**Organising files for easy maintenance and scalability.**

**Folder structure**

/portfolio-root

│── /css

│ ├── styles.css # Global styles (colors, fonts, resets, common layout)

│ ├── header.css # Styles for the header (shared across pages)

│ ├── footer.css # Styles for the footer (shared across pages)

│ ├── home.css # Styles specific to the homepage

│ ├── about.css # Styles specific to the about page

│ ├── projects.css # Styles for the projects page

│ ├── case-studies.css # Styles for project details & case studies

│ ├── animations.css # Reusable animations & transitions

│ ├── utilities.css # Helper classes (spacing, flex/grid utilities)

│── /images

│── /js

│── /pages

│── index.html

│── about.html

│── projects.html

│── case-study1.html

**CSS File structure**

Organizing your CSS files properly will make your portfolio easier to maintain and scale. Here’s a well-structured CSS architecture that balances global styles with page-specific styles:

* styles.css (Global) → Define universal styles like fonts, colors, buttons, and responsiveness.
* header.css **&** footer.css → Keep your nav/footer consistent across pages.
* home.css, about.css, projects.css, case-studies.css → Each page has its unique styles.
* animations.css → Store animations (e.g., fade-in effects, hover transitions).
* utilities.css → Helper classes (e.g., .container, .hidden, .flex-center).

**Naming conventions**

Using **clear, scalable, and maintainable** CSS naming conventions is crucial for organizing styles efficiently. Here are some best practices:

**1. Use Meaningful and Descriptive Names**

* Avoid generic names like div1, button2—instead, use hero-section, cta-button, card-container.
* Names should reflect the **role** or **purpose** of the element.

**2. Follow a Consistent Naming Convention**

Here are some widely used conventions:

**🔹 BEM (Block Element Modifier) – Recommended**

**Example:**

css

.card { /\* Block \*/ }

.card\_\_title { /\* Element \*/ }

.card--highlighted { /\* Modifier \*/ }

**Structure:**

* Block → A standalone component (card, nav, form).
* Element → A part of a block (card\_\_title, nav\_\_link).
* Modifier → A variation of a block or element (card--highlighted).

**🔹 CamelCase or kebab-case**

* mainContainer (CamelCase)
* main-container (kebab-case, **preferred** for CSS)

**3. Keep It Simple, Avoid Deep Nesting**

* Excessive nesting makes styles **hard to maintain**.
* Instead of this:

css

.hero > .container > .row > .column > .title { ... }

* Use **direct classes**:

css

.hero-title { ... }

**4. Prefixing for Specific Contexts**

If your site has multiple pages/components, prefix relevant styles:

* .btn-primary, .btn-secondary
* .navbar-dark, .navbar-light
* .profile-card, .product-card

**5. Avoid Overly Specific Selectors**

* Instead of div > ul > li > a, use .nav-link.

**6. Use a Separate Class for States**

Instead of overriding styles inline:

css

<button class="btn active"></button>

Define state styles separately:

css

.btn--active { background-color: green; }

**7. Keep Class Names Short, But Clear**

Avoid:

css

.super-long-and-confusing-class-name { ... }

Better:

css

.hero-title { ... }

By following these practices, your CSS stays clean, scalable, and easy to manage!