1 Why Project Structure Matters

If we just dump all HTML, CSS, and images in one folder, the project:

- Becomes **messy**
- Is **hard to maintain**
- Makes teamwork harder
- Increases risk of overwriting files

Good folder structure:

- Keeps things **organized**
- Makes **updates faster**
- Helps in **team collaboration**
- Is a **professional habit**

2 Professional HTML/CSS Project Folder Structure

- Here's the most common **root-level layout**:

```
project-name/
|
├── index.html              # Main entry file
├── about.html              # Other HTML pages
├── contact.html
|
├── css/                    # All stylesheets
|   ├── style.css           # Main/global CSS
|   ├── reset.css           # (Optional) CSS reset/normalize
|   └── about.css           # Page-specific styles
|
├── js/                     # All scripts (if needed later)
|   └── script.js
|
├── images/                 # All images
|   ├── logo.png
|   ├── banner.jpg
|   └── icons/
|
├── fonts/                  # Custom fonts (optional)
|
└── assets/                 # Other files (videos, PDFs, downloads)
```

4️⃣ Global vs Page-Specific Styles

- **Global CSS** → Styles shared across **all pages**
  Example: css/style.css
- **Page-specific CSS** → Styles for a **single page** only
  Example: css/about.css

Tip: Always load **global first**, then page-specific.

```
<link rel="stylesheet" href="css/style.css">
<link rel="stylesheet" href="css/about.css">
```

5️⃣ Best Practices

1. **Never keep everything in root** — only main HTML files stay there
2. **Name files/folders in lowercase** with - instead of spaces
   - ✅ about-us.html
   - ❌ About Us.html
3. **Use reset.css or normalize.css** to keep styles consistent across browsers
4. **Use descriptive file names** (avoid img1.jpg, use hero-banner.jpg)
5. **Keep assets separate** → Don't mix CSS, JS, and images in one folder
6. **Avoid absolute paths** → Use relative paths so project works anywhere
   - ✅ images/logo.png
   - ❌ C:/Users/Sarim/Desktop/logo.png

REUSABILITY

The trick: use **CSS variables** in :root and apply global resets and reusable classes.

1️⃣ Start with a CSS Reset (Consistency Across Browsers)

```
/* Global Reset */
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
```

}

2 Define Global Theme in :root

```css
:root {
  /* Colors */
  --primary-color: #007bff;
  --secondary-color: #6c757d;
  --background-color: #f8f9fa;
  --text-color: #333;

  /* Fonts */
  --font-family: 'Roboto', sans-serif;

  /* Font Sizes */
  --heading-size: 2rem;
  --paragraph-size: 1rem;
  --small-text: 0.85rem;

  /* Spacing */
  --padding-default: 1rem;
  --margin-default: 1rem;
}
```

3 Apply Global Styles

```css
body {
  font-family: var(--font-family);
  font-size: var(--paragraph-size);
  background-color: var(--background-color);
  color: var(--text-color);
  line-height: 1.6;
}

h1, h2, h3, h4 {
  font-size: var(--heading-size);
  font-weight: bold;
}

p {
  margin-bottom: var(--margin-default);
}
```

4 Create Reusable Utility Classes

```css
/* Buttons */
.btn {
  padding: 0.5rem 1rem;
  border: none;
  background-color: var(--primary-color);
  color: white;
  cursor: pointer;
  border-radius: 5px;
  font-size: var(--paragraph-size);
}

.btn:hover {
  background-color: darkblue;
}
```

```css
/* Text colors */
.text-primary { color: var(--primary-color); }
.text-secondary { color: var(--secondary-color); }

/* Background colors */
.bg-primary { background-color: var(--primary-color); color: white; }
.bg-secondary { background-color: var(--secondary-color); color: white; }

/* Spacing helpers */
.mt-1 { margin-top: 1rem; }
.mb-1 { margin-bottom: 1rem; }
.p-1 { padding: 1rem; }
```