**SUMÁRIO**

[1. recursos essenciais explicados 1](#_Toc81350311)

[1.1. SOBRE O ARQUIVO ROOT APP.JS 1](#_Toc81350312)

[1.2. SOBRE JSX 1](#_Toc81350313)

[1.3. SOBRE COMPONENTES 2](#_Toc81350314)

[1.4. SOBRE PROPS 3](#_Toc81350315)

[1.5. SOBRE MANIPULAÇÃO DE EVENTOS 3](#_Toc81350316)

[1.6. SOBRE RENDERIZAÇÃO CONDICIONAL 3](#_Toc81350317)

[1.7. SOBRE HOOKS 3](#_Toc81350318)

[1.8. HOOK STATE E CICLO DE VIDA 3](#_Toc81350319)

[1.9. HOOK USEEFFECT 3](#_Toc81350320)

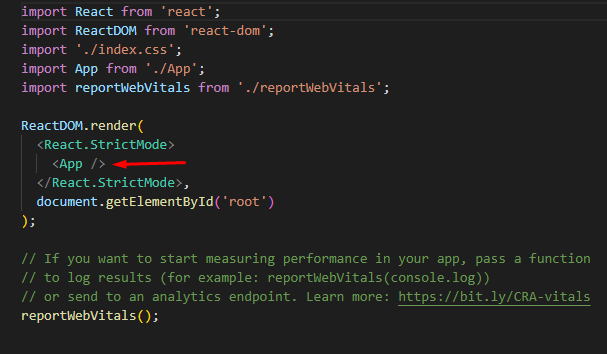
[1.10. HOOK USECONTEXT 3](#_Toc81350321)

# recursos essenciais explicados

## SOBRE O ARQUIVO ROOT APP.JS

No arquivo App.js são realizadas as importações dos componentes que serão utilizados na composição geral da aplicação. Não são todas as importações que são realizadas nesse arquivo, mas, digamos assim, as dos componentes de “maior escopo de agregação”. Por exemplo, se a página possuir um header, um aside, um main, e um footer, esses serão invocados no App.js, mas os diversos outros possíveis subcomponentes de cada um serão invocados neles próprios.

Também vale citar que o próprio root App.js também é um componente no React, e é invocado no “Entry Point” da aplicação, o index.js.



O root da aplicação: index.js

## SOBRE JSX

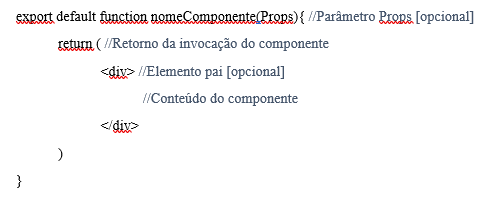
Ao invés de separar tecnologias artificialmente colocando markup e lógica em arquivos separados, o React separa conceitos com unidades pouco acopladas chamadas “componentes” que contém ambos.

Ou seja, o JSX 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

Comumente nomeados dessa forma, os “Props” são objetos enviados entre componentes, e servem para justamente transmitir dados entre diferentes segmentos da aplicação.

Pense na seguinte situação: existe um componente X, e nele existe uma determinada variável com um valor que é essencial para a programação do componente Y. Ou seja, X produz um valor essencial para Y. Como prosseguir? Na invocação de Y, no componente X, basta também definir um atributo de tag, que será enviado para Y como um argumento, e cujo valor será a própria variável, desta forma: <ComponenteY dado = {variável} />.

Assim, no componente Y, o valor será capturado como um parâmetro da função principal. Existem duas formas comuns: com o Props desestrurado seria function ComponenteY({dado}), e a outra forma, não desestruturada, seria function ComponenteY(Props), com o dado podendo ser acessado por Props.dado.

## SOBRE MANIPULAÇÃO DE EVENTOS

## SOBRE RENDERIZAÇÃO CONDICIONAL

## SOBRE HOOKS

## HOOK STATE