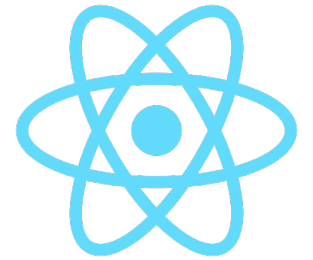


React - Uma biblioteca JavaScript para criar interfaces de usuário

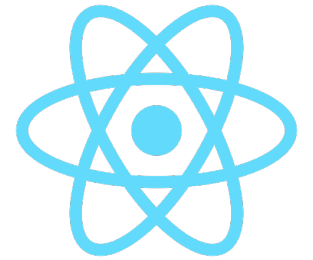
Desenvolvido por engenheiros do Facebook, o React é uma biblioteca JavaScript que revolucionou a maneira como os desenvolvedores projetam e pensam sobre visualizações em aplicativos da web. Ele introduziu uma maneira de os desenvolvedores descreverem declarativamente as interfaces de usuário e modelarem o estado dessas interfaces.



Baseado em componentes

Crie componentes encapsulados que gerenciam seu próprio estado e então, combine-os para criar UIs complexas.





1º passo. – Container no documento HTML

```
<html>

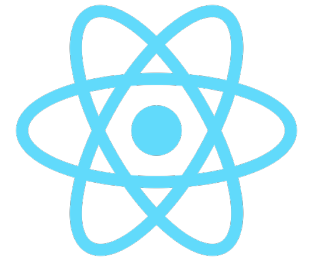
<head>
  <meta charset='utf-8'>
  <meta http-equiv='X-UA-Compatible' content='IE=edge'>
  <title>Exemplo 01 React</title>
  <meta name='viewport' content='width=device-width, initial-scale=1'>
</head>

<body>

  <div id="conteudo"></div>

  <script>
```

Nosso primeiro exemplo será feito em um único arquivo HTML, nesse arquivo a única tag html será uma div com id="conteudo".



2º passo. – Scripts JS / JSX / Links CDN

```
<script type="text/babel">
  class App extends React.Component {
    render() {
      return (
        <div>
          <p> Olá Recode!! Esse é nosso primeiro Componente React </p>
        </div>
      )
    }
  }

  ReactDOM.render(<App />, document.getElementById("conteudo"))
</script>
<script src="https://unpkg.com/react@16/umd/react.development.js" crossorigin></script>
<script src="https://unpkg.com/react-dom@16/umd/react-dom.development.js" crossorigin></script>
<script src="https://unpkg.com/babel-standalone@6/babel.min.js"></script>
```

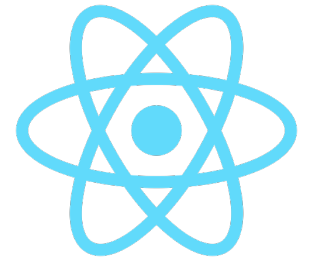
Componente React

Renderiza o conteúdo do componente

Links CDN

Olá Recode!! Esse é nosso primeiro Componente React

Saída no navegador



Detalhes do código

```
<script type="text/babel">
```

Indica que o tipo de script será transpilado pelo Babel

```
class App extends React.Component
```

Essa classe representa um componente React e herda da super classe React.component

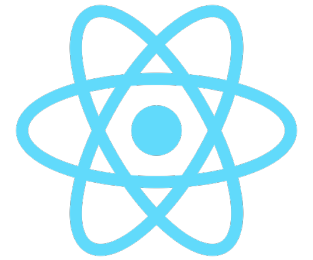
```
render() {  
  return (  
    <div>  
      <p> Olá Recode!! Esse é nosso primeiro Componente React </p>  
    </div>  
  )  
}
```

Método que renderiza o retorno JSX criando o html necessário para compor a UI do usuário

```
ReactDOM.render(<App />, document.getElementById("conteudo"))
```

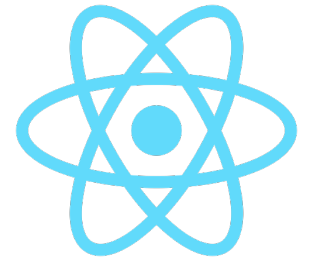
Componente que será renderizado

Container HTML que vai receber o componente App



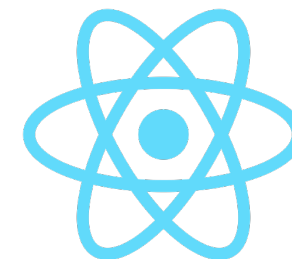
Overview – Código Online

<https://stackblitz.com/edit/react-7y9vcj>



Antes de abordar sobre o JSX

<https://www.w3schools.com/xml/default.asp>



Considere esta declaração de variável:

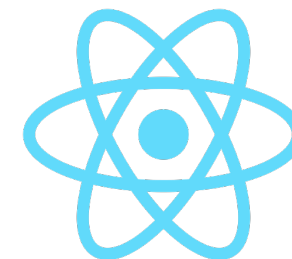
```
const element = <h1>Hello, world!</h1>;
```

Esta sintaxe estranha de tags não é uma string, nem HTML. É chamada JSX e é uma extensão de sintaxe para JavaScript. Recomendamos usar JSX com o React para descrever como a UI deveria parecer.

Por que JSX?

O React adota o fato de que a lógica de renderização é inerentemente acoplada com outras lógicas de UI: como eventos são manipulados, como o state muda com o tempo e como os dados são preparados para exibição.

Fonte: <https://pt-br.reactjs.org/docs/introducing-jsx.html>



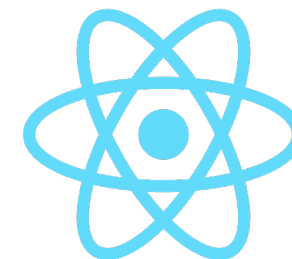
Incorporando Expressões em JSX

No exemplo abaixo, declaramos uma variável chamada `name` e então a usamos dentro do JSX ao envolvê-la com chaves:

```
const name = 'Josh Perez';
const element = <h1>Hello, {name}</h1>;

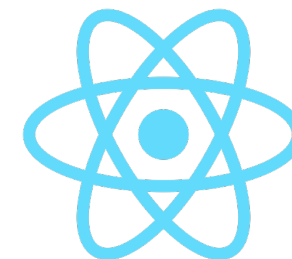
ReactDOM.render(
  element,
  document.getElementById('root')
);
```

Você pode inserir qualquer expressão JavaScript válida dentro das chaves em JSX. Por exemplo, `2 + 2`, `user.firstName`, ou `formatName(user)` são todas expressões JavaScript válidas.



No exemplo abaixo, incorporamos o resultado da chamada de uma função JavaScript, `formatName(user)`, dentro de um elemento `<h1>`.

```
function formatName(user) {  
  return user.firstName + ' ' + user.lastName;  
}  
  
const user = {  
  firstName: 'Harper',  
  lastName: 'Perez'  
};  
  
const element = (  
  <h1>  
    Hello, {formatName(user)}!  
  </h1>  
)  
;  
  
ReactDOM.render(  
  element,  
  document.getElementById('root')  
)
```

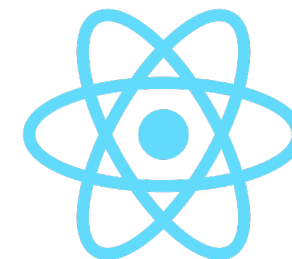


JSX Também é uma Expressão

Depois da compilação, as expressões em JSX se transformam em chamadas normais de funções que retornam objetos JavaScript.

Isto significa que você pode usar JSX dentro de condições `if` e laços `for`, atribuí-lo a variáveis, aceitá-lo como argumentos e retorná-los de funções:

```
function getGreeting(user) {  
  if (user) {  
    return <h1>Hello, {formatName(user)}!</h1>;  
  }  
  return <h1>Hello, Stranger.</h1>;  
}
```



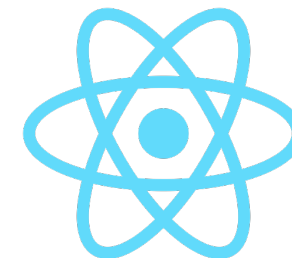
Especificando Elementos Filhos com JSX

Se uma tag está vazia, você pode fechá-la imediatamente com `</>`, como XML:

```
const element = <img src={user.avatarUrl} />;
```

Tags JSX podem conter elementos filhos:

```
const element = (  
  <div>  
    <h1>Hello!</h1>  
    <h2>Good to see you here.</h2>  
  </div>  
);
```



Desafio 01

Partindo do *Olá Mundo* (<https://stackblitz.com/edit/react-7y9vcj>), deverá ser alterado da seguinte forma:

- na classe principal **App**, devem ser adicionados 2 novos métodos: **getTitulo** e **getParagrafo**.
- getTitulo deve receber um parâmetro de texto e retornar um JSX com o texto envolvido pela tag **h1**.
- getParagrafo deve receber dois parâmetros: nome e texto. O parâmetro nome deve ser envolvido pela tag de negrito (**b**), seguido do parâmetro texto, então a função deve retornar um JSX ambos envolvidos pela tag **p**.

Desenvolvedor Full-Stack

Objetivo: Aprender tecnologias incríveis para construir coisas magníficas

Tecnologias aprendidas: JavaScript, TypeScript, ReactJS, Angular, Python, NodeJS entre outras

← Resultado do desafio no navegador