Complete CSS and Tailwind CSS Learning Guide

From Fundamentals to Advanced Techniques

Table of Contents

Introduction

This guide is designed to help you relearn CSS and Tailwind CSS, taking you from the fundamentals to advanced techniques. Whether you're refreshing your knowledge or filling in gaps, you'll find detailed explanations, practical examples, and side-by-side comparisons between vanilla CSS and Tailwind approaches.

How to use this guide:

- Work through each section sequentially if you're starting from scratch
- Jump to specific topics if you need a refresher on particular concepts
- Try the practice projects to solidify your understanding
- Use the reference materials for quick lookups during development

Let's begin your journey back into the world of modern CSS and Tailwind!

CSS Section

CSS Fundamentals and Setup

What is CSS?

CSS (Cascading Style Sheets) is a stylesheet language used to describe the presentation of a document written in HTML. It defines how elements should be rendered on screen, on paper, or in other media.

How CSS Works

CSS operates on a simple principle: it selects HTML elements and applies styles to them. This process involves:

1. **Selectors**: Target specific HTML elements
2. **Properties**: Specify what aspects to style (color, size, etc.)
3. **Values**: Define how to style those aspects (red, 16px, etc.)

The basic syntax looks like this:

```
selector {
  property: value;
  another-property: another-value;
}
```

CSS is applied to HTML documents in three ways:

1. **Inline CSS**: Applied directly to HTML elements using the `style` attribute

```
<p style="color: blue; font-size: 16px;">This is a paragraph.</p>
```

2. **Internal CSS**: Defined within a `<style>` element in the document's `<head>`

```
<head>
  <style>
    p {
      color: blue;
      font-size: 16px;
    }
  </style>
</head>
```

3. **External CSS**: Defined in a separate .css file and linked to the HTML

```
<head>
  <link rel="stylesheet" href="styles.css" />
</head>
```

With styles.css containing:

```
p {
  color: blue;
  font-size: 16px;
}
```

Setting Up Your CSS Environment

To start working with CSS, you'll need:

1. **Text Editor or IDE**: Popular options include:
   - Visual Studio Code (recommended for its extensions)
   - Sublime Text
   - Atom
   - WebStorm
2. **Essential VS Code Extensions**:
   - CSS Peek: Jump to CSS definitions from HTML
   - IntelliSense for CSS: Provides advanced autocompletion
   - Live Server: Preview changes in real-time

- Prettier: Format your CSS code automatically
- CSS Modules: If you're working with CSS Modules
- Live Sass Compiler: If you're using Sass

3. **Browser Developer Tools**:
   - Chrome DevTools
   - Firefox Developer Tools
   - Safari Web Inspector
   - Edge DevTools

4. **Setup Steps**:
   1. Install your preferred code editor
   2. Install relevant extensions
   3. Create a project folder structure:

```
project/
├── index.html
├── css/
│   └── styles.css
├── js/
└── images/
```

   4. Create a basic HTML file linking to your CSS:

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta
      name="viewport"
      content="width=device-width, initial-scale=1.0"
    />
    <title>CSS Learning Project</title>
    <link rel="stylesheet" href="css/styles.css" />
  </head>
  <body>
    <h1>Hello, CSS!</h1>
    <p>This is a paragraph to style.</p>
  </body>
</html>
```

   5. Add some basic CSS to styles.css:

```
body {
  font-family: Arial, sans-serif;
  max-width: 800px;
  margin: 0 auto;
  padding: 20px;
}

h1 {
  color: #333;
}

p {
  line-height: 1.5;
}
```

Modern CSS Workflow Best Practices

1. **Use a CSS Reset or Normalize.css**: Add a CSS reset to ensure consistent rendering across browsers:

```
/* Simple reset */
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}
```

Or link to normalize.css:

```
<link
  rel="stylesheet"

href="https://cdnjs.cloudflare.com/ajax/libs/normalize/8.0.1/normalize.min.c
ss"
/>
```

2. **Organize Your CSS**:
   - Group related styles together
   - Use comments to create sections
   - Consider using a methodology like BEM (discussed later)

```
/* Header styles */
.header {
  /* Header styles here */
}
```

```
/* Main content styles */
.content {
  /* Content styles here */
}
```

3. **Use a Preprocessor** for larger projects:
   - Sass, Less, or Stylus
   - Provides variables, nesting, mixins, and more
4. **Implement a Build Process**:
   - Use tools like Gulp, Webpack, or npm scripts
   - Optimize CSS with minification, autoprefixing
   - Example package.json for a simple setup:

```
{
  "scripts": {
    "build:css": "postcss styles.css -o dist/styles.css"
  },
  "devDependencies": {
    "autoprefixer": "^10.4.0",
    "cssnano": "^5.0.8",
    "postcss": "^8.3.11",
    "postcss-cli": "^9.0.2"
  }
}
```

5. **Version Control**:
   - Track CSS changes with Git
   - Create a .gitignore file for build artifacts
6. **Documentation**:
   - Add comprehensive comments for complex CSS
   - Document your CSS architecture and patterns

Box Model, Positioning, and Layout

The CSS Box Model

The box model is fundamental to CSS layout. Every element on a page is a rectangular box with four parts:

1. **Content**: The actual content (text, images, etc.)
2. **Padding**: Space between content and border
3. **Border**: Line around the padding
4. **Margin**: Space outside the border

```css
.box {
  /* Content dimensions */
  width: 200px;
  height: 100px;

  /* Padding (inner space) */
  padding-top: 10px;
  padding-right: 20px;
  padding-bottom: 10px;
  padding-left: 20px;
  /* Shorthand: padding: 10px 20px 10px 20px; */
  /* Or: padding: 10px 20px; (top/bottom, left/right) */

  /* Border */
  border-width: 2px;
  border-style: solid;
  border-color: #333;
  /* Shorthand: border: 2px solid #333; */

  /* Margin (outer space) */
  margin-top: 10px;
  margin-right: 20px;
  margin-bottom: 10px;
  margin-left: 20px;
  /* Shorthand: margin: 10px 20px 10px 20px; */
  /* Or: margin: 10px auto; (top/bottom, auto centers horizontally) */
}
```

Visual description: The box model creates a series of nested rectangles around the content. First, the content sits in the center, surrounded by padding (like internal cushioning), then a border (like a picture frame), and finally margin space (like the wall space between frames).

Box-sizing property: By default, the width and height properties apply only to the content box. This can make layouts difficult to calculate.

```css
/* Traditional box model */
.box-content {
  box-sizing: content-box; /* Default */
  width: 200px; /* Only the content is 200px wide */
  padding: 20px;
  border: 2px solid #333;
  /* Total width: 200px + 40px + 4px = 244px */
}

/* Alternative box model (recommended) */
.box-border {
  box-sizing: border-box;
  width: 200px; /* The entire box is 200px wide */
  padding: 20px;
  border: 2px solid #333;
```

```
  /* Total width: 200px (content shrinks to accommodate padding and border) */
}

/* Apply border-box globally (recommended) */
* {
  box-sizing: border-box;
}
```

CSS Positioning

CSS offers several positioning schemes to control how elements are placed in the document:

1. **Static Positioning** (default):
   - Elements follow the normal document flow
   - Not affected by top/right/bottom/left properties
2. **Relative Positioning**:
   - Position relative to where it would normally be
   - Doesn't affect the position of other elements

```
.relative-box {
  position: relative;
  top: 20px; /* Moves down 20px from normal position */
  left: 30px; /* Moves right 30px from normal position */
}
```

3. **Absolute Positioning**:
   - Removed from the normal document flow
   - Positioned relative to nearest positioned ancestor (or the document body)

```
.container {
  position: relative; /* Creates a positioning context */
  height: 200px;
}

.absolute-box {
  position: absolute;
  top: 50px; /* 50px from the top of container */
  right: 20px; /* 20px from the right of container */
}
```

4. **Fixed Positioning**:
   - Removed from the normal document flow
   - Positioned relative to the viewport

- Stays in place during scrolling

```
.fixed-header {
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  background-color: white;
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}
```

5. **Sticky Positioning**:
   - Acts like relative positioning until the element scrolls to a threshold
   - Then acts like fixed positioning

```
.sticky-nav {
  position: sticky;
  top: 0; /* Sticks when top reaches 0px */
  background-color: white;
  padding: 10px;
}
```

Visual description: Think of positioning as different ways to place items on a page:

- Static: Items placed in a flow, like text in a book
- Relative: Nudging an item slightly from its normal position
- Absolute: Placing an item exactly where you want within a container, regardless of other items
- Fixed: Pinning an item to the screen so it doesn't move, like a floating chat button
- Sticky: An item that follows you as you scroll until a certain point

CSS Layout Fundamentals

Beyond positioning, CSS provides several ways to create layouts:

1. **Display Property**:
   - Controls how an element behaves in the layout flow

```
/* Block elements take full width and stack vertically */
.block {
  display: block;
}

/* Inline elements flow horizontally and only take needed width */
.inline {
  display: inline;
}
```

```
/* Combination of both behaviors */
.inline-block {
  display: inline-block;
  width: 100px; /* Can set width/height unlike inline */
  height: 100px;
  margin: 10px; /* Can have margins/padding */
}

/* Hides an element completely */
.hidden {
  display: none;
}
```

2. **Floats** (older approach):
    - Float elements to the left or right
    - Other content flows around them

```
.float-left {
  float: left;
  width: 200px;
  margin-right: 20px;
}

/* Clear floats to prevent container collapse */
.clearfix::after {
  content: '';
  display: table;
  clear: both;
}
```

3. **Basic Layout Techniques**:
    - Centering elements horizontally:

```
/* For block elements */
.center-block {
  width: 80%; /* Must have a width */
  margin-left: auto;
  margin-right: auto;
  /* Shorthand: margin: 0 auto; */
}

/* For text */
.center-text {
  text-align: center;
}
```

```
/* For inline/inline-block elements */
.center-inline {
  text-align: center; /* Applied to the parent */
}
```

- Centering elements vertically:

```
/* For single-line text */
.center-text-vertically {
  height: 100px;
  line-height: 100px; /* Equal to height */
}

/* Using positioning (works for any content) */
.absolute-center {
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
}
```

Tailwind equivalent approaches (for comparison):

```
<!-- Box model properties -->
<div class="w-48 h-24 p-4 border-2 border-gray-700 m-4"></div>

<!-- Box sizing border-box is default in Tailwind -->

<!-- Relative positioning -->
<div class="relative top-5 left-8"></div>

<!-- Absolute positioning -->
<div class="relative h-48">
  <div class="absolute top-12 right-5"></div>
</div>

<!-- Fixed positioning -->
<header class="fixed top-0 left-0 w-full bg-white shadow-sm"></header>

<!-- Sticky positioning -->
<nav class="sticky top-0 bg-white p-3"></nav>

<!-- Display properties -->
<div class="block"></div>
<span class="inline"></span>
<div class="inline-block w-24 h-24 m-3"></div>
<div class="hidden"></div>

<!-- Centering -->
```

```
<div class="w-4/5 mx-auto"></div>
<!-- Horizontal center block -->
<div class="text-center"></div>
<!-- Center text -->
<div class="absolute top-1/2 left-1/2 -translate-x-1/2 -translate-y-1/2"></div>
<!-- Absolute center -->
```

Selectors and Specificity

Basic Selectors

CSS selectors allow you to target HTML elements for styling:

1. **Type Selector**: Selects all elements of a given type

```
p {
  /* Styles all <p> elements */
  color: blue;
}
```

2. **Class Selector**: Selects elements with a specific class attribute

```
.highlight {
  /* Styles any element with class="highlight" */
  background-color: yellow;
}
```

3. **ID Selector**: Selects an element with a specific ID attribute

```
#header {
  /* Styles the element with id="header" */
  font-size: 24px;
}
```

4. **Universal Selector**: Selects all elements

```
* {
  /* Styles every element */
  margin: 0;
  padding: 0;
}
```

5. **Attribute Selectors**: Select elements based on attributes

```
[type='text'] {
  /* Styles all elements with type="text" */
  border: 1px solid gray;
}

[href^='https'] {
  /* Styles links starting with https */
  color: green;
}

[href$='.pdf'] {
  /* Styles links ending with .pdf */
  font-weight: bold;
}

[title*='important'] {
  /* Styles elements whose title contains "important" */
  text-decoration: underline;
}
```

Combinators

Combinators allow you to target elements based on their relationship:

1. **Descendant Combinator** (space): Targets a descendant of another element

```
article p {
  /* Styles paragraphs inside articles */
  font-size: 16px;
}
```

2. **Child Combinator** (>): Targets a direct child of another element

```
ul > li {
  /* Styles list items that are direct children of ul */
  margin-bottom: 10px;
}
```

3. **Adjacent Sibling Combinator** (+): Targets an element immediately following another

```
h2 + p {
  /* Styles paragraphs immediately following h2 */
  font-weight: bold;
}
```

4. **General Sibling Combinator** (~): Targets siblings following another element

```
h2 ~ p {
  /* Styles all paragraphs following h2 (same parent) */
  text-indent: 1.5em;
}
```

Pseudo-classes

Pseudo-classes target elements based on their state or position:

1. **User Interaction**:

```
a:link {
  /* Unvisited links */
  color: blue;
}

a:visited {
  /* Visited links */
  color: purple;
}

a:hover {
  /* Mouse over links */
  text-decoration: underline;
}

a:active {
  /* Links being clicked */
  color: red;
}

input:focus {
  /* Input receiving focus */
  border-color: blue;
  outline: none;
}
```

2. **Form States**:

```
input:enabled {
  background-color: white;
}

input:disabled {
  background-color: #f0f0f0;
  cursor: not-allowed;
```

```
}

input:checked {
  /* For checkboxes and radio buttons */
  border-color: green;
}

input:invalid {
  border-color: red;
}
```

3. **Structural Pseudo-classes**:

```
li:first-child {
  font-weight: bold;
}

li:last-child {
  margin-bottom: 0;
}

li:nth-child(odd) {
  background-color: #f9f9f9;
}

li:nth-child(3n) {
  /* Every third item */
  color: red;
}

p:first-of-type {
  font-size: larger;
}

div:empty {
  /* Targets elements with no children */
  display: none;
}
```

Pseudo-elements

Pseudo-elements target specific parts of elements:

```
p::first-line {
  /* Styles the first line of a paragraph */
  font-weight: bold;
}

p::first-letter {
```

```
  /* Styles the first letter of a paragraph */
  font-size: 2em;
  float: left;
  margin-right: 5px;
}

p::before {
  /* Inserts content before an element */
  content: '» ';
  color: blue;
}

p::after {
  /* Inserts content after an element */
  content: ' «';
  color: blue;
}

::selection {
  /* Styles selected text */
  background-color: yellow;
  color: black;
}
```

Specificity

Specificity determines which CSS rule applies when multiple rules target the same element. It's calculated as follows:

1. **Inline styles**: 1000 points
2. **ID selectors**: 100 points per ID
3. **Class, attribute, and pseudo-class selectors**: 10 points each
4. **Element and pseudo-element selectors**: 1 point each

Examples from lowest to highest specificity:

```
/* Specificity: 0,0,0,1 */
p {
  color: black;
}

/* Specificity: 0,0,0,2 */
div p {
  color: red;
}

/* Specificity: 0,0,1,0 */
.text {
  color: green;
}
```

```
/* Specificity: 0,0,1,1 */
p.text {
  color: blue;
}

/* Specificity: 0,0,2,0 */
.sidebar .text {
  color: purple;
}

/* Specificity: 0,1,0,0 */
#content {
  color: orange;
}

/* Specificity: 0,1,0,1 */
#content p {
  color: yellow;
}

/* Specificity: 0,1,1,0 */
#content .text {
  color: pink;
}

/* Specificity: 1,0,0,0 - Inline style */
<p style="color: brown;">Inline styled text</p>

/* !important - Overrides all other styles */
p {
  color: cyan !important;
}
```

Note: `!important` overrides normal specificity rules and should be used sparingly.

Specificity best practices:

- Avoid using IDs for styling when possible
- Prefer classes over element selectors for reusability
- Minimize specificity by keeping selectors simple
- Use specificity deliberately, not accidentally
- Avoid `!important` unless absolutely necessary

Tailwind approach to selectors:

Tailwind uses a flat class structure, avoiding specificity issues by design:

```
<!-- Instead of nested CSS selectors -->
<article class="prose">
  <h2 class="text-xl font-bold mb-4">Heading</h2>
  <p class="text-gray-700 mb-2">First paragraph</p>
```

```
  <p class="text-gray-700 mb-2">Second paragraph</p>
</article>
```

All Tailwind utilities have the same specificity (0,0,1,0), so the order in the HTML determines which takes precedence.

Typography and Text Styling

Font Properties

1. **Font Family**:

```
body {
  /* Prioritized list of font families */
  font-family: 'Helvetica Neue', Arial, sans-serif;
}

/* Using web fonts */
@font-face {
  font-family: 'CustomFont';
  src: url('fonts/custom-font.woff2') format('woff2'), url('fonts/custom-
font.woff')
       format('woff');
  font-weight: normal;
  font-style: normal;
}

.custom-text {
  font-family: 'CustomFont', sans-serif;
}
```

2. **Font Size**:

```
h1 {
  font-size: 2.5rem; /* Relative to root font size */
}

p {
  font-size: 16px; /* Absolute size */
}

.small-text {
  font-size: 0.875em; /* Relative to parent font size */
}

body {
  font-size: 100%; /* Base for rem units (typically 16px) */
}
```

3. **Font Weight**:

```
h1 {
  font-weight: bold; /* Bold (700) */
}

p {
  font-weight: normal; /* Normal (400) */
}

.light {
  font-weight: 300; /* Numeric values: 100-900 */
}

.heavy {
  font-weight: 700;
}
```

4. **Font Style**:

```
.italic {
  font-style: italic;
}

.normal {
  font-style: normal;
}

.oblique {
  font-style: oblique;
}
```

5. **Font Variant**:

```
.small-caps {
  font-variant: small-caps;
}
```

6. **Font Shorthand**:

```
/* font: style variant weight size/line-height family; */
h1 {
  font: italic small-caps bold 2em/1.2 'Times New Roman', serif;
}
```

Text Properties

1. **Text Alignment**:

```
.left {
  text-align: left;
}

.center {
  text-align: center;
}

.right {
  text-align: right;
}

.justify {
  text-align: justify;
}
```

2. **Text Decoration**:

```
.underline {
  text-decoration: underline;
}

.overline {
  text-decoration: overline;
}

.line-through {
  text-decoration: line-through;
}

.fancy-underline {
  text-decoration: underline wavy blue;
}

.no-underline {
  text-decoration: none;
}
```

3. **Text Transform**:

```
.uppercase {
  text-transform: uppercase;
}
```

```
.lowercase {
  text-transform: lowercase;
}

.capitalize {
  text-transform: capitalize; /* First letter of each word */
}
```

4. **Text Spacing**:

```
.letter-spacing {
  letter-spacing: 0.1em;
}

.word-spacing {
  word-spacing: 0.5em;
}

.indent {
  text-indent: 1.5em;
}
```

5. **Line Height**:

```
p {
  line-height: 1.5; /* Unitless (recommended) */
}

.heading {
  line-height: 1.2; /* Tighter for headings */
}

.spaced {
  line-height: 2; /* Double spacing */
}
```

6. **White Space**:

```
.normal {
  white-space: normal; /* Default wrapping */
}

.nowrap {
  white-space: nowrap; /* No wrapping */
}

.pre {
```

```
  white-space: pre; /* Respect spaces and newlines */
}

.pre-wrap {
  white-space: pre-wrap; /* Respect spaces and wrap lines */
}
```

7. **Text Overflow**:

```
.truncate {
  overflow: hidden;
  text-overflow: ellipsis;
  white-space: nowrap;
}

.clip {
  overflow: hidden;
  text-overflow: clip;
  white-space: nowrap;
}
```

8. **Direction and Writing Mode**:

```
.rtl {
  direction: rtl; /* Right to left */
}

.vertical {
  writing-mode: vertical-rl;
}
```

9. **Text Shadow**:

```
.shadow {
  /* horizontal-offset vertical-offset blur color */
  text-shadow: 1px 1px 2px rgba(0, 0, 0, 0.3);
}

.multiple-shadow {
  text-shadow: 1px 1px 2px red, -1px -1px 2px blue;
}
```

Modern Typography Best Practices

1. **Fluid Typography**:

```
body {
  /* Font size scales between 16px and 20px */
  font-size: clamp(1rem, 0.5rem + 1vw, 1.25rem);
}

h1 {
  /* Heading size scales between 2rem and 4rem */
  font-size: clamp(2rem, 1rem + 3vw, 4rem);
}
```

2. **Modular Scale**:

```
:root {
  --ratio: 1.25; /* Major third */
  --base-size: 1rem;
}

body {
  font-size: var(--base-size);
}

h4 {
  font-size: calc(var(--base-size) * var(--ratio));
  /* 1.25rem */
}

h3 {
  font-size: calc(var(--base-size) * var(--ratio) * var(--ratio));
  /* 1.5625rem */
}

h2 {
  font-size: calc(
    var(--base-size) * var(--ratio) * var(--ratio) * var(--ratio)
  );
  /* 1.953125rem */
}

h1 {
  font-size: calc(
    var(--base-size) * var(--ratio) * var(--ratio) * var(--ratio) * var(--
ratio)
  );
  /* 2.44140625rem */
}
```

3. **System Font Stack**:

```css
body {
  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto,
    Oxygen-Sans, Ubuntu, Cantarell, 'Helvetica Neue', sans-serif;
}

/* For monospace */
code {
  font-family: SFMono-Regular, Menlo, Monaco, Consolas, 'Liberation Mono',
    'Courier New', monospace;
}
```

Tailwind typography approach:

```html
<!-- Font family -->
<p class="font-sans">Sans-serif text</p>
<p class="font-serif">Serif text</p>
<p class="font-mono">Monospace text</p>

<!-- Font size -->
<p class="text-xs">Extra small</p>
<p class="text-sm">Small</p>
<p class="text-base">Base size</p>
<p class="text-lg">Large</p>
<p class="text-xl">Extra large</p>
<p class="text-2xl">2X large</p>

<!-- Font weight -->
<p class="font-thin">Thin text</p>
<p class="font-normal">Normal text</p>
<p class="font-medium">Medium text</p>
<p class="font-bold">Bold text</p>

<!-- Font style -->
<p class="italic">Italic text</p>
<p class="not-italic">Non-italic text</p>

<!-- Text alignment -->
<p class="text-left">Left aligned</p>
<p class="text-center">Center aligned</p>
<p class="text-right">Right aligned</p>
<p class="text-justify">Justified text</p>

<!-- Text decoration -->
<p class="underline">Underlined text</p>
<p class="line-through">Strikethrough text</p>
<p class="no-underline">No underline</p>

<!-- Text transform -->
<p class="uppercase">Uppercase text</p>
<p class="lowercase">Lowercase text</p>
<p class="capitalize">Capitalized text</p>
```

```
<p class="normal-case">Normal case</p>

<!-- Letter spacing -->
<p class="tracking-tight">Tight tracking</p>
<p class="tracking-normal">Normal tracking</p>
<p class="tracking-wide">Wide tracking</p>

<!-- Line height -->
<p class="leading-none">No leading</p>
<p class="leading-tight">Tight leading</p>
<p class="leading-normal">Normal leading</p>
<p class="leading-loose">Loose leading</p>

<!-- Text overflow -->
<p class="truncate w-48">
  This text will be truncated with an ellipsis if it's too long
</p>
```

Colors, Gradients, and Backgrounds

Color Values in CSS

CSS supports several color formats:

1. **Named Colors**:

```
.red {
  color: red;
}

.navy {
  color: navy;
}
```

2. **Hexadecimal**:

```
.hex {
  color: #ff0000; /* Red */
}

.hex-short {
  color: #f00; /* Shorthand for #ff0000 */
}

.hex-alpha {
  color: #ff000080; /* Red with 50% opacity */
}
```

3. **RGB and RGBA**:

```
.rgb {
  color: rgb(255, 0, 0); /* Red */
}

.rgba {
  color: rgba(255, 0, 0, 0.5); /* Red with 50% opacity */
}

/* Modern syntax with space separation */
.modern-rgb {
  color: rgb(255 0 0);
}

.modern-rgba {
  color: rgb(255 0 0 / 50%);
}
```

4. **HSL and HSLA** (Hue, Saturation, Lightness):

```
.hsl {
  color: hsl(0, 100%, 50%); /* Red */
}

.hsla {
  color: hsla(0, 100%, 50%, 0.5); /* Red with 50% opacity */
}

/* Modern syntax */
.modern-hsl {
  color: hsl(0 100% 50%);
}

.modern-hsla {
  color: hsl(0 100% 50% / 50%);
}
```

5. **Modern Color Functions**:

```
/* Relative color syntax */
.lighter-blue {
  color: color-mix(in srgb, blue, white 20%);
}

.transparent-red {
  color: color-mix(in srgb, red, transparent 50%);
}
```

Applying Color Properties

1. **Text Color**:

```
.text-color {
  color: #333; /* Text color */
}
```

2. **Background Color**:

```
.bg-color {
  background-color: #f5f5f5;
}
```

3. **Border Color**:

```
.border-color {
  border: 1px solid #ddd;
}

.specific-border {
  border-top-color: red;
  border-right-color: green;
  border-bottom-color: blue;
  border-left-color: yellow;
}
```

4. **Outline Color**:

```
.outline-color {
  outline: 2px solid #007bff;
}
```

5. **Box Shadow Color**:

```
.shadow-color {
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
}
```

6. **Text Shadow Color**:

```
.text-shadow-color {
  text-shadow: 1px 1px 2px rgba(0, 0, 0, 0.3);
}
```

Gradients

CSS gradients create smooth transitions between colors:

1. **Linear Gradients**:

```
.linear-gradient {
  /* Direction, color stops */
  background-image: linear-gradient(to right, red, yellow);
}

.linear-angle {
  background-image: linear-gradient(45deg, red, yellow);
}

.multi-color {
  background-image: linear-gradient(to bottom, red, yellow, green, blue);
}

.positioned-stops {
  background-image: linear-gradient(
    to right,
    red,
    /* 0% position (default) */ yellow 25%,
    /* Yellow at 25% */ green 50%,
    /* Green at 50% */ blue 100% /* Blue at 100% */
  );
}

.hard-stop {
  background-image: linear-gradient(
    to right,
    red 50%,
    /* Red until 50% */ blue 50% /* Blue from 50% */
  );
}
```

2. **Radial Gradients**:

```
.radial-gradient {
  background-image: radial-gradient(circle, red, yellow);
}

.radial-positioned {
```

```
  background-image: radial-gradient(circle at top right, red, yellow);
}

.radial-sized {
  background-image: radial-gradient(circle 200px, /* Size */ red, yellow);
}

.elliptical {
  background-image: radial-gradient(ellipse, red, yellow);
}
```

3. **Conic Gradients**:

```
.conic-gradient {
  background-image: conic-gradient(red, yellow, green, blue, purple, red);
}

.conic-positioned {
  background-image: conic-gradient(
    from 45deg at center,
    red,
    yellow,
    green,
    blue,
    purple,
    red
  );
}

.pie-chart {
  background-image: conic-gradient(
    red 0deg 90deg,
    /* 25% */ yellow 90deg 180deg,
    /* 25% */ green 180deg 270deg,
    /* 25% */ blue 270deg 360deg /* 25% */
  );
  border-radius: 50%;
  width: 200px;
  height: 200px;
}
```

4. **Repeating Gradients**:

```
.repeating-linear {
  background-image: repeating-linear-gradient(
    45deg,
    red,
    red 10px,
    yellow 10px,
```

```
    yellow 20px
  );
}

.repeating-radial {
  background-image: repeating-radial-gradient(
    circle,
    red,
    red 10px,
    yellow 10px,
    yellow 20px
  );
}

.repeating-conic {
  background-image: repeating-conic-gradient(
    red 0deg 10deg,
    yellow 10deg 20deg
  );
}
```

5. **Multiple Gradients**:

```
.multiple-gradients {
  background-image: linear-gradient(
      rgba(0, 0, 255, 0.5),
      rgba(255, 255, 0, 0.5)
    ), linear-gradient(to right, red, purple);
}
```

Backgrounds

1. **Background Images**:

```
.bg-image {
  background-image: url('image.jpg');
}

.multiple-bg {
  background-image: url('overlay.png'), url('background.jpg');
}
```

2. **Background Size**:

```
.bg-auto {
  background-size: auto; /* Original size */
}
```

```css
.bg-cover {
  background-size: cover; /* Cover the entire container */
}

.bg-contain {
  background-size: contain; /* Fit within the container */
}

.bg-specific {
  background-size: 200px 100px;
}

.bg-mixed {
  background-size: 50% auto; /* 50% width, auto height */
}
```

3. **Background Position**:

```css
.bg-top-left {
  background-position: top left;
}

.bg-center {
  background-position: center;
}

.bg-custom {
  background-position: 25% 75%;
}

.bg-specific-pos {
  background-position: 20px 50px;
}
```

4. **Background Repeat**:

```css
.bg-repeat {
  background-repeat: repeat; /* Default */
}

.bg-no-repeat {
  background-repeat: no-repeat;
}

.bg-repeat-x {
  background-repeat: repeat-x; /* Horizontal only */
}
```

```css
.bg-repeat-y {
  background-repeat: repeat-y; /* Vertical only */
}

.bg-space {
  background-repeat: space; /* Space evenly without clipping */
}

.bg-round {
  background-repeat: round; /* Stretch/squeeze to prevent gaps */
}
```

5. **Background Attachment**:

```css
.bg-scroll {
  background-attachment: scroll; /* Scrolls with content (default) */
}

.bg-fixed {
  background-attachment: fixed; /* Fixed to viewport */
}

.bg-local {
  background-attachment: local; /* Scrolls with element's content */
}
```

6. **Background Shorthand**:

```css
.bg-shorthand {
  /* color image position/size repeat attachment */
  background: #f0f0f0 url('bg.jpg') center/cover no-repeat fixed;
}
```

7. **Background Clip and Origin**:

```css
.bg-clip-border {
  background-clip: border-box; /* Default */
}

.bg-clip-padding {
  background-clip: padding-box; /* Only in padding and content */
}

.bg-clip-content {
  background-clip: content-box; /* Only in content */
}
```

```
  .bg-clip-text {
    background-clip: text;
    color: transparent;
    background-image: linear-gradient(to right, red, blue);
  }
```

Tailwind color and background approaches:

```
<!-- Text colors -->
<p class="text-red-500">Red text</p>
<p class="text-blue-700">Dark blue text</p>
<p class="text-green-600">Green text</p>
<p class="text-gray-800">Dark gray text</p>

<!-- Background colors -->
<div class="bg-red-100">Light red background</div>
<div class="bg-blue-200">Light blue background</div>
<div class="bg-green-300">Light green background</div>
<div class="bg-gray-100">Light gray background</div>

<!-- Opacity -->
<div class="text-blue-500 text-opacity-75">Semi-transparent text</div>
<div class="bg-red-500 bg-opacity-50">Semi-transparent background</div>

<!-- Gradients -->
<div class="bg-gradient-to-r from-red-500 to-blue-500">
  Red to blue gradient
</div>
<div class="bg-gradient-to-br from-purple-500 via-pink-500 to-red-500">
  Three-color gradient
</div>

<!-- Background image utilities -->
<div
  class="bg-cover bg-center bg-no-repeat"
  style="background-image: url('image.jpg');"
>
  Background image content
</div>
```

Flexbox

Flexbox Fundamentals

Flexbox (Flexible Box Layout) is a one-dimensional layout method designed for arranging items in rows or columns. It's ideal for distributing space and aligning items when available space is unknown or dynamic.

Basic Flexbox Container:

```
.container {
  display: flex;
  /* or display: inline-flex; for inline flex container */
}
```

Visual description: When you apply `display: flex`, the container transforms into a flex container, and all its direct children become flex items. These items automatically align side by side (by default) and can grow or shrink to fill the available space.

Flex Container Properties

1. **Flex Direction**:

```
.row {
  flex-direction: row; /* Default: left to right */
}

.row-reverse {
  flex-direction: row-reverse; /* Right to left */
}

.column {
  flex-direction: column; /* Top to bottom */
}

.column-reverse {
  flex-direction: column-reverse; /* Bottom to top */
}
```

2. **Flex Wrap**:

```
.nowrap {
  flex-wrap: nowrap; /* Default: all items on one line */
}

.wrap {
  flex-wrap: wrap; /* Items wrap to multiple lines */
}

.wrap-reverse {
  flex-wrap: wrap-reverse; /* Items wrap in reverse */
}
```

3. **Flex Flow** (shorthand for flex-direction and flex-wrap):

```
```

```
.row-wrap {
  flex-flow: row wrap;
}

.column-nowrap {
  flex-flow: column nowrap;
}
```

4. **Justify Content** (alignment along the main axis):

```
.justify-start {
  justify-content: flex-start; /* Default */
}

.justify-end {
  justify-content: flex-end;
}

.justify-center {
  justify-content: center;
}

.justify-between {
  justify-content: space-between; /* Equal space between items */
}

.justify-around {
  justify-content: space-around; /* Equal space around items */
}

.justify-evenly {
  justify-content: space-evenly; /* Equal space between and at edges */
}
```

5. **Align Items** (alignment along the cross axis):

```
.items-start {
  align-items: flex-start;
}

.items-end {
  align-items: flex-end;
}

.items-center {
  align-items: center;
}

.items-baseline {
```

```
  align-items: baseline; /* Align by text baseline */
}

.items-stretch {
  align-items: stretch; /* Default: stretch to fill container */
}
```

6. **Align Content** (alignment of lines when wrapping):

```
.content-start {
  align-content: flex-start;
}

.content-end {
  align-content: flex-end;
}

.content-center {
  align-content: center;
}

.content-between {
  align-content: space-between;
}

.content-around {
  align-content: space-around;
}

.content-stretch {
  align-content: stretch; /* Default */
}
```

7. **Gap** (space between flex items):

```
.row-gap {
  row-gap: 10px;
}

.column-gap {
  column-gap: 20px;
}

.gap {
  gap: 10px; /* Same gap in both directions */
}

.gap-x-y {
```

```
  gap: 10px 20px; /* row-gap column-gap */
}
```

Flex Item Properties

1. **Order** (controls the order of items):

```
.first {
  order: -1; /* Appears before others */
}

.default-order {
  order: 0; /* Default */
}

.last {
  order: 1; /* Appears after others */
}
```

2. **Flex Grow** (ability to grow):

```
.no-grow {
  flex-grow: 0; /* Default: doesn't grow */
}

.grow {
  flex-grow: 1; /* Grows to fill space */
}

.grow-more {
  flex-grow: 2; /* Grows twice as much as .grow */
}
```

3. **Flex Shrink** (ability to shrink):

```
.no-shrink {
  flex-shrink: 0; /* Doesn't shrink */
}

.shrink {
  flex-shrink: 1; /* Default: can shrink */
}

.shrink-more {
  flex-shrink: 2; /* Shrinks twice as much */
}
```

4. **Flex Basis** (initial size before growing/shrinking):

```
.basis-auto {
  flex-basis: auto; /* Default: size based on content */
}

.basis-0 {
  flex-basis: 0; /* Start from zero width */
}

.basis-100 {
  flex-basis: 100px; /* Start at 100px */
}

.basis-50-percent {
  flex-basis: 50%; /* Start at 50% of container */
}
```

5. **Flex** (shorthand for grow, shrink, and basis):

```
.default {
  flex: 0 1 auto; /* Default: don't grow, can shrink, auto basis */
}

.grow-evenly {
  flex: 1; /* Same as flex: 1 1 0 */
}

.flexible {
  flex: 1 1 auto; /* Can grow and shrink from auto size */
}

.fixed-size {
  flex: 0 0 200px; /* Fixed width, won't grow or shrink */
}
```

6. **Align Self** (override align-items for specific items):

```
.self-auto {
  align-self: auto; /* Default: use container's align-items */
}

.self-start {
  align-self: flex-start;
}

.self-end {
  align-self: flex-end;
```

```
}

.self-center {
  align-self: center;
}

.self-baseline {
  align-self: baseline;
}

.self-stretch {
  align-self: stretch;
}
```

Common Flexbox Patterns

1. **Centering an Element**:

```
.center-both {
  display: flex;
  justify-content: center;
  align-items: center;
  height: 300px; /* Container height */
}
```

2. **Navigation Bar**:

```
.navbar {
  display: flex;
  justify-content: space-between;
  padding: 10px;
}

.nav-links {
  display: flex;
  gap: 20px;
}
```

3. **Card Layout**:

```
.card-container {
  display: flex;
  flex-wrap: wrap;
  gap: 20px;
}

.card {
```

```
  flex: 0 0 calc(33.333% - 20px);
  display: flex;
  flex-direction: column;
}

.card-body {
  flex-grow: 1; /* Takes up remaining space */
}
```

4. **Holy Grail Layout**:

```
.holy-grail {
  display: flex;
  flex-direction: column;
  min-height: 100vh;
}

.holy-grail-header,
.holy-grail-footer {
  flex: 0 0 auto;
}

.holy-grail-body {
  display: flex;
  flex: 1 0 auto;
}

.holy-grail-nav,
.holy-grail-aside {
  flex: 0 0 200px;
}

.holy-grail-nav {
  order: -1;
}

.holy-grail-content {
  flex: 1 0 auto;
}

/* Responsive */
@media (max-width: 768px) {
  .holy-grail-body {
    flex-direction: column;
  }

  .holy-grail-nav,
  .holy-grail-aside {
    flex: 0 0 auto;
  }
}
```

5. **Equal Height Columns**:

```
.equal-columns {
  display: flex;
}

.column {
  flex: 1;
}
```

Tailwind Flexbox approach:

```
<!-- Basic flex container -->
<div class="flex">
  <div>Item 1</div>
  <div>Item 2</div>
  <div>Item 3</div>
</div>

<!-- Flex direction -->
<div class="flex flex-row">Row (default)</div>
<div class="flex flex-row-reverse">Row reversed</div>
<div class="flex flex-col">Column</div>
<div class="flex flex-col-reverse">Column reversed</div>

<!-- Flex wrap -->
<div class="flex flex-nowrap">No wrap (default)</div>
<div class="flex flex-wrap">Wrap</div>
<div class="flex flex-wrap-reverse">Wrap reversed</div>

<!-- Justify content -->
<div class="flex justify-start">Start (default)</div>
<div class="flex justify-end">End</div>
<div class="flex justify-center">Center</div>
<div class="flex justify-between">Space between</div>
<div class="flex justify-around">Space around</div>
<div class="flex justify-evenly">Space evenly</div>

<!-- Align items -->
<div class="flex items-start">Start</div>
<div class="flex items-end">End</div>
<div class="flex items-center">Center</div>
<div class="flex items-baseline">Baseline</div>
<div class="flex items-stretch">Stretch (default)</div>

<!-- Gap -->
<div class="flex gap-4">Gap of 1rem</div>
<div class="flex gap-x-4">Horizontal gap</div>
<div class="flex gap-y-4">Vertical gap</div>
```

```
<!-- Flex item properties -->
<div class="flex">
  <div class="order-3">Order 3</div>
  <div class="order-1">Order 1</div>
  <div class="order-2">Order 2</div>
</div>

<div class="flex">
  <div class="flex-grow-0">Don't grow</div>
  <div class="flex-grow">Grow</div>
  <div class="flex-grow-0">Don't grow</div>
</div>

<div class="flex">
  <div class="flex-shrink-0 w-64">Don't shrink</div>
  <div class="flex-shrink w-64">Shrink</div>
</div>

<div class="flex">
  <div class="flex-none w-32">Fixed width</div>
  <div class="flex-1">Grow and shrink</div>
  <div class="flex-auto">Natural width</div>
</div>

<div class="flex items-start">
  <div class="self-auto">Default alignment</div>
  <div class="self-start">Start</div>
  <div class="self-end">End</div>
  <div class="self-center">Center</div>
  <div class="self-stretch">Stretch</div>
</div>

<!-- Common patterns -->
<div class="flex justify-center items-center h-screen">
  <div>Perfectly centered</div>
</div>

<nav class="flex justify-between p-4">
  <div>Logo</div>
  <div class="flex gap-5">
    <a href="#">Link 1</a>
    <a href="#">Link 2</a>
    <a href="#">Link 3</a>
  </div>
</nav>
```

CSS Grid

CSS Grid Fundamentals

CSS Grid Layout is a two-dimensional layout system designed for laying out items in rows and columns. Unlike Flexbox, which deals with either rows OR columns, Grid can handle both simultaneously.

Basic Grid Container:

```
.container {
  display: grid;
  /* or display: inline-grid; for inline grid container */
}
```

Visual description: When you apply `display: grid`, the container becomes a grid container, and all its direct children become grid items. The grid itself is initially invisible — it's a layout framework that items align to.

Grid Container Properties

1. **Grid Template Columns and Rows**:

```
.three-columns {
  /* Three equal columns of 100px each */
  grid-template-columns: 100px 100px 100px;
}

.two-rows {
  /* Two rows: 100px and 200px */
  grid-template-rows: 100px 200px;
}

.repeat-columns {
  /* Same as 100px 100px 100px 100px */
  grid-template-columns: repeat(4, 100px);
}

.mixed-columns {
  /* First and last 100px, middle 200px */
  grid-template-columns: 100px 200px 100px;
}

.fr-unit {
  /* Fractional units - divide available space */
  grid-template-columns: 1fr 2fr 1fr; /* 1:2:1 ratio */
}

.mixed-units {
  /* Fixed and flexible widths */
  grid-template-columns: 100px 1fr 100px; /* Fixed-Flexible-Fixed */
}

.minmax {
  /* Columns with minimum and maximum sizes */
```

```
  grid-template-columns: minmax(100px, 200px) minmax(200px, 1fr);
}

.auto-fill {
  /* Creates as many columns as fit */
  grid-template-columns: repeat(auto-fill, minmax(150px, 1fr));
}

.auto-fit {
  /* Like auto-fill but collapses empty tracks */
  grid-template-columns: repeat(auto-fit, minmax(150px, 1fr));
}
```

2. **Grid Template Areas**:

```
.grid-layout {
  grid-template-areas:
    'header header header'
    'sidebar content content'
    'footer footer footer';
  grid-template-columns: 200px 1fr 1fr;
  grid-template-rows: 100px 1fr 100px;
}

.header {
  grid-area: header;
}

.sidebar {
  grid-area: sidebar;
}

.content {
  grid-area: content;
}

.footer {
  grid-area: footer;
}
```

3. **Grid Template** (shorthand for grid-template-areas, columns, and rows):

```
.grid-template {
  grid-template:
    'header header header' 100px
    'sidebar content content' 1fr
    'footer footer footer' 100px
    / 200px 1fr 1fr;
}
```

4. **Grid Column/Row Gap**:

```
.column-gap {
  grid-column-gap: 20px; /* Space between columns */
}

.row-gap {
  grid-row-gap: 10px; /* Space between rows */
}

.grid-gap {
  /* Shorthand for both */
  grid-gap: 10px 20px; /* row-gap column-gap */
}

/* Modern syntax */
.modern-gap {
  column-gap: 20px;
  row-gap: 10px;
  gap: 10px 20px; /* Shorthand */
}
```

5. **Justify Items** (horizontal alignment):

```
.justify-start {
  justify-items: start; /* Left align */
}

.justify-end {
  justify-items: end; /* Right align */
}

.justify-center {
  justify-items: center; /* Center align */
}

.justify-stretch {
  justify-items: stretch; /* Default: fill cell width */
}
```

6. **Align Items** (vertical alignment):

```
.align-start {
  align-items: start; /* Top align */
}

.align-end {
```

```
  align-items: end; /* Bottom align */
}

.align-center {
  align-items: center; /* Center align */
}

.align-stretch {
  align-items: stretch; /* Default: fill cell height */
}
```

7. **Place Items** (shorthand for align-items and justify-items):

```
.place-center {
  place-items: center; /* Center both horizontally and vertically */
}

.place-start-end {
  place-items: start end; /* align-items: start; justify-items: end; */
}
```

8. **Justify Content** (horizontal alignment of entire grid):

```
.justify-content-start {
  justify-content: start; /* Default: align grid to start */
}

.justify-content-end {
  justify-content: end; /* Align grid to end */
}

.justify-content-center {
  justify-content: center; /* Center grid horizontally */
}

.justify-content-space-between {
  justify-content: space-between; /* Space columns evenly */
}

.justify-content-space-around {
  justify-content: space-around; /* Space around columns */
}

.justify-content-space-evenly {
  justify-content: space-evenly; /* Equal spacing */
}
```

9. **Align Content** (vertical alignment of entire grid):

```css
.align-content-start {
  align-content: start; /* Default: align grid to top */
}

.align-content-end {
  align-content: end; /* Align grid to bottom */
}

.align-content-center {
  align-content: center; /* Center grid vertically */
}

.align-content-space-between {
  align-content: space-between; /* Space rows evenly */
}

.align-content-space-around {
  align-content: space-around; /* Space around rows */
}

.align-content-space-evenly {
  align-content: space-evenly; /* Equal spacing */
}
```

10. **Place Content** (shorthand for align-content and justify-content):

```css
.place-content-center {
  place-content: center; /* Center grid both ways */
}

.place-content-start-end {
  place-content: start end; /* align-content: start; justify-content: end;
*/
}
```

11. **Grid Auto Columns/Rows** (for implicit tracks):

```css
.auto-rows {
  grid-auto-rows: 100px; /* All implicit rows are 100px */
}

.auto-columns {
  grid-auto-columns: 1fr; /* All implicit columns are 1fr */
}
```

12. **Grid Auto Flow** (controls auto-placement algorithm):

```
.auto-flow-row {
  grid-auto-flow: row; /* Default: fill by row */
}

.auto-flow-column {
  grid-auto-flow: column; /* Fill by column */
}

.auto-flow-dense {
  grid-auto-flow: row dense; /* Fill gaps when possible */
}
```

13. **Grid** (super shorthand for all grid properties):

```
.grid-shorthand {
  grid:
    'header header header' 100px
    'sidebar content content' 1fr
    'footer footer footer' 100px
    / 200px 1fr 1fr;
  grid-gap: 10px;
}
```

Grid Item Properties

1. **Grid Column/Row Start/End**:

```
.col-span-2 {
  grid-column-start: 1;
  grid-column-end: 3; /* End at line 3 (spans 2 columns) */
}

.row-span-3 {
  grid-row-start: 2;
  grid-row-end: 5; /* End at line 5 (spans 3 rows) */
}

/* Negative values count backwards from the end */
.col-from-end {
  grid-column-start: -3; /* 3rd line from the end */
  grid-column-end: -1; /* Last line */
}
```

2. **Grid Column/Row** (shorthand for start/end):

```
.col-span-2-shorthand {
  grid-column: 1 / 3; /* Start at line 1, end at line 3 */
}

.row-span-3-shorthand {
  grid-row: 2 / 5; /* Start at line 2, end at line 5 */
}

/* Using span keyword */
.col-span {
  grid-column: 1 / span 2; /* Start at line 1, span 2 columns */
}

.row-span {
  grid-row: span 3; /* Span 3 rows, auto-placement for start */
}

/* Using line names if defined in grid-template */
.header-span {
  grid-column: header-start / header-end;
}
```

3. **Grid Area** (shorthand for row-start, column-start, row-end, column-end):

```
.area-span {
  grid-area: 2 / 1 / 5 / 3; /* row-start / column-start / row-end / column-
end */
}

/* Using named template areas */
.header {
  grid-area: header; /* Uses name from grid-template-areas */
}
```

4. **Justify Self** (horizontal alignment of individual item):

```
.justify-self-start {
  justify-self: start; /* Left align this item */
}

.justify-self-end {
  justify-self: end; /* Right align this item */
}

.justify-self-center {
  justify-self: center; /* Center this item horizontally */
}

.justify-self-stretch {
```

```
  justify-self: stretch; /* Default: fill cell width */
}
```

5. **Align Self** (vertical alignment of individual item):

```
.align-self-start {
  align-self: start; /* Top align this item */
}

.align-self-end {
  align-self: end; /* Bottom align this item */
}

.align-self-center {
  align-self: center; /* Center this item vertically */
}

.align-self-stretch {
  align-self: stretch; /* Default: fill cell height */
}
```

6. **Place Self** (shorthand for align-self and justify-self):

```
.place-self-center {
  place-self: center; /* Center this item both ways */
}

.place-self-start-end {
  place-self: start end; /* align-self: start; justify-self: end; */
}
```

Common CSS Grid Patterns

1. **Basic Grid Layout**:

```
.basic-grid {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-gap: 20px;
}
```

2. **Responsive Grid**:

```
.responsive-grid {
  display: grid;
```

```
  grid-template-columns: repeat(auto-fill, minmax(250px, 1fr));
  grid-gap: 20px;
}
```

3. **Holy Grail Layout with Grid**:

```
.holy-grail {
  display: grid;
  grid-template:
    'header header header' auto
    'nav main aside' 1fr
    'footer footer footer' auto
    / 200px 1fr 200px;
  min-height: 100vh;
}

header {
  grid-area: header;
}
nav {
  grid-area: nav;
}
main {
  grid-area: main;
}
aside {
  grid-area: aside;
}
footer {
  grid-area: footer;
}

@media (max-width: 768px) {
  .holy-grail {
    grid-template:
      'header' auto
      'nav' auto
      'main' 1fr
      'aside' auto
      'footer' auto
      / 1fr;
  }
}
```

4. **Card Layout with Featured Card**:

```
.card-grid {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
```

```
  grid-gap: 20px;
}

.featured-card {
  grid-column: span 2;
  grid-row: span 2;
}
```

5. **Masonry-like Layout**:

```
.masonry-grid {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(250px, 1fr));
  grid-auto-rows: 20px;
  grid-gap: 20px;
}

/* Each item needs a grid-row-end based on its height */
.masonry-item {
  grid-row-end: span calc(var(--item-height) / 20);
}

/* JavaScript would be needed to calculate each item's span */
```

Tailwind Grid approach:

```
<!-- Basic grid -->
<div class="grid grid-cols-3 gap-4">
  <div>Item 1</div>
  <div>Item 2</div>
  <div>Item 3</div>
  <div>Item 4</div>
  <div>Item 5</div>
  <div>Item 6</div>
</div>

<!-- Grid template columns -->
<div class="grid grid-cols-3">3 equal columns</div>
<div class="grid grid-cols-4">4 equal columns</div>
<div class="grid grid-cols-12">12 equal columns</div>
<div class="grid grid-cols-none">No columns</div>

<!-- Grid column span -->
<div class="grid grid-cols-3 gap-4">
  <div class="col-span-2">Spans 2 columns</div>
  <div>Spans 1 column</div>
  <div>Spans 1 column</div>
  <div class="col-span-3">Spans all 3 columns</div>
</div>
```

```
<!-- Grid template rows -->
<div class="grid grid-rows-3 grid-flow-col gap-4">
  <div>Row 1</div>
  <div>Row 2</div>
  <div>Row 3</div>
  <div>Row 1</div>
  <div>Row 2</div>
  <div>Row 3</div>
</div>

<!-- Grid row span -->
<div class="grid grid-rows-3 grid-flow-col gap-4">
  <div class="row-span-3">Spans all 3 rows</div>
  <div class="row-span-2">Spans 2 rows</div>
  <div>Spans 1 row</div>
</div>

<!-- Grid gap -->
<div class="grid grid-cols-3 gap-4">Gap of 1rem</div>
<div class="grid grid-cols-3 gap-x-4">Horizontal gap only</div>
<div class="grid grid-cols-3 gap-y-4">Vertical gap only</div>

<!-- Responsive grid -->
<div
  class="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-4 gap-4"
>
  <div>Responsive item</div>
  <!-- More items... -->
</div>

<!-- Alignment -->
<div class="grid grid-cols-3 justify-items-center">
  <div>Center aligned item</div>
  <!-- More items... -->
</div>

<div class="grid grid-cols-3 content-center h-64">
  <div>Content center</div>
  <!-- More items... -->
</div>

<div class="grid grid-cols-3">
  <div class="justify-self-end">Self end aligned</div>
  <div class="justify-self-center">Self center aligned</div>
  <div class="justify-self-start">Self start aligned</div>
</div>

<!-- Auto grid -->
<div class="grid grid-cols-none auto-cols-auto">Auto columns</div>
<div class="grid grid-rows-none auto-rows-auto">Auto rows</div>
```

Transitions and Animations

CSS Transitions

Transitions provide a way to control animation speed when changing CSS properties. Instead of having property changes take effect immediately, you can cause the changes to occur over a period of time.

1. **Basic Transition**:

```
.button {
  background-color: blue;
  color: white;
  padding: 10px 20px;
  transition: background-color 0.3s ease;
}

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

2. **Transition Properties**:

```
.box {
  width: 100px;
  height: 100px;
  background-color: red;

  /* Individual properties */
  transition-property: width, height, background-color;
  transition-duration: 0.5s, 1s, 2s;
  transition-timing-function: ease, ease-in, ease-out;
  transition-delay: 0s, 0.2s, 0.5s;
}

.box:hover {
  width: 200px;
  height: 200px;
  background-color: blue;
}
```

3. **Transition Shorthand**:

```
.box {
  /* property duration timing-function delay */
  transition: width 0.5s ease 0s, height 1s ease-in 0.2s,
    background-color 2s ease-out 0.5s;
}
```

```
/* All properties */
.transition-all {
  transition: all 0.3s ease;
}
```

4. **Timing Functions**:

```
.ease {
  transition-timing-function: ease; /* Default: slow start, fast middle,
slow end */
}

.linear {
  transition-timing-function: linear; /* Constant speed */
}

.ease-in {
  transition-timing-function: ease-in; /* Slow start, fast end */
}

.ease-out {
  transition-timing-function: ease-out; /* Fast start, slow end */
}

.ease-in-out {
  transition-timing-function: ease-in-out; /* Slow start and end */
}

.step {
  transition-timing-function: steps(5, end); /* 5 discrete steps */
}

.custom {
  /* Define your own timing curve */
  transition-timing-function: cubic-bezier(0.68, -0.55, 0.27, 1.55);
}
```

5. **Transition Events with JavaScript**:

```
// Get the element
const element = document.querySelector('.box');

// Add event listeners
element.addEventListener('transitionstart', () => {
  console.log('Transition started');
});

element.addEventListener('transitionend', () => {
  console.log('Transition ended');
```

```
});

element.addEventListener('transitioncancel', () => {
  console.log('Transition cancelled');
});
```

CSS Animations

While transitions are great for simple state changes, animations are more powerful for complex, multi-step animations.

1. **Basic Animation with @keyframes**:

```
/* Define the animation */
@keyframes slide-in {
  from {
    transform: translateX(-100%);
  }
  to {
    transform: translateX(0);
  }
}

/* Apply the animation */
.slide-in-element {
  animation: slide-in 1s ease-out;
}
```

2. **Multi-step Animation**:

```
@keyframes pulse {
  0% {
    transform: scale(1);
    opacity: 1;
  }
  50% {
    transform: scale(1.2);
    opacity: 0.7;
  }
  100% {
    transform: scale(1);
    opacity: 1;
  }
}

.pulse-element {
  animation: pulse 2s infinite;
}
```

3. **Animation Properties**:

```
.animated-box {
  /* Individual properties */
  animation-name: pulse;
  animation-duration: 2s;
  animation-timing-function: ease-in-out;
  animation-delay: 0.5s;
  animation-iteration-count: infinite;
  animation-direction: alternate;
  animation-fill-mode: forwards;
  animation-play-state: running;
}

/* Pause animation on hover */
.animated-box:hover {
  animation-play-state: paused;
}
```

4. **Animation Shorthand**:

```
.animated-box {
  /* name duration timing-function delay iteration-count direction fill-mode
play-state */
  animation: pulse 2s ease-in-out 0.5s infinite alternate forwards running;
}
```

5. **Multiple Animations**:

```
.multiple-animations {
  animation: pulse 2s infinite, slide-in 1s ease-out;
}
```

6. **Animation Fill Modes**:

```
.fill-none {
  animation: pulse 2s;
  animation-fill-mode: none; /* Default: no styles before or after */
}

.fill-forwards {
  animation: pulse 2s;
  animation-fill-mode: forwards; /* Retain end state */
}

.fill-backwards {
```

```
  animation: pulse 2s;
  animation-fill-mode: backwards; /* Apply first frame during delay */
}

.fill-both {
  animation: pulse 2s;
  animation-fill-mode: both; /* Combine forwards and backwards */
}
```

7. **Animation Direction**:

```
.direction-normal {
  animation: pulse 2s infinite;
  animation-direction: normal; /* Default: 0% to 100% */
}

.direction-reverse {
  animation: pulse 2s infinite;
  animation-direction: reverse; /* 100% to 0% */
}

.direction-alternate {
  animation: pulse 2s infinite;
  animation-direction: alternate; /* 0% to 100%, then 100% to 0% */
}

.direction-alternate-reverse {
  animation: pulse 2s infinite;
  animation-direction: alternate-reverse; /* 100% to 0%, then 0% to 100% */
}
```

8. **Animation Events with JavaScript**:

```
// Get the element
const element = document.querySelector('.animated-box');

// Add event listeners
element.addEventListener('animationstart', () => {
  console.log('Animation started');
});

element.addEventListener('animationend', () => {
  console.log('Animation ended');
});

element.addEventListener('animationiteration', () => {
  console.log('Animation iteration');
});
```

Advanced Animation Techniques

1. **Sprite Animation**:

```
@keyframes sprite {
  0% {
    background-position: 0px 0px;
  }
  100% {
    background-position: -1000px 0px;
  } /* 10 frames at 100px each */
}

.sprite {
  width: 100px;
  height: 100px;
  background-image: url('sprite-sheet.png');
  animation: sprite 1s steps(10) infinite;
}
```

2. **3D Transformations with Animation**:

```
.cube {
  width: 100px;
  height: 100px;
  position: relative;
  transform-style: preserve-3d;
  animation: rotate-cube 5s linear infinite;
}

.face {
  position: absolute;
  width: 100%;
  height: 100%;
  backface-visibility: hidden;
}

.front {
  transform: translateZ(50px);
}
.back {
  transform: rotateY(180deg) translateZ(50px);
}
.right {
  transform: rotateY(90deg) translateZ(50px);
}
.left {
  transform: rotateY(-90deg) translateZ(50px);
}
.top {
```

```
  transform: rotateX(90deg) translateZ(50px);
}
.bottom {
  transform: rotateX(-90deg) translateZ(50px);
}

@keyframes rotate-cube {
  0% {
    transform: rotateY(0) rotateX(0);
  }
  100% {
    transform: rotateY(360deg) rotateX(360deg);
  }
}
```

3. **Staggered Animations**:

```
.staggered-list li {
  animation: fade-in 0.5s ease-out forwards;
  opacity: 0;
}

.staggered-list li:nth-child(1) {
  animation-delay: 0.1s;
}
.staggered-list li:nth-child(2) {
  animation-delay: 0.2s;
}
.staggered-list li:nth-child(3) {
  animation-delay: 0.3s;
}
.staggered-list li:nth-child(4) {
  animation-delay: 0.4s;
}
.staggered-list li:nth-child(5) {
  animation-delay: 0.5s;
}

@keyframes fade-in {
  from {
    opacity: 0;
    transform: translateY(20px);
  }
  to {
    opacity: 1;
    transform: translateY(0);
  }
}
```

4. **Responsive Animations**:

```css
@keyframes responsive-animation {
  from {
    transform: translateX(0);
  }
  to {
    transform: translateX(100px);
  }
}

.responsive-element {
  animation: responsive-animation 2s infinite alternate;
}

@media (max-width: 768px) {
  @keyframes responsive-animation {
    from {
      transform: translateX(0);
    }
    to {
      transform: translateX(50px);
    }
  }
}
```

5. **Scroll-triggered Animations**:

```css
.fade-in-element {
  opacity: 0;
  transform: translateY(20px);
  transition: opacity 0.5s, transform 0.5s;
}

.fade-in-element.visible {
  opacity: 1;
  transform: translateY(0);
}
```

```javascript
// JavaScript to add .visible class when element is in viewport
const elements = document.querySelectorAll('.fade-in-element');

const isInViewport = (element) => {
  const rect = element.getBoundingClientRect();
  return (
    rect.top >= 0 &&
    rect.left >= 0 &&
    rect.bottom <= window.innerHeight &&
    rect.right <= window.innerWidth
  );
```

```
};

const checkElements = () => {
  elements.forEach((element) => {
    if (isInViewport(element)) {
      element.classList.add('visible');
    }
  });
};

// Check on load and scroll
window.addEventListener('load', checkElements);
window.addEventListener('scroll', checkElements);
```

Tailwind transitions and animations approach:

```
<!-- Transitions -->
<button class="bg-blue-500 hover:bg-blue-700 transition duration-300">
  Simple Transition
</button>

<button
  class="bg-green-500 hover:bg-green-700 transition-colors duration-500 ease-in-
out"
>
  Color Transition
</button>

<div
  class="w-24 h-24 bg-red-500 hover:w-48 hover:bg-red-700 transition-all duration-
300 ease-in-out"
>
  Size and Color Transition
</div>

<!-- Animation -->
<div
  class="animate-spin h-12 w-12 border-4 border-blue-500 rounded-full border-t-
transparent"
>
  Spinning Loader
</div>

<div class="animate-pulse bg-blue-500 h-24 w-24">Pulsing Element</div>

<div class="animate-bounce h-12 w-12 bg-purple-500">Bouncing Element</div>

<!-- Custom animation -->
<style>
  @keyframes custom-animation {
    0% {
      transform: scale(1);
```

```
    }
    50% {
      transform: scale(1.2);
    }
    100% {
      transform: scale(1);
    }
  }
  .animate-custom {
    animation: custom-animation 2s infinite;
  }
</style>
<div class="animate-custom h-16 w-16 bg-yellow-500">Custom Animation</div>

<!-- Delay and duration -->
<div class="animate-ping delay-300 duration-1000 h-8 w-8 bg-green-500">
  Delayed Animation
</div>

<!-- Combined animation and transition -->
<div
  class="h-24 w-24 bg-indigo-500 hover:bg-indigo-700 transition duration-300
transform hover:scale-110"
>
  Hover Scale with Color Change
</div>
```

Media Queries and Responsive Design

Media Query Basics

Media queries allow you to apply CSS rules based on device characteristics like screen width, height, or device type.

1. **Basic Media Query Syntax**:

   ```
   @media media-type and (media-feature) {
     /* CSS rules to apply when conditions match */
   }
   ```

2. **Media Types**:
   - `all`: All media types (default)
   - `print`: Printers
   - `screen`: Computer screens, tablets, phones
   - `speech`: Screen readers

3. **Common Media Features**:
   - `width`, `min-width`, `max-width`: Viewport width

- `height`, `min-height`, `max-height`: Viewport height
- `orientation`: Portrait or landscape
- `aspect-ratio`: Width/height ratio
- `resolution`: Pixel density
- `hover`: Whether the device supports hover
- `pointer`: Precision of pointing device (none, coarse, fine)

4. **Basic Examples**:

```
/* Apply when viewport is 768px or wider */
@media (min-width: 768px) {
  .container {
    max-width: 750px;
  }
}

/* Apply when viewport is narrower than 768px */
@media (max-width: 767px) {
  .sidebar {
    display: none;
  }
}
```

/_ Specific range: between 768px and 1024px _/ @media (min-width: 768px) and (max-width: 1024px) { .container { max-width: 960px; } }

/_ Based on orientation _/ @media (orientation: landscape) { .hero-image { height: 70vh; } }

/_ For print _/ @media print { .no-print { display: none; } body { font-size: 12pt; } }

/_ High-resolution screens (Retina displays) _/ @media (min-resolution: 192dpi) { .logo { background-image: url('logo-2x.png'); } }

```
5. **Combining Queries**:
```css
/* AND condition (both must be true) */
@media (min-width: 768px) and (orientation: landscape) {
  /* Apply when width is 768px+ AND in landscape */
}

/* OR condition with comma (either can be true) */
@media (max-width: 767px), (orientation: portrait) {
  /* Apply when width is below 768px OR in portrait */
}

/* NOT condition */
@media not screen and (min-width: 768px) {
  /* Apply to everything except screens 768px+ */
}
```
```

Responsive Design Strategies

1. **Mobile-First Approach**: Start with styles for mobile devices and progressively add styles for larger screens.

```
/* Base styles for all devices (mobile first) */
.container {
  padding: 15px;
}

/* Medium screens (tablets) */
@media (min-width: 768px) {
  .container {
    padding: 30px;
  }
}

/* Large screens (desktops) */
@media (min-width: 1024px) {
  .container {
    padding: 40px;
    max-width: 960px;
    margin: 0 auto;
  }
}
```

2. **Desktop-First Approach**: Start with styles for desktop and add overrides for smaller screens.

```
/* Base styles for desktop */
.container {
  padding: 40px;
  max-width: 960px;
  margin: 0 auto;
}

/* Medium screens (tablets) */
@media (max-width: 1023px) {
  .container {
    padding: 30px;
    max-width: none;
  }
}

/* Small screens (phones) */
@media (max-width: 767px) {
  .container {
    padding: 15px;
```

```
  }
}
```

3. **Common Breakpoints**: These are typical breakpoints, but they should be adapted to your content.

```
/* Small phones: up to 375px */
@media (max-width: 375px) {
  ...;
}

/* Phones: up to 480px */
@media (max-width: 480px) {
  ...;
}

/* Small devices: up to 767px */
@media (max-width: 767px) {
  ...;
}

/* Medium devices: 768px to 1023px */
@media (min-width: 768px) and (max-width: 1023px) {
  ...;
}

/* Large devices: 1024px to 1279px */
@media (min-width: 1024px) and (max-width: 1279px) {
  ...;
}

/* Extra large devices: 1280px+ */
@media (min-width: 1280px) {
  ...;
}
```

4. **Responsive Typography**:

```
/* Base size for mobile */
body {
  font-size: 16px;
}

/* Larger font size for larger screens */
@media (min-width: 768px) {
  body {
    font-size: 18px;
  }
}
```

```
/* Modern fluid typography using clamp() */
body {
  /* Min: 16px, Preferred: 3% of viewport width, Max: 22px */
  font-size: clamp(16px, 3vw, 22px);
}

h1 {
  font-size: clamp(2rem, 5vw, 3.5rem);
}
```

5. **Responsive Images**:

```
/* Make images responsive */
img {
  max-width: 100%;
  height: auto;
}

/* Art direction with picture element */
<picture>
  <source media="(min-width: 1200px)" srcset="large.jpg">
  <source media="(min-width: 768px)" srcset="medium.jpg">
  <img src="small.jpg" alt="Responsive image">
</picture>

/* Responsive background images */
.hero {
  background-image: url('small.jpg');
  background-size: cover;
}

@media (min-width: 768px) {
  .hero {
    background-image: url('medium.jpg');
  }
}

@media (min-width: 1200px) {
  .hero {
    background-image: url('large.jpg');
  }
}
```

6. **Responsive Layouts**:

```
/* Multi-column layout (desktop) to single column (mobile) */
.cards {
  display: grid;
  grid-template-columns: 1fr;
```

```
  gap: 20px;
}

@media (min-width: 768px) {
  .cards {
    grid-template-columns: repeat(2, 1fr);
  }
}

@media (min-width: 1024px) {
  .cards {
    grid-template-columns: repeat(3, 1fr);
  }
}

/* Hide/show elements based on screen size */
.mobile-only {
  display: block;
}

.desktop-only {
  display: none;
}

@media (min-width: 1024px) {
  .mobile-only {
    display: none;
  }

  .desktop-only {
    display: block;
  }
}

/* Responsive navigation */
.nav-links {
  display: none;
  flex-direction: column;
}

.nav-toggle {
  display: block;
}

.nav-links.active {
  display: flex;
}

@media (min-width: 768px) {
  .nav-links {
    display: flex;
    flex-direction: row;
  }
```

```
    .nav-toggle {
      display: none;
    }
  }
```

Viewport Meta Tag

The viewport meta tag is crucial for responsive design on mobile devices. It controls how a page scales and displays on mobile browsers.

```
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
```

This tag tells mobile browsers to:

- Set the width of the viewport to the device width
- Set the initial zoom level to 1 (no zooming)

Additional options:

```
<!-- Prevent user scaling -->
<meta
  name="viewport"
  content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-
scalable=no"
/>

<!-- Set minimum/maximum scale -->
<meta
  name="viewport"
  content="width=device-width, initial-scale=1.0, minimum-scale=1.0, maximum-
scale=2.0"
/>
```

Note: Disabling user scaling can harm accessibility, so it's generally not recommended.

Container Queries (Modern)

While media queries base styling on the viewport, container queries allow you to style elements based on their container's size.

```
/* Define a containment context */
.card-container {
  container-type: inline-size;
  container-name: cards;
}
```

```css
/* Apply styles based on container width */
@container cards (min-width: 400px) {
  .card {
    display: grid;
    grid-template-columns: 1fr 2fr;
    gap: 20px;
  }
}

@container cards (max-width: 399px) {
  .card {
    display: flex;
    flex-direction: column;
  }
}
```

Tailwind responsive design approach:

```html
<!-- Responsive layout -->
<div class="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-4">
  <div class="bg-white p-4 rounded shadow">Item 1</div>
  <div class="bg-white p-4 rounded shadow">Item 2</div>
  <div class="bg-white p-4 rounded shadow">Item 3</div>
</div>

<!-- Responsive typography -->
<h1 class="text-xl sm:text-2xl md:text-3xl lg:text-4xl">Responsive heading</h1>

<!-- Responsive padding -->
<div class="p-4 md:p-6 lg:p-8">Content with responsive padding</div>

<!-- Responsive visibility -->
<div class="hidden md:block">Only visible on medium screens and above</div>

<div class="block md:hidden">Only visible on small screens</div>

<!-- Responsive flexbox direction -->
<div class="flex flex-col md:flex-row">
  <div>Item 1</div>
  <div>Item 2</div>
</div>

<!-- Responsive image size -->
<img src="image.jpg" alt="Responsive image" class="w-full md:w-1/2 lg:w-1/3" />
```

CSS Variables and Custom Properties

Basic Usage

CSS variables (officially called custom properties) allow you to define reusable values that can be referenced throughout your stylesheet.

1. **Defining Variables**:

```
:root {
  /* Define variables at the root level */
  --primary-color: #007bff;
  --secondary-color: #6c757d;
  --font-size-base: 16px;
  --spacing-unit: 8px;
  --border-radius: 4px;
}
```

2. **Using Variables**:

```
.button {
  background-color: var(--primary-color);
  color: white;
  font-size: var(--font-size-base);
  padding: calc(var(--spacing-unit) * 2) calc(var(--spacing-unit) * 3);
  border-radius: var(--border-radius);
}

.button-secondary {
  background-color: var(--secondary-color);
}
```

3. **Fallback Values**:

```
.element {
  /* Use fallback if variable isn't defined */
  color: var(--text-color, #333);
}
```

Scope and Inheritance

Variables are scoped to the element they're declared on and inherit to descendants:

```
:root {
  --main-color: blue;
}

.container {
  --main-color: green; /* Override for this scope */
  --container-padding: 20px;
```

```
  padding: var(--container-padding);
  color: var(--main-color); /* Green */
}

.container .title {
  color: var(--main-color); /* Inherits green */
}

.outside {
  color: var(--main-color); /* Blue from :root */
  padding: var(--container-padding, 10px); /* Uses fallback: 10px */
}
```

Dynamic Variables

1. **Using Media Queries**:

```
:root {
  --header-height: 60px;
  --content-width: 95%;
}

@media (min-width: 768px) {
  :root {
    --header-height: 80px;
    --content-width: 90%;
  }
}

@media (min-width: 1200px) {
  :root {
    --header-height: 100px;
    --content-width: 1140px;
  }
}

header {
  height: var(--header-height);
}

.content {
  width: var(--content-width);
  margin: 0 auto;
}
```

2. **Using JavaScript**:

```
// Get a CSS variable value
const rootStyles = getComputedStyle(document.documentElement);
```

```
const primaryColor = rootStyles.getPropertyValue('--primary-color').trim();
console.log(primaryColor); // "#007bff"

// Set a CSS variable value
document.documentElement.style.setProperty('--primary-color', '#ff0000');
```

3. **Using Class State**:

```
:root {
  --button-bg: #007bff;
  --button-color: white;
}

.theme-dark {
  --button-bg: #6c757d;
  --button-color: #f8f9fa;
}

.button {
  background-color: var(--button-bg);
  color: var(--button-color);
}
```

```
// Toggle dark theme
document.documentElement.classList.toggle('theme-dark');
```

Advanced Use Cases

1. **Theming System**:

```
:root {
  /* Base theme */
  --primary: #3498db;
  --secondary: #2ecc71;
  --text: #333;
  --background: #fff;
}

.dark-theme {
  --primary: #3498db;
  --secondary: #2ecc71;
  --text: #f5f5f5;
  --background: #222;
}

.high-contrast-theme {
  --primary: #ffff00;
```

```
  --secondary: #00ffff;
  --text: #ffffff;
  --background: #000000;
}

body {
  background-color: var(--background);
  color: var(--text);
}

.button-primary {
  background-color: var(--primary);
}

.button-secondary {
  background-color: var(--secondary);
}
```

2. **Creating Design Tokens**:

```
:root {
  /* Colors */
  --color-brand-primary: #3498db;
  --color-brand-secondary: #2ecc71;
  --color-text-primary: #333;
  --color-text-secondary: #666;
  --color-bg-primary: #fff;
  --color-bg-secondary: #f5f5f5;

  /* Typography */
  --font-family-base: 'Roboto', sans-serif;
  --font-family-headings: 'Montserrat', sans-serif;
  --font-size-xs: 0.75rem; /* 12px */
  --font-size-sm: 0.875rem; /* 14px */
  --font-size-md: 1rem; /* 16px */
  --font-size-lg: 1.125rem; /* 18px */
  --font-size-xl: 1.25rem; /* 20px */
  --font-size-2xl: 1.5rem; /* 24px */
  --font-size-3xl: 1.875rem; /* 30px */
  --font-size-4xl: 2.25rem; /* 36px */

  /* Spacing */
  --space-1: 0.25rem; /* 4px */
  --space-2: 0.5rem; /* 8px */
  --space-3: 0.75rem; /* 12px */
  --space-4: 1rem; /* 16px */
  --space-6: 1.5rem; /* 24px */
  --space-8: 2rem; /* 32px */
  --space-12: 3rem; /* 48px */
  --space-16: 4rem; /* 64px */

  /* Borders */
```

```
  --border-radius-sm: 0.125rem; /* 2px */
  --border-radius-md: 0.25rem; /* 4px */
  --border-radius-lg: 0.5rem; /* 8px */
  --border-width-thin: 1px;
  --border-width-thick: 2px;

  /* Shadows */
  --shadow-sm: 0 1px 2px 0 rgba(0, 0, 0, 0.05);
  --shadow-md: 0 4px 6px -1px rgba(0, 0, 0, 0.1);
  --shadow-lg: 0 10px 15px -3px rgba(0, 0, 0, 0.1);

  /* Transitions */
  --transition-fast: 150ms;
  --transition-normal: 300ms;
  --transition-slow: 500ms;
}
```

3. **Calculated Values**:

```
:root {
  --spacing-base: 8px;
  --font-size-base: 16px;
  --line-height-ratio: 1.5;
  --container-max-width: 1200px;
  --sidebar-width: 250px;
}

.container {
  max-width: var(--container-max-width);
  padding: calc(var(--spacing-base) * 2);
}

.content {
  width: calc(100% - var(--sidebar-width) - var(--spacing-base) * 2);
}

p {
  font-size: var(--font-size-base);
  line-height: calc(var(--font-size-base) * var(--line-height-ratio));
  margin-bottom: calc(var(--spacing-base) * 2);
}
```

4. **Component Variables**:

```
.card {
  /* Component-specific variables */
  --card-padding: var(--space-4);
  --card-border-radius: var(--border-radius-md);
  --card-bg: var(--color-bg-primary);
```

```
  /* Apply the variables */
  padding: var(--card-padding);
  border-radius: var(--card-border-radius);
  background-color: var(--card-bg);
}

.card.card-compact {
  --card-padding: var(--space-2);
}

.card.card-large {
  --card-padding: var(--space-6);
}
```

Tailwind approach with CSS variables:

Tailwind uses CSS variables extensively under the hood. You can customize Tailwind by modifying its CSS variables:

```
<style>
  :root {
    --tw-color-primary: #3490dc;
    --tw-color-secondary: #ffed4a;
    --tw-font-family-sans: 'Roboto', sans-serif;
  }
</style>

<!-- Using Tailwind's color classes which use the variables internally -->
<div class="bg-blue-500 text-white p-4">
  This uses Tailwind's predefined colors
</div>

<!-- You can also apply custom CSS variables with Tailwind's utility classes -->
<div
  style="--card-bg: #f8f9fa"
  class="p-4 shadow-md"
  style="background-color: var(--card-bg)"
>
  Custom component with CSS variables
</div>
```

Modern CSS Features

Container Queries

Container queries enable style changes based on the size of a parent container rather than the viewport.

```css
/* Define a containment context */
.card-container {
  container-type: inline-size;
  /* Optional name for the container */
  container-name: card;
}

/* Apply styles based on container width */
@container card (min-width: 400px) {
  .card {
    display: grid;
    grid-template-columns: 1fr 2fr;
  }
}

@container card (max-width: 399px) {
  .card {
    display: flex;
    flex-direction: column;
  }
}

/* You can query unnamed containers too */
@container (min-width: 700px) {
  .sidebar {
    display: block;
  }
}
```

Parent Selector (:has)

The `:has()` pseudo-class allows selecting elements based on their contents.

```css
/* Select paragraphs that contain strong elements */
p:has(strong) {
  background-color: #f8f9fa;
}

/* Style a label when its checkbox is checked */
label:has(input[type='checkbox']:checked) {
  color: green;
}

/* Style a form that has required fields */
form:has(input:required) {
  border: 2px solid red;
}

/* Complex conditions */
.card:has(img):has(.premium-content) {
```

```
  border-color: gold;
}

/* Styling parent based on hover of child */
.parent:has(.child:hover) {
  background-color: #f0f0f0;
}

/* Adjust layout based on number of items */
.grid:has(.item:nth-child(4)) {
  /* Grid has at least 4 items */
  grid-template-columns: repeat(2, 1fr);
}
```

Nesting

Modern CSS now supports nesting similar to preprocessors like Sass.

```
/* Parent selector */
.card {
  background: white;
  border-radius: 8px;

  /* Nested elements */
  & .title {
    font-size: 1.25rem;
    font-weight: bold;
  }

  & .content {
    padding: 16px;

    /* Multiple levels of nesting */
    & p {
      line-height: 1.5;

      /* Targeting specific states */
      &:first-of-type {
        margin-top: 0;
      }
    }
  }

  /* Pseudo-classes */
  &:hover {
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
  }

  /* Complex selectors */
  @media (min-width: 768px) {
    & .content {
```

```
      padding: 24px;
    }
  }
}
```

Color Functions

Modern CSS has enhanced color manipulation capabilities.

```
:root {
  --brand-blue: #0066cc;
}

.element {
  /* Color mixing */
  background-color: color-mix(in srgb, var(--brand-blue), white 30%);

  /* Relative color syntax */
  border-color: color-mix(in srgb, var(--brand-blue), black 20%);

  /* Transparent versions */
  box-shadow: 0 2px 10px color-mix(in srgb, var(--brand-blue), transparent 70%);
}

/* Color contrast */
@media (prefers-color-scheme: dark) {
  :root {
    --text-color: color-contrast(black vs white, #aaa, #bbb);
  }
}
```

Logical Properties

Logical properties and values adapt to the writing mode and direction of text.

```
.box {
  /* Traditional physical properties */
  margin-top: 10px;
  margin-right: 20px;
  margin-bottom: 10px;
  margin-left: 20px;

  /* Logical equivalents */
  margin-block-start: 10px; /* top in horizontal writing mode */
  margin-inline-end: 20px; /* right in LTR, left in RTL */
  margin-block-end: 10px; /* bottom in horizontal writing mode */
  margin-inline-start: 20px; /* left in LTR, right in RTL */

  /* Shorthand for block/inline */
```

```
  margin-block: 10px; /* block-start and block-end */
  margin-inline: 20px; /* inline-start and inline-end */

  /* Other logical properties */
  padding-inline: 16px;
  border-block: 1px solid gray;

  /* Sizing */
  inline-size: 200px; /* width in horizontal writing mode */
  block-size: 100px; /* height in horizontal writing mode */

  /* Position */
  inset-block-start: 20px; /* top in horizontal writing mode */
  inset-inline-end: 20px; /* right in LTR, left in RTL */
}

/* Writing mode examples */
.vertical-text {
  writing-mode: vertical-rl;
  /* Now inline-size controls height and block-size controls width */
}
```

Pseudo-classes and Elements

Modern CSS offers powerful pseudo-classes for targeting elements in different states.

```
/* Form states */
input:placeholder-shown {
  border-color: #ccc;
}

input:focus-visible {
  outline: 2px solid blue;
  outline-offset: 2px;
}

/* Target empty state */
.card:empty {
  display: none;
}

/* Target by index */
li:nth-child(odd) {
  background-color: #f5f5f5;
}

li:nth-child(3n + 1) {
  font-weight: bold;
}

/* Target specific elements */
```

```css
.gallery img:only-child {
  width: 100%;
}

p:first-of-type {
  font-size: 1.2em;
}

/* Target by language */
:lang(fr) {
  quotes: '« ' ' »';
}

/* User interaction */
.tooltip:is(:hover, :focus) {
  opacity: 1;
}

/* Selection styles */
::selection {
  background-color: gold;
  color: black;
}

/* Custom scrollbars */
.custom-scroll::-webkit-scrollbar {
  width: 10px;
}

.custom-scroll::-webkit-scrollbar-track {
  background: #f1f1f1;
}

.custom-scroll::-webkit-scrollbar-thumb {
  background: #888;
  border-radius: 5px;
}
```

Subgrid

Subgrid allows a nested grid to inherit track sizes from its parent grid.

```css
.parent-grid {
  display: grid;
  grid-template-columns: repeat(9, 1fr);
  grid-template-rows: auto auto auto;
  gap: 10px;
}

.child-element {
  grid-column: 2 / 7;
```

```
  grid-row: 2;

  /* Create a subgrid */
  display: grid;
  grid-template-columns: subgrid;
  grid-template-rows: subgrid;
}

/* Items in the child will align to the parent grid's lines */
.child-element .item1 {
  grid-column: 1 / 3; /* Relative to parent grid lines */
}

.child-element .item2 {
  grid-column: 3 / 6; /* Relative to parent grid lines */
}
```

Aspect Ratio

The `aspect-ratio` property maintains a consistent width-to-height ratio.

```
/* 16:9 aspect ratio */
.video-container {
  aspect-ratio: 16 / 9;
  width: 100%;
  background: #000;
}

/* 1:1 square aspect ratio */
.profile-image {
  aspect-ratio: 1;
  width: 100%;
  object-fit: cover;
}

/* Combined with min/max width */
.card {
  aspect-ratio: 4 / 3;
  min-width: 200px;
  max-width: 400px;
}
```

CSS Masking and Clipping

Modern CSS provides powerful ways to mask or clip elements.

```
/* Clip-path for custom shapes */
.hexagon {
  clip-path: polygon(50% 0%, 100% 25%, 100% 75%, 50% 100%, 0% 75%, 0% 25%);
```

```
}

.circle {
  clip-path: circle(50% at center);
}

/* CSS masks with images */
.masked-element {
  mask-image: url('mask.png');
  mask-size: cover;
  mask-repeat: no-repeat;
}

/* Gradient mask */
.fade-out {
  mask-image: linear-gradient(to bottom, black, transparent);
}
```

accent-color

The `accent-color` property allows customizing the color of form controls.

```
/* Set accent color for all form controls */
:root {
  accent-color: #0066cc;
}

/* Specific controls */
input[type='checkbox'] {
  accent-color: #00cc66;
}

input[type='radio'] {
  accent-color: #cc6600;
}

/* Progress bars */
progress {
  accent-color: #cc0066;
}
```

Scroll Snap

Scroll snap creates more controlled scrolling experiences.

```
.gallery {
  display: flex;
  overflow-x: scroll;
  scroll-snap-type: x mandatory;
```

```
  gap: 16px;
}

.gallery img {
  scroll-snap-align: center;
  min-width: 80%;
  height: 300px;
  object-fit: cover;
}

/* Vertical scroll snapping */
.sections {
  height: 100vh;
  overflow-y: scroll;
  scroll-snap-type: y proximity;
}

.section {
  height: 100vh;
  scroll-snap-align: start;
}

/* Control scroll behavior */
.smooth-scroll {
  scroll-behavior: smooth;
}
```

Tailwind doesn't have built-in utilities for some of these modern CSS features yet, but they can be added via plugins or custom CSS when needed.

CSS Architecture Approaches

BEM (Block Element Modifier)

BEM is a methodology for naming CSS classes to create reusable components.

1. **Basic Structure**:
   - **Block**: Standalone entity (`.button`, `.menu`, `.header`)
   - **Element**: Part of a block (`.button__icon`, `.menu__item`)
   - **Modifier**: Different version of a block/element (`.button--large`, `.menu__item--active`)

2. **Example**:

```
<nav class="nav">
  <a class="nav__item nav__item--active" href="/">Home</a>
  <a class="nav__item" href="/about">About</a>
  <a class="nav__item nav__item--disabled" href="/contact">Contact</a>
</nav>
```

```
/* Block */
.nav {
  display: flex;
  background-color: #f5f5f5;
  padding: 1rem;
}

/* Element */
.nav__item {
  color: blue;
  margin-right: 1rem;
  text-decoration: none;
}

/* Modifiers */
.nav__item--active {
  font-weight: bold;
  color: darkblue;
}

.nav__item--disabled {
  color: gray;
  pointer-events: none;
}
```

3. **Benefits**:
    - Clear, consistent naming convention
    - Self-documenting code
    - Reduced style conflicts
    - Easier to understand relationships between HTML and CSS

SMACSS (Scalable and Modular Architecture for CSS)

SMACSS categorizes CSS rules into five types:

1. **Base**: Default styles for HTML elements without classes

```
body {
  font-family: Arial, sans-serif;
}

h1,
h2,
h3 {
  margin-top: 0;
}

a {
```

```
  color: #0066cc;
}
```

2. **Layout**: Major layout components

```
.l-header {
  position: sticky;
  top: 0;
  z-index: 100;
}

.l-sidebar {
  width: 250px;
}

.l-content {
  margin-left: 250px;
}

.l-grid {
  display: grid;
  grid-template-columns: repeat(12, 1fr);
}
```

3. **Module**: Reusable, modular components

```
.card {
  border-radius: 4px;
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}

.btn {
  display: inline-block;
  padding: 0.5em 1em;
  border-radius: 4px;
}
```

4. **State**: Describes how modules or layouts look in a particular state

```
.is-active {
  background-color: #e5e5e5;
}

.is-hidden {
  display: none;
}
```

```
.is-expanded {
  height: auto;
}
```

5. **Theme**: Visual themes (colors, images) separate from layout/structure

```
.theme-dark {
  --bg-color: #222;
  --text-color: #f5f5f5;
}

.theme-light {
  --bg-color: #fff;
  --text-color: #333;
}
```

OOCSS (Object-Oriented CSS)

OOCSS focuses on two main principles:

1. **Separate Structure from Skin**:

```
/* Structure */
.btn {
  display: inline-block;
  padding: 0.5em 1em;
  border-radius: 4px;
}

/* Skin */
.btn-primary {
  background-color: blue;
  color: white;
}

.btn-secondary {
  background-color: gray;
  color: white;
}
```

2. **Separate Container from Content**:

```
/* Content styles independent of location */
.heading {
  font-size: 1.5rem;
  margin-bottom: 0.5em;
}
```

```
/* Not recommended */
.sidebar .heading {
  /* Specific to a container - avoid this */
  font-size: 1.2rem;
}

/* Better approach */
.heading-small {
  font-size: 1.2rem;
}
```

```
<!-- Use in different contexts -->
<main>
  <h2 class="heading">Main Content</h2>
</main>

<aside>
  <h2 class="heading heading-small">Sidebar</h2>
</aside>
```

Atomic CSS (Functional CSS)

Atomic CSS involves using small, single-purpose classes. This approach is similar to what Tailwind CSS implements.

```
/* Display */
.d-block {
  display: block;
}
.d-inline {
  display: inline;
}
.d-flex {
  display: flex;
}

/* Typography */
.text-center {
  text-align: center;
}
.font-bold {
  font-weight: bold;
}
.text-lg {
  font-size: 1.125rem;
}

/* Spacing */
```

```
.m-0 {
  margin: 0;
}
.mt-4 {
  margin-top: 1rem;
}
.p-2 {
  padding: 0.5rem;
}

/* Colors */
.bg-primary {
  background-color: var(--primary-color);
}
.text-white {
  color: white;
}

/* Flex utilities */
.flex-col {
  flex-direction: column;
}
.items-center {
  align-items: center;
}
.justify-between {
  justify-content: space-between;
}
```

```
<div class="d-flex justify-between p-2 bg-primary text-white">
  <h2 class="m-0 text-lg">Header</h2>
  <button class="d-flex items-center">Menu</button>
</div>
```

ITCSS (Inverted Triangle CSS)

ITCSS organizes CSS files in layers from generic to explicit, following specificity:

1. **Settings**: Variables and configurations

   ```
   :root {
     --primary-color: #0066cc;
     --secondary-color: #6c757d;
     --spacing-unit: 8px;
   }
   ```

2. **Tools**: Mixins and functions

```
@mixin heading {
  font-family: var(--font-heading);
  font-weight: bold;
  line-height: 1.2;
}
```

3. **Generic**: Reset and normalize styles

```
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}
```

4. **Elements**: Base styling for HTML elements

```
body {
  font-family: var(--font-body);
  line-height: 1.5;
  color: var(--text-color);
}

a {
  color: var(--link-color);
  text-decoration: none;
}
```

5. **Objects**: Class-based selectors for structure (OOCSS principles)

```
.o-container {
  max-width: 1200px;
  margin-left: auto;
  margin-right: auto;
  padding-left: 20px;
  padding-right: 20px;
}

.o-grid {
  display: grid;
  gap: 20px;
}
```

6. **Components**: UI components with specific styling

```
.c-card {
  border-radius: 4px;
  overflow: hidden;
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}

.c-card__header {
  padding: 1rem;
  border-bottom: 1px solid #eee;
}

.c-button {
  display: inline-block;
  padding: 0.5em 1em;
  border-radius: 4px;
  background-color: var(--primary-color);
  color: white;
}
```

7. **Utilities**: Helper classes with high specificity

```
.u-text-center {
  text-align: center !important;
}

.u-margin-top {
  margin-top: var(--spacing-unit) !important;
}
```

CSS Modules

CSS Modules scope CSS by automatically creating unique class names when compiled.

```
/* button.module.css */
.button {
  display: inline-block;
  padding: 10px 15px;
  background-color: blue;
  color: white;
}

.primary {
  background-color: darkblue;
}
```

```
// In a JavaScript framework (React example)
import styles from './button.module.css';

function Button() {
  return (
    <button className={`${styles.button} ${styles.primary}`}>Click Me</button>
  );
}
```

The compiled HTML might look like:

```
<button class="button_a7c3d button_primary_8d9f0">Click Me</button>
```

CSS-in-JS

CSS-in-JS libraries allow writing CSS directly in JavaScript files.

```
// Styled Components example
import styled from 'styled-components';

const Button = styled.button`
  display: inline-block;
  padding: 10px 15px;
  background-color: ${(props) => (props.primary ? 'darkblue' : 'blue')};
  color: white;
  border-radius: 4px;

  &:hover {
    opacity: 0.9;
  }
`;

function App() {
  return (
    <div>
      <Button>Normal Button</Button>
      <Button primary>Primary Button</Button>
    </div>
  );
}
```

Tailwind's approach to CSS architecture:

Tailwind CSS follows the Atomic/Functional CSS methodology, where utility classes handle most styling needs:

```
<!-- BEM equivalent in Tailwind -->
<nav class="flex bg-gray-100 p-4">
  <a class="text-blue-600 mr-4 font-bold" href="/">Home</a>
  <a class="text-blue-600 mr-4" href="/about">About</a>
  <a class="text-gray-400 cursor-not-allowed" href="/contact">Contact</a>
</nav>

<!-- Component extraction for reuse -->
<button
  class="bg-blue-500 hover:bg-blue-700 text-white font-bold py-2 px-4 rounded"
>
  Button
</button>
```

CSS Preprocessors

CSS preprocessors extend CSS with features like variables, nesting, mixins, and functions. The most popular ones are Sass, Less, and Stylus.

Sass/SCSS

SCSS is a superset of CSS, meaning all valid CSS is also valid SCSS.

1. **Variables**:

```
$primary-color: #3498db;
$secondary-color: #2ecc71;
$font-stack: 'Roboto', sans-serif;
$spacing-unit: 8px;

body {
  font-family: $font-stack;
  color: $primary-color;
}

.button {
  background-color: $primary-color;
  padding: $spacing-unit * 2;
}
```

2. **Nesting**:

```
.card {
  border: 1px solid #ddd;
  border-radius: 4px;

  .card-header {
    padding: 10px;
```

```
    border-bottom: 1px solid #ddd;

    h2 {
      margin: 0;
      font-size: 18px;
    }
  }

  .card-body {
    padding: 15px;
  }

  // Parent selector
  &:hover {
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
  }

  // Special state
  &.is-featured {
    border-color: $primary-color;
  }
}
```

3. **Mixins**:

```
// Define a mixin
@mixin flexCenter {
  display: flex;
  justify-content: center;
  align-items: center;
}

// Mixin with parameters
@mixin button($bg-color, $text-color) {
  background-color: $bg-color;
  color: $text-color;
  padding: 10px 15px;
  border-radius: 4px;
  border: none;
  cursor: pointer;

  &:hover {
    background-color: darken($bg-color, 10%);
  }
}

// Usage
.container {
  @include flexCenter;
  height: 200px;
}
```

```
.button-primary {
  @include button($primary-color, white);
}

.button-secondary {
  @include button($secondary-color, white);
}
```

4. **Functions**:

```
// Custom function
@function calculateWidth($columns, $total: 12) {
  @return percentage($columns / $total);
}

// Usage
.col-4 {
  width: calculateWidth(4); // 33.333%
}

// Built-in functions
.text-light {
  color: lighten($primary-color, 20%);
}

.text-dark {
  color: darken($primary-color, 20%);
}
```

5. **Partials and Import**:

```
// _variables.scss
$primary-color: #3498db;

// _mixins.scss
@mixin flexCenter {
  display: flex;
  justify-content: center;
  align-items: center;
}

// main.scss
@import 'variables';
@import 'mixins';

body {
  color: $primary-color;
}
```

6. **Extend/Inheritance**:

```
%base-button {
  padding: 10px 15px;
  border-radius: 4px;
  border: none;
  cursor: pointer;
}

.button-primary {
  @extend %base-button;
  background-color: $primary-color;
  color: white;
}

.button-secondary {
  @extend %base-button;
  background-color: $secondary-color;
  color: white;
}
```

7. **Control Directives**:

```
// For loop
@for $i from 1 through 4 {
  .col-#{$i} {
    width: calculateWidth($i);
  }
}

// Each loop
$sizes: (
  small: 8px,
  medium: 16px,
  large: 24px,
);

@each $name, $value in $sizes {
  .margin-#{$name} {
    margin: $value;
  }
}

// Conditionals
@mixin text-color($lightness) {
  @if $lightness > 50 {
    color: black;
  } @else {
    color: white;
```

```
    }
  }
```

Less

Less is similar to SCSS but with some syntax differences.

```less
// Variables
@primary-color: #3498db;
@secondary-color: #2ecc71;
@spacing-unit: 8px;

// Nesting
.card {
  border: 1px solid #ddd;

  .card-header {
    padding: 10px;

    h2 {
      margin: 0;
    }
  }

  &:hover {
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
  }
}

// Mixins
.flex-center() {
  display: flex;
  justify-content: center;
  align-items: center;
}

.button(@bg-color, @text-color) {
  background-color: @bg-color;
  color: @text-color;
  padding: 10px 15px;

  &:hover {
    background-color: darken(@bg-color, 10%);
  }
}

// Usage
.container {
  .flex-center();
}
```

```
.button-primary {
  .button(@primary-color, white);
}
```

Stylus

Stylus offers a more minimal syntax with optional colons, semicolons, and braces.

```
// Variables
primary-color = #3498db
secondary-color = #2ecc71

// Nesting
.card
  border 1px solid #ddd

  .card-header
    padding 10px

    h2
      margin 0

  &:hover
    box-shadow 0 2px 5px rgba(0, 0, 0, 0.1)

// Mixins
flex-center()
  display flex
  justify-content center
  align-items center

button(bg-color, text-color)
  background-color bg-color
  color text-color
  padding 10px 15px

  &:hover
    background-color darken(bg-color, 10%)

// Usage
.container
  flex-center()

.button-primary
  button(primary-color, white)
```

PostCSS

PostCSS is a tool for transforming CSS with JavaScript plugins. Unlike traditional preprocessors, PostCSS is modular—you add only the features you need.

```
/* With postcss-preset-env, postcss-nested, etc. */

:root {
  --primary-color: #3498db;
  --secondary-color: #2ecc71;
}

.card {
  border: 1px solid #ddd;

  & .card-header {
    padding: 10px;

    & h2 {
      margin: 0;
    }
  }

  &:hover {
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
  }
}

@custom-media --viewport-medium (width >= 768px);

@media (--viewport-medium) {
  .card {
    display: grid;
    grid-template-columns: 1fr 2fr;
  }
}
```

Tailwind and preprocessors:

Tailwind itself is built with PostCSS. You can use Tailwind with any preprocessor:

```
// SCSS with Tailwind
@import 'tailwindcss/base';
@import 'tailwindcss/components';
@import 'tailwindcss/utilities';

// Custom components with @apply
.btn {
  @apply py-2 px-4 bg-blue-500 text-white rounded;

  &:hover {
    @apply bg-blue-700;
```

```
    }
  }
```

CSS Performance Optimization

Reducing File Size

1. **Minification**:

```
/* Original CSS */
.header {
  background-color: #f5f5f5;
  padding: 20px;
  margin-bottom: 30px;
}

/* Minified CSS */
.header {
  background-color: #f5f5f5;
  padding: 20px;
  margin-bottom: 30px;
}
```

2. **Compression**: Enable GZIP or Brotli compression on your server

3. **CSS Optimization Tools**:

   - PurgeCSS: Removes unused CSS
   - cssnano: Advanced CSS minifier
   - UnCSS: Removes unused CSS

Efficient Selectors

1. **Selector Performance** (from fastest to slowest):

```
/* Fastest: ID selectors */
#header {
  ...;
}

/* Fast: Class selectors */
.header {
  ...;
}

/* Fast: Element selectors */
div {
  ...;
}
```

```
/* Slower: Descendant selectors */
.header h1 {
  ...;
}

/* Slower: Child selectors */
.header > h1 {
  ...;
}

/* Slower: Attribute selectors */
[type='text'] {
  ...;
}

/* Slowest: Pseudo-classes and complex selectors */
.header:hover .dropdown:nth-child(2) a {
  ...;
}
```

2. **Simplify Selectors**:

```
/* Avoid deep nesting */
.header .nav .nav-item .dropdown .menu-item {
  ...;
} /* Bad */

/* Better: Use a dedicated class */
.menu-item {
  ...;
} /* Good */
```

3. **Avoid Universal Selectors in Complex Selectors**:

```
/* Expensive */
.container * {
  ...;
}

/* Better */
.container > * {
  ...;
}

/* Best: Target specific elements */
.container-item {
  ...;
}
```

Reducing Repaints and Reflows

1. **Use Properties That Don't Trigger Layout**:

```
/* Properties that cause reflow/layout when changed */
.bad-animation {
  width: 100px;
  height: 100px;
  animation: grow 2s infinite;
}

@keyframes grow {
  to {
    width: 200px;
    height: 200px;
  }
}

/* Properties that only cause repaint (better) */
.better-animation {
  width: 100px;
  height: 100px;
  animation: fade 2s infinite;
}

@keyframes fade {
  to {
    opacity: 0.5;
  }
}

/* Properties that use GPU acceleration (best) */
.best-animation {
  width: 100px;
  height: 100px;
  animation: move 2s infinite;
}

@keyframes move {
  to {
    transform: translateX(100px);
  }
}
```

2. **Promote to a New Layer**:

```
.hardware-accelerated {
  transform: translateZ(0);
  /* or */
```

```
    will-change: transform;
  }
```

3. **Batch DOM Changes**:

```
// Bad: Multiple style changes causing separate reflows
element.style.width = '100px';
element.style.height = '200px';
element.style.margin = '20px';

// Better: Use a class change to batch updates
element.classList.add('new-dimensions');
```

```
.new-dimensions {
  width: 100px;
  height: 200px;
  margin: 20px;
}
```

Critical CSS

Extract and inline critical above-the-fold CSS for faster initial rendering:

```
<head>
  <!-- Inline critical CSS -->
  <style>
    body {
      font-family: sans-serif;
      margin: 0;
    }
    header {
      background: #f5f5f5;
      padding: 1rem;
    }
    .hero {
      height: 80vh;
      background: url('hero.jpg');
    }
    /* ...other above-the-fold styles */
  </style>

  <!-- Defer non-critical CSS -->
  <link
    rel="preload"
    href="styles.css"
    as="style"
    onload="this.onload=null;this.rel='stylesheet'"
```

```
  />
  <noscript><link rel="stylesheet" href="styles.css" /></noscript>
</head>
```

Loading Strategies

1. **Async CSS Loading**:

```
<link
  rel="preload"
  href="styles.css"
  as="style"
  onload="this.onload=null;this.rel='stylesheet'"
/>
<noscript><link rel="stylesheet" href="styles.css" /></noscript>
```

2. **Split CSS by Media Query**:

```
<link rel="stylesheet" href="base.css" />
<link rel="stylesheet" href="desktop.css" media="(min-width: 1024px)" />
<link rel="stylesheet" href="print.css" media="print" />
```

3. **CSS Import Based on User Interaction**:

```
// Load additional styles when a feature is used
document.querySelector('.video-button').addEventListener('click', () => {
  const link = document.createElement('link');
  link.rel = 'stylesheet';
  link.href = 'video-player.css';
  document.head.appendChild(link);
});
```

CSS Containment

Use the `contain` property to isolate parts of the page:

```
.widget {
  contain: content;
  /* or more specific: */
  contain: layout paint style;
}

.list-item {
```

```
  contain: strict;
}
```

Reduce Unused CSS

1. **Remove Unused Selectors**:

```
/* Before cleanup */
.used-class {
  color: blue;
}
.unused-class {
  color: red;
} /* Remove this */
```

2. **Tools for Finding Unused CSS**:
   - Chrome DevTools Coverage Tab
   - PurgeCSS
   - UnCSS

3. **Reduce Duplicate CSS**:

```
/* Before */
.btn-primary {
  padding: 10px 15px;
  border-radius: 4px;
  background-color: blue;
}

.btn-secondary {
  padding: 10px 15px;
  border-radius: 4px;
  background-color: gray;
}

/* After */
.btn {
  padding: 10px 15px;
  border-radius: 4px;
}

.btn-primary {
  background-color: blue;
}

.btn-secondary {
  background-color: gray;
}
```

Tailwind's approach to performance:

Tailwind uses PurgeCSS and JIT compilation to dramatically reduce CSS file size:

```
// tailwind.config.js
module.exports = {
  purge: ['./src/**/*.html', './src/**/*.js'],
  mode: 'jit',
  // rest of your config
};
```

Browser Compatibility

Understanding Browser Compatibility

Browser compatibility refers to how CSS features are supported across different browsers and versions. Key concepts:

1. **Progressive Enhancement**: Start with basic functionality for all browsers, then enhance for modern browsers
2. **Graceful Degradation**: Build for modern browsers, then ensure a reasonable experience in older ones
3. **Feature Detection**: Test if a feature exists before using it

Checking Browser Support

1. **Resources**:
   - Can I use...: Detailed browser support tables
   - MDN Web Docs: Browser compatibility info for each feature
   - Browser support data: Raw compatibility data

2. **Feature Detection with CSS**:

```
/* @supports rule checks if a feature is supported */
@supports (display: grid) {
  .container {
    display: grid;
    grid-template-columns: repeat(3, 1fr);
  }
}

/* Fallback for browsers without grid support */
.container {
  display: flex;
  flex-wrap: wrap;
}
```

```
.container > * {
  flex: 0 0 calc(33.333% - 20px);
  margin: 10px;
}

/* Multiple conditions */
@supports (display: grid) and (grid-template-areas: 'a b') {
  /* Grid with template areas */
}

/* Negated conditions */
@supports not (display: grid) {
  /* Styles for browsers without grid support */
}
```

3. **Feature Detection with JavaScript**:

```
if ('CSS' in window && 'supports' in window.CSS) {
  if (window.CSS.supports('display', 'grid')) {
    document.documentElement.classList.add('grid-supported');
  } else {
    document.documentElement.classList.add('grid-not-supported');
  }
}
```

Handling Compatibility Issues

1. **Vendor Prefixes**:

```
.box {
  -webkit-box-shadow: 0 2px 4px rgba(0, 0, 0, 0.2);
  -moz-box-shadow: 0 2px 4px rgba(0, 0, 0, 0.2);
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.2);
}
```

2. **Autoprefixer**: Automatically adds vendor prefixes

```
/* Write standard CSS */
.box {
  border-radius: 4px;
  user-select: none;
}

/* Autoprefixer outputs: */
.box {
  border-radius: 4px;
  -webkit-user-select: none;
```

```
  -moz-user-select: none;
  -ms-user-select: none;
  user-select: none;
}
```

3. **Feature Queries**:

```
/* Fallback for all browsers */
.item {
  display: inline-block;
  width: 30%;
  margin: 0 1.5%;
}

/* Enhanced layout for grid-supporting browsers */
@supports (display: grid) {
  .container {
    display: grid;
    grid-template-columns: repeat(3, 1fr);
    grid-gap: 20px;
  }

  .item {
    width: auto;
    margin: 0;
  }
}
```

4. **Browser-Specific Hacks (use sparingly)**:

```
/* Target Internet Explorer 10-11 */
@media all and (-ms-high-contrast: none), (-ms-high-contrast: active) {
  .element {
    display: block;
  }
}

/* Target Safari */
@media not all and (min-resolution: 0.001dpcm) {
  @supports (-webkit-appearance: none) {
    .safari-only {
      color: red;
    }
  }
}
```

5. **Using Polyfills**:

```
<!-- Load polyfills conditionally (example for CSS Variables) -->
<script>
  if (
    !window.CSS ||
    !window.CSS.supports ||
    !window.CSS.supports('--a', '0')
  ) {
    const script = document.createElement('script');
    script.src = 'css-vars-ponyfill.js';
    document.head.appendChild(script);
  }
</script>
```

Normalize.css and Resets

1. **Normalize.css**: Makes browsers render elements more consistently

```
<link
  rel="stylesheet"

href="https://cdnjs.cloudflare.com/ajax/libs/normalize/8.0.1/normalize.min.c
ss"
/>
```

2. **CSS Reset**: Removes browser default styles

```
/* Simple reset */
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

/* More comprehensive reset */
html, body, div, span, h1, h2, h3, h4, h5, h6, p, /* ...etc */ {
  margin: 0;
  padding: 0;
  border: 0;
  font-size: 100%;
  font: inherit;
  vertical-align: baseline;
}
```

Safe Cross-Browser Techniques

1. **Flexbox Fallbacks for Grid**:

```
/* Flexbox for all browsers */
.container {
  display: flex;
  flex-wrap: wrap;
}

.item {
  flex: 0 0 calc(33.333% - 20px);
  margin: 10px;
}

/* Grid for supporting browsers */
@supports (display: grid) {
  .container {
    display: grid;
    grid-template-columns: repeat(3, 1fr);
    grid-gap: 20px;
  }

  .item {
    flex: none;
    margin: 0;
  }
}
```

2. **Fallbacks for CSS Variables**:

```
.button {
  /* Fallback for browsers without CSS variables */
  background-color: #0066cc;
  /* Modern browsers with CSS variables */
  background-color: var(--primary-color, #0066cc);
}
```

3. **Multiple Background Images**:

```
.element {
  /* Fallback for browsers that don't support multiple backgrounds */
  background: url('primary.jpg') center center no-repeat;
  /* Multiple backgrounds for modern browsers */
  background: url('overlay.png') center center no-repeat, url('primary.jpg')
      center center no-repeat;
}
```

4. **Feature Detection for Modern Features**:

```
/* Fallback for all browsers */
.grid {
  overflow: hidden;
}

.grid-item {
  float: left;
  width: 50%;
  padding: 10px;
}

/* Clear fix for float-based layout */
.grid::after {
  content: '';
  display: table;
  clear: both;
}

/* Modern Grid layout */
@supports (display: grid) {
  .grid {
    display: grid;
    grid-template-columns: repeat(auto-fill, minmax(250px, 1fr));
    grid-gap: 20px;
    overflow: visible;
  }

  .grid-item {
    float: none;
    width: auto;
    padding: 0;
  }

  .grid::after {
    content: none;
  }
}
```

Tailwind's approach to browser compatibility:

Tailwind focuses on modern browsers but maintains good compatibility:

```
// tailwind.config.js with browser targets
module.exports = {
  // ...other config
  variants: {
    extend: {
      // Enable responsive variants for backdrop blur
      backdropBlur: ['responsive'],
    },
  },
```

```
  // Target specific browsers for prefixing
  future: {
    purgeLayersByDefault: true,
  },
};
```

Tailwind CSS Section

Philosophy of Utility-First CSS

Utility-first CSS is a styling approach where you apply small, single-purpose classes directly in your HTML rather than writing custom CSS for each component.

Core Principles

1. **Composition Over Inheritance**:
    - Build complex designs by combining simple utility classes
    - Instead of creating and inheriting from larger, more abstract classes
2. **HTML-Centric Development**:
    - Keep styles together with the HTML they affect
    - Avoid context switching between HTML and CSS files
3. **Low-Level Building Blocks**:
    - Use atomic classes that do one thing well
    - Combine them to create any design
4. **Consistency by Default**:
    - Pre-defined values create a consistent design system
    - Constraints help maintain visual consistency
5. **Maximum Reusability**:
    - Small utilities can be reused across the entire project
    - Less duplication of CSS rules

Traditional CSS vs. Utility-First Approach

Traditional CSS:

```
<button class="primary-button">Submit</button>
```

```
.primary-button {
  padding: 8px 16px;
```

```
  background-color: #3490dc;
  color: white;
  border-radius: 4px;
  font-weight: 600;
  font-size: 14px;
}

.primary-button:hover {
  background-color: #2779bd;
}
```

Utility-First (Tailwind) Approach:

```
<button
  class="px-4 py-2 bg-blue-500 hover:bg-blue-700 text-white font-semibold text-sm
rounded"
>
  Submit
</button>
```

Advantages of Utility-First CSS

1. **Elimination of Unused CSS**:
    - Only include utilities actually used in your HTML
    - Results in smaller file sizes in production
2. **No More Naming Things**:
    - Avoid semantic naming challenges
    - No more `.card-header-title-container` style names
    - Less mental overhead
3. **Faster Development**:
    - Make changes directly in HTML without updating CSS files
    - Immediate feedback loop
4. **Safe Changes**:
    - Changes are localized to the element you're working with
    - No unexpected side effects elsewhere
5. **Responsive Design**:
    - Add responsive prefixes like `md:` and `lg:` directly in HTML
    - See the responsive behavior right where you're working
6. **Hover, Focus, and Other States**:

- Add state variants like `hover:` and `focus:` directly in HTML
- Clear visualization of different states

Addressing Common Concerns

1. **"But the HTML gets so cluttered!"**:
   - Component extraction solves this
   - Modern frameworks allow for reusable components

```
// React component example
function Button({ children }) {
  return (
    <button className="px-4 py-2 bg-blue-500 hover:bg-blue-700 text-white
font-semibold rounded">
      {children}
    </button>
  );
}
```

2. **"It's not semantic!"**:
   - Utility classes describe presentation, not semantics
   - HTML elements and ARIA provide semantics
   - Separation of concerns doesn't necessarily mean separation of files
3. **"We lose the ability to make global changes!"**:
   - Tailwind's theming system allows global changes through the configuration
   - Component extraction provides a single place to update styles
4. **"We're basically writing inline styles!"**:
   - Unlike inline styles, utility classes are:
     - Constrained to a design system
     - Responsive
     - Support hover/focus/etc. states
     - Can be purged in production
5. **"What about reusability?"**:
   - Extract components for reusability
   - Use design systems for consistency
   - Tailwind's `@apply` directive for repeating patterns

When Utility-First Makes Sense

Utility-first CSS (like Tailwind) is particularly well-suited for:

1. **Component-based frameworks** (React, Vue, Angular)

2. **Projects with design systems** that benefit from constraints
3. **Teams that value rapid iteration** and prototyping
4. **Projects that require responsiveness** across many breakpoints
5. **Applications with complex UI states** (hover, focus, active, disabled)

When to Consider Alternatives

Traditional CSS approaches might be better for:

1. **Marketing sites or blogs** that need minimal styling
2. **Teams with strong CSS expertise** and established methodology
3. **Projects with very specific design requirements** that don't fit a utility framework
4. **Highly standardized interfaces** with little variation

Installation and Configuration

Basic Installation Options

1. **CDN (simplest, but limited)**:

```
<link

href="https://cdn.jsdelivr.net/npm/tailwindcss@2.2.19/dist/tailwind.min.css"
  rel="stylesheet"
/>
```

2. **PostCSS with npm/yarn (recommended)**:

```
# Install Tailwind CSS and its dependencies
npm install -D tailwindcss postcss autoprefixer

# Initialize configuration files
npx tailwindcss init -p
```

3. **Framework-specific installation**:

   For React (Create React App):

```
npm install -D tailwindcss postcss autoprefixer
npx tailwindcss init -p
```

   For Vue (Vue CLI):

```
vue add tailwind
```

For Next.js:

```
npm install -D tailwindcss postcss autoprefixer
npx tailwindcss init -p
```

Configuration Setup

1. **Create CSS file**:

```
/* src/styles/main.css */
@tailwind base;
@tailwind components;
@tailwind utilities;
```

2. **Configure content paths** (tailwind.config.js):

```
module.exports = {
  content: ['./src/**/*.{html,js,jsx,ts,tsx,vue}'],
  theme: {
    extend: {},
  },
  plugins: [],
};
```

3. **Include in build process**:

   For PostCSS projects (postcss.config.js):

```
module.exports = {
  plugins: {
    tailwindcss: {},
    autoprefixer: {},
  },
};
```

   For webpack:

```
// webpack.config.js
module.exports = {
  // ...
  module: {
    rules: [
      {
```

```
        test: /\.css$/,
        use: ['style-loader', 'css-loader', 'postcss-loader'],
      },
    ],
  },
};
```

Development Workflow

1. **Watch mode**:

```
npx tailwindcss -i ./src/styles/main.css -o ./dist/output.css --watch
```

2. **Development server**:
   - Most frameworks have hot reloading (Next.js, Create React App, Vue CLI)
   - For basic projects, use tools like `live-server`

3. **Editor setup**:

   VS Code extensions:
   - Tailwind CSS IntelliSense
   - PostCSS Language Support

   Snippets:

```
{
  "twbutton": {
    "prefix": "twbutton",
    "body": "class=\"px-4 py-2 bg-blue-500 text-white rounded hover:bg-blue-
700\"",
    "description": "Tailwind button style"
  }
}
```

Production Optimization

1. **Configure content purging**:

```
// tailwind.config.js
module.exports = {
  content: ['./src/**/*.{html,js,jsx,ts,tsx,vue}'],
  theme: {
    extend: {},
  },
```

```
  plugins: [],
};
```

2. **Minification**:

```
# Using PostCSS CLI
npx tailwindcss -i ./src/styles/main.css -o ./dist/output.css --minify
```

With cssnano:

```
// postcss.config.js
module.exports = {
  plugins: {
    tailwindcss: {},
    autoprefixer: {},
    ...(process.env.NODE_ENV === 'production' ? { cssnano: {} } : {}),
  },
};
```

3. **Just-in-Time (JIT) mode**:

```
// tailwind.config.js
module.exports = {
  mode: 'jit', // In Tailwind 3.0+, JIT is on by default
  // rest of your config
};
```

Integrating with CSS Preprocessors

1. **With Sass**:

```
// src/styles/main.scss
@import 'tailwindcss/base';
@import 'tailwindcss/components';
@import 'tailwindcss/utilities';

// Your custom SCSS here
.custom-component {
  @apply rounded shadow bg-white p-4;

  h2 {
    @apply text-xl font-bold mb-2;
  }
}
```

2. **With Less**:

```
// src/styles/main.less
@import 'tailwindcss/base';
@import 'tailwindcss/components';
@import 'tailwindcss/utilities';

// Your custom Less here
```

3. **With Stylus**:

```
// src/styles/main.styl
@import 'tailwindcss/base'
@import 'tailwindcss/components'
@import 'tailwindcss/utilities'

// Your custom Stylus here
```

Multi-Environment Configuration

```
// tailwind.config.js
module.exports = {
  content: ['./src/**/*.{html,js,jsx,ts,tsx,vue}'],
  theme: {
    extend: {
      colors: {
        // Different themes for different environments
        primary:
          process.env.NODE_ENV === 'production'
            ? '#0f172a' // Production theme
            : '#0ea5e9', // Development theme
      },
    },
  },
  plugins: [],

  // Feature flags
  future: {
    removeDeprecatedGapUtilities: true,
    purgeLayersByDefault: true,
  },
};
```

Core Utility Classes

Layout Utilities

1. **Container**:

```
<div class="container mx-auto px-4">
  <!-- Content centered with padding -->
</div>
```

2. **Display**:

```
<div class="block">Block element</div>
<span class="inline">Inline element</span>
<div class="inline-block">Inline-block element</div>
<div class="hidden">Hidden element</div>
<div class="flex">Flex container</div>
<div class="inline-flex">Inline flex container</div>
<div class="grid">Grid container</div>
<div class="table">Table element</div>
```

3. **Floats**:

```
<div class="float-left">Float left</div>
<div class="float-right">Float right</div>
<div class="float-none">No float</div>
<div class="clearfix">Clearfix</div>
```

4. **Clear**:

```
<div class="clear-left">Clear left floats</div>
<div class="clear-right">Clear right floats</div>
<div class="clear-both">Clear both floats</div>
<div class="clear-none">Don't clear floats</div>
```

5. **Object Fit**:

```
<img class="object-contain" src="..." alt="..." />
<img class="object-cover" src="..." alt="..." />
<img class="object-fill" src="..." alt="..." />
<img class="object-none" src="..." alt="..." />
<img class="object-scale-down" src="..." alt="..." />
```

6. **Object Position**:

```
<img class="object-center" src="..." alt="..." />
<img class="object-top" src="..." alt="..." />
<img class="object-right" src="..." alt="..." />
<img class="object-bottom" src="..." alt="..." />
<img class="object-left" src="..." alt="..." />
<img class="object-left-top" src="..." alt="..." />
<img class="object-right-top" src="..." alt="..." />
<img class="object-left-bottom" src="..." alt="..." />
<img class="object-right-bottom" src="..." alt="..." />
```

7. **Overflow**:

```
<div class="overflow-auto">Auto overflow</div>
<div class="overflow-hidden">Hidden overflow</div>
<div class="overflow-visible">Visible overflow</div>
<div class="overflow-scroll">Scrollable overflow</div>
<div class="overflow-x-auto">Horizontal auto overflow</div>
<div class="overflow-y-scroll">Vertical scrollable overflow</div>
```

8. **Overscroll Behavior**:

```
<div class="overscroll-auto">Auto overscroll</div>
<div class="overscroll-contain">Contained overscroll</div>
<div class="overscroll-none">No overscroll</div>
<div class="overscroll-y-auto">Vertical auto overscroll</div>
<div class="overscroll-x-none">No horizontal overscroll</div>
```

9. **Position**:

```
<div class="static">Static positioning</div>
<div class="relative">Relative positioning</div>
<div class="absolute">Absolute positioning</div>
<div class="fixed">Fixed positioning</div>
<div class="sticky">Sticky positioning</div>
```

10. **Top / Right / Bottom / Left**:

```
<div class="top-0">Top 0</div>
<div class="right-0">Right 0</div>
<div class="bottom-0">Bottom 0</div>
<div class="left-0">Left 0</div>
<div class="inset-0">All sides 0</div>
<div class="inset-x-0">Left and right 0</div>
<div class="inset-y-0">Top and bottom 0</div>
```

11. **Visibility**:

```
<div class="visible">Visible element</div>
<div class="invisible">Invisible element</div>
```

12. **Z-Index**:

```
<div class="z-0">z-index: 0</div>
<div class="z-10">z-index: 10</div>
<div class="z-20">z-index: 20</div>
<div class="z-30">z-index: 30</div>
<div class="z-40">z-index: 40</div>
<div class="z-50">z-index: 50</div>
<div class="z-auto">z-index: auto</div>
```

Flexbox Utilities

1. **Flex Direction**:

```
<div class="flex flex-row">Row (default)</div>
<div class="flex flex-row-reverse">Row reversed</div>
<div class="flex flex-col">Column</div>
<div class="flex flex-col-reverse">Column reversed</div>
```

2. **Flex Wrap**:

```
<div class="flex flex-nowrap">No wrapping</div>
<div class="flex flex-wrap">Wrap items</div>
<div class="flex flex-wrap-reverse">Reverse wrap</div>
```

3. **Flex**:

```
<div class="flex flex-1">Flex grow and shrink, flex-basis: 0%</div>
<div class="flex flex-auto">Flex grow and shrink, flex-basis: auto</div>
<div class="flex flex-initial">Flex shrink, don't grow</div>
<div class="flex flex-none">Don't grow or shrink</div>
```

4. **Flex Grow**:

```

```

```
<div class="flex-grow-0">Don't grow</div>
<div class="flex-grow">Grow</div>
```

5. **Flex Shrink**:

```
<div class="flex-shrink-0">Don't shrink</div>
<div class="flex-shrink">Shrink</div>
```

6. **Order**:

```
<div class="order-first">Order -9999</div>
<div class="order-last">Order 9999</div>
<div class="order-none">Order 0</div>
<div class="order-1">Order 1</div>
<div class="order-2">Order 2</div>
<div class="order-3">Order 3</div>
<div class="order-4">Order 4</div>
<div class="order-5">Order 5</div>
<div class="order-6">Order 6</div>
<div class="order-7">Order 7</div>
<div class="order-8">Order 8</div>
<div class="order-9">Order 9</div>
<div class="order-10">Order 10</div>
<div class="order-11">Order 11</div>
<div class="order-12">Order 12</div>
```

7. **Justify Content**:

```
<div class="flex justify-start">Justify start</div>
<div class="flex justify-end">Justify end</div>
<div class="flex justify-center">Justify center</div>
<div class="flex justify-between">Justify between</div>
<div class="flex justify-around">Justify around</div>
<div class="flex justify-evenly">Justify evenly</div>
```

8. **Justify Items**:

```
<div class="grid justify-items-start">Justify items start</div>
<div class="grid justify-items-end">Justify items end</div>
<div class="grid justify-items-center">Justify items center</div>
<div class="grid justify-items-stretch">Justify items stretch</div>
```

9. **Justify Self**:

```
<div class="justify-self-auto">Justify self auto</div>
<div class="justify-self-start">Justify self start</div>
<div class="justify-self-end">Justify self end</div>
<div class="justify-self-center">Justify self center</div>
<div class="justify-self-stretch">Justify self stretch</div>
```

10. **Align Content**:

```
<div class="flex flex-wrap content-center">Align content center</div>
<div class="flex flex-wrap content-start">Align content start</div>
<div class="flex flex-wrap content-end">Align content end</div>
<div class="flex flex-wrap content-between">Align content between</div>
<div class="flex flex-wrap content-around">Align content around</div>
<div class="flex flex-wrap content-evenly">Align content evenly</div>
```

11. **Align Items**:

```
<div class="flex items-start">Align items start</div>
<div class="flex items-end">Align items end</div>
<div class="flex items-center">Align items center</div>
<div class="flex items-baseline">Align items baseline</div>
<div class="flex items-stretch">Align items stretch</div>
```

12. **Align Self**:

```
<div class="self-auto">Align self auto</div>
<div class="self-start">Align self start</div>
<div class="self-end">Align self end</div>
<div class="self-center">Align self center</div>
<div class="self-stretch">Align self stretch</div>
<div class="self-baseline">Align self baseline</div>
```

Grid Utilities

1. **Grid Template Columns**:

```
<div class="grid grid-cols-1">1 column</div>
<div class="grid grid-cols-2">2 columns</div>
<div class="grid grid-cols-3">3 columns</div>
<div class="grid grid-cols-4">4 columns</div>
<div class="grid grid-cols-5">5 columns</div>
<div class="grid grid-cols-6">6 columns</div>
<div class="grid grid-cols-7">7 columns</div>
```

```
<div class="grid grid-cols-8">8 columns</div>
<div class="grid grid-cols-9">9 columns</div>
<div class="grid grid-cols-10">10 columns</div>
<div class="grid grid-cols-11">11 columns</div>
<div class="grid grid-cols-12">12 columns</div>
<div class="grid grid-cols-none">No columns</div>
```

2. **Grid Column Start / End**:

```
<div class="col-auto">Auto column</div>
<div class="col-span-1">Span 1 column</div>
<div class="col-span-2">Span 2 columns</div>
<div class="col-span-3">Span 3 columns</div>
<div class="col-span-4">Span 4 columns</div>
<div class="col-span-5">Span 5 columns</div>
<div class="col-span-6">Span 6 columns</div>
<div class="col-span-7">Span 7 columns</div>
<div class="col-span-8">Span 8 columns</div>
<div class="col-span-9">Span 9 columns</div>
<div class="col-span-10">Span 10 columns</div>
<div class="col-span-11">Span 11 columns</div>
<div class="col-span-12">Span 12 columns</div>
<div class="col-span-full">Span full</div>
<div class="col-start-1">Start at column 1</div>
<div class="col-start-2">Start at column 2</div>
<!-- ... and so on for col-start-3 through col-start-13 -->
<div class="col-end-1">End at column 1</div>
<div class="col-end-2">End at column 2</div>
<!-- ... and so on for col-end-3 through col-end-13 -->
```

3. **Grid Template Rows**:

```
<div class="grid grid-rows-1">1 row</div>
<div class="grid grid-rows-2">2 rows</div>
<div class="grid grid-rows-3">3 rows</div>
<div class="grid grid-rows-4">4 rows</div>
<div class="grid grid-rows-5">5 rows</div>
<div class="grid grid-rows-6">6 rows</div>
<div class="grid grid-rows-none">No explicit rows</div>
```

4. **Grid Row Start / End**:

```
<div class="row-auto">Auto row</div>
<div class="row-span-1">Span 1 row</div>
<div class="row-span-2">Span 2 rows</div>
<div class="row-span-3">Span 3 rows</div>
<div class="row-span-4">Span 4 rows</div>
```

```
<div class="row-span-5">Span 5 rows</div>
<div class="row-span-6">Span 6 rows</div>
<div class="row-span-full">Span full</div>
<div class="row-start-1">Start at row 1</div>
<div class="row-start-2">Start at row 2</div>
<!-- ... and so on for row-start-3 through row-start-7 -->
<div class="row-end-1">End at row 1</div>
<div class="row-end-2">End at row 2</div>
<!-- ... and so on for row-end-3 through row-end-7 -->
```

5. **Grid Auto Flow**:

```
<div class="grid grid-flow-row">Grid flow row</div>
<div class="grid grid-flow-col">Grid flow column</div>
<div class="grid grid-flow-row-dense">Grid flow row dense</div>
<div class="grid grid-flow-col-dense">Grid flow column dense</div>
```

6. **Grid Auto Columns**:

```
<div class="auto-cols-auto">Auto columns auto</div>
<div class="auto-cols-min">Auto columns min</div>
<div class="auto-cols-max">Auto columns max</div>
<div class="auto-cols-fr">Auto columns fractional</div>
```

7. **Grid Auto Rows**:

```
<div class="auto-rows-auto">Auto rows auto</div>
<div class="auto-rows-min">Auto rows min</div>
<div class="auto-rows-max">Auto rows max</div>
<div class="auto-rows-fr">Auto rows fractional</div>
```

8. **Gap**:

```
<div class="gap-0">No gap</div>
<div class="gap-1">0.25rem gap</div>
<div class="gap-2">0.5rem gap</div>
<div class="gap-3">0.75rem gap</div>
<div class="gap-4">1rem gap</div>
<div class="gap-5">1.25rem gap</div>
<div class="gap-6">1.5rem gap</div>
<div class="gap-8">2rem gap</div>
<div class="gap-10">2.5rem gap</div>
<div class="gap-12">3rem gap</div>
<div class="gap-16">4rem gap</div>
<div class="gap-20">5rem gap</div>
```

```
<div class="gap-24">6rem gap</div>
<div class="gap-32">8rem gap</div>
<div class="gap-40">10rem gap</div>
<div class="gap-48">12rem gap</div>
<div class="gap-56">14rem gap</div>
<div class="gap-64">16rem gap</div>
<div class="gap-px">1px gap</div>
<div class="gap-x-0">No horizontal gap</div>
<div class="gap-x-4">1rem horizontal gap</div>
<div class="gap-y-0">No vertical gap</div>
<div class="gap-y-4">1rem vertical gap</div>
```

Spacing Utilities

1. **Padding**:

```
<div class="p-0">No padding</div>
<div class="p-1">0.25rem padding</div>
<div class="p-2">0.5rem padding</div>
<div class="p-3">0.75rem padding</div>
<div class="p-4">1rem padding</div>
<div class="p-5">1.25rem padding</div>
<div class="p-6">1.5rem padding</div>
<div class="p-8">2rem padding</div>
<div class="p-10">2.5rem padding</div>
<div class="p-12">3rem padding</div>
<div class="p-16">4rem padding</div>
<div class="p-20">5rem padding</div>
<div class="p-24">6rem padding</div>
<div class="p-32">8rem padding</div>
<div class="p-40">10rem padding</div>
<div class="p-48">12rem padding</div>
<div class="p-56">14rem padding</div>
<div class="p-64">16rem padding</div>
<div class="p-px">1px padding</div>

<!-- Directional padding -->
<div class="pt-4">Padding top</div>
<div class="pr-4">Padding right</div>
<div class="pb-4">Padding bottom</div>
<div class="pl-4">Padding left</div>
<div class="py-4">Padding top & bottom</div>
<div class="px-4">Padding left & right</div>
```

2. **Margin**:

```
<div class="m-0">No margin</div>
<div class="m-1">0.25rem margin</div>
<div class="m-2">0.5rem margin</div>
```

```
<div class="m-3">0.75rem margin</div>
<div class="m-4">1rem margin</div>
<div class="m-5">1.25rem margin</div>
<div class="m-6">1.5rem margin</div>
<div class="m-8">2rem margin</div>
<div class="m-10">2.5rem margin</div>
<div class="m-12">3rem margin</div>
<div class="m-16">4rem margin</div>
<div class="m-20">5rem margin</div>
<div class="m-24">6rem margin</div>
<div class="m-32">8rem margin</div>
<div class="m-40">10rem margin</div>
<div class="m-48">12rem margin</div>
<div class="m-56">14rem margin</div>
<div class="m-64">16rem margin</div>
<div class="m-auto">Auto margin</div>
<div class="m-px">1px margin</div>

<!-- Directional margin -->
<div class="mt-4">Margin top</div>
<div class="mr-4">Margin right</div>
<div class="mb-4">Margin bottom</div>
<div class="ml-4">Margin left</div>
<div class="my-4">Margin top & bottom</div>
<div class="mx-4">Margin left & right</div>
<div class="mx-auto">Horizontal centering</div>

<!-- Negative margin -->
<div class="-m-4">-1rem margin</div>
<div class="-mt-4">-1rem margin top</div>
<div class="-mr-4">-1rem margin right</div>
<div class="-mb-4">-1rem margin bottom</div>
<div class="-ml-4">-1rem margin left</div>
<div class="-my-4">-1rem margin top & bottom</div>
<div class="-mx-4">-1rem margin left & right</div>
```

3. **Space Between**:

```
<!-- Horizontal spacing -->
<div class="space-x-0">
  <div>Item 1</div>
  <div>Item 2</div>
  <div>Item 3</div>
</div>

<div class="space-x-4">
  <div>Item 1</div>
  <div>Item 2</div>
  <div>Item 3</div>
</div>

<!-- Reverse horizontal spacing -->
```

```
<div class="space-x-reverse space-x-4">
  <div>Item 1</div>
  <div>Item 2</div>
  <div>Item 3</div>
</div>

<!-- Vertical spacing -->
<div class="space-y-0">
  <div>Item 1</div>
  <div>Item 2</div>
  <div>Item 3</div>
</div>

<div class="space-y-4">
  <div>Item 1</div>
  <div>Item 2</div>
  <div>Item 3</div>
</div>

<!-- Reverse vertical spacing -->
<div class="space-y-reverse space-y-4">
  <div>Item 1</div>
  <div>Item 2</div>
  <div>Item 3</div>
</div>
```

Sizing Utilities

1. **Width**:

```
<div class="w-0">0 width</div>
<div class="w-1">0.25rem width</div>
<div class="w-2">0.5rem width</div>
<div class="w-3">0.75rem width</div>
<div class="w-4">1rem width</div>
<div class="w-5">1.25rem width</div>
<div class="w-6">1.5rem width</div>
<div class="w-8">2rem width</div>
<div class="w-10">2.5rem width</div>
<div class="w-12">3rem width</div>
<div class="w-16">4rem width</div>
<div class="w-20">5rem width</div>
<div class="w-24">6rem width</div>
<div class="w-32">8rem width</div>
<div class="w-40">10rem width</div>
<div class="w-48">12rem width</div>
<div class="w-56">14rem width</div>
<div class="w-64">16rem width</div>
<div class="w-auto">Auto width</div>
<div class="w-px">1px width</div>
```

```
<!-- Percentage widths -->
<div class="w-1/2">50% width</div>
<div class="w-1/3">33.33% width</div>
<div class="w-2/3">66.66% width</div>
<div class="w-1/4">25% width</div>
<div class="w-3/4">75% width</div>
<div class="w-1/5">20% width</div>
<div class="w-2/5">40% width</div>
<div class="w-3/5">60% width</div>
<div class="w-4/5">80% width</div>
<div class="w-1/6">16.66% width</div>
<div class="w-5/6">83.33% width</div>
<div class="w-1/12">8.33% width</div>
<div class="w-5/12">41.66% width</div>
<div class="w-7/12">58.33% width</div>
<div class="w-11/12">91.66% width</div>
<div class="w-full">100% width</div>
<div class="w-screen">100vw width</div>
<div class="w-min">Min-content width</div>
<div class="w-max">Max-content width</div>
<div class="w-fit">Fit-content width</div>
```

2. **Min-Width**:

```
<div class="min-w-0">0 min-width</div>
<div class="min-w-full">100% min-width</div>
<div class="min-w-min">Min-content min-width</div>
<div class="min-w-max">Max-content min-width</div>
<div class="min-w-fit">Fit-content min-width</div>
```

3. **Max-Width**:

```
<div class="max-w-none">No max-width</div>
<div class="max-w-xs">20rem max-width</div>
<div class="max-w-sm">24rem max-width</div>
<div class="max-w-md">28rem max-width</div>
<div class="max-w-lg">32rem max-width</div>
<div class="max-w-xl">36rem max-width</div>
<div class="max-w-2xl">42rem max-width</div>
<div class="max-w-3xl">48rem max-width</div>
<div class="max-w-4xl">56rem max-width</div>
<div class="max-w-5xl">64rem max-width</div>
<div class="max-w-6xl">72rem max-width</div>
<div class="max-w-7xl">80rem max-width</div>
<div class="max-w-full">100% max-width</div>
<div class="max-w-min">Min-content max-width</div>
<div class="max-w-max">Max-content max-width</div>
<div class="max-w-fit">Fit-content max-width</div>
<div class="max-w-prose">65ch max-width</div>
```

```
<div class="max-w-screen-sm">640px max-width</div>
<div class="max-w-screen-md">768px max-width</div>
<div class="max-w-screen-lg">1024px max-width</div>
<div class="max-w-screen-xl">1280px max-width</div>
<div class="max-w-screen-2xl">1536px max-width</div>
```

4. **Height**:

```
<div class="h-0">0 height</div>
<div class="h-1">0.25rem height</div>
<div class="h-2">0.5rem height</div>
<div class="h-3">0.75rem height</div>
<div class="h-4">1rem height</div>
<div class="h-5">1.25rem height</div>
<div class="h-6">1.5rem height</div>
<div class="h-8">2rem height</div>
<div class="h-10">2.5rem height</div>
<div class="h-12">3rem height</div>
<div class="h-16">4rem height</div>
<div class="h-20">5rem height</div>
<div class="h-24">6rem height</div>
<div class="h-32">8rem height</div>
<div class="h-40">10rem height</div>
<div class="h-48">12rem height</div>
<div class="h-56">14rem height</div>
<div class="h-64">16rem height</div>
<div class="h-auto">Auto height</div>
<div class="h-px">1px height</div>
<div class="h-full">100% height</div>
<div class="h-screen">100vh height</div>
<div class="h-min">Min-content height</div>
<div class="h-max">Max-content height</div>
<div class="h-fit">Fit-content height</div>
```

5. **Min-Height**:

```
<div class="min-h-0">0 min-height</div>
<div class="min-h-full">100% min-height</div>
<div class="min-h-screen">100vh min-height</div>
<div class="min-h-min">Min-content min-height</div>
<div class="min-h-max">Max-content min-height</div>
<div class="min-h-fit">Fit-content min-height</div>
```

6. **Max-Height**:

```
<div class="max-h-0">0 max-height</div>
<div class="max-h-full">100% max-height</div>
```

```
<div class="max-h-screen">100vh max-height</div>
<div class="max-h-min">Min-content max-height</div>
<div class="max-h-max">Max-content max-height</div>
<div class="max-h-fit">Fit-content max-height</div>
```

Typography Utilities

1. **Font Family**:

```
<p class="font-sans">Sans-serif font</p>
<p class="font-serif">Serif font</p>
<p class="font-mono">Monospace font</p>
```

2. **Font Size**:

```
<p class="text-xs">Extra small text</p>
<p class="text-sm">Small text</p>
<p class="text-base">Base text</p>
<p class="text-lg">Large text</p>
<p class="text-xl">Extra large text</p>
<p class="text-2xl">2XL text</p>
<p class="text-3xl">3XL text</p>
<p class="text-4xl">4XL text</p>
<p class="text-5xl">5XL text</p>
<p class="text-6xl">6XL text</p>
<p class="text-7xl">7XL text</p>
<p class="text-8xl">8XL text</p>
<p class="text-9xl">9XL text</p>
```

3. **Font Weight**:

```
<p class="font-thin">Thin text (100)</p>
<p class="font-extralight">Extra light text (200)</p>
<p class="font-light">Light text (300)</p>
<p class="font-normal">Normal text (400)</p>
<p class="font-medium">Medium text (500)</p>
<p class="font-semibold">Semi-bold text (600)</p>
<p class="font-bold">Bold text (700)</p>
<p class="font-extrabold">Extra bold text (800)</p>
<p class="font-black">Black text (900)</p>
```

4. **Font Style**:

```
<p class="italic">Italic text</p>
<p class="not-italic">Non-italic text</p>
```

5. **Font Variant Numeric**:

```
<p class="normal-nums">Normal numbers</p>
<p class="ordinal">Ordinal numbers (1st, 2nd)</p>
<p class="slashed-zero">Slashed zero (0)</p>
<p class="lining-nums">Lining figures</p>
<p class="oldstyle-nums">Old-style figures</p>
<p class="proportional-nums">Proportional figures</p>
<p class="tabular-nums">Tabular figures</p>
<p class="diagonal-fractions">Diagonal fractions (1/2)</p>
<p class="stacked-fractions">Stacked fractions (1/2)</p>
```

6. **Letter Spacing**:

```
<p class="tracking-tighter">Tighter tracking</p>
<p class="tracking-tight">Tight tracking</p>
<p class="tracking-normal">Normal tracking</p>
<p class="tracking-wide">Wide tracking</p>
<p class="tracking-wider">Wider tracking</p>
<p class="tracking-widest">Widest tracking</p>
```

7. **Line Height**:

```
<p class="leading-none">Leading none</p>
<p class="leading-tight">Tight leading</p>
<p class="leading-snug">Snug leading</p>
<p class="leading-normal">Normal leading</p>
<p class="leading-relaxed">Relaxed leading</p>
<p class="leading-loose">Loose leading</p>
<p class="leading-3">0.75rem line height</p>
<p class="leading-4">1rem line height</p>
<p class="leading-5">1.25rem line height</p>
<p class="leading-6">1.5rem line height</p>
<p class="leading-7">1.75rem line height</p>
<p class="leading-8">2rem line height</p>
<p class="leading-9">2.25rem line height</p>
<p class="leading-10">2.5rem line height</p>
```

8. **Text Alignment**:

```
<p class="text-left">Left-aligned text</p>
<p class="text-center">Center-aligned text</p>
<p class="text-right">Right-aligned text</p>
<p class="text-justify">Justified text</p>
```

```
<p class="text-start">Start-aligned text</p>
<p class="text-end">End-aligned text</p>
```

9. **Text Color**:

```
<p class="text-transparent">Transparent text</p>
<p class="text-current">Current color text</p>
<p class="text-black">Black text</p>
<p class="text-white">White text</p>
<p class="text-gray-50">Gray 50 text</p>
<p class="text-gray-100">Gray 100 text</p>
<p class="text-gray-200">Gray 200 text</p>
<p class="text-gray-300">Gray 300 text</p>
<p class="text-gray-400">Gray 400 text</p>
<p class="text-gray-500">Gray 500 text</p>
<p class="text-gray-600">Gray 600 text</p>
<p class="text-gray-700">Gray 700 text</p>
<p class="text-gray-800">Gray 800 text</p>
<p class="text-gray-900">Gray 900 text</p>
<p class="text-red-500">Red text</p>
<p class="text-blue-500">Blue text</p>
<p class="text-green-500">Green text</p>
<!-- And many other color options -->
```

10. **Text Decoration**:

```
<p class="underline">Underlined text</p>
<p class="overline">Overlined text</p>
<p class="line-through">Strikethrough text</p>
<p class="no-underline">No underline</p>
<p class="underline decoration-solid">Solid underline</p>
<p class="underline decoration-double">Double underline</p>
<p class="underline decoration-dotted">Dotted underline</p>
<p class="underline decoration-dashed">Dashed underline</p>
<p class="underline decoration-wavy">Wavy underline</p>
<p class="underline decoration-2">2px underline</p>
<p class="underline decoration-4">4px underline</p>
<p class="underline decoration-blue-500">Blue underline</p>
```

11. **Text Transform**:

```
<p class="uppercase">Uppercase text</p>
<p class="lowercase">Lowercase text</p>
<p class="capitalize">Capitalized text</p>
<p class="normal-case">Normal case text</p>
```

12. **Text Overflow**:

```
<p class="truncate">Truncated text with ellipsis</p>
<p class="overflow-ellipsis">Ellipsis overflow</p>
<p class="text-ellipsis">Text ellipsis</p>
<p class="overflow-clip">Clipped overflow</p>
<p class="text-clip">Text clip</p>
```

13. **Vertical Alignment**:

```
<span class="align-baseline">Baseline</span>
<span class="align-top">Top</span>
<span class="align-middle">Middle</span>
<span class="align-bottom">Bottom</span>
<span class="align-text-top">Text top</span>
<span class="align-text-bottom">Text bottom</span>
<span class="align-sub">Sub</span>
<span class="align-super">Super</span>
```

14. **Whitespace**:

```
<p class="whitespace-normal">Normal whitespace</p>
<p class="whitespace-nowrap">No wrap whitespace</p>
<p class="whitespace-pre">Pre whitespace</p>
<p class="whitespace-pre-line">Pre-line whitespace</p>
<p class="whitespace-pre-wrap">Pre-wrap whitespace</p>
<p class="whitespace-break-spaces">Break spaces</p>
```

15. **Word Break**:

```
<p class="break-normal">Normal word breaks</p>
<p class="break-words">Break words</p>
<p class="break-all">Break all</p>
<p class="break-keep">Keep all</p>
```

Background Utilities

1. **Background Attachment**:

```
<div class="bg-fixed">Fixed background</div>
<div class="bg-local">Local background</div>
<div class="bg-scroll">Scrollable background</div>
```

2. **Background Clip**:

```
<div class="bg-clip-border">Border box background clip</div>
<div class="bg-clip-padding">Padding box background clip</div>
<div class="bg-clip-content">Content box background clip</div>
<div class="bg-clip-text">Text background clip</div>
```

3. **Background Color**:

```
<div class="bg-transparent">Transparent background</div>
<div class="bg-current">Current color background</div>
<div class="bg-black">Black background</div>
<div class="bg-white">White background</div>
<div class="bg-gray-50">Gray 50 background</div>
<div class="bg-gray-100">Gray 100 background</div>
<div class="bg-gray-200">Gray 200 background</div>
<div class="bg-gray-300">Gray 300 background</div>
<div class="bg-gray-400">Gray 400 background</div>
<div class="bg-gray-500">Gray 500 background</div>
<div class="bg-gray-600">Gray 600 background</div>
<div class="bg-gray-700">Gray 700 background</div>
<div class="bg-gray-800">Gray 800 background</div>
<div class="bg-gray-900">Gray 900 background</div>
<div class="bg-red-500">Red background</div>
<div class="bg-blue-500">Blue background</div>
<div class="bg-green-500">Green background</div>
<!-- And many other color options -->
```

4. **Background Origin**:

```
<div class="bg-origin-border">Border background origin</div>
<div class="bg-origin-padding">Padding background origin</div>
<div class="bg-origin-content">Content background origin</div>
```

5. **Background Position**:

```
<div class="bg-bottom">Bottom position</div>
<div class="bg-center">Center position</div>
<div class="bg-left">Left position</div>
<div class="bg-left-bottom">Left bottom position</div>
<div class="bg-left-top">Left top position</div>
<div class="bg-right">Right position</div>
<div class="bg-right-bottom">Right bottom position</div>
<div class="bg-right-top">Right top position</div>
<div class="bg-top">Top position</div>
```

6. **Background Repeat**:

```
<div class="bg-repeat">Repeat background</div>
<div class="bg-no-repeat">No-repeat background</div>
<div class="bg-repeat-x">Repeat-x background</div>
<div class="bg-repeat-y">Repeat-y background</div>
<div class="bg-repeat-round">Repeat-round background</div>
<div class="bg-repeat-space">Repeat-space background</div>
```

7. **Background Size**:

```
<div class="bg-auto">Auto size background</div>
<div class="bg-cover">Cover background</div>
<div class="bg-contain">Contain background</div>
```

8. **Background Image**:

```
<div class="bg-none">No background image</div>
<div class="bg-gradient-to-t">Top gradient</div>
<div class="bg-gradient-to-tr">Top right gradient</div>
<div class="bg-gradient-to-r">Right gradient</div>
<div class="bg-gradient-to-br">Bottom right gradient</div>
<div class="bg-gradient-to-b">Bottom gradient</div>
<div class="bg-gradient-to-bl">Bottom left gradient</div>
<div class="bg-gradient-to-l">Left gradient</div>
<div class="bg-gradient-to-tl">Top left gradient</div>

<!-- Using from/via/to colors with gradients -->
<div class="bg-gradient-to-r from-red-500 to-blue-500">
  Red to blue gradient
</div>

<div class="bg-gradient-to-r from-indigo-500 via-purple-500 to-pink-500">
  Three-color gradient
</div>
```

Border Utilities

1. **Border Radius**:

```
<div class="rounded-none">No rounded corners</div>
<div class="rounded-sm">Small rounded corners</div>
<div class="rounded">Default rounded corners</div>
<div class="rounded-md">Medium rounded corners</div>
<div class="rounded-lg">Large rounded corners</div>
<div class="rounded-xl">Extra large rounded corners</div>
```

```
<div class="rounded-2xl">2X large rounded corners</div>
<div class="rounded-3xl">3X large rounded corners</div>
<div class="rounded-full">Fully rounded corners</div>

<!-- Specific corners -->
<div class="rounded-t-lg">Top corners rounded</div>
<div class="rounded-r-lg">Right corners rounded</div>
<div class="rounded-b-lg">Bottom corners rounded</div>
<div class="rounded-l-lg">Left corners rounded</div>
<div class="rounded-tl-lg">Top left corner rounded</div>
<div class="rounded-tr-lg">Top right corner rounded</div>
<div class="rounded-bl-lg">Bottom left corner rounded</div>
<div class="rounded-br-lg">Bottom right corner rounded</div>
```

2. **Border Width**:

```
<div class="border-0">No border</div>
<div class="border">1px border</div>
<div class="border-2">2px border</div>
<div class="border-4">4px border</div>
<div class="border-8">8px border</div>

<!-- Specific sides -->
<div class="border-t">Top border</div>
<div class="border-r">Right border</div>
<div class="border-b">Bottom border</div>
<div class="border-l">Left border</div>
<div class="border-t-2">2px top border</div>
<div class="border-r-4">4px right border</div>
<div class="border-b-8">8px bottom border</div>
<div class="border-l-0">No left border</div>
```

3. **Border Color**:

```
<div class="border border-transparent">Transparent border</div>
<div class="border border-current">Current color border</div>
<div class="border border-black">Black border</div>
<div class="border border-white">White border</div>
<div class="border border-gray-500">Gray border</div>
<div class="border border-red-500">Red border</div>
<div class="border border-blue-500">Blue border</div>

<!-- Specific sides with colors -->
<div class="border-t border-t-blue-500">Blue top border</div>
<div class="border-r border-r-green-500">Green right border</div>
<div class="border-b border-b-red-500">Red bottom border</div>
<div class="border-l border-l-yellow-500">Yellow left border</div>
```

4. **Border Style**:

```
<div class="border border-solid">Solid border</div>
<div class="border border-dashed">Dashed border</div>
<div class="border border-dotted">Dotted border</div>
<div class="border border-double">Double border</div>
<div class="border border-none">No border style</div>
```

5. **Divide Width** (for children):

```
<div class="divide-y divide-solid">
  <div>Item 1</div>
  <div>Item 2</div>
  <div>Item 3</div>
</div>

<div class="divide-x divide-solid">
  <div>Item 1</div>
  <div>Item 2</div>
  <div>Item 3</div>
</div>

<div class="divide-y-2">
  <div>Item 1</div>
  <div>Item 2</div>
  <div>Item 3</div>
</div>

<div class="divide-x-4">
  <div>Item 1</div>
  <div>Item 2</div>
  <div>Item 3</div>
</div>

<div class="divide-y divide-dashed">
  <div>Item 1</div>
  <div>Item 2</div>
  <div>Item 3</div>
</div>

<div class="divide-x divide-dotted">
  <div>Item 1</div>
  <div>Item 2</div>
  <div>Item 3</div>
</div>
```

6. **Divide Color**:

```
<div class="divide-y divide-blue-500">
  <div>Item 1</div>
  <div>Item 2</div>
  <div>Item 3</div>
</div>

<div class="divide-x divide-gray-200">
  <div>Item 1</div>
  <div>Item 2</div>
  <div>Item 3</div>
</div>
```

7. **Ring** (focus outlines):

```
<button class="ring">Default ring</button>
<button class="ring-0">No ring</button>
<button class="ring-1">1px ring</button>
<button class="ring-2">2px ring</button>
<button class="ring-4">4px ring</button>
<button class="ring-8">8px ring</button>

<button class="ring ring-blue-500">Blue ring</button>
<button class="ring ring-inset">Inset ring</button>
<button class="focus:ring focus:ring-blue-500">Focus ring</button>
```

Effect Utilities

1. **Box Shadow**:

```
<div class="shadow-sm">Small shadow</div>
<div class="shadow">Default shadow</div>
<div class="shadow-md">Medium shadow</div>
<div class="shadow-lg">Large shadow</div>
<div class="shadow-xl">Extra large shadow</div>
<div class="shadow-2xl">2X large shadow</div>
<div class="shadow-inner">Inner shadow</div>
<div class="shadow-none">No shadow</div>

<!-- Colored shadows (via CSS variables) -->
<div class="shadow-lg shadow-blue-500/50">Blue shadow</div>
```

2. **Opacity**:

```
<div class="opacity-0">0% opacity</div>
<div class="opacity-5">5% opacity</div>
<div class="opacity-10">10% opacity</div>
```

```
<div class="opacity-20">20% opacity</div>
<div class="opacity-25">25% opacity</div>
<div class="opacity-30">30% opacity</div>
<div class="opacity-40">40% opacity</div>
<div class="opacity-50">50% opacity</div>
<div class="opacity-60">60% opacity</div>
<div class="opacity-70">70% opacity</div>
<div class="opacity-75">75% opacity</div>
<div class="opacity-80">80% opacity</div>
<div class="opacity-90">90% opacity</div>
<div class="opacity-95">95% opacity</div>
<div class="opacity-100">100% opacity</div>
```

3. **Mix Blend Mode**:

```
<div class="mix-blend-normal">Normal blend</div>
<div class="mix-blend-multiply">Multiply blend</div>
<div class="mix-blend-screen">Screen blend</div>
<div class="mix-blend-overlay">Overlay blend</div>
<div class="mix-blend-darken">Darken blend</div>
<div class="mix-blend-lighten">Lighten blend</div>
<div class="mix-blend-color-dodge">Color dodge blend</div>
<div class="mix-blend-color-burn">Color burn blend</div>
<div class="mix-blend-hard-light">Hard light blend</div>
<div class="mix-blend-soft-light">Soft light blend</div>
<div class="mix-blend-difference">Difference blend</div>
<div class="mix-blend-exclusion">Exclusion blend</div>
<div class="mix-blend-hue">Hue blend</div>
<div class="mix-blend-saturation">Saturation blend</div>
<div class="mix-blend-color">Color blend</div>
<div class="mix-blend-luminosity">Luminosity blend</div>
```

4. **Background Blend Mode**:

```
<div class="bg-blend-normal">Normal bg blend</div>
<div class="bg-blend-multiply">Multiply bg blend</div>
<div class="bg-blend-screen">Screen bg blend</div>
<div class="bg-blend-overlay">Overlay bg blend</div>
<div class="bg-blend-darken">Darken bg blend</div>
<div class="bg-blend-lighten">Lighten bg blend</div>
<div class="bg-blend-color-dodge">Color dodge bg blend</div>
<div class="bg-blend-color-burn">Color burn bg blend</div>
<div class="bg-blend-hard-light">Hard light bg blend</div>
<div class="bg-blend-soft-light">Soft light bg blend</div>
<div class="bg-blend-difference">Difference bg blend</div>
<div class="bg-blend-exclusion">Exclusion bg blend</div>
<div class="bg-blend-hue">Hue bg blend</div>
<div class="bg-blend-saturation">Saturation bg blend</div>
```

```
<div class="bg-blend-color">Color bg blend</div>
<div class="bg-blend-luminosity">Luminosity bg blend</div>
```

5. **Filter Effects**:

```
<div class="blur-none">No blur</div>
<div class="blur-sm">Small blur</div>
<div class="blur">Default blur</div>
<div class="blur-md">Medium blur</div>
<div class="blur-lg">Large blur</div>
<div class="blur-xl">Extra large blur</div>
<div class="blur-2xl">2X large blur</div>
<div class="blur-3xl">3X large blur</div>

<div class="brightness-0">Brightness 0</div>
<div class="brightness-50">Brightness 50%</div>
<div class="brightness-75">Brightness 75%</div>
<div class="brightness-90">Brightness 90%</div>
<div class="brightness-95">Brightness 95%</div>
<div class="brightness-100">Brightness 100%</div>
<div class="brightness-105">Brightness 105%</div>
<div class="brightness-110">Brightness 110%</div>
<div class="brightness-125">Brightness 125%</div>
<div class="brightness-150">Brightness 150%</div>
<div class="brightness-200">Brightness 200%</div>

<div class="contrast-0">Contrast 0</div>
<div class="contrast-50">Contrast 50%</div>
<div class="contrast-75">Contrast 75%</div>
<div class="contrast-100">Contrast 100%</div>
<div class="contrast-125">Contrast 125%</div>
<div class="contrast-150">Contrast 150%</div>
<div class="contrast-200">Contrast 200%</div>

<div class="drop-shadow-sm">Small drop shadow</div>
<div class="drop-shadow">Default drop shadow</div>
<div class="drop-shadow-md">Medium drop shadow</div>
<div class="drop-shadow-lg">Large drop shadow</div>
<div class="drop-shadow-xl">Extra large drop shadow</div>
<div class="drop-shadow-2xl">2X large drop shadow</div>
<div class="drop-shadow-none">No drop shadow</div>

<div class="grayscale-0">No grayscale</div>
<div class="grayscale">Full grayscale</div>

<div class="hue-rotate-0">No hue rotate</div>
<div class="hue-rotate-15">15deg hue rotate</div>
<div class="hue-rotate-30">30deg hue rotate</div>
<div class="hue-rotate-60">60deg hue rotate</div>
<div class="hue-rotate-90">90deg hue rotate</div>
<div class="hue-rotate-180">180deg hue rotate</div>
<div class="-hue-rotate-30">-30deg hue rotate</div>
```

```
<div class="-hue-rotate-60">-60deg hue rotate</div>
<div class="-hue-rotate-90">-90deg hue rotate</div>
<div class="-hue-rotate-180">-180deg hue rotate</div>

<div class="invert-0">No invert</div>
<div class="invert">Full invert</div>

<div class="saturate-0">No saturation</div>
<div class="saturate-50">50% saturation</div>
<div class="saturate-100">100% saturation</div>
<div class="saturate-150">150% saturation</div>
<div class="saturate-200">200% saturation</div>

<div class="sepia-0">No sepia</div>
<div class="sepia">Full sepia</div>

<!-- Combining filters -->
<div class="blur-sm brightness-125 contrast-125">Multiple filters</div>
```

6. **Backdrop Filter**:

```
<div class="backdrop-blur-sm">Small backdrop blur</div>
<div class="backdrop-blur">Default backdrop blur</div>
<div class="backdrop-blur-md">Medium backdrop blur</div>
<div class="backdrop-brightness-50">Backdrop brightness 50%</div>
<div class="backdrop-contrast-200">Backdrop contrast 200%</div>
<div class="backdrop-grayscale">Full backdrop grayscale</div>
<div class="backdrop-hue-rotate-90">90deg backdrop hue rotate</div>
<div class="backdrop-invert">Full backdrop invert</div>
<div class="backdrop-opacity-50">50% backdrop opacity</div>
<div class="backdrop-saturate-150">150% backdrop saturation</div>
<div class="backdrop-sepia">Full backdrop sepia</div>
```

Transform Utilities

1. **Scale**:

```
<div class="scale-0">Scale 0</div>
<div class="scale-50">Scale 50%</div>
<div class="scale-75">Scale 75%</div>
<div class="scale-90">Scale 90%</div>
<div class="scale-95">Scale 95%</div>
<div class="scale-100">Scale 100%</div>
<div class="scale-105">Scale 105%</div>
<div class="scale-110">Scale 110%</div>
<div class="scale-125">Scale 125%</div>
<div class="scale-150">Scale 150%</div>

<div class="scale-x-0">Scale X 0</div>
```

```
<div class="scale-y-0">Scale Y 0</div>
<div class="scale-x-50">Scale X 50%</div>
<div class="scale-y-50">Scale Y 50%</div>
<div class="scale-x-150">Scale X 150%</div>
<div class="scale-y-150">Scale Y 150%</div>
```

2. **Rotate**:

```
<div class="rotate-0">Rotate 0deg</div>
<div class="rotate-1">Rotate 1deg</div>
<div class="rotate-2">Rotate 2deg</div>
<div class="rotate-3">Rotate 3deg</div>
<div class="rotate-6">Rotate 6deg</div>
<div class="rotate-12">Rotate 12deg</div>
<div class="rotate-45">Rotate 45deg</div>
<div class="rotate-90">Rotate 90deg</div>
<div class="rotate-180">Rotate 180deg</div>
<div class="-rotate-1">Rotate -1deg</div>
<div class="-rotate-2">Rotate -2deg</div>
<div class="-rotate-3">Rotate -3deg</div>
<div class="-rotate-6">Rotate -6deg</div>
<div class="-rotate-12">Rotate -12deg</div>
<div class="-rotate-45">Rotate -45deg</div>
<div class="-rotate-90">Rotate -90deg</div>
<div class="-rotate-180">Rotate -180deg</div>
```

3. **Translate**:

```
<div class="translate-x-0">Translate X 0</div>
<div class="translate-y-0">Translate Y 0</div>
<div class="translate-x-1">Translate X 0.25rem</div>
<div class="translate-y-1">Translate Y 0.25rem</div>
<div class="translate-x-2">Translate X 0.5rem</div>
<div class="translate-y-2">Translate Y 0.5rem</div>
<div class="translate-x-4">Translate X 1rem</div>
<div class="translate-y-4">Translate Y 1rem</div>
<div class="translate-x-8">Translate X 2rem</div>
<div class="translate-y-8">Translate Y 2rem</div>
<div class="translate-x-px">Translate X 1px</div>
<div class="translate-y-px">Translate Y 1px</div>
<div class="translate-x-1/2">Translate X 50%</div>
<div class="translate-y-1/2">Translate Y 50%</div>
<div class="translate-x-full">Translate X 100%</div>
<div class="translate-y-full">Translate Y 100%</div>
<div class="-translate-x-1">Translate X -0.25rem</div>
<div class="-translate-y-1">Translate Y -0.25rem</div>
<div class="-translate-x-full">Translate X -100%</div>
<div class="-translate-y-full">Translate Y -100%</div>
```

4. **Skew**:

```
<div class="skew-x-0">Skew X 0deg</div>
<div class="skew-y-0">Skew Y 0deg</div>
<div class="skew-x-1">Skew X 1deg</div>
<div class="skew-y-1">Skew Y 1deg</div>
<div class="skew-x-2">Skew X 2deg</div>
<div class="skew-y-2">Skew Y 2deg</div>
<div class="skew-x-3">Skew X 3deg</div>
<div class="skew-y-3">Skew Y 3deg</div>
<div class="skew-x-6">Skew X 6deg</div>
<div class="skew-y-6">Skew Y 6deg</div>
<div class="skew-x-12">Skew X 12deg</div>
<div class="skew-y-12">Skew Y 12deg</div>
<div class="-skew-x-1">Skew X -1deg</div>
<div class="-skew-y-1">Skew Y -1deg</div>
<div class="-skew-x-2">Skew X -2deg</div>
<div class="-skew-y-2">Skew Y -2deg</div>
<div class="-skew-x-3">Skew X -3deg</div>
<div class="-skew-y-3">Skew Y -3deg</div>
<div class="-skew-x-6">Skew X -6deg</div>
<div class="-skew-y-6">Skew Y -6deg</div>
<div class="-skew-x-12">Skew X -12deg</div>
<div class="-skew-y-12">Skew Y -12deg</div>
```

5. **Transform Origin**:

```
<div class="origin-center">Origin center</div>
<div class="origin-top">Origin top</div>
<div class="origin-top-right">Origin top right</div>
<div class="origin-right">Origin right</div>
<div class="origin-bottom-right">Origin bottom right</div>
<div class="origin-bottom">Origin bottom</div>
<div class="origin-bottom-left">Origin bottom left</div>
<div class="origin-left">Origin left</div>
<div class="origin-top-left">Origin top left</div>
```

Interactivity Utilities

1. **Accent Color**:

```
<input type="checkbox" class="accent-pink-500" />
<input type="radio" class="accent-blue-500" />
<input type="range" class="accent-green-500" />
<progress class="accent-purple-500" value="70" max="100"></progress>
```

2. **Appearance**:

```
<input class="appearance-none" type="checkbox" />
<select class="appearance-none"></select>
```

3. **Cursor**:

```
<div class="cursor-auto">Auto cursor</div>
<div class="cursor-default">Default cursor</div>
<div class="cursor-pointer">Pointer cursor</div>
<div class="cursor-wait">Wait cursor</div>
<div class="cursor-text">Text cursor</div>
<div class="cursor-move">Move cursor</div>
<div class="cursor-help">Help cursor</div>
<div class="cursor-not-allowed">Not-allowed cursor</div>
<div class="cursor-none">No cursor</div>
<div class="cursor-context-menu">Context menu cursor</div>
<div class="cursor-progress">Progress cursor</div>
<div class="cursor-cell">Cell cursor</div>
<div class="cursor-crosshair">Crosshair cursor</div>
<div class="cursor-vertical-text">Vertical text cursor</div>
<div class="cursor-alias">Alias cursor</div>
<div class="cursor-copy">Copy cursor</div>
<div class="cursor-no-drop">No-drop cursor</div>
<div class="cursor-grab">Grab cursor</div>
<div class="cursor-grabbing">Grabbing cursor</div>
<div class="cursor-all-scroll">All-scroll cursor</div>
<div class="cursor-col-resize">Column resize cursor</div>
<div class="cursor-row-resize">Row resize cursor</div>
<div class="cursor-n-resize">North resize cursor</div>
<div class="cursor-e-resize">East resize cursor</div>
<div class="cursor-s-resize">South resize cursor</div>
<div class="cursor-w-resize">West resize cursor</div>
<div class="cursor-ne-resize">Northeast resize cursor</div>
<div class="cursor-nw-resize">Northwest resize cursor</div>
<div class="cursor-se-resize">Southeast resize cursor</div>
<div class="cursor-sw-resize">Southwest resize cursor</div>
<div class="cursor-ew-resize">East-west resize cursor</div>
<div class="cursor-ns-resize">North-south resize cursor</div>
<div class="cursor-nesw-resize">Northeast-southwest resize cursor</div>
<div class="cursor-nwse-resize">Northwest-southeast resize cursor</div>
<div class="cursor-zoom-in">Zoom in cursor</div>
<div class="cursor-zoom-out">Zoom out cursor</div>
```

4. **Pointer Events**:

```
<div class="pointer-events-none">No pointer events</div>
<div class="pointer-events-auto">Auto pointer events</div>
```

5. **Resize**:

```
<div class="resize-none">Not resizable</div>
<div class="resize">Resizable (both directions)</div>
<div class="resize-y">Vertically resizable</div>
<div class="resize-x">Horizontally resizable</div>
```

6. **Scroll Behavior**:

```
<div class="scroll-auto">Auto scroll</div>
<div class="scroll-smooth">Smooth scroll</div>
```

7. **Scroll Margin**:

```
<div class="scroll-m-0">Scroll margin 0</div>
<div class="scroll-m-1">Scroll margin 0.25rem</div>
<div class="scroll-mt-4">Scroll margin top 1rem</div>
<div class="scroll-mr-4">Scroll margin right 1rem</div>
<div class="scroll-mb-4">Scroll margin bottom 1rem</div>
<div class="scroll-ml-4">Scroll margin left 1rem</div>
<div class="scroll-mx-4">Scroll margin x-axis 1rem</div>
<div class="scroll-my-4">Scroll margin y-axis 1rem</div>
```

8. **Scroll Padding**:

```
<div class="scroll-p-0">Scroll padding 0</div>
<div class="scroll-p-1">Scroll padding 0.25rem</div>
<div class="scroll-pt-4">Scroll padding top 1rem</div>
<div class="scroll-pr-4">Scroll padding right 1rem</div>
<div class="scroll-pb-4">Scroll padding bottom 1rem</div>
<div class="scroll-pl-4">Scroll padding left 1rem</div>
<div class="scroll-px-4">Scroll padding x-axis 1rem</div>
<div class="scroll-py-4">Scroll padding y-axis 1rem</div>
```

9. **Scroll Snap Align**:

```
<div class="snap-start">Snap start</div>
<div class="snap-end">Snap end</div>
<div class="snap-center">Snap center</div>
<div class="snap-align-none">No snap align</div>
```

10. **Scroll Snap Stop**:

```
<div class="snap-normal">Normal snap stop</div>
<div class="snap-always">Always snap stop</div>
```

11. **Scroll Snap Type**:

```
<div class="snap-none">No snap type</div>
<div class="snap-x">Horizontal snap type</div>
<div class="snap-y">Vertical snap type</div>
<div class="snap-both">Both directions snap type</div>
<div class="snap-mandatory">Mandatory snap</div>
<div class="snap-proximity">Proximity snap</div>
```

12. **Touch Action**:

```
<div class="touch-auto">Auto touch action</div>
<div class="touch-none">No touch action</div>
<div class="touch-pan-x">Horizontal pan touch action</div>
<div class="touch-pan-left">Pan left touch action</div>
<div class="touch-pan-right">Pan right touch action</div>
<div class="touch-pan-y">Vertical pan touch action</div>
<div class="touch-pan-up">Pan up touch action</div>
<div class="touch-pan-down">Pan down touch action</div>
<div class="touch-pinch-zoom">Pinch zoom touch action</div>
<div class="touch-manipulation">Manipulation touch action</div>
```

13. **User Select**:

```
<div class="select-none">No text selection</div>
<div class="select-text">Text selection</div>
<div class="select-all">All text selection</div>
<div class="select-auto">Auto text selection</div>
```

Transition and Animation Utilities

1. **Transition Property**:

```
<div class="transition-none">No transition</div>
<div class="transition-all">All properties transition</div>
<div class="transition">Default transition</div>
<div class="transition-colors">Colors transition</div>
<div class="transition-opacity">Opacity transition</div>
<div class="transition-shadow">Shadow transition</div>
<div class="transition-transform">Transform transition</div>
```

2. **Transition Duration**:

```
<div class="duration-0">0ms duration</div>
<div class="duration-75">75ms duration</div>
<div class="duration-100">100ms duration</div>
<div class="duration-150">150ms duration</div>
<div class="duration-200">200ms duration</div>
<div class="duration-300">300ms duration</div>
<div class="duration-500">500ms duration</div>
<div class="duration-700">700ms duration</div>
<div class="duration-1000">1000ms duration</div>
```

3. **Transition Timing Function**:

```
<div class="ease-linear">Linear timing</div>
<div class="ease-in">Ease-in timing</div>
<div class="ease-out">Ease-out timing</div>
<div class="ease-in-out">Ease-in-out timing</div>
```

4. **Transition Delay**:

```
<div class="delay-0">0ms delay</div>
<div class="delay-75">75ms delay</div>
<div class="delay-100">100ms delay</div>
<div class="delay-150">150ms delay</div>
<div class="delay-200">200ms delay</div>
<div class="delay-300">300ms delay</div>
<div class="delay-500">500ms delay</div>
<div class="delay-700">700ms delay</div>
<div class="delay-1000">1000ms delay</div>
```

5. **Animation**:

```
<div class="animate-none">No animation</div>
<div class="animate-spin">Spin animation</div>
<div class="animate-ping">Ping animation</div>
<div class="animate-pulse">Pulse animation</div>
<div class="animate-bounce">Bounce animation</div>
```

Configuration and Customization

Understanding the Configuration File

The Tailwind configuration file (`tailwind.config.js`) is where you can customize your project's design system.

```
// Default tailwind.config.js
module.exports = {
  content: ['./src/**/*.{html,js,jsx,ts,tsx,vue}'],
  theme: {
    screens: {
      sm: '640px',
      md: '768px',
      lg: '1024px',
      xl: '1280px',
      '2xl': '1536px',
    },
    colors: {
      transparent: 'transparent',
      current: 'currentColor',
      black: '#000',
      white: '#fff',
      gray: {
        50: '#f9fafb',
        100: '#f3f4f6',
        // ...and so on
      },
      // ...other default colors
    },
    spacing: {
      px: '1px',
      0: '0px',
      0.5: '0.125rem',
      1: '0.25rem',
      // ...and so on
    },
    // ...other theme settings
    extend: {},
  },
  plugins: [],
};
```

Theme Customization

1. **Extending vs. Overriding**:

```
// Extending adds your values while keeping defaults
module.exports = {
  theme: {
    extend: {
      colors: {
        'brand-blue': '#1992d4',
      },
```

```
    },
  },
};

// Overriding replaces all defaults
module.exports = {
  theme: {
    colors: {
      // You must redefine ALL colors here or lose defaults
      black: '#000',
      white: '#fff',
      'brand-blue': '#1992d4',
    },
  },
};
```

2. **Colors**:

```
module.exports = {
  theme: {
    extend: {
      colors: {
        // Named colors
        primary: '#3490dc',
        secondary: '#ffed4a',
        danger: '#e3342f',

        // Color object with shades
        brand: {
          50: '#f0f9ff',
          100: '#e0f2fe',
          200: '#bae6fd',
          300: '#7dd3fc',
          400: '#38bdf8',
          500: '#0ea5e9',
          600: '#0284c7',
          700: '#0369a1',
          800: '#075985',
          900: '#0c4a6e',
        },
      },
    },
  },
};
```

3. **Spacing**:

```
module.exports = {
  theme: {
```

```
    extend: {
      spacing: {
        72: '18rem',
        84: '21rem',
        96: '24rem',
        128: '32rem',
      },
    },
  },
};
```

4. **Font Families**:

```
module.exports = {
  theme: {
    extend: {
      fontFamily: {
        sans: [
          'Roboto',
          'ui-sans-serif',
          'system-ui',
          '-apple-system',
          'sans-serif',
        ],
        serif: ['Merriweather', 'ui-serif', 'Georgia', 'serif'],
        mono: ['JetBrains Mono', 'ui-monospace', 'monospace'],
        heading: ['Montserrat', 'sans-serif'],
      },
    },
  },
};
```

5. **Breakpoints**:

```
module.exports = {
  theme: {
    screens: {
      xs: '475px',
      sm: '640px',
      md: '768px',
      lg: '1024px',
      xl: '1280px',
      '2xl': '1536px',
      tablet: '640px',
      laptop: '1024px',
      desktop: '1280px',
    },
  },
};
```

6. **Border Radius**:

```
module.exports = {
  theme: {
    extend: {
      borderRadius: {
        xs: '0.125rem',
        '4xl': '2rem',
        primary: '0.375rem',
      },
    },
  },
};
```

7. **Box Shadows**:

```
module.exports = {
  theme: {
    extend: {
      boxShadow: {
        outline: '0 0 0 3px rgba(66, 153, 225, 0.5)',
        'inner-lg': 'inset 0 2px 10px 0 rgba(0, 0, 0, 0.05)',
        card: '0 4px 6px -1px rgba(0, 0, 0, 0.1), 0 2px 4px -1px rgba(0, 0,
0, 0.06)',
      },
    },
  },
};
```

Custom Plugins

1. **Creating a Custom Plugin**:

```
// In a separate file (e.g., plugins/buttons.js)
const plugin = require('tailwindcss/plugin');

module.exports = plugin(function ({ addComponents, theme }) {
  const buttons = {
    '.btn': {
      padding: `${theme('spacing.2')} ${theme('spacing.4')}`,
      borderRadius: theme('borderRadius.md'),
      fontWeight: theme('fontWeight.semibold'),
      display: 'inline-flex',
      alignItems: 'center',
      justifyContent: 'center',
      whiteSpace: 'nowrap',
      transition: '0.15s ease',
```

```
    },
    '.btn-primary': {
      backgroundColor: theme('colors.blue.500'),
      color: theme('colors.white'),
      '&:hover': {
        backgroundColor: theme('colors.blue.600'),
      },
    },
    '.btn-secondary': {
      backgroundColor: theme('colors.gray.200'),
      color: theme('colors.gray.800'),
      '&:hover': {
        backgroundColor: theme('colors.gray.300'),
      },
    },
  };

  addComponents(buttons);
});
```

2. **Using Custom Plugins**:

```
// tailwind.config.js
module.exports = {
  theme: {
    // ...theme options
  },
  plugins: [
    require('./plugins/buttons'),
    require('@tailwindcss/forms'),
    require('@tailwindcss/typography'),
    require('@tailwindcss/aspect-ratio'),
  ],
};
```

3. **Using Plugin Helper Functions**:

```
const plugin = require('tailwindcss/plugin');

module.exports = plugin(function ({
  addUtilities,
  addComponents,
  addBase,
  matchUtilities,
  theme,
}) {
  // Add base styles
  addBase({
    h1: { fontSize: theme('fontSize.2xl') },
```

```
      h2: { fontSize: theme('fontSize.xl') },
    });

    // Add components
    addComponents({
      '.card': {
        borderRadius: theme('borderRadius.lg'),
        padding: theme('spacing.6'),
        boxShadow: theme('boxShadow.xl'),
      },
    });

    // Add utilities
    addUtilities({
      '.content-auto': {
        contentVisibility: 'auto',
      },
    });

    // Add dynamic utilities
    matchUtilities(
      {
        'text-shadow': (value) => ({
          textShadow: value,
        }),
      },
      { values: theme('textShadow') }
    );
  });
```

Extending Core Plugins

```
// tailwind.config.js
module.exports = {
  theme: {
    extend: {
      // Add new variants for existing plugins
      backgroundColor: ({ theme }) => ({
        ...theme('colors'),
        twitter: '#1da1f2',
        facebook: '#1877f2',
        instagram: '#e4405f',
      }),

      // Add new font sizes
      fontSize: {
        tiny: '0.625rem',
        '2xs': '0.7rem',
        '10xl': '10rem',
      },
```

```
      // Add new keyframes
      keyframes: {
        wiggle: {
          '0%, 100%': { transform: 'rotate(-3deg)' },
          '50%': { transform: 'rotate(3deg)' },
        },
      },

      // Add custom animations
      animation: {
        wiggle: 'wiggle 1s ease-in-out infinite',
        'spin-slow': 'spin 3s linear infinite',
      },
    },
  },
};
```

Customizing Variants

```
// tailwind.config.js
module.exports = {
  // In Tailwind v3, all variants are enabled by default
  // For v2, you would use:
  variants: {
    extend: {
      // Enable group-focus for backgroundColor
      backgroundColor: ['group-focus'],

      // Enable hover, focus, and active for textColor
      textColor: ['hover', 'focus', 'active'],

      // Enable disabled for opacity
      opacity: ['disabled'],
    },
  },
};
```

Using CSS Variables

```
// tailwind.config.js
module.exports = {
  theme: {
    extend: {
      colors: {
        primary: 'var(--color-primary)',
        secondary: 'var(--color-secondary)',
        background: 'var(--color-background)',
      },
```

```
    },
  },
};
```

```
/* In your CSS */
:root {
  --color-primary: #3490dc;
  --color-secondary: #ffed4a;
  --color-background: #f8fafc;
}

.dark {
  --color-primary: #63b3ed;
  --color-secondary: #fef08a;
  --color-background: #1a202c;
}
```

Function-Based Configuration

```
// tailwind.config.js
const defaultTheme = require('tailwindcss/defaultTheme');

module.exports = {
  theme: {
    extend: {
      // Using functions to extend default values
      fontFamily: {
        sans: ['Inter var', ...defaultTheme.fontFamily.sans],
      },

      // Function to generate color shades
      colors: ({ colors }) => ({
        primary: {
          50: colors.blue[50],
          100: colors.blue[100],
          // ...and so on
          900: colors.blue[900],
        },
      }),

      // Using a function to compute spacing
      spacing: ({ theme }) => ({
        72: '18rem',
        84: '21rem',
        96: '24rem',
        'xs-max': `calc(${theme('screens.xs')} - 1px)`,
      }),
    },
```

```
  },
};
```

Presets

```
// my-preset.js
module.exports = {
  theme: {
    colors: {
      blue: {
        500: '#3b82f6',
        600: '#2563eb',
      },
      // ...other colors
    },
    borderRadius: {
      primary: '0.375rem',
    },
  },
  plugins: [require('@tailwindcss/forms')],
};
```

```
// tailwind.config.js
module.exports = {
  presets: [require('./my-preset.js')],
  // Project-specific customizations
  theme: {
    extend: {
      // ...
    },
  },
};
```

Responsive Design with Tailwind

Responsive Basics

Tailwind uses a mobile-first approach, where styles are applied to all screen sizes unless you specify otherwise with responsive prefixes.

```
<div class="w-full md:w-1/2 lg:w-1/3">
  <!-- Full width on mobile, half on medium, third on large -->
</div>
```

Default Breakpoints

Tailwind provides default breakpoints that you can use with any utility class:

- `sm`: 640px and up
- `md`: 768px and up
- `lg`: 1024px and up
- `xl`: 1280px and up
- `2xl`: 1536px and up

```
<!-- Text size responsive example -->
<h1 class="text-2xl sm:text-3xl md:text-4xl lg:text-5xl">Responsive Heading</h1>

<!-- Flex direction responsive example -->
<div class="flex flex-col md:flex-row">
  <div>Sidebar</div>
  <div>Content</div>
</div>

<!-- Grid columns responsive example -->
<div class="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 xl:grid-cols-4">
  <div>Item 1</div>
  <div>Item 2</div>
  <div>Item 3</div>
  <div>Item 4</div>
</div>

<!-- Spacing responsive example -->
<div class="p-4 md:p-6 lg:p-8">Responsive padding</div>

<!-- Visibility responsive example -->
<div class="hidden md:block">Only visible on medium screens and larger</div>

<div class="block md:hidden">Only visible on small screens</div>
```

Custom Breakpoints

You can customize breakpoints in your Tailwind configuration:

```
// tailwind.config.js
module.exports = {
  theme: {
    screens: {
      xs: '475px',
      sm: '640px',
      md: '768px',
      lg: '1024px',
      xl: '1280px',
      '2xl': '1536px',
      tablet: '640px',
      laptop: '1024px',
```

```
      desktop: '1280px',
      wide: '1440px',
    },
  },
};
```

Usage:

```
<div class="hidden xs:block">Visible from 475px+</div>

<div class="hidden tablet:flex laptop:justify-between">
  Appear as flex from tablet, with space-between from laptop
</div>
```

Max-Width Breakpoints

By default, Tailwind uses min-width breakpoints. For max-width behavior:

```
// tailwind.config.js
module.exports = {
  theme: {
    screens: {
      sm: '640px',
      md: '768px',
      lg: '1024px',
      xl: '1280px',
      '2xl': '1536px',
      // Max-width breakpoints
      'max-2xl': { max: '1535px' },
      'max-xl': { max: '1279px' },
      'max-lg': { max: '1023px' },
      'max-md': { max: '767px' },
      'max-sm': { max: '639px' },
    },
  },
};
```

Usage:

```
<div class="hidden max-md:block md:hidden">
  Only visible on screens smaller than md breakpoint
</div>
```

Complex Breakpoint Ranges

Define precise breakpoint ranges:

```
// tailwind.config.js
module.exports = {
  theme: {
    screens: {
      sm: '640px',
      md: '768px',
      lg: '1024px',
      // Complex range (only between md and lg)
      'md-only': { min: '768px', max: '1023px' },
      // Portrait mode tablets
      'tablet-portrait': {
        min: '640px',
        max: '1023px',
        orientation: 'portrait',
      },
      // Landscape mode
      landscape: { raw: '(orientation: landscape)' },
      // Dark mode
      dark: { raw: '(prefers-color-scheme: dark)' },
      // High DPI screens
      retina: { raw: '(min-resolution: 2dppx)' },
    },
  },
};
```

Usage:

```
<div class="hidden md-only:block">Only visible between 768px and 1023px</div>

<div class="bg-gray-100 dark:bg-gray-900">
  Light or dark depending on system preferences
</div>

<img class="h-10 retina:h-20" src="logo.png" alt="Logo" />
```

Creating Responsive Layouts

1. **Responsive Grid**:

```
<div
  class="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-4 gap-
4"
>
  <div class="bg-white p-4 shadow rounded">Item 1</div>
  <div class="bg-white p-4 shadow rounded">Item 2</div>
  <div class="bg-white p-4 shadow rounded">Item 3</div>
```

```
  <div class="bg-white p-4 shadow rounded">Item 4</div>
  <!-- More items... -->
</div>
```

2. **Responsive Navigation**:

```
<nav class="bg-blue-600 text-white">
  <div class="container mx-auto px-4 py-2">
    <div class="flex justify-between items-center">
      <div class="text-xl font-bold">Logo</div>

      <!-- Mobile Menu Button -->
      <button class="md:hidden">
        <!-- Menu icon -->
        <svg
          class="w-6 h-6"
          fill="none"
          stroke="currentColor"
          viewBox="0 0 24 24"
          xmlns="http://www.w3.org/2000/svg"
        >
          <path
            stroke-linecap="round"
            stroke-linejoin="round"
            stroke-width="2"
            d="M4 6h16M4 12h16M4 18h16"
          ></path>
        </svg>
      </button>

      <!-- Desktop Navigation -->
      <div class="hidden md:flex space-x-4">
        <a href="#" class="py-2">Home</a>
        <a href="#" class="py-2">About</a>
        <a href="#" class="py-2">Services</a>
        <a href="#" class="py-2">Contact</a>
      </div>
    </div>

    <!-- Mobile Navigation (hidden by default) -->
    <div class="hidden">
      <div class="flex flex-col space-y-2 py-4">
        <a href="#" class="py-2">Home</a>
        <a href="#" class="py-2">About</a>
        <a href="#" class="py-2">Services</a>
        <a href="#" class="py-2">Contact</a>
      </div>
    </div>
  </div>
</nav>
```

3. **Responsive Card Layout**:

```
<div class="p-4">
  <div class="bg-white rounded-lg shadow-md overflow-hidden">
    <div class="md:flex">
      <!-- Image takes full width on mobile, half on medium screens -->
      <div class="md:w-1/2">
        <img
          class="h-48 w-full object-cover md:h-full"
          src="image.jpg"
          alt="Image"
        />
      </div>

      <!-- Content area -->
      <div class="p-4 md:w-1/2">
        <h2 class="text-xl font-bold mb-2">Card Title</h2>
        <p class="text-gray-700 mb-4">
          This card layout changes from stacked to side-by-side at the
medium
          breakpoint.
        </p>
        <button class="bg-blue-500 text-white px-4 py-2 rounded">
          Learn More
        </button>
      </div>
    </div>
  </div>
</div>
```

4. **Responsive Table**:

```
<!-- Traditional table on desktop, card view on mobile -->
<div class="overflow-x-auto">
  <!-- Desktop table (hidden on small screens) -->
  <table class="hidden md:table w-full">
    <thead>
      <tr class="bg-gray-200">
        <th class="p-2 text-left">Name</th>
        <th class="p-2 text-left">Email</th>
        <th class="p-2 text-left">Role</th>
        <th class="p-2 text-left">Actions</th>
      </tr>
    </thead>
    <tbody>
      <tr class="border-b">
        <td class="p-2">John Doe</td>
        <td class="p-2">john@example.com</td>
        <td class="p-2">Admin</td>
        <td class="p-2">
```

```
            <button class="text-blue-500">Edit</button>
          </td>
        </tr>
        <!-- More rows... -->
      </tbody>
    </table>

    <!-- Mobile card view (hidden on medium screens and up) -->
    <div class="md:hidden space-y-4">
      <div class="bg-white p-4 rounded shadow">
        <div class="flex justify-between border-b pb-2">
          <span class="font-bold">Name:</span>
          <span>John Doe</span>
        </div>
        <div class="flex justify-between border-b py-2">
          <span class="font-bold">Email:</span>
          <span>john@example.com</span>
        </div>
        <div class="flex justify-between border-b py-2">
          <span class="font-bold">Role:</span>
          <span>Admin</span>
        </div>
        <div class="pt-2">
          <button class="text-blue-500">Edit</button>
        </div>
      </div>
      <!-- More cards... -->
    </div>
  </div>
```

5. **Responsive Typography**:

```
<div class="px-4 max-w-4xl mx-auto">
  <h1 class="text-3xl md:text-4xl lg:text-5xl font-bold mb-4">
    Responsive Title
  </h1>
  <h2 class="text-xl md:text-2xl lg:text-3xl font-semibold mb-2">
    Subtitle Scales Too
  </h2>
  <p class="text-base md:text-lg leading-relaxed mb-6">
    This paragraph text gets slightly larger on medium screens, maintaining
    good readability across devices.
  </p>
  <div class="space-y-4">
    <p class="text-sm md:text-base">
      Smaller text also scales appropriately to maintain hierarchy.
    </p>
  </div>
</div>
```

Responsive Component Design

1. **Responsive Hero Section**:

```
<section class="relative bg-gray-900 text-white">
  <!-- Background image -->
  <div class="absolute inset-0">
    <img src="hero-bg.jpg" alt="" class="w-full h-full object-cover" />
    <div class="absolute inset-0 bg-black opacity-60"></div>
  </div>

  <!-- Content -->
  <div class="relative container mx-auto px-4 py-16 md:py-24 lg:py-32">
    <div class="max-w-3xl">
      <h1 class="text-3xl md:text-4xl lg:text-5xl font-bold mb-4">
        Responsive Hero Heading
      </h1>
      <p class="text-lg md:text-xl mb-8">
        This subtitle is responsive and adjusts based on screen size.
      </p>
      <div class="space-x-4">
        <button
          class="bg-blue-600 hover:bg-blue-700 px-6 py-3 rounded font-
semibold"
        >
          Primary CTA
        </button>
        <button
          class="bg-transparent border border-white hover:bg-white
hover:text-gray-900 px-6 py-3 rounded font-semibold"
        >
          Secondary CTA
        </button>
      </div>
    </div>
  </div>
</section>
```

2. **Responsive Footer**:

```
<footer class="bg-gray-800 text-white pt-12 pb-8">
  <div class="container mx-auto px-4">
    <div class="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-8">
      <!-- Company Info -->
      <div>
        <h3 class="text-xl font-bold mb-4">Company Name</h3>
        <p class="mb-4">
          A brief description of the company and what it does.
        </p>
        <div class="flex space-x-4">
```

```
          <!-- Social icons -->
        </div>
      </div>

      <!-- Quick Links -->
      <div>
        <h3 class="text-xl font-bold mb-4">Quick Links</h3>
        <ul class="space-y-2">
          <li><a href="#" class="hover:underline">Home</a></li>
          <li><a href="#" class="hover:underline">About</a></li>
          <li><a href="#" class="hover:underline">Services</a></li>
          <li><a href="#" class="hover:underline">Blog</a></li>
          <li><a href="#" class="hover:underline">Contact</a></li>
        </ul>
      </div>

      <!-- Resources -->
      <div>
        <h3 class="text-xl font-bold mb-4">Resources</h3>
        <ul class="space-y-2">
          <li><a href="#" class="hover:underline">Documentation</a></li>
          <li><a href="#" class="hover:underline">Support</a></li>
          <li><a href="#" class="hover:underline">Privacy Policy</a></li>
          <li><a href="#" class="hover:underline">Terms of Service</a></li>
        </ul>
      </div>

      <!-- Contact -->
      <div>
        <h3 class="text-xl font-bold mb-4">Contact Us</h3>
        <p class="mb-2">123 Street Name, City</p>
        <p class="mb-2">contact@example.com</p>
        <p>(123) 456-7890</p>
      </div>
    </div>

    <div
      class="border-t border-gray-700 mt-8 pt-8 text-center text-gray-400"
    >
      <p>&copy; 2023 Company Name. All rights reserved.</p>
    </div>
  </div>
</footer>
```

3. **Responsive Pricing Table**:

```
<div class="container mx-auto px-4 py-12">
  <h2 class="text-3xl font-bold text-center mb-12">Pricing Plans</h2>

  <!-- Mobile: Stacked, Desktop: Side by side -->
  <div class="grid grid-cols-1 md:grid-cols-3 gap-8">
    <!-- Basic Plan -->
```

```
    <div
      class="border rounded-lg overflow-hidden hover:shadow-lg transition-
shadow"
    >
      <div class="bg-gray-100 p-6">
        <h3 class="text-xl font-bold">Basic</h3>
        <div class="mt-4">
          <span class="text-4xl font-bold">$9</span>
          <span class="text-gray-600">/month</span>
        </div>
      </div>
      <div class="p-6">
        <ul class="space-y-3">
          <li class="flex items-center">
            <svg
              class="h-5 w-5 text-green-500 mr-2"
              fill="none"
              stroke="currentColor"
              viewBox="0 0 24 24"
              xmlns="http://www.w3.org/2000/svg"
            >
              <path
                stroke-linecap="round"
                stroke-linejoin="round"
                stroke-width="2"
                d="M5 13l4 4L19 7"
              ></path>
            </svg>
            Feature 1
          </li>
          <li class="flex items-center">
            <svg
              class="h-5 w-5 text-green-500 mr-2"
              fill="none"
              stroke="currentColor"
              viewBox="0 0 24 24"
              xmlns="http://www.w3.org/2000/svg"
            >
              <path
                stroke-linecap="round"
                stroke-linejoin="round"
                stroke-width="2"
                d="M5 13l4 4L19 7"
              ></path>
            </svg>
            Feature 2
          </li>
          <li class="flex items-center">
            <svg
              class="h-5 w-5 text-gray-400 mr-2"
              fill="none"
              stroke="currentColor"
              viewBox="0 0 24 24"
              xmlns="http://www.w3.org/2000/svg"
```

```
            >
              <path
                stroke-linecap="round"
                stroke-linejoin="round"
                stroke-width="2"
                d="M6 18L18 6M6 6l12 12"
              ></path>
            </svg>
            Feature 3
          </li>
        </ul>
        <button
          class="mt-6 w-full bg-blue-500 hover:bg-blue-600 text-white py-2
rounded font-semibold"
        >
          Get Started
        </button>
      </div>
    </div>

    <!-- Pro Plan -->
    <div class="border rounded-lg overflow-hidden shadow-lg border-blue-
500">
      <div class="bg-blue-500 text-white p-6">
        <h3 class="text-xl font-bold">Pro</h3>
        <div class="mt-4">
          <span class="text-4xl font-bold">$29</span>
          <span>/month</span>
        </div>
      </div>
      <div class="p-6">
        <ul class="space-y-3">
          <!-- Features... -->
        </ul>
        <button
          class="mt-6 w-full bg-blue-500 hover:bg-blue-600 text-white py-2
rounded font-semibold"
        >
          Get Started
        </button>
      </div>
    </div>

    <!-- Enterprise Plan -->
    <div
      class="border rounded-lg overflow-hidden hover:shadow-lg transition-
shadow"
    >
      <!-- Similar structure -->
    </div>
  </div>
</div>
```

Best Practices for Responsive Design

1. **Start with Mobile First**: Build for mobile screens first, then add responsive prefixes for larger screens.

```
<!-- Mobile first approach -->
<div class="text-sm md:text-base lg:text-lg">
  Starts small, gets larger on bigger screens
</div>
```

2. **Use Containers Effectively**:

```
<div class="container mx-auto px-4 md:px-6 lg:px-8">
  <!-- Content is centered and has appropriate padding -->
</div>
```

3. **Consider Touch Targets**:

```
<button class="py-3 px-4 md:py-2 md:px-3">
  <!-- Larger touch target on mobile, smaller on desktop -->
</button>
```

4. **Logical Progression of Layout Changes**:

```
<div class="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-3 xl:grid-cols-4">
  <!-- Progressively more columns as screen size increases -->
</div>
```

5. **Avoid "Jumpy" Layouts**:

```
<div class="h-64 md:h-auto">
  <!-- Fixed height on mobile prevents content jump during loading -->
</div>
```

6. **Test Real Devices**: While browser dev tools are helpful, always test on real devices if possible.

7. **Use Aspect Ratio for Media**:

```
<div class="aspect-w-16 aspect-h-9">
  <iframe
    src="https://www.youtube.com/embed/dQw4w9WgXcQ"
    frameborder="0"
    allow="accelerometer; autoplay; clipboard-write; encrypted-media;
```

```
gyroscope; picture-in-picture"
    allowfullscreen
  ></iframe>
</div>
```

8. **Optimize Images**:

```
<picture>
  <source srcset="image-large.jpg" media="(min-width: 1024px)" />
  <source srcset="image-medium.jpg" media="(min-width: 640px)" />
  <img src="image-small.jpg" alt="Responsive image" />
</picture>
```

Component Extraction Strategies

Why Extract Components?

One common criticism of utility-first CSS is HTML becoming cluttered with many classes. Component extraction addresses this by:

1. Reducing repetition
2. Improving maintainability
3. Creating a consistent design system
4. Keeping HTML clean
5. Enabling better collaboration

Using Tailwind's @apply Directive

The `@apply` directive lets you extract repeated utility patterns into custom CSS classes.

1. **Basic Extraction**:

```
/* In your CSS file */
.btn {
  @apply inline-flex items-center px-4 py-2 border border-transparent
rounded-md font-semibold text-white bg-indigo-600 hover:bg-indigo-700
focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-indigo-500;
}

.btn-secondary {
  @apply bg-gray-200 text-gray-700 hover:bg-gray-300 focus:ring-gray-500;
}
```

2. **Extracting Complex Components**:

```
/* Card component */
.card {
  @apply bg-white rounded-lg shadow-md overflow-hidden;
}

.card-header {
  @apply px-6 py-4 border-b border-gray-200;
}

.card-title {
  @apply text-xl font-semibold text-gray-800;
}

.card-body {
  @apply p-6;
}

.card-footer {
  @apply px-6 py-4 bg-gray-50 border-t border-gray-200;
}
```

3. **Media Query Support**:

```
.responsive-element {
  @apply w-full p-4;

  @screen md {
    @apply w-1/2 p-6;
  }

  @screen lg {
    @apply w-1/3 p-8;
  }
}
```

4. **State Variants**:

```
.input {
  @apply w-full px-3 py-2 border border-gray-300 rounded-md shadow-sm;
}

.input:focus {
  @apply outline-none border-blue-500 ring-1 ring-blue-500;
}

.input:disabled {
  @apply bg-gray-100 text-gray-500 cursor-not-allowed;
}
```

5. **Pseudo-Element Support**:

```
.custom-checkbox {
  @apply relative pl-8;
}

.custom-checkbox::before {
  @apply absolute left-0 h-5 w-5 border border-gray-300 rounded;
  content: '';
}

.custom-checkbox:checked::before {
  @apply bg-blue-500 border-blue-500;
}
```

Framework Component Systems

Different frontend frameworks have different ways to extract components:

1. **React Components**:

```
// Button.jsx
function Button({ children, variant = 'primary', ...props }) {
  const baseClasses =
    'inline-flex items-center px-4 py-2 border border-transparent rounded-md
font-semibold focus:outline-none focus:ring-2 focus:ring-offset-2';

  const variantClasses = {
    primary:
      'text-white bg-indigo-600 hover:bg-indigo-700 focus:ring-indigo-500',
    secondary:
      'text-gray-700 bg-gray-200 hover:bg-gray-300 focus:ring-gray-500',
    danger: 'text-white bg-red-600 hover:bg-red-700 focus:ring-red-500',
  };

  return (
    <button
      className={`${baseClasses} ${variantClasses[variant]}`}
      {...props}
    >
      {children}
    </button>
  );
}

// Usage
function App() {
  return (
    <div>
```

```
      <Button>Primary Button</Button>
      <Button variant="secondary">Secondary Button</Button>
      <Button variant="danger">Danger Button</Button>
    </div>
  );
}
```

2. **Vue Components**:

```
<!-- Button.vue -->
<template>
  <button :class="[baseClasses, variantClasses[variant]]">
    <slot></slot>
  </button>
</template>

<script>
export default {
  props: {
    variant: {
      type: String,
      default: 'primary',
      validator: (value) =>
        ['primary', 'secondary', 'danger'].includes(value),
    },
  },
  computed: {
    baseClasses() {
      return 'inline-flex items-center px-4 py-2 border border-transparent
rounded-md font-semibold focus:outline-none focus:ring-2 focus:ring-offset-
2';
    },
    variantClasses() {
      return {
        primary:
          'text-white bg-indigo-600 hover:bg-indigo-700 focus:ring-indigo-
500',
        secondary:
          'text-gray-700 bg-gray-200 hover:bg-gray-300 focus:ring-gray-500',
        danger: 'text-white bg-red-600 hover:bg-red-700 focus:ring-red-500',
      };
    },
  },
};
</script>

<!-- Usage -->
<template>
  <div>
    <Button>Primary Button</Button>
    <Button variant="secondary">Secondary Button</Button>
    <Button variant="danger">Danger Button</Button>
```

```
    </div>
  </template>
```

3. **Angular Components**:

```
// button.component.ts
import { Component, Input } from '@angular/core';

@Component({
  selector: 'app-button',
  template: `
    <button [ngClass]="[baseClasses, variantClasses[variant]]">
      <ng-content></ng-content>
    </button>
  `,
})
export class ButtonComponent {
  @Input() variant: 'primary' | 'secondary' | 'danger' = 'primary';

  baseClasses =
    'inline-flex items-center px-4 py-2 border border-transparent rounded-md
font-semibold focus:outline-none focus:ring-2 focus:ring-offset-2';

  variantClasses = {
    primary:
      'text-white bg-indigo-600 hover:bg-indigo-700 focus:ring-indigo-500',
    secondary:
      'text-gray-700 bg-gray-200 hover:bg-gray-300 focus:ring-gray-500',
    danger: 'text-white bg-red-600 hover:bg-red-700 focus:ring-red-500',
  };
}

// Usage in template
// <app-button>Primary Button</app-button>
// <app-button variant="secondary">Secondary Button</app-button>
// <app-button variant="danger">Danger Button</app-button>
```

Template Partials (HTML/PHP/etc.)

For non-JavaScript frameworks or traditional server-rendered sites:

```
<!-- _button.php -->
<?php
function button($text, $variant = 'primary', $attributes = '') {
  $baseClasses = 'inline-flex items-center px-4 py-2 border border-transparent
rounded-md font-semibold focus:outline-none focus:ring-2 focus:ring-offset-2';

  $variantClasses = [
    'primary' => 'text-white bg-indigo-600 hover:bg-indigo-700 focus:ring-indigo-
```

```
500',
    'secondary' => 'text-gray-700 bg-gray-200 hover:bg-gray-300 focus:ring-gray-
500',
    'danger' => 'text-white bg-red-600 hover:bg-red-700 focus:ring-red-500',
  ];

  $classes = $baseClasses . ' ' . $variantClasses[$variant];

  echo "<button class=\"{$classes}\" {$attributes}>{$text}</button>";
}
?>

<!-- Usage -->
<?php
include '_button.php';
button('Primary Button');
button('Secondary Button', 'secondary');
button('Danger Button', 'danger', 'id="danger-btn"');
?>
```

Composition with JS Functions

JavaScript utility for generating class names:

```
// utils/classNames.js
function classNames(...classes) {
  return classes.filter(Boolean).join(' ');
}

// Usage
import classNames from './utils/classNames';

function Button({ children, variant, disabled }) {
  return (
    <button
      className={classNames(
        'px-4 py-2 rounded-md font-semibold',
        variant === 'primary' && 'bg-blue-500 text-white',
        variant === 'secondary' && 'bg-gray-200 text-gray-800',
        disabled && 'opacity-50 cursor-not-allowed'
      )}
      disabled={disabled}
    >
      {children}
    </button>
  );
}
```

Component Libraries

When to build your own component library:

1. **Pros**:
    - Tailored to your specific design language
    - Greater control over functionality and API
    - No dependency on third-party updates
    - Potentially smaller bundle size
2. **Cons**:
    - Time-consuming to build from scratch
    - Requires maintenance
    - Need to handle cross-browser issues
    - Potential for inconsistency

Popular Tailwind CSS component libraries:

1. **Tailwind UI**: Official premium component library
2. **Headless UI**: Unstyled, accessible components
3. **DaisyUI**: Free component library with themes
4. **Flowbite**: Open-source components with interactivity

Best Practices for Component Extraction

1. **Extract at the Right Time**:
    - Start with utility classes directly in HTML
    - Extract when patterns emerge or repeat
    - Don't prematurely abstract
2. **Maintain Consistency**:
    - Use consistent naming conventions
    - Document component props and variants
    - Build a component showcase or storybook
3. **Balance Flexibility and Standardization**:
    - Make components customizable but with guardrails
    - Use props for variations (size, color, etc.)
    - Allow extension through composition
4. **Progressive Enhancement**:
    - Build simple components first
    - Add complexity as needed
    - Test across different contexts
5. **Documentation**:
    - Document usage patterns

- Provide examples
- Explain props and customization options

Tailwind Directives

Core Tailwind Directives

Tailwind provides several directives to control how the framework generates styles:

1. **@tailwind**: Includes Tailwind's base, components, and utilities styles.

```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

2. **@layer**: Adds custom styles to Tailwind's base, components, or utilities layers.

```
@layer base {
  h1 {
    @apply text-2xl font-bold;
  }
  h2 {
    @apply text-xl font-semibold;
  }
}

@layer components {
  .btn {
    @apply px-4 py-2 rounded;
  }
}

@layer utilities {
  .text-shadow {
    text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.5);
  }
}
```

3. **@apply**: Extracts Tailwind utility classes into custom CSS.

```
.btn-primary {
  @apply bg-blue-500 text-white font-bold py-2 px-4 rounded;
}
```

4. **@variants**: Generates utility variations (hover, focus, etc.) for custom utilities.

```
@variants hover, focus {
  .text-shadow {
    text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.5);
  }
}
```

5. **@responsive**: Generates responsive variants for custom utilities.

```
@responsive {
  .text-shadow {
    text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.5);
  }
}
```

6. **@screen**: Reference Tailwind breakpoints in media queries.

```
.card {
  @apply p-4;

  @screen md {
    @apply p-6;
  }

  @screen lg {
    @apply p-8;
  }
}
```

@tailwind Directive

The `@tailwind` directive is used to include Tailwind's styles in your CSS:

```
/* Include Tailwind's base styles (resets, etc.) */
@tailwind base;

/* Include Tailwind's component classes */
@tailwind components;

/* Include Tailwind's utility classes */
@tailwind utilities;

/* Include Tailwind's variant prefix utilities (added in newer versions) */
@tailwind variants;
```

The order matters:

1. `base`: Global resets and base styles
2. `components`: Component classes (both from Tailwind and your custom ones)
3. `utilities`: Utility classes for granular styling
4. `variants`: Variant utilities (hover, focus, etc.)

@layer Directive

The `@layer` directive helps you organize custom styles into Tailwind's layers, which:

- Maintains proper specificity
- Respects the cascade
- Ensures proper purging of unused styles

1. **Base Layer**:

```
@layer base {
  /* Styling HTML elements */
  body {
    @apply bg-gray-50 text-gray-900;
  }

  h1,
  h2,
  h3,
  h4,
  h5,
  h6 {
    @apply font-bold;
  }

  /* Custom CSS variables */
  :root {
    --my-color: #1e293b;
  }
}
```

2. **Components Layer**:

```
@layer components {
  /* Custom card component */
  .card {
    @apply bg-white rounded-lg shadow-md overflow-hidden;
  }

  .card-header {
    @apply px-6 py-4 border-b border-gray-200;
  }

  .card-body {
```

```
    @apply p-6;
  }

  .card-footer {
    @apply px-6 py-4 bg-gray-50 border-t border-gray-200;
  }

  /* Custom button component */
  .btn {
    @apply inline-flex items-center px-4 py-2 rounded-md font-medium
focus:outline-none focus:ring-2 focus:ring-offset-2;
  }

  .btn-primary {
    @apply bg-blue-600 text-white hover:bg-blue-700 focus:ring-blue-500;
  }
}
```

3. **Utilities Layer**:

```
@layer utilities {
  /* Custom utilities */
  .text-shadow-sm {
    text-shadow: 0 1px 2px rgba(0, 0, 0, 0.1);
  }

  .text-shadow {
    text-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
  }

  .text-shadow-lg {
    text-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
  }

  /* Gradient text utility */
  .text-gradient {
    @apply bg-clip-text text-transparent bg-gradient-to-r;
  }

  /* Subtle animations */
  .hover-lift {
    @apply transition duration-300;
  }

  .hover-lift:hover {
    @apply transform -translate-y-1;
  }
}
```

@apply Directive

The `@apply` directive extracts Tailwind utilities into custom CSS classes:

1. **Basic Usage**:

```
.btn {
  @apply py-2 px-4 bg-blue-500 text-white rounded hover:bg-blue-700;
}
```

2. **Using with Responsive Utilities**:

```
.responsive-card {
  @apply p-4 md:p-6 lg:p-8;
}
```

3. **Using with Pseudo-Class Variants**:

```
.input-field {
  @apply border rounded px-4 py-2;
  @apply focus:outline-none focus:ring-2 focus:ring-blue-500;
  @apply hover:border-gray-400;
}
```

4. **Combining with Regular CSS**:

```
.custom-component {
  @apply flex items-center p-4 bg-white rounded;
  box-shadow: 0 4px 6px -1px rgba(0, 0, 0, 0.1);
  transition: box-shadow 0.3s ease;
}

.custom-component:hover {
  @apply bg-gray-50;
  box-shadow: 0 10px 15px -3px rgba(0, 0, 0, 0.1);
}
```

5. **Advanced Usage with Multiple Classes**:

```
.complex-btn {
  /* Base styles */
  @apply inline-flex items-center justify-center;
  @apply px-4 py-2 rounded-md font-medium;
  @apply focus:outline-none focus:ring-2 focus:ring-offset-2;

  /* Transitions */
```

```
    @apply transition-colors duration-200;

    /* Default variant styles */
    @apply bg-blue-500 text-white;
    @apply hover:bg-blue-600 focus:ring-blue-500;
  }
```

@variants Directive

The `@variants` directive generates variant versions of your custom utilities:

```
@variants hover, focus, active, disabled {
  .btn-outline {
    @apply border border-gray-300 bg-transparent;
  }
}
```

This generates:

```
.btn-outline {
  border: 1px solid #d1d5db;
  background-color: transparent;
}

.hover\:btn-outline:hover {
  border: 1px solid #d1d5db;
  background-color: transparent;
}

.focus\:btn-outline:focus {
  border: 1px solid #d1d5db;
  background-color: transparent;
}

.active\:btn-outline:active {
  border: 1px solid #d1d5db;
  background-color: transparent;
}

.disabled\:btn-outline:disabled {
  border: 1px solid #d1d5db;
  background-color: transparent;
}
```

@responsive Directive

The `@responsive` directive generates responsive variants of your custom utilities:

```
@responsive {
  .text-shadow {
    text-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
  }
}
```

This generates:

```
.text-shadow {
  text-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}

@media (min-width: 640px) {
  .sm\:text-shadow {
    text-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
  }
}

@media (min-width: 768px) {
  .md\:text-shadow {
    text-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
  }
}

@media (min-width: 1024px) {
  .lg\:text-shadow {
    text-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
  }
}

@media (min-width: 1280px) {
  .xl\:text-shadow {
    text-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
  }
}

@media (min-width: 1536px) {
  .\32xl\:text-shadow {
    text-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
  }
}
```

@screen Directive

The `@screen` directive lets you create media queries that reference your Tailwind breakpoints:

```
.sidebar {
  width: 100%;
```

```
  @screen md {
    width: 16rem;
  }

  @screen lg {
    width: 20rem;
  }
}
```

Compiles to:

```
.sidebar {
  width: 100%;
}

@media (min-width: 768px) {
  .sidebar {
    width: 16rem;
  }
}

@media (min-width: 1024px) {
  .sidebar {
    width: 20rem;
  }
}
```

Combining Directives

You can combine directives for powerful custom styling:

```
/* Custom component with responsive and variant styles */
@layer components {
  .badge {
    @apply inline-flex items-center px-2.5 py-0.5 rounded-full text-xs font-
medium;
  }

  .badge-blue {
    @apply bg-blue-100 text-blue-800;
  }

  .badge-green {
    @apply bg-green-100 text-green-800;
  }

  .badge-red {
    @apply bg-red-100 text-red-800;
```

```
  }

  /* Add hover and focus variants */
  @variants hover, focus {
    .badge-interactive {
      @apply transform transition-transform;
    }

    .badge-interactive {
      @apply scale-105;
    }
  }

  /* Responsive adjustments */
  @screen md {
    .badge {
      @apply px-3 py-1 text-sm;
    }
  }
}
```

Best Practices for Using Directives

1. **Use `@layer` for Organizing**: Group custom styles appropriately in the right layer to maintain correct cascade and specificity.
2. **Be Selective with `@apply`**:
   - Don't overuse—it reduces the benefit of in-HTML styling
   - Extract only common patterns that repeat frequently
   - Keep utilities simple and composable
3. **Prefer Modern Syntax**:
   - In newer Tailwind versions, `@variants` is less commonly used
   - You can now add variants directly in `tailwind.config.js`
4. **Avoid Unnecessary Abstractions**:
   - Don't create a custom class for every component variation
   - Use HTML composition when possible before extracting CSS
5. **Layer Management**:
   - Put custom resets in `@layer base`
   - Put reusable components in `@layer components`
   - Put custom utilities in `@layer utilities`

Dark Mode Implementation

Basic Dark Mode Setup

Tailwind supports dark mode through two main strategies: class-based and media query-based.

1. **Configuration**:

```
// tailwind.config.js
module.exports = {
  darkMode: 'class', // or 'media' for system preference
  // rest of your config
};
```

2. **Class-based Dark Mode**: Uses a `dark` class on an ancestor element (usually `html` or `body`).

```
<!-- Light mode by default, toggle to dark mode with JS -->
<html class="dark">
  <body>
    <div class="bg-white dark:bg-gray-800 text-gray-900 dark:text-white">
      Content adapts to dark mode when the "dark" class is present
    </div>
  </body>
</html>
```

3. **Media Query-based Dark Mode**: Automatically follows system preferences using `prefers-color-scheme`.

```
<!-- Automatically uses dark mode based on system preferences -->
<div class="bg-white dark:bg-gray-800 text-gray-900 dark:text-white">
  Content adapts to dark mode when system preference is dark
</div>
```

Dark Mode Variants

Add `dark:` prefix to any utility to apply it only in dark mode:

```
<!-- Text colors -->
<p class="text-gray-900 dark:text-white">
  This text is dark in light mode, light in dark mode
</p>

<!-- Background colors -->
<div class="bg-white dark:bg-gray-800">Dark background in dark mode</div>

<!-- Border colors -->
<div class="border border-gray-200 dark:border-gray-700">
  Border color changes in dark mode
</div>
```

```
<!-- Multiple properties -->
<button
  class="bg-blue-500 hover:bg-blue-600 text-white dark:bg-blue-600 dark:hover:bg-
blue-700"
>
  Button adapts to dark mode
</button>

<!-- Combining with responsive variants -->
<div class="bg-white dark:bg-gray-800 md:bg-gray-100 md:dark:bg-gray-900">
  Responsive AND dark mode aware
</div>
```

JavaScript Toggle Implementation

For class-based dark mode, you'll need JavaScript to toggle the `dark` class:

```
// Simple toggle
function toggleDarkMode() {
  document.documentElement.classList.toggle('dark');
}

// Toggle with local storage persistence
function toggleDarkMode() {
  if (document.documentElement.classList.contains('dark')) {
    document.documentElement.classList.remove('dark');
    localStorage.setItem('darkMode', 'light');
  } else {
    document.documentElement.classList.add('dark');
    localStorage.setItem('darkMode', 'dark');
  }
}

// Initialize from saved preference
function initDarkMode() {
  if (
    localStorage.getItem('darkMode') === 'dark' ||
    (localStorage.getItem('darkMode') === null &&
      window.matchMedia('(prefers-color-scheme: dark)').matches)
  ) {
    document.documentElement.classList.add('dark');
  } else {
    document.documentElement.classList.remove('dark');
  }
}

// Call on page load
initDarkMode();
```

Complete Dark Mode Toggle Example

```
<html>
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Dark Mode Demo</title>
    <link
      href="https://cdn.jsdelivr.net/npm/tailwindcss@2.2.19/dist/tailwind.min.css"
      rel="stylesheet"
    />
    <script>
      // Check for saved theme preference or use system preference
      const prefersDark = window.matchMedia(
        '(prefers-color-scheme: dark)'
      ).matches;
      const savedTheme = localStorage.getItem('theme');

      if (savedTheme === 'dark' || (!savedTheme && prefersDark)) {
        document.documentElement.classList.add('dark');
      }

      function toggleDarkMode() {
        const isDark = document.documentElement.classList.toggle('dark');
        localStorage.setItem('theme', isDark ? 'dark' : 'light');
      }
    </script>
  </head>
  <body
    class="min-h-screen bg-gray-100 dark:bg-gray-900 transition-colors duration-
200"
  >
    <div class="container mx-auto px-4 py-8">
      <header class="flex justify-between items-center mb-8">
        <h1 class="text-2xl font-bold text-gray-900 dark:text-white">
          Dark Mode Demo
        </h1>

        <button
          onclick="toggleDarkMode()"
          class="p-2 rounded-full bg-gray-200 dark:bg-gray-700 text-gray-800
dark:text-white focus:outline-none"
        >
          <!-- Sun icon for dark mode -->
          <svg
            class="h-6 w-6 hidden dark:block"
            fill="none"
            viewBox="0 0 24 24"
            stroke="currentColor"
          >
            <path
              stroke-linecap="round"
              stroke-linejoin="round"
              stroke-width="2"
```

```
              d="M12 3v1m0 16v1m9-9h-1M4 12H3m15.364 6.364l-.707-.707M6.343
6.343l-.707-.707m12.728 0l-.707.707M6.343 17.657l-.707.707M16 12a4 4 0 11-8 0 4 4
0 018 0z"
            />
          </svg>
          <!-- Moon icon for light mode -->
          <svg
            class="h-6 w-6 block dark:hidden"
            fill="none"
            viewBox="0 0 24 24"
            stroke="currentColor"
          >
            <path
              stroke-linecap="round"
              stroke-linejoin="round"
              stroke-width="2"
              d="M20.354 15.354A9 9 0 018.646 3.646 9.003 9.003 0 0012 21a9.003
9.003 0 008.354-5.646z"
            />
          </svg>
        </button>
      </header>

      <main>
        <div class="bg-white dark:bg-gray-800 shadow rounded-lg p-6 mb-6">
          <h2 class="text-xl font-semibold text-gray-900 dark:text-white mb-4">
            Card Title
          </h2>
          <p class="text-gray-700 dark:text-gray-300">
            This card's background, text, and border colors all adapt based on
            the current theme.
          </p>
        </div>

        <div class="grid grid-cols-1 md:grid-cols-2 gap-6">
          <div class="bg-white dark:bg-gray-800 shadow rounded-lg p-6">
            <h3
              class="text-lg font-semibold text-gray-900 dark:text-white mb-2"
            >
              Features
            </h3>
            <ul class="space-y-2 text-gray-700 dark:text-gray-300">
              <li class="flex items-center">
                <svg
                  class="h-5 w-5 text-green-500 mr-2"
                  fill="none"
                  viewBox="0 0 24 24"
                  stroke="currentColor"
                >
                  <path
                    stroke-linecap="round"
                    stroke-linejoin="round"
                    stroke-width="2"
                    d="M5 13l4 4L19 7"
```

```
                  />
                </svg>
                Automatic system preference detection
              </li>
              <li class="flex items-center">
                <svg
                  class="h-5 w-5 text-green-500 mr-2"
                  fill="none"
                  viewBox="0 0 24 24"
                  stroke="currentColor"
                >
                  <path
                    stroke-linecap="round"
                    stroke-linejoin="round"
                    stroke-width="2"
                    d="M5 13l4 4L19 7"
                  />
                </svg>
                Manual toggle with persistent preference
              </li>
              <li class="flex items-center">
                <svg
                  class="h-5 w-5 text-green-500 mr-2"
                  fill="none"
                  viewBox="0 0 24 24"
                  stroke="currentColor"
                >
                  <path
                    stroke-linecap="round"
                    stroke-linejoin="round"
                    stroke-width="2"
                    d="M5 13l4 4L19 7"
                  />
                </svg>
                Smooth transition between themes
              </li>
            </ul>
          </div>

          <div class="bg-white dark:bg-gray-800 shadow rounded-lg p-6">
            <h3
              class="text-lg font-semibold text-gray-900 dark:text-white mb-4"
            >
              Actions
            </h3>
            <div class="space-y-3">
              <button
                class="w-full bg-blue-500 hover:bg-blue-600 dark:bg-blue-600
dark:hover:bg-blue-700 text-white font-medium py-2 px-4 rounded"
              >
                Primary Button
              </button>
              <button
                class="w-full bg-gray-200 hover:bg-gray-300 dark:bg-gray-700
```

```
dark:hover:bg-gray-600 text-gray-800 dark:text-white font-medium py-2 px-4
rounded"
              >
                Secondary Button
              </button>
            </div>
          </div>
        </div>
      </main>

      <footer
        class="mt-12 pt-6 border-t border-gray-200 dark:border-gray-700 text-gray-
600 dark:text-gray-400 text-center"
      >
        Dark mode implementation with Tailwind CSS
      </footer>
    </div>
  </body>
</html>
```

Theming Beyond Dark/Light

For more complex theming needs, you can extend beyond just dark/light:

```
// tailwind.config.js
module.exports = {
  theme: {
    extend: {
      colors: {
        // Theme-specific colors using CSS variables
        primary: 'var(--color-primary)',
        secondary: 'var(--color-secondary)',
        background: 'var(--color-background)',
        text: 'var(--color-text)',
      },
    },
  },
  plugins: [],
};
```

```
/* Base CSS with theme variables */
:root {
  /* Light theme (default) */
  --color-primary: #3b82f6;
  --color-secondary: #10b981;
  --color-background: #ffffff;
  --color-text: #1f2937;
}
```

```
.dark {
  /* Dark theme */
  --color-primary: #60a5fa;
  --color-secondary: #34d399;
  --color-background: #111827;
  --color-text: #f9fafb;
}

.theme-blue {
  /* Blue theme */
  --color-primary: #1e40af;
  --color-secondary: #0369a1;
  --color-background: #e0f2fe;
  --color-text: #0c4a6e;
}

.theme-purple {
  /* Purple theme */
  --color-primary: #7c3aed;
  --color-secondary: #c026d3;
  --color-background: #f5f3ff;
  --color-text: #4c1d95;
}

.theme-high-contrast {
  /* High contrast theme for accessibility */
  --color-primary: #ffff00;
  --color-secondary: #ff00ff;
  --color-background: #000000;
  --color-text: #ffffff;
}
```

```
<!-- Using theme variables in HTML -->
<div class="bg-background text-text">
  <h1 class="text-primary">Themed Heading</h1>
  <p>
    Regular text uses the
    <span class="text-secondary">theme variables</span> too.
  </p>
</div>
```

Color System with HSL

For more sophisticated theming, using HSL colors can be powerful:

```
:root {
  /* Base hues */
  --hue-primary: 220; /* Blue */
  --hue-secondary: 160; /* Green */
```

```
  /* Light theme (default) */
  --saturation-primary: 90%;
  --lightness-primary: 60%;
  --saturation-secondary: 80%;
  --lightness-secondary: 50%;
  --bg-lightness: 98%;
  --text-lightness: 15%;
}

.dark {
  /* Dark theme adjustments */
  --saturation-primary: 70%;
  --lightness-primary: 50%;
  --saturation-secondary: 60%;
  --lightness-secondary: 40%;
  --bg-lightness: 10%;
  --text-lightness: 90%;
}

/* Generate color properties */
:root {
  --color-primary: hsl(
    var(--hue-primary),
    var(--saturation-primary),
    var(--lightness-primary)
  );
  --color-primary-light: hsl(
    var(--hue-primary),
    var(--saturation-primary),
    calc(var(--lightness-primary) + 15%)
  );
  --color-primary-dark: hsl(
    var(--hue-primary),
    var(--saturation-primary),
    calc(var(--lightness-primary) - 15%)
  );

  --color-secondary: hsl(
    var(--hue-secondary),
    var(--saturation-secondary),
    var(--lightness-secondary)
  );
  --color-secondary-light: hsl(
    var(--hue-secondary),
    var(--saturation-secondary),
    calc(var(--lightness-secondary) + 15%)
  );
  --color-secondary-dark: hsl(
    var(--hue-secondary),
    var(--saturation-secondary),
    calc(var(--lightness-secondary) - 15%)
  );
```

```
  --color-background: hsl(220, 20%, var(--bg-lightness));
  --color-text: hsl(220, 20%, var(--text-lightness));
}
```

Best Practices for Dark Mode

1. **Don't Just Invert Colors**:
   - Dark mode isn't just "light mode inverted"
   - Adjust contrast, saturation, and lightness carefully
   - Test for readability and eye strain
2. **Use Semantic Colors**:
   - Don't hardcode `dark:bg-gray-800` everywhere
   - Use variables or utility classes like `bg-background`
   - This helps when adding more themes later
3. **Consider Contrast and Accessibility**:
   - Maintain WCAG contrast requirements in both modes
   - Test with accessibility tools
   - Provide sufficient contrast for text and UI elements
4. **Smooth Transitions**:
   - Add transitions for a smoother experience when switching modes

```
<body
  class="bg-white dark:bg-gray-900 text-gray-900 dark:text-white transition-
colors duration-200"
></body>
```

5. **Handle Images and Media**:
   - Adjust image brightness/contrast in dark mode
   - Consider providing alternate images with `<picture>` element

```
<picture>
  <source srcset="logo-dark.png" media="(prefers-color-scheme: dark)" />
  <img src="logo-light.png" alt="Logo" />
</picture>
```

With Tailwind:

```
<img src="logo-light.png" class="block dark:hidden" alt="Logo" />
<img src="logo-dark.png" class="hidden dark:block" alt="Logo" />
```

```
```

6. **Remember Non-Color Properties**:
   - Adjust shadow intensity in dark mode
   - Consider reducing animation intensity
   - Adjust blur effects

```
<div class="shadow-lg dark:shadow-xl dark:shadow-black/20"></div>
```

Plugins and Ecosystem

Understanding Tailwind Plugins

Tailwind plugins extend the framework's functionality by adding new utilities, components, or variants. They can:

1. Add new utility classes
2. Add new component classes
3. Add variant modifiers (like `hover:`, `focus:`, etc.)
4. Add base styles
5. Register new values for theme configuration
6. Override or extend existing functionality

Official Tailwind Plugins

1. **@tailwindcss/typography**:
   - Adds a `prose` class for styling rich text content
   - Automatically formats article content with proper spacing, typography, and responsive design

```
npm install @tailwindcss/typography
```

```
// tailwind.config.js
module.exports = {
  plugins: [require('@tailwindcss/typography')],
};
```

```
<article class="prose lg:prose-xl">
  <h1>Article Title</h1>
  <p>Article content with proper typography...</p>
  <blockquote>A nice quote that's styled properly</blockquote>
  <ul>
    <li>List items with proper indentation</li>
```

```
    <li>And proper bullet styling</li>
  </ul>
</article>
```

2. **@tailwindcss/forms**:
   - Provides basic reset styles for form elements
   - Makes form elements adopt your theme styles automatically

```
npm install @tailwindcss/forms
```

```
// tailwind.config.js
module.exports = {
  plugins: [require('@tailwindcss/forms')],
};
```

```
<input
  type="text"
  class="mt-1 block w-full rounded-md border-gray-300 shadow-sm
focus:border-indigo-500 focus:ring focus:ring-indigo-200 focus:ring-opacity-
50"
/>
```

3. **@tailwindcss/aspect-ratio**:
   - Provides utilities for maintaining aspect ratios

```
npm install @tailwindcss/aspect-ratio
```

```
// tailwind.config.js
module.exports = {
  plugins: [require('@tailwindcss/aspect-ratio')],
};
```

```
<div class="aspect-w-16 aspect-h-9">
  <iframe
    src="https://www.youtube.com/embed/dQw4w9WgXcQ"
    frameborder="0"
    allow="accelerometer; autoplay; clipboard-write; encrypted-media;
gyroscope; picture-in-picture"
```

```
    allowfullscreen
  ></iframe>
</div>
```

4. **@tailwindcss/line-clamp**:
   - Provides utilities for truncating text to a specific number of lines
   - Now integrated into Tailwind core

```
<p class="line-clamp-3">
  This text will be truncated after 3 lines with an ellipsis at the end. Any
  additional text beyond that point will not be displayed.
</p>
```

Popular Community Plugins

1. **tailwindcss-transforms**:
   - Adds more transform utilities

```
npm install tailwindcss-transforms
```

```
// tailwind.config.js
module.exports = {
  plugins: [
    require('tailwindcss-transforms')({
      // options
    }),
  ],
};
```

```
<div class="rotate-45">This element is rotated 45 degrees</div>
```

2. **tailwindcss-fluid-typography**:
   - Provides fluid typography based on viewport size

```
// tailwind.config.js
module.exports = {
  plugins: [require('tailwindcss-fluid-typography')],
};
```

```
<h1 class="fluid-text-4xl">
  This heading scales smoothly between breakpoints
</h1>
```

3. **tailwindcss-elevation**:
   - Material Design-inspired elevation/shadow system

```
// tailwind.config.js
module.exports = {
  plugins: [require('tailwindcss-elevation')(['responsive'])],
};
```

```
<div class="elevation-2 hover:elevation-8">
  This element has Material Design shadows
</div>
```

4. **tailwindcss-interaction-variants**:
   - Adds interaction-based variants

```
// tailwind.config.js
module.exports = {
  plugins: [require('tailwindcss-interaction-variants')],
};
```

```
<div class="group-hover:text-blue-500">
  This text changes color when parent is hovered
</div>
```

Creating Your Own Plugin

A Tailwind plugin is a function that receives a set of utilities that you can use to inject styles:

```
// my-plugin.js
const plugin = require('tailwindcss/plugin');

module.exports = plugin(function ({
  addUtilities,
  addComponents,
  addBase,
```

```
  theme,
}) {
  // Add simple utilities
  const newUtilities = {
    '.text-shadow-sm': {
      textShadow: '0 1px 2px rgba(0, 0, 0, 0.1)',
    },
    '.text-shadow': {
      textShadow: '0 2px 4px rgba(0, 0, 0, 0.1)',
    },
    '.text-shadow-lg': {
      textShadow: '0 4px 6px rgba(0, 0, 0, 0.1)',
    },
  };

  addUtilities(newUtilities, ['responsive', 'hover']);

  // Add custom component styles
  const buttons = {
    '.btn': {
      padding: `${theme('spacing.2')} ${theme('spacing.4')}`,
      fontWeight: theme('fontWeight.semibold'),
      borderRadius: theme('borderRadius.default'),
      '&:focus': {
        outline: 'none',
        boxShadow: `0 0 0 3px rgba(66, 153, 225, 0.5)`,
      },
    },
    '.btn-blue': {
      backgroundColor: theme('colors.blue.500'),
      color: theme('colors.white'),
      '&:hover': {
        backgroundColor: theme('colors.blue.600'),
      },
    },
  };

  addComponents(buttons);

  // Add base styles
  addBase({
    h1: {
      fontSize: theme('fontSize.2xl'),
      fontWeight: theme('fontWeight.bold'),
    },
    h2: {
      fontSize: theme('fontSize.xl'),
      fontWeight: theme('fontWeight.semibold'),
    },
  });
});
```

Then use it in your configuration:

```
// tailwind.config.js
module.exports = {
  plugins: [require('./my-plugin.js')],
};
```

Plugin with Options

Create a configurable plugin:

```
// gradient-plugin.js
const plugin = require('tailwindcss/plugin');

module.exports = plugin.withOptions(function (options = {}) {
  return function ({ addUtilities, theme, e }) {
    const defaultOptions = {
      variants: ['responsive', 'hover'],
      gradients: {
        'blue-green': ['#4facfe', '#00f2fe'],
        'purple-pink': ['#667eea', '#764ba2'],
        'red-yellow': ['#f5576c', '#f2b851'],
      },
      directions: {
        t: 'to top',
        tr: 'to top right',
        r: 'to right',
        br: 'to bottom right',
        b: 'to bottom',
        bl: 'to bottom left',
        l: 'to left',
        tl: 'to top left',
      },
    };

    const opts = { ...defaultOptions, ...options };

    const utilities = {};

    // Generate the utilities
    Object.entries(opts.gradients).forEach(([name, colors]) => {
      Object.entries(opts.directions).forEach(([direction, value]) => {
        const className = `.${e(`bg-gradient-${direction}-${name}`)}`;

        utilities[className] = {
          backgroundImage: `linear-gradient(${value}, ${colors.join(', ')})`,
        };
      });
    });

    addUtilities(utilities, opts.variants);
```

```
  };
});
```

Use the plugin with custom options:

```
// tailwind.config.js
module.exports = {
  plugins: [
    require('./gradient-plugin.js')({
      gradients: {
        brand: ['#3490dc', '#6574cd'],
        sunset: ['#f6ad55', '#ed64a6'],
      },
    }),
  ],
};
```

Tailwind UI

Tailwind UI is the official premium component library from Tailwind Labs. It provides:

- Production-ready UI components
- Accessible, responsive designs
- Marketing, application UI, and ecommerce components
- Available in HTML, React, and Vue formats

Example Tailwind UI component (simplified):

```
<!-- Dropdown component -->
<div class="relative inline-block text-left">
  <div>
    <button
      type="button"
      class="inline-flex justify-center w-full rounded-md border border-gray-300
shadow-sm px-4 py-2 bg-white text-sm font-medium text-gray-700 hover:bg-gray-50
focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-offset-gray-100
focus:ring-indigo-500"
      id="options-menu"
      aria-expanded="true"
      aria-haspopup="true"
    >
      Options
      <svg
        class="-mr-1 ml-2 h-5 w-5"
        xmlns="http://www.w3.org/2000/svg"
        viewBox="0 0 20 20"
        fill="currentColor"
        aria-hidden="true"
      >
```

```
        <path
          fill-rule="evenodd"
          d="M5.293 7.293a1 1 0 011.414 0L10 10.586l3.293-3.293a1 1 0 111.414
1.414l-4 4a1 1 0 01-1.414 0l-4-4a1 1 0 010-1.414z"
          clip-rule="evenodd"
        />
      </svg>
    </button>
  </div>

  <div
    class="origin-top-right absolute right-0 mt-2 w-56 rounded-md shadow-lg bg-
white ring-1 ring-black ring-opacity-5 focus:outline-none"
    role="menu"
    aria-orientation="vertical"
    aria-labelledby="options-menu"
  >
    <div class="py-1" role="none">
      <a
        href="#"
        class="block px-4 py-2 text-sm text-gray-700 hover:bg-gray-100 hover:text-
gray-900"
        role="menuitem"
        >Account settings</a
      >
      <a
        href="#"
        class="block px-4 py-2 text-sm text-gray-700 hover:bg-gray-100 hover:text-
gray-900"
        role="menuitem"
        >Support</a
      >
      <a
        href="#"
        class="block px-4 py-2 text-sm text-gray-700 hover:bg-gray-100 hover:text-
gray-900"
        role="menuitem"
        >License</a
      >
      <form method="POST" action="#" role="none">
        <button
          type="submit"
          class="block w-full text-left px-4 py-2 text-sm text-gray-700 hover:bg-
gray-100 hover:text-gray-900"
          role="menuitem"
        >
          Sign out
        </button>
      </form>
    </div>
  </div>
</div>
```

Headless UI

Headless UI provides unstyled, fully accessible UI components that integrate well with Tailwind CSS.

Example of a Headless UI Dropdown in React:

```
import { Menu } from '@headlessui/react';

function MyDropdown() {
  return (
    <Menu as="div" className="relative inline-block text-left">
      <Menu.Button className="inline-flex justify-center w-full px-4 py-2 text-sm
font-medium text-white bg-black rounded-md bg-opacity-20 hover:bg-opacity-30
focus:outline-none focus-visible:ring-2 focus-visible:ring-white focus-
visible:ring-opacity-75">
        Options
      </Menu.Button>
      <Menu.Items className="absolute right-0 w-56 mt-2 origin-top-right bg-white
divide-y divide-gray-100 rounded-md shadow-lg ring-1 ring-black ring-opacity-5
focus:outline-none">
        <div className="px-1 py-1">
          <Menu.Item>
            {({ active }) => (
              <button
                className={`${
                  active ? 'bg-violet-500 text-white' : 'text-gray-900'
                } group flex rounded-md items-center w-full px-2 py-2 text-sm`}
              >
                Edit
              </button>
            )}
          </Menu.Item>
          <Menu.Item>
            {({ active }) => (
              <button
                className={`${
                  active ? 'bg-violet-500 text-white' : 'text-gray-900'
                } group flex rounded-md items-center w-full px-2 py-2 text-sm`}
              >
                Duplicate
              </button>
            )}
          </Menu.Item>
        </div>
      </Menu.Items>
    </Menu>
  );
}
```

Design System Frameworks

Several frameworks combine Tailwind with design systems:

1. **DaisyUI**:
   - Component library with customizable themes
   - Simple class-based API (`class="btn btn-primary"`)
   - Built on top of Tailwind

```
<button class="btn btn-primary">Button</button>
<div class="card w-96 bg-base-100 shadow-xl">
  <figure><img src="..." alt="..." /></figure>
  <div class="card-body">
    <h2 class="card-title">Card title</h2>
    <p>Card content</p>
    <div class="card-actions justify-end">
      <button class="btn btn-primary">Buy Now</button>
    </div>
  </div>
</div>
```

2. **Flowbite**:
   - Open-source UI component library
   - Interactive components without JavaScript frameworks
   - Accessible and responsive

```
<button
  type="button"
  class="text-white bg-blue-700 hover:bg-blue-800 focus:ring-4 focus:ring-
blue-300 font-medium rounded-lg text-sm px-5 py-2.5 text-center"
>
  Click me
</button>
```

3. **WindiCSS**:
   - Alternative to Tailwind with additional features
   - Faster compilation, attributify mode, shortcuts
   - Compatible with Tailwind

```
<!-- Attributify mode -->
<button
  bg="blue-500 hover:blue-700"
  text="white sm"
  font="medium"
  p="y-2 x-4"
  border="rounded"
>
```

```
  Button
</button>
```

Resources and Community

Key resources in the Tailwind ecosystem:

1. **Official Resources**:
   - Tailwind CSS Documentation
   - Tailwind UI
   - Tailwind Play (interactive playground)
   - Headless UI
2. **Community Libraries**:
   - Tailwind Components (free component library)
   - Tailblocks (ready-to-use blocks)
   - Meraki UI (responsive components)
   - Kometa UI Kit (UI sections)
3. **Tools**:
   - Tailwind CSS IntelliSense (VS Code extension)
   - Headwind (class sorter)
   - Windy (Tailwind design tool)
   - Tailwind Config Viewer
4. **Learning**:
   - Tailwind CSS: From Zero to Production (official video series)
   - Refactoring UI (design guide from Tailwind creators)
   - Advanced Tailwind CSS Course

JIT Compiler and Performance

What is the JIT Compiler?

The Just-in-Time (JIT) Compiler for Tailwind CSS was introduced to dramatically improve the development experience and build performance. In Tailwind v3.0+, JIT mode is the default.

Key benefits:

1. **Lightning-fast build times**
2. **No more build size concerns during development**
3. **Generate utilities on-demand**
4. **Support for arbitrary values**
5. **Support for all variants**
6. **Better browser performance**

How JIT Works

The traditional Tailwind approach:

1. Generate ALL possible utility classes
2. Purge unused classes in production

The JIT approach:

1. Scan your source files for used class names
2. Generate ONLY the CSS you need on-demand
3. No purging needed - only what you use gets generated

Using Arbitrary Values

JIT enables arbitrary value support using square bracket notation:

```
<!-- Specific width -->
<div class="w-[32.25rem]">Custom width</div>

<!-- Complex gradient with arbitrary values -->
<div class="bg-gradient-to-r from-[#d53369] to-[#daae51]">Custom gradient</div>

<!-- Custom font size -->
<p class="text-[18px]">Custom text size</p>

<!-- Arbitrary grid template -->
<div class="grid grid-cols-[1fr_500px_2fr]">
  Three columns with custom widths
</div>

<!-- Functions and calculations -->
<div class="w-[calc(100%-theme('spacing.4'))]">
  Using calculations and theme values
</div>

<!-- Complex arbitrary values -->
<div class="bg-[url('/img/hero.jpg')]">Background image</div>

<div class="grid-rows-[repeat(20,minmax(0,1fr))]">20 equal rows</div>
```

All Variants Support

With JIT, all variants are available for all utilities without configuration:

```
<!-- Previously not possible without configuration -->
<div class="dark:focus:hover:bg-blue-500">Multiple stacked variants</div>

<!-- Pseudo-element variants -->
```

```
<div class="before:content-['Hello_World'] before:block before:font-bold">
  Uses before pseudo-element
</div>

<!-- Arbitrary variants -->
<div class="[@media(min-width:800px)]:text-red-500">
  Custom media query variant
</div>

<!-- Group and peer variants -->
<div class="group">
  <div class="hidden group-hover:block">Shows on parent hover</div>
</div>

<input type="checkbox" class="peer" />
<div class="hidden peer-checked:block">Shows when checkbox is checked</div>
```

Performance Improvements

1. **Development Performance**:
   - Traditional builds could take 3-30+ seconds
   - JIT builds typically take less than 50ms
2. **Runtime Performance**:
   - Smaller CSS files (only what you use)
   - Modern browsers parse and process CSS faster
   - No unused CSS for the browser to handle
3. **Production Optimization**:
   - No separate PurgeCSS step needed
   - More accurate tree-shaking built in
   - Content-aware optimizations

Setting Up JIT

In Tailwind CSS v3.0+, JIT is the default mode. For older versions:

```
// tailwind.config.js
module.exports = {
  mode: 'jit',
  purge: ['./src/**/*.{html,js,jsx,ts,tsx,vue}'],
  theme: {
    // ...
  },
  // ...
};
```

Optimizing JIT for Large Projects

For larger projects, you can optimize JIT performance:

1. **Be specific with content paths**:

```
// tailwind.config.js
module.exports = {
  content: [
    './src/pages/**/*.{js,jsx,ts,tsx}',
    './src/components/**/*.{js,jsx,ts,tsx}',
    './src/layouts/**/*.{js,jsx,ts,tsx}',
    // Don't include files that don't need to be scanned
    // './src/utils/**/*.{js,ts}',
    // './src/api/**/*.{js,ts}',
  ],
  // ...
};
```

2. **Using safelist for dynamic classes**:

```
// tailwind.config.js
module.exports = {
  content: ['./src/**/*.{html,js}'],
  safelist: [
    // Specific classes
    'bg-red-500',
    'text-3xl',
    // Regular expression patterns (text-red-100 through text-red-900)
    {
      pattern: /bg-(red|green|blue)-(100|200|300|400|500|600|700|800|900)/,
    },
  ],
  // ...
};
```

3. **Optimizing for CI/CD pipelines**:

```
// package.json
{
  "scripts": {
    "dev": "tailwindcss -i ./src/input.css -o ./dist/output.css --watch",
    "build": "tailwindcss -i ./src/input.css -o ./dist/output.css --minify"
  }
}
```

JIT Pitfalls and Solutions

1. **Dynamic Class Names**:

   Problem: Dynamic concatenated classes might not be detected.

   ```
   // This might not be detected
   const dynamicClass = `text-${color}-500`;
   ```

   Solutions:

   a. Use safelist:

   ```
   // tailwind.config.js
   module.exports = {
     safelist: [
       {
         pattern: /text-(red|green|blue)-
   (100|200|300|400|500|600|700|800|900)/,
       },
     ],
   };
   ```

   b. Use complete class names:

   ```
   // Instead of concatenation, use an object lookup
   const colorMap = {
     red: 'text-red-500',
     green: 'text-green-500',
     blue: 'text-blue-500',
   };

   // Then use
   const className = colorMap[color];
   ```

2. **Build-time vs. Runtime**:

   Problem: JIT runs at build time, so classes generated at runtime won't work.

   Solution: Include all possible class combinations in your safelist.

3. **Increased Development Dependencies**:

   Problem: JIT requires PostCSS and more dependencies.

   Solution: Use the Tailwind CLI or bundled version for simpler projects.

Development Workflow with JIT

1. **VS Code Extension**:

   Install "Tailwind CSS IntelliSense" for autocomplete, syntax highlighting, and linting.

2. **Watch Mode**:

```
npx tailwindcss -i ./src/input.css -o ./dist/output.css --watch
```

3. **Browser Sync or Live Server**:

```
npm install -g browser-sync
browser-sync start --server --files "css/*.css, *.html"
```

4. **Framework Integration**:

   Most frameworks have hot-reloading built-in, which works well with JIT:

   - Next.js: Works out of the box
   - Create React App: Works with CRACO
   - Vue CLI: Works with PostCSS plugin

Framework Integration

React and Tailwind CSS

1. **Setup with Create React App**:

```
npx create-react-app my-app
cd my-app
npm install -D tailwindcss postcss autoprefixer
npx tailwindcss init -p
```

   Configure Tailwind:

```
// tailwind.config.js
module.exports = {
  content: ['./src/**/*.{js,jsx,ts,tsx}'],
  theme: {
    extend: {},
  },
  plugins: [],
};
```

   Add Tailwind directives to CSS:

```
/* src/index.css */
@tailwind base;
@tailwind components;
@tailwind utilities;
```

Import CSS in your app:

```
// src/index.js
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';

ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById('root')
);
```

2. **React Component Example**:

```
// src/components/Button.js
import React from 'react';

function Button({ children, primary }) {
  const baseClasses =
    'px-4 py-2 rounded font-semibold focus:outline-none focus:ring-2
focus:ring-offset-2';
  const primaryClasses =
    'bg-blue-500 text-white hover:bg-blue-600 focus:ring-blue-500';
  const secondaryClasses =
    'bg-gray-200 text-gray-800 hover:bg-gray-300 focus:ring-gray-500';

  return (
    <button
      className={`${baseClasses} ${
        primary ? primaryClasses : secondaryClasses
      }`}
    >
      {children}
    </button>
  );
}

export default Button;
```

Usage:

```
// src/App.js
import Button from './components/Button';

function App() {
  return (
    <div className="p-8">
      <h1 className="text-3xl font-bold mb-6">React with Tailwind</h1>
      <div className="space-x-4">
        <Button primary>Primary Button</Button>
        <Button>Secondary Button</Button>
      </div>
    </div>
  );
}
```

3. **Class Composition Libraries**:

a. **clsx**:

```
import clsx from 'clsx';

function Button({ primary, disabled }) {
  return (
    <button
      className={clsx(
        'px-4 py-2 rounded',
        primary ? 'bg-blue-500 text-white' : 'bg-gray-200 text-gray-800',
        disabled && 'opacity-50 cursor-not-allowed'
      )}
      disabled={disabled}
    >
      Button
    </button>
  );
}
```

b. **Tailwind Merge**:

```
import { twMerge } from 'tailwind-merge';

function Button({ className, ...props }) {
  // Base classes can be overridden by className prop
  return (
    <button
      className={twMerge(
        'px-4 py-2 bg-blue-500 text-white rounded',
```

```
        className
      )}
      {...props}
    />
  );
}
```

4. **Next.js with Tailwind**:

```
npx create-next-app --example with-tailwindcss my-app
```

Or manual setup:

```
npx create-next-app my-app
cd my-app
npm install -D tailwindcss postcss autoprefixer
npx tailwindcss init -p
```

```
// tailwind.config.js
module.exports = {
  content: [
    './pages/**/*.{js,ts,jsx,tsx}',
    './components/**/*.{js,ts,jsx,tsx}',
  ],
  theme: {
    extend: {},
  },
  plugins: [],
};
```

```
/* styles/globals.css */
@tailwind base;
@tailwind components;
@tailwind utilities;
```

```
// pages/_app.js
import '../styles/globals.css';

function MyApp({ Component, pageProps }) {
  return <Component {...pageProps} />;
}

export default MyApp;
```

Vue and Tailwind CSS

1. **Setup with Vue CLI**:

```
vue create my-app
cd my-app
vue add tailwind
```

Or manual setup:

```
vue create my-app
cd my-app
npm install -D tailwindcss postcss autoprefixer
npx tailwindcss init -p
```

```
// tailwind.config.js
module.exports = {
  content: ['./src/**/*.{vue,js,ts,jsx,tsx}'],
  theme: {
    extend: {},
  },
  plugins: [],
};
```

```
/* src/assets/tailwind.css */
@tailwind base;
@tailwind components;
@tailwind utilities;
```

```
// src/main.js
import { createApp } from 'vue';
import App from './App.vue';
import './assets/tailwind.css';

createApp(App).mount('#app');
```

2. **Vue Component Example**:

```
<!-- src/components/Button.vue -->
<template>
  <button
    :class="[
      'px-4 py-2 rounded font-semibold focus:outline-none focus:ring-2
focus:ring-offset-2',
      primary
        ? 'bg-blue-500 text-white hover:bg-blue-600 focus:ring-blue-500'
        : 'bg-gray-200 text-gray-800 hover:bg-gray-300 focus:ring-gray-500',
    ]"
  >
    <slot></slot>
  </button>
</template>

<script>
export default {
  props: {
    primary: {
      type: Boolean,
      default: false,
    },
  },
};
</script>
```

Usage:

```
<!-- src/App.vue -->
<template>
  <div class="p-8">
    <h1 class="text-3xl font-bold mb-6">Vue with Tailwind</h1>
    <div class="space-x-4">
      <Button primary>Primary Button</Button>
      <Button>Secondary Button</Button>
    </div>
  </div>
</template>

<script>
import Button from './components/Button.vue';

export default {
  components: {
    Button,
  },
};
</script>
```

3. **Nuxt.js with Tailwind**:

```
npx create-nuxt-app my-app
cd my-app
npm install -D @nuxtjs/tailwindcss
```

```
// nuxt.config.js
export default {
  buildModules: ['@nuxtjs/tailwindcss'],
  tailwindcss: {
    // Options
  },
};
```

4. **Vue 3 Composition API Example**:

```
<!-- ButtonWithState.vue -->
<template>
  <button :class="buttonClasses" @click="toggle">
    {{ active ? 'Active' : 'Inactive' }}
  </button>
</template>

<script setup>
import { ref, computed } from 'vue';

const active = ref(false);

const buttonClasses = computed(() => {
  return [
    'px-4 py-2 rounded transition-colors',
    active.value ? 'bg-green-500 text-white' : 'bg-gray-200 text-gray-800',
  ];
});

const toggle = () => {
  active.value = !active.value;
};
</script>
```

Angular and Tailwind CSS

1. **Setup with Angular CLI**:

```
ng new my-app
cd my-app
```

```
npm install -D tailwindcss postcss autoprefixer
npx tailwindcss init
```

```
// tailwind.config.js
module.exports = {
  content: ['./src/**/*.{html,ts}'],
  theme: {
    extend: {},
  },
  plugins: [],
};
```

```
/* src/styles.css */
@tailwind base;
@tailwind components;
@tailwind utilities;
```

Update your build configuration:

```
// angular.json
{
  "projects": {
    "my-app": {
      "architect": {
        "build": {
          "builder": "@angular-devkit/build-angular:browser",
          "options": {
            "styles": ["src/styles.css"]
          }
        }
      }
    }
  }
}
```

2. **Angular Component Example**:

```
// src/app/button/button.component.ts
import { Component, Input } from '@angular/core';

@Component({
  selector: 'app-button',
  template: `
    <button
      [ngClass]="[
```

```
        'px-4 py-2 rounded font-semibold focus:outline-none focus:ring-2
focus:ring-offset-2',
        primary
          ? 'bg-blue-500 text-white hover:bg-blue-600 focus:ring-blue-500'
          : 'bg-gray-200 text-gray-800 hover:bg-gray-300 focus:ring-gray-
500'
      ]"
    >
      <ng-content></ng-content>
    </button>
  `,
})
export class ButtonComponent {
  @Input() primary = false;
}
```

```
// src/app/app.component.html
<div class="p-8">
  <h1 class="text-3xl font-bold mb-6">Angular with Tailwind</h1>
  <div class="space-x-4">
    <app-button [primary]="true">Primary Button</app-button>
    <app-button>Secondary Button</app-button>
  </div>
</div>
```

3. **Using Tailwind with Angular Components**:

```
// Component with dynamic classes
import { Component, Input } from '@angular/core';

@Component({
  selector: 'app-card',
  template: `
    <div class="rounded overflow-hidden shadow-lg">
      <div [ngClass]="['p-6', colorClass]">
        <h2 class="text-xl font-semibold">{{ title }}</h2>
        <p class="mt-2">{{ content }}</p>
      </div>
    </div>
  `,
})
export class CardComponent {
  @Input() title = '';
  @Input() content = '';
  @Input() color = 'default';

  get colorClass(): string {
    const colorMap: { [key: string]: string } = {
      default: 'bg-white',
```

```
      primary: 'bg-blue-100',
      success: 'bg-green-100',
      warning: 'bg-yellow-100',
      danger: 'bg-red-100',
    };

    return colorMap[this.color] || colorMap.default;
  }
}
```

Laravel and Tailwind CSS

1. **Setup with Laravel**:

```
composer create-project laravel/laravel my-app
cd my-app
npm install -D tailwindcss postcss autoprefixer
npx tailwindcss init -p
```

```
// tailwind.config.js
module.exports = {
  content: [
    './resources/**/*.blade.php',
    './resources/**/*.js',
    './resources/**/*.vue',
  ],
  theme: {
    extend: {},
  },
  plugins: [],
};
```

```
/* resources/css/app.css */
@tailwind base;
@tailwind components;
@tailwind utilities;
```

Update webpack.mix.js:

```
// webpack.mix.js
const mix = require('laravel-mix');

mix
```

```
    .js('resources/js/app.js', 'public/js')
    .postCss('resources/css/app.css', 'public/css', [require('tailwindcss')]);
```

2. **Blade Component Example**:

```
<!-- resources/views/components/button.blade.php -->
@props([
    'primary' => false,
])

@php
$classes = 'px-4 py-2 rounded font-semibold focus:outline-none focus:ring-2
focus:ring-offset-2';
$classes .= $primary
    ? ' bg-blue-500 text-white hover:bg-blue-600 focus:ring-blue-500'
    : ' bg-gray-200 text-gray-800 hover:bg-gray-300 focus:ring-gray-500';
@endphp

<button {{ $attributes->merge(['class' => $classes]) }}>
    {{ $slot }}
</button>
```

Usage:

```
<!-- resources/views/welcome.blade.php -->
<div class="p-8">
    <h1 class="text-3xl font-bold mb-6">Laravel with Tailwind</h1>
    <div class="space-x-4">
        <x-button primary>Primary Button</x-button>
        <x-button>Secondary Button</x-button>
    </div>
</div>
```

3. **TALL Stack** (Tailwind, Alpine.js, Laravel, Livewire):

```
# Install dependencies
composer require livewire/livewire
npm install alpinejs
```

```
// resources/js/app.js
import Alpine from 'alpinejs';
window.Alpine = Alpine;
Alpine.start();
```

Example component:

```
<!-- resources/views/livewire/counter.blade.php -->
<div class="p-6 bg-white rounded shadow">
    <h2 class="text-xl font-semibold mb-4">Counter Example</h2>

    <div class="flex items-center space-x-4">
        <button wire:click="decrement" class="px-4 py-2 bg-red-500 text-
white rounded">
            -
        </button>

        <span x-data="{ count: @entangle('count') }" x-text="count"
class="text-2xl"></span>

        <button wire:click="increment" class="px-4 py-2 bg-green-500 text-
white rounded">
            +
        </button>
    </div>
</div>
```

Other Framework Integrations

1. **Svelte and Tailwind**:

```
npx degit sveltejs/template my-app
cd my-app
npm install -D tailwindcss postcss autoprefixer
npx tailwindcss init -p
```

```
// tailwind.config.js
module.exports = {
  content: ['./src/**/*.{html,js,svelte,ts}'],
  theme: {
    extend: {},
  },
  plugins: [],
};
```

```
/* src/app.css */
@tailwind base;
@tailwind components;
@tailwind utilities;
```

```
// rollup.config.js (add PostCSS)
// ...
plugins: [
  svelte({
    // ...
  }),
  // ...
  postcss({
    extract: 'bundle.css',
    plugins: [require('tailwindcss'), require('autoprefixer')],
  }),
  // ...
];
```

2. **Express/Node.js and Tailwind**:

```
mkdir my-app && cd my-app
npm init -y
npm install express ejs
npm install -D tailwindcss postcss autoprefixer postcss-cli
npx tailwindcss init -p
```

```
// tailwind.config.js
module.exports = {
  content: ['./views/**/*.ejs'],
  theme: {
    extend: {},
  },
  plugins: [],
};
```

```
/* public/css/input.css */
@tailwind base;
@tailwind components;
@tailwind utilities;
```

Add build script to package.json:

```
"scripts": {
  "build:css": "postcss public/css/input.css -o public/css/styles.css",
  "watch:css": "postcss public/css/input.css -o public/css/styles.css --
watch"
}
```

```
// app.js
const express = require('express');
const app = express();

app.set('view engine', 'ejs');
app.use(express.static('public'));

app.get('/', (req, res) => {
  res.render('index');
});

app.listen(3000, () => {
  console.log('Server running on port 3000');
});
```

```
<!-- views/index.ejs -->
<!DOCTYPE html>
<html>
  <head>
    <title>Express with Tailwind</title>
    <link rel="stylesheet" href="/css/styles.css" />
  </head>
  <body class="bg-gray-100">
    <div class="container mx-auto p-8">
      <h1 class="text-3xl font-bold text-center mb-8">
        Express with Tailwind CSS
      </h1>
      <!-- Content here -->
    </div>
  </body>
</html>
```

3. **Gatsby and Tailwind**:

```
npm install -g gatsby-cli
gatsby new my-app
cd my-app
npm install -D tailwindcss postcss autoprefixer gatsby-plugin-postcss
npx tailwindcss init -p
```

```
// gatsby-config.js
module.exports = {
  plugins: [
    'gatsby-plugin-postcss',
    // other plugins...
```

```
  ],
};
```

```
// tailwind.config.js
module.exports = {
  content: ['./src/**/*.{js,jsx,ts,tsx}'],
  theme: {
    extend: {},
  },
  plugins: [],
};
```

```
/* src/styles/global.css */
@tailwind base;
@tailwind components;
@tailwind utilities;
```

```
// gatsby-browser.js
import './src/styles/global.css';
```

Production Optimization

Optimizing CSS Size

1. **Content Configuration**:

   Be specific with your content paths to avoid scanning unnecessary files:

```
// tailwind.config.js
module.exports = {
  content: [
    './src/pages/**/*.{js,jsx,ts,tsx}',
    './src/components/**/*.{js,jsx,ts,tsx}',
    './public/**/*.html',
  ],
  // ...
};
```

2. **Minification**:

   Enable minification for production builds:

```
# Using Tailwind CLI
npx tailwindcss -i input.css -o output.css --minify
```

```
// PostCSS config with cssnano
// postcss.config.js
module.exports = {
  plugins: {
    tailwindcss: {},
    autoprefixer: {},
    ...(process.env.NODE_ENV === 'production' ? { cssnano: {} } : {}),
  },
};
```

3. **Removing Unnecessary Features**:

   Disable features you don't use:

```
// tailwind.config.js
module.exports = {
  corePlugins: {
    float: false, // Disable if not using float utilities
    objectFit: false, // Disable if not using object-fit utilities
    objectPosition: false, // Disable if not using object-position utilities
  },
  // ...
};
```

4. **Advanced PurgeCSS Options**:

   Configure safelist for dynamically generated classes:

```
// tailwind.config.js
module.exports = {
  content: ['./src/**/*.{html,js}'],
  safelist: [
    'bg-red-500',
    'text-center',
    'lg:text-right',
    {
      pattern: /bg-(red|green|blue)-(100|200|300|400|500)/,
      variants: ['lg', 'hover', 'focus', 'lg:hover'],
    },
  ],
  // ...
};
```

Build Process Optimization

1. **Separate Development and Production Configs**:

```
// tailwind.config.js
const defaultTheme = require('tailwindcss/defaultTheme');

module.exports = {
  content: ['./src/**/*.{html,js}'],
  theme: {
    extend: {
      fontFamily: {
        sans: ['Inter var', ...defaultTheme.fontFamily.sans],
      },
    },
  },
  plugins: [
    require('@tailwindcss/forms'),
    require('@tailwindcss/typography'),
  ],
  ...(process.env.NODE_ENV === 'production'
    ? {
        // Production-specific settings
        future: {
          removeDeprecatedGapUtilities: true,
          purgeLayersByDefault: true,
        },
      }
    : {
        // Development-specific settings
      }),
};
```

2. **Using Modern Build Tools**:

   a. **Webpack**:

```
// webpack.config.js
module.exports = {
  // ...
  module: {
    rules: [
      {
        test: /\.css$/,
        use: [
          'style-loader',
          { loader: 'css-loader', options: { importLoaders: 1 } },
          'postcss-loader',
        ],
      },
```

```
    ],
  },
  // ...
};
```

b. **Vite**:

```
// vite.config.js
import { defineConfig } from 'vite';
import react from '@vitejs/plugin-react';

export default defineConfig({
  plugins: [react()],
  css: {
    postcss: './postcss.config.js',
  },
});
```

3. **Cache Optimization**:

   Use hashed filenames for better caching:

```
// webpack.config.js
module.exports = {
  output: {
    filename: '[name].[contenthash].js',
    chunkFilename: '[name].[contenthash].js',
    assetModuleFilename: 'assets/[hash][ext][query]',
    path: path.resolve(__dirname, 'dist'),
    clean: true,
  },
  // ...
};
```

Performance Considerations

1. **Split CSS by Media Queries**:

```
// postcss.config.js
module.exports = {
  plugins: [
    require('tailwindcss'),
    require('autoprefixer'),
    process.env.NODE_ENV === 'production' &&
      require('@fullhuman/postcss-purgecss')({
        content: ['./src/**/*.{js,jsx,ts,tsx,html}'],
        defaultExtractor: (content) =>
```

```
            content.match(/[\w-/:]+(?<!:)/g) || [],
        }),
      require('postcss-extract-media-query')({
        output: {
          path: './dist/css',
          name: '[name]-[query].[ext]',
        },
        queries: {
          'screen and (min-width: 768px)': 'tablet',
          'screen and (min-width: 1024px)': 'desktop',
        },
      }),
    ].filter(Boolean),
};
```

2. **Critical CSS Extraction**:

```
// Using critical npm package
const critical = require('critical');

critical.generate({
  inline: true,
  base: 'dist/',
  src: 'index.html',
  target: {
    html: 'index-critical.html',
    css: 'critical.css',
  },
  width: 1300,
  height: 900,
});
```

3. **Preloading CSS**:

```
<link rel="preload" href="/css/styles.css" as="style" />
<link rel="stylesheet" href="/css/styles.css" />
```

Deployment Best Practices

1. **CDN Integration**:

```
<!-- Using a CDN for delivering assets -->
<link rel="stylesheet" href="https://cdn.example.com/css/styles.css" />
```

2. **HTTP/2 and HTTP/3**:

Configure your server for HTTP/2 or HTTP/3 to improve performance:

```
# Nginx configuration for HTTP/2
server {
    listen 443 ssl http2;
    server_name example.com;

    # SSL configuration
    ssl_certificate /path/to/cert.pem;
    ssl_certificate_key /path/to/key.pem;

    # Other configuration...
}
```

3. **Content Compression**:

```
# Nginx configuration for Gzip compression
server {
    # Other configuration...

    gzip on;
    gzip_comp_level 6;
    gzip_types text/plain text/css application/json application/javascript
text/xml application/xml application/xml+rss text/javascript;

    # Other configuration...
}
```

Browser Support and Polyfills

1. **Autoprefixer Configuration**:

```
// postcss.config.js
module.exports = {
  plugins: [
    require('tailwindcss'),
    require('autoprefixer')({
      overrideBrowserslist: [
        '> 1%',
        'last 2 versions',
        'Firefox ESR',
        'not dead',
      ],
    }),
  ],
};
```

2. **Feature Queries**:

   Use `@supports` for progressive enhancement:

```
/* Base styles for all browsers */
.card {
  display: block;
  margin: 1rem;
}

/* Enhanced styles for browsers that support grid */
@supports (display: grid) {
  .card-container {
    display: grid;
    grid-template-columns: repeat(auto-fill, minmax(250px, 1fr));
    gap: 1rem;
  }

  .card {
    margin: 0;
  }
}
```

3. **Browser-Specific CSS**:

   Handle IE11 or older browsers:

```
/* IE11 specific fix */
@media all and (-ms-high-contrast: none), (-ms-high-contrast: active) {
  .flex-container {
    display: flex;
    flex-wrap: wrap;
  }

  .flex-item {
    flex: 0 0 calc(33.333% - 1rem);
    margin: 0.5rem;
  }
}
```

Monitoring and Analytics

1. **Performance Metrics**:

   Track Core Web Vitals and other performance metrics:

```
// Using web-vitals library
import { getCLS, getFID, getLCP } from 'web-vitals';
```

```
function sendToAnalytics({ name, delta, id }) {
  // Send metrics to your analytics service
  console.log({ name, delta, id });
}

getCLS(sendToAnalytics);
getFID(sendToAnalytics);
getLCP(sendToAnalytics);
```

2. **Lighthouse Integration**:

   Automate Lighthouse tests in your CI/CD pipeline:

```
// Using lighthouse-ci
// .lighthouserc.js
module.exports = {
  ci: {
    collect: {
      url: ['http://localhost:8080/'],
      numberOfRuns: 3,
    },
    upload: {
      target: 'temporary-public-storage',
    },
    assert: {
      preset: 'lighthouse:recommended',
      assertions: {
        'categories:performance': ['warn', { minScore: 0.9 }],
        'first-contentful-paint': ['warn', { maxNumericValue: 2000 }],
        interactive: ['warn', { maxNumericValue: 3500 }],
      },
    },
  },
};
```

Migration from CSS to Tailwind

Planning Your Migration

1. **Assessment and Inventory**:
   - Audit existing CSS files and patterns
   - Identify common components and styles
   - Document design tokens (colors, spacing, typography, etc.)
   - Determine which parts to migrate first

2. **Setting Migration Goals**

- Set clear objectives for the migration

- Decide between incremental or full migration
- Create a timeline and prioritize components
- Establish coding standards for the new Tailwind implementation

3. **Team Preparation**:
   - Train team members on Tailwind CSS basics
   - Establish shared understanding of utility-first approach
   - Create documentation for common patterns
   - Set up pair programming for knowledge sharing

Incremental Migration Strategies

1. **Parallel Approaches**:
   - Keep existing CSS while gradually adding Tailwind
   - Use namespacing to avoid conflicts

```
<!-- Original component -->
<div class="card">
  <h2 class="card-title">Title</h2>
  <p class="card-text">Content</p>
</div>

<!-- Migrated component with Tailwind -->
<div class="tw-bg-white tw-rounded-lg tw-shadow-md tw-p-6">
  <h2 class="tw-text-xl tw-font-bold tw-mb-2">Title</h2>
  <p class="tw-text-gray-700">Content</p>
</div>
```

```
/* tailwind.config.js */
module.exports = {
  prefix: 'tw-',
    // Add prefix to all Tailwind classes
    // ...
;
}
```

2. **Component-by-Component Migration**:
   - Identify isolated components to migrate first
   - Create a list of components ordered by dependencies
   - Start with leaf components (those with no dependencies)
   - Document each migrated component

   Example migration plan:

   1. Buttons and form elements
   2. Cards and media objects

3. Navigation components
4. Layout containers
5. Page templates

3. **New Features with Tailwind**:
   - Use Tailwind for all new features
   - Leave legacy code untouched initially
   - Create migration backlog for old components

4. **CSS-in-JS to Tailwind**:
   - Replace styled-components or Emotion with Tailwind classes

```
// Before: styled-components
const Button = styled.button`
  padding: 0.5rem 1rem;
  background-color: #3b82f6;
  color: white;
  border-radius: 0.25rem;
  font-weight: 600;

  &:hover {
    background-color: #2563eb;
  }
`;

// After: Tailwind
function Button({ children, ...props }) {
  return (
    <button
      className="px-4 py-2 bg-blue-500 hover:bg-blue-600 text-white rounded
font-semibold"
      {...props}
    >
      {children}
    </button>
  );
}
```

5. **SCSS/LESS to Tailwind**:
   - Analyze nested selectors and convert to component extraction
   - Replace variables with Tailwind theme values
   - Convert mixins to Tailwind utilities

```
// Before: SCSS
.card {
  background-color: $white;
  border-radius: $border-radius;
```

```
  box-shadow: $shadow-md;
  padding: $spacing-6;

  .card-title {
    font-size: $font-lg;
    font-weight: $font-bold;
    margin-bottom: $spacing-2;
  }

  .card-content {
    color: $gray-700;
  }

  &:hover {
    box-shadow: $shadow-lg;
  }
}

// After: Tailwind
<div class="bg-white rounded-lg shadow-md p-6 hover:shadow-lg">
  <h2 class="text-lg font-bold mb-2">Card Title</h2>
  <div class="text-gray-700">Card Content</div>
</div>
```

Tools and Techniques for Migration

1. **CSS-to-Tailwind Converters**:
   - Use tools like CSS to Tailwind or Windy
   - Verify and clean up automated conversions
2. **CSS Analysis Tools**:
   - Use tools like Wallace to analyze CSS complexity
   - Run PurgeCSS to identify unused styles
3. **Customizing Tailwind to Match Current Design System**:

```
// tailwind.config.js
module.exports = {
  theme: {
    extend: {
      colors: {
        // Map your current color system to Tailwind
        'brand-primary': '#0073e6',
        'brand-secondary': '#ff6b6b',
        'brand-accent': '#46c93a',
        // ...
      },
      spacing: {
        // Map your current spacing system
```

```
        xs: '0.25rem',
        sm: '0.5rem',
        md: '1rem',
        lg: '1.5rem',
        xl: '2rem',
        // ...
      },
      borderRadius: {
        // Match your current border radiuses
        button: '0.25rem',
        card: '0.5rem',
        pill: '9999px',
      },
      // ...other theme extensions
    },
  },
};
```

4. **Hybrid Approach with `@apply`**:
   - Use `@apply` as a bridge during migration

```
/* Before */
.btn-primary {
  background-color: #3b82f6;
  color: white;
  padding: 0.5rem 1rem;
  border-radius: 0.25rem;
  font-weight: 600;
}

.btn-primary:hover {
  background-color: #2563eb;
}

/* During migration */
.btn-primary {
  @apply bg-blue-500 text-white px-4 py-2 rounded font-semibold;

  &:hover {
    @apply bg-blue-600;
  }
}

/* Final goal: direct Tailwind classes in HTML */
<button class="bg-blue-500 hover:bg-blue-600 text-white px-4 py-2 rounded
font-semibold">
  Button
</button>
```

5. **Class Mapping Documentation**: Create a reference document that maps old CSS classes to new Tailwind equivalents:

Legacy Class	Tailwind Equivalent	Notes
`.btn`	`px-4 py-2 rounded`	Basic button styles
`.btn-primary`	`bg-blue-500 text-white hover:bg-blue-600`	Primary button variant
`.card`	`bg-white rounded-lg shadow-md p-6`	Card container
`.container`	`max-w-6xl mx-auto px-4`	Content container

Common Challenges and Solutions

1. **Handling Complex Selectors**:

   Challenge: CSS with complex nesting and specificity.

   Solution: Decompose into smaller components with targeted classes.

```
/* Before: Complex CSS */
.sidebar .nav-menu > ul > li.active > a {
  color: #3b82f6;
  font-weight: bold;
}
```

```
<!-- After: Tailwind with component extraction -->
<nav class="sidebar">
  <ul class="nav-menu">
    <li class="nav-item">
      <a class="nav-link" href="#">Link</a>
    </li>
    <li class="nav-item">
      <a class="nav-link active" href="#">Active Link</a>
    </li>
  </ul>
</nav>
```

```
// Component extraction
function NavLink({ children, active, ...props }) {
  return (
    <a
      className={`block py-2 px-4 ${
        active
          ? 'text-blue-500 font-bold'
          : 'text-gray-700 hover:text-gray-900'
      }`}
```

```
      {...props}
    >
      {children}
    </a>
  );
}
```

2. **Global Styles and Resets**:

   Challenge: Migrating global styles and CSS resets.

   Solution: Use Tailwind's base styles or create custom base styles.

```
/* tailwind.config.js */
module.exports = {
  plugins: [ require('@tailwindcss/typography'), ];
}
```

```
/* In your CSS */
@layer base {
  h1 {
    @apply text-2xl font-bold mb-4;
  }

  h2 {
    @apply text-xl font-semibold mb-3;
  }

  a {
    @apply text-blue-600 hover:text-blue-800 hover:underline;
  }
}
```

3. **Third-Party Components**:

   Challenge: Integrating third-party components with existing CSS.

   Solution: Wrap third-party components and override styles systematically.

```
// Wrapping a third-party component with Tailwind classes
function CustomDatePicker(props) {
  return (
    <div className="tailwind-datepicker-wrapper">
      <DatePicker
        className="block w-full px-3 py-2 border border-gray-300 rounded-md
shadow-sm focus:outline-none focus:ring-blue-500 focus:border-blue-500"
        {...props}
      />
```

```
    </div>
  );
}
```

```
/* Additional CSS to handle third-party styling conflicts */
.tailwind-datepicker-wrapper .react-datepicker-wrapper {
  @apply block w-full;
}
```

4. **CSS Animations and Transitions**:

   Challenge: Complex animations in CSS are hard to convert to utilities.

   Solution: Keep animations in CSS and combine with Tailwind classes.

```
/* Keep complex animations in CSS */
@keyframes fadeInUp {
  from {
    opacity: 0;
    transform: translateY(1rem);
  }
  to {
    opacity: 1;
    transform: translateY(0);
  }
}

.animate-fade-in-up {
  animation: fadeInUp 0.5s ease-out forwards;
}
```

```
<!-- Use with Tailwind classes -->
<div class="bg-white shadow-lg rounded-lg p-6 animate-fade-in-up">
  Animated content
</div>
```

5. **Media Queries and Responsive Design**:

   Challenge: Converting complex media query patterns.

   Solution: Use Tailwind's responsive prefixes and custom screens.

```
/* Before: Custom media queries */
@media (min-width: 768px) and (max-width: 1023px) and (orientation:
landscape) {
```

```
  .special-element {
    width: 50%;
    padding: 1.5rem;
  }
}
```

```
// tailwind.config.js
module.exports = {
  theme: {
    screens: {
      'md-landscape': {
        raw: '(min-width: 768px) and (max-width: 1023px) and (orientation:
landscape)',
      },
      // ... other screens
    },
  },
};
```

```
<!-- Using custom responsive prefix -->
<div class="w-full md-landscape:w-1/2 p-4 md-landscape:p-6">
  Responsive content
</div>
```

Measuring Migration Success

1. **Performance Metrics**:
   - Compare bundle size before and after migration
   - Measure Core Web Vitals improvements
   - Track time-to-interactive changes
2. **Code Quality Metrics**:
   - Measure reduction in CSS complexity
   - Count CSS selectors before and after
   - Track specificity issues resolved
3. **Developer Experience**:
   - Survey team on development speed
   - Measure time to implement new features
   - Track bugs related to styling
4. **Migration Progress Tracking**:
   - Percentage of components migrated

- Legacy CSS file size reduction
- Components left in backlog

Advanced Topics

Building Complex UI Components with CSS/Tailwind

Design System Principles

1. **Tokens vs. Components**:

   Design tokens are the foundational values of your design system:

```
/* CSS Custom Properties as Design Tokens */
:root {
  /* Colors */
  --color-primary: #3b82f6;
  --color-primary-light: #60a5fa;
  --color-primary-dark: #2563eb;

  /* Spacing */
  --spacing-xs: 0.25rem;
  --spacing-sm: 0.5rem;
  --spacing-md: 1rem;
  --spacing-lg: 2rem;
  --spacing-xl: 4rem;

  /* Typography */
  --font-size-xs: 0.75rem;
  --font-size-sm: 0.875rem;
  --font-size-md: 1rem;
  --font-size-lg: 1.125rem;
  --font-size-xl: 1.25rem;

  /* Borders */
  --border-radius-sm: 0.125rem;
  --border-radius-md: 0.25rem;
  --border-radius-lg: 0.5rem;
  --border-radius-full: 9999px;
}
```

   With Tailwind:

```
// tailwind.config.js
module.exports = {
  theme: {
    extend: {
      colors: {
        primary: {
          DEFAULT: '#3b82f6',
```

```
          light: '#60a5fa',
          dark: '#2563eb',
        },
      },
      spacing: {
        xs: '0.25rem',
        sm: '0.5rem',
        md: '1rem',
        lg: '2rem',
        xl: '4rem',
      },
      fontSize: {
        xs: '0.75rem',
        sm: '0.875rem',
        md: '1rem',
        lg: '1.125rem',
        xl: '1.25rem',
      },
      borderRadius: {
        sm: '0.125rem',
        md: '0.25rem',
        lg: '0.5rem',
        full: '9999px',
      },
    },
  },
};
```

2. **Atomic Components**:

   Build small, single-purpose components:

```
// Avatar.jsx
function Avatar({ src, alt, size = 'md' }) {
  const sizeClasses = {
    sm: 'w-8 h-8',
    md: 'w-12 h-12',
    lg: 'w-16 h-16',
  };

  return (
    <img
      src={src}
      alt={alt}
      className={`${sizeClasses[size]} rounded-full object-cover`}
    />
  );
}

// Badge.jsx
function Badge({ children, variant = 'default' }) {
  const variantClasses = {
```

```
    default: 'bg-gray-100 text-gray-800',
    primary: 'bg-blue-100 text-blue-800',
    success: 'bg-green-100 text-green-800',
    warning: 'bg-yellow-100 text-yellow-800',
    danger: 'bg-red-100 text-red-800',
  };

  return (
    <span
      className={`inline-flex items-center px-2.5 py-0.5 rounded-full text-
xs font-medium ${variantClasses[variant]}`}
    >
      {children}
    </span>
  );
}
```

3. **Compound Components**:

   Create more complex components from atomic ones:

```
// UserCard.jsx
function UserCard({ user, actions }) {
  return (
    <div className="bg-white rounded-lg shadow-md p-6">
      <div className="flex items-center space-x-4">
        <Avatar src={user.avatar} alt={user.name} />

        <div>
          <h3 className="text-lg font-semibold">{user.name}</h3>
          <p className="text-gray-500">{user.title}</p>

          <div className="flex mt-2 space-x-2">
            {user.tags.map((tag) => (
              <Badge
                key={tag}
                variant={tag === 'admin' ? 'primary' : 'default'}
              >
                {tag}
              </Badge>
            ))}
          </div>
        </div>
      </div>

      {actions && (
        <div className="mt-4 flex justify-end space-x-2">{actions}</div>
      )}
    </div>
  );
}
```

```
// Usage
<UserCard
  user={{
    name: 'Jane Smith',
    title: 'Product Designer',
    avatar: '/images/avatar.jpg',
    tags: ['admin', 'design'],
  }}
  actions={
    <>
      <Button variant="outline">Message</Button>
      <Button>View Profile</Button>
    </>
  }
/>;
```

Complex UI Patterns

1. **Multi-Level Navigation**:

```
// Dropdown.jsx
function Dropdown({ trigger, items }) {
  const [isOpen, setIsOpen] = useState(false);

  return (
    <div className="relative">
      {/* Trigger button */}
      <button
        className="flex items-center space-x-1"
        onClick={() => setIsOpen(!isOpen)}
      >
        <span>{trigger}</span>
        <svg
          className={`w-4 h-4 transition-transform ${
            isOpen ? 'rotate-180' : ''
          }`}
          viewBox="0 0 20 20"
          fill="currentColor"
        >
          <path
            fillRule="evenodd"
            d="M5.293 7.293a1 1 0 011.414 0L10 10.586l3.293-3.293a1 1 0
111.414 1.414l-4 4a1 1 0 01-1.414 0l-4-4a1 1 0 010-1.414z"
            clipRule="evenodd"
          />
        </svg>
      </button>

      {/* Dropdown menu */}
      {isOpen && (
        <div className="absolute right-0 mt-2 w-48 bg-white rounded-md
```

```
shadow-lg py-1 z-10">
          {items.map((item, index) => (
            <a
              key={index}
              href={item.href}
              className="block px-4 py-2 text-sm text-gray-700 hover:bg-
gray-100"
            >
              {item.label}
            </a>
          ))}
        </div>
      )}
    </div>
  );
}

// Multi-level Navigation
function Navigation() {
  return (
    <nav className="bg-white shadow">
      <div className="max-w-7xl mx-auto px-4">
        <div className="flex justify-between h-16">
          {/* Logo */}
          <div className="flex items-center">
            <a href="/" className="text-xl font-bold">
              Logo
            </a>
          </div>

          {/* Navigation items */}
          <div className="hidden md:flex items-center space-x-4">
            <a
              href="/"
              className="px-3 py-2 text-gray-700 hover:text-gray-900"
            >
              Home
            </a>

            <Dropdown
              trigger="Products"
              items={[
                { label: 'Category 1', href: '/category-1' },
                { label: 'Category 2', href: '/category-2' },
                { label: 'Category 3', href: '/category-3' },
              ]}
            />

            <Dropdown
              trigger="Services"
              items={[
                { label: 'Service 1', href: '/service-1' },
                { label: 'Service 2', href: '/service-2' },
                { label: 'Service 3', href: '/service-3' },
```

```
              ]}
            />

            <a
              href="/about"
              className="px-3 py-2 text-gray-700 hover:text-gray-900"
            >
              About
            </a>
            <a
              href="/contact"
              className="px-3 py-2 text-gray-700 hover:text-gray-900"
            >
              Contact
            </a>
          </div>

          {/* Mobile menu button */}
          <div className="md:hidden flex items-center">
            <button className="text-gray-500 hover:text-gray-700">
              <svg
                className="h-6 w-6"
                fill="none"
                viewBox="0 0 24 24"
                stroke="currentColor"
              >
                <path
                  strokeLinecap="round"
                  strokeLinejoin="round"
                  strokeWidth={2}
                  d="M4 6h16M4 12h16M4 18h16"
                />
              </svg>
            </button>
          </div>
        </div>
      </div>

      {/* Mobile menu (hidden by default) */}
      {/* ... Mobile menu implementation ... */}
    </nav>
  );
}
```

2. **Data Tables with Sorting, Filtering, and Pagination**:

```
function DataTable({ data, columns }) {
  const [sortColumn, setSortColumn] = useState(null);
  const [sortDirection, setSortDirection] = useState('asc');
  const [page, setPage] = useState(1);
  const [rowsPerPage, setRowsPerPage] = useState(10);
  const [filters, setFilters] = useState({});
```

```
  // Apply sorting
  const sortedData = React.useMemo(() => {
    if (!sortColumn) return data;

    return [...data].sort((a, b) => {
      const aValue = a[sortColumn];
      const bValue = b[sortColumn];

      if (aValue < bValue) return sortDirection === 'asc' ? -1 : 1;
      if (aValue > bValue) return sortDirection === 'asc' ? 1 : -1;
      return 0;
    });
  }, [data, sortColumn, sortDirection]);

  // Apply filtering
  const filteredData = React.useMemo(() => {
    return sortedData.filter((row) => {
      return Object.entries(filters).every(([key, value]) => {
        if (!value) return true;
        return String(row[key]).toLowerCase().includes(value.toLowerCase());
      });
    });
  }, [sortedData, filters]);

  // Apply pagination
  const paginatedData = React.useMemo(() => {
    const startIndex = (page - 1) * rowsPerPage;
    return filteredData.slice(startIndex, startIndex + rowsPerPage);
  }, [filteredData, page, rowsPerPage]);

  // Handle sorting
  const handleSort = (column) => {
    if (sortColumn === column) {
      setSortDirection(sortDirection === 'asc' ? 'desc' : 'asc');
    } else {
      setSortColumn(column);
      setSortDirection('asc');
    }
  };

  // Handle filtering
  const handleFilterChange = (column, value) => {
    setFilters((prev) => ({
      ...prev,
      [column]: value,
    }));
    setPage(1); // Reset to first page on filter change
  };

  return (
    <div className="bg-white shadow-md rounded-lg overflow-hidden">
      {/* Filter inputs */}
      <div className="p-4 bg-gray-50 border-b flex flex-wrap gap-4">
```

```
        {columns.map((column) => (
          <div key={`filter-${column.key}`} className="flex flex-col">
            <label className="text-sm font-medium text-gray-700">
              {column.label}
            </label>
            <input
              type="text"
              className="mt-1 block w-full rounded-md border-gray-300
shadow-sm focus:border-blue-500 focus:ring-blue-500"
              placeholder={`Filter ${column.label}`}
              value={filters[column.key] || ''}
              onChange={(e) =>
                handleFilterChange(column.key, e.target.value)
              }
            />
          </div>
        ))}
      </div>

      {/* Table */}
      <div className="overflow-x-auto">
        <table className="min-w-full divide-y divide-gray-200">
          <thead className="bg-gray-50">
            <tr>
              {columns.map((column) => (
                <th
                  key={column.key}
                  scope="col"
                  className="px-6 py-3 text-left text-xs font-medium text-
gray-500 uppercase tracking-wider cursor-pointer"
                  onClick={() => handleSort(column.key)}
                >
                  <div className="flex items-center space-x-1">
                    <span>{column.label}</span>
                    {sortColumn === column.key && (
                      <svg
                        className="w-4 h-4"
                        viewBox="0 0 20 20"
                        fill="currentColor"
                      >
                        {sortDirection === 'asc' ? (
                          <path
                            fillRule="evenodd"
                            d="M5.293 7.707a1 1 0 010-1.414l4-4a1 1 0
011.414 0l4 4a1 1 0 01-1.414 1.414L10 4.414l-3.293 3.293a1 1 0 01-1.414 0z"
                            clipRule="evenodd"
                          />
                        ) : (
                          <path
                            fillRule="evenodd"
                            d="M14.707 12.293a1 1 0 010 1.414l-4 4a1 1 0 01-
1.414 0l-4-4a1 1 0 111.414-1.414L10 15.586l3.293-3.293a1 1 0 011.414 0z"
                            clipRule="evenodd"
                          />
```

```
                        )}
                      </svg>
                    )}
                  </div>
                </th>
              ))}
            </tr>
          </thead>
          <tbody className="bg-white divide-y divide-gray-200">
            {paginatedData.map((row, rowIndex) => (
              <tr key={rowIndex} className="hover:bg-gray-50">
                {columns.map((column) => (
                  <td
                    key={`${rowIndex}-${column.key}`}
                    className="px-6 py-4 whitespace-nowrap"
                  >
                    {column.render ? column.render(row) : row[column.key]}
                  </td>
                ))}
              </tr>
            ))}
          </tbody>
        </table>
      </div>

      {/* Pagination */}
      <div className="bg-white px-4 py-3 flex items-center justify-between
border-t border-gray-200 sm:px-6">
        <div className="flex-1 flex justify-between sm:hidden">
          <button
            className="relative inline-flex items-center px-4 py-2 border
border-gray-300 text-sm font-medium rounded-md text-gray-700 bg-white
hover:bg-gray-50"
            onClick={() => setPage(page > 1 ? page - 1 : 1)}
            disabled={page === 1}
          >
            Previous
          </button>
          <button
            className="ml-3 relative inline-flex items-center px-4 py-2
border border-gray-300 text-sm font-medium rounded-md text-gray-700 bg-white
hover:bg-gray-50"
            onClick={() =>
              setPage(
                page < Math.ceil(filteredData.length / rowsPerPage)
                  ? page + 1
                  : page
              )
            }
            disabled={page >= Math.ceil(filteredData.length / rowsPerPage)}
          >
            Next
          </button>
        </div>
```

```
        <div className="hidden sm:flex-1 sm:flex sm:items-center sm:justify-
between">
          <div>
            <p className="text-sm text-gray-700">
              Showing{' '}
              <span className="font-medium">
                {Math.min((page - 1) * rowsPerPage + 1,
filteredData.length)}
              </span>{' '}
              to{' '}
              <span className="font-medium">
                {Math.min(page * rowsPerPage, filteredData.length)}
              </span>{' '}
              of <span className="font-medium">{filteredData.length}</span>
{' '}
              results
            </p>
          </div>
          <div>
            <select
              className="mr-4 rounded-md border-gray-300 shadow-sm
focus:border-blue-500 focus:ring-blue-500"
              value={rowsPerPage}
              onChange={(e) => {
                setRowsPerPage(Number(e.target.value));
                setPage(1);
              }}
            >
              <option value={5}>5 per page</option>
              <option value={10}>10 per page</option>
              <option value={25}>25 per page</option>
              <option value={50}>50 per page</option>
            </select>

            <nav
              className="relative z-0 inline-flex rounded-md shadow-sm -
space-x-px"
              aria-label="Pagination"
            >
              <button
                className="relative inline-flex items-center px-2 py-2
rounded-l-md border border-gray-300 bg-white text-sm font-medium text-gray-
500 hover:bg-gray-50"
                onClick={() => setPage(page > 1 ? page - 1 : 1)}
                disabled={page === 1}
              >
                <span className="sr-only">Previous</span>
                <svg
                  className="h-5 w-5"
                  xmlns="http://www.w3.org/2000/svg"
                  viewBox="0 0 20 20"
                  fill="currentColor"
                  aria-hidden="true"
                >
```

```
                  <path
                    fillRule="evenodd"
                    d="M12.707 5.293a1 1 0 010 1.414L9.414 10l3.293 3.293a1
1 0 01-1.414 1.414l-4-4a1 1 0 010-1.414l4-4a1 1 0 011.414 0z"
                    clipRule="evenodd"
                  />
                </svg>
              </button>

              {/* Page numbers */}
              {Array.from({
                length: Math.min(
                  5,
                  Math.ceil(filteredData.length / rowsPerPage)
                ),
              }).map((_, i) => {
                const pageNumber = i + 1;
                return (
                  <button
                    key={pageNumber}
                    onClick={() => setPage(pageNumber)}
                    className={`relative inline-flex items-center px-4 py-2
border text-sm font-medium ${
                      page === pageNumber
                        ? 'z-10 bg-blue-50 border-blue-500 text-blue-600'
                        : 'bg-white border-gray-300 text-gray-500 hover:bg-
gray-50'
                    }`}
                  >
                    {pageNumber}
                  </button>
                );
              })}

              <button
                className="relative inline-flex items-center px-2 py-2
rounded-r-md border border-gray-300 bg-white text-sm font-medium text-gray-
500 hover:bg-gray-50"
                onClick={() =>
                  setPage(
                    page < Math.ceil(filteredData.length / rowsPerPage)
                      ? page + 1
                      : page
                  )
                }
                disabled={
                  page >= Math.ceil(filteredData.length / rowsPerPage)
                }
              >
                <span className="sr-only">Next</span>
                <svg
                  className="h-5 w-5"
                  xmlns="http://www.w3.org/2000/svg"
                  viewBox="0 0 20 20"
```

```
                  fill="currentColor"
                  aria-hidden="true"
                >
                  <path
                    fillRule="evenodd"
                    d="M7.293 14.707a1 1 0 010-1.414L10.586 10 7.293 6.707a1
1 0 011.414-1.414l4 4a1 1 0 010 1.414l-4 4a1 1 0 01-1.414 0z"
                    clipRule="evenodd"
                  />
                </svg>
              </button>
            </nav>
          </div>
        </div>
      </div>
    </div>
  );
}

// Usage
<DataTable
  data={[
    {
      id: 1,
      name: 'John Doe',
      email: 'john@example.com',
      role: 'Admin',
      status: 'Active',
    },
    {
      id: 2,
      name: 'Jane Smith',
      email: 'jane@example.com',
      role: 'Editor',
      status: 'Active',
    },
    // More data...
  ]}
  columns={[
    { key: 'id', label: 'ID' },
    { key: 'name', label: 'Name' },
    { key: 'email', label: 'Email' },
    { key: 'role', label: 'Role' },
    {
      key: 'status',
      label: 'Status',
      render: (row) => (
        <span
          className={`px-2 py-1 rounded-full text-xs font-medium ${
            row.status === 'Active'
              ? 'bg-green-100 text-green-800'
              : 'bg-red-100 text-red-800'
          }`}
        >
```

```
          {row.status}
        </span>
      ),
    },
  ]}
/>;
```

3. **Tabs with Dynamic Content**:

```
function Tabs({ tabs }) {
  const [activeTab, setActiveTab] = useState(0);

  return (
    <div>
      {/* Tab headers */}
      <div className="border-b border-gray-200">
        <nav className="-mb-px flex space-x-8">
          {tabs.map((tab, index) => (
            <button
              key={index}
              className={`py-4 px-1 border-b-2 font-medium text-sm ${
                activeTab === index
                  ? 'border-blue-500 text-blue-600'
                  : 'border-transparent text-gray-500 hover:text-gray-700
hover:border-gray-300'
              }`}
              onClick={() => setActiveTab(index)}
            >
              {tab.label}
            </button>
          ))}
        </nav>
      </div>

      {/* Tab content */}
      <div className="py-4">{tabs[activeTab].content}</div>
    </div>
  );
}

// Usage
<Tabs
  tabs={[
    {
      label: 'Profile',
      content: (
        <div className="space-y-4">
          <h3 className="text-lg font-medium">User Profile</h3>
          <p>Profile information and settings.</p>
        </div>
      ),
    },
```

```
    {
      label: 'Account',
      content: (
        <div className="space-y-4">
          <h3 className="text-lg font-medium">Account Settings</h3>
          <p>Manage your account preferences.</p>
        </div>
      ),
    },
    {
      label: 'Notifications',
      content: (
        <div className="space-y-4">
          <h3 className="text-lg font-medium">Notification Settings</h3>
          <p>Configure how you receive notifications.</p>
        </div>
      ),
    },
  ]}
/>;
```

4. **Modal Dialogs and Overlays**:

```
function Modal({ isOpen, onClose, title, children, footer }) {
  if (!isOpen) return null;

  return (
    <>
      {/* Backdrop */}
      <div
        className="fixed inset-0 bg-black bg-opacity-50 transition-opacity"
        onClick={onClose}
      ></div>

      {/* Modal */}
      <div
        className="fixed inset-0 overflow-y-auto"
        aria-labelledby="modal-title"
        role="dialog"
        aria-modal="true"
      >
        <div className="flex items-center justify-center min-h-screen p-4">
          <div className="relative bg-white rounded-lg shadow-xl max-w-md w-
full">
            {/* Header */}
            <div className="flex justify-between items-center p-4 border-b">
              <h3
                className="text-lg font-medium text-gray-900"
                id="modal-title"
              >
                {title}
              </h3>
```

```
              <button
                type="button"
                className="text-gray-400 hover:text-gray-500"
                onClick={onClose}
              >
                <span className="sr-only">Close</span>
                <svg
                  className="h-6 w-6"
                  fill="none"
                  viewBox="0 0 24 24"
                  stroke="currentColor"
                >
                  <path
                    strokeLinecap="round"
                    strokeLinejoin="round"
                    strokeWidth={2}
                    d="M6 18L18 6M6 6l12 12"
                  />
                </svg>
              </button>
            </div>

            {/* Content */}
            <div className="p-6">{children}</div>

            {/* Footer */}
            {footer && (
              <div className="px-4 py-3 bg-gray-50 flex justify-end space-x-
2 rounded-b-lg">
                {footer}
              </div>
            )}
          </div>
        </div>
      </div>
    </>
  );
}

// Usage example with state
function App() {
  const [isModalOpen, setIsModalOpen] = useState(false);

  return (
    <div className="p-8">
      <button
        className="px-4 py-2 bg-blue-500 text-white rounded hover:bg-blue-
600"
        onClick={() => setIsModalOpen(true)}
      >
        Open Modal
      </button>

      <Modal
```

```
        isOpen={isModalOpen}
        onClose={() => setIsModalOpen(false)}
        title="Example Modal"
        footer={
          <>
            <button
              className="px-4 py-2 text-gray-700 border border-gray-300
rounded hover:bg-gray-50"
              onClick={() => setIsModalOpen(false)}
            >
              Cancel
            </button>
            <button
              className="px-4 py-2 bg-blue-500 text-white rounded hover:bg-
blue-600"
              onClick={() => {
                alert('Confirmed!');
                setIsModalOpen(false);
              }}
            >
              Confirm
            </button>
          </>
        }
      >
        <p className="text-gray-600">
          This is an example modal dialog. You can add any content here.
        </p>
        <div className="mt-4">
          <input
            type="text"
            className="w-full border-gray-300 rounded-md shadow-sm
focus:border-blue-500 focus:ring-blue-500"
            placeholder="Enter some text..."
          />
        </div>
      </Modal>
    </div>
  );
}
```

Advanced Layout Techniques

1. **Dashboard Grid**:

```
function Dashboard() {
  return (
    <div className="min-h-screen bg-gray-100 p-4 lg:p-8">
      <div className="grid grid-cols-1 md:grid-cols-4 gap-4 lg:gap-8">
        {/* Header - full width */}
        <div className="md:col-span-4 bg-white rounded-lg shadow-md p-6">
```

```
          <h1 className="text-2xl font-bold">Dashboard</h1>
          <p className="text-gray-500">Welcome back, User</p>
        </div>

        {/* Sidebar - 1/4 width on md+ screens */}
        <div className="md:col-span-1 bg-white rounded-lg shadow-md p-6">
          <nav className="space-y-2">
            <a
              href="#"
              className="block py-2 px-4 rounded-md bg-blue-50 text-blue-
700"
            >
              Dashboard
            </a>
            <a
              href="#"
              className="block py-2 px-4 rounded-md text-gray-700 hover:bg-
gray-50"
            >
              Analytics
            </a>
            <a
              href="#"
              className="block py-2 px-4 rounded-md text-gray-700 hover:bg-
gray-50"
            >
              Reports
            </a>
            <a
              href="#"
              className="block py-2 px-4 rounded-md text-gray-700 hover:bg-
gray-50"
            >
              Settings
            </a>
          </nav>
        </div>

        {/* Main content - 3/4 width on md+ screens */}
        <div className="md:col-span-3 space-y-4 lg:space-y-8">
          {/* Stats row */}
          <div className="grid grid-cols-1 sm:grid-cols-3 gap-4">
            <div className="bg-white rounded-lg shadow-md p-6">
              <h3 className="text-gray-500 text-sm font-medium">
                Total Users
              </h3>
              <p className="text-3xl font-bold">12,345</p>
              <p className="text-green-600 text-sm mt-2">
                ↑ 12% from last month
              </p>
            </div>

            <div className="bg-white rounded-lg shadow-md p-6">
              <h3 className="text-gray-500 text-sm font-medium">Revenue</h3>
```

```
              <p className="text-3xl font-bold">$34,567</p>
              <p className="text-green-600 text-sm mt-2">
                ↑ 7% from last month
              </p>
            </div>

            <div className="bg-white rounded-lg shadow-md p-6">
              <h3 className="text-gray-500 text-sm font-medium">
                Active Projects
              </h3>
              <p className="text-3xl font-bold">27</p>
              <p className="text-red-600 text-sm mt-2">
                ↓ 2% from last month
              </p>
            </div>
          </div>

          {/* Chart section */}
          <div className="bg-white rounded-lg shadow-md p-6">
            <h2 className="text-lg font-semibold mb-4">Monthly Revenue</h2>
            <div className="h-64 bg-gray-100 rounded border border-gray-
200">
              {/* Chart placeholder */}
              <div className="h-full flex items-center justify-center text-
gray-400">
                Chart Placeholder
              </div>
            </div>
          </div>

          {/* Table section */}
          <div className="bg-white rounded-lg shadow-md p-6">
            <h2 className="text-lg font-semibold mb-4">
              Recent Transactions
            </h2>
            <div className="overflow-x-auto">
              <table className="min-w-full divide-y divide-gray-200">
                <thead>
                  <tr>
                    <th className="px-4 py-3 text-left text-xs font-medium
text-gray-500 uppercase tracking-wider">
                      Date
                    </th>
                    <th className="px-4 py-3 text-left text-xs font-medium
text-gray-500 uppercase tracking-wider">
                      Customer
                    </th>
                    <th className="px-4 py-3 text-left text-xs font-medium
text-gray-500 uppercase tracking-wider">
                      Amount
                    </th>
                    <th className="px-4 py-3 text-left text-xs font-medium
text-gray-500 uppercase tracking-wider">
                      Status
```

```
                    </th>
                  </tr>
                </thead>
                <tbody className="bg-white divide-y divide-gray-200">
                  <tr>
                    <td className="px-4 py-3 whitespace-nowrap">
                      2023-04-23
                    </td>
                    <td className="px-4 py-3 whitespace-nowrap">
                      John Smith
                    </td>
                    <td className="px-4 py-3 whitespace-nowrap">$123.45</td>
                    <td className="px-4 py-3 whitespace-nowrap">
                      <span className="px-2 py-1 rounded-full text-xs font-
medium bg-green-100 text-green-800">
                        Completed
                      </span>
                    </td>
                  </tr>
                  {/* More rows... */}
                </tbody>
              </table>
            </div>
          </div>
        </div>

        {/* Footer - full width */}
        <div className="md:col-span-4 bg-white rounded-lg shadow-md p-6
text-center text-gray-500">
          &copy; 2023 Your Company
        </div>
      </div>
    </div>
  );
}
```

2. **Masonry Layout**:

```
function MasonryGrid({ children, columns = 3, gap = 16 }) {
  const [columnHeights, setColumnHeights] =
useState(Array(columns).fill(0));
  const columnRef = useRef([]);

  useEffect(() => {
    // Reset heights when columns change
    setColumnHeights(Array(columns).fill(0));
    columnRef.current = Array(columns)
      .fill()
      .map(() => React.createRef());
  }, [columns]);

  const childrenArray = React.Children.toArray(children);
```

```
  // Calculate layout - this is a simple approach for demonstration
  // In production, you would use more sophisticated measurement techniques
  const getColumnItems = () => {
    const columnItems = Array(columns)
      .fill()
      .map(() => []);

    childrenArray.forEach((child, index) => {
      // Find column with lowest height
      const shortestColumn = columnHeights.indexOf(
        Math.min(...columnHeights)
      );

      // Add child to shortest column
      columnItems[shortestColumn].push(child);

      // Update column height (assumes fixed height per item for simplicity)
      // In reality, you would measure actual DOM elements
      const estimatedHeight = 200; // Placeholder value
      columnHeights[shortestColumn] += estimatedHeight + gap;
    });

    return columnItems;
  };

  return (
    <div className="flex flex-wrap" style={{ margin: `-${gap / 2}px` }}>
      {getColumnItems().map((columnItems, columnIndex) => (
        <div
          key={columnIndex}
          ref={columnRef.current[columnIndex]}
          className="flex-grow"
          style={{
            padding: `${gap / 2}px`,
            width: `${100 / columns}%`,
          }}
        >
          <div className="flex flex-col space-y-4">{columnItems}</div>
        </div>
      ))}
    </div>
  );
}

// Usage
function App() {
  return (
    <div className="p-8">
      <h1 className="text-3xl font-bold mb-8">Masonry Gallery</h1>

      <MasonryGrid columns={3} gap={16}>
        {[...Array(20)].map((_, index) => (
          <div
```

```
            key={index}
            className="bg-white rounded-lg shadow-md overflow-hidden"
            style={{
              height: `${200 + Math.floor(Math.random() * 200)}px`,
              backgroundColor: `hsl(${index * 20}, 70%, 80%)`,
            }}
          >
            <div className="p-4">
              <h3 className="font-bold">Item {index + 1}</h3>
              <p>Random height content.</p>
            </div>
          </div>
        ))}
      </MasonryGrid>
    </div>
  );
}
```

3. **Responsive Sidebar Layouts**:

```
function SidebarLayout() {
  const [sidebarOpen, setSidebarOpen] = useState(false);

  return (
    <div className="min-h-screen bg-gray-100">
      {/* Mobile sidebar overlay */}
      {sidebarOpen && (
        <div
          className="fixed inset-0 bg-gray-600 bg-opacity-75 z-20 lg:hidden"
          onClick={() => setSidebarOpen(false)}
        ></div>
      )}

      {/* Sidebar */}
      <div
        className={`
        fixed inset-y-0 left-0 z-30 w-64 bg-white shadow-md transform
transition-transform duration-300 ease-in-out
        ${sidebarOpen ? 'translate-x-0' : '-translate-x-full'}
        lg:translate-x-0 lg:static lg:h-screen
      `}
      >
        <div className="flex flex-col h-full">
          <div className="flex items-center justify-between px-4 py-6
border-b">
            <h1 className="text-xl font-bold">Dashboard</h1>
            <button
              className="lg:hidden text-gray-500 hover:text-gray-700"
              onClick={() => setSidebarOpen(false)}
            >
              <svg
                className="h-6 w-6"
```

```
                fill="none"
                viewBox="0 0 24 24"
                stroke="currentColor"
              >
                <path
                  strokeLinecap="round"
                  strokeLinejoin="round"
                  strokeWidth={2}
                  d="M6 18L18 6M6 6l12 12"
                />
              </svg>
            </button>
          </div>

          <nav className="flex-1 px-2 py-4 overflow-y-auto">
            <div className="space-y-1">
              <a
                href="#"
                className="group flex items-center px-2 py-2 rounded-md bg-
blue-100 text-blue-700"
              >
                <svg
                  className="mr-3 h-6 w-6"
                  fill="none"
                  viewBox="0 0 24 24"
                  stroke="currentColor"
                >
                  <path
                    strokeLinecap="round"
                    strokeLinejoin="round"
                    strokeWidth={2}
                    d="M3 12l2-2m0 0l7-7 7 7M5 10v10a1 1 0 001 1h3m10-11l2
2m-2-2v10a1 1 0 01-1 1h-3m-6 0a1 1 0 001-1v-4a1 1 0 011-1h2a1 1 0 011 1v4a1
1 0 001 1m-6 0h6"
                  />
                </svg>
                Dashboard
              </a>
              <a
                href="#"
                className="group flex items-center px-2 py-2 rounded-md
text-gray-700 hover:bg-gray-50"
              >
                <svg
                  className="mr-3 h-6 w-6"
                  fill="none"
                  viewBox="0 0 24 24"
                  stroke="currentColor"
                >
                  <path
                    strokeLinecap="round"
                    strokeLinejoin="round"
                    strokeWidth={2}
                    d="M9 19v-6a2 2 0 00-2-2H5a2 2 0 00-2 2v6a2 2 0 002
```

```
2h2a2 2 0 002-2zm0 0V9a2 2 0 012-2h2a2 2 0 012 2v10m-6 0a2 2 0 002 2h2a2 2 0
002-2m0 0V5a2 2 0 012-2h2a2 2 0 012 2v14a2 2 0 01-2 2h-2a2 2 0 01-2-2z"
                  />
                </svg>
                Analytics
              </a>
              {/* More navigation items... */}
            </div>
          </nav>

          <div className="border-t px-4 py-4">
            <div className="flex items-center">
              <img
                className="h-8 w-8 rounded-full"
                src="https://randomuser.me/api/portraits/men/1.jpg"
                alt=""
              />
              <div className="ml-3">
                <p className="text-sm font-medium">John Smith</p>
                <p className="text-xs text-gray-500">View Profile</p>
              </div>
            </div>
          </div>
        </div>
      </div>

      {/* Main content */}
      <div className="lg:pl-64">
        {/* Top navbar */}
        <div className="sticky top-0 z-10 flex-shrink-0 flex h-16 bg-white
shadow">
          <button
            className="px-4 text-gray-500 lg:hidden"
            onClick={() => setSidebarOpen(true)}
          >
            <svg
              className="h-6 w-6"
              fill="none"
              viewBox="0 0 24 24"
              stroke="currentColor"
            >
              <path
                strokeLinecap="round"
                strokeLinejoin="round"
                strokeWidth={2}
                d="M4 6h16M4 12h16M4 18h16"
              />
            </svg>
          </button>

          <div className="flex-1 px-4 flex justify-between">
            <div className="flex-1 flex">
              <div className="w-full flex md:ml-0">
                <div className="relative w-full text-gray-400 focus-
```

```
within:text-gray-600">
                  <div className="absolute inset-y-0 left-0 pl-3 flex items-
center pointer-events-none">
                    <svg
                      className="h-5 w-5"
                      fill="currentColor"
                      viewBox="0 0 20 20"
                    >
                      <path
                        fillRule="evenodd"
                        clipRule="evenodd"
                        d="M8 4a4 4 0 100 8 4 4 0 000-8zM2 8a6 6 0 1110.89
3.476l4.817 4.817a1 1 0 01-1.414 1.414l-4.816-4.816A6 6 0 012 8z"
                        />
                    </svg>
                  </div>
                  <input
                    id="search-field"
                    className="block w-full h-full pl-10 pr-3 py-2 rounded-
md text-gray-900 placeholder-gray-500 focus:outline-none focus:placeholder-
gray-400 sm:text-sm"
                    placeholder="Search"
                    type="search"
                  />
                </div>
              </div>
            </div>

            <div className="ml-4 flex items-center md:ml-6">
              {/* Notification bell */}
              <button className="p-1 text-gray-400 rounded-full hover:text-
gray-500">
                <svg
                  className="h-6 w-6"
                  fill="none"
                  viewBox="0 0 24 24"
                  stroke="currentColor"
                >
                  <path
                    strokeLinecap="round"
                    strokeLinejoin="round"
                    strokeWidth={2}
                    d="M15 17h5l-1.405-1.405A2.032 2.032 0 0118
14.158V11a6.002 6.002 0 00-4-5.659V5a2 2 0 10-4 0v.341C7.67 6.165 6 8.388 6
11v3.159c0 .538-.214 1.055-.595 1.436L4 17h5m6 0v1a3 3 0 11-6 0v-1m6 0H9"
                    />
                </svg>
              </button>
            </div>
          </div>
        </div>

        {/* Main content */}
        <main className="p-4 sm:p-6 lg:p-8">
```

```
          <h1 className="text-2xl font-bold mb-6">Dashboard</h1>

          <div className="bg-white rounded-lg shadow-md p-6">
            <p>
              Welcome to your dashboard. This is a responsive sidebar layout
              built with Tailwind CSS.
            </p>
          </div>
        </main>
      </div>
    </div>
  );
}
```

Creating Design Systems

Design Token Methodology

1. **Design Token Architecture**:

```
// tokens.js - Design Token Structure
const tokens = {
  colors: {
    brand: {
      primary: '#0066ff',
      secondary: '#ff6b6b',
      tertiary: '#4ecdc4',
    },
    neutral: {
      white: '#ffffff',
      gray: {
        50: '#f9fafb',
        100: '#f3f4f6',
        200: '#e5e7eb',
        300: '#d1d5db',
        400: '#9ca3af',
        500: '#6b7280',
        600: '#4b5563',
        700: '#374151',
        800: '#1f2937',
        900: '#111827',
      },
      black: '#000000',
    },
    status: {
      success: '#10b981',
      warning: '#f59e0b',
      error: '#ef4444',
      info: '#3b82f6',
    },
  },
```

```
    spacing: {
      0: '0',
      px: '1px',
      0.5: '0.125rem',
      1: '0.25rem',
      1.5: '0.375rem',
      2: '0.5rem',
      2.5: '0.625rem',
      3: '0.75rem',
      3.5: '0.875rem',
      4: '1rem',
      5: '1.25rem',
      6: '1.5rem',
      8: '2rem',
      10: '2.5rem',
      12: '3rem',
      16: '4rem',
      20: '5rem',
      24: '6rem',
      32: '8rem',
      40: '10rem',
      48: '12rem',
      56: '14rem',
      64: '16rem',
    },
    typography: {
      fonts: {
        sans: ['"Inter"', 'sans-serif'],
        serif: ['"Merriweather"', 'serif'],
        mono: ['"JetBrains Mono"', 'monospace'],
      },
      sizes: {
        xs: '0.75rem',
        sm: '0.875rem',
        base: '1rem',
        lg: '1.125rem',
        xl: '1.25rem',
        '2xl': '1.5rem',
        '3xl': '1.875rem',
        '4xl': '2.25rem',
        '5xl': '3rem',
        '6xl': '3.75rem',
      },
      weights: {
        thin: 100,
        extralight: 200,
        light: 300,
        normal: 400,
        medium: 500,
        semibold: 600,
        bold: 700,
        extrabold: 800,
        black: 900,
      },
```

```
    lineHeights: {
      none: 1,
      tight: 1.25,
      snug: 1.375,
      normal: 1.5,
      relaxed: 1.625,
      loose: 2,
    },
    letterSpacing: {
      tighter: '-0.05em',
      tight: '-0.025em',
      normal: '0',
      wide: '0.025em',
      wider: '0.05em',
      widest: '0.1em',
    },
  },
  borders: {
    widths: {
      none: '0',
      xs: '1px',
      sm: '2px',
      md: '4px',
      lg: '8px',
    },
    radii: {
      none: '0',
      sm: '0.125rem',
      default: '0.25rem',
      md: '0.375rem',
      lg: '0.5rem',
      xl: '0.75rem',
      '2xl': '1rem',
      '3xl': '1.5rem',
      full: '9999px',
    },
  },
  shadows: {
    sm: '0 1px 2px 0 rgba(0, 0, 0, 0.05)',
    DEFAULT:
      '0 1px 3px 0 rgba(0, 0, 0, 0.1), 0 1px 2px 0 rgba(0, 0, 0, 0.06)',
    md: '0 4px 6px -1px rgba(0, 0, 0, 0.1), 0 2px 4px -1px rgba(0, 0, 0,
0.06)',
    lg: '0 10px 15px -3px rgba(0, 0, 0, 0.1), 0 4px 6px -2px rgba(0, 0, 0,
0.05)',
    xl: '0 20px 25px -5px rgba(0, 0, 0, 0.1), 0 10px 10px -5px rgba(0, 0, 0,
0.04)',
    '2xl': '0 25px 50px -12px rgba(0, 0, 0, 0.25)',
    inner: 'inset 0 2px 4px 0 rgba(0, 0, 0, 0.06)',
    none: 'none',
  },
  animations: {
    durations: {
      fast: '150ms',
```

```
      normal: '300ms',
      slow: '500ms',
    },
    timingFunctions: {
      linear: 'linear',
      in: 'cubic-bezier(0.4, 0, 1, 1)',
      out: 'cubic-bezier(0, 0, 0.2, 1)',
      'in-out': 'cubic-bezier(0.4, 0, 0.2, 1)',
    },
  },
  breakpoints: {
    sm: '640px',
    md: '768px',
    lg: '1024px',
    xl: '1280px',
    '2xl': '1536px',
  },
  zIndices: {
    0: 0,
    10: 10,
    20: 20,
    30: 30,
    40: 40,
    50: 50,
    auto: 'auto',
  },
};

export default tokens;
```

2. **CSS Variable Generation**:

```
// tokensToCSSVariables.js
const tokens = require('./tokens');
const fs = require('fs');

// Function to flatten a nested object into key-value pairs
function flattenTokens(obj, prefix = '') {
  return Object.entries(obj).reduce((acc, [key, value]) => {
    const newKey = prefix ? `${prefix}-${key}` : key;

    if (typeof value === 'object' && !Array.isArray(value)) {
      return { ...acc, ...flattenTokens(value, newKey) };
    }

    return { ...acc, [newKey]: value };
  }, {});
}

// Generate CSS variables
function generateCSSVariables() {
  const flatTokens = flattenTokens(tokens);
```

```
  let cssOutput = ':root {\n';

  Object.entries(flatTokens).forEach(([key, value]) => {
    // Convert array values (like font stacks) to proper CSS
    const cssValue = Array.isArray(value) ? value.join(', ') : value;
    cssOutput += `  --${key}: ${cssValue};\n`;
  });

  cssOutput += '}\n';
  return cssOutput;
}

// Write CSS to file
const cssVariables = generateCSSVariables();
fs.writeFileSync('src/styles/variables.css', cssVariables);
console.log('CSS variables generated successfully!');
```

3. **Tailwind Configuration from Tokens**:

```
// tailwind.config.js
const tokens = require('./tokens');

module.exports = {
  theme: {
    colors: {
      transparent: 'transparent',
      current: 'currentColor',
      ...tokens.colors,
    },
    spacing: tokens.spacing,
    fontFamily: tokens.typography.fonts,
    fontSize: tokens.typography.sizes,
    fontWeight: tokens.typography.weights,
    lineHeight: tokens.typography.lineHeights,
    letterSpacing: tokens.typography.letterSpacing,
    borderWidth: tokens.borders.widths,
    borderRadius: tokens.borders.radii,
    boxShadow: tokens.shadows,
    transitionDuration: tokens.animations.durations,
    transitionTimingFunction: tokens.animations.timingFunctions,
    screens: tokens.breakpoints,
    zIndex: tokens.zIndices,
  },
  variants: {
    extend: {},
  },
  plugins: [],
};
```

Component Documentation

1. **Storybook Integration**:

```
// Button.stories.jsx
import React from 'react';
import Button from './Button';

export default {
  title: 'Components/Button',
  component: Button,
  argTypes: {
    variant: {
      control: {
        type: 'select',
        options: ['primary', 'secondary', 'outline', 'text'],
      },
      description: 'The button variant',
      defaultValue: 'primary',
    },
    size: {
      control: { type: 'select', options: ['sm', 'md', 'lg'] },
      description: 'The button size',
      defaultValue: 'md',
    },
    disabled: {
      control: 'boolean',
      description: 'Whether the button is disabled',
      defaultValue: false,
    },
    onClick: { action: 'clicked' },
  },
};

// Template for stories
const Template = (args) => <Button {...args} />;

// Primary button
export const Primary = Template.bind({});
Primary.args = {
  variant: 'primary',
  children: 'Primary Button',
};

// Secondary button
export const Secondary = Template.bind({});
Secondary.args = {
  variant: 'secondary',
  children: 'Secondary Button',
};

// Outline button
export const Outline = Template.bind({});
Outline.args = {
  variant: 'outline',
```

```
  children: 'Outline Button',
};

// Text button
export const Text = Template.bind({});
Text.args = {
  variant: 'text',
  children: 'Text Button',
};

// Small button
export const Small = Template.bind({});
Small.args = {
  variant: 'primary',
  size: 'sm',
  children: 'Small Button',
};

// Large button
export const Large = Template.bind({});
Large.args = {
  variant: 'primary',
  size: 'lg',
  children: 'Large Button',
};

// Disabled button
export const Disabled = Template.bind({});
Disabled.args = {
  variant: 'primary',
  disabled: true,
  children: 'Disabled Button',
};

// With icon
export const WithIcon = Template.bind({});
WithIcon.args = {
  variant: 'primary',
  children: (
    <>
      <svg
        className="w-4 h-4 mr-2"
        fill="none"
        stroke="currentColor"
        viewBox="0 0 24 24"
      >
        <path
          strokeLinecap="round"
          strokeLinejoin="round"
          strokeWidth={2}
          d="M12 6v6m0 0v6m0-6h6m-6 0H6"
        />
      </svg>
      Button with Icon
```

```
    </>
  ),
};
```

2. **Component API Documentation**:

```
// Button.jsx
import React from 'react';
import PropTypes from 'prop-types';

/**
 * Button component for user interactions.
 *
 * @component
 * @example
 * ```jsx
 * <Button variant="primary" onClick={() => alert('Clicked!')}>
 *   Click Me
 * </Button>
 * ```
 */
function Button({
  children,
  variant = 'primary',
  size = 'md',
  disabled = false,
  className = '',
  onClick,
  ...props
}) {
  const baseClasses =
    'inline-flex items-center justify-center font-medium rounded
focus:outline-none focus:ring-2 focus:ring-offset-2';

  const variantClasses = {
    primary:
      'bg-brand-primary text-white hover:bg-brand-primary-dark focus:ring-
brand-primary',
    secondary:
      'bg-brand-secondary text-white hover:bg-brand-secondary-dark
focus:ring-brand-secondary',
    outline:
      'border border-brand-primary text-brand-primary bg-transparent
hover:bg-brand-primary hover:text-white focus:ring-brand-primary',
    text: 'bg-transparent text-brand-primary hover:bg-brand-primary
hover:bg-opacity-10 focus:ring-brand-primary',
  };

  const sizeClasses = {
    sm: 'text-sm px-3 py-1.5',
    md: 'text-base px-4 py-2',
    lg: 'text-lg px-6 py-3',
```

```
  };

  const disabledClasses = disabled
    ? 'opacity-50 cursor-not-allowed'
    : 'cursor-pointer';

  return (
    <button
      className={`${baseClasses} ${variantClasses[variant]}
${sizeClasses[size]} ${disabledClasses} ${className}`}
      disabled={disabled}
      onClick={onClick}
      {...props}
    >
      {children}
    </button>
  );
}

Button.propTypes = {
  /**
   * Button content
   */
  children: PropTypes.node.isRequired,

  /**
   * The variant of the button
   */
  variant: PropTypes.oneOf(['primary', 'secondary', 'outline', 'text']),

  /**
   * The size of the button
   */
  size: PropTypes.oneOf(['sm', 'md', 'lg']),

  /**
   * Whether the button is disabled
   */
  disabled: PropTypes.bool,

  /**
   * Additional classes to apply
   */
  className: PropTypes.string,

  /**
   * Click handler
   */
  onClick: PropTypes.func,
};

export default Button;
```

3. **Style Guide Generation**:

   Using Storybook Docs addon:

```
// .storybook/main.js
module.exports = {
  stories: ['../src/**/*.stories.@(js|jsx|ts|tsx)'],
  addons: [
    '@storybook/addon-links',
    '@storybook/addon-essentials',
    '@storybook/addon-a11y',
    '@storybook/addon-docs',
  ],
};
```

   Creating a theme overview page:

```
// ColorPalette.stories.jsx
import React from 'react';
import tokens from '../tokens';

export default {
  title: 'Design System/Colors',
  parameters: {
    docs: {
      description: {
        component: 'Color palette for the design system',
      },
    },
  },
};

export const BrandColors = () => {
  return (
    <div className="space-y-6">
      <h2 className="text-2xl font-bold">Brand Colors</h2>
      <div className="grid grid-cols-1 md:grid-cols-3 gap-4">
        {Object.entries(tokens.colors.brand).map(([name, value]) => (
          <div key={name} className="border rounded-lg overflow-hidden">
            <div
              className="h-24 w-full"
              style={{ backgroundColor: value }}
            ></div>
            <div className="p-4">
              <h3 className="font-medium">{name}</h3>
              <p className="text-sm text-gray-500">{value}</p>
            </div>
          </div>
        ))}
      </div>
```

```
    </div>
  );
};

export const NeutralColors = () => {
  return (
    <div className="space-y-6">
      <h2 className="text-2xl font-bold">Neutral Colors</h2>
      <div className="grid grid-cols-1 md:grid-cols-5 gap-4">
        {Object.entries(tokens.colors.neutral.gray).map(([shade, value]) =>
(
          <div key={shade} className="border rounded-lg overflow-hidden">
            <div
              className="h-24 w-full"
              style={{ backgroundColor: value }}
            ></div>
            <div className="p-4">
              <h3 className="font-medium">Gray {shade}</h3>
              <p className="text-sm text-gray-500">{value}</p>
            </div>
          </div>
        ))}
      </div>
    </div>
  );
};

export const StatusColors = () => {
  return (
    <div className="space-y-6">
      <h2 className="text-2xl font-bold">Status Colors</h2>
      <div className="grid grid-cols-1 md:grid-cols-4 gap-4">
        {Object.entries(tokens.colors.status).map(([name, value]) => (
          <div key={name} className="border rounded-lg overflow-hidden">
            <div
              className="h-24 w-full"
              style={{ backgroundColor: value }}
            ></div>
            <div className="p-4">
              <h3 className="font-medium">{name}</h3>
              <p className="text-sm text-gray-500">{value}</p>
            </div>
          </div>
        ))}
      </div>
    </div>
  );
};
```

Design System Implementation

1. **Component Library Structure**:

```
src/
├── components/
│   ├── atoms/
│   │   ├── Button/
│   │   │   ├── Button.jsx
│   │   │   ├── Button.test.jsx
│   │   │   ├── Button.stories.jsx
│   │   │   └── index.js
│   │   ├── Input/
│   │   ├── Checkbox/
│   │   └── ...
│   ├── molecules/
│   │   ├── Card/
│   │   ├── FormField/
│   │   ├── Dropdown/
│   │   └── ...
│   ├── organisms/
│   │   ├── Header/
│   │   ├── Footer/
│   │   ├── Sidebar/
│   │   └── ...
│   ├── templates/
│   │   ├── DashboardLayout/
│   │   ├── LandingLayout/
│   │   └── ...
│   └── index.js
├── tokens/
│   ├── index.js
│   ├── colors.js
│   ├── typography.js
│   └── ...
├── styles/
│   ├── index.css
│   ├── variables.css
│   └── utilities.css
└── utils/
    ├── classNames.js
    └── ...
```

2. **Consistent Component Pattern**:

```jsx
// FormField.jsx (molecule component example)
import React from 'react';
import PropTypes from 'prop-types';
import Input from '../atoms/Input';
import Label from '../atoms/Label';
import HelperText from '../atoms/HelperText';

function FormField({
  id,
```

```
  label,
  type = 'text',
  helperText,
  error,
  className = '',
  ...props
}) {
  return (
    <div className={`space-y-1 ${className}`}>
      {label && (
        <Label htmlFor={id} required={props.required}>
          {label}
        </Label>
      )}

      <Input id={id} type={type} error={!!error} {...props} />

      {(helperText || error) && (
        <HelperText error={!!error}>{error || helperText}</HelperText>
      )}
    </div>
  );
}

FormField.propTypes = {
  id: PropTypes.string.isRequired,
  label: PropTypes.string,
  type: PropTypes.string,
  helperText: PropTypes.string,
  error: PropTypes.string,
  className: PropTypes.string,
  required: PropTypes.bool,
};

export default FormField;
```

3. **Composition vs. Configuration**:

```
// Bad: Overly complex configuration
<Button
  primary
  large
  withIcon
  iconPosition="left"
  rounded
  uppercase
  fullWidth
  loading
  loadingText="Processing..."
>
  Submit
</Button>
```

```
// Better: Composition with sensible defaults
<Button variant="primary" size="lg" isLoading={isSubmitting}>
  <Icon name="send" className="mr-2" />
  Submit
</Button>

// Even better: Specialized components through composition
<SubmitButton isLoading={isSubmitting}>
  Submit
</SubmitButton>
```

4. **Theme Switching**:

```
// ThemeProvider.jsx
import React, {
  createContext,
  useContext,
  useState,
  useEffect,
} from 'react';

const ThemeContext = createContext();

export function useTheme() {
  return useContext(ThemeContext);
}

export function ThemeProvider({ children }) {
  // Initialize from localStorage or system preference
  const [theme, setThemeState] = useState(() => {
    const savedTheme = localStorage.getItem('theme');
    if (savedTheme) return savedTheme;

    return window.matchMedia('(prefers-color-scheme: dark)').matches
      ? 'dark'
      : 'light';
  });

  const setTheme = (newTheme) => {
    setThemeState(newTheme);
    localStorage.setItem('theme', newTheme);
  };

  // Apply theme to document
  useEffect(() => {
    document.documentElement.classList.remove('light', 'dark');
    document.documentElement.classList.add(theme);
  }, [theme]);

  // Listen for system preference changes
  useEffect(() => {
```

```
    const mediaQuery = window.matchMedia('(prefers-color-scheme: dark)');

    const handleChange = () => {
      if (!localStorage.getItem('theme')) {
        setThemeState(mediaQuery.matches ? 'dark' : 'light');
      }
    };

    mediaQuery.addEventListener('change', handleChange);
    return () => mediaQuery.removeEventListener('change', handleChange);
  }, []);

  const value = {
    theme,
    setTheme,
    isDark: theme === 'dark',
  };

  return (
    <ThemeContext.Provider value={value}>{children}</ThemeContext.Provider>
  );
}
```

Usage:

```
function App() {
  return (
    <ThemeProvider>
      <NestedComponent />
    </ThemeProvider>
  );
}

function NestedComponent() {
  const { theme, setTheme } = useTheme();

  return (
    <div>
      <p>Current theme: {theme}</p>
      <button
        onClick={() => setTheme(theme === 'light' ? 'dark' : 'light')}
        className="px-4 py-2 bg-primary text-white rounded"
      >
        Toggle Theme
      </button>
    </div>
  );
}
```

5. **Accessibility Integration**:

```jsx
// AccessibleButton.jsx
import React from 'react';
import PropTypes from 'prop-types';

function AccessibleButton({
  children,
  variant = 'primary',
  size = 'md',
  disabled = false,
  loading = false,
  className = '',
  'aria-label': ariaLabel,
  onClick,
  ...props
}) {
  // Base classes similar to before
  const baseClasses = '...';
  const variantClasses = {...};
  const sizeClasses = {...};

  // Handle loading state
  const isDisabled = disabled || loading;
  const showLoading = loading;

  return (
    <button
      className={`${baseClasses} ${variantClasses[variant]}
${sizeClasses[size]} ${isDisabled ? 'opacity-50 cursor-not-allowed' :
'cursor-pointer'} ${className}`}
      disabled={isDisabled}
      onClick={onClick}
      aria-label={ariaLabel || undefined}
      aria-busy={loading}
      aria-disabled={isDisabled}
      {...props}
    >
      {showLoading ? (
        <span className="flex items-center">
          <svg className="animate-spin -ml-1 mr-2 h-4 w-4"
xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24">
            <circle className="opacity-25" cx="12" cy="12" r="10"
stroke="currentColor" strokeWidth="4"></circle>
            <path className="opacity-75" fill="currentColor" d="M4 12a8 8 0
018-8V0C5.373 0 0 5.373 0 12h4zm2 5.291A7.962 7.962 0 014 12H0c0 3.042 1.135
5.824 3 7.938l3-2.647z"></path>
          </svg>
          {props['aria-label'] || 'Loading...'}
        </span>
      ) : children}
    </button>
  );
}
```

```
AccessibleButton.propTypes = {
  children: PropTypes.node.isRequired,
  variant: PropTypes.oneOf(['primary', 'secondary', 'outline', 'text']),
  size: PropTypes.oneOf(['sm', 'md', 'lg']),
  disabled: PropTypes.bool,
  loading: PropTypes.bool,
  className: PropTypes.string,
  'aria-label': PropTypes.string,
  onClick: PropTypes.func,
};

export default AccessibleButton;
```

Animation Techniques

CSS Transitions

1. **Basic Transitions**:

```
.button {
  background-color: #3b82f6;
  color: white;
  padding: 8px 16px;
  border-radius: 4px;
  /* Add transition */
  transition: background-color 0.3s ease;
}

.button:hover {
  background-color: #2563eb;
}
```

Tailwind equivalent:

```
<button
  class="bg-blue-500 hover:bg-blue-600 text-white px-4 py-2 rounded
transition duration-300 ease-in-out"
>
  Hover me
</button>
```

2. **Multiple Property Transitions**:

```
.card {
  transform: translateY(0);
  box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
  transition: transform 0.3s ease, box-shadow 0.3s ease;
```

```
}

.card:hover {
  transform: translateY(-5px);
  box-shadow: 0 10px 15px rgba(0, 0, 0, 0.1);
}
```

Tailwind equivalent:

```
<div
  class="transform translate-y-0 shadow-md hover:-translate-y-1
hover:shadow-lg transition-all duration-300 ease-in-out"
>
  <div class="p-6 bg-white rounded-lg">Card content</div>
</div>
```

3. **Transition Timing Functions**:

```
.ease-linear {
  transition-timing-function: linear;
}
.ease-in {
  transition-timing-function: cubic-bezier(0.4, 0, 1, 1);
}
.ease-out {
  transition-timing-function: cubic-bezier(0, 0, 0.2, 1);
}
.ease-in-out {
  transition-timing-function: cubic-bezier(0.4, 0, 0.2, 1);
}
.ease-custom {
  transition-timing-function: cubic-bezier(0.68, -0.55, 0.27, 1.55);
}
```

Tailwind equivalent:

```
<button
  class="bg-blue-500 hover:bg-blue-600 text-white px-4 py-2 rounded
transition duration-300 ease-linear"
>
  Linear
</button>

<button
  class="bg-blue-500 hover:bg-blue-600 text-white px-4 py-2 rounded
transition duration-300 ease-in"
>
```

```
  Ease In
</button>

<button
  class="bg-blue-500 hover:bg-blue-600 text-white px-4 py-2 rounded
transition duration-300 ease-out"
>
  Ease Out
</button>

<button
  class="bg-blue-500 hover:bg-blue-600 text-white px-4 py-2 rounded
transition duration-300 ease-in-out"
>
  Ease In Out
</button>
```

CSS Animations

1. **Basic Keyframe Animation**:

```
@keyframes fadeIn {
  from {
    opacity: 0;
  }
  to {
    opacity: 1;
  }
}

.fade-in {
  animation: fadeIn 1s ease-out forwards;
}
```

Tailwind (with custom animation):

```
/* In your CSS */
@keyframes fadeIn {
  from {
    opacity: 0;
  }
  to {
    opacity: 1;
  }
}

.animate-fade-in {
  animation: fadeIn 1s ease-out forwards;
}
```

```
<div class="animate-fade-in">This content fades in</div>
```

2. **Multi-step Animation**:

```
@keyframes pulse {
  0% {
    transform: scale(1);
    opacity: 1;
  }
  50% {
    transform: scale(1.1);
    opacity: 0.7;
  }
  100% {
    transform: scale(1);
    opacity: 1;
  }
}

.pulse {
  animation: pulse 2s infinite ease-in-out;
}
```

Tailwind (built-in animation):

```
<div class="animate-pulse">This content pulses</div>
```

3. **Staggered Animations**:

```
.staggered-item:nth-child(1) {
  animation-delay: 0.1s;
}
.staggered-item:nth-child(2) {
  animation-delay: 0.2s;
}
.staggered-item:nth-child(3) {
  animation-delay: 0.3s;
}
.staggered-item:nth-child(4) {
  animation-delay: 0.4s;
}
.staggered-item:nth-child(5) {
  animation-delay: 0.5s;
}
```

Tailwind (with custom classes):

```
@keyframes fadeInUp {
  from {
    opacity: 0;
    transform: translateY(20px);
  }
  to {
    opacity: 1;
    transform: translateY(0);
  }
}

.animate-fade-in-up {
  animation: fadeInUp 0.5s ease-out forwards;
}

.delay-1 {
  animation-delay: 0.1s;
}
.delay-2 {
  animation-delay: 0.2s;
}
.delay-3 {
  animation-delay: 0.3s;
}
.delay-4 {
  animation-delay: 0.4s;
}
.delay-5 {
  animation-delay: 0.5s;
}
```

```
<ul>
  <li class="animate-fade-in-up delay-1">Item 1</li>
  <li class="animate-fade-in-up delay-2">Item 2</li>
  <li class="animate-fade-in-up delay-3">Item 3</li>
  <li class="animate-fade-in-up delay-4">Item 4</li>
  <li class="animate-fade-in-up delay-5">Item 5</li>
</ul>
```

Advanced Animation Techniques

1. **Intersection Observer for Scroll Animations**:

```
import React, { useEffect, useRef, useState } from 'react';
```

```
function FadeInOnScroll({ children, threshold = 0.1, rootMargin = '0px' }) {
  const [isVisible, setIsVisible] = useState(false);
  const ref = useRef(null);

  useEffect(() => {
    const observer = new IntersectionObserver(
      ([entry]) => {
        if (entry.isIntersecting) {
          setIsVisible(true);
          observer.disconnect();
        }
      },
      {
        threshold,
        rootMargin,
      }
    );

    if (ref.current) {
      observer.observe(ref.current);
    }

    return () => {
      if (ref.current) {
        observer.unobserve(ref.current);
      }
    };
  }, [threshold, rootMargin]);

  return (
    <div
      ref={ref}
      className={`transition-opacity duration-1000 ease-out ${
        isVisible ? 'opacity-100' : 'opacity-0'
      }`}
    >
      {children}
    </div>
  );
}

// Usage
function App() {
  return (
    <div className="space-y-32 py-20">
      <FadeInOnScroll>
        <div className="bg-white p-6 rounded-lg shadow-lg">
          <h2 className="text-2xl font-bold mb-4">Section 1</h2>
          <p>This section fades in when it enters the viewport.</p>
        </div>
      </FadeInOnScroll>

      <FadeInOnScroll>
        <div className="bg-white p-6 rounded-lg shadow-lg">
```

```
            <h2 className="text-2xl font-bold mb-4">Section 2</h2>
            <p>This section also fades in when scrolled into view.</p>
          </div>
        </FadeInOnScroll>

        <FadeInOnScroll threshold={0.5} rootMargin="20px">
          <div className="bg-white p-6 rounded-lg shadow-lg">
            <h2 className="text-2xl font-bold mb-4">Section 3</h2>
            <p>This section has custom threshold and rootMargin values.</p>
          </div>
        </FadeInOnScroll>
      </div>
    );
}
```

2. **GSAP Integration**:

```
import React, { useEffect, useRef } from 'react';
import { gsap } from 'gsap';
import { ScrollTrigger } from 'gsap/ScrollTrigger';

// Register ScrollTrigger plugin
gsap.registerPlugin(ScrollTrigger);

function AdvancedAnimation() {
  const containerRef = useRef(null);
  const headerRef = useRef(null);
  const paragraphRef = useRef(null);
  const imageRef = useRef(null);

  useEffect(() => {
    const container = containerRef.current;
    const header = headerRef.current;
    const paragraph = paragraphRef.current;
    const image = imageRef.current;

    // Timeline for sequenced animations
    const tl = gsap.timeline({
      scrollTrigger: {
        trigger: container,
        start: 'top 80%',
        end: 'bottom 20%',
        toggleActions: 'play none none reverse',
        markers: false,
      },
    });

    tl.from(header, {
      y: 50,
      opacity: 0,
      duration: 0.6,
      ease: 'power3.out',
```

```
    })
      .from(
        paragraph,
        {
          y: 30,
          opacity: 0,
          duration: 0.6,
          ease: 'power3.out',
        },
        '-=0.3'
      )
      .from(
        image,
        {
          scale: 0.8,
          opacity: 0,
          duration: 0.8,
          ease: 'power2.out',
        },
        '-=0.3'
      );

    return () => {
      if (tl.scrollTrigger) {
        tl.scrollTrigger.kill();
      }
      tl.kill();
    };
  }, []);

  return (
    <div ref={containerRef} className="max-w-3xl mx-auto p-6">
      <h2 ref={headerRef} className="text-3xl font-bold mb-4">
        GSAP Animation Example
      </h2>
      <p ref={paragraphRef} className="text-lg mb-6">
        This section animates with GSAP's ScrollTrigger for advanced
effects.
        Each element animates in sequence as you scroll.
      </p>
      <div
        ref={imageRef}
        className="bg-gradient-to-r from-blue-500 to-purple-500 rounded-lg
h-64 w-full"
      ></div>
    </div>
  );
}
```

3. **CSS Grid Animation**:

```jsx
import React, { useState } from 'react';

function GridAnimation() {
  const [expanded, setExpanded] = useState(false);

  return (
    <div className="p-6">
      <button
        className="mb-6 px-4 py-2 bg-blue-500 text-white rounded hover:bg-
blue-600 transition-colors"
        onClick={() => setExpanded(!expanded)}
      >
        {expanded ? 'Collapse' : 'Expand'} Grid
      </button>

      <div
        className={`
          grid grid-cols-2 md:grid-cols-3 lg:grid-cols-4 gap-4
          transition-all duration-500 ease-in-out
          ${expanded ? 'grid-rows-[100px_100px_100px]' : 'grid-rows-
[100px]'}
        `}
      >
        {[...Array(12)].map((_, i) => (
          <div
            key={i}
            className={`
              rounded-lg bg-gradient-to-br from-indigo-500 to-purple-500
              flex items-center justify-center text-white font-bold
              transition-all duration-500 ease-in-out
              ${
                i >= 4 && !expanded
                  ? 'opacity-0 scale-95'
                  : 'opacity-100 scale-100'
              }
            `}
            style={{
              transitionDelay: `${(i % 4) * 50}ms`,
            }}
          >
            Item {i + 1}
          </div>
        ))}
      </div>
    </div>
  );
}
```

4. **Animated Micro-Interactions**:

```jsx
import React, { useState } from 'react';

// Animated Toggle Button
function Toggle({ label, onChange, checked }) {
  return (
    <label className="inline-flex items-center cursor-pointer">
      <div className="relative">
        <input
          type="checkbox"
          className="sr-only"
          checked={checked}
          onChange={onChange}
        />
        <div
          className={`
          block w-14 h-8 rounded-full transition-colors duration-300 ease-
in-out
          ${checked ? 'bg-blue-500' : 'bg-gray-300'}
        `}
        ></div>
        <div
          className={`
          absolute left-1 top-1 bg-white w-6 h-6 rounded-full transition-
transform duration-300 ease-in-out
          ${checked ? 'transform translate-x-6' : ''}
        `}
        ></div>
      </div>
      <span className="ml-3 text-gray-700">{label}</span>
    </label>
  );
}

// Animated Button with Icon
function IconButton({ icon, label, onClick }) {
  const [isAnimating, setIsAnimating] = useState(false);

  const handleClick = () => {
    setIsAnimating(true);
    setTimeout(() => setIsAnimating(false), 700);
    onClick?.();
  };

  return (
    <button
      className="px-4 py-2 bg-purple-500 text-white rounded-md flex items-
center space-x-2 overflow-hidden relative"
      onClick={handleClick}
    >
      <span
        className={`
        transform transition-transform duration-700 ease-in-out
        ${isAnimating ? 'rotate-360' : ''}
```

```
        `}
        >
          {icon}
        </span>
        <span>{label}</span>

        {isAnimating && (
          <span className="absolute inset-0 bg-white opacity-20 animate-
ripple"></span>
        )}
      </button>
    );
}

// Animated Input
function AnimatedInput({ label, placeholder }) {
  const [isFocused, setIsFocused] = useState(false);
  const [value, setValue] = useState('');

  return (
    <div className="relative">
      <label
        className={`
          absolute left-3 transition-all duration-300 ease-in-out pointer-
events-none
          ${
            isFocused || value
              ? 'transform -translate-y-6 scale-75 text-blue-500'
              : 'top-2 text-gray-500'
          }
        `}
      >
        {label}
      </label>
      <input
        type="text"
        className="block w-full border-0 border-b-2 border-gray-300
focus:border-blue-500 px-3 pt-4 pb-2 focus:outline-none transition-colors
duration-300"
        placeholder={isFocused ? placeholder : ''}
        onFocus={() => setIsFocused(true)}
        onBlur={() => setIsFocused(false)}
        value={value}
        onChange={(e) => setValue(e.target.value)}
      />
    </div>
  );
}

// Demo All Micro-interactions
function MicroInteractions() {
  const [isToggled, setIsToggled] = useState(false);

  return (
```

```
      <div className="space-y-8 p-6 max-w-xl mx-auto">
        <div>
          <h3 className="text-lg font-semibold mb-3">Animated Toggle</h3>
          <Toggle
            label="Enable Feature"
            checked={isToggled}
            onChange={() => setIsToggled(!isToggled)}
          />
        </div>

        <div>
          <h3 className="text-lg font-semibold mb-3">Animated Button</h3>
          <IconButton
            icon={
              <svg
                className="w-5 h-5"
                fill="none"
                stroke="currentColor"
                viewBox="0 0 24 24"
              >
                <path
                  strokeLinecap="round"
                  strokeLinejoin="round"
                  strokeWidth={2}
                  d="M12 4v16m8-8H4"
                />
              </svg>
            }
            label="Add Item"
            onClick={() => console.log('Clicked')}
          />
        </div>

        <div>
          <h3 className="text-lg font-semibold mb-3">Animated Input</h3>
          <AnimatedInput
            label="Email Address"
            placeholder="your.email@example.com"
          />
        </div>
      </div>
    );
  }
```

5. **SVG Animations**:

```
import React, { useState, useEffect } from 'react';

function SVGAnimation() {
  const [isAnimating, setIsAnimating] = useState(false);

  return (
```

```
      <div className="p-6 text-center">
        <button
          className="mb-6 px-4 py-2 bg-indigo-500 text-white rounded hover:bg-
indigo-600"
          onClick={() => setIsAnimating(!isAnimating)}
        >
          {isAnimating ? 'Pause' : 'Animate'} SVG
        </button>

        <svg
          width="200"
          height="200"
          viewBox="0 0 200 200"
          className="mx-auto"
        >
          {/* Circle path that draws itself */}
          <circle
            cx="100"
            cy="100"
            r="80"
            fill="none"
            stroke="#6366f1"
            strokeWidth="4"
            strokeDasharray="502"
            strokeDashoffset={isAnimating ? '0' : '502'}
            className="transition-all duration-1500 ease-in-out"
          />

          {/* Line that moves when activated */}
          <line
            x1="40"
            y1="100"
            x2="160"
            y2="100"
            stroke="#f43f5e"
            strokeWidth="4"
            strokeLinecap="round"
            className={`
              transition-transform duration-1000 ease-in-out
              ${isAnimating ? 'rotate-180' : ''}
            `}
            style={{ transformOrigin: '100px 100px' }}
          />

          {/* Pulsing rectangle */}
          <rect
            x="70"
            y="70"
            width="60"
            height="60"
            fill="#ec4899"
            rx="8"
            className={isAnimating ? 'animate-pulse' : ''}
          />
```

```
      </svg>

      {/* Animated progress bar */}
      <div className="max-w-xs mx-auto mt-10 h-3 bg-gray-200 rounded-full
overflow-hidden">
        <div
          className="h-full bg-green-500 transition-all duration-1000 ease-
out"
          style={{ width: isAnimating ? '100%' : '0%' }}
        ></div>
      </div>
    </div>
  );
}
```

Accessibility Considerations

Creating Accessible Styles

1. **Color Contrast**:

```
/* Poor contrast - fails WCAG guidelines */
.poor-contrast {
  color: #777;
  background-color: #fff;
}

/* Good contrast - passes WCAG AA (4.5:1 ratio) */
.good-contrast {
  color: #595959;
  background-color: #fff;
}

/* Excellent contrast - passes WCAG AAA (7:1 ratio) */
.excellent-contrast {
  color: #333;
  background-color: #fff;
}
```

Tailwind approach:

```
<!-- Poor contrast (avoid) -->
<p class="text-gray-400 bg-white">Low contrast text</p>

<!-- Good contrast (WCAG AA) -->
<p class="text-gray-600 bg-white">Better contrast text</p>

<!-- Excellent contrast (WCAG AAA) -->
<p class="text-gray-800 bg-white">High contrast text</p>
```

2. **Focus Styles**:

```
/* Basic focus style */
button:focus {
  outline: 2px solid #4f46e5;
  outline-offset: 2px;
}

/* Enhanced focus style for better visibility */
.enhanced-focus:focus {
  outline: none;
  box-shadow: 0 0 0 3px rgba(66, 153, 225, 0.5);
}

/* Hide outline only when using mouse, keep for keyboard */
.mixed-focus:focus {
  outline: 2px solid #4f46e5;
  outline-offset: 2px;
}

.mixed-focus:focus:not(:focus-visible) {
  outline: none;
}
```

Tailwind approach:

```
<!-- Default focus styles -->
<button
  class="px-4 py-2 bg-blue-500 text-white rounded focus:outline-none
focus:ring-2 focus:ring-blue-500 focus:ring-offset-2"
>
  Accessible Button
</button>

<!-- Focus styles with mouse/keyboard differentiation -->
<button
  class="px-4 py-2 bg-blue-500 text-white rounded focus:outline-none focus-
visible:ring-2 focus-visible:ring-blue-500 focus-visible:ring-offset-2"
>
  Smart Focus Button
</button>
```

3. **Skip Navigation Links**:

```
.skip-link {
  position: absolute;
  top: -40px;
```

```
  left: 0;
  padding: 8px 16px;
  background-color: #4f46e5;
  color: white;
  z-index: 100;
  transition: top 0.2s;
}

.skip-link:focus {
  top: 0;
}
```

Usage:

```
<a href="#main-content" class="skip-link">Skip to main content</a>

<!-- Rest of header and navigation -->

<main id="main-content">
  <!-- Main content here -->
</main>
```

4. **Reduced Motion**:

```
/* Default animation */
.animated-element {
  transition: transform 0.5s ease;
}

.animated-element:hover {
  transform: scale(1.1);
}

/* Reduced animation for users who prefer reduced motion */
@media (prefers-reduced-motion: reduce) {
  .animated-element {
    transition: none;
  }

  .animated-element:hover {
    transform: none;
  }
}
```

Tailwind approach:

```
<div
  class="transform hover:scale-110 transition duration-300 motion-
reduce:transform-none motion-reduce:transition-none"
>
  This element has animation that respects user preferences
</div>
```

Semantic HTML with Tailwind

1. **Proper Heading Structure**:

```
<!-- Good practice -->
<article>
  <h1 class="text-3xl font-bold mb-4">Main Article Title</h1>

  <section>
    <h2 class="text-2xl font-semibold mb-3">Section Title</h2>
    <p class="mb-4">Section content goes here...</p>

    <div>
      <h3 class="text-xl font-medium mb-2">Subsection Title</h3>
      <p class="mb-4">Subsection content...</p>
    </div>
  </section>
</article>

<!-- Avoid this (incorrect structure) -->
<article>
  <div class="text-3xl font-bold mb-4">Main Article Title</div>

  <section>
    <div class="text-2xl font-semibold mb-3">Section Title</div>
    <p class="mb-4">Section content goes here...</p>

    <div>
      <div class="text-xl font-medium mb-2">Subsection Title</div>
      <p class="mb-4">Subsection content...</p>
    </div>
  </section>
</article>
```

2. **Forms and Labels**:

```
<!-- Bad practice -->
<div class="mb-4">
  <span class="block text-gray-700 mb-2">Email:</span>
  <input
    type="email"
```

```
    class="w-full p-2 border rounded"
    placeholder="Enter your email"
  />
</div>

<!-- Good practice -->
<div class="mb-4">
  <label for="email" class="block text-gray-700 mb-2">Email:</label>
  <input
    id="email"
    type="email"
    class="w-full p-2 border rounded"
    aria-describedby="email-help"
  />
  <p id="email-help" class="text-sm text-gray-500 mt-1">
    We'll never share your email.
  </p>
</div>
```

3. **ARIA Attributes with Tailwind**:

```
<!-- Tabs implementation -->
<div role="tablist" class="flex border-b">
  <button
    id="tab-1"
    role="tab"
    aria-selected="true"
    aria-controls="panel-1"
    class="px-4 py-2 text-blue-600 border-b-2 border-blue-600"
  >
    Tab 1
  </button>
  <button
    id="tab-2"
    role="tab"
    aria-selected="false"
    aria-controls="panel-2"
    class="px-4 py-2 text-gray-500 hover:text-gray-700"
  >
    Tab 2
  </button>
</div>

<div id="panel-1" role="tabpanel" aria-labelledby="tab-1" class="p-4">
  Panel 1 content
</div>

<div id="panel-2" role="tabpanel" aria-labelledby="tab-2" class="p-4
hidden">
  Panel 2 content
</div>
```

4. **Accessible Toggle Components**:

```
function AccessibleToggle({ label, checked, onChange }) {
  const id = useId();

  return (
    <div className="flex items-center">
      <button
        role="switch"
        id={id}
        aria-checked={checked}
        aria-label={label}
        className={`
          relative inline-flex flex-shrink-0 h-6 w-11 border-2 border-
transparent rounded-full
          transition-colors duration-200 ease-in-out focus:outline-none
focus:ring-2 focus:ring-blue-500 focus:ring-offset-2
          ${checked ? 'bg-blue-600' : 'bg-gray-200'}
        `}
        onClick={() => onChange(!checked)}
      >
        <span
          className={`
            pointer-events-none inline-block h-5 w-5 rounded-full bg-white
shadow ring-0
            transition duration-200 ease-in-out
            ${checked ? 'translate-x-5' : 'translate-x-0'}
          `}
        />
      </button>
      <span className="ml-3" id={`${id}-label`}>
        {label}
      </span>
    </div>
  );
}
```

Screen Reader Considerations

1. **Hidden Text for Screen Readers**:

```
.sr-only {
  position: absolute;
  width: 1px;
  height: 1px;
  padding: 0;
  margin: -1px;
  overflow: hidden;
  clip: rect(0, 0, 0, 0);
```

```
  white-space: nowrap;
  border-width: 0;
}
```

Tailwind already provides this utility:

```
<button class="bg-blue-500 p-2 rounded text-white">
  <svg class="h-5 w-5" fill="none" viewBox="0 0 24 24"
stroke="currentColor">
    <path
      stroke-linecap="round"
      stroke-linejoin="round"
      stroke-width="2"
      d="M15 19l-7-7 7-7"
    />
  </svg>
  <span class="sr-only">Previous Page</span>
</button>
```

2. **ARIA Live Regions**:

```
<!-- Polite announcement (waits for screen reader to finish) -->
<div aria-live="polite" aria-atomic="true" class="sr-only">
  <div id="notification">3 new messages</div>
</div>

<!-- Assertive announcement (interrupts screen reader) -->
<div aria-live="assertive" aria-atomic="true" class="sr-only">
  <div id="alert">Error: Form submission failed</div>
</div>
```

JavaScript to update:

```
function showNotification(message) {
  document.getElementById('notification').textContent = message;
}

function showAlert(message) {
  document.getElementById('alert').textContent = message;
}
```

3. **Visually Hidden Elements that Take Space**:

```
.visually-hidden {
  clip: rect(1px, 1px, 1px, 1px);
```

```
  clip-path: inset(50%);
  height: 1px;
  width: 1px;
  margin: -1px;
  overflow: hidden;
  padding: 0;
  position: absolute;
}
```

Example use case:

```
<table class="w-full">
  <caption class="visually-hidden">
    Monthly expenses breakdown
  </caption>
  <thead>
    <!-- table headers -->
  </thead>
  <tbody>
    <!-- table data -->
  </tbody>
</table>
```

Testing Accessibility

1. **Keyboard Navigation Testing**:

   Ensure all interactive elements are focusable and usable with keyboard:

   - Tab key navigates between elements
   - Enter or Space activates buttons
   - Arrow keys navigate within components (like menus)
   - Escape dismisses modals/popups

2. **Color Contrast Tools**:

```
// ContrastChecker.jsx component
function ContrastChecker({ backgroundColor, textColor }) {
  // Using WCAG contrast ratio formula
  const getContrastRatio = (bg, text) => {
    // Convert hex to RGB
    const hexToRgb = (hex) => {
      const r = parseInt(hex.slice(1, 3), 16);
      const g = parseInt(hex.slice(3, 5), 16);
      const b = parseInt(hex.slice(5, 7), 16);
      return [r, g, b];
    };

    // Calculate relative luminance
```

```
    const getLuminance = (rgb) => {
      const [r, g, b] = rgb.map((v) => {
        v /= 255;
        return v <= 0.03928 ? v / 12.92 : Math.pow((v + 0.055) / 1.055,
2.4);
      });
      return 0.2126 * r + 0.7152 * g + 0.0722 * b;
    };

    const bgRgb = hexToRgb(bg);
    const textRgb = hexToRgb(text);

    const bgLuminance = getLuminance(bgRgb);
    const textLuminance = getLuminance(textRgb);

    const ratio =
      bgLuminance > textLuminance
        ? (bgLuminance + 0.05) / (textLuminance + 0.05)
        : (textLuminance + 0.05) / (bgLuminance + 0.05);

    return ratio.toFixed(2);
  };

  const ratio = getContrastRatio(backgroundColor, textColor);

  // WCAG standards
  const passesAA = ratio >= 4.5;
  const passesAAA = ratio >= 7;

  return (
    <div className="p-4 border rounded-lg">
      <div
        className="p-6 mb-4 rounded text-center"
        style={{ backgroundColor, color: textColor }}
      >
        Sample Text
      </div>

      <div className="space-y-2">
        <p>
          Contrast Ratio: <strong>{ratio}</strong>
        </p>
        <p>
          WCAG AA:{' '}
          <span className={passesAA ? 'text-green-600' : 'text-red-600'}>
            {passesAA ? 'Pass' : 'Fail'}
          </span>
        </p>
        <p>
          WCAG AAA:{' '}
          <span className={passesAAA ? 'text-green-600' : 'text-red-600'}>
            {passesAAA ? 'Pass' : 'Fail'}
          </span>
        </p>
```

```
      </div>
    </div>
  );
}

// Usage
<ContrastChecker backgroundColor="#ffffff" textColor="#333333" />;
```

3. **Automated Testing Tools**:

```
// Using axe-core with Jest
import React from 'react';
import { render } from '@testing-library/react';
import { axe, toHaveNoViolations } from 'jest-axe';
import Button from './Button';

expect.extend(toHaveNoViolations);

describe('Button component', () => {
  it('should not have accessibility violations', async () => {
    const { container } = render(<Button>Click me</Button>);
    const results = await axe(container);
    expect(results).toHaveNoViolations();
  });
});
```

Performance Optimization Strategies

CSS Performance Best Practices

1. **Selector Efficiency**:

```
/* Inefficient selector (evaluates right-to-left) */
body div ul li a.nav-link {
  color: red;
}

/* More efficient selector */
.nav-link {
  color: red;
}
```

2. **Reducing Repaints and Reflows**:

```
/* Properties that cause reflow when changed */
.causes-reflow {
  width: 100px;
```

```
  height: 100px;
  padding: 20px;
  margin: 20px;
  position: absolute;
  top: 50px;
  left: 50px;
}

/* Properties that only cause repaint (better) */
.causes-repaint {
  color: red;
  background-color: blue;
  text-shadow: 1px 1px 2px black;
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.5);
}

/* Properties optimized for GPU acceleration (best) */
.gpu-accelerated {
  transform: translateZ(0);
  will-change: transform, opacity;
}
```

3. **CSS Containment**:

```
/* Isolate rendering performance impact */
.contained-component {
  contain: layout paint style;
  /* or for all containment */
  contain: strict;
}
```

4. **Avoiding @import**:

```
/* Avoid this - creates waterfall loading */
@import url('other-styles.css');

/* Better approach - use multiple <link> tags in HTML */
<link rel="stylesheet" href="main.css">
<link rel="stylesheet" href="other-styles.css">
```

5. **Critical CSS**:

```
<head>
  <!-- Inline critical CSS for fast first paint -->
  <style>
    /* Critical CSS for above-the-fold content */
    body {
```

```
      font-family: sans-serif;
      margin: 0;
    }
    header {
      background: #f5f5f5;
      padding: 1rem;
    }
    .hero {
      height: 80vh;
      background: url('hero.jpg');
    }
  </style>

  <!-- Load non-critical CSS asynchronously -->
  <link
    rel="preload"
    href="styles.css"
    as="style"
    onload="this.onload=null;this.rel='stylesheet'"
  />
  <noscript><link rel="stylesheet" href="styles.css" /></noscript>
</head>
```

Tailwind-Specific Optimizations

1. **JIT (Just-in-Time) Mode**:

```
// tailwind.config.js - JIT is default in Tailwind v3+
module.exports = {
  mode: 'jit', // Only needed for Tailwind v2.x
  // ... rest of config
};
```

2. **Content Configuration**:

```
// tailwind.config.js
module.exports = {
  content: [
    // Be specific about which files to scan
    './src/pages/**/*.{js,jsx,ts,tsx}',
    './src/components/**/*.{js,jsx,ts,tsx}',
    './src/layouts/**/*.{js,jsx,ts,tsx}',
    // Avoid including files that don't use Tailwind classes
    // './src/utils/**/*.{js,ts}', // Don't include utility files
  ],
  // ... rest of config
};
```

3. **Safelist for Dynamic Classes**:

```
// tailwind.config.js
module.exports = {
  // ... other config
  safelist: [
    // Simple values
    'bg-red-500',
    'text-center',

    // Regular expression patterns
    {
      pattern: /bg-(red|green|blue)-(100|200|300|400|500)/,
      variants: ['hover', 'focus', 'lg:hover'],
    },
  ],
};
```

4. **Disabling Unused Core Plugins**:

```
// tailwind.config.js
module.exports = {
  // ... other config
  corePlugins: {
    float: false, // Disable if not using float utilities
    clear: false, // Disable if not using clear utilities
    skew: false, // Disable if not using skew transforms
    // ... other unused plugins
  },
};
```

5. **Optimizing for Production**:

```
# In package.json scripts
"build:css": "NODE_ENV=production tailwindcss -i src/styles/main.css -o
dist/styles.css --minify"
```

Image and Asset Optimization

1. **Responsive Images with srcset**:

```
<!-- Responsive images based on viewport width -->
<img
  src="image-800w.jpg"
  srcset="image-400w.jpg 400w, image-800w.jpg 800w, image-1200w.jpg 1200w"
  sizes="(max-width: 600px) 100vw, (max-width: 1200px) 50vw, 800px"
```

```
  alt="Responsive image"
/>

<!-- Responsive images based on pixel density -->
<img
  src="image.jpg"
  srcset="image.jpg 1x, image@2x.jpg 2x, image@3x.jpg 3x"
  alt="Density-aware image"
/>
```

2. **Modern Image Formats**:

```
<!-- Using picture element for format switching -->
<picture>
  <source srcset="image.avif" type="image/avif" />
  <source srcset="image.webp" type="image/webp" />
  <img src="image.jpg" alt="Image with modern format fallback" />
</picture>
```

3. **Lazy Loading**:

```
<!-- Native lazy loading -->
<img src="image.jpg" loading="lazy" alt="Lazy loaded image" />

<!-- For older browsers, you can use Intersection Observer in JavaScript -->
```

4. **CSS Background Image Optimization**:

```
/* Different image resolutions for different screen sizes */
.hero {
  background-image: url('hero-small.jpg');
}

@media (min-width: 768px) {
  .hero {
    background-image: url('hero-medium.jpg');
  }
}

@media (min-width: 1200px) {
  .hero {
    background-image: url('hero-large.jpg');
  }
}
```

Loading Strategy Optimization

1. **CSS Code Splitting**:

```
// webpack.config.js example
module.exports = {
  // ...
  optimization: {
    splitChunks: {
      cacheGroups: {
        styles: {
          name: 'styles',
          test: /\.css$/,
          chunks: 'all',
          enforce: true,
        },
      },
    },
  },
};
```

2. **Loading CSS Based on Media Queries**:

```
<!-- Only download if the condition is met -->
<link rel="stylesheet" href="desktop.css" media="(min-width: 1024px)" />
<link rel="stylesheet" href="print.css" media="print" />
```

3. **Resource Hints**:

```
<!-- Preconnect to CDN domains -->
<link rel="preconnect" href="https://cdn.example.com" />

<!-- Preload critical CSS -->
<link rel="preload" href="critical.css" as="style" />

<!-- Prefetch likely-to-be-used resources -->
<link rel="prefetch" href="next-page.css" />
```

4. **Loading Non-Critical CSS Asynchronously**:

```
<head>
  <!-- Critical CSS inline -->
  <style>
    /* Critical styles here */
  </style>

  <!-- Non-critical CSS loaded asynchronously -->
  <link
```

```
    rel="preload"
    href="non-critical.css"
    as="style"
    onload="this.onload=null;this.rel='stylesheet'"
  />
  <noscript><link rel="stylesheet" href="non-critical.css" /></noscript>
</head>
```

Common Pitfalls and Solutions

CSS Pitfalls

1. **Specificity Issues**:

   Problem:

```
/* Highly specific selector */
#header .navigation ul li.active a {
  color: red;
}

/* This won't work because the above selector is more specific */
.active-link {
  color: blue !important; /* Bad practice to use !important */
}
```

   Solution:

```
/* Use classes with consistent, low specificity */
.nav-link {
  color: gray;
}

.nav-link.active {
  color: blue;
}
```

2. **Z-index Stacking Context**:

   Problem:

```
/* Elements in different stacking contexts */
.dropdown {
  position: relative;
  z-index: 100;
}
```

```
.modal {
  /* This will still appear below .dropdown even with higher z-index
     if .dropdown's parent creates a new stacking context */
  z-index: 1000;
}
```

Solution:

```
/* Make sure both elements are in the same stacking context,
   or manage stacking contexts hierarchically */
.stacking-context-1 {
  position: relative;
  z-index: 10;
}

.stacking-context-2 {
  position: relative;
  z-index: 20; /* Higher than stacking-context-1 */
}

/* Now elements inside stacking-context-2 will appear above
   elements in stacking-context-1, regardless of their z-index */
```

3. **Box Model Confusion**:

Problem:

```
.box {
  width: 200px;
  padding: 20px;
  border: 2px solid black;
  /* Total width: 200px + 40px + 4px = 244px */
}
```

Solution:

```
/* Use box-sizing: border-box globally */
*,
*::before,
*::after {
  box-sizing: border-box;
}

.box {
  width: 200px;
  padding: 20px;
  border: 2px solid black;
```

```
  /* Total width: 200px (content area shrinks to accommodate padding and
border) */
}
```

4. **Unintended CSS Inheritance**:

   Problem:

```
body {
  font-size: 16px;
}

/* All elements inside containers will have white text */
.container {
  color: white;
}
```

   Solution:

```
/* Be specific about what should inherit */
.container {
  /* Only apply color to direct text */
  color: white;
}

.container-text {
  color: white;
}

/* Override for specific elements */
.container .special-text {
  color: yellow;
}
```

5. **Media Query Overlaps**:

   Problem:

```
/* These overlap between 768px and 1024px */
@media (max-width: 1024px) {
  .element {
    font-size: 14px;
  }
}

@media (min-width: 768px) {
  .element {
```

```
    font-size: 16px;
  }
}
```

Solution:

```
/* Be explicit about ranges */
@media (max-width: 767px) {
  .element {
    font-size: 14px;
  }
}

@media (min-width: 768px) and (max-width: 1023px) {
  .element {
    font-size: 15px;
  }
}

@media (min-width: 1024px) {
  .element {
    font-size: 16px;
  }
}
```

Tailwind Pitfalls

1. **Utility Bloat in HTML**:

   Problem:

```
<button
  class="px-4 py-2 bg-blue-500 hover:bg-blue-600 text-white font-semibold
rounded shadow hover:shadow-md transition-all duration-300 ease-in-out
focus:outline-none focus:ring-2 focus:ring-blue-500 focus:ring-opacity-50"
>
  Button with too many classes
</button>
```

   Solution:

```
<!-- Extract component -->
<button>Simple Button</button>

<!-- Or use @apply in CSS -->
<button class="btn-primary">Simple Button</button>
```

```
@layer components {
  .btn-primary {
    @apply px-4 py-2 bg-blue-500 hover:bg-blue-600 text-white font-semibold
rounded shadow hover:shadow-md transition-all duration-300 ease-in-out
focus:outline-none focus:ring-2 focus:ring-blue-500 focus:ring-opacity-50;
  }
}
```

2. **Dynamic Class Names**:

   Problem:

```
// This won't work with PurgeCSS/JIT
const bg = isPrimary ? 'bg-blue-500' : 'bg-gray-500';
<div className={`${bg} text-white p-4`}></div>;
```

   Solution:

```
// Use complete classes
<div
  className={`${isPrimary ? 'bg-blue-500' : 'bg-gray-500'} text-white p-4`}
></div>;

// Or add to safelist in tailwind.config.js
module.exports = {
  // ...
  safelist: ['bg-blue-500', 'bg-gray-500'],
};
```

3. **Default vs. Extended Theme**:

   Problem:

```
// tailwind.config.js
module.exports = {
  theme: {
    colors: {
      // This REPLACES all default colors
      primary: '#3b82f6',
      secondary: '#10b981',
    },
  },
};
```

   Solution:

```
// tailwind.config.js
const colors = require('tailwindcss/colors');

module.exports = {
  theme: {
    // Keep defaults but override specific colors
    colors: {
      ...colors,
      primary: '#3b82f6',
      secondary: '#10b981',
    },

    // Or better yet, use extend
    extend: {
      colors: {
        primary: '#3b82f6',
        secondary: '#10b981',
      },
    },
  },
};
```

4. **Arbitrary Value Syntax Errors**:

   Problem:

```
<!-- These won't work -->
<div class="w-[calc(100%-20px)]"></div>
<div class="bg-[rgba(255,255,255,0.5)]"></div>
```

   Solution:

```
<!-- Fix the syntax -->
<div class="w-[calc(100%_-_20px)]"></div>
<div class="bg-[rgba(255,255,255,0.5)]"></div>
```

5. **Misunderstanding Responsive Prefixes**:

   Problem:

```
<!-- This applies padding at all screen sizes -->
<div class="lg:p-4">Not what you'd expect</div>
```

   Solution:

```
<!-- This starts with 0 padding, then adds padding at lg breakpoint -->
<div class="p-0 lg:p-4">Starts with no padding, adds at lg</div>

<!-- OR: Different padding at different breakpoints -->
<div class="p-2 md:p-3 lg:p-4">Gradually increasing padding</div>
```

General Web Development Pitfalls

1. **Ignoring Mobile Experience**:

```
/* Develop for mobile first */
.element {
  /* Mobile styles (default) */
  padding: 1rem;
  font-size: 1rem;
}

@media (min-width: 768px) {
  .element {
    /* Tablet enhancements */
    padding: 1.5rem;
    font-size: 1.1rem;
  }
}

@media (min-width: 1024px) {
  .element {
    /* Desktop enhancements */
    padding: 2rem;
    font-size: 1.25rem;
  }
}
```

2. **Neglecting Web Performance**:

   Solution checklist:

   - Minimize HTTP requests
   - Optimize and compress images
   - Use lazy loading for off-screen content
   - Minify CSS and JavaScript
   - Use efficient selectors
   - Implement critical CSS
   - Defer non-critical CSS and JavaScript
   - Optimize web fonts loading

3. **Poor Accessibility Practices**:

Accessibility checklist:

- Proper semantic HTML structure
- Sufficient color contrast
- Keyboard navigation support
- Accessible form labels and controls
- Meaningful alt text for images
- ARIA attributes where appropriate
- Skip navigation links
- Proper heading hierarchy
- Focus management for interactive components
- Support for screen readers
- Respect user preferences (reduced motion, etc.)

Practice Projects

Basic Projects

Simple Profile Card

HTML Structure:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Profile Card</title>
    <link rel="stylesheet" href="styles.css" />
  </head>
  <body>
    <div class="card">
      <div class="card-header">
        <img
          src="https://randomuser.me/api/portraits/men/32.jpg"
          alt="Profile Picture"
          class="avatar"
        />
        <h2 class="name">John Doe</h2>
        <p class="title">Frontend Developer</p>
      </div>
      <div class="card-body">
        <p class="bio">
          Passionate developer with experience in HTML, CSS, and JavaScript.
          Loves building user-friendly interfaces and learning new technologies.
        </p>
        <div class="stats">
          <div class="stat">
            <span class="stat-value">142</span>
            <span class="stat-label">Projects</span>
          </div>
```

```
            <div class="stat">
              <span class="stat-value">8.5k</span>
              <span class="stat-label">Followers</span>
            </div>
            <div class="stat">
              <span class="stat-value">57</span>
              <span class="stat-label">Likes</span>
            </div>
          </div>
        </div>
        <div class="card-footer">
          <button class="btn btn-primary">Follow</button>
          <button class="btn btn-secondary">Message</button>
        </div>
      </div>
    </body>
</html>
```

CSS Implementation:

```
/* Base Styles */
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

body {
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
  background-color: #f5f7fb;
  display: flex;
  justify-content: center;
  align-items: center;
  min-height: 100vh;
  padding: 1rem;
}

/* Card Component */
.card {
  width: 100%;
  max-width: 360px;
  background-color: white;
  border-radius: 12px;
  box-shadow: 0 10px 25px rgba(0, 0, 0, 0.05);
  overflow: hidden;
}

.card-header {
  padding: 1.5rem;
  background-color: #5c6bc0;
  color: white;
  text-align: center;
```

```
}

.avatar {
  width: 100px;
  height: 100px;
  border-radius: 50%;
  border: 4px solid rgba(255, 255, 255, 0.3);
  margin-bottom: 1rem;
  object-fit: cover;
}

.name {
  font-size: 1.5rem;
  font-weight: 600;
  margin-bottom: 0.25rem;
}

.title {
  font-size: 0.875rem;
  opacity: 0.8;
}

.card-body {
  padding: 1.5rem;
}

.bio {
  color: #555;
  line-height: 1.5;
  margin-bottom: 1.5rem;
}

.stats {
  display: flex;
  justify-content: space-between;
  text-align: center;
}

.stat-value {
  display: block;
  font-size: 1.25rem;
  font-weight: 600;
  color: #333;
}

.stat-label {
  font-size: 0.75rem;
  color: #777;
}

.card-footer {
  padding: 1.5rem;
  border-top: 1px solid #f0f0f0;
  display: flex;
```

```
  gap: 0.75rem;
}

.btn {
  flex: 1;
  padding: 0.75rem 1rem;
  border-radius: 6px;
  font-weight: 600;
  font-size: 0.875rem;
  cursor: pointer;
  transition: background-color 0.2s, transform 0.1s;
  border: none;
}

.btn:active {
  transform: translateY(1px);
}

.btn-primary {
  background-color: #5c6bc0;
  color: white;
}

.btn-primary:hover {
  background-color: #4d5ab3;
}

.btn-secondary {
  background-color: #f5f7fb;
  color: #5c6bc0;
}

.btn-secondary:hover {
  background-color: #e8ecf5;
}

/* Media Queries */
@media (max-width: 480px) {
  .card {
    max-width: 100%;
  }

  .stats {
    flex-direction: column;
    gap: 1rem;
  }
}
```

Tailwind Implementation:

```
<!DOCTYPE html>
<html lang="en">
```

```
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Profile Card</title>
    <script src="https://cdn.tailwindcss.com"></script>
  </head>
  <body class="bg-gray-100 flex justify-center items-center min-h-screen p-4">
    <div class="bg-white rounded-xl shadow-lg overflow-hidden w-full max-w-sm">
      <div class="bg-indigo-500 p-6 text-center text-white">
        <img
          src="https://randomuser.me/api/portraits/men/32.jpg"
          alt="Profile Picture"
          class="w-24 h-24 rounded-full border-4 border-white/30 mx-auto mb-4
object-cover"
        />
        <h2 class="text-xl font-semibold mb-1">John Doe</h2>
        <p class="text-sm text-white/80">Frontend Developer</p>
      </div>

      <div class="p-6">
        <p class="text-gray-600 leading-relaxed mb-6">
          Passionate developer with experience in HTML, CSS, and JavaScript.
          Loves building user-friendly interfaces and learning new technologies.
        </p>

        <div class="flex justify-between text-center mb-6 sm:mb-0">
          <div>
            <span class="block text-lg font-semibold text-gray-800">142</span>
            <span class="text-xs text-gray-500">Projects</span>
          </div>
          <div>
            <span class="block text-lg font-semibold text-gray-800">8.5k</span>
            <span class="text-xs text-gray-500">Followers</span>
          </div>
          <div>
            <span class="block text-lg font-semibold text-gray-800">57</span>
            <span class="text-xs text-gray-500">Likes</span>
          </div>
        </div>
      </div>

      <div class="p-6 border-t border-gray-100 flex gap-3">
        <button
          class="flex-1 bg-indigo-500 hover:bg-indigo-600 text-white font-semibold
py-3 px-4 rounded-md transition-colors active:translate-y-0.5"
        >
          Follow
        </button>
        <button
          class="flex-1 bg-gray-100 hover:bg-gray-200 text-indigo-500 font-
semibold py-3 px-4 rounded-md transition-colors active:translate-y-0.5"
        >
          Message
        </button>
```

```
      </div>
    </div>
  </body>
</html>
```

Responsive Navbar

HTML Structure:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Responsive Navbar</title>
    <link rel="stylesheet" href="styles.css" />
  </head>
  <body>
    <header class="navbar">
      <div class="navbar-container">
        <a href="#" class="navbar-logo"> Company Name </a>

        <button class="navbar-toggle">
          <span class="toggle-bar"></span>
          <span class="toggle-bar"></span>
          <span class="toggle-bar"></span>
        </button>

        <nav class="navbar-menu">
          <ul class="navbar-nav">
            <li class="nav-item active">
              <a href="#" class="nav-link">Home</a>
            </li>
            <li class="nav-item">
              <a href="#" class="nav-link">About</a>
            </li>
            <li class="nav-item">
              <a href="#" class="nav-link">Services</a>
            </li>
            <li class="nav-item">
              <a href="#" class="nav-link">Portfolio</a>
            </li>
            <li class="nav-item">
              <a href="#" class="nav-link">Contact</a>
            </li>
          </ul>
        </nav>
      </div>
    </header>

    <main class="main-content">
```

```
      <h1>Responsive Navbar</h1>
      <p>Resize the window to see the navbar change.</p>
    </main>

    <script>
      // Toggle mobile menu
      const navToggle = document.querySelector('.navbar-toggle');
      const navMenu = document.querySelector('.navbar-menu');

      navToggle.addEventListener('click', () => {
        navMenu.classList.toggle('active');
        navToggle.classList.toggle('active');
      });
    </script>
  </body>
</html>
```

CSS Implementation:

```
/* Base Styles */
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

body {
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
  line-height: 1.6;
  color: #333;
}

a {
  text-decoration: none;
}

ul {
  list-style: none;
}

/* Navbar Styles */
.navbar {
  background-color: #2c3e50;
  position: sticky;
  top: 0;
  z-index: 1000;
  box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
}

.navbar-container {
  max-width: 1200px;
  margin: 0 auto;
```

```
  padding: 0 1rem;
  display: flex;
  justify-content: space-between;
  align-items: center;
  height: 70px;
}

.navbar-logo {
  color: white;
  font-size: 1.5rem;
  font-weight: 700;
}

.navbar-toggle {
  display: none;
  flex-direction: column;
  justify-content: space-between;
  width: 30px;
  height: 21px;
  background: transparent;
  border: none;
  cursor: pointer;
}

.toggle-bar {
  width: 100%;
  height: 3px;
  background-color: white;
  border-radius: 3px;
  transition: all 0.3s ease-in-out;
}

.navbar-menu {
  display: flex;
}

.navbar-nav {
  display: flex;
}

.nav-item {
  margin-left: 2rem;
}

.nav-link {
  color: #ecf0f1;
  font-size: 1rem;
  font-weight: 500;
  padding: 0.5rem 0;
  transition: color 0.3s ease;
  position: relative;
}

.nav-link::after {
```

```
    content: '';
    position: absolute;
    bottom: 0;
    left: 0;
    width: 0;
    height: 2px;
    background-color: #3498db;
    transition: width 0.3s ease;
  }

  .nav-link:hover {
    color: #3498db;
  }

  .nav-link:hover::after,
  .active .nav-link::after {
    width: 100%;
  }

  .active .nav-link {
    color: #3498db;
  }

  /* Main Content Styles */
  .main-content {
    max-width: 1200px;
    margin: 0 auto;
    padding: 3rem 1rem;
  }

  .main-content h1 {
    margin-bottom: 1rem;
  }

  /* Media Queries */
  @media (max-width: 768px) {
    .navbar-toggle {
      display: flex;
    }

    .navbar-menu {
      position: fixed;
      top: 70px;
      left: 0;
      width: 100%;
      height: 0;
      background-color: #2c3e50;
      overflow: hidden;
      transition: height 0.3s ease-in-out;
    }

    .navbar-menu.active {
      height: calc(100vh - 70px);
    }
```

```
  .navbar-nav {
    flex-direction: column;
    width: 100%;
    padding: 1rem;
  }

  .nav-item {
    margin: 0;
    width: 100%;
    border-bottom: 1px solid rgba(255, 255, 255, 0.1);
  }

  .nav-link {
    display: block;
    padding: 1rem 0;
  }

  .navbar-toggle.active .toggle-bar:nth-child(1) {
    transform: translateY(9px) rotate(45deg);
  }

  .navbar-toggle.active .toggle-bar:nth-child(2) {
    opacity: 0;
  }

  .navbar-toggle.active .toggle-bar:nth-child(3) {
    transform: translateY(-9px) rotate(-45deg);
  }
}
```

Tailwind Implementation:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Responsive Navbar</title>
    <script src="https://cdn.tailwindcss.com"></script>
  </head>
  <body class="text-gray-800">
    <header class="bg-gray-800 sticky top-0 z-50 shadow-md">
      <div class="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">
        <div class="flex justify-between h-16">
          <!-- Logo -->
          <div class="flex items-center">
            <a href="#" class="text-white text-xl font-bold">Company Name</a>
          </div>

          <!-- Mobile menu button -->
          <div class="flex items-center md:hidden">
```

```
            <button
              id="mobile-menu-button"
              class="text-white focus:outline-none"
            >
              <svg
                class="h-6 w-6"
                xmlns="http://www.w3.org/2000/svg"
                fill="none"
                viewBox="0 0 24 24"
                stroke="currentColor"
              >
                <path
                  id="menu-icon"
                  stroke-linecap="round"
                  stroke-linejoin="round"
                  stroke-width="2"
                  d="M4 6h16M4 12h16M4 18h16"
                />
                <path
                  id="close-icon"
                  class="hidden"
                  stroke-linecap="round"
                  stroke-linejoin="round"
                  stroke-width="2"
                  d="M6 18L18 6M6 6l12 12"
                />
              </svg>
            </button>
          </div>

          <!-- Desktop Navigation -->
          <nav class="hidden md:flex md:items-center">
            <ul class="flex space-x-8">
              <li>
                <a
                  href="#"
                  class="text-blue-400 relative after:absolute after:w-full
after:h-0.5 after:bg-blue-400 after:bottom-0 after:left-0"
                  >Home</a
                >
              </li>
              <li>
                <a
                  href="#"
                  class="text-gray-300 hover:text-blue-400 transition-colors
relative after:absolute after:w-0 after:h-0.5 after:bg-blue-400 after:bottom-0
after:left-0 hover:after:w-full after:transition-all"
                  >About</a
                >
              </li>
              <li>
                <a
                  href="#"
                  class="text-gray-300 hover:text-blue-400 transition-colors
```

```
relative after:absolute after:w-0 after:h-0.5 after:bg-blue-400 after:bottom-0
after:left-0 hover:after:w-full after:transition-all"
                  >Services</a
                >
              </li>
              <li>
                <a
                  href="#"
                  class="text-gray-300 hover:text-blue-400 transition-colors
relative after:absolute after:w-0 after:h-0.5 after:bg-blue-400 after:bottom-0
after:left-0 hover:after:w-full after:transition-all"
                  >Portfolio</a
                >
              </li>
              <li>
                <a
                  href="#"
                  class="text-gray-300 hover:text-blue-400 transition-colors
relative after:absolute after:w-0 after:h-0.5 after:bg-blue-400 after:bottom-0
after:left-0 hover:after:w-full after:transition-all"
                  >Contact</a
                >
              </li>
            </ul>
          </nav>
        </div>
      </div>

      <!-- Mobile Navigation -->
      <nav id="mobile-menu" class="hidden md:hidden">
        <ul class="bg-gray-800 py-2 px-4 space-y-1 border-t border-gray-700">
          <li>
            <a
              href="#"
              class="block py-3 text-blue-400 border-b border-gray-700"
              >Home</a
            >
          </li>
          <li>
            <a
              href="#"
              class="block py-3 text-gray-300 border-b border-gray-700"
              >About</a
            >
          </li>
          <li>
            <a
              href="#"
              class="block py-3 text-gray-300 border-b border-gray-700"
              >Services</a
            >
          </li>
          <li>
            <a
```

```
              href="#"
              class="block py-3 text-gray-300 border-b border-gray-700"
              >Portfolio</a
            >
          </li>
          <li>
            <a href="#" class="block py-3 text-gray-300">Contact</a>
          </li>
        </ul>
      </nav>
    </header>

    <main class="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8 py-12">
      <h1 class="text-3xl font-bold mb-4">Responsive Navbar</h1>
      <p>Resize the window to see the navbar change.</p>
    </main>

    <script>
      // Toggle mobile menu
      const mobileMenuButton = document.getElementById('mobile-menu-button');
      const mobileMenu = document.getElementById('mobile-menu');
      const menuIcon = document.getElementById('menu-icon');
      const closeIcon = document.getElementById('close-icon');

      mobileMenuButton.addEventListener('click', () => {
        mobileMenu.classList.toggle('hidden');
        menuIcon.classList.toggle('hidden');
        closeIcon.classList.toggle('hidden');
      });
    </script>
  </body>
</html>
```

Price Comparison Table

HTML Structure:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Pricing Table</title>
    <link rel="stylesheet" href="styles.css" />
  </head>
  <body>
    <div class="pricing-container">
      <h1 class="pricing-title">Choose Your Plan</h1>
      <p class="pricing-subtitle">Select the perfect plan for your needs</p>

      <div class="pricing-toggle">
```

```
      <span>Monthly</span>
      <label class="toggle">
        <input type="checkbox" id="billing-toggle" />
        <span class="toggle-slider"></span>
      </label>
      <span>Yearly <span class="discount">Save 20%</span></span>
    </div>

    <div class="pricing-plans">
      <!-- Basic Plan -->
      <div class="pricing-plan">
        <div class="plan-header">
          <h2 class="plan-name">Basic</h2>
          <div class="plan-price">
            <div class="monthly-price">$9<span>/month</span></div>
            <div class="yearly-price">$86<span>/year</span></div>
          </div>
          <p class="plan-description">Perfect for individuals</p>
        </div>

        <div class="plan-features">
          <ul>
            <li>5 Projects</li>
            <li>10GB Storage</li>
            <li>Basic Support</li>
            <li class="not-included">Access to all features</li>
            <li class="not-included">Priority Support</li>
          </ul>
        </div>

        <button class="plan-button">Get Started</button>
      </div>

      <!-- Pro Plan -->
      <div class="pricing-plan popular">
        <div class="popular-badge">Popular</div>
        <div class="plan-header">
          <h2 class="plan-name">Pro</h2>
          <div class="plan-price">
            <div class="monthly-price">$19<span>/month</span></div>
            <div class="yearly-price">$182<span>/year</span></div>
          </div>
          <p class="plan-description">For small teams</p>
        </div>

        <div class="plan-features">
          <ul>
            <li>Unlimited Projects</li>
            <li>50GB Storage</li>
            <li>Priority Support</li>
            <li>Access to all features</li>
            <li class="not-included">Dedicated Account Manager</li>
          </ul>
        </div>
```

```
          <button class="plan-button">Get Started</button>
        </div>

        <!-- Enterprise Plan -->
        <div class="pricing-plan">
          <div class="plan-header">
            <h2 class="plan-name">Enterprise</h2>
            <div class="plan-price">
              <div class="monthly-price">$49<span>/month</span></div>
              <div class="yearly-price">$470<span>/year</span></div>
            </div>
            <p class="plan-description">For larger organizations</p>
          </div>

          <div class="plan-features">
            <ul>
              <li>Unlimited Projects</li>
              <li>500GB Storage</li>
              <li>Priority Support</li>
              <li>Access to all features</li>
              <li>Dedicated Account Manager</li>
            </ul>
          </div>

          <button class="plan-button">Get Started</button>
        </div>
      </div>
    </div>

    <script>
      // Toggle between monthly and yearly pricing
      const billingToggle = document.getElementById('billing-toggle');
      const pricingPlans = document.querySelector('.pricing-plans');

      billingToggle.addEventListener('change', function () {
        if (this.checked) {
          pricingPlans.classList.add('yearly');
        } else {
          pricingPlans.classList.remove('yearly');
        }
      });
    </script>
  </body>
</html>
```

CSS Implementation:

```
/* Base Styles */
* {
  margin: 0;
  padding: 0;
```

```
  box-sizing: border-box;
}

body {
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
  line-height: 1.6;
  color: #333;
  background-color: #f8fafc;
  padding: 2rem 1rem;
}

/* Pricing Container */
.pricing-container {
  max-width: 1200px;
  margin: 0 auto;
  text-align: center;
}

.pricing-title {
  font-size: 2.5rem;
  margin-bottom: 0.5rem;
  color: #1e293b;
}

.pricing-subtitle {
  font-size: 1.125rem;
  color: #64748b;
  margin-bottom: 2rem;
}

/* Pricing Toggle */
.pricing-toggle {
  display: flex;
  align-items: center;
  justify-content: center;
  margin-bottom: 3rem;
  font-weight: 500;
}

.toggle {
  position: relative;
  display: inline-block;
  width: 60px;
  height: 30px;
  margin: 0 1rem;
}

.toggle input {
  opacity: 0;
  width: 0;
  height: 0;
}

.toggle-slider {
```

```
  position: absolute;
  cursor: pointer;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
  background-color: #e2e8f0;
  transition: 0.4s;
  border-radius: 34px;
}

.toggle-slider:before {
  position: absolute;
  content: '';
  height: 22px;
  width: 22px;
  left: 4px;
  bottom: 4px;
  background-color: white;
  transition: 0.4s;
  border-radius: 50%;
}

input:checked + .toggle-slider {
  background-color: #3b82f6;
}

input:checked + .toggle-slider:before {
  transform: translateX(30px);
}

.discount {
  background-color: #bae6fd;
  color: #0369a1;
  font-size: 0.75rem;
  padding: 0.25rem 0.5rem;
  border-radius: 9999px;
  margin-left: 0.5rem;
}

/* Pricing Plans */
.pricing-plans {
  display: flex;
  flex-wrap: wrap;
  justify-content: center;
  gap: 2rem;
}

.pricing-plan {
  background-color: white;
  border-radius: 0.5rem;
  box-shadow: 0 4px 6px rgba(0, 0, 0, 0.05);
  width: 100%;
  max-width: 350px;
```

```
  overflow: hidden;
  transition: transform 0.3s, box-shadow 0.3s;
  position: relative;
}

.pricing-plan:hover {
  transform: translateY(-5px);
  box-shadow: 0 10px 15px rgba(0, 0, 0, 0.05);
}

.popular {
  border: 2px solid #3b82f6;
  transform: scale(1.05);
}

.popular:hover {
  transform: scale(1.05) translateY(-5px);
}

.popular-badge {
  position: absolute;
  top: 12px;
  right: 12px;
  background-color: #3b82f6;
  color: white;
  font-size: 0.75rem;
  font-weight: 600;
  padding: 0.25rem 0.75rem;
  border-radius: 9999px;
}

.plan-header {
  padding: 2rem;
  background-color: #f8fafc;
  border-bottom: 1px solid #e2e8f0;
}

.plan-name {
  font-size: 1.5rem;
  margin-bottom: 1rem;
  color: #1e293b;
}

.plan-price {
  font-size: 2.5rem;
  font-weight: 700;
  color: #1e293b;
  margin-bottom: 0.5rem;
}

.plan-price span {
  font-size: 1rem;
  font-weight: 400;
  color: #64748b;
```

```
}

.yearly-price {
  display: none;
}

.pricing-plans.yearly .monthly-price {
  display: none;
}

.pricing-plans.yearly .yearly-price {
  display: block;
}

.plan-description {
  color: #64748b;
  font-size: 0.875rem;
}

.plan-features {
  padding: 2rem;
}

.plan-features ul {
  list-style-type: none;
  text-align: left;
}

.plan-features li {
  padding: 0.5rem 0;
  display: flex;
  align-items: center;
}

.plan-features li::before {
  content: '✓';
  color: #10b981;
  font-weight: bold;
  display: inline-block;
  width: 1.5rem;
  margin-right: 0.5rem;
}

.plan-features li.not-included {
  color: #94a3b8;
}

.plan-features li.not-included::before {
  content: 'X';
  color: #ef4444;
}

.plan-button {
  width: calc(100% - 4rem);
```

```
  margin: 0 2rem 2rem;
  padding: 0.75rem 1.5rem;
  background-color: #3b82f6;
  color: white;
  border: none;
  border-radius: 0.25rem;
  font-size: 1rem;
  font-weight: 600;
  cursor: pointer;
  transition: background-color 0.3s;
}

.plan-button:hover {
  background-color: #2563eb;
}

/* Media Queries */
@media (max-width: 768px) {
  .pricing-plans {
    flex-direction: column;
    align-items: center;
  }

  .pricing-plan {
    max-width: 100%;
  }

  .popular {
    transform: none;
    order: -1;
  }

  .popular:hover {
    transform: translateY(-5px);
  }
}
```

Tailwind Implementation:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Pricing Table</title>
    <script src="https://cdn.tailwindcss.com"></script>
  </head>
  <body class="bg-gray-50 p-4 md:p-8">
    <div class="max-w-7xl mx-auto text-center">
      <h1 class="text-4xl font-bold text-gray-800 mb-2">Choose Your Plan</h1>
      <p class="text-lg text-gray-500 mb-8">
        Select the perfect plan for your needs
```

```
      </p>

      <div class="flex items-center justify-center mb-12">
        <span class="font-medium">Monthly</span>
        <label class="relative inline-flex items-center mx-4 cursor-pointer">
          <input type="checkbox" id="billing-toggle" class="sr-only" />
          <div
            class="w-14 h-7 bg-gray-200 rounded-full peer peer-checked:bg-blue-
500"
          >
            <div
              class="absolute w-5 h-5 bg-white rounded-full left-1 top-1
transition-all peer-checked:left-8"
            ></div>
          </div>
        </label>
        <span class="font-medium"
          >Yearly
          <span
            class="bg-blue-100 text-blue-700 text-xs py-1 px-2 rounded-full ml-1"
            >Save 20%</span
          ></span
        >
      </div>

      <div
        id="pricing-plans"
        class="grid md:grid-cols-3 gap-8 max-w-5xl mx-auto"
      >
        <!-- Basic Plan -->
        <div
          class="bg-white rounded-lg shadow-sm hover:shadow-lg transition-all
hover:-translate-y-1 duration-300"
        >
          <div class="bg-gray-50 p-8 border-b">
            <h2 class="text-2xl font-bold text-gray-800 mb-4">Basic</h2>
            <div
              id="basic-monthly"
              class="text-4xl font-bold text-gray-800 mb-2"
            >
              $9<span class="text-base font-normal text-gray-500">/month</span>
            </div>
            <div
              id="basic-yearly"
              class="text-4xl font-bold text-gray-800 mb-2 hidden"
            >
              $86<span class="text-base font-normal text-gray-500">/year</span>
            </div>
            <p class="text-gray-500 text-sm">Perfect for individuals</p>
          </div>

          <div class="p-8">
            <ul class="text-left mb-8 space-y-2">
              <li class="flex items-center text-gray-700">
```

```
                <svg
                  class="w-5 h-5 text-green-500 mr-2"
                  fill="none"
                  stroke="currentColor"
                  viewBox="0 0 24 24"
                >
                  <path
                    stroke-linecap="round"
                    stroke-linejoin="round"
                    stroke-width="2"
                    d="M5 13l4 4L19 7"
                  ></path>
                </svg>
                5 Projects
              </li>
              <li class="flex items-center text-gray-700">
                <svg
                  class="w-5 h-5 text-green-500 mr-2"
                  fill="none"
                  stroke="currentColor"
                  viewBox="0 0 24 24"
                >
                  <path
                    stroke-linecap="round"
                    stroke-linejoin="round"
                    stroke-width="2"
                    d="M5 13l4 4L19 7"
                  ></path>
                </svg>
                10GB Storage
              </li>
              <li class="flex items-center text-gray-700">
                <svg
                  class="w-5 h-5 text-green-500 mr-2"
                  fill="none"
                  stroke="currentColor"
                  viewBox="0 0 24 24"
                >
                  <path
                    stroke-linecap="round"
                    stroke-linejoin="round"
                    stroke-width="2"
                    d="M5 13l4 4L19 7"
                  ></path>
                </svg>
                Basic Support
              </li>
              <li class="flex items-center text-gray-400">
                <svg
                  class="w-5 h-5 text-red-500 mr-2"
                  fill="none"
                  stroke="currentColor"
                  viewBox="0 0 24 24"
                >
```

```
                  <path
                    stroke-linecap="round"
                    stroke-linejoin="round"
                    stroke-width="2"
                    d="M6 18L18 6M6 6l12 12"
                  ></path>
                </svg>
                Access to all features
              </li>
              <li class="flex items-center text-gray-400">
                <svg
                  class="w-5 h-5 text-red-500 mr-2"
                  fill="none"
                  stroke="currentColor"
                  viewBox="0 0 24 24"
                >
                  <path
                    stroke-linecap="round"
                    stroke-linejoin="round"
                    stroke-width="2"
                    d="M6 18L18 6M6 6l12 12"
                  ></path>
                </svg>
                Priority Support
              </li>
            </ul>

            <button
              class="w-full bg-blue-500 text-white py-3 rounded font-semibold
hover:bg-blue-600 transition-colors"
            >
              Get Started
            </button>
          </div>
        </div>

        <!-- Pro Plan -->
        <div
          class="bg-white rounded-lg shadow-sm hover:shadow-lg transition-all
hover:-translate-y-1 duration-300 border-2 border-blue-500 relative md:scale-110
z-10"
        >
          <div class="absolute top-3 right-3">
            <span
              class="bg-blue-500 text-white text-xs font-bold py-1 px-3 rounded-
full"
              >Popular</span
            >
          </div>
          <div class="bg-gray-50 p-8 border-b">
            <h2 class="text-2xl font-bold text-gray-800 mb-4">Pro</h2>
            <div id="pro-monthly" class="text-4xl font-bold text-gray-800 mb-2">
              $19<span class="text-base font-normal text-gray-500">/month</span>
            </div>
```

```
            <div
              id="pro-yearly"
              class="text-4xl font-bold text-gray-800 mb-2 hidden"
            >
              $182<span class="text-base font-normal text-gray-500">/year</span>
            </div>
            <p class="text-gray-500 text-sm">For small teams</p>
          </div>

          <div class="p-8">
            <ul class="text-left mb-8 space-y-2">
              <li class="flex items-center text-gray-700">
                <svg
                  class="w-5 h-5 text-green-500 mr-2"
                  fill="none"
                  stroke="currentColor"
                  viewBox="0 0 24 24"
                >
                  <path
                    stroke-linecap="round"
                    stroke-linejoin="round"
                    stroke-width="2"
                    d="M5 13l4 4L19 7"
                  ></path>
                </svg>
                Unlimited Projects
              </li>
              <li class="flex items-center text-gray-700">
                <svg
                  class="w-5 h-5 text-green-500 mr-2"
                  fill="none"
                  stroke="currentColor"
                  viewBox="0 0 24 24"
                >
                  <path
                    stroke-linecap="round"
                    stroke-linejoin="round"
                    stroke-width="2"
                    d="M5 13l4 4L19 7"
                  ></path>
                </svg>
                50GB Storage
              </li>
              <li class="flex items-center text-gray-700">
                <svg
                  class="w-5 h-5 text-green-500 mr-2"
                  fill="none"
                  stroke="currentColor"
                  viewBox="0 0 24 24"
                >
                  <path
                    stroke-linecap="round"
                    stroke-linejoin="round"
                    stroke-width="2"
```

```
                    d="M5 13l4 4L19 7"
                  ></path>
                </svg>
                Priority Support
              </li>
              <li class="flex items-center text-gray-700">
                <svg
                  class="w-5 h-5 text-green-500 mr-2"
                  fill="none"
                  stroke="currentColor"
                  viewBox="0 0 24 24"
                >
                  <path
                    stroke-linecap="round"
                    stroke-linejoin="round"
                    stroke-width="2"
                    d="M5 13l4 4L19 7"
                  ></path>
                </svg>
                Access to all features
              </li>
              <li class="flex items-center text-gray-400">
                <svg
                  class="w-5 h-5 text-red-500 mr-2"
                  fill="none"
                  stroke="currentColor"
                  viewBox="0 0 24 24"
                >
                  <path
                    stroke-linecap="round"
                    stroke-linejoin="round"
                    stroke-width="2"
                    d="M6 18L18 6M6 6l12 12"
                  ></path>
                </svg>
                Dedicated Account Manager
              </li>
            </ul>

            <button
              class="w-full bg-blue-500 text-white py-3 rounded font-semibold
hover:bg-blue-600 transition-colors"
            >
              Get Started
            </button>
          </div>
        </div>

        <!-- Enterprise Plan -->
        <div
          class="bg-white rounded-lg shadow-sm hover:shadow-lg transition-all
hover:-translate-y-1 duration-300"
        >
          <div class="bg-gray-50 p-8 border-b">
```

```
            <h2 class="text-2xl font-bold text-gray-800 mb-4">Enterprise</h2>
            <div
              id="enterprise-monthly"
              class="text-4xl font-bold text-gray-800 mb-2"
            >
              $49<span class="text-base font-normal text-gray-500">/month</span>
            </div>
            <div
              id="enterprise-yearly"
              class="text-4xl font-bold text-gray-800 mb-2 hidden"
            >
              $470<span class="text-base font-normal text-gray-500">/year</span>
            </div>
            <p class="text-gray-500 text-sm">For larger organizations</p>
          </div>

          <div class="p-8">
            <ul class="text-left mb-8 space-y-2">
              <li class="flex items-center text-gray-700">
                <svg
                  class="w-5 h-5 text-green-500 mr-2"
                  fill="none"
                  stroke="currentColor"
                  viewBox="0 0 24 24"
                >
                  <path
                    stroke-linecap="round"
                    stroke-linejoin="round"
                    stroke-width="2"
                    d="M5 13l4 4L19 7"
                  ></path>
                </svg>
                Unlimited Projects
              </li>
              <li class="flex items-center text-gray-700">
                <svg
                  class="w-5 h-5 text-green-500 mr-2"
                  fill="none"
                  stroke="currentColor"
                  viewBox="0 0 24 24"
                >
                  <path
                    stroke-linecap="round"
                    stroke-linejoin="round"
                    stroke-width="2"
                    d="M5 13l4 4L19 7"
                  ></path>
                </svg>
                500GB Storage
              </li>
              <li class="flex items-center text-gray-700">
                <svg
                  class="w-5 h-5 text-green-500 mr-2"
                  fill="none"
```

```
                  stroke="currentColor"
                  viewBox="0 0 24 24"
                >
                  <path
                    stroke-linecap="round"
                    stroke-linejoin="round"
                    stroke-width="2"
                    d="M5 13l4 4L19 7"
                  ></path>
                </svg>
                Priority Support
              </li>
              <li class="flex items-center text-gray-700">
                <svg
                  class="w-5 h-5 text-green-500 mr-2"
                  fill="none"
                  stroke="currentColor"
                  viewBox="0 0 24 24"
                >
                  <path
                    stroke-linecap="round"
                    stroke-linejoin="round"
                    stroke-width="2"
                    d="M5 13l4 4L19 7"
                  ></path>
                </svg>
                Access to all features
              </li>
              <li class="flex items-center text-gray-700">
                <svg
                  class="w-5 h-5 text-green-500 mr-2"
                  fill="none"
                  stroke="currentColor"
                  viewBox="0 0 24 24"
                >
                  <path
                    stroke-linecap="round"
                    stroke-linejoin="round"
                    stroke-width="2"
                    d="M5 13l4 4L19 7"
                  ></path>
                </svg>
                Dedicated Account Manager
              </li>
            </ul>

            <button
              class="w-full bg-blue-500 text-white py-3 rounded font-semibold
hover:bg-blue-600 transition-colors"
            >
              Get Started
            </button>
          </div>
        </div>
```

```
        </div>
      </div>

      <script>
        // Toggle between monthly and yearly pricing
        const billingToggle = document.getElementById('billing-toggle');

        const monthlyElements = document.querySelectorAll('[id$="-monthly"]');
        const yearlyElements = document.querySelectorAll('[id$="-yearly"]');

        billingToggle.addEventListener('change', function () {
          if (this.checked) {
            // Show yearly prices
            monthlyElements.forEach((el) => el.classList.add('hidden'));
            yearlyElements.forEach((el) => el.classList.remove('hidden'));
          } else {
            // Show monthly prices
            monthlyElements.forEach((el) => el.classList.remove('hidden'));
            yearlyElements.forEach((el) => el.classList.add('hidden'));
          }
        });
      </script>
    </body>
  </html>
```

Intermediate Projects

Responsive Portfolio Website

For a complete responsive portfolio website, please refer to the standalone article about building a portfolio with HTML, CSS, and JavaScript. The implementation should include:

1. Responsive header with navigation
2. Hero section with call-to-action
3. About me section with skills
4. Project gallery with filtering
5. Contact form
6. Footer with social media links

Image Gallery with Filtering

For an image gallery with category filtering, see the dedicated article on building an interactive image gallery with CSS Grid and JavaScript.

Responsive Dashboard Layout

For a responsive admin dashboard layout, refer to the example in the Advanced Layout Techniques section of this guide.

Advanced Projects

E-commerce Product Page

A complete e-commerce product page implementation would include:

1. Product image gallery with thumbnails
2. Product information section
3. Variant selection (size, color, etc.)
4. Quantity selector
5. Add to cart functionality
6. Product tabs (description, specifications, reviews)
7. Related products section

Social Media Interface

A social media interface implementation would include:

1. Header with search and notifications
2. Sidebar with navigation
3. Feed of posts
4. Post interaction controls
5. User profile section
6. Responsive design for mobile and desktop

Interactive Landing Page

An interactive landing page would include:

1. Animated hero section
2. Features section with animations
3. Testimonial carousel
4. Pricing comparison
5. FAQ accordion
6. Contact form with validation
7. Smooth scrolling navigation

Reference Materials

CSS Cheat Sheet

Selectors

Selector	Example	Description
Element	`p { }`	Selects all paragraph elements
Class	`.example { }`	Selects elements with class="example"
ID	`#example { }`	Selects element with id="example"
Universal	`* { }`	Selects all elements

Selector	Example	Description
Attribute	`[type="text"] { }`	Selects elements with type="text"
Descendant	`div p { }`	Selects all p inside div
Child	`div > p { }`	Selects p that are direct children of div
Adjacent Sibling	`h1 + p { }`	Selects p that directly follows h1
General Sibling	`h1 ~ p { }`	Selects p that follows h1
Pseudo-class	`a:hover { }`	Selects a elements on hover
Pseudo-element	`p::first-line { }`	Selects first line of every p element

Box Model

Property	Description
`width`	Element width
`height`	Element height
`padding`	Space between content and border
`border`	Border around element
`margin`	Space outside border
`box-sizing`	How width and height are calculated
`min-width`	Minimum element width
`max-width`	Maximum element width
`min-height`	Minimum element height
`max-height`	Maximum element height
`overflow`	How to handle content that exceeds dimensions

Display and Positioning

Property	Values
`display`	`block`, `inline`, `inline-block`, `flex`, `grid`, `none`
`position`	`static`, `relative`, `absolute`, `fixed`, `sticky`
`top`, `right`, `bottom`, `left`	Distance from respective edge when positioned
`z-index`	Element stacking order
`float`	`left`, `right`, `none`
`clear`	`left`, `right`, `both`, `none`

Flexbox

Property	Description
`display: flex`	Creates a flex container
`flex-direction`	`row`, `row-reverse`, `column`, `column-reverse`
`flex-wrap`	`nowrap`, `wrap`, `wrap-reverse`
`flex-flow`	Shorthand for flex-direction and flex-wrap
`justify-content`	Alignment along the main axis
`align-items`	Alignment along the cross axis
`align-content`	Alignment of multiple lines along the cross axis
`gap`, `row-gap`, `column-gap`	Space between flex items
`flex`	Shorthand for flex-grow, flex-shrink, flex-basis
`flex-grow`	How much item grows relative to others
`flex-shrink`	How much item shrinks relative to others
`flex-basis`	Initial size of flex item
`align-self`	Override align-items for specific flex item
`order`	Controls order of items in flex container

Grid

Property	Description
`display: grid`	Creates a grid container
`grid-template-columns`	Defines columns and their sizes
`grid-template-rows`	Defines rows and their sizes
`grid-template-areas`	Defines named grid areas
`grid-template`	Shorthand for grid-template-*
`grid-column-gap`	Space between columns
`grid-row-gap`	Space between rows
`grid-gap`	Shorthand for grid-row-gap and grid-column-gap
`justify-items`	Alignment along the row axis
`align-items`	Alignment along the column axis
`place-items`	Shorthand for align-items and justify-items
`justify-content`	Alignment of grid within container (row axis)

Property	Description
`align-content`	Alignment of grid within container (column axis)
`place-content`	Shorthand for align-content and justify-content
`grid-auto-columns`	Size of implicitly created columns
`grid-auto-rows`	Size of implicitly created rows
`grid-auto-flow`	How items are placed in grid
`grid`	Shorthand for all grid properties
`grid-column-start`	Grid item start position (column)
`grid-column-end`	Grid item end position (column)
`grid-row-start`	Grid item start position (row)
`grid-row-end`	Grid item end position (row)
`grid-column`	Shorthand for grid-column-start and grid-column-end
`grid-row`	Shorthand for grid-row-start and grid-row-end
`grid-area`	Shorthand for grid-row and grid-column

Typography

Property	Description
`font-family`	Font family name
`font-size`	Font size
`font-weight`	Font weight (boldness)
`font-style`	Font style (italic, normal)
`font-variant`	Font variant (small-caps)
`font`	Shorthand for font properties
`line-height`	Line height (leading)
`letter-spacing`	Space between characters
`word-spacing`	Space between words
`text-align`	Text alignment
`text-decoration`	Text decoration (underline, line-through)
`text-transform`	Text transformation (uppercase, lowercase)
`text-indent`	First line indentation
`text-shadow`	Text shadow

Property	Description
`vertical-align`	Vertical alignment
`white-space`	How white space is handled
`word-break`	How to break words
`word-wrap`	Allow long words to be broken
`color`	Text color

Color and Background

Property	Description
`color`	Text color
`background-color`	Background color
`background-image`	Background image
`background-repeat`	How background image repeats
`background-position`	Background image position
`background-size`	Background image size
`background-attachment`	Background image scrolling
`background-origin`	Background positioning area
`background-clip`	Background painting area
`background`	Shorthand for background properties
`opacity`	Element opacity

Borders and Shadows

Property	Description
`border-width`	Border width
`border-style`	Border style
`border-color`	Border color
`border`	Shorthand for border properties
`border-radius`	Rounded corners
`box-shadow`	Box shadow
`outline`	Outline around element (outside border)

Transitions and Animations

Property	Description
`transition-property`	CSS properties to transition
`transition-duration`	Duration of transition
`transition-timing-function`	Timing curve of transition
`transition-delay`	Delay before transition
`transition`	Shorthand for all transition properties
`animation-name`	Name of @keyframes animation
`animation-duration`	Duration of animation
`animation-timing-function`	Timing curve of animation
`animation-delay`	Delay before animation
`animation-iteration-count`	Number of times to play animation
`animation-direction`	Direction of animation
`animation-fill-mode`	Styles before/after animation
`animation-play-state`	Running or paused
`animation`	Shorthand for all animation properties
`@keyframes`	Defines animation keyframes

Media Queries

```
/* Basic media query */
@media screen and (max-width: 768px) {
  /* Rules for screens up to 768px wide */
}

/* Media query with multiple conditions */
@media screen and (min-width: 768px) and (max-width: 1024px) {
  /* Rules for screens between 768px and 1024px */
}

/* Media query for print */
@media print {
  /* Rules for print styling */
}

/* Media query for orientation */
@media (orientation: landscape) {
  /* Rules for landscape orientation */
}

/* Media query for dark mode */
@media (prefers-color-scheme: dark) {
```

```
  /* Rules for dark mode */
}
```

CSS Variables (Custom Properties)

```
/* Defining variables */
:root {
  --primary-color: #3490dc;
  --secondary-color: #ffed4a;
  --spacing-unit: 8px;
}

/* Using variables */
.element {
  color: var(--primary-color);
  margin: var(--spacing-unit);
  padding: calc(var(--spacing-unit) * 2);
}

/* Fallback value */
.element {
  color: var(--accent-color, #7e8ea1);
}

/* Scoped variables */
.dark-theme {
  --primary-color: #61dafb;
}
```

Tailwind Cheat Sheet

Core Concepts

Category	Prefix + Examples	Description
Responsive Prefixes	`sm:`, `md:`, `lg:`, `xl:`, `2xl:`	Add before any utility to apply at specific breakpoint and above
State Variants	`hover:`, `focus:`, `active:`, `disabled:`, `group-hover:`	Add before any utility to apply in specific state
Dark Mode	`dark:`	Add before any utility to apply in dark mode

Layout Utilities

Category	Utilities	Description
Container	`container`	Set max-width to match min-width of current breakpoint

Category	Utilities	Description
Display	`block`, `inline-block`, `inline`, `flex`, `inline-flex`, `grid`, `hidden`	Set display property
Floats	`float-left`, `float-right`, `float-none`	Set float property
Clear	`clear-left`, `clear-right`, `clear-both`, `clear-none`	Set clear property
Object Fit	`object-contain`, `object-cover`, `object-fill`, `object-none`, `object-scale-down`	Set object-fit property
Object Position	`object-bottom`, `object-center`, `object-left`, `object-right`, `object-top`	Set object-position property
Overflow	`overflow-auto`, `overflow-hidden`, `overflow-visible`, `overflow-scroll`	Set overflow property
Position	`static`, `fixed`, `absolute`, `relative`, `sticky`	Set position property
Top/Right/Bottom/Left	`top-0`, `right-0`, `bottom-0`, `left-0`, `inset-0`	Set position coordinates
Visibility	`visible`, `invisible`	Set visibility property
Z-Index	`z-0`, `z-10`, `z-20`, `z-30`, `z-40`, `z-50`, `z-auto`	Set z-index property

Flexbox Utilities

Category	Utilities	Description
Flex Direction	`flex-row`, `flex-row-reverse`, `flex-col`, `flex-col-reverse`	Set flex-direction property
Flex Wrap	`flex-wrap`, `flex-nowrap`, `flex-wrap-reverse`	Set flex-wrap property
Flex	`flex-1`, `flex-auto`, `flex-initial`, `flex-none`	Set flex property
Flex Grow	`flex-grow-0`, `flex-grow`	Set flex-grow property
Flex Shrink	`flex-shrink-0`, `flex-shrink`	Set flex-shrink property
Order	`order-1` through `order-12`, `order-first`, `order-last`	Set order property
Justify Content	`justify-start`, `justify-end`, `justify-center`, `justify-between`, `justify-around`, `justify-evenly`	Set justify-content property
Justify Items	`justify-items-start`, `justify-items-end`, `justify-items-center`, `justify-items-stretch`	Set justify-items property

Category	Utilities	Description
Justify Self	`justify-self-auto`, `justify-self-start`, `justify-self-end`, `justify-self-center`, `justify-self-stretch`	Set justify-self property
Align Content	`content-center`, `content-start`, `content-end`, `content-between`, `content-around`, `content-evenly`	Set align-content property
Align Items	`items-start`, `items-end`, `items-center`, `items-baseline`, `items-stretch`	Set align-items property
Align Self	`self-auto`, `self-start`, `self-end`, `self-center`, `self-stretch`, `self-baseline`	Set align-self property
Gap	`gap-0` through `gap-16`, `gap-px`, `gap-x-{size}`, `gap-y-{size}`	Set gap between flex/grid items

Grid Utilities

Category	Utilities	Description
Grid Template Columns	`grid-cols-1` through `grid-cols-12`, `grid-cols-none`	Set grid-template-columns property
Grid Column Span	`col-auto`, `col-span-1` through `col-span-12`, `col-span-full`	Set grid column span
Grid Column Start/End	`col-start-1` through `col-start-13`, `col-end-1` through `col-end-13`	Set grid column start/end
Grid Template Rows	`grid-rows-1` through `grid-rows-6`, `grid-rows-none`	Set grid-template-rows property
Grid Row Span	`row-auto`, `row-span-1` through `row-span-6`, `row-span-full`	Set grid row span
Grid Row Start/End	`row-start-1` through `row-start-7`, `row-end-1` through `row-end-7`	Set grid row start/end
Grid Auto Flow	`grid-flow-row`, `grid-flow-col`, `grid-flow-row-dense`, `grid-flow-col-dense`	Set grid-auto-flow property
Grid Auto Columns	`auto-cols-auto`, `auto-cols-min`, `auto-cols-max`, `auto-cols-fr`	Set grid-auto-columns property
Grid Auto Rows	`auto-rows-auto`, `auto-rows-min`, `auto-rows-max`, `auto-rows-fr`	Set grid-auto-rows property

Spacing Utilities

Category	Utilities	Description
Padding	`p-0` through `p-64`, `p-px`, `p-{size}`	Set padding on all sides

Category	Utilities	Description
Padding X/Y	`px-{size}`, `py-{size}`	Set horizontal or vertical padding
Padding Sides	`pt-{size}`, `pr-{size}`, `pb-{size}`, `pl-{size}`	Set padding on specific side
Margin	`m-0` through `m-64`, `m-px`, `m-auto`, `m-{size}`	Set margin on all sides
Margin X/Y	`mx-{size}`, `my-{size}`, `mx-auto`	Set horizontal or vertical margin
Margin Sides	`mt-{size}`, `mr-{size}`, `mb-{size}`, `ml-{size}`	Set margin on specific side
Space Between	`space-x-{size}`, `space-y-{size}`, `space-x-reverse`, `space-y-reverse`	Set spacing between child elements

Sizing Utilities

Category	Utilities	Description
Width	`w-0` through `w-64`, `w-auto`, `w-px`, `w-full`, `w-screen`	Set width
Width Fractions	`w-1/2`, `w-1/3`, `w-2/3`, `w-1/4`, `w-3/4`, etc.	Set fractional width
Min-Width	`min-w-0`, `min-w-full`, `min-w-min`, `min-w-max`	Set minimum width
Max-Width	`max-w-xs`, `max-w-sm`, `max-w-md`, `max-w-lg`, `max-w-xl`, `max-w-2xl`, etc.	Set maximum width
Height	`h-0` through `h-64`, `h-auto`, `h-px`, `h-full`, `h-screen`	Set height
Min-Height	`min-h-0`, `min-h-full`, `min-h-screen`	Set minimum height
Max-Height	`max-h-full`, `max-h-screen`	Set maximum height

Typography Utilities

Category	Utilities	Description
Font Family	`font-sans`, `font-serif`, `font-mono`	Set font family
Font Size	`text-xs`, `text-sm`, `text-base`, `text-lg`, `text-xl`, `text-2xl` through `text-9xl`	Set font size

Category	Utilities	Description
Font Weight	`font-thin`, `font-extralight`, `font-light`, `font-normal`, `font-medium`, `font-semibold`, `font-bold`, `font-extrabold`, `font-black`	Set font weight
Font Style	`italic`, `not-italic`	Set font style
Letter Spacing	`tracking-tighter`, `tracking-tight`, `tracking-normal`, `tracking-wide`, `tracking-wider`, `tracking-widest`	Set letter spacing
Line Height	`leading-none`, `leading-tight`, `leading-snug`, `leading-normal`, `leading-relaxed`, `leading-loose`	Set line height
Text Alignment	`text-left`, `text-center`, `text-right`, `text-justify`	Set text alignment
Text Color	`text-{color}-{shade}`, e.g., `text-blue-500`, `text-gray-700`	Set text color
Text Decoration	`underline`, `line-through`, `no-underline`	Set text decoration
Text Transform	`uppercase`, `lowercase`, `capitalize`, `normal-case`	Set text transformation
Text Overflow	`truncate`, `overflow-ellipsis`, `overflow-clip`	Set text overflow handling
Vertical Alignment	`align-baseline`, `align-top`, `align-middle`, `align-bottom`, `align-text-top`, `align-text-bottom`	Set vertical alignment

Background Utilities

Category	Utilities	Description
Background Attachment	`bg-fixed`, `bg-local`, `bg-scroll`	Set background attachment
Background Clip	`bg-clip-border`, `bg-clip-padding`, `bg-clip-content`, `bg-clip-text`	Set background clip
Background Color	`bg-{color}-{shade}`, e.g., `bg-blue-500`, `bg-gray-200`	Set background color
Background Origin	`bg-origin-border`, `bg-origin-padding`, `bg-origin-content`	Set background origin
Background Position	`bg-bottom`, `bg-center`, `bg-left`, `bg-right`, `bg-top`	Set background position
Background Repeat	`bg-repeat`, `bg-no-repeat`, `bg-repeat-x`, `bg-repeat-y`	Set background repeat
Background Size	`bg-auto`, `bg-cover`, `bg-contain`	Set background size

Category	Utilities	Description
Background Image	`bg-none`, `bg-gradient-to-{direction}`	Set background image
Gradient Color Stops	`from-{color}-{shade}`, `via-{color}-{shade}`, `to-{color}-{shade}`	Set gradient colors

Border Utilities

Category	Utilities	Description
Border Radius	`rounded-none`, `rounded-sm`, `rounded`, `rounded-md`, `rounded-lg`, `rounded-xl`, `rounded-2xl`, `rounded-3xl`, `rounded-full`	Set border radius
Border Width	`border-0`, `border`, `border-2`, `border-4`, `border-8`	Set border width
Border Color	`border-{color}-{shade}`, e.g., `border-gray-200`	Set border color
Border Style	`border-solid`, `border-dashed`, `border-dotted`, `border-double`, `border-none`	Set border style
Divide Width	`divide-x-{size}`, `divide-y-{size}`	Set border between children
Divide Color	`divide-{color}-{shade}`	Set color of dividing borders
Divide Style	`divide-solid`, `divide-dashed`, `divide-dotted`, `divide-double`, `divide-none`	Set style of dividing borders
Ring	`ring-{size}`, `ring-inset`, `ring-{color}-{shade}`	Add focus rings

Effect Utilities

Category	Utilities	Description
Box Shadow	`shadow-sm`, `shadow`, `shadow-md`, `shadow-lg`, `shadow-xl`, `shadow-2xl`, `shadow-inner`, `shadow-none`	Set box shadow
Opacity	`opacity-0`, `opacity-5`, `opacity-10`, `opacity-20`, `opacity-25`, `opacity-30`, `opacity-40`, `opacity-50`, `opacity-60`, `opacity-70`, `opacity-75`, `opacity-80`, `opacity-90`, `opacity-95`, `opacity-100`	Set opacity
Mix Blend Mode	`mix-blend-normal`, `mix-blend-multiply`, `mix-blend-screen`, etc.	Set mix blend mode

Category	Utilities	Description
Background Blend Mode	`bg-blend-normal`, `bg-blend-multiply`, `bg-blend-screen`, etc.	Set background blend mode

Filter Utilities

Category	Utilities	Description
Blur	`blur-none`, `blur-sm`, `blur`, `blur-md`, `blur-lg`, `blur-xl`, `blur-2xl`, `blur-3xl`	Apply blur filter
Brightness	`brightness-0`, `brightness-50`, `brightness-75`, `brightness-90`, `brightness-95`, `brightness-100`, `brightness-105`, `brightness-110`, `brightness-125`, `brightness-150`, `brightness-200`	Apply brightness filter
Contrast	`contrast-0`, `contrast-50`, `contrast-75`, `contrast-100`, `contrast-125`, `contrast-150`, `contrast-200`	Apply contrast filter
Drop Shadow	`drop-shadow-sm`, `drop-shadow`, `drop-shadow-md`, `drop-shadow-lg`, `drop-shadow-xl`, `drop-shadow-2xl`, `drop-shadow-none`	Apply drop shadow filter
Grayscale	`grayscale-0`, `grayscale`	Apply grayscale filter
Hue Rotate	`hue-rotate-0`, `hue-rotate-15`, `hue-rotate-30`, `hue-rotate-60`, `hue-rotate-90`, `hue-rotate-180`, `-hue-rotate-30`, `-hue-rotate-60`, `-hue-rotate-90`, `-hue-rotate-180`	Apply hue rotation filter
Invert	`invert-0`, `invert`	Apply invert filter
Saturate	`saturate-0`, `saturate-50`, `saturate-100`, `saturate-150`, `saturate-200`	Apply saturation filter
Sepia	`sepia-0`, `sepia`	Apply sepia filter
Backdrop Filter	`backdrop-filter-none`, `backdrop-blur-{size}`, etc.	Apply filter to background

Transform Utilities

Category	Utilities	Description

Category	Utilities	Description
Scale	`scale-0`, `scale-50`, `scale-75`, `scale-90`, `scale-95`, `scale-100`, `scale-105`, `scale-110`, `scale-125`, `scale-150`	Set scale transform
Rotate	`rotate-0`, `rotate-1`, `rotate-2`, `rotate-3`, `rotate-6`, `rotate-12`, `rotate-45`, `rotate-90`, `rotate-180`, `-rotate-1`, etc.	Set rotation transform
Translate	`translate-x-0`, `translate-x-1`, etc., `translate-y-0`, `translate-y-1`, etc.	Set translation transform
Skew	`skew-x-0`, `skew-x-1`, etc., `skew-y-0`, `skew-y-1`, etc.	Set skew transform
Transform Origin	`origin-center`, `origin-top`, `origin-top-right`, `origin-right`, etc.	Set transform origin

Transition and Animation Utilities

Category	Utilities	Description
Transition Property	`transition-none`, `transition-all`, `transition`, `transition-colors`, `transition-opacity`, `transition-shadow`, `transition-transform`	Set transition property
Transition Duration	`duration-75`, `duration-100`, `duration-150`, `duration-200`, `duration-300`, `duration-500`, `duration-700`, `duration-1000`	Set transition duration
Transition Timing	`ease-linear`, `ease-in`, `ease-out`, `ease-in-out`	Set transition timing function
Transition Delay	`delay-75`, `delay-100`, `delay-150`, `delay-200`, `delay-300`, `delay-500`, `delay-700`, `delay-1000`	Set transition delay
Animation	`animate-none`, `animate-spin`, `animate-ping`, `animate-pulse`, `animate-bounce`	Apply animation

Interactivity Utilities

Category	Utilities	Description
Accent Color	`accent-{color}-{shade}`	Set accent color for form controls
Appearance	`appearance-none`	Remove native styling
Cursor	`cursor-auto`, `cursor-default`, `cursor-pointer`, `cursor-wait`, `cursor-text`, `cursor-move`, `cursor-not-allowed`	Set cursor style
Pointer Events	`pointer-events-none`, `pointer-events-auto`	Control whether element responds to pointer events

Category	Utilities	Description
Resize	`resize-none`, `resize`, `resize-y`, `resize-x`	Set element resizability
Scroll Behavior	`scroll-auto`, `scroll-smooth`	Set scrolling behavior
User Select	`select-none`, `select-text`, `select-all`, `select-auto`	Control text selection

CSS vs Tailwind Comparison

Feature	CSS Approach	Tailwind Approach
Centering a div	`css .center-div {  display: flex;  justify-content: center;  align-items: center; }`	`html <div class="flex justify-center items-center">...</div>`
Margin and Padding	`css .content {  margin-top: 1rem;  margin-bottom: 2rem;  padding: 1.5rem; }`	`html <div class="mt-4 mb-8 p-6">...</div>`
Typography	`css .title {  font-size: 1.875rem;  font-weight: 700;  color: #1a202c;  line-height: 1.25; }`	`html <h1 class="text-3xl font-bold text-gray-900 leading-tight">...</h1>`
Responsive Design	`css @media (min-width: 768px) {  .card {  width: 50%;  } }  @media (min-width: 1024px) {  .card {  width: 33.333333%;  } }`	`html <div class="w-full md:w-1/2 lg:w-1/3">...</div>`
Hover States	`css .button:hover {  background-color: #2c5282;  transform: translateY(-1px); }`	`html <button class="hover:bg-blue-800 hover:-translate-y-1">...</button>`
Flexbox Layout	`css .container {  display: flex;  flex-direction: column;  gap: 1rem; }  @media (min-width: 768px) {  .container {  flex-direction: row;  } }`	`html <div class="flex flex-col md:flex-row gap-4">...</div>`

Feature	CSS Approach	Tailwind Approach

Grid Layout	css
	.grid {
	 display: grid;
	 grid-template-columns: repeat(1, 1fr);
	 gap: 1rem;
	}
	
	@media (min-width: 640px) {
	 .grid {
	 grid-template-columns: repeat(2, 1fr);
	 }
	}
	
	@media (min-width: 1024px) {
	 .grid {
	 grid-template-columns: repeat(3, 1fr);
	 }
	}
		html
		 ...

Card Component	css
	.card {
	 background-color: white;
	 border-radius: 0.5rem;
	 box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
	 padding: 1.5rem;
	 margin-bottom: 1rem;
	}
		html
		 ...

Button Styles	css
	.btn {
	 padding: 0.5rem 1rem;
	 background-color: #3182ce;
	 color: white;
	 border-radius: 0.25rem;
	 font-weight: 600;
	}
	
	.btn:hover {
	 background-color: #2c5282;
	}
	
	.btn:focus {
	 outline: none;
	 box-shadow: 0 0 0 3px rgba(66, 153, 225, 0.5);
	}
		html
		<button class="px-4 py-2 bg-blue-500 text-white rounded font-semibold hover:bg-blue-700 focus:outline-none focus:ring-2 focus:ring-blue-500 focus:ring-opacity-50">...</button>

Custom Variables	css
	:root {
	 --primary-color: #3182ce;
	 --spacing-unit: 1rem;
	}
	
	.element {
	 color: var(--primary-color);
	 margin: calc(var(--spacing-unit) * 2);
	}
		js
		// tailwind.config.js
		module.exports = {
		 theme: {
		 extend: {
		 colors: {
		 primary: '#3182ce',
		 },
		 spacing: {
		 '2u': '2rem',
		 },
		 },
		 },
		}
		
		html
		 ...

Dark Mode	css
	@media (prefers-color-scheme: dark) {
	 .card {
	 background-color: #1a202c;
	 color: #f7fafc;
	 }
	}
		html
		 ...

Recommended Tools and Resources

Development Tools

1. **Code Editors and IDEs**:
   - **Visual Studio Code**: Free editor with excellent CSS and Tailwind support
   - **WebStorm**: Full-featured IDE for web development
   - **Sublime Text**: Lightweight, fast editor with extensive plugin system
   - **Atom**: Customizable editor with good web development support

2. **Browser Developer Tools**:
   - **Chrome DevTools**: Comprehensive tools for debugging and testing
   - **Firefox Developer Tools**: Excellent CSS inspection and grid visualization
   - **Safari Web Inspector**: Performance and debugging tools for Safari
   - **Edge DevTools**: Similar to Chrome with some unique features

3. **VS Code Extensions**:
   - **Tailwind CSS IntelliSense**: Autocomplete, syntax highlighting for Tailwind
   - **CSS Peek**: Jump to CSS definitions from HTML
   - **Live Server**: Local development server with live reload
   - **Prettier**: Code formatter with CSS/HTML support
   - **ESLint**: Linting for JavaScript and JSX
   - **HTML CSS Support**: CSS class name completion
   - **Color Highlight**: Highlights color codes in your files

4. **Build Tools**:
   - **Vite**: Next-generation frontend build tool
   - **Webpack**: Powerful and configurable bundler
   - **Parcel**: Zero-configuration bundler
   - **PostCSS**: Tool for transforming CSS with JavaScript plugins
   - **Gulp**: Automation toolkit for development workflows

Online Tools and Services

1. **CSS Tools**:
   - **Clippy**: CSS clip-path generator
   - **Cubic-Bezier**: Create custom timing functions
   - **CSS Grid Generator**: Visual CSS grid builder
   - **Flexbox Froggy**: Learn Flexbox through a game
   - **Grid Garden**: Learn CSS Grid through a game
   - **Can I Use**: Browser compatibility tables

2. **Tailwind Tools**:
   - **Tailwind Play**: Official Tailwind CSS playground
   - **Tailwind Components**: Community components
   - **Tailblocks**: Ready-to-use Tailwind blocks

- **Tailwind Toolbox**: Templates and components
- **Headless UI**: Unstyled, accessible UI components
- **Tailwind UI**: Premium component library (paid)

3. **Design and Prototyping**:

- **Figma**: Collaborative design tool
- **Adobe XD**: UI/UX design tool
- **Sketch**: Digital design tool for Mac
- **InVision**: Digital product design platform

4. **Color Tools**:

- **Coolors**: Color scheme generator
- **Adobe Color**: Create color palettes
- **Tailwind Color Shades**: Generate Tailwind color shades
- **Color Designer**: Create accessible color palettes

Learning Resources

1. **Documentation**:

- **MDN Web Docs**: Comprehensive CSS documentation
- **Tailwind CSS Docs**: Official Tailwind documentation
- **CSS-Tricks**: Articles and tutorials on CSS
- **web.dev**: Web development guidance from Google

2. **Courses and Tutorials**:

- **freeCodeCamp**: Free web development courses
- **CSS Grid by Wes Bos**: Free CSS Grid course
- **Flexbox Zombies**: Learn Flexbox through a game
- **Scrimba's Learn CSS**: Interactive CSS lessons
- **Tailwind From Scratch**: YouTube tutorial series

3. **Books**:

- **"CSS: The Definitive Guide"** by Eric Meyer & Estelle Weyl
- **"CSS Secrets"** by Lea Verou
- **"Refactoring UI"** by Adam Wathan & Steve Schoger
- **"Every Layout"** by Heydon Pickering & Andy Bell
- **"Tailwind CSS: From Zero to Production"** various online guides

4. **Blogs and Newsletters**:

- **CSS-Tricks**: Articles about CSS and web design
- **Smashing Magazine**: Web design and development articles
- **A List Apart**: Web design best practices
- **Tailwind Weekly**: Weekly Tailwind CSS newsletter
- **CSS Weekly**: Weekly CSS newsletter

5. **YouTube Channels**:
    - **Kevin Powell**: CSS tutorials and tips
    - **Adam Wathan**: Tailwind CSS creator's channel
    - **Traversy Media**: Web development tutorials
    - **Web Dev Simplified**: Clear explanations of web concepts
    - **DesignCourse**: UI/UX and frontend tutorials

Community and Support

1. **Forums and Communities**:
    - **Stack Overflow**: Q&A for programming
    - **Reddit r/css**: CSS community
    - **Reddit r/tailwindcss**: Tailwind community
    - **Dev.to**: Community of software developers
    - **Frontend Mentor**: Frontend coding challenges
2. **Discord and Slack Communities**:
    - **Tailwind CSS Discord**: Official Tailwind community
    - **Frontend Developers**: Slack for frontend devs
    - **CSS-Tricks Spectrum**: CSS-Tricks community
    - **Web Dev Discord**: Web development community

Conclusion

This comprehensive guide has covered both CSS and Tailwind CSS from fundamentals to advanced techniques. By working through each section, you've gained a thorough understanding of both traditional CSS approaches and the utility-first methodology of Tailwind CSS.

Remember that both CSS and Tailwind have their places in modern web development. Pure CSS offers complete control and works everywhere, while Tailwind provides rapid development and consistent design constraints. Many developers use both in their projects, leveraging the strengths of each.

Continue your learning journey by building projects, experimenting with new techniques, and staying up-to-date with evolving web standards. The included reference materials and resources will help you solve problems and find inspiration as you create beautiful, responsive web experiences.

Happy coding!