# Front-End Web Development

## Overview:

* Slack is the best communication tool for her
* Office hours on Weekends through Google Hangout
* Use Sublime and Github
* Dashboard: FEWD

Boiler Plate:

<!DOCTYPE html> <html>

<head>

<title> This is a Web Page </title>

</head>

<body>

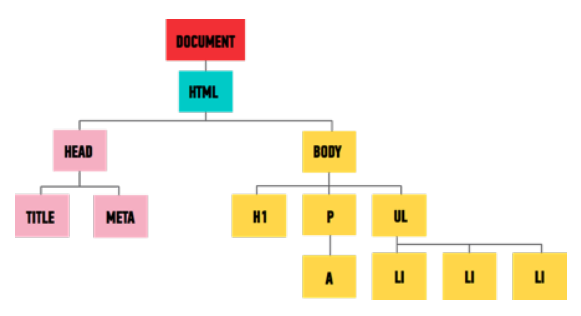
<p> This is a paragraph about web pages. </p>

<a href=“....”> This is a link to another page. </a>

</body>

</html>

\*\*quick command for this is to save the file as a .html file and then once you are in the file type “html” and then “tab” and it will create the boiler plate for you



When making a website:

-think about the wireframe

-then headings/font

-add content

-Lorem Ipsum for generating fake paragraphs

Tag= <h1>

Element= <h1>Hello</h1>

Tags

Bold=<strong></strong>

Italize=<em></em>

“Quotes”= <q></q>

Link= <a href=”url”>TEXT</a>

Navigation= <nav></nav>

Horizontal Rule(creates line)=<hr></hr>

Type:

Text-align: left, center, right, justify

Text-transform: uppercase, lowercase, capitalize

Text-decoration: none, underline

Line-height: 20px

Font-size:12px, 26px

Font-family: sans-serif, serif

Font-weight: normal, bold

Font-style: regular, italic

\*\*can select more than 1 google font types at once and put them in the html header as 1 line item

Tips:

-Command + D = change multiple same tags at once by selecting and clicking Command + D before changing item

-If you open the document in Chrome and then right click the page, you can inspect the code and see what is going on

-In HTML if you start typing and element and hit TAB the rest will show up

-Comment: Command + /

CSS:

Selector=h1

Declaration= {color: yellow;}

Property= color: (what you can change)

Value= yellow; (what you want to change it to)

Text color= color: blue;

Background color = background-color: black;

DOM tree=helps you get a feel for nesting and how to indent (h1, p, ul is in first line..then indent a, li..etc.)

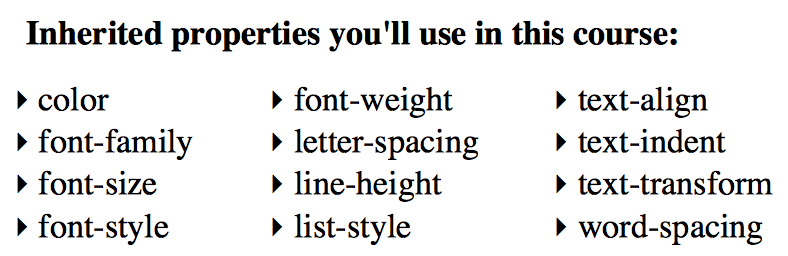
\*\*in order to normalize the browser code for your website you can use **Normalize.css**

**-Adriana will give us the .css file and we should add the link in our html file…**

**<link rel=”stylesheet” type=”text/css” href=”normalize.css”> (load this before the style sheet link)**

**-CSS is read from bottom to top!!**

Inheritance-parent elements passing down properties to their child elements. Make sure to specify the proper selector, the more specific selector takes priority. CSS rules cascade downward until they are canceled out by another rule.



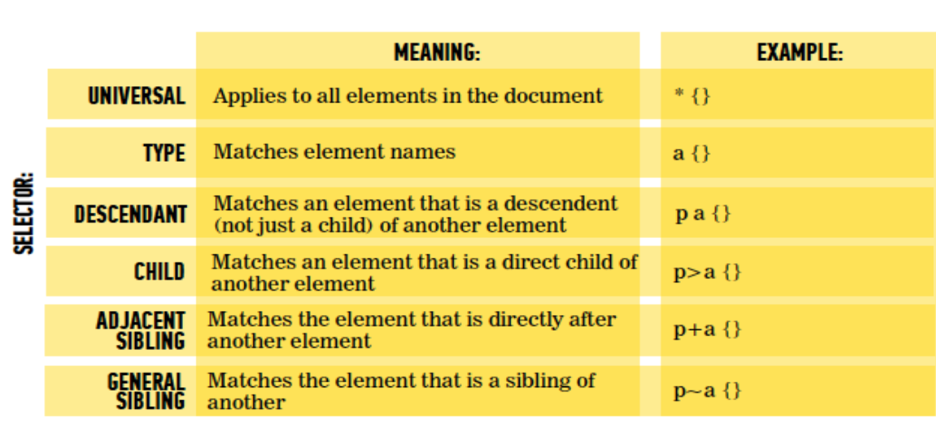
Classes & IDs-allow us to assign labels to our elements so we can target in our stylesheet

* Class-can be reused
  + Html= <li class=”emphasis”>Content</li>
  + CSS= .emphasis { color: pink; }
  + Can have multiple classes <div class=”blocks-wrapper clearfix”>
* ID-can only be used for 1 thing
  + HTML= <h3 id=”about”>Content</h3>
  + ID= #about {color: pink; }

Inline styles-styling directly in your html (this is usually really bad🡪 do NOT use)

Importance-only use this when necessary! Basically do NOT use

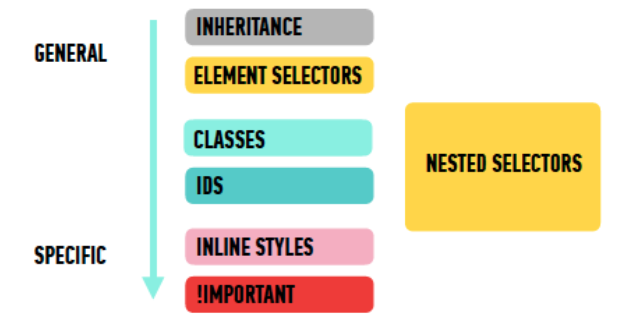
li { font-size: 100px !important; }

Nested selectors-

Descendant- p a =any link within the paragraph

Child- p > a = link has to be a direct child with NO selector between it

\*\*the table below shows what will override what…usually the most specific rule will take precedent over other rules



\*\*if two selectors have the same importance and specificity, the one that comes later in the stylesheet will take precedence

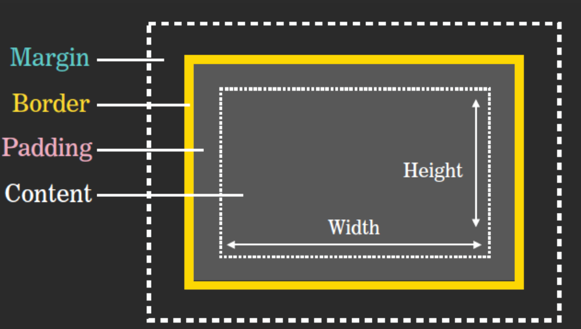
**Box Model:** 3 things to change the appearance

Border-immediately after content

-Can change: color, style, width

-Can style in one line: p { border: 3px solid #ff0000; }

Padding-space between content and border

Margin-space between border of one element and other surrounding elements. Used for spacing BETWEEN elements

Sublime Shortcuts:

**Command + D**=select word and then keep hitting that sequence until you have selected everything you want. Then change them all at the same time

Terminal Commands:

**Open folder/directory**: cd Documents

**Go back one folder/directory**: cd ..

**Create new folder/directory**: mkdir dummy\_folder

**Create new file**: touch index.html

**List content of file**: ls

**Full path to working directory**: pwd

Github: website where programmers store their code

-Github is similar to Google Docs. You have the ability to access your code anywhere

-can copy your projects onto your computer (clone)

-allows you to keep different versions of code (version control)

-lets others collaborate on projects (repository)

Git Commands:

**Put Git into file**: git clone (repository url)

**Adds all files you made change too**: git add. (this is kind of like saving the file)

**Adds individual files names:** git add file\_name1 file\_name2

**Message about changes**: git commit –m “detailed message about changes you made”

**Publishes code to GitHub:** Git push

**Shows recent commits:** git log

🡪 Adding code to git hub

1. git add .
2. git commit -m “what changes you made”
3. git push origin master

Git add vs. Git commit

-git add saves the code and then for the changes to be pushed through you have to do git commit and then git push to see the changes on the actual website

**Workflow:**

Make a change in the code (in Sublime)🡪 git add(in Terminal)🡪git commit(in Terminal)🡪git push(in Terminal)🡪GitHub

**CSS Layout: Display**

Block-level elements: every element is setup to be a block

* <h1,h2,h3, etc.>
* <li>
* <ul>
* <p>
* <ol>
* <div>

In-line elements: will always appear to continue on the same line as their neighbor elements

<a>

<img>

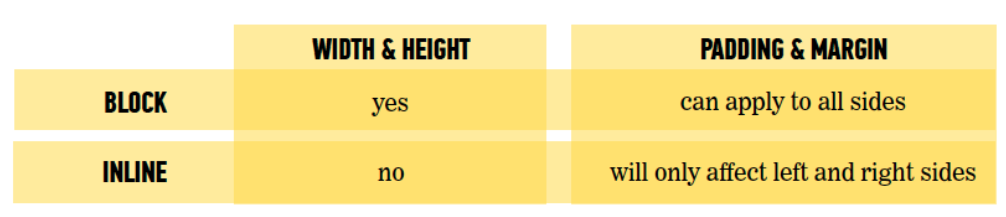
<em>

<strong>

<q>

<span>

\*\*You cannot add width, height, padding, margin to in-line elements\*\*



You can change wheter elements are displayed as inline elements or block elements by using the **display** property

*Examples:*

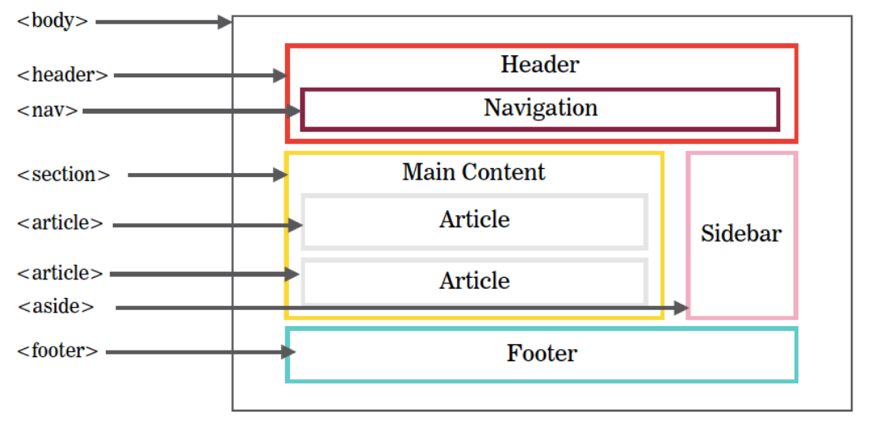
* li {display: inline;} 🡨make block level element inline
* a {display: block;} 🡨make inline element block
* h2 {display: inline-block} 🡨 Make a block-level element flow like an inline element, while retaining width, height, padding, and margin
* img {display: none;} 🡨hides the element

**DIVS:**

div-defines a section in your html, this allows developers to create sections

span-used to style one little piece of text within a larger paragraph, contain several inline elements

Structural Elements:

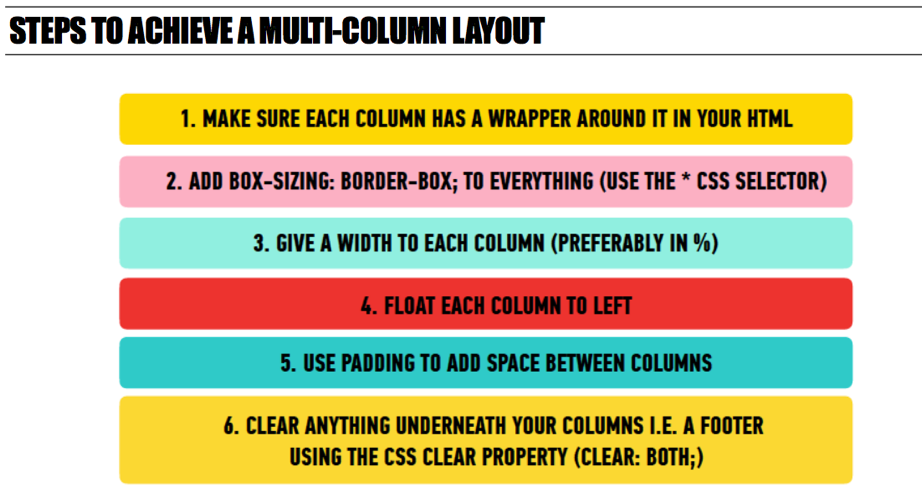




Multi Column Layout:

**Box-sizing: border-box** — the padding and border press their way inside the box instead of expanding the box

* CSS: \* {box-sizing: border-box;}



Floats-

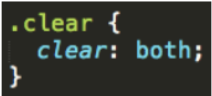
Float: left, float: right, float: none

Clear-

Clear: Both-*make sure an element starts on a new line (it may appear in the wrong place in your document)*

The **clear** property specifies which side(s) of an element other floating elements are not allowed

\*so when you float something and something else goes over it funny you use clear:both



Left: no floating elements allowed on left side

Right: no floating elements allowed on right side

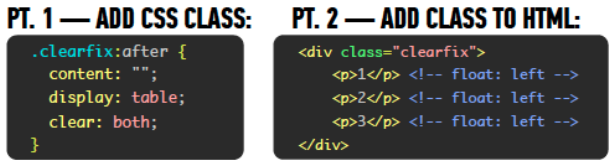
Both: no floating elements allowed on either side

None: allows floating elements on both sides

Clearfix- *fixes collapsed parent (it may appear like its not there)*

If a container element only contains floated elements use this code:

\*so if you have a section that contains something but isn’t showing that way use clearfix



Chrome Dev Tools:

* Click on the three dots on the right to change the view of the tools

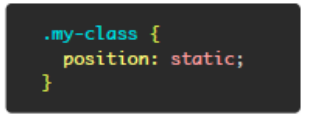
Refactoring: *cleans up code*

* Code refactoring is the process of restructuring existing computer code—changing the factoring—without changing its external behavior
* Refactoring improves nonfunctional attributes of the software
* Golden Rule: Keep code DRY (Don’t Repeat Yourself)
* Refactoring code: indent correctly, add helpful comments, organize code, clearly name code

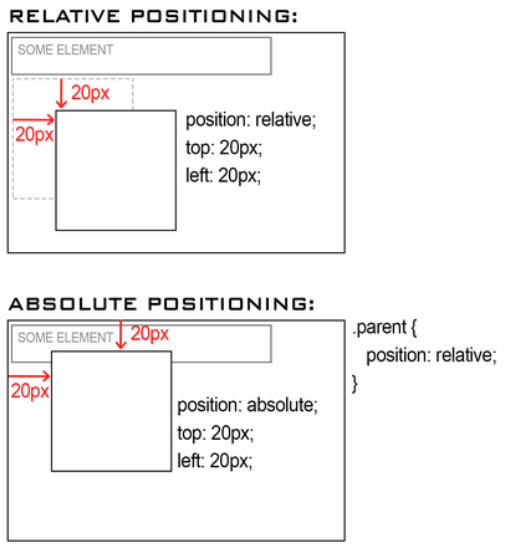
Advanced CSS Positioning:

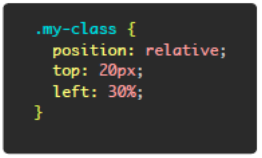
Static Positioning-

* this is the default; elements render in the order that they appear in the document



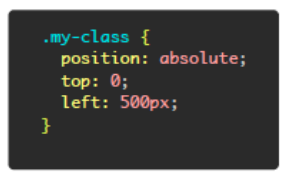
Relative Positioning-*relative to something else*

* Relative positioning moves an element relative to where it would have been in normal flow.
* For example, "left: 20px" adds 20px to an element's *left* position
* Creates a coordinate system for child elements.



Absolute Positioning-*always in same position*

* When the position property is given a value of absolute, an element is taken out of the normal flow of the document.
* This element no longer affects the position of other elements on the page (they act like it's not there).
* You can add the right, top, left and bottom properties to specify where the element should appear relative to its first positioned (not static) ancestor element
* Can use WITH relative positioning
* If you use relative AND absolute, it will stick something on top of another thing
  + i.e. position img relative and a blurb absolute (they need to be inside same div)



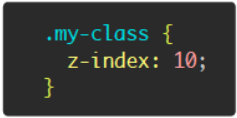
Fixed Positioning-*i.e. keeps a nav bar at the top*

* When the position property is given a value of fixed, the element is positioned in relation to the browser window
* When the user scrolls down the page, it stays in the same place.
* You can add the right, top, left and bottom properties to specify where the element should appear in relation to the browser window.



Z-Index: Use to overlap elements

* When using relative, fixed or absolute positioning, elements can overlap.
* When elements overlap, the elements that appear later in the HTML code sit on top of those that appear earlier in the page.
* If you want to control which elements are layered on top of each other, you can use the z-index property.
* This property takes a number — the higher the number the closer that element is to the front. Similar to 'bring to front' and 'send to back' in programs like Adobe Illustrator.
  + i.e. z-index: 1, z-index: 2, z-index: 3

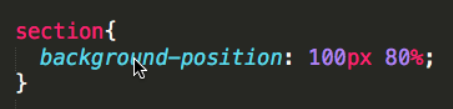


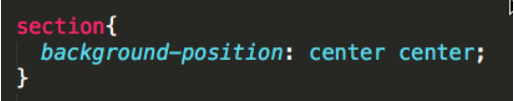
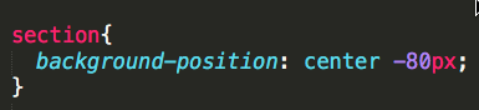
Background URL

* You can also add a background image in css
* The syntax is background-image: url(image\_url); You must specify that the image shouldn’t repeat (tile effect)
  + background-repeat: no-repeat;



* You can position the background using the position property





**\*Make sure code doesn’t use any DIVs**

**\*Come up with site design**

**\*\*need to be inside same div**

**div {**

**position: relative;**

**}**

**div p {**

**position: absolute’**

**bottom: 50px;**

**color: red;**

**}**