认识Hook

课堂目标

1. 掌握hook基本使用

资源

1. Hook简介
2. hook视频介绍：React Today and Tomorrow

知识要点

认识Hook

Hook 是什么？ Hook 是一个特殊的函数，它可以让你“钩入” React 的特性。例如，`useState` 是允许你在 React 函数组件中添加 state 的 Hook。

什么时候我会用 Hook？ 如果你在编写函数组件并意识到需要向其添加一些 state，以前的做法是必须将其它转化为 class。现在你可以在现有的函数组件中使用 Hook。

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

export default function HookPage(props) {
  // 声明一个叫 “count” 的 state 变量，初始化为0
  const [count, setCount] = useState(0);
  return (
    <div>
      <h3>HookPage</h3>
      <p>{count}</p>
      <button onClick={() => setCount(count + 1)}>add</button>
    </div>
  );
}
```

使用 Effect Hook

Effect Hook 可以让你在函数组件中执行副作用操作。

数据获取，设置订阅以及手动更改 React 组件中的 DOM 都属于副作用。不管你知不知道这些操作，或是“副作用”这个名字，应该都在组件中使用过它们。

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

export default function HookPage(props) {
  // 声明一个叫 “count” 的 state 变量，初始化为0
  const [count, setCount] = useState(0);
  // 与 componentDidMount 和 componentDidUpdate相似
  useEffect(() => {
    // 更新 title
    document.title = `You clicked ${count} times`;
  });

  return (
    <div>
      <h3>HookPage</h3>
      <p>{count}</p>
      <button onClick={() => setCount(count + 1)}>add</button>
    </div>
  );
}
```

在函数组件主体内（这里指在 React 渲染阶段）改变 DOM、添加订阅、设置定时器、记录日志以及执行其他包含副作用的操作都是不被允许的，因为这可能会产生莫名其妙的 bug 并破坏 UI 的一致性。

使用 `useEffect` 完成副作用操作。赋值给 `useEffect` 的函数会在组件渲染到屏幕之后执行。你可以把 effect 看作从 React 的纯函数式世界通往命令式世界的逃生通道。

默认情况下，effect 将在每轮渲染结束后执行，但你可以选择让它 在只有某些值改变的时候 才执行。

effect 的条件执行

默认情况下，effect 会在每轮组件渲染完成后执行。这样的话，一旦 effect 的依赖发生变化，它就会被重新创建。

然而，在某些场景下这么做可能会矫枉过正。比如，在上一章节的订阅示例中，我们不需要在每次组件更新时都创建新的订阅，而是仅需要在 `source` props 改变时重新创建。

要实现这一点，可以给 `useEffect` 传递第二个参数，它是 effect 所依赖的值数组。更新后的示例如下：

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

export default function HookPage(props) {
  // 声明一个叫 “count” 的 state 变量，初始化为0
  const [count, setCount] = useState(0);
  const [date, setDate] = useState(new Date());

  // 与 componentDidMount 和 componentDidUpdate相似
  useEffect(() => {
    // 更新 title
    document.title = `You clicked ${count} times`;
  }, [count]);

  useEffect(() => {
    const timer = setInterval(() => {
      setDate(new Date());
    }, 1000);
  }, []);

  return (
    <div>
      <h3>HookPage</h3>
      <p>{count}</p>
      <button onClick={() => setCount(count + 1)}>add</button>
      <p>{date.toLocaleTimeString()}</p>
    </div>
  );
}
```

此时，只有当 useEffect第二个参数数组里的数值 改变后才会重新创建订阅。

清除 effect

通常，组件卸载时需要清除 effect 创建的诸如订阅或计时器 ID 等资源。要实现这一点，`useEffect` 函数需返回一个清除函数，以防止内存泄漏，清除函数会在组件卸载前执行。

```
useEffect(() => {
  const timer = setInterval(() => {
    setDate(new Date());
  }, 1000);
  return () => clearInterval(timer);
}, []);
```