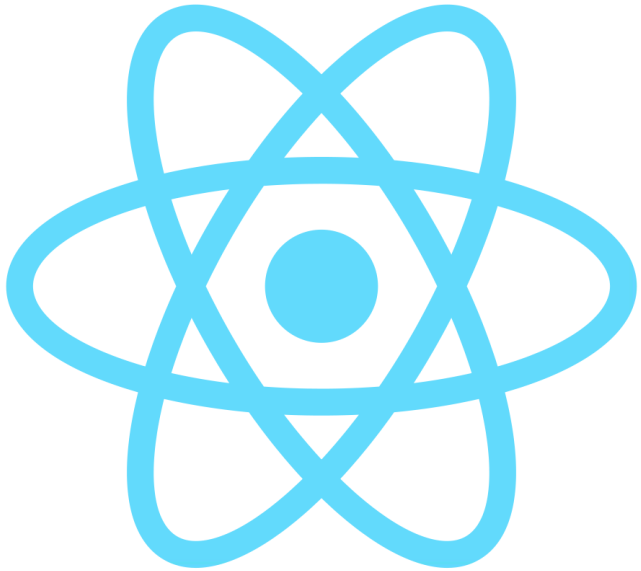
RECODE pro



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.

Fonte: https://freefrontend.com/reactjs-books/

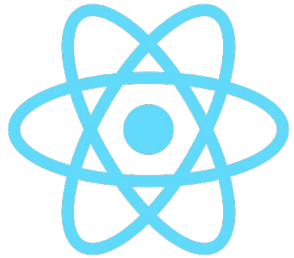
Baseado em componentes

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

https://www.webdesignemfoco.com/cursos/react-js/react-js-botao-show-hidden-toggle

RECODE pro

Instalação em uma página HTML



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".

RECODE pro

Instalação em uma página HTML

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

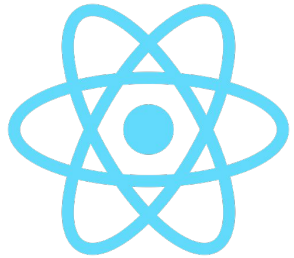
Links CDN

Olá Recode!! Esse é nosso primeiro Componente React

Saída no navegador

RECODE pro

Instalação em uma página HTML



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

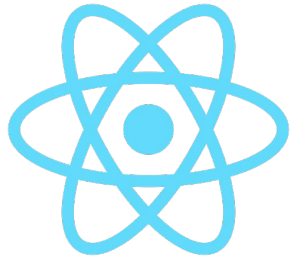
```
ReactDOM.render(<App />, document.getElementById("conteudo"))
```

Componente que será renderizado

Container HTML que vai receber o componente App

RECODE pro

Introduzindo JSX (JavaScript XML)



Considere esta declaração de

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

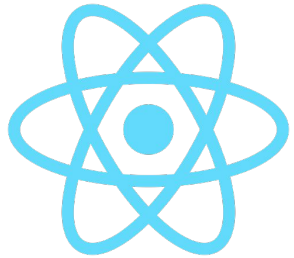
Esta sintaxe estranha de tags não é uma string, nem HTML.
É chamada JSX e é uma extensão de sintaxe para JavaScript. Recomendamos usar JSX com o React para descrever como a UI deveria parecer.

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

RECODE pro

Introduzindo JSX (JavaScript XML)



Incorporando Expressões em JSX

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

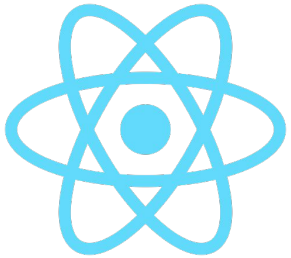
```
const name = 'Josh Perez';
const element = <h1>Hello, {name}</h1>;

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.

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

RECODE pro

Introduzindo JSX (JavaScript XML)



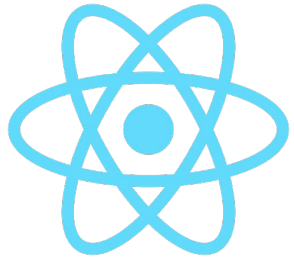
No exemplo abaixo, incorporamos o resultado da chamada de uma função JavaScript, `formatName(user)`, dentro de um elemento `<h1>`.

```
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')
);
```

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

RECODE pro

Introduzindo JSX (JavaScript XML)



JSX Também é uma Expressão

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

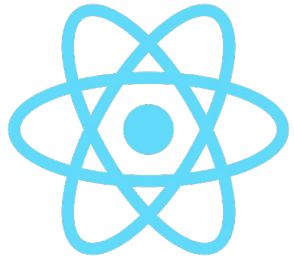
Isto significa que você pode usar JSX dentro de condições `if` e laços `for`, atribuí-lo a variáveis, aceitá-lo como argumentos e retorná-los de funções:

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

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

RECODE pro

Introduzindo JSX (JavaScript XML)



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>
);
```

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

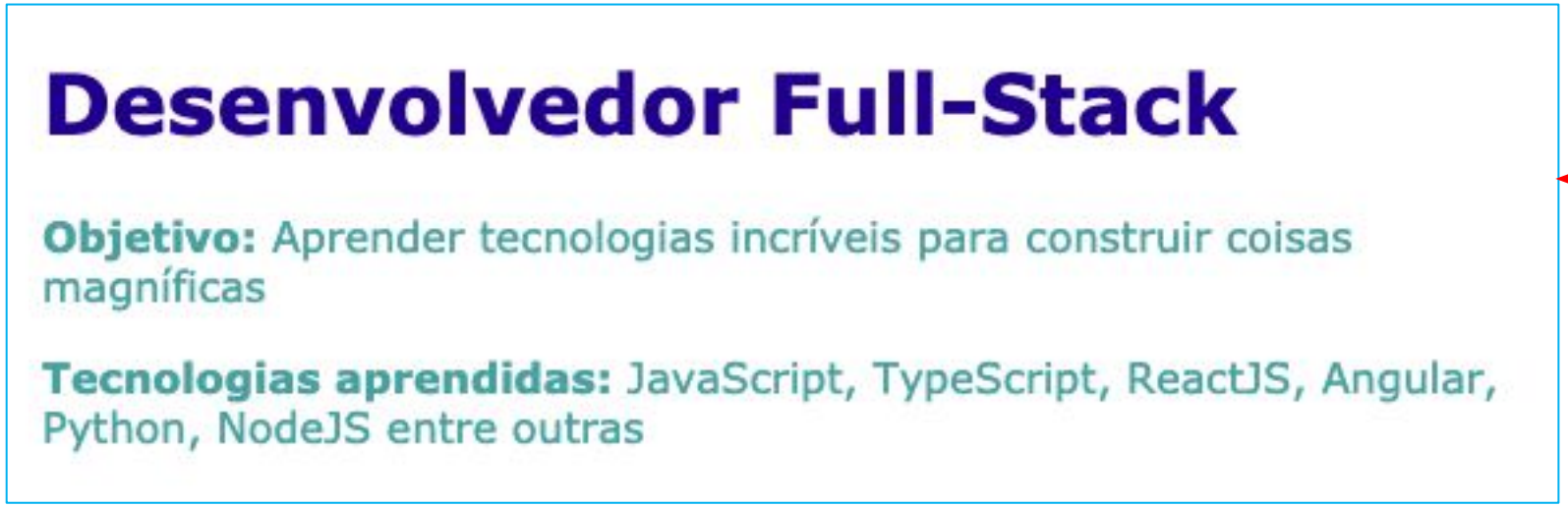
RECODE pro

Introduzindo JSX (JavaScript XML)

Desafio 01

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

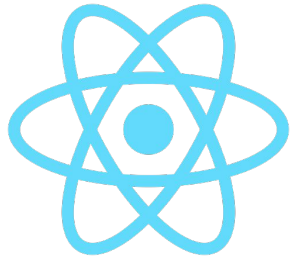
- na classe principal ***App***, devem ser adicionados 2 novos métodos: ***getTitulo*** e ***getParagrafo***.
- getTitulo deve receber um parâmetro de texto e retornar um JSX com o texto envolvido pela tag ***h1***.
- getParagrafo deve receber dois parâmetros: nome e texto. O parâmetro nome deve ser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



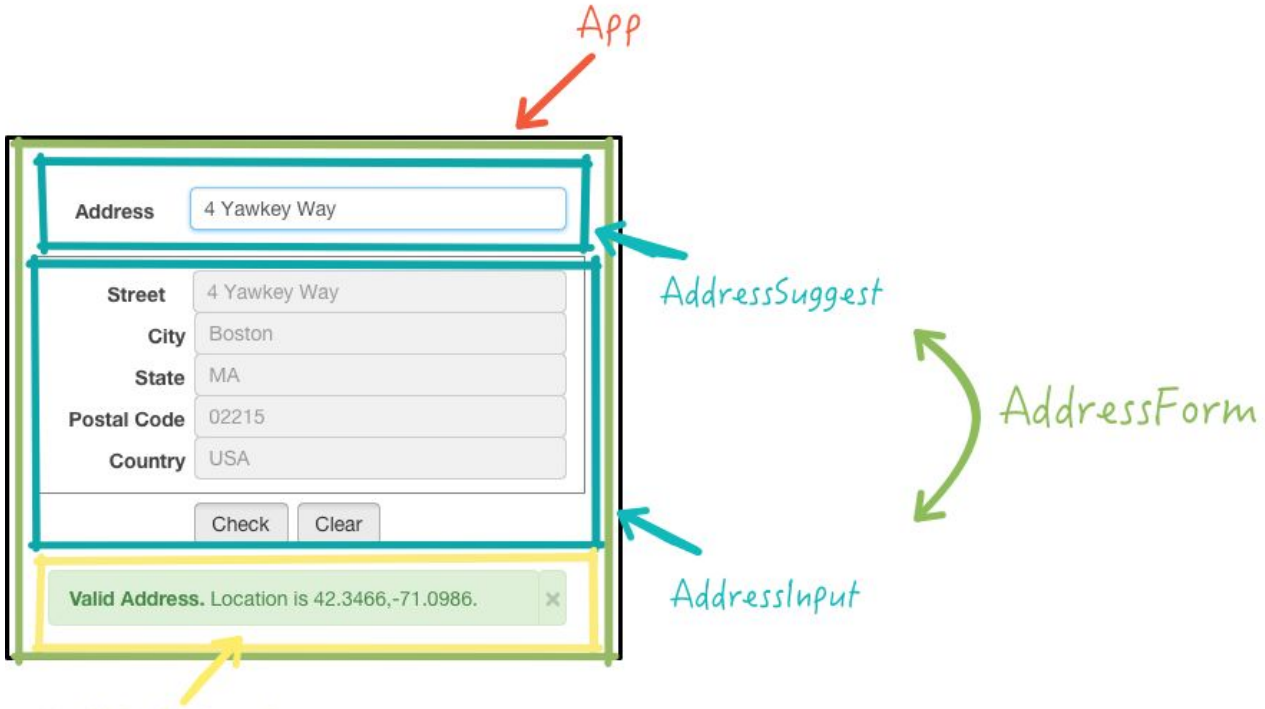
Resultado do desafio no navegador

RECODE pro

Componentes e Props



Componentes permitem você dividir a UI em partes independentes, reutilizáveis e pensar em cada parte isoladamente.
Conceitualmente, componentes são como funções JavaScript. Eles aceitam entradas arbitrárias (chamadas “props”) e retornam elementos React que descrevem o que deve aparecer na tela.

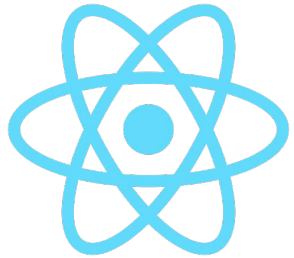


Fonte: https://pt-br.reactjs.org/docs/components-and-props.html

Fonte: https://developer.here.com/blog/street-address-validation-with-reactjs-and-here-geocoder-autocomplete

RECODE pro

Componentes e Props



Componentes de Função e Classe

A maneira mais simples de definir um componente é escrever uma função JavaScript:

```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}
```

Essa função é um componente React válido porque aceita um único argumento de objeto "props" (que significa propriedades) com dados e retorna um elemento React. Nós chamamos esses componentes de "componentes de função" porque são literalmente funções JavaScript.

Você também pode usar uma classe ES6 para definir um componente:

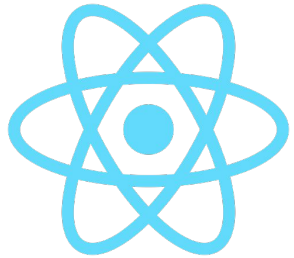
```
class Welcome extends React.Component {
  render() {
    return <h1>Hello, {this.props.name}</h1>;
  }
}
```

Os dois componentes acima são equivalentes do ponto de vista do React.

Fonte: https://pt-br.reactjs.org/docs/components-and-props.html

RECODE pro

Componentes e Props



```jsx
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}

const element = <Welcome name="Sara" />;
ReactDOM.render(
  element,
  document.getElementById('root')
);
```

Try it on CodePen

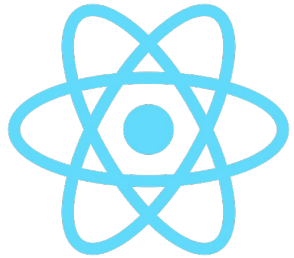
Vamos recapitular o que acontece nesse exemplo:

1. Nós chamamos `ReactDOM.render()` com o elemento `<Welcome name="Sara" />`.
2. React chama o componente `Welcome` com `{name: 'Sara'}` como props.
3. Nosso componente `Welcome` retorna um elemento `<h1>Hello, Sara</h1>` como resultado.
4. React DOM atualiza eficientemente o DOM para corresponder `<h1>Hello, Sara</h1>`.

Fonte: https://pt-br.reactjs.org/docs/components-and-props.html

RECODE pro

Componentes e Props



Compondo Componentes

Componentes podem se referir a outros componentes em sua saída. Isso nos permite usar a mesma abstração de componente para qualquer nível de detalhe. Um botão, um formulário, uma caixa de diálogo, uma tela: em aplicativos React, todos esses são normalmente expressos como componentes.

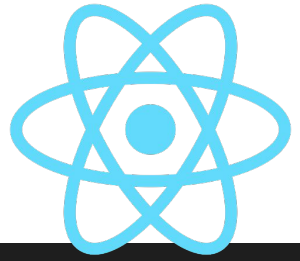
Por exemplo, nós podemos criar um componente `App` que renderiza `Welcome` muitas vezes:

```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}

function App() {
  return (
    <div>
      <Welcome name="Sara" />
      <Welcome name="Cahal" />
      <Welcome name="Edite" />
    </div>
  );
}

ReactDOM.render(
  <App />,
  document.getElementById('root')
);
```

Fonte: https://pt-br.reactjs.org/docs/components-and-props.html

RECODE pro

Exemplo - Props



Index.js parte 01

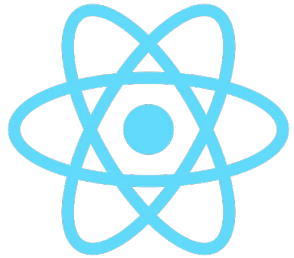
```
class MyButton extends React.Component {
    render() {
        return (
        <button>{this.props.nome}</button>
        )
    }
}

class MyLabel extends React.Component {
    render() {
        return (
            <p>{this.props.texto}</p>
        )
    }
}
```

Index.js parte 02

```
class App extends React.Component {
    render() {
        return (
            <div>
                <MyLabel texto="Recode Pro 2019"/>
                <MyButton nome="Botão 01"/>
                <MyButton nome="Botão 02"/>
                <MyButton nome="Botão 03"/>
            </div>
        )
    }
}


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

RECODE pro

Exemplo - Props



Index.html

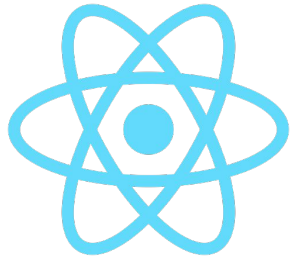
```
<!DOCTYPE html>
<html>

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

</head>

<body>

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


    <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>
    <script type="text/babel" src="index.js"> </script>

</body>

</html>
```

Saída no navegador

Recode Pro 2019

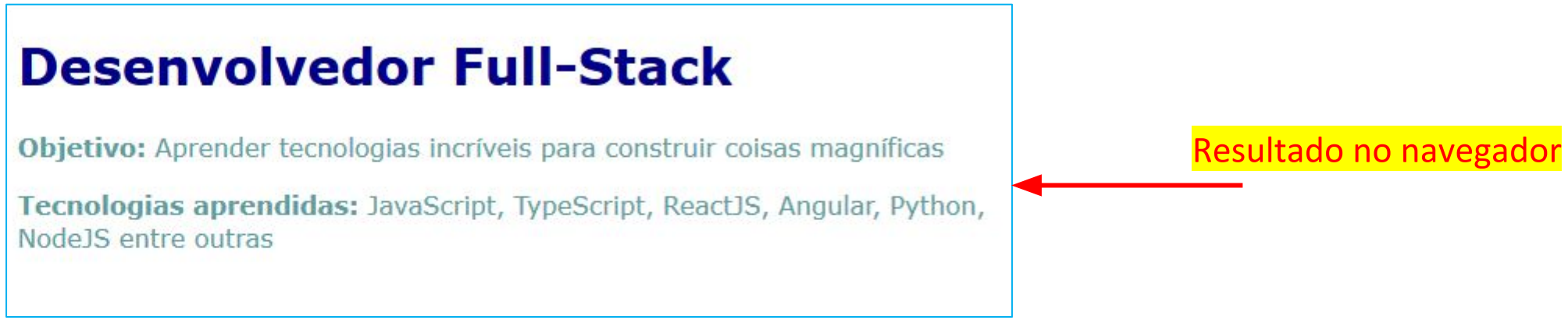
Botão 01 | Botão 02 | Botão 03

RECODE pro



Desafio

Neste desafio, vamos alterar o projeto anterior e construir 2 novos componentes: *CursoHeader* e *CursoContent*. O primeiro cria o título e o segundo criando as linhas seguintes, ambos recebendo os dados por ***props***.



Resultado no navegador