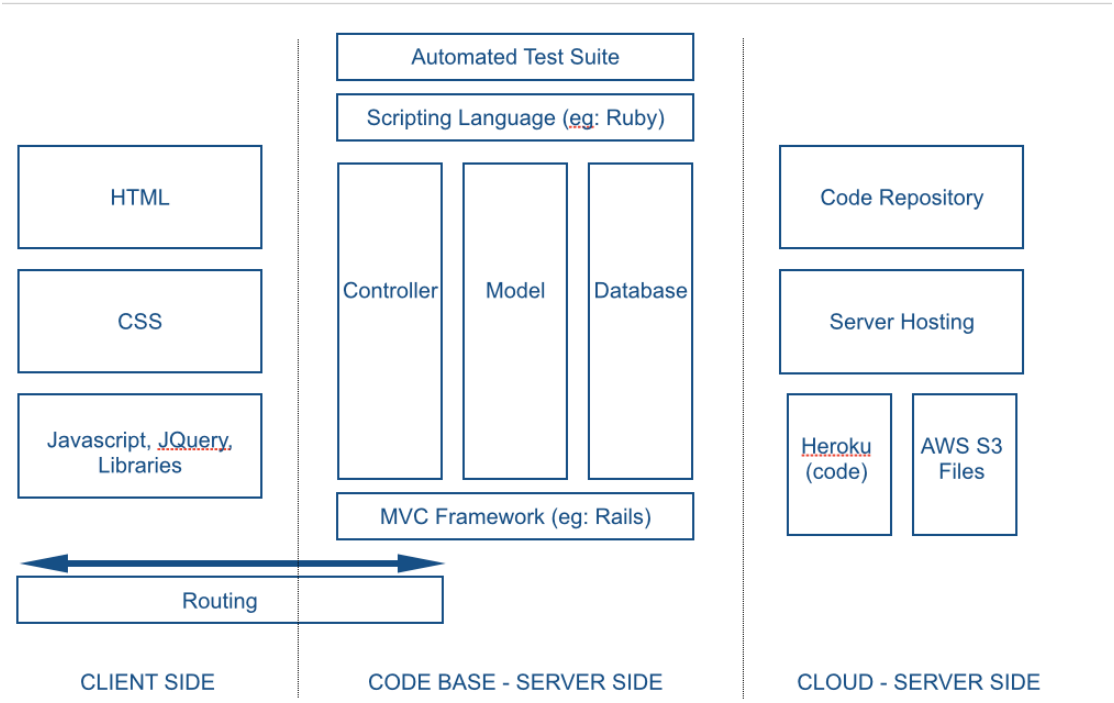
CODING FOR PRODUCT MANAGERS

INTRODUCTION

Web development brings together a varied of languages, frameworks, and technologies, each of which is a system in and of its own. When combined their complexity is compounded and the process can seem impenetrable.

In this course, we're going to break down each component from the outside in, building a blog app (https://product-school-blog.herokuapp.com/) that brings them all together in a functioning, live system. In the process, you'll get an understanding of both the parts and the whole. By the end of this course, our goal is that you'll be able to build on this base of learning to create more complex apps as well as collaborate with developers as a product manager to build products that far exceed what you could have created without an understanding of the development process (not to mention building them in a more frictionless, time-efficient manner!).

We'll tackle three core pillars of the web development process: front-end web development, backend web development, and cloud-side code deployment and hosting. Here's a snapshot of what we'll be learning:



In our front-end module, we'll start with HTML, CSS, and mobile-optimized responsive styling.

We'll then dive into Ruby, your first backend scripting language. We'll cover the basics of the language and how you can use it to build functionality into your system. Once we have a sense of Ruby, we'll dive into Rails, a Ruby-based Model-View-Controller framework that will allow you to quickly build out your blog app. In the process we'll also get familiar with Git, Github, versioning, and databases – which are a constant in any good web development process. We'll round out our backend module by diving into automated testing as a way of building reliability and efficiency as your codebase scales.

Finally, you'll learn how to push your application onto Heroku and integrate with Amazon's AWS S3 service to dynamically serve images that are uploaded by users of your app.

If we happen to have time, we can dive back into JavaScript, which is a client-side scripting language, and JQuery a JavaScript library, which will allow to you create interactivity in and additional functionality into your web page.

You can access all of the code for this course on the Product School github account at: https://github.com/product-school/ .

TIMELINE (tentative):

CLASS 1 – HTML, CSS & Responsive Web Styling

CLASS 2 – Git, Github & Ruby

CLASS 3 – Introduction to the Rails MVC framework – Database Structure, Models, Views, and Controllers

CLASS 4 – Rails Continued

CLASS 5 – Automated Testing

CLASS 6 – Hosting on Heroku & Amazon AWS S3. Overview of Javascript & JQuery and other relevant languages/frameworks.

Unit 1 - HTML, CSS, & Responsive Styling

1.1 HTML

Introduction:

HTML (Hyper-text Markup Language) and CSS (Cascading Style Sheets) represent the skeleton and outer-facade of any page that you see on the Internet. Any frontend or full-stack developer will be working with these daily. The basic concepts within HTML and CSS can be learned quickly and you'll have time to deepen your knowledge of specific attributes, tricks, and resources over time.

Pre-Work:
- Download Sublime Text (http://www.sublimetext.com/2). You'll use this as your code editor for the rest of the program.
- Download Chrome (if you don't yet have it downloaded).
- Install Xcode on your laptop (via the app store). Open up Xcode, in the Xcode dropdown (upper left hand side of the screen) go to Preferences and click on the Download tab. Check "Command Line Tools" and then click the "Check and Install Now" button at the bottom left of the window. This will take a while, so feel free to let the download run in the background. Note: If you don't see the 'Command Line Tools' option, open the Utilities folder within Applications and then open the Terminal application. Then enter the command xcode-select –install . If still causes an error, hold off for help in class.
- Create a ProductSchool folder on your computer where you'll house all of your code
- Check out the blog you'll be building: https://product-school-blog.herokuapp.com/

In-Class:

i. HTML page structure, head & body

```
<! DOCTYPE Html>
<html>
        <head>
        <title>Page Title</title>
        </head>
    <body>
            [Page's HTML content]
    </body>
</html>
```

ii. HTML body tags (full list: http://www.w3schools.com/tags/)

Most Commonly Used Tags:

- Full Width HTML Tags
  - Headings: <h1> - <h6>

    <h1>Hello World</h1>
  - Paragraph:

    <p>Copy text <p>
  - Div (aka 'box'):

     [content to be placed inside]
  - Ordered (numbered) & Unordered (bulleted) Lists: <ul> &


```
    <li>Point one</li>
    <li>Point two</li>
    <li>Point three</li>
  </ul>
```

- Blockquote:

```
<blockquote>Copy worth quoting</blockquote>
```

- Horizontal Rule (aka: horizontal divider line): <hr>

- Line break:

- Nav: <nav> </nav>

```
<nav>
      <a href="/html/">HTML</a> |
      <a href="/css/">CSS</a> |
      <a href="/js/">JavaScript</a> |
</nav>
```

- Footer:

```
<footer>
  <ul>
        <li><a href="/about/">About Us</a></li>
        <li><a href="/privacy/">Privacy</a></li>
        <li><a href="/contact/">Contact Us</a></li>
  </ul>
</footer>
```

- Form:

<form>[1]

```
    <div>
      <label for="post_title">Title</label>
      <input required="required" type="text">
    </div>

    <div>
      <label for="post_image">Image</label>
      <input required="required" type="file">
    </div>

    <div>
      <label for="post_show_image">Show Photo?</label>
  <input type="checkbox" name="post_show_photo" value="1">
    </div>

    <div>
       <label for="post_category">Category</label>
       <select>
         <option value="1">Product School</option>
         <option value="2">Travel</option>
    </div>

    <div class="actions">
      <label for="post_submit">Submit</label>
      <input type="submit" value="Create Post">
    </div>
</form>
```

[1] Form code accessible @ https://github.com/product-school/html-css-exercises/blob/master/form.html

Common form text field types: text, email, password, search, URL

Form field types without text field 'input': checkbox, radio, number range

Form buttons: button, image, reset, submit

iii. In-Line HTML Tags

- Image: <img src="linked-file-path" alt="alternative title">
- Anchor (aka: 'link'): Linked copy

iv. Formatting HTML Tags

- Bold: bolded text ; alternate: bolded text
- Underline: <u>underlined text</u>
- Italics: <i>Italicized text<i>

1.1.2 In-line Styling

- <tag style="attribute: value;">
- Starter styles: width (px or %), height (px or %), text-align (left, center, right), color (hex), background-color, border (px color solid/dashed)

In-Class Exercises:

- Create the HTML structure for your blog index page[2]

1.1.3 Boxes: Positioning, Display Types & Floats

In-Class Reading:

- Read: http://learnlayout.com/position.html
- Read up on floats: http://css.maxdesign.com.au/floatutorial/introduction.htm

i. Box Positions

- Static:
  - Default, 'unstyled' position
- Relative:
  - Behaves the same as static unless you add some extra properties.
  - Setting right, left, top, bottom, will adjust the position of a relative box.
  - Other elements will not fill open gaps (eg: if the width of a relatively positioned box is only 40% of the line, the other 60% of the line will be empty).
- Fixed:
  - A fixed position element is positioned relative to the viewport and will stay in the same place in the browser as you scroll.
  - Does not leave a gap in the page where it would have otherwise been placed.
- Absolute
  - Behaves like a fixed position element, but is fixed relative to it's nearest parent rather than the viewport

ii. Box Display Types:

- Inline Element: has no line break before or after it, and it tolerates HTML elements next to it.
  - Respect left & right margins and padding, but **not** top & bottom
  - **Cannot** have a width and height set

[2] HTML + CSS rendering of the page for reference: http://product-school.github.io/blog-html-css/index.html

- Allows other elements to sit to their left and right.

- Block Element: Owns the full width of the page
  - Respect top & bottom margins and padding
  - Respects height and width
  - Forces a line break after the block element

- Inline-block Element: is placed as an inline element (on the same line as adjacent content), but it behaves as a block element.
  - Allows other elements to sit to their left and right
  - Respects top & bottom margins and padding
  - Respects height and width

iii. Floats

- Floated boxes will move to the left or right until their outer edge touches the containing block edge or the outer edge of another float.

- If there isn't enough horizontal room on the current line for the floated box, it will move downward, line-by-line, until a line has room for it.

- Block level elements above a floated element will not be affected by it. However, elements below will wrap around the floated element.

In-Class Exercises[3]:

- Add CSS to the html scaffolding of this blank page: https://github.com/product-school/html-css-exercises/blob/master/blank_face.html into a face[4]: http://product-school.github.io/html-css-exercises/face.html

1.2 CSS

Resources:
- For a full-list of css properties: http://www.w3.org/TR/CSS21/propidx.html .
- For more information on CSS selectors: https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Getting_started/Selectors

i. File Structure & Syntax

- Linking your css file to your html file

  <link rel="stylesheet" href="/application.css">

- Syntax

```
h1 {
    property: value;
    color: red;
    font-size: 16px;
    text-align: center;
}
```

ii. CSS Selector Types

- HTML Tags:

[3] Here's a list of common style properties to get you started: http://tech.journalism.cuny.edu/documentation/css-cheat-sheet/
[4] Feel free to use this triangle generator: http://apps.eky.hk/css-triangle-generator/

```
h1, p, div{
    color: red;
}
```

- Classes & Id's: HTML tag attributes used for CSS & JavaScript targeting

  - Classes: Assigned to one or more HTML elements on a page.

```
.my-class {
    text-align: center;
    margin: 0 auto;
}
```

- IDs: Assigned to only ONE HTML element on a page

```
#my-Id {
    color: green;
}
```

- Pseudo Selectors: styling that is assigned to a state of a page element or specific subset of elements

  - a:hover { color: red;} : Links will turn red when a mouse hovers over them
  - p:first-child { background-color: gray;}: The first paragraph element on a page will have a gray background color
  - div:nth-child(3) { text-align: center; }: Text in the 3rd div on the page will be centered

- Selectors based off of relationships

  - A E {}: Any E element that is a *descendant* of an A element (that is: a child, or a child of a child, *etc*.)
  - A > E {}: Any E element that is a child of an A element
  - E:first-child {}: Any E element that is the first child of its parent
  - B + E {}: Any E element that is the next *sibling* of a B element (that is: the next child of the same parent)

iii. Chrome developer tools

- Elements + CSS
- Console

1.3 Responsive Styling

Introduction:

Responsive styling enables you to customize how a page is displayed at different browser sizes. With media queries, you can set custom CSS for any browser with you'd like to target.

i. Typical Viewport Sizes

- Smartphones: 680 x 960 pixels
- Tablets: 768 x 1024 pixels
- Laptops: 1440 x 900 pixels
- Desktops: 1920 x 1080 pixels

ii. Responsive media queries

```
@media (max-width: 767px) and (min-width: 480px) {
    h2 {
        font-size: 12px;
```

```
        }
        nav {
                display: none;
        }
    }
```

iii. Head Meta-tags:

Add this to the head of your file to let your mobile browser know your site is optimized for mobile: <meta name="viewport" content="width=device-width, initial-scale=1">

Homework:

i. Finishing Up The HTML + CSS For Your Blog:

- Review the most common CSS style properties: http://tech.journalism.cuny.edu/documentation/css-cheat-sheet/
- Finish the HTML for the index and individual blog page of your blog.
- Add responsive CSS styling to your Product School blog index and post page. The end result should look something like this: http://product-school.github.io/blog-html-css-responsive/index.html . Here is a link to a final code version on Github, if your absolutely need it! https://github.com/product-school/blog-html-css

i. Prepping For Next Class

- Git & Github: Do all of the steps included in the Unit 2 pre-work
- Ruby:
  - Do CodeSchool's 15 minute Intro to Ruby exercise: http://tryruby.org/levels/1/challenges/0
  - Go the full way through this intro to Ruby tutorial: https://www.ruby-lang.org/en/documentation/quickstart/

Bonus Exercise (if you're bored and want to continue honing your skills)

- Recreate the image below using box positioning and inline styling: Live version here: http://product-school.github.io/html-css-exercises/box-positioning.html