CSS Variables

CSS Variables, also known as **CSS Custom Properties**, allow you to define reusable values in your stylesheets. These variables make it easier to maintain and update styles, especially when dealing with large projects or themes, as you only need to change the value in one place to update multiple elements across your styles.

Syntax

To declare a CSS variable, you use the `--` prefix followed by a variable name, and then assign it a value.

Defining a Variable

CSS variables are usually defined inside a `:root` selector or any other valid selector. The `:root` selector represents the highest-level parent in the document, similar to the `html` element, and is a good place to define global variables.

```
:root {
    --primary-color: #3498db;
    --secondary-color: #2ecc71;
    --padding-size: 10px;
}
```

Using a CSS Variable

You use the `var()` function to access the value of the variable anywhere in your CSS.

```
button {
    background-color: var(--primary-color);
    padding: var(--padding-size);
    color: white;
}

.card {
    border: 2px solid var(--secondary-color);
    padding: var(--padding-size);
}
```

Benefits of CSS Variables:

1. **Reusability**: You can define a variable once and use it multiple times, reducing redundancy.
2. **Easy Maintenance**: If you need to change a style (like a color or spacing), you only update the variable's value, and all references will be automatically updated.

3. **Theming**: CSS variables are ideal for creating different themes (like light and dark mode) by changing a few key variables.

CSS Variables vs Preprocessor Variables

CSS variables differ from variables in CSS preprocessors like SASS or LESS because:

- **Dynamic**: CSS variables can be updated and changed at runtime using JavaScript, while SASS/LESS variables are static and compiled at build time.
- **Scope**: CSS variables follow the normal rules of CSS, meaning they can be scoped to specific elements, while SASS/LESS variables are global or scoped to the file/module they are defined in.

Example of Dynamic Behavior

CSS variables can be updated dynamically in JavaScript:

```
<!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: #3498db;
        }

        body {
            background-color: var(--primary-color);
        }

        button {
            margin-top: 20px;
        }
    </style>
</head>
<body>
    <h1>CSS Variables</h1>
    <button onclick="changeColor()">Change Background Color</button>

    <script>
        function changeColor() {
            document.documentElement.style.setProperty('--primary-color', '#e74c3c');
        }
    </script>
```

```
</body>
</html>
```

In this example:

- Initially, the background is set to `#3498db` (blue) using the `--primary-color` variable.
- When the button is clicked, JavaScript dynamically changes the value of `--primary-color` to `#e74c3c` (red), which immediately updates the background color without reloading the page.

Scoping Variables

You can also define variables inside specific elements to override global variables:

```
:root {
    --primary-color: #3498db; /* Global */
}

.card {
    --primary-color: #e74c3c; /* Local to .card */
    background-color: var(--primary-color); /* Uses the local variable */
}
```

In this case, the `.card` element will have a red background (`#e74c3c`), while other elements using `--primary-color` will remain blue (`#3498db`).

Conclusion

CSS Variables give you flexibility in managing and reusing styles across a project. They help keep your CSS clean, maintainable, and easier to update by centralizing common style values.