

Executar os processos de codificação, manutenção e documentação de aplicativos computacionais para desktop

SENAC PE

Thiago Nogueira

Instrutor de Educação Profissional

React Components

React Components

Overview

- Componentes são pedaços de código independentes e reutilizáveis.
- Eles servem ao mesmo propósito que funções JavaScript, mas funcionam isoladamente e retornam HTML.
- Os componentes vêm em dois tipos: componentes de classe e componentes de função.
- Em bases de código React mais antigas, você encontra principalmente componentes Class. Agora é sugerido usar componentes Function junto com Hooks, que foram adicionados no React 16.8.

React Components

Convenções

- Nomes dos componentes definidos pelo desenvolvedor **DEVEM** começar com letra maiúscula.
- Com isso o React mapeia em um `React.createElement` correspondente.
- Tags começando com letras minúsculas são integradas como uma tag DOM.
- Isso faz com que o React use alguns dos componentes integrados correspondentes a essas tags HTML no React.

React Components

Class Component

- Um componente Class deve incluir a declaração `extends React.Component`. Esta declaração cria uma herança para `React.Component` e dá ao seu componente acesso às funções de `React.Component`.
- Exemplo: componente de classe chamado `Car`

```
class Car extends React.Component {  
  render() {  
    return <h2>Hi, I am a Car!</h2>;  
  }  
}
```

React Components

Function Component

- Um componente Function também retorna HTML e se comporta da mesma forma que um componente Class, mas os componentes Function podem ser escritos usando muito menos código e são mais fáceis de entender.
- Exemplo: mesmo exemplo **Car**, mas criado usando um componente Function

```
function Car() {  
  return <h2>Hi, I am a Car!</h2>;  
}
```

React Components

Renderizando um Component

- Para usar um component em uma aplicação, use uma sintaxe semelhante ao HTML normal: `<NomeComponent />`
- Exemplo: Exibir o componente **Car** no elemento "root":

```
const root =  
ReactDOM.createRoot(document.getElementById('root'));  
root.render(<Car />);
```

React Components

Props

- Os componentes podem ser passados como **props**, que significa propriedades.
- Props são como argumentos de função, e você os envia para o componente como atributos.
- Exemplo: Atributo para passar uma cor para o componente Car e usar na função render():

```
function Car(props) {  
  return <h2>I am a {props.color} Car!</h2>;  
}  
  
const root =  
ReactDOM.createRoot(document.getElementById('root'));  
root.render(<Car color="red"/>);
```


React Components

Componentes em Componentes

- Podemos nos referir a componentes dentro de outros componentes:
- Exemplo: Car Component dentro do Garagem Component

```
function Car() {  
  return <h2>I am a Car!</h2>;  
}  
  
function Garage() {  
  return (  
    <>  
      <h1>Who lives in my Garage?</h1>  
      <Car />  
    </>  
  );  
}  
  
const root =  
ReactDOM.createRoot(document.getElementById('root'));  
root.render(<Garage />);
```

React Components

Componentes em arquivos

- O React tem como objetivo reutilizar código, e é recomendável dividir seus componentes em arquivos separados.
- Para fazer isso, crie um novo arquivo.js e coloque o código dentro dele.
 - Lembre-se que o nome do arquivo deve começar com uma letra maiúscula.
- Exemplo: novo arquivo "Car.js":
 - Para poder usar o componente, você precisa importar o arquivo em sua aplicação.

```
function Car() {  
  return <h2>Hi, I am a Car!</h2>;  
}  
  
export default Car;
```

React Components

Componentes em arquivos

- Agora importamos o arquivo "Car.js" no aplicativo (por exemplo App.js) e podemos usar o componente Car como se ele tivesse sido criado aqui.

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import Car from './Car.js';

const root =
ReactDOM.createRoot(document.getElementById('root'));
root.render(<Car />);
```

Exercícios

Exercícios

Instruções


- Seguir o exercício roteiro da aula 09
- Documentar o que foi feito no README.md
- A documentação deve ter a explicação de cada bloco de código do novo componente criado bem como a explicação do que é o novo componente
- Enviar o exercício roteiro da aula 09 para um repositório no GitHub
- Anexar link do repo na atividade do Teams


*“Ensinar é impregnar
de sentido o que
fazemos a cada
instante”*


Paulo Freire

Obrigado!

Thiago Nogueira

 [linkedin.com/tdn](https://www.linkedin.com/tdn)

 thiago.nogueira@pe.senac.br

 (81) 9 9627-0419