Week 9 - Responsive Design Review

Monday

1. What is a CSS rule?
    - A CSS rule consists of a selector and curly braces around property-value combinations.
2. How to import other CSS files into your CSS file
   How to link a stylesheet into an HTML page
    - css into css : with use of the `@import` tag!
    - css into html : by using the `link` HTML element.
        - The link element has to have the `rel` and `href` attributes configured like so:

          ```
          <link rel="stylesheet" href="file.css">
          ```

3. Explain how CSS rules are applied based on their order and specificity, and be able to calculate the specificity of CSS rules and determine which rule override the properties of another.
    - `IDs` are unique, thus are considered highly specific since they always target one element.
    - Calculating the `score specificity` of selectors, and the one with the highest score is the most specific.
    - `in-line styling > id > pseudo-class selectors > class > tag name`
4. Write "combinators" to create compound selector statements to target specific elements
    - *Direct child selector*: the carrot, `>` means a **direct** child.
    - *All descendants/children selector*: denotes with a `space` in between selectors
    - *Multiple items selection*: `comma` dileniated selectors (ex-> body, div, html {})
5. Explain and apply pseudo selectors for specific elements in specific states (i.e. :hover)
    - See example from Monday's code demo:
      html:

      ```
      <a href="https://google.com">Link</a>
      ```

      css:

      ```
      a:hover {
          font-family: "Roboto Condensed", sans-serif;
          color: #4fc3f7;
          text-decoration: none;
          border-bottom: 2px solid #4fc3f7;
      }
      ```

- The **not** pseudo-class selector:
  html:

  ```
  <div class="hello">Hello</div>
  <div class="world">World</div>
  <div>Random</div>


  div:not(.world) {
      background-color: red;
  }
  ```

  Only `div` s containing the text `"Hello"` and `"Random"` will have a `background-color` of `red` .

6. Explain and apply the `::before` and `::after` pseudo elements, & Use the content CSS property to define the content of an element:
   - See exerpt from W9D1 demo:
     html:

     ```
     <p> this is a paragraph tag</p>
     ```

     css:

     ```
     p::after {
         background-color: lightblue;
         border-right: 1px solid violet;
         content: ':-D';
         margin-right: 4px;
         margin-left: 4px;
     }
     ```

7. What are these ?!
   - `font-size` : size of letters
   - `font-weight` : boldness of letters
   - `font-style` : italicization
   - `font-family` : actualy font
     - Some general font families: [sans-serif, serif, cursive]
   - `text-transform` : text casing
   - `text-decoration` : underlining
   - `text-align` : text justification (left, right, etc.)
8. Colors expressed as names, hexadecimal RGB values, and decimal RGB values
   - The `color` css property changes the color of the *text*.
   - The `background-color` css property does just what it says
   - `rgb()` - takes in 3 integer values denoting levels of `red, green and blue`

- `rgba()` - Same as above but with additional argument (the 'a') called **alpha** which represents how **transparent** the color will be and is on a scale from 0 to 1 where 0 is transparent and 1 is opaque.

9. Everything about borders, Shadows, & Opacity
    - takes in three arguments: thickness, line style, line color
    - Text and box shadows
    - Opacity set to 0 can make an element completely transparent.

```
h2 {
    border: 4px solid red;
    text-shadow: 0px 2px 10px orange;
    box-shadow: 5px 5px yellow;
    opacity: 0.5;
}
```

10. Covering an element with a background image

```
#picture-here {
    background-image: url(https://appacademy.github.io/styleguide/assets/logo/logo-embl
    background-size: cover;
    height: 100px;
    width: 100px;
}
```

11. `Explain why using Web fonts helps with consistent experience across viewing devices:
    - `Your explanation here.`
    - Web fonts are good to use because every browser has different default font families, but there is a drawback. Google tracks when you use their fonts using google fonts
    - Will be the same in any browser.
12. Explain absolute and relative length units in CSS
    - Relative :
        - `rem` - relative to root
        - `em` - relative to parent
    - Absolute Measure :
        - `pt`
        - `px`
        - `cm`

Wednesday

1. `display` property:
    - `inline`

    - `inline-block`
    - `block`
    - `flex`
    - `grid`
2. Identify the different types of media that a media query can target
    - all
    - print
    - screen
    - speech
3. Know the Box Model

Box Model

1. Describe the following
    - padding
    - margin
    - `position: fixed` -
    - `position: relative` - positioned relative to closest parent ancestor
    - `position: absolute` - removes from regular flow of the page.
    - `position: static` - the default positioning of all elements.
    - The MC quiz from earlier in the week is good for positioning stuff.
2. Identify elements rendered with specific padding and margin settings
3. Apply padding and margins to HTML elements to achieve a desired layout
4. Apply positioning settings to elements (fixed, relative, and absolute) to HTML elements to achieve a desired layout
5. Identify which HTML elements have a default "inline" display value
    - `fill in`
6. Identify which HTML elements have a default "block" display value
    - `fill in`
7. Describe and use z-index positioning of elements
    - `z-index` : refers to the "third dimension" i.e. stacking elements on top of each other. a higher z-index means bringing the element to the top.
8. Explain how flexible box layout lays out elements
    - displays items in flexible container so layout is responsive
    - containers height/width adjust to fit viewport
    - `justify-content`
        - alignment of items along main axis
        - distributes extra space around/between items
    - `align-items`

        - justify content for cross axis
9. For the following LOs please revisit your project work and project solutions (i.e. AA Times, Wednesday EOD demo, etc.) for how you've done the following:
    - Use the flex property to specify `grow`, `shrink`, and `basis` values.
    - `grow` determines how much available space it will take up
    - `shrink` determines how much the element can shrink
    - `basis` is considered to be the default size of the element
    - Use the `flex-direction` property to direct the layout of the content.
    - Use the `flex-wrap` property to affect the wrap of content layout within an element using flexible box layout.
    - Use `align-self`, `justify-content`, and `align-items` to change the way that children elements are laid out in a flexible box layout.
    - Use the `order` property to change the order in which elements will appear in a flexible box layout.
    - Use the `grid-template-columns`, `grid-template-rows`, and `grid-template` properties to specify the layout of the grid using relative and absolute measures.
    - Use `grid-template-areas` to label areas of a grid and `grid-area` to assign an element to the area.
    - Use `grid-column-gap`, `grid-row-gap`, and `grid-gap` to set the "gutter" areas between elements in a grid layout.
    - Use `grid-column-start` / `grid-column-end` and `grid-row-start` / `grid-row-end` to create spans across multiple columns and rows with positive integers, negative integers, and in conjunction with the "span" operator.
    - Use `justify-items`, `align-items`, `justify-content` and `align-content` to layout items in each grid area.
    - Use the `order` property to change the default order in which items are laid out
10. Explain how grid layout lays out elements
    - Sections off your document into smaller sections that can be organized and customised via the
      css grid layout properties.
11. Explain and use the shorthand versions of `grid-column` and `grid-row` to define how an element will span a grid layout
    - `grid-column: grid-column-start / grid-column-end`
    - `grid-row: grid-row-start / grid-row-end`
    - can use grid-area with grid-template-areas to visualize grid layout
        - grid-template-areas: each string represents one row
        - use grid-areas name to define region element will take up
        - `grid-area: grid-row-start / grid-column-start / grid-row-end / grid-column-end`
12. Explain and use the "fr" unit of measure

  - `fr` : fraction unit of measure used for creating grid layout

Thursday

1. Describe what `Block` , `Element` , and `Modifier` means in BEM
   - `block__element--modifier` `element` has `n` for `underscore` and `modifier` has `d` for `dash`
   - Block: standalone entity that is meaningful on its own
     - header, container, menu, navbar, input
     - name can contain latin letters, digits, dashes
   - Element - part of a block, no meaning on its own
     - menu item, list-item, header title, input label
     - name can contain latin letters, digits, dashes, underscore
     - class formed by block name + two underscores + element name
   - Modifier - flag on a block or element used to change appearance or behavior
     - disabled, color yellow, size big, fixed
     - add modifier class to blocks/el they modify and keep original class
     - css class formed as block or el name plus two dashes
2. Identify CSS class names that follow the BEM principle.
   - Go to wednesday EOD code demo for more (W9D3)

   ```
   <header class="header header--color>
   ```

3. Describe and use the transition property show animated changes due to class and pseudo-class CSS rule application
   - `transition-property`
     - name/names of css props to which transitions should apply
     - transition effect starts when specified CSS prop changes
   - `transition-duration`
     - duration over which transition occurs
   - `transition-delay`
     - how long to wait between time prop is changed and transition begins
   - `transition`
     - shorthand: property, duration, delay
4. Describe and use the overflow, overflow-x, and overflow-y properties to effect clipping and scrolling on elements
   - `overflow: auto`
   - `overflow: scroll`
   - `overflow: hidden`

Tuesday - AJAX

1. Explain what an AJAX request is
2. Identifying the advantages of using an AJAX request.
    - We don't have to refresh the entire HTML page.
    - It's a smaller amount of data that needs to be transferred.
3. Identify what the acronym AJAX means and how it relates to modern Web programming
    - Asynchronous JavaScript and XML
    - Asynchronous: We don't lock up the page when we are waiting on a response. We are still able to interact and the response's data will be handled whenever it returns.
    - JavaScript: The engine behind AJAX. We use JavaScript to make the request to the server, then we also use it to process the response and make any updates to the DOM that are needed based on this new data.
    - XML: The original format of the data that was sent back on the response. Nowadays we will almost always be using JSON as the format.
4. Describe the different steps in an AJAX request/response cycle
    - An event listener is set up to wait for an specific action that will trigger a request to our server. Clicking on a button or submitting a form would be a popular example.
    - When the event is triggered, we use JavaScript to formulate an appropriate request to a server. In our project we used `fetch` in order to send a request to a specific route on our server, along with an options object to indicate the methods, headers, etc., that differ from the default values, a body with necessary data, etc..
    - The request is sent asynchronously to the server. The user is still able to interact with our application since the request is not blocking the call stack.
    - The server receives the request and does whatever it needs to do on its end to create/read/update/destroy data related to the request. After it performs the requested action, it creates a response and sends it back to the client. This is almost always going to be in a JSON format.
    - The client receives the response and is able to parse the data and do any updates that it needs to do to the DOM. In our project, we used a `.then` on our call to `fetch`, which allowed us to then convert the response's JSON into a usable POJO when the response came back. The data inside of this object is then accessible and used to manipulate the DOM.
5. Fully use the fetch API to make dynamic Web pages without refreshing the page
    - Look over the AJAX project from Friday. Be comfortable with creating many different request types, such as `GET`, `PATCH`, `POST`, and `DELETE`.
    - Be comfortable with using both the `.then` promise chains that we used in the project as well as how we could convert them into an async/await format:

```
// Using Promise chains for .then and .catch
document.querySelector('#downvote').addEventListener('click', () => {
        fetch('http://localhost:3000/kitten/downvote', { method: 'PATCH' })
                .then(handleResponse) // handleResponse defined below for reference
    .then(updateImageScore) // updateImageScore defined below for reference
    .catch(handleError); // handleError defined below for reference
});


// Using async/await
document.querySelector('#downvote').addEventListener('click', async () => { // Notice the async
// We create a standard try/catch block
  try {
    // We await each asynchronous function call
    const resJSON = await fetch('http://localhost:3000/kitten/downvote', { method: 'PATCH' });
    const resObj = await handleResponse(resJSON);
    // updateImageScore is synchronous, so we do not have to await its response
    updateImageScore(resObj);
  } catch (e) {
    handleError(e)
  }
});


// Functions used above, for reference
const handleResponse = (response) => {
        stopLoader();
        clearError();

        if (!response.ok) {
                throw response;
        }
        return response.json();
};

const handleError = (error) => {
        if (error.json) {
                error.json().then((errorJSON) => {
                        document.querySelector('.error').innerHTML = `Error occured: ${errorJSON
                });
        } else {
                console.error(error);
                alert('Something went wrong. Please try again!');
        }
};

const updateImageScore = (data) => {
        const { score } = data;
        document.querySelector('.score').innerHTML = score;
};
```