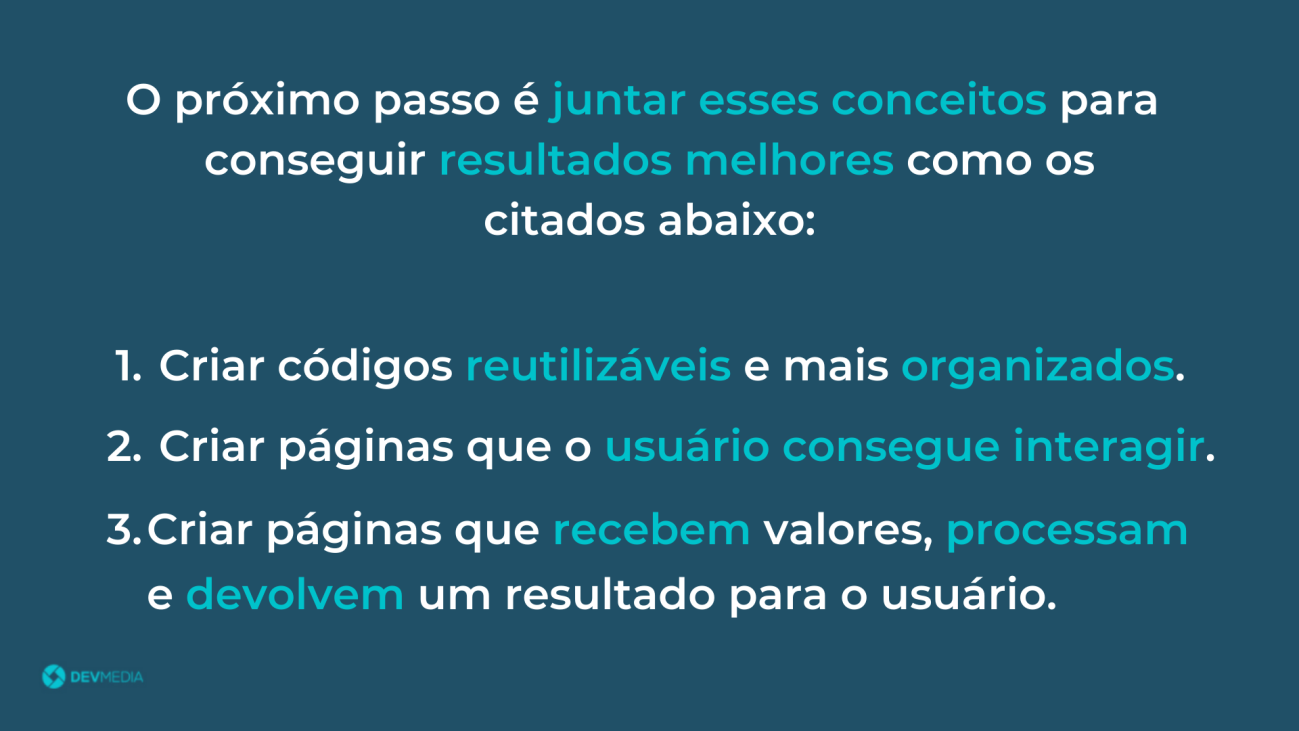
Guia de React

Iniciar uma aplicação REACT

npx create-react-app “nome da pasta”

React é uma biblioteca JavaScript para a criação de interfaces de usuário e, nesse contexto, uma das tecnologias mais utilizadas pelo mercado



Reutilizar códigos acelera o processo de desenvolvimento de uma página já que o programador escreve menos código.

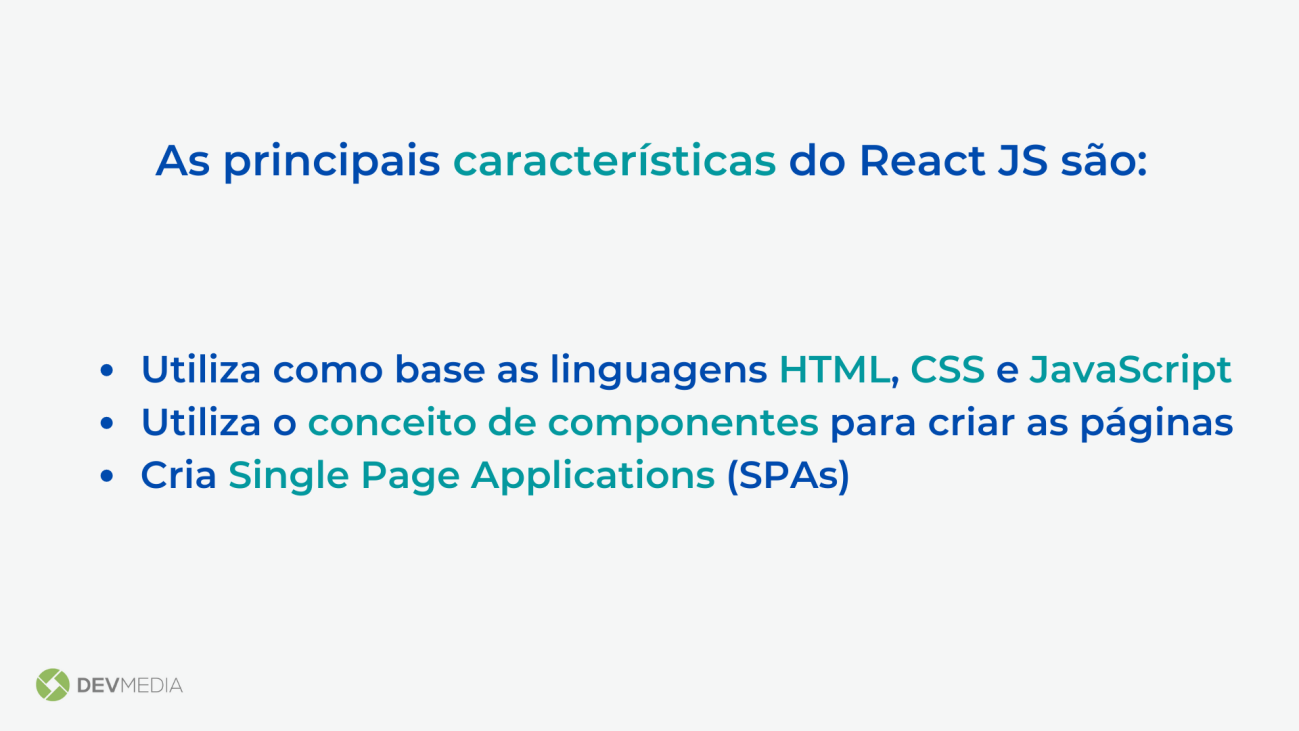
Aprender a criar códigos reutilizáveis, páginas interativas e que recebem, processam e devolvem um valor, é o seu próximo passo na carreira como programador front-end. Você aprenderá a aplicar esses conceitos utilizando a biblioteca React JS que é a tecnologia web com maior número de vagas no mercado.

As principais características do React JS são:

Utiliza como base as linguagens HTML, CSS e JavaScript

Utiliza o conceito de componentes para criar as páginas

Cria Single Page Applications (SPAs)



O React JS foi criado pelo Facebook e adotado pela comunidade de programadores front-end que cresce a cada dia. Com isso, diversas empresas adotaram o React JS em seus projetos e aprender a utilizá-lo e te deixa mais próximo de entrar no mercado como programador front-end.

npx create-react-app minha-pagina

Sempre que precisar criar uma aplicação React utilize o mesmo comando 'npx create-react-app'. A única palavra que muda é o nome da aplicação que é inserida no final deste comando.

npm start

No React JS, exibir o conteúdo na tela significa criar uma função que retorna elementos HTML. Essa função é chamada de componente.

import './App.css';

function App() {

return (

<div>

<h1>Hello world</h1>

<p>Com React JS</p>

<p className="autor">Criado por Mateus Correa</p>

</div>

);

}

export default App;

O que são componentes

Componentes são, basicamente, elementos visuais exibidos na tela para o usuário.

Por exemplo, em uma página Web podemos dizer que um botão é um componente, um item de menu é um componente, um campo de edição de dados é um componente e assim por diante.

Além disso, a própria página Web, em si, também é um componente (que contém componentes filhos).

Uma página construída em React é um conjunto de componentes.

React cria a interface do usuário utilizando componentes.

Precisamos saber como criar, importar e exportar componentes para trabalhar com React.

Ou seja, a mecânica fundamental de programação React funciona assim:

Primeiro você vai codificar o seu componente (por exemplo, um botão na tela);

Com o componente codificado, o segundo passo é exportar esse componente - o que significa deixá-lo exposto para que possa ser usado na sua página;

Por último você vai ‘importar’ o componente previamente exposto e dessa forma apresentá-lo na tela.

Em termos mais técnicos:

Criar o componente filho;

Exportar o componente filho;

Importar o componente filho dentro do componente pai (neste exemplo na própria página).

Mais para frente no curso veremos, na prática, como realizar estes três passos do Slide 1. Neste momento o mais importante é você entender a mecânica da programação, pois esses passos serão repetidos muitas e muitas vezes sempre que você criar um app React.

O React consiste na composição de componentes, ou seja, um componente pode ser composto de componentes menores.

O componente App é o principal da aplicação, devendo conter todos os outros componentes dentro dele.

Um componente pode ser criado a partir de outros componentes.

A sintaxe que usamos dentro do return se chama JSX e é bem parecida com HTML. Você não precisa se preocupar em se tornar um expert em JSX, pois a sintaxe é facilmente aprendida durante a programação em React - e é muito similar ao HTML tradicional.

Usamos a palavra chave 'default' para exportar um componente padrão.

Devemos seguir o padrão de escrita case sensitive utilizado no arquivo do componente. Ou seja, usamos nesse exemplo a primeira letra maiúscula, seguindo o padrão utilizado. A convenção é criar componentes com a primeira letra em maiúsculo.

State Hook

O State Hook (useState), basicamente nos permite criar uma variável que ao mudar de valor, automaticamente, será alterada na tela do usuário.

Por exemplo, se a variável nomeProduto for alterada, precisamos que o usuário veja o novo nome do produto - o React permite que isso aconteça de forma automática - ou - reativa, como o nome da tecnologia implica.

Evento onClick

Ao clicar em um botão é comum que algo aconteça. Para chamarmos esta ação utilizamos o evento onClick.

Neste exemplo, quando o usuário clicar no botão, o código associado ao evento onClick será executado (a geração do número aleatório).

Esse é o conceito da programação de eventos - associamos um código a um evento, ou trigger. Quando este evento/trigger for disparado pela aplicação (ou pelo usuário) o código associado será executado.

Com state Hook conseguimos atualizar a tela automaticamente ao alterar o valor de uma variável.

import React, { useState } from 'react';

import './estilo.css';

export default function PaginaInicial() {

const [ numeroAleatorio, setNumeroAleatorio ] = useState(1);

return(

<div className="conteudo-centralizado">

<h3>Número aleatório:</h3>

<h1>{ numeroAleatorio }</h1>

<div className='area-botao'>

<label>

Click no botão abaixo para gerar um número aleatório:

</label>

<button>

Gerar número

</button>

</div>

</div>

);

}

Importante:

Funções não possuem estado, por isso ao serem executadas suas variáveis deixam de ser acessíveis. Criar uma variável de estado utilizando o Hook useState significa preservá-la e mantê-la acessível.

Hooks são como gatilhos: mecanismo que permite ao React reagir a um momento específico. Em nosso exemplo, usamos Hooks para renderizar a tela automaticamente toda vez que a variável é modificada, e por isso a chamamos de variável de estado.

Aqui utilizamos o Hook useState para criar uma variável de estado.

const [NOME\_DA\_VARIAVEL\_DE\_ESTADO, FUNCAO\_PARA\_USO\_INTERNO] =

useState(VALOR\_INICIAL\_DA\_VARIAVEL);

function gerarNumero() {

const novoNumero = Math.floor(Math.random() \* (100-1) + 1);

setNumeroAleatorio(novoNumero);

}

import './App.css';

import { useState } from "react";

export default function App() {

const [ novoEstilo, setNovoEstilo ] = useState('bg-vermelho');

const alterarEstilo = () => {

setNovoEstilo('bg-azul');

}

return (

<div className='display'>

<div className='container'>

<h2>Alterando estilo do elemento</h2>

<div className={ novoEstilo }></div>

<p>Clique no botão para alterar o estilo do elemento acima.</p>

<button onClick={ alterarEstilo }>Alterar</button>

</div>

</div>

);

}

import './App.css';

import { useState } from "react";

export default function App() {

const [ novaCor, setNovaCor ] = useState('bg-vermelho');

const alterarCor = () => {

if (novaCor === 'bg-vermelho'){

setNovaCor('bg-azul');

} else {

setNovaCor('bg-vermelho')

}

}

return (

<div className='display'>

<div className='container'>

<h2>Altere o estilo do elemento</h2>

<div className={novaCor}></div>

<p>Clique no botão abaixo para alterar a cor do elemento.</p>

<button onClick={alterarCor}>Clique aqui</button>

</div>

</div>

);

}

import React, { useState } from 'react';

export default function PaginaPrincipal() {

const [ mensagem, setMensagem ] = useState("Nenhuma mensagem encontrada");

return (

<h1>{mensagem}</h1>

);

}

<button onClick = { patrocinarClube }>Patrocinar</button>

import React from 'react';

import './style.css';

import { Link } from 'react-router-dom';

const PaginaInicial = () => (

<div>

<h1>Escolha seu caminho em JavaScript</h1>

<div className="conteudo-opcoes">

<Link className="opcao-front-end" to='/front-end'>Front-end</Link>

<Link className="opcao-back-end" to='/back-end'>Back-end</Link>

</div>

</div>

);

export default PaginaInicial;

## Rotas

### O que são Rotas?

Uma rota é basicamente a associação entre uma URL e um componente. Para criarmos rotas no React utilizamos a biblioteca react-router-dom.



A principal funcionalidade do React no front-end é criar SPAs. Uma [SPA](https://www.devmedia.com.br/ja-ouviu-falar-em-single-page-applications/39009) é uma página que basicamente não é recarregada para executar uma ação ou para exibir uma nova tela para o usuário.

import React from "react";

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

import PaginaInicial from "../PaginaInicial";

import PaginaMobile from "../PaginaMobile";

const Rotas = () => (

<BrowserRouter>

<Routes>

<Link exact path='/' element={ <PaginaInicial /> }/>

<Link path='/mobile' element={ <PaginaMobile /> }/>

</Routes>

</BrowserRouter>

);

export default Rotas;

useRedirect() foi substituído por useNavigate()

mesmo conjunto de ferramentas.

Import {useNavigate} from ‘react-router-dom’;

<Route path=”\*” element={<useNavegate to=’/pagian-principal’ />} />;

npm install --save react-router-dom

npm install -D @types/react-router-dom

## props

O conceito de props é usado diversas vezes na criação de componentes

Utilizar props nos permite reaproveitar componentes

import React from 'react';

import './estilo.css';

export default function ItemLista(props) {

return (

<div className="boxSigno">

<div className="nomeSigno">

{props.signo}

</div>

<img className="imagemSigno" src={props.imagem} alt={props.signo}/>

<div className="periodoNascimento">

{props.dataInicio} - {props.dataFim}

</div>

</div>

)

}

Método alternativo:

export default function ItemLista({signo, imagem, dataInicio, dataFim}) {

return (

<div className = "boxSigno">

<div className = "nomeSigno">

{ signo }

</div>

<img className = "imagemSigno" src = { imagem } alt = { signo } />

<div className="periodoNascimento">

{ dataInicio } - { dataFim }

</div>

</div>

)

}

import "./styles.css";

import { useState } from "react";

import imagemCachorro from "../public/assets/cachorro.jpg";

import imagemGato from "../public/assets/gato.jpg";

export default function App() {

let imagemAnimal = "";

let informacaoAnimal = "";

let tipoAnimal = "Cachorro";

const [tipoDoComponenteCard, setTipoDoComponenteCard] = useState("cachorro");

const alterarState = () => {

if (tipoDoComponenteCard === "cachorro") {

setTipoDoComponenteCard("gato");

} else {

setTipoDoComponenteCard("cachorro");

}

};

if (tipoDoComponenteCard === "cachorro") {

imagemAnimal = imagemCachorro;

informacaoAnimal = "É um mamífero carnívoro da família dos canídeos.";

tipoAnimal = "Cachorro";

} else {

imagemAnimal = imagemGato;

informacaoAnimal = "É um mamífero carnívoro da família dos felídeos.";

tipoAnimal = "Gato";

}

return (

<div className="App">

{/\* componente Topo \*/}

<header>

<h1>Projeto Props + useState</h1>

<p>Clique no botão para mudar os componentes abaixo</p>

<button onClick=>Mudar Animal</button>

</header>

{/\* Compomente CardAnimal \*/}

<div id="CardAnimal">

<img src= alt="animal" />

</div>

{/\* Componente CardInformacoes \*/}

<div className="CardInformacao">

<h3>Informação sobre o </h3>

<p></p>

</div>

</div>

);

}

## Comando para execultar projeto baixado:

npm install

npm run dev

npm start

Deploy

Você pode criar o ícone e a logo da sua aplicação. O logo512.png deve ter o tamanho 512x512 e o logo192.png deve ter 192x192.

Ícones
e imagens

Para alterar o título abra o arquivo index.html, localizado em public/index.html

npm run build

comando para gerar os arquivos finais do projeto.

Fazer deploy somente da pasta build? Ou a pasta toda?