ESLintのローカルルールで独自のコーディング規約を実装する

berlysia / Lightning TechTalks #2

誰

- berlysia（べるりしあ、と読む）
  - Web、特にブラウザの周りに関心

- 株式会社ドワンゴ所属
  教育事業のWebフロント担当
  - Webフロントをやる人
  - Webフロントのためにいろいろやる人

ドワンゴの教育事業

N予備校

KADOKAWA・ドワンゴが創るネットの高校

N高等学校

S高等学校

ZEN University

(仮称) (設置認可申請中)

disclaimer

- ESLintでローカルルールを実装する方法について話します
- 具体的なルール・プラグインの作り方の話はしません

コーディング規約

ありますか

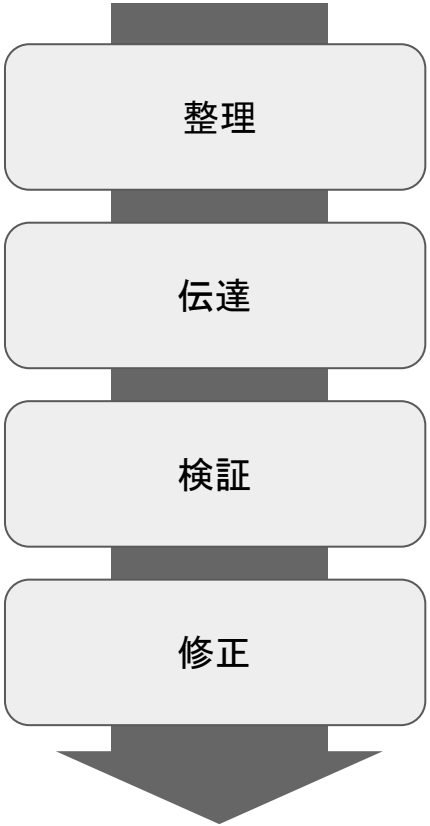
コーディング規約

どうしてますか

コーディング規約の対象

- コードフォーマット
- 命名のお作法
- コメント、(コード内)ドキュメンテーションのお作法
- エラー処理のお作法
- モジュール、コンポーネントの分割のお作法
- 依存関係のお作法
- ディレクトリ構造のお作法
- テストのお作法
- ↑↑今日の話題はここまで↑↑
- ~~コミットメッセージのお作法~~
- ~~プルリクエストのお作法~~
- etc…

コーディング規約の実現のカタ



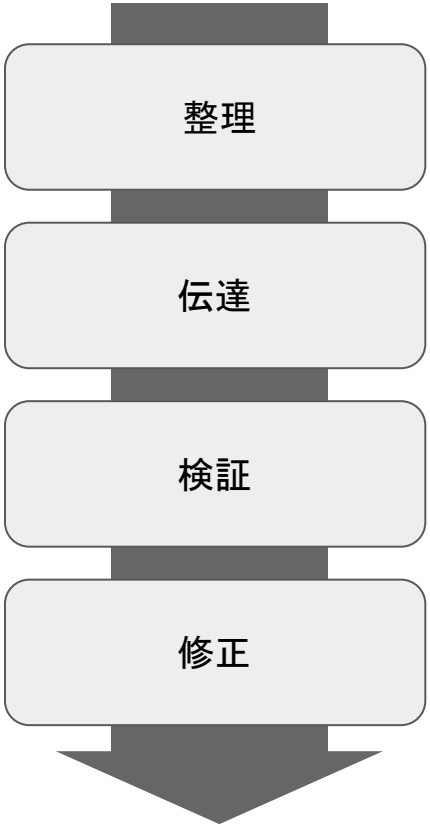
- 文書としてまとめる
- スタイルガイドにまとめる

- 研修やワークショップ
- ペアプログラミング、コードレビュー

- コードレビューで指摘する
- Linterによる自動検知

- 手で直す
- Linter、Formatterによる自動修正

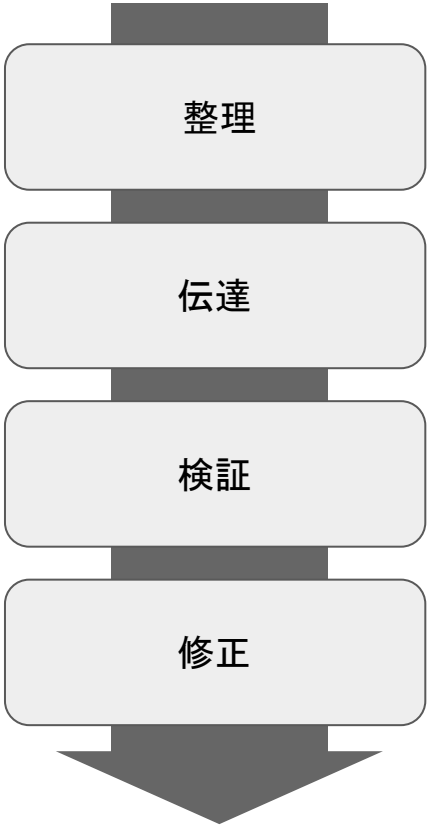
コーディング規約の実現のカタ



- **文書としてまとめる**
- **スタイルガイドにまとめる**
- **研修やワークショップ**
- **ペアプログラミング、コードレビュー**
- **コードレビューで指摘する**
- Linterによる自動検知
- **手で直す**
- Linter、Formatterによる自動修正

コスト高いが
背景を残したり
高次な話までできる

コーディング規約の実現のカタ



- 文書としてまとめる
- スタイルガイドにまとめる

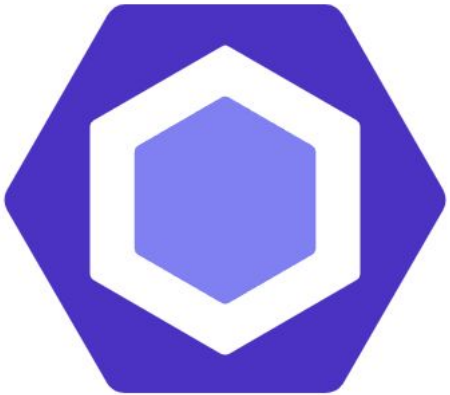
- 研修やワークショップ
- ペアプログラミング、コードレビュー

- コードレビューで指摘する
- **Linterによる自動検知**

- 手で直す
- **Linter、Formatterによる自動修正**

検出と修正は
できるだけ
自動でやりたい
何なら文書を
読みたくない

使っていますか



ESLint

https://eslint.org/

ESLint alternatives

- Deno Linter
  - https://docs.deno.com/runtime/manual/tools/linter
  - Nodeのプロジェクトでも普通に使える。Rust製
- Biome Linter
  - https://biomejs.dev/linter/
  - 元Rome。Rust製
- oxlint
  - https://github.com/web-infra-dev/oxc#-linter
  - Rust製

よくありそう[要出典]な会話

「あ、これはこっちの書き方をするやつですよね」

「これってESLintのルールないですかね」

1. 「しらないですね」
    - はい
2. 「このプラグインにありますね」
    - 歩くESLintプラグインカタログや検索巧者がいるとこうなりがち
3. 「このコアルールを使うと表現できますね」
    - no-restricted-syntaxを使うと任意の構文を検知できる
        - コードからASTが透けて見えない人は AST explorerを使うとよい
        - 構文以外の情報が必要な場合はむずかしい

よくありそう[要出典]な会話、修正編

「あ、これはこっちの書き方をするやつですよね」

「これってESLintのルールないですかね」

「このルールで検知できそうですね」

「いちいち直すのもあれですし、自動修正できないもんですかね？」

「このルールになければないですね……」

自動修正までしたいと思うと、ピッタリの道具がないことがある

ルールがなければ、作りたい

- ESLint: **ある**
  - （皆さんご存じ）プラグインの仕組みがある https://eslint.org/docs/latest/extend/plugins
- Deno Linter: **ない**
  - Issueはある https://github.com/denoland/deno_lint/issues/175
  - PoC実装されていたようだが消えたようだ https://github.com/denoland/deno_lint/pull/1045
- Biome Linter: **ない**、予定もない
  - Discussionは立っているが、調査から必要だね、という状況
- oxlint: **ないが、予定はある**
  - trustfallというGraphQLサブセット言語で書くプラグインシステムを計画中
  - https://github.com/web-infra-dev/oxc#linter-plugin

実は独自のルールを作れるのは現状ESLintしかない

プラグイン／ルールを作るのは別に難しくない

```
module.exports = {
  rules: {
    "no-function-call": {
      create(context) {
        return {
          CallExpression(node) {
            context.report({
              node,
              message: "関数の呼び出しは禁止されています"
            })
          }
        }
      }
    }
  }
};
```

が、ルールは元々こうやって参照する

```
{
  "rules": [
    "import/no-deprecated": "error"
  ]
}
```

eslint-plugin-import の no-deprecated というルール

ESLintが内部的に require(“eslint-plugin-import”) みたいなことをする
ESLint pluginが node_modules 配下にあることを期待している
プラグインはパッケージとして参照できなければいけない……

ローカルのルールを読む仕組みがかつてあった、が

```
--rulesdir
```

Deprecated: Use rules from plugins instead.

This option allows you to specify another directory from which to load rules files. This allows you to dynamically load new rules at run time. This is useful when you have custom rules that aren't suitable for being bundled with ESLint.

プラグインとしてpublishしないためのアプローチ

rulesdirがやっていたようなことをやるESLint pluginが作られた

- https://github.com/cletusw/eslint-plugin-local-rules
  - eslint-local-rules.js / eslint-local-rules/index.js をルールの一覧として受け取る
- https://github.com/not-an-aardvark/eslint-plugin-rulesdir
  - ルールが配置されているディレクトリを文字列で与える
- https://github.com/taskworld/eslint-plugin-local
  - .eslintplugin.js / .eslintplugin/index.js をプラグインとして受け取る

プロジェクト固有のルールが欲しければ、こういった道具を使うか、
観念してpluginとしてpublishする必要があった——**だが今は違う！**

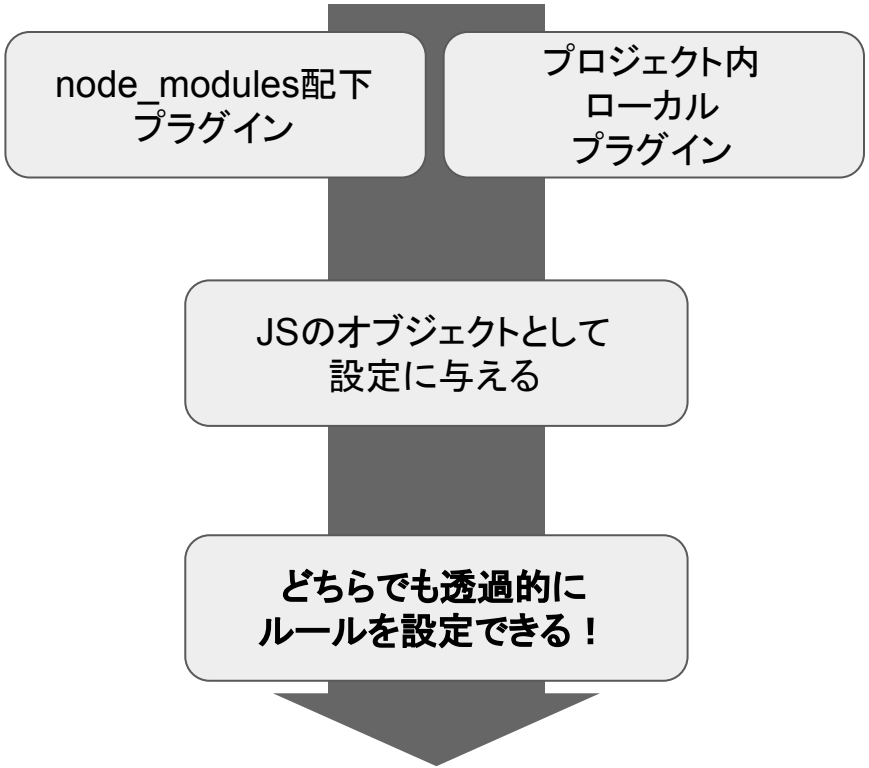
新しいFlat Configの世界では——

Published 05 Aug, 2022 under API Changes

ESLint's new config system, Part 2: Introduction to flat config

ESLint's new config system, nicknamed flat config, is designed to be both familiar and much simpler than the original config system.

Flat Configの世界ではプラグインを利用者が解決する




```
import react from "eslint-plugin-react";
import localPlugin from "./local-plugin";

export default [
  {
    files: ["src/**/*.js"],
    plugins: {
      react,
      local: localPlugin,
    },
    rules: {
      "react/jsx-uses-react": "error",
      "local/no-function-call": "error",
    },
  },
];
```

Flat Configの世界ではプラグインを利用者が解決する

node_modules配下
プラグイン

プロジェクト内
ローカル
プラグイン

JSのオブジェクトとして
設定に与える

どちらでも透過的に
ルールを設定できる！

ローカルルール
誰でも簡単
書き放題

書こう！

```
import react from "eslint-plugin-react";
import localPlugin from "./local-plugin";

export default [
  {
    files: ["**/*.js"],
    plugins: {
      react,
      local: localPlugin,
    },
    rules: {
      "react/jsx-uses-react": "error",
      "local/no-function-call": "error",
    },
  },
];
```

追記：ご参考にどうぞ

新卒エンジニアがESLintのFlat Config移行と格闘した話 - ドワンゴ教育サービス開発者ブログ

おまけ：我々のコーディング規約の考え方

大きいことは文書化、細かいことはLinter/Formatterに任せて自動化

- 文書化して周知やストック
  - 全体の思想みたいな文書
  - 大きいレベルのリファクタリング方針
- Linterで検知だけ
  - モジュールの参照方向違反など自動修正しにくいもの
  - 実装パターン移行中で新規に増えてほしくないもの
- Linterで検知して自動修正
  - 良い感じに揃ってることに嬉しさがあるもの
- **「気にしない」と決める**
  - 好みの問題のやつ