React

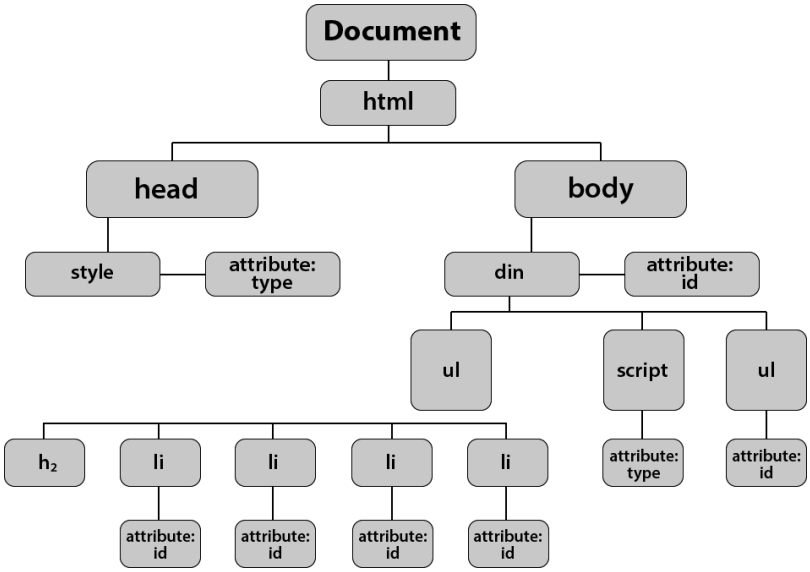
O que é React JS?

O **React.js** é uma **biblioteca JavaScript** criada pela **Meta**, em 2011, para facilitar o desenvolvimento de interfaces de usuário. Ele é muito utilizado em **Single Page Applications (SPAs)**, que são páginas web mais rápidas e dinâmicas, pois carregam o conteúdo uma única vez e atualizam apenas as partes necessárias.

Isso acontece porque o React trabalha com **componentes**, que são pequenos blocos de código reutilizáveis. Quando algo muda na tela, em vez de recarregar tudo, o React atualiza somente o que foi alterado, deixando a navegação mais rápida e eficiente.

O que é a DOM?

A **DOM (Document Object Model)** é a estrutura que representa os elementos de uma página web, como se fosse um "mapa" do HTML.



Quando um site é carregado, o navegador cria essa estrutura para exibir e manipular os elementos na tela, mas sempre que ocorre uma mudança no conteúdo da página, o navegador precisa recriar a DOM inteira, o que pode deixar o site lento (menos performático).

O que é a Virtual DOM?

Para resolver esse problema, o React utiliza a **Virtual DOM**, que é uma versão leve e otimizada da DOM real.

O funcionamento é simples:

1. O React mantém uma cópia da interface, chamada de **Virtual DOM;**
2. Quando ocorre alguma alteração na página, ao invés de em vez de modificar diretamente a **DOM**, o React primeiro atualiza a **Virtual DOM;**
3. Em seguida, ele compara a nova versão com a anterior e identifica as mudanças;
4. Então atualiza na **DOM** somente os elementos que foram modificados, tornando o processo mais rápido e eficiente.

Ciclo de Vida de um Componente React

No React, os componentes passam por três fases principais ao longo de sua existência:

1. **Montagem (Mounting)** – O componente é criado e inserido na tela.
    - Exemplo: Um botão que aparece quando a página carrega.
2. **Atualização (Updating)** – O componente é atualizado quando suas informações mudam.
    - Exemplo: Um contador que aumenta quando o usuário clica em um botão.
3. **Desmontagem (Unmounting)** – O componente é removido da tela.
    - Exemplo: Um aviso que desaparece após alguns segundos.

JSX (JavaScript XML)

O JSX é uma extensão de sintaxe para JavaScript, onde escrevemos código HTML dentro do JS.

obs.: O React converte o código JSX para JS pruro antes de exibir em tela, esse processo se chama transpilação.

Componentes no React

O React é baseado em componentes, que são pequenos blocos de código que podem ser reutilizados ao longo detoda a aplicação.

Existem duas formas de criarmos componentes no React, os componentes com **exportação nomeada (named export)** e os **componentes com exportação padrão (default exports).**

- **Default Export:** permite a importação do componente com qualquer nome. O problema deste tipo de exportação é a falta de padronização e organização do código.

```
// componente
function saudacao() {
   return <h1>Olá, mundo!</h1>
}

export default saudacao
```

```
// importação do componente
import saudacao from './saudacao'

import Batatinha123 from './saudacao'
```

- **Named Export:** o nome declarado na função de criação do componente é o mesmo que será utilizado na sua importação.

```
// componente
export function saudacao() {
   return <h1>Olá, mundo!</h1>
}
```

```
// importação do componente
import { saudacao } from './saudacao'
```

Como Criar um Projeto React?

Create React App (CRA): conjunto de ferramentas e funcionalidades desenvolvidas pelo prórprio time do React. Entretanto faz uso de outras duas ferramentas por “de baixo dos

panos", sendo elas o Babel (transpilação do código) e o Webpack (empacotamento do código).

Vite: ferramenta de construção de projetos front-end, desenvolvida pelo criador do framework web Vue.JS. Sendo mais rápido e eficiente que o CRA.

Atualmente na documentação oficial do React, eles não recomendam mais o uso do CRA para criação de **novas** aplicações React.JS

Para criamos uma aplicação React utilizando o Vite, precisamos rodar o seguinte comando no terminal: `npm create vite@latest`

Passo a passo:

1. `npm create vite@latest` → dizemos que queremos criar um projeto utilizando a última versão do Vite
2. Definimos um nome para o nosso projeto
3. Selecionamos o framework que queremos utilizar, no nosso caso, `React`
4. Agora vamos selecionar a "variante", como vamos trabalhar com JS puro, neste primeiro momento, selecionamos `JavaScript`
5. Executamos o comando `cd nomeDoProjeto` para abrirmos o terminal na pasta do nosso projeto e então executamos `npm install` ou `npm i` para instalar os pacotes necessários
6. Por fim, para executarmos nossa aplicação rodamos o comando `npm run dev`

Renderização Condicional

O React permite que um componente seja exibido em tela de acordo com a condição. Exemplo:

```
function mensagem(props) {
  if (props.logado) {
    return <h1>Bem-vindo de volta!</h1>;
  } else {
```

```
    return <h1>Por favor, faça login.</h1>;
  }
}

// operador ternário
<p>{props.logado ? "Bem-vindo!" : "Faça login"}</p>
```

Renderização de Listas

Quando precisamos renderizar elementos que estão em uma lista, um array, utilizamos o método **map** ao invés do método **forEach**.

- Por que o método forEach não é recomendado?
    - O método forEach não cria um novo array com elementos manipulados, fazendo com que se torne inadequado na renderização dos elementos, já que o React precisa deste novo array para atualizar a DOM corretamente.

Além de utilizarmos o método **map** na renderização de listas, precisamos passar uma **chave** para cada elemento do array. Ela será a responsável por indiciar qual elemento especificamente precisa ser reconstruído na DOM, caso haja alguma alteração.

Exemplo:

```
const numbers = [1, 2, 3, 4, 5];

// utilizando  o prórprio elemento como chave
const listItems = numbers.map((number) =>
  <li key={number.toString()}>
    {number}
  </li>
);

// utilizando o index do array como chave
const numbers = [1, 2, 3, 4, 5];
const listItems = numbers.map((number, index) =>
  <li key={index}>
    {number}
```

```
  </li>
);
```

obs.: a chave deve ser única para cada elemento, o mais indicado é utilizarmos um ID (identificador único) como chave para cada elemento da nossa lista