CODE IN ONE DAY:

HTML & CSS Crash Course

INTRODUCTIONS

GitHub

jmeade11

Skype

jmeade11

LinkedIn

linkedin.com/in/jenniferannmeade

CREATING CAREERS DAILY

learn the skills to get the job you ❤

CODING

DATA

PRODUCT MANAGEMENT

UX & DESIGN

DIGITAL MARKETING

BY THE NUMBERS

20
WORLDWIDE CAMPUSES

60k+
GLOBAL ALMUNI

2,500+
HIRING PARTNERS

250+
EXPERT INSTRUCTORS

WORKSHOP RULEZ

- Have a text editor installed:
  ATOM and **VS Code** are free
- Work through the **Code Along** and **Lab** exercises
- Wifi: **GA Guest**
  Password: **yellowpencil**
- Live presentation link:
  slides.com/jennifermeade/crashcourse/live

Ask what you don't know and **share** what you do

A QUIZ ALREADY?

CHECKPOINT #1: About You

Where are you in your journey to learning to program the Web?

- [ ] Just testing the waters
- [ ] My journey stalled around the time when tables were used for page layout
- [ ] Working my way through YouTube on a path to becoming a professional coding ninja

WORKSHOP TOPICS

- How the web works
- Building web pages with HTML & CSS
- How stuff behaves in the (sometimes weird) world of the web
- Where to go from here

HOW WEB PAGES WORK

HOW THE INTERNET WORKS

WEB PAGE COMPONENTS

HYPERTEXT MARKUP LANGUAGE

HTML provides the ***structure*** for your web pages

CASCADING STYLE SHEETS

CSS defines how the elements of the web page should ***look***

JAVASCRIPT

Javascript adds
interactivity

HISTORY OF WEB STANDARDS

PART 1

PART 1: OBJECTIVES

- Describe the anatomy of a web page and create the basic structure of a page in HTML
- Apply HTML tags to web page content
- Style web pages with CSS
- Link external CSS files

HTML BASICS

HTML SYNTAX

HEADS UP: Some tags only have a start tag.

ADDING ATTRIBUTES

WHAT'S UP DOC?

- The HTML5 doctype tells the browser to behave **strictly** according to the standards
- Case-insensitive but conventionally written as:

```
<!DOCTYPE html>
```

ANATOMY OF A WEBPAGE

Every HTML page has the same foundational structure

```html
<!DOCTYPE html>
<html>
    <head>

        <meta charset="UTF-8">
        <title>Page Title</title>

    </head>
    <body>

        <!-- Where your content goes -->

    </body>
</html>
```

CONTENT TAGS

Your content needs structure too! Content tags are used to delineate different page contents. They go in between the `<body>` ... `</body>` tags in your html page. Let's look at some...

Content Tags

HEADING TAGS

```
<h1>Largest Heading</h1>

<h2> . . . </h2>

<h3> . . . </h3>

<h4> . . .</h4>

<h5> . . . </h5>

<h6>Smallest Heading</h6>
```

Content Tags

TEXT ELEMENTS

```
<p>paragraph text</p>

<code>computer code</code>
```

Content Tags

UNORDERED LIST

```
<ul>

</ul>
```

Content Tags

LIST ITEM

```
<ul>

    <li>A List Item</li>

    <li>Another list item</li>

</ul>
```

Content Tags

ANCHOR TAGS (LINKS)

```
<a href="url-or-link-address">Link Text</a>
```

PRESS RELEASE

BLOCK ELEMENTS

Block elements ***block*** other elements from sitting next to them.

```
<p>I'm a block element.</p>
<div>So am I!</div>
```

I'm a block element.

So am I!

INLINE ELEMENTS

Inline elements ***wrap*** inside their containing elements.

```
<p>
  Both <span>span</span> and
  <a href="#top">anchor</a>
  are inline elements.
</p>
```

Both span and anchor are inline elements.

WEBPAGE BUILDING BLOCKS

- **header**: used for repetitive elements at the top of the page such as branding and menus.
- **nav**: used to wrap any kind of navigation menu.
- **main**: used for the main content of the page.
- **aside**: used for ancillary content.
- **section**: a general purpose section commonly wraps divs.
- **div**: a general purpose section (division)
- **footer**: used to wrap the page footer contents.

🚩 **HEADS UP:** Don't confuse `header` with `head`.

HELPFUL INLINE TAGS

- **span**: a general purpose inline tag.
- **strong**: used to add weight (replaced the b tag).
- **em**: used to add emphasis (replaced the i tag).
- **a**: anchor tags creating links to different content on the page or pages on the web.
- **code**: used to wrap example code.
- **img**: the img (image) tag is a little bit of both, but technically inline!

For more details, see Block-level Elements and Inline-level Elements on MDN.

COOKIE RECIPE

LAB SETUP

https://github.com/jmeade11/crashcourse

1. Go to the URL above
2. Click the green **Clone or download button**
3. Click the link to **Download ZIP**
4. Uncompress the crashcourse-master.zip file and move it to your desktop
5. Open the crashcourse-master and navigate to the labs folder

Lab 1:

LAB TASKS

https://github.com/jmeade11/crashcourse/labs

1. In Sublime Text, choose **File>Open** and open the the labs folder
2. Expand the lab folder, then expand the **lab1** folder and click on the index.html file to open it
3. Add your HTML boilerplater
4. Copy the recipe from the online instructions and paste it in the body
5. Use what you've learned to add tags to the recipe
6. Control + click on the file in Finder and choose **Open With** > **Chrome** to review your work

CSS BASICS

SELECTORS, PROPERTIES, AND VALUES

(OH MY!)

```
selector {
  property: value;
}
```

CSS SELECTORS

Selectors **target** the elements you want to style. They can be:

1. **Element Tags**
2. Classes & IDs
3. Attributes
4. Pseudo Classes
5. Combinators

PROPERTIES

Properties are the **styles** you want to apply to the targeted elements.

- border
- color
- background-color

VALUES

Acceptable values are specific to the property:

- border has values for the width, color and style
- color and background-color accept a color value

STYLE TAG

CSS ***can*** be added to your HTML page with a `style` tag.

```
<!DOCTYPE html>
<head>
  <title>My Page Title</title>
  <style>
  <!--Style tags only go inside your head tags-->
    h1 {
      color: blue;
    }
  </style>
</head>
<body>
  ...
```

CSS EXAMPLE

```
<style>
  h1 {
    color: darkblue;
    font-family: sans-serif;
  }
  h2 {
    color: maroon;
    font-family: sans-serif;
  }
  body {
    background-color: lightgray;
    border: 5px solid darkblue;
    padding: 5px;
  }
</style>
```

HEADS UP: Property names and values are case-sensitive!

CSS Properties

COLOR PROPERTIES

The `color` property sets the font color.

```
/* make the fonts on
   the page blue */

body {
  color: blue;
}
```

The `background-color` property sets the fill color.

```
/* make the page red */

body {
  background-color: red;
}
```

BORDER PROPERTIES

```
/* border is shorthand for:
   border-width
   border-style
   border-color */

ul {
  border: 1px solid darkgray;
}
```

Borders can be set for all sides or individually (e.g., border-bottom).
Properties can also be specified individually (e.g., border-top-style).

FONT PROPERTIES

```
/* font is shorthand for:
   font-style, font-variant,
   font-weight, font-size,
   line-height, font-family; */
p {
  font: bold 16px sans-serif;
}
```

HEADS UP: This shorthand requires the ***font-family*** and ***font-size*** be specified, and font-family must appear *last*.

CSS BASICS

EXTERNAL STYLESHEETS

LINKING FILES

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

🚩 **HEADS UP:** No `<style>` tags are used!

LINKING FILES RELATIVELY

index.html → styles.css:

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

1. Start in the folder where the current file is
2. Go into the folder called **css**
3. Get the file called **styles.css**

COOKIE RECIPE CSS

Lab 2:

LAB TASKS

https://github.com/jmeade11/crashcourse/labs

1. Open the lab2 folder and use the starter index.html file.
2. Make the background for the html element the color **moccasin**.
3. Set the font for the page to **16px** and **sans-serif**
4. Style the link color as **chocolate**
5. Add a **solid**, **15px** wide, **chocolate** colored border around the body.
6. The body should have its background set to **oldlace**, font color set to **saddlebrown**.
7. BONUS Change the h1 to **40px** and **center** it.

GIVE YOURSELF A 👏

Ready for more?

PART 2

QUICK REVIEW

- Describe the different parts of an HTML page
- What is the doctype used for?
- What tag is used to link to an external stylesheet?

PART 2: OBJECTIVES

- Apply margins, padding and borders to elements
- Use fixed and relative units to adjust the height and width of elements
- Add colors with transparency
- Add images to the page and background

MARGINS & PADDING

CSS BOX MODEL

1. The **content box** is defined by the height and width of the elements inside it.
2. Adding **padding** creates interior space around the content.
3. The background extends to the **border**.
4. The **margin** creates space on the outside of the element.

MARGINS & PADDING

```
div {
    padding-top: 16px;
    padding-right: 30px;
    padding-bottom: 16px;
    padding-left: 30px;
}
```

```
div {
    margin-top: 24px;
    margin-right: 16px;
    margin-bottom: 24px;
    margin-left: 16px;
}
```

🚩 **HEADS UP:** Margins can have **negative** values.
Padding can only have positive values.

SHORTHAND VERSION

with 1 value

```
div {
    margin: 20px;
    padding: 20px;
}
```

All sides are the same.

SHORTHAND VERSION

with 2 values

```
div {
    margin: 1em 20px;
    padding: 1em 20px;
}
```

Top/Bottom are 1em

Right/Left are 20px

SHORTHAND VERSION

with 3 values

```
div {
  margin: 0 1rem;
  padding: 20px 50px 10px;
}
```

Top padding is 20px

Right/Left padding is 50px

Bottom padding is 10px

SHORTHAND VERSION

with 4 values

```
div {
  margin: 0 0 1rem 1rem;
}
```

Trouble remembering?

Top - **R**ight - **B**ottom - **L**eft

UNITS OF MEASURE

- **px**: A fixed (aka absolute) value in pixels
- **em**: Relative to the font-size of the element (e.g., 2em = 2 x the size of the current font)
- **rem**: Relative to the root element font-size
- **vh**: % of the viewport height (e.g., 50vh = 50% of the viewport height)
- **vw**: % of the viewport width
- **vmax**: % of viewport's larger dimension
- **vmin**: % of viewport's smaller dimension
- **%**: It depends

HEIGHT & WIDTH

Height and width can only be applied to **block** elements.

```
div {
    height: 50vh;
    width: 50vh;
}
span {
    height: 50px; /* HAS NO EFFECT */
}
```

BOX-SIZING: CONTENT-BOX

```
div {
  border: 2px solid;
  padding: 50px;
  width: 200px;
  box-sizing: content-box;
}
```

CONTENT-BOX (DEFAULT)

PADDING LEFT: 50PX

PADDING RIGHT: 50PX

BORDER LEFT: 2PX

BORDER RIGHT: 2PX

+ WIDTH (CONTENT): 200PX

TOTAL SIZE: 304PX

HEADS UP: content-box is the default value.

BOX-SIZING: BORDER-BOX

```
div {
  padding: 50px;
  border: 2px solid;
  width: 200px;
  box-sizing: border-box;
}
```

BORDER-BOX

WIDTH (TOTAL): 200PX

- BORDER-LEFT: 2PX
- BORDER-RIGHT: 2PX
- PADDING LEFT: 50PX
- PADDING RIGHT: 50PX

CONTENT: 96PX

HEADS UP: border-box is the preferred value.

FIXING THE BOX MODEL

We use a **reset** to apply the box-sizing to everything with the `*` universal selector.

```
/* This special selector means apply this to everything! */

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

MARGINS, PADDING & BORDERS

COLORS

RED × GREEN × BLUE

16,777,216 possible color combinations

COLOR VALUES IN CSS

- Keyword
- Hexidecimal
- RGB & RGBA
- HSL & HSLA

HEADS UP: Alpha channel range is 0 (transparent) to 1 (opaque)

```
/* Keyword Syntax */

h1 {
  background-color: gray;
}

/* RGB & HSL Syntax */

p {
  color: rgba(0,0,0,1);
  border: 2px solid hsl(0,0%,0%);
}

/* Hexadecimal Syntax */

div {
  background: #ff0000;
}
```

IMAGES

ADDING IMAGES TO YOUR PAGE

- Images can be added with the `<img>` tag in HTML or in the background via CSS
- Background images are design elements only and are ignored by screen-readers

CONTENT IMAGES

- The `<img>` tag has no closing tag
- The `src` attribute links the file and is required
- The `alt` attribute is used by screenreaders and for SEO

```
<img src="images/dogs.jpg" alt="A picture of my dogs">
```

LINKING FILES RELATIVELY

index.html → logo.png:

```
<img src="imgs/logo.png">
```

1 Start in the folder where the current file is

2 Go into the folder called **imgs**

3 Get the file called **logo.png**

IMAGES

BACKGROUND IMAGES

Background images are added through CSS

```
/* Use the background-image or
   shorthand background property */

selector {
  background-image: url('path/to/file');
}
```

HEADS UP: FYI... multiple background images are stackable!

LINKING FILES RELATIVELY

styles.css → logo.png:

```
div {
  background-image: url("../imgs/logo.png");
}
```

2 3 4

1. Start in the folder where the current file is
2. Go back one folder (to **website**)
3. Go into the **imgs** folder
4. Get the **logo.png** file

BACKGROUND PROPERTIES

- **background-repeat**: tile the image or place once
- **background-position**: position the image in its containing element
- **background-attachment**: scroll with the page or remain fixed in one place
- **background-size**: the size of the image in the background
- **background-clip**: crop the image at the content-box, border-box or padding-box
- **background-origin**: place the image origin at the content-box, border-box or padding-box

BACKGROUND IMAGE SYNTAX

```
header {
  background-image: url('../images/bg.jpg');
  background-size: cover;
}

section { /* Multiple Backgrounds */
  background-image: url('../images/logo.png'),
                    url('../images/bg.jpg');
  background-repeat: no-repeat, repeat;
  background-position: bottom right, top left;
}

main { /* Shorthand Format */
  background: url('../images/logo.png')
              no-repeat
              bottom right / 30%
              fixed;
}
```

BACKGROUND IMAGES

IMAGES

Lab 3:

LAB TASKS

https://github.com/jmeade11/crashcourse/labs

1. Grandma is pretty happy but thinks her recipe page could use some pizzaz. Open the index.html and the styles.css files in the **lab3** folder.
2. Add **1rem** of space between the body border and content (remember your box model 😊).
3. The body has a default margin of 8px. Change that to **1rem** as well.
4. Insert the cookie image into the page above the first heading.
5. Set the image **width to be 100%** so it scales to fit the width of its container.

PHEW...YOU MADE IT

Break time!!!

PART 3

QUICK REVIEW

- What is one type of color value?
- Can you describe the box model?
- What tag do you use to add an image to your HTML?

PART 3: OBJECTIVES

- Use classes, IDs, and combinators to apply styles
- Understand how styles inherit/cascade
- Describe how conflicting styles are applied

CLASSES & ID

CSS SELECTORS

1. Element Tags
2. **Classes & IDs**
3. Combinators
4. Attributes
5. Pseudo Classes

CLASSES & IDS

- Classes and IDs give us more flexibility to target elements on our page and apply styles to them
- They also make it easier for us to develop and manage our webpages
- You can combine them with other selectors

IDS

- An ID name is unique. It may only be used once on a page.
- An element may only have one ID

```
<div id="extra-special">
```

```
#extra-special {
...
}
```

CLASSES

- Classes are reusable as many times as you want
- An element can have as many classes as you want

```
<div class="big primary">
```

```
.primary {
  ...
}
.big {
  ...
}
```

CLASSES & IDS

COMBINATORS

Document Object Model

DOM: A WEBPAGE FAMILY TREE

USING COMBINATORS

- **Descendant** *space* : Applies to any matching descendants (**any number of levels below**) of the first selector.
- **Child** `>`: Applies only to matching children (only **one level below** the first selector) of the first selector.
- **General Sibling** `~`: Applies to any element that is a sibling of the first element, as long as it appears **after** the first element in the source order.
- **Adjacent Sibling** `+`: Must be the **very next sibling** in the source order following the first selector.

COMBINATORS IN ACTION

```
/* targets paragraphs that are descendants of divs */

div p {
  color: red;
}

/* targets paragraphs that are direct children of sections */

section > p {
  color: blue;
}

/* targets paragraphs are adjacent siblings of another paragraph */

p + p {
  background-color: yellow;
}

/* targets paragraphs are siblings of divs */

div ~ p {
  background-color: orange;
}
```

HEADS UP: It's always the **last** element that we're targeting.

USING COMBINATORS

CASCADING & INHERITANCE

CASCADING & INHERITANCE

- Many CSS properties inherit their values from their ancestors and most accept a value of **inherit**
- Properties can be overridden when you provide a rule that has **more specificity**
- Any rules that are **not specifically overridden** continue to be inherited

SPECIFICITY

Specificity is a calculation based on the selectors used. Each type of selector is weighted:

1. Inline Styles (highest)
2. IDs
3. Classes & Attributes
4. Element Tags (lowest)

CALCULATING SPECIFICITY

	inline	id	class	element
`#main-nav > a.nav-item { background-color: red; }`	0	1	1	1
`header > nav .nav-item { background-color: blue; }`	0	0	1	2
`header nav#main-nav > a.nav-item { background-color: green; }`	0	1	1	3

WHEN SPECIFICITY MATTERS

- When there are multiple rules that contradict one another, the specificity is calculated to determine which is applied.
- For identically weighted rules, the last declaration for wins!
- Inline styles trump **almost** all others.
- One exception is the special `!important` attribute.

GUESS WHICH?

CSS CLASSES & IDS

Lab 4:

LAB TASKS

https://github.com/jmeade11/crashcourse/labs

1. Open the index.html and styles.css files in **lab4** folder.
2. The h1 is centered, but we want to also **center** the h4 tag that credits grandma and the paragraph with the link at the bottom of the page.
3. Let's group each of the main areas of the recipe (ingredients, instructions, and nutrition) inside a set of section tags.
4. Give each one a corresponding **id**.
5. Make the list of ingredients only **italic**.

ANOTHER ONE DOWN!

PART 4

QUICK REVIEW

- What does cascade mean?
- What selector is weighted highest?
- How can you override a style attribute on a tag?

PART 4: OBJECTIVES

- Use floats to have more control over your layout
- Be aware of more modern methods for layout control
- Review some helpful shorthand properties

CSS FLOATS

WHAT IS FLOAT?

Float places an element on the **left** or **right** of its container and allows other elements to wrap around it.

```
/* Float accepts left, right or none
   The none value is the default   */

.my-floated-element {
    float: right;
}
```

CLEARING THE FLOAT

The **clear** property prevents elements from sitting next to the floated elements that precede it — forcing the cleared element below the floated element.

```
/* Clear accepts left, right or both */

.my-cleared-element {
    clear: both;
}
```

FLOAT EXAMPLES

HTML CSS

LIVE

```
1 div {
2   height: 100px;
3   width: 100px;
4   background-color: rgba(255,0,0,.25);
5   border: 5px solid #e41a23;
6   float: right;
7 }
```

Result

FORK ON CODEPEN

Lorem ipsum dolor sit amet consectetur adipisicing elit. Eius id ipsa vel sint exercitationem ipsum veniam incidunt asperiores quis. Distinctio aliquid qui repellat incidunt dolorem officiis at numquam deleniti quo?

Lorem ipsum dolor sit amet consectetur adipisicing elit. Ullam officiis possimus debitis? Iure in animi id, optio illum nihil, hic facilis similique nam nesciunt nostrum nisi ullam est sint itaque!

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Reprehenderit, nulla. Quis at cupiditate nostrum distinctio soluta saepe molestias quam voluptate, officia ullam laudantium reprehenderit quia aspernatur et quaerat nemo tenetur.

FLOATS

Lab 5:

LAB TASKS

https://github.com/jmeade11/crashcourse/labs

1. Open the **lab5** folder.
2. You'll need to set both the ingredients and instruction sections to be **50% in width** and then float them both to the **left**, so they fit next to one another.
3. Make sure you address the nutrition section so that it **doesn't float** with the other sections.
4. Lastly, let's give the nutrition section some additional padding, but just on top! Make it **20px**.

GO BUILD AWESOME THINGS!