Version

v8.23.0

Search

USER GUIDE

Getting Started
Configuring
Configuration Files (New)
Configuration Files
Configuring Language Options
Configuring Rules
Configuring Plugins
Ignoring Code
Command Line Interface
Rules
Formatters
Integrations
Migrating to v8.x

DEVELOPER GUIDE

Architecture
Getting the Source Code
Set Up a Development Environment
Run the Tests
Working with Rules
Working with Plugins
Working with Custom Formatters
Working with Custom Parsers
Shareable Configs
Node.js API
Contributing

MAINTAINER GUIDE

Managing Issues
Reviewing Pull Requests
Managing Releases
Governance

Ignoring Code

`ignorePatterns` in Config Files

You can tell ESLint to ignore specific files and directories using `ignorePatterns` in your config files. `ignorePatterns` patterns follow the same rules as `.eslintignore`. Please see the the `.eslintignore` file documentation to learn more.

```
{
    "ignorePatterns": ["temp.js", "**/vendor/*.js"],
    "rules": {
        //...
    }
}
```

- Glob patterns in `ignorePatterns` are relative to the directory that the config file is placed in.
- You cannot write `ignorePatterns` property under `overrides` property.
- Patterns defined in `.eslintignore` take precedence over the `ignorePatterns` property of config files.

If a glob pattern starts with `/`, the pattern is relative to the base directory of the config file. For example, `/foo.js` in `lib/.eslintrc.json` matches to `lib/foo.js` but not `lib/subdir/foo.js`.

If a config is provided via the `--config` CLI option, the ignore patterns that start with `/` in the config are relative to the current working directory rather than the base directory of the given config. For example, if `--config config/.eslintrc.json` is present, the ignore patterns in the config are relative to `.` rather than `./configs`.

The `.eslintignore` File

You can tell ESLint to ignore specific files and directories by creating an `.eslintignore` file in your project's root directory. The `.eslintignore` file is a plain text file where each line is a glob pattern indicating which paths should be omitted from linting. For example, the following will omit all JavaScript files:

```
**/*.js
```

When ESLint is run, it looks in the current working directory to find an `.eslintignore` file before determining which files to lint. If this file is found, then those preferences are applied when traversing directories. Only one `.eslintignore` file can be used at a time, so `.eslintignore` files other than the one in the current working directory will not be used.

Globs are matched using node-ignore, so a number of features are available:

- Lines beginning with `#` are treated as comments and do not affect the ignore patterns.
- Paths are relative to the current working directory. This is also true of paths passed in via the `--ignore-pattern` command.
- Lines preceded by `!` are negated patterns that re-include a pattern that was ignored by an earlier pattern.
- Ignore patterns behave according to the `.gitignore` specification.

Of particular note is that like `.gitignore` files, all paths used as patterns for both `.eslintignore` and `--ignore-pattern` must use forward slashes as their path separators.

```
# Valid
/root/src/*.js

# Invalid
\root\src\*.js
```

Please see `.gitignore`'s specification for further examples of valid syntax.

In addition to any patterns in the `.eslintignore` file, ESLint always follows a couple of implicit ignore rules even if the `--no-ignore` flag is passed. The implicit rules are as follows:

- `node_modules/` is ignored.
- dot-files (except for `.eslintrc.*`), as well as dot-folders and their contents, are ignored.

There are also some exceptions to these rules:

- If the path to lint is a glob pattern or directory path and contains a dot-folder, all dot-files and dot-folders will be linted. This includes dot-files and dot-folders that are buried deeper in the directory structure.

  For example, `eslint .config/` will lint all dot-folders and dot-files in the `.config` directory, including immediate children as well as children that are deeper in the directory structure.

- If the path to lint is a specific file path and the `--no-ignore` flag has been passed, ESLint will lint the file regardless of the implicit ignore rules.

  For example, `eslint .config/my-config-file.js --no-ignore` will cause `my-config-file.js` to be linted. It should be noted that the same command without the `--no-ignore` line will not lint the `my-config-file.js` file.

- Allowlist and denylist rules specified via `--ignore-pattern` or `.eslintignore` are prioritized above implicit ignore rules.

  For example, in this scenario, `.build/test.js` is the desired file to allowlist. Because all dot-folders and their children are ignored by default, `.build` must first be allowlisted so that eslint becomes aware of its children. Then, `.build/test.js` must be explicitly allowlisted, while the rest of the content is denylisted. This is done with the following `.eslintignore` file:

```
# Allowlist 'test.js' in the '.build' folder
# But do not allow anything else in the '.build' folder to be lint
!.build
.build/*
!.build/test.js
```

The following `--ignore-pattern` is also equivalent:

```
eslint --ignore-pattern '!.build' --ignore-pattern '.build/*' --ig
```

Using an Alternate File

If you'd prefer to use a different file than the `.eslintignore` in the current working directory, you can specify it on the command line using the `--ignore-path` option. For example, you can use `.jshintignore` file because it has the same format:

```
eslint --ignore-path .jshintignore file.js
```

You can also use your `.gitignore` file:

```
eslint --ignore-path .gitignore file.js
```

Any file that follows the standard ignore file format can be used. Keep in mind that specifying `--ignore-path` means that any existing `.eslintignore` file will not be used. Note that globbing rules in `.eslintignore` follow those of `.gitignore`.

Using eslintIgnore in package.json

If an `.eslintignore` file is not found and an alternate file is not specified, ESLint will look in package.json for an `eslintIgnore` key to check for files to ignore.

```
{
    "name": "mypackage",
    "version": "0.0.1",
    "eslintConfig": {
        "env": {
            "browser": true,
            "node": true
        }
    },
    "eslintIgnore": ["hello.js", "world.js"]
}
```

Ignored File Warnings

When you pass directories to ESLint, files and directories are silently ignored. If you pass a specific file to ESLint, then you will see a warning indicating that the file was skipped. For example, suppose you have an `.eslintignore` file that looks like this:

```
foo.js
```

And then you run:

```
eslint foo.js
```

You'll see this warning:

```
foo.js
  0:0  warning  File ignored because of a matching ignore pattern. Us

✖ 1 problem (0 errors, 1 warning)
```

This message occurs because ESLint is unsure if you wanted to actually lint the file or not. As the message indicates, you can use `--no-ignore` to omit using the ignore rules.

Consider another scenario where you may want to run ESLint on a specific dot-file or dot-folder, but have forgotten to specifically allow those files in your `.eslintignore` file. You would run something like this:

```
eslint .config/foo.js
```

You would see this warning:

```
.config/foo.js
  0:0  warning  File ignored by default.  Use a negated ignore patter

✖ 1 problem (0 errors, 1 warning)
```

This message occurs because, normally, this file would be ignored by ESLint's implicit ignore rules (as mentioned above). A negated ignore rule in your `.eslintignore` file would override the implicit rule and reinclude this file for linting. Additionally, in this specific case, `--no-ignore` could be used to lint the file as well.

Edit this page

Light Dark

Language

vs English (US)