ESLint

Version

Search

v8.23.0

USER GUIDE

Getting Started

Configuration Files (New)

Configuring Language Options

Configuration Files

Configuring Rules

Configuring Plugins

Command Line Interface

Ignoring Code

Rules

Formatters

Integrations

Architecture

Run the Tests

Working with Rules

Working with Plugins

Shareable Configs

MAINTAINER GUIDE

Reviewing Pull Requests

Managing Releases

Governance

Managing Issues

Node.js API

Contributing

Migrating to v8.x

DEVELOPER GUIDE

Getting the Source Code

Set Up a Development Environment

Working with Custom Formatters

Working with Custom Parsers

Configuring

Plugins

~

^

^

Specifying Parser

By default, ESLint uses Espree as its parser. You can optionally specify that a different parser should be used in your configuration file so long as the parser meets the following requirements:

- 1. It must be a Node module loadable from the config file where the parser is used. Usually, this means you should install the parser package separately using npm.
- 2. It must conform to the parser interface.

Note that even with these compatibilities, there are no guarantees that an external parser will work correctly with ESLint and ESLint will not fix bugs related to incompatibilities with other parsers.

To indicate the npm module to use as your parser, specify it using the parser option in your .eslintrc file. For example, the following specifies to use Esprima instead of Espree:

```
"parser": "esprima",
         "rules": {
             "semi": "error"
5
6
```

<u>Esprima</u>

The following parsers are compatible with ESLint:

- <u>@babel/eslint-parser</u> A wrapper around the <u>Babel</u> parser that makes it compatible with
- ESLint. • otypescript-eslint/parser - A parser that converts TypeScript into an ESTree-compatible form
- so it can be used in ESLint. Note when using a custom parser, the parserOptions configuration property is still required for

parserOptions and may or may not use them to determine which features to enable. Specifying Processor

ESLint to work properly with features not in ECMAScript 5 by default. Parsers are all passed

Plugins may provide processors. Processors can extract JavaScript code from other kinds of files,

then let ESLint lint the JavaScript code or processors can convert JavaScript code in preprocessing for some purpose. To specify processors in a configuration file, use the processor key with the concatenated string of

a plugin name and a processor name by a slash. For example, the following enables the processor

 α -processor that the plugin α -plugin provided:

```
"plugins": ["a-plugin"],
               "processor": "a-plugin/a-processor"
    4
To specify processors for specific kinds of files, use the combination of the overrides key and the
processor key. For example, the following uses the processor \alpha-plugin/markdown for *.md files.
```

```
"plugins": ["a-plugin"],
              "overrides": [
    3
                      "files": ["*.md"],
                       "processor": "a-plugin/markdown"
    8
    9
Processors may make named code blocks such as 0. js and 1. js. ESLint handles such a named
```

code blocks in the overrides section of the config. For example, the following disables the strict rule for the named code blocks which end with . js in markdown files.

code block as a child file of the original file. You can specify additional configurations for named

```
"plugins": ["a-plugin"],
             "overrides": [
                      "files": ["*.md"],
    5
                      "processor": "a-plugin/markdown"
                      "files": ["**/*.md/*.js"],
                      "rules": {
  10
                           "strict": "off"
  11
  12
  13
  14
  15
ESLint checks the file path of named code blocks then ignores those if any overrides entry didn't
```

Configuring Plugins

match the file path. Be sure to add an overrides entry if you want to lint named code blocks other

npm.

To configure plugins inside of a configuration file, use the plugins key, which contains a list of plugin names. The eslint-plugin- prefix can be omitted from the plugin name.

ESLint supports the use of third-party plugins. Before using the plugin, you have to install it using

"plugins": ["plugin1",

```
"eslint-plugin-plugin2"
   5
   6
And in YAML:
```

4

plugins:

- plugin1

than *.js.

```
- eslint-plugin-plugin2
Notes:
  1. Plugins are resolved relative to the config file. In other words, ESLint will load the plugin as a
    user would obtain by running require('eslint-plugin-pluginname') in the config file.
```

2. Plugins in the base configuration (loaded by extends setting) are relative to the derived config file. For example, if . / .eslintrc has extends: ["foo"] and the eslint-config-foo has

- plugins: ["bar"], ESLint finds the eslint-plugin-bar from ./node_modules/ (rather than ./node_modules/eslint-config-foo/node_modules/) or ancestor directories. Thus every plugin in the config file and base configurations is resolved uniquely.
- Naming convention

The eslint-plugin- prefix can be omitted for non-scoped packages

5

6

Use a plugin

convention:

Include a plugin

// ... "plugins": ["jquery", // means eslint-plugin-jquery

```
5
             // ...
   6
The same rule does apply to scoped packages:
             // ...
              "plugins": [
```

"afoobar" // means afoobar/eslint-plugin

"ajquery/jquery", // means ajquery/eslint-plugin-jquery

// ...

 eslint-plugin-foo → foo/a-rule afoo/eslint-plugin → afoo/α-config

When using rules, environments or configs defined by plugins, they must be referenced following the

- afoo/eslint-plugin-bar → afoo/bar/a-environment For example:

```
// ...
          "plugins": [
              "jquery", // eslint-plugin-jquery
              "afoo/foo", // afoo/eslint-plugin-foo
 5
                          // abar/eslint-plugin
              "abar"
 6
          "extends": [
 8
              "plugin:@foo/foo/recommended",
              "plugin:abar/recommended"
10
11
          "rules": {
12
              "jquery/a-rule": "error",
13
              "afoo/foo/some-rule": "error",
14
15
              "abar/another-rule": "error"
16
          },
          "env": {
17
18
              "jquery/jquery": true,
19
              "afoo/foo/env-foo": true,
              "abar/env-bar": true,
20
21
22
23
```

© OpenJS Foundation and ESLint

□ Dark

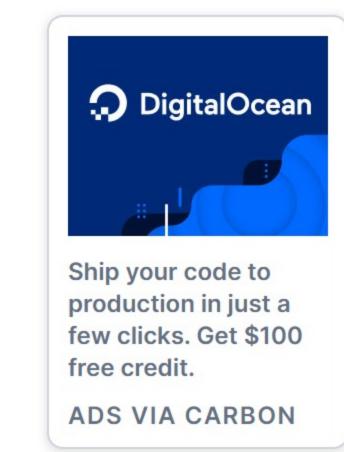
~

contributors, www.openjsf.org

us English (US)

Language

Edit this page



Playground

Donate

Blog

Docs

Store

Team

- Specifying Parser
- **Specifying Processor**
- L Use a plugin

