

# React Native



— Domine o Futuro Mobile —

Kelven Silva

# Índice

## **1. Introdução ao React**

- O que é React?
- Instalação e configuração do ambiente
- Criando seu primeiro componente

## **2. Componentes e Props**

- Estrutura de componentes
- Utilização de Props para passagem de dados
- Exemplos práticos e exercícios

## **3. Estado e Eventos**

- Introdução ao estado (useState)
- Manipulação de eventos
- Atualização dinâmica de componentes

# Introdução

E aí, meu amigo ou minha amiga dev! Bem-vindo ao seu guia para o universo do React! Se você está começando sua jornada no desenvolvimento web ou quer dar um upgrade nas suas habilidades, este ebook é perfeito para você.

Vamos explorar o básico do React de forma simples e prática, dividido em três módulos essenciais:

**Introdução ao React** – Aqui, vamos entender o que é o React, como instalar e configurar seu ambiente, e criar seu primeiro componente. É o ponto de partida para se familiarizar com a biblioteca.

**Componentes e Props** – Neste módulo, vamos nos aprofundar na construção de componentes e na utilização das props para passar dados entre eles. Você vai aprender a estruturar seu código e criar componentes reutilizáveis.

**Estado e Eventos** – Finalmente, vamos explorar como gerenciar o estado com o `useState` e como lidar com eventos para tornar suas aplicações interativas. Esses conceitos são fundamentais para criar experiências de usuário dinâmicas.

Cada módulo vem com exemplos práticos e exercícios para consolidar seu aprendizado. Prepare-se para se tornar um dev React habilidoso e criar interfaces web impressionantes!

**Vamos nessa?**

# Introdução ao React

E aí, meu amigo ou minha amiga dev! Vamos começar nossa jornada com o React? Aqui, vamos ver o básico para você dar os primeiros passos e começar a criar interfaces web incríveis.

## *O que é React?*

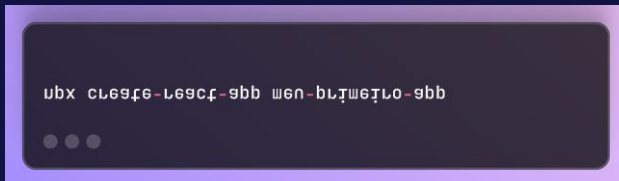
React é uma biblioteca JavaScript desenvolvida pelo Facebook que facilita a criação de interfaces de usuário dinâmicas e interativas. Em vez de escrever grandes blocos de código para cada parte da sua aplicação, o React divide tudo em pequenos componentes reutilizáveis. Isso torna o desenvolvimento mais organizado e o código mais fácil de manter.

## **Instalação e configuração do ambiente**

Para começar, você vai precisar configurar seu ambiente de desenvolvimento. Aqui estão os passos básicos:

Instale o Node.js: O React precisa do Node.js, então, se você ainda não tem, baixe e instale a versão mais recente do site oficial do Node.js.

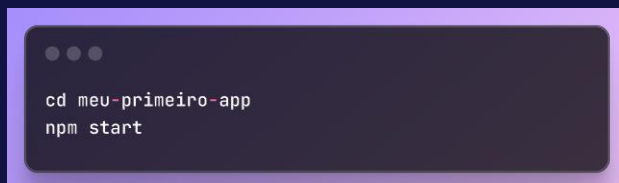
Crie um novo projeto com Create React App: O Create React App é uma ferramenta que configura tudo para você. Abra seu terminal e execute o comando:

A terminal window with a dark background and light blue text. It shows the command `npm create-react-app meu-primeiro-app` being entered. There are three small circles on the left side of the terminal window, indicating it's a window in a window manager.

```
npm create-react-app meu-primeiro-app
```

Isso vai criar uma nova pasta chamada `meu-primeiro-app` com todos os arquivos e configurações necessárias.

Inicie o servidor de desenvolvimento: Navegue até a pasta do seu projeto e inicie o servidor com:

A terminal window with a dark background and light blue text. It shows the commands `cd meu-primeiro-app` and `npm start` being entered. There are three small circles on the left side of the terminal window, indicating it's a window in a window manager.

```
cd meu-primeiro-app  
npm start
```

Isso vai abrir seu navegador com a aplicação React em funcionamento.

# Introdução ao React

## ***Criando seu primeiro componente***

Vamos criar um componente simples para ver o React em ação:

Abra o projeto: No seu editor de código, vá até a pasta src e abra o arquivo App.js.

Edite o arquivo App.js: Substitua o conteúdo do arquivo pelo seguinte código:



```
import React from 'react';

function App() {
  return (
    <div>
      <h1>Olá, React!</h1>
      <p>Bem-vindo ao seu primeiro componente React.</p>
    </div>
  );
}

export default App;
```

Veja o resultado: Salve o arquivo e veja o resultado no navegador. Você deve ver a mensagem "Olá, React!" e uma breve introdução.

Pronto! Você acabou de criar seu primeiro componente React. Continue praticando e explore mais sobre como criar e usar componentes para deixar suas aplicações ainda mais interessantes.

É isso aí, meu amigo ou minha amiga dev! Na próxima seção, vamos mergulhar nos componentes e nas props. Fique ligado!

## Componentes e Props

E aí, meu amigo ou minha amiga dev! Agora que você já tem o básico do React na ponta dos dedos, é hora de explorar um dos conceitos mais importantes: Componentes e Props. Esses são os blocos de construção que vão te ajudar a criar aplicações mais organizadas e funcionais.

### *Estrutura de componentes*

No React, tudo gira em torno de componentes. Pense neles como peças de Lego que você pode montar para criar sua aplicação. Cada componente pode ser uma parte da interface, como um botão, um formulário, ou até mesmo uma página inteira. Eles ajudam a manter seu código modular e fácil de entender.

### *Aqui está a estrutura básica de um componente:*

Importação do React: Todo componente React precisa importar a biblioteca React.

Função ou Classe: Você pode criar um componente usando uma função ou uma classe. Para iniciantes, a forma mais simples é com funções.

JSX: Dentro da função, você retorna um código chamado JSX (JavaScript XML) que define o que será exibido na tela.

Exportação: Por fim, você exporta o componente para poder usá-lo em outras partes do seu código.

Veja um exemplo de um componente simples:

```
import React from "react";

function MeuComponente() {
  return (
    <div>
      <h2>Bem-vindo ao Meu Componente!</h2>
    </div>
  );
}

export default MeuComponente;
```

## Utilização de Props para passagem de dados

Props são como parâmetros que você passa para os seus componentes. Elas permitem que você envie dados de um componente pai para um componente filho, tornando seus componentes mais reutilizáveis e dinâmicos.

### *Aqui está como você pode usar props:*

Defina Props no Componente Filho: No componente onde você deseja receber dados, você pode acessar as props diretamente nos parâmetros da função.

```
function Saudacao(props) {  
  return <h1>Olá, {props.nome}!</h1>;  
}
```

Passe Props do Componente Pai: Quando você usa o componente filho, você passa as props como atributos.

```
function App() {  
  return (  
    <div>  
      <Saudacao nome="Maria" />  
      <Saudacao nome="João" />  
    </div>  
  );  
}
```

No exemplo acima, o componente Saudacao recebe um prop chamado nome e o usa para personalizar a mensagem.

## Exemplos práticos e exercícios

Vamos colocar a mão na massa com alguns exemplos e exercícios para você praticar:

Crie um Componente de Boas-Vindas:

Crie um componente chamado BoasVindas que exibe uma mensagem de boas-vindas personalizada usando props.

Teste o componente passando diferentes nomes como props.

Construa um Componente de Lista:

Crie um componente chamado ListaDeItens que recebe uma lista de itens através de props e exibe cada item em uma lista HTML.

Desenvolva um Componente de Perfil:

Crie um componente Perfil que exibe o nome e a idade de uma pessoa. Passe essas informações como props e mostre uma mensagem personalizada, como "Meu nome é [nome] e eu tenho [idade] anos."

Com esses exercícios, você vai ganhar mais confiança no uso de componentes e props. Continue praticando e explorando para dominar esses conceitos!

1

```
// BoasVindas.jsx
import React from "react";
const BoasVindas = ({ nome }) => {
  return (
    <div>
      <h1>Bem-vindo, {nome}!</h1>
    </div>
  );
};
export default BoasVindas;
```

3

```
// Perfil.jsx
import React from "react";
const Perfil = ({ nome, idade }) => {
  return (
    <div>
      <p>
        Meu nome é {nome} e eu tenho {idade} anos.
      </p>
    </div>
  );
};
export default Perfil;
```

2

```
// ListaDeItens.jsx
import React from "react";
const ListaDeItens = ({ itens }) => {
  return (
    <ul>
      {itens.map((item, index) => (
        <li key={index}>{item}</li>
      ))}
    </ul>
  );
};
export default ListaDeItens;
```

Resultado :

```
// App.jsx
import React from "react";
import BoasVindas from "../BoasVindas";
import ListaDeItens from "../ListaDeItens";
import Perfil from "../Perfil";

const App = () => {
  const itens = ["Maçã", "Banana", "Laranja"];

  return (
    <div>
      <BoasVindas nome="João" />
      <ListaDeItens itens={itens} />
      <Perfil nome="Ana" idade={25} />
    </div>
  );
};
export default App;
```

E aí, pronto para colocar a mão na massa? Na próxima seção, vamos explorar o gerenciamento de estado e a manipulação de eventos no React. Fique ligado!



## Estado e Eventos


E aí, meu amigo ou minha amiga dev! Chegou a hora de explorar como tornar suas aplicações React interativas e dinâmicas. Vamos falar sobre o estado e como lidar com eventos. Com esses conceitos, suas aplicações vão ganhar vida!

### *Introdução ao estado (useState)*

O estado no React é como uma caixa de armazenamento que mantém informações sobre o seu componente e pode mudar ao longo do tempo. Para gerenciar o estado em componentes funcionais, usamos o hook useState.

*Aqui está o básico sobre como usar useState:*

Importe o Hook: Primeiro, você precisa importar useState do React.



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

Declare o Estado: Dentro do seu componente, declare o estado com useState. Ele retorna um array com dois elementos: o valor atual do estado e uma função para atualizá-lo.



```
function Contador() {  
  const [contador, setContador] = useState(0);  
  return (  
    <div>  
      <p>Você clicou {contador} vezes</p>  
      <button onClick={() => setContador(contador + 1)}>Clique aqui</button>  
    </div>  
  );  
}
```

No exemplo acima, contador é o valor do estado e setContador é a função para atualizá-lo. Cada vez que você clica no botão, o valor do contador aumenta.

## Manipulação de eventos

Eventos são interações do usuário com sua aplicação, como cliques de botão, entradas de texto e muito mais. No React, você pode manipular eventos diretamente em JSX.

Aqui estão alguns exemplos básicos:

### *Eventos de Clique:*

```
function App() {  
  const handleClick = () => {  
    alert("Botão clicado!");  
  };  
  return <button onClick={handleClick}>Clique em mim</button>;  
}
```

Quando o botão é clicado, a função handleClick é chamada, exibindo um alerta.

### *Eventos de Entrada:*

```
function Formulario() {  
  const [texto, setTexto] = useState("");  
  const handleChange = (event) => {  
    setTexto(event.target.value);  
  };  
  return <input type="text" value={texto} onChange={handleChange} />;  
}
```

Neste exemplo, o valor do input é controlado pelo estado, e a função handleChange atualiza o estado com o texto digitado.

## Atualização dinâmica de componentes

Quando o estado muda, o React atualiza automaticamente a interface para refletir as novas informações. Isso permite que suas aplicações reajam a eventos e mudanças de maneira fluida.

Por exemplo, se você atualizar o estado com um novo valor, o componente vai re-renderizar para mostrar as mudanças:

```
function App() {  
  const [mensagem, setMensagem] = useState("Olá, mundo!");  
  const mudarMensagem = () => {  
    setMensagem("Mensagem atualizada!");  
  };  
  return (  
    <div>  
      <p>{mensagem}</p>  
      <button onClick={mudarMensagem}>Mudar Mensagem</button>  
    </div>  
  );  
}
```

Quando você clica no botão, a mensagem é atualizada e o componente re-renderiza para mostrar a nova mensagem.

Pronto para ver sua aplicação ganhar vida com interatividade? Na próxima, você estará pronto para aplicar esses conceitos e criar interfaces ainda mais incríveis. Continue praticando e explorando!

## Conclusão

E aí, meu amigo ou minha amiga dev! Chegamos ao final deste mini ebook sobre React, e eu quero agradecer por ter acompanhado o conteúdo até aqui. Foi um prazer compartilhar com você os fundamentos essenciais do React, desde a criação do seu primeiro componente até o gerenciamento de estado e eventos.

Lembre-se: a prática é a chave para se tornar um desenvolvedor habilidoso. Continue explorando, criando projetos e aplicando o que você aprendeu. Cada linha de código é uma oportunidade para melhorar suas habilidades e entender ainda mais como o React pode transformar suas aplicações.

Se você tiver dúvidas ou quiser compartilhar suas criações, fique à vontade para entrar em contato. Aqui estão meus detalhes para que possamos continuar conectados:

Kelven Silva

<https://github.com/KelvenPer>

Obrigado por sua dedicação e boa sorte em sua jornada com React. Até a próxima!