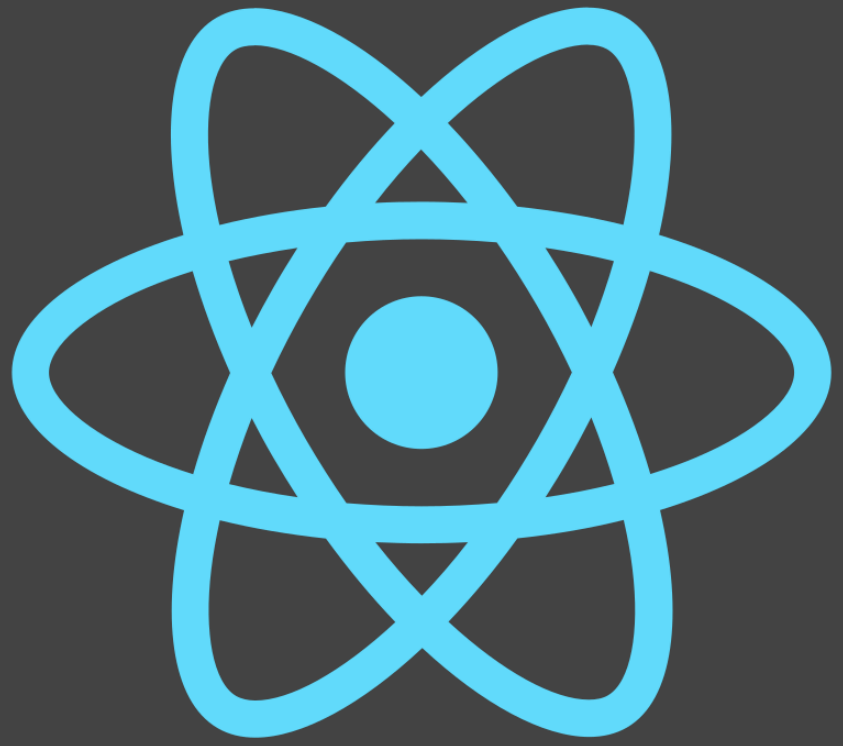


Aula 3
ReactJS

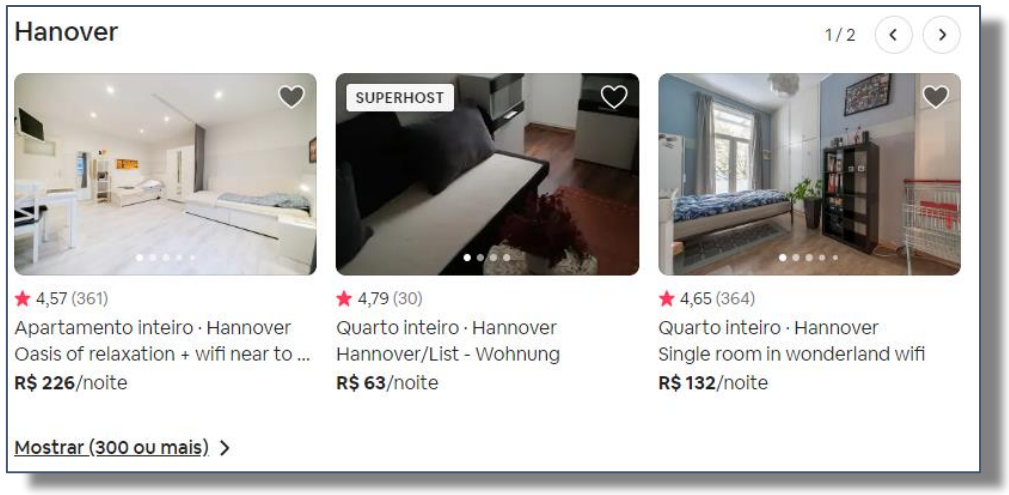
Firjan SENAI

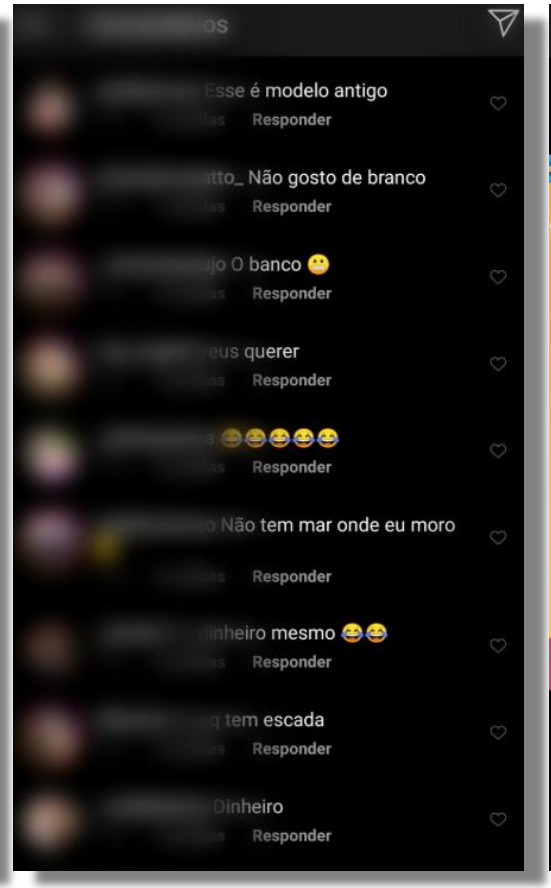
RESIDÊNCIA DE SOFTWARE

serratec
Parque Tecnológico Região Serrana

Principais Conceitos Práticos

Componentes

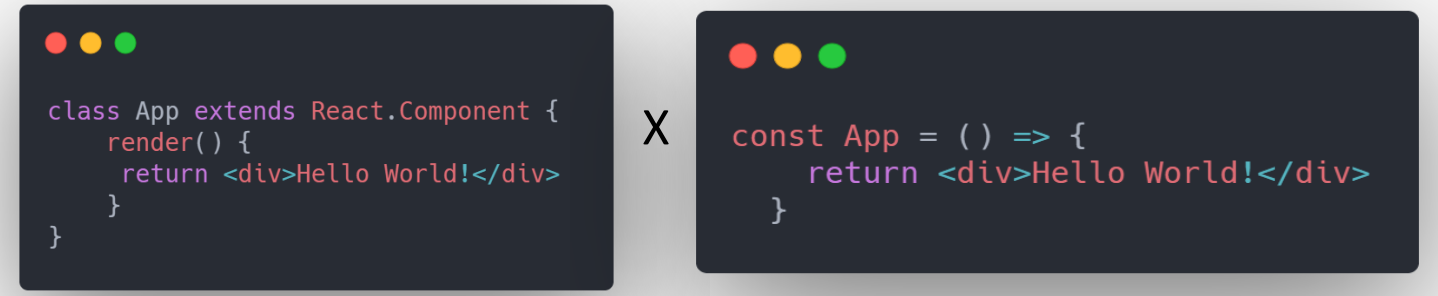




Principais Conceitos Práticos

Componentes

- Componentes de Classe x Componentes de Função



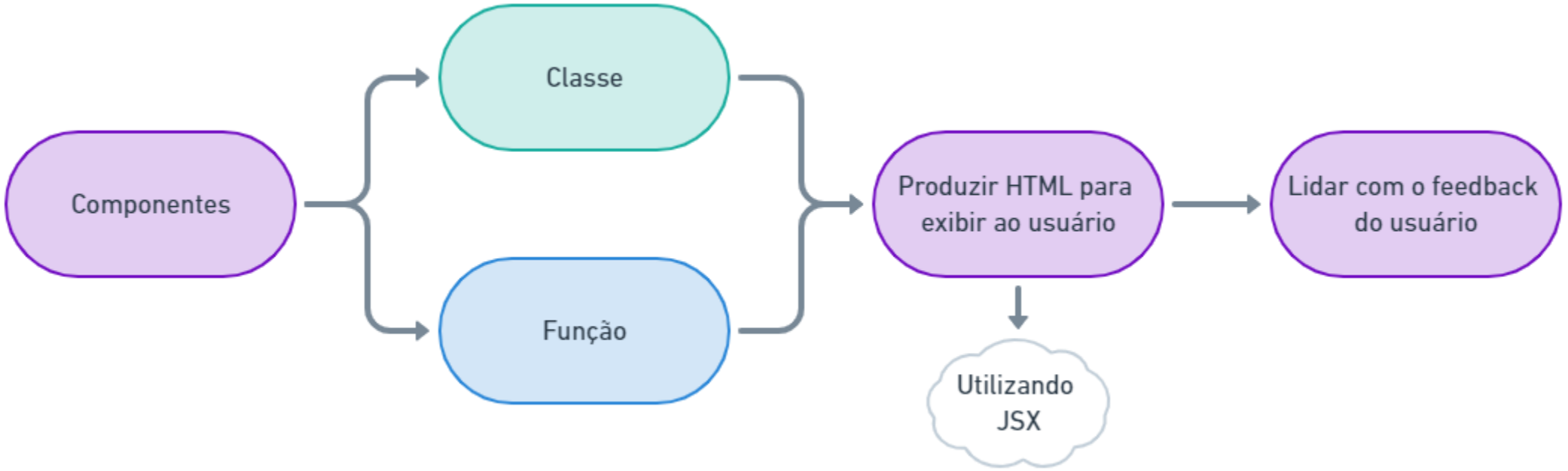
* Aprenderemos futuramente que os componentes funcionais não ficarão com esse conceito engessado. Mas até lá, deixaremos alinhado desta forma.

- Classe: Precisa do método render(), permite a utilização de states e permite o uso de métodos de ciclo de vida (lifecycle methods);

- Função: Maneira mais simples de definir um componente. Não mantém states e nem permite a utilização de lifecycle methods.*

Principais Conceitos Práticos

Componentes



Principais Conceitos Práticos

Componentes de Classe - States

- Gerenciados de dentro do componente (como variáveis declaradas dentro de uma função);
- Diferentemente de variáveis comuns, só podem ter seu valor alterado através do setState();
- **Quando o state muda, o componente responde renderizando novamente;**
- O setState() é **assíncrono**. O React intencionalmente "espera" até todos os componentes terem chamado setState() em seus manipuladores de evento antes de começar a renderizar novamente. Isso aumenta performance por evitar renderizações desnecessárias.

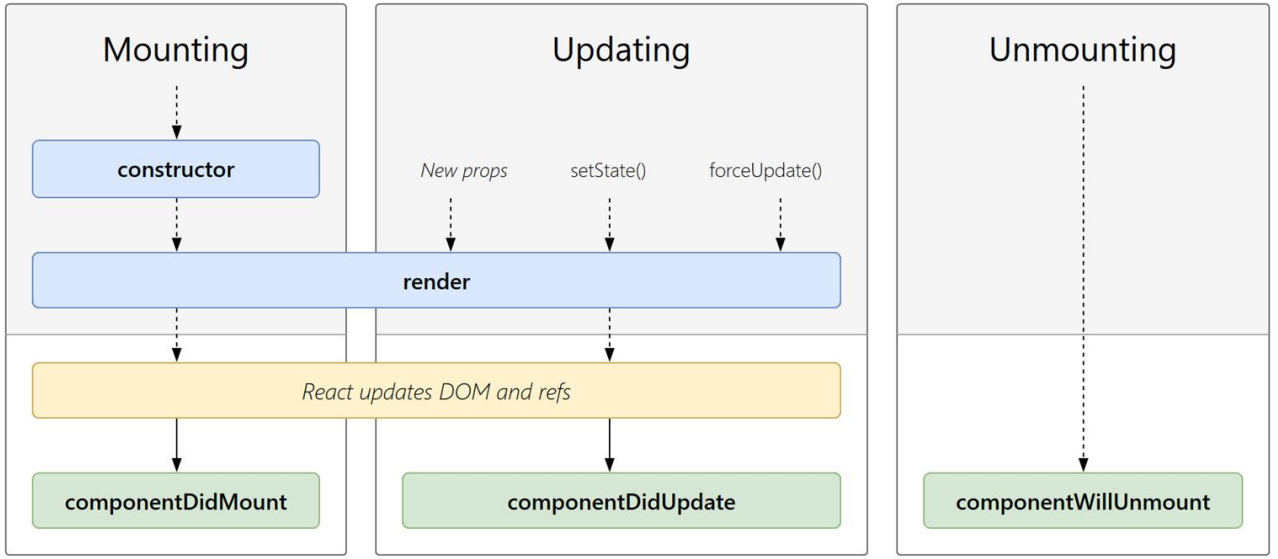
```jsx
import React from 'react';

class App extends React.Component {
    state = {
        nome: ''
    }

    handleClick = () => {
        if (this.state.nome === 'Yuri Weilemann') {
            this.setState({nome: ''})
        } else {
        this.setState({nome: 'Yuri Weilemann'})
        }
    }

    render() {
        return (
        <>
            <div>
                Hello {this.state.nome}!
            </div>
            <button onClick={this.handleClick}>Click me!</button>
        </>
        )
    }
}

export default App;
```

Fonte: https://pt-br.reactjs.org/

Principais Conceitos Práticos

Componentes de Classe - Métodos de Ciclo de Vida (Lifecycle Methods)

- Cada componente do React tem um ciclo de vida que você pode monitorar e manipular durante suas três fases principais.
- As três fases são: Montagem, Atualização e Desmontagem.
- **Montagem**: inserção de elementos no DOM.
- **Atualização**: Como o nome diz, acontece quando um componente é atualizado (mudança no state ou props do componente).
- **Desmontagem**: remoção de elementos no DOM.



Principais Conceitos Práticos

Componentes de Classe - Métodos de Ciclo de Vida (Lifecycle Methods)

- componentDidMount: É executado quando todos os componentes da tela são renderizados corretamente. **Geralmente** é utilizado para fazer requisições a APIs.

```
componentDidMOunt() {
  axios.get('https://pokeapi.co/api/v2/pokemon/')
  .then(response => {
    this.setState(response.data)
  }
}
```

- componentDidUpdate: É executado quando um componente é atualizado (quando um state é alterado, por exemplo). Tem, como seus dois primeiros argumentos, as props anteriores e o state anterior.

```
componentDidUpdate(prevProps, prevState) {
  if (prevState.pokemons !== this.state.pokemons) {
    console.log('O state dos pokemons foi alterado.')
  }
}
```

Principais Conceitos Práticos

Props

```jsx
import React from 'react';
import ReactDOM from 'react-dom';

const Welcome = (props) => {
  return <h1>Hello, {props.name}</h1>;
}

const element = <Welcome name="Yuri Weilemann" />;
ReactDOM.render(
  element,
  document.getElementById('root')
);
```

- Quando o React vê um elemento representando um componente definido pelo usuário, ele passa atributos JSX e componentes filhos para esse componente como um único objeto. Nós chamamos esse objeto de “props”.
- Utilizamos as props geralmente para passar informações armazenadas em states para componentes filhos, porém podemos passar qualquer tipo de objeto para estes componentes.

Fonte: https://pt-br.reactjs.org/

Principais Conceitos Práticos

Componentes de Função - Hooks

- Componentes funcionais eram utilizados somente quando não havia necessidade de se manipular um state ou de fazer uso dos lifecycle methods;
- Com a criação dos chamados Hooks, agora é possível fazer uso de ambas ferramentas citadas acima em componentes funcionais;
- A utilização dessas ferramentas é um pouco diferente através dos Hooks, então vamos dar uma olhada.

```
class Welcome extends React.Component {
  render() {
    return <h1>Hello, {this.props.name}</h1>;
  }
}
```

```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}
```

Componentes equivalentes no ponto de vista do React

Principais Conceitos Práticos

Componentes de Função - Hooks

- States: Os states deverão ser importados como **useState** de dentro do pacote 'react'. Para isso, utilizaremos as chaves para fazer uma importação não-default.

- Lifecycle Methods (componentDidMount e componentDidUpdate): Os Lifecycle Methods deverão ser importados como **useEffect** de dentro do pacote 'react'. Para isso, utilizaremos as chaves para fazer uma importação não-default.

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

Principais Conceitos Práticos

Componentes de Função - Hooks

Sintaxes

useState

```
const [state, setstate] = useState(initialState)
```

useEffect

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

OBS: Se o array de dependência do useEffect permanecer vazio, ele será executado apenas uma vez! (comportamento semelhante ao componentDidMount)