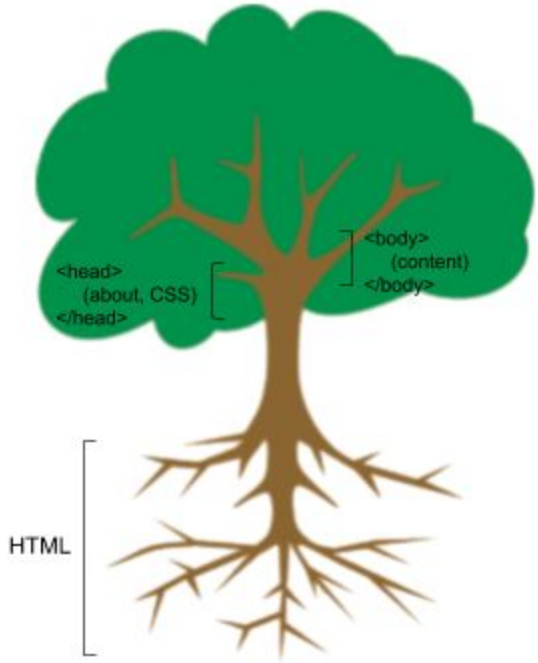
Creating a Responsive Website (HTML, CSS, and JavaScript)

Getting started on a new endeavor has consistently been one of the most challenging aspects no matter the subject matter. I've always been a very hands-on learner



HTML: provides the **basic** structure

CSS (Cascading Style Sheets): controls the presentation, formatting, and layout

JavaScript: controls the behavior of different elements

`head`	element is for metadata (information about the site)	
`body`	element is for the content	
`element`	first is **self-closing**; doesn't have a value	` ` (line-break)
`element`	second is a **common element**: a value is communicated that belongs to an element	`<p>hello, Mordor!</p>`
`element`	**element with an attribute** gives the element more meaning; attributes can have multiple values, elements can have multiple attributes	`<element attribute="value" attribute="value value">value</element>`
	HTML to CSS	To link the CSS file to the HTML, use a self-closing <link>

`<link>`	Between the <head>; self-closing tag needed to link the HTML file to the CSS. MUST INCLUDE: `href` anchor element), must be address to the CSS file `Type` the type of document being linked `rel` relationship between HTML and CSS files	`<link href="./style.css" type="text/css" rel="stylesheet">`
Paragraphs	`HTML= <p></p>` `CSS= p {}`	

`class`	Class attribute specifies >= 1 class names for an HTML element and can be used by CSS and JavaScript to assign certain tasks for elements with the class name. The example to the right shows HTML, CSS, and JS, respectively.
	
	Yes, it's case sensitive.
	Can have different tags with the same class. Just means they'll have the same style.
		`<p class="brand">Sole Shoe Company</p>`
		--- 	
		`.brand {`
		`}`
		--- 	
		`document.getElementsByClassName("brand");`

Multiple Classes	Can make rules in CSS that you can then transfer into HTML, for instance if you want an element to be both *x* and *y*.
	
	To add multiple classes to HTML element, separate them with a space.
		`.green {`
		`  Color: green;`
		`}`
		`.bold {`
		`  Font-weight: bold;`
		`}`
		--- 	
		`<h1 class="green bold">...</h1>`

`Class vs. id`	Class: reused over many elements HTML: `<h1 class="large-title"> ... </h1>` CSS: `.large-title {}` id: style only **one** element; override styles of tags and classes *use sparingly and only on elements that always appear the same* HTML: `<h1 id="large-title"> ... </h1>` CSS: `#large-title {}`	Page with two headlines--both bold but different colors: create CSS rules for `.bold, .green, .blue` ⇒ "bold green"; "bold blue";
Specificity + **Chaining**	!important > ID > class > tag * !important is very hard to override and should be avoided	`!important` > `#large-title {}` > `.large-title {}` > `h1 {}`

	^ chaining = ^ specificity	
Chaining Selectors	CSS can require HTML element to have 2+ selectors at the same time (i.e. chaining)	`h1.special {}` Must be both h1 and special (i.e. != p and special)

Nested elements	CSS can select elements that are nested within other elements
	HTML:
	`<u1 class="main-list">`
	`<li> …</li>`
	`<li> …</li>`
	`<li> …</li>`
	`</u1>`
	CSS: `.main-list li {}`	

Multiple Selectors and Conciseness	`h1 { font-family: Georgia; }` `.menu { font-family: Georgia; }`	`h1, .menu { font-family: Georgia; }`

*HTML has predefined elements - the name of the element is used in angular brackets (<>, </>). End tags on empty elements are neither required nor allowed.

Takeaways:

- CSS can change the look of HTML elements. In order to do this, CSS must select HTML elements, then apply styles to them.
- CSS can select HTML elements by tag, class, or ID.
- Multiple CSS classes can be applied to one HTML element.
- Classes can be reusable, while IDs can only be used once.
- IDs are more specific than classes, and classes are more specific than tags. That means IDs will override any styles from a class, and classes will override any styles from a tag selector.
- Multiple selectors can be chained together to select an element. This raises the specificity, but can be necessary.
- Nested elements can be selected by separating selectors with a space.
- The !important flag will override any style, however it should almost never be used, as it is extremely difficult to override.
- Multiple unrelated selectors can receive the same styles by separating the selector names with commas.

I downloaded Font Awesome--the free version because it's free \\I've got bills to pay and mouths to feed and yes, Cage the Elephant is in my top 5 favorite bands. In order to use it I attached the link in my HTML file like so: `<link href="fontawesome-all.css" rel="stylesheet">` Just a heads up: I'm going to start leaving my code from VSC (Visual Studio Code, the editor I'm using on my rigged up PC) as is instead of converting it to the same format I've been using for note taking.

At this point, I needed to be able to push code to Github for easy access and because it's vital information for programmers. Git is incredibly important for collaborative development because when you're working with a team, it's almost guaranteed that everyone's code will be a little bit different so you need a central location where code can get put and seen by everyone working on the team.

Git	Git keeps track of the history of things that you're working on. Git makes it easy to collaborate with others while maintaining the history and tracking the changes.	Software that allows you to do version control
Github (you can use Git and Github separately)	A web service that organizes Git. Like a social network for projects done through git	Username: gracehartzell
Repository (repo)	essentially the project; can hold multiple files within it	Project name: mySite
commit	Essentially the "save". Allows you to make changes to the files and preserve the history	
hash	Unique identifier for each commit	
git init	Initialize an empty repository	

Udacity - Intro to JavaScript

// brilliant brevity can be accomplished with this comment notation
/* Telling a story like a Hartzell (AKA long-winded and not necessarily consecutive) is best for this type of marking */

Quiz 2-1: First Expression
Write an expression that uses at least 3 different arithmetic operators; should equal **42**.

1. Identify the arithmetic operators: +, -, *, /, %
2. Since you're not working in the console, you'll need to use console.log()
3. Come up with some numbers!

My code:

```
console.log(10 + 15 - 3 + (40/2));
```

Quiz 2-2: Converting Temperatures
The Celsius-to-Fahrenheit formula:
F = C x 1.8 + 32
Set the fahrenheit variable to the correct value using the celsius variable and the formula above.
Log the fahrenheit variable to the console.

```
> var celsius = 12;          // 12 is given
> var fahrenheit = celsius * 1.8 + 32;
> console.log(fahrenheit);
< 53.6
```

Quiz 2-5: All Tied Up
Build a single string that resembles the following joke.

```
Why couldn't the shoes go out and play?
They were all "tied" up!
```

Your joke should take the format of a question and answer. The first line should be a question and the second line should be an answer.
Code:

```
var joke = "Why couldn't the shoes go out and play? \nThey were all \"tied\" up!"
console.log(joke);
```

Quiz: Yosa Buson (2-6)
Build a string using concatenation by combining the lines from this famous haiku poem by Yosa Buson. Each string should be printed on its own line.

```
Blowing from the west
Fallen leaves gather
In the east.
```

```
var haiku = "Blowing from the west" + "\nFallen leaves gather" +
"\nIn the east.";
console.log(haiku);
```

Funbits:

- camelCase= lowerCaseUpperCase
- Strings= make sure to use quotes otherwise the console will evaluate it as a variable with the value of whatever you entered in (Uncaught ReferenceError)

	STRINGS	
string	collection of characters enclosed in quotation marks (“”, ‘’)	“Captain is sleeping in her bag.”
concatenation	Combines two strings. Pay attention to whitespace and punctuation. Whitespace will not be auto-added.	“Captain,” + “ get off the keyboard,” Returns: “Captain, get off the keyboard.”
Implicit type coercion	JS will convert data to be the same type so it can then be added to a string.	entered: “Hello” + 5*10 returned: “Hello50”
VARIABLES	Conveniently provide access to reuse strings later in the code	
var variableName=	Be as succinct as possible!	var firstName= “Grace”
INDEXING charAt() varName.charAt() varName[]	Locating an individual character in a string. *Strings start at 0	
ESCAPING STRINGS	Cannot have two sets of quotation marks	Special characters: \\ \ (backslash) \” “ (double quote)

		`\'` ' (single quote) `\n` newline `\t` tab
Comparing strings	* case matters: lowercase > uppercase	`"Y" > "y"` → false `"Green" > "blue"` → false `"green" > "Green"` → true