



Front-end: React

**React - Aula 02 -
Eventos, Imagens e
Hooks**

Prof. MSc. Kelson Almeida

Agenda

- Eventos
- Imagens
- Hooks
 - useState

Eventos de Clique

- Os eventos para o front-end são fundamentais.
- No React, evento de clique é uma interação do usuário que ocorre quando o elemento da interface é clicado.
- O React fornece uma maneira simples de lidar com eventos de clique em seus componentes.

```
javascript

import React from 'react';

function MeuComponente(props) {
  function handleClick() {
    console.log('O botão foi clicado.');
```

Eventos de Clique

- Em muitas situações vamos precisar de eventos de clique
 - Exemplo: Enviar formulários
- Em React, os eventos já estão “prontos”.
 - Podemos utilizar “onClick”
 - Criada na própria função do componente.
- Para adicionar um evento de clique no React, podemos adicionar uma *listener* de eventos de clique ao elemento em questão.
- No geral, é melhor utilizar a sintaxe de JSX para lidar com eventos em vez de adicionar listeners de eventos diretamente ao DOM, já que o próprio React vai lidar com a criação e gerenciamento de DOM para nós.

```
javascript

import React from 'react';

function MeuComponente(props) {
  function handleClick() {
    console.log('O botão foi clicado.');
```

Eventos de Clique

- Neste exemplo, estamos adicionando um evento de clique ao botão usando o atributo “onClick”.
- Quando o botão é clicado, a função “handleClick” é executada e imprime uma mensagem no console.
- Observe que a função “**handleClick**” é definida dentro do componente “MeuComponente”, para que possa acessar as propriedades e o estado do componente, se necessário.

```
javascript

import React from 'react';

function MeuComponente(props) {
  function handleClick() {
    console.log('O botão foi clicado.');
```

Renderizando com Funções

- É possível criar funções que retornem JSX.
- Situações que dependam de outras condições
- Exemplo: O JSX a ser renderizado pode mudar de acordo com alguma variável.

```
10
11   const renderSomething = (x) => {
12     if (x) {
13       return <h1>Renderizando isso!</h1>
14     } else {
15       return <h1>Renderizando aquilo!</h1>
16     }
17   }
18
19   return (
20     <div>
21       <div>
22         <button onClick={handleMyEvent}>Clique aqui</button>
23       </div>
24       <div>
25         {renderSomething(true)}
26         {renderSomething(false)}
27       </div>
28     </div>
29   )
30 }
31
32 export default Events
```

Vamos criar nosso projeto?

- De uma maneira mais rápida que a aula passada:
 - `npm create vite@latest`

Vamos praticar?

- Crie um componente chamado: `RenderizandoComFuncoes`
- Neste componente crie uma função chamada: `escolhaDeRenderizacao` que terá um parâmetro chamado: `oQueRenderizar`
 - Nesta função, se o valor de `"oQueRenderizar"` for a string `"h1"`, terá como retorno um JSX com o texto `"Texto em h1"` envolto pela tag `h1`.
 - Se a string for diferente de `"h1"`, o retorno será: `"Texto em h2"` envolto pela tag `h2`.
- Na tela chame essa função através de duas template expressions, uma passando como parâmetro `"h1"` e outra `"h2"`.
- Segue o exemplo do resultado na tela ao lado:

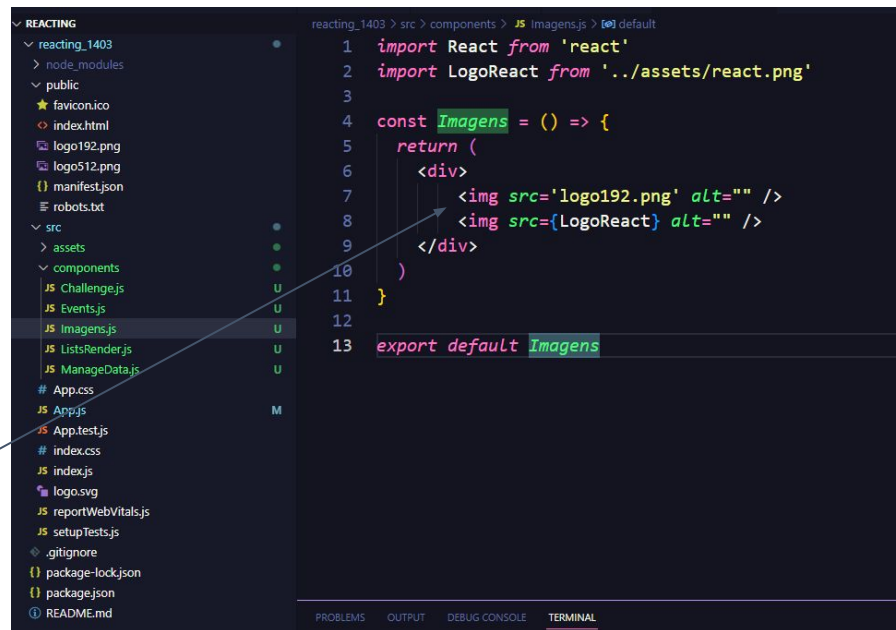


Texto em h1

Texto em h2

Trabalhando com Imagens no React

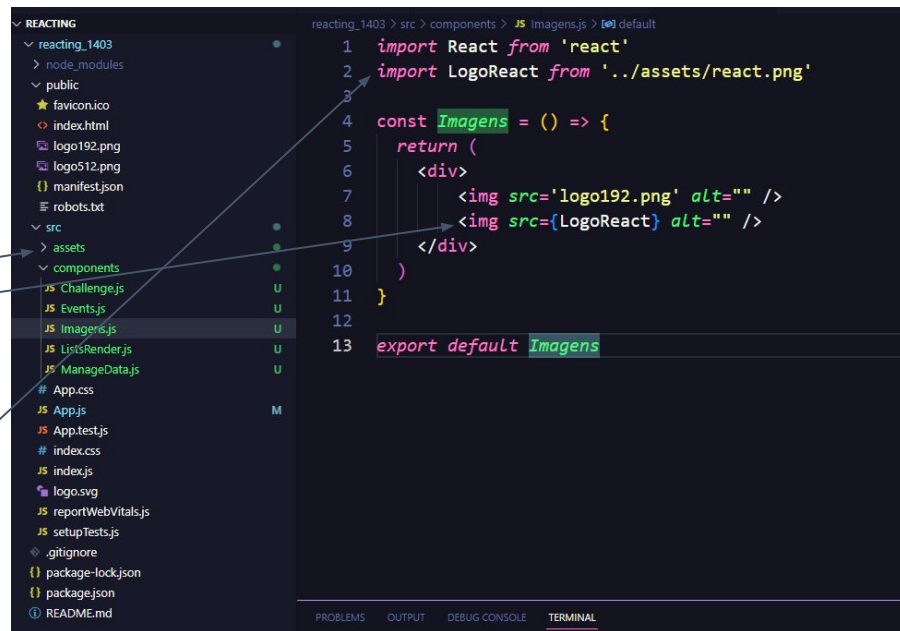
- A pasta public do nosso projeto pode abrigar as imagens públicas.
- Quando as imagens estão nessa pasta, podemos chamá-las nos nossos componentes através da tag `img` e `/nome-da-imagem.jpg`
- Isso se dá, pois a pasta public fica conectada ao src
- Por boas práticas, sempre devemos ao menos inicializar a propriedade `alt` do `img`.



```
1 import React from 'react'
2 import LogoReact from '../assets/react.png'
3
4 const Imagens = () => {
5   return (
6     <div>
7       <img src='logo192.png' alt='' />
8       <img src={LogoReact} alt='' />
9     </div>
10   )
11 }
12
13 export default Imagens
```

Trabalhando com Assets no React

- Outro padrão bem utilizado para inserir imagens em projetos, é utilizar uma pasta chamada “assets” dentro de src.
- Podemos encontrar projetos utilizando as duas abordagens.
- Caso optemos por assets, precisamos importar as imagens.



```
1 import React from 'react'
2 import LogoReact from '../assets/react.png'
3
4 const Imagens = () => {
5   return (
6     <div>
7       <img src='logo192.png' alt='' />
8       <img src={LogoReact} alt='' />
9     </div>
10   )
11 }
12
13 export default Imagens
```

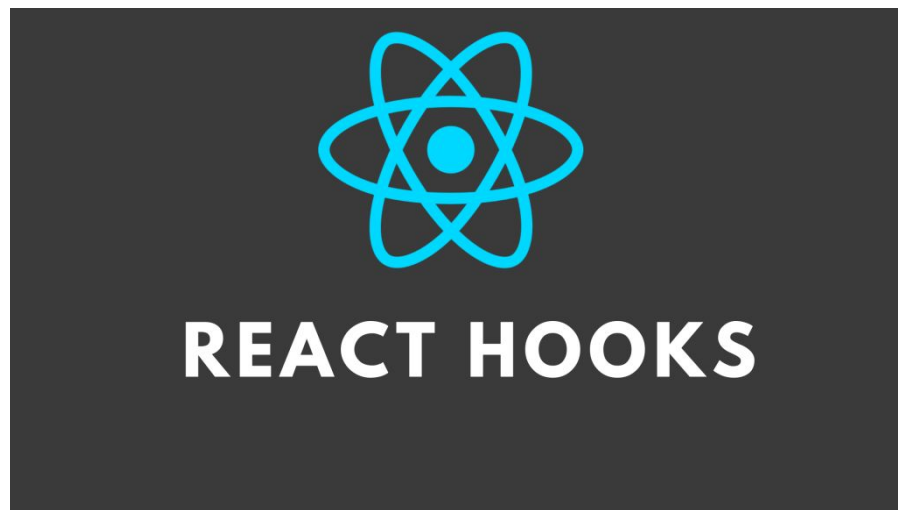
Vamos praticar?

- Crie um componente chamado `TrabalhandoComImagens`
- Baixe na internet duas imagens quaisquer.
- Exiba essas duas imagens na tela, uma através da pasta `public` e a outra através de `assets` importado.



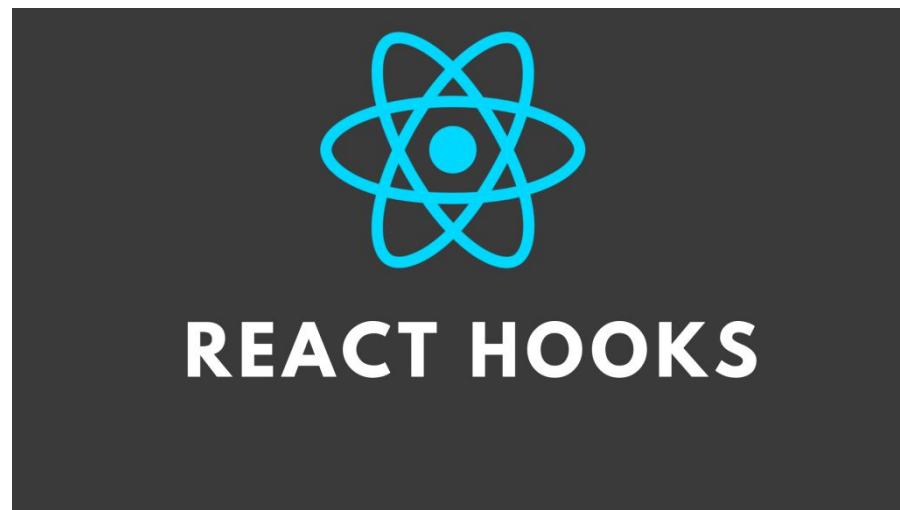
Hooks

- Recurso do React que tem muitas funções!
- Guardar/alterar o estado de algum dado.
- Em nomenclatura, os hooks iniciam com a palavra “use”.
 - Exemplo: useState.
- Precisam ser importados!
- São bastante utilizados nas aplicações.
- Utilizaremos bastante durante as nossas práticas.



Hooks

- Até a versão 16.7, algumas funcionalidades só eram possíveis serem utilizadas através de classes.
- Na versão 16.8 do React, foram lançados os hooks, que permitem uso de vários recursos através de **funções**.
Ajudando também a organizar a lógica dentro dos componentes.



Hooks

- Uma forma de adicionar funcionalidades de estado e ciclo de vida a componentes funcionais do React.
- Existem vários tipos de Hooks disponíveis, os mais comuns são o “useState” e o “useEffect”.
- No “useState” é permitido adicionar estado a um componente funcional.

javascript

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

function Example() {
  // Declara uma variável de estado chamada "count"
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>Você clicou {count} vezes</p>
      <button onClick={() => setCount(count + 1)}>
        Clique aqui
      </button>
    </div>
  );
}
```

Hooks

- No exemplo ao lado, o “useState” é utilizado para adicionar **um estado chamado “count”** ao componente “Example”. O **“count” começa com o valor de 0** e é atualizado sempre que o botão é clicado.
- Quando o estado é atualizado, o componente é renderizado novamente com o valor do estado.

javascript

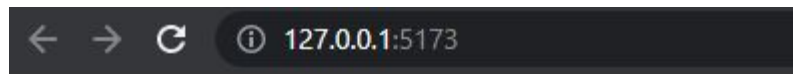
```
import React, { useState } from 'react';

function Example() {
  // Declara uma variável de estado chamada "count"
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>Você clicou {count} vezes</p>
      <button onClick={() => setCount(count + 1)}>
        Clique aqui
      </button>
    </div>
  );
}
```

Vamos praticar?

- Crie um componente chamado: HookContador
- Seu componente terá um `useState` “contador” com estado inicial de valor **1**.
- Crie uma função “incrementarContador” que altera o valor do contador para valor anterior+1
- A página de exibição deve seguir o exemplo ao lado:
 - Crie o button “Incrementar contador para exibir o valor de cada incremento.



Contador com `useState`

Incrementar contador

O contador está em: 16

Já apertou 15 vezes no botão

Vamos praticar?

- Crie um componente chamado “HookMegaSena”
- Nele, inicialize um useState que armazena um número sorteado, que tem o estado inicial vazio.
- Crie outro useState com o estado inicial de um array vazio para armazenar os números sorteados
- Seu componente terá uma função chamada: `sortearNumero`
 - Essa função deve abastecer o “useState” de número sorteado com um número aleatório entre 1 e 60.
 - Sintaxe: `Math.floor(Math.random() * 60) + 1`
 - Também deve ir armazenando os valores já sorteados em um array.
 - Sintaxe: `[...arrayNumerosSorteados, sorteado]`
 - Não permitir que um novo número seja sorteado se o array já tiver o tamanho de 6 elementos.
 - Se isso acontecer, exibir um alert: “Já temos 6 números sorteados!”
- Exibir na tela as informações como demonstra o print ao lado.
 - Criar o button Sortear Número para chamar a sua função.



Sorteador da Mega em React! :)

Sortear Número

Último Número sorteado: 35

Sorteados: 20 - 17 - 47 - 59 - 18 - 35

