http://bit.ly/workshop-attend
ance-19

http://bit.ly/WeatherApp19

Build a Weather App

DAY 3

Introduction!

For new students and another new mentor :P

Important Notes

- You **need** to know Basic HTML & Basic CSS to continue with this workshop.
- We will be using **HTML5**, **CSS3**, and **JavaScript** for building this complete web app. And today we will be learning the most interesting part - **JavaScript** **:o**
- This workshop will be spread across 6 - 1 = 5 parts each week from 5:30 to 7:30 in this same room (SMITH 407).
- In the first workshop, we made the HTML structure for our website. Then we designed our page using CSS. *“I hope everyone remembers what we did!”*

Review Last Class

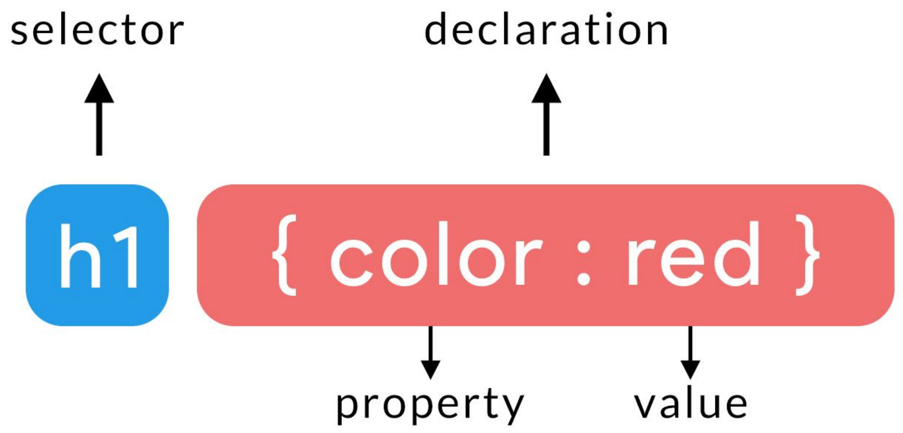
Let’s review CSS!

What is CSS?

- Short for Cascading Style Sheet
- Used for styling pages
- Latest Version is **CSS3**
- It can be implemented in different ways
- CSS is ***mostly*** case-insensitive

CSS Syntax

- First is **selector** : all the styles are applied to this.
- Second is **declaration** : what those styles are.
    - Property : what property are you changing.
    - Value : what are you changing it to.



Inline CSS

- Recall attributes!!
- Attributes are like properties to tags and **style** is also property.
- Rarely used.
- For example:

```
<p style="color : red;">This is a paragraph.</p>
```

CSS inside <head>

- We use <**style**></**style**> tag
- How do we use this:

```
<head>
    <style>
        p {
            color : red;
        }
    </style>
</head>
```

Separate CSS file

- For longer CSS and for extensive websites we love to keep the structuring and styles separately
- We do this by creating a new **.CSS** file and linking that with our **.HTML** file
- For linking these two files we use <**link** /> tag -

  ```
  <link rel="stylesheet" type="text/css" href="style.css" />
  ```
- After linking you can write it in the same way you wrote inside the style tag.

Height and Width

- There are many different measurements for height and width
- Rem, em, px
- When to use rem vs em vs px
- Short summary of above article: don't use px because it's not accessible. Read above article to find out more
- p {
    - width : 150em;

      height: 500em;

  }

Other Common Properties - Color

- Used to specify color of an element
- Can be passed a hex code
    - A hex code is a way of representing a color. Ex: #4263f5 is a blue
    - Check out Google color picker to play with hex codes
    - Mix and match with digits from 0 to 9 and alphabets from a to f (must be 3 or 6 character long)
- Can also be passed as an rgb value, but hex codes are more commonly used.
- Example -

  p { color : red; }

Other Common Properties - Font

- Broadly used to specify font properties.
- Fonts like ***Arial*** and ***Times New Roman*** are common fonts.
- Example -
  ```
  p { font : 15px arial, sans-serif; }
  ```
- But this is not the easiest way to add fonts. We can further specify individual font properties.
- Example -
  ```
  p {
          font-size : 15px;
          font-family : arial, sans-serif;
  }
  ```

Other Common Properties - Background

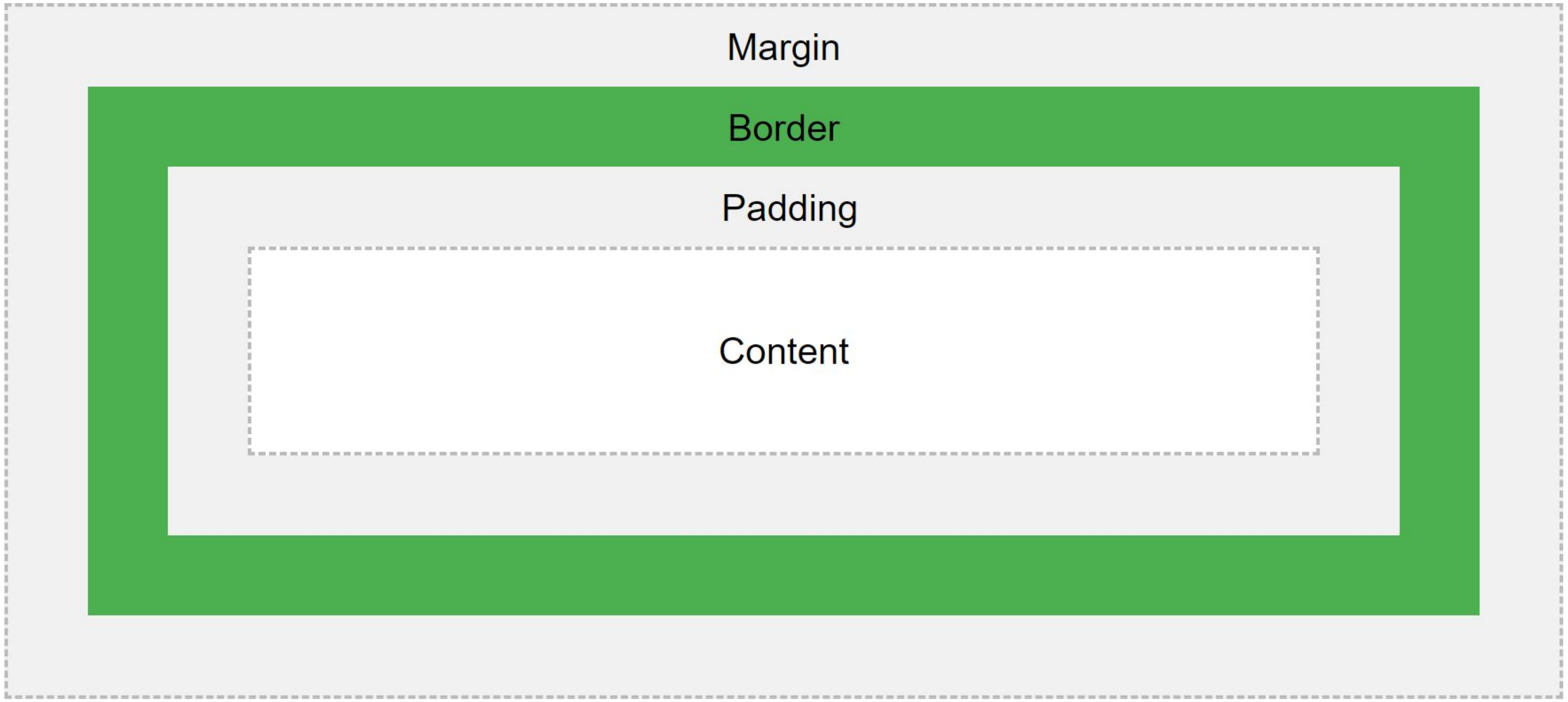
- Broadly used to specify background properties.
- Example -

  p { background : white url("img.png") repeat left bottom; }

- But this is not the easiest way to add background.
- Example -

```
p {
        background-color : white;
        background-image : url("img.png");
        background-repeat : repeat;
        background-position : left bottom;
}
```

CSS Box Model



Margin

- Margin is the outermost layer of space from an element

```
p {
        margin-left : 10px;
        margin-right: 10px;
        margin-top: 10px;
        margin-bottom: 10px;
}
```

Border

2nd outermost space from element

```
p {
        border-left : 10px;
        border-right: 10px;
        border-top: 10px;
        border-bottom: 10px;
}
```

Padding

Innermost spacing on an HTML element

```
p {
	padding-left : 10px;
	padding-right: 10px;
	padding-top: 10px;
	padding-bottom: 10px;
}
```

HTML classes

- Classes are properties that you can give an HTML element
- They are case-sensitive.
- Ex: <div class=”box”></box>

What do classes do?

- On their own classes don’t do anything
- Classes are useful though because they allow you to group HTML elements and apply a collective style

```
.apple { /* notice how there is a . in front of apple! */
        padding-left : 10px;
        padding-right: 10px;
        padding-top: 10px;
        padding-bottom: 10px;
}
```

HTML Ids

- Ids are similar to classes except that ids must be unique - two elements cannot have the same id
- They are case-sensitive.

```
#apple { /* notice how there is a # in front of apple! */
        padding-left : 10px;
        color: 10px;
}
```

The display property

- The display property can take a couple of values
- Some notable ones are block, inline, block-inline
- Inline vs block-inline
- Other useful display properties are hidden and none

```
div {
        display: inline;
}
```

Flexbox

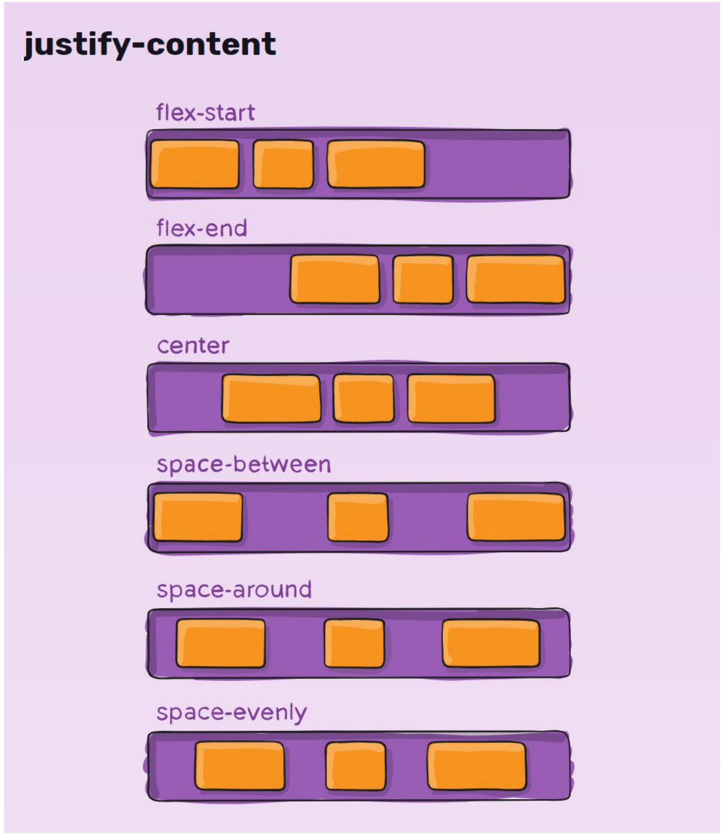
- Flexbox is a term used to describe elements with the property display:flex
- A flexbox is a unique way of displaying elements and can come in handy in a lot of situations
- Make a flexbox using the display property
- Every child element of a flexbox will become a flex item

```
div {
        display: flex;
}
```

Flexbox - Justify Content

The justify-content property is used to

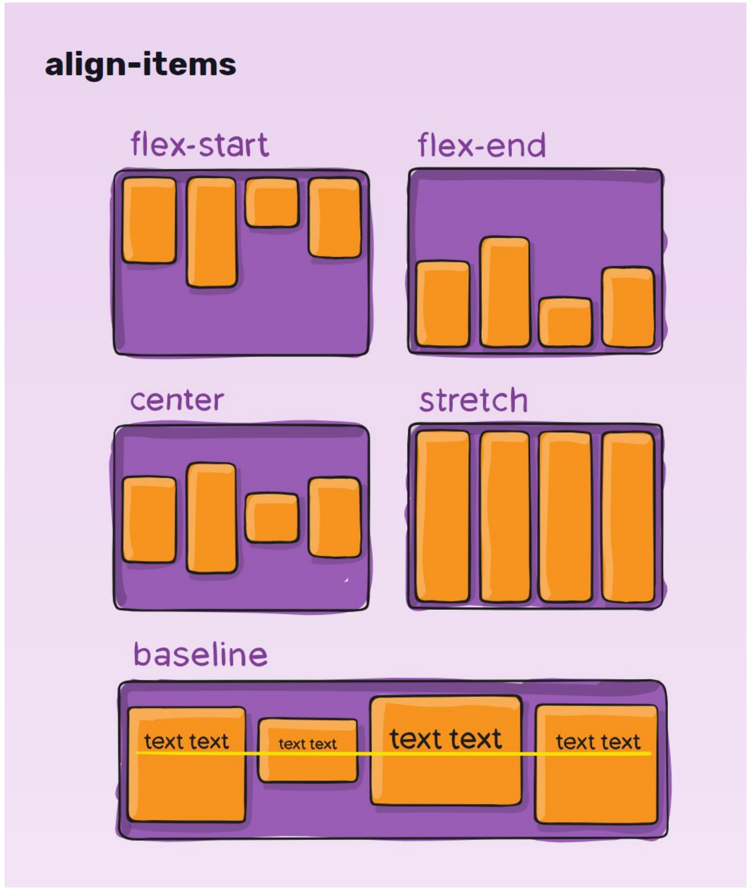
align items horizontally



Flexbox - align-items

Used to align elements vertically

in the container



More on flexbox

- An element is made into a flex box by specifying **display:flex**
- **flex-direction** - direction that elements go
- **flex-wrap** - whether or not to wrap elements once they reach the end of the container
- Read more about Flexbox on CSS Tricks
- Practice on Flexbox Froggy

Pseudo-selectors

- Pseudo-selectors allow us to specify a style only when a certain event occurs
- Ex we can use the :hover pseudo-selector to make an element red only when a user hovers over it
- Ex:

```
p:hover {
        background-color : red;
}
```

Any questions up till now?

Don’t feel shy!

JavaScript

```
let x = “student” + “ ” + “workshop”
```

What is JavaScript?

- JavaScript is a high-level, general purpose programming language.
- JavaScript is an interpreted language, so the computer translates JavaScript into machine code at *runtime*. This is in contrast with compiled languages like C and Java.
- This means most of our errors will appear at runtime and might be a little bit harder to fix!

Creating a JavaScript File

- JavaScript are linked to the HTML document using the <script> tag, just like how we linked the CSS to the HTML with the <style> tag!
    - Something like this:
    - `<script src="path/to/my/script.js"></script>`
- We can also learn JavaScript in the console!!! - We will talk about this later when we work on the weather app.

Switch to Nam

Variables

- Variables in JavaScript are dynamically typed
- JavaScript variables are declared using the *let* keyword:
    - `let a = 10; //This is a number with the value 10`
    - `let b = "hello world"; //This is a string with the value “hello world”`
- *Numbers* are used to represent numeric data (think 1, 2, 3, 4)
- *Strings* are used to represents a collection of characters (think a word, or a sentence)
- *let* are block-scoped, and *var* is functional scoped
    - TLDR: use let

Booleans

- *Booleans* are conditional statement, and can be produced using relational operators:
    - == (Equals To). Returns TRUE if both side are equal and FALSE otherwise
        - (3 == 3 => true, 3 == 4 => false)
    - && (And). Returns TRUE if both side are TRUE, and FALSE otherwise
        - 3 == 3 && 4 == 4 => true because TRUE && TRUE is TRUE
        - 3 == 4 && 4 == 4 => false because TRUE && FALSE is FALSE
    - || (Or). Returns TRUE if one side of the equation is TRUE.
        - 3 == 3 || 4 == 3 => true because 3 == 3 is TRUE
        - 3 == 2 || 4 == 3 => false because both sides are FALSE
    - ! (Not). Reverse the sign of a boolean value
        - !(True) == False => true (not true is false, false is equals to false)
        - !(3 == 2) && 3 == 3 => true (not 3 == 2 is true, and 3 == 3 is true, true and true is true)

Arrays

- Array is a collection of element, declared using the [ ] bracket.
  - `let food = ["apple", "orange", "banana"]; // an array of Strings representing food name`
  - `let numbers = [1, 2, 3, 4, 5]; // an array of numbers from 1 to 5`
- Access an array using the bracket notation and 0 indexing
  - `food[0] == "apple"`
  - `numbers[1] == 2`
- Adding to an array using push
  - `food.push("lemon");`

JavaScript Objects

- A one-dimensional sequence of values that are all stored in a single variable
- Instead of using an integer index as key (like an array), an Object uses *String*
  - Imagine a dictionary
- Object uses Key-Value pairs. Key can be used to look up values using the dot notation (.). You don't use position to refers to a key-value pair like an array.
- Objects uses { } to represents itself.
  - `let values = {"hritik" : 1, "kevin" : 2, "nam" : 3};`
  - `values.hritik == 1; values.kevin == 2; values.nam == 3`

If Else

- In JavaScript, you use conditional statement (if-else) for control structure

-
  ```
  if (condition) {
      // do something
  } else {
      // do something
  }
  ```
- condition can be any expression that evaluates to a boolean value (true/ false)

For loop

-
  ```
  // an example for loop. The `i` is not declared as an int. This loops over the array and log
  // out elements at the ith position.
  let array = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
  for (let i = 0; i < array.length; i++){
    console.log(array[i]);
  }
  ```

While loop

-
  ```
  // an example while loop. This also loops over the array and log out elements at the ith position.

  // array.length = 10;

  let array = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];

  let i = 0;

  while (i < array.length){

    console.log(array[i]);

    i++;

  }
  ```

Functions

- Declared using the *function* keyword in order to abstract code

```
//A function named `double` that takes 1 arguments

//and returns the doubled value of that argument

function double(num) {

  //Function body: perform tasks in here

  let doubled = num * 2;

  // Return: what you want the function to output

  return doubled;

}
```

```
// Call the double() function with the values 10

// Assign the result to `twenty`

let twenty = double(10);

// console.log(twenty) logs the number 20
```

Anonymous function

- In JavaScript, functions ARE variables:

```
let double = function(num) {

      return num * 2;

};

// console.log(double(10)) logs the number 20
```

- These produce the same function

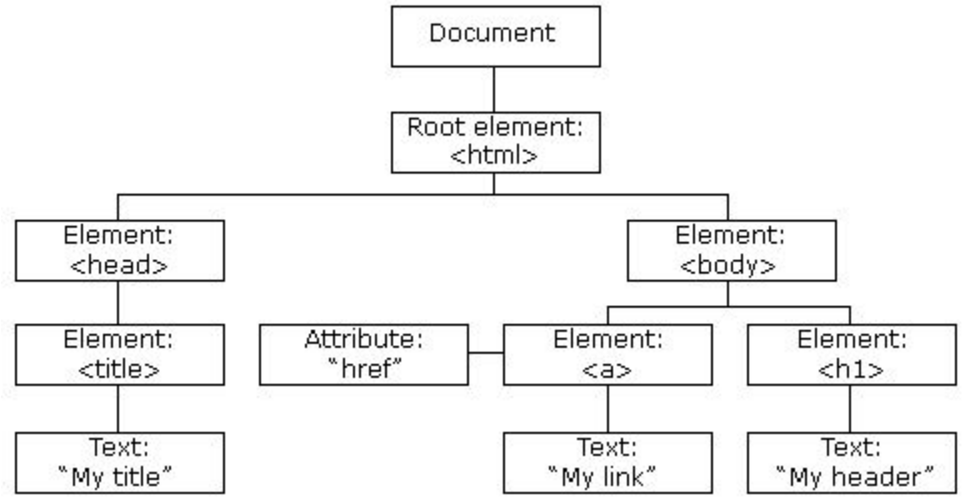
```
function foo(bar) {}

let foo = function(bar) {}
```

Open exercises folder > exercise-3 folder into VS Code and read the prompt in the .html file

Switch to Kevin

Document Object Model (DOM)

- When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel
- The HTML DOM Object is constructed as a tree of objects



What do DOM selectors do?

- Allow us to access HTML element(s) in JavaScript using ids,

DOM selectors

- document.getElementById
- document.querySelector
- document.querySelectorAll

What can we do with the DOM and JavaScript?

- JavaScript can change all the HTML elements in the page
- JavaScript can change all the HTML attributes in the page
- JavaScript can change all the CSS styles in the page
- JavaScript can remove existing HTML elements and attributes
- JavaScript can add new HTML elements and attributes
- JavaScript can react to all existing HTML events in the page
- JavaScript can create new HTML events in the page

document.getElementById

- Used to access HTML elements by their Id
- Ex: ***let header = document.getElementById(‘header’);***

document.querySelector

- Use a css selector to access an HTML element in JavaScript
- Ex: ***let header = document.querySelector(‘body h1’);***

document.querySelectorAll

- Get an array of elements that match the css selector
- Ex: ***let header = document.querySelectorAll('p'); // get an array of p elements***

Change classes with .classList

- We can use the *.classList* property to change the class of an element

Ex: ***let header = document.getElementById('header');***

header.classList.add("blue"); // add the class "blue" to #header

header.classList.remove("blue"); // remove the class "blue" from #header

Change css properties using .style

We can change css styles of an element by using the ***.style*** property

```
let header = document.getElementById('header');

header.style.backgroundColor = "red" // change background color to red

header.style.color = "red" // change text color color to red

header.style.marginLeft = "20px" // change margin left distance
```

Hide/Show an element using display

We can use JavaScript and the css display property to hide/show elements on a page.

```
let header = document.getElementById('header');

header.style.display = "none"; // hide an element on the page

header.style.display = "block" // unhide the element
```

Open exercises folder > exercise-4 folder into VS Code and read the prompt in the .html file

Any questions up till now?

Don’t feel shy!

That’s a lot for a day!

I think we should rest up a bit!