

Linting



Cory House

@housecor

bitnative.com



Here's the Plan



Why Lint?

Linting Tools

Configuration approaches for ESLint

Set up ESLint



Enforce Consistency. Avoid Mistakes.

Enforce Consistency

Curly brace position

confirm / alert

Trailing commas

Globals

eval

Avoid Mistakes

Extra parenthesis

Overwriting function

Assignment in conditional

Missing default case in switch

debugger / console.log



Pick a Linter



Use this



Configuring ESLint



Decision #1: Config Format



Configuration File Formats

ESLint supports configuration files in several formats:

- **JavaScript** - use `.eslintrc.js` and export an object containing your configuration.
- **YAML** - use `.eslintrc.yaml` or `.eslintrc.yml` to define the configuration structure.
- **JSON** - use `.eslintrc.json` to define the configuration structure. ESLint's JSON files also allow JavaScript-style comments.
- **Deprecated** - use `.eslintrc`, which can be either JSON or YAML.
- **package.json** - create an `eslintConfig` property in your `package.json` file and define your configuration there.

If there are multiple configuration files in the same directory, ESLint will only use one. The priority order is:

1. `.eslintrc.js`
2. `.eslintrc.yaml`
3. `.eslintrc.yml`
4. `.eslintrc.json`
5. `.eslintrc`
6. `package.json`

Decision #1: Configuration Location

Dedicated config file

Not tied to npm

package.json

One less file



Configuring ESLint via package.json

```
{  
  "name": "mypackage",  
  "version": "0.0.1",  
  "eslintConfig": {  
    "plugins": ["example"],  
    "env": {  
      "example/custom": true  
    }  
  }  
}
```



Decision #2:
Which rules?



Rules

Rules in ESLint are divided into several categories to help you better understand their value. Though none are enabled by default, you can turn on rules that ESLint recommends by specifying your configuration to inherit from `eslint:recommended`. The rules that will be enabled when you inherit from `eslint:recommended` are indicated below as "(recommended)". For more information on how to configure rules and inherit from `eslint:recommended`, please see the [configuration documentation](#).

Some rules are fixable using the `--fix` command line flag. Those rules are marked as "(fixable)" below.

Possible Errors

The following rules point out areas where you might have made mistakes.

- `comma-dangle` - disallow or enforce trailing commas (recommended)
- `no-cond-assign` - disallow assignment in conditional expressions (recommended)
- `no-console` - disallow use of `console` in the node environment (recommended)
- `no-constant-condition` - disallow use of constant expressions in conditions (recommended)
- `no-control-regex` - disallow control characters in regular expressions (recommended)
- `no-debugger` - disallow use of `debugger` (recommended)
- `no-dupe-args` - disallow duplicate arguments in functions (recommended)
- `no-dupe-keys` - disallow duplicate keys when creating object literals (recommended)
- `no-duplicate-case` - disallow a duplicate case label. (recommended)
- `no-empty-character-class` - disallow the use of empty character classes in regular expressions (recommended)
- `no-empty` - disallow empty statements (recommended)
- `no-ex-assign` - disallow assigning to the exception in a `catch` block (recommended)
- `no-extra-boolean-cast` - disallow double-negation boolean casts in a boolean context (recommended)
- `no-extra-parens` - disallow unnecessary parentheses
- `no-extra-semi` - disallow unnecessary semicolons (recommended) (fixable)
- `no-func-assign` - disallow overwriting functions written as function declarations (recommended)
- `no-inner-declarations` - disallow function or variable declarations in nested blocks (recommended)
- `no-invalid-regexp` - disallow invalid regular expression strings in the `RegExp` constructor (recommended)
- `no-irregular-whitespace` - disallow irregular whitespace outside of strings and comments (recommended)
- `no-negated-in-lhs` - disallow negation of the left operand of an `in` expression (recommended)
- `no-obj-calls` - disallow the use of object properties of the global object (`Math` and `JSON`) as functions (recommended)
- `no-regex-spaces` - disallow multiple spaces in a regular expression literal (recommended)
- `no-sparse-arrays` - disallow sparse arrays (recommended)



Core Decisions

1

Config format?

2

Which built-in rules?

3

Warnings or errors?

4

Which plugins?

5

Use preset instead?



Decision #3: Warnings or errors?



Decision #3: Warnings vs Errors

Warning

Can continue development

Can be ignored

Team must agree: Fix warnings

Error

Breaks the build

Cannot be ignored

Team is forced to comply



Decision #4: Which plugins?





find packages



Greetings, coxautokc



★ eslint-plugin-react public

React specific linting rules for ESLint

status **maintained** npm **v6.2.0** build **passing** build **passing** devDependencies **up-to-date** coverage **97%**
code climate **2.9**

React specific linting rules for ESLint

Installation

Install **ESLint** either locally or globally.

```
$ npm install eslint
```

If you installed **ESLint** globally, you have to install React plugin globally too. Otherwise, install it locally.

```
$ npm install eslint-plugin-react
```

Configuration

Add plugins section and specify ESLint-plugin-React as a plugin.

```
{  
  "plugins": [  
    "react"  
  ]  
}
```

📄 npm i eslint-plugin-react

[how? learn more](#)

yannickcr published 5 days ago

6.2.0 is the latest of 88 releases

github.com/yannickcr/eslint-plugin-react

MIT

Collaborators



Stats

64 357 downloads in the last day

338 628 downloads in the last week

1 455 252 downloads in the last month

128 open issues on GitHub

24 open pull requests on GitHub





find packages



Greetings, coxautokc



★ eslint-plugin-angular public

ESLint rules for AngularJS projects

ESLint rules for your angular project with checks for best-practices, conventions or potential errors.

build **passing** dependencies **none** devDependencies **out-of-date** chat on **gitter** coverage **100%**

Summary

This repository will give access to new rules for the ESLint tool. You should use it only if you are developing an AngularJS application.

Since the 0.0.4 release, some rules defined in **John Papa's Guideline** have been implemented. In the description below, you will have a link to the corresponding part of the guideline, in order to have more information.

Contents

- [Usage with shareable config](#)
- [Usage without shareable config](#)
- [Rules](#)
- [Need your help](#)
- [How to create a new rule](#)
- [Default ESLint configuration file](#)
- [Who uses it?](#)
- [Team](#)

npm i eslint-plugin-angular

[how? learn more](#)

 **emmanueldemey** published a month ago

1.3.1 is the latest of 49 releases

github.com/Gillespie59/eslint-plugin-angularjs

MIT 

Collaborators



Stats

4 791 downloads in the last day

28 488 downloads in the last week

115 073 downloads in the last month

56 open issues on GitHub

3 open pull requests on GitHub





find packages



Greetings, coxautokc



★ eslint-plugin-node public

Additional ESLint's rules for Node.js

npm v2.0.0 downloads 15k/month build passing coverage 99% dependencies up to date

Additional ESLint's rules for Node.js

Install & Usage

```
> npm install --save-dev eslint eslint-plugin-node
```

- Requires Node.js `^0.10.0 || ^0.12.0 || ^4.0.0 || >=6.0.0`
- Requires ESLint `>=2.0.0`

.eslintrc

```
{
  "plugins": [ "node" ],
  "extends": [ "eslint:recommended", "plugin:node/recommended" ]
}
```

Configs

This plugin provides `plugin:node/recommended` preset config. This preset config:

- enables the environment of ES2015 (ES6) and Node.js.
- enables rules which are given ★ in the following table.

Note: It recommends a use of the "engines" field of `package.json`. The "engines" field is used by **no-unsupported-features** rule.

Rules

Private packages for the whole team

It's never been easier to manage developer teams with varying permissions and multiple projects. [Learn more about Private Packages and Organizations...](#)

`npm i eslint-plugin-node`[how? learn more](#) **mysticatea** published 2 months ago**2.0.0** is the latest of 23 releasesgithub.com/mysticatea/eslint-plugin-node

MIT

Collaborators



Stats

235 downloads in the last day**3 790** downloads in the last week**15 063** downloads in the last month**5 open issues** on GitHub**No open pull requests** on GitHub

Awesome ESLint

A list of awesome ESLint configs, plugins, etc.

`build` `passing` `PRs` `welcome`

If you want to contribute, please read the [contribution guidelines](#).



Table of Contents

- [Configs](#)
- [Parsers](#)
- [Plugins](#)
 - [Frameworks and Libraries](#)
 - [Misc.](#)
 - [Practices](#)
 - [Style](#)
- [Preconfigured Tools with ESLint Set up](#)
- [Tools](#)
- [Tutorials](#)

Configs

- [Airbnb](#) - Shareable config for [Airbnb's style guide](#)
- [Canonical](#) - Shareable config for [Canonical style guide](#)
- [ESLint](#) - Shareable config for ESLint's default settings
- [Google](#) - Shareable config for the [Google style](#)
- [Shopify](#) - Shareable config for [Shopify's style guide](#)
- [Standard](#) - Shareable config for JavaScript [Standard Style](#)
- [XO](#) - Shareable config for [XO](#)

Parsers

- [Babel](#) - Use Babel's parser for linting all Babel features

Decision #5:
Use a preset?



Decision #5: Use a Preset?



From scratch



Recommended



Presets

Issue:
ESLint doesn't watch files.



eslint-loader

Re-lints all files upon save.

eslint-watch

ESLint wrapper that adds file watch

Not tied to webpack

Better warning/error formatting

Displays clean message

Easily lint tests and build scripts too



Issue:
ESLint doesn't support many
experimental JavaScript features



Run ESLint directly

Supports ES6 and ES7 natively

Also supports object spread

Babel-eslint

Also lints stage 0 - 4 features



Stage-X (Experimental Presets)

Any transforms in stage-x presets are changes to the language that haven't been approved to be part of a release of Javascript (such as ES6/ES2015).

“Changes to the language are developed by way of a process which provides guidelines for evolving an addition from an idea to a fully specified feature”

Subject to change

These proposals are subject to change so ***use with extreme caution***, especially for anything pre stage-3.

The [TC39](#) categorises proposals into 4 stages:

- [stage-0](#) - Strawman: just an idea, possible Babel plugin.
- [stage-1](#) - Proposal: this is worth working on.
- [stage-2](#) - Draft: initial spec.
- [stage-3](#) - Candidate: complete spec and initial browser implementations.
- stage-4 - Finished: will be added to the next yearly release.

**Want to use these features?
Try babel-eslint.**

Also check out the [current tc39 proposals](#) and its [process document](#).

Why Lint via an Automated Build Process?



1. One place to check
2. Universal configuration
3. Part of continuous integration

Demo



Set up ESLint

- ESLint Recommended
- eslint-watch



Demo



Run linting as part of npm start



Wrap Up



Why Lint?

- Enforce consistency
- Avoid mistakes

We chose ESLint

Configuration choices

- Config format
- Which rules?
- Warnings vs errors?
- Which plugins?
- Use preset instead?

ESLint runs automatically

Next up: HTTP

