
SOFTEX
PERNAMBUCO

Softex

MINISTÉRIO DA
CIÊNCIA, TECNOLOGIA
E INOVAÇÃO

GOVERNO FEDERAL
BRASIL
UNIÃO E RECONSTRUÇÃO

# Aula 23 | Módulo: React JS (Introdução)

- ➔ Introdução ao estado com useState.
- ➔ Projeto prático: criação de uma interface simples com múltiplos componentes e props.

# React JS (Introdução)

## Configurando o ambiente para rodar o React

- Node.js (LTS recomendado >= 20):
  - **Windows/macOS:**
    - baixe o instalador LTS no site do Node.
  - **Linux (Deb/Ubuntu):**
    - curl -fsSL https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.7/install.sh | bash
    - nvm install --lts
    - nvm use --lts
- Confirmar instalação rodando os comandos abaixo em algum terminal:
  - **node -v**
  - **npm -v**
- Se “npm/node não reconhecido”, feche o terminal e tente pelo Command Prompt.

## Abrindo/Criando um projeto

- Para ABRIR um projeto existente:
  - Abra o VSCode, clique em File > Open Folder… (Arquivo > Abrir Pasta…) e procurar a pasta do projeto existente.
  - No terminal, rode o comando ao lado para iniciar: **npm run dev**
- Para CRIAR um projeto novo
  - Abra o VSCode, clique em File > Open Folder… (Arquivo > Abrir Pasta…).
  - Escolha um local para criar a sua pasta, crie uma nova pasta e dê o nome de **seunome_aula_react_22**. Depois dê dois clique nessa pasta criada e clique em Selecionar pasta. O VSCode reabrirá dentro dessa pasta que foi criada.

## Criar o projeto com Vite

- Criando um projeto com JavaScript puro:
  - **npm create vite@latest . -- --template react**
- Durante a instalação do React usando o Vite, algumas opções podem surgir:

```
PS E:\hygorrasec\curso_react> npm create vite@latest . -- --template react
Need to install the following packages:
create-vite@8.0.2
Ok to proceed? (y) y


> npx
> create-vite . --template react

◇  Use rolldown-vite (Experimental)?:
   No

◇  Install with npm and start now?
   Yes
```

# React JS (Introdução)

**Se algo travar: portas ou permissões**

- Porta ocupada (erro ao iniciar ou tela vazia):
  - Iniciar em outra porta: **npm run dev -- --port 5174**
- Windows: “npm não reconhecido”
  - Feche e reabra o PowerShell/CMD.
  - Verifique se o Node foi instalado para “All Users” (às vezes a variável de ambiente não atualiza).
- Linux/macOS: erro de permissão (EACCES)
  - mkdir -p ~/.npm-global
  - npm config set prefix ~/.npm-global
  - echo 'export PATH=$HOME/.npm-global/bin:$PATH' >> ~/.bashrc && source ~/.bashrc

# React JS (Introdução)

## Relembrando!

- Criação e exportação de componentes funcionais com JSX.
- Em React, componentes são funções JavaScript que retornam elementos JSX (parecidos com HTML).
- Eles permitem reutilizar e organizar partes da interface.
- Um componente pode ser criado, exportado e usado em outros arquivos.

src > components > Saudacao.jsx > ...

```jsx
function Saudacao() {
    return <h2>Olá, mundo!</h2>;
}

export default Saudacao;
```

src > App.jsx > ...

```jsx
import Saudacao from './components/Saudacao';

function App() {
    return <Saudacao />
}

export default App
```

Lembra? O nome do componente **sempre** começa com letra maiúscula.

# React JS (Introdução)

## Relembrando!

- Uso de Props: Passagem de Dados entre Componentes.
- Props (propriedades) são a forma de enviar dados de um componente pai para um filho.
- Elas tornam o componente dinâmico e personalizável.
- Props são imutáveis, o componente que recebe não pode alterar o valor.

src > components > Jogo.jsx > ...

```jsx
function Jogo({ nome }) {
    return <h2>{nome}</h2>;
}

export default Jogo;
```

src > App.jsx > ...

```jsx
import Jogo from './components/Jogo';

function App() {
    return <Jogo nome="Desolara RPG" />;
}

export default App
```

# React JS (Introdução)

**Agora vamos tornar os componentes interativos**

- Até agora, nossos componentes recebiam informações, mas não mudavam por conta própria.
- Hoje aprenderemos a dar “vida” a eles com o estado (**useState**), que permite armazenar e atualizar informações internas de um componente.

## O que é Estado?

- “Estado” é o conjunto de informações que mudam ao longo do tempo dentro de um componente.
- Quando o estado muda, o React re-renderiza automaticamente o componente.
- É como uma “memória interna” do componente.

- Analogias:
  - Um interruptor: ligado/desligado.
  - Um contador: valor que aumenta quando clicamos.
  - Um campo de formulário: texto que muda conforme digitamos.

## Por que usar o useState?

- Em React, os componentes não guardam informações por conta própria, eles são “estáticos”.
- Para armazenar valores que mudam com o tempo, usamos o hook **useState**.
- Hooks são funções especiais do React que adicionam “superpoderes” aos componentes.
- **useState** é o hook que permite criar variáveis reativas, quando mudam, o React atualiza automaticamente a interface.

# React JS (Introdução)

## Sintaxe Básica

src > components > Exemplo.jsx > ...

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

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

export default Exemplo;
```

- contador > é o valor atual do estado.
- setContador > é a função que atualiza o valor.
- 0 > é o valor inicial do estado.

# React JS (Introdução)

## Exemplo: Contador Interativo

src > components > Contador.jsx > ...

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

function Contador() {
    const [numero, setNumero] = useState(0);

    return (
        <>
            <h1>Você clicou {numero} vezes!</h1>
            <button onClick={() => setNumero(numero + 1)}>Clique aqui</button>
        </>
    );
}

export default Contador;
```

# React JS (Introdução)

**Por que isso é importante?**

- Antes do **useState**, teríamos que:
  - Manipular o DOM manualmente (com document.querySelector, etc.)
  - Lidar com recarregamentos desnecessários
- Agora o React cuida disso sozinho!
- O estado muda > o React re-renderiza o componente > a tela mostra o novo valor automaticamente.

**Atividade rápida:**

➔ Duplique o botão que criamos e faça com que ele diminua o valor:

- setNumero(numero - 1)

## Múltiplos Estados no Mesmo Componente

- Gerenciando mais de uma variável

src > components > Perfil.jsx > ...

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

function Perfil() {
    const [nome, setNome] = useState('Hygor');
    const [idade, setIdade] = useState(37);

    return (
        <>
            <h1>{nome} - {idade} anos</h1>
            <button onClick={() => setIdade(idade + 1)}>Fazer Aniversário!!!</button>
        </>
    );
}

export default Perfil;
```

## Múltiplos Estados no Mesmo Componente

- Podemos usar quantos useState precisarmos.
- Cada um controla uma parte do comportamento.

# React JS (Introdução)

## Exercício 1

1. Crie um componente Jogador.jsx
2. Use useState para:
    - nome (string)
    - nível (número)
    - vocação (string)
3. Exiba as informações na tela.
4. Adicione botões para aumentar o nível e alterar a vocação.
5. Desafio extra:
    - Permitir editar o nome com um <input> controlado.

## Estado em Componentes Filhos

- Queremos que o componente **Painel** mostre o número do contador, mas o botão está no componente **Botao.jsx**

src > components > Painel.jsx > ...

```jsx
function Painel({ contador }) {
    return <div>Contador: {contador}</div>;
}

export default Painel;
```

src > components > Botao.jsx > ...

```jsx
function Botao({ aumentar }) {
    return <button onClick={aumentar}>Aumentar</button>;
}

export default Botao;
```

## Estado em Componentes Filhos

- Queremos que o componente **Painel** mostre o número do contador, mas o botão está no componente **Botao.jsx**

src > App.jsx > ...

```jsx
import Painel from './components/Painel';
import Botao from './components/Botao';
import { useState } from 'react';

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

    return (
        <>
            <Painel contador={contador} />
            <Botao aumentar={() => setContador(contador + 1)} />
        </>
    );
}

export default App;
```

## Boas Práticas com useState

- Sempre inicialize o estado com um valor do mesmo tipo esperado.
- Use nomes descritivos (isVisible, contador, mensagem).
- Evite lógica pesada dentro de **setState**, calcule antes.

# React JS (Introdução)

## Mini Projeto: “Painel de Cliques”

- **Objetivo:** Criar uma interface com múltiplos componentes que compartilham estado.
- **Componentes sugeridos:**
  - **App.jsx:** controla o estado principal (cliques).
  - **Contador.jsx:** exibe o total.
  - **BotaoAdicionar.jsx:** adiciona +1.
  - **BotaoZerar.jsx:** reseta para 0.
- **Requisitos:**
  - Use **useState** no App.
  - Passe dados e funções via **props**.
  - Exiba tudo com estilo simples (CSS básico).
- **Extra:**
  - Mostrar mensagem **“Parabéns!”** quando cliques for maior ou igual a 10.

# React JS (Introdução)

## Gabarito Exercício 1

src > components > Jogador.jsx > ...

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

function Jogador() {
    const [nome, setNome] = useState('Hygor');
    const [nivel, setNivel] = useState(1);
    const [vocacao, setVocacao] = useState('Guerreiro');
    function trocarVocacao() {
        setVocacao((vocacao === 'Guerreiro') ? 'Mago' : 'Guerreiro');
    }
    return (
        <>
            <p><strong>Nome:</strong> {nome}</p>
            <p><strong>Nível:</strong> {nivel}</p>
            <p><strong>Vocação:</strong> {vocacao}</p>
            <button onClick={() => setNivel(nivel + 1)}>Subir de Nível</button>
            <button onClick={trocarVocacao}>Trocar Vocação</button>
            <input type="text" value={nome} onChange={(e) => setNome(e.target.value)} />
        </>
    );
}

export default Jogador;
```

## Gabarito Mini Projeto

1. No terminal, rode: **git clone https://github.com/hygorrasec/painel_de_cliques_react**
2. Depois entre na pasta que baixar, e rode: **npm install**
3. E depois inicie o projeto para visualizar: **npm run dev**

# ATÉ A PRÓXIMA AULA!

*Front-end - Design. Integração. Experiência.*

**Professor:** Hygor Rasec
https://www.linkedin.com/in/hygorrasec
https://github.com/hygorrasec