

Reactのいいなと思ったところ

Natsuki

# 今日話すこと

普段の業務ではVueを使用していて、最近Reactも業務外で勉強し始めました  
初心者の目線で「Reactのココいいな」と思った点を共有したいと思います

## 自己紹介

Natsuki


普段はLaravel + Vueで開発




# 宣伝

Reactを学ぶ上で、VSCodeの拡張機能を作ってみました。


## Your Themes

 Visual Studio | Marketplace

Sign in 

Visual Studio Code > Visualization > Your Themes

New to Visual Studio Code? [Get it now.](#)



### Your Themes

Natsuki | 39 installs | ★★★★★ (0) | Free

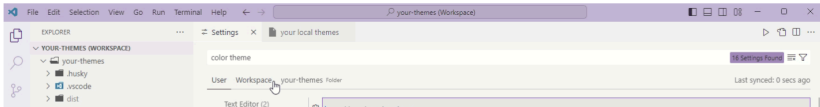
Previews of your local vscode themes

[Install](#) [Trouble Installing?](#)

[Overview](#) [Version History](#) [Q & A](#) [Rating & Review](#)

### Your Themes

This extension shows color theme previews you installed in VSCode.



### Categories

Visualization

### Works with

Universal

### Resources

[Issues](#)

demo

<https://github.com/natsuki-engr/your-themes/blob/a1ce518a99009fd5baf489e592958c3c35b5dfcf/img/demo.gif>

# 本題

「Reactのいいなと思ったところ」

# dangerouslySetInnerHTML

```
const markup = '<p>some raw html</p>';  
return <div dangerouslySetInnerHTML={{__html: markup}} />;
```

タグを含んだ文字列をHTMLタグとしてレンダリングしたい場合の`setInnerHTML`と同じ

XSS攻撃の危険性があるので、注意が必要

「危険なんだぞ」という強い警告を感じるところがいい

vueだと`v-html`で同じ実装ができるが、罪悪感が無い

```
<script setup>  
const markup = '<p>some raw html</p>'  
</script>
```

一貫してJS

「JSを知っていれば大体書ける」

たとえばJSX

# JSX

- JSXの構成はJSとタグなので、テンプレートのために必要な知識が少ない
- JSのパラダイム/設計を持ち込める

```
function ItemList({ items: Item[] }) {  
  if(items.length === 0) {  
    return <p>empty...</p>  
  }  
  
  return (  
    <ul>  
      {items.map(item => (  
        <li key={item.id}>{item.name}</li>  
      ))}  
    </ul>  
  );  
}
```



# レンダリングの更新が分かりやすい

基本的に、引数やステートが変わって再レンダリングされるときに  
「全て再計算される」と思っておいて、

- ・ 特定の再計算をスキップしたければ ``useMemo``

配列から値の検索とか

- ・ 特定の処理をスキップしたければ ``useEffect``

APIの呼び出しとか

# useEffect/useMemo の悪い例

## 不要な `useEffect`

```
function Form() {  
  const [firstName, setFirstName] = useState('Taylor');  
  const [lastName, setLastName] = useState('Swift');  
  
  const [fullName, setFullName] = useState('');  
  useEffect(() => {  
    setFullName(firstName + ' ' + lastName);  
  }, [firstName, lastName]);  
  
  return /** ... */  
}
```


## 不要な `useMemo`

```
function Form() {  
  const [firstName, setFirstName] = useState("Taylor");  
  const [lastName, setLastName] = useState("Swift");  
  
  const fullName = useMemo(  
    () => firstName + " " + lastName,  
    [firstName, lastName]  
  );  
  
  return; /** ... */  
}
```

```
const fullName = firstName + " " + lastName
```

<https://react.dev/learn/you-might-not-need-an-effect#updating-state-based-on-props-or-state>

# なぜ今日この話をしようと思ったか

 **jordwalke** ✓  
@jordwalke

**templating syntaxes** →

```
.jsx  
(condition ? (  
  <div>Content</div>  
) : otherCondition ? (  
  <div>Other Content</div>  
) : (  
  <div>Fallback Content</div>  
))
```

**mental illnesses** →

```
.svelte  
{#if condition}  
  <div>Content</div>  
{:else if otherCondition}  
  <div>Other Content</div>  
{:else}  
  <div>Fallback Content</div>  
{/if}
```

```
.vue  
<div v-if="condition">  
  Content  
</div>  
<div v-else-if="otherCondition">  
  Other Content  
</div>  
<div v-else>  
  Fallback Content  
</div>
```

# 参考

【React】誤解される useMemo と 誤用される useState — 「A の変更に対応して B の値が変わる」と考えるべきでない

<https://qiita.com/honey32/items/58e56e407d4d87e294a4>

【React】 useEffect の標準動作は「依存配列の中身が変わると実行」ではない

<https://qiita.com/honey32/items/62edf5165aced7d0c4bf>

useState + useEffect -> 再レンダリング中に毎回計算 (重ければ useMemo)

[https://scrapbox.io/honey32/useState\\_+\\_useEffect\\_->\\_再レンダリング中に毎回計算\\_\(重ければ\\_useMemo\)](https://scrapbox.io/honey32/useState_+_useEffect_->_再レンダリング中に毎回計算_(重ければ_useMemo))

props または state に基づいて state を更新する - そのエフェクトは不要かも

<https://ja.react.dev/learn/you-might-not-need-an-effect#updating-state-based-on-props-or-state>