



SOFTEX
PERNAMBUCO

 **Softex**

MINISTÉRIO DA
CIÊNCIA, TECNOLOGIA
E INOVAÇÃO

GOVERNO FEDERAL
BRASIL
UNIÃO E RECONSTRUÇÃO



Aula 21 | Módulo: React JS (Introdução)



- Conceito de componentes: reutilização e modularidade
- Configuração de ambiente com Vite ou Create React App (já foi ensinado na aula passada)



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 no terminal:
 - **`node -v`**
 - **`npm -v`**
- Se “npm/node não reconhecido”, feche o terminal e tente pelo Command Prompt.



React JS (Introdução)



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_21**. Depois dê dois clique nessa pasta criada e clique em Selecionar pasta. O VSCode reabrirá dentro dessa pasta que foi criada.



React JS (Introdução)



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)



O que é um componente? (revisão e novos contextos)

- Um componente é uma função JavaScript que representa uma parte da interface (visual) do site.
- No React, tudo é componente: botões, menus, cabeçalhos, rodapés, formulários...
- O React lê e exibe os componentes através do App.jsx, que é o ponto inicial da aplicação.
- Analogia:
 - Pense no site como um carro. Ele não é feito de um bloco só, é feito de peças: o volante, as rodas, o painel, o motor.
- O React segue o mesmo princípio. Em vez de montar a página inteira num arquivo só, criamos partes isoladas chamadas componentes. Cada um tem sua função específica, e juntos formam a aplicação completa.”



React JS (Introdução)



Onde os componentes ficam?

- Depois de criar o projeto com o Vite (ou usar o que já foi feito na aula anterior), vocês verão a pasta **src**. É dentro dela que ficam os arquivos React.

- O fluxo funciona assim:

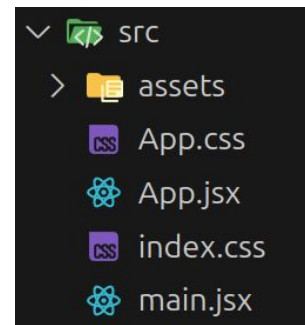
index.html



main.jsx (renderiza o componente `<App />`)



App.jsx (exibe os outros componentes que criamos)



Cria o componente > importa no App.jsx > ele aparece no site.




React JS (Introdução)



Exemplo prático

- Dentro da pasta **src**, crie um arquivo chamado **Mensagem.jsx**

```
src >  Mensagem.jsx > ...  
1  function Mensagem() {  
2    |    return <p>Olá, React!</p>;  
3  }  
4  
5  export default Mensagem;  
6
```



React JS (Introdução)



Exemplo prático

- Dentro do **App.jsx**, vamos importar e exibir o componente criado. Depois visualize no navegador (**localhost:5173**) se a mensagem aparece.

```
src > App.jsx > ...
1  import Mensagem from './Mensagem';
2
3  function App() {
4    return (
5      <>
6        <Mensagem />
7      </>
8    );
9  }
10
11  export default App;
12
```



React JS (Introdução)



Vamos entender o que fizemos em Mensagem.jsx

- **function Mensagem() { ... }**
- Define uma função JavaScript chamada Mensagem.
- Essa função é o componente em si, cada componente React é uma função.
- O nome deve começar com letra maiúscula, pois é assim que o React reconhece que se trata de um componente (e não uma tag HTML).
- Analogia:
 - Pense que “Mensagem” é uma mini-fábrica. Sempre que o React encontrar `<Mensagem />`, ele vai chamar essa função e montar o que ela retorna.



React JS (Introdução)



Vamos entender o que fizemos em Mensagem.jsx

- **return <p>Olá, React!</p>;**
- Tudo que está dentro do return é o que o componente vai mostrar na tela.
- É um trecho de JSX, que parece HTML, mas é HTML dentro do JavaScript.
- Aqui, o componente retorna um parágrafo (<p>) com o texto “Olá, React!”.
- Analogia:
 - O return é como dizer: “quando alguém usar este componente, mostre este conteúdo na tela”.



React JS (Introdução)



Vamos entender o que fizemos em Mensagem.jsx

- **export default Mensagem;**
- **export** “Exporta” o componente, ou seja, permite que outros arquivos o utilizem.
- **default** indica que este é o principal item exportado por esse arquivo.
- Um arquivo pode exportar apenas um item “default”.
- Quando um componente é exportado como default, ele pode ser importado com qualquer nome.

```
src > Mensagem.jsx > ...  
1 function Mensagem() {  
2   |   return <p>Olá, React!</p>;  
3 }  
4  
5 export default Mensagem;
```

```
src > App.jsx > ...  
1 import Mensagem from './Mensagem'; // funciona!  
2 import Banana from './Mensagem'; // aqui também funciona!  
3  
4 function App() {  
5   |   return (  
6     |     <>  
7       |     <Mensagem />  
8       |     <Banana />  
9     |     </>  
10  |   )  
11 }
```



React JS (Introdução)



Vamos entender o que fizemos em Mensagem.jsx

- Export sem default (export nomeado)

```
src > Mensagem.jsx > ...  
1 export function Mensagem() {  
2   |   return <p>Olá, React!</p>;  
3 }  
4
```

```
src > App.jsx > ...  
1 import { Mensagem } from './Mensagem'; // agora precisa das chaves {}  
2  
3 function App() {  
4   |   return (  
5     |     <>  
6     |     <Mensagem />  
7     |   </>  
8   )  
9 }
```

- Aqui o nome precisa ser exatamente igual ao nome da função exportada.
- É obrigatório usar chaves {}.



React JS (Introdução)



Vamos entender o que fizemos em App.jsx

- **import Mensagem from './Mensagem';**
- Importa o componente que criamos em Mensagem.jsx.
- O caminho './Mensagem' diz ao React: “Procure o arquivo Mensagem.jsx dentro da mesma pasta”.
- Após importar, podemos usar o componente dentro do JSX usando a tag
 - **<Mensagem />**
- No React, sempre que você quiser usar um componente externo, precisa importá-lo primeiro.



React JS (Introdução)



Vamos entender o que fizemos em App.jsx

- **function App() { ... }**
- Cria o componente principal do aplicativo.
- O React começa exibindo o conteúdo desse componente porque ele é chamado dentro do **main.jsx** (onde o React inicia o app).
- Analogia:
 - Se o site fosse uma casa, o App seria o corpo principal dela, e os outros componentes (como Mensagem, Header, Footer) seriam os cômodos.



React JS (Introdução)



Vamos entender o que fizemos em App.jsx

- `return (<> <Mensagem /> </>);`
- Tudo dentro do return é o que será mostrado na tela.
- `<Mensagem />` chama o componente Mensagem, que mostramos antes.
- As tags `<>` e `</>` são chamadas de **fragmentos** (React.Fragment):
- Elas agrupam vários elementos sem criar uma div extra no HTML.
- São muito usadas quando você quer retornar mais de um elemento sem adicionar tags desnecessárias.
- O `<> </>` é como uma caixa invisível que serve apenas para “segurar” os elementos dentro dela.



React JS (Introdução)



Vamos entender o que fizemos em App.jsx

- **export default App;**
- Exporta o componente principal App para que ele possa ser usado no main.jsx (que é o arquivo onde o React realmente coloca o site no navegador).



React JS (Introdução)



Fluxo geral até o navegador

index.html

carregado primeiro pelo navegador



main.jsx

inicializa o React e renderiza `<App />` no `#root`



App.jsx

componente principal, que importa e usa `<Mensagem />`



Mensagem.jsx

componente que retorna o conteúdo visual



React monta tudo e exibe no navegador



React JS (Introdução)



Fluxo geral até o navegador

1. O navegador sempre começa pelo index.html, mas esse arquivo só tem uma div vazia chamada root.
2. O React entra em cena através do main.jsx, que coloca dentro dessa div o conteúdo do nosso aplicativo (App).
3. A partir daí, o React monta todos os componentes que o App usa, como o Mensagem, e mostra tudo na tela.

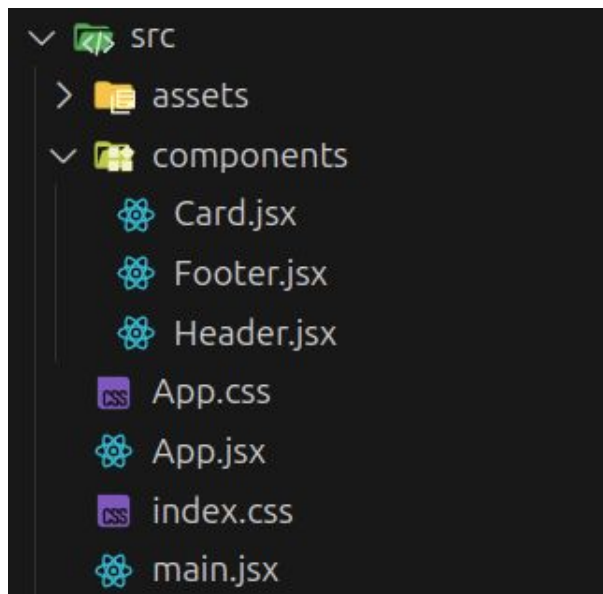


React JS (Introdução)



Reutilização visual com componentes (Header, Cards, Footer)

- Vamos ver na prática como um mesmo componente pode ser usado várias vezes.
- Entender a organização mínima (pasta components/) e estilos.





React JS (Introdução)



Reutilização visual com componentes (Header, Cards, Footer)

- Estilos globais (src/index.css)
 - Esse arquivo deve conter cores, fontes e resets gerais do site.

```
src > index.css > ...
1  :root {
2    --bg: #0f172a;
3    --text: #e5e7eb;
4  }
5
6  * {
7    box-sizing: border-box;
8    margin: 0;
9    padding: 0;
10 }
11
12 body {
13   background: var(--bg);
14   color: var(--text);
15   font-family: Arial, Verdana;
16 }
```



React JS (Introdução)



Reutilização visual com componentes (Header, Cards, Footer)

- Estilos do aplicativo (**src/App.css**)
 - Aqui entram os estilos específicos da
 - página e dos componentes.

```
src > App.css > ...
1  body {
2    font-family: Arial, Verdana;
3    margin: 0;
4    padding: 0;
5  }
6
7  .header, .footer {
8    background-color: #333;
9    color: white;
10   text-align: center;
11   padding: 10px;
12 }
13
14 .main { padding: 20px; text-align: center; }
15
16 .card {
17   border: 1px solid #ccc;
18   border-radius: 5px;
19   padding: 15px;
20   margin: 10px;
21   display: inline-block;
22   width: 200px;
23 }
24
```



React JS (Introdução)



Agora vamos criar o component (src/components/Header.jsx)


```
src > components > Header.jsx > ...  
1  function Header() {  
2      return (  
3          <header className="header">  
4              <h1>Meu Site em React</h1>  
5          </header>  
6      );  
7  }  
8  
9  export default Header;  
10
```




React JS (Introdução)



Agora vamos criar o component (src/components/Card.jsx)

```
src > components >  Card.jsx > ...
1  function Card() {
2      return (
3          <div className="card">
4              <h3>Título do Card</h3>
5              <p>Esse é um exemplo de componente reutilizável.</p>
6          </div>
7      );
8  }
9
10 export default Card;
11
```



React JS (Introdução)



Agora vamos criar o component (src/components/Footer.jsx)

```
src > components >  Footer.jsx > ...  
1   function Footer() {  
2       return (  
3           <footer className="footer">  
4               <p>2025 - Criado na Aula de React</p>  
5           </footer>  
6       );  
7   }  
8  
9   export default Footer;  
10  
11
```



React JS (Introdução)



Agora vamos ajustar o (src/App.jsx)

```
src > App.jsx > ...
1  import './App.css';
2  import Header from './components/Header';
3  import Card from './components/Card';
4  import Footer from './components/Footer';
5
6  function App() {
7    return (
8      <>
9        <Header />
10       <main className="main">
11         <Card /><Card /><Card />
12       </main>
13       <Footer />
14     </>
15   );
16 }
17
18 export default App;
19
```



React JS (Introdução)



Se tudo deu certo, sua página deve ficar mais ou menos assim





React JS (Introdução)



Exercício prático

- Crie um novo componente chamado **Menu.jsx** dentro da pasta **components/**, com o seguinte conteúdo:

```
src > components > Menu.jsx > ...
1  function Menu() {
2      return (
3          <nav className="menu">
4              <a href="#">Início</a> | <a href="#">Sobre</a> | <a href="#">Contato</a>
5          </nav>
6      );
7  }
8
9  export default Menu;
10
```

→ Agora importe-o e use dentro do **App.jsx** logo abaixo do **Header**.



React JS (Introdução)



Exercício prático

- Você pode observar que o menu ficará feio, então vamos dar um estilo para ele.

```
src > App.css > ...  
29 .menu {  
30   background-color: #eee;  
31   padding: 10px;  
32   text-align: center;  
33 }  
34  
35 .menu a {  
36   color: #333;  
37   text-decoration: none;  
38   margin: 0 10px;  
39   font-weight: bold;  
40 }  
41  
42 .menu a:hover {  
43   color: #007bff;  
44 }
```



React JS (Introdução)



Se tudo deu certo, sua página agora deve ficar mais ou menos assim





React JS (Introdução)



Boas Práticas de Componentização

- Nomeação Correta dos Componentes
 - O nome do componente sempre começa com letra maiúscula.
 - O nome do arquivo deve ser igual ao nome do componente.
- Exemplo correto:
 - Componente: Ola
 - Arquivo: Ola.jsx
- Evite:
 - `function header() { ... } // errado`
 - `function componenteheader() { ... } // ruim`
- O React reconhece um elemento como componente quando ele começa com letra maiúscula.
- Se estiver minúsculo, ele pensa que é uma tag HTML.

```
src > components > Ola.jsx > ...  
1  function Ola() {  
2    |    return <p>Olá!</p>;  
3  }  
4  
5  export default Ola;
```




React JS (Introdução)



Exercício 1

- Crie um novo componente chamado **Alerta**, que exibe uma mensagem simples na tela.
- Depois, importe e use esse componente no App.jsx, logo abaixo do menu.
- Adicione um estilo básico para destacar o alerta (ex: cor de fundo clara e borda simples).



React JS (Introdução)



Exercício 2

- Adicione uma imagem para exibir dentro do componente Card.

src/

|— **assets/**

| |— **minha-imagem.jpg** (*exemplo*)


- Importe a imagem dentro do componente Card:
 - **import imagemExemplo from '../assets/minha-imagem.jpg';**
- Adicione a tag **** dentro do componente, usando a variável importada:
 - ****
- Adicione um estilo básico no CSS (ex: largura máxima, bordas arredondadas ou sombra leve) para deixar a imagem visualmente agradável.



React JS (Introdução)



Exercício 1 - Gabarito

```
src > components >  Alerta.jsx > ...  
1  function Alerta() {  
2      return (  
3          <div className="alerta">  
4              <p>Atenção: Este é um alerta importante!</p>  
5          </div>  
6      );  
7  }  
8  
9  export default Alerta;  
10
```



React JS (Introdução)



Exercício 1 - Gabarito





```
src > App.jsx > ...
1 import './App.css';
2 import Header from './components/Header';
3 import Menu from './components/Menu';
4 import Alerta from './components/Alerta';
5 import Card from './components/Card';
6 import Footer from './components/Footer';
7
8 function App() {
9   return (
10     <>
11       <Header />
12       <Menu />
13       <Alerta />
14       <main className="main">
15         <Card /><Card /><Card />
16       </main>
17       <Footer />
18     </>
19   );
20 }
21
22 export default App;
```



React JS (Introdução)



Exercício 1 - Gabarito

```
src >  App.css > ...  
46  .alerta {  
47      background-color:  #fff3cd;  
48      color:  #856404;  
49      border: 1px solid  #ffeeba;  
50      padding: 10px;  
51      margin: 15px;  
52      border-radius: 5px;  
53      text-align: center;  
54  }  
55
```



ATÉ A PRÓXIMA AULA!

Front-end - Design. Integração. Experiência.

Professor: Hygor Rasec

<https://www.linkedin.com/in/hygorrasec>

<https://github.com/hygorrasec>