




HELLO
WORLD



INTRODUÇÃO AO

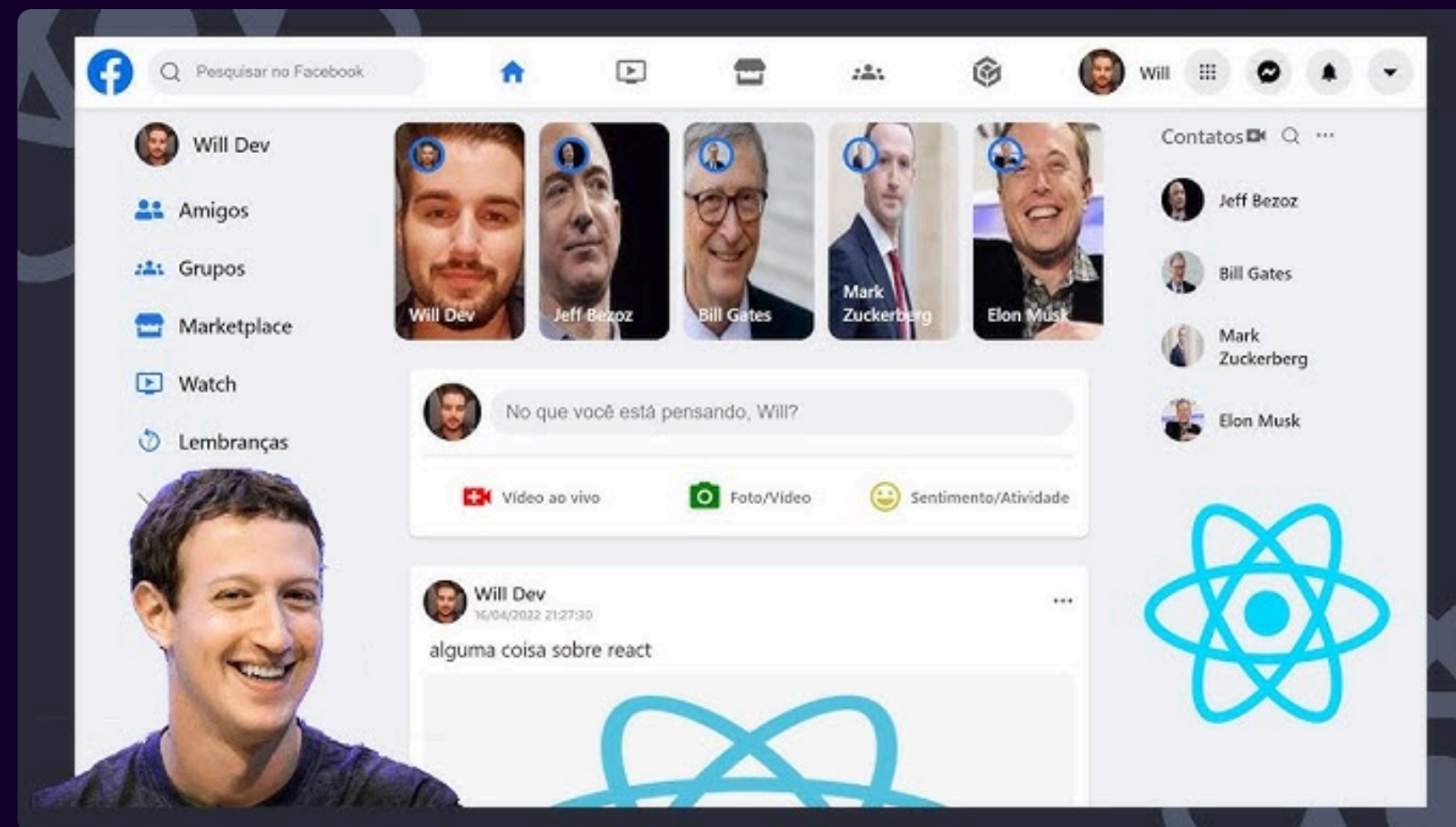
REACT.JS

LIBRARY VS FRAMEWORK

- Library (Biblioteca): Conjunto de funcionalidades focado em resolver problemas específicos.
 - Exemplos: React, Axios.
- Framework: Conjunto de funcionalidades mais abrangente que resolve diversos tipos de problemas.
 - Exemplos: Angular, Vue.js.

HISTÓRIA

O Facebook criou o React em 2011 para resolver problemas de desempenho e complexidade em sua interface, especialmente no feed de notícias. Na época, as atualizações dinâmicas de conteúdo eram lentas e difíceis de gerenciar. Jordan Walke, engenheiro do Facebook, desenvolveu o React como uma solução para atualizar apenas os elementos necessários na interface usando um conceito inovador chamado Virtual DOM, que tornou as mudanças muito mais rápidas e eficientes. Em 2013, o React foi lançado como open-source, ganhando popularidade rapidamente pela simplicidade e desempenho no desenvolvimento de interfaces.





O QUE É REACT.JS?

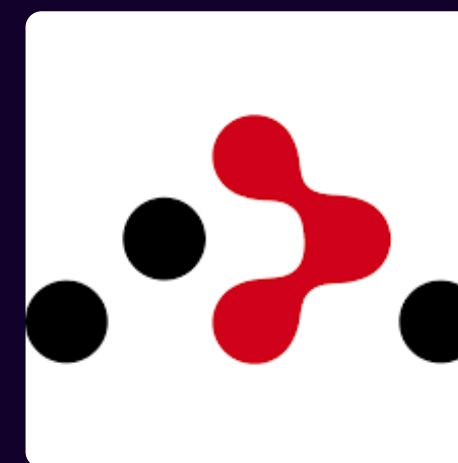
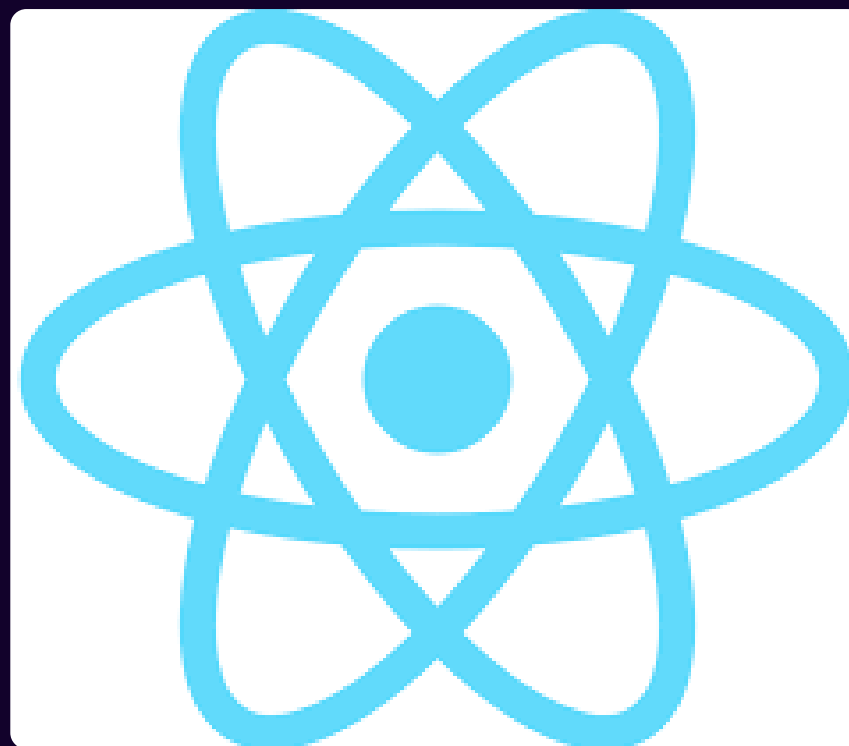
React.js é uma biblioteca JavaScript criada pelo Facebook para construir interfaces de usuário dinâmicas e eficientes, com foco em componentes reutilizáveis. React permite dividir a interface em "componentes", tornando o código mais organizado e fácil de manter.

Principais características:

- Componentes: Blocos reutilizáveis de código que representam partes da interface.
- Virtual DOM: React atualiza apenas as partes necessárias da interface, melhorando o desempenho.
- Declaratividade: Ao descrever o que deve aparecer, React cuida de como atualizar a interface.

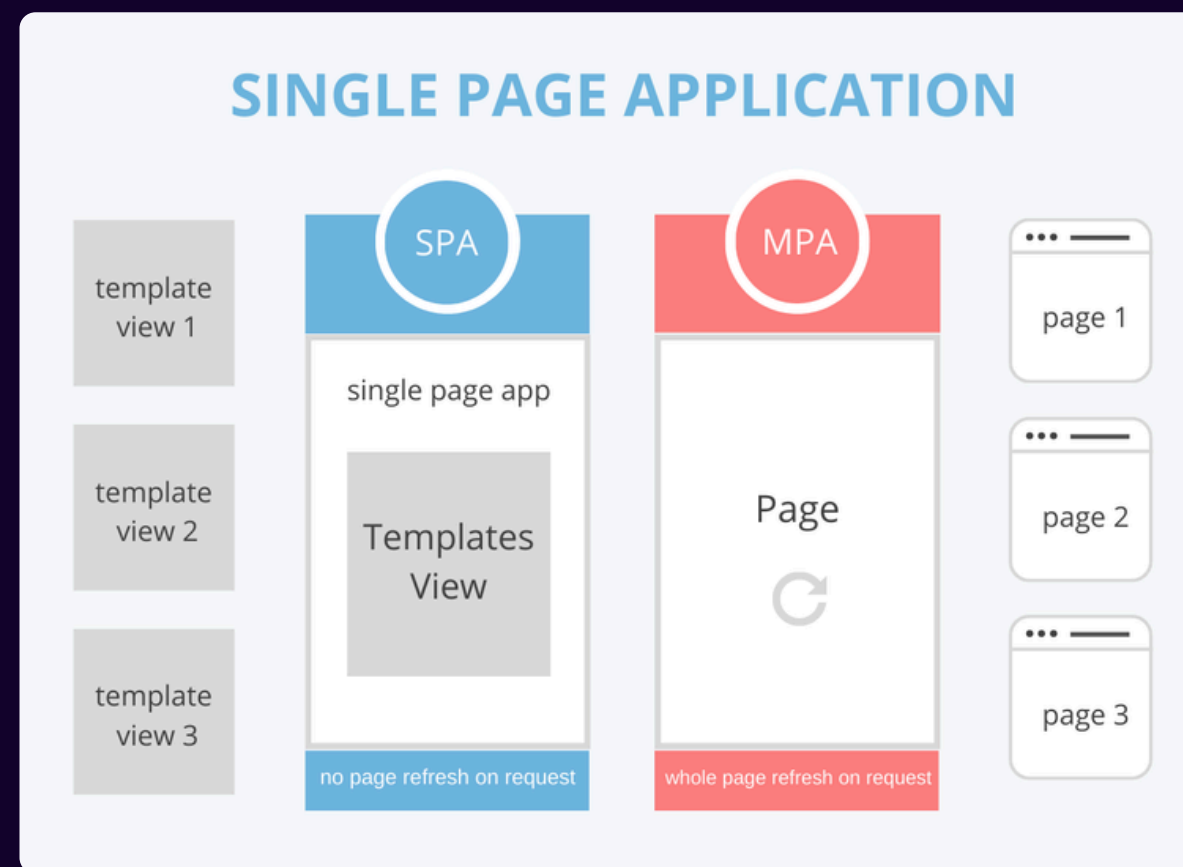
[SITE OFICIAL](#)

REACT E SEUS DERIVADOS

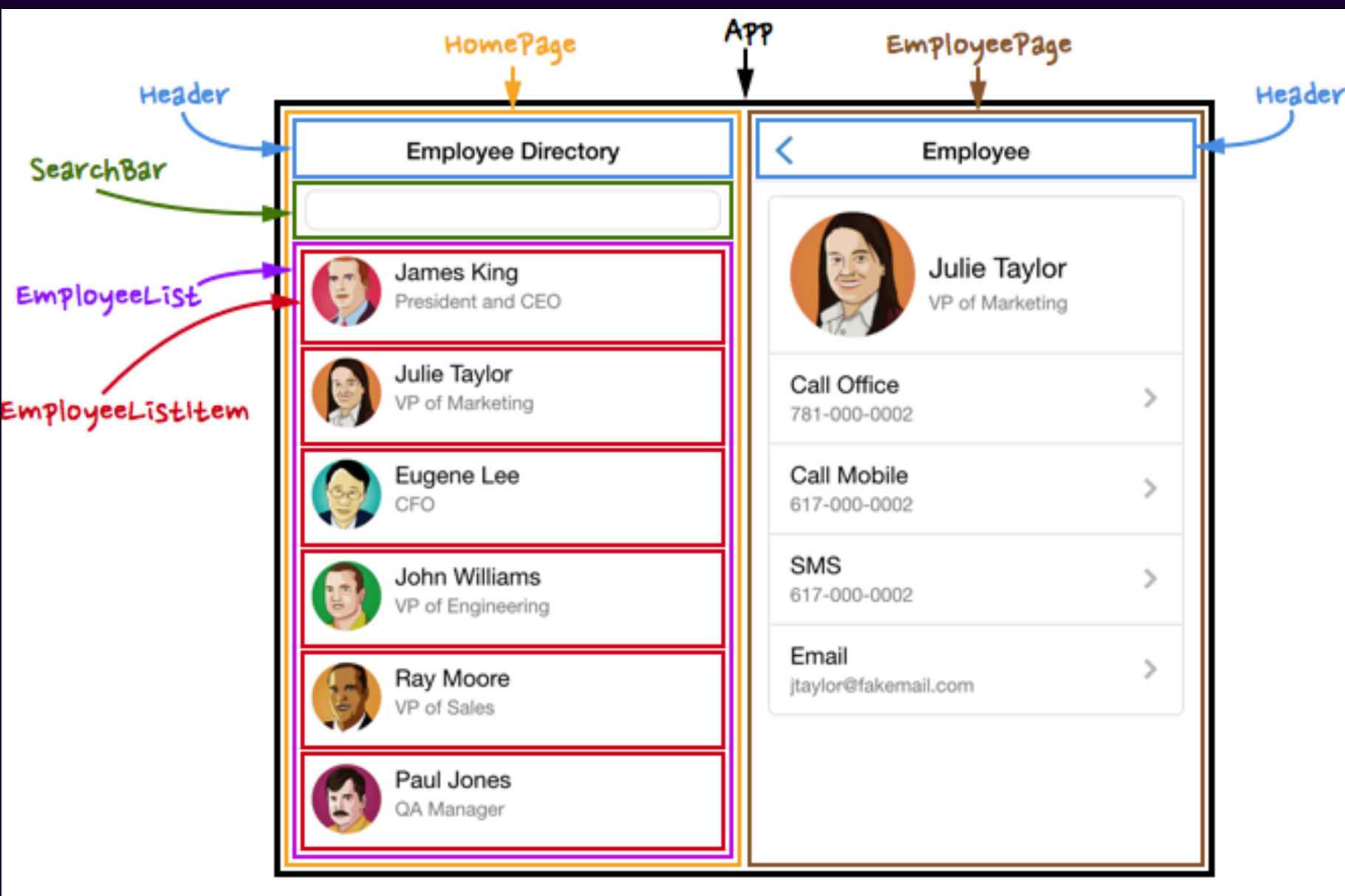


SINGLE PAGE APPLICATION

- Vantagens:
 - Navegação mais rápida e fluida.
 - Experiência de usuário mais parecida com aplicativos de desktop.
 - Menos recarregamento de página, o que melhora a usabilidade.
- Desvantagens:
 - Pode ser mais complicado para otimizar para SEO (embora isso esteja melhorando).
 - Geralmente precisa de mais JavaScript e pode ser mais pesada no carregamento inicial.



- Vantagens:
 - Melhor para SEO, pois cada página tem sua própria URL e é mais facilmente indexada.
 - Estrutura mais simples e mais leve no carregamento inicial, sem depender tanto de JavaScript.
- Desvantagens:
 - Navegação mais lenta, já que cada clique aciona um novo carregamento de página.
 - Menos fluidez na experiência do usuário.



COMPONENTES

Os componentes são as unidades fundamentais do React, representando partes isoladas da interface.

01

Reutilizáveis

02

Isolados: Alterar um componente não afeta os outros

03

Combináveis: Vários componentes podem ser usados juntos para criar interfaces complexas



TIPOS DE COMPONENTES

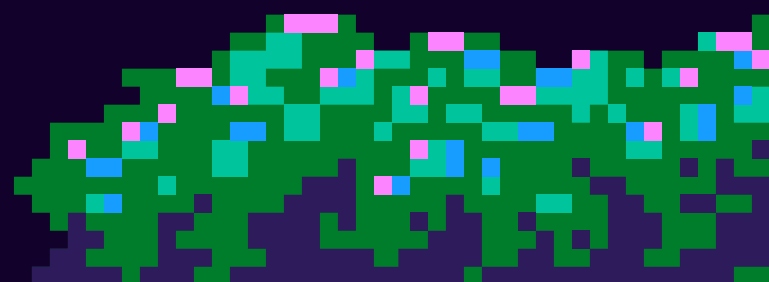
Componentes funcionais: Criados como funções JavaScript que retornam elementos HTML.

```
1 function Botao() {  
2   return <button>Click me!</button>;  
3 }
```

VS

Componentes de classe: Herdam de `React.Component` e utilizam o método `render()`.

```
1 class Botao extends React.Component {  
2   render() {  
3     return <button>Click me!</button>;  
4   }  
5 }
```

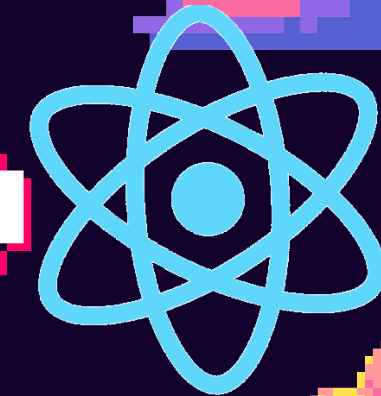




BASICAMENTE JAVASCRIPT

```
1 import { useState } from 'react';
2
3 export default function Counter() {
4   const [count, setCount] = useState(0);
5
6   function handleClick() {
7     setCount(count + 1);
8   }
9
10  return (
11    <button onClick={handleClick}>
12      You pressed me {count} times
13    </button>
14  );
15 }
16
```

CONFIGURAÇÃO DO AMBIENTE MANUAL



PASSO 1: CRIAÇÃO DO PROJETO E INSTALAÇÃO DAS DEPENDÊNCIAS

Iniciar o Projeto: No terminal, navegue até a pasta onde quer criar o projeto e execute:

```
mkdir meu-projeto-react  
cd meu-projeto-react  
npm init -y
```

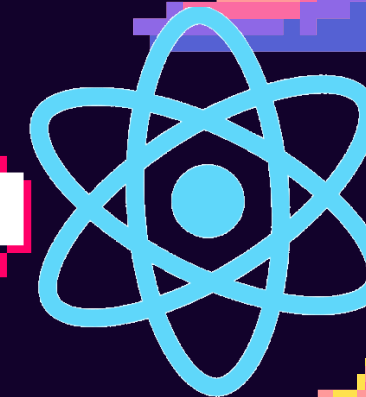
Instalar React e React DOM:

```
npm install react react-dom
```

Instalar e Configurar Webpack e Babel: Webpack e Babel são ferramentas que permitem transformar o código React (JSX) em JavaScript compreensível para navegadores.

```
npm install webpack webpack-cli webpack-dev-server --save-dev  
npm install @babel/core babel-loader @babel/preset-env @babel/preset-react --save-dev
```

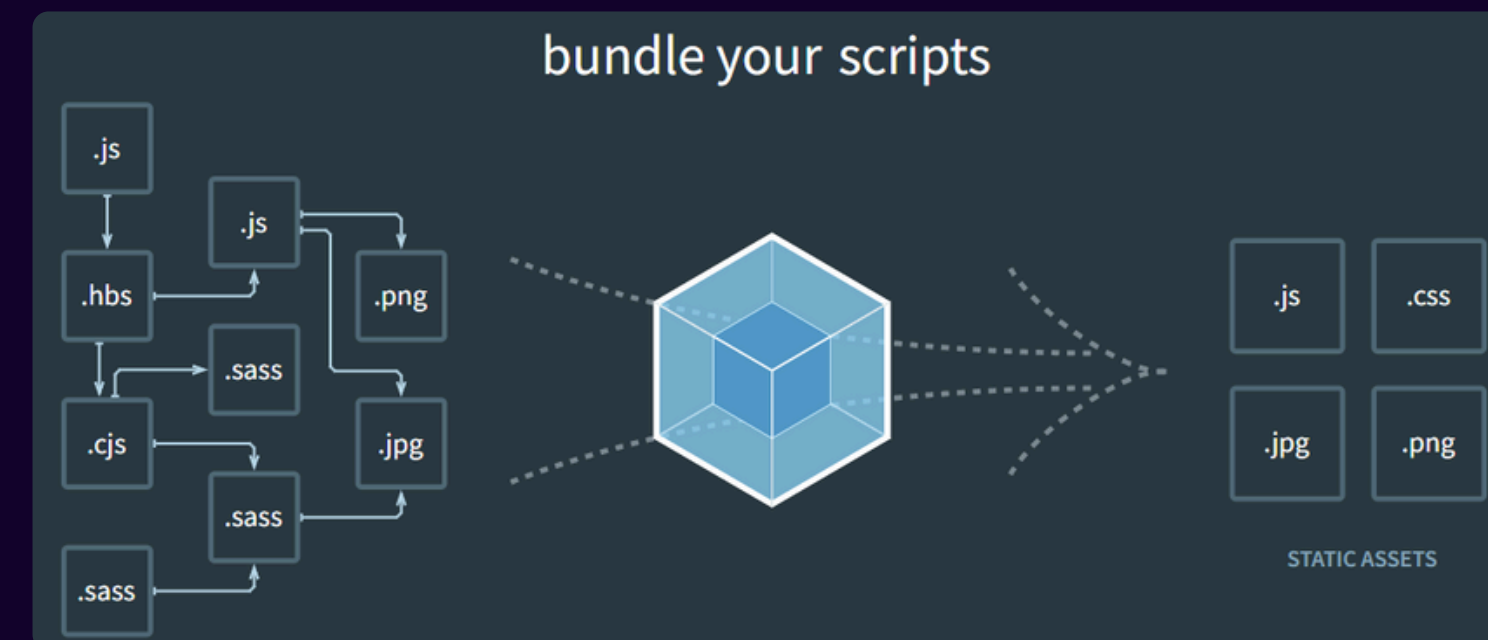
CONFIGURAÇÃO DO AMBIENTE MANUAL



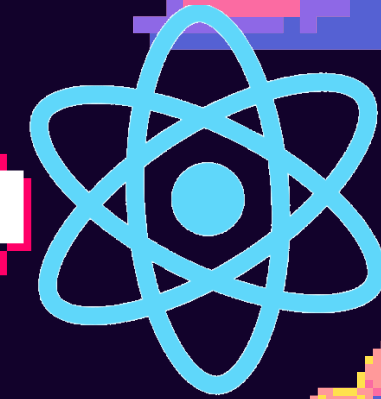
PASSO 2: CONFIGURAÇÃO DO WEBPACK

Crie um arquivo webpack.config.js na
raiz do projeto

```
1 const path = require('path');
2
3 module.exports = {
4   entry: './src/index.js',
5   output: {
6     path: path.resolve(__dirname, 'dist'),
7     filename: 'bundle.js',
8   },
9   module: {
10    rules: [
11      {
12        test: /\.?(js|jsx)$/,
13        exclude: /node_modules/,
14        use: {
15          loader: 'babel-loader',
16        },
17      },
18    ],
19  },
20  resolve: {
21    extensions: ['.js', '.jsx'],
22  },
23  devServer: {
24    contentBase: path.join(__dirname, 'dist'),
25    compress: true,
26    port: 9000,
27  },
28 };
```



CONFIGURAÇÃO DO AMBIENTE MANUAL

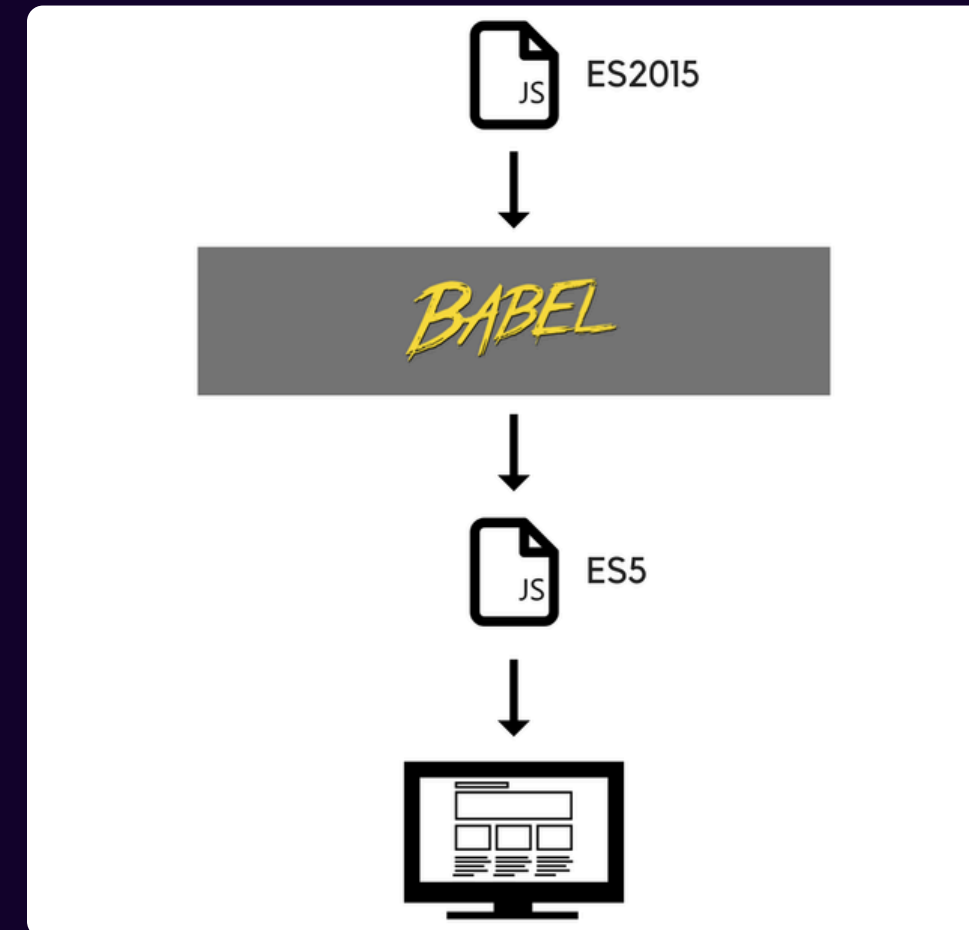


PASSO 3: CONFIGURAÇÃO DO BABEL

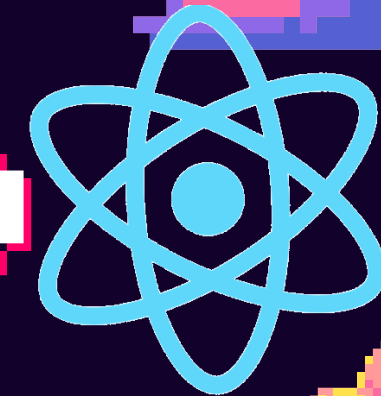
Crie um arquivo .babelrc:

```
1 {  
2   "presets": ["@babel/preset-env", "@babel/preset-react"]  
3 }
```

BABEL



CONFIGURAÇÃO DO AMBIENTE MANUAL

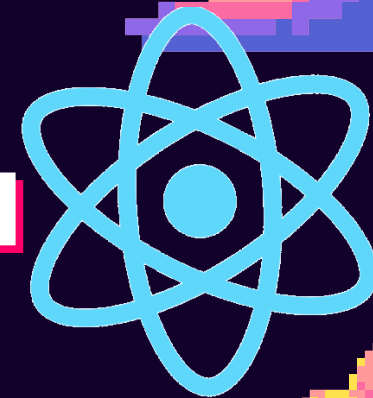


PASSO 4: ESTRUTURA DO PROJETO

Crie a estrutura de diretórios e arquivos:

```
meu-projeto-react/  
├── src/  
│   ├── index.js  
│   └── App.js  
├── dist/  
│   └── index.html  
├── webpack.config.js  
└── .babelrc
```

CONFIGURAÇÃO DO AMBIENTE MANUAL



PASSO 5: ARQUIVOS E CONTEÚDO INICIAL

Arquivo index.html:

```
1 <!DOCTYPE html>
2 <html lang="pt-BR">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Meu Projeto React</title>
7 </head>
8 <body>
9   <div id="root"></div>
10  <script src="bundle.js"></script>
11 </body>
12 </html>
```

Arquivo index.js:

```
1 import React from 'react';
2 import ReactDOM from 'react-dom';
3 import App from './App';
4
5 ReactDOM.render(<App />, document.getElementById('root'));
```

O QUE É O JSX?

JSX (JavaScript XML) é uma extensão de sintaxe para JavaScript usada principalmente com o React para construir interfaces de usuário. Ele permite escrever código que parece HTML dentro do JavaScript, o que facilita a criação de componentes visuais com uma sintaxe familiar.

```
1 function App() {  
2   return (  
3     <div>  
4       <h1>Olá, Mundo! </h1>  
5     </div>  
6   );  
7 }
```

Por que usar JSX?

- **Sintaxe Familiar:** Parece HTML, o que facilita para devs web que já conhecem essa linguagem.
- **Modularidade:** Como os componentes React utilizam JSX, é fácil quebrar a interface em partes reutilizáveis.
- **Eficiente:** JSX é compilado para `React.createElement` no JavaScript, o que torna o código mais otimizado.

PRIMEIRO COMPONENTE



```
1 import React from 'react';  
2  
3 const App = () => {  
4     return <h1>Olá, React!</h1>;  
5 };  
6  
7 export default App;
```



ESTILOS INLINE NO REACT

Estilos inline no React são uma maneira de aplicar CSS diretamente a um elemento, utilizando objetos JavaScript em vez de classes ou arquivos CSS externos. Cada propriedade CSS é definida em camelCase (exemplo: backgroundColor ao invés de background-color), e os valores são passados como strings (ou números para propriedades sem unidades).

```
● ● ●
1 function App() {
2   return (
3     <div style={{ backgroundColor: "lightblue", padding: "20px" }}>
4       <h1 style={{ color: "darkblue" }}>Olá, Mundo!</h1>
5     </div>
6   );
7 }
8
```





VANTAGENS

01

Escopo Local: Como são aplicados apenas no componente específico, evita conflitos de estilo.

02

Dinamicidade: É fácil passar valores de variáveis e props diretamente, o que permite manipular os estilos com base no estado do componente ou props.

```
1 function Button({ isActive }) {  
2   return (  
3     <button style={{ color: isActive ? "green" : "gray" }}>  
4       {isActive ? "Ativo" : "Inativo"}  
5     </button>  
6   );  
7 }
```

Casos para Usar Inline Styles

- **Estilização Condicional Simples:** Como mudar uma cor com base em uma condição.
- **Customização Temporária ou Única:** Quando você precisa de um estilo único para um elemento específico sem impactar o resto da aplicação.

DESVANTAGENS

01

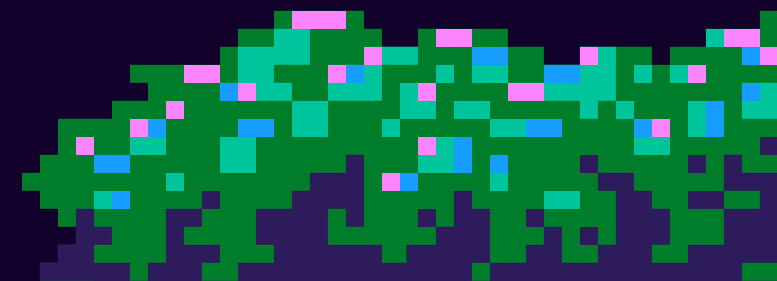
Manutenção e Reutilização: Com muitos estilos inline, o código pode se tornar longo e difícil de manter.

02

Falta de Pseudo-Classes: Estilos inline não suportam pseudo-classes como :hover, :focus, etc.

03

Limitações de CSS Completo: Técnicas avançadas, como animações complexas e media queries, não são suportadas com inline styles.





PORQUE USAR REACT?

01

Desempenho com Virtual DOM

02

Componente Reutilizáveis

03

Arquitetura Simples e Declarativa

04

React Hooks

05

Grande Comunidade e Ecossistema

06

SEO Amigável (com SSR e SSG)

07

Suporte para Desenvolvimento Mobile com React Native

08

Facilidade de Aprendizado e Documentação Completa





EDRA REAGIA

NOSSOS CONTATOS:



27 99500-7495



<https://beacons.ai/prismatech>



producaoprismatech@gmail.com



Avenida Jerônimo Monteiro 145, Vitória





THANK
YOU