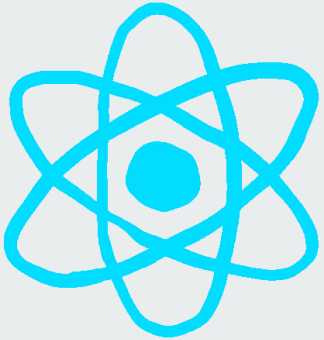
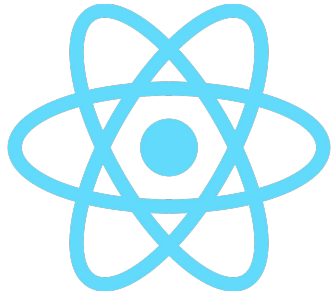
INTRODUÇÃO AO REACT



Profº: Me. Lucas Ferreira Mendes

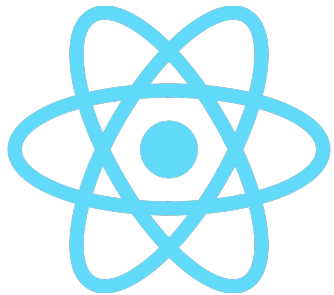
O que é React?

- React é uma biblioteca JavaScript para renderização de interfaces de usuário (UI)
- UI é constituída de pequenas unidades como botões, textos e imagens
- Através do React é possível combinar essas unidades em componentes **reutilizáveis** e **aninhados**
- As interfaces construídas com React são **dinâmicas** e **reativas**



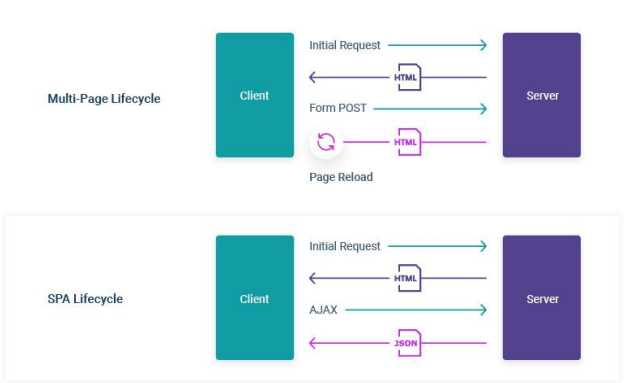
Principais características do React

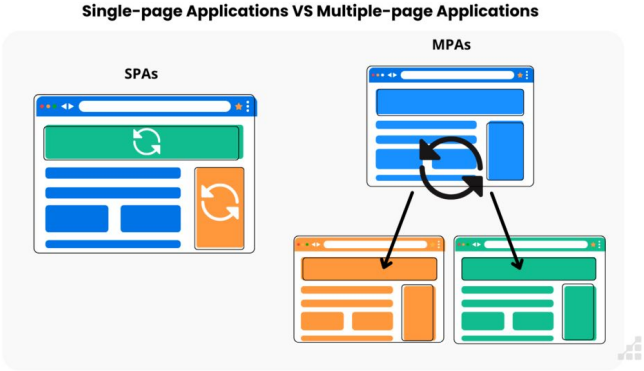
- **Componentização**
  - UIs divididas em componentes reutilizáveis
  - Facilita a manutenção e organização do código
- **Virtual DOM:**
  - Atualiza apenas as partes necessárias da interface, minimizando renderizações
  - Melhora o desempenho da aplicação

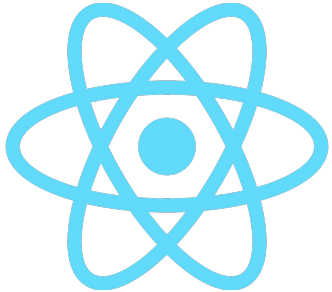


Principais características do React

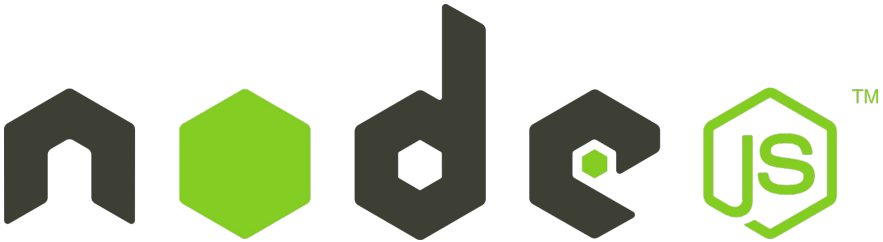
- Facilita a criação de Single Page Aplications (SPA)



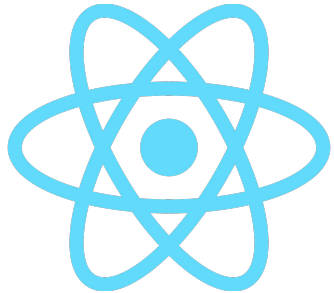




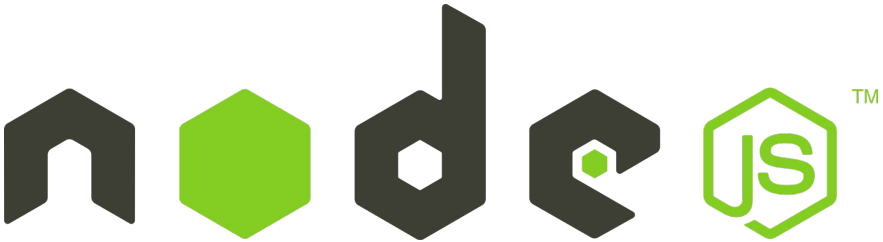
O que é Node.js?



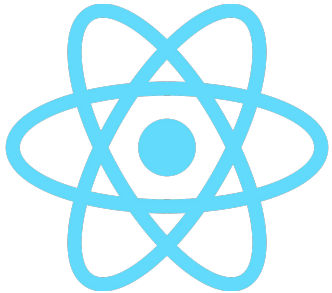
- O Node.js é uma plataforma *open source* que fornece um ambiente de execução de JavaScript, que permite os programadores criar servidores, aplicações da Web, ferramentas CLI e programas para automação de tarefas.
- Com o Node.js, o programa JavaScript não precisa do navegador para executar.
- Trabalha fortemente com execução assíncrona e orientação a eventos.
- Foi criado com o mecanismo V8 do Google Chrome.



Pra que serve o Node.js

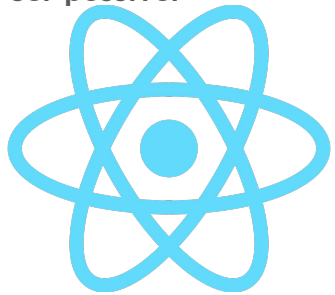


- Criar servidores Web
- Criar APIs
- Manipular arquivos
- IoT
- Etc...



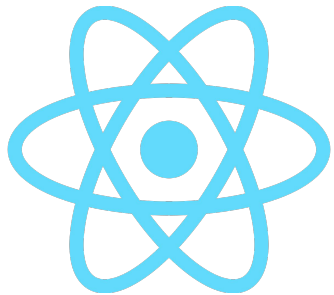
Preciso do Node para usar React?

- A resposta rápida é: **não**
- Podemos usar o React em uma aplicação web tradicional (site) simplesmente adicionando os arquivos de script js básicos da biblioteca
- Quando queremos desenvolver uma aplicação web reativa com uma UI mais complexa com React, podemos usar ferramentas como o Vite a nosso favor
- Essas ferramentas nos ajudam a criar toda a estrutura inicial de um projeto React, além de ser possível com o Node.js, adicionar facilmente novos pacotes e bibliotecas ao nosso projeto



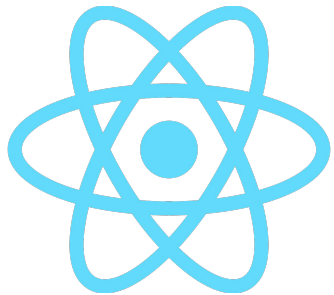
Hello World no React

- No VS Code podemos instalar a extensão **ES7 + React/Redux/React-Native snippets** para nos auxiliar na criação de código com o React
- Além disso, é importante configurar a extensão **Emmet** no VS Code para o react (caso não esteja configurada por padrão)
    - **File > Settings > Extensions**
    - **Emmet => javascript - javascriptreact**



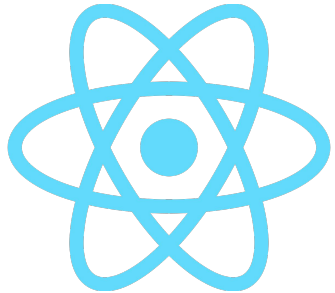
Hello World no React

- Para criar um projeto em React utilizamos o npx (executor de scripts do Node)
  - `npx create-react-app <nome>`
  - `npm start` (dentro da pasta do projeto)
- Atualmente tem-se optado pela utilização de outra ferramenta para esse processo, o Vite
  - `npm create vite@latest`
  - `npm install` (dentro da pasta do projeto)
  - `npm run dev` (dentro da pasta do projeto)



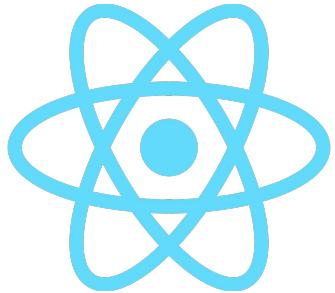
Hello World no React

- Estrutura base de arquivos e pastas do projeto
  - `node_modules` (todos os pacotes e módulos que seu projeto utiliza)
  - `public` (arquivos com acesso público)
  - `src` (pasta onde, geralmente, colocamos todo o código fonte da aplicação)
  - `src/main.jsx` (entrypoint de uma aplicação React)
  - `src/App.jsx` (componente principal, criado automaticamente)



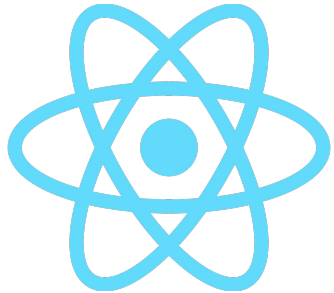
Desafio Inicial

- Entre no arquivo App.jsx e adicione algum elemento HTML da sua escolha
- Crie uma regra de estilos em App.css que altere a cor desse elemento
- Vá até o arquivo index.html e altere o título da página



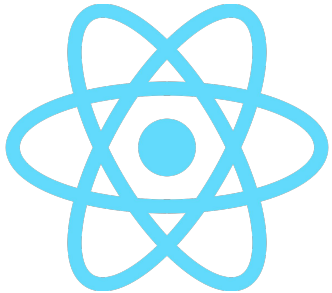
Componentes

- Componentes são os **elementos fundamentais** em uma aplicação React
- Os componentes podem ser constituídos de diferentes elementos de interface a depender de sua finalidade
- Uma **combinação de componentes** pode ser utilizada dentro de uma página/tela de sua aplicação
- Componentes são **reutilizáveis** e podem trabalhar com **dados dinâmicos**



Componentes - criação

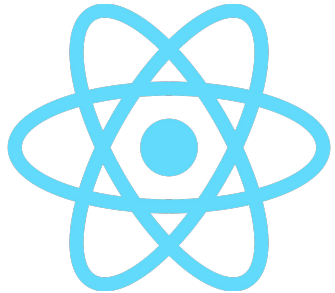
- Geralmente criamos os componentes de nossa aplicação em uma pasta chamada **components**
- Por convenção utilizamos **camelCase** para a definição do nome do componente: `FirtsComponent.jsx`
- Um componente em React é definido como uma **função JavaScript** contendo seu código (lógica e template)
- Por fim. precisamos **exportar** essa função para reutilizar o componente em outros lugares



Componentes - criação

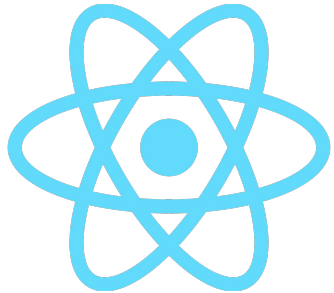
```jsx
const FirstComponent = () => {
  return (
    <div>
      <h1>Meu primeiro componente!</h1>
      <p>Este é um componente reutilizável</p>
    </div>
  );
};

export default FirstComponent;
```



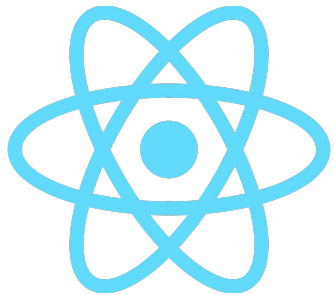
Componentes - importação

- A importação é a maneira pela qual podemos **reutilizar um componente**
- A sintaxe de importação é a seguinte: **import MyComponent from “./components/MyComponent”**
- Para utilizar um componente importado, devemos colocá-lo em forma de tag: **<MyComponent />**



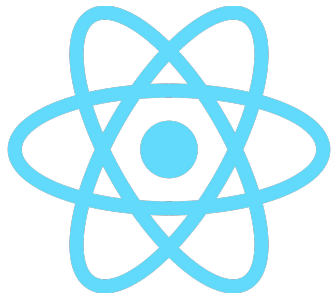
Componentes - hierarquia

- Componentes podem ser reutilizados em vários outros componentes
- Ainda, é possível importar componentes dentro de outros, formando assim uma **hierarquia de componentes**



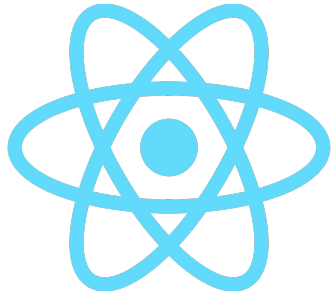
JSX

- JSX é uma extensão da linguagem JavaScript que permite escrever marcações, como HTML, dentro do código JavaScript
- O template JSX fica no **return** do component
- Temos algumas especificidades em relação ao HTML, como os atributos **className** e **htmlFor**



JSX

- O componente JSX deve retornar **apenas um elemento pai**
- Todas as tags devem ser fechadas
- Uso do camel case para nome de atributos e demais propriedades que entram em conflito com palavras ou símbolos reservados do JavaScript, por exemplo: **className**, **backgroundColor**

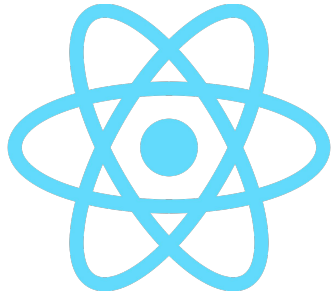


JSX - Template Expressions

```jsx
const CarDetails = ({ car }) => {
  return (
    <div>
      <h2>Modelo: {car.model}</h2>
      <p>Marca: {car.brand}</p>
      <p>Cor: {car.color}</p>
      <p>Ano: {car.year}</p>
      <p>Preço Inicial: R$ {car.price}</p>
      <p>Desconto: R$ {car.discount}</p>
      <p>Valor Final: R$ {car.price - car.discount}</p>
    </div>
  );
};

export default CarDetails;
```

- Podemos adicionar dados dinâmicos dentro de códigos JSX com uso de ***Template Expressions*** - **{}**
- Dessa forma, podemos ter um componente renderizado de forma dinâmica de acordo com os dados que ele recebe



Atividade Prática

- Criação de uma página de catálogo de produtos, consumindo dados da API Fake Store (https://fakestoreapi.com/docs).
- Planeje a interface de forma componentizada para facilitar a reutilização de código e a organização do projeto.
- Você pode tomar como base o código desenvolvido na atividade anterior (que envolvia a criação da mesma página somente com HTML, CSS e JS): https://codepen.io/prof_lucasmendes/pen/WbvMGXr