自定义Hook与Hook使用规则

课堂目标

1. 掌握自定义Hook与Hook使用规则

资源

1. 自定义Hook
2. Hook规则

知识要点

自定义Hook

有时候我们会想要在组件之间重用一些状态逻辑。目前为止，有两种主流方案来解决这个问题：高阶组件和 render props。自定义 Hook 可以让你在不增加组件的情况下达到同样的目的。

自定义 Hook 是一个函数，其名称以 “use” 开头，函数内部可以调用其他的 Hook。

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

export default function CustomHookPage(props) {
  //定义一个叫count的state变量，初始化为0
  const [count, setCount] = useState(0);
  //和didMount、didUpdate类似
  useEffect(() => {
    console.log("count effect");
    // 只需要在count发生改变的时候执行就可以啦
    document.title = `点击了${count}次`;
  }, [count]);

  return (
    <div>
```

```
      <h3>自定义Hook</h3>
      <p>{count}</p>
      <button onClick={() => setCount(count + 1)}>add</button>
      <p>{useClock().toLocaleTimeString()}</p>
    </div>
  );
}

//自定义hook，命名必须以use开头
function useClock() {
  const [date, setDate] = useState(new Date());
  useEffect(() => {
    console.log("date effect");
    //只需要在didMount时候执行就可以了
    const timer = setInterval(() => {
      setDate(new Date());
    }, 1000);
    //清除定时器，类似willUnmount
    return () => clearInterval(timer);
  }, []);
  return date;
}
```

✌️ Hook 使用规则

Hook 就是 JavaScript 函数，但是使用它们会有两个额外的规则：

- 只能在**函数最外层**调用 Hook。不要在循环、条件判断或者子函数中调用。
- 只能在 **React 的函数组件**中调用 Hook。不要在其他 JavaScript 函数中调用。（还有一个地方可以调用 Hook —— 就是自定义的 Hook 中。）