Hook API之useMemo与useCallback

课堂目标

1. 掌握useMemo、useCallback

资源

1. Hook API 索引

知识要点

useMemo

把"创建"函数和依赖项数组作为参数传入 `useMemo`，它仅会在某个依赖项改变时才重新计算 memoized 值。这种优化有助于避免在每次渲染时都进行高开销的计算。

```
import React, { useState, useMemo } from "react";

export default function UseMemoPage(props) {
  const [count, setCount] = useState(0);
  const expensive = useMemo(() => {
    console.log("compute");
    let sum = 0;
    for (let i = 0; i < count; i++) {
      sum += i;
    }
    return sum;
    //只有count变化，这里才重新执行
  }, [count]);
  const [value, setValue] = useState("");
  return (
    <div>
      <h3>UseMemoPage</h3>
      <p>expensive:{expensive}</p>
      <p>{count}</p>
      <button onClick={() => setCount(count + 1)}>add</button>
      <input value={value} onChange={event => setValue(event.target.value)} />
    </div>
  );
}
```

useCallback

把内联回调函数及依赖项数组作为参数传入 `useCallback`，它将返回该回调函数的 memoized 版本，该回调函数仅在某个依赖项改变时才会更新。当你把回调函数传递给经过优化的并使用引用相等性去避免非必要渲染（例如 `shouldComponentUpdate`）的子组件时，它将非常有用。

```
import React, { useState, useCallback, PureComponent } from "react";

export default function UseCallbackPage(props) {
  const [count, setCount] = useState(0);
  const addClick = useCallback(() => {
    let sum = 0;
    for (let i = 0; i < count; i++) {
      sum += i;
    }
    return sum;
  }, [count]);
  const [value, setValue] = useState("");
  return (
    <div>
      <h3>UseCallbackPage</h3>
      <p>{count}</p>
      <button onClick={() => setCount(count + 1)}>add</button>
      <input value={value} onChange={event => setValue(event.target.value)} />
      <Child addClick={addClick} />
    </div>
  );
}

class Child extends PureComponent {
  render() {
    console.log("child render");
    const { addClick } = this.props;
    return (
      <div>
        <h3>Child</h3>
        <button onClick={() => console.log(addClick())}>add</button>
      </div>
    );
  }
}
```

`useCallback(fn, deps)` 相当于 `useMemo(() => fn, deps)`。

> 注意
>
> 依赖项数组不会作为参数传给“创建”函数。虽然从概念上来说它表现为：所有“创建”函数中引用的值都应该出现在依赖项数组中。未来编译器会更加智能，届时自动创建数组将成为可能。