* O que são **hooks**: na versão 16.8 do React, foram lançados os **hooks**, que permitem o uso de vários recursos de forma simples através de funções. Eles também ajudam a organizar a lógica utilizada dentro dos componentes
* Todos os hooks são importadas da biblioteca React
* Os hooks são funções

Tipos de Hooks

* **useState**: utilizado para lidar com estado (comportamento) dentro de um componente, ou seja, lida com as alterações e diferenças que um componente pode ter ao decorrer do tempo.

Recebe o valor inicial do componente, ou seja, qual estado ele terá ao ser renderizado em tela e retorna um **array** com duas opções: uma delas é o estado (valor do componente) e a outra é uma **função para alterar o estado** que é criada no seguinte padrão: **setNome-do-estado**

Exemplo:

const [name, setName] = useState('Eduardo');

* **useEffect**: hook que trabalha com o ciclo de vida do componente. O primeiro parâmetro desse hook é uma função que faz algo e o segundo parâmetro é o **array de dependência** (quando queremos que o useEffect seja executado novamente)

Exemplo:

useEffect(() => {}, **[]**);

* Se usarmos o useEffect conforme o exemplo acima, sem a especificação do segundo parâmetro , o componente só será renderizado uma vez, quando ele for montado em tela
* É importante sempre passar o **array de dependência**, para evitar que a renderização entre em loop infinito
* **useMemo**: memoriza algum cálculo, alguma ação, algum valor que não precisamos renderizar o tempo todo. Dessa forma, quando temos algo que consome bastante nossa memória no programa, podemos passar esse algo para dentro do useMemo, fazendo com que aquilo seja executado somente uma vez e memorizado pelo hook, ao invés de executar sempre que renderizar

Obs.: o hook useMemo também tem um **array de dependência**

Exemplo:

const calculo = useMemo(() => {

return 10 \* 26589

}, []);

* No exemplo acima, o calculo com números grandes consumiria muita memória
* Para que algo não fique fixo no useMemo, passamos no **array de dependência** o que queremos que seja alterado conforme as renderizações
* **useCallback**: bem semelhante ao useMemo, porém ao invés de memorizar uma variável/valor, ele memoriza uma função. O useCallback também tem como parâmetro o **array de dependência**
* Ele evita rerenderizações desnecessárias