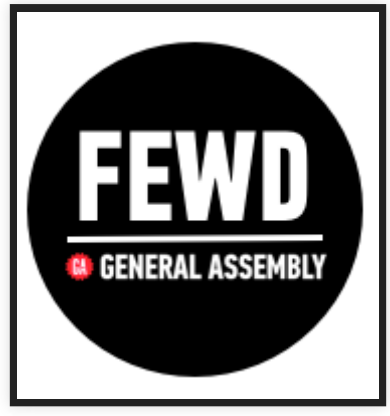


LESSON 2 - CSS BASICS (AKA HTML 2.0)

LESSON 2 AGENDA

- Introduce Learning Objectives
- Review Lesson 1 - HTML & External Style Sheets
- Understanding the DOM
- Building A Simple Web Page
  - Images
  - Nav
  - CSS
  - Colors
  - Linking To Other Pages
- Lab Time

CLASSROOM CULTURE & VALUES

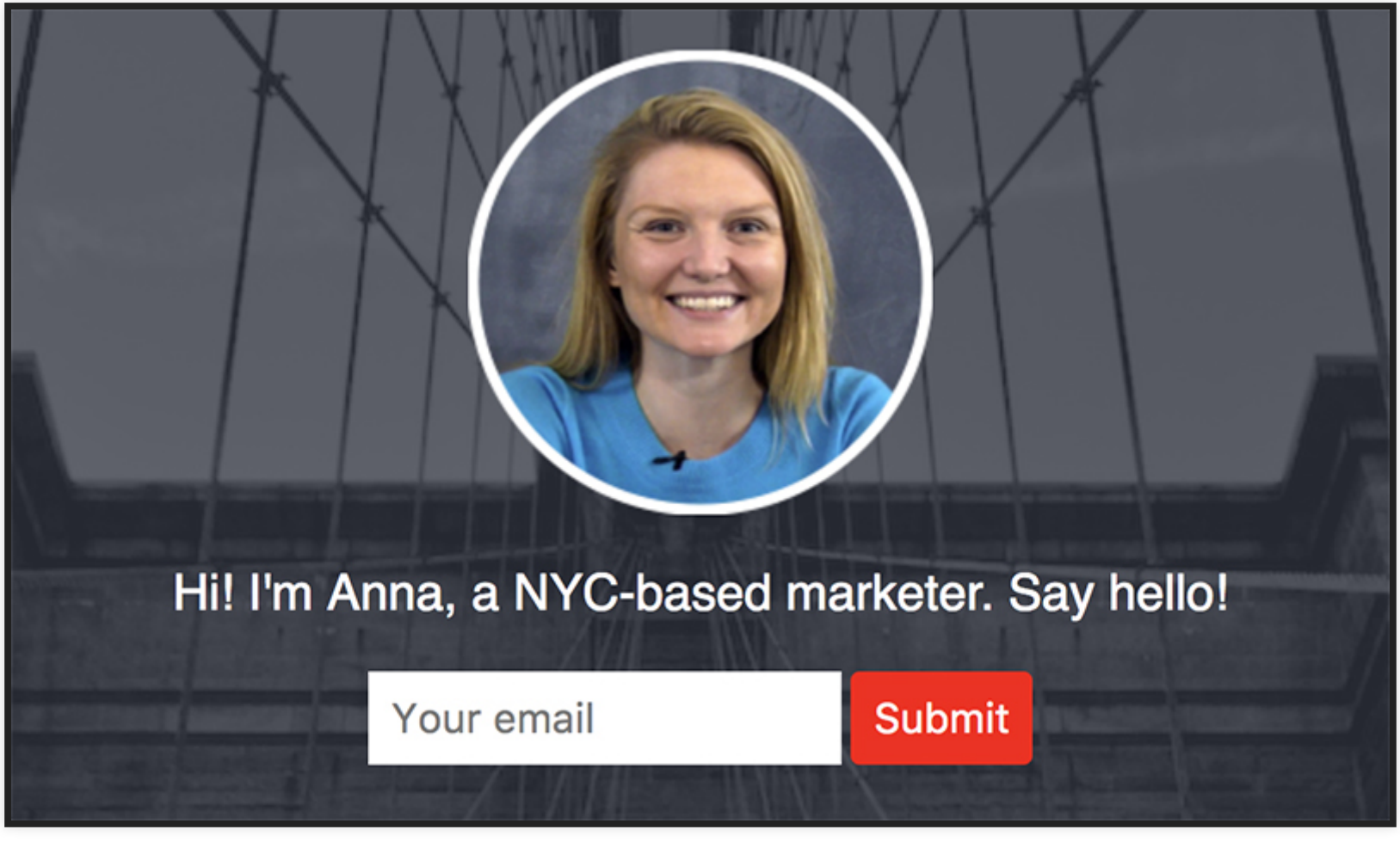
- Be present and involved
- Be respectful of other people's time
- Honor your commitments
- Be patient!
- Step out of your comfort zone
- Share your Knowledge

LEARNING OBJECTIVES:

AFTER TODAY, YOU SHOULD BE ABLE TO...

- Describe the DOM and draw a simple DOM tree
- Predict image paths and apply relative paths to `<img>` and `<a>` tags.
- Apply and explain CSS "cascade" including: importance, specificity and inheritance.
- Experiment with CSS Colors and margin/padding

HTML BASICS REVIEW


Hi! I'm Anna, a NYC-based marketer. Say hello!
Your email
Submit

KIDDING...

The Best Chocolate Chip Cookies

Recipe by: My Grandma

Prep Time

45 Min

Ingredients

- 1 1/2 cup (3 sticks) butter, softened
- 1 cup brown sugar
- 1 cup granulated sugar
- 1 Tbl vanilla instant pudding powder
- 2 Tbl milk
- 2 Tbl vanilla extract
- 2 eggs
- 4 cups all purpose flour
- 2 tsp baking soda
- 1/2 tsp salt
- 4 cups chocolate chips
- 1 cup chopped walnuts or pecans (optional)

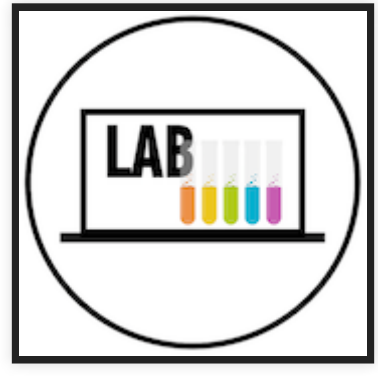
How to make Mama's Recipe: The Best Chocolate Chip Cookies

1. Preheat oven to 350 degrees
2. Beat butter and sugars together until light and fluffy
3. Stir in pudding mix milk and vanilla extract.
4. Beat in eggs.
5. Add dry ingredients and stir until combined.
6. Stir in chocolate chips and nuts.
7. Place 1 1/2 inch balls of dough 2 inches apart on an un greased cookie sheet.
8. Bake 8-10 minutes or until golden brown.

Nutrition Information

Probably bad for you, but who cares. MMMMMM COOKIES!!!! nom nom nom

This recipe was altered from Open Source Recipe.



WHAT TAG IS IT

About Me | Portfolio

JOE JOHNSON

I ENJOY LIFE AS A DEVELOPER

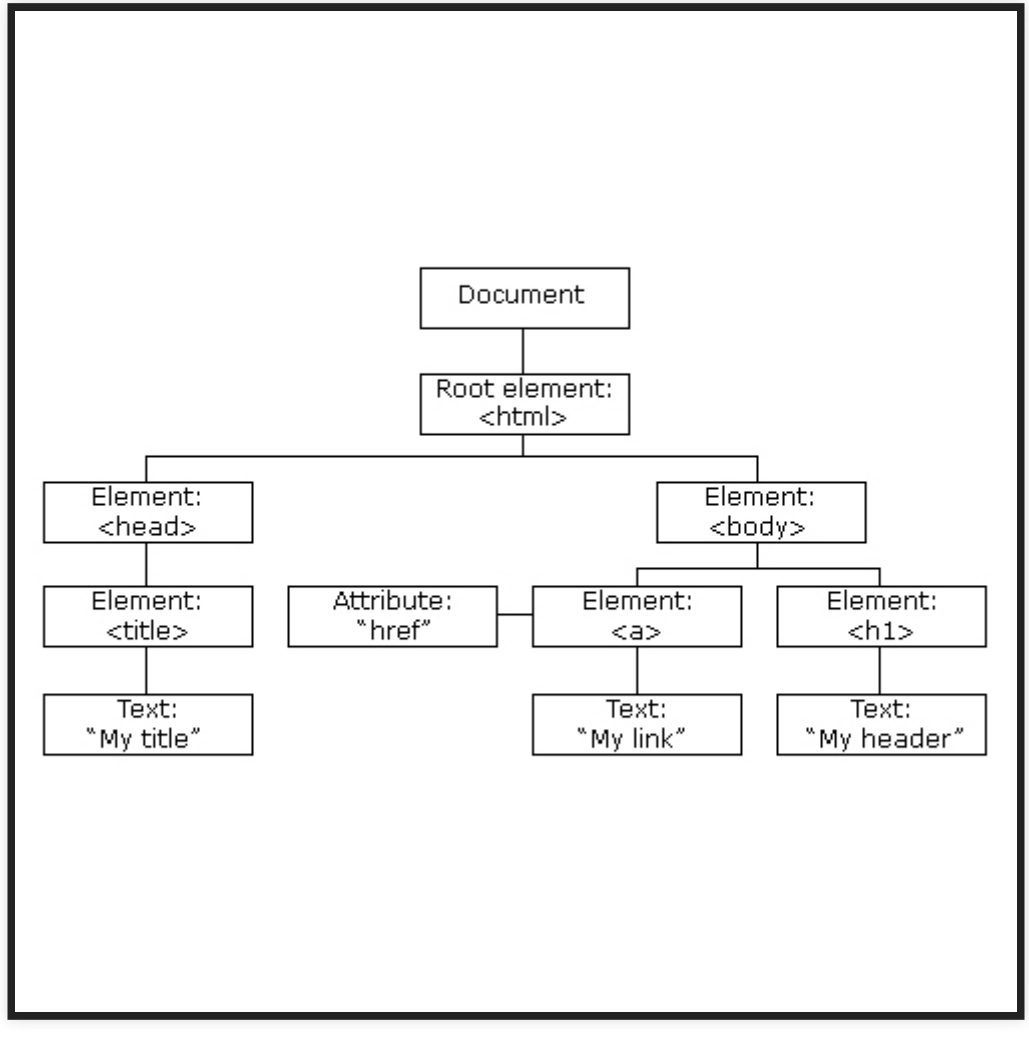
I'm Joe Johnson, a Developer based in NYC. I Have ten years of experience in the graphic design world, specializing in the creation of responsive websites.

Facebook | Twitter | Instagram | LinkedIn

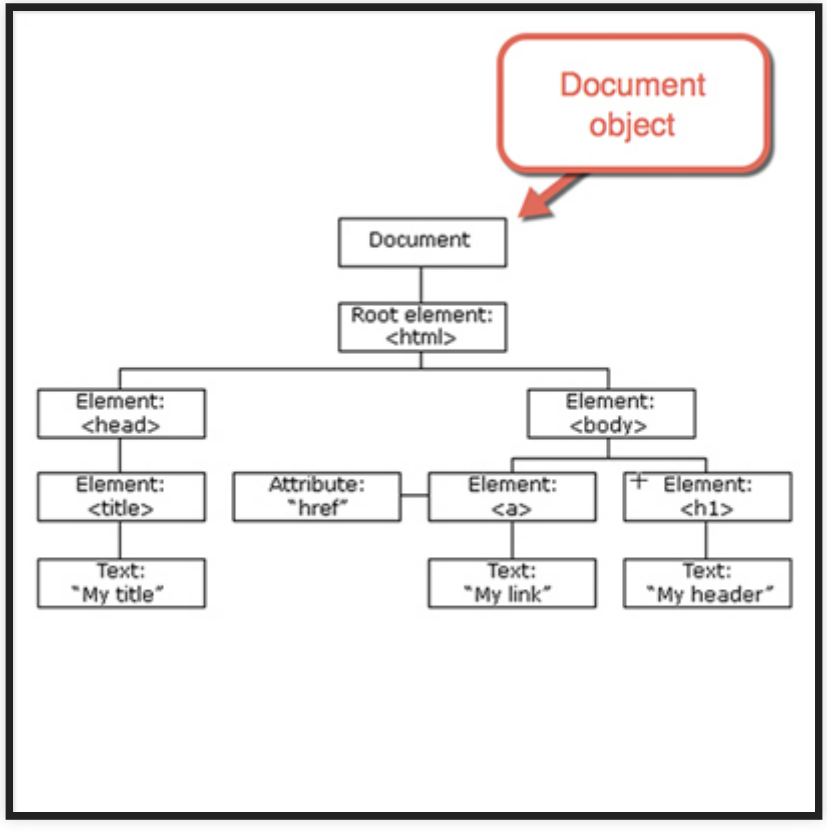
UNDERSTANDING THE DOCUMENT OBJECT MODEL (DOM)

WHAT IS THE DOM?

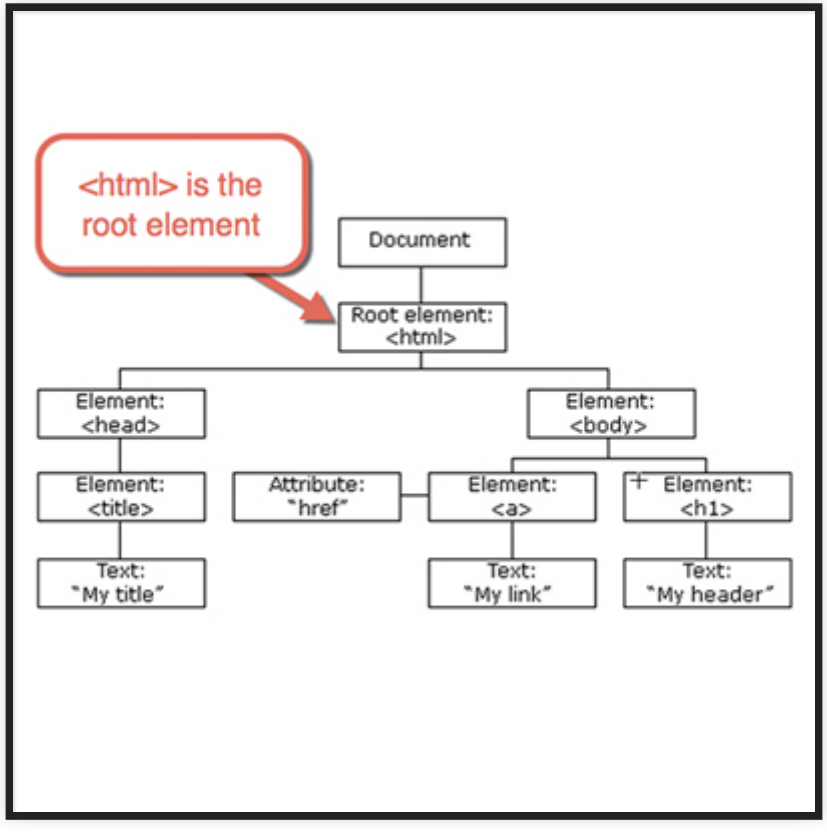
- When a web page is loaded, the browser renders a **Document Object Model (DOM)** of the page in HTML.
- The DOM model is constructed as a tree of Objects


Document
Root element:
<html>
Element:
<head>
Element:
<body>
Element:
<title>
Attribute:
“href”
Element:
<a>
Element:
<h1>
Text:
“My title”
Text:
“My link”
Text:
“My header”

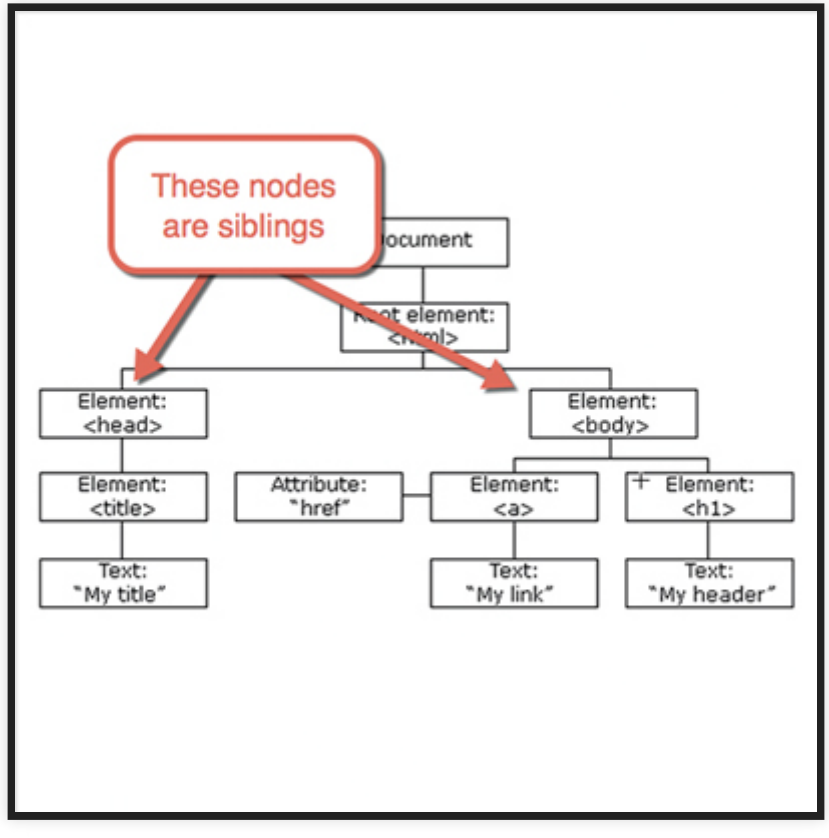
At the top we have the **document object**



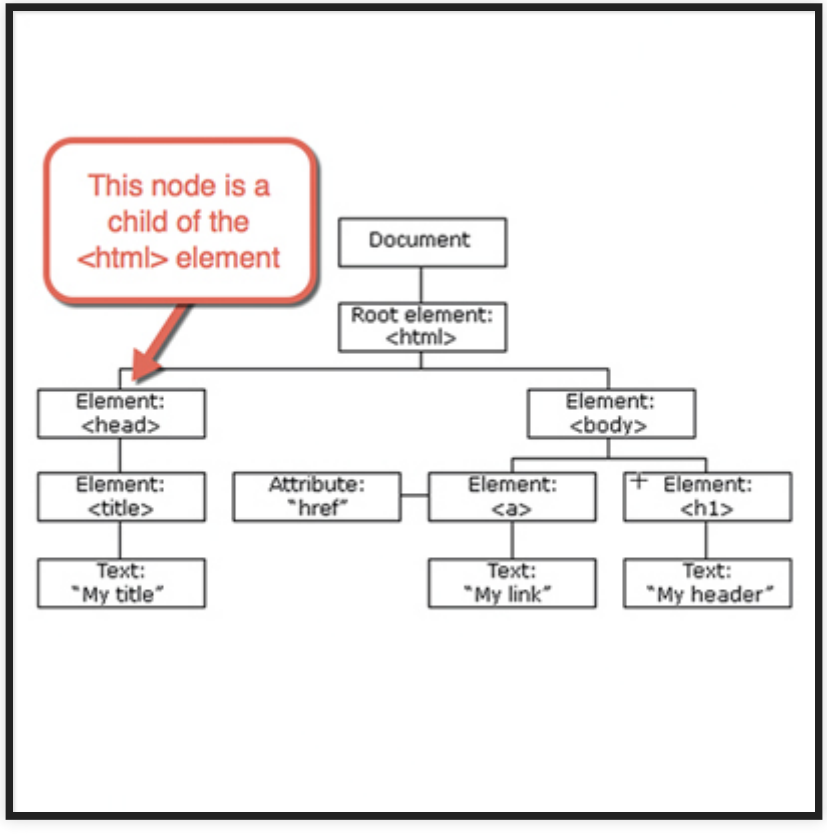
Next, we have the **root element** which is always the `<html>` tag



Then comes the <head> and <body> elements. Notice they are on the same level -- this makes them siblings



Also, notice this makes them both the first children of the `<html>` element

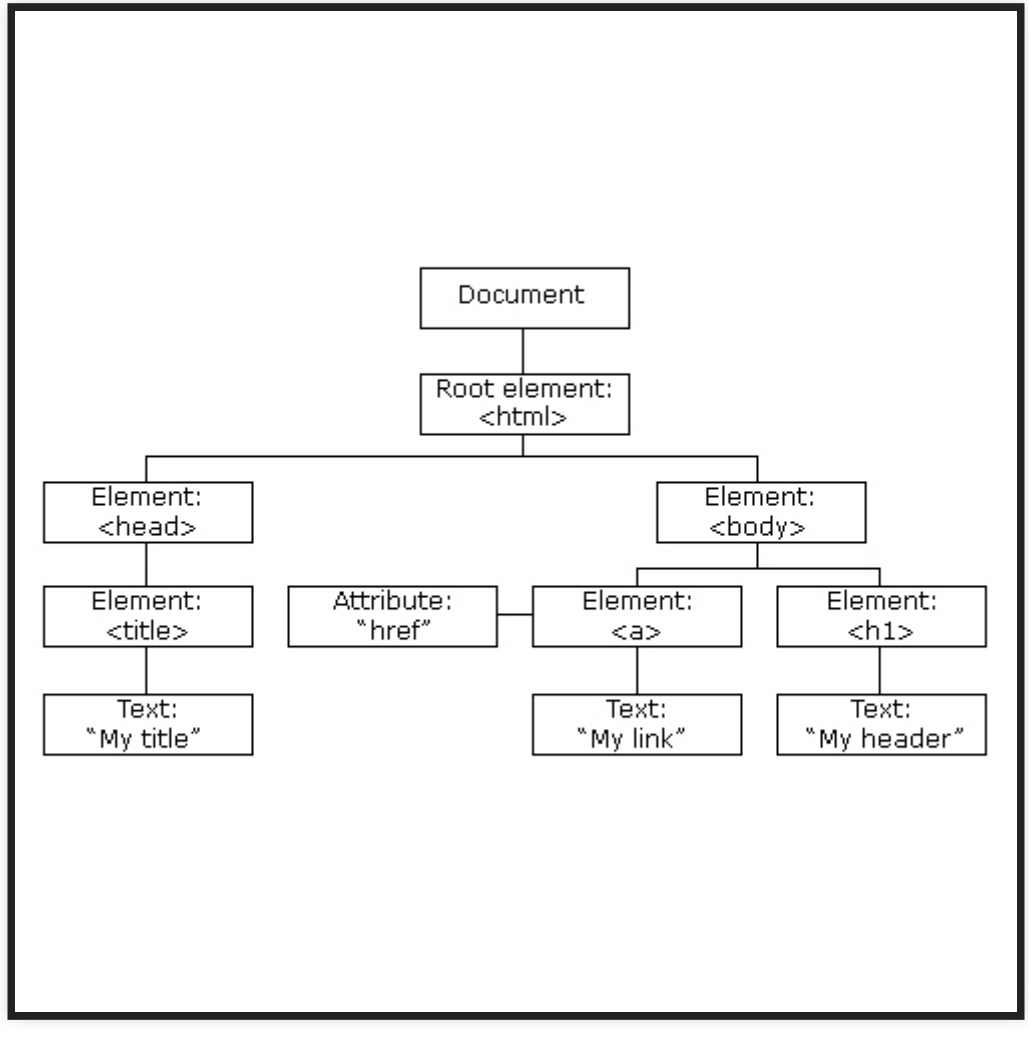


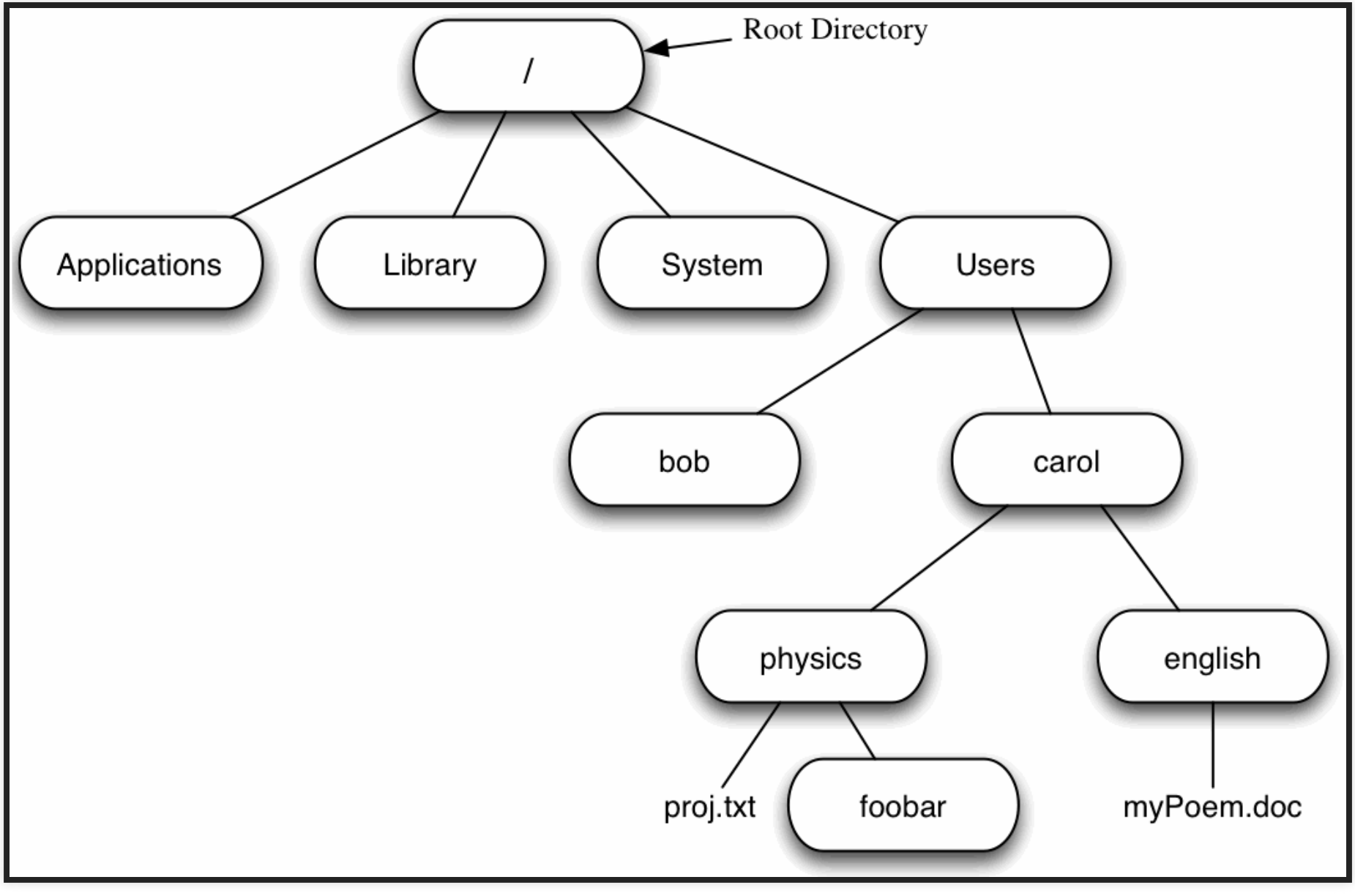
And so on, for all of the subsequent HTML elements in the web page...

BASICALLY...

The DOM is a family tree four your webpage

Where else have you seen a similar structure?




Root Directory
/
Applications
Library
System
Users
bob
carol
physics
english
proj.txt
foobar
myPoem.doc

WHY SHOULD I CARE?

- Parent/child relationship of HTML elements
- CSS specificity and inheritance
- Major portion of Javascript interaction
- Readability of HTML Code (indentation)

PULSE CHECK

Let's determine the DOM model for the "About Me" page from earlier...

BUILDING WEBSITES - IMAGE TAGS

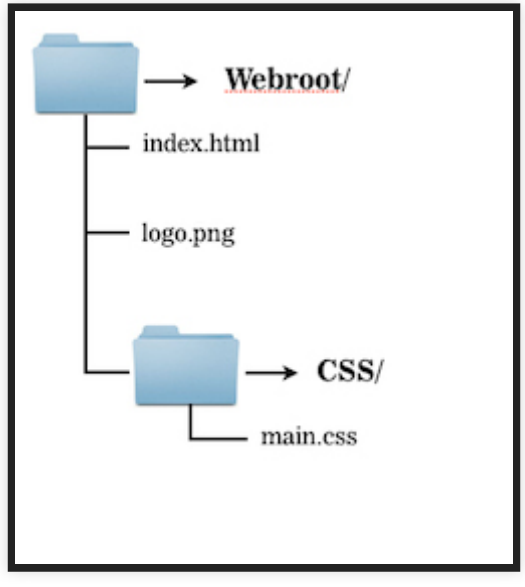
HTML Images are placed using the <img> tag:

```
<img src="folder/image_name.jpg"
     alt="alternative text">
```

The `<img>` tag requires a `src` attribute, which tells the browser where to find the image.

The `alt` attribute is optional, but strongly encouraged for a number of reasons

CHECK: How might you write the `src` path to access *logo.png* from *index.html*?



There are 2 different ways to specify the path to an image...

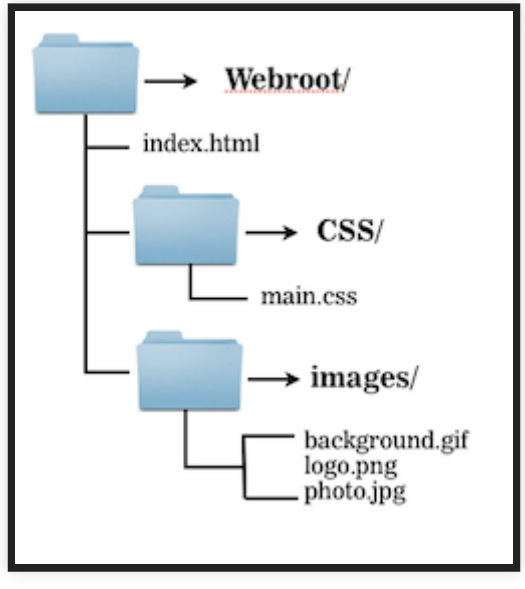
RELATIVE VS ABSOLUTE

RELATIVE PATH

As long as the image you are pointing to is inside your `webroot`, you can use a relative path:

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

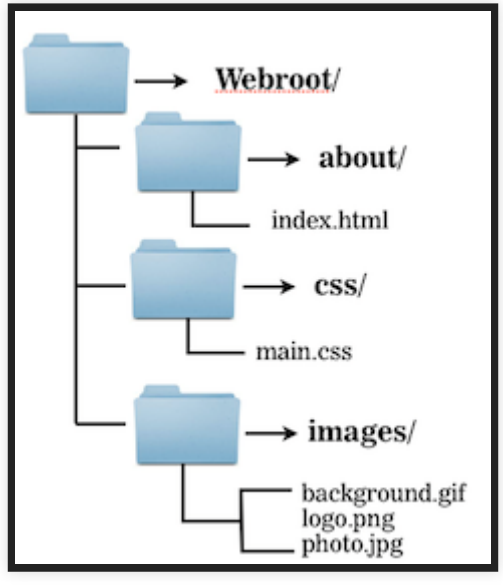
What about this example; how might you write the `src` path here?



Since it is in a subfolder in relation to `index.html`, we need to include that subfolder in our path:

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

Last one -- can anyone figure out the `src` path for this one:



In this case, the HTML file is in its own subfolder like the image.

When this happens, you need to go up a directory, and then go down into the subfolder containing the image.

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

To go up a directory, we use `..`

It can also be used repeatedly to go up multiple directories

For example, `../../..` would go up three directories

ABSOLUTE PATH

An Absolute path is a full URL that and is required when the file is located on an external domain.

It can also be used in place of relative URLs on your own `webroot`, but its not required

```
<img
src="http://www.example.com/images/logo
```

PULSE CHECK

What are some issues we could run into when using relative vs absolute paths?

BUILDING WEBSITES - IMAGE FILE FORMATS

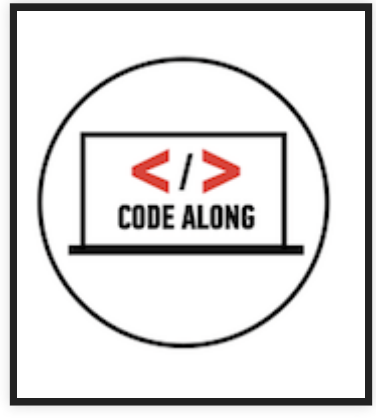
There are three main image file formats:

.JPEG / .JPG

.GIF

.PNG

15 MINUTE BREAK



ABOUT ME STYLES

About Me | Portfolio

JOE JOHNSON

I ENJOY LIFE AS A DEVELOPER

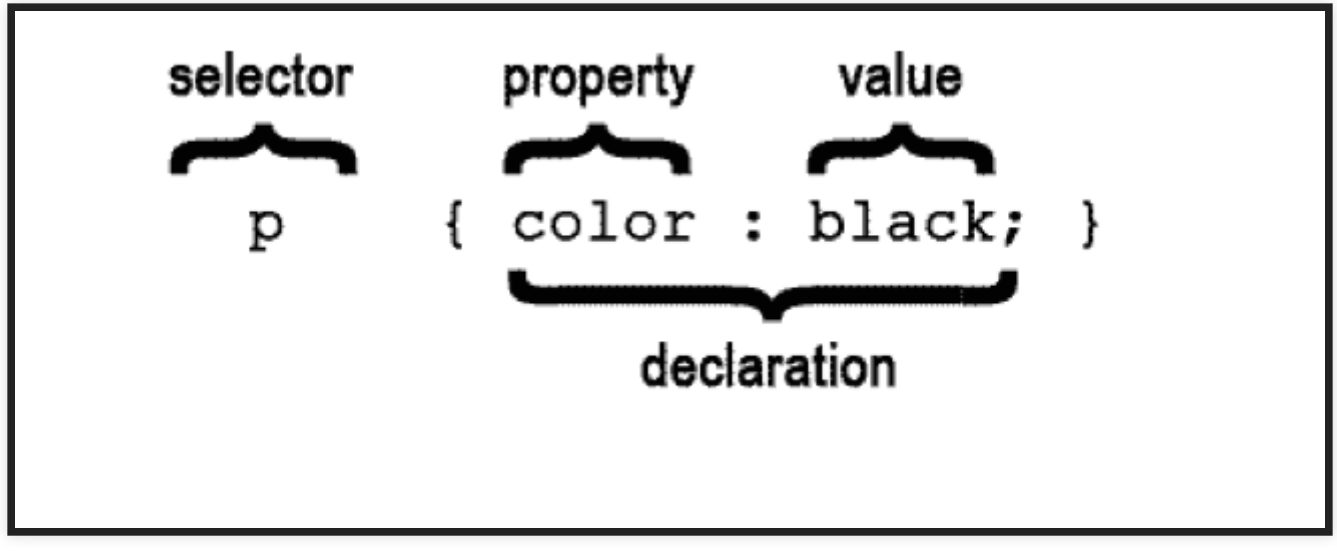
I'm Joe Johnson, a Developer based in NYC. I Have ten years of experience in the graphic design world, specializing in the creation of responsive websites.

Facebook | Twitter | Instagram | LinkedIn

BUILDING WEBSITES - CSS

WHAT IS CSS?

CSS Stands for **Cascading Style Sheets**



CSS RULE

Below is a sample CSS **rule** for the <p> HTML element.

It consists of the element **selector** -- p -- and is followed by a set of **declarations** in a **declaration block**.

```
p {
    color: red;
    font-weight: bold;
}
```

CSS SELECTOR

The **selector** -- `p` in this case -- specifies what parts of the HTML document should be styled by the declaration.

This selector will style all <p> elements.

CSS DECLARATION BLOCK

Everything between the open curly bracket: { and closing curly bracket: } is considered the **declaration block**

The declaration block is made up of individual **declarations**

```
{
    color: red;
    font-weight: bold;
}
```

CSS DECLARATIONS

This example has two declarations. Here's the first:

```
color: red;
```

PULSE CHECK

Now, let's look at the second declaration:

```
font-weight: bold;
```

What style **property** are we specifying here, and what **value** are we setting it to?

WHERE DOES CSS GO?

- Inline
- In the `head` (internal or embedded CSS)
- **In a separate file (external CSS)**

INLINE CSS

Inline CSS is when the style **declarations** are placed directly in the specific HTML element via the `style` attribute:

```
<p style="color: red; font-weight: bold;">
```

INTERNAL CSS

Internal CSS is when the style **rules** are placed in the `<head>` of the HTML page, inside of the `<style>` HTML element.

```
<head>
    <style>
        p {
            color: red;
            font-weight: bold;
        }
    </style>
</head>
```

EXTERNAL CSS

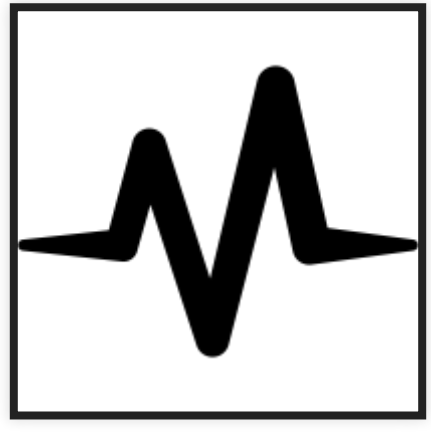
You link to this file from the <head> of your HTML.

Its best practice to put CSS in its own file.

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

style.css:

```
p {
    color: red;
    font-weight: bold;
}
```



PULSE CHECK

Which method should I choose?

BUILDING WEBSITES - CSS SPECIFICITY AND INHERITANCE

CSS is called "Cascading" Style Sheets for a reason...

This can get pretty crazy, but let's just focus on the basics for now.

SPECIFICITY

In general, "Cascading" means that styles can fall (or cascade) from one style sheet to another

For example, lets say we have the following 2 external stylesheets in our webpage

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

And then the stylesheets have the following:

style.css:

```
h1 {
    color: blue;
}

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

custom.css:

```
h2 {
    color: green;
}

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

As you can see, CSS specifies a priority scheme to determine which style rules apply if more than one rule matches against a particular element

This also comes into play when using a combination of inline, internal, and external styles

IMPORTANCE

You can override all of this by simply including "!important" in your style declaration, like so:

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

INHERITANCE

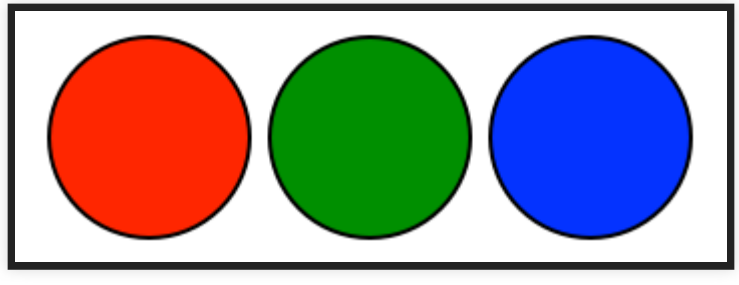
This is pretty straightforward -- in CSS, a child HTML element will inherit any applicable styles from its parent HTML element, unless otherwise specified.

```
body {
    color: blue;
    text-align: center;
}

p {
    color: red;
}
```

BUILDING WEBSITES - CSS COLORS

Colors can be specified in CSS in a variety of ways:



- Color Keywords
- Hex codes
- rgb / rgba
- hsl / hsla

COLOR KEYWORDS

These are used less frequently, but are handy for basic colors like `black` and `white`. There are several

See here for more

HEX CODES (RGB)

RGB COLOR VALUES

`RGB(0,0,0)`

- The first value is red, the second green, the third blue.
- Each value can range from 0 to 255, which expresses the same number of color steps as 00 to FF in base-16.

RGBA COLORS

- RGBa works identically to RGB, expect that it takes a 4th value called the "alpha".
- This is a value between 0 and 1 which will be used to determine a color's opacity on the page,

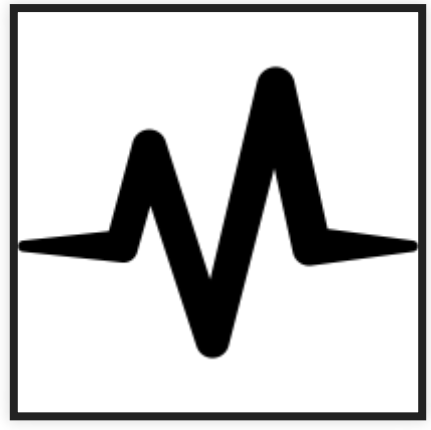
HSL COLORS

HSL

- Similar notation to RGB values, but specify colors using hue, saturation, and lightness.
- Not used very often

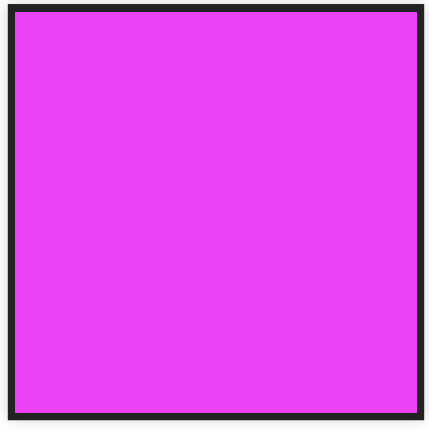
HSLA

- As with RGBa, HSLa is exactly like HSL for the first 3 values, but takes a 4th alpha-channel value.



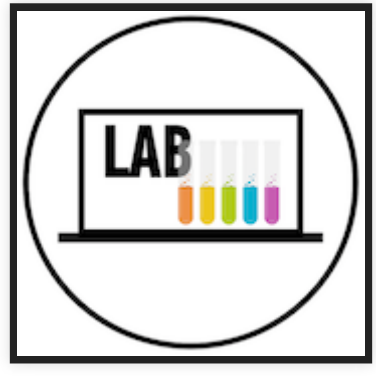
PULSE CHECK

Name that color:



CSS REVIEW

- A CSS style rule consists of a selector and declarations (each with a property and value) inside a declaration block
- CSS can be placed inline, internal, or external -- but you will almost always want to do external
- CSS priority is determined by a combination of inheritance, specificity, and importance -- a great resource can be found here



LAB TIME

- Your Portfolio!

LEARNING OBJECTIVES REVIEW

- We learned the DOM and can draw a simple DOM tree
- We can predict image paths and apply relative paths to `<img>` and `<a>` tags.
- We applied the CSS "cascade" including: importance, specificity and inheritance, and can explain the basic principles
- We experimented with CSS Colors and margin/padding

HOMEWORK

Complete the Portfolio and About pages according to the specifications outlined in the **FEWD-ATL-17** repo under Week 1/Assignment

Be sure it meets the criteria outlined in the Rubric.md document

Also, follow the instructions at the bottom of the page to create a new repo in your Github account, and submit your work via this method