CSS Variables

The var() function is used to insert the value of a CSS variable.

CSS variables have access to the DOM, which means that you can create variables with local or global scope, change the variables with JavaScript, and change the variables based on media queries.

A good way to use CSS variables is when it comes to the colors of your design. Instead of copy and paste the same colors over and over again, you can place them in variables.

Specifity

**CSS specificity** is a set of rules that determines which styles are applied to an element when there are conflicting CSS rules. Understanding specificity is crucial for managing styles effectively and ensuring that the desired rules take precedence.

**1. Understanding Specificity**

Specificity is calculated based on the types of selectors used in a CSS rule. The more specific a selector is, the higher its specificity value, which helps determine which styles are applied to an element.

**2. Specificity Hierarchy**

Specificity is calculated using four different categories, often represented in a numeric format (a, b, c, d):

* **Inline styles (a)**: Styles applied directly to an element using the style attribute. This has the highest specificity.
  + Example: <div style="color: red;">Text</div> (specificity: 1, 0, 0, 0)
* **IDs (b)**: Selectors that use the ID of an element (e.g., #example). IDs are more specific than classes and element selectors.
  + Example: #header { color: blue; } (specificity: 0, 1, 0, 0)
* **Classes, attributes, and pseudo-classes (c)**: Selectors that use classes (e.g., .className), attributes (e.g., [type="text"]), or pseudo-classes (e.g., :hover).
  + Example: .button { color: green; } (specificity: 0, 0, 1, 0)
* **Elements and pseudo-elements (d)**: Selectors that target HTML elements (e.g., div, p) or pseudo-elements (e.g., ::before, ::after).
  + Example: p { color: black; } (specificity: 0, 0, 0, 1)

**3. Calculating Specificity**

To calculate specificity, you can represent it as a four-part value (a, b, c, d):

* **Inline styles**: Count as 1 in the first position (a).
* **IDs**: Count as 1 in the second position (b).
* **Classes, attributes, and pseudo-classes**: Count as 1 in the third position (c).
* **Elements and pseudo-elements**: Count as 1 in the fourth position (d).

**Example Calculation**:

css

Copy code

#header .nav a:hover {

color: red; /\* Specificity: 0, 1, 1, 1 \*/

}

.header a {

color: blue; /\* Specificity: 0, 0, 1, 1 \*/

}

div {

color: green; /\* Specificity: 0, 0, 0, 1 \*/

}

In this example:

* #header .nav a:hover has a specificity of (0, 1, 1, 1) = 1 + 1 + 1 = 3.
* .header a has a specificity of (0, 0, 1, 1) = 1 + 1 = 2.
* div has a specificity of (0, 0, 0, 1) = 1.

**4. Specificity in Action**

When multiple CSS rules apply to the same element, the rule with the highest specificity wins. If two rules have the same specificity, the last one declared in the CSS will take precedence.

**5. Important Notes**

* **Inline styles** always take precedence over styles defined in CSS files, regardless of specificity.
* Specificity does not consider the order of styles; it only considers the selector types.
* Use IDs sparingly to avoid high specificity that can complicate CSS management.
* To manage styles effectively, prefer classes over IDs for styling elements and keep your CSS organized.

**6. Overriding Styles**

If you need to override styles, you can:

* Increase specificity by adding more classes or IDs.
* Use the !important declaration (though this is generally discouraged as it can lead to maintenance issues).

**Example**:

css

Copy code

.button {

color: blue !important; /\* This will override other rules \*/

}

Advanced Selectors

**Advanced CSS selectors** allow you to target specific elements or groups of elements in more complex and nuanced ways than simple class, ID, or element selectors. These selectors can help you create more efficient, maintainable, and dynamic stylesheets by reducing the need for extra classes or redundant rules. Here’s a breakdown of the most common advanced CSS selectors and their uses:

**1. Attribute Selectors**

Attribute selectors target elements based on the presence or value of their attributes.

* **[attribute]**: Selects elements with a specific attribute.

css

Copy code

[type] {

border: 1px solid red;

}

/\* Selects elements with a "type" attribute (e.g., <input type="text">, <button type="submit">) \*/

* **[attribute="value"]**: Selects elements with a specific attribute and exact value.

css

Copy code

[type="text"] {

background-color: yellow;

}

/\* Selects only <input type="text"> elements \*/

* **[attribute^="value"]**: Selects elements where the attribute value **starts** with a specific value.

css

Copy code

[href^="https"] {

color: green;

}

/\* Selects links that start with "https" \*/

* **[attribute$="value"]**: Selects elements where the attribute value **ends** with a specific value.

css

Copy code

[href$=".pdf"] {

text-decoration: underline;

}

/\* Selects links that end with ".pdf" \*/

* **[attribute\*="value"]**: Selects elements where the attribute value **contains** a specific value.

css

Copy code

[class\*="button"] {

padding: 10px;

}

/\* Selects all elements with "button" in their class name \*/

**2. Pseudo-classes**

Pseudo-classes are used to define a special state of an element. They typically target elements based on their state (hovered, focused, etc.) or their position within the document.

* **:hover**: Applies styles when the user hovers over an element.

css

Copy code

a:hover {

color: red;

}

* **:focus**: Targets an element that has focus (e.g., input fields when clicked).

css

Copy code

input:focus {

border-color: blue;

}

* **:nth-child(n)**: Selects the nth child of its parent, where n can be a number, keyword, or formula.

css

Copy code

li:nth-child(2) {

color: red;

}

/\* Selects the second child of all <li> elements \*/

* **:nth-of-type(n)**: Similar to nth-child, but only counts elements of a specific type.

css

Copy code

p:nth-of-type(3) {

font-weight: bold;

}

/\* Selects the third <p> element within its parent \*/

* **:nth-last-child(n)**: Same as nth-child but starts counting from the last child.

css

Copy code

li:nth-last-child(1) {

color: blue;

}

/\* Selects the last <li> child \*/

* **:first-child / :last-child**: Selects the first or last child element.

css

Copy code

p:first-child {

font-size: 20px;

}

/\* Applies to the first <p> child in its parent \*/

* **:not(selector)**: Excludes elements that match the given selector.

css

Copy code

p:not(.intro) {

color: black;

}

/\* Selects all <p> elements except those with the class "intro" \*/

* **:empty**: Selects elements that have no children (including text nodes).

css

Copy code

div:empty {

background-color: grey;

}

* **:checked**: Targets checked <input> elements (e.g., checkboxes, radio buttons).

css

Copy code

input:checked {

background-color: yellow;

}

**3. Pseudo-elements**

Pseudo-elements create and style specific parts of an element, such as the first letter, line, or before/after content.

* **::before / ::after**: Inserts content before or after the element's content.

css

Copy code

h1::before {

content: "★ ";

color: gold;

}

h1::after {

content: " ★";

color: gold;

}

/\* Adds stars before and after <h1> text \*/

* **::first-letter**: Styles the first letter of an element.

css

Copy code

p::first-letter {

font-size: 2em;

color: red;

}

/\* Enlarges and colors the first letter of each <p> \*/

* **::first-line**: Styles the first line of an element.

css

Copy code

p::first-line {

font-weight: bold;

}

**4. Combinators**

Combinators allow you to combine multiple selectors to target elements based on their relationship to other elements.

* **Descendant Selector ( )**: Selects elements that are descendants of a specified element.

css

Copy code

div p {

color: blue;

}

/\* Selects all <p> elements that are inside a <div> \*/

* **Child Selector (>)**: Selects elements that are direct children of a specified element.

css

Copy code

div > p {

color: green;

}

/\* Selects <p> elements that are direct children of a <div> \*/

* **Adjacent Sibling Selector (+)**: Selects an element that is immediately preceded by another specific element.

css

Copy code

h2 + p {

color: orange;

}

/\* Selects the first <p> element after an <h2> \*/

* **General Sibling Selector (~)**: Selects all elements that are siblings of a specified element.

css

Copy code

h2 ~ p {

color: purple;

}

/\* Selects all <p> elements that are siblings of an <h2> \*/

**5. Universal Selector (\*)**

The universal selector selects all elements. It is often used to apply global styles.

css

Copy code

\* {

margin: 0;

padding: 0;

}

**6. Group Selector (,)**

The group selector applies the same styles to multiple selectors, separated by commas.

css

Copy code

h1, h2, h3 {

font-family: Arial, sans-serif;

}

**7. Advanced :nth-child() Usage**

The nth-child pseudo-class allows more complex targeting based on formulas:

* **:nth-child(odd)**: Targets odd-numbered children.

css

Copy code

li:nth-child(odd) {

background-color: lightgrey;

}

* **:nth-child(3n+1)**: Targets every third element, starting from the first.

css

Copy code

li:nth-child(3n+1) {

background-color: yellow;

}

**8. Intersection of Selectors**

By combining multiple types of selectors, you can create very specific and powerful rules:

css

Copy code

div#content ul > li.special:first-child {

font-weight: bold;

}

This selector targets:

* <li> elements with class "special",
* That are the first child of their parent,
* And are direct children of an <ul>,
* Which is inside a <div> with the ID "content".

**Summary of Advanced Selectors:**

1. **Attribute Selectors**: Target elements based on attribute presence or value.
2. **Pseudo-classes**: Target elements based on their state or position (e.g., :hover, :nth-child).
3. **Pseudo-elements**: Style parts of an element (e.g., ::before, ::first-letter).
4. **Combinators**: Select elements based on relationships (descendants, siblings, etc.).
5. **Universal Selector (\*)**: Targets all elements.
6. **Group Selector (,)**: Apply the same styles to multiple elements.
7. **Advanced :nth-child()**: Allows complex targeting based on position.
8. **Intersection of Selectors**: Combines multiple selectors for precision.

Sibling Selectors

**CSS sibling selectors** allow you to target elements based on their relationship to other elements, specifically their position as siblings in the DOM (Document Object Model). There are two types of sibling selectors:

1. **Adjacent Sibling Selector (+)**
2. **General Sibling Selector (~)**

**1. Adjacent Sibling Selector (+)**

The adjacent sibling selector (+) selects an element that **immediately follows** another element. In other words, it selects the first sibling that comes directly after a specified element.

**Syntax:**

css

Copy code

A + B {

/\* styles \*/

}

Where B is the element that immediately follows A.

**Example:**

html

Copy code

<h2>Title 1</h2>

<p>This paragraph follows Title 1.</p>

<h2>Title 2</h2>

<p>This paragraph follows Title 2.</p>

css

Copy code

h2 + p {

color: blue;

}

* **Explanation**: The rule h2 + p targets the <p> elements that immediately follow an <h2>. In this case, both paragraphs will be blue because each one comes directly after an <h2>.

**2. General Sibling Selector (~)**

The general sibling selector (~) selects **all sibling elements** that follow the specified element, not just the immediate one. It applies to any element that is a sibling of the given element and appears later in the DOM, regardless of whether they are adjacent.

**Syntax:**

css

Copy code

A ~ B {

/\* styles \*/

}

Where B is any sibling that comes after A.

**Example:**

html

Copy code

<h2>Title 1</h2>

<p>This paragraph follows Title 1.</p>

<p>Another paragraph after Title 1.</p>

<h2>Title 2</h2>

<p>This paragraph follows Title 2.</p>

css

Copy code

h2 ~ p {

color: green;

}

* **Explanation**: The rule h2 ~ p selects all <p> elements that are siblings of any <h2> element. In this case, every <p> element that appears after an <h2> will be styled, not just the immediate one. So all four paragraphs in the example will turn green.

**Comparison: Adjacent vs. General Sibling Selector**

* **Adjacent Sibling Selector (+)**: Only applies to the **next** sibling immediately following the target element.

Example:

html

Copy code

<h2>Title</h2>

<p>Paragraph 1</p>

<p>Paragraph 2</p>

css

Copy code

h2 + p {

color: blue;

}

**Result**: Only the first paragraph will be blue, as it is the adjacent sibling of the <h2>.

* **General Sibling Selector (~)**: Applies to **all siblings** that follow the target element.

Example:

html

Copy code

<h2>Title</h2>

<p>Paragraph 1</p>

<p>Paragraph 2</p>

css

Copy code

h2 ~ p {

color: green;

}

**Result**: Both paragraphs will be green since they both follow the <h2> and are its siblings.

CSS Properties

CSS properties are used to define the styles applied to HTML elements. These properties control various aspects of the appearance and layout of elements, such as color, typography, spacing, borders, and positioning. Below is an overview of some common categories of CSS properties, along with examples for each.

**1. Typography**

These properties control the appearance of text, such as font, size, weight, and alignment.

* **font-family**: Specifies the font to be used for text.

css

Copy code

p {

font-family: Arial, sans-serif;

}

* **font-size**: Specifies the size of the text.

css

Copy code

h1 {

font-size: 2em;

}

* **font-weight**: Controls the thickness (weight) of the text.

css

Copy code

strong {

font-weight: bold;

}

* **line-height**: Sets the amount of space between lines of text.

css

Copy code

p {

line-height: 1.5;

}

* **text-align**: Aligns the text horizontally within its container.

css

Copy code

h1 {

text-align: center;

}

* **color**: Specifies the color of the text.

css

Copy code

p {

color: #333;

}

* **text-transform**: Controls capitalization (e.g., uppercase, lowercase, capitalize).

css

Copy code

h2 {

text-transform: uppercase;

}

**2. Box Model (Padding, Margin, Borders)**

The CSS box model governs the space around elements, including padding, borders, and margins.

* **margin**: Specifies the space outside an element's border.

css

Copy code

div {

margin: 20px;

}

* **padding**: Specifies the space inside an element, between the content and the border.

css

Copy code

div {

padding: 10px;

}

* **border**: Specifies the border around an element.

css

Copy code

div {

border: 1px solid black;

}

* **width** and **height**: Define the width and height of an element.

css

Copy code

img {

width: 100px;

height: auto;

}

**3. Background**

These properties allow you to control the background of an element, including colors and images.

* **background-color**: Specifies the background color of an element.

css

Copy code

body {

background-color: #f0f0f0;

}

* **background-image**: Sets an image as the background of an element.

css

Copy code

div {

background-image: url('background.jpg');

}

* **background-size**: Controls the size of the background image.

css

Copy code

div {

background-size: cover;

}

* **background-position**: Specifies the starting position of a background image.

css

Copy code

div {

background-position: center;

}

* **background-repeat**: Controls whether the background image is repeated.

css

Copy code

div {

background-repeat: no-repeat;

}

**4. Positioning**

These properties define how elements are positioned on the page.

* **position**: Specifies the type of positioning method used for an element (static, relative, absolute, fixed, sticky).

css

Copy code

div {

position: absolute;

top: 50px;

left: 100px;

}

* **top**, **right**, **bottom**, **left**: Define offsets for positioned elements.

css

Copy code

div {

position: relative;

top: 20px;

}

* **z-index**: Controls the stacking order of positioned elements (higher values are in front).

css

Copy code

div {

z-index: 10;

}

* **float**: Floats an element to the left or right, allowing text or other elements to wrap around it.

css

Copy code

img {

float: left;

}

* **clear**: Prevents elements from floating next to floated elements.

css

Copy code

div {

clear: both;

}

**5. Flexbox**

Flexbox is a layout model that allows items within a container to align and distribute space.

* **display: flex**: Enables flexbox on a container.

css

Copy code

div {

display: flex;

}

* **justify-content**: Aligns flex items along the main axis.

css

Copy code

div {

justify-content: center;

}

* **align-items**: Aligns flex items along the cross axis.

css

Copy code

div {

align-items: stretch;

}

* **flex-direction**: Defines the direction of flex items (row, column).

css

Copy code

div {

flex-direction: column;

}

* **flex-wrap**: Controls whether flex items wrap onto multiple lines.

css

Copy code

div {

flex-wrap: wrap;

}

**6. Grid**

CSS Grid is a layout system for creating complex grid-based layouts.

* **display: grid**: Enables grid layout on a container.

css

Copy code

div {

display: grid;

}

* **grid-template-columns**: Defines the columns in a grid.

css

Copy code

div {

grid-template-columns: 100px 200px auto;

}

* **grid-template-rows**: Defines the rows in a grid.

css

Copy code

div {

grid-template-rows: 100px 100px;

}

* **grid-gap**: Adds spacing between grid items.

css

Copy code

div {

grid-gap: 10px;

}

**7. Transformations and Transitions**

These properties allow you to animate and transform elements on the page.

* **transform**: Applies a 2D or 3D transformation (e.g., rotate, scale).

css

Copy code

div {

transform: rotate(45deg);

}

* **transition**: Defines how changes in properties occur over time, creating smooth animations.

css

Copy code

div {

transition: background-color 0.5s ease;

}

**8. Visibility and Display**

These properties control how elements are rendered or hidden.

* **display**: Controls the display behavior of an element (block, inline, none, etc.).

css

Copy code

div {

display: none;

}

* **visibility**: Controls the visibility of an element (visible, hidden).

css

Copy code

div {

visibility: hidden;

}

**9. Overflow**

These properties handle content that overflows an element’s box.

* **overflow**: Controls how overflow content is displayed (visible, hidden, scroll, auto).

css

Copy code

div {

overflow: auto;

}

**10. List and Table Styling**

These properties allow you to style lists and tables.

* **list-style-type**: Changes the marker type of list items (e.g., disc, circle, square, none).

css

Copy code

ul {

list-style-type: square;

}

* **list-style-position**: Specifies whether the marker is inside or outside the list item.

css

Copy code

ul {

list-style-position: inside;

}

* **border-collapse**: Controls whether table borders are separated or collapsed.

css

Copy code

table {

border-collapse: collapse;

}

* **caption-side**: Specifies the position of the table caption.

css

Copy code

table {

caption-side: top;

}

**11. Color and Opacity**

These properties control the color and transparency of elements.

* **background-color**: Sets the background color of an element.

css

Copy code

div {

background-color: #f00;

}

* **opacity**: Controls the transparency level of an element.

css

Copy code

div {

opacity: 0.5;

}

**12. Shadows**

CSS allows you to add shadows to elements.

* **box-shadow**: Adds a shadow to an element’s box.

css

Copy code

div {

box-shadow: 5px 5px 10px #888;

}

* **text-shadow**: Adds a shadow to text.

css

Copy code

h1 {

text-shadow: 2px 2px 5px black;

}