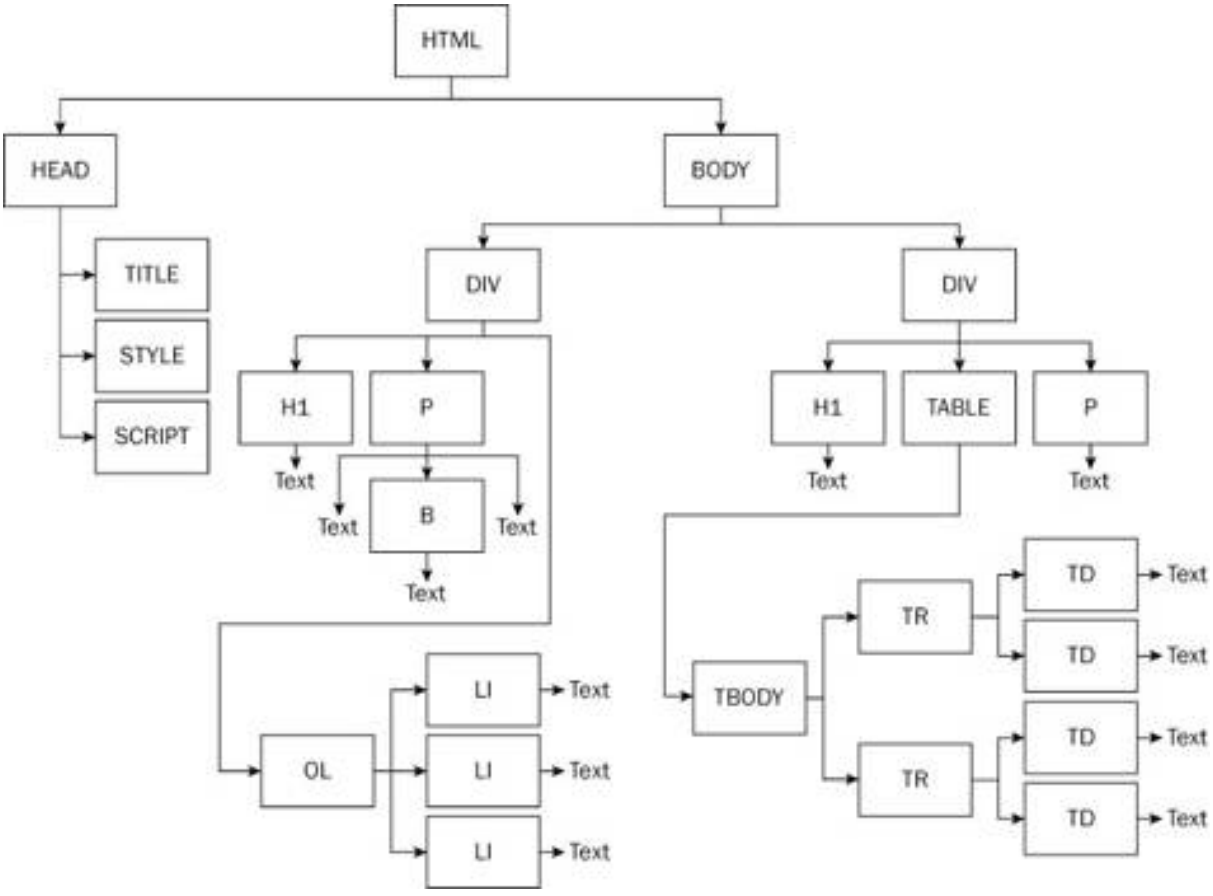
JAVASCRIPT FOR FRONT-END

Introdução ao DOM:

O DOM é uma representação em forma de árvore da estrutura de um documento HTML. Ele permite que os scripts do lado do cliente, especialmente JavaScript, manipulem dinamicamente a estrutura, o estilo e o conteúdo de um documento HTML.



HTML:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, init
ial-scale=1.0">
    <title>DOM e JavaScript</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div id="container">
        <h1>Manipulando o DOM com JavaScript</h1>
        <button id="btn">Clique aqui</button>
    </div>

    <script src="script.js"></script>
</body>
```

```
</html>
```

CSS (styles.css):

```
#container {
    text-align: center;
    margin-top: 50px;
}

button {
    padding: 10px 20px;
    font-size: 16px;
    cursor: pointer;
}
```

JavaScript (script.js):

```
// Selecionando elementos do DOM
const container = document.getElementById('container');
const button = document.getElementById('btn');

// Adicionando um evento de clique ao botão
button.addEventListener('click', function() {
    // Criando um novo elemento
    const newParagraph = document.createElement('p');
    newParagraph.textContent = 'Novo parágrafo criado dinam
icamente!';

    // Adicionando o novo elemento ao DOM
    container.appendChild(newParagraph);
});
```

Conceitos Fundamentais de JavaScript para React:

1. **Seleção de Elementos do DOM**: Utilize `document.getElementById`, `document.querySelector`, ou métodos similares para selecionar elementos do DOM.
2. **Manipulação do DOM**: Adicione, remova ou modifique elementos do DOM usando métodos como `createElement`, `appendChild`, `removeChild`, `setAttribute`, `textContent`, entre outros.
3. **Eventos**: Adicione ouvintes de eventos aos elementos do DOM usando métodos como `addEventListener`.
4. **Funções**: Utilize funções para encapsular comportamentos e reutilizá-los em diferentes partes do seu código.
5. **Escopo e Closure**: Entenda como o escopo funciona em JavaScript e como as closures podem ser usadas para encapsular estados e comportamentos em componentes.
6. **Manipulação de Arrays e Objetos**: Aprenda a manipular arrays e objetos para armazenar e manipular dados de forma eficiente.

Estes são apenas alguns dos conceitos fundamentais de JavaScript que são essenciais para o aprendizado e entendimento do React JS. Com uma compreensão sólida desses conceitos, você estará bem preparado para mergulhar no desenvolvimento com React.

EXEMPLOS DESSES CONCEITOS

1. Seleção de Elementos do DOM:

```
// Selecionando um elemento pelo ID
const container = document.getElementById('container');

// Selecionando o primeiro elemento que corresponde a um se
letor CSS
const button = document.querySelector('button');
```

2. Manipulação do DOM:

```
// Criando um novo elemento
const newParagraph = document.createElement('p');

// Definindo o conteúdo de texto do novo parágrafo
newParagraph.textContent = 'Novo parágrafo criado dinamicam
ente!';

// Adicionando o novo parágrafo ao elemento com ID 'contain
er'
container.appendChild(newParagraph);

// Removendo um elemento do DOM
container.removeChild(newParagraph);
```

3. Eventos:

```
// Adicionando um ouvinte de evento ao botão
button.addEventListener('click', function() {
    // Função a ser executada quando o botão for clicado
    console.log('Botão clicado!');
});
```

4. Funções:

```
// Definindo uma função que multiplica dois números
function multiply(x, y) {
    return x * y;
}

// Chamando a função e armazenando o resultado em uma variá
vel
const result = multiply(3, 4);
console.log(result); // Saída: 12
```

5. Escopo e Closure:

```
// Função que retorna outra função
function outerFunction() {
    const outerVar = 'Outer';

    function innerFunction() {
        console.log(outerVar); // Inner function tem acesso
à variável da função externa
    }

    return innerFunction;
}

const innerFunc = outerFunction();
innerFunc(); // Saída: 'Outer'
```

6. Manipulação de Arrays e Objetos:

```
// Array de números
const numbers = [1, 2, 3, 4, 5];

// Adicionando um elemento ao final do array
numbers.push(6);

// Removendo o último elemento do array
numbers.pop();

// Objeto com propriedades
const person = {
    name: 'João',
    age: 30,
    city: 'São Paulo'
};

// Acessando propriedades do objeto
```

```
console.log(person.name); // Saída: 'João'
console.log(person.age); // Saída: 30
```