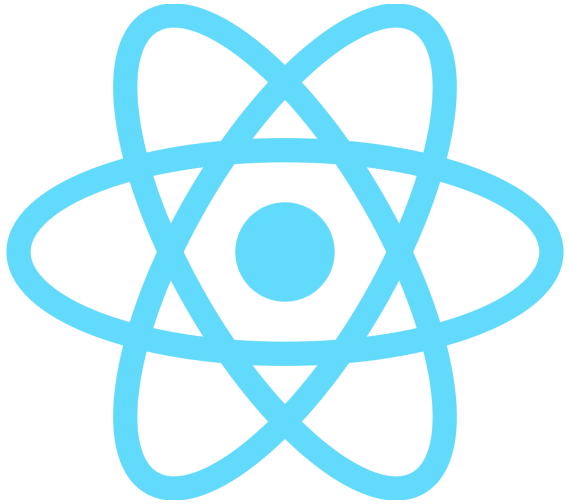


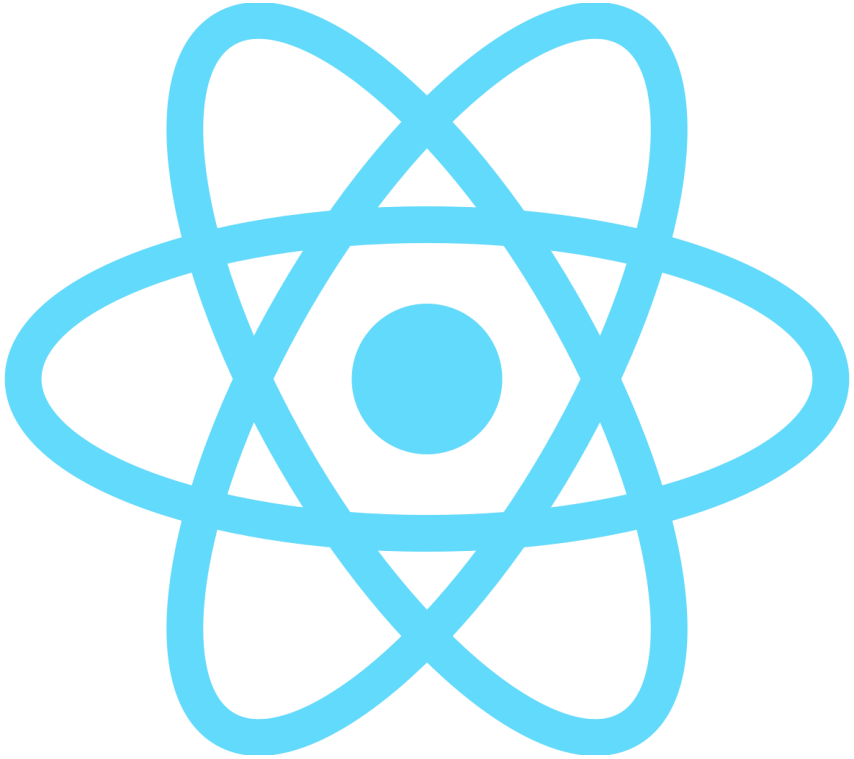
React JS
Aula 06

Instrutor: Rodolfo Moura

Intro

O que são classes?

Quando utilizá-las?



Classes no JavaScript

Consideramos como “funções especiais”

Maneira mais simples e clara de criar objetos e lidar com herança

Classes no JavaScript

Declarando classes:

```
class Retangulo {
    constructor(altura, largura) {
        this.altura = altura;
        this.largura = largura;
    }
}
```

Criando um objeto:

```
const Quadrado = new Retangulo(3,3);
```

Classes no JavaScript

Criando funções/métodos na classe:

```
TextoLargura () {
    return `Largura: ${this.largura}`
}
```

Utilizando tal função referenciando o novo objeto, podemos ter um retorno com a função usando os dados do objeto.

```
const Quadrado = new Retangulo(3,3);

console.log(Quadrado.TextoLargura());
```

```
C:\Users\Rodolfo\Documents\SENAC\rascunhos>node classRetangulo
Largura: 3
```

Componentes de Classe React

Diferente sintaxe dos Componentes apresentados anteriormente, os Componentes Funcionais

São criados usando os conceitos de Classe que temos no JavaScript

Atenção para a diferença entre Componentes de Classe e os Componentes Funcionais

Componentes de Classe x Funcionais

Componentes Funcionais (ou Principais) são funções que recebem um único argumento de objeto, chamado de props, e retorna dados com um elemento React.

Componentes de Classe estendem um Componente padrão do React, e ao fazer isso criamos uma herança do React.Component e daí podemos acessar suas funções, com o render retornando HTML.

Componentes de Classe React

Criação do componente de classe:

Importa React

```
import React from 'react';
```

Criamos o componente com a herança da Classe “Component” do React

```
class NomeDoComponente extends React.Component
```

Componentes de Classe React

Criação do componente de classe:

Adicionamos o render (ação necessária)

```
class NomeDoComponente extends React.Component {

    render()
```

E em seguida o return com o resto do passo-a-passo sendo similar

```
render() {

    return <h1> Código </h1>

}
```

Ao final, exportamos.

Componentes de Classe React

Ao final teríamos:

```
import React from 'react';

class NomeDoComponente extends React.Component {

    render() {

        return <h1> Código </h1>

    }
}

export default NomeDoComponente
```

A chamada do componente no App.js é similar.

Componentes de Classe React

Para o uso de Props, os Componentes de Classe vão fazer o uso do Método Construtor, que em classes Javascript estabelecem as propriedades dentro do objeto.

```
class NomeDoComponente extends React.Component {

constructor(props){

    super (props)
}
    render() {
```

Além disso, temos que fazer o uso do “super” para poder fazer uso das classes herdadas quando estendemos de react.component

Componentes de Classe React

Para o uso interno das props dentro desse componente temos que fazer o uso de **this** como é feito em JavaScript com o método constructor.

```
render() {
    return(
    <div>
    <h1> Código</h1>
    <p>Nome: {this.props.nome}</p>
    </div>
    )
}
```

Eventos no React

Os eventos são similares aos eventos do DOM

Por ex.: **onClick**.

Porém no React colocamos diretamente na tag, diferente de quando usamos querySelector.

Geralmente um método é atribuído ao evento. Método normalmente criado no componente.

Eventos no React

Exemplificando com onClick, criamos uma componente que execute uma função ao ser efetuado o clique:

```
function Evento (){

    function meuEvento(){
        console.log("Evento ocorreu");
    }
    return (
        <div>
            <p>Evento onClick:</p>
            <button onClick={meuEvento}>Evento</button>
        </div>
    )
}
```

Eventos no React

Importaremos o componente normalmente para nosso App.js e o chamamos do mesmo modo:

```
function App() {
  return(
    <p>
    <Evento/>
    </p>
  )
}
```

Verificar realização do evento.

Eventos no React

Do mesma maneira com props, podemos colocar o componente com diferentes propriedades sendo chamadas, e com isso teremos diferentes resultados para diferentes componentes.

```
function Evento ({numero}){

    function meuEvento(){
        console.log(`Evento ${numero} ocorreu`);
    }
    return (
        <div>
            <p>Evento onClick:</p>
            <button onClick={meuEvento}>Evento</button>
        </div>
    )
}

export default Evento
```

Eventos no React

Ao usarmos a nossa componente assim, teremos diferentes resultados para cara interação onClick.

```
function App() {
  return(
    <p>
    <Evento numero="1"/>
    <Evento numero="2"/>
    </p>
  )
}
```

Eventos no React

Existem outros exemplos de uso de Eventos no React, dentro eles também podemos citar mais diretamente **onChange** e **onSubmit**