**What is SASS/SCSS?**

**SASS = Syntactically Awesome Style Sheets**

* It’s a **CSS preprocessor** → a tool that extends CSS with features like **variables, nesting, mixins, inheritance, and functions**.
* You write styles in .sass or .scss files → then **compile** into normal .css (which browsers understand).

**SASS vs. SCSS**

SASS actually has **two syntaxes**:

1. **SASS (Indented Syntax)**

Uses **indentation** instead of curly braces {} and semicolons ;.

Example:

$primary: #3498db

body

background: $primary

font-size: 16px

1. **SCSS (Sassy CSS)**

Newer and more popular syntax.

Fully **compatible with CSS** (every valid CSS file is also valid SCSS).

Example:

$primary: #3498db;

body {

background: $primary;

font-size: 16px;

}

**Why use SASS/SCSS instead of plain CSS?**

**Variables** → store and reuse colors, fonts, spacing  
**Nesting** → cleaner, structured styles  
**Partials & Imports** → split large CSS into small files  
**Mixins** → reusable blocks of CSS (like functions)  
**Math & Logic** → do calculations (width: 100% / 3;)

In short:  
**SASS/SCSS = a “smarter” way to write CSS** → makes your stylesheets **more powerful, reusable, and maintainable**.

**Here’s exactly how you can compile SCSS into CSS so browsers can use it:**

## 1. Install Sass (the compiler)

You need the **Sass tool** installed. Options:

### a) With ****Node.js**** (most common):

npm install -g sass

### b) Or download Dart Sass standalone:

https://sass-lang.com/install

## 2. Basic Compile Command

From your project folder, run:

sass styles.scss styles.css

This takes styles.scss and outputs a normal styles.css.

## 3. Watch Mode (auto-recompile on save)

Instead of running manually every time, let Sass **watch** your file:

sass --watch styles.scss:styles.css

Now, whenever you edit & save styles.scss, it automatically updates styles.css.

## 4. Project Structure Example

project/

index.html

scss/

main.scss

css/

main.css ← compiled output

Run:

sass scss/main.scss css/main.css --watch

## 5. Link in HTML

In your index.html, include the **compiled CSS**:

<link rel="stylesheet" href="css/main.css">

You **cannot** link .scss files directly in HTML, only the compiled .css.

# **Introduction to SCSS/SASS**

## 1. Variables

* Variables let you **store values** (colors, fonts, sizes) and reuse them throughout your stylesheet.
* Helps maintain consistency and makes updates easy (change in one place → applied everywhere).

**SCSS Example:**

$primary-color: #3498db;

$font-stack: 'Segoe UI', sans-serif;

body {

font-family: $font-stack;

background-color: $primary-color;

}

button {

background: $primary-color;

color: white;

}

**Compiled CSS:**

body {

font-family: 'Segoe UI', sans-serif;

background-color: #3498db;

}

button {

background: #3498db;

color: white;

}

## 2. Nesting

* Instead of writing long selectors repeatedly, you can **nest child selectors inside parent selectors**.
* Makes code more readable and mirrors the HTML structure.

**SCSS Example:**

nav {

background: #333;

ul {

list-style: none;

li {

display: inline-block;

a {

color: white;

text-decoration: none;

&:hover {

color: yellow;

}

}

}

}

}

**Compiled CSS:**

nav {

background: #333;

}

nav ul {

list-style: none;

}

nav ul li {

display: inline-block;

}

nav ul li a {

color: white;

text-decoration: none;

}

nav ul li a:hover {

color: yellow;

}

## 3. Mixins

* A **mixin** is like a **function for CSS** — you define reusable chunks of code and include them wherever needed.
* Great for vendor prefixes or repeating patterns.

**SCSS Example:**

@mixin flex-center {

display: flex;

justify-content: center;

align-items: center;

}

@mixin box-shadow($color) {

box-shadow: 0 4px 6px $color;

}

.container {

@include flex-center;

height: 100vh;

@include box-shadow(rgba(0, 0, 0, 0.2));

}

**Compiled CSS:**

.container {

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

box-shadow: 0 4px 6px rgba(0, 0, 0, 0.2);

}

**Quick Recap**

* **Variables** → store values once, reuse everywhere
* **Nesting** → cleaner, structured selectors
* **Mixins** → reusable blocks (like CSS functions)

**Examples**

## 1) Variables & Nesting — A pill button

**HTML**

<button class="btn">Click me</button>

<link rel="stylesheet" href="styles.css">

**SCSS**

$primary: #2563eb;

.btn {

background: $primary; color: #fff; border: 0; padding: .6rem 1rem;

border-radius: 999px; cursor: pointer;

&:hover { background: darken($primary, 7%); }

}

## 2) Mixin for Flex Center — Center a box

**HTML**

<div class="box">Centered</div>

<link rel="stylesheet" href="styles.css">

**SCSS**

@mixin flex-center { display:flex; align-items:center; justify-content:center; }

html, body { height: 100%; margin:0; }

.box { @include flex-center; height: 100vh; background:#f3f4f6; font:700 24px/1 system-ui; }

## 3) Accordion (CSS only with checkbox)

**HTML**

<div class="accordion">

<input type="checkbox" id="acc1">

<label class="accordion\_\_toggle" for="acc1">Section 1</label>

<div class="accordion\_\_panel">Content 1</div>

</div>

<link rel="stylesheet" href="styles.css">

**SCSS**

.accordion {

border:1px solid #e5e7eb; border-radius:8px; overflow:hidden;

input { display:none; }

&\_\_toggle { display:block; padding:1rem; cursor:pointer; background:#fff; }

&\_\_panel { display:none; padding:1rem; background:#f9fafb; }

input:checked ~ &\_\_panel { display:block; }

}

## 4) Tabs (CSS only with radio buttons)

**HTML**

<div class="tabs">

<input type="radio" name="tab" id="tabA" checked>

<input type="radio" name="tab" id="tabB">

<div class="tabs\_\_nav">

<label for="tabA">A</label>

<label for="tabB">B</label>

</div>

<div class="tabs\_\_panel" id="panelA">Panel A</div>

<div class="tabs\_\_panel" id="panelB">Panel B</div>

</div>

<link rel="stylesheet" href="styles.css">

**SCSS**

.tabs {

&\_\_nav { display:flex; gap:.5rem; margin-bottom:.75rem;

label { padding:.4rem .8rem; border-radius:999px; border:1px solid #e5e7eb; cursor:pointer; }

}

&\_\_panel { display:none; border:1px solid #e5e7eb; padding:1rem; border-radius:8px; }

#tabA:checked ~ #panelA, #tabB:checked ~ #panelB { display:block; }

#tabA:checked ~ .tabs\_\_nav label[for=tabA],

#tabB:checked ~ .tabs\_\_nav label[for=tabB] { background:#111827; color:#fff; }

}

## 5) Responsive Grid

**HTML**

<div class="grid">

<div class="card">1</div>

<div class="card">2</div>

<div class="card">3</div>

</div>

<link rel="stylesheet" href="styles.css">

**SCSS**

$bp: (md:768px, lg:1024px);

@mixin up($k){ @media(min-width:map-get($bp,$k)){ @content; } }

.grid {

display:grid; gap:1rem; grid-template-columns:1fr;

@include up(md){ grid-template-columns:repeat(2,1fr); }

@include up(lg){ grid-template-columns:repeat(3,1fr); }

}

.card { background:#fff; border:1px solid #e5e7eb; padding:1rem; border-radius:8px; }

## 6) Button Variants via Map Loop

**HTML**

<button class="btn btn--primary">Primary</button>

<button class="btn btn--success">Success</button>

<button class="btn btn--danger">Danger</button>

<link rel="stylesheet" href="styles.css">

**SCSS**

$variants: ( primary:#2563eb, success:#16a34a, danger:#dc2626 );

.btn { color:#fff; border:0; padding:.6rem 1rem; border-radius:8px; cursor:pointer; }

@each $n,$c in $variants {

.btn--#{$n}{ background:$c; &:hover{ background:darken($c,6%); } }

}

## 7) Card with Hover

**HTML**

<article class="card">

<div class="card\_\_media"></div>

<div class="card\_\_body">

<h3 class="card\_\_title">Title</h3>

<p class="card\_\_text">Description</p>

</div>

</article>

<link rel="stylesheet" href="styles.css">

**SCSS**

.card {

border:1px solid #e5e7eb; border-radius:12px; overflow:hidden; background:#fff;

transition: box-shadow .2s ease, transform .2s ease;

&:hover{ box-shadow:0 10px 20px rgba(0,0,0,.08); transform:translateY(-2px); }

&\_\_media { width:100%; aspect-ratio:16/9; background:#e5e7eb; }

&\_\_body { padding:1rem; }

&\_\_title { margin:0 0 .25rem; font:700 1.1rem/1.2 system-ui; }

&\_\_text { margin:0; color:#6b7280; }

}

## 8) Toast (CSS-only with :target)

**HTML**

<a href="#toast" class="btn">Show Toast</a>

<div id="toast" class="toast">Saved! <a href="#">✖</a></div>

<link rel="stylesheet" href="styles.css">

**SCSS**

.toast {

position:fixed; bottom:1rem; right:1rem;

background:#111827; color:#fff; padding:.6rem .9rem; border-radius:10px;

opacity:0; pointer-events:none; transition:.2s;

a { color:#fff; margin-left:.5rem; }

}

:target.toast { opacity:1; pointer-events:auto; }

## 9) Progress Bar (CSS-only with animation)

**HTML**

<div class="progress">

<div class="progress\_\_bar"></div>

</div>

<link rel="stylesheet" href="styles.css">

**SCSS**

@keyframes fill { from{width:0} to{width:80%} }

.progress { background:#e5e7eb; height:10px; border-radius:999px; overflow:hidden; }

.progress\_\_bar { width:0; height:100%; background:#2563eb; animation:fill 2s forwards; }

## 10) Dark Mode Toggle (CSS-only with checkbox)

**HTML**

<label><input type="checkbox" id="dark"> Dark Mode</label>

<div class="panel">

<h2>Theme Demo</h2>

<p>Switch themes with SCSS + checkbox.</p>

</div>

<link rel="stylesheet" href="styles.css">

**SCSS**

$bg:#fff; $fg:#111827; $muted:#6b7280;

$bg-dark:#0b1020; $fg-dark:#e5e7eb; $muted-dark:#9ca3af;

body { margin:0; font-family:system-ui; background:$bg; color:$fg; }

.panel { border:1px solid #ddd; border-radius:12px; padding:1rem; color:$muted; }

#dark:checked ~ .panel { background:$bg-dark; color:$fg-dark; border-color:$muted-dark; }

#dark:checked ~ .panel p { color:$muted-dark; }

## 11) Theming with Token Maps → auto CSS variables

**HTML**

<div class="theme theme--light">

<div class="card">Light theme card</div>

</div>

<div class="theme theme--dark mt"></div>

<link rel="stylesheet" href="styles.css">

**SCSS**

$themes: (

light: (bg:#ffffff, fg:#111827, border:#e5e7eb),

dark: (bg:#0b1020, fg:#e5e7eb, border:#15213a)

);

@each $name, $tokens in $themes {

.theme--#{$name} {

@each $k, $v in $tokens { --#{$k}: #{$v}; }

background: var(--bg); color: var(--fg);

.card { border:1px solid var(--border); padding:1rem; border-radius:12px; }

}

}

.mt { margin-top: 1rem; }

## 12) Responsive “Burger” Nav (checkbox hack)

**HTML**

<nav class="site-nav">

<input id="nav-toggle" type="checkbox">

<label for="nav-toggle" class="site-nav\_\_burger">☰</label>

<ul class="site-nav\_\_links">

<li><a href="#">Docs</a></li><li><a href="#">Blog</a></li><li><a href="#">About</a></li>

</ul>

</nav>

<link rel="stylesheet" href="styles.css">

**SCSS**

$bp: 768px;

.site-nav {

background:#111827; color:#fff; padding:.5rem 1rem; position:relative;

&\_\_burger { display:inline-block; cursor:pointer; padding:.4rem .6rem; }

&\_\_links { list-style:none; margin:0; padding:0; display:none;

li { border-top:1px solid rgba(255,255,255,.1); }

a { display:block; color:#fff; padding:.6rem; text-decoration:none; }

}

#nav-toggle { display:none; }

#nav-toggle:checked ~ .site-nav\_\_links { display:block; }

@media (min-width:$bp){

&\_\_burger { display:none; }

&\_\_links { display:flex; gap:1rem; border:0; li{border:0;} a{padding:.4rem 0;} }

}

}

## 13) Tooltip via data-attr + mixin

**HTML**

<button class="tip" data-tip="Save changes">💾 Save</button>

<link rel="stylesheet" href="styles.css">

**SCSS**

@mixin tooltip($bg:#111827, $fg:#fff) {

position:relative;

&::after {

content: attr(data-tip);

position:absolute; inset:auto 0 120% 50%;

transform: translateX(-50%);

background:$bg; color:$fg; padding:.25rem .5rem; border-radius:6px;

font-size:.8rem; white-space:nowrap; opacity:0; pointer-events:none; transition:.15s;

}

&:hover::after { opacity:1; inset:auto 0 140% 50%; }

}

.tip { @include tooltip(); border:1px solid #e5e7eb; border-radius:8px; padding:.4rem .6rem; background:#fff; }

## 14) CSS-only Dropdown with <details>

**HTML**

<details class="dropdown">

<summary>Choose an option</summary>

<ul>

<li><a href="#">One</a></li>

<li><a href="#">Two</a></li>

<li><a href="#">Three</a></li>

</ul>

</details>

<link rel="stylesheet" href="styles.css">

**SCSS**

.dropdown {

width: 200px; font-family: system-ui;

summary { list-style:none; cursor:pointer; border:1px solid #e5e7eb; padding:.5rem .75rem; border-radius:8px; }

ul { margin:.4rem 0 0; padding:.4rem 0; border:1px solid #e5e7eb; border-radius:8px; display:none; background:#fff; }

li { list-style:none; }

a { display:block; padding:.5rem .75rem; text-decoration:none; color:#111827; &:hover{background:#f3f4f6;} }

&[open] ul { display:block; }

}

## 15) Masonry-ish Grid with grid-auto-rows & spans

**HTML**

<div class="masonry">

<div class="item item--h2">A</div>

<div class="item">B</div>

<div class="item item--h3">C</div>

<div class="item">D</div>

<div class="item item--h2">E</div>

</div>

<link rel="stylesheet" href="styles.css">

**SCSS**

.masonry {

display:grid; grid-template-columns: repeat(4, 1fr);

grid-auto-rows: 80px; gap: .75rem;

}

.item { background:#e5e7eb; border-radius:10px; display:flex; align-items:center; justify-content:center; font:600 1.25rem/1 system-ui; }

.item--h2 { grid-row: span 2; }

.item--h3 { grid-row: span 3; }

@media (max-width:900px){ .masonry { grid-template-columns: repeat(2,1fr); } }

## 16) Form States + Inline Validation Look (no JS)

**HTML**

<form class="form">

<label>Email <input type="email" required placeholder="you@example.com"></label>

<small class="hint">We’ll never share your email.</small>

<label>Password <input type="password" required minlength="6"></label>

<button type="submit" class="btn">Create account</button>

</form>

<link rel="stylesheet" href="styles.css">

**SCSS**

$form-border:#e5e7eb; $valid:#16a34a; $invalid:#dc2626;

.form { max-width: 360px; font-family: system-ui; display:grid; gap:.75rem; }

label { display:grid; gap:.25rem; color:#111827; }

input {

padding:.5rem .6rem; border:1px solid $form-border; border-radius:8px; outline: none;

&:focus{ border-color:#2563eb; box-shadow:0 0 0 3px rgba(37,99,235,.15); }

&:user-valid{ border-color:$valid; }

&:user-invalid{ border-color:$invalid; }

}

.hint { color:#6b7280; }

.btn { padding:.55rem 1rem; border:0; border-radius:8px; background:#111827; color:#fff; cursor:pointer; }

## 17) Table with Sticky Header + Zebra + Compact Utilities

**HTML**

<table class="table table--compact">

<thead><tr><th>Name</th><th>Role</th><th>Status</th></tr></thead>

<tbody>

<tr><td>Ada</td><td>Engineer</td><td>Active</td></tr>

<tr><td>Grace</td><td>Scientist</td><td>On Leave</td></tr>

<tr><td>Linus</td><td>Architect</td><td>Active</td></tr>

</tbody>

</table>

<link rel="stylesheet" href="styles.css">

**SCSS**

.table {

border-collapse: separate; border-spacing:0; width:100%; overflow:hidden; border:1px solid #e5e7eb; border-radius:10px; font-family: system-ui;

thead th { position:sticky; top:0; background:#f8fafc; text-align:left; font-weight:700; }

th, td { padding:.75rem 1rem; border-bottom:1px solid #e5e7eb; }

tbody tr:nth-child(odd) { background:#fcfdff; }

&--compact { th, td { padding:.5rem .75rem; } }

}

## 18) CSS-only Slider (radio) + SCSS loops

**HTML**

<div class="slider">

<input type="radio" name="s" id="s1" checked>

<input type="radio" name="s" id="s2">

<input type="radio" name="s" id="s3">

<div class="slides">

<figure class="slide">Slide 1</figure>

<figure class="slide">Slide 2</figure>

<figure class="slide">Slide 3</figure>

</div>

<div class="dots">

<label for="s1"></label>

<label for="s2"></label>

<label for="s3"></label>

</div>

</div>

<link rel="stylesheet" href="styles.css">

**SCSS**

$w: 320px; $h: 160px;

.slider { width:$w; margin:auto; font-family:system-ui; }

input { display:none; }

.slides { width:$w; height:$h; overflow:hidden; position:relative; border:1px solid #e5e7eb; border-radius:10px; }

.slide { position:absolute; inset:0; display:flex; align-items:center; justify-content:center; background:#e5e7eb; opacity:0; transition:.25s; }

#s1:checked ~ .slides .slide:nth-child(1),

#s2:checked ~ .slides .slide:nth-child(2),

#s3:checked ~ .slides .slide:nth-child(3) { opacity:1; }

.dots { display:flex; justify-content:center; gap:.5rem; margin-top:.5rem; }

.dots label { width:10px; height:10px; border-radius:999px; background:#cbd5e1; cursor:pointer; }

#s1:checked ~ .dots label:nth-child(1),

#s2:checked ~ .dots label:nth-child(2),

#s3:checked ~ .dots label:nth-child(3) { background:#111827; }

## 19) Skeleton Loader (keyframes + mixin)

**HTML**

<div class="skeleton-card">

<div class="sk sk--media"></div>

<div class="sk sk--line"></div>

<div class="sk sk--line sk--short"></div>

</div>

<link rel="stylesheet" href="styles.css">

**SCSS**

@keyframes shimmer {

from { background-position: 200% 0; }

to { background-position: -200% 0; }

}

@mixin skeleton {

background: linear-gradient(90deg, #f1f5f9 25%, #e2e8f0 37%, #f1f5f9 63%);

background-size: 400% 100%;

animation: shimmer 1.2s infinite linear;

}

.skeleton-card { width:320px; border:1px solid #e5e7eb; border-radius:12px; padding:1rem; }

.sk { @include skeleton; border-radius:8px; height:12px; margin:.4rem 0; &--short{ width:60%; } &--media{ height:160px; } }

## 20) Aspect-Ratio Helper + Auto Media Classes

**HTML**

<div class="media ar-16x9"></div>

<div class="media ar-1x1 mt"></div>

<link rel="stylesheet" href="styles.css">

**SCSS**

$ratios: ( '16x9': 56.25%, '4x3': 75%, '1x1': 100% );

.media { width: 320px; background:#e5e7eb; border-radius:12px; position:relative; overflow:hidden; }

.media::before { content:""; display:block; }

@each $name, $pct in $ratios {

.ar-#{$name}::before { padding-top: $pct; }

}

.mt { margin-top: 1rem; }