

Sidebar

Programming · VS Code Astro Cheat Sheet

The Astro extension's settings and commands, plus the `.astro` component syntax, config options and command line. Checked against extension 2.16.20 installed here and the current Astro documentation. Sheet 19 in the VS Code series.

THE EXTENSION

- Extension ID** `astro-build.astro-vscode` (version 2.16.20 here): language support for `.astro` files
- Features** Highlighting, errors, completions with auto-imports, Emmet in HTML/CSS, props completion for React/Vue/Svelte components, formatting, hover, go to definition, inlay hints, folding
- Formatting** Uses Prettier with `prettier-plugin-astro`; outside the editor, install that plugin yourself
- TypeScript plugin** Also lets `.ts` and `.js` files import and export Astro components
- Snippets** `page_html` (page with full HTML), `page_layout` (page using a layout), `component`
- Astro: Reload Projects** Command Palette: re-read the project after changing config or adding files
- Astro: Find File References** Command Palette: find where a file is imported
- Astro: Select TypeScript Version...** Command Palette: choose the workspace or the bundled TypeScript
- Astro: Open TypeScript config** Command Palette: open the `tsconfig.json` that applies

RELATED VS CODE SETTINGS

- typescript.inlayHints.*** Astro is TypeScript-only, so set the `typescript.` versions, e.g. `{ "typescript.inlayHints.parameterNames.enabled": "all" }`
- html.* / css.*** HTML and CSS behaviour inside `.astro` files follows the `html` and `css` settings
- [astro]** A `settings.json` language block for overrides that apply only to `.astro` files
- editor.defaultFormatter** For Astro files, usually `"astro-build.astro-vscode"` inside `[astro]` (or Prettier with the Astro plugin)
- tailwindCSS.includeLanguages** Only if Tailwind class suggestions do not appear: `{ "astro": "html" }`
- eslint.probe** Already includes `astro`; ESLint needs an Astro plugin to lint the template

EXTENSION SETTINGS (ASTRO.*)

- astro.auto-import-cache.enabled** Faster auto-imports; manually reloaded (default `true`)
- astro.updateImportsOnFileMove.enabled** Update imports on file move (default `false`)
- astro.content-intellisense** Experimental content collection for MDX and Markdown (default `false`)
- astro.language-server.ls-path** Use a specific language server (rarely needed)
- astro.language-server.runtime** Path to the Node program (default none)
- astro.trace.server** `off` (default), `messages`, `verbose`

COMPONENT SYNTAX (.ASTRO FILE)

- **...** **---** The frontmatter fence: JavaScript/TypeScript at the top when the page is built
- {expression}** Insert a value or run an expression in the template: `{items.map(i => <li>{i}</li>)}`
- {cond && <p>...</p>}** Conditional output; ternaries work too
- class={name}** Dynamic attributes use curly braces
- <Fragment> or <> ... </>** Group elements without a wrapper
- <slot />** Where a parent's child content goes
- <slot name="sidebar" />** A named slot; the parent fills it with content
- const { title } = Astro.props;** Read the props passed to the component
- Differences from JSX** Use `class`, not `className`; several HTML comments appear in the output

THE ASTRO OBJECT

<code>Astro.props</code>	Props passed in by the parent
<code>Astro.url</code>	The current page's URL
<code>Astro.params</code>	Route parameters of a dynamic route
<code>Astro.request</code>	The full request object
<code>Astro.redirect(path)</code>	Send the visitor elsewhere (server rendering)
<code>Astro.slots</code>	<code>has()</code> and <code>render()</code> for working with slot content
<code>Astro.self</code>	Refer to the component itself, for recursion

PROJECT LAYOUT & ROUTING (MY KNOWLEDGE)

<code>src/pages/</code>	Every <code>.astro</code> , <code>.md</code> or <code>.html</code> file here becomes a URL
<code>[slug].astro</code>	A dynamic route; in static builds it exports <code>getStaticPaths()</code>
<code>[...slug].astro</code>	A catch-all route that matches any depth of path
<code>src/layouts/</code> and <code>src/components/</code>	Reusable page frames and building blocks
<code>src/content/</code>	Content collections: files with a typed schema
<code>public/</code>	Copied to the output as they are, with no processing
<code>dist/</code>	Where <code>astro build</code> writes the finished site
<code>.astro/types.d.ts</code>	Types generated by <code>astro sync</code> ; list it in <code>tsconfig.json</code> <code>include</code>
<code>tsconfig.json</code>	Usually extends <code>astro/tsconfigs/strict</code> (<code>base</code> and <code>strictest</code> also exist)

COMMAND LINE (NPX ASTRO ...)

<code>dev</code>	Start the dev server with hot reloading (port 4321 by default)
<code>build</code>	Build the site into <code>dist/</code>
<code>preview</code>	Serve the built <code>dist/</code> folder locally
<code>check</code>	Type-check and report diagnostics (exits with code 1 on errors)
<code>sync</code>	Generate TypeScript types for Astro modules
<code>add name</code>	Install and configure an integration, e.g. <code>astro add react</code>
<code>info</code>	Print your environment details (useful in bug reports)
<code>--port --host --open</code>	Dev/preview flags: port number, network address, open the browser
<code>--config --root --site --base</code>	Point at a different config, root, site URL or base path
<code>--force --verbose --silent</code>	Clear the content cache / more logging / no logging

TEMPLATE DIRECTIVES

<code>set:html={string}</code>	Insert a string as raw HTML (only with co
<code>set:text={string}</code>	Insert a string as plain text
<code>class:list={[...]}</code>	Build a class list from strings, objects and a
<code>define:vars={{ x }}</code>	Pass server values into a <code><style></code> as
<code>is:global</code>	On <code><style></code> ; not scoped to the component
<code>is:inline / is:raw</code>	Leave a script or style as written / do no
<code>client:load client:idle client:visible</code>	Hydrate a fram / when idle / w
<code>client:media="(min-width: 768px)"</code>	Hydrate when a med
<code>client:only="react"</code>	Skip server rendering; render only in th
<code>transition:name transition:animate</code>	View transitions for

ASTRO.CONFIG.MJS

<code>export default defineConfig({ ... })</code>	The file shape; <code>def</code> <code>astro/config</code>
<code>site</code>	Your final URL, used for canonical links and sitemaps
<code>base</code>	A sub-path the site is deployed under
<code>output</code>	<code>static</code> (default) or <code>server</code>
<code>trailingSlash</code>	<code>always</code> , <code>never</code> , <code>ignore</code> (default)
<code>build.format</code>	<code>directory</code> (default), <code>file</code> , <code>preserve</code> ; h
<code>build.assets</code>	Folder name for built assets (default <code>_astro</code>)
<code>srcDir / publicDir / outDir</code>	Defaults <code>./src</code> , <code>./publi</code>
<code>integrations</code>	Array of integrations such as React, MDX or site
<code>redirects</code>	Map old paths to new ones: <code>{ "/old": "/new"</code>
<code>server.port / server.host / server.open</code>	Dev server op <code>false</code>)
<code>devToolbar.enabled</code>	The in-browser dev toolbar (default <code>tr</code>
<code>compressHTML / scopedStyleStrategy</code>	Whitespace handlin scoped (default <code>at</code>
<code>vite</code>	Pass options through to Vite

A REASONABLE STARTER SET (MY SUGGESTIONS, NOT I

<code>"typescript.inlayHints.parameterNames.enabled": "lit</code>	
<code>"[astro]": { "editor.defaultFormatter": "astro-build</code>	
<code>"astro.updateImportsOnFileMove.enabled": true</code>	Import move like it)
<code>"emmet.includeLanguages": { "astro": "html" }</code>	Only i expan

NO TYPES, NO COMPLETIONS OR RED ERRORS THAT MAKE NO SENSE IN .ASTRO FILES

1) Run `npx astro sync` to regenerate the types, and make sure `.astro/types.d.ts` is in the `include` list of `tsconfig.json`. 2) Run *Astro: Reload Projects*. 3) Run *Astro: Select TypeScript Version...* and pick the workspace version if your project has its own. 4) Run *TypeScript: Restart TS Server*. 5) For a brand-new file that auto-import cannot find, reload the window or turn off `astro.auto-import-cache.enabled`.

FORMATTING .ASTRO FILES WITH PRETTIER

Inside VS Code the extension formats `.astro` files for you. For the command line or CI, install `prettier-plugin-astro` and list it in the Prettier config:

```
{ "plugins": ["prettier-plugin-astro"], "overrides": [{ "files": "*.astro", "options": { "parser": "astro" } }] }
```

. See the Prettier cheat sheet for the other options.