🛴5.React Hooks 原理

5-1.内部实现

> useState 其实是一个简版的 useReducer，因为 Redux 的作者入职了 React 以后，将原有的 Redux 的核心理论移植给了 Hooks

5-2.简版的实现

```
let state

function useState(defaultState) {
  function setState(newState) {
    state = newState
  }

  if (!state) {
    state = defaultState
  }
  return [state, setState]
}

function render() {
  const [state, setState] = useState(0)
  console.log(state)
  setState(state + 1)
}
```

状态堆与上下文栈存储多个状态多个函数

```
let contextStack = []

function useState(defaultState) {
  const context = contextStack[contextStack.length - 1]
  const nu = context.nu++
  const { states } = context

  function setState(newState) {
    states[nu] = newState
  }

  if (!states[nu]) {
    states[nu] = defaultState
  }
```

```
  return [states[nu], setState]
}

function withState(func) {
  const states = {}
  return (...args) => {
    contextStack.push({ nu: 0, states })
    const result = func(...args)
    contextStack.pop()
    return result
  }
}

const render = withState(
  function render() {
    const [state, setState] = useState(0)

    render1()

    console.log('render', state)
    setState(state + 1)
  }
)

const render1 = withState(
  function render1() {
    const [state, setState] = useState(0)

    console.log('render1', state)
    setState(state + 2)
  }
)
```

我们再来看看useReducer

```
let memoizedState;
function useReducer(reducer, initialArg, init) {
  let initState = void 0;
  if (typeof init !== 'undefined') {
    initState = init(initialArg)
  } else {
    initState = initialArg
  }
  function dispatch(action) {
    memoizedState = reducer(memoizedState, action)
    render()
  }
  memoizedState = memoizedState || initState
  return [memoizedState, dispatch]
```

```
}

function useState(initState) {
  return useReducer((oldState, newState) => newState, initState)
}
```

志佳老师@2019