# REACT HOOKS

Segue abaixo um lista com todos os Hooks do React.

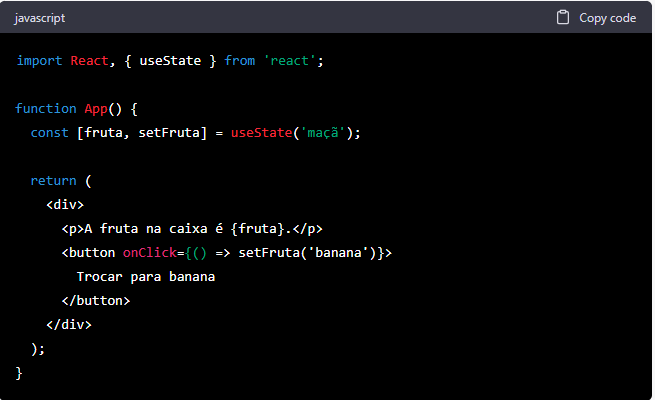
1. useState
2. useEffect
3. useContext
4. useReducer
5. useCallback
6. useMemo
7. useRef
8. useLayoutEffect
9. useImperativeHandle
10. useDebugValue
11. useErrorBoundary
12. useTransition
13. useDeferredValue
14. useMutableSource
15. useOpaqueIdentifier
16. useDebugState
17. useResponder
18. useEvent
19. useRootClose
20. useReducedMotion
21. useSlider
22. useCombobox
23. useMultiSelect
24. useSelectable
25. useRadioGroup
26. useCheckboxGroup
27. useTree
28. useMap
29. useSet
30. useForkRef
31. useIsomorphicLayoutEffect
32. useIntersectionObserver
33. useScrollTrigger
34. useMediaQuery
35. usePrefersReducedMotion
36. usePrefersColorScheme
37. useDocumentVisibility
38. useLocation
39. useParams
40. useHistory
41. useRouteMatch
42. usePrompt

Segue uma descrição da funcionalidadade de cada um deles, com um breve exemplo de como ele funciona.

1 - UseState

Imagine que você tem uma caixa onde pode guardar uma fruta. Com o useState, você pode trocar qual fruta está dentro dessa caixa sempre que quiser. Por exemplo, você pode colocar uma maçã dentro da caixa e depois trocar por uma banana, ou por uma laranja, ou qualquer outra fruta que quiser.

Exemplo:



2 - useEffect

Agora, imagine que você quer fazer uma lista de compras, mas você precisa esperar um pouco para saber quais frutas estão disponíveis na loja. Com o useEffect, você pode dizer para o seu programa esperar um pouco e depois fazer algo, como mostrar a lista de compras na tela.

Exemplo:



3 - useContext

Imagine que você tem vários amigos e cada um deles tem um brinquedo diferente. Com o useContext, você pode compartilhar um brinquedo com todos os seus amigos, para que eles possam brincar juntos.

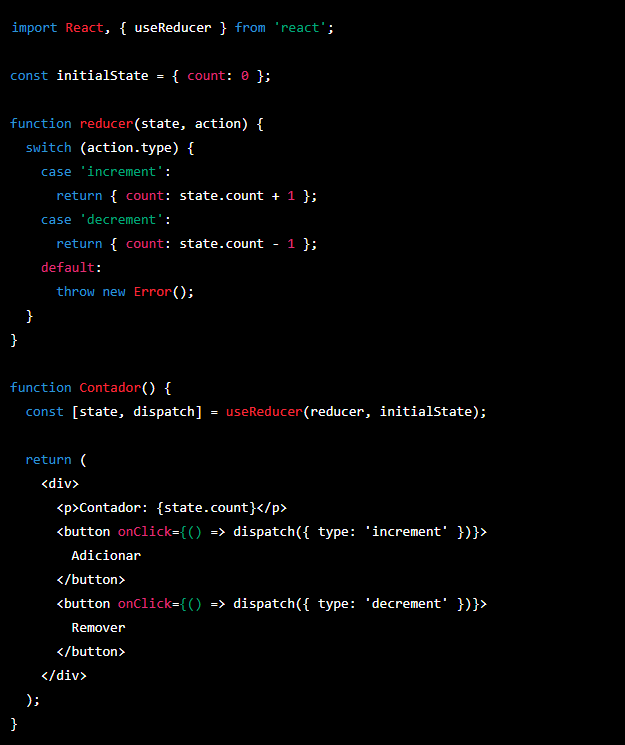
Exemplo:



4 - useReducer

Imagine que você é um chef de cozinha e tem vários ajudantes. Com o useReducer, você pode dizer para cada ajudante o que fazer em cada etapa da receita, como cortar legumes ou misturar ingredientes. Cada ajudante sabe qual é a sua tarefa e trabalha em conjunto para fazer a refeição.

Exemplo:



5 - useCallback

Imagine que você tem vários amigos e quer chamar cada um deles para brincar, mas você só quer ligar para eles uma vez. Com o useCallback, você pode guardar o número de telefone de cada amigo e só ligar para eles quando quiser.

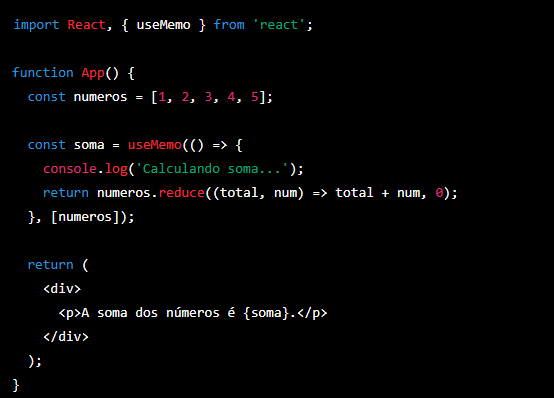
Exemplo:



6 - useMemo

Imagine que você precisa fazer uma conta matemática muito complicada várias vezes. Com o useMemo, você pode guardar o resultado da conta para usá-lo mais tarde, sem precisar fazer a conta de novo toda vez que precisar do resultado.

Exemplo:



7 - useRef

Imagine que você é um jogador de futebol e precisa treinar seus chutes a gol. Com o useRef, você pode guardar a posição do gol para saber onde chutar a bola, mesmo que o gol se mova ou mude de posição.

Exemplo:

**import React, { useRef } from 'react';**

**function Gol() {**

**const golRef = useRef();**

**function chutar() {**

**const gol = golRef.current;**

**console.log(`Chutando na posição (${gol.offsetLeft}, ${gol.offsetTop})...`);**

**}**

**return (**

**<div ref={golRef} style={{ position: 'relative', width: '500px', height: '300px', background: 'green' }}>**

**<button onClick={chutar}>Chutar na trave</button>**

**</div>**

**);**

**}**

8 - useLayoutEffect

Imagine que você é um pintor e precisa pintar uma tela. Com o useLayoutEffect, você pode esperar a tela secar antes de adicionar uma nova camada de tinta. Isso garante que as cores não se misturem e que o resultado final seja bonito e bem feito.



9 - useImperativeHandle

Imagine que você é um mágico e tem vários truques na manga. Com o useImperativeHandle, você pode criar uma caixa mágica que guarda seus truques e só mostra quando você quiser. Assim, você pode impressionar sua plateia com truques incríveis e deixar todo mundo boquiaberto.

Exemplo:

import React, { useRef, useImperativeHandle, forwardRef } from 'react';

const CaixaMagica = forwardRef((props, ref) => {

const caixaRef = useRef();

useImperativeHandle(ref, () => ({

abrir: () => {

console.log('Abrindo a caixa mágica...');

caixaRef.current.style.display = 'block';

},

fechar: () => {

console.log('Fechando a caixa mágica...');

caixaRef.current.style.display = 'none';

},

}));

return (

<div ref={caixaRef} style={{ display: 'none', background: 'purple', width: '200px', height: '200px' }}>

<p>Truques mágicos dentro desta caixa!</p>

</div>

);

});

function App() {

const caixaRef = useRef();

function abrirCaixa() {

caixaRef.current.abrir();

}

function fecharCaixa() {

caixaRef.current.fechar();

}

return (

<div>

<button onClick={abrirCaixa}>Abrir caixa mágica</button>

<button onClick={fecharCaixa}>Fechar caixa mágica</button>

<CaixaMagica ref={caixaRef} />

</div>

);

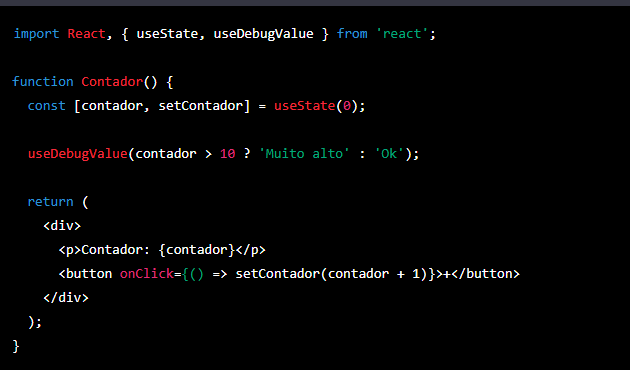
}

10 - useDebugValue

O hook useDebugValue é usado para exibir informações adicionais durante o desenvolvimento de uma aplicação React. Ele permite que você forneça um valor que será exibido na ferramenta de desenvolvedor quando o componente que utiliza o hook for selecionado.

Por exemplo, se você tem um contador em um componente e quer saber se ele está "Muito alto" ou "Ok" quando o valor exceder 10, você pode utilizar o useDebugValue para exibir essa informação na ferramenta de desenvolvedor quando o componente estiver selecionado.

Exemplo:



11 -useErrorBoundary

É um hook utilizado para criar um limite de erro em um componente React. Quando um erro ocorre dentro do componente, ele é capturado e pode ser tratado de forma personalizada. Isso ajuda a melhorar a experiência do usuário e a evitar que a aplicação quebre completamente

Exemplo:



12 - useTransition

É um hook utilizado para criar uma transição suave entre as mudanças de estado do componente. Ele ajuda a evitar atrasos ou quebras durante as atualizações de estado, melhorando a experiência do usuário.

Exemplo:



13 - useDeferredValue

É um hook utilizado para adiar a atualização de um valor em um componente React. Isso ajuda a melhorar o desempenho e a reduzir a carga no navegador, principalmente em componentes com atualizações frequentes.

Exemplo:



1. useMutableSource

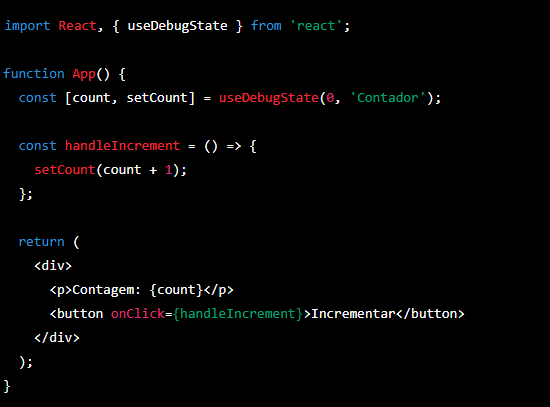
O useMutableSource é utilizado para acessar um objeto fonte mutável, que pode ser atualizado por outro componente ou módulo externo. Abaixo está um exemplo de como utilizá-lo:

Exemplo:



15 - useDebugState

O useDebugState é utilizado para adicionar informações de depuração ao estado de um componente. Ele funciona de maneira semelhante ao useState, mas com a adição de uma mensagem de depuração que é exibida no console. Aqui está um exemplo simples de como utilizá-lo:



16 - useResponder

O useResponder é utilizado para lidar com eventos de toque em dispositivos móveis. Ele permite que um componente seja configurado para responder a gestos específicos, como toque, deslize e rotação. Aqui está um exemplo simples de como utilizá-lo:

import React, { useResponder } from 'react';

function App() {

const handleTouchMove = (event) => {

console.log('Movimento detectado:', event.touches[0].clientX, event.touches[0].clientY);

};

const responder = useResponder({

onMoveShouldSetResponder: () => true,

onMoveShouldSetResponderCapture: () => true,

onMove: handleTouchMove,

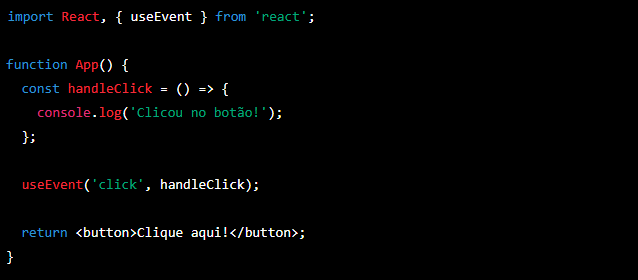
});

return <div {...responder}>Toque e mova o dedo</div>;

}

1. useEvent

O useEvent é utilizado para adicionar e remover ouvintes de eventos de forma declarativa. Ele permite que um componente seja configurado para responder a eventos específicos, como clique, toque e teclado. Aqui está um exemplo simples de como utilizá-lo:



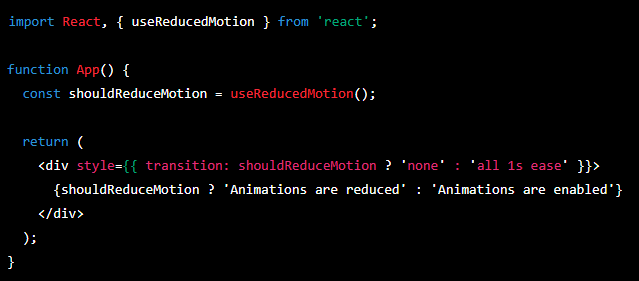
18 - useRootClose

O useRootClose é utilizado para fechar um elemento quando o usuário clica fora dele. Ele permite que um componente seja configurado para fechar automaticamente quando o usuário clica em qualquer lugar da página, exceto no próprio componente. Aqui está um exemplo simples de como utilizá-lo:



19 - useReduceMotion

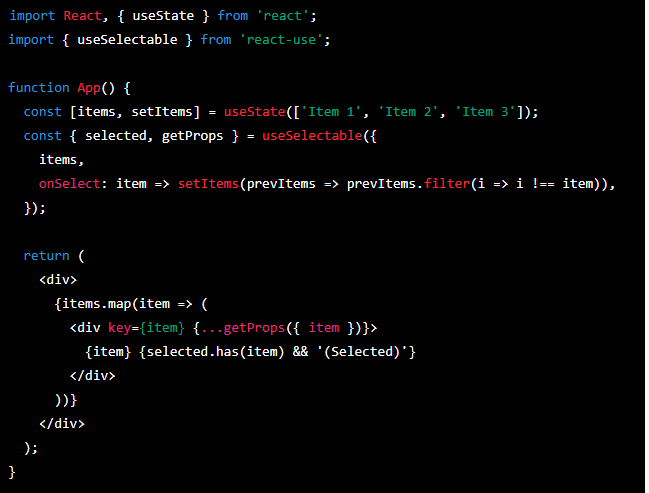
O useReducedMotion é utilizado para detectar se o usuário prefere animações reduzidas. Ele permite que um componente seja configurado para reduzir as animações quando o usuário prefere menos movimento na interface. Aqui está um exemplo simples de como utilizá-lo:



20 -

21 - useSelectable

O useSelectable é um hook que permite a criação de componentes selecionáveis. Com ele é possível definir um estado de seleção, selecionar e deselecionar elementos, além de obter informações sobre os itens selecionados. Aqui está um exemplo simples:



22 - useRadioGroup

O useRadioGroup é um hook que permite criar componentes de seleção de radio buttons. Com ele, é possível definir um estado de seleção, selecionar e deselecionar elementos, além de obter informações sobre o item selecionado. Aqui está um exemplo simples:

import React from 'react';

import { useRadioGroup } from 'react-use';

function App() {

const { value, setValue, getRadioProps } = useRadioGroup({

defaultValue: 'option1',

onChange: e => console.log('Selected option:', e.target.value),

});

return (

<div>

<label {...getRadioProps({ value: 'option1' })}>

Option 1 <input type="radio" value="option1" checked={value === 'option1'} onChange={setValue} />

</label>

<label {...getRadioProps({ value: 'option2' })}>

Option 2 <input type="radio" value="option2" checked={value === 'option2'} onChange={setValue} />

</label>

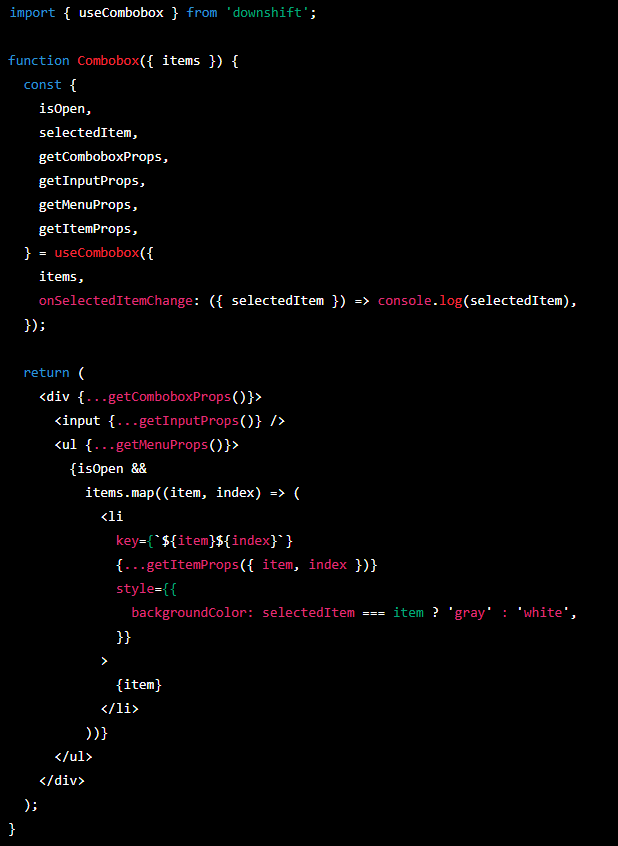
<label {...getRadioProps({ value: 'option

Demais Hooks que eu entendo como importantes seguem abaixo

**useCombobox**

Esse hook é usado para criar um componente de seleção de itens com uma caixa de texto que permite filtrar os itens disponíveis. É útil quando há muitos itens na lista e o usuário precisa de uma maneira rápida de pesquisar ou filtrar.

Exemplo?



useMultiSelect

Esse hook é usado para criar um componente de seleção múltipla que permite selecionar vários itens em uma lista.



useSelectable

Esse hook é usado para criar um componente de seleção de um item em uma lista.



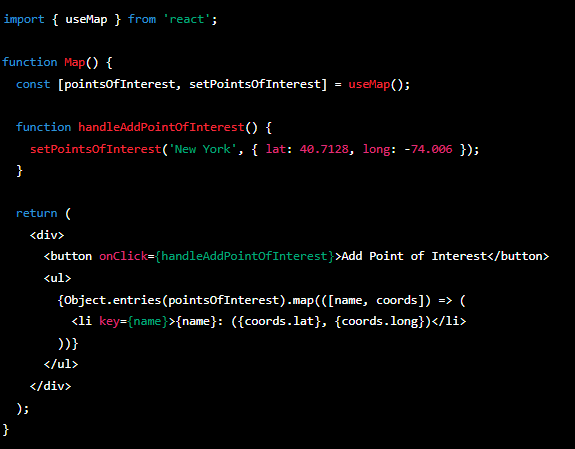
useTree

Esse hook é usado para criar uma árvore de elementos para renderizar dados hierárquicos. Ele é especialmente útil para criar componentes de navegação em menus de árvore e outras estruturas de dados. Um exemplo de uso seria para representar a estrutura de pastas e arquivos em um explorador de arquivos.



useMap

é um hook que permite criar e manipular um mapa (map) de dados em React. É especialmente útil para armazenar dados em pares de chave-valor, onde a chave pode ser qualquer valor único e a valor pode ser qualquer valor válido. Um exemplo prático de uso do hook useMap seria armazenar informações do usuário como nome, idade, endereço e outros detalhes em um mapa.



useSet

é um hook que permite criar e manipular um conjunto (set) de dados em React. É útil para armazenar coleções de dados exclusivos, semelhante a um array, mas sem permitir elementos duplicados. Um exemplo prático de uso do hook useSet seria armazenar uma lista de tags para um artigo em um conjunto.

import { useSet } from 'react';

function SelectableList({ items }) {

const [selectedItems, setSelectedItems] = useSet();

function handleItemClick(item) {

setSelectedItems(prevSelectedItems => new Set([...prevSelectedItems, item]));

}

function handleItemDeselect(item) {

setSelectedItems(prevSelectedItems => new Set([...prevSelectedItems].filter(i => i !== item)));

}

return (

<ul>

{items.map(item => (

<li

key={item}

onClick={() => selectedItems.has(item) ? handleItemDeselect(item) : handleItemClick(item)}

style={{ fontWeight: selectedItems.has(item) ? 'bold' : 'normal' }}

>

{item}

</li>

))}

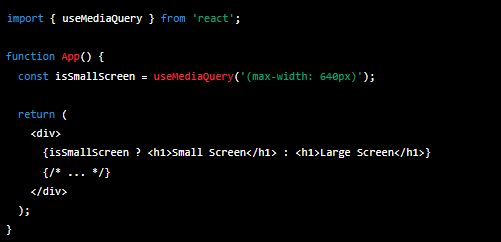
</ul>

);

}

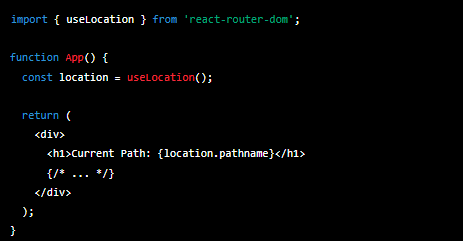
useMediaQuery

é um hook que permite verificar o estado de uma consulta de mídia em React. É útil para criar componentes responsivos que se adaptam a diferentes tamanhos de tela ou condições de exibição. Um exemplo prático de uso do hook useMediaQuery seria exibir um botão de menu de navegação somente quando a tela tiver uma largura inferior a um determinado tamanho.



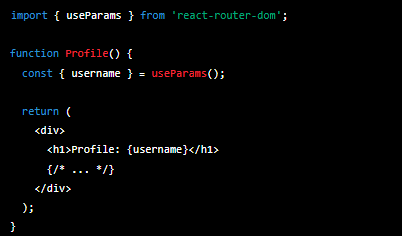
useLocation

é um hook que permite acessar e manipular o objeto de localização atual do navegador em React. É útil para criar componentes que dependem da localização do usuário, como um mapa ou um sistema de navegação. Um exemplo prático de uso do hook useLocation seria acessar as informações de localização do usuário para exibir os eventos ou locais próximos.



useParams

é um hook que permite acessar e manipular os parâmetros de URL em React. É útil para criar componentes que dependem de informações específicas de uma rota, como um identificador de usuário ou um nome de categoria. Um exemplo prático de uso do hook useParams seria exibir informações de um usuário específico com base no ID fornecido na URL.



usePrompt

é um hook que permite exibir uma mensagem de confirmação antes de navegar para outra página em React. É útil para garantir que o usuário não perca nenhum dado ou informações importantes. Um exemplo prático de uso do hook usePrompt seria exibir uma mensagem de confirmação antes de permitir que o usuário saia de um formulário que ainda não foi enviado.



useRouteMatch

é um hook que permite verificar se a URL atual corresponde a uma determinada rota em React. É útil para criar componentes que respondem a rotas específicas, como um componente que só é exibido na página inicial. Um exemplo prático de uso do hook useRouteMatch seria exibir um banner promocional apenas na página inicial do site.

