

React JS

★ AULA 03 ★

★ PROFESSOR
MAROMO ★

React JS

State Hook

Sobre manutenção de estados



Agenda

- Componentes e estados
- State Hook
- Evento onClick
- Geração de números aleatórios



State Hook

O State Hook, também conhecido como `useState`, é uma das funcionalidades do React que permite que componentes funcionais - uma forma mais simples e moderna de criar componentes no React - tenham seu próprio estado interno, algo que anteriormente era limitado a componentes de classe.



useState

O useState é uma função que aceita um argumento, que é o estado inicial, e retorna um array com duas posições. A primeira posição é o valor atual do estado, e a segunda é uma função que pode ser usada para atualizá-lo.



Aqui está um exemplo básico de como useState é usado em um componente funcional do React:

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

function Example() {
  // Declare uma nova variável de estado, que chamaremos de "count"
  const [count, setCount] = useState(0);

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



Evento onClick

Ao clicar em um botão é comum que algo aconteça. Para chamarmos esta ação utilizamos o evento onClick.

Neste exemplo, quando o usuário clicar no botão, o código associado ao evento onClick será executado.



por que o
evento **onClick**
é útil ?



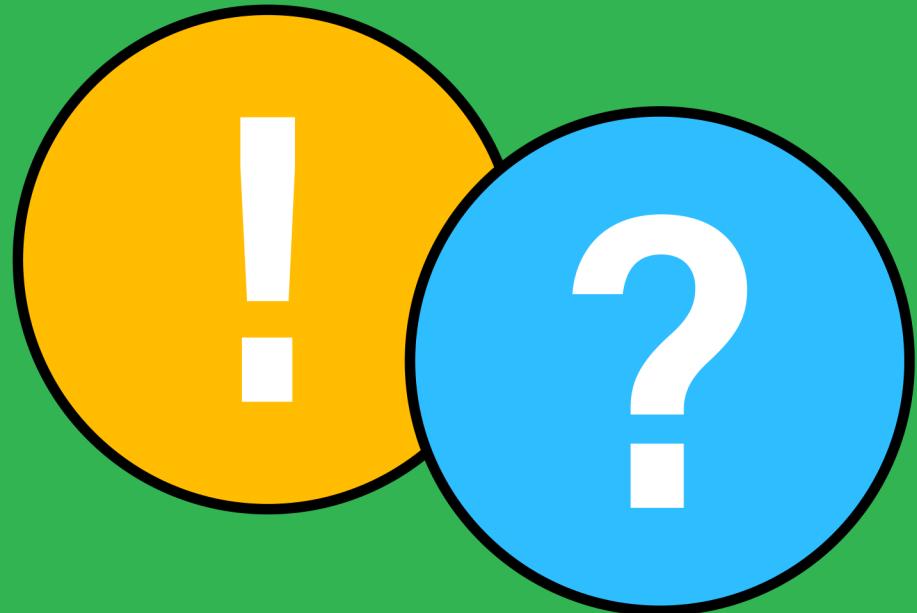


Utilidade do onClick

Em resumo, o evento **onClick** é uma ferramenta crucial na criação de páginas web e aplicações que respondem às ações dos usuários, melhorando a experiência do usuário e permitindo interações mais complexas

Essa característica permite que os desenvolvedores criem interfaces de usuário interativas e dinâmicas. Por exemplo, um botão pode ter um evento **onClick** que aciona a submissão de um formulário, a abertura de uma nova página, a alteração de um estado dentro de um componente do React, entre outras ações.

E o State
Hook, por quê
é útil ?



Importância do State Hook



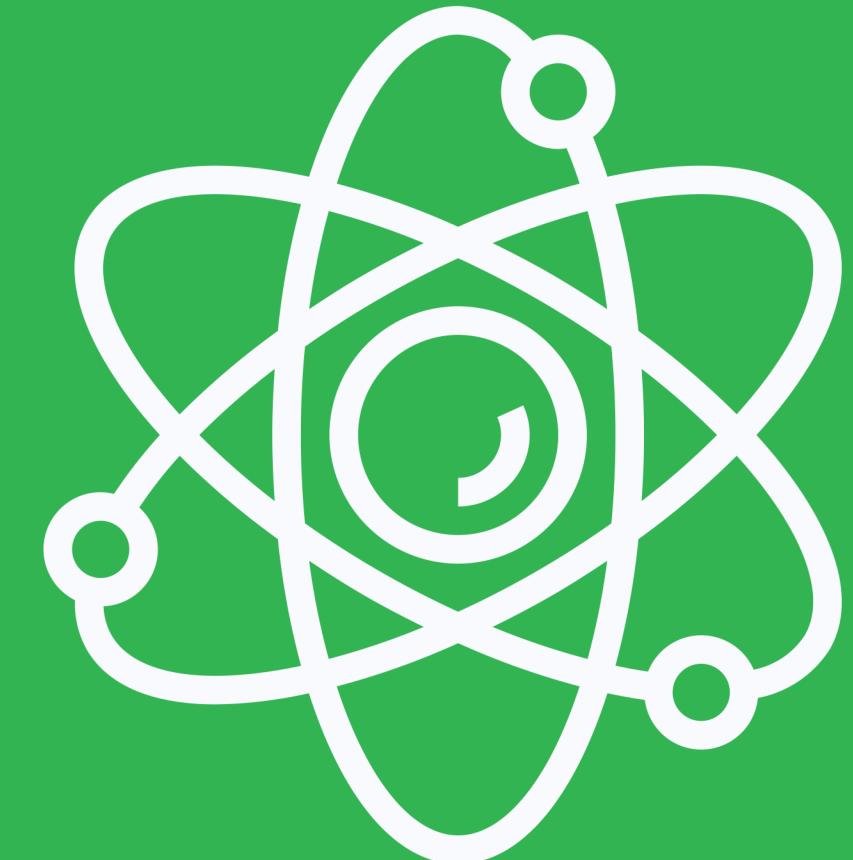
1. Gerenciamento de estado em componentes funcionais
2. Reatividade
3. Simplicidade e legibilidade
4. Modularidade

Gerenciamento de estado em componentes funcionais



Antes da introdução dos Hooks, o gerenciamento de estado era restrito a componentes de classe. Com o **useState**, é possível ter estado local em componentes funcionais, o que torna o código mais legível e fácil de manter.

Reatividade



O **useState** permite a criação de componentes reativos. Quando o estado de um componente é alterado utilizando a função de atualização fornecida pelo **useState**, o React sabe que precisa renderizar esse componente para refletir as alterações.

Simplicidade e legibilidade

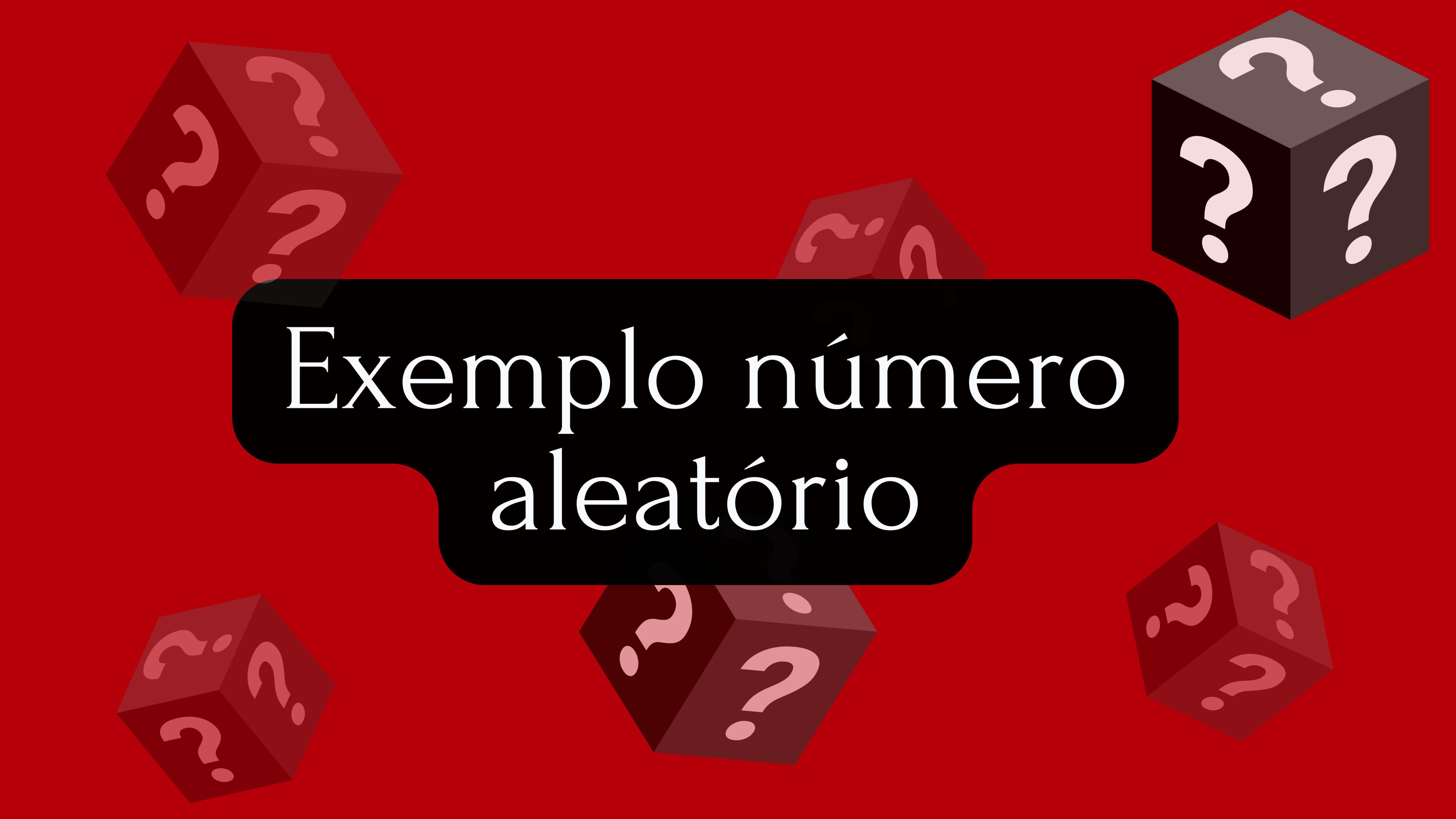


O **useState** torna o código mais limpo, pois elimina a necessidade de métodos de ciclo de vida complexos que eram necessários em componentes de classe para gerenciamento de estado.

Modularidade



Com o **useState**, é possível definir múltiplos estados independentes dentro de um único componente. Cada chamada ao useState cria um novo estado local que pode ser gerenciado de forma independente, o que permite uma maior modularidade e reutilização de código



Exemplo número
aleatório

Arquitetura do Exemplo



Vamos criar um componente funcional simples em React que gera um número aleatório entre 1 e 100 quando um botão é clicado.

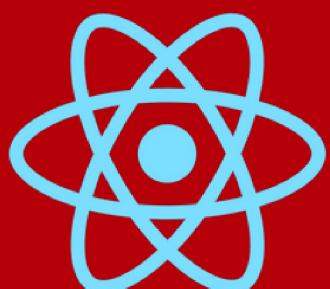
Esses são os passos importantes

1. Importar React e useState
2. Criar o componente funcional
3. Inicializar o estado
4. Criar a função de geração de números aleatórios
5. Renderizar o componente

Laboratório



Desenvolvimento de uma aplicação SPA usando o React JS. Nome da aplicação **sample03**.



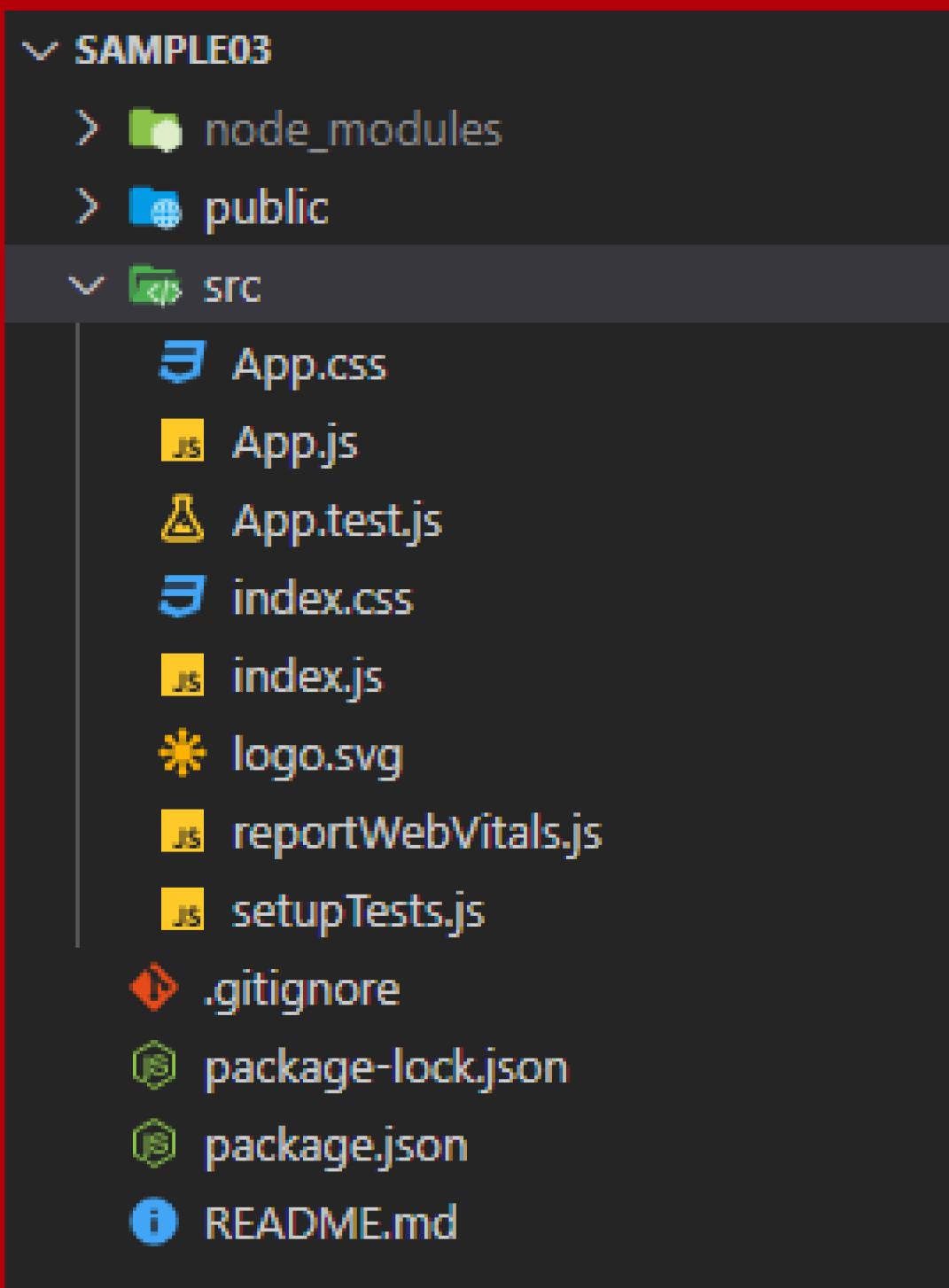
React JS



Passo 1: Configuração do ambiente de desenvolvimento

1. Execute o comando `npx create-react-app sample03` para criar um novo projeto React chamado "sample03". Aguarde até que a criação do projeto seja concluída.
2. O processo anterior, de criação do projeto, pode levar alguns instantes enquanto create-react-app monta todo o **boilerplate** mínimo que uma aplicação React para a web precisa ter.
3. Entre no diretório [cd sample03] e digite [code .] para abrir o VS Code.

Passo 2: Estrutura do projeto React





Antes de começarmos criar o componente Home com os arquivos index.js e style.css de nossa aplicação. Veja a estrutura abaixo:

A screenshot of a file explorer window titled "SAMPLE03". The structure is as follows:

- > node_modules
- > public
- < src
 - < components
 - < Home
 - index.js
 - style.css

A large white arrow points from the left towards the "src" folder, and a curly brace is placed below the "components" folder to indicate its scope.



Passo 3: Preparando o componente pai

Esse componente é chamado **App** e já foi criado junto com o projeto na aula anterior.

O que precisamos fazer é remover o conteúdo que foi automaticamente criado dentro dele e inserir o nosso código.

Nossa aplicação funcionará da seguinte forma:

- O usuário clica no botão **Gerar número**;
- Um número aleatório substitui o número 0.

```
src > JS App.js > [e] default
1 | import React from "react";
2 | import './App.css';
3 |
4 | function App(){
5 |   return (
6 |     <div className="App">
7 |       </div>
8 |     );
9 |   }
10| }
11| export default App;
```



Passo 4: Preparando Componente Home [index.js]

Com o nosso componente App criado, agora podemos começar a construir o único componente filho que vai compor nossa página. Componente Home, como mostra a Figura.

```
1 import React from "react";
2 import './style.css';
3
4 function Home() {
5   const [numeroAleatorio, setNumeroAleatorio] = useState(0);
6
7   function gerarNumero() {
8     const novoNumero = Math.floor(Math.random() * (100 - 1) + 1);
9     setNumeroAleatorio(novoNumero);
10 }
11
12 return (
13   <div className="conteudo-centralizado">
14     <h3>Número aleatório:</h3>
15     <h1>{numeroAleatorio}</h1>
16     <div className='area-botao'>
17       <label>
18         Click no botão abaixo para gerar um número aleatório:
19       </label>
20       <button onClick={gerarNumero}>
21         Gerar número
22       </button>
23     </div>
24   </div>
25 );
26 }
27
28 export default Home;
```

A black and white photograph of a person from the side, wearing a dark, zip-up jacket with a high collar. They are standing in a minimalist, modern interior with large windows in the background.

Estilizando os
componentes

STYLE. STYLE.
STYLE. STYLE.
STYLE. STYLE.



Diretório src - Estilizando a página principal

Arquivo App.css:

```
1 < .App {  
2   display: flex;  
3   justify-content: center;  
4   align-items: center;  
5   flex-direction: column;  
6   width: 100%;  
7   height: 100vh;  
8   background-color: #f8fafb;  
9 }  
10
```



Diretório src - Estilizando a página Home

Arquivo Home/style.css

```
1  .conteudo-centralizado {  
2      display: flex;  
3      justify-content: center;  
4      align-items: center;  
5      flex-direction: column;  
6      background-color: #fff;  
7      padding: 20px;  
8      box-shadow: 3px 6px 10px 6px rgba(0, 0, 0, 0.05);  
9      border-radius: 12px;  
10     box-sizing: border-box;  
11  }  
12  
13 .area-botao {  
14     display: flex;  
15     justify-content: center;  
16     align-items: center;  
17     flex-direction: column;  
18     text-align: center;  
19  }
```

```
21  button {  
22      margin-top: 20px;  
23      color: #FFFFFF;  
24      font-size: 12px;  
25      font-weight: 500;  
26      text-decoration: none;  
27      border: none;  
28      border-radius: 5px;  
29      margin-top: 15px;  
30      background: #599465;  
31      box-shadow: 0px 3px 6px rgba(119, 119, 119, .1);  
32      padding: 10px 15px;  
33      outline: none;  
34      cursor: pointer;  
35  }  
36
```



Diretório src - Atualizando a página principal

Arquivo App.js:

```
1 import logo from './logo.svg';
2 import './App.css';
3 import Home from './components/Home';
4
5 function App() {
6   return (
7     <div className='App' >
8       <Home/>
9     </div>
10   );
11 }
12
13 export default App;
```

Resultado da Execução



Número aleatório:

70

Click no botão abaixo para gerar um número aleatório:

[Gerar número](#)



Resultado

Este componente irá inicialmente exibir a mensagem "Clique no botão abaixo para gerar um número aleatório:". Quando o botão for clicado, um número aleatório entre 1 e 100 será gerado e exibido na tela, substituindo a mensagem inicial.



O sucesso é a soma de
pequenos esforços
repetidos dia após dia.