HTML NOTES

Resource:

MDN (Mozilla Developer Network)
VS Code Extensions: VS code icons, github theme, jellyfish theme, live preview,

Revisit mobile preview trick

The client (browser) sends a request for a webpage to the server. The server reads this message and responds by sending the index.html file back to the browser

DOM (Data Object Model) basically represents all the building blocks of a webpage and their structure e.g. images, divs, headings and so on. The browser after receiving index.html, starts constructing the DOM and requests resources that need to be loaded along the way

HTTP is a language for communication between client and server. Https has added encryption

URL is the address of a unique resource on the web. The URL always contains the scheme (https) and the authority (domain) and can optionally include a file path, parameters (?p=a1&q=a2) and an anchor (bookmark within the page)

Index.html is considered the default file for the homepage

Check out Chrome developer tools 🔥

Html tags

are used to wrap the various parts of a webpage
<html> start and end of page
<title> title of webpage. Metadata, not displayed
<body> content of page
<h1> heading 1
<p> paragraph
<head> wraps the metadata. <meta> metadata may include the charset, description, keywords, viewport, and author. Any external resources like css and JavaScript files are also linked within this head element
<a> Anchor tag, for creating links

Any html file should start with a **DOCTYPE** that tells the browser which version of html we're using.
<!DOCTYPE html> tells the browser we're using html 5

Use Emmet abbreviations in VS Code
html tag followed by **lang="en-US"** tells the browser what language the document is in

Tags:

part of the element. Opening and closing, enclosed within<>

Elements:

building blocks of a webpage: image, paragraph, heading etc
There are some elements which are self enclosing meaning they don't have a close tag. These don't have explicit content. They are either empty or their content comes from attributes, like <img src=heroImage.jpg>

An **attribute** contains additional information about the element

HTML COMMENTS

<!-- Tis I, a comment -->. These are not displayed and are ignored by the browser

Inline elements:

flow horizontally (words), anchor tags, spans, they are only as wide as their elements and cannot accept margins and different sizes

Block elements:

flow vertically, stacked (new lines, paragraphs, heading)
Linking css file: <link>
Linking js file: <script:src>

THE <div> ELEMENT

It's a block level element that works as a generic container. Useful for grouping content
Adding customizations: <div style='color: color-code ; border : 2px

Adding Links

Done with the anchor tag: <a>
<a href="https://www.google.com"> Open Google </a>
To open link in new tab: <a target="_blank" href="https://www.google.com"> Open Google </a>

Strong element an inline element that indicates that the particular piece of content is very important. Reflects urgency, seriousness
Emphasis element to put emphasis on a content so that it's italicized and screen readers pause at it
Side comment element
The small element: used for small text like captions

Quote element

<q> OR <blockquote>
<blockquote cite="ref link">

Separator element

Line break: <br>
Thematic break/ horizontal rule: <hr>

Lists

Ordered list: numbered list <ol>
Unordered list(bullets): <ul>
Lists can be nested in html

For **referencing images** a level up the current folder:

<img src="../images/examples/photo.jpeg">

Form element

<form>, the action attribute is where we specify where the form information will be sent
Leaving action blank submits the form to the current page
The method represents how the form information will be sent
disabled- an attribute for input element that disables any input from the user. A default value can be displayed in the field by using the value attribute

Select Element: Creates a drop-down menu with options:

```
<select name=”fruits>
        <option value = “Apple”>Apple</option>
<option value = “Banana”>Banana</option>
<option value = “Grapes”>Grapes</option>
```

Table element

<table>, <tr> table row
<th> table heading
<td> table data
colspan attribute: defines how many columns the element will occupy
rowspan attribute: defines how many rows the element will occupy
<caption> tag: Adds a caption/ main heading for the table. Write under table tag

To use **radio buttons**, give type= radio under the input tag
<textarea> tag: To control the situation size of input field box by specifying rows and columns

```
<textarea name="comment" rows="4" cols="50" >
Enter your comment<\textarea>
```

Span element: Works just like a div, but it is INLINE and is used to wrap around other elements to form a section.

ID ATTRIBUTE:

An ID is an attribute, a unique identifier assigned to only one HTML element within a page. It is often used for unique styling and JS manipulations.

CLASS ATTRIBUTE:

It lets you give the same name to multiple HTML elements. That way, you can easily change their look or behavior all at once. They are not unique. One element can be assigned more than one class.
<div id=”firstdiv” class=”red”>First</div>
<div id=”seconddiv” class=”red”>Second</div>
To reference class in CSS file: Add **(.)** before class name

To reference id in CSS file: Add **(#)** before id name

❖ If we add an id in the URL preceded by **#**, then the browser takes us directly to the section having that particular id instead of taking us to the start of the page.

Video, Audio & Media

- Video: Added using <video> tag. Adding the **controls** attribute enables us to interact with the video player.
- Other attribs: loop autoplay muted poster(like a thumbnail)
  <video src=”video.mp4” width=”” height=”” controls></video>
- Audio: <audio src=”audio.mp3” controls></audio>
- Attributes same as video but also has **preload** attrib with these values:
  - None, metadata, auto
- Embedding SVG files: <svg width=”500” height=”500”><circle cx=”50” cy=”50” stroke-width=”3” stroke=”black”></svg>
- iFrames (inline frames): allow us to embed another HTML webpage within our current page
  - <iframe src=”URL” width=”” height=””> </iframe>

Semantic Tags

Semantic tags tell our browser or web crawlers about our site’s content and the purpose of each tag. It is important for SEO purposes e.g. a text wrapped in <h1> has different semantic meaning than a text with same size as h1 but wrapped in a <span>. They don’t affect the content, only wrap around it.

Tags:

➔ <header>
➔ <footer>
➔ <article>
➔ <section>
➔ <main>
➔ <figure> or <figcaption>: images, diagrams, or charts
➔ <nav>
➔ <aside>
➔ <time>: Used for time-related information

<pre> tag:

The new lines and spaces between whatever is written inside this element are preserved. Examples present in ‘Notes’ folder

<code tag>:

Used to display code in the webpage, with appropriate font.

<role=”” attribute>

To increase the page accessibility, the *role* attribute can be used to indicate the purpose behind an element on the page to assistive technologies

CSS NOTES

Definition:

CSS stands for Cascading Style Sheets. It’s a stylesheet language used to describe visual presentation of a webpage written in html.

Selectors:

Selectors are used to select elements of a certain class, id, or related group in order to target and style them. It basically ‘selects’ the elements that you want to style.

- Simple selectors (select elements based on name, id, class)
- Combinator selectors (select elements based on a specific relationship between them)
- Pseudo-class selectors (select elements based on a certain state)
- Pseudo-elements selectors (select and style a part of an element)
- Attribute selectors (select elements based on an attribute or attribute value)

Declarations:

Declaration is the text written within the {} of a selector. It consists of property value pairs.

Inline CSS:

Added to html using <style> tag, with individual elements or classes.

Internal CSS:

Put all the CSS code within the <style> tag in the <head> element of the html document.

External CSS:

Create a separate stylesheet and link it to the html document.

Types of Selectors

Selectors can also be written combined: h1.class_name { }; and this combines their specificity.

Class selector:

precede by a full stop (.) in stylesheet.

ID selector:

precede by a hashtag (#) in stylesheet.

Child selectors:

Used to select an element within another element.

```
div > p {
  Color: white
  Margin: 10px
}
```

The path provided must match the actual nesting. If p is in a section that is within a div, changes won't apply

Descendant selectors:

Ignore sequence of nesting and applies changes if child is present within parent
div p { font-size: 18pt}
Will work even if p is wrapped within another element

Universal selectors:

apply to all elements. *{ }

Pseudo selectors:

Apply to links.
a:link {*Change link styling*\}
a: visited {}, a: active, a:hover

Attribute selectors:

E.g, <input type=”text”/>
In stylesheet: [type=”text] { }
OR <h1 class=”heading” data-x = “a”>Heading</h1>
In stylesheet: [data-x] { /*styling*/ }

Giving multiple classes to one element:

div { class="class1 class2"}

Box Sizing Property:

border-box: Include the padding and border width in the total width specified for the element. w: 200; b:2px; padding: 8px; content width -> 190px

Fonts

Font-family:

CSS tries to set the first font appearing in a font family, if it's available in the system. If it is not installed, it moves on to the next listed font. The selected font is main font & the rest are called 'fallback fonts'.

Using External Fonts:

There are two ways to add external fonts to the document:

1. Including the font - copy the font's import code and paste in stylesheet or inside style tag.
2. Using link - copy the link and paste in document under link tag in head. Exclude from the style sheet.

Colors

Colors can be represented in following ways:

1. Color keywords
2. Hex codes
3. RGB
4. RGBA (r,g,b,alpha)
5. HSL

CSS Specificity:

Used to resolve override conflicts. Tells how specific an element is.

- Selectors with higher specificity get precedence.
- If importance level is same, order of the selectors is considered.
- !important has top-most specificity
- Inline styles have second-highest specificity, followed by id.

User agent stylesheet:

Styles applied by the browser automatically.

Responsive units: px, em, rem, vh, vw, max-width, max-height, vmin, vmax

Display property:
inline, inline-block, none,

Visibility property
Inline elements do not respect top padding
Box shadow
Text shadow
Outline
List styling

Overflow property:

overflow : scroll
overflow : auto
overflow: clip
overflow-x, overflow-y
White space property:
white-space : nowrap
text-overflow property
Position property:
position:static, sticky, absolute, fixed, relative
(transform, filter, perspective properties can also make element appear positioned

CSS Global Variables:

```
:root {
  –color : blue;
  –seccolor: black;
```

How to use a variable
background-color: var(--color)

CSS Local Variables:

Same syntax. Declare within block of selector

Media queries

@media only screen and (expression)

Grouping selectors:

```
a,p,h1{
  font-size: 2rem;}
```

Before & after pseudo selectors:
::before, ::after

CSS Flexbox:

- align-items: when flex items occupy a single row or column
- align-content: when flex items occupy more than a single row or column
- justify-content: align items according to the flex direction axis
- Gap property
- Row-gap

- Column-gap

Hiding Content:

This will be visible to screen readers only. Do this by setting these properties:

```
border: 0;
clip: rect(1px, 1px, 1px, 1px);
clip-path: inset(50%);
height: 1px;
width: 1px;
overflow: hidden;
whitespace: nowrap;
position: absolute;
padding: 0;
margin: -1px;
```

CSS Clip property:

The CSS clip property is used to define the visible portions of an element. The clip-path property determines the shape the clip property should take. Set the clip-path property to the value of inset(50%), forming the clip-path into a rectangle within the element.

Clip: rect(1px, 1px, 1px, 1px)

First-of-type pseudo selector:

The :first-of-type pseudo-selector is used to target the first element that matches the selector.

Last-of-type pseudo selector:

The :last-of-type pseudo-selector does the exact opposite - it targets the last element that matches the selector.

The calc() function:

The calc() function is a CSS function that allows you to calculate a value based on other values. For example, you can use it to calculate the width of the viewport minus the margin of an element:

```
.example {
  margin: 10px;
  width: calc(100% - 20px)

}
```

!important:

You can use the !important keyword to ensure these properties are always applied, regardless of order or specificity.

Border Collapse property:

Set the border-collapse property to collapse, which will allow cell borders to collapse into a single border, instead of a border around each cell.

:nth-of-type() pseudo-selector

The :nth-of-type() pseudo-selector is used to target specific elements based on their order among siblings of the same type.

CSS Positioning:

CSS positioning lets you set how you want an element to be positioned in the browser. It has a position property you can set to static, absolute, relative, sticky or fixed.

Once you set the position property of the element, you can move the element around by setting a pixel or a percentage value for one or more of the top, right, left, or bottom properties.

Position: static

static is the default positioning for all elements. If you assign it to an element, you won't be able to move it around with top, right, left, or bottom.

Position: absolute

The next position property is absolute. When you use the absolute value for your position property, the element is taken out of the normal flow of the document, and then its position is determined by the top, right, bottom, and left properties.

Position: fixed

fixed is a position property value that lets you make an element fixed to the page no matter where the user scrolls to on the page.

Position: sticky

The last position property value is sticky. sticky positioning is a hybrid of relative and fixed positioning. It allows an element to stick to a specific position within its containing element or viewport, based on the scroll position.

CSS Transform:

The transform property allows you to modify the shape, position, and size of an element without changing the layout or affecting the surrounding elements. It has functions such as:

- translate(),
- rotate(),
- scale(),
- skew(),
- matrix().

The **::before** selector creates a pseudo-element which is the first child of the selected element, while the **::after** selector creates a pseudo-element which is the last child of the selected element.

The content property is used to set or override the content of the element. By default, the pseudo-elements created by the ::before and ::after pseudo-selectors are empty, and the elements will not be rendered to the page. Setting the content property to an empty string "" will ensure the element is rendered to the page while still being empty.

Media Query:

The @media at-rule, also known as a media query, is used to conditionally apply CSS. Media queries are commonly used to apply CSS based on the viewport width using the max-width and min-width properties

Logical operators can be used to construct more complex media queries. The and logical operator is used to query two media conditions.

Loading Attribute:

The loading attribute on an img element can be set to lazy to tell the browser not to fetch the image resource until it is needed (as in, when the user scrolls the image into view)

Referer HTTP header:

The Referer HTTP header contains information about the address or URL of a page that a user might be visiting from. This information can be used in analytics to track how many users from your page visit freecodecamp.org

Minmax function:

Use the **minmax** function to make your columns responsive on any device. The minmax function takes two arguments, the first being the minimum value and the second being the maximum.

Grid-column property:

The grid-column property tells the grid item which grid line to start and end at.

Object-Fit Property:

The object-fit property tells the browser how to position the element within its container. In this case, cover will set the image to fill the container, cropping as needed to avoid changing the aspect ratio.

Grid-auto-flow property:

This property takes either row or column as the first value, with an optional second value of dense.
grid-auto-flow uses an auto-placement algorithm to adjust the grid layout. Setting it to column will tell the algorithm to create new columns for content as needed. The dense value allows the algorithm to backtrack and fill holes in the grid with smaller items, which can result in items appearing out of order.

Place-items property:

The place-items property can be used to set the `align-items` and `justify-items` values at the same time. The place-items property takes one or two values. If one value is provided, it is used for both the align-items and justify-items properties. If two values are provided, the first value is used for the align-items property and the second value is used for the justify-items property.

Style the text within your #years element by creating a #years span[class] selector. The span[class] syntax will target any span element that has a class attribute set, regardless of the attribute's value.

CSS ANIMATIONS:

Transform Origin:

The `transform-origin` property is used to set the point around which a CSS transformation is applied. For example, when performing a rotate (which you will do later in this project), the transform-origin determines around which point the element is rotated.

```
transform-origin: 0% 0%;
```

This will offset the origin point by 0% from the left and 0% from the top, setting it to the top left corner of the element

Keyframes Rule:

The `@keyframes` at-rule is used to define the flow of a CSS animation. Within the @keyframes rule, you can create selectors for specific points in the animation sequence, such as 0% or 25%, or use from and to to define the start and end of the sequence.

`@keyframes` rules require a name to be assigned to them, which you use in other rules to reference. For example, the `@keyframes freeCodeCamp { }` rule would be named freeCodeCamp.

You now need to define how your animation should start. To do this, create a 0% rule within your @keyframes wheel rule. The properties you set in this nested selector will apply at the beginning of your animation.

```
@keyframes freecodecamp {
  12% {
    color: green;
  }

}
```

Animation-Name:

The `animation-name` property is used to link a @keyframes rule to a CSS selector. The value of this property should match the name of the @keyframes rule.

Animation-Duration:

The `animation-duration` property is used to set how long the animation should sequence to complete. The time should be specified in either seconds (s) or milliseconds (ms).

Animation-Iteration-Count:

The `animation-iteration-count` property sets how many times your animation should repeat. This can be set to a number, or to infinite to indefinitely repeat the animation. Your Ferris wheel should never stop, so set the .wheel selector to have an animation-iteration-count of infinite.

Animation-Timing-Function:

The `animation-timing-function` property sets how the animation should progress over time. There are a few different values for this property, but you want the Ferris wheel animation to run at the same rate from start to finish. Set the animation-timing-function to linear in your .wheel selector.

Skew Transform:

To make the mountain look more like a mountain, you can use the `skew transform` function, which takes two arguments. The first being an angle to shear the x-axis by, and the second being an angle to shear the y-axis by.