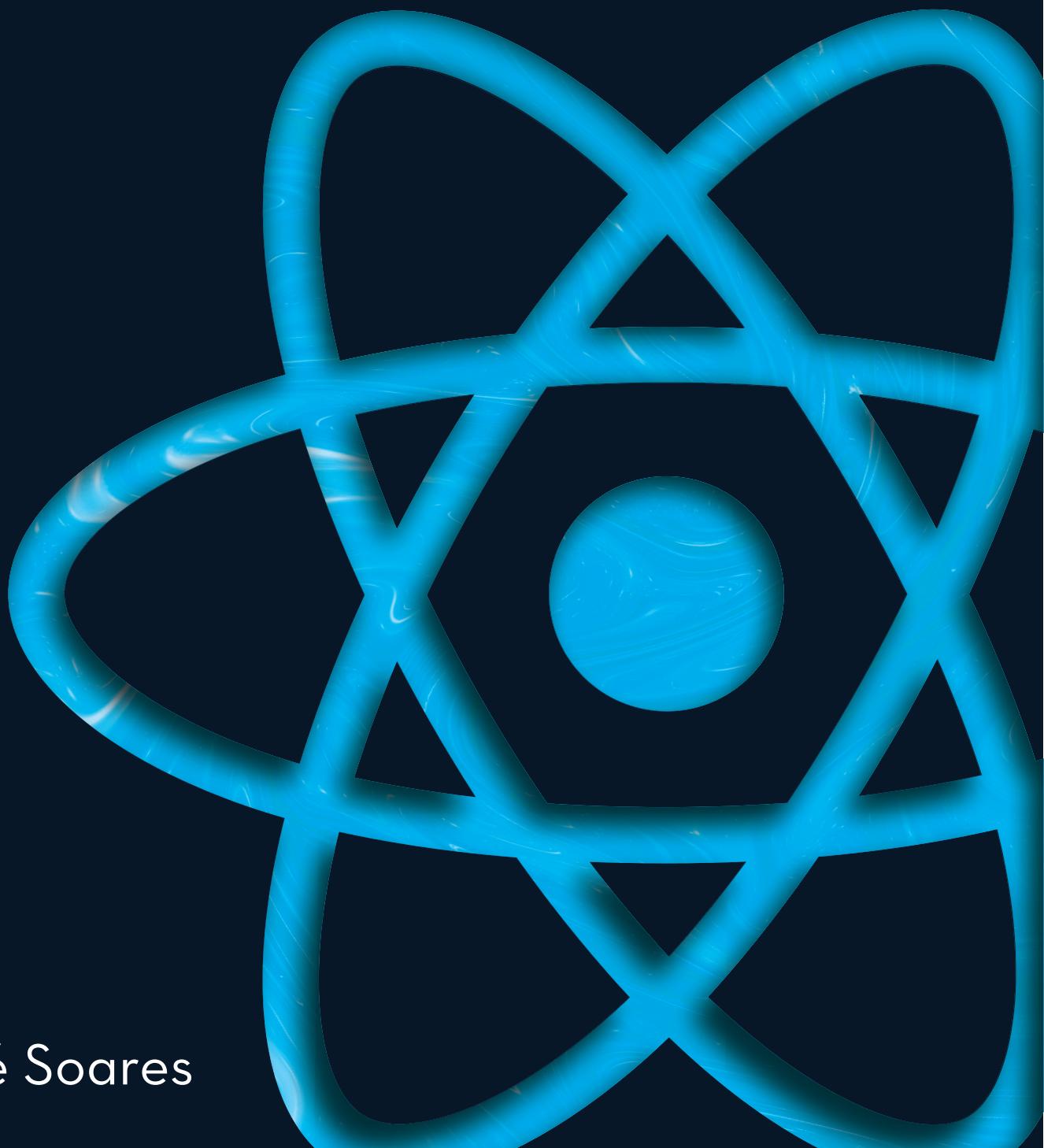


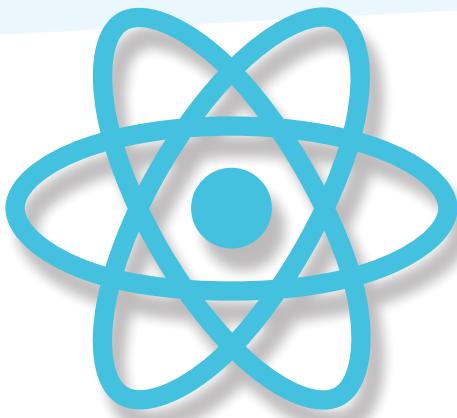
React

A biblioteca que move a web moderna



André Soares

O poder do React na Web Moderna



Se você é do tipo que vibra com um código limpo, componentes reutilizáveis e uma interface que responde instantaneamente às ações do usuário, bem-vindo ao universo do React.

Criado pelo Facebook (hoje Meta), o React é a espinha dorsal da maioria das interfaces modernas: Netflix, Instagram, Airbnb, GitHub e até a própria Meta utilizam React para entregar experiências rápidas e fluidas.

Neste eBook, vamos explorar suas principais funcionalidades, com explicações claras, diretas e exemplos de código reais, no dia a dia de um desenvolvedor.



Componentes: O DNA do React

Em React, **tudo é componente**.

Pense em um componente como um “lego digital”, você cria peças pequenas que se encaixam para formar interfaces poderosas.

```
... header

export function Header() {
  return (
    <header>
      <h1>🚀 Galactic Store</h1>
      <p>Os melhores gadgets do universo geek</p>
    </header>
  );
}
```

Esses componentes são funções que retornam JSX, uma sintaxe que mistura JavaScript com HTML. Isso torna o código direto, intuitivo e fácil de manter.

Dica de sênior: quanto menor e mais específico o componente, mais fácil será testá-lo e reutilizá-lo.



Props: Passando dados como quem faz trocas entre portais

As props são a forma como os componentes se comunicam. Imagine que você está enviando uma “mensagem” de um componente pai para um filho.

```
● ● ●          ProductCard

type ProductCardProps = {
  name: string;
  price: number;
};

export function ProductCard({ name, price }: ProductCardProps) {
  return (
    <div>
      <h2>{name}</h2>
      <p>$ {price.toFixed(2)}</p>
    </div>
  );
}

<ProductCard name="Sabre de Luz" price={599.99} />
```

Simples e poderoso. Props tornam seus componentes dinâmicos e flexíveis, como uma armadura de Tony Stark: cada peça tem sua função e pode ser atualizada sem quebrar o resto.



Estado (useState): A energia vital do componente

Componentes com **estado** são capazes de reagir às ações do usuário. É o que dá vida à sua interface.

```
...                                         Counter
import { useState } from "react";

export function Counter() {
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>🚀 Contagem: {count}</p>
      <button onClick={() => setCount(count + 1)}>Aumentar</button>
    </div>
  );
}
```

Toda vez que o estado muda, o componente **re-renderiza automaticamente**. É o mesmo princípio dos portais interdimensionais: cada vez que o valor muda, o React abre um novo “portal visual” com a interface atualizada.



UseEffect: Controle o ciclo de vida (como um Jedi do código)

O useEffect permite reagir a mudanças e eventos externos, como chamadas de API, timers ou listeners.

```
••• GithubUser

import { useEffect, useState } from "react";

export function GithubUser({ username }: { username: string }) {
  const [user, setUser] = useState<any>(null);

  useEffect(() => {
    fetch(`https://api.github.com/users/${username}`)
      .then(res) => res.json()
      .then(setUser);
  }, [username]);

  if (!user) return <p>Carregando...</p>;

  return (
    <div>
      <h2>{user.name}</h2>
      <p>👤 {user.bio}</p>
    </div>
  );
}
```

Resumo nerd: o useEffect é o “Observador” da Marvel do seu componente, ele vigia tudo e reage quando algo muda.



Reutilização com Custom Hooks: Crie seus próprios poderes

Hooks personalizados permitem encapsular lógica reutilizável. É como criar uma “habilidade especial” para ser usada em diferentes partes do app.

```
useLocalStorage

import { useState, useEffect } from "react";

export function useLocalStorage(key: string, initialValue: any) {
  const [value, setValue] = useState(() => {
    const saved = localStorage.getItem(key);
    return saved ? JSON.parse(saved) : initialValue;
  });

  useEffect(() => {
    localStorage.setItem(key, JSON.stringify(value));
  }, [key, value]);

  return [value, setValue] as const;
}
```

Agora, você pode usar esse hook
como se fosse nativo:

```
const useLocalStorage

const [theme, setTheme] = useLocalStorage("theme", "dark");
```

Prático, limpo e escalável.



Context API: Compartilhando estado global

Quando vários componentes precisam dos mesmos dados, o Context API resolve o problema sem precisar repassar props manualmente.

```
● ● ●           ThemeProvider

import { createContext, useContext, useState } from "react";
const ThemeContext = createContext<any>(null);

export function ThemeProvider({ children }: { children: React.ReactNode }) {
  const [theme, setTheme] = useState("dark");
  return (
    <ThemeContext.Provider value={{ theme, setTheme }}>
      {children}
    </ThemeContext.Provider>
  );
}

export function useTheme() {
  return useContext(ThemeContext);
}
```

Agora é só usar em qualquer parte do app:

```
● ● ●           const ThemeProvider

const { theme, setTheme } = useTheme();
```

Contexto global sem prop drilling!



Renderização Condicional e Listas: Código com lógica e estética

Em React, a renderização condicional é simples como um “if com estilo”.

```
... if  
{isLoggedIn ? <Dashboard> : <Login>}
```

E listas são renderizadas com map(), perfeito para montar interfaces dinâmicas:

```
... map  
{products.map((item) =>  
  <ProductCard key={item.id} name={item.name} price={item.price}>  
)}
```

Essas duas técnicas são o que tornam React tão elegante: o código reflete exatamente **o que você quer ver na tela**.



Ferramentas do Ecosistema React

- **Vite:**

Build ultra-rápido (adeus, esperas eternas).

- **React Router:**

Navegação entre páginas com mágica.

- **Redux / Zustand / Jotai:**

Controle de estado global com diferentes abordagens.

- **Next.js:**

O lado full stack da força.

- **TailwindCSS:**

Estilização ágil e moderna.

Essas ferramentas transformam o React em um verdadeiro arsenal para o desenvolvedor moderno.

Conclusão

Seu próximo passo na jornada React

Dominar React é mais do que entender hooks e componentes, é pensar em interfaces **como sistemas vivos**.

Agora que você entende seus fundamentos, está pronto para explorar bibliotecas mais avançadas e arquiteturas modernas, como **Next.js, Server Components e React Suspense**.

Continue praticando, refatorando e construindo. Porque, no fim das contas, **código é poder**. E o React é sua **espada lendária**.

Referências:

React Official Docs - react.dev

Vite - vitejs.dev

React Router - reactrouter.com

Next.js Documentation - nextjs.org/docs

TailwindCSS - tailwindcss.com

MDN Web Docs - [JSX Guide](#)