Understanding CSS Variables (Custom Properties)

CSS Variables, also known as Custom Properties, are entities defined by CSS authors that contain specific values to be reused throughout a document. They help maintain consistency in styles and make it easier to manage and update values in your stylesheets.

How CSS Variables Work

Definition: CSS variables are defined using a specific syntax: --variable-name: value;. The variable name must start with two hyphens (--).

Scope: CSS variables can be defined globally (on the :root selector) or locally (within a specific selector).

Usage: To use a CSS variable, you reference it with the var() function: var(--variable-name).

Dynamic Changes: CSS variables can be changed dynamically with JavaScript or within CSS by adjusting the values in different states or media queries.

Example: Defining and Using CSS Variables

Here’s an example HTML document that demonstrates how to define and use CSS variables for consistent styling.

html

Copy code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>CSS Variables Example</title>

<style>

:root {

--primary-color: #4CAF50; /\* Define a primary color variable \*/

--secondary-color: #f0f0f0; /\* Define a secondary color variable \*/

--text-color: #ffffff; /\* Define a text color variable \*/

--font-size: 16px; /\* Define a base font size variable \*/

}

body {

background-color: var(--secondary-color); /\* Use secondary color \*/

font-size: var(--font-size); /\* Use base font size \*/

color: var(--primary-color); /\* Use primary color \*/

margin: 0;

padding: 20px;

font-family: Arial, sans-serif;

}

h1 {

color: var(--primary-color); /\* Heading color \*/

}

.card {

background-color: var(--primary-color); /\* Card background \*/

color: var(--text-color); /\* Card text color \*/

padding: 20px;

border-radius: 5px;

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);

margin-bottom: 20px;

}

/\* Change variable values for different states \*/

@media (max-width: 600px) {

:root {

--font-size: 14px; /\* Smaller font size for mobile \*/

}

}

/\* Change variable values for a hover effect \*/

.card:hover {

background-color: var(--text-color); /\* Change background on hover \*/

color: var(--primary-color); /\* Change text color on hover \*/

}

</style>

</head>

<body>

<h1>CSS Variables Example</h1>

<div class="card">

<h2>Card Title</h2>

<p>This is an example of a card using CSS variables for styling.</p>

</div>

<div class="card">

<h2>Another Card Title</h2>

<p>This card also uses the same CSS variables for consistent styling.</p>

</div>

</body>

</html>

Explanation of the Example

Defining Variables:

CSS variables are defined in the :root selector, making them globally accessible throughout the document. Here, we define four variables: --primary-color, --secondary-color, --text-color, and --font-size.

Using Variables:

These variables are used throughout the styles using the var() function:

background-color: var(--secondary-color); sets the background of the body to the secondary color.

color: var(--text-color); sets the text color of the card to white.

Dynamic Changes:

Media queries allow us to change the value of the --font-size variable for smaller screens, which results in a smaller font size when the screen width is less than 600px.

Hover Effects:

The hover effect on the .card class demonstrates how you can change the values of the CSS variables dynamically. When the card is hovered over, the background color changes to the text color, and the text color changes to the primary color.

Advantages of Using CSS Variables

Maintainability: Changing the value of a variable in one place automatically updates all instances where the variable is used.

Consistency: Ensures consistent use of colors, font sizes, and other values across a stylesheet.

Dynamic Updates: CSS variables can be manipulated in real-time using JavaScript, enabling interactive styling changes.

Conclusion

CSS variables provide a powerful way to manage styles in a more organized and efficient manner. They improve maintainability, consistency, and flexibility in web design, allowing developers to create responsive and dynamic layouts with ease. By using variables, you can quickly update styles without needing to modify multiple rules, enhancing the overall development experience.