



## Core Concepts & Setup

### Installation

**Using npm:**

```
npm install -D tailwindcss postcss autoprefixer
npx tailwindcss init -p
```

**Using yarn:**

```
yarn add -D tailwindcss postcss autoprefixer
npx tailwindcss init -p
```

Configure your template paths in `tailwind.config.js`:

```
/** @type {import('tailwindcss').Config} */
module.exports = {
  content: [
    './src/**/*.{html,js}"
  ],
  theme: {
    extend: {},
  },
  plugins: [],
}
```

Add Tailwind directives to your CSS:

```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

### Utility-First Fundamentals

Tailwind CSS is a utility-first CSS framework. Instead of pre-designed components, it provides low-level utility classes that you compose to build custom designs directly in your HTML.

Example:

```
<button class="bg-blue-500 hover:bg-blue-700 text-white font-bold py-2 px-4 rounded">
  Click me
</button>
```

This creates a blue button that changes color on hover, has white text, bold font, padding, and rounded corners.

## Common Utilities

### Layout

<code>container</code>	Centers content horizontally with maximum width based on breakpoints.
<code>block</code> , <code>inline</code> , <code>inline-block</code>	Sets the display property of an element.
<code>grid</code> , <code>flex</code>	Enables CSS Grid and Flexbox layouts.
<code>static</code> , <code>relative</code> , <code>absolute</code> , <code>fixed</code> , <code>sticky</code>	Sets the position property of an element.

### Spacing

<code>p-*</code> (padding)	Sets padding on all sides of an element. E.g., <code>p-4</code> .
<code>pt-*</code> , <code>pb-*</code> , <code>pl-*</code> , <code>pr-*</code>	Sets padding on the top, bottom, left, and right sides respectively. E.g., <code>pt-2</code> .
<code>m-*</code> (margin)	Sets margin on all sides of an element. E.g., <code>m-8</code> .
<code>mt-*</code> , <code>mb-*</code> , <code>ml-*</code> , <code>mr-*</code>	Sets margin on the top, bottom, left, and right sides respectively. E.g., <code>mb-4</code> .
<code>space-x-*</code> , <code>space-y-*</code>	Adds space between elements in a flex or grid container. E.g., <code>space-x-2</code> .

### Typography

<code>font-*</code>	Sets the font family. E.g., <code>font-sans</code> , <code>font-serif</code> , <code>font-mono</code> .
<code>text-*</code>	Sets the text size. E.g., <code>text-sm</code> , <code>text-lg</code> , <code>text-xl</code> .
<code>font-bold</code> , <code>font-medium</code> , <code>font-light</code>	Sets the font weight.
<code>italic</code> , <code>not-italic</code>	Sets the font style.
<code>underline</code> , <code>line-through</code> , <code>no-underline</code>	Sets text decoration.
<code>text-left</code> , <code>text-center</code> , <code>text-right</code> , <code>text-justify</code>	Sets text alignment.

## Customization & Directives

### Configuration

The `tailwind.config.js` file allows you to customize Tailwind's default configuration, including colors, fonts, breakpoints, and more.

Example: Extending the theme

```
module.exports = {
  theme: {
    extend: {
      colors: {
        'custom-blue': '#123456',
      },
      fontFamily: {
        'brand': ['Roboto', 'sans-serif'],
      },
    },
  },
}
```

This adds a custom color `custom-blue` and a custom font family `brand`.

### Directives

<code>@tailwind</code>	Used to inject Tailwind's base, components, and utilities styles.
<code>@apply</code>	Allows you to use Tailwind's utility classes directly in your own CSS. Useful for extracting reusable component styles.
<code>@layer</code>	Used to organize custom CSS into layers (base, components, utilities). This helps Tailwind properly optimize and order the CSS.
<code>@config</code>	Specifies a custom Tailwind configuration file.

## Responsive Design & State Variants

### Responsive Modifiers

Tailwind uses breakpoint prefixes to apply styles conditionally based on screen size.

`sm:`, `md:`, `lg:`, `xl:`, `2xl:`

Example:

```
<div class="text-center md:text-left">
  This text will be centered on small
  screens and left-aligned on medium
  screens and up.
</div>
```

### State Variants

<code>hover:</code>	Applies styles on hover.
<code>focus:</code>	Applies styles when the element is focused.
<code>active:</code>	Applies styles when the element is active (e.g., being clicked).
<code>disabled:</code>	Applies styles when the element is disabled.
<code>focus-within:</code>	Applies styles to the parent when a child element is focused.
<code>focus-visible:</code>	Applies styles when the element is focused using the keyboard (accessibility).

### Dark Mode

Tailwind supports dark mode using the `dark:` variant. You need to configure dark mode in your `tailwind.config.js` file.

```
// tailwind.config.js
module.exports = {
  darkMode: 'class', // or 'media'
  // ...
}
```

Then, you can use the `dark:` prefix to apply styles in dark mode:

```
<div class="bg-white dark:bg-gray-800
text-gray-800 dark:text-white">
  This text will be dark on light
  backgrounds and light on dark
  backgrounds.
</div>
```