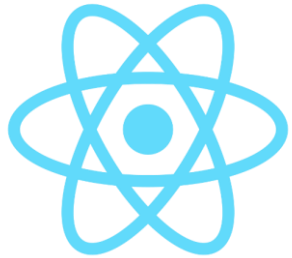
RECODE pro

Resolução Desafio 01 anterior



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

Nome: Joao

Alterar

Nome: Ana

Alterar

Nome: Carlos

Alterar

Estado após a execução do método através dos botões

Nome: Ribeiro

Alterar

Nome: Catarina

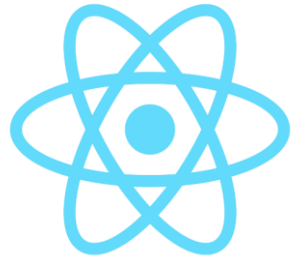
Alterar

Nome: António

Alterar

RECODE pro

Resolução Desafio 01 anterior



```
class Conteudo extends React.Component {

    constructor(props) {
        super(props)
        this.state = {
            nome: this.props.nome_inicial
        }

        this.mudaNome = this.mudaNome.bind(this)

    }

    mudaNome(){
        this.setState({ nome: this.props.nome_final })
    }
    render() {
        return (
            <div>
                <p>Nome: {this.state.nome}</p>
                <button onClick={this.mudaNome}> Alterar </button>
            </div>

        )
    }
}
```

State nome inicia com o props nome_inicial

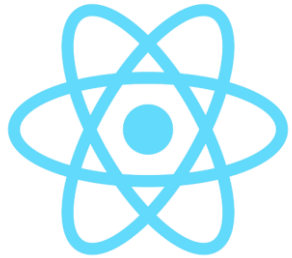
Faz o vinculo do método mudaNome() com uma instancia do componente Conteudo

Método mudaNome chama o método setState para alterar o estado da propriedade nome usando o props nome_final

Quando o componente Conteudo for renderizado já será mostrado o valor do state nome e o método mudaNome() estará associado ao click do botão.

RECODE pro

Resolução Desafio 01 anterior



```
class App extends React.Component {
    render() {
        return (
            <div>
                <Conteudo nome_inicial = "João" nome_final = "Ribeiro" />
                <Conteudo nome_inicial = "Ana" nome_final = "Catarina" />
                <Conteudo nome_inicial = "Carlos" nome_final = "Antonio" />
            </div>
        )
    }
}


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

O componente Conteudo sendo renderizado três vezes com valores diferente para seus props nome_inicial e nome_final.

RECODE pro

Resolução Desafio 02 anterior

Estado inicial

Após clicar no botão English

Aproveitando o projeto anterior!!

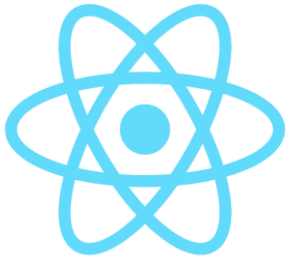
Usando **states, props e eventos**, refatorar o exemplo anterior conforme a imagem ao lado.

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

Flávio Mota

RECODE pro

Resolução Desafio 02 anterior

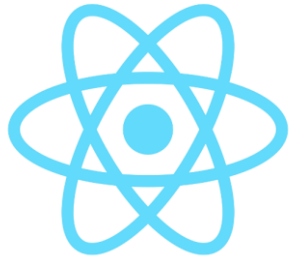


Segue o link da resolução do desfio 02

https://stackblitz.com/edit/react-2aefij

RECODE pro

Renderização condicional



Em React, você pode criar componentes distintos que encapsulam o comportamento que você precisa. Então, você pode renderizar apenas alguns dos elementos, dependendo do estado da sua aplicação

Renderização condicional em React funciona da mesma forma que condições funcionam em JavaScript. Use operadores de JavaScript como if ou operador condicional para criar elementos representando o estado atual, e deixe o React atualizar a UI para corresponde-los.

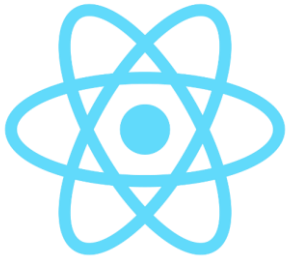
Considere esses dois componentes:

```
function UserGreeting(props) {
  return <h1>Welcome back!</h1>;
}

function GuestGreeting(props) {
  return <h1>Please sign up.</h1>;
}
```

Fonte: https://pt-br.reactjs.org/docs/conditional-rendering.html

RECODE pro

Renderização condicional



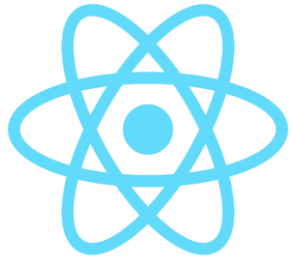
Nós vamos criar um componente `Greeting` que mostra um dos outros dois componentes se o usuário estiver logado ou não:

```
function Greeting(props) {
  const isLoggedIn = props.isLoggedIn;
  if (isLoggedIn) {
    return <UserGreeting />;
  }
  return <GuestGreeting />;
}

ReactDOM.render(
  // Try changing to isLoggedIn={true}:
  <Greeting isLoggedIn={false} />,
  document.getElementById('root')
);
```

Fonte: https://pt-br.reactjs.org/docs/conditional-rendering.html

RECODE pro

Renderização condicional



Variáveis de Elementos

Você pode usar variáveis para guardar elementos. Isto pode te ajudar a renderizar condicionalmente parte do componente enquanto o resto do resultado não muda.

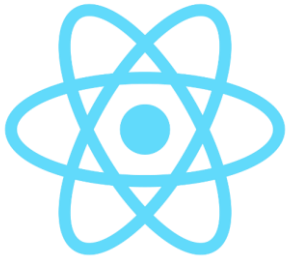
Considere esses dois novos componentes representando os botões de Logout e Login:

```
function LoginButton(props) {
  return (
    <button onClick={props.onClick}>
      Login
    </button>
  );
}

function LogoutButton(props) {
  return (
    <button onClick={props.onClick}>
      Logout
    </button>
  );
}
```

Fonte: https://pt-br.reactjs.org/docs/conditional-rendering.html

RECODE pro

Renderização condicional



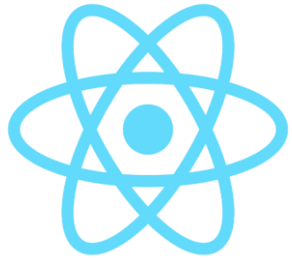
O componente irá renderizar o `<LoginButton />` ou `<LogoutButton />` dependendo do estado atual. Ele tambem irá renderizar `<Greeting />` do exemplo anterior:

```
class LoginControl extends React.Component {
  constructor(props) {
    super(props);
    this.handleLoginClick = this.handleLoginClick.bind(this);
    this.handleLogoutClick = this.handleLogoutClick.bind(this);
    this.state = {isLoggedIn: false};
  }

  handleLoginClick() {
    this.setState({isLoggedIn: true});
  }

  handleLogoutClick() {
    this.setState({isLoggedIn: false});
  }

  render() {
    const isLoggedIn = this.state.isLoggedIn;
    let button;
```

Fonte: https://pt-br.reactjs.org/docs/conditional-rendering.html

RECODE pro

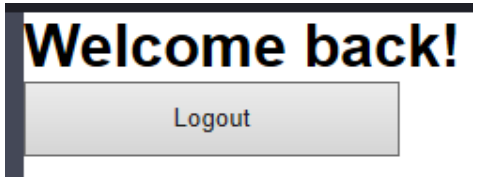
Renderização condicional



```
    if (isLoggedIn) {
      button = <LogoutButton onClick={this.handleLogoutClick} />;
    } else {
      button = <LoginButton onClick={this.handleLoginClick} />;
    }

    return (
      <div>
        <Greeting isLoggedIn={isLoggedIn} />
        {button}
      </div>
    );
  }
}

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

State 01

State 02

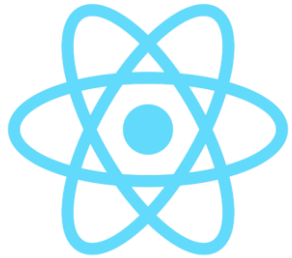


Código online: https://codepen.io/gaearon/pen/QKzAgB?editors=0010

Fonte: https://pt-br.reactjs.org/docs/conditional-rendering.html

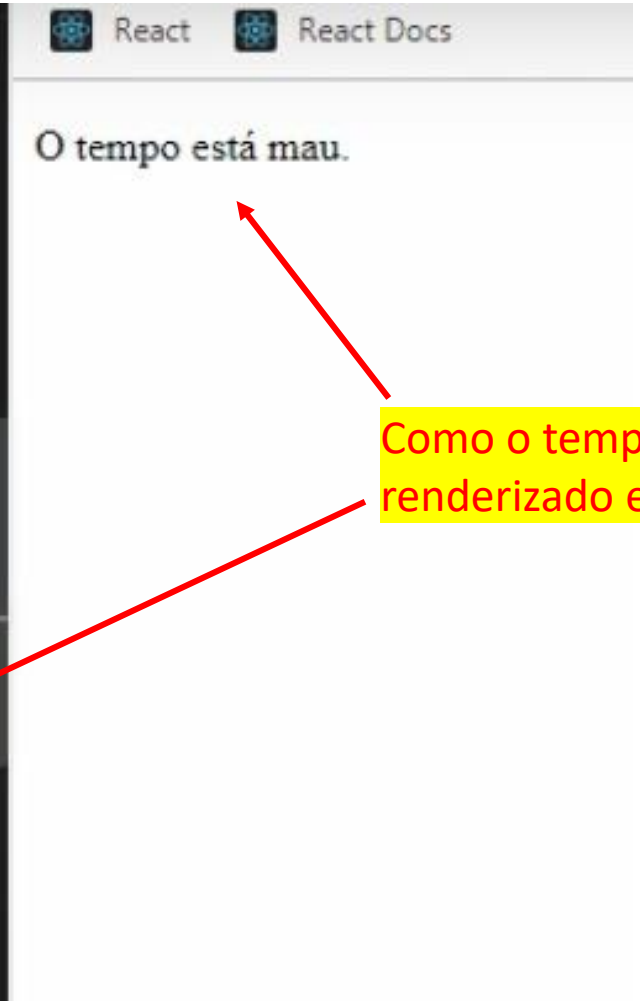
RECODE pro

Renderização condicional



Usando o operador ternário

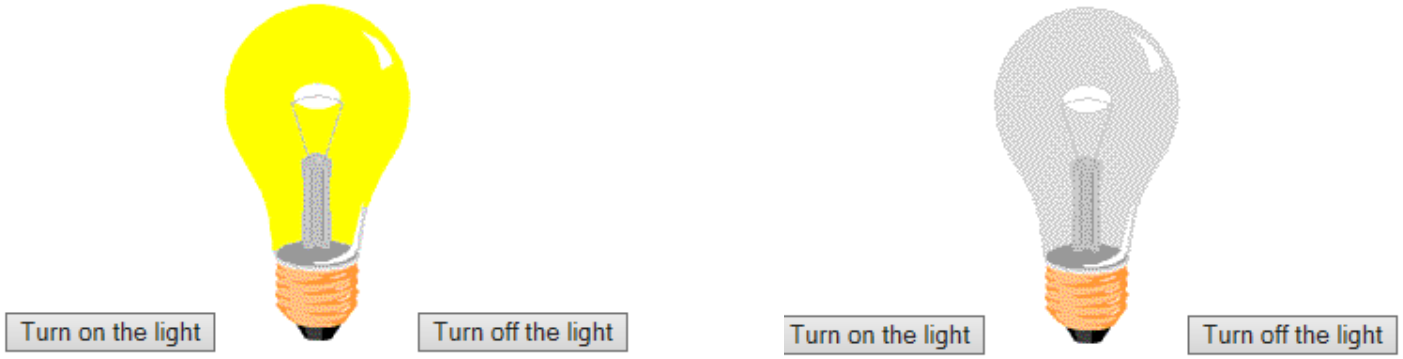
```
constructor(){
    super()
    this.state = {
        tempoBom: false
    }
}

render(){

    // método ternário (condição ternária
    return(
        this.state.tempoBom ?
        <p>O tempo está bom.</p> :
        <p>O tempo está mau.</p>
    )
```



Como o tempo é false será renderizado essa opção

RECODE pro

Desafio 01

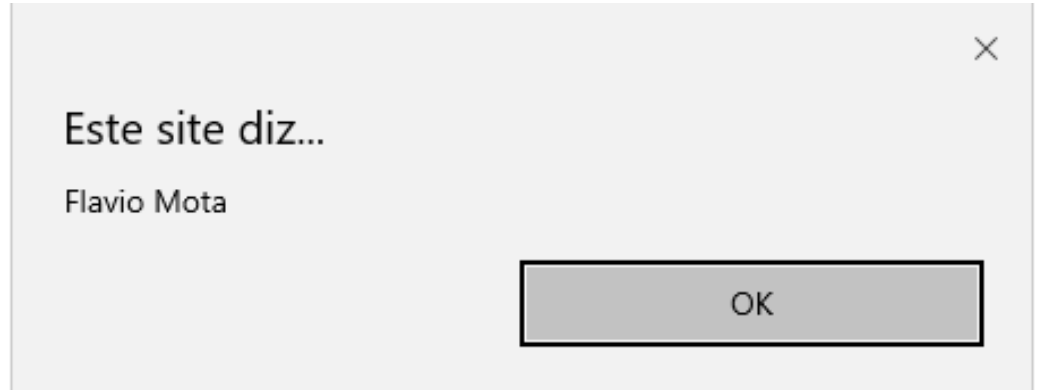


https://www.w3schools.com/js/tryit.asp?filename=tryjs_intro_lightbulb

Transforme o código do link acima em React, de preferencia usando renderização condicional

Elementos de Formulários em React

Nome: Flavio Mota | Pegar Nome




```
class App extends React.Component {
    constructor(props) {
        super(props);
        this.state = { texto: '' };

        this.pegaTexto = this.pegaTexto.bind(this);
        this.mostraTexto = this.mostraTexto.bind(this);

    }

    pegaTexto(event) {
        this.setState({ texto: event.target.value });
    }
    mostraTexto(){
        alert(this.state.texto)
    }
```

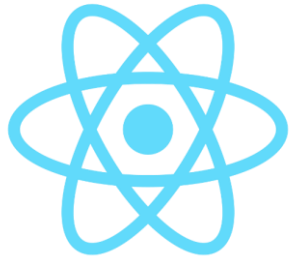
Faz o vinculo dos métodos ao objeto component

Seta o valor do input text no state texto

Mostra o valor do state texto em um alert

RECODE pro

Elementos de Formulários em React



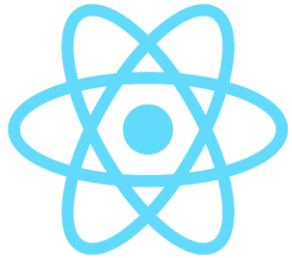
```
    render() {
        return (
            <form>
                Nome:
                <input type="text" onChange={this.pegaTexto} />

                <input type="button" value="Pegar Nome" onClick={this.mostraTexto} />
            </form>
        );
    }
}
ReactDOM.render(<App />, document.getElementById("conteudo"))
```

Campo de texto vinculando evento onChange ao método pegarTexto()

Chama o método mostraTexto()

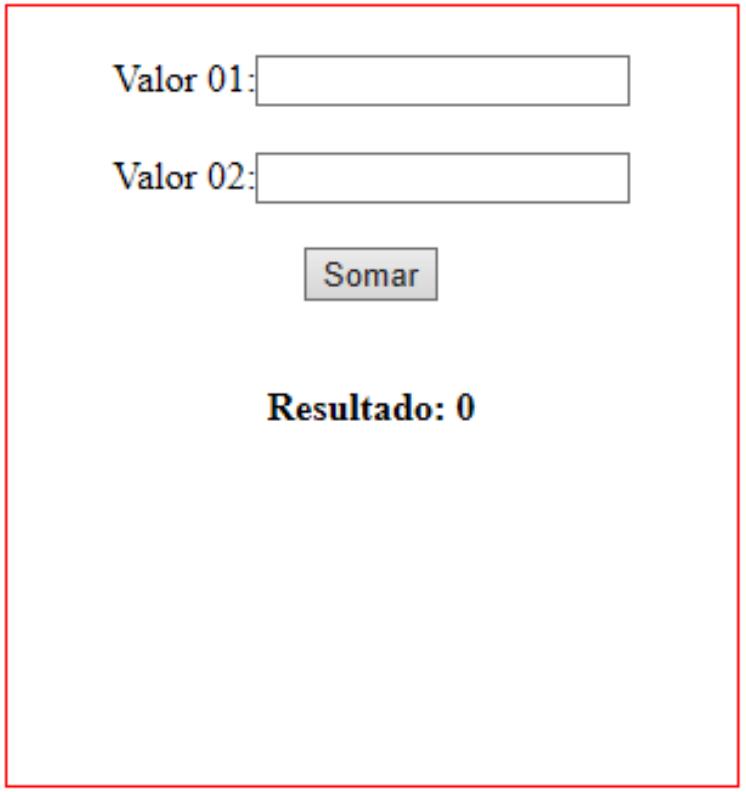
RECODE pro

Elementos de Formulários em React



Exemplo de cálculo básico

Saída no navegador

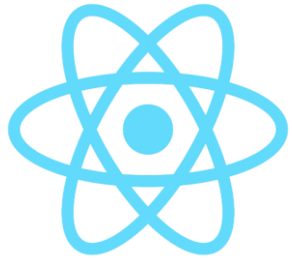


Formatacao.css

```css
.Formatacao{
    width: 300px;
    height: 300px;
    border: 1px solid red;
    margin: auto;
    text-align: center;
    padding-top: 20px;
}
```

RECODE pro

Elementos de Formulários em React

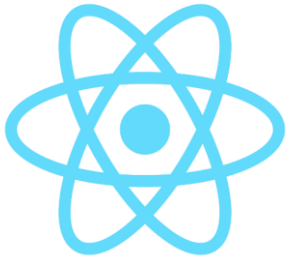


Exemplo de cálculo básico – arquivo index.js parte 01

```
class App extends React.Component {
    constructor(props) {
        super(props);
        this.state = {
            num01: 0,
            num02: 0,
            resultado: 0
        };

        this.manipulador01 = this.manipulador01.bind(this);
        this.manipulador02 = this.manipulador02.bind(this);
        this.calculo = this.calculo.bind(this)

    }

    manipulador01(event) {
        this.setState({ num01: event.target.value });
    }
    manipulador02(event) {
        this.setState({ num02: event.target.value });
    }

    calculo() {
        this.setState({ resultado: (parseFloat(this.state.num01) + parseFloat(this.state.num02)) })
    }
```

RECODE pro

Elementos de Formulários em React

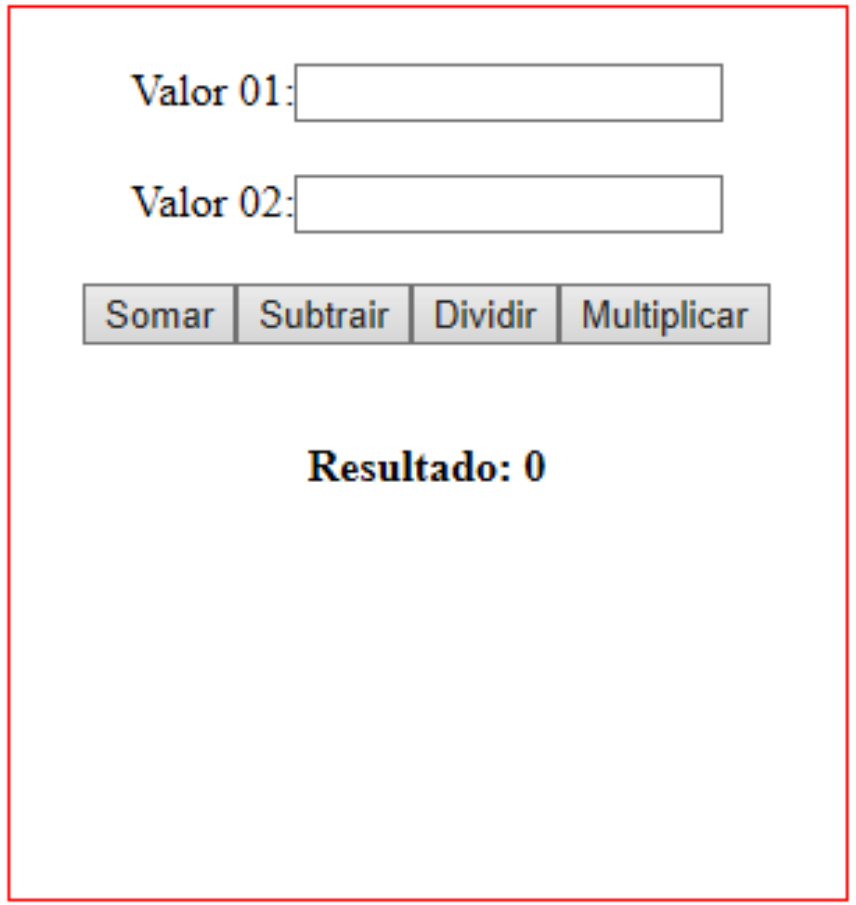


Exemplo de cálculo básico – arquivo index.js parte 02

```
    render() {
        return (
            <div className="Formatacao">
                <form>
                    Valor 01:
                    <input type="text" onChange={this.manipulador01} />
                    <br /><br />

                    Valor 02:
                    <input type="text" onChange={this.manipulador02} />
                    <br /><br />

                    <input type="button" value="Somar" onClick={this.calculo} /><br /><br />

                    <p><b>Resultado: {this.state.resultado}</b></p>
                </form>
            </div>
        );
    }
}
ReactDOM.render(<App />, document.getElementById("conteudo"))
```

RECODE pro

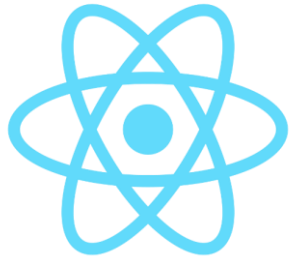
Desafio 02



Com base no exemplo anterior implemente esse modelo.

RECODE pro

Elementos de Formulários em React



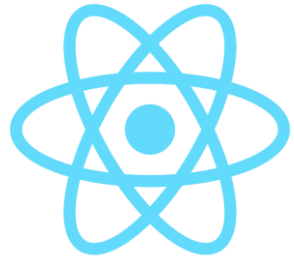
EXEMPLO TAG SELECT

```
class FlavorForm extends React.Component {
  constructor(props) {
    super(props);
    this.state = {value: 'coco'};

    this.handleChange = this.handleChange.bind(this);
    this.handleSubmit = this.handleSubmit.bind(this);
  }

  handleChange(event) {
    this.setState({value: event.target.value});
  }

  handleSubmit(event) {
    alert('Seu sabor favorito é: ' + this.state.value);
    event.preventDefault();
  }
```

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

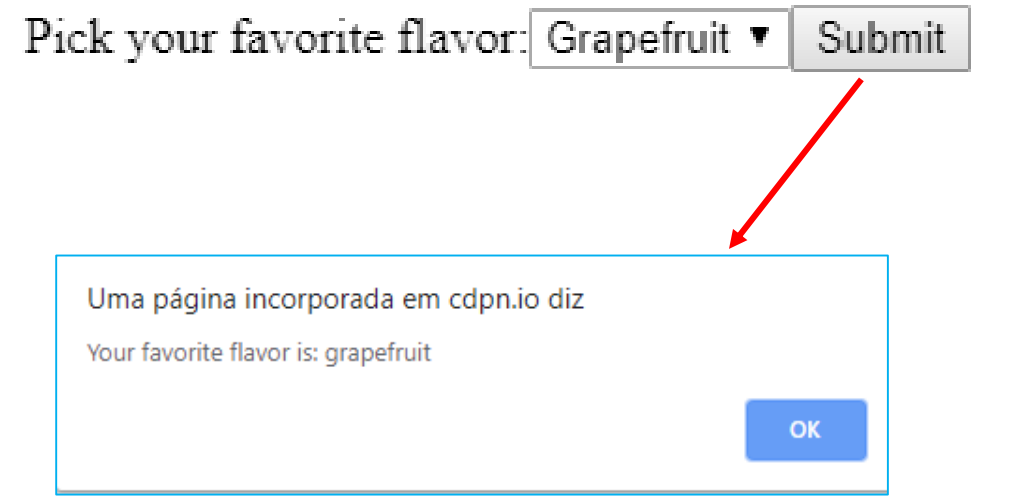
RECODE pro

Elementos de Formulários em React



EXEMPLO TAG SELECT

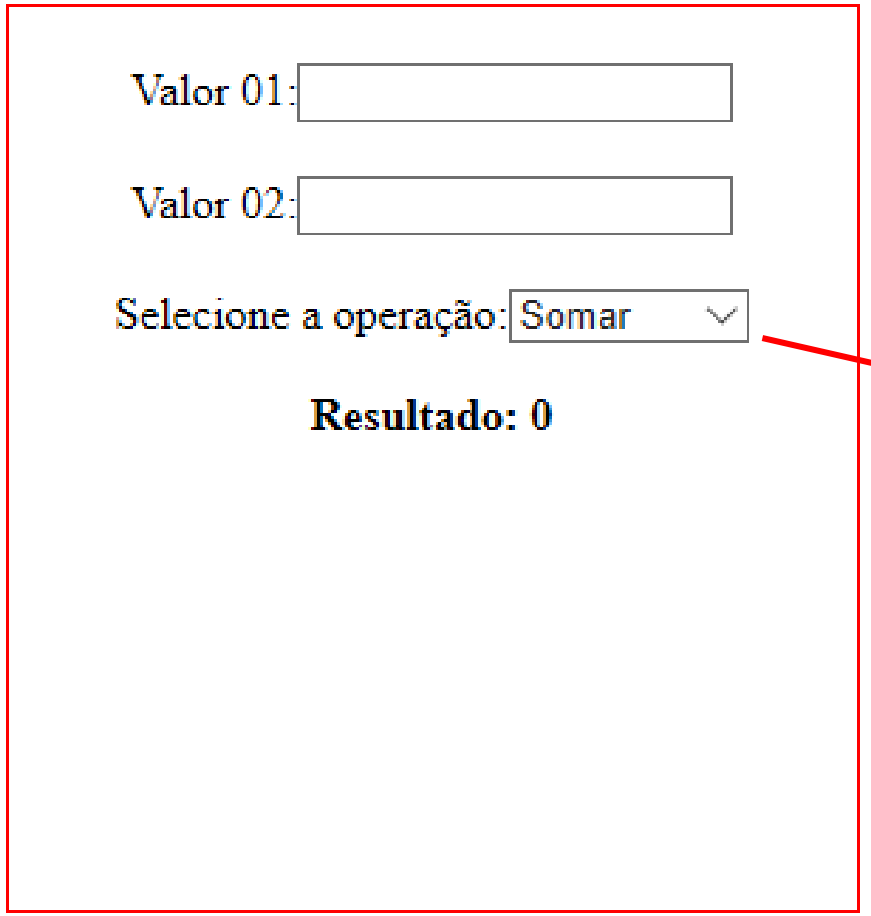
```
  render() {
    return (
      <form onSubmit={this.handleSubmit}>
        <label>
          Escolha seu sabor favorito:
          <select value={this.state.value} onChange={this.handleChange}>
            <option value="laranja">Laranja</option>
            <option value="limao">Limão</option>
            <option value="coco">Coco</option>
            <option value="manga">Manga</option>
          </select>
        </label>
        <input type="submit" value="Enviar" />
      </form>
    );
  }
}
```

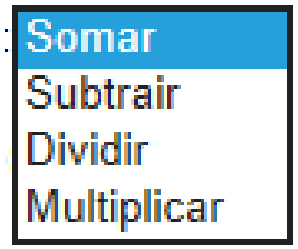


Fonte: https://pt-br.reactjs.org/docs/forms.html

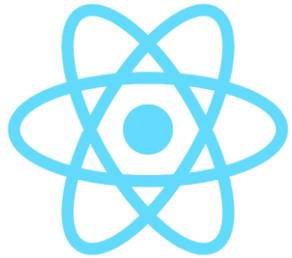
RECODE pro

Desafio 03





Com base no exemplo anterior implemente esse modelo.



Para ir além!! Consulte os seguintes endereços para conhecer outros controles de formulários usando JSX

https://pt-br.reactjs.org/docs/forms.html

https://www.w3schools.com/react/react_forms.asp