL" class="form-control" id="mce-EMAIL"
placeholder="Enter email">
      </div>
    </div>
    <div class="col-sm-3">
      <div id="mce-responses" class="clear">
        <div class="response" id="mce-error-response" style="display:none"></div>
        <div class="response" id="mce-success-response" style="display:none"></div>
      </div>    <!-- real people should not fill this in and expect good things - do
not remove this or risk form bot signups-->
      <div style="position: absolute; left: -5000px;"><input type="text"
name="b_636e49cc7c70ceee939312b50_14d21b31a8" tabindex="-1" value=""></div>
      <p class="center"><input type="submit" value="Join The Fun" name="subscribe"
class="btn btn-warning"></p>
    </div>
    </form>
  </div>
</div>
```

6. Change the offset div to have an ID of home

index.html

```
<div id="home" class="col-md-10 col-md-offset-1 blog-main">
```

7. Add images in a div with an ID of home to the same CSS styling rules as those images in a div with ID of about

css/blog.css

```
#about img, #home img {
  border-radius: 100px;
  margin-right: 30px;
  margin-bottom: 10px;
}
```

8. Make sure to add the "-moz-border-radius" and "-webkit-border-radius" to all border radius properties.

This makes sure that border-radius displays correctly with older versions of Safari, Firefox, and Chrome.

Lesson 68: Background Properties

Have a little fun with all the different background possibilities

1. Take out the row and offset div

index.html

delete

```
<div class="row">
  <div id="home" class="col-md-10 col-md-offset-1 blog-main">
```

2. Rename the blog-page div with an ID of home-top and take the ID of home out of the third div

3. Fix the CSS to style the image correctly

css/blog.css

```
#about img, #home-top img {
  border-radius: 100px;
  -moz-border-radius: 100px;
  -webkit-border-radius: 100px;
  margin-right: 30px;
```

```
  margin-bottom: 10px;
}
```

4. Delete the blog-main div

5. Move the first paragraph outside of the “home-top” div

index.html

```
<div class="container blog-main">
  <p id="first-paragraph">I am the founder of MASSIVE Academy and Online Director for Miles 2 Give. As you can see above I have a strong passion for improving education, specifically online education and have a dream that one day every one will have the chance to learn how to code and build things online. </p>
</div>
```

6. Style home-top correctly

css/blog.css

```
#home-top {
  padding-top: 60px;
  padding-bottom:60px;
  background: url(../images/background.jpg) repeat;
}
```

****NOTE: Make sure to grab the background image from http://massive.academy/baw**

Lesson 69: Finishing Up

Put on the finishing touches and you have yourself a good looking blog template!

1. Edit the home-top h1

index.html

```
<h1>I send out learning hacks, business tips, and all other things I think are cool and can give you a competitive advantage. I'd love for you to join.</h1>
```

2. Style the form and button to be bigger

css/blog.css

```
#home-top h1 {
  font-weight: 500;
}
#home-top .btn {
  font-size: 25px;
}
#home-top input.form-control {
  font-size: 25px;
  height: 50px;
}
```

****NOTE:** You could also have used “#home-top input[type="email"]” to be more specific in case you wanted to add other inputs like a first name.

3. Change up and style the first paragraph

index.html

```
<p id="first-paragraph-home">I am the founder of MASSIVE Academy and Online Director for Miles 2 Give. I have a strong passion for improving education and the way we learn. Head over to my <a href="blog.html">blog</a>, check my <a href="about.html">about page</a> to learn more about me, or sign up for a <a href="https://www.udemy.com/build-your-first-website-in-1-week/">my course</a> to learn to build your first website.</p>
```

css/blog.css

```
#first-paragraph-home {
  padding-top: 50px;
}
```

Lesson 70: Building A Full-Width Magazine Layout

Let's learn how to make an awesome, full-width magazine layout site

You know the drill when getting started...

1. Download bootstrap, unzip the folder, rename it "magazine-layout", put it in your desktop and open it in Sublime

2. Copy the basic template from bootstrap and create an index.html file within your *magazine-layout* folder

3. Replace the current *Hello World! h1* with a jumbotron div

index.html

```
<div class="jumbotron">

  <div class="container">

    <h1>We are <strong>improving education</strong><br> through the material we teach and the method by which we deliver our lessons.</h1>

  </div>

</div>
```

Lesson 71: Recreating A Jumbotron Section

Lets take a walk down memory lane and revisit how to create a jumbotron section

1. Grab image from Massive Academy site or download from downloadable materials on the right panel in Udemy and save it in *images* folder

2. Create a custom.css file and link to the background image for the jumbotron

custom.css

```
.jumbotron {

    background-image: url('../images/background.jpg');

}
```

3. Link to the custom.css in your index.html

custom.css

```
<link href="css/custom.css" rel="stylesheet">
```

4. Consolidate the background properties and give it some padding and border to make it look good

custom.css

```
.jumbotron {
    background: url('../images/background.jpg') no-repeat center center;
    -o-background-size: cover;
    -moz-background-size: cover;
    -webkit-background-size: cover;
    background-size: cover;
    padding-top: 170px;
    padding-bottom: 170px;
    border-top: 3px solid #e74c3c;
}
```

Lesson 72: Background Opacity And Text Shadows

We'll add another tool to your website making arsenal with text shadows and revisit to rgba

1. Properly style the H1 to look good and be readable for our viewers

custom.css

```
.jumbotron h1 {
    font-family: klavika, arvo, "Helvetica neue", serif;
    padding: 10px;
    text-align: center;
    margin: 10px 20px;
```

```
    font-weight: 600;
    font-size: 50px;
    text-transform: uppercase;
    color: white;
    background-color: rgba(0,0,0,.1);
    text-shadow: 1px 1px 2px #333;
}
```

2. Change the strong tags to be span tags with a class of highlight

index.html

```
<h1>We are <span class="highlight">improving education</span><br> through the material we teach and the method by which we deliver our lessons.</h1>
```

3. Add the color rule to the highlight class

custom.css

```
.highlight {
    color: #e74c3c;
}
```

Lesson 73: Importing Fonts From Your Computer

We've learned in the past how to add Google Fonts to our site. Now let's learn how to add fonts we have on our computer

NOTE: I don't have the rights to share the Klavika font with you so if you don't have a font you want to import follow along in this lesson until we import the Arvo font and then pick up there. Use this lesson as a future reference when you want to import your own fonts. I wanted to include the first part of this lesson to show you in case you or your client ever wants to import a specific font that you can't link to online like we have before with Google Fonts.

1. Bring your fonts into your folder's font folder

2. Link to your fonts in CSS with @font-face

custom.css

```
@font-face {
      font-family: klavika;
      src: url(../fonts/Klavika-Regular.otf);
      font-weight: 300;
}
@font-face {
      font-family: klavika;
      src: url(../fonts/Klavika-Bold.otf);
      font-weight: 600;
}
@font-face {
      font-family: klavika;
      src: url(../fonts/Klavika-Light.otf);
      font-weight: 100;
}
```

3. Connect to Arvo font in your index file as a fallback if you don't have the Klavika font or if it doesn't load properly.

index.html

```
<!-- Arvo font -->
<link href='https://fonts.googleapis.com/css?family=Arvo:400,700' rel='stylesheet'
type='text/css'>
```

Lesson 74: Static Navbars

Up to this point you've been working with fixed navbars. Now it's time to learn how to create a static navbar

1. Grab the starter template * * and bring into your site above the jumbotron section

2. Move the navbar to the right of the screen

index.html

```
<ul class="nav navbar-nav navbar-right">
```

3. Update the links in the navbar

index.html

```
<li><a href="#about">Start Learning</a></li>
```

```
<li><a href="#contact">About Us</a></li>
```

4. Change navbar to not be fixed, but instead have a static positioning

index.html

change

```
<div class="navbar navbar-inverse navbar-fixed-top" role="navigation">
```

to

```
<div class="navbar navbar-inverse navbar-default" role="navigation">
```

5. Change the background color to be transparent and fix the margin

custom.css

```
.navbar {
    background-color: transparent;
    border: none;
    margin-bottom: 0px;
}
```

6. Give jumbotron a negative margin so that it moves up

custom.css

```
.jumbotron {
    background: url('../images/background.jpg') no-repeat center center;
    -o-background-size: cover;
    -moz-background-size: cover;
    -webkit-background-size: cover;
    background-size: cover;
    padding-top: 170px;
    padding-bottom: 170px;
    border-top: 3px solid #e74c3c;
    margin-top: -50px;
}
```

7. Grab the logo from the site or from this lesson's downloadable materials and save it in the images folder

8. Include the logo on your site

index.html

```
<a class="navbar-brand" href="#"><img src="images/logo.png" alt="MASSIVE Academy"></a>
```

9. Change the width of the logo

custom.css

```
.navbar-brand img {
    width: 90%;
    max-width: 353px;
}
```

Lesson 75: Finishing Up The Navbar

And now it's time to put the final touches on our beautiful, transparent navbar

1. Correct the spacing with the navbar and jumbotron

custom.css

```
.navbar {
	background-color: transparent;
	border: none;
	margin-bottom: 0px;
	padding-top: 10px;
}
.jumbotron {
	background: url('../images/background.jpg') no-repeat center center;
	-o-background-size: cover;
	-moz-background-size: cover;
	-webkit-background-size: cover;
	background-size: cover;
	padding-top: 170px;
	padding-bottom: 170px;
	border-top: 3px solid #e74c3c;
	margin-top: -60px;
}
```

2. Fix the links in the navbar to have the correct styling

custom.css

```
.navbar-inverse .navbar-nav>li>a {
	color: #fefefe;
	opacity: .7;
```

```
    text-transform: uppercase;
    padding-top: 25px;
    padding-bottom: 10px;
}
```

3. Fix margin of jumbotron so that it now goes to top of page with navbar being 65px tall

custom.css

```
.jumbotron {
    background: url('../images/background.jpg') no-repeat center center;
    -o-background-size: cover;
    -moz-background-size: cover;
    -webkit-background-size: cover;
    background-size: cover;
    padding-top: 170px;
    padding-bottom: 170px;
    border-top: 3px solid #e74c3c;
    margin-top: -65px;
}
```

4. Take away the font family from *.jumbotron h1* and create styling for the body

custom.css

```
body {
    font-family: klavika, arvo, "Helvetica neue", serif;
    font-size: 16px;
}
```

Lesson 76: Fluid Containers

Let's learn how to make website layouts that span the width of the screen. Oh how the possibilities are endless now!

1. Time to set up the bottom section with 3 equal-width divs

index.html

```
<div id="bottom-section" class="container-fluid">
  <div class="row">
    <div id="byfw" class="col-md-4">
    </div>
    <div id="baw" class="col-md-4">
    </div>
    <div id="oml" class="col-md-4">
    </div>
  </div>
</div>
```

2. Grab each of the images from the site or from the downloadable materials section in this lesson

3. Include the images in CSS and add correct color for the background

custom.css

```
#byfw {
      background-image: url(../images/byfw.jpg);
      background-color: #e74c3c;
}
#baw {
      background-image: url(../images/baw.png);
      background-color: #f39c12;
}
```

```
#oml {
      background-image: url(../images/oml.jpg);
      background-color: #f1c40f;
}
```

4. Style each div correctly

custom.css

```
#bottom-section .col-md-4 {
      min-height: 300px;
      background-size: 100%;
      background-repeat: no-repeat;
}
```

5. Fix the spacing between the bottom section and the jumbotron

custom.css

```
.jumbotron {
      background: url('../images/background.jpg') no-repeat center center;
      -o-background-size: cover;
      -moz-background-size: cover;
      -webkit-background-size: cover;
      background-size: cover;
      padding-top: 170px;
      padding-bottom: 170px;
      border-top: 3px solid #e74c3c;
      margin-top: -65px;
      margin-bottom: 0px;
}
```

Lesson 77: Linking Divs

Sometimes you want to make entire divs clickable. Let's learn how to do so

1. Bring in the paragraphs and buttons for each div

index.html

```
<div id="bottom-section" class="container-fluid">
  <div class="row">
    <div id="byfw" class="col-md-4">
      <p>Learn the basics of HTML5 and CSS3 and have your first website up in 1 week. Hecks yeah!</p>
      <p><button class="btn btn-success">Learn More</button></p>
    </div>
    <div id="baw" class="col-md-4">
      <p>Be able to build beautiful, startup quality sites. Learn HTML5 and CSS3 in less than 1 month.</p>
      <p><button class="btn btn-success">Learn More</button></p>
    </div>
    <div id="oml" class="col-md-4">
      <p>We are bringing coding in to the Tallahassee school system to empower's Tallahassee's future today.</p>
      <p><button class="btn btn-success">Get Involved</button></p>
    </div>
  </div>
</div>
```

2. Fix the styling for the divs and their paragraphs

custom.css

```
#bottom-section .col-md-4 {
      min-height: 300px;
```

```
    background-size: 100%;
    background-repeat: no-repeat;
    text-align: center;
    padding-top: 250px;
}
#bottom-section .col-md-4 p {
    color: white;
    font-family: klavika, 'arvo', sans-serif;
    font-size: 18px;
    font-weight: 100;
    margin-bottom: 10px;
}
```

3. Style the button

index.html

```
#bottom-section .col-md-4 .btn-success {
    background-color: #27ae60;
    margin-bottom: 5px;
    font-size: 20px;
    font-weight: 100;
}
```

4. Wrap each div in a link

5. Add opacity to the divs when hovered for a nice subtle effect

custom.css

```
#bottom-section .col-md-4:hover {
    opacity: .9;
}
```

Lesson 78: Finishing Up With The Footer

Let's finish it all up with a simple footer section

1. Let's set up the foundation for our footer

index.html

```
<footer>
  <div class="container">
    <div class="col-sm-6">
    </div>
    <div class="col-sm-6 right-align">
    </div>
  </div>
</footer>
```

2. Download and bring in [Font Awesome]

index.html

```
<!-- Include Font-Awesome -->
<link rel="stylesheet" href="fonts/font-awesome/css/font-awesome.min.css">
```

3. Put in the paragraph on the right

index.html

```
<p>Made in Tallahassee with <i class="fa fa-heart"></i> | &#169; 2014 MASSIVE Academy</p>
```

4. Style the footer correctly

custom.css

```
footer {
    background-color: #333;
    padding-top: 20px;
```

```
    padding-bottom: 20px;
    font-weight: 200;
    color: #fefefe;
}
```

5. Bring in the icons and align them to the right with the correct color

index.html

```
<div class="col-sm-6 right-align">
  <a href="#"><span class="fa-stack">
    <i class="fa fa-circle fa-stack-2x"></i>
    <i class="fa fa-question fa-stack-1x fa-inverse"></i>
  </span></a>
  <a href="#" target="_blank"><span class="fa-stack">
    <i class="fa fa-circle fa-stack-2x"></i>
    <i class="fa fa-envelope fa-stack-1x fa-inverse"></i>
  </span></a>
  <a href="#" target="_blank"><span class="fa-stack">
    <i class="fa fa-circle fa-stack-2x"></i>
    <i class="fa fa-facebook fa-stack-1x fa-inverse"></i>
  </span></a>
  <a href="#" target="_blank"><span class="fa-stack">
    <i class="fa fa-circle fa-stack-2x"></i>
    <i class="fa fa-twitter fa-stack-1x fa-inverse"></i>
  </span></a>
</div>
```

custom.css

```
footer a {
```

```
    color: #ccc;
}
.right-align {
    text-align: right;
}
```

6. Give the left side of the footer some margin-top

custom.css

```
footer p {
    margin-top: 5px;
}
```

Lesson 79: Revisiting Responsive Design

Did you try to do the responsiveness yourself? If not try to before you watch this video. I think you'll surprise yourself at how much you know!

Note: Remember there is more than one way to a solution in web development. So if the responsive design you did looks good, but my code and the way I do it in this video is different there is no shame in that game.

1. Change the divs to have a class of *col-sm-4*

2. Make sure change any CSS rule that has a class selector of *col-md-4* to *col-sm-4*

3. Change the styling for the *col-sm-4* divs

custom.css

```
#bottom-section .col-sm-4 {
    min-height: 300px;
    background-size: 100%;
    background-repeat: no-repeat;
    text-align: center;
    padding-top: 20%;
```

```
}
```

4. Put in styling for bottom section when we hover over a link

custom.css

```
#bottom-section a:hover {
	text-decoration: none;
}
```

5. Put in the responsive design

custom.css

```
/* responsive design */
@media (max-width: 768px) {
	#bottom-section .col-sm-4 {
		min-height: 425px;
		padding-top: 55%;
		padding-bottom: 5px;
	}
	.jumbotron h1 {
		font-size: 25px;
	}
	.navbar-brand {
		width: 50%;
	}
	.navbar-brand img {
		width: 220px;
	}
	.navbar-inverse .navbar-brand {
		border: none;
```

```
    }
    .jumbotron {
        margin-top: -186px;
        padding-top: 210px;
        padding-bottom: 90px;
    }
}
```

Lesson 80: What's Next?

Thank you for taking this course! 3 quick things:

1. At this point you now know how to build a tower site, a blog template, and a portfolio/magazine layout among many other things. You have the knowledge and tools to make a variety of sites you can dream up, but there may be some questions you have or some ideas for something I didn't cover. So I'd love to hear your feedback and know what else you'd like to learn in this course. I want to add more videos as the course goes along in order to keep improving the course so please respond. You can email me at ryan@massive.academy, tweet me @massive_academy, or post on the discussion here to let me know what else you'd like to learn in this course. If it's something outside of html and css let me know too and I may make another course or do a video lesson on it on our blog.

2. Also, I'd love to see what you're creating from this course! Once again, please email me at ryan@massive.academy, tweet me @massive_academy, or post on the discussion here a link to your site.

3. And finally, if you enjoyed this course and think others would too I'd really appreciate a review. The more reviews I get the more likely someone will sign up for this course when they come to the page. Also here's a buddy pass for 50% off of this course if you want to share it with a friend: http://bit.ly/1kyHR7Y