



# DESENVOLVIMENTO FULL STACK

---

## Aula – Introdução React

---

Caio Eduardo do Prado Ireno

[caio.ireno@faculdadeimpacta.com.br](mailto:caio.ireno@faculdadeimpacta.com.br)

# O curso...



**React:** Componentes Funcionais, Hooks, Ciclo de Vida, Formulários, Propriedades, Router e Mais



**Pré-requisitos:** HTML / CSS / JavaScript\*



**Ferramentas:** Visual Studio Code, Node.js e NPM

# Ambiente

---

**Instalar o Node.js / NPM:**

<https://nodejs.org/en/download/>

---

**Instalar o Git:** <https://git-scm.com/downloads>

---

**Instalar o Visual Studio Code:**

<https://code.visualstudio.com/>

---

**React Developer Tools:**

<https://chrome.google.com/webstore/detail/react-developer-tools/fmkadmapgofadopljbjfkapdkoienihi?hl=en>

# Visual Studio Code (Extensões)

**Live Server**, Color  
Highlight, Prettier,  
ESLint, ES7+  
React/Redux/React-Na  
tive/JS snippets...

*Extensions > Buscar >  
Instalar*

# JS Estudar

Objeto

Arrow Function

Destructuring

Rest e Spread

Module (Export/Import)

Fetch

Async / Await

Arrays (map e filter)

Expressões

# JS - Revisão

## TUDO É OBJETO

Um objeto possui propriedades e métodos que podem ser diretos ou herdados (protótipo).

```
const menu = {  
  selector: ".header",  
  ativo() {  
    const headerElement = document.querySelector(this.selector);  
    headerElement.classList.add("ativo");  
  },  
};  
  
menu.selector;  
menu.ativo();
```

```
['1', '2', '3'].map(Number); // [1, 2, 3];  
'JavaScript'.toUpperCase(); // JAVASCRIPT
```

# JS - Destructuring

```
// Extraíndo valores diretamente das propriedades do evento
function logMousePosition(event) {
  const posX = event.clientX;
  const posY = event.clientY;
  console.log(posX, posY);
}

// Usando desestruturação para simplificar a extração
function logMousePosition(event) {
  const { clientX: posX, clientY: posY } = event;
  console.log(posX, posY);
}

// Desestruturando diretamente no parâmetro
function logMousePosition({ clientX: posX, clientY: posY }) {
  console.log(posX, posY);
}

// Ligando o evento de mousemove ao método
document.documentElement.addEventListener("mousemove", logMousePosition);
```

# JS - Expressões

```
// Comparação de dois grupos
const pontosA = 100;
const pontosB = 300;
const resultado = pontosA > pontosB ? 'Vitória do Grupo A' : 'Vitória do Grupo B';

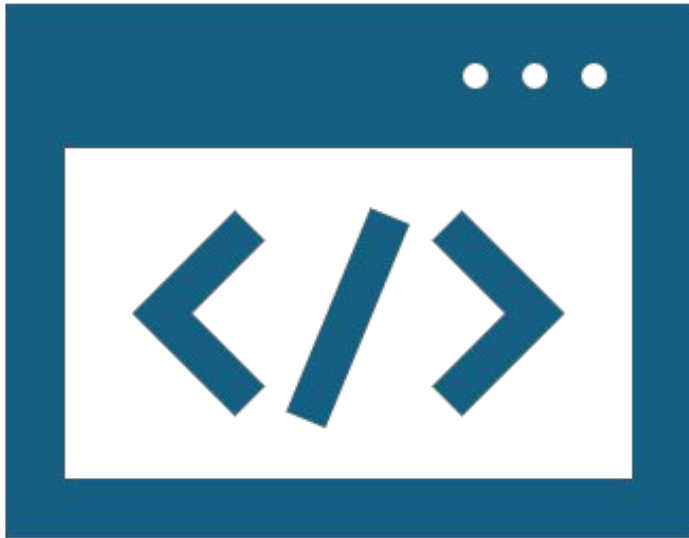
// Filtrando números maiores que um valor específico
const valores = [2, 3, 4, 5, 6];
const filtrados = valores.filter((valor) => valor > 4);

// Verificando estado ativo para exibir mensagem
const estadoAtivo = true;
const mensagemBotao = estadoAtivo ? 'O botão está ativo' : null;
```



# O QUE É O REACT?

# Uma biblioteca JavaScript para criar interfaces de usuário



*"O React permite que você crie interfaces de usuário a partir de peças individuais chamadas de componentes. Crie seus próprios componentes React como Thumbnail, LikeButton, e Video. Em seguida, combine-os em telas inteiras, páginas, e aplicativos."*

<https://pt-br.react.dev/>

# O QUE É O REACT?



**Componentes:** são como blocos de construção reutilizáveis para a interface de usuário.



**JSX (JavaScript XML):** JSX é uma extensão de sintaxe do JavaScript que permite escrever "códigos" HTML



**DOM Virtual (Virtual DOM):** React usa o **Virtual DOM**, que é uma representação em memória do DOM real.



**Reatividade com Estados e Propriedades:**



**Ciclo de Vida dos Componentes**



**Unidirecionalidade de Dados**



**Ecossistema e Ferramentas**

# HANDS ON

A thin, vertical white line is positioned to the right of the text 'HANDS ON', extending from the top of the text to the bottom of the slide.

# Ferramenta Front End



Cria um ambiente de desenvolvimento já configurado e otimizado para a criação de aplicativos com React.

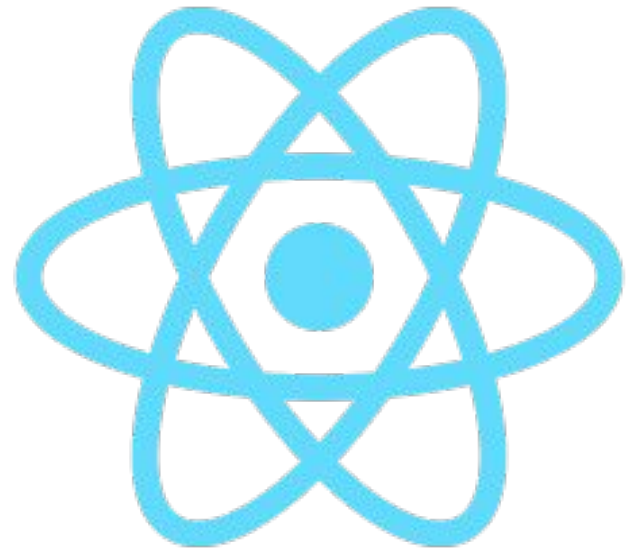
<https://vitejs.dev/>

Na pasta que deseja instalar:

-> `npm create vite@latest .`



+



NA PRÁTICA...

# JSX

**JavaScript XML (JSX)** é uma extensão de sintaxe para JavaScript que permite escrever elementos de interface como se fossem HTML ou XML. Esses elementos JSX são convertidos em chamadas para funções do React, como `React.createElement`, permitindo que o código seja mais legível e declarativo.

```
const MyComponent = () => {  
  return <div className="action-button">Finalizar Compra</div>;  
};
```



```
const MyComponent = () => {  
  return React.createElement('div', { className: 'action-button' }, 'Finalizar Compra');  
};
```

# JSX - Atributos

## Atributos

No **JSX**, atributos podem ser passados para os elementos de forma semelhante ao HTML. Entretanto, há algumas diferenças e casos especiais para evitar conflitos com palavras reservadas do JavaScript.

```
const MyComponent = () => {  
  return (  
    <a href="https://www.example.com" title="Visite o site">  
      Example  
    </a>  
  );  
};
```



# JSX - Atributos

## Casos Especiais

### className no Lugar de class

No JSX, usamos **className** para definir classes CSS, pois **class** é uma palavra reservada no JavaScript.

```
const MyComponent = () => {  
  return <section className="container">Conteúdo da Seção</section>;  
};
```

Doc: <https://legacy.reactjs.org/docs/dom-elements.html>

# JSX - Atributos

No JSX, atributos compostos (ou seja, que têm mais de uma palavra) seguem a convenção **camelCase**. Ou seja, palavras subsequentes começam com letra maiúscula e a primeira letra de cada palavra subsequente é maiúscula.

```
const MyComponent = () => {  
  return <video autoPlay />;  
};
```

# JSX - Expressões / Variáveis

No JSX, você pode inserir expressões e variáveis dentro do código, utilizando chaves {}. Isso permite que você inclua valores dinâmicos diretamente no retorno do JSX, tornando o código mais flexível e interativo.

```
const MyComponent = () => {  
  const userName = 'Caio';  
  return <p>{userName}</p>;  
};
```

# JSX - Expressões / Variáveis

Exemplo com Cálculos e Expressões

```
const MyComponent = () => {  
  const discount = 30;  
  const price = 200;  
  return <p>{price - discount}</p>;  
};
```

# JSX - Expressões / Variáveis

## Exemplo com Condicionais

```
const MyComponent = () => {  
  const isActive = false;  
  return <p className={isActive ? 'active' : 'inactive'}>Produto</p>;  
};
```

# JSX - Expressões / Variáveis

## Atribuindo JSX a Variáveis

Você também pode atribuir um pedaço de JSX a uma variável e utilizá-la dentro do JSX de outros componentes.

```
const Header = <h1>Bem-vindo ao site!</h1>;

const MyComponent = () => {
  return <div>{Header}</div>;
};
```

# JSX - Style

O style irá receber um objeto com as propriedades do elemento em camelCase.

```
function App() {  
  const estiloH1 = {  
    color: "blue",  
    fontSize: "20px",  
    fontFamily: "Helvetica",  
  };  
  
  return (  
    <>  
      <div>  
        <h1 style={estiloH1}>Desenvolvimento FullStack</h1>  
        <p>Esse é o conteúdo da página principal.</p>  
      </div>  
    </>  
  );  
}
```

# Exercício

Crie uma aplicação que mostre o status de projetos de uma equipe de trabalho, onde você deve verificar se o projeto está concluído ou em andamento. Além disso, calcule a quantidade total de horas trabalhadas e mostre uma mensagem se o total de horas ultrapassar 100 horas.

## Projeto Alpha

Horas trabalhadas: 60

Concluído

## Projeto Beta

Horas trabalhadas: 50

Em andamento

```
const projetoA = {
  nome: 'Projeto Alpha',
  horasTrabalhadas: 60,
  concluido: true,
};

const projetoB = {
  nome: 'Projeto Beta',
  horasTrabalhadas: 50,
  concluido: false,
};

const App = () => {
  const projetoSelecioneado = projetoA;

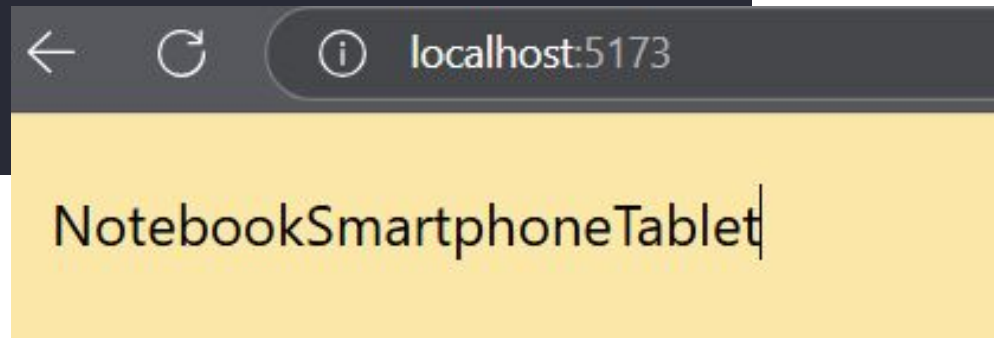
  return (
    <div>
      {/* Exibir dados do projeto */}
    </div>
  );
};
```



# JSX - array

O JSX irá listar cada um dos itens da array. Ele não irá separar ou colocar vírgula, é você que deve modificar a array para o resultado desejado.

```
function App() {  
  const produtos = ["Notebook", "Smartphone", "Tablet"];  
  return (  
    <>  
      <div>{produtos}</div>  
    </>  
  );  
}  
export default App;
```

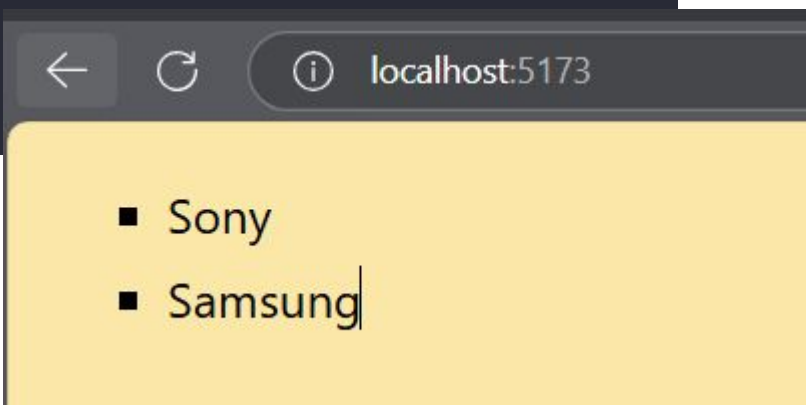


# JSX – array

## KEY

As **keys** no JSX são identificadores únicos que o React usa para rastrear os elementos de uma lista ao renderizá-los, garantindo que ele possa identificar quais itens foram adicionados, removidos ou atualizados de forma eficiente

```
function App() {  
  const marcas = [<li key="m1">Sony</li>, <li key="m2">Samsung</li>];  
  
  return <ul>{marcas}</ul>;  
}  
export default App;
```

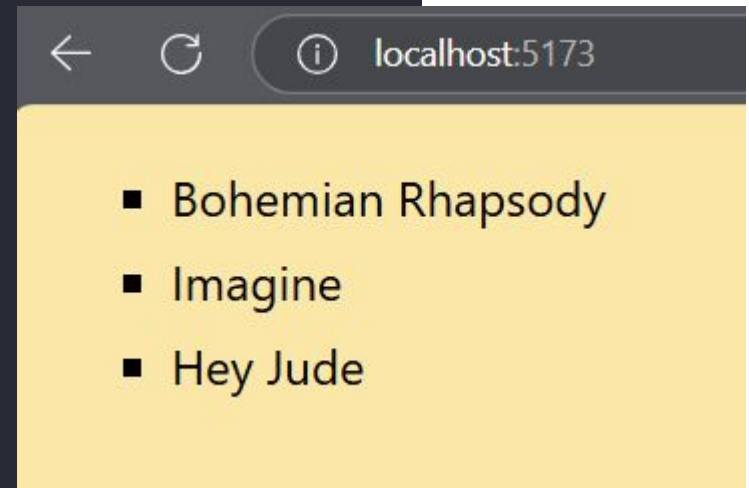
- 
- Sony
  - Samsung

# JSX – array

## MAP

É comum usarmos o **map** direto na array como uma expressão, retornando um elemento para cada item novo da Array.

```
function App() {  
  const musicas = ["Bohemian Rhapsody", "Imagine", "Hey Jude"];  
  
  return (  
    <ul>  
      {musicas.map((musica) => (  
        <li key={musica}>{musica}</li>  
      ))}  
    </ul>  
  );  
}  
  
export default App;
```

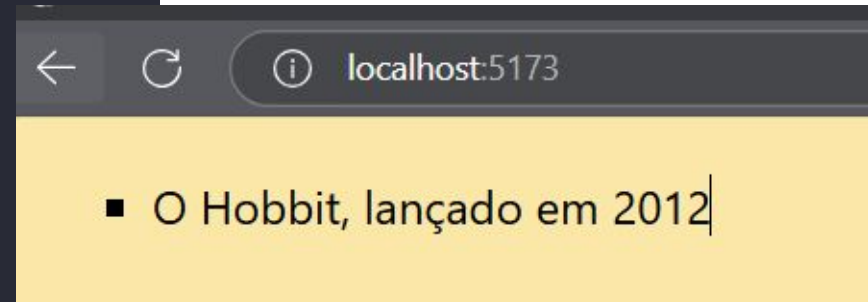


# JSX – array

## Array de Objetos

O cenário mais comum é trabalhar com array's de objetos.

```
function App() {  
  const filmes = [  
    { titulo: "O Senhor dos Anéis", ano: 2001 },  
    { titulo: "Harry Potter", ano: 2002 },  
    { titulo: "O Hobbit", ano: 2012 },  
  ];  
  
  return (  
    <ul>  
      {filmes  
        .filter((filme) => filme.ano > 2005)  
        .map((filme) => (  
          <li key={filme.titulo}>  
            {filme.titulo}, lançado em {filme.ano}  
          </li>  
        ))}  
    </ul>  
  );  
}  
  
export default App;
```



# Exercício

Crie uma aplicação que mostre o status de projetos de uma equipe de trabalho, onde você deve verificar se o projeto está concluído ou em andamento. Além disso, calcule a quantidade total de horas trabalhadas e mostre uma mensagem se o total de horas ultrapassar 100 horas.

## Projeto Alpha

Horas trabalhadas: 60

Concluído

## Projeto Beta

Horas trabalhadas: 50

Em andamento

```
const projetoA = {
  nome: 'Projeto Alpha',
  horasTrabalhadas: 60,
  concluido: true,
};

const projetoB = {
  nome: 'Projeto Beta',
  horasTrabalhadas: 50,
  concluido: false,
};

const App = () => {
  const projetoSelecioneado = projetoA;

  return (
    <div>
      {/* Exibir dados do projeto */}
    </div>
  );
};
```

# Links

[HTML Color Codes](#)

[50 Gorgeous Color Schemes From Stunning Websites](#)



Obrigado!

---

```

export const AppDois = () => {
  const projetos = [projetoA, projetoB];

  return (
    <>
      {projetos.map((projeto) => (
        <div key={projeto.nome}>
          <h2>{projeto.nome}</h2>
          <p style={{ color: projeto.horasTrabalhadas > 100 ? "green" : "red" }}>
            Horas trabalhadas: {projeto.horasTrabalhadas}
          </p>
          <p style={{ color: projeto.concluido ? "green" : "orange" }}>
            {projeto.concluido ? "Concluído" : "Em andamento"}
          </p>
        </div>
      ))}
    </>
  );
};

```