ScopeTreasure Courses

CSS COURSES · LESSON 2.14 · CODE SCOPE SERIES

CSS Variables + Dark Mode Project

Define colours once in :root, use them everywhere with var() — then flip the whole page to dark mode by redefining them. Video coming soon.

Introduction

Every real design repeats itself: the brand colour appears in the navbar, the buttons, the links, the footer — forty places. Change it the hard way and you edit forty lines (and miss three). **CSS variables** (officially *custom properties*) end that: you name a value once — `--brand: #0b6e4f;` — and spend the *name* everywhere with `var(--brand)`. This lesson builds the full pattern, and ends with the project it was born for: an entire page that flips to **dark mode** by redefining a handful of variables — not one style rewritten. *(Roman Urdu: Variable aik naam hai jise aap aik dafa value dete hain — phir poori site wohi naam istemal karti hai.)*

Key points

- **Define on `:root`:** `:root` means the very top element (`<html>`) — variables defined there are visible to **every** element on the page. Names begin with two dashes: `--brand`, `--ink`, `--radius`. A variable can hold **any** value: a colour, a size, a shadow, a font stack.
- **Spend with `var()`:** `color: var(--brand);` — at render time the browser swaps in the value. Change the definition and *every* usage updates in the same instant — that is the entire magic, and it works live in DevTools.
- **Fallbacks for safety:** `var(--accent, gold)` — "use `--accent`; if it was never defined, use gold". A fallback saves you when a variable name is mistyped, because…
- **…a bad variable fails silently:** an undefined variable (no fallback) makes the property *invalid* — the element quietly falls back to inherited/default styling. No error, no warning. When a colour "randomly" refuses to apply, check the variable's spelling first.

  (Roman Urdu: Variable ghalat likha to style chup chaap gayab — spelling hamesha check karo.)
- **Scope is inheritance:** a variable defined on `.card` exists only inside that card (and its children) — a local override. `:root` = global, a class = a local edition of the same name.
- **Dark mode the variables way:** your styles never name raw colours — they name *roles*: `--bg`, `--surface`, `--ink`, `--muted`. A dark theme simply **redefines the roles**: `@media (prefers-color-scheme: dark)` for the visitor's system choice, or a `[data-theme="dark"]` block for a manual switch. Zero rules rewritten.

Code examples

Example 1 — One palette, spent everywhere

```
:root {
  --brand: #0b6e4f;
  --brand-soft: #e7f6ef;
  --ink: #14231d;
  --radius: 12px;
}

h1 { color: var(--brand); }
.btn { background: var(--brand); border-radius: var(--radius); color: white; }
.tag { background: var(--brand-soft); color: var(--brand); border-radius: var(--radius); }
```

What it does: Four definitions at the top; every rule below spends names, not values. The tag and the button share a radius because they share `--radius` — the design is *consistent by construction*, not by your memory.

What you will see: a normally styled page — the variables are invisible in the result. Now the lesson's party trick: in DevTools, edit `--brand` on `:root` to `#7c3aed` (purple). The heading, button and tag all turn purple *together, live*. That is what "redefine once" means.

Example 2 — Automatic dark mode (prefers-color-scheme)

```
:root {
  --bg: #faf9f6;
  --surface: #ffffff;
  --ink: #14231d;
  --muted: #5a6a63;
}
@media (prefers-color-scheme: dark) {
  :root {
    --bg: #0d1512;
    --surface: #16211c;
    --ink: #eef5f1;
    --muted: #9db3a8;
  }
}
body { background: var(--bg); color: var(--ink); }
.card { background: var(--surface); color: var(--ink); }
.card p { color: var(--muted); }
```

What it does: The component rules are written **once**, against role variables. The media query — the same tool as Lesson 2.9, aimed at the visitor's *system theme* instead of width — redefines the four roles for dark-

mode users. Not one component rule is duplicated.

What you will see: flip your operating system (or the DevTools rendering emulation) between light and dark: the whole page inverts — paper becomes deep green-black, cards lift to a lighter surface, muted text stays muted. Visitors who run dark mode get a dark site *for free*.

Example 3 — The project: a manual theme switch, pure CSS

```
<label class="switch">
  <input type="checkbox" id="theme-toggle">
  <span>🌙 Dark mode</span>
</label>
```

```
:root {
  --bg: #faf9f6; --surface: #ffffff; --ink: #14231d; --brand: #0b6e4f;
}
/* when the checkbox (anywhere above the page content) is ticked… */
:root:has(#theme-toggle:checked) {
  --bg: #0d1512; --surface: #16211c; --ink: #eef5f1; --brand: #2fbf8f;
}
body { background: var(--bg); color: var(--ink); transition: background-color 0.3s ease, color 0.3s ease; }
```

What it does: The modern `:has()` selector lets the **root** react to a checkbox state inside the page — tick the box and `:root` redefines its palette; untick and the light palette returns. Because every component spends variables, the *entire page* re-themes from four redefinitions, and the 2.10 transition makes the flip glide instead of snap. (Production sites usually flip a `data-theme` attribute with two lines of JavaScript and save the choice — the variable half, the half that matters, is identical to what you see here.)

What you will see: a profile page with a small "🌙 Dark mode" switch — tick it and the page melts into its dark edition; untick and the light edition returns. You have built the feature that billion-dollar apps advertise, with zero rewritten styles.

Quick recap

- **Define** on `:root` (`--name: value`), **spend** with `var(--name)`, insure with a **fallback** `var(--name, default)`.
- Variables are **live**: one redefinition re-themes everything that spends the name.
- **Dark mode = redefining role variables** (`--bg`, `--surface`, `--ink`) under `prefers-color-scheme` or a theme switch — never rewriting component styles.

Common mistakes

1. **The silent typo** — `var(--brnad)` references a variable that does not exist; the declaration quietly dies and the element shows inherited styling. No console error will save you — spelling is the debugging.
2. **Defining locally, spending globally** — a variable declared inside `.sidebar` does not exist in the footer. If a `var()` "works in one place only", check *where* the definition lives; page-wide values belong on `:root`.
3. **Hard-coding colours beside the variables** — one raw `#0b6e4f` typed directly in a footer rule will not follow the theme flip, and will glow wrongly in dark mode. If it is a theme colour, it goes through a variable — no exceptions.

Practice

1. Take the ChaiPoint page from Lesson 2.12 and convert every raw colour to a role variable (`--bg`, `--surface`, `--ink`, `--muted`, `--brand`). The page must look identical — then produce a second colour scheme by editing `:root` only.
2. Add the Example 2 automatic dark scheme to that page, then the Example 3 manual switch on top. Verify: light system + unticked = light; dark system = dark; the switch flips either way.

Practice quiz

1. Where are page-wide CSS variables defined?

- Inside any single element that uses them
- On :root, the very top element, so every element can see them
- Only inside @media rules
- In the HTML head as attributes

Answer: On :root, the very top element, so every element can see them

2. What does var(--accent, gold) mean?

- Use --accent and also paint gold behind it
- Use --accent; if it was never defined, use gold
- Define a variable called gold
- It is an error, fallbacks do not exist

Answer: Use --accent; if it was never defined, use gold

3. How is dark mode done the variables way?

- By rewriting every component rule with dark colours
- By redefining the role variables (--bg, --surface, --ink) under prefers-color-scheme or a theme switch
- By adding a dark class to every element by hand
- Dark mode cannot be done with variables

Answer: By redefining the role variables (--bg, --surface, --ink) under prefers-color-scheme or a theme switch

---

Watch the Code Scope channel: https://www.youtube.com/@CodeScopePK — this lesson's own video joins the channel when it is recorded.

Read online: https://scopetreasure.com/notes/css-variables-dark-mode · More free tools and notes at https://scopetreasure.com