

Sidebar

Programming · VS Code Tailwind CSS Cheat Sheet

The Tailwind CSS IntelliSense extension's settings, the VS Code settings it recommends, and the Tailwind v4 CSS directives. Checked against extension 0.16.0 installed here and the Tailwind CSS v4.3 documentation. Sheet 18 in the VS Code series.

GETTING IT WORKING

- Extension ID

`bradlc.vscode-tailwindcss` (version 0.16.0 here): autocomplete, hover previews and linting for Tailwind classes
- Needs `tailwindcss` installed

In your project (npm, pnpm or yarn); the extension cannot work from the editor alone
- Tailwind v4 and later

A `.css` file that contains `@import "tailwindcss";` (the extension finds it automatically)
- Tailwind v3 and earlier

A `tailwind.config.js` (or `.cjs`, `.mjs`, `.ts`, `.cts`, `.mts`) in the workspace, or a stylesheet that points to one with `@config`
- Tailwind CSS: Show Output

Command Palette: opens the language server log; first place to look if nothing works
- Tailwind CSS: Sort Selection

Command Palette (pre-release): sort selected classes in Tailwind's recommended order
- Tailwind CSS language mode

An alternative to VS Code's CSS mode that understands Tailwind at-rules

RECOMMENDED VS CODE SETTINGS

- `files.associations`

`{ "*.css": "tailwindcss" }`; open CSS in Tailwind CSS mode so `@apply`, `@theme` and other directives are not flagged as unknown
- `editor.quickSuggestions`

`{ "strings": "on" }`; lets class suggestions appear inside quoted attribute values (off for strings by default)
- `editor.colorDecorators`

Must be on (default) for the extension's colour swatches

WHERE IT WORKS (LANGUAGE AND CLASS DETECTION)

- `tailwindCSS.classAttributes`

Attributes treated as class lists (default `["class", "className", "ngClass", "class:list"]`)
- `tailwindCSS.classFunctions`

Function or tagged-template names whose strings are classes, e.g. `["clsx", "cn", "tw"]` (default none)
- `tailwindCSS.includeLanguages`

Map an unsupported language to a supported one, e.g. `{ "plaintext": "html" }` (default `{}`)
- `tailwindCSS.emmetCompletions`

Class completions inside Emmet abbreviations such as `div.bg-red-500` (default `false`)
- `tailwindCSS.files.exclude`

Globs ignored by all features; inherits from `files.exclude` (default skips `.git`, `node_modules`, `.hg`, `.svn`)
- `tailwindCSS.experimental.classRegex`

Your own patterns for finding class strings (experimental, default none)
- `tailwindCSS.experimental.configFile`

Name the CSS entry file (v4) or config file (v3) yourself; a string, or an object mapping files to globs for several projects

FEATURES ON / OFF

- `tailwindCSS.suggestions`

Autocomplete (default `true`)
- `tailwindCSS.hovers`

Hover preview showing the CSS a class generates (default `true`)
- `tailwindCSS.codeActions`

Quick fixes, e.g. for conflicting classes (default `true`)
- `tailwindCSS.codeLens`

Code lens annotations (default `true`)
- `tailwindCSS.colorDecorators`

Colour swatches next to colour class names and functions (default `true`)
- `tailwindCSS.validate`

Linting master switch (default `true`)
- `tailwindCSS.showPixelEquivalents`

Show `px` next to `rem` values (default `true`)
- `tailwindCSS.rootFontSize`

Root font size used for that conversion (default 16)

LINT RULES (EACH: IGNORE, WARNING OR ERROR)

<code>tailwindCSS.lint.cssConflict</code>	Two classes on one element set the same CSS property (default <code>warning</code>)
<code>tailwindCSS.lint.suggestCanonicalClasses</code>	A class could be written in a shorter form (default <code>warning</code>)
<code>tailwindCSS.lint.recommendedVariantOrder</code>	Variants not in the recommended order (default <code>warning</code>)
<code>tailwindCSS.lint.usedBlocklistedClass</code>	A class you excluded with <code>@source not inline(...)</code> is used (default <code>warning</code>)
<code>tailwindCSS.lint.deprecatedAtRule</code>	Old Tailwind at-rules (default <code>warning</code>)
<code>tailwindCSS.lint.invalidApply</code>	<code>@apply</code> used in an unsupported way (default <code>error</code>)
<code>tailwindCSS.lint.invalidConfigPath</code>	Unknown path in the <code>theme()</code> function (default <code>error</code>)
<code>tailwindCSS.lint.invalidTailwindDirective</code>	Unknown value after <code>@tailwind</code> (default <code>error</code>)
<code>tailwindCSS.lint.invalidScreen / lint.invalidVariant</code>	Unknown screen or variant name in older directives (default <code>error</code>)

TAILWIND V4 CSS DIRECTIVES

<code>@import "tailwindcss";</code>	Load Tailwind itself; this line is what the extension looks for
<code>@theme { ... }</code>	Define design tokens such as colours, fonts and break as CSS variables, e.g. <code>--color-brand: #38bdf8;</code>
<code>@source "path";</code>	Tell Tailwind to scan extra files for class names
<code>@utility name { ... }</code>	Add your own utility class that works with var
<code>@variant dark { ... }</code>	Apply a built-in variant to your own CSS
<code>@custom-variant name (selector);</code>	Create a new variant
<code>@apply a b c;</code>	Inline existing utilities into your own CSS rule
<code>@reference "file";</code>	Use the theme and utilities in a component stylesheet without repeating the output CSS
<code>@config / @plugin</code>	Load a v3-style config file or plugin (kept for compatibility)
<code>--alpha()</code> <code>--spacing()</code>	Functions: change a colour's opacity, and spacing value, e.g. <code>--spacing(4)</code>
<code>theme()</code>	Older function for reading theme values (deprecated in v4; use CSS variables)

TAILWIND HABITS WORTH KNOWING (MY KNOWLEDGE)

State variants	<code>hover:</code> , <code>focus:</code> , <code>dark:</code> , <code>md:</code> , <code>group-hover:</code> : put the variant in front of any utility
Arbitrary values	<code>w-[137px]</code> , <code>bg-[#38bdf8]</code> : square brackets for one-off values
Important	Put <code>!</code> at the end (v4) or the start (v3) of a class to force it; check which your version uses
Prettier plugin	<code>prettier-plugin-tailwindcss</code> sorts classes on save (see the Prettier sheet)
Dark mode	In v4, <code>dark:</code> follows the OS by default; use <code>@custom-variant</code> for a class-based switch

A REASONABLE STARTER SET (MY SUGGESTIONS, NOT DEFAULTS)

<code>"files.associations": { "*.css": "tailwindcss" }</code>	Recommended by the extension
<code>"editor.quickSuggestions": { "strings": "on" }</code>	Class completion inside attribute strings
<code>"tailwindCSS.classFunctions": ["clsx", "cn"]</code>	If you use helper functions to build class strings
<code>"tailwindCSS.includeLanguages": { "astro": "html" }</code>	Only suggest completions for <code>do not appear</code> in <code>.astro</code>
<code>"tailwindCSS.lint.cssConflict": "error"</code>	Make conflicting class impossible to miss

NO SUGGESTIONS OR HOVERS? CHECK THESE

1) Is `tailwindcss` installed in the project? 2) For v4, does a `.css` file contain `@import "tailwindcss";`? (Sass, Less and Stylus files do not count.) For v3, is there a `tailwind.config.*`? 3) Are those files hidden by `files.exclude`, `files.watcherExclude` or `tailwindCSS.files.exclude`? 4) Run *Tailwind CSS: Show Output*. 5) With several Tailwind installs or entry stylesheets, set `tailwindCSS.experimental.configFile`. 6) For attribute strings, set `editor.quickSuggestions` as above.

"UNKNOWN AT RULE @APPLY" FROM VS CODE ITSELF

VS Code's built-in CSS checker does not know Tailwind's at-rules and underlines them. Fix it by opening CSS files in Tailwind CSS mode with `files.associations` (see the second card), or by turning off the built-in check with `{ "css.lint.unknownAtRules": "ignore" }`.