O que são Hooks

**React Hooks, o que são?**

Publicado por [**Cristhian Germano Rodrigues**](https://www.garagemdigital.io/author/cristhian-rodrigues/) em 19/03/2021

O que é um Hook? Hook nada mais é que uma função especial que te permite utilizar recursos do React.

React Hooks foi lançado há algum tempo, e com eles vários problemas foram solucionados, tais como:

– Dificuldade na reutilização de código

– Componentes complexos

– Confusão com classes

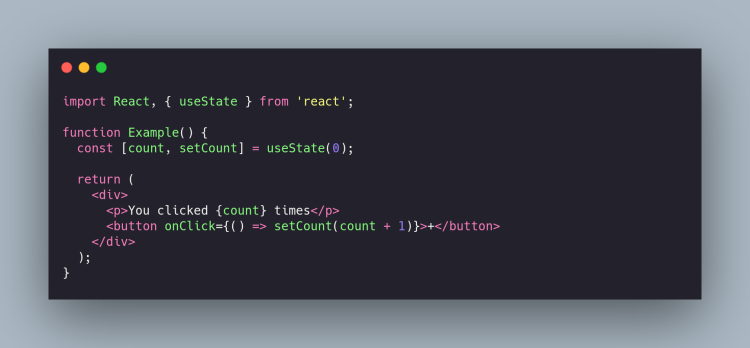
Os hooks permitem que você use o state e outros recursos do React sem escrever uma classe.

Agora vamos listar os basic hooks e mostrar como funcionam.

useState

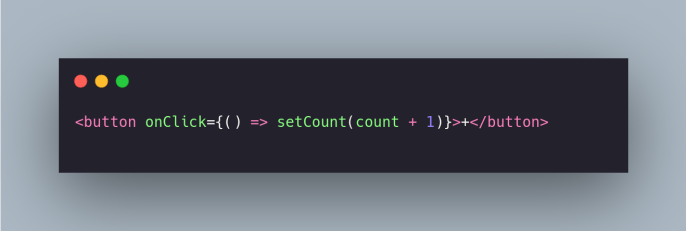
Esse primeiro hook nos permite criar estados dentro de um componente de função. O useState( ) nos retorna um par de valores: o estado atual e uma função que atualiza o estado.

Siga o exemplo abaixo:



O primeiro valor “count” representa o valor do estado que será manipulado pela função “setCount”, recebida através da desestruturação realizada no useState. O valor 0 passado ao hook é o valor inicial do estado.

Para manipularmos o valor do “count”, podemos simplesmente executar:

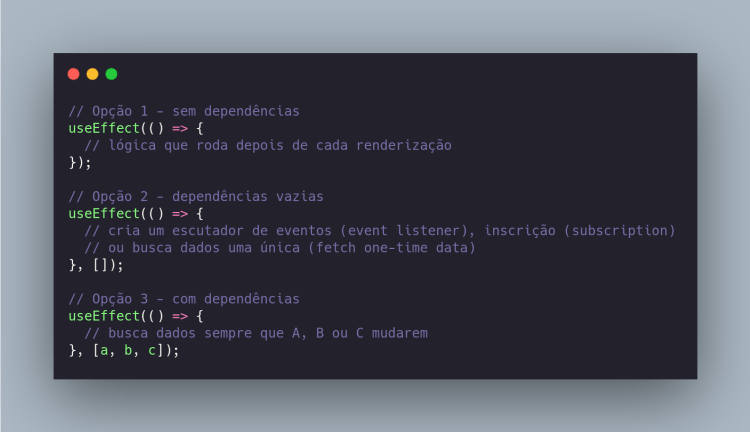


useEffect

O useEffect é um hook de efeito e nos permite executar efeitos colaterais em nossos componentes. Sempre que você está buscando dados, conectando-se a um serviço, assinatura ou manipulando manualmente o DOM, está realizando um efeito colateral.

Um efeito colateral é qualquer coisa que torne a função imprevisível, como dados ou estado. Um hook de efeito sempre é executado após cada renderização, o motivo pra isso acontecer é que os efeitos podem conter uma lógica pesada, e no caso, seria melhor executar após a renderização.

O hook recebe dois argumentos: A função a ser executada e um array com valores que serão avaliados após cada renderização. Esses valores são chamados de dependências.



A lógica de cada opção é:

* **Se não estiver presente,**o efeito será executado após cada renderização. Esta opção não é normalmente usada, mas é útil em algumas situações, como a necessidade de fazer cálculos pesados após cada renderização.
* **Com um array vazio,**[] o efeito é executado apenas uma vez, após a montagem e na primeira renderização. Isso é ótimo para efeitos únicos, como criar um escutador de eventos (*event listener*).
* **Um array com valores**[a, b, c] faz com que o efeito avalie as dependências, sempre que uma dependência mudar o efeito será executado. Isso é útil para executar efeitos quando *props*ou o estado mudam, como buscar novos dados.

É muito importante que você use array de dependências corretamente, incluindo todas as variáveis usadas em useEffect, caso contrário, o efeito fará referência a valores obsoletos de renderizações anteriores durante a execução, causando bugs.

Nova forma de usar Context API

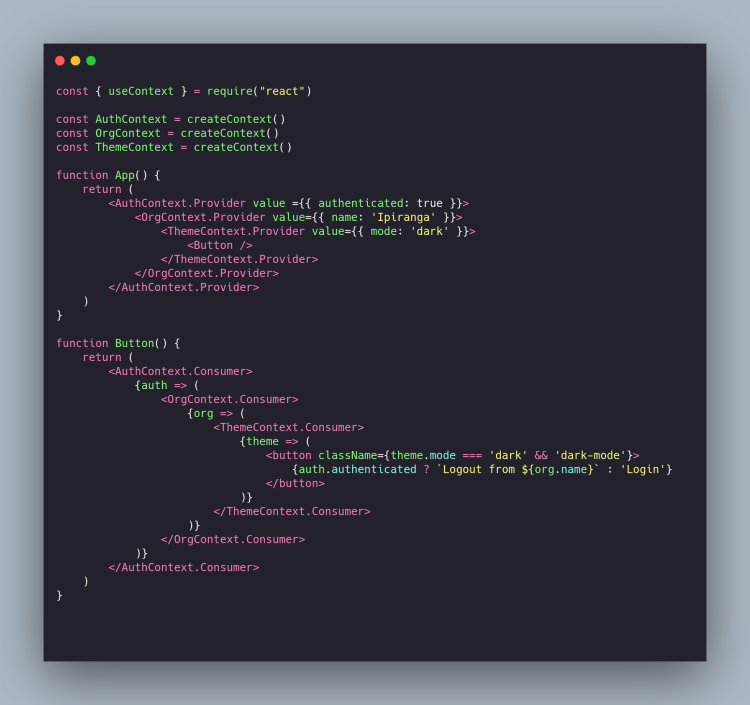
Diga adeus ao render props!

Até então, nós precisávamos usar uma técnica chamada render props dentro de um componente Consumer sempre que quiséssemos acessar valores de um contexto.

Vamos apenas relembrar como era o uso do Context API antes da chegada dos hooks:



Neste exemplo temos um caso simples, o problema aparece mesmo quando precisamos acessar vários contextos ao mesmo tempo, logo, temos um monstrinho assim:



Claro que aqui dei um exemplo que talvez não seja tão convencional, mas que se fosse necessário, teria que ser feito dessa forma. Te convenci que isso de fato era, de fato, um problema?

Então agora vamos reescrever este mesmo código usando o hook useContext:



Muito mais simples, não é?

O primeiro ponto a destacar é que a forma como declaramos os nossos contextos não mudou, ou seja, ainda precisamos da função createContext e do componente Provider por volta de quem vai acessar seus valores.

A mudança mesmo está na forma em que consumimos os contextos: basta invocar a função useContext enviando como único argumento o contexto que queremos pegar informações e o hook retorna todos os valores para gente, sem precisar de render props nem nada do tipo.

Esses são os basic hooks, explorem toda a possibilidade que eles trazem para o desenvolvimento de suas aplicações!