Good one 🙂 — the word **“variable”** shows up in two different ways in the web world:

**1️⃣ HTML <var> Element (semantic variable)**

* HTML has a built-in tag called <var>.
* It’s used to **mark text that represents a variable**, usually in math, programming, or documentation.
* Example:
* <p>The formula for area is <var>a</var> × <var>b</var>.</p>

👉 In browsers, <var> usually appears in *italic* by default.

* It doesn’t *store* data like a programming variable — it’s just semantic markup (meaningful text).

**2️⃣ CSS Custom Properties (often called “CSS variables”)**

* In styling, we use **custom properties** inside :root or other selectors:
* :root {
* --main-color: #13315C;
* --font-size-md: 16px;
* }
* p {
* color: var(--main-color);
* font-size: var(--font-size-md);
* }
  + --main-color and --font-size-md are **variables**.
  + var(--main-color) is how you use them.
  + They make your design reusable and easy to update (change once, update everywhere).

**3️⃣ JavaScript Variables (dynamic data)**

* In JavaScript, variables hold values that can change while your program runs.
* Example:
* <script>
* let name = "Arvind";
* document.write("Hello " + name);
* </script>
  + name is a variable storing the string "Arvind".
  + You can reuse and change it later (name = "Meena";).

**🎯 Quick Recap**

* **<var> in HTML** → semantic tag for marking variable names in text.
* **CSS variables** → reusable style values (--my-color).
* **JavaScript variables** → data containers that actually store/change values.

👉 So the meaning of “variable in HTML” depends on context:

* If you’re asking about the **tag**, it’s <var>.
* If you’re asking about **styling**, it’s CSS variables.
* If you’re asking about **dynamic content**, that’s handled with JavaScript variables.

🎯 — CSS variables (official name: **CSS custom properties**) are one of the most powerful modern features in web design. Let’s go deeper into how they work:

**🔹 1. Defining a CSS Variable**

* Written as --name: value;
* Usually defined in :root so they’re **global** (available everywhere).

:root {

--main-bg: #f0f0f0;

--main-color: #13315C;

--spacing: 1rem;

}

**🔹 2. Using a CSS Variable**

* Call it with var(--name) wherever you need it.

body {

background: var(--main-bg);

color: var(--main-color);

padding: var(--spacing);

}

👉 Now, if you ever change --main-color, it updates **everywhere** you used it.

**🔹 3. Fallback Values**

You can give a backup value if the variable isn’t set:

p {

color: var(--text-color, black);

}

👉 If --text-color doesn’t exist, the text will be black.

**🔹 4. Scope (Local vs Global)**

* **Global**: defined in :root → available everywhere.
* **Local**: defined inside a selector → only applies inside that element.

:root {

--color: blue;

}

.card {

--color: red; /\* local variable just for .card \*/

color: var(--color);

}

👉 Normal text will be blue, but text inside .card will be red.

**🔹 5. Dynamic Updates (with JavaScript)**

You can even change CSS variables with JS at runtime.

:root {

--main-color: blue;

}

document.documentElement.style.setProperty('--main-color', 'green');

👉 Suddenly, everything using --main-color turns green — instant theming!

**🔹 6. Real-World Uses**

* **Theming** (light/dark mode):

:root {

--bg: white;

--text: black;

}

.dark-mode {

--bg: black;

--text: white;

}

body {

background: var(--bg);

color: var(--text);

}

👉 Toggle .dark-mode on <html> → the whole site changes theme.

* **Consistency**: define one spacing unit (--gap: 16px;) and reuse it everywhere.
* **Maintainability**: easier to update design tokens (colors, fonts, shadows).

**⚡ Why CSS Variables Are Better Than Old “Sass/LESS Variables”**

* Old preprocessors (like Sass) replaced variables **at compile time** (static).
* CSS variables work **in the browser**, so they can:
  + Be updated live with JS.
  + Be scoped (different values in different parts of the page).
  + React to media queries (e.g. smaller spacing on mobile).

👉 In short: CSS variables are like **design Lego blocks 🧩** — define once, reuse everywhere, and re-theme instantly.

Perfect ⚡—here’s a **tiny self-contained HTML demo** that shows how CSS variables can be used to switch between light and dark mode with just a button.

**🌗 Light/Dark Mode Demo**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>CSS Variables Demo</title>

<style>

/\* Define default (light mode) variables \*/

:root {

--bg: white;

--text: black;

--accent: #13315C;

}

/\* Dark mode overrides \*/

.dark {

--bg: #0b1f33;

--text: #f0f0f0;

--accent: #FF6A13;

}

body {

background: var(--bg);

color: var(--text);

font-family: sans-serif;

padding: 2rem;

transition: all 0.3s ease;

}

button {

background: var(--accent);

color: var(--bg);

padding: 0.5rem 1rem;

border: none;

border-radius: 6px;

cursor: pointer;

font-size: 1rem;

}

</style>

</head>

<body>

<h1>Hello, world!</h1>

<p>This text and background color are controlled by CSS variables.</p>

<button onclick="document.body.classList.toggle('dark')">

Toggle Dark Mode

</button>

</body>

</html>

**🔍 How It Works**

1. :root defines variables for light mode (--bg, --text, --accent).
2. .dark overrides those same variables with darker colors.
3. The body and button styles **use the variables**.
4. When you click the button, JavaScript toggles the .dark class on <body>.
5. Instantly, the entire theme switches without rewriting any CSS. 🎉

👉 This is the magic of CSS variables: you **change a few tokens**, and your whole design adapts.