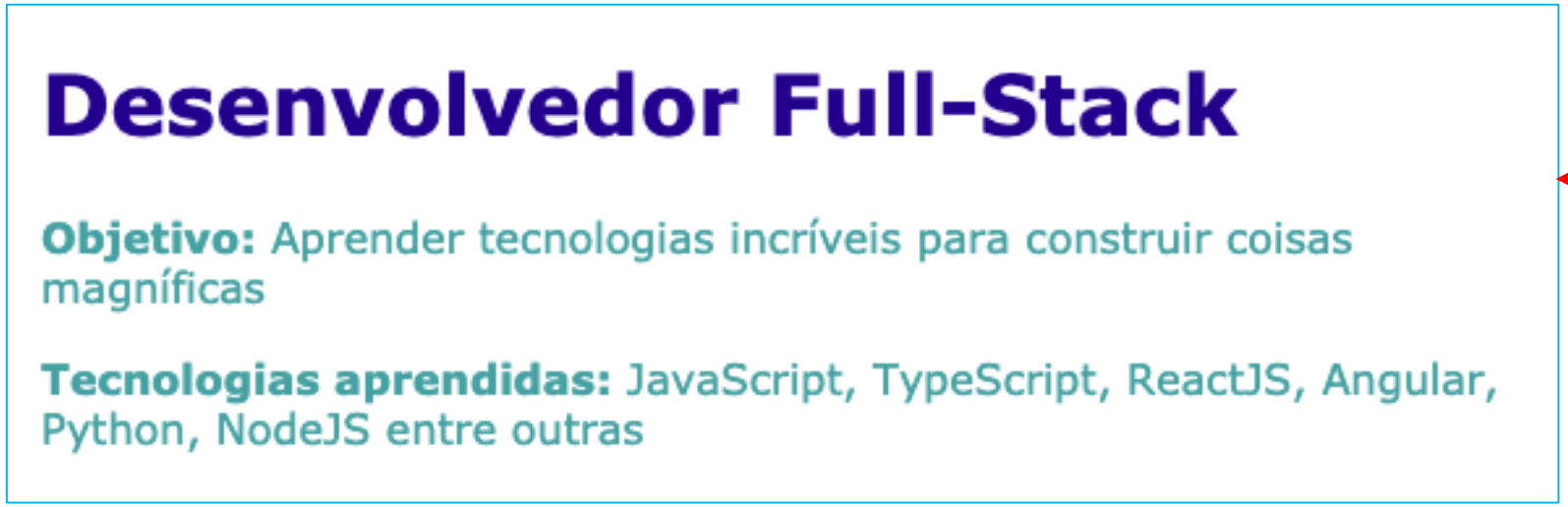
RECODE pro

Resolução desafio anterior

Desafio 02

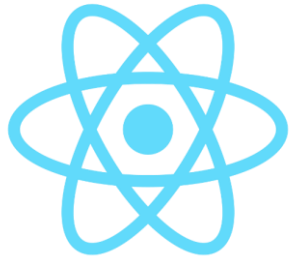
Neste desafio, vamos alterar o projeto anterior e construir 2 novos componentes: *CursoHeader* e *CursoContent*. O primeiro cria o título e o segundo criando as linhas seguintes, ambos recebendo os dados por ***props***.



Resultado do desafio no navegador

RECODE pro

Resolução desafio anterior



Desafio 02

```
import React, { Component } from 'react';
import { render } from 'react-dom';
import './style.css';

class CursoHeader extends Component {
  render(){
    return <h1>{this.props.nome}</h1>
  }
}

class CursoContent extends Component {
  render(){
    return <p><b>{this.props.item}: </b> {this.props.valor}</p>
  }
}
```

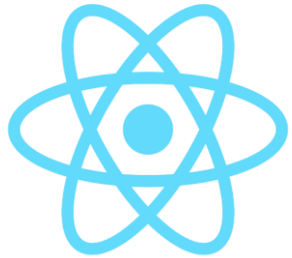
Componente CursoHeader

Componente CursoContent

RECODE pro

Resolução desafio anterior

Desafio 02



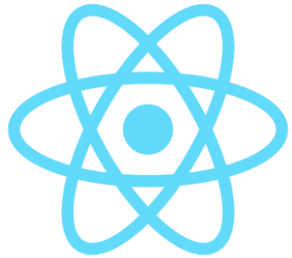
```
class App extends Component {

  render() {
    return (
      <div>

        <CursoHeader nome='Desenvolvedor Full-Stack'/>

        <CursoContent item='Objetivo' valor='Aprender tecnologias incríveis para construir coisas
        magníficas' />

        <CursoContent item='Tecnologias aprendidas' valor='JavaScript, TypeScript, ReactJS,
        Angular, Python, NodeJS entre outras' />

      </div>
    );
  }
}

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

Chamada dos componentes passando valores para seus props(propriedades)

Chamada do render principal

RECODE pro

Objeto State



Os Componentes do React possuem um objeto `state` interno.
O objeto `state` é onde você armazena valores de propriedade que pertencem ao componente. Quando o objeto `state` muda, o componente é renderizado novamente.

Criando um objeto state

O objeto `state` é inicializado no construtor do componente:

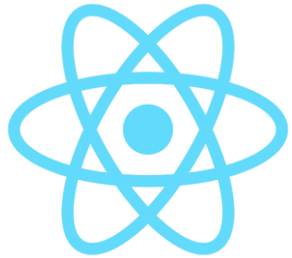
```
class Car extends React.Component {
  constructor(props) {
    super(props);
    this.state = {brand: "Ford"};
  }
  render() {
    return (
      <div>
        <h1>My Car</h1>
      </div>
    );
  }
}
```

State do componente Car

Fonte: https://www.w3schools.com/react/react_state.asp

RECODE pro

Objeto State



Exemplo:

Especifique todas as propriedades que seu componente precisa:

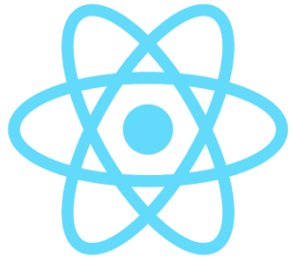
```
class Car extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      brand: "Ford",
      model: "Mustang",
      color: "red",
      year: 1964
    };
  }
  render() {
    return (
      <div>
        <h1>My Car</h1>
      </div>
    );
  }
}
```

Objeto state do componente Car com varias propriedades

Fonte: https://www.w3schools.com/react/react_state.asp

RECODE pro

Objeto State



```
class Car extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      brand: "Ford",
      model: "Mustang",
      color: "red",
      year: 1964
    };
  }
  render() {
    return (
      <div>
        <h1>My {this.state.brand}</h1>
        <p>
          It is a {this.state.color}
          {this.state.model}
          from {this.state.year}.
        </p>
      </div>
    );
  }
}
```

Usando o state do componente

Fonte: https://www.w3schools.com/react/react_state.asp

RECODE pro

Objeto State

Alterando o estado do objeto

Quando um valor no `state` do objeto é alterado, o componente é renderizado novamente, o que significa que a saída será alterada de acordo com os novos valores.

```
class Car extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      brand: "Ford",
      model: "Mustang",
      color: "red",
      year: 1964
    };
  }
  changeColor = () => {
    this.setState({color: "blue"});
  }
```

Para alterar o valor de um state temos que usar o método setState()

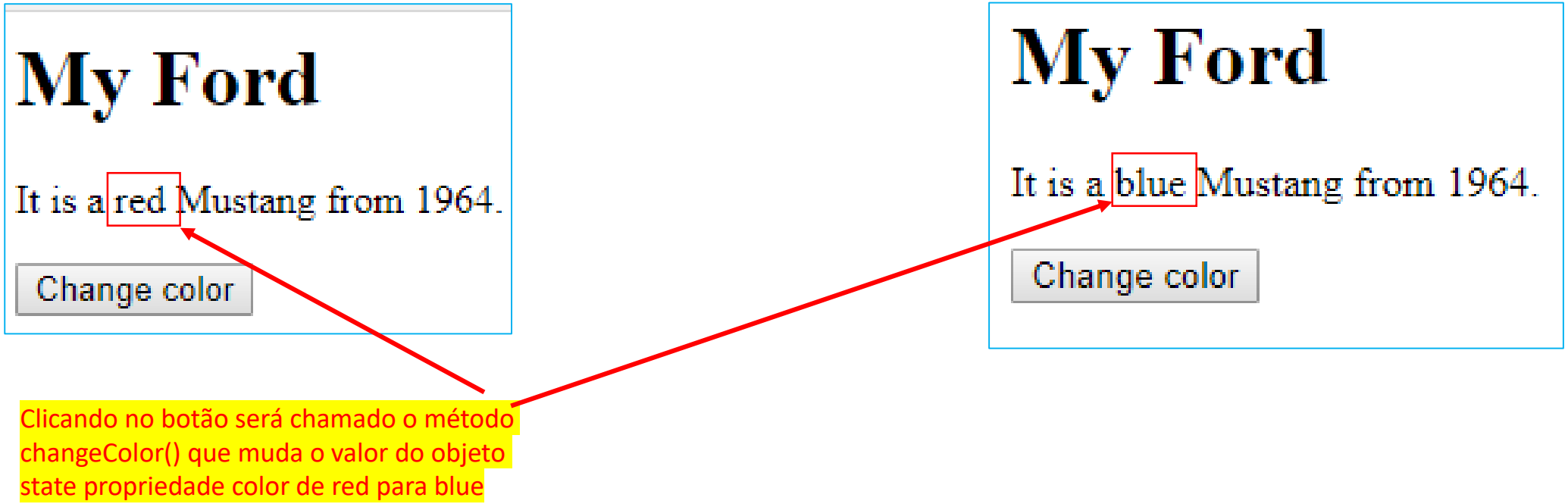
```
render() {
  return (
    <div>
      <h1>My {this.state.brand}</h1>
      <p>
        It is a {this.state.color}
        {this.state.model}
        from {this.state.year}.
      </p>
      <button
        type="button"
        onClick={this.changeColor}
      >Change color</button>
    </div>
  );
}
```

Botão que chama o método changeColor()

Fonte: https://www.w3schools.com/react/react_state.asp

Objeto State

Alterando o estado do objeto

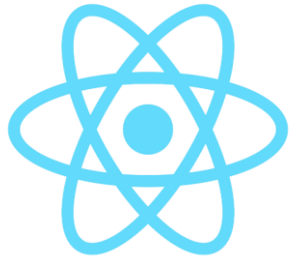


Clicando no botão será chamado o método changeColor() que muda o valor do objeto state propriedade color de red para blue

Fonte: https://www.w3schools.com/react/react_state.asp

RECODE pro

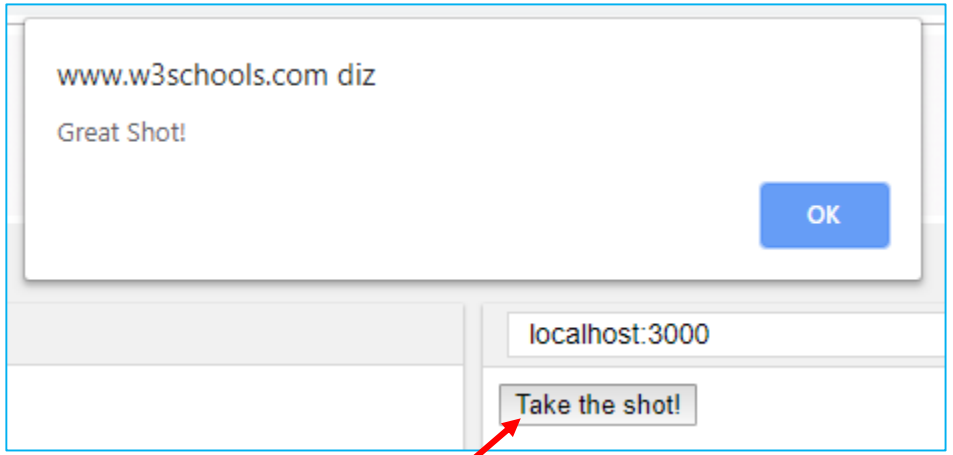
Adicionando eventos



Assim como o HTML, o React pode executar ações com base nos eventos do usuário.
O React possui os mesmos eventos que o HTML: click, change, mouseover etc

Os eventos do React são gravados na sintaxe camelCase:
`onClick` em vez de `onclick`.
Os manipuladores de eventos do React são escritos dentro de chaves:
`onClick={shoot}` em vez de `onClick="shoot()"`.

```
class Football extends React.Component {
  shoot() {
    alert("Great Shot!");
  }
  render() {
    return (
      <button onClick={this.shoot}>Take the shot!</button>
    );
  }
}

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

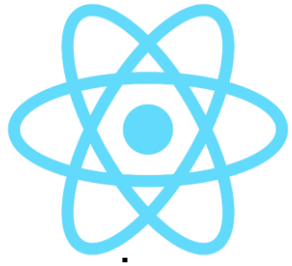


Chamada do método shoot no botão take the shoot

Fonte: https://www.w3schools.com/react/react_state.asp

RECODE pro

Adicionando eventos



Bind this

Para métodos em React, a palavra-chave `this` deve representar o componente que possui o método. É por isso que você deve usar as funções de seta. Com as funções de seta, `this` sempre representará o objeto que definiu a função.

Exemplo:

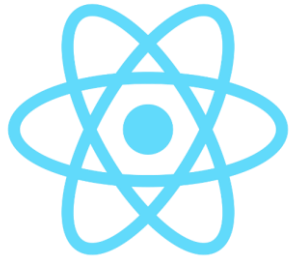
```
class Football extends React.Component {
  shoot = () => {
    alert(this);
    /*
    The 'this' keyword refers to the component object
    */
  }
  render() {
    return (
      <button onClick={this.shoot}>Take the shot!</button>
    );
  }
}

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

Arrow function (função de seta) recomendado pra agilizar os binds no React

Fonte: https://www.w3schools.com/react/react_events.asp

RECODE pro

Adicionando eventos



Bind this

```
class Football extends React.Component {
  constructor(props) {
    super(props)
    this.shoot = this.shoot.bind(this)
  }
  shoot() {
    alert(this);
    /*
    Thanks to the binding in the constructor function,
    the 'this' keyword now refers to the component object
    */
  }
  render() {
    return (
      <button onClick={this.shoot}>Take the shot!</button>
    );
  }
}

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

Vinculo do evento com o componente

Fonte: https://www.w3schools.com/react/react_events.asp

RECODE pro

Adicionando eventos

Passando argumentos (Parâmetros)

Se você deseja enviar parâmetros para um manipulador de eventos, você tem duas opções:

1. Faça uma função de seta anônima:

Exemplo:

Envie "Objetivo" como um parâmetro para a `shoot` função, usando a função de seta:

```
class Football extends React.Component {
  shoot = (a) => {
    alert(a);
  }
  render() {
    return (
      <button onClick={() => this.shoot("Goal")}>Take the shot!</button>
    );
  }
}

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

Passando parâmetros para a função de seta shoot

Fonte: https://www.w3schools.com/react/react_events.asp

RECODE pro

Adicionando eventos

Passando argumentos (Parâmetros)

Se você deseja enviar parâmetros para um manipulador de eventos, você tem duas opções:

2. Ligue o manipulador de eventos a `this`.

Observe que o primeiro argumento deve ser `this`.

Exemplo:

Envie "Objetivo" como um parâmetro para a `shoot` função:

```
class Football extends React.Component {
  shoot(a) {
    alert(a);
  }
  render() {
    return (
      <button onClick={this.shoot.bind(this, "Goal")}>Take the shot!</button>
    );
  }
}

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

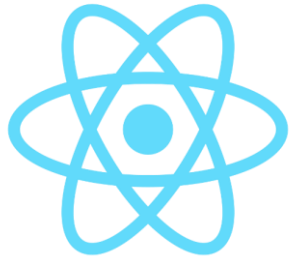
Passando parâmetros para a função shoot

Nota no segundo exemplo: Se você enviar argumentos sem usar o método `bind` (em `this.shoot(this, "Goal")` em vez de `this.shoot.bind(this, "Goal")`), a função `shoot` será executada quando a página for carregada, em vez de esperar que o botão seja clicado.

Fonte: https://www.w3schools.com/react/react_events.asp

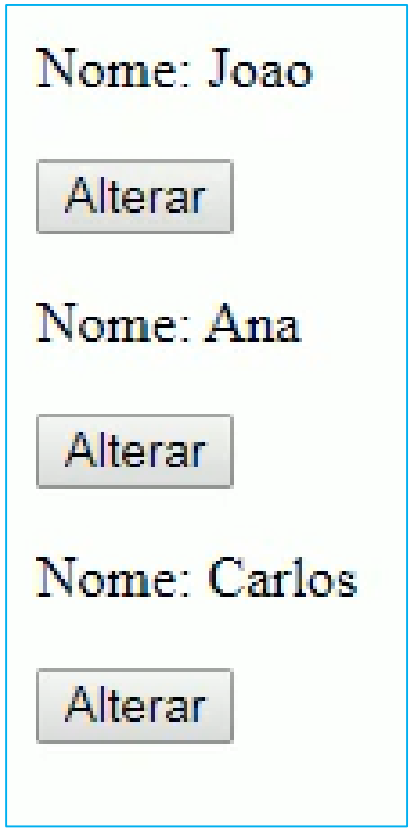
RECODE pro

Desafio 01

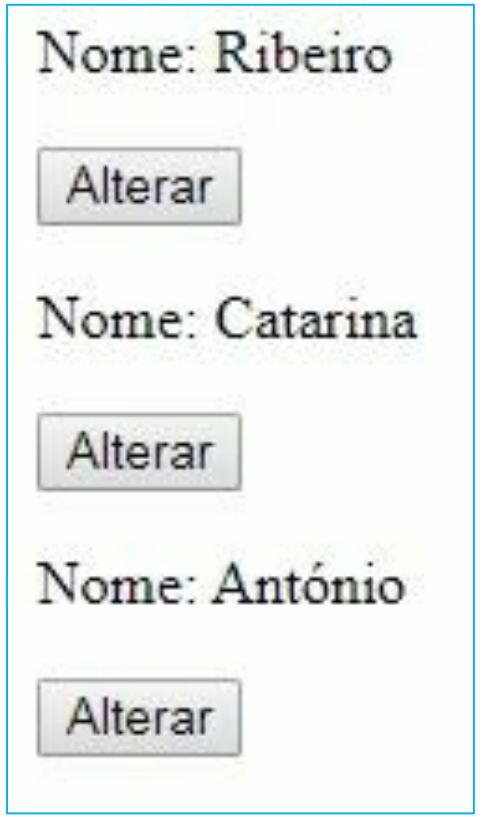


Usando **states, props e eventos**, crie um projeto com as seguintes características:
Observação: o projeto deverá conter apenas dois componentes <App> e <Conteudo>

Estado inicial



Estado após a execução do método através dos botões



RECODE pro

Desafio 02

Estado inicial

Após clicar no botão English

Aproveitando o projeto anterior!!

Usando **states, props e eventos**, refatorar o exemplo anterior conforme a imagem ao lado.

Observação: após clicar no <botão Português> a página deve voltar para o idioma inicial português.

Flávio Mota