title: React Hooks 调研
date: 2019-06-04
tags: React, Hooks

description: React Hooks 使用函数式（FC）进编码。

Hooks 是为了解决 React 目前存在的一些问题：

- 有状态的逻辑很难在组件之间重用
- React 没有提供一个有效的可重用方法。render props、HOC 模式的引入会导致重构组件，使得问题变得麻烦，组件难以维护。
- 复杂组件难以阅读和理解
- 类（Classes）让人困惑、影响机器运行效率

强依赖

react-scripts

CRA 升级 — react-scripts 3.0.0 以上。CHANGELOG

react-scripts 3.0.0 新增特性：

- Jest 24
- Hooks support — 强制执行 Hooks 规则 eslint-plugin-react-hooks

API

useState

```
import React, { useState } from 'react';
export default function Example() {
  const [ count, setCount ] = useState(0);
  const addCounter = () => {
    const newValue = count + 1;
    setCount(newValue);
  };
  return (
    <div>
      <p>You clicked {count} times</p>
      <button onClick={addCounter}>Click me</button>
    </div>
  )
}
```

useEffect

```
useEffect(() => {
  document.title = `You clicked ${count} times`;
}, [count]); // Only re-run the effect if count changes
// 等价于
componentDidUpdate(prevProps, prevState) {
  if (prevState.count !== this.state.count) {
    document.title = `You clicked ${this.state.count} times`;
  }
}
```

限制

Hooks 的使用要遵循固定的规则 https://www.npmjs.com/package/eslint-plugin-react-hooks CRA 升级 react-scripts 到 3.0.0 默认遵守这个规则

实战

封装 useInterval Hook

```
// Hooks && setInterval
// https://overreacted.io/zh-hans/making-setinterval-declarative-with-react-hooks/
import React, { useState, useEffect, useRef } from 'react';
export default function Counter() {
  const [ count, setCount ] = useState(0);
  /**
   * 尝试一
   * React 会在每次渲染后重执行 effects
   * setInterval 执行时会进入到时间队列里，如果频繁的更新 effects，interval 有可能没有机会被执行。
   * 比如给 Counter 外层包装一层定时器，没 100 ms 执行一次，这时候 effects 内部的 interval 不会被执行
  */
  // useEffect(() => {
  //   let id = setInterval(() => {
  //     setCount(count + 1);
  //   }, 10000);
  //   return () => clearInterval(id);
  // });
  /**
   * 尝试二
   * useEffect() 允许“选择性”执行 effects
   * 设定一个依赖数组作为第二个参数， React 只会在数值变化时运行
   * mount 执行 effect，unmount 时清理，传入 []
   * 效果：计时器更新到 1 就不变了
   * 原因：effect 仅执行一次渲染 count 为 0， setInterval 一直引用闭包 count 为 0 。。。
   * 修复方法：
   * 一： setCount(c => c + 1) update 可以更新 state 变量。当时无法更新 props
   * 二： useReducer()
   */
  // useEffect(() => {
  //   let id = setInterval(() => {
  //     console.log('count', count);
  //     setCount(count + 1);
  //   }, 1000);
  //   return () => clearInterval(id);
  // }, []);

  /**
   * useRef() 返回一个有带有 current 可变属性的普通对象在 renders 间共享，可以保存新的 interval 回调给它：
  */
  useInterval(() => {
    setCount(count + 1);
  }, 1000);
  // 封装 Hook
  function useInterval(callback, delay) {
    const savedCallback = useRef();
    useEffect(() => {
      savedCallback.current = callback;
    });
    useEffect(() => {
      function tick() {
        savedCallback.current();
      }
      if (delay !== null) {
        let id = setInterval(tick, delay);
        return () => clearInterval(id);
      }
    }, [delay]);
  }
  return <h2>{count}</h2>;
}
```

思考

- Hooks 的使用还是处于探索阶段，使用技巧并不成熟
- 对待 Hooks，建议是在新项目中尝试使用

参考文档

- https://medium.com/javascript-in-plain-english/state-management-with-react-hooks-no-redux-or-context-api-8b3035ceecf8
- https://www.netlify.com/blog/2019/03/11/deep-dive-how-do-react-hooks-really-work/