



**DATAPREV**



UMA BIBLIOTECA JAVASCRIPT PARA CRIAR INTERFACES DE USUÁRIO

Ricardo Glodzinski  
Analista de Tecnologia da Informação

Tiago de Andrade Freire  
Analista de Tecnologia da Informação







# AULA 2

## PLANEJAMENTO

- Listas e Chaves
- Forms
- Elevando o estado
- Composição vs Herança
- Pensando do jeito React
- Atividades assíncronas: exercícios

# LISTAS E CHAVES

TRANSFORMANDO ARRAYS EM LISTAS DE ELEMENTOS REACT

- Percorrer o array e retornar um elemento React para cada item.
- Definir uma key para cada elemento.
- A key deve ser um valor que identifique de forma ÚNICA cada elemento.
- É recomendado o uso de IDs estáveis, como uma PK de um registro em um banco de dados relacional.
- É possível usar o índice do elemento no array, mas não é recomendado pois o item pode mudar de posição.

# LISTAS E CHAVES

TRANSFORMANDO ARRAYS EM LISTAS DE ELEMENTOS REACT

```
import React from 'react';

const NumberList = props => {

  const ListItem = props => (<li>{props.value}</li>);

  const items = props.items;

  const listItems = items.map(item =>
    <ListItem key={item.id} value={item.value} />
  );

  return (
    <ul>
      {listItems}
    </ul>
  );
};

export default NumberList;
```



Hands on!

# LISTAS E CHAVES

TRANSFORMANDO ARRAYS EM LISTAS DE ELEMENTOS REACT

- As chaves devem ser únicas apenas entre elementos irmãos, não sendo necessário serem únicas globalmente.
- As chaves não são passadas para o elemento. Caso o valor seja necessário nele é preciso passar explicitamente.



# FORMULÁRIOS

## COMPONENTES CONTROLADOS E NÃO CONTROLADOS

- Em HTML, elementos de formulário como `<input>`, `<textarea>` e `<select>` normalmente mantêm seu próprio estado e o atualiza baseado na entrada do usuário. **[NÃO CONTROLADOS]**
- Em React, o estado mutável é normalmente mantido na propriedade `state` dos componentes e atualizado apenas com `setState()`. **[CONTROLADOS]**
- Podemos combinar os dois fazendo o estado React ser a “única fonte da verdade”.

# FORMULÁRIOS

## COMPONENTES CONTROLADOS E NÃO CONTROLADOS

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

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

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

  handleSubmit(event) {
    alert('Um nome foi enviado: ' + this.state.value);
    event.preventDefault();
  }

  render() {
    return (
      <form onSubmit={this.handleSubmit}>
        <label>
          Nome:
          <input type="text" value={this.state.value} onChange={this.handleChange} />
        </label>
        <input type="submit" value="Enviar" />
      </form>
    );
  }
}
```



Hands on!



# FORMULÁRIOS

<textarea>

- Em HTML, o texto de um elemento <textarea> é definido por seus filhos:

```
<textarea>  
  Apenas algum texto em uma área de texto  
</textarea>
```

- Em React, em vez disso, o <textarea> usa um atributo value.

```
<textarea value={this.state.value} onChange={this.handleChange} />
```

# FORMULÁRIOS

<select>

- Em HTML, <select> cria uma lista suspensa (drop-down).

```
<select>
  <option value="laranja">Laranja</option>
  <option value="limao">Limão</option>
  <option selected value="coco">Coco</option>
  <option value="manga">Manga</option>
</select>
```

- Em React, em vez de usar este atributo selected, usa-se um atributo value na raiz da tag select.

```
<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>
```

É possível passar um array para o atributo value, permitindo que sejam selecionadas várias opções.

# FORMULÁRIOS

`<input type="file">`

- Em HTML, o `<input type="file">` permite ao usuário escolher um ou mais arquivos de seu dispositivo para serem enviados para um servidor ou manipulados por JavaScript através da File API.
- Como seu valor é de somente leitura, ele é um componente não controlado do React.

# FORMULÁRIOS

MANIPULANDO MÚLTIPLOS INPUTS

- É possível adicionar um atributo name a cada elemento e deixar a função manipuladora escolher o que fazer com base no valor de event.target.name.
- Utilizar a sintaxe ES6 nomes de propriedades computados para atualizar a chave de estado correspondente ao nome de entrada fornecido.

```
handleInputChange(event) {  
  const target = event.target;  
  const value = target.name === 'isGoing' ? target.checked : target.value;  
  const name = target.name;  
  
  this.setState({  
    [name]: value  
  });  
}
```



# FORMULÁRIOS

VALOR NULO EM UM INPUT CONTROLADO

- A especificação de uma prop value em um componente controlado impede que o usuário altere a entrada, a menos que você deseje.
- Se foi especificada uma prop value, mas o input ainda é editável, acidentalmente foi definido o value como undefined ou null.

```
ReactDOM.render(<input value="hi" />, mountNode);

setTimeout(function() {
  ReactDOM.render(<input value={null} />, mountNode);
}, 1000);
```

# FORMULÁRIOS

## INPUTS NÃO CONTROLADOS

- Na maioria das situações o recomendável é utilizar componentes controlados. Mas, se necessário, é possível deixar o controle por conta do DOM.

```
class NameForm extends React.Component {
  constructor(props) {
    super(props);
    this.handleSubmit = this.handleSubmit.bind(this);
    this.input = React.createRef();
  }

  handleSubmit(event) {
    alert('A name was submitted: ' + this.input.current.value);
    event.preventDefault();
  }

  render() {
    return (
      <form onSubmit={this.handleSubmit}>
        <label>
          Name:
          <input type="text" ref={this.input} />
        </label>
        <input type="submit" value="Submit" />
      </form>
    );
  }
}
```

# ELEVANDO O ESTADO

- Com frequência, a modificação de um dado tem que ser refletida em vários componentes.
- É recomendável, nesse caso, elevar o estado compartilhado ao elemento pai comum mais próximo.

Informe a temperatura em Celsius:

Informe a temperatura em Fahrenheit:

A água ferveria.

Informe a temperatura em Celsius:

Informe a temperatura em Fahrenheit:

A água não ferveria.



Hands on!

# COMPOSIÇÃO VS HERANÇA

- O React tem um poderoso modelo de composição.
- É recomendado o uso de composição ao invés de herança para reutilizar código entre componentes.



# COMPOSIÇÃO VS HERANÇA

CONTENÇÃO

- Alguns componentes não tem como saber quem serão seus elementos filhos.
- É recomendado, nesse caso, o uso da prop especial children. Isso permite que outros componentes passem elementos filhos no próprio JSX.

```
function FancyBorder(props) {  
  return (  
    <div className={'FancyBorder' +  
      props.color}>  
      {props.children}  
    </div>  
  );  
}
```

```
function WelcomeDialog() {  
  return (  
    <FancyBorder color="blue">  
      <h1 className="Dialog-title">  
        Bem-vindo  
      </h1>  
      <p className="Dialog-message">  
        Obrigado por visitar a nossa espaçonave!  
      </p>  
    </FancyBorder>  
  );  
}
```

# COMPOSIÇÃO VS HERANÇA

## ESPECIALIZAÇÃO

- Algumas vezes pensamos em componentes que são uma especialização de outro componente.
- Em React, é possível criar essa especialização utilizando composição.

```
function Dialog(props) {  
  return (  
    <FancyBorder color="blue">  
      <h1 className="Dialog-title">  
        {props.title}  
      </h1>  
      <p className="Dialog-message">  
        {props.message}  
      </p>  
    </FancyBorder>  
  );  
}
```

```
function WelcomeDialog() {  
  return (  
    <Dialog  
      title="Bem-vindo"  
      message="Obrigado por nos visitar!" />  
  );  
}
```

# COMPOSIÇÃO VS HERANÇA

```
class SignUpDialog extends Component {
  constructor(props) {
    super(props);
    this.handleChange = this.handleChange.bind(this);
    this.handleSignUp = this.handleSignUp.bind(this);
    this.state = { login: "" };
  }

  render() {
    const { id, show, onClose } = this.props;
    return (
      <Modal id={id} show={show} onClose={onClose} title="Programa de Exploração de Marte">
        <p>Como gostaria de ser chamado?</p>
        <input value={this.state.login} onChange={this.handleChange} />
        <button onClick={this.handleSignUp}>Enviar</button>
      </Modal>
    );
  }

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

  handleSignUp() {
    alert(`Bem-vindo a bordo, ${this.state.login}!`);
  }
}
```



Hands on!

# COMPOSIÇÃO VS HERANÇA

E A HERANÇA??

- Ao usar React no cotidiano será possível perceber: NÃO há nenhum caso onde o uso de composição não consiga resolver.
- O uso de props e composição dá toda a flexibilidade necessária para customizar o comportamento e aparência dos componentes, de uma maneira explícita e segura.
- Se for necessário reutilizar funcionalidades (não gráficas) entre componentes, é possível extrair em módulos JavaScript. Os componentes podem importar essa função, objeto ou classe sem precisar estender.



# PENSANDO DO JEITO REACT

- Uma das muitas excelentes partes do React é o modo que ele nos faz pensar sobre as aplicações enquanto as construímos.
- É possível extrair componentes a partir de uma API Rest/JSON e até mesmo um desenhado “mock” de uma determinada funcionalidade.
- Para entender melhor, vamos ver na prática:

# PENSANDO DO JEITO REACT

☐ Exibir apenas produtos em estoque

| Nome                      | Valor       |
|---------------------------|-------------|
| <b>Artigos Esportivos</b> |             |
| Bola de futebol           | R\$ 95,00   |
| Bola de basquete          | R\$ 88,90   |
| Kit Bolas de Tennis       | R\$ 29,99   |
| <b>Eletrônicos</b>        |             |
| iPod Touch                | R\$ 750,00  |
| iPhone X                  | R\$ 5399,99 |
| Nexus 7                   | R\$ 1890,99 |

Mock da Tela

```
const PRODUCTS = [  
  { category: 'Artigos Esportivos', price: 'R$ 95,00', stocked: true, name: 'Bola de futebol' },  
  { category: 'Artigos Esportivos', price: 'R$ 88,90', stocked: true, name: 'Bola de basquete' },  
  { category: 'Artigos Esportivos', price: 'R$ 29,99', stocked: false, name: 'Kit Bolas de Tennis' },  
  { category: 'Eletrônicos', price: 'R$ 750,00', stocked: true, name: 'iPod Touch' },  
  { category: 'Eletrônicos', price: 'R$ 5399,99', stocked: false, name: 'iPhone X' },  
  { category: 'Eletrônicos', price: 'R$ 1890,99', stocked: true, name: 'Nexus 7' }  
];
```

Dados retornados pela API JSON

# PENSANDO DO JEITO REACT

PASSO 1 – SEPARAR A UI EM UMA HIERARQUIA DE COMPONENTES

☐ Exibir apenas produtos em estoque

| Nome                | Valor       |
|---------------------|-------------|
| Artigos Esportivos  |             |
| Bola de futebol     | R\$ 95,00   |
| Bola de basquete    | R\$ 88,90   |
| Kit Bolas de Tennis | R\$ 29,99   |
| Eletrônicos         |             |
| iPod Touch          | R\$ 750,00  |
| iPhone X            | R\$ 5399,99 |
| Nexus 7             | R\$ 1890,99 |



- FilterableProductTable
- SearchBar
- ProductTable
- ProductCategoryRow
- ProductRow



- FilterableProductTable
- SearchBar
- ProductTable
  - ProductCategoryRow
  - ProductRow

# PENSANDO DO JEITO REACT

## PASSO 2 – CRIAR UMA VERSÃO ESTÁTICA

- Construir uma versão que recebe o seu modelo de dados e renderiza a UI, mas sem interatividade.
- Criar componentes que reutilizem outros componentes e passem dados utilizando props.
- O componente no topo da hierarquia (FilterableProductTable) receberá o modelo de dados como uma prop.



Hands on!



# PENSANDO DO JEITO REACT

PASSO 3 – IDENTIFICAR A REPRESENTAÇÃO MÍNIMA (MAS COMPLETA) DO ESTADO DA UI

- Valores que não são passados via props.
- Valores que não podem ser computados a partir de outros valores passados via props.
- Valores que sofrem alteração ao longo do tempo.



Hands on!

# PENSANDO DO JEITO REACT

PASSO 4 – IDENTIFICAR ONDE O ESTADO DEVE SER ARMAZENADO

- Identificar todo componente que renderiza alguma coisa baseado no state.
- Encontrar o componente-pai comum (um único componente acima dos outros na hierarquia que necessita do estado).
- O componente-pai comum ou algum outro acima na hierarquia deve possuir o state.



Hands on!

# PENSANDO DO JEITO REACT

PASSO 5 – ADICIONAR O FLUXO DE DADOS INVERSO

- Os elementos de formulário na base da hierarquia precisam atualizar o state em `FilterableProductTable`.
- Utilizar o `onChange` dos elementos do formulário para que executem um callback que atualize o estado de `FilterableProductTable`.



Hands on!



# Obrigado!

**Ricardo Glodzinski**

**Analista de Tecnologia da Informação**

ricardo.glodzinski@dataprev.gov.br

**Tiago de Andrade Freire**

**Analista de Tecnologia da Informação**

tiago.freire@dataprev.gov.br

Agosto de 2020



[www.facebook.com/dataprevtecnologia](https://www.facebook.com/dataprevtecnologia)



[@dataprev](https://twitter.com/dataprev)



[DATAPREV](https://www.linkedin.com/company/dataprev)