




React Components

Prof. Guilherme Alves

Ambiente de desenvolvimento






Auto Rename Tag

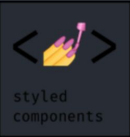
v0.1.10

Jun Han | 9,810,344 | ★★★★★ (156)

Auto rename paired HTML/XML tag


[Disable](#) [Uninstall](#) 

This extension is enabled globally.




vscode-styled-components


v1.7.4

 Styled Components | 272,318 | ★★★★★ (9)

Syntax highlighting for styled-components

[Disable](#) [Uninstall](#) 

This extension is enabled globally.




Live Server


v5.7.5

Ritwick Dey | 23,707,100 | ★★★★★ (375)

Launch a development local Server with live reload feature for static & dynamic pages

[Disable](#) [Uninstall](#) 

This extension is enabled globally.




cdnjs

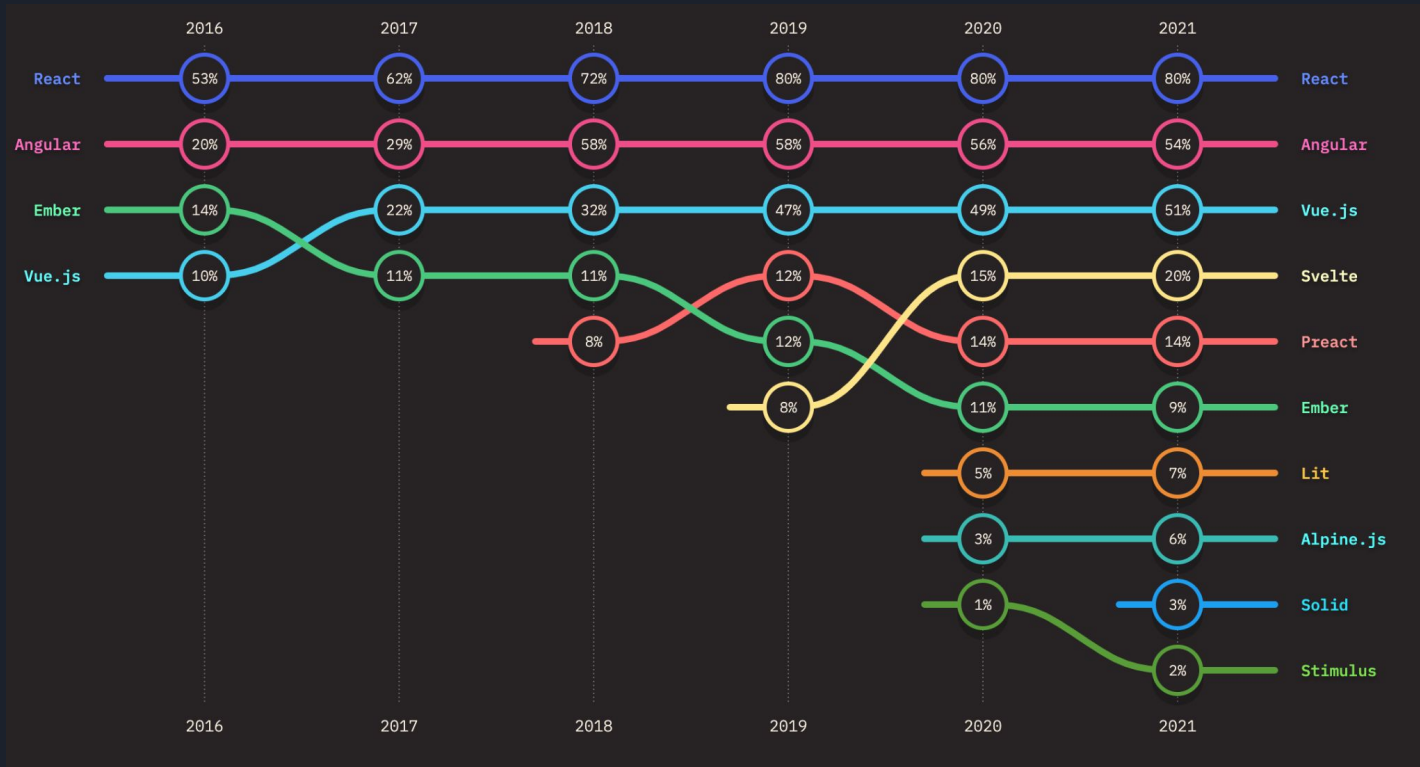
v0.19.0

Jake Wilson | 63,523 | ★★★★★ (2)

Search and embed libraries from cdnjs.com in Visual Studio Code

[Disable](#) [Uninstall](#) 

This extension is enabled globally.

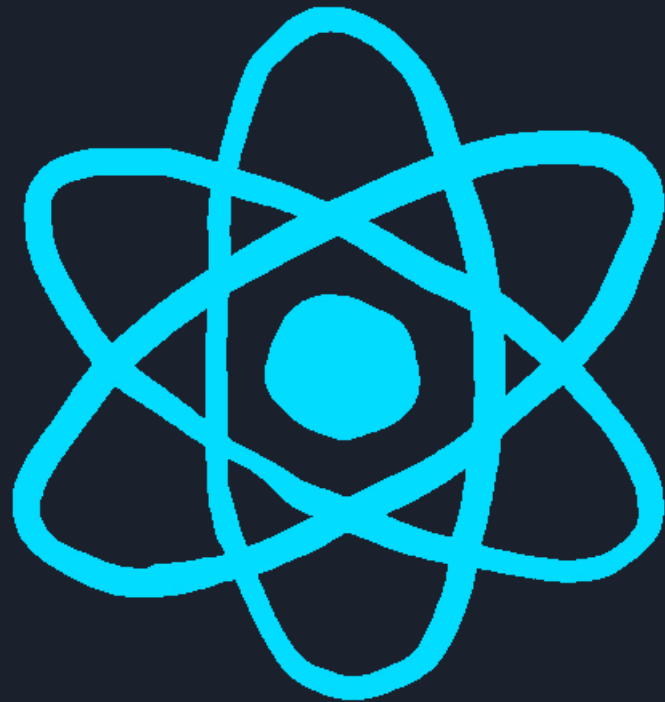


<https://2021.stateofjs.com/en-US/libraries/front-end-frameworks>




Documentação

<https://reactis.org/>




JSX

O React utiliza o JSX como sintaxe para criação de UI, sendo uma ferramenta muito poderosa e prática para escrever elementos em JavaScript. Ao contrário do que parece o JSX não é HTML e sim JavaScript.



```
1  return (  
2      <div>  
3          <h1>Título</h1>  
4          <p>Parágrafo</p>  
5      </div>  
6  )
```



```
1  return React.createElement("div", null,  
2      React.createElement("h1", null, "Título"),  
3      React.createElement("p", null, "Parágrafo"));  
4
```



Componentes

Os componentes no React são divididos em duas categorias, sendo elas stateless que são componentes sem estado e stateful que são componentes que possuem estado. Existem duas formas de criar um componente, sendo funcionais, que são componentes baseados em funções e componentes baseados em classes.

Um dos principais diferenciais do React é a facilidade em componetizar e reutilizar componentes.

Componentes baseados em classe

Os componentes de classe por padrão possuem manipulação de estado e métodos para gerenciamento do ciclo de vida que podem ser implementados. Estes componentes não são mais recomendados pela documentação, mas ainda têm uma adoção abrangente. Todo componente de classe deve estender de classes disponibilizadas pela biblioteca.

```
1  class Component extends React.Component {  
2      state = {};  
3  
4      render() {  
5          const { state, props } = this;  
6  
7          return (<dic></dic>);  
8      }  
9  }
```

Componentes baseados em classe

Os componentes baseados em classe podem estender também da classe **PureComponent** do react, tendo como diferença a implementação do método **shouldComponentUpdate** e a comparação das propriedades é realizado de maneira superficial utilizando o modo **shallow**. Isto significa que em uma comparação de não objetos o conteúdo é utilizado como item para comparação, quando você compara objetos apenas a referência é verificada.



```
1 class Component extends React.PureComponent {  
2   state = {};  
3  
4   render() {  
5     const { state, props } = this;  
6  
7     return (<dic></dic>);  
8   }  
9 }
```




Componentes baseados em classe ciclo de vida

<https://projects.woitekmai.pl/react-lifecycle-methods-diagram/>

<https://reactjs.org/docs/react-component.html>



Componentes funcionais

A criação de componentes funcionais é indicada pela documentação como tipo recomendado, por convenção utiliza-se arrow functions que recebem um único objeto como argumento. Por definição este tipo de componente não possui manipulação de estado. Com o surgimento e utilização dos hooks este tipo de componente passou a possibilitar o controle de estado e ciclo de vida.



```
1  const Component (props) => (<div></div>);
```