# Prettier Opinionated formatter

**START HERE**

**`npx prettier --write .`**
Format everything

**`npx prettier --check .`**
Verify, change nothing

**`prettier -l .`**
List unformatted files

**`prettier --write "src/**/*.ts"`**
Format a glob

**`prettier --single-quote <file>`**
Try an option

**`prettier --stdin-filepath <path>`**
Format piped text

**`.prettierrc`**
Project options

**`.prettierignore`**
Skip files

**Parse** source to AST → **discard layout** → **Reprint** printWidth rules → **write / check** → **Output** stdout, file or exit code

## CONCEPTS

Without --write or --check, Prettier prints the formatted code to stdout and touches nothing.

Prettier is opinionated: few options, and it reprints from the AST, so your original line breaks mostly do not survive.

Config lookup goes up the tree from each file; there is no global config. CLI flags override the file by default.

## RUN

**`prettier --write <path>`**
Format files in place

**`prettier --check <path>`**
Report unformatted files, exit 1

**`prettier --list-different <path>`**
Print names of differing files

**`prettier <file>`**
Print formatted result to stdout

**`prettier --ignore-unknown --write .`**
Skip file types it cannot parse

**`prettier --no-error-on-unmatched-pattern <glob>`**
No error when nothing matches

**`prettier --with-node-modules <path>`**
Include node_modules

**`prettier --log-level silent <path>`**
Quieter or louder output
*error, warn, log (default), debug or silent.*

**`prettier --debug-check <file>`**
Test that formatting is safe
*Cannot be combined with --write.*

## INPUT

**`prettier --stdin-filepath <path>`**
Pick parser and config from a name
*Reads stdin as if it were the file at <path>; also applies its ignore rules.*

**`prettier --parser <name>`**
Force a parser (babel, ts, json, md...)

**`prettier --range-start <n> --range-end <n>`**
Format only a character range

## INPUT (cont.)

**`prettier --insert-pragma <file>`**
Add @format header comment

**`prettier --require-pragma <path>`**
Only format files with the pragma

## COMMON OPTIONS

**`--print-width <n>`**
Line length to wrap at (80)

**`--tab-width <n>`**
Spaces per indent level (2)

**`--use-tabs`**
Indent with tabs

**`--no-semi`**
Omit semicolons

**`--single-quote`**
Single quotes in JS/TS

**`--trailing-comma <all|es5|none>`**
Trailing commas (default all)

**`--prose-wrap <always|never|preserve>`**
Wrap Markdown text

**`--end-of-line <lf|crlf|cr|auto>`**
Line ending style (lf)

**`--arrow-parens <always|avoid>`**
Parens around lone arrow param

**`--quote-props <as-needed|consistent|preserve>`**
Quote object keys

**`--no-bracket-spacing`**
No spaces in { a }

**`--bracket-same-line`**
Put > of multiline tag on last line

**`--jsx-single-quote`**
Single quotes in JSX

**`--object-wrap <preserve|collapse>`**
Keep or collapse object first line

**`--single-attribute-per-line`**
One HTML/JSX attribute per line

**`--html-whitespace-sensitivity <css|strict|ignore>`**
HTML whitespace handling

**`--vue-indent-script-and-style`**
Indent script and style in Vue

## COMMON OPTIONS (cont.)

**`--embedded-language-formatting <off|auto>`**
Format code embedded in files

## CONFIGURATION

**`prettier --config <file> <path>`**
Use a specific config file

**`prettier --no-config <path>`**
Ignore config files, use defaults

**`prettier --find-config-path <file>`**
Show which config applies

**`prettier --config-precedence <mode>`**
cli-override, file-override, prefer-file

**`prettier --no-editorconfig <path>`**
Do not read .editorconfig

**`package.json "prettier" key`**
Config inside package.json
*Highest precedence among config locations.*

**`.prettierrc / .prettierrc.json`**
JSON or YAML options file

**`prettier.config.js / .mjs / .ts`**
Config as code, export default

**`"overrides": [{ "files", "options" }]`**
Per-glob options
*Patterns resolve relative to the config file; add excludeFiles to carve out exceptions.*

**`.editorconfig`**
indent_size, max_line_length are read
*Config file values win over EditorConfig. charset and insert_final_newline are not mapped.*

## IGNORING

**`.prettierignore`**
Gitignore syntax, project root

**`prettier --ignore-path <file>`**
Use another ignore file
*Repeating it replaces the defaults (.gitignore, .prettierignore); list them again to keep them.*

**`// prettier-ignore`**
Skip next JS/TS node

**`<!-- prettier-ignore -->`**
Skip next HTML or Markdown block

## IGNORING (cont.)

**`/* prettier-ignore */`**
Skip next CSS rule

**`# prettier-ignore`**
Skip next YAML node

**`<!-- prettier-ignore-start -->`**
Begin skipped Markdown range
*Close with <!-- prettier-ignore-end -->. Needs a blank line before each comment and top-level placement.*

node_modules and VCS folders are skipped by default, and .gitignore is honoured.

## CACHE AND CI

**`prettier --cache --write .`**
Only reformat changed files

**`prettier --cache-location <file>`**
Where the cache lives
*Default is node_modules/.cache/prettier/.prettier-cache.*

**`prettier --cache-strategy <metadata|content>`**
How changes are detected (content)
*Running without --cache deletes the cache. Plugin versions are not part of the key.*

Exit codes: 0 all formatted, 1 some files unformatted, 2 Prettier error. Use --check in CI.

## RECIPES

### CI gate

```
# fails the job on any unformatted file
npx prettier --check .
```

### Format only staged files

```
git diff --name-only --cached --diff-filter=d -z \
  | xargs -0 prettier --write --ignore-unknown
git add -u
```

### Minimal .prettierrc

```
{
  "singleQuote": true,
  "printWidth": 100,
  "overrides": [{ "files": "*.md", "options": { "proseWrap": "always" } }]
}
```

### Format a snippet from a pipe

```
cat messy.json | prettier --stdin-filepath x.json
echo '{"a":1}' | prettier --parser json
```