HTML

- Comments
  -Comments start with <!-- and finish with → in HTML

- Elements
  - Delete unnecessary elements to save space
    - Descriptive HTML Tags
      - header
      - footer
      - nav
      - article
      - section
      - Main (helps engines find the main content on the page.
        Ex. use of main
        <main>
        hello world
        </main>

- Images
  - -Adding an image
    -Use img element
    -point to specific url of the image by src
    Ex.
    <img src="https:.....">

    -img should have an alt feature for improved accessibility and is displayed if image is not displayed.

    Ex.
    <img src="https:....." alt="sample text">

- Linking to External Pages with Anchors
  -Use anchor to link to external content outside the current page
  -anchors need a web address called with an attribute called href
  Ex
  sample info

- Linking to Internal Sections of a page with anchors
  -To make internal a link have a # after href and add an id attribute to item being linked to
  Ex: target
  <footer id="footer">text</footer>

- Nesting an Anchor Element in a Paragraph

-Ex:

Here is a <a target=”_blank”
hfref=”https:....>link to freecode for you to see this example

- Making Dead Links Using the Hash Symbol
  -ex href=”#””http:.....”>

- Turning an Image into a Link
  -nest image within an element
  Ex:

<img src="https://bit.ly/fcc-running-cats" alt="Three kittens running towards the camera.">

-always remember to use # to href to make it a dead link
The problem I completed-
<img src="https://bit.ly/fcc-relaxing-cat" alt="A cute orange cat lying on its back.">cat photos

- Create a Bulleted Unordered List
  - Starts with <ul> followed by …then
  - Ex:

    milk
    cheese

- Creating an Ordered List
  - Starts with an <ol> followed by <li>... and then
  - Ex:

    Shadow
    Sadie
    Remy

- Creating a Text Field
  - How to make a text input line - <input type=”text”>
    - Inputs are self closing

-
- Adding Placeholder Text to a Text Field

    - Create a placeholder text by- <input type=”text” placeholder=”this is placeholder text”>
- Creating a Form Element
    - They submit data to a server by using the form element.
      Ex:
      <form action=”/url-where-you-want-to-submit-form-data”></form> then you can add text input by adding *<input type="text" placeholder="cat photo URL">*
      To the form to get

      Ex: <form action=”/submit-cat-photo”><input type="text" placeholder="cat photo URL"></form>

- Add a Submit Button to a Form
    - A submit button on a form will send data to specific URL with the form’s action attribute
    - Ex: <button type=”submit”>this is the button that submits the form</button>

- Use HTML5 to Require a Field
    - Just add required attribute within the input element
    - Ex: <input type=”text required>.....

- Creating a Set of Radio Buttons
    - Radio Buttons are used for questions when you want a user to give one answer out of multiple options.
    - Radio buttons are an input type and they all have the same name if they are related.
    - Each can be nested within its own element
        - Ex:
          <label>
            <input type=”radio” name=”indoor-outdoor”>Indoor
          </label>

            - Ex: ID
              <label for =”indoor”>
              <input id=”indoor” type=”radio”
              name=”indoor-outdoor”>Indoor
              </label>

- Create a Set of Checkboxes
    - Checkboxes are used commonly for questions with multiple answers
    - They are a type of input
    - Can be nested in its own label

- By wrapping an input element into a label it will associate the checkbox input with the element around it
- They should all have the same name characteristic
  Ex:
  <label for="loving"><input id="loving" type="checkbox" name="personality">
  Loving</label>

- Check Radio Buttons and Checkboxes by Default
  - Set checkbox or radio button using checked
    Ex:
    <input type=”radio” name=”test-name” checked>

- Nest Many Elements within a Single div Element
  - Div also called a division element is a container
  - You have to open and close it ….

- Declare the Doctype of an HTML Document
  - You tell the browser what version of HTML your page is currently using
  - You do this by
    Ex- <!DOCTYPE html>
    <html>

    <!-- Your HTML code goes here →

    </html>
- Define the Head and Body of an HTML Document
  - What the markup info is, it is put in the head tag
  - The body tag is where the info the user sees is stored.
  - Ex: <html>
    <head>
    </head>
    <body>
    </body>
    </html>

CSS

- Changing the Color of theText
  - Change it in the style
    Ex: <h2 style=”color”: blue;”>sample text</h2>

- sing CSS Selectors to Style Elements

- Use style blocks
  ```
  <style>
    h2{color:red;}
  </style>
  ```

- Use a CSS Class to Style an Element
  - Classes are styles that can be reused and be added to HTML elements
  - Ex:
    ```
    <style>
      .blue-text {
        Color: blue;
      }
    </style>
    <h2 class="blue-text">
    ```

- Style Multiple Elements with a CSS Class
  - Classes allow you to use the same style on multiple HTML parts.
  - Ex;
    ```
    <p class="red-text">
    ```

- Change the Font Size of an Element
  - Font size is controlled by font-size
  - Ex:
    ```
    h1{
    Font-size: 30px;
    }
    ```

- Set the Font Family of an Element
  - Set font by using the font-family property
  - Ex;
    ```
    h2 {
      Font-family: arial;
    }
    ```

- Importing a Google Font
  - Copy the URL to Google Fonts (Which is Free)
  - Ex:
    ```
    <link href="https://fonts.googleapis.com/css?family=Lobster" rel="stylesheet"
    type="text/css">
    ```

- Specify How Fonts Should Degrade
  - When a font is not available in the browser you can degrade it to another font
  - Generic fonts: monospace,serif, sans-serif

- Ex:
  ```
  p {
    Font-family: Helvetica, sans-serif;
  }
  ```

- Size Your Images
  - Width controls the element's width and uses px to specify the size
  - Ex:
    ```
    <style>
      .larger-image{
          Width: 500px;
    }
    </style>
    ```

- Adding Borders Around Your Elements
  - Use style, color, width
  - Ex:
    ```
    <style>
      .thin-red-border {
        border-color: red;
        border-width: 5px;
        border-style: solid;
      }
    </style>
    ```

    And you can apply it to more classes with
    ex:
    ```
    <img class="class1 class2">
    ```

- Add Rounded Corners with border-radius
  - Add rounded corners with border-radius
  - Ex:
    ```
    Border-radius: 10px;
    ```

- Make Circular Images with a border-radius
  - Use percentages when doing this
  - Ex:
    ```
    Border-radius: 50%;
    ```

- Give a Background Color to a div Element
  - Set background with background-color
  - Ex:
    ```
    .green-background {
    ```

Background-color: green;
}

- Set the id of an Element
  - Don't give more than one element the same id
  - Ex:
    Cat-photo-app:
    <h2 id="cat-photo-app">

- Use an id Attribute to Style an Element
  - You can style id attributes like classes
  - An id is not reusable (use on only one element)
  - Ex:
    #cat-photo-element {
    Background-color: green;
    }

- Adjusting the Padding of an Element
  - Properties
    - Padding-controls space between content and border

- Adjust the Margin of an Element
  - Properties
    - Margin- controls space between border and other elements around it

- Add at Negative Margin to and Element
  - Negative margin makes the element larger

- Add Different Padding to Each Side of an Element
  - Properties
    - Padding-top, padding-bottom, padding-right, padding-left

- Add Different Margins to Each Side of an Element
  - Properties
    - Margin-top, margin-right, margin-bottom, margin-left

- Using Clockwise Notation to Specify the Padding of an Element
  - Can be specified in one line
    Ex:
    Padding: 10px 20px 10px 20px

- Using Clockwise Notation to Specify the Margin of an Element

- Margin can be specified in one line as well
  Ex:(clockwise notation)
  Margin: 10px 20px 10px 20px

- Using Attribute Selectors to Style Elements
  - [attract=value]
  - ex:
    [type='radio'] {
      Margin: 20px 0px 20px opx;
    }

    Ex:

    <style>
      [type='checkbox']{
      margin: 10px 0px 15px 0px;
    }
    </style>

- Understanding Absolute versus Relative Units
  - Em and rem(relative) in mm (absolute)
  - Em is based on an elements font size
  - ex:
    1.5em

- Style the HTML Body Element
  - Every html page has a body and it can be styled
  - Ex:
    Body {
      Background-color: black;
    }

- Inherit Styles from the Body Element
  - Other elements will inherit the body styles
    <h1 class="body">Hello World</h1>

- Prioritize One Style Over Another
    - Creating a class to override body since the body element can't be 2 colors
    - So make it it's own class
    - Ex:

      .pink-text {

        Color:pink;

      }

      <h1 class="pink-text">Sample text</h1>

- Override Styles in Subsequent CSS
    - When using multiple attributes to HTML elements do this

      Ex:

      class="class1 class2"

- Override Class Declarations by Styling ID Attributes
    - Give the h1 element an id of …-text
    - Ex:

      <h1 id='orange-text">
    - Make a declaration for the id element in the style element
        - Ex:

          #brown-text {

            Color:brown;

          }

          <h1 id="brown-text" class="....">text</h1>
- Override Class Declarations with Inline Styles
    - Use inline-style to change h1
    - Ex:

      <h1 style="color: green;">

- Override All Other Styles by using Important
    - Use ! important to make sure that the element will be specific to what you want
    - Ex:

      Color: red !important;

- Use Hex Code for Specific Colors
    - Ex:

      #000000 (model)

      Ex:

      Body {

      Color: #000000;

      }

- Use Hex Code to Mix Colors
    - F- highest number in hex for maximum brightness
    - 0- lowest number represents the complete absence of color
    - Ex:

      .text{

       Color: #FFA500

      }

- Use Abbreviated Hex Code
    - Hex code can be shortened
    - Ex: Instead of #FF0000 you can do #F00
- Use RGB values to Color Elements
    - RGB is another way to show colors

      Ex:

      rgb(0, 0, 0) (black)

      Or

      rgb(225, 225, 225)

      And

      Body {

        Background-color: rgb(225, 165, 0);

      }

- Use CSS Variables to change several elements at once
    - You can change many values at once by changing a single value
    - Ex:

      .penguin{

      --penguin-skin:gray;

      --penguin-belly:white;

      --penguin-beak:orange;

      (the rest of the info……..0

      }

- Create a Custom CSS Variable
    - Give it a name with -- in front of the name then give it a value

      Ex:

      --dolphin-skin: blue;

- Use a custom CSS Variable
    - To create a variable assign the value to other properties by referencing its name that it is given.
    - Ex:

      Background: var(--penguin-skin);

- Attach a Fallback value to a CSS Variable
    - If the given variable isn't valid attach a fallback so the browser can revert to that.
    - Ex:

      background: var(--penguin-skin, black);

- Improve Compatibility with Browser Fallbacks
    - Add a different background as a fallback

- Cascading CSS Variables
    - Use :root as a container for the whole page like in HTML having the body element

Applied Visual Design

- Create Visual Balance Using the text-align Property
    - Text-align
        - Text-align: justify; -causes all lines of text except the last line to meet the left and right edges of the line box.
        - Text-align: center;- centers text
        - Text-align: right; - aligns to the right
        - Text-align: left; - aligns to the left

- Adjust the Width of an Element Using the width Property
    - Specify image width
        - Ex:

          Img {

            Width: 220px;

          }

- Adjust the Height of an Element Using the height Property
    - Specify the height
        - Ex:

          Img {

           Height: 25px;

          }

- Use the strong Tag to Make Text Bold

  Ex:

  Font-weight: bold;

  And

   …..

- Use the u Tag to Underline Text
    - Ex:

      Text-decoration: underline;

- Use the em Tag to Italicize Text
    - font-style:italic;

      And

       …

- Use the s Tag to Strikethrough Text
    - <s>....</s>

      and
    - text-decoration: line-through;
- Create a Horizontal Line Using the hr Element
    - Hr is a self closing tab
    - <hr> (no closing tab needed)

- Adjust the background-color Property of Text
    - Use RGB or Hex codes

- Adjust the Size of a Header(h1 through h6) Versus a Paragraph Tag(<p>)
    - Remember that header tags should be larger than the font size of the paragraph tags!

- Adding a box-shadow to a Card-like Element
    - Applies one or more shadows to an element
    - Blur-radius
    - spread-radius
    - Ex:

      box-shadow: 0 10px 20px rgba(0,0,0,0.19), 0 6px 6px rgba(0,0,0,0.23);

More detail

#thumbnail {

/* offset-x | offset-y | blur-radius | spread-radius | color */

box-shadow: 0 10px 20px rgba(0,0,0,0.19), 0 6px 6px rgba(0,0,0,0.23);

}

- Decrease the Opacity of an Element
    - Opacity adjusts the transparency for certain items
    - 1-means not transparent at all
    - .5 - is half transparent
    - 0 -is completely transparent
- Use the text-transform Property to Make Text Uppercase
    - Text-transform (1st)
    - Lowercase
    - Uppercase
    - Capitalize
    - Initial
    - Inherit
    - None
- Set the font-size for Multiple Heading Elements

```
h1{
  font-size: 68px;
}
h2{
  font-size: 52px;
}
h3{
  font-size: 40px;
}
h4{
  font-size: 32px;
}
h5{
  font-size: 21px;
}
h6{
  font-size: 14px;
}
```

- Set the font-weight form Multiple Heading Elements

```
<style>
 h1 {
   font-size: 68px;
   font-weight: 800;
 }
 h2 {
   font-size: 52px;
   font-weight: 600;
 }
 h3 {
   font-size: 40px;
   font-weight: 500;
 }
 h4 {
   font-size: 32px;
   font-weight: 400;
 }
 h5 {
   font-size: 21px;
   font-weight: 300;
 }
 h6 {
   font-size: 14px;
   font-weight: 200;
 }
</style>
<h1>This is h1 text</h1>
<h2>This is h2 text</h2>
<h3>This is h3 text</h3>
<h4>This is h4 text</h4>
<h5>This is h5 text</h5>
<h6>This is h6 text</h6>
```

- Set the font-size of Paragraph Text
    - Ex:

      P{

      Font-size: 10px;

      }

- Set the line-height of Paragraphs
    - Ex:

      P {

      Line-height: 25px:

      }

- Adjust the Hover State of an Anchor Tag
    - Ex:after a {

      }

      Add

      A: hover {

      Color: blue;

      }

- Change an Element's Relative Position
    - Ex:

      P {

      Position: relative;

      Bottom: 10px;

      }

- Move a Relatively Positioned Element with CSS Offsets
    - Top
    - Bottom
    - Left
    - Right

- Ex:

```
H2{
 Position: relative;
Bottom: 10px;
Left: 15px
}
```

- Lock an Element to its Parent with Absolute Positioning
  - Absolute for the position property locks an element into place no matter what

- Lock an Element to the Browser Window with Fixed-Positioning
  - Fixed position won't move when a page is scrolled.

- Push Elements Left or Right with the float Property
  - Float properties are not in the normal page of a document so they are on the right or left side containing their element.
  - Ex:

```
#left {
 Float: left:
Width: 50%
}
```

(Same for right side)

- Change the Position of Overlapping Elements with the z-index Property
  - Z-index specifies the order of the elements on top of each other
  - Ex:

```
.first {
        Z-index: 2;
        Background-color: red;
        Position: absolute;
}
```

- Center an Element Horizontally Using the margin Property
  - Use Margin: auto;

- Learn about Complementary Colors
    - Use the colors hex codes

- Learn about Tertiary Colors
    - Also use hex codes

- Adjust the Color of Various Elements to Complementary Colors
    - Add colors to header, footer and buttons
    - Ex:

      Header{
        Background-color: orange;
        Color: white
      }

- Adjust the Hue of a Color
    - hsl(red) or hsl(0, 100%, 50%)
    - Background-color: hsl()
- Adjust the Tone of a Color
    - Background-color: hsl(180,80%,25%);
- Create a Gradual CSS Linear Gradient
    - Background: linear-gradient(90deg, red, yellow, rgb(204, 204, 255));

      Or
    - Have hex codes #ffcccc

- Use a CSS Linear Gradient to Create a Striped Element
    - Repeating-linear-gradient() just repeats the pattern of the gradient pattern and it allows a bunch of different values
    - Ex:

      0px [yellow—blend—blue] 40px [green—blend—red] 80px

      Yellow 0px,

      yellow 40px,

      Black 40px,

Green 40px,

Red 80px,

Black 80px,

- Create Texture by Adding a Subtle Pattern as a Background Image
    - Background: url(.....);
- Use the CSS Transform scale Property to Change the Size of an Element
    - Ex: p {

        Transform:scale(2);

      }
- Use the CSS Transform Scale Property to Scale an Element on Hover
    - P:hover{
    - Transform: scale(2.1)
- Use the CSS Transform Property skewX to skew an Element Along the X-Axis
    - SkewX() selects element on the x-axis(horizontal)
    - P {
    - Transform: skewX(-32deg);
- Use the CSS Transform Property skewY to Skeew an Element Along the Y-Axis
    - SkewY is the same as skewX but on the y-axis
- Create a Graphic Using CSS also for More Complex Shapes using CSS and HTML
    - Use all the tools from HTML and CSS that you acquired
- Learn How the CSS@keyframes and animation Properties Work
    - Animation-name sets the name
    - Animation-duration sets the length of time for it
    - @keyframes specifies what the outcome of the animation of the duration is
    - Ex.

      <style>

      div {

        height: 40px;

        width: 70%;

        background: black;

        margin: 50px auto;

        border-radius: 5px;

      }

```
#rect {
  animation-name: rainbow;
  animation-duration: 4s;



}
@keyframes rainbow{

  0%{
    background-color: blue;
  }
}
  50% {
    background-color: green;

  }
  100% {
    background-color: yellow;
  }

 </style>
<div id="rect"></div>
```

- Use CSS Animation to Change the Hover State of a Button
    - You can also use @keyframes to change the color of a button when its hovered over
    - Ex:

```
@keyframes background-color{
  100%{
        Background-color: #4791d0
        }
```

- Modify Fill Mode of an Animation

    - To keep highlighted use animation-fill-mode and it specifies the look applied to the element when the animation concludes
    - Ex: animation-fill-mode: forwards;
- Create Movement Using CSS Animation

```
<style>
 div {
   height: 40px;
   width: 70%;
   background: black;
   margin: 50px auto;
   border-radius: 5px;
   position: relative;
 }

#rect {
 animation-name: rainbow;
 animation-duration: 4s;
}

@keyframes rainbow {
 0% {
   background-color: blue;
   top: 0px;
   left: 0px;

 }
 50% {
   background-color: green;
   top: 50px;
   left: 25px;

 }
 100% {
```

```
  background-color: yellow;
  top: 0px;
  left: -25px;

 }
}
</style>

<div id="rect"></div>
```

- Create Visual Direction by Fading an Element Left to Right
    - Add opacity to .1 to the @keyframes
    - Ex:

```
@keyframes fade {
    50%{
        Left: 60%;
        Opacity: .1}
    }
```

- Animate Elements Continauly Using an Infinite Animation Count
    - Animation-iteration-count
    - Ex.

      Animation-iteration-count: infinite;
- Make a CSS Heartbeat using an Infinite Animation Count
    - Add the animation-iteration-count to a class value
- Animate Elements at Variable Rates
    - Adjust the percentages after the @keyframes
- Animate Multiple Elements at Variable Rates
    - Adjust the percentages for each different value
- Change Animation Timing with Keywords
    - Animation-timing-function
    - Speeds vary
        - Ease- slow throughout
        - Ease-out - quick then slow

        - Ease-in-speedy at the end
        - Linear- a constant speed
- Learn How Bezier Curves Work
    - The cubic-bezier uses Bézier curves which shows how the animation plays out
    - It has four main points
        - P0
        - P1
        - P2
        - P3
    - Then for one p it has anchor points
        - (X1, y1, x2, y2)
    - Ex: animation-timing-function: cubic-bezier(0.25, 0.25, 0.75, 0.75);
    - Speed(timing affects this as well)
- Use a Bezier Curve to Move a Graphic
    - Use the coordinates to do this like in the notes above
- Make Motion More Natural Using a Bezier Curve
    - Use @keyframes to slow or speed up the motions
- **Applied Accessibility**

- Add a Text Alternative to Images for Visually Impaired Accessibility
    - <Img src=“.....” alt=“CompanyLogo”>
- Use Headings to Show Hierarchical Relationships of Content
    - Go from <h1> - <h5>
- Jump Straight to the Content Using the Main Element
    - Main is used to wrap the primary content
    - Use

  <header>

  </header

  <main>

  </main>

  <footer>

  </footer>
- Wrap Content in the article Element

    - - groups content
    - <section> - groups related content
    - <article> - groups independent, self-contained content
- Make Screen Reader Navigation Easier with the Header Landmark
    - Use the header in the body
- Make Screen Reader Navigation Easier with the nav Element
    - Nav is an element that wraps around the main links on the page

Ex: <nav>

<h1></h1>

</nav>

- Make Screen Reader Navigation Easier with the footer Landmark
    - Footer primarily holds the copyrights and other info at the bottom of the page
- Improve Accessibility of Audio Content with the audio Element
    - Audio supports the control attribute
    - Ex:

<audio id="meowClip" controls>

<source src="audio/meow.mp3" type="audio/mpeg"/>

<source src="audio/meow.ogg" type="audio/ogg"/>

</audio>

- Improve Chart Accessibility with the figure Element
    - Figcaption goes inside figure Ex:

<figure>

<img src="roundhouseDestruction.jpeg" alt="Photo of Camper Cat executing a roundhouse kick">

<figcaption>

Master Camper Cat demonstrates proper form of a roundhouse kick.

</figcaption>

</figure>

- Improve Form Field Accessibility with the label Element
    - Label wraps around text for specific control of an item
    - For associates with the label with the form control
    - For attribute be the same as the id attribute

```
<form>
    <label for="name">Name:</label>
    <input type="text" Id ="name" name="name">
</form>
```

- Wrap Radio Buttons in a fieldset Element for Better Accessibility
    - Fieldset - the entire grouping of radio buttons
    - Ex: do this

```
<fieldset>
    <legend></legend>
    <input id=" " type="radio" name="levels" value=" ">
</fieldset>
```

- Add an Accessible Data Picker
- Text and submit inputs specify the date field
    - Ex:

```
<input type="date" id="pickdate" name="date">
<input type="submit" name="submit" value="submit">
```

- Standardize Time with the HTML5 datetime Attribute
    - Time tags wrap around text with (day, month dateth
    - Datetime can't be empty
    - Close the time tag
    - Ex:

```
<time datetime="2017-06-15">Friday, July 15<sup>th</sup></time>
```

- Make Elements Only Visible to a Screen Reader by Using Custom CSS
    - To have what will fit in the screen in the window so some is offscreen
    - Ex:

```
.sr-only {
    Position: absolute;
    Left: -10000px;
    Width: 1px;
    Height: 1px;
    Top:auto;
    Overflow: hidden;
```

}

- Display:none; and visibility:hidden; hides content from everyone
- Improve Readability with High Contrast Text
    - Adjust colors so it improves visibility with a ratio
- Avoid Color Blindness Issues by Using Sufficient Contrast
    - Adjust lightness percentages in hsl()
- Avoid Color Blindness Issues by Carefully Choosing Colors that Convey Information
    - This is just a quick bit of info to keep in mind
- Give Links Meaning by Using Descriptive Text
    - Move the anchor tags (a) so they describe the links
    - info is here
- Make Links Navigatable with HTML Access Keys

  The access key attribute specify a shortcut to go to an element
    - Add it to a link to quickly navigate
    - Ex:

      <button accesskey="b">important button</button>

      <h2> text</h2>
    - And abbreviate to a single letter like Garfield= "g"

- Use tabindex to Add Keyboard Focus to an Element
    - Tabindex values can be positive negative or zero
    - focus
- Use tabindex to Specify the Order of Keyboard Focus for Several Elements
    - Tabindex can specify the exact order of the elements
    - Setting tabindex="1" will focus on the first element first then it cycles through the rest
    - Ex:

      I get it first

      I get it second
- **Responsive Web Desigh Principles**

- Create a Media Query
    - Media Queries present content based on different view sizes

- Ex: when the width is less than or equal to 100px:
- @media (max-width: 100px)
- Ex:When the device is greater than or equal to 350px:
- @media (min-height: 350px)

- Make an Image Responsive
  - Instead of applying an absolute width to an element
  - Do this
    - Img{
      Max-width: 100%;
      Display: block;
      Height: auto;
      }
- Use a Retina Image for Higher Resolution Displays
  - For retina displays define width and height as half of their original values
  - Ex: only using half of the original
    <style>
      Img { height: 250px; width: 250px;}
    </style>
    <Img src=“.....” alt=“.....”>]
- Make Typography Responsive
  - Use these instead of em or px for text size
    - Vw: 10vw would be 10% of the viewports width
    - Vh: 3vh would be 3% of the viewports height
    - Vmin: 70vmin would be 70% of the viewports smaller dimension(height vs width)
    - Vmax: 100vmax would be 100% of the viewports bigger dimension(height vs width)
    - Ex:
      - P{
        Width: 75vmin;
        }
- **CSS Flexbox**

- Use display: flex to Position Two Boxes
    - Putting the CSS Property display: flex; on an element lets you use other flex properties to construct a page that is responsive
    - Ex: adding display to a #box-container

      <style>

      #box-container{

      Height: 500px;

      Display:flex;

      }

      </style>
- Add Flex Superpowers to the Tweet Embed
    - Add display:flex; to

      Header
        - The headers .profile-name
        - The headers .follow-btn
        - Headers h3 h4
        - The footer
        - Footer .stats
    - Ex:

      <style>

      Header{

      Display: flex;

      }

      </style>

- Use the flex-direction Property to Make a Row
    - Adding display:text turns into a flex container which can allow you to arrange the result of the element(s) into columns or rows
    - Then add flex-direction to the original(parent) and set it to row or column
    - Row= horizontal column= vertical
    - The default flex-direction is a row
    - Then add flex-direction to a #box-container to then add a value called row-reverse

- Ex:

```
<style>
    #box-container {
        Display: flex;
        Height: …;
        flex-direction: row-reverse;
    }
```

- Apply the flex-direction Property to Create Rows in the Tweet Embed
  - You can add flex-direction to headers and footers so that their (children) can be arranged as rows
  - Ex:

```
<style>
    Header {
        Display: flex;
        Flex-direction: row;
    }
</style>U
```

- Use the flex-direction Property to Make a Column
  - <style>

```
    #box-container {
        Display: flex;
        Height: ...px;
        Flex-direction: column;
    }
</style>
```

- Apply the flex-direction Property to Create a Column in the Tweet Embed
  - Add flex direction to headers .profile-name to set its value to a column
  - Ex:

```
<style>
    Header .profile-name{
        Display: flex;
        Flex-direction: column;
        margin -left: 10px;
```

}

- Align Elements Using the justify-content Property
    - Justify-content has several options for spacing flex items side by side from right to left
    - Justify-content: center; aligns all flex items in the center of the flex container
    - Flex-start - aligns items to start of the flex container
    - Flex-end - aligns to the end of flex container
    - Space-between - aligns to center with extra space around items
    - Space-around -first and last items are not locked to the containers edges
    - Ex:

```
<style>
    #box-container {
        Background: gray;
        Display: flex;
        Height: 500px;
        Justify-content: center;
        }
```

- Use the justify-content Property in the Tweet Embed
    - You can also add justify-content to the headers .profile-name
- Align Elements Using the align-items Property
    - Align-items property tells CSS to and how to push items up or down in the container(row) and for(column) left or right in the container
    - Different values for align-items
        - Flex-start- aligns to the start of the flex container
        - Flex-end- aligns to the end of the flex container
        - Center- aligns to the center
        - Stretch-it stretches the items to fill the container
        - Baseline- aligns items to where the text would sit on
    - Ex:

```
<style>
    #box-container {
        Background: gray;
        Display: flex;
```

```
height: 500px;
Align-items: center;
}
```

- Use the align-items Property in the Tweet Embed
    - Align-items can be applied to a bit of tweet embeds to align the flex items inside of it
    - Ex:

```
Header .follow-btn {
    Display: flex;
    Align-items: center;
    Margin: auto;
}
```

- Use the flex-wrap Property to Wrap a Row or Column
    - Flex-wrap allows CSS to wrap items meaning moving elements to a brand new row or column
    - Options for the direction of the Wrap
        - Nowrap- doesn't wrap items
        - Wrap - wraps left-right or top to bottom
        - Wrap-reverse -wraps items from bottom to top or right to left
    - Ex:

```
<style>
    #box-container {
        Background: gray;
        Display: flex;
        height:100%,
        Flex-wrap: wrap;
    }
```

- Use the flex-shrink Property to Shrink Items
    - Flex-shrink lets an item shrink into a flex container if the container is too small
    - It's values are numbers the higher the number
        - The higher the number the smaller it is
    - Ex:

```
<style>
```

```
#box-1 {
        Background-color: red;
        width: 100%;
        Height: 200px;
        flex-shrink: 1;
}
```

- Use the flex-grow Property to Expand Items
    - Flex-grow is the opposite of flex-shrink
    - The higher the value the larger it gets
    - Ex:

```
#box-1 {
        Background-color: red
        Height: 200px;
        Flex-grow: 1;
}
```

- Use the flex-basis Property to Set the Initial Size of an Item
    - Flex-basis specifies the original size of an item before flex-shrink or flex-grow
    - Units used are used with all of the other properties
    - Ex:

```
#box-1 {
Background-color: red
Height: 200px;
Flex-basis:10em;
}
```

- Use the flex Shorthand Property
    - All flex properties(flex-grow flex-shrink flex-basis) can be set together with the flex property
    - Don't put the values in ()
    - Ex:

```
Flex: 1 0 10px;
        And
The order is flex:(grow, shrink, basis);
        And
```

```
#box-1 {
        Background-color: red;
        flex: 2 2 150px;
        Height:......;
}
```

- Use the order Property to Rearrange Items
    - The order property is used with CSS to let it know how the flex items are in order to appear in the container
    - Ex:

```
#box-1{
        Order: 1;
        ……..
        }

#box-2{
        order:2;
        ……..
        }
```

- Use the align-self Property
    - Accepts the same align-items values and it will overpower any value from the align-items property
    - Uses other values like flex-end as well
    - Ex:

```
#box-1 {
        Align-self: flex-end;
        }
```

- **CSS Grid**

- Create Your First CSS Grid
    - Turn HTML elements into a grid by setting the display property to grid
    - This allows you to use every other grid associated properties]

- Ex:

```
.container {
    Display:grid;
    …….
    }
```

- Add Columns with grid-template-columns
  - Use the grid-template-columns to add columns on a grid container
  - Ex:

```
.container {
    display: grid;
    Grid-template-columns: 50px 50px;
}
```

- Add Rows with grid-template-rows
  - To adjust rows manually and it works the same way as grid-template-columns, but instead with rows
  - Ex:

```
.container {
    Display:grid;
    grid-template-rows: 50px 50px;
}
```

- Use CSS Grids units to Change the Size of Columns and Rows
  - You can use either relative or absolute units like (px or em)
    - Fr- sets the row/column to a fraction of open space
    - Auto-sets the row/column to width or height automatically
    - % - adjusts the row/columns to the percent width of the container there
  - Ex:

    Grid-template-columns: auto 50px 10% 2fr 1fr; (creates 5 columns and it goes 1-5 with different widths throughout this example)(it can be more than 5 columns)

- Create a Column Gap Using grid-column-gap
  - Ex:

    Grid-column-gap: 20px;

    (Creates a 20px gap between every single column)

- Create a Row Gap Using grid-row-gap
    - Works the same way as the column version above but it begins with grid-row-gap instead
- Add Gaps Faster with grid-gap
    - Grid-gap is basically row and column grid-gaps combine
    - Grid-gap goes with row then column
    - Ex:

        Grid-gap: 10px 20px;
- Use grid-column to Control Spacing
    - The grid-column is the first property to use on the actual grid
    - Ex:

        Grid-column: ⅓;

        Which makes a grid with 3 spaces per column
- Use grid-row to Control Spacing
    - Makes the rows on the grid
    - Ex:

        Grid-row: ½;
- Align an Item Horizontally using justify-self
    - (For Horizontal use)Change justify-self's default value of default to either since it will automatically fill the whole width of the cell
        - Start- content aligns to left cell
        - Center- content aligns to center cell
        - End- content aligns to right cell

- Align an Item Vertically using align-self
    - Use self-align for vertical use
    - And it can use the same property names as the justify-self (start center end)
    -
- Align All Items Horizontally using justify-items
    - Use justify-items to align many things horizontally (use start center end)
    -
- Align All Items Vertically using align-items

- Use for aligning many things vertically at once (start center end can be used again)

- Divide the Grid into an Area Template
  - Group cells together in an area and give it a name by using grid-template-areas on the container
  - Ex: (use this template to reference placing or what to change in the grid)

```
Grid-template-areas:
    "Header header header"
    "advert content content"
    "footer footer footer";
```

    - Header - top 3 cells merged together
      - Footer - bottom 3 cells merged together and it makes 2 areas in the middle row called advert and content
      - Use a period to designate an empty cell on the grid

- Place Items in Grid Areas Using the grid-area Property
  - Ex:

```
.item1 {
    grid-area: header;
    }
```

- Use grid-area Without Creating an Areas Template
  - Ex:

```
Item1 {
    grid-area: 1/1/2/4;
    }
```

- Reduce Repetition Using the repeat Function
  - How to enter a massive amount of rows and columns at once with repeat
  - Rows- grid-template-rows: repeat(100, 50px);
  - Columns - grid-template-columns: repeat(2, 1fr 50px) 20px; or grid-template-columns: 1fr 50px 1fr 50px 20px;
- Limit Item Size Using the minmax Function
  - Minmax is used to make a restriction on the size of an item
  - Ex:

Grid-template-columns: 100px minmax(50px, 200px); and repeat(3, minmax(90px, 1fr));

- Create Flexible Layouts Using auto-fill
    - Repeat has an option names auto-fill to insert as many rows and columns as needed
    - Ex:

        grid-template-columns set to

        Repeat(auto fill, minmax(60px, 1fr));
- Creat Flexible Layouts Using auto-fit
    - Works identically to auto-fill but it is used when a container's size exceeds size of all items combined
    - Ex:

        Grid-template-columns set to

        Repeat(auto-fit, minmax(50px, 1fr));

- Use Media Queries to Create Responsive Layouts

  Change the grid with headers when it gets to a certain size

```
@media (min-width: 400px){
  .container{
   /* change the code below this line */

   grid-template-areas:
    "header header" <--
    "advert content"
    "footer footer";
```

- Create Grids within Grids
    - Add to an item grid-template-columns and display
    - Ex:

```
.Item2{
        Background: …;
        Grid-area: content;
        Grid-template-columns: auto 1fr;
        Display: grid;
```

```
}
```