

# FrontEnd Engineer 集会

2024 04

# 4月のトピック

~ 2024 / 4 / 16

# Storybook

2024 / 03 / 08



# Storybook は 何?

3行で

- 実 DOM を使うコンポーネントドキュメンテーションツールだよ
- React, Vue, 他ほとんど js フレームワークでもサポートしてるよ
- プラグイン入れればテストもできるよ

# 何があったか?

Storybook v8 がリリースされた

レンダリングエンジンに React が使われていたが、これに依存しなくなった

# Reactへの依存解消

Storybook を導入すると、それだけで React が node\_modules にバンドルされる問題

- TypeScript を使ってると TSX 使用時に React がデフォルトで使われてしまう
- vue と react が同じ node\_modules にある気持ち悪さ
- 単純なレンダリングパフォーマンスが悪い

glup

2024 / 03 / 29

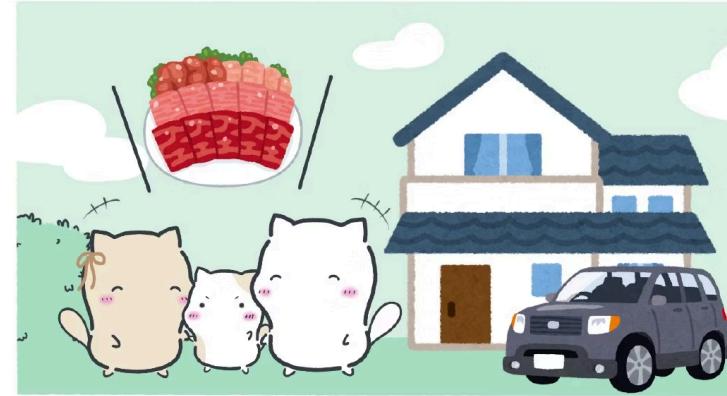


# glup is 何?

3行でまとめ

- webpack が出る前に使われていたツールチェイン  
(対抗: grunt)
- 昔は、TS をビルドして、SCSS をビルドして、バン  
ドルして、ミニファイして・・  
を grupfile.js に書いてたんだよ
- wordpress とか使ってる現場だと現役かも？

理想



現実



# お前も・・生きてたんだ

v5 がリリースされた

前回の v4.0.2 から 5年ぶりのアップデート

# bun

2024 / 04 / 01



# bun は 何?

3行でまとめ

- node.js, deno に並ぶ サーバサイド js ランタイム
- 兼 npm, yarn に並ぶパッケージマネージャー
- パッケージインストールが気持ち悪いほど早い

# 何があったか?

bun v1.1 リリースで、Windows での動作がサポートされた!

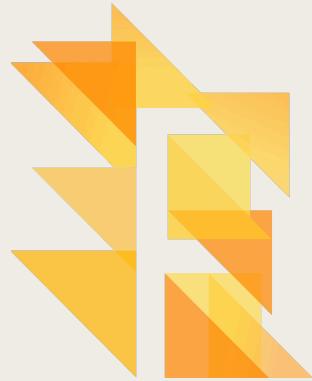
powershell で動くよ! やったね!



```
powershell -c "irm bun.sh/install.ps1 | iex"
```

# flow

2024 / 04 / 04



# flow is 何?

3行でまとめ

- CoffeeScript / TypeScript の仲間、AltJS だよ
- TypeScript と同じく、型安全性を謳っていたけど、完全に負けたよ
- facebook 製で、React のソースコードも flow で書かれてるよ
  - `@types/react` が必要なのはこのせい
  - 他で使われているところを見たことがないので、実質 React 開発用言語だよ

# 何があったか?

- flow v0.233.0 にて、新構文 `component` が追加された

# これまでの flow での component の書き方

function を使う感じ

typescript に似てるので何となく分かる・・

今まで

```
type Props = $ReadOnly<{
  text?: string,
  onClick: () => void,
}>;
export default function HelloWorld({
  text = 'Hello!',
  onClick,
}: Props): React.MixedElement {
  return <div onClick={onClick}>{text}</div>;
}
```

# component syntax での書き方

function の代わりに component を使う

component syntax

```
export default component HelloWorld(  
  text: string = 'Hello!',  
  onClick: () => void,  
) {  
  return <div onClick={onClick}>{text}</div>;  
}
```

# component syntax のメリット

- 引数 (props) が readonly になる
- useRef, useState などを間違った使い方したとき

ランタイムエラーをコンパイル時に検知できる



```
component MyComponent() {
  const renderCount = useRef<number>(0);
  renderCount.current += 1; // error
  return <div>{renderCount.current}</div> // error
}

component MyComponent() {
  const renderCount = useRef<number>(0);
  const [count, setCount] = useState(0);
  useEffect(() => {
    renderCount.current += 1; // ok
    setCount(renderCount.current); // ok
  }, [renderCount]);
  return <div>{count}</div>
}
```

おわり