

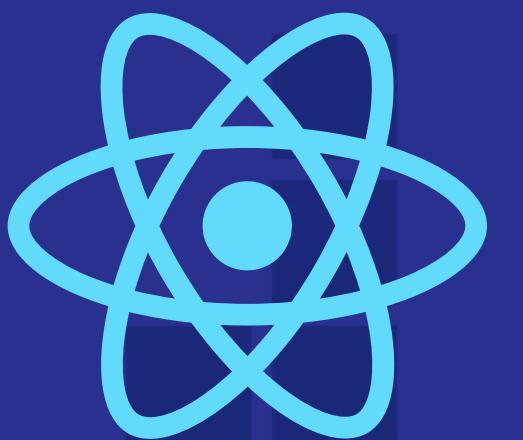


React Native

Entendendo estados (salvar e atualizar)

O que é React Native?

O React Native é um framework de desenvolvimento multiplataforma criado e mantido pela Meta para aplicações mobile.



Através dele é possível criar aplicativos para iOS e Android utilizando uma única base de código usando apenas JavaScript/ TypeScript.

Por que usar React Native?

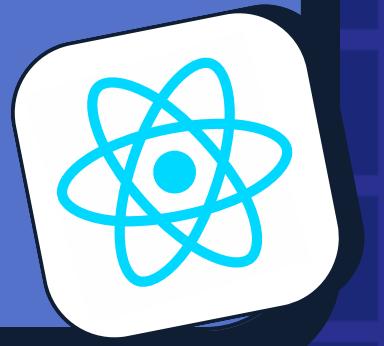


- **Vantagens:**

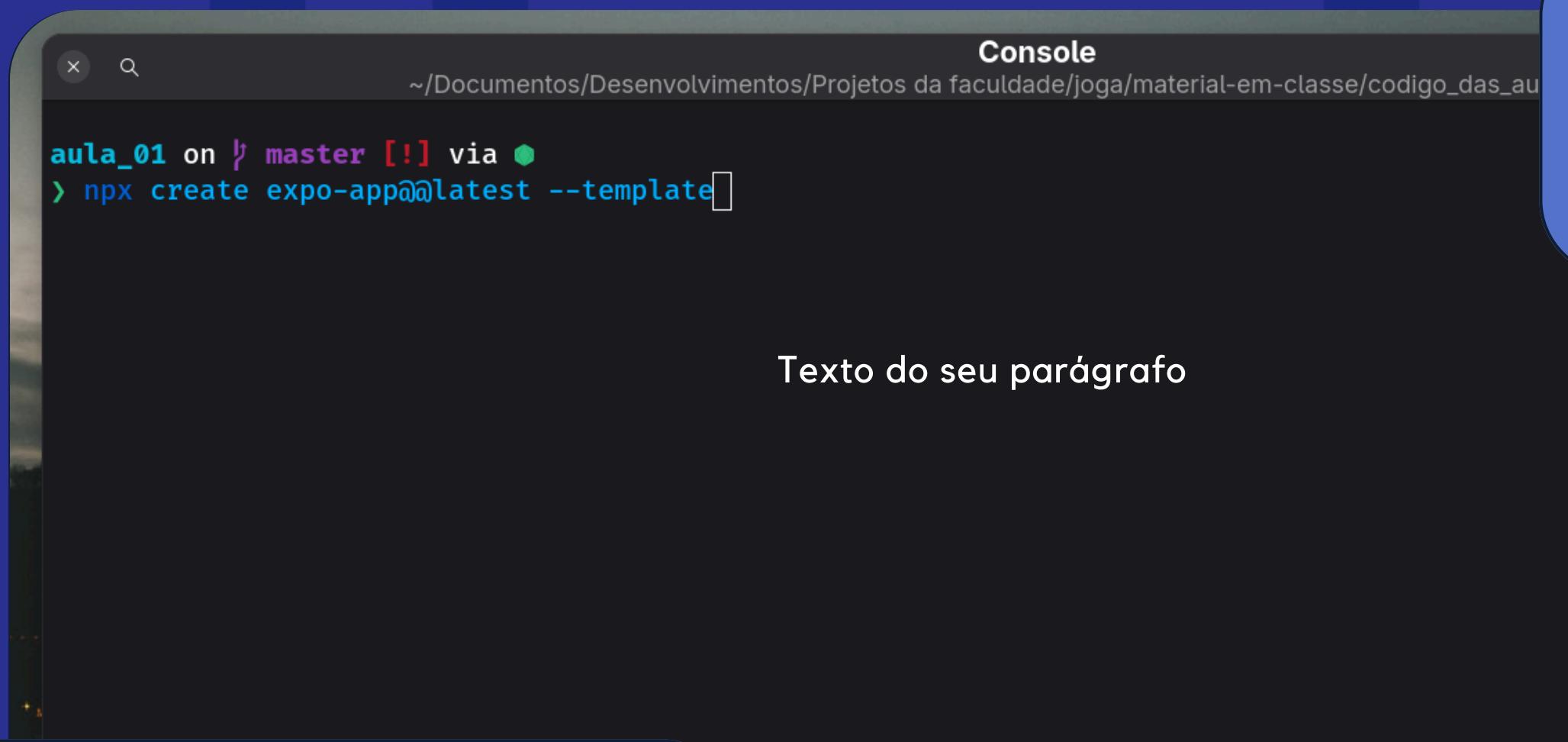
- Código único para duas plataformas.
- Performance próxima de apps nativos.
- Hot Reload (atualizações em tempo real).
- Ecossistema robusto (bibliotecas, comunidade).

Primeiro passo: pré-requisitos

1. Instalar o aplicativo **Expor Go** no smartphone
2. Instalar o **Node.js** (LTS)
3. Instalar uma **IDE** (ambiente de desenvolvimento integrado)
ou **VS Code**



Segundo passo: Iniciar um projeto



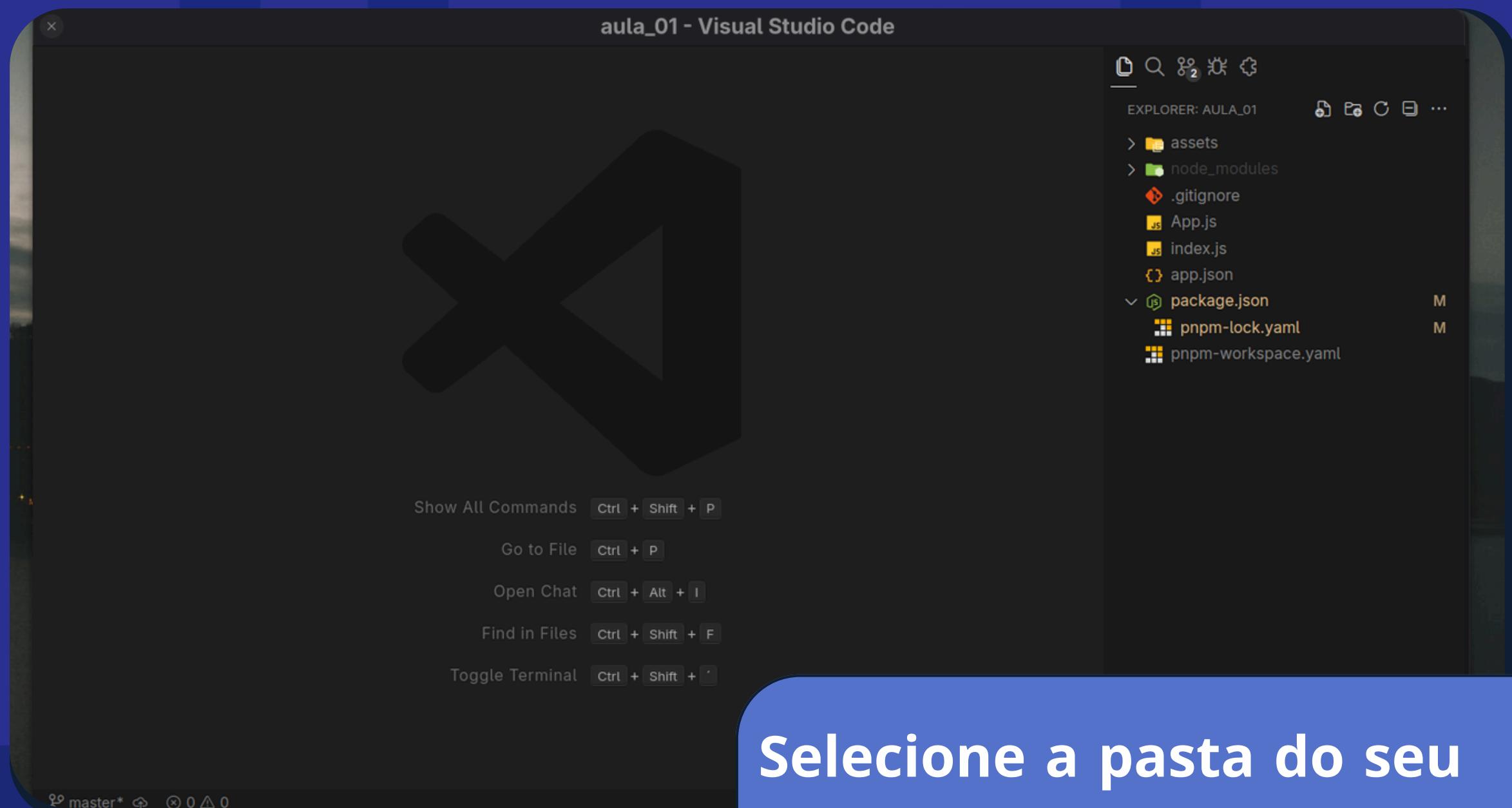
A screenshot of a terminal window titled "Console". The window shows the path: ~/Documentos/Desenvolvimentos/Projetos da faculdade/joga/material-em-classe/codigo_das_aulas. The command "npx create expo-app@latest --template" is being typed into the terminal. The terminal has a dark background with light-colored text.

Texto do seu parágrafo

npx create expo-
app@latest --template

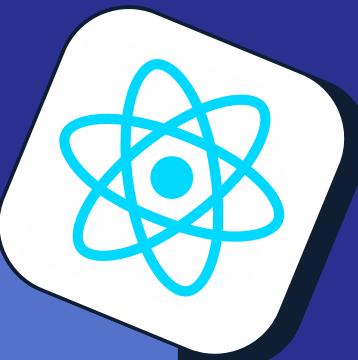
Com um terminal
aberto, uso o
comando:

Terceiro passo: abra o projeto



Selecione a pasta do seu
projeto.

Introdução



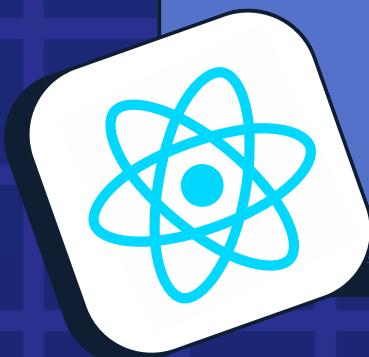
No **React Native**, o estado (**state**) é uma estrutura fundamental para **controlar** e **atualizar** informações que mudam ao longo do tempo na sua interface. Ele serve para **armazenar dados** que influenciam o que é **renderizado** na tela — por exemplo, texto digitado pelo usuário, valor de um contador ou se um botão foi clicado.



Renderizar e confirmar

No React Native, toda vez que os dados mudam — por exemplo, ao digitar em um campo ou clicar em um botão — o framework precisa atualizar a interface. Esse processo é dividido em duas etapas: **render** e **commit**.

Na **fase de render**, o React calcula o que precisa mudar. Já na **fase de commit**, ele aplica essas mudanças na tela. Esse ciclo garante que a interface sempre reflita o estado mais recente dos dados do componente.



Exemplo!



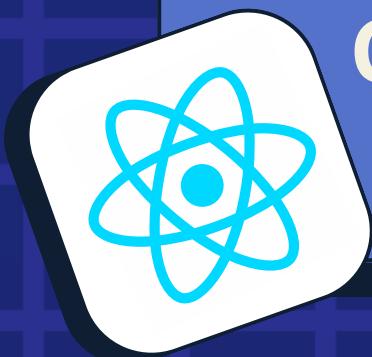
Renderizar e confirmar

Imagine que seus **componentes** são cozinheiros na cozinha, montando pratos saborosos a partir dos ingredientes. Nesse cenário, o React é o garçom que recebe os pedidos dos clientes e entrega os pratos. Esse **processo** de **solicitar** e **servir** a interface (UI) tem três etapas:

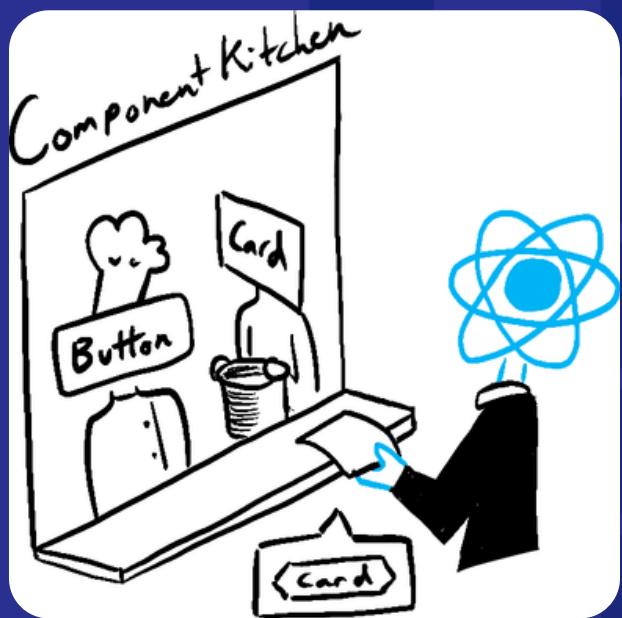
Acionar uma renderização (entregar o pedido do cliente para a cozinha)

Renderizar o componente (preparar o pedido na cozinha)

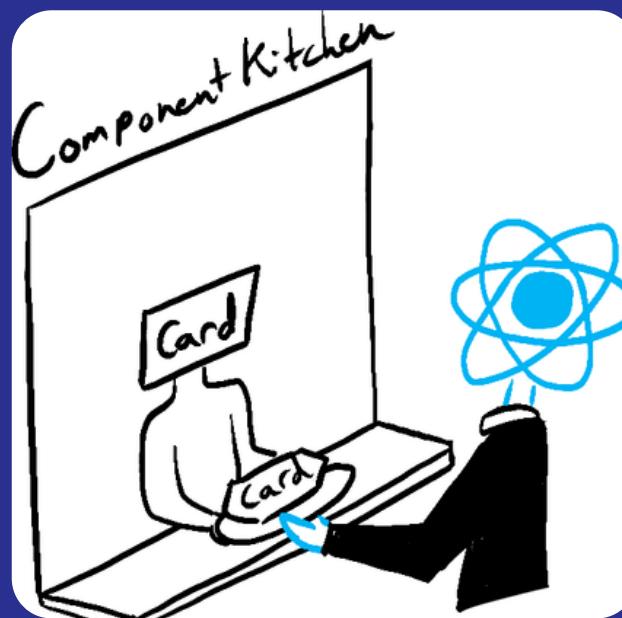
Confirmar no DOM (colocar o pedido na mesa)



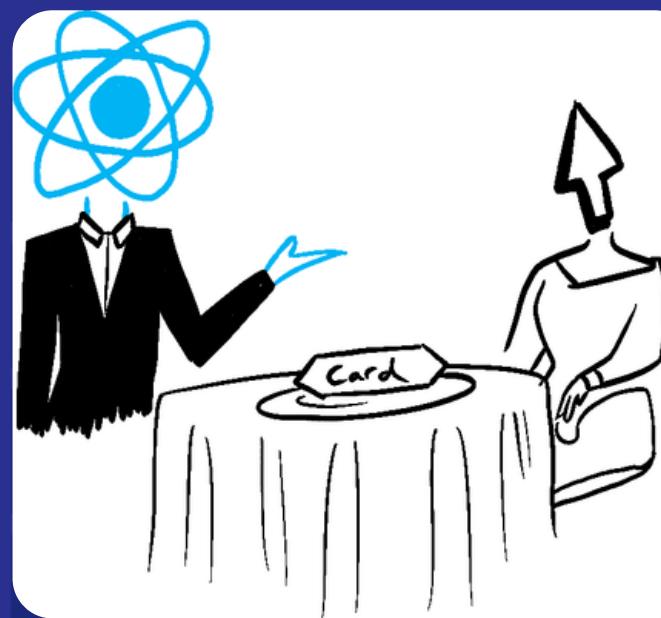
Renderizar e confirmar



Accionar



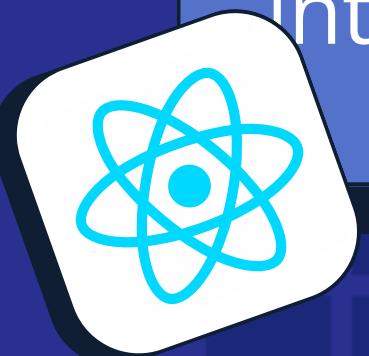
Renderizar



Confirmar

Definição

useStatus: é útil para lidar com dados que mudam com o tempo ou que vêm da interação do usuário.



Hook: é uma função do React que permite que você use recursos como *status* e outros comportamentos

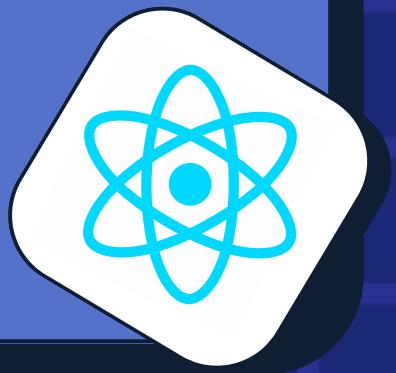


Vamos prática!
{ JavaScript }

Como usar

Embora se baseie nos mesmos **conceitos do CSS**, ela funciona de forma **diferente** por estar dentro do ambiente JavaScript. O React Native não usa arquivos .css nem classes CSS.

Toda a estilização é feita por meio de **objetos JavaScript**, usando geralmente a função StyleSheet.create(). Essa função do React Native que cria um objeto de **estilos imutável** e otimizado para performance.



Referências

- <https://react.dev/learn/updating-objects-in-state>
- <https://react.dev/learn/state-as-a-snapshot>
- <https://react.dev/learn/render-and-commit>
- <https://react.dev/learn/state-a-components-memory>
- [https://reactnative.dev/docs/next/intro-react?
language=javascript#state](https://reactnative.dev/docs/next/intro-react?language=javascript#state)



