Assignment 1: CSS Variables- Primary and Secondary Colors

Setup/Assignment Description:

Create a webpage with a container that showcases the use of CSS variables for defining primary and secondary colors. Utilize the :root selector to set global variables --primary-color and --secondary-color. Apply these colors to different elements in the container, demonstrating the flexibility of CSS variables.

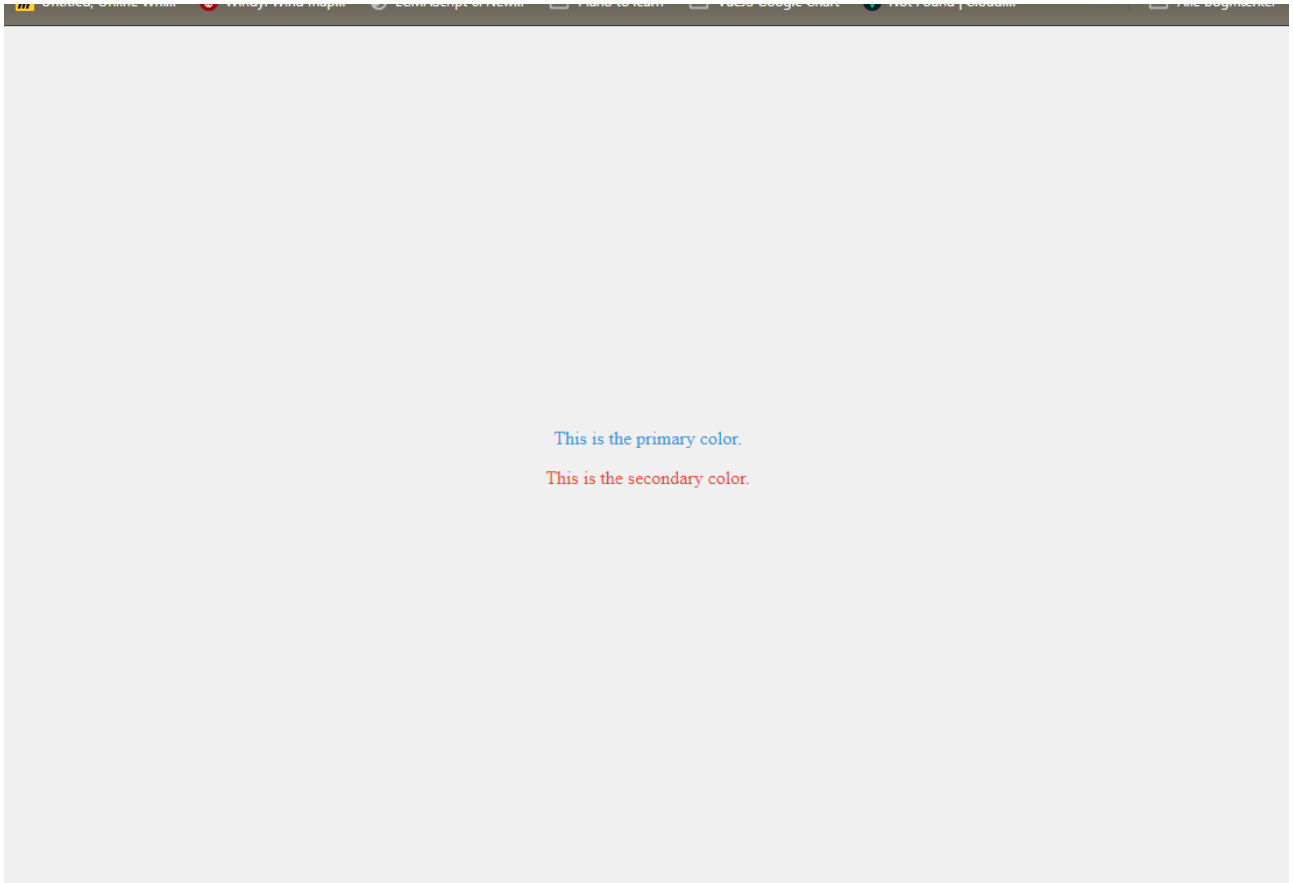
Instructions:

1. In the :root selector of your CSS, define the following CSS variables:

   --primary-color with a default value of #3498db (blue).

   --secondary-color with a default value of #e74c3c (red).

2. Create HTML elements within a container, such as paragraphs or headings.
3. Apply the primary color to one set of elements (e.g., using a class like .primary-color) and the secondary color to another set of elements (e.g., using a class like .secondary-color).
4. Observe how changing the values of the global CSS variables influences the colors of the elements throughout the container.



Assignment 2: Advanced CSS Variables- Dynamic Layout

Setup/Assignment Description:

Create a webpage with a dynamic layout using variables for spacing and sizing. create a multi-column layout with flexible spacing between columns and dynamic element sizes. Use CSS variables to control the layout parameters, allowing for easy adjustments.

Instructions:

1. Define the following CSS variables in the :root selector:

   --column-width for the width of each column (default: 200px).

   --column-gap for the spacing between columns (default: 20px).

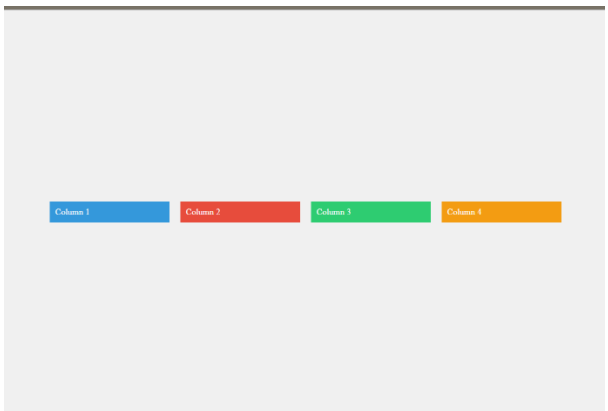
   --container-padding for the padding of the container (default: 20px).

   - **Main Version:**
     - Define separate CSS variables for the background color of each column (--color-1, --color-2, --color-3, --color-4).
     - Introduce a font color variable (--font-color) with a default value of #fff.
   - **Bonus/Advanced Version 1:**
     - Define a single CSS variable (--box-color) for the background color of all boxes in the main version.
   - **Bonus/Advanced Version 2:**
     - Instead of using color: #fff;, use the font color variable (--font-color) for text color in both the main and bonus/advanced versions.
2. Create a container element and populate it with multiple column elements.

3. Apply the CSS variables to control the width of each column, the spacing between columns, and the container's padding.

4. Observe how adjusting the values of the CSS variables influences the overall layout dynamically.

5. Main Version: Utilize the color and font variables within the :nth-child pseudo-class for each column to give them distinct background colors and text colors.
   - Bonus/Advanced Versions: Experiment with the additional color and font variables to enhance the visual appeal and maintainability of your multi-column layout.



Assignment 3: Dark Theme Switcher

Setup/Assignment Description:

Implement a Dark Theme Switcher for your webpage using CSS variables and JavaScript. Allow users to toggle between light and dark modes manually.
Additionally/bonus, leverage the @media (prefers-color-scheme) feature to automatically switch between light and dark modes based on the user's system preferences. What issues can this bring? Using (prefers-color-scheme)?

Instructions:

1. **CSS Variables**:
   - Define CSS variables for background colors and text colors for both light and dark modes in the :root selector.
   - Set up default styles for the body element, applying the light mode styles.

2. **Dark Mode Styles:**
   - Create styles for dark mode by defining a class, for example, .dark-mode. Apply this class to the body element when the user toggles to dark mode.

3. **Toggle Button:**
   - Add a button in the HTML (e.g., with the id themeToggle) that users can click to switch between light and dark modes manually.

4. **JavaScript**:
   - Write/Paste JavaScript code (in a separate script file, e.g., script.js) to toggle the dark-mode class on the body when the button is clicked.

   - Link to JS file to the index.html. just before the closing </body> tag, paste this in:

```
<script src="script.js"></script>
```

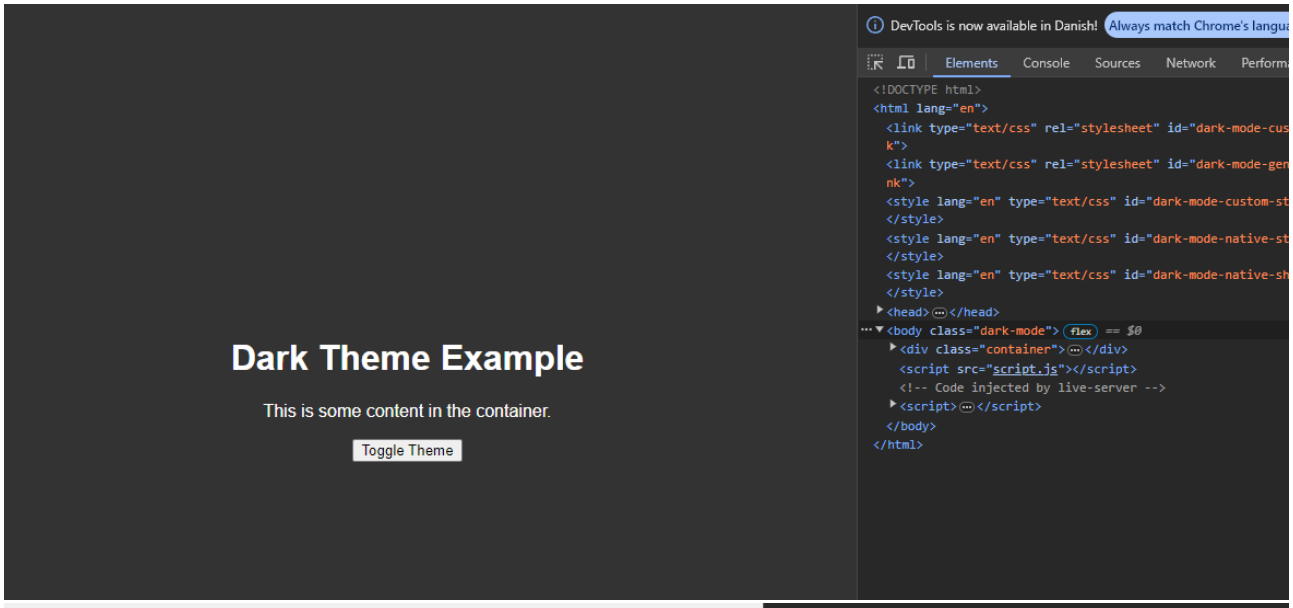
   - copy paste this js code, and put it into a file named “script.js”.

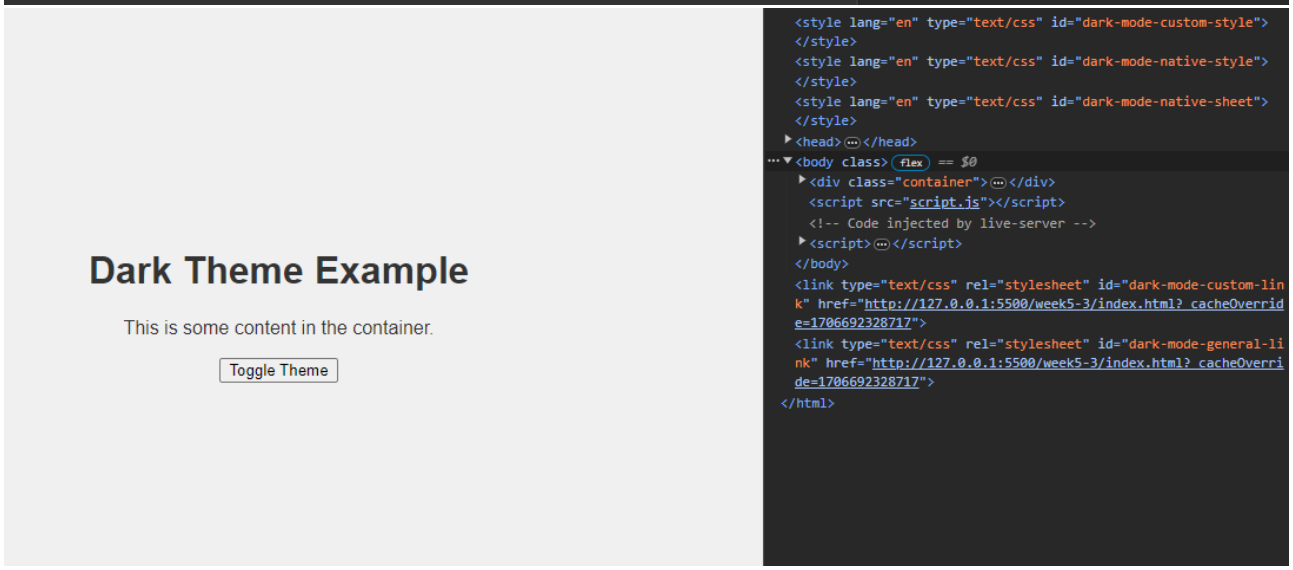
```
document.body.onclick = (e) => {
    const body = document.body;

    // Toggle dark mode class when the body is clicked
    body.classList.toggle('dark-mode');
};
```

5. (**optional**) Automatic Dark Mode:
   - Utilize @media (prefers-color-scheme: dark) in your CSS to automatically switch to dark mode based on the user's system preferences.

6. Testing:

- Test the Dark Theme Switcher by manually toggling between light and dark modes using the button.
- (optional) Observe how the webpage automatically switches between light and dark modes based on the user's system preferences.





Assignment 4: HTML/CSS Animation- Bounce

Setup/Assignment Description:

Create a simple animated element. This assignment focuses on keyframe animations to bring dynamic movement to a webpage.

Instructions:

1. HTML Setup:
   - Create an HTML document with a div element. This will be the element that you animate.

2. CSS Animation:
   - Style the div element in the CSS file, giving it a width, height, background color, and any other desired properties.
   - Implement a keyframe animation named bounce that creates a bouncing effect for the div. Utilize the @keyframes rule to define different steps of the animation, adjusting the transform: translateY() property to create the bounce.
   - Apply the bounce animation to the div.

Testing:

Open the HTML file in a browser to observe the animated effect.

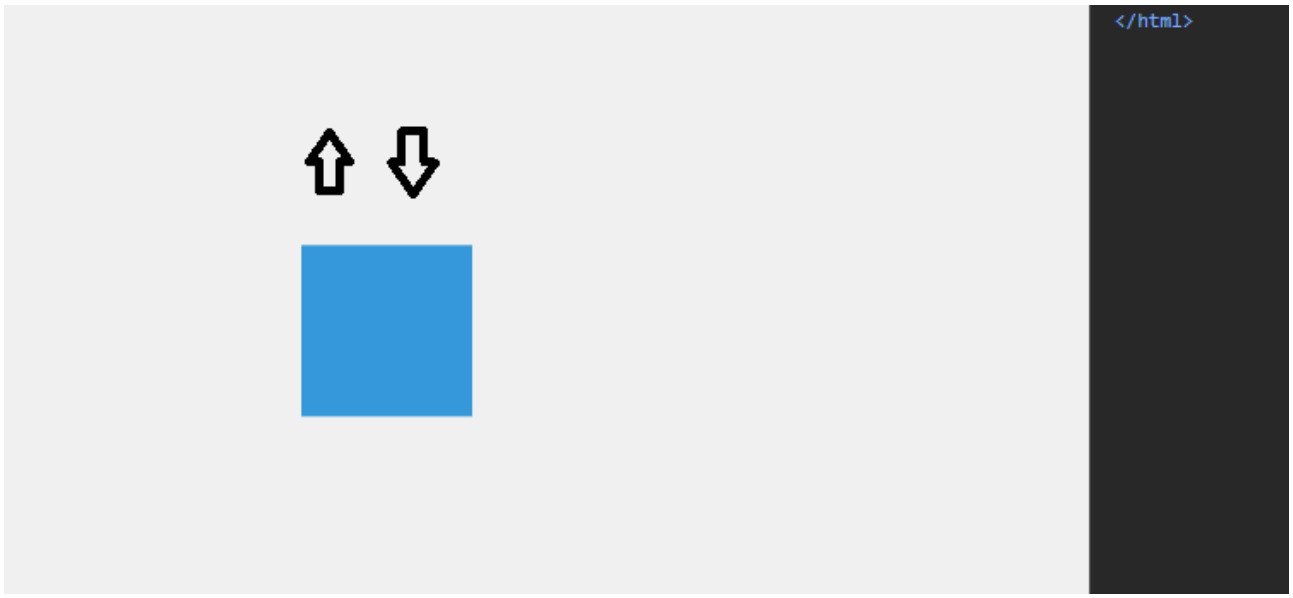
Experiment with the animation duration, easing, and other properties to fine-tune the visual impact.

Additional Challenges (Optional):

Explore other keyframe animation properties like scale, rotate, or color changes.

Create a sequence of animations to produce a more complex visual effect.

Experiment with the animation-delay property to stagger animations.



Assignment 5: CSS Animation- Unique Entry

Setup/Assignment Description:

Copy your assignment-4 files (or continue in them). Create a set of boxes with unique entry animations. Each box will fade in while entering the viewport from different directions.

Instructions:

1. HTML Setup:
   - create/reuse assignment4 HTML document and make/copy-paste four div elements, each having the class “box”.

2. CSS Animation - Fade In:
   - Style the box elements in the CSS file, setting their width, height, background color, and any other desired properties. (square, light blue, centered in browser)
   - Implement a keyframe animation named fadeIn that fades in each box.
   - Utilize the opacity property and the @keyframes rule to smoothly transition each box from an initial opacity of 0 to 1. (invisible on load, fades into full opacity over 1sec)
   - Apply the fadeIn animation to each box, ensuring they appear with a gradual fade-in effect.

Fade in to this

3. CSS Animation - Dual Entry: (drop in + fade in)
   - Implement/rename a keyframe animation named dualEntry that combines a fade-in effect with a drop-down animation. Attach the animation the the .box selector
   - Utilize the opacity property and the @keyframes rule to smoothly transition each box from an initial opacity of 0 to 1.
   - Simultaneously, apply a transform: translateY(-100px) property in the keyframes to create a drop-down effect, causing the boxes to enter from the top.
   - Adjust the duration of the animation to achieve a balanced and visually appealing effect.
   - Make it run only once!

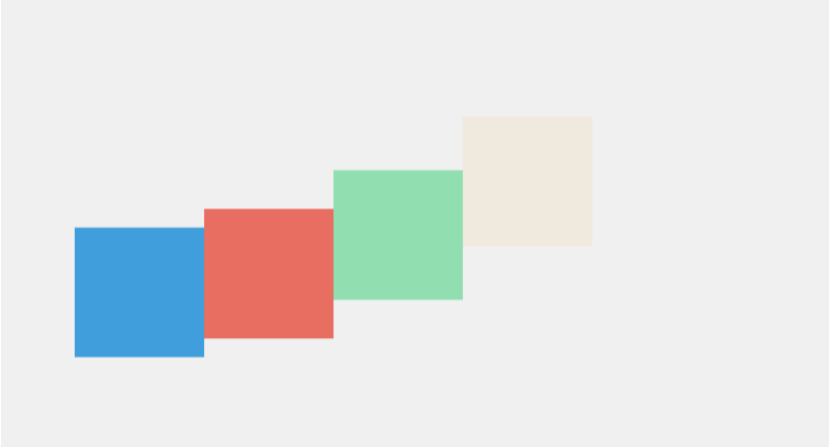
Fade in + drop down, as the image above

4. Staggered Animation:
    - Use the animation-delay property for each box to stagger their entry animations, creating a sequence of unique appearances. Attach the

```
.box:nth-child(1) {
    transform: translateY(-100px);
}
```

    - For each individual .box. nth-child(1) is first box, (2) is for second and so on.

Animation in progress (start from all invisible)



Animation end

5. Unique Entry Directions:
    - Assign a unique transform property to each box, specifying a starting position that will determine the direction from which it enters.
        - Hints: translateY(), translateX() and animation-delay

    - Experiment with different starting positions, such as top-left, top-right, bottom-left, and bottom-right, for a visually appealing effect.

Gif if it works: