Drupal + Technology

TRACK SUPPORTED BY

platform.sh
Code with style!
Prettier + ESLint

David Corbacho Román

https://www.drupaleurope.org/session/code-style-prettier-eslint
We want to deliver software without flaws.
How to deliver software without flaws.

Ask your team members to review your code
Code reviews

reviewed on 6 May 2015

```javascript
import { moduleForModel, test } from 'ember-qunit'
```

Minor style nitpick, this blank line is not needed
Code reviews

Reviewed on 6 May 2015

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>@-1,7</td>
<td>+1,7</td>
<td>@</td>
<td>@</td>
<td>@</td>
<td>@</td>
<td>@</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>import { Router } from 'express'</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>import mongoose from 'mongoose'</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>import { NotFoundError } from '../src/error'</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>- import { Task, Attendance } from '../models'</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>+ import { Task, Activity } from '../models'</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Minor style nitpick, align from other from keywords
I don't really like this indentation, what about:

```javascript
'\n'military',
'school',
'civil defence'
```
Code reviews

reviewed on 12 Jun

```css
public/styles.css

360 + font-size: 15px;
361 + line-height: 20px;
362 + color: #333333;
363 + }

```

on 12 Jun  Contributor

Watch the indentation and newline here again!
Add a space after colon
friction noun [U] (DISAGREEMENT)

Disagreement or unfriendliness caused by people having different opinions:

There's a lot of friction between my wife and my mother.
Unpopular opinion:
Maybe... perfect formatting is not *that* important
Unpopular opinion:

Maybe... perfect formatting is not *that* important

... and one semicolon here
Following a convention is important
How to deliver software without flaws.

Static Code Analysis
Tools that inspect the code without executing the software.

Linters

Dynamic Code Analysis
aka “testing”
How to deliver software without flaws.

Static Code Analysis
Tools that inspect the code **without** executing the software.

Linters

- CODE SMELL
- PATTERNS
- BEST PRACTICES
- SECURITY FLAWS
- STYLE
Linting in JavaScript. Evolution

**JSLint**

*2002* by Douglas Crockford, JSON creator.
Linting evolution

2002 by Douglas Crockford, JSON creator.
Too opinionated.

Yeah, well, you know, that's just, like,

YOUR OPINION, MAN
Linting evolution

JSLint

2002 by Douglas Crockford, JSON creator.
Too opinionated.

JS Hint

2011. Fork of JSLint
Community driven
Not opinionated
More configurable
Linting evolution

2002 by Douglas Crockford, JSON creator. 
Too opinionated.

2011. Fork of JSLint
Community driven
Not opinionated
More configurable

2013 by Nicholas Zakas.
Not opinionated
Lots of configuration options
Extensible with plugins
+ Supports NodeJS
+ Keeps up with ES evolution
+ In 2014 support for JSX
ESLint demo

```javascript
var day;
let week = 5;
console.log(week);
```

<table>
<thead>
<tr>
<th>Messages</th>
<th>Fixed Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:5 - 'day' is defined but never used. (no-unused-vars)</td>
<td></td>
</tr>
<tr>
<td>2:5 - 'week' is never reassigned. Use 'const' instead. (prefer-const)</td>
<td></td>
</tr>
<tr>
<td>3:1 - Unexpected console statement. (no-console)</td>
<td></td>
</tr>
</tbody>
</table>
Too much configuration

Rules

Rules in ESLint are grouped by category to help you understand their purpose.

Possible Errors

These rules relate to possible syntax or logic errors in JavaScript code.
Too much configuration

ESLint “Shareable configurations”

- **Airbnb** - Shareable config for Airbnb's style guide
- **Canonical** – Shareable config for Canonical style guide
- **ESLint** - Contains the ESLint configuration used for projects maintain
- **ES** - Shareable config for very strict code
- **Facebook** - Shareable config for Facebook’s style guide.
- **Google** - Shareable config for the Google style
- **Problems** - Shareable config that only catches actual problems, and
- **Shopify** - Shareable config for Shopify's style guide
- **Standard** - Shareable config for JavaScript Standard Style
- **Supermind** - Shareable config for Supermind style
- **XO** - Shareable config for XO

https://github.com/dustinspecker/awesome-eslint
## Airbnb JS Style guide

### 2,728,137 repository results

<table>
<thead>
<tr>
<th>Repository</th>
<th>License</th>
<th>Updated</th>
<th>Issues</th>
<th>Stars</th>
</tr>
</thead>
<tbody>
<tr>
<td>freeCodeCamp/freeCodeCamp</td>
<td>JavaScript</td>
<td>a day ago</td>
<td>39</td>
<td>294k</td>
</tr>
<tr>
<td>twbs/bootstrap</td>
<td>CSS</td>
<td>6 hours ago</td>
<td>10</td>
<td>127k</td>
</tr>
<tr>
<td>vuejs/vue</td>
<td>JavaScript</td>
<td>3 days ago</td>
<td>2</td>
<td>113k</td>
</tr>
<tr>
<td>EbookFoundation/free-programming-books</td>
<td>JavaScript</td>
<td>a day ago</td>
<td>5</td>
<td>111k</td>
</tr>
<tr>
<td>facebook/react</td>
<td>JavaScript</td>
<td>3 days ago</td>
<td>3</td>
<td>111k</td>
</tr>
<tr>
<td>tensorflow/tensorflow</td>
<td>C++</td>
<td>2 hours ago</td>
<td></td>
<td>109k</td>
</tr>
<tr>
<td>sindresorhus/awesome</td>
<td></td>
<td>5 days ago</td>
<td>5</td>
<td>92.1k</td>
</tr>
<tr>
<td>getify/You-Dont-Know-JS</td>
<td></td>
<td>12 days ago</td>
<td></td>
<td>88k</td>
</tr>
<tr>
<td>d3/d3</td>
<td>JavaScript</td>
<td></td>
<td></td>
<td>78.6k</td>
</tr>
<tr>
<td>robbyrussell/oh-my-zsh</td>
<td>Shell</td>
<td>3 hours ago</td>
<td></td>
<td>76k</td>
</tr>
<tr>
<td>airbnb/javascript</td>
<td>JavaScript</td>
<td>3 days ago</td>
<td></td>
<td>75.8k</td>
</tr>
</tbody>
</table>
Airbnb JS Style guide

Excellent opinionated guide to ES6 using ESLint rules.

References

- 2.1 Use `const` for all of your references; avoid using `var`.
  
  ```javascript
  // bad
  var a = 1;
  var b = 2;
  
  // good
  const a = 1;
  const b = 2;
  ```

https://github.com/airbnb/javascript
Yeah, well, you know, that's just, like, YOUR OPINION, MAN
Too opinionated?
Creates friction in the team?

Escape hatch:
Override Airbnb rules

Example Drupal 8 core

```json
{
    "extends": "eslint-config-airbnb",
    "root": true,
    "env": {
        "browser": true,
        "es6": true,
        "node": true
    },
    "globals": {
        "Drupal": true,
        "drupalSettings": true,
        ...
    }
}
```

```json
"rules": {
    "consistent-return": [0],
    "no-underscore-dangle": [0],
    "max-nested-callbacks": [1, 3],
    "no-mutable-exports": [1],
    "no-plusplus": [1, {
        "allowForLoopAfterthoughts": true
    }],
    "no-param-reassign": [0],
    "no-prototype-builtins": [0],
    "valid-jscs": [1, {
        "prefer": {
            "returns": "return",
            "property": "prop"
        },
        "requireReturn": false
    }],
    "brace-style": ["error", "stroustrup"],
    "no-unused-vars": [1]
}
```
How Linters work? AST parser

```javascript
var foo = bar;
```

```json
{
  "type": "Program",
  "start": 0,
  "end": 14,
  "body": [
    {
      "type": "VariableDeclaration",
      "start": 0,
      "end": 14,
      "declarations": [
        {
          "type": "VariableDeclarator",
          "start": 4,
          "end": 13,
          "id": {
            "type": "Identifier",
            "start": 4,
            "end": 7,
            "name": "foo"
          },
          "init": {
            "type": "Identifier",
          }
        }
      ]
    }
  ]
}
```
WHAT IF...
We use the output of the AST parser to **rewrite** the source code?

gofmt was born. 2013 [Go formatter]

Also rfmt [Reason formatter]
WHAT IF...

We do it for JavaScript? using Wadler's algorithm from "A prettier printer" and lots of smart tweaks.

Prettier was born
Jan. 2017
by James Long
Prettier

prettier.io
Prettier

https://www.youtube.com/watch?v=hkfBvpEfWdA
```javascript
}

getValue() {

render() {
    return (
        <div>
            <Field>
                <input
                    type="text"
                    value={getValue()}
                    onFocus={() => this.setState({ mode: 'edit' })}
                    onBlur={this.onBlur.bind(this)}
                    onChange={this.onChange.bind(this)}
                    onKeyDown={this.onKeyDown.bind(this)}
                    style={this.props.style}
                
```
Man, after using @prettiercode for a month, I'd love to have all the time back that I've wasted over the years manually formatting code!

Prettier is a great example of this maxim: "A luxury, once sampled, becomes a necessity."
Prettier. Not only JS.

Sept 2018 supported languages

ES2017
JSX
Flow
TypeScript
Vue
JSON

CSS3+
Less
SCSS
styled-components
styled-jsx

GraphQL
GraphQL Schemas

CommonMark
GitHub Flavored Markdown
MDX

YAML

Work in Progress
Elm (via elm-format)
Java
PHP
PostgreSQL
Python
Ruby
Swift
Prettier. Opinionated.

Only 9 style rules

- Print Width
- Tab Width
- Tabs
- Semicolons
- Quotes
- Trailing Commas
- Bracket Spacing
- JSX Brackets
- Arrow Function Parentheses
Prettier. Opinionated.

A hard pill to swallow for artisans

... and one semicolon here
Prettier will hurt your feelings

It will not respect columns alignment or other whitespace “decorations”

```javascript
const columns = [
  { name: 'id', index: 'id' },
  { name: 'invdate', index: 'invdate' },
  { name: 'name', index: 'name_asc, invdate' },
  { name: 'amount', index: 'amount' },
  { name: 'tax', index: 'tax' },
  { name: 'total', index: 'total' },
  { name: 'note', index: 'note' }
];
```
I don't like prettier
I don't like prettier

Nobody loves what prettier does to their syntax
Everyone loves what prettier does to their coworkers' syntax
Prettier. Opinionated.

... but a quick win for juniors
Who uses Prettier

Facebook is 💯% Prettier

Another exciting announcement is that all the JavaScript files in the Facebook codebase have now been converted to Prettier!

Feb. 2018 https://prettier.io/blog/2018/02/26/1.11.0.html
Benefits of Prettier

- Reduces Friction. [Prettier chooses formatting for you]
- Speed up Code Reviews
- Instant feedback
- 1 thing less to multi-task while coding
- Faster copy/pasting and refactoring
Cons of Prettier

- One more tool in the belt to maintain
- No custom formatting possible
- Not for everyone

My approach to code formatting is the best.

If it weren’t, I wouldn’t use it. Formatting is part of my being—an extension of my genetic makeup. And, when I write code, my formatting is the fingerprint that I leave behind. When I’m asked to change my style, I am—quite literally—asked to deny a fundamental Truth of my being.

— Ben Nadel
Ok, this needs to be said: Prettier is not a replacement for ESLint. Prettier is a source code formatter. ESLint is a code quality tool.
+ ### Does Prettier replace ESLint?

+ No, ESLint does both traditional linting (looking for problematic patterns) and style checking (enforcement of conventions). You can use ESLint for everything, or you can combine both using Prettier to format your code and ESLint to catch possible errors.
Conclusion: Before

- Linters
  - CODE SMELL
  - PATTERNS
    - BEST PRACTICES
    - SECURITY FLAWS
  - FORMATTING
Conclusion: Now

- CODE SMELL
- PATTERNS
- BEST PRACTICES
- SECURITY FLAWS
- FORMATTING

**Linters**

**Code reviews**

**Prettier**
Configure Prettier + ESLint

- `yarn add -D eslint eslint-config-airbnb`
  - (you might want just `eslint-config-airbnb-base`, if not using React)

- `yarn add -D prettier eslint-plugin-prettier eslint-config-prettier`
  - it will override (disable) eslint rules not compatible with prettier

Order matters

---

`.prettierrc.json`

```json`
{
  "printWidth": 80,
  "semi": true,
  "singleQuote": true,
  "trailingComma": "all"
}
```

`.eslintrc.json`

```json`
{
  "extends": [
    "airbnb",
    "plugin:prettier/recommended"
  ],
}
```
Drupal 8.6 comes with Prettier. This is an effort to standardize the formatting of our JavaScript in addition to the rules provided by ESLint.

https://www.drupal.org/node/2986680
Drupal. PHP Linter

PHP Code Sniffer
Installing Coder Sniffer
https://www.drupal.org/node/1419988

Helps to enforce coding standards
https://www.drupal.org/docs/develop/standards/coding-standards

Coder module contains the “sniffs”
https://www.drupal.org/project/coder
Using Prettier in your project

**Option 1.** Ask every developer to install Prettier on their editor and auto-save (it adds friction to the team, and could limit editor choice, personal workflow)

**Option 2.** Run in a pre-commit hook
Use lint-staged for easily setting up git commit hooks.
Introduce Prettier to an existing project

First: Agree with the team!

If only part of the team uses Prettier, it will warranty a lot of friction. Prettier by default formats the whole file, not only your changes. This multiplies the chances of git conflicts for the whole team and lengthy code reviews because of unrelated changes.

**Option 1.**
Apply Prettier to the whole project (if medium size project)
It will dirt git history

**Option 2.**
Go back to the past. Rewrite git history like Prettier would have been there from the day one.
Drupal. CSS linters

In the process of migrating from CSS Lint to STYLE Lint

Drupal stylelint config:

```json
{
    "extends": "stylelint-config-standard",
    "plugins": [
        "stylelint-no-browser-hacks/lib"
    ],
    "rules": {
        "comment-empty-line-before": null,
        "function-linear-gradient-no-nonstandard-direction": null,
        "function-whitespace-after": null,
        "no-descending-specificity": null,
        "no-duplicate-selectors": null,
        "no-unknown-animations": true,
        "media-feature-name-no-unknown": [true, {
            "ignoreMediaFeatureNames": ["prefers-reduced-motion"]
        }],
        "number-leading-zero": "always",
        "plugin/no-browser-hacks": [true, {
```
David Corbacho Román
Developer at Wunder
@dcorbacho
Become a Drupal contributor
Friday from 9am

- First timers workshop
- Mentored contribution
- General contribution