Desenvolvimento Front-end com Frameworks

Fundamentos de React

Agenda

Etapa 1: O que é React?

- Apresentações.
- Introdução ao React.
- Paradigmas de Programação.
- Laboratório.

Apresentações

Armênio Cardoso

Iniciei minha carreira profissional em **1986** e desde **1990** procuro conciliar o trabalho em Desenvolvimento de Software com o de Professor.

Participei em diversos projetos, atuando com modelagem, arquitetura e programação. Fui desenvolvedor Pascal, Clipper, Visual Basic, C/C++, Java, JavaScript.

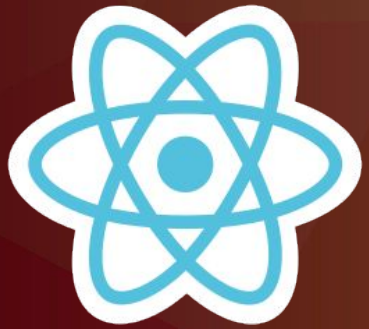
Trabalhei como tradutor / revisor de livros técnicos e professor em várias instituições.

Em **2002** vim para o Infnet onde dou aulas nos cursos de Graduação, Pós-Graduação e de Extensão.

Desde **2012** trabalho como Consultor de TI e Tech Leader na DASA - Diagnósticos da América S.A., empresa da área de medicina diagnóstica.

http://www.linkedin.com/in/**armeniocardoso**

Introdução ao React

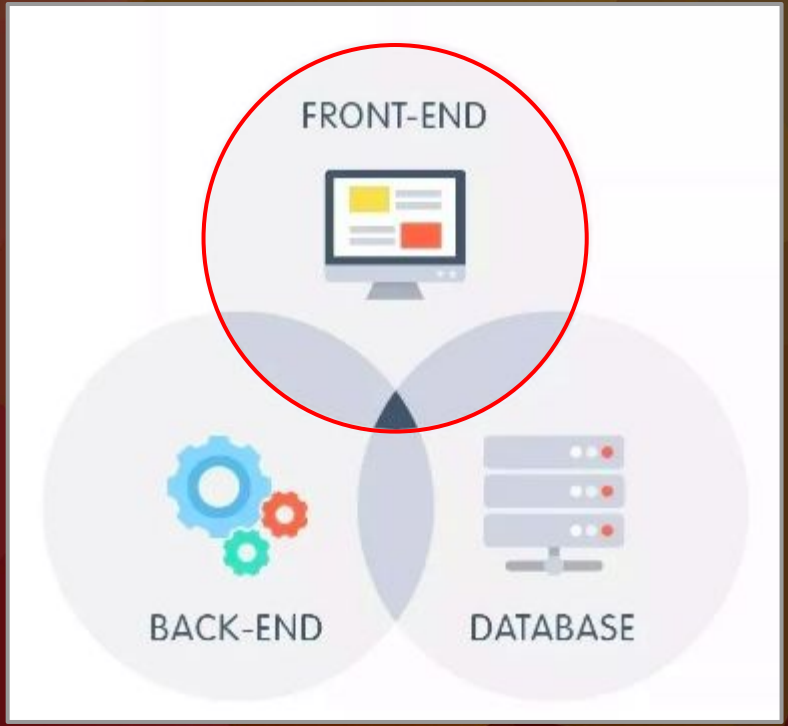


ReactJS é uma **biblioteca JavaScript** - um código criado para ser reutilizado.

Esse código compartilha funções que resolvem problemas com uma abordagem genérica.

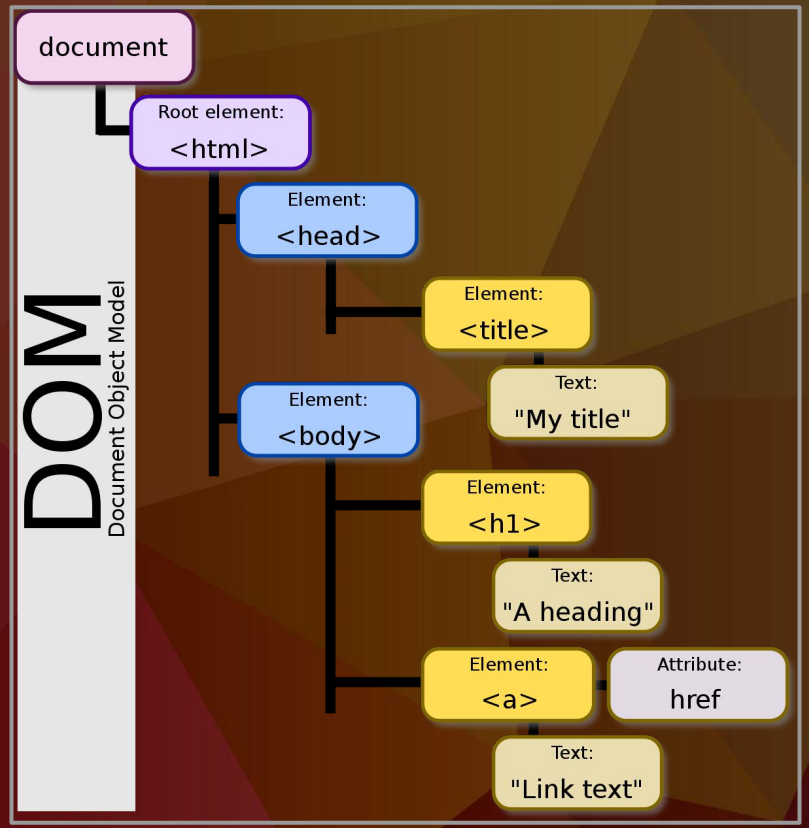
Em projetos complexos, permite ao desenvolvedor focar apenas no seu problema principal, não necessitando ‘reinventar a roda’.

React traz soluções relacionadas à construção de **interfaces de usuário - Front-end**, não apenas para aplicações web com o **React**, mas também *mobile* com **React Native** e realidade virtual com **React 360**.



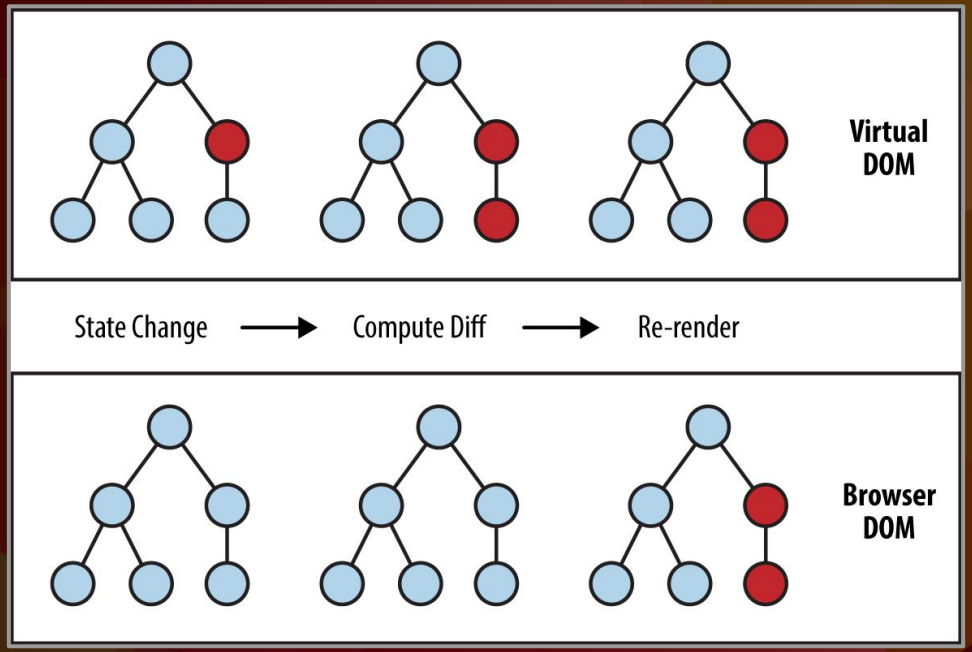
O foco principal do **React** é a **componentização** que visa reutilizar partes de interface e manipulação eficiente do **DOM** - Document Object Model.

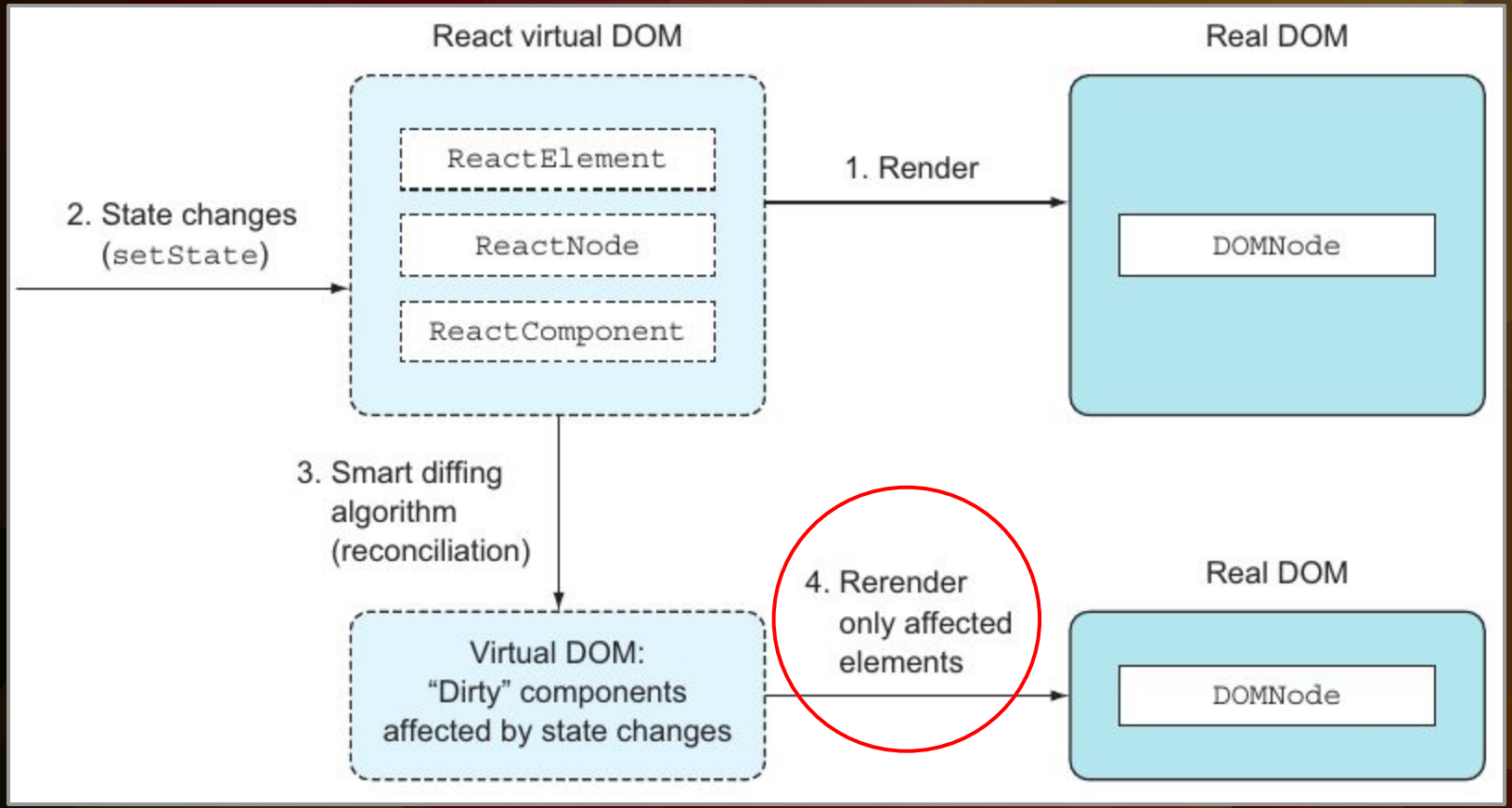
O **DOM** é uma forma de exibir a estrutura de um documento HTML como uma árvore lógica.



Virtual DOM é uma representação JavaScript leve do **Document Object Model** usado em estruturas declarativas da Web, como **React**, Vue.js e Elm.

Atualizar o **Virtual DOM** é comparativamente mais rápido do que atualizar o DOM real.




React virtual DOM
Real DOM
ReactElement
ReactNode
ReactComponent
1. Render
2. State changes
(setState)
DOMNode
3. Smart diffing
algorithm
(reconciliation)
4. Rerender
only affected
elements
Real DOM
Virtual DOM:
"Dirty" components
affected by state changes
DOMNode

Qual é o Problema a ser Resolvido?

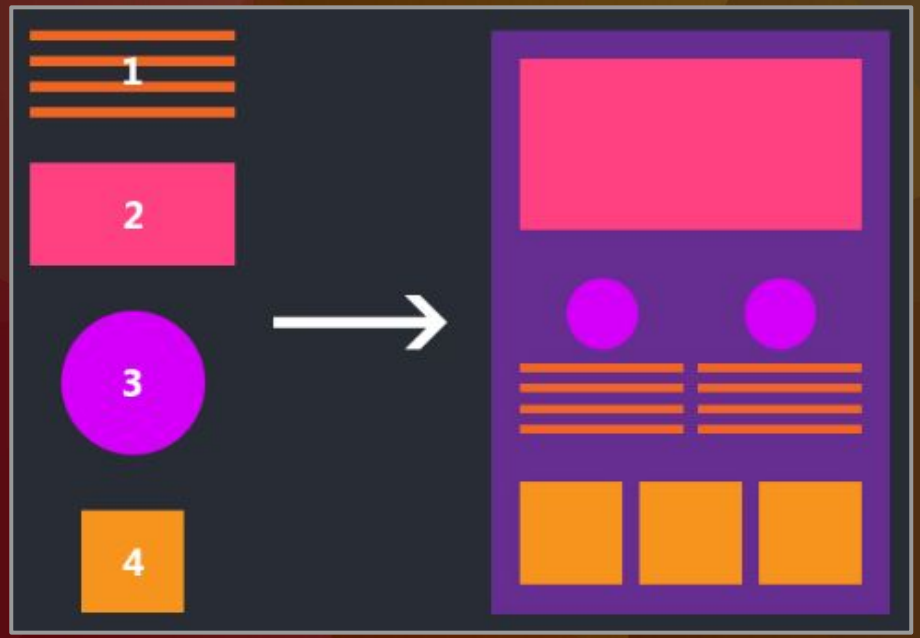
Arquitetura baseada em componentes já existia antes do **React** - visa uma construção mais fácil para reutilizar, manter e estender do que as *user interfaces* monolíticas.

Observando os últimos anos de desenvolvimento web, vemos problemas na criação e gerenciamento de *user interfaces* complexas - o **React** nasceu principalmente para abordá-los e resolvê-los.

Uma das tarefas mais complexas ao desenvolver essas *user interfaces* complexas é gerenciar como as visualizações mudam em resposta às alterações de dados.

Os componentes do **React** são blocos de funções altamente independentes e têm comportamento próprio. Tais componentes têm uma representação visual e lógica dinâmica.

Alguns componentes podem até conversar com o servidor por conta própria a fim de executar consultas e atualizações.



Um dos maiores desafios enfrentados por quem inicia no **React** é entender como a biblioteca funciona, juntamente com os pré-requisitos para usá-la.

Como o React usa **JSX**, **conhecer HTML e JavaScript é uma obrigação**.

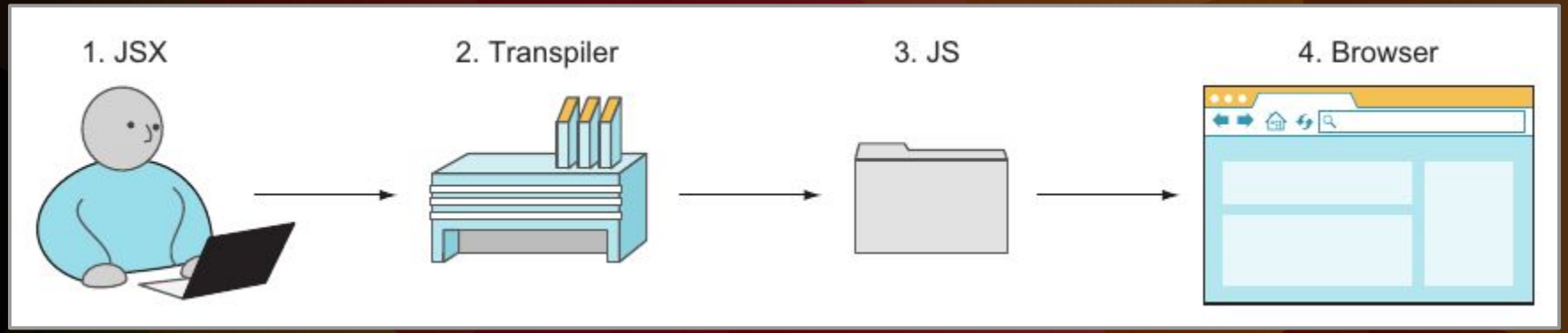
Além disso, você também deve conhecer CSS ou um framework CSS moderno para projetar seus aplicativos web.

JSX é uma extensão React para a sintaxe da linguagem JavaScript que fornece uma maneira de estruturar a renderização de componentes usando uma sintaxe familiar a muitos desenvolvedores.

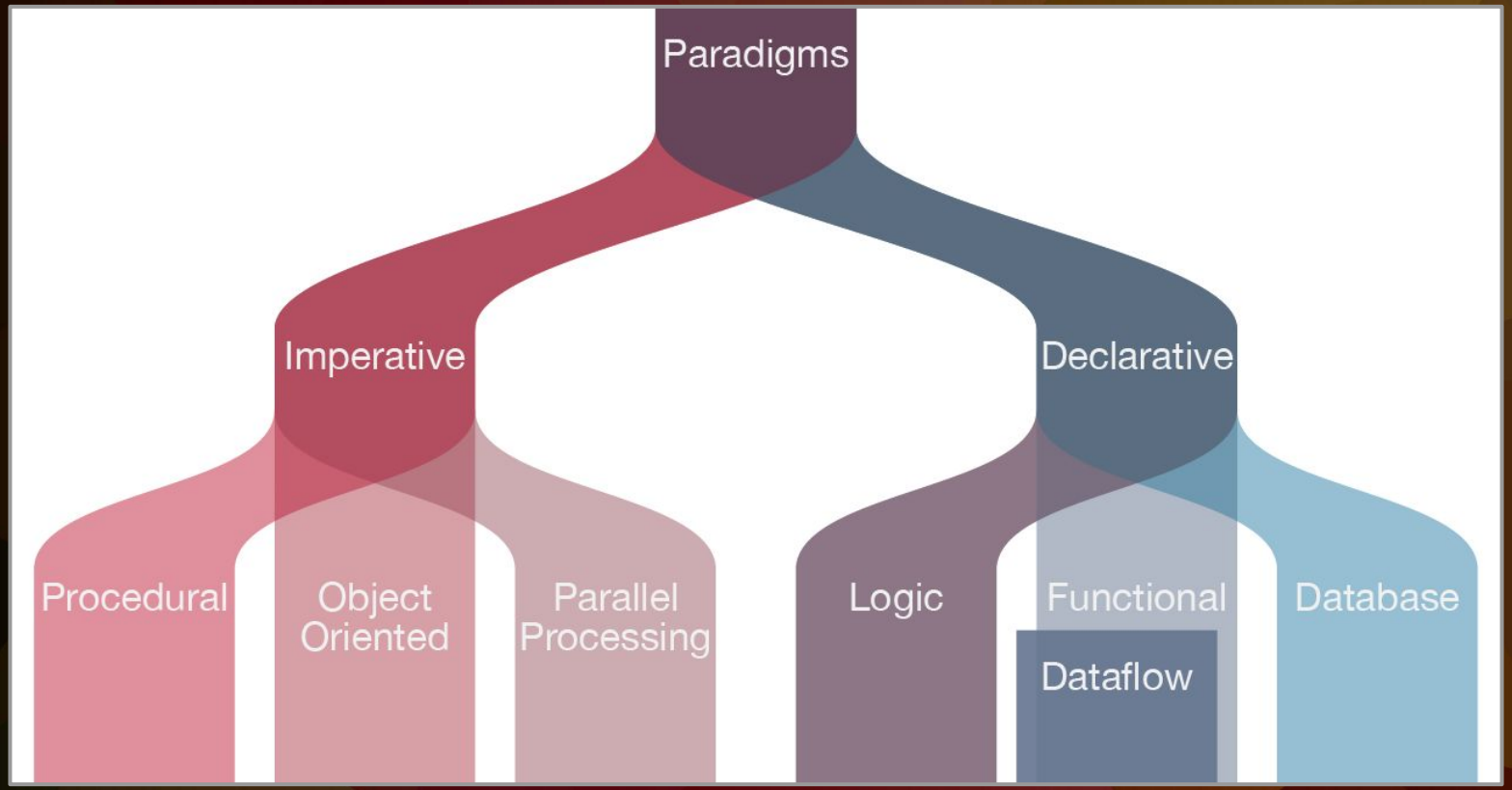
É semelhante em aparência ao HTML.

JSX não faz parte da especificação **JavaScript** e não possui nenhuma semântica definida - é próprio do **React**.

Um **transcompilador** ou **transpilador** (compilador fonte a fonte) é um tipo de compilador que usa o código fonte de um programa escrito em uma linguagem de programação como entrada e produz o código fonte equivalente em outra linguagem de programação.



Paradigmas de Programação


Paradigms
Imperative
Declarative
Procedural
Object Oriented
Parallel Processing
Logic
Functional
Dataflow
Database

Todas as linguagens de programação podem ser classificadas, com base em seus recursos, em algum **paradigma de programação**.

Alguns desses paradigmas estão preocupados com as implicações para o modelo de execução da linguagem ou se a **sequência de operações** é definida pelo modelo de execução.

Outros com a **forma como o código é organizado**, como agrupar o código em unidades junto com o estado que é modificado pelo código.

E outros ainda estão preocupados com o **estilo de sintaxe e gramática**.

A **Programação Imperativa** se concentra em COMO executar a lógica do programa e **define o fluxo de controle** como instruções que alteram o estado do programa.

As principais características incluem atribuições diretas, estruturas de dados comuns e variáveis globais.

Exemplo: C, C++, Java, PHP, Python, Ruby.

Cada declaração muda o estado do programa, desde a atribuição de valores a cada variável até a adição final desses valores.

A **Programação Declarativa** enfoca O QUE executar e define a lógica do programa, mas não um fluxo de controle detalhado.

As principais características incluem linguagens de quarta geração, planilhas e geradores de programa de relatório.

Exemplo: SQL, expressões regulares, Prolog.

Os programas declarativos podem ser descritos como independentes do contexto. o que significa que **eles apenas declaram qual é o objetivo final**, não as etapas intermediárias para atingir esse objetivo.

A **Programação Funcional** trata os programas como avaliadores de funções matemáticas e evita dados de estado e mutáveis.

As principais características incluem cálculo lambda, fórmula, recursão, transparência referencial, sem efeitos colaterais.

Exemplo: C++, Clojure, Elixir, Erlang, F #, Haskell, Kotlin, Lisp, Python, Ruby, Scala, Elm, Standard ML, JavaScript.

Programação Imperativa

Refere-se a um paradigma de programação onde fornecemos ao computador instruções passo a passo sobre **como** executar uma tarefa específica.

- Programação Estruturada
- Programação Procedural
- Programação Modular

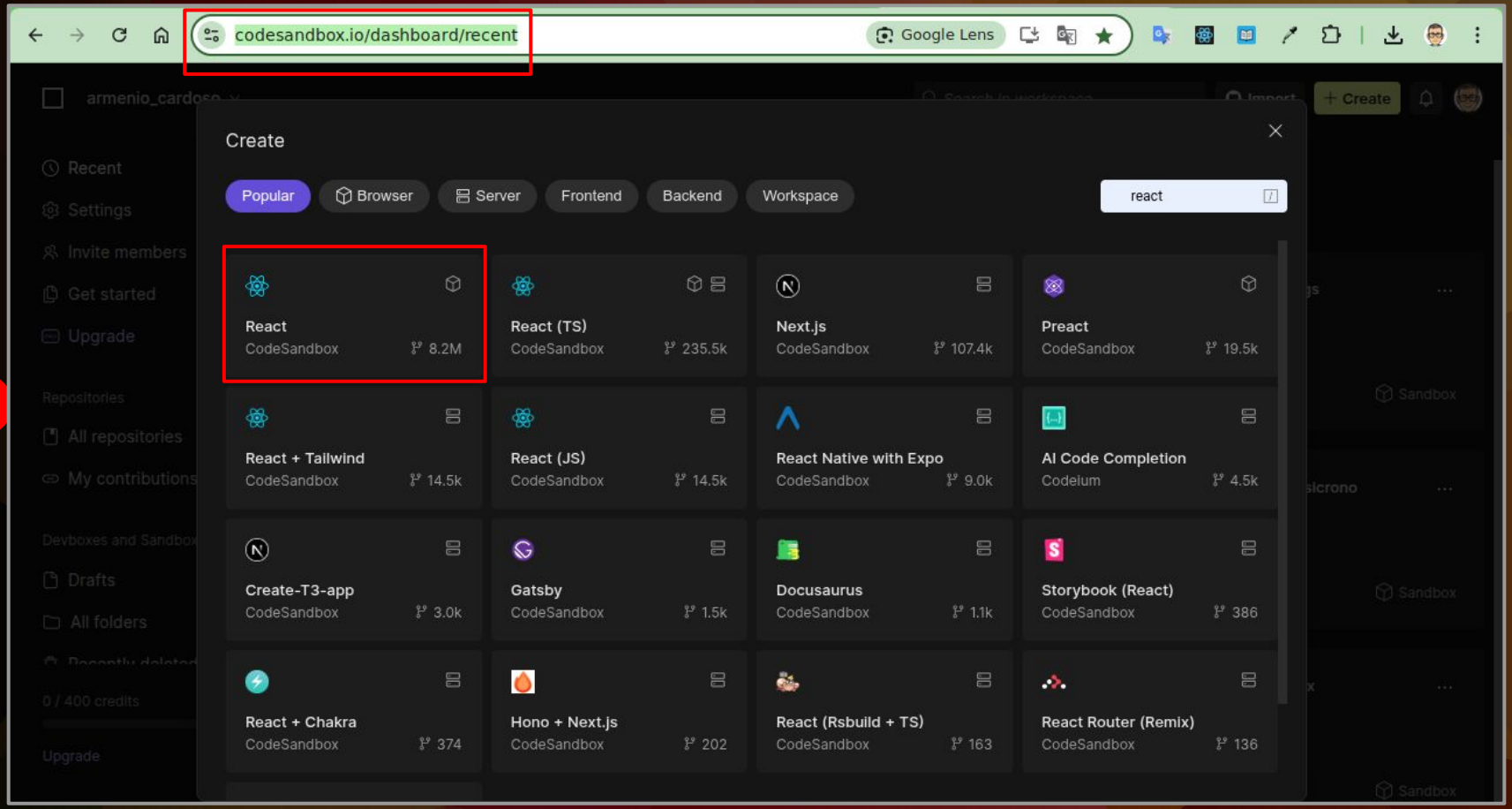
Programação Declarativa

Envolve a especificação do **resultado** que esperamos do nosso código.

Isso é conseguido principalmente através de funções e ferramentas especiais fornecidas por diferentes estruturas e bibliotecas de uma linguagem de programação.

- Programação Lógica
- Programação Funcional

Laboratório


codesandbox.io/dashboard/recent
Google Lens
armenio_cardoso
Create
Recent
Settings
Invite members
Get started
Upgrade
Repositories
All repositories
My contributions
Devboxes and Sandbox
Drafts
All folders
0 / 400 credits
Upgrade
Create
Popular
Browser
Server
Frontend
Backend
Workspace
react
1
React
CodeSandbox
8.2M
React (TS)
CodeSandbox
235.5k
Next.js
CodeSandbox
107.4k
Preact
CodeSandbox
19.5k
React + Tailwind
CodeSandbox
14.5k
React (JS)
CodeSandbox
14.5k
React Native with Expo
CodeSandbox
9.0k
AI Code Completion
Codeium
4.5k
Create-T3-app
CodeSandbox
3.0k
Gatsby
CodeSandbox
1.5k
Docusaurus
CodeSandbox
1.1k
Storybook (React)
CodeSandbox
386
React + Chakra
CodeSandbox
374
Hono + Next.js
CodeSandbox
202
React (Rsbuild + TS)
CodeSandbox
163
React Router (Remix)
CodeSandbox
136

Aqui temos o **Ponto de Entrada** da aplicação onde podemos implementar nossas telas.

2

```
# App.css M ×   JS App.js M
src > # App.css > ...
1  .container {
2    background-color: aquamarine;
3    border: 1px solid black;
4    margin: 5px;
5    padding: 5px;
6  }
```

3

```
# App.css M   JS App.js M ×
src > JS App.js > ...
1  import './App.css';
2
3  function App() {
4    return (
5      <div className="container">
6        <h3>Seja Bem Vindo ao React!</h3>
7      </div>
8    );
9  }
10
11 export default App;
```

4

App.js M

src > App.js > ...

```
1  import './App.css';
2
3  function App() {
4
5    const nome = "Machado de Assis";
6
7    return (
8      <div>
9        <div className="container">
10         <h3>{nome}, Seja Bem Vindo ao React!</h3>
11       </div>
12       <div className="container">
13         <h4>Essa é uma expressão aritmética = {2 + 2}</h4>
14       </div>
15     </div>
16   );
17 }
18
19 export default App;
```

Aqui apresentamos a sintaxe com **expressões JavaScript** que são processadas no momento de transformação do código.

5

App.js M

src > App.js > default

```
1  import './App.css';
2
3  function App() {
4
5    const nome = "Machado de Assis";
6
7    function formatarNome(umUsuario) {
8      return umUsuario.primeiroNome + ' ' + umUsuario.ultimoNome;
9    }
10
11   const usuario = {
12     primeiroNome: 'Clarice',
13     ultimoNome: 'Lispector'
14   };
15
16   return (
17     <div>
18       <div className="container">
19         <h3>{nome}, Seja Bem Vindo ao React!</h3>
20       </div>
21       <div className="container">
22         <h4>Essa é uma expressão aritmética = {2 + 2}</h4>
23       </div>
24       <div className="container">
25         <h4>Expressão processando uma funcao = {formatarNome(usuario)}</h4>
26       </div>
27     </div>
28   );
29 }
30
31 export default App;
```

Esse é um outro exemplo de uso de expressões JavaScript.

6

```
JS App.js M ×
src > JS App.js > App
1  import './App.css';
2
3  function App() {
4
5    const nome = "Machado de Assis";
6
7    function formatarNome(umUsuario) {
8      if(umUsuario) {
9        return <h4>Bem Vindo, {umUsuario.primeiroNome + ' ' + umUsuario.ultimoNome}</h4>;
10     } else {
11       return <h4>Bem Vindo Estranho</h4>;
12     }
13
14   }
```

7

```
JS App.js M ×
src > JS App.js > ...
21     return (
22       <div>
23         <div className="container">
24           <h3>{nome}, Seja Bem Vindo ao React!</h3>
25         </div>
26         <div className="container">
27           <h4>Essa é uma expressão aritmética = {2 + 2}</h4>
28         </div>
29         <div className="container">
30           {formatarNome()}
31         </div>
32       </div>
33     );
34   }
```

Aqui a função gera o elemento de tela dinamicamente.

8

App.js M | Saudacao.jsx U

src > Saudacao.jsx > ...

```
1  export default function Saudacao(props) {
2      return (
3          <div className="container">
4              <h4>Seja Bem Vindo(a), {props.nome}</h4>
5          </div>
6      )
7  }
```

Todo componente React recebe props como parâmetro

Este é um componente de tela bem simples que pode ser parametrizado.

9

App.js M | Saudacao.jsx U

src > App.js > App

```
1  import './App.css';
2  import Saudacao from './Saudacao';
```

10

App.js M | Saudacao.jsx U

src > App.js > App

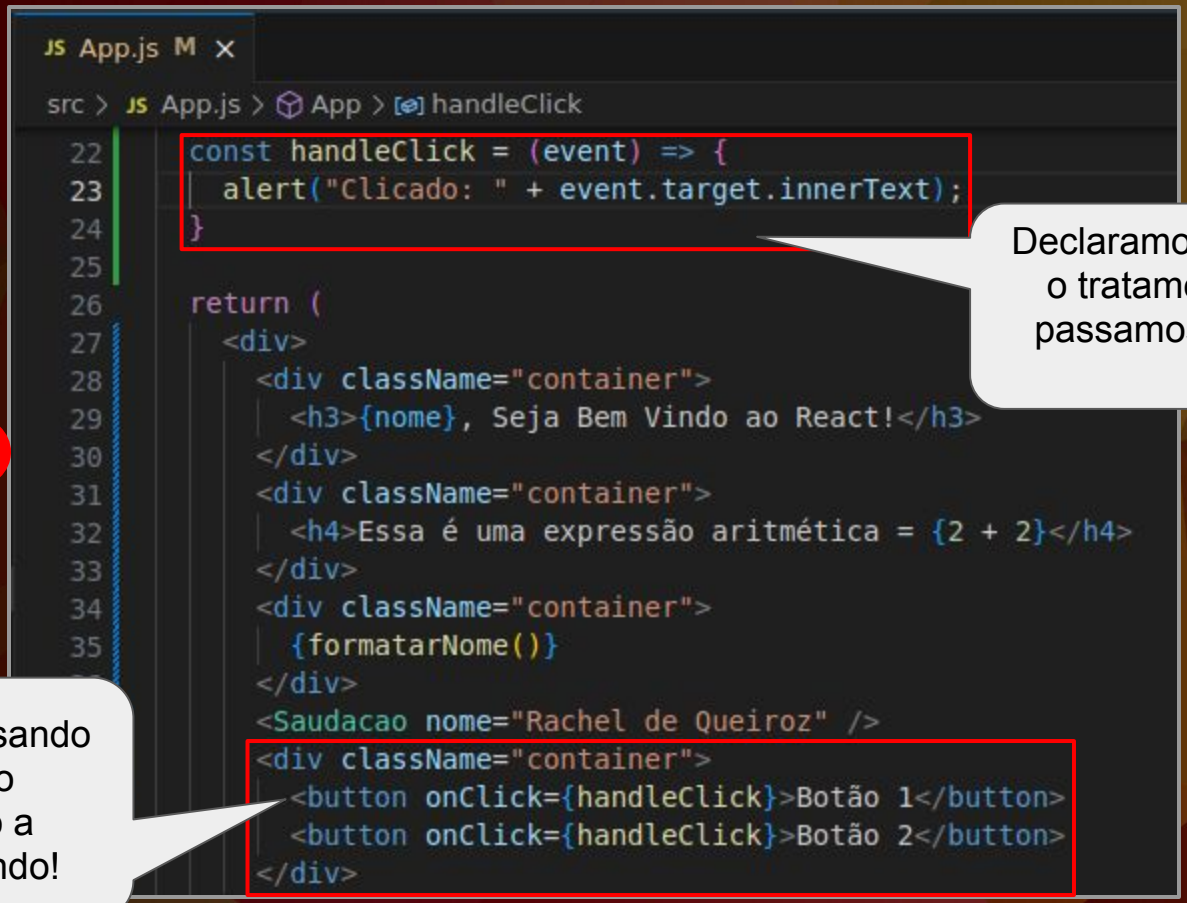
```
30             <div className="container">
31               {formatarNome()}
32             </div>
33             <Saudacao nome="Rachel de Queiroz" />
34         </div>
35     );
36 }
```

DICA

Sempre inicie os nomes dos componentes com uma letra maiúscula.

O React trata componentes começando com letras minúsculas como tags do DOM.

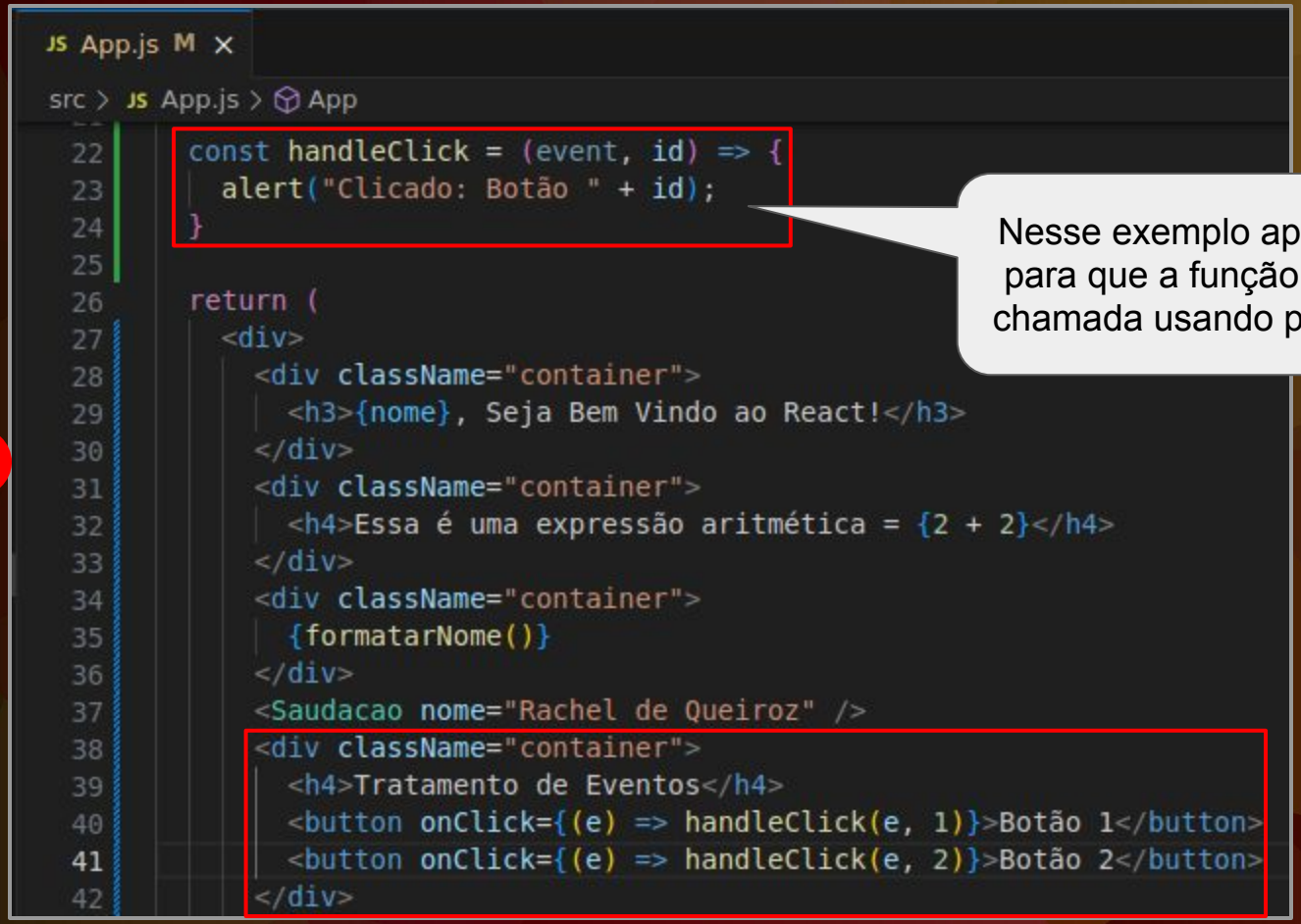
Por exemplo, representa uma tag div do HTML, mas <Welcome /> representa um componente e requer que Welcome esteja no escopo.


App.js M
src > App.js > App > handleClick
22 const handleClick = (event) => {
23 alert("Clicado: " + event.target.innerText);
24 }
25
26 return (
27
28
29 <h3>{nome}, Seja Bem Vindo ao React!</h3>
30
31
32 <h4>Essa é uma expressão aritmética = {2 + 2}</h4>
33
34
35 {formatarNome()}

<Saudacao nome="Rachel de Queiroz" />

<button onClick={handleClick}>Botão 1</button>
<button onClick={handleClick}>Botão 2</button>

11
Declaramos a função que faz o tratamento do evento e passamos a função para o objeto.
Aqui estamos passando a função como parâmetro: não a estamos chamando!


App.js M
src > App.js > App
const handleClick = (event, id) => {
alert("Clicado: Botão " + id);
}
return (

<h3>{nome}, Seja Bem Vindo ao React!</h3>

<h4>Essa é uma expressão aritmética = {2 + 2}</h4>

{formatarNome()}

<Saudacao nome="Rachel de Queiroz" />

<h4>Tratamento de Eventos</h4>
<button onClick={(e) => handleClick(e, 1)}>Botão 1</button>
<button onClick={(e) => handleClick(e, 2)}>Botão 2</button>

Nesse exemplo aprimoramos para que a função possa ser chamada usando parâmetros.

12