CSS Basics

Learning Objectives

- Review HTML
- Predict and apply relative and absolute paths to images and links
- Describe the DOM and draw a simple DOM tree
- Apply and explain CSS "cascade", including specificity and inheritance
- Experiment with CSS (things like margin, border etc.)

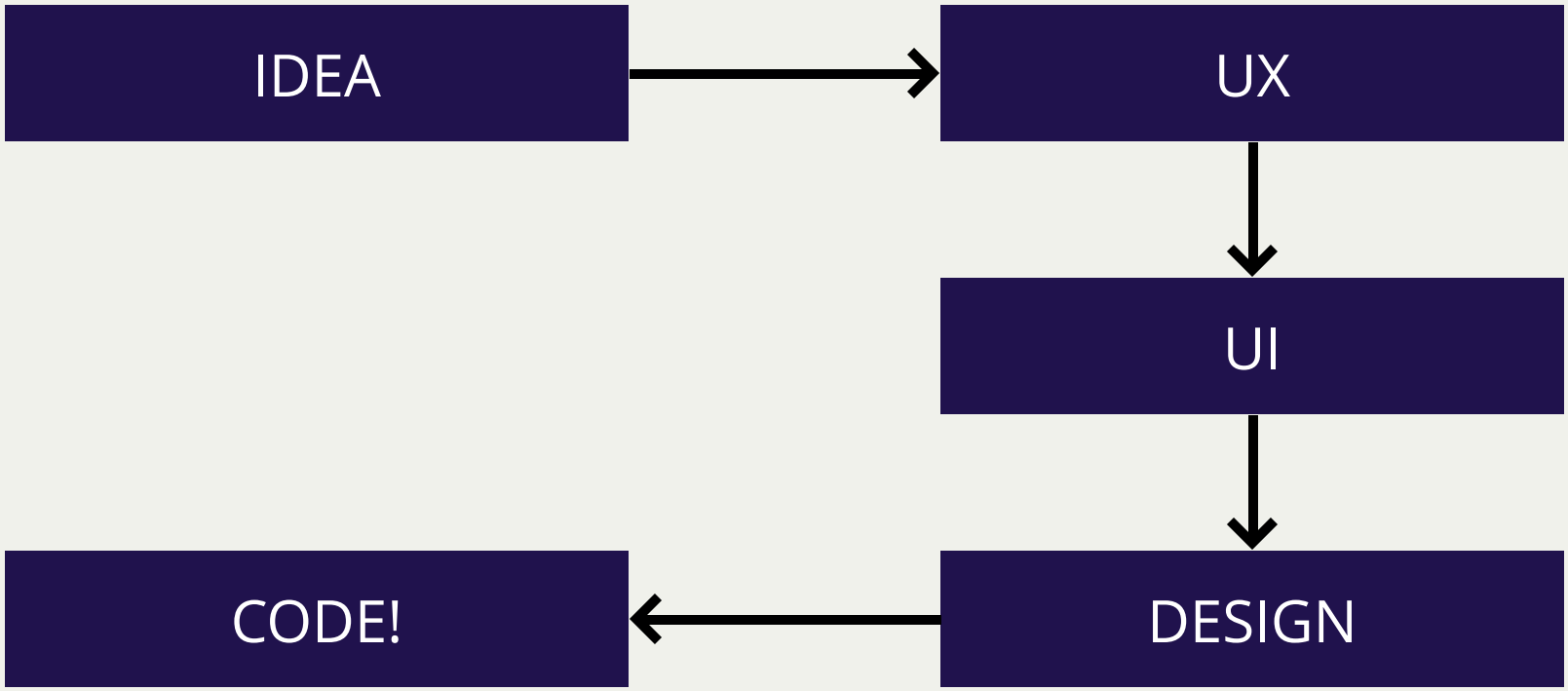
Agenda

- HTML Review
- Working with Absolute and Relative Paths
- The DOM
- CSS History and Syntax
- CSS Selectors and Colors
- Building a website
- Lab time
- Questions and Answers

A Brief Interlude...

- Homework
- Projects
- Office Hours

The Ideal Workflow



Wireframing?

- Search for inspiration
- Gather content for your site
  - Group data
- Define your grid
- Create layout with boxes
- Define information hierarchy

Wireframing?

- Marvel
- Axure
- Invision
- Axure
- Balsamiq
- Sketch
- Illustrator || InDesign || Photoshop
- Wireframe.cc
- Pen and paper

HTML Review

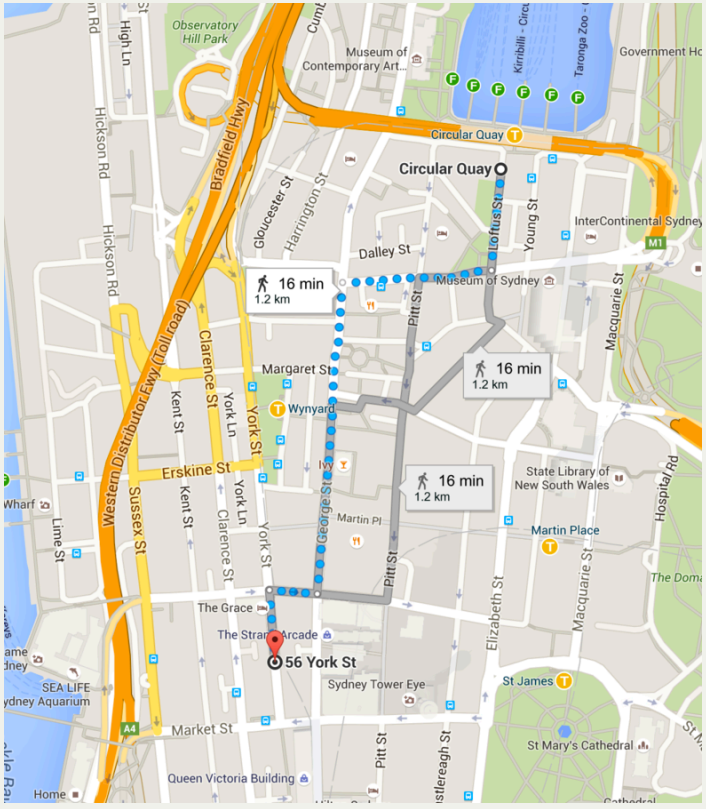
Some Questions

- What does HTML do?
- What makes up an HTML document?
- What is an HTML element?
- What is an HTML element made up of?
- Does every element need a closing tag?
- What do attributes do?
- How would you create a link in an HTML page?

Paths

Absolute Paths

Relative Paths



Paths

Absolute Paths

```
<img src="https://fillmurray.com/200/200">
<img src="/Users/username/GeneralAssembly/absolute.png">
```

Relative Paths

```
<img src="relative.jpg">
<img src="images/relative.jpg">
<img src="../relative.jpg">
<img src="../assets/relative.jpg">
<img src="../../assets/relative.jpg">
```

Exercise!

Muck around with Paths!

- Look through the file structure
- Add images to the page
- Bonus:
  - Add information to the images (alt and title)
  - Link the HTML files
  - Style all images to have a width of 300px

Image Types

PNG - Transparency

JPG - No transparency

GIF - Animatable, and transparency

SVG - Transparency, Responsive, Small, Animatable

Plus, others

CSS

CSS - The Style (Clothes)

- **C**ascading **S**tyle **S**heets
- Currently at version 3
- You apply properties and values to HTML. It's a presentation language

```
h1 {
  font-family: "Comic Sans";
  color: hotpink;
}

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

CSS - The Style (Clothes)

- A CSS **Rule** is made up of:
  - A **selector**
  - Curly brackets
  - **Declaration**(s)
- A CSS **Declaration** is made up of:
  - **Property**
  - **Value**

CSS - The Style (Clothes)

```
selector {
  /* DECLARATION */
  property:  value;
}

body {
  background-color: limegreen;
  color: chartreuse;
}
```

CSS - The Style (Clothes)

You reference HTML with CSS

```
<!-- THE HTML -->

<h1>Hello World</h1>


<p>Angle brackets - ugh!</p>
```

```
/* THE CSS */

h1 {
    font-size: 25px;
}

p {
    font-style: italic;
}
```

CSS - The Style (Clothes)

How do we add styles to our page?

Three ways:

- ~~Inline Styles~~
- ~~Style Tag~~
- External File ✔

CSS - The Style (Clothes)

Adding an external CSS file

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

CSS - The Style (Clothes)

Adding a style tag

```
<!DOCTYPE html>
<html>
<head>
    <title>Basic web page</title>
    <style>
        h1 {
            color: #FFF;
        }
    </style>
</head>
<body>

</body>
</html>
```

CSS - The Style (Clothes)

Adding inline styles

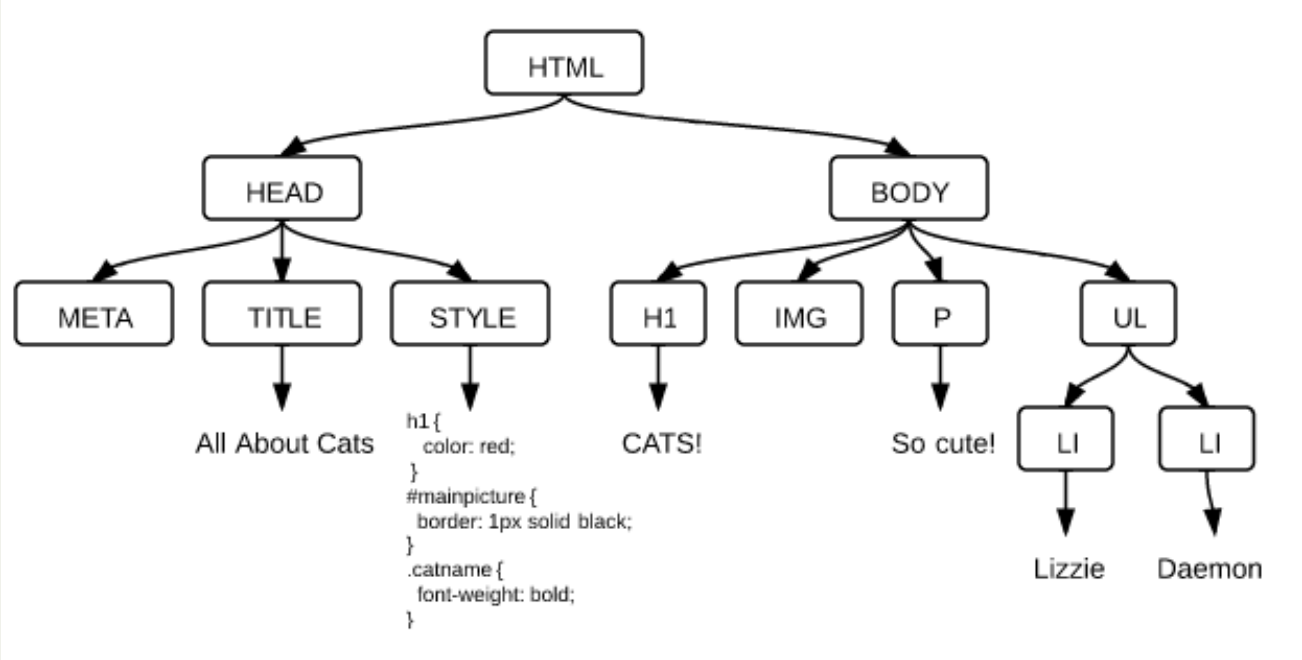
```
<p style="font-size: 20px; color: hotpink">
    Ahhhhhh
</p>
```

DOM

Document Object Model

- The browser's view of our HTML
- Defines how we access and manipulate our HTML
- It's represented as a tree

Document Object Model



Document Object Model

- **Node**
- **Parent Node**
- **Child Node**
- **Sibling Node**

Exercise!

Look at your GA press release

What would the DOM tree look like?

CSS - The Style (Clothes)

Adding inline styles

```
<h1 style="color: #FFF">Page Title</h1>
```

CSS - The Style (Clothes)

Color Names

```
h1 {
  color: red;
}

h2 {
  color: rebeccapurple;
}

h3 {
  color: papayawhip;
}

h4 {
  color: lightgoldenrodyellow;
}
```

CSS - The Style (Clothes)

Hexadecimal Colors

```
h1 {
  color: #000000;
}

h2 {
  color: #00ff00;
}

h3 {
  color: #12facd;
}

h4 {
  color: #123;
}
```

CSS - The Style (Clothes)

RGB Colors

```
h1 {
  color: rgb(0, 0, 0);
}

h2 {
  color: rgb(12, 123, 255);
}

h3 {
  color: rgb(255, 255, 255);
}

h4 {
  color: rgb(0, 255, 0);
}
```

CSS - The Style (Clothes)

RGBA Colors

```
h1 {
  color: rgba(0, 0, 0, 0.0);
}

h2 {
  color: rgba(12, 123, 255, 0.3);
}

h3 {
  color: rgba(255, 255, 255, 0.6);
}

h4 {
  color: rgba(0, 255, 0, 1.0);
}
```

CSS - The Style (Clothes)

HSL and HSLA Colors

```
h1 {
  color: hsl(0, 100%, 50%);
}

h2 {
  color: hsl(110, 100%, 52%);
}

h3 {
  color: hsla(110, 100%, 52%, 0.0);
}

h4 {
  color: hsla(0, 100%, 50%, 1.0);
}
```

Cascading?

- Like a waterfall
- If you style a parent element
  - You will style it's children!
  - Though, CSS specificity can overwrite this

Specificity?

- 1 point for an elements name
- 10 points for a selection based on a class
- 50 points for a selection based on an ID

For more:

Pop Culture

Smashing Magazine

CSS Tricks

CSS Selectors

CSS Selector: Tag

```
h1 {} /* All h1 tags */

p {} /* All p tags */

a {} /* All a tags */

ul {} /* All ul tags */
```

CSS Selector: Descendant

```
ol li {} /* All li tags, within ol tags */

p a {} /* All a tags, within p tags */

header h1 {} /* All h1 tags, within header tags */

footer ul li {} /* All li tags, within ul tags (within footer tags) */
```

CSS Selector: Child

```
ol > li {} /* li tags directly within ol tags */

p > a {} /* a tags directly within p tags */

header > h1 {} /* h1 tags directly within header tags */

footer > ul > li {}

/*
li tags directly within ul tags (that are directly within footer tags)
*/
```

CSS Selector: Class

```
a.importantLink {} /* a tags with the class of importantLink */

img.bill {} /* img tags with the class of bill */

ul li.highlight {}
/* lis with the class of highlight, within uls */

nav .currentLink {}
/* any element with the class of currentLink with navs */
```

CSS Selector: ID

```
a#importantLink {} /* a tags with the ID of importantLink */

img#bill {} /* img tags with the ID of bill */

ul li#highlight {}
/* lis with the ID of highlight, within uls */

nav #currentLink {}
/* any element with the ID of currentLink with navs */
```

Classes vs. IDs

- Both are attributes used to identify specific elements
- Classes can be used as many times as you want
- Each ID can only be used once on a page, and they are very specific

CSS Selectors

We use CSS Selectors to identify specific elements

We use CSS Selectors to reuse styles

Exercise!

CSS Diner

- Get through as much as you can
- If you get it done, post a screenshot of you being called a CSS Ninja

Quiz!

```
p {}

header h1 {}

a.important.link {}

img#bill {}

.fullWidth {}

footer > ol > li {}

h3, h4 {}

* {}
```

Coding Style!

- Indent each declaration
- Only be as specific as you need to be
- One selector per line
- Comment for clarity
- Aim for reusability
- CSS Guidelines, Google's CSS guidelines and MDO's CSS guidelines

Homework

- Practice HTML and CSS
- e.g. Create a resume for Bill Murray
  - Use everything you know
- Finish off CSS Diner
- Try to get through this
- Start working on your project
- It will save you lots of time if you go through some of this

Q & A

GENERAL ASSEMBLY

Feedback Time

Lesson 2: **CSS Basics**

https://ga.co/fewd32syd

Thanks!