No-config JavaScript/TypeScript style enforcement with StandardJS and Prettier

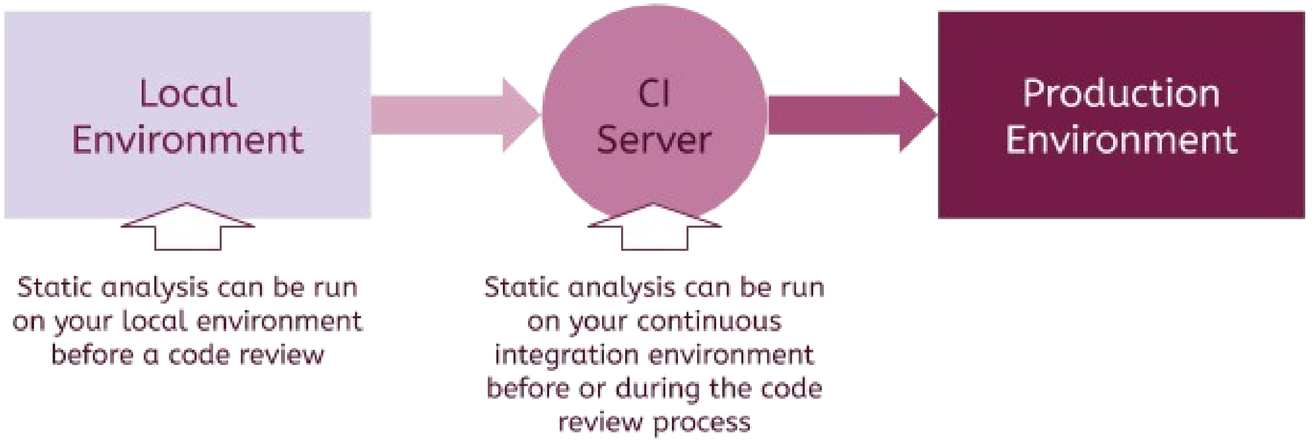
Marcin Lewandowski, mlewandowski@pl.ibm.com
Senior Software Engineer, Krakow Software Lab, IBM

What is Static Analysis?

Static analysis, also called **Static Code Analysis**, is the practice of analyzing source code before it is running.

Static analysis can be fully automated, therefore it’s one of the best ways to improve the quality of your JavaScript code without investing developer time.

Programs which automate static code analysis are called **Linters**. Except for reporting problems, modern linters can also automatically fix some of the issues if you request them to do so.



Analyzing JavaScript with ESLint

ESLint is a configurable JavaScript linter that helps you find and fix problems in your JavaScript code. Problems can be anything from potential runtime bugs, to not following best practices, to styling issues.

Rules are the core building block of ESLint. A rule validates if your code meets a certain expectation, and what to do if it does not meet that expectation.


```
module.exports = {
  parser: '@babel/eslint-parser',
  extends: [
    'airbnb',
    'plugin:react-hooks/recommended',
    'plugin:react-redux/recommended',
    'plugin:security/recommended',
  ],
  env: {···
  },
  plugins: [
    '@babel',
    'react',
    'jsx-a11y',
    'react-intl',
    'flowtype',
    'eslint-plugin-formatjs',
    'react-redux',
  ],
  parserOptions: {···
  },
  rules: {
    'no-underscore-dangle': 'off',
    'no-multiple-empty-lines': 'off',
    'no-tabs': 'error',
    'no-restricted-exports': 'warn',
    'default-case-last': 'error',
    'no-unsafe-optional-chaining': 'error',
    'function-call-argument-newline': ['error', 'consistent'],
    'no-promise-executor-return': 'error',
    'prefer-exponentiation-operator': 'off',
    'prefer-regex-literals': 'off',
    'arrow-parens': [
      'error',
      'always',
    ],
    'arrow-body-style': [
      'error',
      'as-needed',
```

StandardJS: JavaScript Standard Style

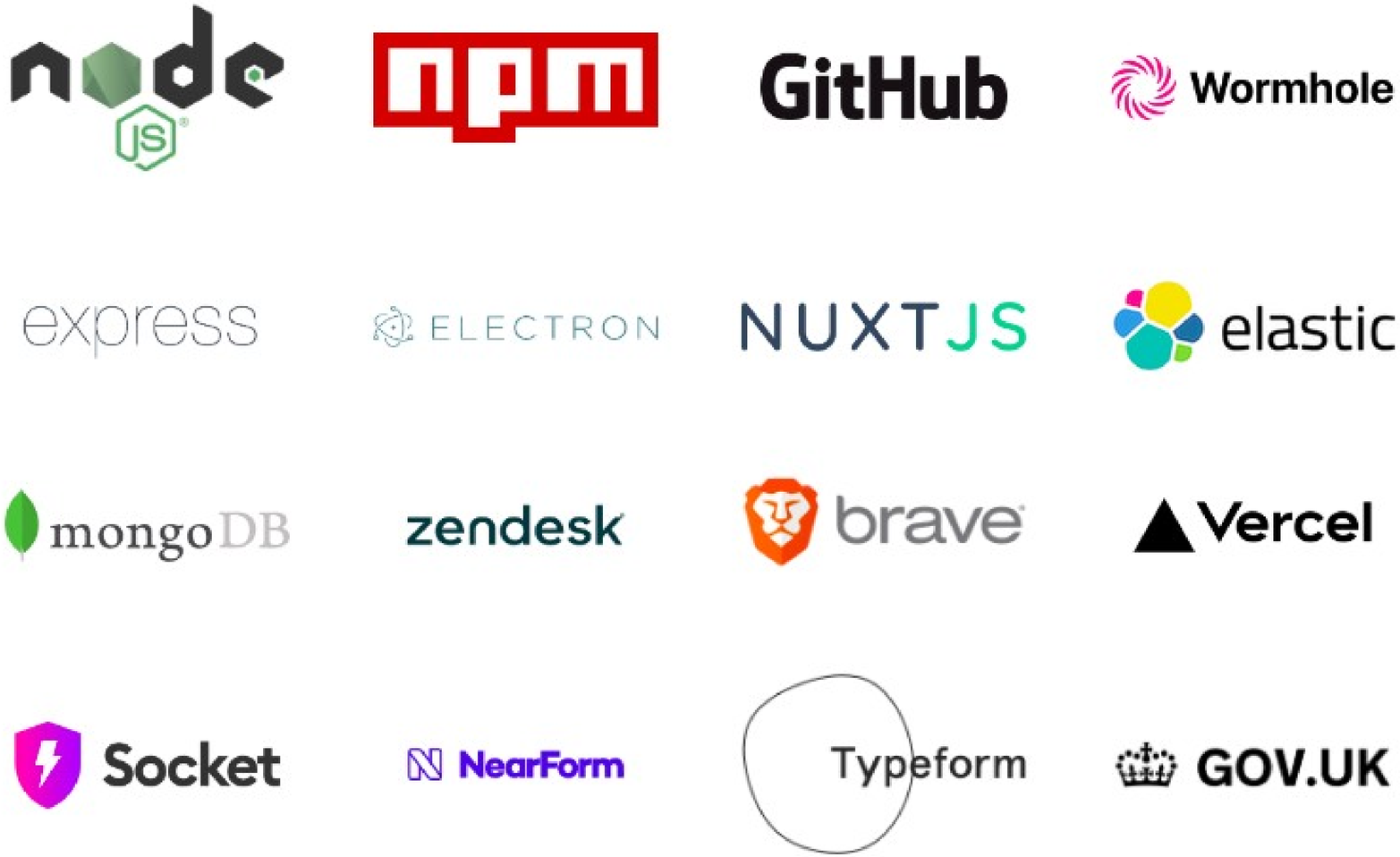
No configuration. The easiest way to enforce code quality in your project. No decisions to make. No .eslintrc files to manage. It just works.

Automatically format code. Just run standard --fix and say goodbye to messy or inconsistent code.

Catch style issues & programmer errors early. Save precious code review time by eliminating back-and-forth between reviewer & contributor.

StandardJS uses ESLint / ESLint rules under the hood.

Who uses JavaScript Standard Style



ts-standard

TypeScript Style Guide, with linter and automatic code fixer based on StandardJS.

Install globally with `sudo npm i -g ts-standard` and just run:

```
ts-standard
```

OR:

```
ts-standard --fix
```

Integrate with your existing `eslint` setup:

```
yarn add -D \
  eslint-config-standard-with-typescript \
  @typescript-eslint/parser \
  @typescript-eslint/eslint-plugin \
  eslint-plugin-import \
  eslint-plugin-n \
  eslint-plugin-promise

# .eslintrc.json
{
  "extends": ["next/core-web-vitals", "standard-with-typescript"],
  "parserOptions": {
    "project": "./tsconfig.json"
  }
}
```

Prettier: performance at stake

Formatting rules:

- max-len
- no-mixed-spaces-and-tabs
- keyword-spacing
- comma-style
- ...

Code-quality rules:

- no-unused-vars
- no-extra-bind
- no-implicit-globals
- prefer-promise-reject-errors
- ...

Enabling Prettier with standard styles

Necessary packages:

- prettier – the tool itself
- prettier-config-standard – configures Prettier to match Standard style
- eslint-config-prettier – disables eslint styling rules (must come last)

Configure Prettier for Standard style:

Disable styling rules in `eslint`:

```
yarn add -D prettier \
            prettier-config-standard
            eslint-config-prettier



# package.json
{
  "scripts": {
    ...
    "format": "prettier --write '**/*.+(js|ts|tsx|json)'"
  }
  ...
  "prettier": "prettier-config-standard"
}


# .eslintrc.json
{
  "extends": ["next/core-web-vitals", "standard-with-typescript", "prettier"],
  "parserOptions": {
    "project": "./tsconfig.json"
  }
}
```

References

- ESLint: https://eslint.org/
- StandardJS: https://standardjs.com/
- Prettier: https://prettier.io/
- Going Beyond ESLint: An Overview of Static Analysis in JavaScript: https://www.telerik.com/blogs/going-beyond-eslint-overview-static-analysis-javascript
- Feross Aboukhadijeh: Write Perfect Code With Standard And ESLint - JSConf.Asia 2018: https://www.youtube.com/watch?v=kuHfMw8j4xk
- eslint-config-standard-with-typescript: https://www.npmjs.com/package/eslint-config-standard-with-typescript
- prettier-config-standard: https://www.npmjs.com/package/prettier-config-standard

Thank you.