📘 O que é o React?

React é uma **biblioteca JavaScript** criada pelo Facebook (Meta) para construir **interfaces de usuário** (UIs) de forma rápida, modular e reativa. Com React, você cria **componentes reutilizáveis**, cada um responsável por uma parte da tela — por exemplo, um botão, uma lista ou um formulário.

> 💡 Diferente de um framework completo (como Angular ou Vue), o React foca apenas na **camada de visualização** (View) da aplicação.

🧱 Conceito de Componentes

Um **componente** é como uma "peça de LEGO" da interface. Cada componente pode:

- Ter seu próprio **HTML (JSX)**
- Ter **estilos (CSS)**
- E sua **lógica (JavaScript)**

Exemplo de componente simples:

```
function Mensagem() {
  return <h1>Olá, mundo!</h1>;
}
```

E no arquivo `App.js`:

```
function App() {
  return (
    <div>
      <Mensagem />
    </div>
  );
}
```

⚙ Estrutura de um Projeto React

Ao criar um projeto com:

```
npx create-react-app meu-projeto
```

Você terá uma estrutura como:

```
meu-projeto/
├── node_modules/
├── public/
├── src/
│   ├── App.js
│   ├── index.js
│   └── ...
├── package.json
└── README.md
```

- **src/** → onde ficam os arquivos principais da aplicação.
- **App.js** → componente principal.
- **index.js** → ponto de entrada do React no HTML.

🔤 JSX – JavaScript + HTML

O React usa **JSX**, uma extensão do JavaScript que permite escrever **HTML dentro do código JS**.

Exemplo:

```
function App() {
  const nome = "Maria";
  return <h2>Olá, {nome}!</h2>;
}
```

> As chaves `{}` permitem inserir **expressões JavaScript** dentro do HTML.

🔁 Estado e Hooks

React usa **Hooks** para lidar com **estado** e **eventos**. O Hook mais básico é o `useState`.

Exemplo de contador:

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

function App() {
  const [contador, setContador] = useState(0);

  return (
    <div>
      <p>Você clicou {contador} vezes</p>
      <button onClick={() => setContador(contador + 1)}>Clique aqui</button>
    </div>
  );
}
```

📥 Props (Propriedades)

As **props** servem para **passar informações de um componente para outro**, como parâmetros de função.

Exemplo:

```
function Saudacao(props) {
  return <h2>Olá, {props.nome}!</h2>;
}

function App() {
  return <Saudacao nome="João" />;
}
```

🔄 Renderização de Listas

Você pode exibir listas dinamicamente com o método `.map()`:

```
function App() {
  const livros = ["Dom Casmurro", "1984", "O Hobbit"];

  return (
    <ul>
      {livros.map((livro, i) => (
        <li key={i}>{livro}</li>
      ))}
    </ul>
  );
}
```

Exercícios Práticos

> Use o editor de código online PlayCode - React para completar os exercícios abaixo:

1. Crie uma aplicação React que exiba uma **mensagem de boas-vindas** com o seu nome.
2. Crie um **componente** `Mensagem` que receba uma **prop** com o nome e exiba: “Bem-vindo(a), [nome]!”
3. Crie um **contador simples** com `useState`.
4. Mostre uma **lista de 3 livros** na tela.
5. Crie um **campo de texto** que mostre em tempo real o que o usuário digita.

Curso Grátis

- Curso gratuito (inglês): *Learn React* – Codecademy