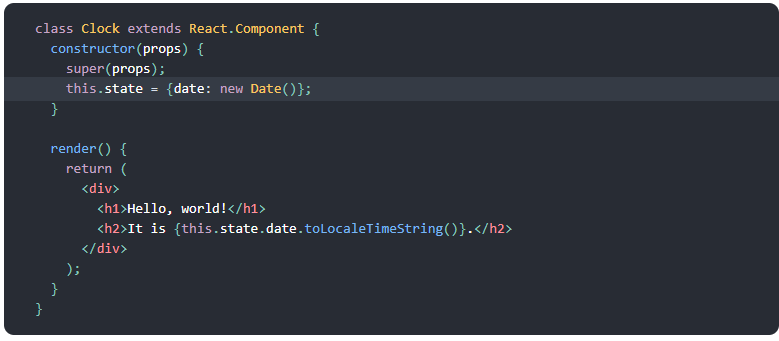
React

# O que é State (Estado)

No React, existe um objeto chamada State (Estado) responsável por observar as mudanças de um componente. Todas as propriedades desse objeto ficarão sendo *observados* pelo React e **caso** haja uma mudança na propriedade, todos os elementos que tiverem aquele propriedade alterada serão atualizados. Para utilizar o state é necessário converter um componente de função para classe.

A classe *clock* herda as propriedades de um componente React. Ele possui um construtor onde recebe as *props*, usa o *super* para passar a *props* para o componente Pai, e então, é declarado o *state* daquele componente, que possui uma única propriedade, *date*. Essa classe também possui o primeiro método do *Ciclo de Vida* do React que iremos aprender, o Render, responsável por “*Desenhar*” o componente. Note que dentro do JSX também há uma lógica JavaScript, onde acessamos a informação dentro do *State* e mostramos na tela.

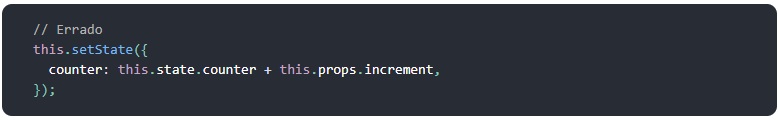
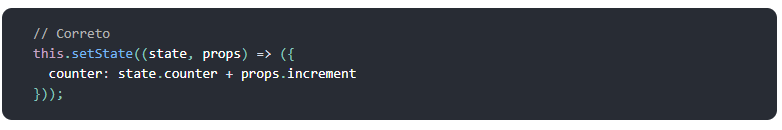
## Como atualizar o State

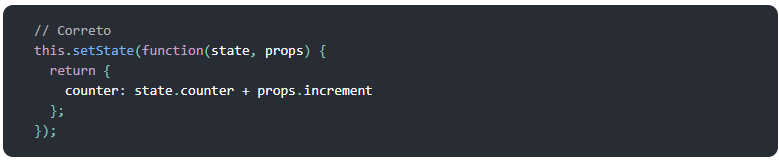
É natural pensarmos que para fazer alguma mudança no estado do componente, devemos apenas acessar *this.state* e alterar o valor do estado. **Porém**, isso não é nem um pouco recomendável, pois, não há nenhuma garantia que a mudança será propagada para todos os componentes.

Dito isso, o recomendável utilizarmos o método *setState*:



Fazendo dessa maneira, onde o novo valor da propriedade do state **NÃO** depende do valor anterior à ele, o React irá alterar **APENAS** a propriedade passada. Entretanto, esse código abaixo pode falhar:

 Isso acontece pq o método setState ocorre **assincronamente**, e por motivos de otimização, ele pode realizar a mudança de estado de mais de um componente ao mesmo tempo. Então, se por acaso um outro componente também alterar algo desse estado, podemos ter problemas. Por isso, o recomendado é **sempre** utilizar uma função:



# React Lifecycle Methods (Métodos de Ciclo de Vida do React)

O React possui, internamente, um loop onde executa em ordem, algumas funções. Os principais métodos desse ciclo de vida são os: *ComponentDidMount* e *ComponentWillUnmount*. Eles são os mais utilizados no nosso Dia a Dia.

Veja o diagrama completo em: <https://projects.wojtekmaj.pl/react-lifecycle-methods-diagram/>

Esses dois componentes são muito úteis e simples. O primeiro método, *ComponentDidMount*, é executado quando o componente está pronto no html/tela para ser usado e manipulado, dizemos que ele foi **Montado**. É utilizado para “ligar” qualquer coisa “externa” necessária para o componente funcionar, como por exemplo, uma requisição ao banco de dados.

 O segundo método, *ComponentWillUnmount*, é um método executado **antes** do componente sair da tela, nele é interessante retirar qualquer coisa “externa” ligada ao componente para ele funcionar. Veja o exemplo da documentação do React:

Nesse exemplo, o objetivo é desenvolver um relógio. Então, a cada 1 segundo devemos executar a função para atualizar o relógio.

Quando o componente for desmontado por algum motivo, como por exemplo: mudança de rota. Devemos deletar esse intervalo responsável por chamar o tick, pois, mesmo com o componente **desmontado**, o intervalo irá continuar na memória do navegador e continuará chamando a função *this.tick*. Porém, no momento em que voltarmos a montar o componente, um novo intervalo será criado, e agora 2 intervalos chamarão a mesma função, e assim por diante.... além de gerar um problema na nossa aplicação, também teremos problema de performance.

# Manipulando Eventos

O React funciona através de **componentes controlados**. Isso significa que toda a “**verdade**” do documento é proveniente de seu **estado**. Realizar as manipulações de eventos no React é semelhante ao DOM, **mas** devemos ter alguns cuidados.

* Eventos em React são nomeados usando camelCase ao invés de letras minúsculas.
* Com o JSX você passa uma função como manipulador de eventos ao invés de um texto.

Por exemplo, com HTML:

É ligeiramente diferente com React:

Fontes:

<https://pt-br.reactjs.org/docs/handling-events.html>