JavaScript - Code Quality - ESLint

Overview

To ensure consistent code style and readability, it is common to use a code linter to check and enforce a set of coding styles. ESLint is one of the most popular applications for linting JavaScript code, being very configurable and accepting a range of different style rule sets, from Standard to Google.

Install

Add the ESLint npm package using npm or yarn:

yarn add eslint --dev

Setup the initial configuration using the init command and following the steps:

yarn eslint --init

Run

To run use the eslint command and the file to lint:

eslint <options> <file>

Directories can also be passed to eslint and all files within will be scanned. Files and directories can be chained.

ESLint also accepts the glob syntax in string form, for example to scan all in a directory:

eslint "lib/\*\*"

Fix

ESLint can fix problems automatically using the fix flag:

eslint --fix index.js

Add Script to Yarn

To make linting directories quicker, add the eslint script to the package.json, for example:

scripts": {

"test": "jest --no-cache",

"eslint": "eslint --fix \"app/javascript/packs/controllers/\*\*/\*\"",

"eslint-spec": "eslint --fix \"spec/javascript/\*\*/\*\""

},

ESLint can then be run using:

yarn eslint

Add Babel

ESLint sometimes cannot parse ES6/7 features, therefore it is common to add babel-eslint to transpile any new features to ESLint compatible code.

First add the babel-eslint dependency to the project:

yarn add --dev babel-eslint

Then add the parser details to the .eslintrc configuration file:

"parser": "babel-eslint"

Rules For Directories

ESLint configuration files are hierarchical and can therefore be added to directories to give them specific configuration. For example, to turn off some rules for a specific directory, add a new .eslintrc file in the directory with the following:

{

"rules": {

"no-undef": "off"

}

}