React Básico

Material das Aulas

- Slides

Boas Vindas

Documentação

- React

O que veremos nas Aulas

- Por que React
- Preperando ambiente
- Entendendo a arquitetura do React
- Primeiros passos e conceitos básicos
- Consumindo dados de uma API com Axios
- Utilizando React Router para criar rotas
- Boas práticas: organização e estilo
- Entendendo, na prática, o poder da programação declarativa.

Por que React?

Material Complementar

- Código Fonte sobre React

Motivos para usar React

- Lib para criação de interfaces
- Utilizada para a construção de Single Page Applications
- Podemos chamar de framework, devido a seu ecossistema: ReactJS React Native, Redux, Webpack, React Router DOM.
- Tudo fica dentro do JS (elementos visuais, lógica e estilo).

JSX

- Versão de HTML dentro do JavaScript

Styled Components

- Estilização usando JavaScript.

Vantagens

- Organização do código
  - Dividir nosso app/código em blocos específicos
  - O funcionamento de um componente não interfere no outro
  - Se um componente é removido, o resto continua funcionando
- Divisão de responsabilidades
  - Back-end: regras de negócio
  - Front-end: interface
- Programação declarativa
  - Com JS puro temos programação imperativa, damos ordem, dizemos como.
  - Dizemos ao código o que queremos, ele se encarrega do resto, não dizemos como, só dizemos o que queremos.

Preparando o Ambiente

Material

- Documentação React

Instalar

- NodeJS (LTS)
  - Menos Bugs
- Current
  - Mais recente e em desenvolvimento e com bugs.
- Escolha o sistema operacional e instale, ou pelo gerenciador de pacotes ou pelo instalador.

node -v

- Vê qual a versão do Node instalado

NPM ou Yarn / npm install -g yarn / npm install -g

- Instale o gerenciador de pacotes chamado NPM ou Yarn.

Instale o VSCode.

Instale o HyperX.

npx create-react-app | npx create-vite-app @latest

- Comando pronto desenvolvido pelo Facebook para configuração de projeto, todo configurado
- Dizemos o nome do nosso projeto e irá construir nossa aplicação em React.

yarn run dev or npm start

- Nossa aplicação em React estará rodando!

Fundamentos do React

npm start

- Para rodar o projeto

acesso o localhost!

- Agora podemos ver a aplicação.

Como funciona a renderização do HTML

- O HTML é todo renderizado através da `div` com ID `root` que está dentro da public no index.html.
- Dentro do App.js estão os componentes que iremos criar e importar para dentro dele.
- Ou seja, criamos o componente, importamos no App.js.
- Lá dentro do index.js tudo que está no `App` é enviado para a `div` de id `root`

JSX

- Chamamos de JSX todo HTML dentro do JavaScript

Componentes funcionais

- No React para criamos nosso componente precisamos utilizar uma função que retorna um conteúdo HTML
- Esse conteúdo HTML dentro do JavaScript chamamos de `JSX.`

export function App() em vez de export default App

- Na hora de importarmos no index.js importamos com chaves `{ App }`

className em vez de class

- Usamos className, pois class é uma palavra reservada do JavaScript, e como estamos usando HTML dentro do JS, isso pode dar conflito.

Propriedade

- São como atributos do HTML, mas passados para nosso componente.

```
ReactDOM.render(
  <App title="Hello Gama Academy" />,
  document.getElementById('root')
);
```

Props

- Pegamos essa propriedade através de `PROPS`
- Todo componente tem esse parâmetro disponível
- Temos que colocar um `<div>` ou `<>` embrulhando, quando tivermos mais de um elemento HTML ou componente.

```
function App(props) {
  return (
    <div>
      <h1>{ props.title } { props.user }</h1>
      <input name="usuario" id="usuario" className="usuarioInput" placeholder="Usuário" />
    </div>
  );
}

export default App;
```

Fragment

- É uma tag vazia, sem nada, ele substitui uma `div` no contexto do React.
- Serve apenas para embrulhar os nossos componentes ou elementos HTML.
- O ideal é usar o `<>`, para evitar usar um elemento a mais.

Estado

- Usamos Hooks para controlar Estados no React
- É uma API que controla a manipulação de Estados.

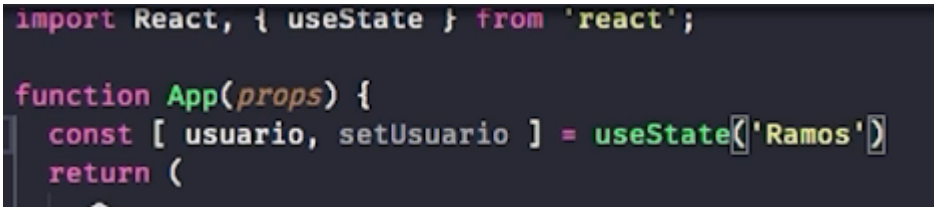
useState

- Permite modificar e citar estados.
- Precisamos importar antes.
- Ele é uma função que retorna uma array
  - Na primeira posição ele retorna o valor do Estado

- Na segunda posição ele retorna uma função, que é usada para setar esse valor

→ Por convenção temos `[state, setState]`

Exemplo



Consumindo dados de uma API

Estado

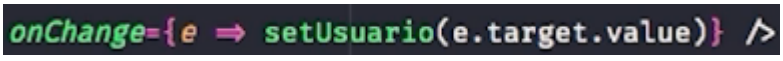
- É a única fonte de dados do React
- Teremos duas fontes de dados
  - Input
- Preicsamos ter componentes controlados
- Vamos ter o valor através dos `states.`

Eventos no React

- No React colocamos em **camelCase**
- Vamos capturar qualquer alteração no input

`onChange { e => console.log(e.target.value)}`

- Agora vamos alterar esse evento para receber os dados do useState



- Armazenado o valor do input no estado.

```
function App(props)
const[usuario,setUsuario]useState("");
return(
<>
<input className="usuarioInput" placeholder="Usuário" value={usuario}
  onChange={ e => setUsuario(e.target.value)} />
```

```
  <button type="button"> </button>

  </>
    )
  }
  export default App;
```

```
return (
  <>
    <p>{ usuario }</p>
    <input className="usuarioInput" placeholder="Usuário" value={usuario} onChange={e => setUsuario(e.target.value)}
```

const usuario: string

Buscando dados na API do github com Axios

- Usamos o `onClick` e a função entre chaves.
- Usamos o padrão `handleOqueVamosFazer`
- Quando clicarmos no button, pesquisamos o usuário

```
return (
  <>
    <input className="usuarioInput" placeholder="Usuário" value={usuario} onChange={e => setUsuario(e.target.value)}
    <button type="button" onClick={handlePesquisa}>Pesquisar</button>
  </>
);
```

- Fazemos requisição para fazer API

```
npm install axios
```

```
import axios from 'axios'
```

criando a função de buscar os repositórios

```
function handlePesquisa() {
  axios.get(`https://api.github.com/users/${usuario}/repos`).then(response => console.log(response.data));
}
```

Por padrão o React sempre importa o index.ts, então não precisamos colocar o formato

Instalando e configurando React-Router-DOM

Documentação

- Consultar documentação, pois a forma de uso dessa lib pode ter mudado

O que é o React-Router-DOM?

- Permite a reconstrução de SPA
- Mudamos de página, sem dar o refresh da página
- Termoes uma página para buscar os dados.
- Outra página para listar o repositório

Repositories.js

- Vai listar todos os repositórios
- Usando named exports.

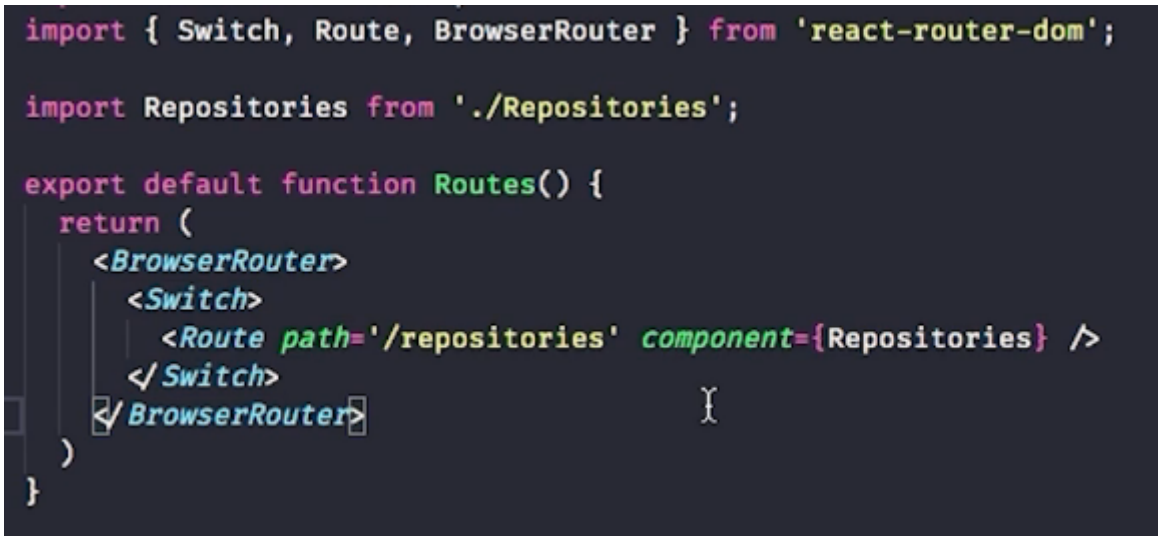
React Router DOM

- Responsável por fazer o sistema de Roteamento da nossa aplicação

npm install react-router-dom@5.2.0

configurando nossas rotas | routes.js

- BrowserRouter é o container principal do RRD
- O `Switch` é o container que irá armazenar a nossa rota.
- No Route passamos o caminho que queremos que aparece na URL e no componente o nome do nosso componente.

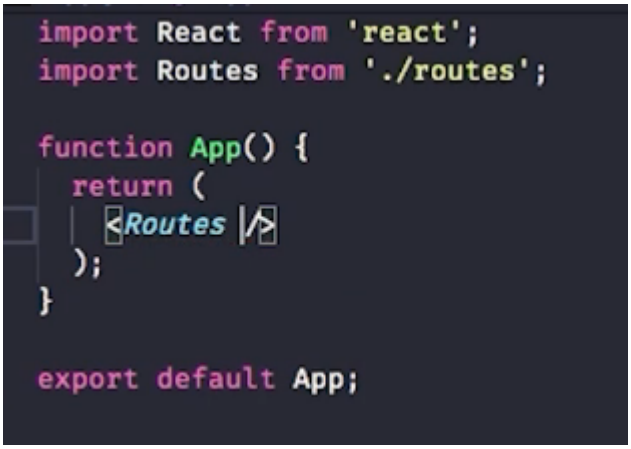


Importando as rotas no componente App.

Colocando o arquivo de rotas dentro do App

- Vamos refatorar nosso código e criar outro arquivo chamado Home.js

App.js



```
import React from 'react';
import Routes from './routes';

function App() {
  return (
    <Routes />
  );
}

export default App;
```

Routes.ts

- Definindo a Rota raiz com o componente Home.js
- Usamos exact pra não retornar a home, sempre que encontrar a `/`

```
import { Switch, Route, BrowserRouter } from 'react-router-dom';

import Repositories from './Repositories';
import Home from './Home';

export default function Routes() {
  return (
    <BrowserRouter>
      <Switch>
        <Route path='/' exact component={Home} />
        <Route path='/repositories' component={Repositories} />
      </Switch>
    </BrowserRouter>
  )
}
```

Home.js

- Iremos tirar tudo que é referente a nossa pesquisa de repositório do App e colocar nele

```jsx
import React, { useState } from 'react';
import axios from 'axios';

function App(props) {
  const [ usuario, setUsuario ] = useState('');

  function handlePesquisa() {
    axios.get(`https://api.github.com/users/${usuario}/repos`).then(response
  }

  return (
    <>
      <input className="usuarioInput" placeholder="Usuário" value={usuario}
      <button type="button" onClick={handlePesquisa}>Pesquisar</button>
    </>
  );
}

export default App;
```