

[Skip to main content](#)

[Donate](#)

- [Team](#)
- [Blog](#)
- [Docs](#)
- [Store](#)
- [Playground](#)

[Versions](#)

Version Switcher

Selecting a version will take you to the chosen version of the ESLint docs.

Version HEAD v9.9.0 v8.57.0 Previous Versions

[Versions](#)

Version Switcher

Selecting a version will take you to the chosen version of the ESLint docs.

Version HEAD v9.9.0 v8.57.0 Previous Versions

Index

Search

Results will be shown and updated as you type.

[Powered by](#)

Clear search

- [Use ESLint in Your Project](#)
 - [Getting Started](#)
 - [Core Concepts](#)
 - [Glossary](#)
 - [Configure ESLint](#)
 - [Configuration Files](#)
 - [Configure Language Options](#)
 - [Configure Rules](#)
 - [Configure Plugins](#)
 - [Configure a Parser](#)
 - [Combine Configs](#)
 - [Ignore Files](#)
 - [Debug Your Configuration](#)
 - [Configuration Migration Guide](#)
 - [Command Line Interface Reference](#)
 - [Rules Reference](#)
 - [Feature Flags](#)
 - [Formatters Reference](#)
 - [Integrations](#)

- [Migrate to v9.x](#)
- [Version Support](#)
- [Troubleshooting](#)
 - [ESLint couldn't determine the plugin ... uniquely](#)
 - [ESLint couldn't find the config ... to extend from](#)
 - [ESLint couldn't find the plugin ...](#)
 - [TypeError: context.getScope is not a function](#)
- [Extend ESLint](#)
 - [Ways to Extend ESLint](#)
 - [Create Plugins](#)
 - [Custom Rule Tutorial](#)
 - [Custom Rules](#)
 - [Custom Processors](#)
 - [Migration to Flat Config](#)
 - [Share Configurations](#)
 - [Custom Formatters](#)
 - [Custom Parsers](#)
 - [Stats Data](#)
- [integrate ESLint](#)
 - [Integrate with the Node.js API Tutorial](#)
 - [Node.js API Reference](#)
- [Contribute to ESLint](#)
 - [Code of Conduct](#)
 - [Report Bugs](#)
 - [Propose a New Rule](#)
 - [Propose a Rule Change](#)
 - [Request a Change](#)
 - [Architecture](#)
 - [Set up a Development Environment](#)
 - [Run the Tests](#)
 - [Package.json Conventions](#)
 - [Work on Issues](#)
 - [Submit a Pull Request](#)
 - [Contribute to Core Rules](#)
 - [Governance](#)
 - [Report a Security Vulnerability](#)
- [Maintain ESLint](#)
 - [How ESLint is Maintained](#)
 - [Manage Issues and Pull Requests](#)
 - [Review Pull Requests](#)
 - [Manage Releases](#)
 - [Working Groups](#)

Package.json Conventions

Table of Contents

1. [Names](#)
2. [Order](#)
3. [Main Script Names](#)
 1. [Build](#)
 2. [Fetch](#)
 3. [Release](#)
 4. [Lint](#)
 5. [Start](#)
 6. [Test](#)
4. [Modifiers](#)
 1. [Fix](#)
 2. [Target](#)
 3. [Options](#)
 4. [Watch](#)

The following applies to the “scripts” section of package.json files.

Names

npm script names MUST contain only lower case letters, : to separate parts, - to separate words, and + to separate file extensions. Each part name SHOULD be either a full English word (e.g. coverage not cov) or a well-known initialism in all lowercase (e.g. wasm).

Here is a summary of the proposal in ABNF.

```
name           = life-cycle / main target? option* ":watch"?
life-cycle     = "prepare" / "preinstall" / "install" / "postinstall" /
"prepublish" / "preprepare" / "prepare" / "postprepare" / "prepack" /
"postpack" / "prepublishOnly"
main           = "build" / "lint" ":fix"? / "release" / "start" / "test"
/ "fetch"
target         = ":" word ("-" word)* / extension ("+" extension)*
option         = ":" word ("-" word)*
word           = ALPHA +
extension      = ( ALPHA / DIGIT )+
```

- 1
- 2
- 3
- 4
- 5
- 6
- 7

Order

The script names MUST appear in the package.json file in alphabetical order. The other

conventions outlined in this document ensure that alphabetical order will coincide with logical groupings.

Main Script Names

With the exception of [npm life cycle scripts](#) all script names MUST begin with one of the following names.

Build

Scripts that generate a set of files from source code and / or data MUST have names that begin with build.

If a package contains any build:* scripts, there MAY be a script named build. If so, SHOULD produce the same output as running each of the build scripts individually. It MUST produce a subset of the output from running those scripts.

Fetch

Scripts that generate a set of files from external data or resources MUST have names that begin with fetch.

If a package contains any fetch:* scripts, there MAY be a script named fetch. If so, it SHOULD produce the same output as running each of the fetch scripts individually. It MUST produce a subset of the output from running those scripts.

Release

Scripts that have public side effects (publishing the web site, committing to Git, etc.) MUST begin with release.

Lint

Scripts that statically analyze files (mostly, but not limited to running eslint itself) MUST have names that begin with lint.

If a package contains any lint:* scripts, there SHOULD be a script named lint and it MUST run all of the checks that would have been run if each lint:* script was called individually.

If fixing is available, a linter MUST NOT apply fixes UNLESS the script contains the :fix modifier (see below).

Start

A start script is used to start a server. As of this writing, no ESLint package has more than one start script, so there's no need start to have any modifiers.

Test

Scripts that execute code in order to ensure the actual behavior matches expected behavior MUST have names that begin with test.

If a package contains any test:* scripts, there SHOULD be a script named test and it MUST run of all of the tests that would have been run if each test:* script was called individually.

A test script SHOULD NOT include linting.

A test script SHOULD report test coverage when possible.

Modifiers

One or more of the following modifiers MAY be appended to the standard script names above. If a target has modifiers, they MUST be in the order in which they appear below (e.g.

lint:fix:js:watch not lint:watch:js:fix)

Fix

If it's possible for a linter to fix problems that it finds, add a copy of the script with :fix appended to the end that also fixes.

Target

The name of the target of the action being run. In the case of a build script, it SHOULD identify the build artifact(s), e.g. "javascript" or "css" or "website". In the case of a lint or test script, it SHOULD identify the item(s) being linted or tested. In the case of a start script, it SHOULD identify which server is starting.

A target MAY refer to a list of affected file extensions (such as cjs or less) delimited by a +. If there is more than one extension, the list SHOULD be alphabetized. When a file extension has variants (such as cjs for CommonJS and mjs for ESM), the common part of the extension MAY be used instead of explicitly listing out all of the variants (e.g. js instead of cjs+jsx+mjs).

The target SHOULD NOT refer to name of the tool that's performing the action (eleventy, webpack, etc.)

Options

Additional options that don't fit under the other modifiers.

Watch

If a script watches the filesystem and responds to changes, add :watch to the script name.

[Edit this page](#)

Table of Contents

1. [Names](#)
2. [Order](#)
3. [Main Script Names](#)
 1. [Build](#)

2. [Fetch](#)
3. [Release](#)
4. [Lint](#)
5. [Start](#)
6. [Test](#)
4. [Modifiers](#)
 1. [Fix](#)
 2. [Target](#)
 3. [Options](#)
 4. [Watch](#)

Social Media

-
-
-
-

© OpenJS Foundation and ESLint contributors, www.openjsf.org. Content licensed under [MIT License](#).



Theme Switcher

Light System Dark

[Change Language](#)

Language Switcher

Selecting a language will take you to the ESLint website in that language.

Language  English (US) (Latest)  简体中文 (最新)