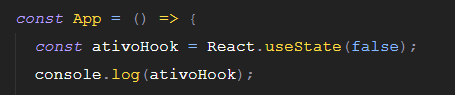
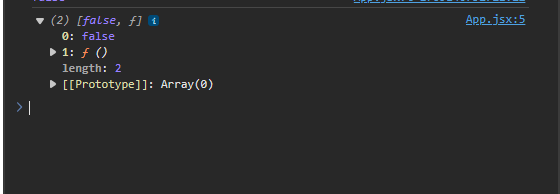
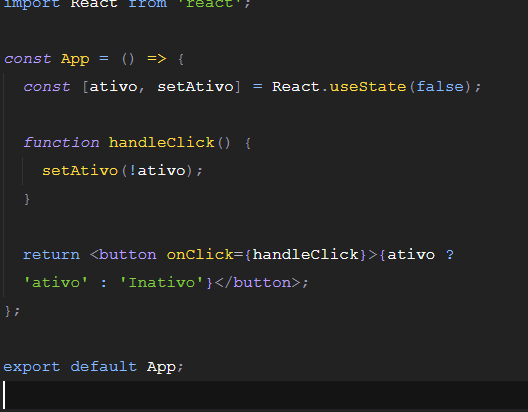
**REACT**

1. **HOOKS**
   1. **– UseState**

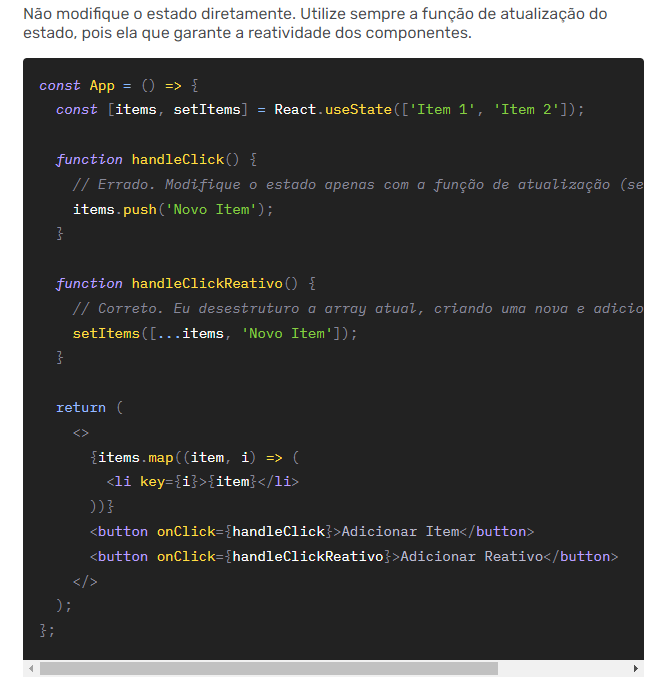
****

****

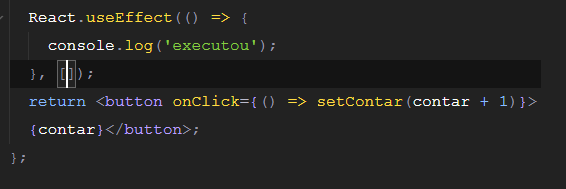
**O useState retorna nada mais que uma array com dois valores, onde o primeiro elemento é o estado inicial que no exemplo foi atribuído como false e o segundo vai ser uma função que vai ser responsável por atualizar o valor inicial.**

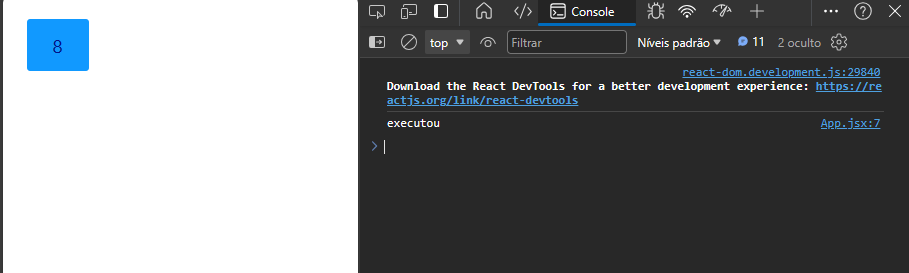
****

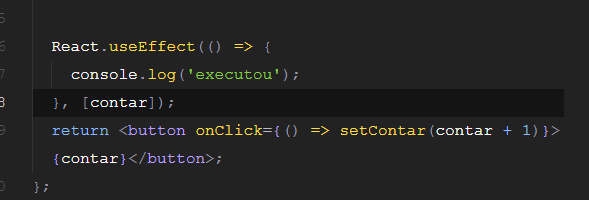
**Ele é realmente utilizado dessa forma, desestruturado.**

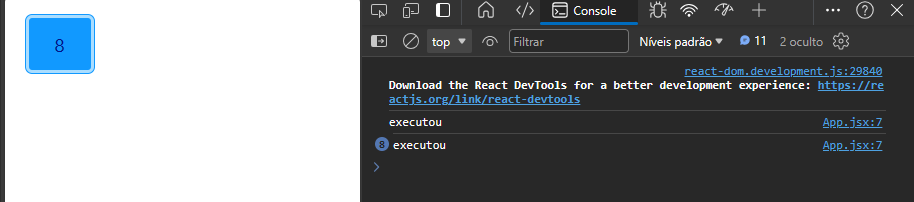
****

* 1. **– UseEffect**

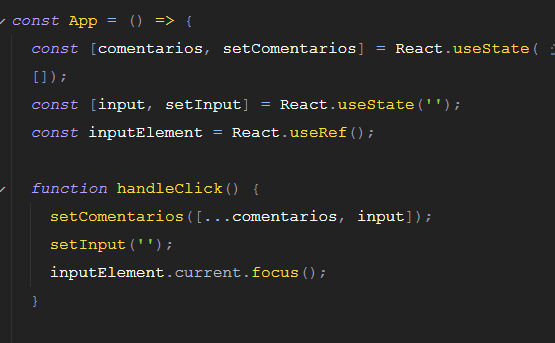
****

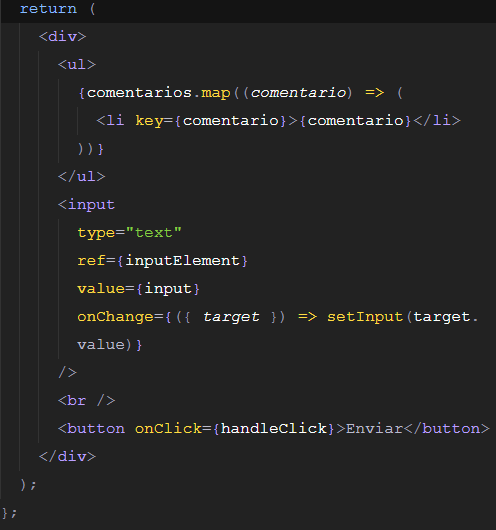
****

****

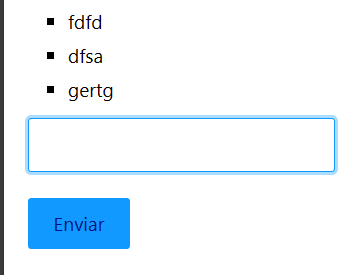
****

* 1. **– UseRef**

****

****

**Comentários com um valor inicial de lista vazia, o que acontece é o seguinte. Foi definido que vai ser mostrado uma lista de comentários a partir da lista de comentários. Podemos observar com o map em comentários onde cada li vai ser um comentário. O comentário é escrito através do input, e por isso precisamos de um useState de input também pois ele começa com valor inicial vazio, porém podemos perceber no onChange que a função de setInput que é responsável por alterar o valor de input pega o valor escrito no campo de input e logo abaixo em button a função de onClick que aponta para handleClick mostra que setComentarios adiciona o valor presente em input na lista de comentários e logo após isso retorna o valor de input para vazio. O useRef é utilizado para destacar o campo de input como mostrado na figura abaixo:**

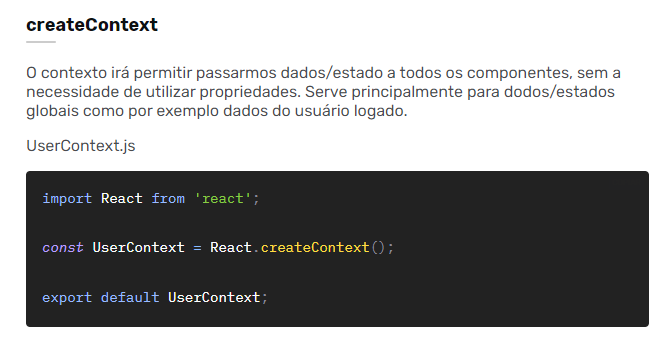
****

* 1. **– UseMemo & UseCallback**

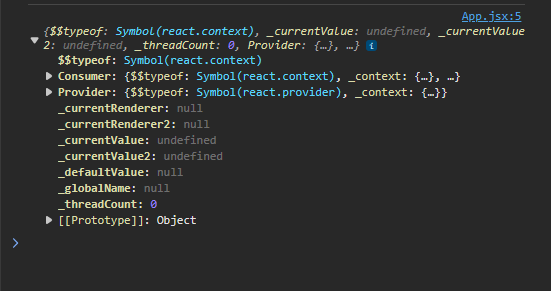
****

**Só executa na criação.**

* 1. **– UseContext**

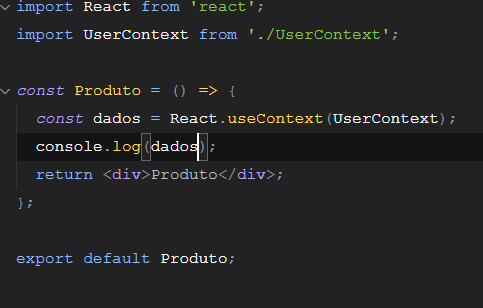


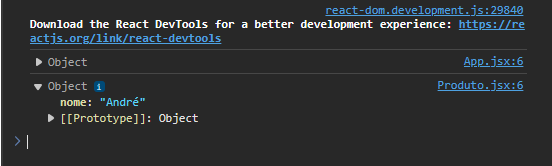
**Usado geralmente na parte mais global do app, ou da parte que você quiser que tenha acesso a esse contexto**

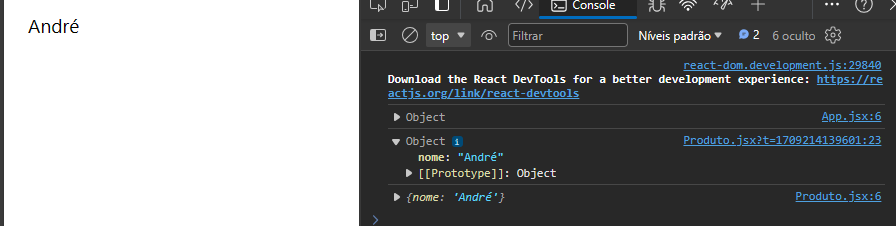
****

**Tem o Consumer e o Provider, sendo o provider o responsável por prover o contexto global para todos os filhos.**

****

****

****

****

* 1. **– Custom Hooks**

****

****