eslint-plugin-vue を使用して継続的に Vue3 移行する

Vue Fes Japan Online 2022 - 2022.10.16 SUN

Press Space for next page →

自己紹介

太田 洋介

- 年齢: アラフォー　📍: 神奈川県
- GitHub: @ota-meshi ♡ Sponsor , npm: ota-meshi, Twitter: @omoteota, Qiita: @ota-meshi
- 所属:
  - フューチャー株式会社 **（プラチナスポンサー！）** 社員 (2015/06 -)
  - Vue.js eslint-plugin-vue メンテナー (eslint-plugin-vue 2018/08 - , Vue 2019/07 -)
  - Stylelint Owners チーム (2020/09 -)
  - Intlify eslint-plugin-vue-i18n メンテナー (2020/07 -)
  - Stylus チーム (2022/06 -)
  - ESLint Community チーム (2022/09 -)
- WEB+DB PRESS Vol.120 「最新 Vue.js 3 入門」 共同執筆 (2020/12/24)
- Google Open Source Peer Bonus 2022 受賞

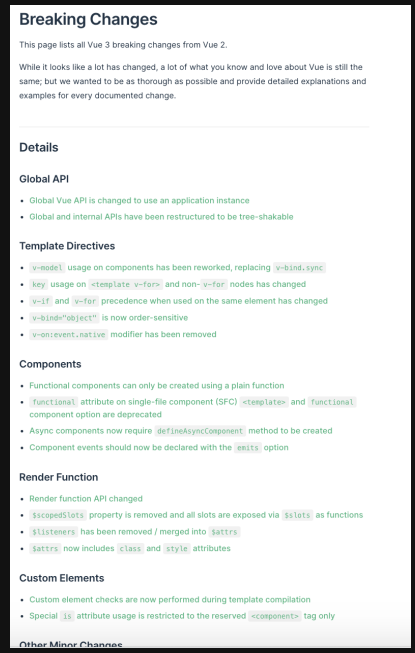
アジェンダ

1. Vue3 の Breaking Changes
2. eslint-plugin-vue とは
3. eslint-plugin-vue を使用して Vue3 移行してみた
4. Vue3 移行に ESLint を使うメリット
5. まとめ

Vue3 移行が大変ですか? 🤔

Vue3 の Breaking Changes

https://v3-migration.vuejs.org/breaking-changes/



Breaking Changes

This page lists all Vue 3 breaking changes from Vue 2.

While it looks like a lot has changed, a lot of what you know and love about Vue is still the same; but we wanted to be as thorough as possible and provide detailed explanations and examples for every documented change.

Details

Global API

- Global Vue API is changed to use an application instance
- Global and internal APIs have been restructured to be tree-shakable

Template Directives

- `v-model` usage on components has been reworked, replacing `v-bind.sync`
- `key` usage on `<template v-for>` and non-`v-for` nodes has changed
- `v-if` and `v-for` precedence when used on the same element has changed
- `v-bind="object"` is now order-sensitive
- `v-on:event.native` modifier has been removed

Components

- Functional components can only be created using a plain function
- `functional` attribute on single-file component (SFC) `<template>` and `functional` component option are deprecated
- Async components now require `defineAsyncComponent` method to be created
- Component events should now be declared with the `emits` option

Render Function

- Render function API changed
- `$scopedSlots` property is removed and all slots are exposed via `$slots` as functions
- `$listeners` has been removed / merged into `$attrs`
- `$attrs` now includes `class` and `style` attributes

Custom Elements

- Custom element checks are now performed during template compilation
- Special `is` attribute usage is restricted to the reserved `<component>` tag only

Other Minor Changes

などなど、38 項目

Vue3 の Breaking Changes

Global API

- Global Vue API is changed to use an application instance
- Global and internal APIs have been restructured to be tree-shakable

Template Directives

- `v-model` usage on components has been reworked, replacing `v-bind.sync`
- `key` usage on `<template v-for>` and non-`v-for` nodes has changed
- `v-if` and `v-for` precedence when used on the same element has changed
- `v-bind="object"` is now order-sensitive
- `v-on:event.native` modifier has been removed

Components

- Functional components can only be created using a plain function
- `functional` attribute on single-file component (SFC) `<template>` and `functional` component option are deprecated
- Async components now require `defineAsyncComponent` method to be created
- Component events should now be declared with the `emits` option

Render Function

- Render function API changed
- `$scopedSlots` property is removed and all slots are exposed via `$slots` as functions
- `$listeners` has been removed / merged into `$attrs`
- `$attrs` now includes `class` and `style` attributes

Custom Elements

- Custom element checks are now performed during template compilation
- Special `is` attribute usage is restricted to the reserved `<component>` tag only

Other Minor Changes

- The `destroyed` lifecycle option has been renamed to `unmounted`
- The `beforeDestroy` lifecycle option has been renamed to `beforeUnmount`
- Props `default` factory function no longer has access to `this` context
- Custom directive API changed to align with component lifecycle and `binding.expression` removed
- The `data` option should always be declared as a function
- The `data` option from mixins is now merged shallowly
- Attributes coercion strategy changed
- Some transition classes got a rename
- `<TransitionGroup>` now renders no wrapper element by default
- When watching an array, the callback will only trigger when the array is replaced. If you need to trigger on mutation, the `deep` option must be specified.
- `<template>` tags with no special directives (`v-if/else-if/else`, `v-for`, or `v-slot`) are now treated as plain elements and will result in a native `<template>` element instead of rendering its inner content.
- Mounted application does not replace the element it's mounted to
- Lifecycle `hook:` events prefix changed to `vnode-`

Removed APIs

- `keyCode` support as `v-on` modifiers
- $on, $off and $once instance methods
- Filters
- Inline templates attributes
- `$children` instance property
- `propsData` option
- `$destroy` instance method. Users should no longer manually manage the lifecycle of individual Vue components.
- Global functions `set` and `delete`, and the instance methods `$set` and `$delete`. They are no longer

eslint-plugin-vue

eslint-plugin-vue とは

Vue.js 用の ESLint プラグイン（Vue オフィシャル！）

- ESLint は JavaScript のコードをチェックするリンター
- `*.vue` ファイルを解析したり、Vue に特化した検証ルールを提供します
- チェックルールによっては問題を**自動的に修正**します（ESLint の機能）
- **Vue3 移行に便利**なルールもいくつか提供しています
- 過去の資料:
  - Vue.js v-tokyo オンライン Meetup#12 の発表資料
    https://docs.google.com/presentation/d/1JFS9DiTxUsrlGfYr72n9QRPibgYB-TzSTB8hi6mq4wY/edit?usp=sharing
  - TechFeed Conference 2022 での発表資料
    https://docs.google.com/presentation/d/18Q8nn69Hi8d39k51HduArKkrCx3CY_ZsbPf-F-tb8Pg/edit?usp=sharing
- 私は eslint-plugin-vue のメンテナーです

eslint-plugin-vue を使用して Vue3 移行してみた

～ ESLint でどの程度 Vue3 移行をサポートできたのか? ～

Vue3 の Breaking Changes

https://v3-migration.vuejs.org/breaking-changes/

Global API

- Global Vue API is changed to use an application instance
- Global and internal APIs have been restructured to be tree-shakable

Template Directives

- `v-model` usage on components has been reworked, replacing `v-bind.sync`
- `key` usage on `<template v-for>` and non-`v-for` nodes has changed
- `v-if` and `v-for` precedence when used on the same element has changed
- `v-bind="object"` is now order-sensitive
- `v-on:event.native` modifier has been removed

Components

- Functional components can only be created using a plain function
- `functional` attribute on single-file component (SFC) `<template>` and `functional` component option are deprecated
- Async components now require `defineAsyncComponent` method to be created
- Component events should now be declared with the `emits` option

Render Function

- Render function API changed
- `$scopedSlots` property is removed and all slots are exposed via `$slots` as functions
- `$listeners` has been removed / merged into `$attrs`
- `$attrs` now includes `class` and `style` attributes

Custom Elements

- Custom element checks are now performed during template compilation
- Special `is` attribute usage is restricted to the reserved `<component>` tag only

Other Minor Changes

- The `destroyed` lifecycle option has been renamed to `unmounted`
- The `beforeDestroy` lifecycle option has been renamed to `beforeUnmount`
- Props `default` factory function no longer has access to `this` context
- Custom directive API changed to align with component lifecycle and `binding.expression` removed
- The `data` option should always be declared as a function
- The `data` option from mixins is now merged shallowly
- Attributes coercion strategy changed
- Some transition classes got a rename
- `<TransitionGroup>` now renders no wrapper element by default
- When watching an array, the callback will only trigger when the array is replaced. If you need to trigger on mutation, the `deep` option must be specified.
- `<template>` tags with no special directives (`v-if/else-if/else`, `v-for`, or `v-slot`) are now treated as plain elements and will result in a native `<template>` element instead of rendering its inner content.
- Mounted application does not replace the element it's mounted to
- Lifecycle `hook:` events prefix changed to `vnode-`

Removed APIs

- `keyCode` support as `v-on` modifiers
- $on, $off and $once instance methods
- Filters
- Inline templates attributes
- `$children` instance property
- `propsData` option
- `$destroy` instance method. Users should no longer manually manage the lifecycle of individual Vue components.

Vue3 の Breaking Changes

以下ではこのように表記します。

Global API

- createApp()等
- 名前付き Export

Template Directives

- `v-model`の`modelValue`, `emit('input')`
- `v-bind.sync`廃止
- `<template v-for>`の key
- `v-if`・`v-for`の優先度
- `v-bind="object"`が順序に影響
- `v-on:event.native`廃止

Components

- 関数型コンポーネントの定義方法
- SFC の関数型コンポーネント廃止
- 非同期コンポーネントの定義方法
- `emits`オプション

Render Function

- `render`関数の変更
- `$scopedSlots`廃止
  `$slots`の関数化
- `$listeners`廃止
- `$attrs`の変更

Custom Elements

- カスタム要素の扱い
- `is`の記法

Other Minor Changes

- `unmounted`
- `beforeUnmount`
- `default`の`this`
- カスタムディレクティブ定義方法
- `data`関数
- mixin の`data`の扱い
- `:attr=false`
- Transition クラス名変更
- `<TransitionGroup>`のルート要素
- 配列の watch
- ディレクティブ無しの`<template>`
- Vue アプリのルート
- `hook:`イベント

Removed APIs

- `v-on`キー修飾子
- `$on`・`$off`・`$once`
- フィルター
- Inline テンプレート
- `$children`
- `propsData`
- `$destroy()`
- `Vue.set()`・`Vue.delete()`

Vue3 の Breaking Changes を ESLint でチェックする

💚: eslint-plugin-vue でチェック可能 / 🔧: 自動修正可能 / 💡: エディタ上での提案

createApp()等	❌
名前付き Export	❌
`v-model`の`modelValue`, `emit('input')`	💚💡(※)
`v-bind.sync`廃止	💚🔧
`<template v-for>`の key	💚
`v-if`・`v-for`の優先度	💚
`v-bind="object"`が順序に影響	❌
`v-on:event.native`廃止	💚
関数型コンポーネントの定義方法	❌
SFC の関数型コンポーネント廃止	💚
非同期コンポーネントの定義方法	❌
`emits`オプション	💚💡
`render`関数の変更	❌
`$scopedSlots`廃止	💚🔧
`$slots`の関数化	💚
`$listeners`廃止	💚
`$attrs`の変更	❌
カスタム要素の扱い	❌
`is`の記法	💚
`unmounted`	💚💡
`beforeUnmount`	💚💡

`default`の`this`	💚
カスタムディレクティブ定義方法	❌
`data`関数	💚🔧
mixin の`data`の扱い	❌
`:attr=false`	❌
Transition クラス名変更	❌
`<TransitionGroup>`のルート要素	❌
配列の watch	❌
ディレクティブ無しの`<template>`	💚
Vue アプリのルート	❌
`hook:`イベント	❌
`v-on`キー修飾子	💚🔧
`$on`・`$off`・`$once`	💚
フィルター	💚
Inline テンプレート	💚
`$children`	❌
`propsData`	❌
`$destroy()`	❌
`Vue.set()`・`Vue.delete()`	❌

※ 設定が必要です。詳細は vue/no-restricted-custom-eventとvue/no-restricted-propsを参照してください。

半分しかチェックできない?

まだまだチェックできます！

`$children`、`$destroy()`もチェックする

プラグインではなく ESLint 本体に付属しているルールに、
任意のプロパティをチェックできるルールがあるので、それを使用してチェックできます。

https://eslint.org/docs/latest/rules/no-restricted-properties

```
module.exports = {
  // ...
  rules: {
    // ...
    "no-restricted-properties": [
      "error",
      {
        property: "$children",
        message: "廃止されたプロパティです",
      },
      {
        property: "$destroy",
        message: "廃止されたメソッドです",
      },
    ],
    // ...
  },
  // ...
};
```

```
foo () {
  const vm = this
  const child = vm.$children[0]
```

any

'$children' is restricted from being used. 廃止されたプロパティです
eslint(no-restricted-properties)

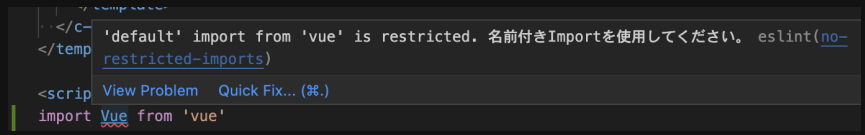
View Problem Quick Fix... (⌘.)

`import Vue from 'vue'`もチェックする

ESLint 本体に付属しているルールに、任意の`import`をチェックできるルールがあるので、
それを使用してチェックできます。

https://eslint.org/docs/latest/rules/no-restricted-imports

```
module.exports = {
  // ...
  rules: {
    // ...
    "no-restricted-imports": [
      "error",
      {
        paths: [
          {
            name: "vue",
            importNames: ["default"],
            message: "名前付きImportを使用してください。",
          },
        ],
      },
    ],
    // ...
  },
```



Vue3 の Breaking Changes を ESLint でチェックする

💚, 💙 : eslint-plugin-vue, ESLint でチェック可能 / 🔧 : 自動修正可能 / 💡 : エディタ上での提案

createApp()等	💙
名前付き Export	💙
`v-model`の`modelValue`, `emit('input')`	💚💡
`v-bind.sync`廃止	💚🔧
`<template v-for>`の key	💚
`v-if`・`v-for`の優先度	💚
`v-bind="object"`が順序に影響	❌
`v-on:event.native`廃止	💚
関数型コンポーネントの定義方法	❌
SFC の関数型コンポーネント廃止	💚
非同期コンポーネントの定義方法	❌
`emits`オプション	💚💡
`render`関数の変更	❌
`$scopedSlots`廃止	💚🔧
`$slots`の関数化	💚
`$listeners`廃止	💚
`$attrs`の変更	❌
カスタム要素の扱い	❌
`is`の記法	💚
`unmounted`	💚💡
`beforeUnmount`	💚💡

`default`の`this`	💚
カスタムディレクティブ定義方法	❌
`data`関数	💚🔧
mixin の`data`の扱い	❌
`:attr=false`	❌
Transition クラス名変更	❌
`<TransitionGroup>`のルート要素	❌
配列の watch	❌
ディレクティブ無しの`<template>`	💚
Vue アプリのルート	❌
`hook:`イベント	❌
`v-on`キー修飾子	💚🔧
`$on`・`$off`・`$once`	💚
フィルター	💚
Inline テンプレート	💚
`$children`	💙
`propsData`	❌
`$destroy()`	💙
`Vue.set()`・`Vue.delete()`	💙

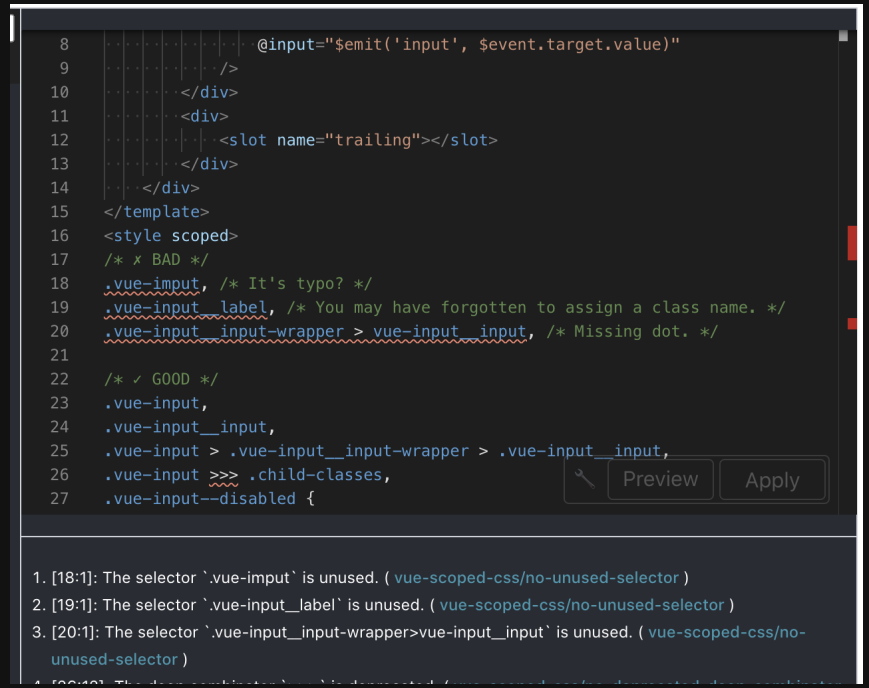
25 / 40 半分以上がチェック可能

まだチェックできます！

eslint-plugin-vue-scoped-css も導入する

SFC のスコープ付き CSS をチェックする ESLint プラグイン。
https://github.com/future-architect/eslint-plugin-vue-scoped-css

- 未使用セレクタをチェックできたりするプラグイン
- フューチャー（弊社）の OSS リポジトリ で管理
- 元々は社内で開発運用されていたチェックルールの一部を OSS 化したもの
- 23,000 ダウンロード / 週 程度には利用されている (2022 年 9 月の情報)



FUTURE

Vue3 の Breaking Changes を ESLint でチェックする

💚, 💙, 🧡: eslint-plugin-vue, ESLint, **eslint-plugin-vue-scoped-css** でチェック可能
🔧: 自動修正可能 / 💡: エディタ上での提案

createApp()等	💙
名前付き Export	💙
`v-model` の `modelValue`, `emit('input')`	💚💡
`v-bind.sync` 廃止	💚🔧
`<template v-for>` の key	💚
`v-if` ・ `v-for` の優先度	💚
`v-bind="object"` が順序に影響	❌
`v-on:event.native` 廃止	💚
関数型コンポーネントの定義方法	❌
SFC の関数型コンポーネント廃止	💚
非同期コンポーネントの定義方法	❌
`emits` オプション	💚💡
`render` 関数の変更	❌
`$scopedSlots` 廃止	💚🔧
`$slots` の関数化	💚
`$listeners` 廃止	💚
`$attrs` の変更	❌
カスタム要素の扱い	❌
`is` の記法	💚
`unmounted`	💚💡
`beforeUnmount`	💚💡

`default` の `this`	💚
カスタムディレクティブ定義方法	❌
`data` 関数	💚🔧
mixin の `data` の扱い	❌
`:attr=false`	❌
Transition クラス名変更	🧡
`<TransitionGroup>` のルート要素	❌
配列の watch	❌
ディレクティブ無しの `<template>`	💚
Vue アプリのルート	❌
`hook:` イベント	❌
`v-on` キー修飾子	💚🔧
`$on` ・ `$off` ・ `$once`	💚
フィルター	💚
Inline テンプレート	💚
`$children`	💙
`propsData`	❌
`$destroy()`	💙
`Vue.set()` ・ `Vue.delete()`	💙

26 / 40 がチェック可能

eslint-plugin-社内用 に追加チェックを実装

フューチャー（弊社）のプロジェクトで使用するためだけに社内で開発・運用されている ESLint プラグイン

- ここから、eslint-plugin-vue にチェックルールを輸出することもある
  - vue/no-undef-properties （未定義プロパティをチェックするルール），
    vue/no-unused-refs （未使用の`ref="..."`をチェックするルール）など

Vue3 の Breaking Changes を ESLint でチェックする

💚, 💙, 🧡, 💖: eslint-plugin-vue, ESLint, eslint-plugin-vue-scoped-css, **社内用プラグイン** でチェック可能
🔧: 自動修正可能 / 💡: エディタ上での提案 / 🚀: **社内のプロジェクトでは影響なし**

createApp()等	💙
名前付き Export	💙
`v-model` の `modelValue`, `emit('input')`	💚💡
`v-bind.sync` 廃止	💚🔧
`<template v-for>` の key	💚
`v-if` ・ `v-for` の優先度	💚
`v-bind="object"` が順序に影響	💖
`v-on:event.native` 廃止	💚
関数型コンポーネントの定義方法	🚀
SFC の関数型コンポーネント廃止	💚
非同期コンポーネントの定義方法	🚀
`emits` オプション	💚💡
`render` 関数の変更	💖
`$scopedSlots` 廃止	💚🔧
`$slots` の関数化	💚
`$listeners` 廃止	💚
`$attrs` の変更	❌
カスタム要素の扱い	🚀
`is` の記法	💚
`unmounted`	💚💡
`beforeUnmount`	💚💡

`default` の `this`	💚
カスタムディレクティブ定義方法	❌
`data` 関数	💚🔧
mixin の `data` の扱い	🚀
`:attr=false`	🚀
Transition クラス名変更	🧡
`<TransitionGroup>` のルート要素	❌
配列の watch	❌
ディレクティブ無しの `<template>`	💚
Vue アプリのルート	❌
`hook:` イベント	❌
`v-on` キー修飾子	💚🔧
`$on` ・ `$off` ・ `$once`	💚
フィルター	💚💖🔧
Inline テンプレート	💚
`$children`	💙
`propsData`	🚀
`$destroy()`	💙
`Vue.set()` ・ `Vue.delete()`	💙

のこり 6 項目

フィルター廃止に伴う対応について

フィルター廃止に伴って、社内フレームワークでは関数呼び出しに置き換える方針としました。
しかし、eslint-plugin-vue でもフィルターの使用箇所を検出することができますが*自動的に修正することはできません*。
フィルターはかなり多くの箇所で使用していたため、**社内用 ESLint プラグインに自動修正を持つルールを追加**して対応しました。

```
module.exports = {
  // ...
  rules: {
    // ...
    "internal/no-vue-filter": [
      "error",
      {
        /* フィルター名: 修正後の関数情報 */
        formatNum: { method: "$util.formatNum" },
        className: {
          method: "$util.getClassName",
          argumentIndex: 1,
        },
        // ...
      },
    ],
    // ...
```

Before:

```
<template>
  <span>{{ value | formatNum }}</span>
  <span>{{ kindValue | className('KIND') }}</span>
</template>
```

After:

```
<template>
  <span>{{ $util.formatNum(value) }}</span>
  <span>{{ $util.getClassName('KIND', kindValue) }}</span>
</template>
```

残り 6 項目はどうした?

残りはどうした?

残りの 6 項目は以下のように対応しました。
どれもすぐに見つけられすぐ修正できたので、ESLint ルールを使用しませんでした。
（今後追加するかもしれません。）

- `$attrs`の変更
  … 全て`$listeners`と併用していたため`$listeners`廃止の対応をして作業終了。
- カスタムディレクティブ定義方法
  … カスタムディレクティブはファイル構成としてまとめていたので見つけるための工夫は不要。修正して対応。
- `<TransitionGroup>`のルート要素
  … むしろ変更後の動作を好んだため動作確認をして作業終了。
- 配列の watch
  … 一箇所配列に対する watch を発見。そもそも watch はあまり使用していなかった。
- Vue アプリのルート
  … 各アプリで 1 箇所修正。
- `hook:`イベント
  … 全て`$once`と併用していたため`$on`・`$off`・`$once`の対応をして作業終了。

Vue3 以外の Breaking Changes を ESLint でチェックする

Vue3 をサポートするライブラリへのアップグレードに伴う Breaking Changes や、Vue3 以前に非推奨にはなっていて、Vue3 で削除された機能等も ESLint でチェックできる場合があります。

💚, 🧡, 🌐: eslint-plugin-vue, eslint-plugin-vue-scoped-css, @intlify/eslint-plugin-vue-i18n でチェック可能
🔧: 自動修正可能

`scope="..."`の廃止(Vue.js 2.5.0+)、`v-slot`を使用	💚🔧
`slot="..."`の廃止(Vue.js 2.6.0+)、`v-slot`を使用	💚🔧
`slot-scope="..."`の廃止(Vue.js 2.6.0+)、`v-slot`を使用	💚🔧
`slot-scope="..."`の廃止(Vue.js 2.6.0+)、`v-slot`を使用	💚🔧
古い Deep セレクター`>>>`の代わりに`::v-deep`を使用	🧡🔧
`::v-deep`の代わりに、`:deep()`または`::v-deep()`を使用	🧡🔧
vue-i18n の`<i18n>`コンポーネントの代わりに`<i18n-t>`を使用	🌐🔧
vue-i18n の`place`属性廃止	🌐
vue-i18n の`place`Prop 廃止	🌐
vue-i18n のメッセージ構文の変更	🌐

Vue3 移行に
ESLint を使うメリット

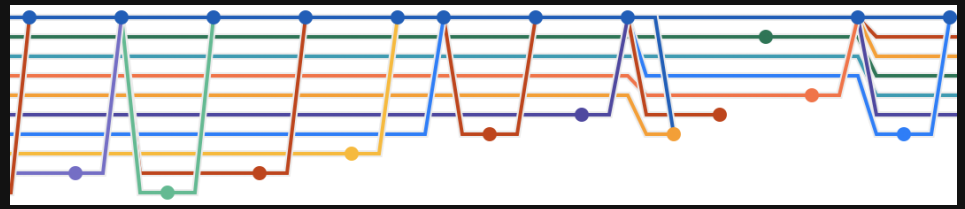
Vue3 作業に ESLint を使うメリット

- **自動化**
  - 移行が必要な箇所を自動でチェック・修正できる
  - 移行が必要な箇所をエディタ上で確認できる
- **継続的に使える**
  - Shareable Config を作成しておけば、別プロジェクトでもすぐに使える
  - *移行後に新しく Vue2 でしか動かないコードを書かれない*

移行後に新しく Vue2 でしか動かないコードを書かれない

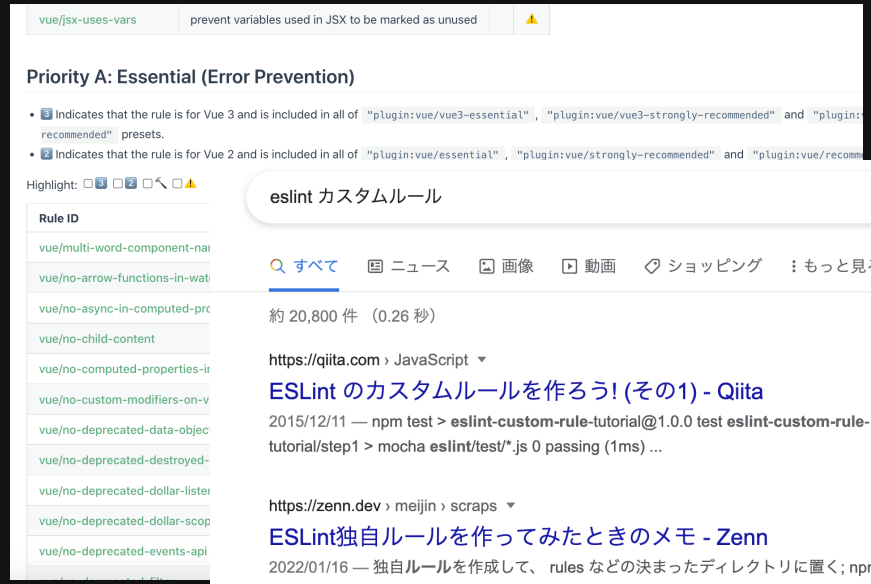
例えば、

- チームメンバー
  - Vue3 移行後に Vue2 でこれまで開発してきたメンバーがうっかりフィルタを使おうとしてビット論理和をしてしまう、などを防げる
  - Vue 初心者な新規メンバーが、ネット記事を参考に書いた Vue2 でしか動かないコードをそのまま使ってしまう、などを防げる
- 共通ライブラリの 2 バージョン管理
  - 共通ライブラリの Vue2 用バージョンで追加した機能を Vue3 用にマージするとき自動でチェック・修正される



Vue3 移行に ESLint を使うデメリット

- プラグインが存在しない場合はカスタムルールを自作しないといけないかもしれない
- そんなに簡単に（社内用）ESLint プラグイン・カスタムルール作れない（？）
  - 公式ドキュメントを読んでみる
  - eslint-plugin-vue の 200 を超えるルールの実装を参考にしてみる
  - 「eslint カスタムルール」で検索してみる
  - ESLint の Discord の Japanese チャンネルで聞いてみる
- ESLint は移行ツールではないので、移行ツールに比べれば弱い点もある



まとめ

まとめ

- Vue3 移行は**一回の作業で終わりではありません**。
  - 作ってるアプリケーションの数だけ移行作業が必要
  - Vue2 用と Vue3 用の両バージョンをサポートする共通ライブラリのマージ管理
  - 開発メンバーへの Vue2 ↔ Vue3 の違いの周知

継続的に移行するためのツールが必要です!

- ESLint を使用して移行が必要な箇所を自動でチェック・修正できます。
  - OSS のプラグインで Vue3 の Breaking Changes の **65%**(26/40) 程度の項目はカバーできる
    （弊社の場合、影響なしの項目を除けば 76%(26/34))
    - 移行作業コストの感覚値では 95%ぐらいは自動でチェックできていると思います
  - ローカルルールを決めて個別に特化したカスタムルールを実装すれば、
    ほとんど自動で移行できる可能性がある

ESLint と eslint-plugin-vue で継続的な Vue3 移行を実現しましょう

宣伝

プラチナスポンサーしてます。

https://vuefes.jp/2022/sponsors#future

詳しくはスポンサーセッションをご覧ください。

エンジニア・スペシャリスト
募集してます
↓
https://www.future.co.jp/recruit/

Thank you for your attention

Support me ❤ or follow me!!

GitHub: https://github.com/ota-meshi

Twitter: https://twitter.com/omoteota

Qiita: https://qiita.com/ota-meshi