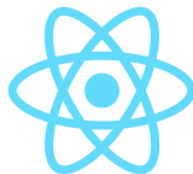


# Introdução ao React.js

Laboratório de Programação



# O que é?

- Framework Javascript desenvolvido pelo Facebook <https://pt-br.reactjs.org/>
  - concorrente do AngularJS
- Diferença em relação ao AngularJs:

“

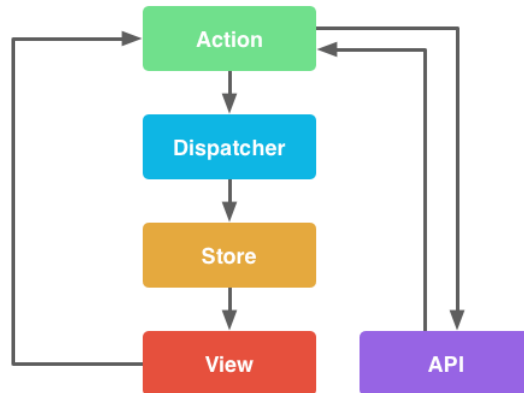
Usa o modelo Flux ao invés do MVC

”

- Tende a ser de simples aprendizado
- Doc: [React.js Getting Started](#)

# Flux

- Arquitetura desenvolvida pelo Facebook para desenvolver aplicações no lado do cliente
- Organizada em 3 elementos:
  - Dispatcher, Store, Views



# Flux



- Dispatcher: responsável por executar ações que envolvam o servidor
- Store: mantém o estado da aplicação
- Views: componentes responsáveis pela renderização de informação

“

a ideia é que o ciclo continue para manter a integridade da arquitetura como um todo

”

# + Características

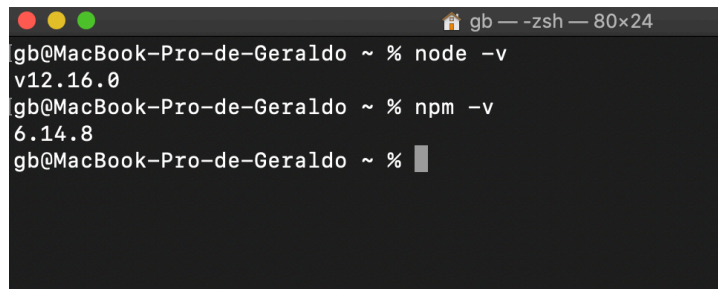
- Gerencia automaticamente seu estado e atualizações de interface
  - Quando ocorre mudança em dados, o React automaticamente renderiza os componentes ligados
- "Componentização" dos elementos de interface
  - legal poder reutilizar código
- Componentização de estruturas JSX
  - legal por poder usar herança, ...
- Single Page Application
- JSX -> extensão de javascript: DOC: [Introducing JSX](#)

# Montando o ambiente e criando o primeiro projeto

Mãos a obra!

# Pré-Requisitos:

- Node.js (precisa ser instalado)
  - Servidor para javascript
  - já vem com o gerenciador de pacotes, npm
- para testar instalação digite `node -v` num terminal

A terminal window with a dark background and light text. The window title bar shows 'gb — zsh — 80x24'. The terminal content shows the user 'gb' at the prompt 'gb@MacBook-Pro-de-Geraldo ~ %' running 'node -v' which outputs 'v12.16.0', then running 'npm -v' which outputs '6.14.8', and finally a prompt 'gb@MacBook-Pro-de-Geraldo ~ %' with a cursor.

```
gb@MacBook-Pro-de-Geraldo ~ % node -v
v12.16.0
gb@MacBook-Pro-de-Geraldo ~ % npm -v
6.14.8
gb@MacBook-Pro-de-Geraldo ~ %
```

# Pré-Requisitos:

- instale via node.js o `create-react-app`

“

`npm install -g create-react-app`

”

```
gb@MacBook-Pro-de-Geraldo ~ % node -v
v12.16.0
gb@MacBook-Pro-de-Geraldo ~ % npm -v
6.14.8
gb@MacBook-Pro-de-Geraldo ~ % npm install -g create-react-app
((...)) : fetchMetadata: sill removeObsoleteDep removing builtins@
```



# Criando a aplicação react

“

`cd` na pasta que queira criar a aplicação

```
create-react-app demo-app
```

- Atenção: aguarde ele baixar as dependências

”

```
gb@MacBook-Pro-de-Geraldo ~ % cd /Users/gb/Documents/Workspaces/labprog
gb@MacBook-Pro-de-Geraldo labprog % create-react-app demo-app

Creating a new React app in /Users/gb/Documents/Workspaces/labprog/demo-app.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...

( [REDACTED] ) ^ fetchMetadata: sill resolveWithNewModule scheduler@0.20.1 che
```

# Criando a aplicação react

```
Created git commit.

Success! Created demo-app at /Users/gb/Documents/Workspaces/labprog/demo-app
Inside that directory, you can run several commands:

  npm start
    Starts the development server.

  npm run build
    Bundles the app into static files for production.

  npm test
    Starts the test runner.

  npm run eject
    Removes this tool and copies build dependencies, configuration files
    and scripts into the app directory. If you do this, you can't go back!

We suggest that you begin by typing:

  cd demo-app
  npm start

Happy hacking!
```

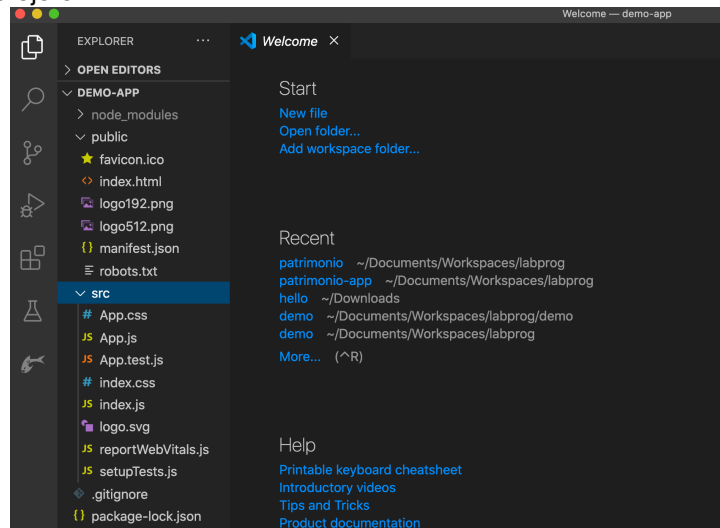
# Colocando para rodar:

- Entre na pasta do projeto: `cd demo-app`
- Use `npm start`
- Ele disponibiliza um endereço web. Por padrão <http://localhost:3000/>



# Abrindo no VS Code

- Basta abrir a pasta do projeto
  - Navegue pelo projeto!



# Criando o primeiro componente

hello world

# Componenete no React

- `function`
- inclua `export default` para publicar
- `return` indica como o componente será renderizado

```
export default function Hello() {  
  return (  
    <div>  
      <label>Hello!</label>  
    </div>  
  );  
}
```

# Componenete no React

- Adicione o componente no `index.js`

DE:

```
root.render(  
  <React.StrictMode>  
    <App />  
  </React.StrictMode>  
);
```

Para:

```
root.render(  
  <React.StrictMode>  
    <Hello />  
  </React.StrictMode>  
);
```

# Agora basta olhar o index.html no endereço

```
“  
localhost:3000  
”
```

- lembre-se que vc tem que ter efetuado `npm start` antes



# Variáveis

- Você pode declarar variáveis

```
const user = {  
  name : 'Geraldo'  
}
```

- e usá-las dentro do código com { variável }

```
const user = {  
  name : 'Geraldo'  
}  
export default function Hello() {  
  return (  
    <div>  
      <label>Hello {user.name}!</label>  
    </div>  
  );  
}
```

# Lidando com listas

- basicamente a lista é construída como um vetor JSON

```
const producao = [  
  {id: 1, titulo: 'Artigo 1'},  
  {id: 2, titulo: 'Artigo 2'},  
  {id: 3, titulo: 'Artigo 3'},  
];
```

- use `map` para iterar nos elementos e construir o componente:

```
export default function Listas() {  
  const lstItens = producao.map (producao =>  
    <li key={producao.id}>  
      {producao.titulo}  
    </li>  
  )  
  return (  
    <ul>{lstItens}</ul>  
  );  
}
```

## Exemplo similar para Tabela

```
const producao = [
  {id: 1, titulo:'Artigo 1'},
  {id: 2, titulo:'Artigo 2'},
  {id: 3, titulo:'Artigo 3'},
];

export default function Tabela() {
  const lstItens = producao.map (producao =>
    <tr>
      <td>{producao.id}</td>
      <td>{producao.titulo}</td>
    </tr>
  )
  return (
    <table>
      <tr>
        <th>ID</th>
        <th>Titulo</th>
      </tr>
      {lstItens}
    </table>
  );
}
```

## Tabela com click

- Para capturar o evento do click, vamos criar uma função `handleClick`

```
function handleClick(indice) {  
  alert('Você clicou!');  
}
```

- No botão basta ser:

```
<button onClick={handleClick}>Botão</button>
```

- Não se pode passar parâmetro dessa maneira, senão fica num loop

- **Mas, e se quiser receber um parâmetro? Use uma função arrow** `() =>`

```
function handleClick(indice) {  
  alert('Você clicou no ID=' + indice);  
}  
...  
<button onClick={ () => handleClick(producao.id)}>Ação</button>  
...
```

## Tabela com click

```
const producao = [
  {id: 1, titulo: 'Artigo 1'}, {id: 2, titulo: 'Artigo 2'}, {id: 3, titulo: 'Artigo 3'},
];

export default function Click() {
  function handleClick(indice) {
    alert('Você clicou no ID=' + indice);
  }

  const lstItens = producao.map (producao =>
    <tr>
      <td>{producao.id}</td>
      <td>{producao.titulo}</td>
      <td>
        <button onClick={ () => handleClick(producao.id)}>Ação</button>
      </td>
    </tr>
  )
  return (
    <table> <tr><th>ID</th><th>Título</th><th>Click</th> </tr>
      {lstItens}
    </table>
  );
}
```

# State

- O react fornece mecanismo de estado e gerenciamento de variáveis
- Para utilizar você precisa:
  - importar `import { useState } from 'react';`
  - e declarar o estado `const [count, setCount] = useState(0);`

```
import { useState } from 'react';  
  
export default function Estado() {  
  const [num1, setNum1] = useState(0);  
  const [num2, setNum2] = useState(0);  
  const [soma, setSoma] = useState(0);
```

- `num1`, `num2`, ... são estados e podem ser obtidos por `{num1}`
- `setNum1`, `setNum2`, ... são funções para ajuste de valor
- `useState` é uma função `hook` definida no React. Outros: [API Hooks](#)

# State

- Você pode associar o estado a um componente:

```
<input type='text'  
  onChange={e => setNum1(e.target.value)}></input>
```

- Use a função `arrow` para coletar o evento `e` ao modificar o valor via `onChange`
- `setNum1` é a função disponibilizada pelo estado acima

# State

- Podemos adicionar ações como subfunções:

```
function somar() {  
  setSoma(Number(num1)+Number(num2));  
}
```

- `setSoma` é a atualização do estado
- `Number(num1)` converter string -> número
- Quando a função é chamada no react, ela não recebe parâmetro

```
<button onClick={somar}>Somar</button>
```



## State - Completo

```
import { useState } from 'react';

export default function Estado() {
  const [num1, setNum1] = useState(0);
  const [num2, setNum2] = useState(0);
  const [soma, setSoma] = useState(0);

  function somar() {
    setSoma(Number(num1)+Number(num2));
  }

  return (
    <div>
      <label>Num1:</label>
      <input type='text'
        onChange={e => setNum1(e.target.value)}></input>
      <label>Num2:</label>
      <input type='text'
        onChange={e => setNum2(e.target.value)}></input>
      <br></br>
      <button onClick={somar}>Somar</button>
      <br></br>
      <label>Resultado:{soma}</label>
    </div>
  );
}
```

# Props

- É possível passar informações para o componente por `props`

```
function Campo ({ value }) {  
  return (  
    <>  
      <label>{value}</label>  
      <br></br>  
    </>  
  );  
}
```

- onde é informado via:

```
<Campo value = "Digite algo:" />
```

# Props

```
import { useState } from 'react';

export default function Props() {
  const [texto, setTexto] = useState("");

  return (
    <div>
      <Campo value = "Digite algo:"/>
      <input type='text'
        onChange={e => setTexto(e.target.value)}></input>
      <Campo value={texto}/>
    </div>
  );
}

function Campo ({ value }) {
  return (
    <>
      <label>{value}</label>
      <br></br>
    </>
  );
}
```

Exemplo `Thinking in React`

# Thinking in React

- Exemplo de filtragem de conteúdo numa tabela

<input type="text" value="Search..."/>	
<input type="checkbox"/> Only show products in stock	
Name	Price
Fruits	
Apple	\$1
Dragonfruit	\$1
Passionfruit	\$2
Vegetables	
Spinach	\$2
Pumpkin	\$4
Peas	\$1

- Disponível completo em: [Link](#)

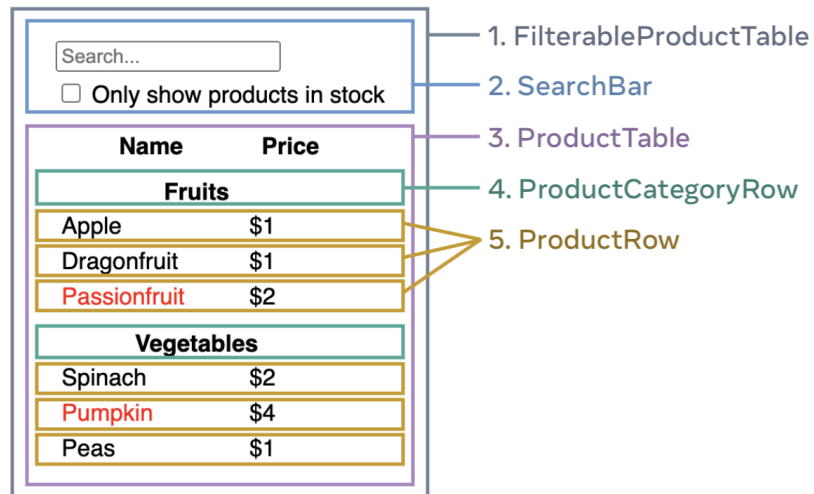
# Passos:

Dado que recebeu o seguinte JSON de resultado:

```
[
  { category: "Fruits", price: "$1", stocked: true, name: "Apple" },
  { category: "Fruits", price: "$1", stocked: true, name: "Dragonfruit" },
  { category: "Fruits", price: "$2", stocked: false, name: "Passionfruit" },
  { category: "Vegetables", price: "$2", stocked: true, name: "Spinach" },
  { category: "Vegetables", price: "$4", stocked: false, name: "Pumpkin" },
  { category: "Vegetables", price: "$1", stocked: true, name: "Peas" }
]
```

- 1) Transforme a interface em componentes
- 2) Construa uma versão estática
- 3) Defina a representação do estado
- 4) Construa as funcionalidades do estado nos componentes

# 1) UI para componentes



Hierarquia: FilterableProductTable (SearchBar, ProductTable (ProductCategoryRow, ProductRow))

## 2) Versão Estática

```
function ProductCategoryRow({ category }) {  
  return (  
    <tr>  
      <th colspan="2">  
        {category}  
      </th>  
    </tr>  
  );  
}
```



## 2) Versão Estática

```
function ProductRow({ product }) {  
  const name = product.stocked ? product.name :  
    <span style={{ color: 'red' }}>  
      {product.name}  
    </span>;  
  
  return (  
    <tr>  
      <td>{name}</td>  
      <td>{product.price}</td>  
    </tr>  
  );  
}
```

## 2) Versão Estática

```
function ProductTable({ products }) {  
  const rows = [];  
  let lastCategory = null;  
  
  products.forEach((product) => {  
    if (product.category !== lastCategory) {  
      rows.push(  
        <ProductCategoryRow  
          category={product.category}  
          key={product.category} />  
      );  
    }  
    rows.push(  
      <ProductRow  
        product={product}  
        key={product.name} />  
    );  
    lastCategory = product.category;  
  });  
  ...  
}
```

## 2) Versão Estática

```
return (  
  <table>  
    <thead>  
      <tr>  
        <th>Name</th>  
        <th>Price</th>  
      </tr>  
    </thead>  
    <tbody>{rows}</tbody>  
  </table>  
)  
);  
}
```

## 2) Versão Estática

```
function SearchBar() {  
  return (  
    <form>  
      <input type="text" placeholder="Search..." />  
      <label>  
        <input type="checkbox" />  
        {' '}  
        Only show products in stock  
      </label>  
    </form>  
  );  
}
```

## 2) Versão Estática

```
function FilterableProductTable({ products }) {  
  return (  
    <div>  
      <SearchBar />  
      <ProductTable products={products} />  
    </div>  
  );  
}
```

```
export default function App() {  
  return <FilterableProductTable products={PRODUCTS} />;  
}
```

### 3) Definindo os estados

- Quais dados são estado ou propriedades?
  - Estados quando sofrem modificações
- Estados relacionados a UI
  - O texto da busca
  - O valor do checkbox

## 4) Implementando os estados

- Os estados ficam em `FilterableProductTable` por ser hierarquicamente o agregador

```
function FilterableProductTable({ products }) {  
  const [filterText, setFilterText] = useState('');  
  const [inStockOnly, setInStockOnly] = useState(false);  
  ...  
}
```

## 4) Implementando os estados

- Os estados podem ser passados por `props` para os demais componentes:

```
function FilterableProductTable({ products }) {  
  const [filterText, setFilterText] = useState('');  
  const [inStockOnly, setInStockOnly] = useState(false);  
  
  return (  
    <div>  
      <SearchBar  
        filterText={filterText}  
        inStockOnly={inStockOnly}  
        onFilterTextChange={setFilterText}  
        onInStockOnlyChange={setInStockOnly} />  
      <ProductTable  
        products={products}  
        filterText={filterText}  
        inStockOnly={inStockOnly} />  
    </div>  
  );  
}
```



## 4) Implementando os estados

- Você deve atualizar os subcomponentes para refletir a nova situação

```
function ProductTable({ products, filterText, inStockOnly }) {  
  const rows = [];  
  let lastCategory = null;  
  
  products.forEach((product) => {  
    if (product.name.toLowerCase().indexOf(filterText.toLowerCase()) === -1) {  
      return;  
    }  
    if (inStockOnly && !product.stocked) {  
      return;  
    }  
    if (product.category !== lastCategory) {  
      rows.push(  
        <ProductCategoryRow category={product.category} key={product.category} />;  
      )  
      rows.push(  
        <ProductRow product={product} key={product.name} />;  
        lastCategory = product.category;  
      );  
    }  
  });  
  ....  
}
```

## 4) Implementando os estados

- Você deve atualizar os subcomponentes para refletir a nova situação

```
function SearchBar({
  filterText,
  inStockOnly,
  onFilterTextChange,
  onInStockOnlyChange
}) {
  return (
    <form>
      <input
        type="text"
        value={filterText} placeholder="Search..."
        onChange={(e) => onFilterTextChange(e.target.value)} />
      <label>
        <input
          type="checkbox"
          checked={inStockOnly}
          onChange={(e) => onInStockOnlyChange(e.target.checked)} />
        {' '}
        Only show products in stock
      </label>
    </form>
  );
}
```