Module 9 Day 4 Notes

Dark/Light Mode Toggle

Starter Code Explanation

Importing Fonts

```
@import
url('https://fonts.googleapis.com/css2?family=Poppins:ital,wght@0,100;0,200;0,300;
0,400;0,500;0,600;0,700;0,800;0,900;1,100;1,200;1,300;1,400;1,500;1,600;1,700;1,80
0;1,900&display=swap');
```

- This line imports the “Poppins” font in various weights and styles from Google Fonts, a free library of web fonts.
- The URL specifies different weights (100, 200, … 900) and styles (italic and regular).

Google Fonts (MDN)
Google Fonts Guide (W3Schools)

Universal Styling

```
*, *::before, *::after {
  box-sizing: border-box;
  margin: 0;
  padding: 0;
}
```

- The `*` selector targets all HTML elements.
- `*::before` and `*::after` target all pseudo-elements.
- `box-sizing: border-box;` ensures padding and borders are included in the total width and height of an element.
- By setting `margin` and `padding` to `0`, it removes any default spacing, giving you a clean slate.

CSS Universal Selector (MDN)
CSS `box-sizing` Property (W3Schools)

Body Styling

```css
body {
  font-family: "Poppins", sans-serif;
  background-color: #f0f0f0;
  color: #333;
  transition: all 0.4s ease-in;
}
```

- Sets the default font for the webpage to “Poppins”, and if it’s not available, it falls back to any sans-serif font.
- Defines the default background color and text color.
- The transition ensures any color change in the body appears smoothly.

CSS `font-family` Property (MDN)
CSS `background-color` Property (W3Schools)

Header Styling

```css
h1 {
  text-align: center;
  margin-top: 1.25rem;
}
```

- Aligns the text in the `<h1>` element to the center.
- Adds a top margin for spacing.

[CSS `text-align` Property (MDN)]
[CSS `margin` Property (W3Schools)]

Container Styling

```
.container {
  display: grid;
  width: 100%;
  min-height: calc(100vh - 4.25rem);
  place-content: center;
  gap: 1rem;
  text-align: center;
}
```

- Sets the display type to grid.
- Ensures the container takes up the full width and a minimum height based on the viewport height.
- Centers content both vertically and horizontally.
- The `gap` provides spacing between grid items.
- Content inside the container is centered.

The Computation: `calc(100vh - 4.25rem)`

This calculation dynamically determines the `min-height` of an element. It's essentially saying:

"Set the minimum height of this element to be the full height of the viewport (`100vh`) minus `4.25` times the font size of the root element."

If the root font size is `16px` (a common default), `4.25rem` would translate to `68px`. So, if your viewport height is `900px`, the `min-height` would be `900px - 68px = 832px`.

CSS Grid Layout (MDN)
CSS `place-content` Property (W3Schools)

CSS `calc()` Function (MDN)

CSS Units `vh` and `rem` (W3Schools)

HTML Structure

- Used a `<div>` with the class `container` to wrap the content.
- Inside the container, an `<h2>` displays the current theme.
- The toggle switch is composed of a `<label>`, a `<span>`, and a checkbox `<input>`.

HTML `<label>` Element (MDN)
HTML `<input>` Element (MDN)

CSS Styling

- `.dark-mode`: Defines the appearance for the dark theme.
- `.toggle-switch`: Positions and sizes the toggle switch container.
- `.slider`: Styles the background of our custom toggle switch.
- `.slider:before`: Creates the actual switch that moves left and right.
- Adjacent Sibling Combinator (`+`): Used to style elements based on the state or condition of their adjacent siblings.

CSS `transition` Property (MDN)
CSS `:before` Pseudo-element (MDN)
CSS Adjacent Sibling Combinator (`+`) (W3Schools)

JavaScript Interactivity

- DOM Selection: Used methods like `querySelector` and `getElementById` to access elements from our HTML.
- `toggleMode` Function: This function toggles between the dark and light mode and updates the status message.
- Event Listener: Added a click listener to our switch, which calls the `toggleMode` function when clicked.

JavaScript `querySelector` Method (MDN)
JavaScript `getElementById` Method (MDN)
JavaScript Event Listeners (W3Schools)

Further Learning and Resources:

- MDN Learning Web Development
- W3Schools HTML Tutorial
- W3Schools CSS Tutorial
- W3Schools JavaScript Tutorial

Image Slider Project Notes:

1. Introduction:

- We built an image slider that fades images in and out.
- Enhanced our skills in DOM manipulation.

2. HTML Structure:

- The slider contains multiple `<img>` elements and two `<button>` elements for navigation.
- All images and buttons are wrapped inside a `<div>` with an id of `slider`.

3. CSS Styling:

- Slider Styling:
  - Relative position, setting a canvas for images and buttons.
- Image Styling:
  - Absolute position to stack images.

    - Opacity set to `0` to hide images initially.
    - `transition` property for a smooth fade effect.
- Active Image Styling:
    - Opacity set to `1` to display the image.
- Button Styling:
    - Absolute position within the slider.
    - Circular shape with a border.
    - Hover effect to indicate interactivity.

4. JavaScript Functionality:

- Element Selection:
    - Used DOM methods to select images and buttons.
- Navigation Logic:
    - `currentIndex` to track the displayed image.
    - `reset` function to clear any active image.
    - `initializeSlider` to set the initial state.
    - `slideLeft` and `slideRight` functions to navigate through images.

5. Concepts Explained:

- NodeList:
    - Collection of nodes, often resulting from methods like `document.querySelectorAll()`.
    - Not a true array, but array-like.
- Accessing NodeList items (`images[i]`):
    - `i` is an index, starting from `0`.
    - Allows individual element manipulation within the NodeList.
- Navigation Functions:
    - `slideLeft` decrements `currentIndex`.
    - `slideRight` increments `currentIndex`.
    - Boundary conditions ensure circular navigation.

Resources:

HTML:

1. Images:
   - HTML Images on MDN
   - HTML Images on W3Schools
2. Div Element:
   - The Div element on MDN
   - HTML div element on W3Schools

CSS:

1. Position Property:
   - Position on MDN
   - CSS Positioning on W3Schools
2. Opacity Property:
   - Opacity on MDN
   - CSS Opacity on W3Schools
3. Transition Property:
   - Transition on MDN
   - CSS Transitions on W3Schools

JavaScript:

1. DOM Manipulation:
   - Document Object Model (DOM) on MDN
   - HTML DOM Tutorial on W3Schools
2. NodeList:
   - NodeList on MDN
   - HTML Collection vs. NodeList on W3Schools
3. Event Listeners:
   - addEventListener on MDN
   - DOM Event Listeners on W3Schools

Additional Google Doc Notes:

1. Objects and "this"

2. Intro to DOM Manipulation
3. Web APIS, BOM & DOM
4. CSS Units Cheat Sheet

Project 9 Guide: How to Approach This Project

1. Understand the Requirements: Before diving in, ensure you've read and understood what's expected in the project. This includes the core functionality and the optional (but recommended) projects section.
2. Planning:
   - Sketch out a rough design of the changes you wish to implement on paper or use a digital tool like Figma.
   - Decide on the core functionality you want to implement: image slider, dark/light mode toggle, or something else.
3. Research:
   - If you're implementing a feature you're unfamiliar with, do some research. Sites like MDN Web Docs, W3Schools, or CSS-Tricks can be invaluable.
4. Coding:
   - Start with the core functionality. Ensure you've backed up your website's current version before making changes.
   - Test as you code. This way, if something breaks, you know it's related to the most recent changes you've made.
   - If you choose to implement the projects section, design the layout first (using Grid or Flexbox), then populate it with your projects.
5. Accessibility: If you're implementing the dark/light mode toggle, ensure the colors you choose are accessible. Tools like the WebAIM Color Contrast Checker can help.
6. Review and Test:
   - Once you've added the desired functionalities, review your site on different devices and browsers to ensure compatibility.
   - Ask peers or mentors to review your site. They might spot issues you've missed.
7. Submission: Once you're satisfied, submit your HTML, CSS, and JavaScript for grading.
8. Reflection: After submitting, take a moment to reflect on the changes you've made. Write a brief summary of what you added to your website and why.