**SUMÁRIO**

[1. TÍTULO1 4](#_Toc51005538)

[1.1. Subtítulo1 4](#_Toc51005539)

[2. Título 2 4](#_Toc51005540)

[2.1. Subtítulo2 4](#_Toc51005541)

[2.1.1. Subtítulo do subtítulo do título2 5](#_Toc51005542)

[3. Título3 5](#_Toc51005543)

[4. BIBLIOGRAFIA 6](#_Toc51005544)

# recursos explicados

## SOBRE JSX

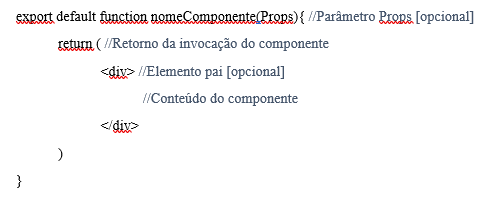
Ao invés de separar tecnologias artificialmente colocando markup e lógica em arquivos separados, o React separa conceitos com unidades pouco acopladas chamadas “componentes” que contém ambos.

Ou seja, o JSX, como descrito na introdução, é uma junção poderosa de um HTML emulado (pois não é HTML, de fato) com Javascript, o que normalmente seria impossível ou evitado ao máximo.

Para saber mais, clique [aqui](https://pt-br.reactjs.org/docs/introducing-jsx.html).

## SOBRE COMPONENTES

**Componentes, na prática, são** criados exatamente como funções, mas que são sempre exportadas e retornam elementos para renderização, e são escritas em **JSX** – a linguagem do React. Essa linguagem é uma mixagem da programação Javascript com uma “emulação” da linguagem de marcação HTML. Segue abaixo o exemplo da estrutura básica de um componente:



**O nome do componente**, que é uma função com retorno, deve ser o mesmo do arquivo js – sim, para cada componente deve haver um arquivo.

**Por sua vez, o retorno do componente**, que é o conteúdo que será renderizado ou utilizado na renderização de alguma forma, segue para onde foi invocado na aplicação, podendo ser no arquivo root App.js ou em outro componente.

**Muitas vezes o componente terá um elemento pai**, como a <div> </div> do exemplo, mas, podem haver **casos em que um não é necessário**, e para isso basta que seja escrita uma tag sem nome, <> </>, ou, ainda, para uma melhor semântica, <Fragment><Fragment/>. O nome formal para esta prática é “***React Fragment***”.

**Por último, sobre o parâmetro do componente**, “Props”, é um objeto que recebe valores, via atributos, a partir da invocação do componente na aplicação, que se dá com tags e o nome dele, assim: <nome\_componente />, ou, se houverem atributos para serem enviados, <nome\_componente atributo1 = {variavel} atributo2 = “valor” />. Outra possibilidade é utilizar o objeto Props desestruturado, function nomeComponente({atributo1: valor, atributo2: valor}).

## SOBRE PROPS

## SOBRE STATE E CICLO DE VIDA

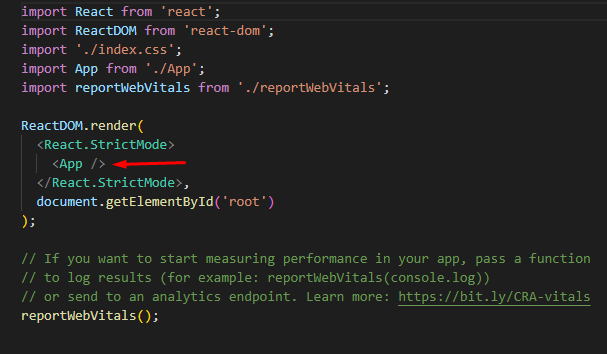
## SOBRE MANIPULAÇÃO DE EVENTOS

## SOBRE RENDERIZAÇÃO CONDICIONAL

## SOBRE O ARQUIVO ROOT APP.JS

No arquivo App.js são realizadas as importações dos componentes que serão utilizados na renderização, e também é onde os próprios componentes, é claro, são invocados para que seus retornos sejam obtidos.

Na verdade, o próprio root App.js também é um componente, e é invocado no “Entry Point” da aplicação, o index.js.



O root da aplicação: index.js