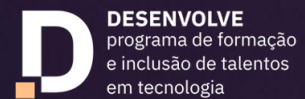







KORU





JS PURO VS REACT

AGENDA

-  Desafios do JS Puro vs UIs Complexas
-  Introdução ao React
-  Comparativo Prático
-  Quando usar cada um?
-  Próximos passos

ONDE ESTAMOS?

Nos Módulos 1 e 2, construímos interfaces web usando HTML, CSS e JavaScript Puro (Vanilla JS).

Aprendemos a estruturar conteúdo, estilizar páginas e adicionar interatividade básica com DOM e Eventos.

ONDE ESTAMOS?

O que conseguimos
fazer com JS e
Páginas Web?

Já sabemos fazer sites
interativos!

```
// Manipular o DOM
const button = document.getElementById('myButton');
button.addEventListener('click', function() {
  // Fazer algo
});

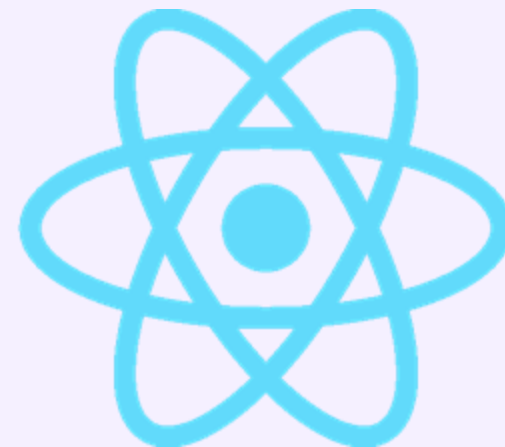
// Alterar elementos
element.textContent = 'Novo texto';
element.style.color = 'red';
```

PARA ONDE VAMOS

React!

Desenvolvimento
moderno de aplicações
web

A partir de agora, vamos
usar Typescript ao invés
de Javascript!



PARA ONDE VAMOS

Primeiro ponto: React é um pacote do NPM.

- Cuidado para não confundir React com JS!
- React foi concebido para ser utilizado em navegadores
 - Ou em mobile, com react-native.
- Quando falamos em React, falamos em desenvolvimento web
 - O que é diferente, por exemplo de fazer um CLI



JS PURO E UIS COMPLEXAS (PROBLEMAS)

JS PURO E UIS COMPLEXAS (PROBLEMAS)

[JS Puro] Manipulação Manual e Imperativa do DOM:

Adicionar, remover, atualizar elementos HTML e seus atributos/estilos exige muito código manual.

Exemplo do contador ao lado: muito código para algo simples!

```
// Código imperativo - dizemos COMO fazer cada passo
const counter = document.getElementById('counter');
const button = document.getElementById('button');
let count = 0;

button.addEventListener('click', function() {
  count++;
  counter.textContent = count;

  if (count > 5) {
    counter.style.color = 'red';
    counter.classList.add('highlight');
  }

  if (count > 10) {
    button.disabled = true;
    button.textContent = 'Máximo atingido';
  }
});
```

JS PURO E UIS COMPLEXAS (PROBLEMAS)

[JS Puro] Dificuldade em Gerenciar Estado Complexo:

Manter o controle de dados que mudam ao longo do tempo (count, lista de itens, dados de usuário).

Sincronizar esses dados com a interface visual se torna propenso a erros.

```
// Estado em variáveis globais
let userLoggedIn = false;
let cartItems = [];
let currentPage = 'home';
let searchQuery = '';
let filters = { category: 'all', price: 'any' };

// Funções que modificam estado em lugares diferentes
function login() { userLoggedIn = true; updateUI(); }
function addToCart() { cartItems.push(item); updateCartUI(); }
function search() { searchQuery = input.value; updateSearchUI(); }
```

JS PURO E UIS COMPLEXAS (PROBLEMAS)

[JS Puro] Baixa Reutilização de Código:

É difícil criar "peças" de UI que possam ser facilmente usadas em diferentes partes da aplicação.

Você usa muito *ctrl/+c* e *ctrl/+v*

Copiar e colar código leva a manutenção difícil.

```
<!-- Card do Produto 1 -->
<div class="product-card" id="product1">
  
  <h3>Produto 1</h3>
  <p>R$ 99,90</p>
  <button onclick="addToCart(1)">Adicionar</button>
</div>

<!-- Card do Produto 2 -->
<div class="product-card" id="product2">
  
  <h3>Produto 2</h3>
  <p>R$ 149,90</p>
  <button onclick="addToCart(2)">Adicionar</button>
</div>

<!-- Repetindo 100x... 🤖 -->
```

JS PURO E UIS COMPLEXAS (PROBLEMAS)

[JS Puro] Dificuldade em Manter a Sincronia (Dados vs UI):

Quando os dados mudam, você precisa lembrar de atualizar manualmente todas as partes da UI que dependem desses dados.

Ex: Se um nome de usuário muda, atualizar o cabeçalho, perfil, comentários...

```
// Quando o estado muda, temos que lembrar de atualizar TUDO
function updateUser(newUser) {
  currentUser = newUser;

  // Lembrar de atualizar todos os lugares:
  document.getElementById('userName').textContent = newUser.name;
  document.getElementById('userAvatar').src = newUser.avatar;
  document.getElementById('userEmail').textContent = newUser.email;
  document.querySelector('.header-user').textContent = newUser.name;
  document.querySelector('.sidebar-user').textContent = newUser.name;

  // E se esquecermos algum lugar? 🐛
}
```




**PRECISAMOS DE
UMA NOVA
ABORDAGEM PARA
UIS COMPLEXAS.**

PRECISAMOS DE UMA NOVA ABORDAGEM

Desenvolvimento web moderno exige interfaces dinâmicas, reativas e com grande interatividade.

Gerenciar essa complexidade apenas com JS Puro se torna lento, propenso a erros e difícil de escalar.

Precisamos de ferramentas que nos ajudem a construir UIs de forma mais eficiente e organizada.

PRECISAMOS DE UMA NOVA ABORDAGEM

Uma abordagem mais simples do que a que sabemos (imperativa) é a abordagem declarativa

PRECISAMOS DE UMA NOVA ABORDAGEM

🔍 Analogia: Fazendo um Omelete

Imperativo (JS Puro):

- Vá até a cozinha
- Abra a geladeira
- Pegue 2 ovos
- Quebre os ovos na tigela
- Bata os ovos
- Esquente a frigideira
- Despeje os ovos...

Declarativo (React):

- "Quero um omelete de 2 ovos" ✨



**ENTRA EM
CENA O
REACT!**

O QUE É REACT

Biblioteca JavaScript

Criada pelo **Facebook em 2013**

Foco: construir interfaces de usuário

"Descreveva como a UI deve parecer para qualquer estado, e o React cuida do resto"

ECOSSISTEMA FRONTEND

Biblioteca/Framework	Empresa
React	Meta (Facebook)
Vue	Evan You
Angular	Google
Svelte	Rich Harris

ECOSSISTEMA FRONTEND

Adivinha qual é o mais popular? 🙌

Biblioteca/Framework	Empresa
React	Meta (Facebook)
Vue	Evan You
Angular	Google
Svelte	Rich Harris



PRINCIPAIS CARACTERÍSTICAS DO REACT

PRINCIPAIS CARACTERÍSTICAS DO REACT

Componentização - Blocos reutilizáveis de código

Virtual DOM - Performance otimizada

Fluxo Unidirecional - Dados fluem de forma previsível

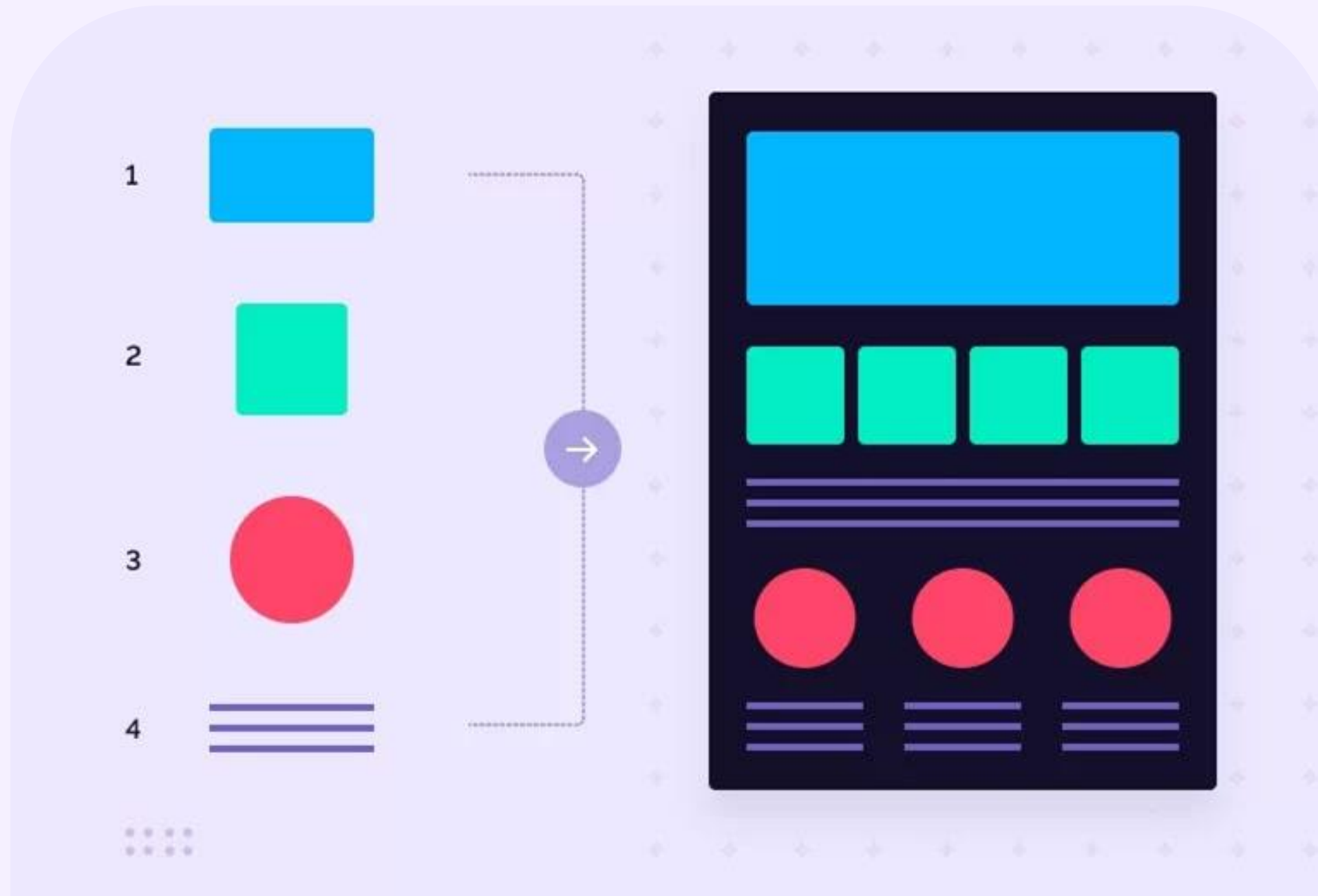
Estado Reativo - UI atualiza automaticamente

Ecossistema Rico - Milhares de bibliotecas



CONCEITOS FUNDAMENTAIS

1. COMPONENTES (COMO LEGO)



1. COMPONENTES (COMO LEGO)

```
function ProductCard({ name, price, image }) {  
  return (  
    <div className="product-card">  
      <img src={image} alt={name} />  
      <h3>{name}</h3>  
      <p>R$ {price}</p>  
      <button>Adicionar ao Carrinho</button>  
    </div>  
  );  
}
```

componente

```
// Usar quantas vezes quiser!
```

```
<div>  
  <ProductCard name="iPhone" price="5000" image="iphone.jpg" />  
  <ProductCard name="Galaxy" price="4000" image="galaxy.jpg" />  
</div>;
```


2. JSX

JSX é uma nova sintaxe para utilizar **HTML dentro do JS**

```
// Isso é JSX (não é string!)
const element = <h1>Olá, mundo.</h1>;

// Expressões JavaScript
const isLoggedIn = true;
const content = isLoggedIn ? <Dashboard /> : <p>Você não está logado.</p>;
```


3. VIRTUAL DOM

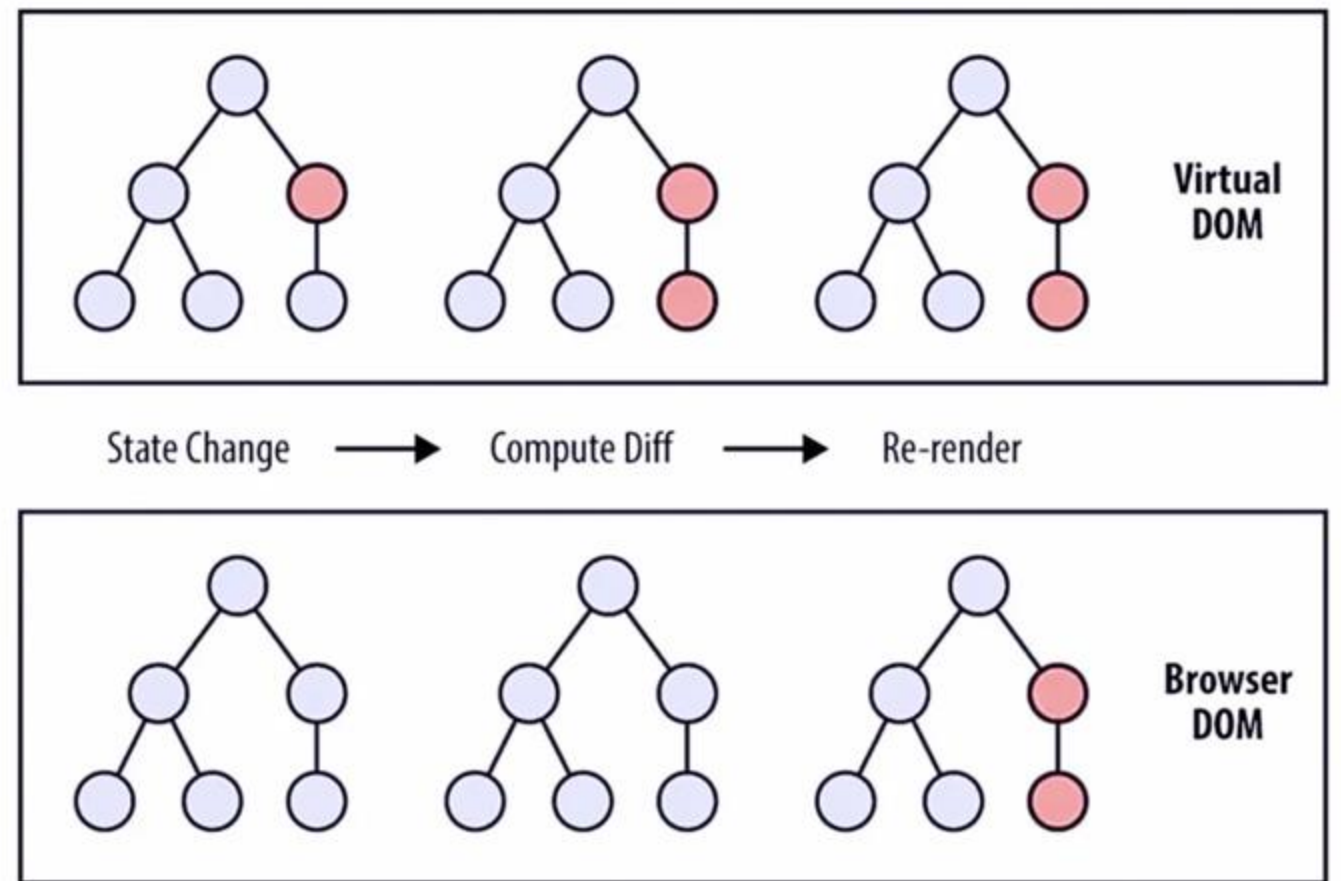
Como funciona:

Estado muda → React cria nova árvore virtual

Diff Algorithm → Compara com árvore anterior

Reconciliation → Atualiza apenas o que mudou no DOM real

Resultado: Performance otimizada ⚡



4. FLUXO UNIDIRECIONAL

Dados fluem numa
direção:

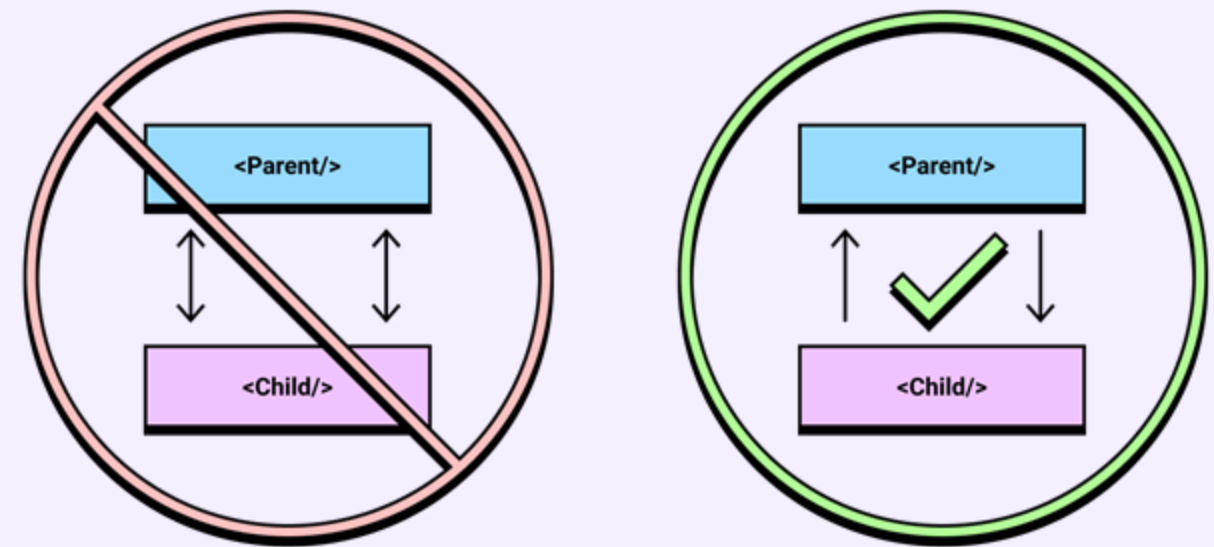
Parent Component

↓ (props)

Child Component

↓ (props)

Grandchild Component





**VAMOS VER NA
PRÁTICA!**

INSTRUÇÕES

Versão 01 do Contador.

Crie um contador usando html puro e vanilla js com os seguintes requisitos.

Mostrar número atual

Botões +1, -1, Reset

INSTRUÇÕES

Versão 02 do Contador.

Baseado no contador anterior, adicione o seguinte:

Botões de +10 e -10

INSTRUÇÕES

Versão 03 do Contador.

Baseado no contador anterior, adicione o seguinte:

Quando o contador está negativo, o número deverá aparecer com a cor vermelha.



**AGORA COM
REACT!**

INSTRUÇÕES

Versão final do Contador.

Crie um contador usando html puro e vanilla js com os seguintes requisitos.

Mostrar número atual

Botões +1, -1, Reset

Botões de +10 e -10

Quando o contador está negativo, o número deverá aparecer com a cor vermelha.

DESAFIO PRA PENSAR



INSTRUÇÕES

Agora, nosso cliente quer 5 contadores na mesma página, cada um com contagem independente.

Como você faria com vanilla JS?


Como podemos fazer com React?



VANTAGENS DO REACT


VANTAGENS DO REACT

 Produtividade - Menos código, mais resultado

 Reutilização - Componentes como LEGO

 Manutenção - Código organizado e legível

 Comunidade - Milhões de desenvolvedores

 Ecossistema - Bibliotecas para tudo

 Mercado - Muito demandado pelas empresas



**QUANDO USAR JS
PURO AO INVÉS
DE REACT?**

QUANDO USAR JS PURO AO INVÉS DE REACT?

- ✓ Projetos muito pequenos (landing page simples)
- ✓ Interatividade básica (toggle, modal simples)
- ✓ Performance crítica (casos muito específicos)
- ✓ Sem build process (HTML direto)
- ✓ Aprendizado (entender os fundamentos)

... ou nunca?
(<https://justfuckingusereact.com>)



REACT NO MERCADO

O MUNDO TODO USA REACT



Uber



A group of diverse young people, including men and women of various ethnicities, are smiling and waving at the camera. The image is overlaid with a semi-transparent purple gradient. The word "RESUMO" is written in large, bold, white capital letters on the right side of the image.

RESUMO

RESUMO

- ✓ JS Puro tem limitações em projetos complexos
- ✓ React resolve problemas reais de desenvolvimento
- ✓ Componentes são como LEGO - reutilizáveis
- ✓ JSX mistura HTML e JavaScript
- ✓ Virtual DOM otimiza performance
- ✓ Use a ferramenta certa para cada projet



**Momento de
avaliar a aula
de hoje!**

KORU

