Biome Lint and format JS/TS

START HERE

`biome init`
Create biome.json

`biome check .`
Format, lint, organise imports

`biome check --write .`
Apply safe fixes

`biome format --write <path>`
Format only

`biome lint <path>`
Lint only

`biome ci .`
Read-only check for CI

`biome check --staged`
Only staged files

`biome explain <rule>`
What a rule means

biome check one pass, all three
- **Formatter** layout
  - **Linter** rules, safe and unsafe fixes
    - **Assist** organise imports

CONCEPTS

biome check is the one command to remember: it runs the formatter, the linter and import sorting in one pass.

Without --write nothing on disk changes; biome only reports. Fixes marked unsafe also need --unsafe.

Biome supports JS, TS, JSX, JSON, CSS and more, but is not a drop-in for every ESLint plugin or Prettier option.

SETUP

`npm i -D -E @biomejs/biome`
Install pinned in a project
-E pins the exact version so formatting stays stable.

`biome init`
Create biome.json

`biome init --jsonc`
Create biome.jsonc instead

`biome version`
Print the CLI version

`biome migrate --write`
Update config after an upgrade
Run without --write first to preview the changes.

`biome migrate prettier --write`
Import Prettier settings

`biome migrate eslint --write`
Import ESLint settings
Add --include-inspired to also carry over rules Biome only approximates.

`biome upgrade`
Upgrade a standalone binary
Not for npm installs; bump the package instead.

CHECK

`biome check <path>`
Format, lint and assist, report only

`biome check --write <path>`
Apply formatting and safe fixes
--fix is an alias.

`biome check --write --unsafe <path>`
Also apply unsafe fixes
Review the diff; unsafe fixes can change behaviour.

`biome check --staged`
Only staged files

CHECK (cont.)

`biome check --changed`
Files changed versus the default branch
Needs vcs.defaultBranch set, or use --since.

`biome check --changed --since=<ref>`
Changed since a given ref

`biome check --linter-enabled=false`
Skip the linter for this run

`biome check --formatter-enabled=false`
Skip the formatter for this run

`biome check --stdin-file-path=<file>`
Process stdin to stdout

`biome check --watch`
Re-run when files change

FORMAT

`biome format <path>`
Show formatting differences

`biome format --write <path>`
Format files in place

`biome format --staged --write`
Format staged files

`biome format --indent-style=space`
Use spaces instead of tabs
Biome defaults to tabs, unlike Prettier.

`biome format --line-width=<n>`
Set the line width

`biome format --stdin-file-path=<file>`
Format code piped on stdin

LINT

`biome lint <path>`
Lint only

`biome lint --write <path>`
Apply safe lint fixes

`biome lint --write --unsafe <path>`
Apply unsafe lint fixes too

`biome lint --only=<rule>`
Run a single rule or group
For example --only=suspicious or --only=correctness/noUnusedVariables.

`biome lint --skip=<rule>`
Turn a rule or group off for this run

LINT (cont.)

`biome lint --suppress --reason="<why>"`
Add suppression comments to violations
Handy for adopting a rule on a big codebase gradually.

`biome explain <rule>`
Show docs for a rule

`biome search '<pattern>'`
Find code with a GritQL pattern
Experimental; add -l css or -l json for other languages.

CI & OUTPUT

`biome ci .`
Check without modifying files
Fails on format, lint and assist problems; has no --write.

`biome ci --changed`
Only files changed on this branch

`biome ci --reporter=github`
Annotate a GitHub run

`--reporter=<name>`
Pick the output format
Includes json, junit, sarif, gitlab, summary and concise.

`--reporter-file=<path>`
Write report to a file

`--error-on-warnings`
Fail the run on warnings

`--diagnostic-level=error`
Show only errors

`--max-diagnostics=none`
Show every diagnostic
The default cuts the output at 20.

`--no-errors-on-unmatched`
Do not fail when no file matches

`--threads=<n>`
Limit parallelism in CI

CONFIGURATION

biome.json basics

```
{
  "vcs": { "enabled": true,
"clientKind": "git",
           "useIgnoreFile":
true, "defaultBranch":
"main" },
  "files": { "includes": ["src/
**"] },
  "formatter": { "indentStyle":
"space", "lineWidth": 100 },
  "linter": { "rules":
{ "preset": "recommended" } }
}
```

`--config-path=<dir>`
Use a config from another folder

CONFIGURATION (cont.)

`biome rage`
Print a troubleshooting report

Config sections

Key	Controls
`formatter`	indent, width, line ending
`linter`	rules and their severity
`javascript.formatter`	quotes, semicolons, commas
`assist`	import organising and other actions
`overrides`	per-glob settings

Use files.includes with ! patterns to exclude paths; there is no separate ignore file.

A rule can be silenced inline with // biome-ignore lint/<group>/<rule>: <reason>.

EDITOR & HOOKS

`biome lsp-proxy`
Run the language server over stdio

`biome start`
Start the background daemon

`biome stop`
Stop the background daemon

`biome clean`
Remove daemon log files

RECIPES

Adopt in an existing project

```
npm i -D -E @biomejs/biome
npx biome init
npx biome migrate prettier --
write
npx biome migrate eslint --
write
npx biome check --write .
```

Pre-commit on staged files

```
# .git/hooks/pre-commit
npx biome check --staged --
write --no-errors-on-unmatched
git update-index --again
```

CI job

```
npx biome ci --reporter=github
# or only this branch
npx biome ci --changed --
since=origin/main
```

package.json scripts

```
"lint": "biome lint .",
"format": "biome format --
write .",
"check": "biome check --
write ."
```