

DESVENDANDO O CÓDIGO

REACT DO ZERO

Seus primeiros passos na prática



COMPONENTES • JSX • PROPS • USESTATE

Aprenda os fundamentos e descubra projetos reais.

PROF. MARCOS CALDAS

Antes de começar

Você já viu uma página atualizar informações sem recarregar tudo? Ou um formulário que responde imediatamente ao que foi digitado? O React ajuda a construir interfaces com esse tipo de comportamento.

Neste guia, você vai entender os fundamentos e criar uma pequena aplicação interativa. A proposta não é decorar comandos: é descobrir por que o código funciona.

O que você vai aprender

- O papel do React em uma aplicação web.
- Como criar um projeto usando Vite.
- O que são componentes, JSX e props.
- Como usar o useState para atualizar a tela.
- Como praticar e continuar para projetos maiores.

Pré-requisito: noções básicas de HTML, CSS e JavaScript, especialmente funções, variáveis e eventos.

React: o que ele resolve?

React é uma biblioteca JavaScript para construir interfaces de usuário. Em vez de organizar uma página inteira como um bloco único, você pode dividi-la em componentes que representam partes da tela.

Imagine um catálogo de produtos

Ele pode ter um cabeçalho, um formulário de cadastro, uma lista e cartões individuais. Cada parte pode virar um componente com sua própria responsabilidade.

- Header: apresenta o título da aplicação.
- FormProduto: recebe as informações digitadas.
- ListaProdutos: organiza os itens na tela.
- Produto: apresenta um item da lista.

Essa divisão facilita a leitura, a reutilização e a manutenção do código. O React não substitui o JavaScript: ele utiliza JavaScript para descrever como a interface deve se comportar.

Uma ideia essencial

Quando os dados usados por um componente mudam, o React pode renderizar novamente a interface para refletir o novo estado. Você descreve o resultado desejado; o React coordena a atualização.

CAPÍTULO 2

Prepare o ambiente

Você precisa ter o Node.js instalado para usar as ferramentas de desenvolvimento. Neste exemplo, vamos criar uma aplicação com Vite, que prepara a estrutura inicial do projeto.

Criando o projeto

```
npm create vite@latest meu-primeiro-react  
cd meu-primeiro-react  
npm install  
npm run dev
```

No assistente de criação, escolha React e JavaScript. Depois, abra no navegador o endereço local mostrado no terminal.

O que esses comandos fazem?

- `npm create vite@latest`: inicia a criação do projeto.
- `npm install`: instala as dependências.
- `npm run dev`: inicia o servidor de desenvolvimento.

A pasta `src` concentra o código da aplicação. Um arquivo importante é o `App.jsx`, onde você pode começar a criar sua interface.

CAPÍTULO 3

Seu primeiro componente

Um componente React pode ser uma função JavaScript que retorna o conteúdo da interface. Por convenção, usamos nomes iniciados por letra maiúscula.

Exemplo: uma mensagem de boas-vindas

```
function BoasVindas() {  
  return <h2>Bem-vindo ao React!</h2>;  
}  
  
export default BoasVindas;
```

Para mostrar o componente dentro de App.jsx, importe-o e use sua tag:

```
import BoasVindas from "./BoasVindas";  
  
function App() {  
  return <BoasVindas />;  
}  
  
export default App;
```

Perceba a ideia: em vez de repetir o mesmo trecho em vários lugares, podemos transformar uma parte da interface em componente e reutilizá-la.

Pense como programador

Se a aplicação tivesse três páginas com a mesma mensagem, faria sentido copiar o código três vezes? Ou criar um componente reutilizável?

CAPÍTULO 4

JSX: HTML dentro do JavaScript?

O JSX permite escrever uma sintaxe semelhante a HTML dentro dos componentes. Apesar da aparência, JSX não é simplesmente um arquivo HTML: ele é transformado em código JavaScript durante a construção da aplicação.

```
function Saudacao() {  
  const nome = "Ana";  
  
  return <h2>Olá, {nome}!</h2>;  
}
```

As chaves { } permitem inserir expressões JavaScript no JSX. Neste exemplo, o valor da variável aparece no texto.

Duas regras úteis

- Use `className` para classes CSS em JSX.
- Um componente deve retornar um único elemento raiz ou um fragmento.

```
return (  
  <>  
    <h1>Minha aplicação</h1>  
    <p>Aprendendo React.</p>  
  </>  
);
```

O fragmento `<>...</>` agrupa elementos sem acrescentar uma nova `div` ao HTML renderizado.

CAPÍTULO 5

Props: dados entre componentes

Imagine um cartão que mostra o nome de uma pessoa. Você não precisa criar um componente diferente para cada nome: pode passar informações por props.

```
function Cartao({ nome }) {  
  return <p>Aluno: {nome}</p>;  
}  
  
function App() {  
  return (  
    <>  
      <Cartao nome="Ana" />  
      <Cartao nome="Pedro" />  
    </>  
  );  
}
```

O componente Cartao recebe o valor de nome e o utiliza na interface. O componente pai escolhe o dado; o filho o apresenta.

Atenção

Props são entradas recebidas pelo componente. Não devemos alterá-las diretamente dentro do componente filho. Quando uma informação precisa mudar ao longo do tempo, normalmente utilizamos estado.

useState: quando a tela precisa mudar

Uma variável JavaScript comum pode guardar um valor, mas mudar essa variável não faz o React atualizar automaticamente o componente. Para dados que afetam a interface, podemos usar o hook `useState`.

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

function Contador() {
  const [numero, setNumero] = useState(0);

  function aumentar() {
    setNumero(numero + 1);
  }

  return (
    <>
      <p>Cliques: {numero}</p>
      <button onClick={aumentar}>
        Aumentar
      </button>
    </>
  );
}
```

Aqui, `numero` representa o valor atual e `setNumero` solicita uma atualização do estado. Quando o usuário clica, o React renderiza o componente com o novo valor.

CAPÍTULO 7

Construindo seu primeiro projeto

Vamos juntar os conceitos em uma pequena aplicação: um contador de estudos. A pessoa marca quantos exercícios resolveu e acompanha o progresso na tela.

Substitua o conteúdo de App.jsx

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

function App() {
  const [exercicios, setExercicios] =
    useState(0);

  function concluir() {
    setExercicios(valor => valor + 1);
  }

  function reiniciar() {
    setExercicios(0);
  }

  return (
    <main>
      <h1>Meu progresso</h1>
      <p>Exercícios: {exercicios}</p>
      <button onClick={concluir}>
        Concluir exercício
      </button>
      <button onClick={reiniciar}>
        Reiniciar
      </button>
    </main>
  );
}

export default App;
```

CAPÍTULO 8

Entenda o que você construiu

Agora observe o funcionamento antes de continuar adicionando código.

- O componente App organiza a interface.
- O `useState(0)` define o valor inicial.
- O botão chama a função `concluir`.
- A função usa `setExercicios` para atualizar o estado.
- O React mostra a nova quantidade na tela.

Por que usar `valor => valor + 1`?

A função recebe o estado anterior e calcula o próximo valor. Essa forma é útil quando a atualização depende do valor que já estava armazenado.

Desafio 1

Adicione um terceiro botão para diminuir a quantidade de exercícios, sem permitir valores negativos.

Desafio 2

Exiba a mensagem "Meta alcançada!" quando o contador chegar a cinco exercícios.

Desafio 3

Crie um componente separado para mostrar a quantidade de exercícios. Passe o número usando props.

Erros comuns de quem começa

1. Chamar a função durante a renderização

```
// Correto: passa a função
<button onClick={concluir}>OK</button>

// Evite: executa imediatamente
<button onClick={concluir()}>OK</button>
```

2. Alterar o estado diretamente

```
// Evite
exercicios = exercicios + 1;

// Prefira
setExercicios(valor => valor + 1);
```

3. Confundir React com o backend

React cuida da interface no navegador. Quando precisamos consultar ou salvar informações em um servidor, geralmente fazemos requisições a uma API. O backend recebe essas requisições, aplica regras de negócio e pode acessar um banco de dados.

Essa separação fica mais clara quando você passa de um contador simples para um catálogo, uma lista de tarefas ou um sistema de agendamentos.

Do primeiro componente ao projeto real

O contador ensina a relação entre evento, estado e interface. Mas o próximo passo é conectar os conceitos em aplicações que resolvem problemas mais completos.

O que muda em um projeto maior?

- Formulários: recebem dados do usuário e validam informações.
- Listas: exibem vários itens usando arrays e `map()`.
- CRUD: permite cadastrar, consultar, editar e excluir.
- APIs: permitem trocar informações com um servidor.
- Persistência: mantém dados disponíveis depois de atualizar a página, conforme a solução utilizada.

Uma evolução possível é começar com uma lista local, depois usar o `localStorage`, integrar uma API simulada com JSON Server e, por fim, conectar o React a um backend real.

O raciocínio continua o mesmo

Antes de escrever o código, pergunte: qual dado preciso representar, quem pode alterá-lo e qual parte da interface depende dele?

Seu próximo passo no React

Você já criou componentes, usou JSX, passou props e trabalhou com estado. Agora pode começar a construir aplicações com mais de uma tela, formulários, listagens e integração com APIs.

No curso React na Prática — Do Zero ao Full Stack com Projetos Reais, esses fundamentos são aplicados progressivamente em projetos como CRUD com armazenamento local, integração com JSON Server, consumo de APIs e aplicações que se comunicam com backend Node.js e Express.

A proposta é acompanhar a construção e entender cada decisão, saindo dos exemplos isolados para projetos completos.

Antes de seguir, confira seu aprendizado

- Consigo explicar o que é um componente?
- Sei a diferença entre props e estado?
- Entendo o que acontece quando clico em um botão?
- Consigo alterar o contador sem copiar o código?
- Sei qual seria o próximo recurso a acrescentar?

Se você conseguiu responder, já deu um passo importante: está começando a pensar como programador.

CALCULADORA DE IMC

Um primeiro projeto interativo

Estados, eventos e atualização da interface.

PROJETO REACT

Calculadora de IMC

Informe seu peso e sua altura para descobrir seu Índice de Massa Corporal.

01 Informe seus dados

Peso

75

kg

Altura

1,75

m

Calcular meu IMC →

Limpar

SEU IMC É

24.49

PESO NORMAL

Peso normal

Seu IMC está dentro da faixa considerada saudável. Continue cuidando da sua saúde.

02 REFERÊNCIA

Faixas de IMC

Abaixo do peso

Menor que 18,5

Peso normal

18,5 até 24,9

você

Sobrepeso

25 até 29,9

Obesidade grau I

30 até 34,9

Obesidade grau II

35 até 39,9

Obesidade grau III

40 ou mais

❗

O resultado é uma estimativa. Idade, massa muscular e outras características também devem ser consideradas.

O IMC é apenas uma referência e não substitui avaliação médica.

PROJETO 01 / 03

Veja a evolução dos conceitos no curso React na Prática.

GERENCIADOR DE TAREFAS

Dados que vão além do navegador

React consumindo uma API simulada com JSON Server.

</> DEV TASKS

React + JSON Server

Gerenciador de Tarefas

React consumindo uma API simulada com JSON Server

Adicionar

Aprender JSON-Server

Concluída

Reabrir

Excluir

Consumir uma API

Pendente

Concluir

Excluir

estudar javascript

Pendente

Concluir

Excluir

ESTUDAR HTML

Pendente

Concluir

Excluir

Desenvolvido com React + JSON Server

© 2026 • Desvendando o Código

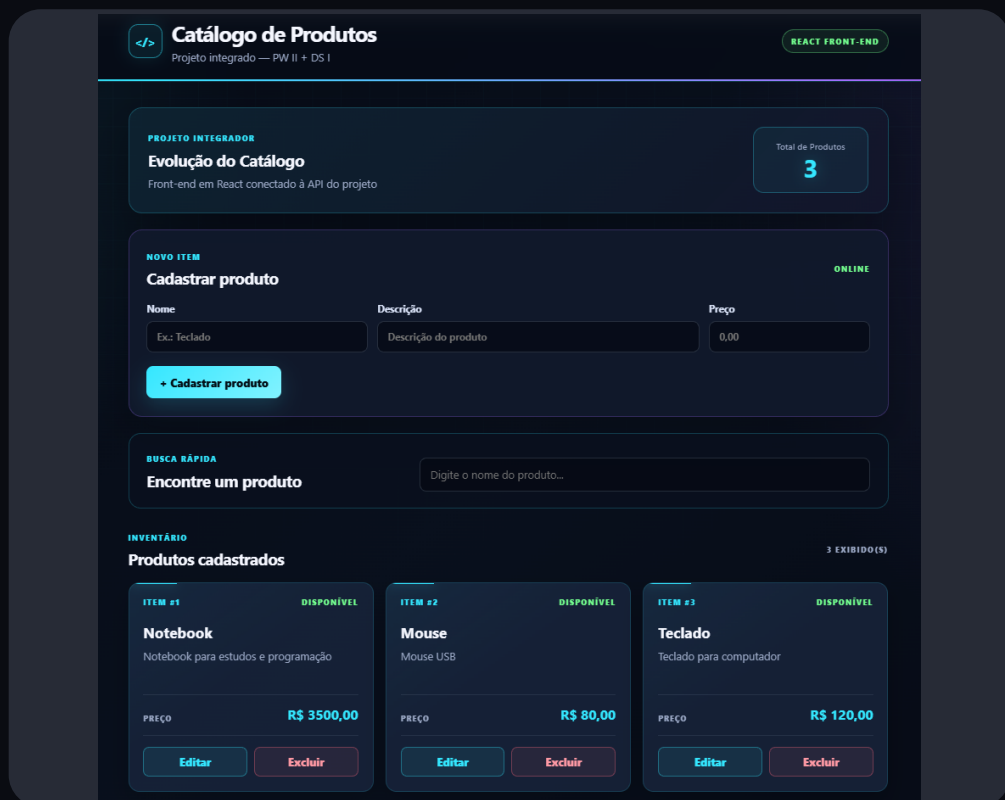
PROJETO 02 / 03

Veja a evolução dos conceitos no curso React na Prática.

CATÁLOGO DE PRODUTOS

A evolução para uma aplicação integrada

Formulários, listagem, busca e operações de CRUD.



PROJETO 03 / 03

Veja a evolução dos conceitos no curso React na Prática.

Continue aprendendo na prática

Aprender os fundamentos é o começo. O próximo desafio é construir aplicações completas, integrar diferentes tecnologias e compreender como o frontend conversa com uma API.

Conheça o curso React na Prática — Do Zero ao Full Stack com Projetos Reais, do Desvendando o Código.



Aponte a câmera para o QR Code ou toque no link abaixo.

CONHECER O CURSO REACT NA PRÁTICA

Professor Marcos Caldas • Desvendando o Código
Aprenda programação construindo de verdade.