


# ARQUITETURA DE FRONT-END

 Professor: Nélío Frazão

 Email: [neliofrazac@gmail.com](mailto:neliofrazac@gmail.com)



## **SOBRE NÉLIO**

Nélio Frazão, Frontend, com 15 anos de experiência. Possui MBA em Marketing Digital, pela Faculdade Estácio/Idéz em /João Pessoa/PB. e especialista em Desenvolvimento de Aplicações para WEB pelo UNIPÊ



# O QUE VEREMOS

- Conceitos básicos
- Introdução ao ReactJS e seu ecossistema
- Composição dos componentes em ReactJS
- Introdução aos Hooks
- Gerenciamento de estado em ReactJS usando useState Hook
- useEffect Hook
- Trabalhando com APIs em ReactJS
- Rotas em ReactJS
- Formulários



# AVALIAÇÃO

- Criar uma aplicação utilizando **The Rick and Morty API** (<https://rickandmortyapi.com/>)
- Página de personagens listando todos os personagens
- Detalhamento dos personagens
- Página de episódios
- Detalhamento dos Episódios
- Listagem de Localizações
- Detalhamento das localizações



## Conceitos Básicos - Web Components

Componentes Web são um conjunto de normas produzidas por engenheiros do Google como também uma especificação da W3C que permitem a criação de componentes reutilizáveis em documentos e aplicações web. A intenção por trás deles é trazer a engenharia de software baseada em componentes para a web

# Conceitos Básicos - Web Components

## Disciplina

Nome da disciplina

carga horária: 16h

Professor: Pessoa da Silva

Status **Obrigatória**

Lorem ipsum dolor sit amet  
consectetur adipisicing elit.

Componente

Nome da disciplina

carga horária: 16h

Professor: Pessoa da Silva

Status **Opcional**

Lorem ipsum dolor sit amet  
consectetur adipisicing elit.

Componente

Nome da disciplina

carga horária: 16h

Professor: Pessoa da Silva

Status **Opcional**

Lorem ipsum dolor sit amet  
consectetur adipisicing elit.

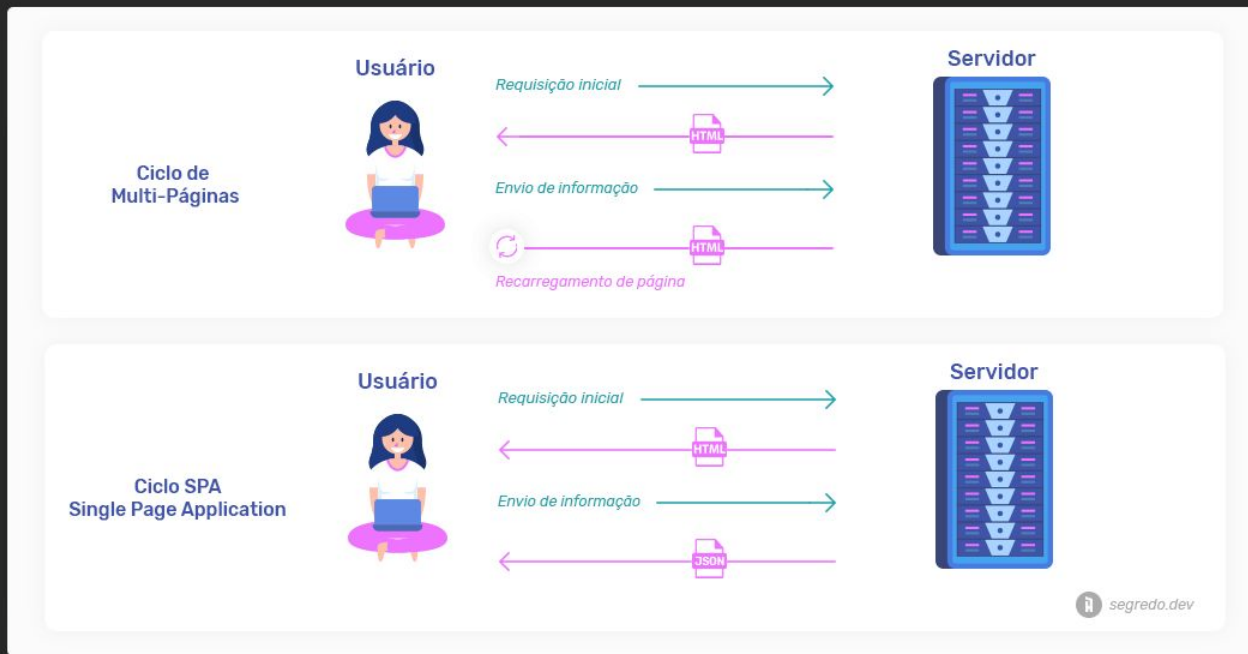
Componente

## Conceitos Básicos - Single Page Application

Uma SPA (Single Page Application) é uma aplicação web que carrega apenas uma página e usa JavaScript para atualizar dinamicamente ao interagir com o usuário, sem a necessidade de carregar uma nova página do servidor. Geralmente usa frameworks JavaScript populares, como React, Angular e Vue, para facilitar o desenvolvimento.



# Conceitos Básicos - Single Page Application

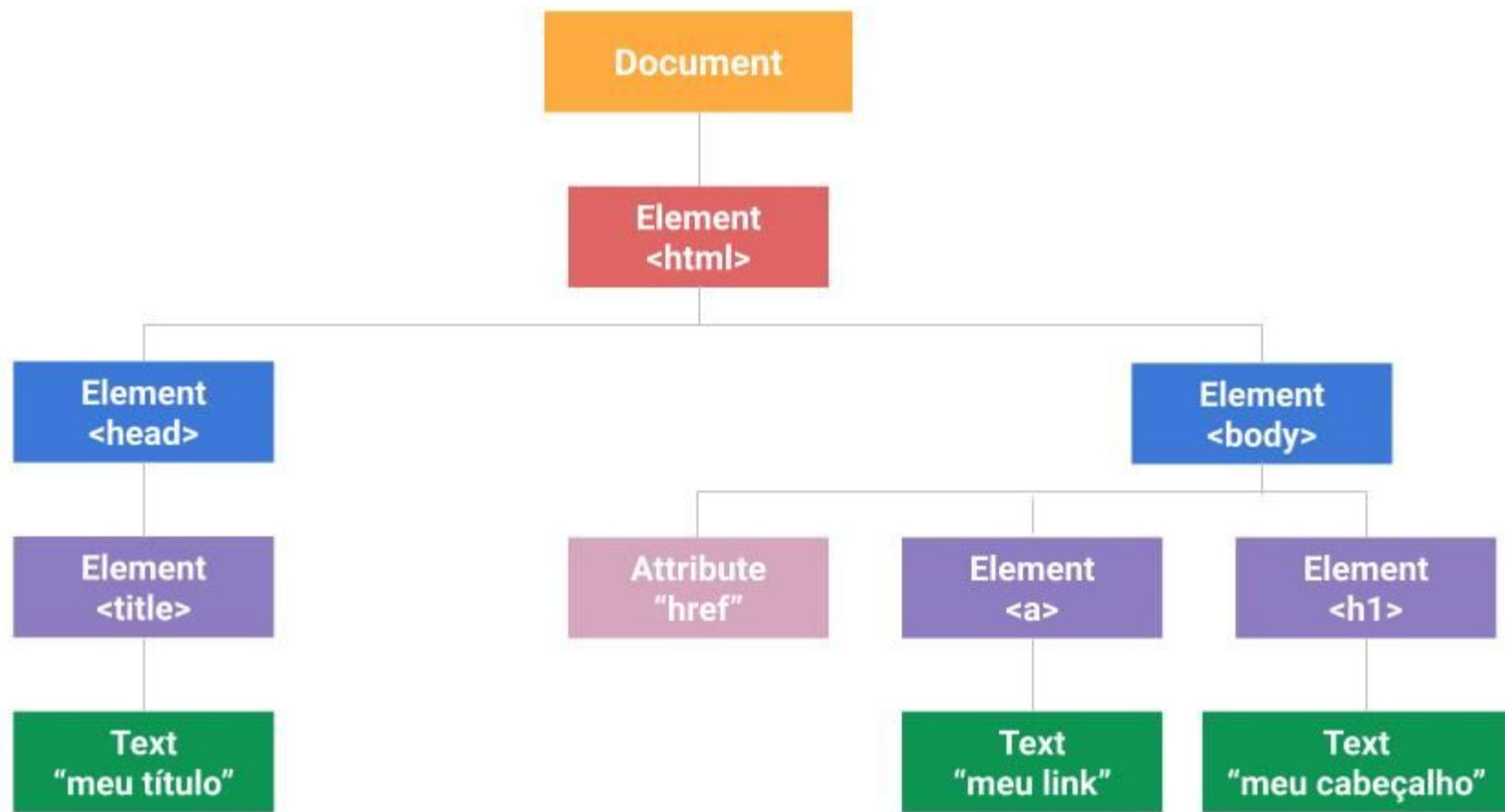


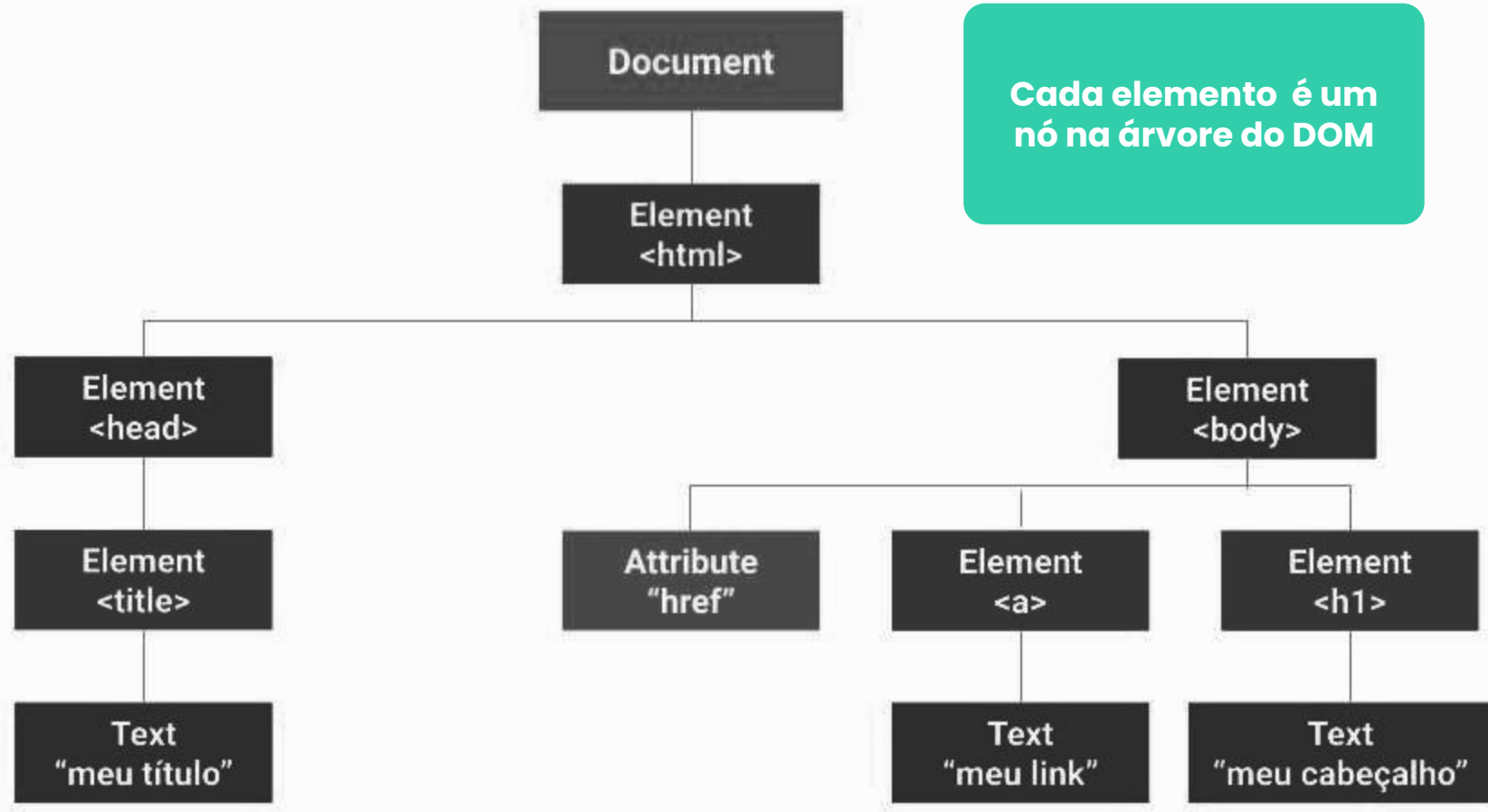


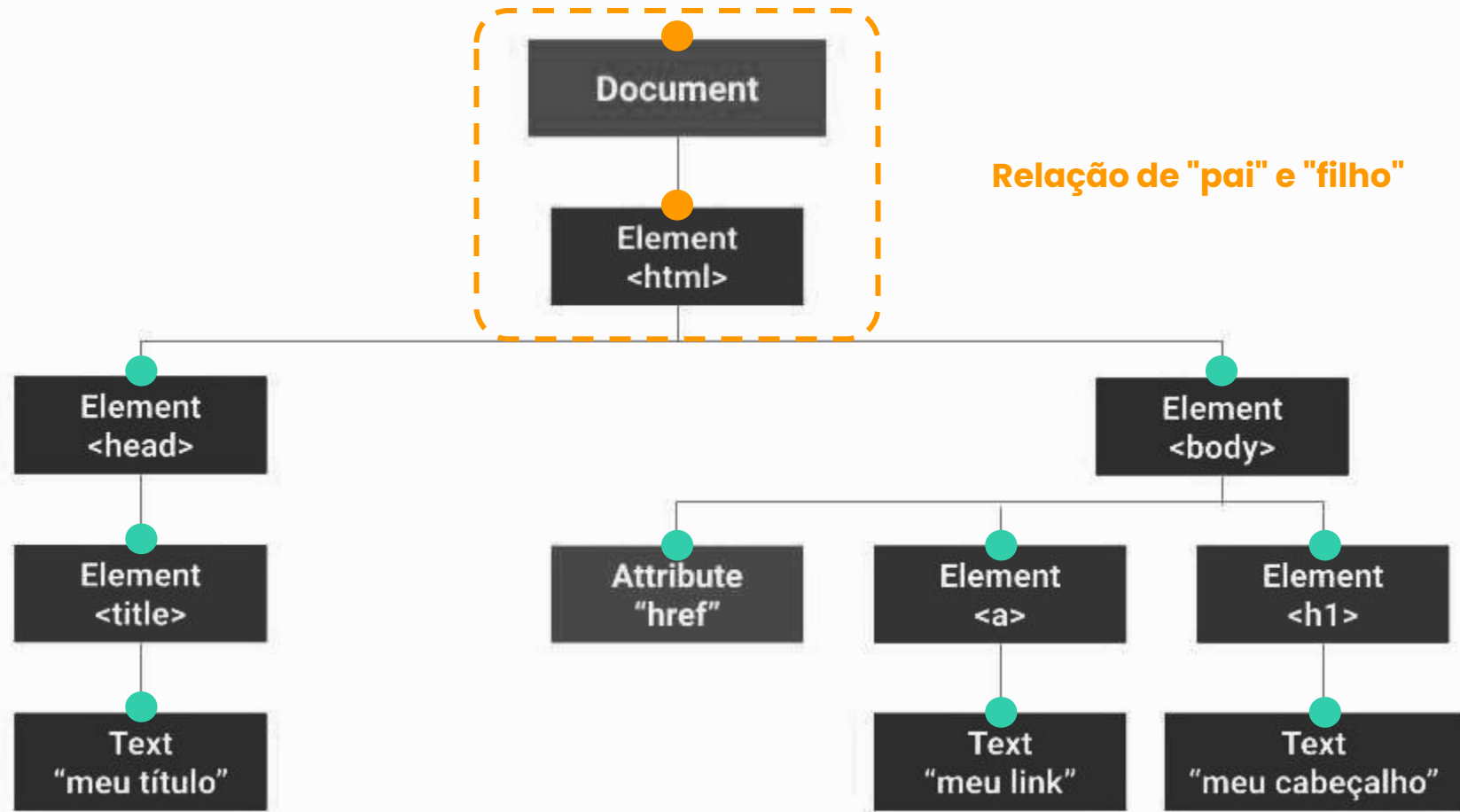


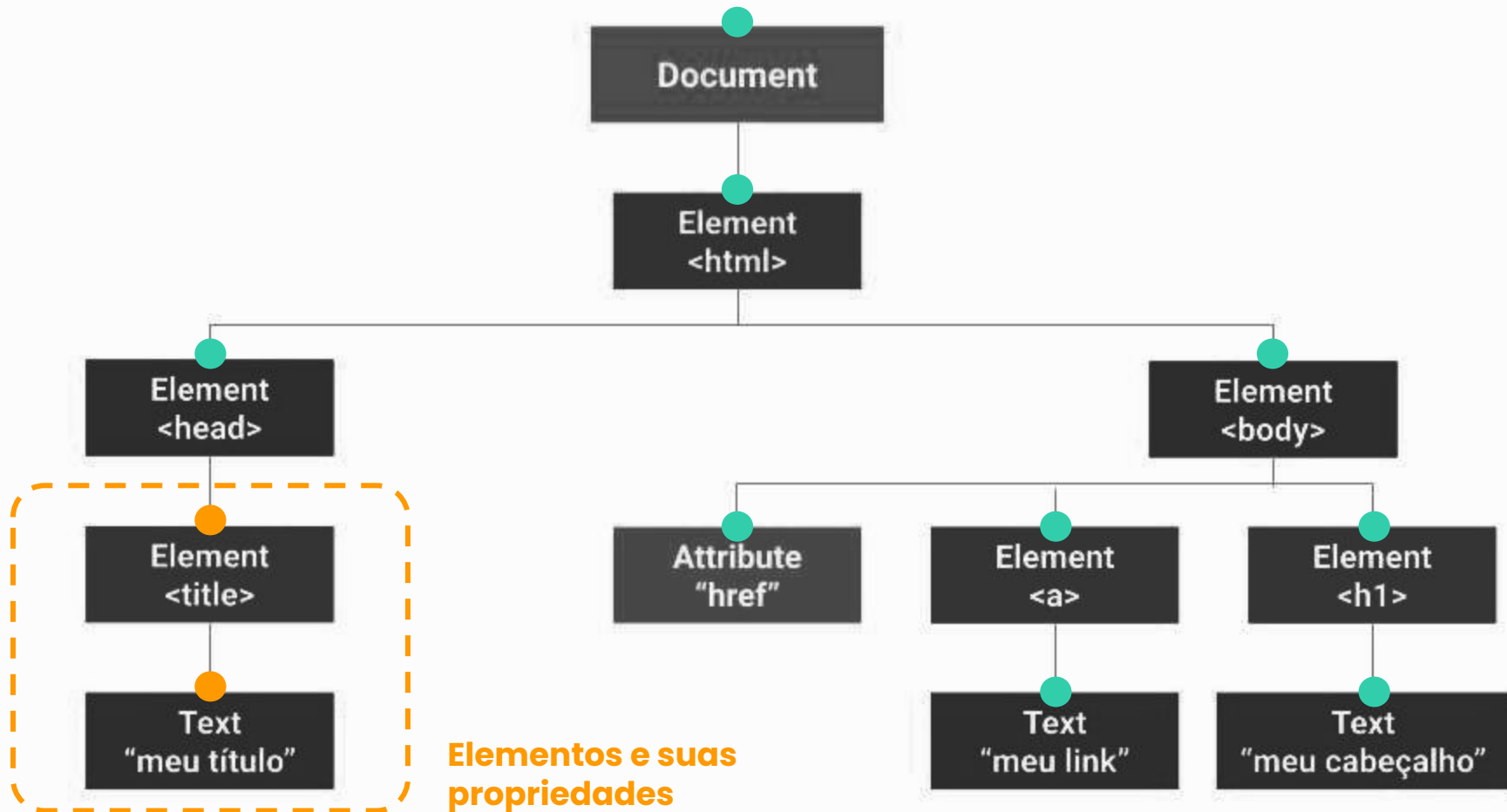
## Conceitos Básicos - DOM

É a sigla para **Document Object Model**, que numa tradução livre seria **Modelo de Documento por Objetos** e consiste em uma representação estruturada e hierarquizada dos elementos que compõem uma página na web.











## Conceitos Básicos - Virtual DOM

O **Virtual DOM** (VDOM) é uma representação do DOM mantida em memória. Dessa forma, quando precisamos fazer alguma alteração, ela é feita no Virtual DOM, que é bem mais rápido que o DOM.

Com isso ele analisa todos os lugares que serão afetados e sincroniza com o DOM em um processo chamado **Reconciliação**. A vantagem disso é que essa análise permite que haja o menor número possível de acessos ao DOM, melhorando muito a performance das aplicações.

É importante destacar que o Virtual DOM não é uma característica do navegador, mas sim um conceito implementado por bibliotecas como o React.

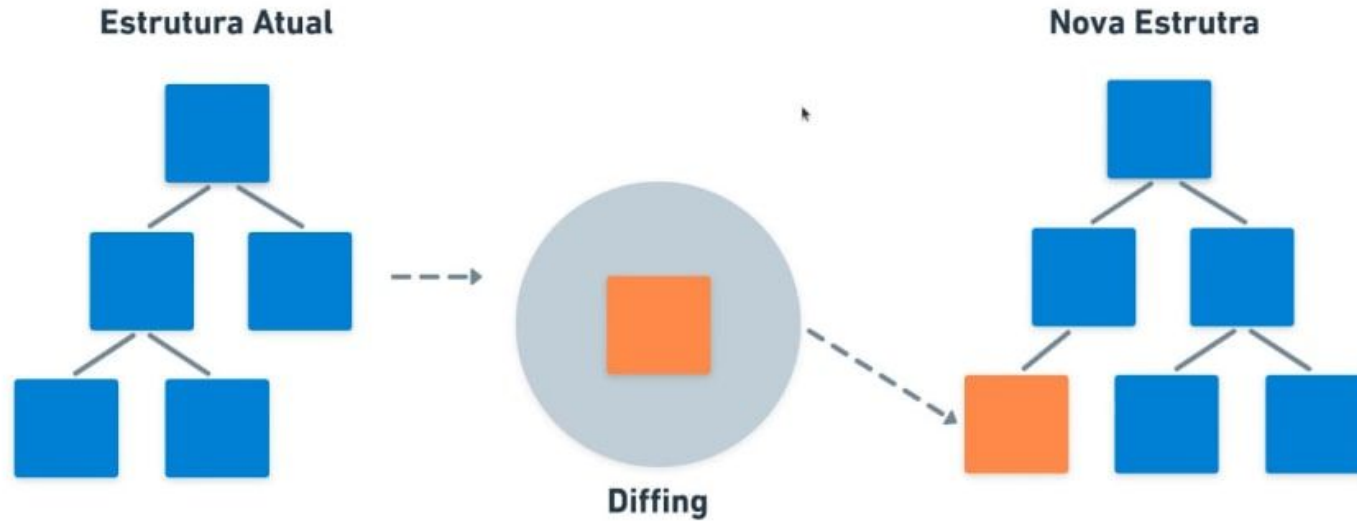


## Conceitos Básicos - Reconciliation

A reconciliação é o processo de atualização da árvore de elementos do DOM para refletir as mudanças no estado dos componentes.

- O React possui o DOM Virtual ( cópia do DOM em tela ) em memória.
- Quando um componente é atualizado, um novo DOM Virtual é criado.
- Então é feita uma comparação pelo Algoritmo de Diferenciação. Essa comparação é realizada em memória, dessa forma o componente ainda não foi atualizado na DOM.
- Após a comparação, o React cria um novo DOM Virtual com as alterações necessárias.
- Em seguida, ele atualiza o DOM do navegador com o menor número possível de alterações sem renderizar todo o DOM novamente. Isso melhora drasticamente a performance da aplicação.

## Conceitos Básicos - Reconciliation







## Conceitos Básicos - Reconciliation

### **Importante lembrar que:**

Embora a reconciliação seja uma tarefa crítica, a maioria dos desenvolvedores do React não precisa se preocupar com ela, pois o React cuida da maior parte desse processo automaticamente. No entanto, é importante entender os conceitos por trás da reconciliação para otimizar o desempenho de aplicativos React mais complexos e entender como o React atualiza o DOM em resposta a mudanças de estado ou props.



## **Ecosystem React - O que é React**

React é uma biblioteca JavaScript de código aberto usada para construir interfaces de usuário (UI) para aplicativos da web. Desenvolvida pelo Facebook, ela foi lançada em 2013 e se tornou uma das bibliotecas JavaScript mais populares no desenvolvimento de aplicações web.

React permite criar componentes reutilizáveis que representam partes independentes da interface do usuário, como botões, formulários, tabelas e muito mais. Esses componentes podem ser compostos para construir interfaces complexa



## Ecossistema React - O que é JSX

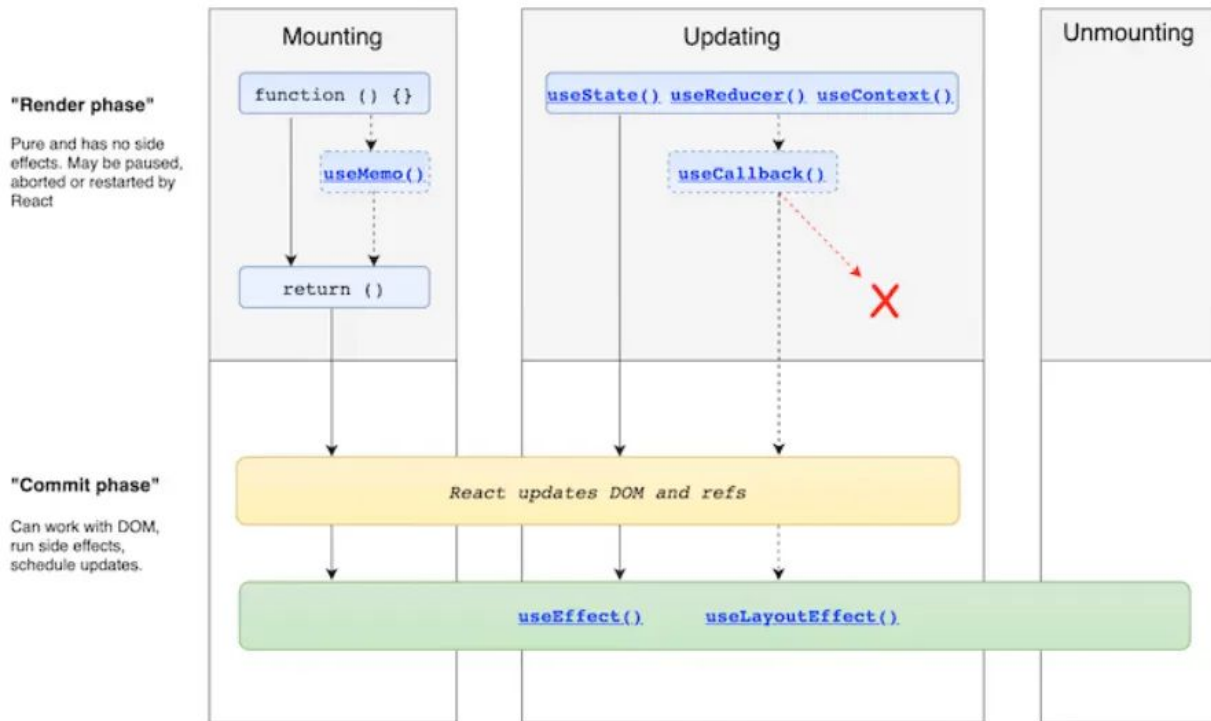
JSX é uma extensão de sintaxe JavaScript usada pelo React para permitir a escrita de código HTML dentro de um arquivo JavaScript. Com JSX, é possível escrever código que mistura elementos HTML e JavaScript em um único arquivo, facilitando a criação de componentes de interface do usuário no React.

Por exemplo, em vez de escrever HTML e JavaScript em arquivos separados, um desenvolvedor pode escrever código JSX que define a aparência e o comportamento de um componente em um único arquivo. O JSX é então traduzido em código JavaScript válido pelo compilador do React antes de ser executado no navegador.

# Ecossistema React - Ciclo de vida de um componente



## React Hooks Lifecycle





# Typescript

TypeScript é um superconjunto de JavaScript, desenvolvido pela Microsoft, que oferece uma camada adicional de tipagem estática ao código. Ao contrário do JavaScript tradicional, que é uma linguagem de script interpretada, o TypeScript permite a definição de tipos para variáveis, parâmetros de função e outros elementos do código, proporcionando um desenvolvimento mais seguro e robusto.



## Typescript - Vantagens

### **Tipagem Estática em Comparação com Linguagens Dinâmicas:**

Uma das principais vantagens do TypeScript em relação a linguagens dinamicamente tipadas, está na introdução da tipagem estática. Isso significa que as variáveis têm tipos definidos durante o desenvolvimento, permitindo a detecção precoce de erros e garantindo maior segurança e confiabilidade no código.



# Typescript - Vantagens

## **Identificação de Erros no Desenvolvimento:**

A tipagem estática do TypeScript proporciona a detecção imediata de erros durante o desenvolvimento, reduzindo substancialmente a possibilidade de erros em tempo de execução. Essa abordagem previne bugs comuns e facilita a manutenção do código.



# Typescript - Vantagens

## **Melhorias na Qualidade do Código:**

A tipagem estática também contribui para a melhoria geral da qualidade do código, fornecendo informações claras sobre a estrutura do programa e tornando a leitura e compreensão do código mais eficientes, especialmente em projetos grandes.





## Ecossistema React - Criando um projeto react

- <https://vitejs.dev/guide/>

NPM Yarn PNPM Bun

```
$ npm create vite@latest
```

bash



## Introdução à hooks - O que São Hooks

Adicionado na versão 16.8 do React, os hooks permite uma escrita de código mais simples e menos verbosa. Esse recurso disponibiliza maneiras de trabalharmos com componentes funcionais que possuem um estado interno e acesso ao próprio ciclo de vida de uma maneira mais direta sem a necessidade de criar uma classe (stateful component)

O uso de hooks em componentes funcionais também pode melhorar a performance da aplicação, já que eles permitem que o React evite criar novas instâncias de componentes a cada renderização, além de possibilitar a criação de componentes mais simples e independentes.



## Introdução à hooks - Motivações

- Facilitar a reutilização da lógica com estados entre componentes
- Componentes menos complexos e mais fáceis de entender
- Escrita de código mais simples



## Introdução à hooks - Hooks que usaremos

- `useState()`
- `useEffect()`



## Introdução à hooks - Hooks de estado

```
import React from 'react';

export default function App () {
  return (
    <>
      Count
    </>
  );
}
```



## Introdução à hooks - Hooks de estado

```
import React, { useState } from 'react';
```



## Introdução à hooks - Hooks de estado

```
import React, { useState } from 'react';
```

```
const [ count ]
```

State Variable



## Introdução à hooks - Hooks de estado

```
import React, { useState } from 'react';
```

Setter Function

```
const [ count, setCount ]
```

State Variable



## Introdução à hooks - Hooks de estado

```
import React, { useState } from 'react';
```

Setter Function

```
const [ count, setCount ] = useState(0)
```

Initial Value

State Variable



## Introdução à hooks - Hooks de estado

```
import React, { useState } from 'react';
```

```
const [ count, setCount ] = useState(0)
```



Destructuring Assignment

## Introdução à hooks - Hooks de estado

Sempre passe um valor inicial para o estado,  
isso evita comportamentos inesperados na sua  
aplicação

```
import React, { useState } from 'react';
```

```
const [ count, setCount ] = useState(0)
```



# Introdução à hooks - Hooks de estado

```
import React, { useState } from 'react';
```

```
export default function App () {
```

```
  const [ count, setCount ] = useState(0)
```

```
  return (
```

```
    <>
```

```
    Count: {count}
```

Acessando o Estado

```
  </>
```

```
);
```

```
}
```

# Introdução à hooks - Hooks de estado

```
import React, { useState } from 'react';  
  
export default function App () {  
  const [ count, setCount ] = useState(0)  
  const handleClick = () => setCount(count + 1)
```


Modificando o estado

```
return (  
  <>  
    Count: {count}   
    <button onClick={handleClick}>  
      new count  
    </button>  
  </>  
);  
}
```



## Introdução à hooks - Hooks de efeito

```
import { useEffect } from 'react';
```

- 
- > componentDidMount()
  - > componentDidUpdate()
  - > componentWillUnmount()

# Introdução à hooks - Hooks de efeito

```
import { useEffect } from 'react';
```

```
useEffect(( ) => {  
  }, [])
```

Corpo da função

Quando a função deve  
ser executada



## Introdução à hooks - Hooks de efeito

```
import { useEffect } from 'react';  
  
useEffect(() => {  
  document.title = `Você clicou ${count} vezes.`  
})
```



State Variable





## Introdução à hooks - Hooks de efeito

```
import { useEffect } from 'react';  
  
useEffect(() => {  
  document.title = `Você clicou ${count} vezes.`  
}, [ count ])
```

Array de dependências



# OBRIGADO

✉ [neliofrazac@gmail.com](mailto:neliofrazac@gmail.com)

 <http://www.linkedin.com/in/neliofrazao>

