**Web Fundamentals Notes:**

**3/6/2023-4/5/2023**

**HTML:**

**How The Web Works:**

**Objectives:**

* Gain a high-level understanding of how the internet works.
* Learn about the request-response cycle.

**Internet:** A large network of computers that are connected and can communicated with each other. The web is made up of computers that we call **Clients** and **Servers**.

**Client:** Are computers and other internet-connected devices that use the internet.

**Servers:** Are computers that sore web pages or apps. They are computers, but typically without the keyboard, trackpad, or screen.

**The Request-Response Cycle:** When a **Client** wants to access a web pages, a copy of the is downloaded rom the server onto the **Client** machine to be displayed in the user’s web browser. In other words, the **Client** makes a **Request** and the servers answers back with a **Response**.

**Requests:**

* Typing a URL in the address bar of a browser.
* Clicking a link.
* Submitting a form.
* Refreshing a page.

**Responses:** The server we are trying to reach then **Responds** with some kind of content. Maybe it’s search results, a friend’s profile or a video content. In all of these cases, this content is providing to the **Client** as some combination of **HTML**, **CSS** and **JavaScript**.

**Intro To HTML:**

**Hypertext Markup Language HTML:** Is the main markup language for all web pages. HTML Elements are the basic building-blocks of the internet.

**Objectives:**

* Understand the theoretical components of the web and how they interact.
* Understand the **Client-Server** model.
* Understand the most important parts of what is displayed on the border.
* Understand the roles of **HTML** VS **CSS** VS **JavaScript**.
* Understand how to incrementally break down building a web page into its fundamental components.

**What is HTML?:**

**Hypertext Markup Language HTML:** Is the main markup language for all web pages. HTML Elements are the basic building-blocks of the internet. Its purpose is to let us communicate with the browser the meaning of the content that we wish to place on a web page. To achieve this, HTML defines a number of **Tags** that we can wrap the content with:

* Headings and Paragraphs of text.
* Images.
* Links.
* Lists
* Tables.
* Forms.

**HTML Tags** describe the content they contain. For example, there are **Paragraph Tags** for **Paragraphs**, **Anchor Tags** for **Links**. The browser does not display the **HTML Tag** but uses the **Tags** to interpret the content of the pages.

**Why HTML and CSS?:**

The **HTML** and **CSS** chapters are designed to take you through the theoretical components of the web as well as the first step in creating a web app (building out the Client-Side.) When building a web app we believe it is best to always start with a clickable prototype involving only HTML, CSS and JavaScript so it is only natural to reach these technologies first.

**Creating a simple HTML Document:**

* Create a new file in the Text Editor of your choice.
* Type “Hello world!” Into the file.
* Save the file as hello\_world.html.
* Drag the file into your browser to open it.

**Opening and Closing Tags:**

Since **Tags** are supposed to wrap things, most **Tags** come in pairs: one opening and one closing **Tag** that denotes the beginning and end of content. **Tags** without a leading toward slash are called **Opening Tags** while **Tags** with leading forward slashes are called **Closing** **Tags**. (<p></p>.)

**Nesting:**

Notice that between the **Opening Tags** and **Closing Tags HTML Tags**, we also have **Head Tags** and **Body Tags**. Tags can encapsulate other **Tags**, this is called **Nesting**. **Nested** items are indented with tab to make the document easier to read. The hierarchy that rises from **Nesting** is called **Dom: Document Object Model**.

**Commenting:**

**Comments:** A section of test that is not process by the web browser.

<!--This is a comment in **HTML**-->

**Text Elements:**

**Objectives:**

* Learn how **HTML** affects what we see on a page.
* Recognize the **Syntax** of **HTML Elements**.

**Syntax:**

Every **HTML** web page is composed to **Elements**, which specify different types of content. The **Syntax** for indicating the types of **Elements** is with **Tags**. A section of content is wrapped in a **Tag**, where the content beings with an **Opening Tag** and ends with a **Closing Tag**. **HTML** has a finite, pre-defined list of **Tags** that browser will recognize. The browser does not display the **HTML Tags** but uses the **Tag** to interpret the content of the page. **HTML Tags** describe the content they contain.

**Common Elements:**

**Headers:** <h1> through <h6>.

**Paragraphs:** <p></p>.

**Anchor Tags:** <a> Attribute: href.

**Emphasis and Strong:** <em> and <strong>.

**Lists and Table:**

* Recognize the **Syntax** of more **HTML** Elements.
* Learn about **Parent**, **Child**, **Sibling** relationships among **HTML** **Elements**.

**Syntax:**

Every **HTML** web page is composed to **Elements**, which specify the different types of content. The **Syntax** for indicating the type of **Element** with **Tags**. A section of content is wrapped in a **Tag**, where the contain begins with an **Opening Tag** and ends with a **Closing Tag**.

**Lists and Tables:**

**Unordered Lists:** <ul> with **Nested** <li> that contain the **List** items.

**Ordered Lists:** <ol> with **Nested** <li> that contain the **List** Items.

**Table:** <table> with **Nested** <tr> (table rows) that contain Nested <th> or <td> for **Header** or regular data cells.

**Parent, Child, Sibling:**

**Tags** like <li> or <td> are always used inside the other **Tags** like <ul> or <tr>. When an **HTML** **Tag** is **Nested** inside of another **Tag**, we call the inside **Tag** a **Child** of the outer **Tag**. Adjacent child **Tags** like multiple <li> are considered **Siblings** of each other.

**Images and Videos:**

**Objectives:**

* Learn how to use img and video **Elements**.
* Learn how to make video autoplay in modern browsers.

**More Elements:**

**Images:**

<img> src: Image path. Alt: Image description.

**Videos:**

<video> src: Video path. Control: Boolean and control display.

**Why Indentation?:**

Indent based on Child/Parent relationship for neatness and readability.

**Forms:**

**Objectives:**

* Learn which **Elements** to use inside of **Forms**.

**Forms** allow users to interact with our website in a more meaningful way. They can tell us about themselves, give us feedback, or even created their own stories and upload meaningful images and videos using **Forms**. As such, it’s important that we are familiar with a wide varity of different Input types.

**Common Inputs:**

**Text Input:** <input type="text" name="first\_name">

**Number Input:** <input type="number" name="age">

**Passwords:** <input type="password" name="password">

**Date:** <input type="date" name="dob">

**Color:** <input type="color" name="text\_color">

**Radio:**

<input type="radio" name="font" value="bold" id="bold">

<label for="bold"><strong>Bold</strong></label>

<input type="radio" name="font" value="normal" id="normal">

<label for="normal">Normal</label>

**Checkbox:**

<input type="checkbox" name="accept" id="accept">

<label for="accept">I agree</label>

**Textarea:** <textarea name="comment" cols="20" rows="3"></textarea>

**Select:**

<select name="snack">

<option>Almonds</option>

<option>Cheese and Crackers</option>

<option>Pita and Hummus</option>

<option>Pears</option>

</select>

**About Divs:**

The <div></div> **Elements** exist for use to create a division in our code where we can group **Tags** that have a common purpose together. In the example above we do this so that the <label> and <input> **Tags** are lined up next to each other.

**HTML Basics Lecture Notes:**

Lecture Link: <https://www.youtube.com/watch?v=tE1huAA2BbU&t=1060s>

**Request/responds Cycle:** The client is the computer. When you type in an URL you are submitting a request to servers. Servers is the internet.

**HTML:** Skeleton of the webpage.

**CSS:** Skin of the webpage.

**JS:** Brain of the webpage.

HTML naming convention: index.html.

**HTML:** Hypertext markup language.

**Title Tag**: Title of the webpage tab.

**H1:** Title of the webpage.  
**P Tag: Paragraph Tag** for writing text on webpage.

**Opening Tag/Closing Tag:** Most tags have an **Opening Tag**, **Content**, **Closing Tag**.

**Anchor Tag:** For adding links. <a href=“”></a>

**Image Tag:** Self closes <img src=“” alt= “”>

**Lists Tags:**

**Unordered List: UL:** <ul></ul>

**Ordered List: OL:** <ol></ol>

**Parent Elements** with the **Child Element** being the list within the **UL** or **OL**.

**Table Tag:** Also have **Child Elements**.

**Forms:** Requesting information from the user.

**div:** Used for grouping Elements together.

**CSS:**

**Selector Basics:**

**Objectives:**

* Learn about selecting HTML in our CSS using Tags, Classes and ID.

**Selecting An Element:**

Any **HTML** **Element** in our code can be selected like h1, a, table, button. We can even select some **Tags** you might not expect like body (useful if we want to define an overall padding or background color to the website.

**Selecting A Class:**

Another way we can select a **Tag** is by its **Class**. In the **HTML** we can add a **Class** attribute with the name of the **Class** we want to select. In the **CSS** we can then select a **Class** by using a . followed by the name of the **Class**.

**Selecting An ID:**

Sometimes there only needs to be one of the given **Element**, in those cases we can use ID as well. In **CSS** we can then select an **Element** by using a # followed by **Class** name.

**Selector Advanced:**

* Learn about **Wildcard**, **Descendant**, **Direct Descendant**, and **Attribute Selector**.
* Learn how combining **Selector** affects **Selector Specificity**.

**The Wildcard:**

If you ever want to select all the **Elements**, the **Wildcard** selector can do just that.

**Descendant Selector:**

Sometimes we know the **Element** we want to select is inside another **Element**. I the “**Parent**” **Element** is one we can easily select (maybe it had a **Class** on it) then we can use this to our advantage to access the **Child** **Element**. If we want to select the **Paragraphs** that are outside the “quote” we can use a space in our selector between the **Parent Element** and the **Child**, we want to select.

**Direct Descendant Selector:**

Similar to the **Descendant Selector**, we can select a **Child Element** with the **Direct Descendant** Selector ‘>’ but unlike the **Descendant Selector** it won’t select a grandchild.

**Attribute Selector:**

Sometimes we have a situation where we want to style two similar **Elements** that rely on an attribute for how they behave. For instance if we are making a Form we likely use ‘<input type=“text”> for the user to input data into and ‘<input type= “submit”> which acts as the button the user clicks to submit the **Form**. **CSS** we would want to use on one we might not want to use on other. While we could use a **Class** to differentiate between the two, we could also use a **Selector** that looks at the **Attribute**.

**Box Model:**

**CSS Properties - Color:**

**Objectives:**

* Learn about named, rgb and hexadecimal colors.
* Learn the different between color and background color.

Color is one the best properties to work with when setting look and feel of the website. There are some tricky things to be aware of when working with it. Color can be set, using a few different systems.

**Named Colors:**

Selecting color by the name of the color.

**RBG:**

Each number must be tween 0 and 255 and they are set for red, green and blue.

**Hexadecimal:**

Works very similar to RBG with each pair of numbers/letters representing the red, green and blue.

**Color VS Background:**

When we apply CSS we need to keep in mind that color changes the color of text, and background color changes the color in the background of whatever Element we have selected.

**CSS Properties – Text:**

**Objectives:**

* Learn about text-align and text-decoration.
* Learn about font-weight, font-style and font-family.

**Text Alignment:**

We can adjust the text alignment using text align.

text-align: left;

text-align: center;

text-align: right;

**Text Decoration:**

We can apply or remove text font, size and color.

**Sizes:**

**Width and Height:**

The width and height properties are used to determine the size of your **Element**. The values can be expressed in pixels (px) and percentages (%). When working with sta5tic content using pc is suitable as you will as you will be defining your page does not change. If you are working with responsive design, you will want to use %.

**Font-Size:**

The font-size propter values can be express in four different unites pt, px, em.

**The Flex Advantage:**

**Objectives:**

* Learn how **Display: Flex** is applied to a **Parent Element** to style its **Children**.
* Learn how **Display: Flex**, ignored **Whitespaces** in our **HTML** when positioning **Elements**.

When using **Inline-Block** to have out columns display next to each other, the whitespace all counts as blank “text” **Element** that takes up space. If we rewrite our codes so that next **div** begins after the previous **div** this will eliminate extra space.

**Using Flex:**

If we change how we write the code to use **display: flex** on the **Parent** “container” **div** then we can write our HTML with our usual indentation and we can avoid having the newline and tab between **divs** take up space.

**Justify The Content:**

**Objectives:**

* Learn about **align-items**.
* Learn about the different values we can apply to **justify-content**.

Flex can be used to position small Elements (like images or boxes) within a larger Element. Properties like **align-items** and **justify-content**.

**Being Flex-ible:**

* Learn about **display: flex;**
* Learn about the flex:1; shorthand.

**The Flex Shorthand:**

One of the most useful things we can do when using display: flex is to make use of a shorthand property called flex! This property combines: flex-grow, flex-shrink, flex-bias and can provide an easy way for us to create columns in our layout. Think of each Element with flex:1; like one slice of pie. If another column were to have flex:2; it will essentially take 2 slices of the available.

**CSS Resets:**

The simplest CSS resets that web developers commonly use is to simply zero out the margin and passing for all visible Elements on the page. This will make our design consistent from one browser to another.

**CSS Basics Lecture Notes:**

**Lecture Link:** <https://www.youtube.com/watch?v=jRkA_2sU6Qs&t=5s>

**ID Selector:** The strict name of one **Element** and one **Element** alone. Call this in the CSS with #.

**Naming conventions: Name CSS** stylesheet.

<link rel= “style.CSS”>

**Text Alignment:** For aligning text, left, right, center. If moves the text within the **Element** you are targeting. **Text align** takes up the entire width of the page.

**Wildcard:** Selects everything on the page.

**CSS Reset:** Set margin and padding to 0 with Wildcard to get rid of needless default space.

**Margin:** Used to create space around Elements.

**Padding:** Properties are used to generate space around an Element’s content, inside of any defined borders.

**Height:** How tall the container is.

**Width:** How wide the container is.

**CSS Flexbox/Position/Lego Concept Lecture Notes:**

**Lecture Link:** <https://www.youtube.com/watch?v=JHFtFVOYAtE&t=27s>

**Flex:** All **Children** follow the **Parent**. Put on **Parent Element** to affect the **Children**.

**Flex Box:** Flexes display for different displays.

**Inline Block:** Stacking blocks against each other.

**Display Flex:** Makes content fit for different displays.

**Justify Content:** Allows browsers to distribute space between and around content items along the main axis of a flesh container and the inline axis of a grid container. Works on the X axis.

**JS:**

**Variables, Types, Operators:**

**Objectives:**

* Learn how to create and use Variables.
* Learn about basic Data Types in JS.
* Learn about Operators in Js.

**Variables:**

In JS we can define **Variables**, basically a name we can use to store and modify some information. When declaring a **Variable,** we need to start by using the var keyword followed by giving it some sort of unique name.

**Operators:**

(+) Adds two **Variables** together.

(-) Subtracts two **Variables** from one another.

(\*) Multiplies two **Variables**.

(/) Divides two **Variables**.

**Functions:**

**Functions:** A block of code will run when called.

**Adding Parameters:**

**Functions** are great, but they become even more useful when we’re able to send in different values so that the same code runs, but it may vary slightly depending on input! We indicate that a **Function** requires input by specifying **Parameters** in the parente4hse next to the **Function’s** name. Then, when we call on the **Function**, we pass in **Arguments**, or actual values to be used in the **Function**.

**Click Event:**

**Onclick:** Is triggered when user left clicks.

**What is “this”?:**

**Objectives:**

* Learn what the “**this**” keyword is used for.
* Learn some basic ways we can manipulate Elements.

“**This**” – keyword refers to an object that is executing the current piece of code.

**Hover Event:**

* Learn about onmouseover and onmouseout.
* Learn how “**this**” can be used with the **Elements** we interact with.

Add events that happen without having to click.

**Query Selector:**

* Learn how to select an **Element** on the page using a CSS style Selector.

query Selector() is a method used for searching and returning the very first **Element** within the document that matches the given selector.

**Changing HTML and CSS:**

* Learn how to alter HTML attributes.
* Learn about. style and .classlist.

Using JS we can modify the **HTML** **Elements** that we select. If the **Tag** displays test, like h3 and a p **Tag** then that text is something we can modify. The way we can do this by using innerText. We can also alter **HTML** **Elements** by changing their attributes. If for instance our **HTML** Element were an img **Tag**, we would be abler to change its src and alt attributes. If we were to change the src from one particular file to a different file it will change the image that is displayed on the page.

**Changing An Element’s Style:**

The **HTML Tags** we can alter also have **CSS** that is applied to them. This is something that we can change using **JS** as well. The **Element’s. style** is a **JS** object that contains all of the **CSS** styles we can apply for the **Element**. In this we change its color and also its background color. Note that we’re using a **CSS** that is normally hyphenated, in out **JS** we will need to rewrite it with alternating capitalization, for example margin-left becomes marginLeftSometimes we want to apply a number of **CSS** properties at once and while we can do that with. style it may be more efficient for us to use **CSS** class. This way whatever style are present in the **CSS** class dark-mode are applied to the button. note that an **Element** is allowed to have multiple classes so we treat the classlist as if it is like an **Array**. The classlist can be used with .add to push a new **Class** to the list and .remove to remover a specific **Class**. We can combine all of what we’ve learned make a button that we can toggle from one mode to another.

**Timeout:**

* Learn how to use setTimeout

Sometimes we want to apply a delay in out cold which can be done in **JS** by using a **Function** called **setTimeout**. The **setTimeout** **Function** takes two **Parameters**: (1) a **Function** to be called and (2) and the delay in milliseconds. The **Function** we provide is often referred to as **Callback Function** and we do not need to invoke it ourselves, we just need to put it’s name in as a **Parameter**. The **setTimeout** will call the callback **Function** after the amount of time provided.

**Intro To JS Lecture Notes:**

**Lecture Link:** <https://www.youtube.com/watch?v=gg-Tf4TTTgc> **JS** is the brain of the program. It gives Functionality. What is **JS**? A scripting language.

**String:** A **JS** **String** is zero or more characters written inside quotes.

var name = “Courtney”

**Boolean:** True or false.

**Operators:**

**+:** Adding

**-:** Subtraction

**\*:** Multiplication

**%:** Division

**Console Log:** Prints to the terminal.

**!=:** Not equal

**==:** Exactly equal. A comparison.  
**Array:** A list of items stored.

**Object:** A stand alone entity with properties and type.

**Function:** A block of code designed to perform a particular task.

**JS Events Lecture Notes:**

**Lecture Link:** <https://www.youtube.com/watch?v=vWS8VLy8NuI&t=617s>

**While Loop:** Will go until criteria is met.

**For Loop:** For a amount of time.

**=:** Assigning value to = sign.

**==:** Equals.

**if:** Use if to specify a block of code to be executed, if a specified condition is true.

**JS Likes Lecture Notes:**

**Lecture Link:** <https://www.youtube.com/watch?v=vqJq3-c0b1>

**Query Selectors:** Method returns the first **Element** that marches a **CSS** selector.

Document is the webpage.

**Object:** A container that holds different traits.