**Hooks no React e React Native: **

Em React, um "Hook" é uma função especial que permite que componentes funcionais, como peças de LEGO, tenham superpoderes adicionais.

Imagine que você está construindo um robô de LEGO (seu aplicativo React). Para dar ao seu robô habilidades especiais, você usa esses "ganchos" mágicos. Aqui estão alguns dos principais "ganchos" em React:

1. **useState**: Com este Hook, você pode adicionar uma "memória" ao seu robô. Ele pode lembrar informações e usá-las mais tarde. Por exemplo, você pode lembrar a contagem de balas em um jogo.

2. **useEffect**: Este Hook é como um temporizador para o seu robô. Ele diz ao robô para fazer algo em um momento específico, como atualizar uma tela quando um botão é pressionado.

Então, esses Hooks são como ferramentas especiais que você usa para dar ao seu robô (seu aplicativo React) habilidades extras. Eles tornam o desenvolvimento mais rápido, organizado e poderoso, como dar superpoderes ao seu robô LEGO.

Hooks são uma característica do React (e do React Native) que permitem a você adicionar funcionalidades de componentes de classe aos componentes funcionais. Em termos técnicos, eles são funções que permitem que você "ligue-se" ao sistema de gerenciamento de estado e ciclo de vida do React em componentes funcionais.

Novo conceito de Hook:

1. **useState**: Este Hook permite adicionar estado a um componente funcional. Isso significa que você pode criar variáveis que armazenam informações que podem mudar ao longo do tempo. Quando o estado muda, o componente é renderizado novamente.

2. **useEffect**: Este Hook permite que você realize efeitos colaterais em componentes funcionais. Efeitos colaterais são coisas como buscar dados de um servidor, atualizar o título

da página ou se inscrever em eventos. O código dentro de `useEffect` é executado após a renderização do componente.

Exemplo de useState Hook: Contador Simples

Imagine que estamos criando um aplicativo React Native com um contador. Queremos que, quando o usuário toque em um botão, o contador aumente em 1. Usaremos o Hook `useState` para gerenciar o estado do contador.

Aqui está o código do nosso exemplo:

```
import React, { useState } from 'react';
import { View, Text, Button } from 'react-native';

function Contador() {
 // Usamos o useState para criar uma variável de estado 'contador' com valor inicial 0.
 // A função 'setContador' nos permite atualizar o valor do contador.
 const [contador, setContador] = useState(0);

 // Função que incrementa o contador quando o botão é pressionado.
 const incrementarContador = () => {
  setContador(contador + 1); // Atualiza o estado do contador
 };

 return (
  <View>
   <Text>Contagem: {contador}</Text>
   <Button title="Incrementar" onPress={incrementarContador} />
  </View>
 );
```

```
}

export default Contador;
```

Neste exemplo:

1. Importamos o Hook `useState` do React. Ele nos permite criar uma variável de estado chamada `contador` com um valor inicial de 0.

2. Usamos `contador` para exibir o valor atual do contador na interface.

3. Quando o botão é pressionado, chamamos a função `incrementarContador`, que usa `setContador` para atualizar o estado do `contador`. Isso faz com que o componente seja re-renderizado com o novo valor do contador, refletindo a mudança na interface do usuário.

O `useState` é um Hook que nos permite adicionar estado a componentes funcionais. Com ele, podemos gerenciar e atualizar o estado em um componente funcional, tornando-o interativo e responsivo às ações do usuário.

Portanto, este é um exemplo prático de como o Hook `useState` é usado para gerenciar o estado em um componente React Native. Ele é uma ferramenta importante para adicionar funcionalidades dinâmicas aos seus aplicativos.

Em resumo, Hooks são ferramentas poderosas no React e no React Native que tornam o desenvolvimento de aplicativos mais simples, organizado e eficiente, permitindo que você adicione estado, lógica e efeitos colaterais aos seus componentes funcionais. Eles são uma maneira mais moderna e flexível de escrever código em React.

Ex Hook

```
import React, { useState } from 'react';

import { View, Text, Button } from 'react-native';

function Contador() {
```

```
  // Usamos o useState para criar uma variável de estado 'contador' com valor inicial 0.
  // A função 'setContador' nos permite atualizar o valor do contador.
  const [contador, setContador] = useState(0);

  // Função que incrementa o contador quando o botão é pressionado.
  const incrementarContador = () => {
    setContador(contador + 1); // Atualiza o estado do contador
  };

  return (
    <View>
      <Text>Contagem: {contador}</Text>
      <Button title="Incrementar" onPress={incrementarContador} />
    </View>
  );
}

export default Contador;
```

Exemplo 2

```
import React, { useState, useEffect } from 'react';
import { View, Text } from 'react-native';

function Relogio() {
  const [hora, setHora] = useState(new Date());

  // useEffect é usado para realizar efeitos colaterais em componentes funcionais.
  // Neste caso, atualizaremos o estado 'hora' a cada segundo.
  useEffect(() => {
    const intervalo = setInterval(() => {
      setHora(new Date());
```

```
    }, 1000);

    // Retorna uma função de limpeza para parar o intervalo quando o componente é
desmontado.
    return () => {
      clearInterval(intervalo);
    };
  }, []); // O array vazio [] indica que este efeito é executado apenas uma vez na montagem.

  return (
    <View>
      <Text>Horário atual: {hora.toLocaleTimeString()}</Text>
    </View>
  );
}

export default Relogio;
```

Em React e React Native, um "efeito colateral" se refere a qualquer ação que ocorre em um componente que não está diretamente relacionada à renderização da interface do usuário. São operações que ocorrem "ao lado" (ou seja, em segundo plano) da renderização do componente. Alguns exemplos comuns de efeitos colaterais incluem: