

eslint-plugin-vue の現状と今後

Vue Fes Japan 2023 - 2023.10.28 SAT

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](#)  , npm: [ota-meshi](#), Twitter: [@omoteota](#), Qiita: [@ota-meshi](#)
- 所属:
 - [フューチャー株式会社](#) (ゴールドスポンサー!) 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 -)
 - [Stylus](#) チーム (2022/06 -)
 - [ESLint Community](#) チーム (2022/09 -)
 - [Svelte](#) チーム (2023/05 -)
 - [ESLint](#) チーム (2023/08 -)
- WEB+DB PRESS Vol.120 「最新 Vue.js 3 入門」 共同執筆 (2020/12/24)



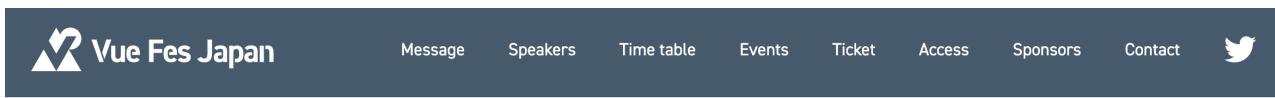


F U T U R E

宣传

ゴールドスポンサーです。

<https://vuefes.jp/2023/sponsors/future-architect>



The header navigation bar for Vue Fes Japan 2023. It includes the logo, navigation links (Message, Speakers, Time table, Events, Ticket, Access, Sponsors, Contact), and social media icons for Twitter and GitHub.

Sponsors

——— スポンサー ———

Gold



フューチャーでは、各分野に精通するエンジニアが多数在籍しコミッターとしても活躍しています。エンジニアが実装のみならず業務改革などのコンサルティングも行い、様々な業界のお客様の「経営と IT」を支援しています。現在も社会にインパクトのあるプロジェクトを数多く手掛けており、エンジニアを募集中です！Vue.jsは多くのプロジェクトで活用しており、今後もコミュニティへの貢献を通じて社会の発展に寄与します。

スポンサーブースにもお立ち寄りください！1Fの廊下でやってます。

と言いたいところですが、この話が終わったらもう終わりらしいです😭

エンジニア・スペシャリスト
募集します



<https://www.future.co.jp/recruit/>

アジェンダ

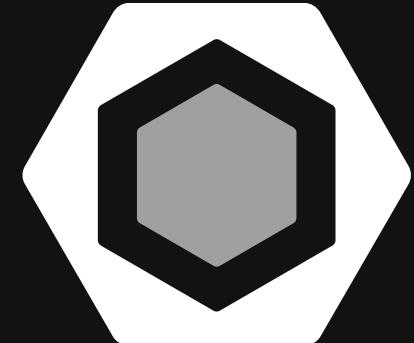
1. eslint-plugin-vue とは
2. eslint-plugin-vue の変遷
3. Plans for version 10
4. 将来的な機能
5. 現状と今後

ESLint

ESLint とは

JavaScriptのリンター

- JavaScriptの問題・スタイル違反を見つけるツール
- 自動的に修正可能
- 全ての検証ルールはON・OFF可能、レポート内容をオプションで変更可能
- プラグインによって検証ルールを追加可能、JavaScript以外もチェック可能
- 私は *ESLint* のコミッターです



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!)

- `*.vue` ファイルの解析を可能にし、Vue に特化した問題を見つける検証ルールを提供
- 230もの追加ルールがあります
- 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](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](https://docs.google.com/presentation/d/18Q8nn69Hi8d39k51HduArKkrCx3CY_ZsbPf-F-tb8Pg/edit?usp=sharing)
 - [Vue Fes Japan Online 2022 での発表資料](#)
<https://ota-meshi.github.io/vue-fes-japan-online-2022-slide/>
- 私は `eslint-plugin-vue` のメンテナーです

eslint-plugin-vue の変遷

eslint-plugin-vue の変遷 と関連パッケージの変遷

2013

- ESLint登場

2014

- Vue.js登場

2015

- ESLint v1.0.0
- Vue.js v1.0.0

2016

- Vue.js v2.0.0

- eslint-plugin-vue v1.0.0

当時は eslint-plugin-html に依存していた。

<template> ブロックのチェックはできなかった。

仕組み的には現在のものとは全くの別物。

2017

- eslint-plugin-vue v3.0.0 

カスタムパーサーを使用した全く新しい仕組みで実装された。

<template> ブロックのチェックが可能に!

カスタムパーサーを使用(2017)

vue-eslint-parser の登場

*.vue ファイルをパースして ESLint と互換性のある AST を生成する ESLint カスタムパーサー

vue-eslint-parser が開発された。

当時 ESLint カスタムパーサーは TypeScript 用と Babel を使用したものぐらいしか存在していなかった中での Vue 専用カスタムパーサーの登場であったため、個人的にとても注目した 😊

これによって、

- (現在の) eslint-plugin-vue が開発できるようになった
- eslint-plugin-vuejs-accessibility や eslint-plugin-vue-i18n などの何かに特化したチェックを行うプラグインが登場する基礎にもなった

eslint-plugin-vue の変遷 と関連パッケージの変遷

2018

- *Prettier v1.10*

.vue ファイルの
部分的サポート開始!

- ota-meshi (私) が
コラボレーターとして
eslint-plugin-vue に参加

- *eslint-plugin-vue v4.0.0*

Vueスタイルガイドをフォローする
ための多くのルールが追加された!

- Vue Fes Japan 2018 開催
- Vue 3 Plan発表
- eslint-plugin-vue-a11y 登場

フォーマッターとしての eslint-plugin-vue (2018)

Vueのスタイルガイド

Vueには実は公式の[スタイルガイド](#)が存在していて（現在は隠れていて公式トップからは辿り着けません 😊）それに従っているかどうかをチェックするルール多くのルールが追加されました。

eslint-plugin-vue は `*.vue` ファイルのフォーマットをフルサポート

`<template>` ブロックも処理するインデントのルールなどのフォーマットに関連する多くのルールも追加されました。

現在ほとんどのユーザーはPrettierでフォーマットしていると思いますが、当時はPrettierがまだ `*.vue` をフルサポートしていなかったため、`prettyhtml` のようなHTMLのフォーマッターを使用するか、`eslint-plugin-vue` を使用するか、部分的サポートであってもPrettierを使用するかの選択肢があったと思います。

（HTMLのフォーマッターを使用する場合、ディレクティブ内のスクリプトがフォーマットされないなどの問題もありました）



eslint-plugin-vue の変遷 と関連パッケージの変遷

2019

- `@typescript-eslint` 登場!
- `eslint-plugin-vue v5.0.0`
 - Vue v2.6のサポート
 - `v-slot` への移行を助けるルールが追加された
 - 一部のルールがNuxtをサポート
- `Vue.js v2.6`
 - `v-slot` の導入など
- `eslint-plugin-vue-i18n` 登場

2020

- `Vue.js v3.0.0`
- `eslint-plugin-vue v7.0.0`
 - Vue v3.0のサポート
 - Vue v3 移行を助ける多くのルールが追加された
 - 66個もの新しいルールが追加された
- `vue-eslint-parser v7.3.0`
 - `eslint-plugin-jsonc`、`eslint-plugin-yml` などを使用したカスタムブロックのチェックをサポート

移行ツールとしての eslint-plugin-vue (2019 - 2020)

eslint-plugin-vue は廃止された機能が使用されていないかどうかをチェックするルールを多く持っています。またチェックルールによっては自動修正や提案機能を使用して、自動的に新しい書き方に置き換えてくれます。

これによって、[Vueのバージョンアップをサポートする移行ツール](#)としての側面も持つようになりました。

前回のVueFesでの発表内容もチェックしてね!

v2 → v3

また eslint-plugin-vue-i18n では vue-i18n の移行を助けるルールが実装され、さらにその後 eslint-plugin-vuetify というプラグインも登場しこれはVuetifyの移行を助けてくれるそうです。

eslint-plugin-vue の変遷 と関連パッケージの変遷

2021

- *Vue.js v3.1.0*
- *eslint-plugin-vue v7.11.0*
Vue v3.1のサポート
- *Vue.js v3.2.0*
- *eslint-plugin-vue v7.16.0*
Vue v3.2のサポート

2022

- *eslint-plugin-vuejs-accessibility v1.0.0*
(*eslint-plugin-vue-a11y*の後継)
- *eslint-plugin-vue v8.0.0*
Vue3用Preset構成の大幅な改善
Vue3.0->Vue3.2での変化に追従など
さらなるNuxtのサポート
- *FloEdelmann* がメンテナーとして *eslint-plugin-vue* に参加🎉
- *eslint-plugin-vue v9.0.0*
Vue2用Preset構成の大幅な改善
Vue2でも `<script setup>` を使用する人たちを考慮した構成など
- *Vue.js v2.7.0*
Vue2でも `<script setup>` が公式に使用可能になった

`eslint-plugin-vue` の変遷 と関連パッケージの変遷

2023

- `Vue.js v3.3.0` そして...
- `eslint-plugin-vue v9.13.0`

Vue v3.3のサポート

eslint-plugin-vue v10 計画

Nuxt サポートの強化

eslint-plugin-vue v10 計画

Nuxt サポート

もう一人のメンテナである [FloEdelmann](#) と話し合った結果、Nuxt を正式にサポートしていくことに決めた。



手伝ってくれる人歓迎です ❤️

eslint-plugin-vue の 将来的な機能構想

TypeScriptの 型情報を使用したチェック

TypeScriptの型情報を使用したチェック

現在は、eslint-plugin-vue と @typescript-eslint の型情報を使用したルールとの組み合わせで弱い。特に eslint-plugin-vue の使用しているパーサー vue-eslint-parser は `<template>` ブロックの型情報を捨ててしまっているため、`<template>` ブロックでは型情報を使用したルールは使用できないし、型情報を使用した独自のルールも実装できない。

TypeScriptの型情報を使用したチェック

例えば、

```
<script setup lang="ts">
let str: string = "";
</script>
<template>
  <MyInput @update:model-value="(e) => (str = 'Value is ' + e)" />
</template>
```

の感覚で、

```
<template>
  <input @input="(e) => (str = 'Value is ' + e)" />
</template>
```

と書いてしまっても、TypeScriptではエラーにならないが、型情報を使用したESLintカスタムルールを作ることができればチェックできるはず。
しかし型情報を使用したルールを実装できないので現在は不可能。

Why?

現在、vue-eslint-parser は `<script>` ブロックの `<template>` ブロック内のディレクティブのスクリプトは、別々にパースした結果を組み合わせてASTを構築している。

例えば：

```
<script setup lang="ts">
let str: string = "";
</script>
<template>
  <input @input="(e) => (str = 'Value is ' + e)" />
</template>
```

は、`let str: string = "";` と `@input="(e) => (str = "Value is " + e))";` のスクリプトを別々にパースしている。

結果として `<template>` ブロック内のディレクティブのスクリプトは型情報をとったところで意味のない情報となってしまっている。
(つまりそもそも使える型情報になっていないので捨ててしまっている)

どうすれば解決できる?

<script> ブロックの <template> ブロック内のディレクティブのスクリプトをTypeScriptが理解できる形でくっつけてパースした結果を、構築し直してASTを作ればいい。

例えば、

```
<script setup lang="ts">
let str: string = "";
</script>
<template>
  <input @input="(e) => (str = 'Value is ' + e)" />
</template>
```

は、

```
let str: string = "";

function __render() {
  ((e) => (str = "Value is " + e)) as (
    e: "input" extends infer EVT
      ? EVT extends keyof HTMLElementEventMap
        ? HTMLElementEventMap[EVT]
        : CustomEvent<any>
      : never,
  ) => void;
}
```

で、パースしてその結果からASTを構築すれば、<template> ブロック内のディレクティブも正しい型情報が得られる。

やってみた

実際にこのアイディアをやってみた。

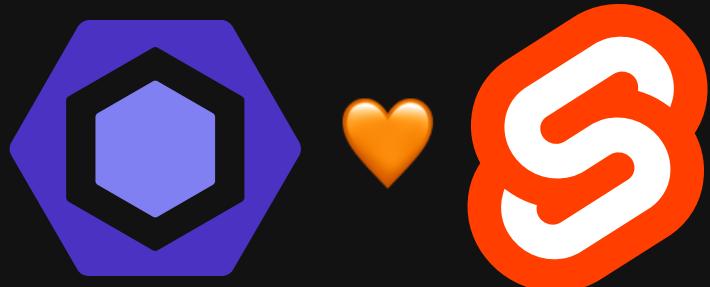
[eslint-plugin-svelte](#) で。

Playground

```
> src/example.svelte
1 <script lang="ts">
2   export let str: string = "";
3 </script>
4
5 <input on:input={(e) => (str = 'Value is ' + e)} />
6
```

Invalid operand for a '+' operation. Operands must each be a number or string, allowing a string + any `RegExp`, `undefined`. Got `Event & { currentTarget: EventTarget & HTMLInputElement; }`. ESLint(@typescript-eslint/operands)
'e' will evaluate to '[object Object]' when stringified. ESLint(@typescript-eslint/no-base-to-string)
[View Problem \(CF8\)](#) No quick fixes available

うまく動いていそう。



TypeScriptの型情報を使用したチェック

次の目標はこのアイディアを [eslint-plugin-vue](#) に取り込むこと。

(またその他にも試したいいろいろなアイディアがあるので、
うまくいったものは eslint-plugin-vue に取り込む)

ただし...

- eslint-plugin-vue の大きな破壊的な変更を伴う
 - おそらくほぼ全てのチェックルールは書き直しになると思います 😅
- 時間がかかりそう
 - 過去のしがらみのない eslint-plugin-svelte でこれを実現するのに 2 年以上かかるってる
 - (私はフルタイムで OSS やってるわけではないのでとても時間がかかります)

ESLintの完全な書き直し計画

Complete rewrite of ESLint

<https://github.com/eslint/eslint/discussions/16557>

Complete rewrite of ESLint #16557

 Closed

 Locked

Unanswered

nzakas asked this question in Ideas



nzakas [on Nov 1, 2022](#)

Maintainer

edited

...

Introduction

ESLint was first released in 2013, meaning it will be ten years old next year. During that time, the way people write JavaScript has changed dramatically and we have been using the incremental approach to updating ESLint. This has served us well, as we've been able to keep up with changes fairly quickly while building off the same basic core as in 2013. However, I don't believe continually to make incremental changes will get us to where ESLint needs to go if it wants to be around in another ten years.

Even though we are close to rolling out the new config system, which is the first significant rearchitecture we've done, that effort is what led me to believe that it's time for a larger rewrite. We are still stuck on implementing things like async parsers and rules because it's difficult to plot a path forward that doesn't cause a lot of pain for a lot of users. This seems like the right time to stop and take stock of where we are and where we want to go.

Goals

ESLintの完全な書き直し計画

この中で、**Make ESLint project-aware** というトピックがあり、これはファイルをLintする前にプロジェクト全体の情報を収集してその情報をファイル単位のLintに活かそうというもの。

These are some ideas that aren't fully hatched in my mind and I'm not sure how we might go about implementing them or even if they are good ideas, but they are worth exploring.

- **Make ESLint type-aware.** This seems to be something we keep banging our heads against -- we just don't have any way of knowing what type of value a variable contains. If we knew that, we'd be able to catch a lot more errors. Maybe we could find a way to consume TypeScript data for this?
- **Make ESLint project-aware.** More and more we are seeing people wanting to have some insights into the surrounding project and not just individual files. `typescript-eslint` and `eslint-plugin-import` both work on more than one file to get a complete picture of the project. Figuring out how this might work in the core seems worthwhile to explore.
- **Standalone ESLint executable.** With Rust's ability to call into JavaScript, it might be worth exploring whether or not we could create a standalone ESLint executable that can be distributed without the need to install a separate runtime. Deno also has the ability to compile JavaScript into a standalone executable, so Rust isn't even required to try this.

ESLintの完全な書き直し計画

プロジェクトの情報をを集められるということは、例えば、

- 登録されたコンポーネント情報を収集して、その情報を使用したチェック
- Vue Router で登録しているルーター情報を収集して、その情報を使用したチェック
- vue-i18n で登録しているメッセージ情報を収集して、その情報を使用したチェック
- グローバルに宣言されたCSSセレクタ情報を収集して、その情報を使用したチェック

など、さまざまなことがESLintのエンジンを使用してできるようになるかもしれません。

現状と今後 まとめ

現状と今後まとめ

- eslint-plugin-vue は Vue を強力にサポートし続けてきました
 - 強力なlintingツールとして
 - フォーマッターとして
 - マイグレーションツールとして
- eslint-plugin-vue は今後も**より強力に発展**していきます
 - **Nuxt**のサポートの拡充
 - **型情報を活用したより強力なチェック**を可能にする
 - (ESLintエンジン自体の新機能を活用して) **プロジェクト情報の事前収集などによる強力なチェック**を可能にする

そしてもちろん
Vue v3.4のサポートにも
取り組みます ❤

eslint-plugin-vueは
Vueを強力にサポートする
ツールとしてこれからも
発展していきます
応援よろしくお願ひします ❤

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>