简易版的React hook实现

React Hooks原理

```
let memorizedHookValue = []; // hooks 存放在这个数组，每use一个hook，就把hook内要存储的数据放在数组
let cursor = 0; // 当前 memorizedHookValue 下标

function useState(initialValue) {
    memorizedHookValue[cursor] = memorizedHookValue[cursor] || initialValue;
    const currentCursor = cursor;
    function setState(newState) {
        memorizedHookValue[currentCursor] = newState;
        render();
    }
    return [memorizedHookValue[cursor++], setState]; // 返回当前 state，并把 cursor 加 1
}

function useEffect(callback, depArray) {
    const hasNoDeps = !depArray;
    const deps = memorizedHookValue[cursor];
    const hasChangedDeps = deps
        ? !depArray.every((el, i) => el === deps[i]) // 依次判断本次传入的dependencies内的依赖值是
        : true;
    if (hasNoDeps || hasChangedDeps) {
        callback();
        memorizedHookValue[cursor] = depArray;
    }
    cursor++;
}
```

memorizedHookValue：用来记录hook要存储的值的数组，对于state hook来说，要存储的是状态的值; 对于effect hook来说，要存储的是这个effect的依赖的值
cursor：记录在组件内部，运行到了第几个hook。每use一个hook，都会给cursor加一，hook内部通过这个cursor值，去memorizedHookValue中拿到存储的值来使用

useState内做的事情：

- 从memorizedHookValue[cursor]中取出当前state的值
- 声明用来改变状态的函数（set函数），该函数内部的逻辑就是改变memorizedHookValue[cursor]的值，并引发组件重渲染
- 将cursor加一，并返回当前state的值和对应的set函数

useEffect内做的事情：

- 从memorizedHookValue[cursor]中取出上一次的依赖值

- 判断上一次的依赖值和本次传入的依赖值是否相等
  - 如果依赖值有改变或者压根没有依赖数组传入，则执行callback，并将memorizedHookValue[cursor]置为当前传入的依赖值
  - 否则不执行callback(因此，当依赖值是一个空数组的时候，效果相当于componentDidMount)
- 将cursor加一

Q&A

- **Q:** 为什么只能在函数最外层调用Hook？为什么不要在循环、条件判断或者子函数中调用。
  A: memorizedHookValue 数组是按 hook定义的顺序来放置数据的，如果 hook 顺序变化，memorizedHookValue 并不会感知到。
- **Q:** 为什么useEffect第二个参数是空数组，相当于 componentDidMount ？
  A: 因为依赖一直不变化，callback 不会二次执行。
- **Q:** 在React代码中，所谓的memorizedHookValue， cursor存在哪里？
- **A:** React 中是通过类似单链表的形式来代替memorizedHookValue数组。通过 next指针按顺序串联所有的 hook。在渲染时，react 会生成一棵组件树（或Fiber链表），树中每个节点对应了一个组件，hooks 的数据就作为组件的一个信息，存储在这些节点上，伴随组件一起出生，一起死亡。