珠峰架构公开课 React Hooks（微信 zhufengjiagou）

1. React Hooks

- Hook 是 React 16.8 的新增特性,它可以让你在不编写 class 的情况下使用 state 以及其他的 React 特性
- 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

2. 注意事项

- 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

3. 搭建项目

```
npx create-react-app zhufeng_hooks
cd zhufeng_hooks
yarn start
```

4. useState

- useState 就是一个 Hook
- 通过在函数组件里调用它来给组件添加一些内部 state,React 会在重复渲染时保留这个 state
- useState 会返回一对值：当前状态和一个让你更新它的函数，你可以在事件处理函数中或其他一些地方调用这个函数。它类似 class 组件的 this.setState，但是它不会把新的 state 和旧的 state 进行合并
- useState 唯一的参数就是初始 state
- 返回一个 state，以及更新 state 的函数
    - 在初始渲染期间，返回的状态 (state) 与传入的第一个参数 (initialState) 值相同
    - setState 函数用于更新 state。它接收一个新的 state 值并将组件的一次重新渲染加入队列

`const [state, setState] = useState(initialState);`

4.1 使用useState

src\index.js

```
import React,{useState} from 'react';
import ReactDOM from 'react-dom';
function Counter(){
    const [number,setNumber] = useState(0);
    return (
        <>
            <p>{number}</p>
            <button onClick={()=>setNumber(number+1)}>+</button>
        </>
    )
```

```
}
function render(){
    ReactDOM.render(<Counter/>,document.getElementById('root'));
}
render();
```

4.2 实现useState

src\index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
let memoizedState;
function useState(initialState){
    memoizedState = memoizedState||initialState;
    function setState(newState){
       memoizedState = newState;
       render();
    }
    return [memoizedState,setState];
}

function Counter(){
    const [number,setNumber] = useState(0);
    return (
        <>
            <p>{number}</p>
            <button onClick={()=>setNumber(number+1)}>+</button>
        </>
    )
}
function render(){
    ReactDOM.render(<Counter/>,document.getElementById('root'));
}
render();
```

5. useReducer

- useState 的内部实现
- 它接收一个形如 `(state, action) => newState` 的 reducer，并返回当前的 state 以及与其配套的 dispatch 方法
- 在某些场景下，useReducer 会比 useState 更适用，例如 state 逻辑较复杂且包含多个子值，或者下一个 state 依赖于之前的 state 等

5.1 使用useReducer

```
import React,{useReducer} from 'react';
import ReactDOM from 'react-dom';
const initialArg = 0;

function reducer(state, action) {
  switch (action.type) {
    case 'increment':
      return {number: state.number + 1};
    case 'decrement':
```

```
      return {number: state.number - 1};
    default:
      throw new Error();
  }
}
function init(initialArg){
    return {number:initialArg};
}
function Counter(){
    debugger;
    const [state, dispatch] = useReducer(reducer, initialArg,init);
    return (
        <>
          Count: {state.number}
          <button onClick={() => dispatch({type: 'increment'})}>+</button>
          <button onClick={() => dispatch({type: 'decrement'})}>-</button>
        </>
    )
}
function render(){
    ReactDOM.render(<Counter/>,document.getElementById('root'));
}
render();
```

5.2 实现useReducer

```
import React from 'react';
import ReactDOM from 'react-dom';
let memoizedState ;
function useReducer(reducer, initialArg,init){
     var initialState = void 0;
      if (init !== undefined) {
        initialState = init(initialArg);
      } else {
        initialState = initialArg;
      }
      function dispatch(action){
          memoizedState = reducer(memoizedState,action);
          render();
      }
      memoizedState =  memoizedState||initialState;
      return  [memoizedState, dispatch];
}
const initialArg = 0;

function reducer(state, action) {
  switch (action.type) {
    case 'increment':
      return {number: state.number + 1};
    case 'decrement':
      return {number: state.number - 1};
    default:
      throw new Error();
  }
}
function init(initialArg){
    return {number:initialArg};
```

```
}
function Counter(){
    debugger;
    const [state, dispatch] = useReducer(reducer, initialArg,init);
    return (
        <>
          Count: {state.number}
          <button onClick={() => dispatch({type: 'increment'})}>+</button>
          <button onClick={() => dispatch({type: 'decrement'})}>-</button>
        </>
    )
}
function render(){
    ReactDOM.render(<Counter/>,document.getElementById('root'));
}
render();
```

5.3 useReducer实现useState

```
import React from 'react';
import ReactDOM from 'react-dom';
let memoizedState;
function useReducer(reducer, initialArg,init){
     var initialState = void 0;
      if (init !== undefined) {
        initialState = init(initialArg);
      } else {
        initialState = initialArg;
      }
      function dispatch(action){
          memoizedState = reducer(memoizedState,action);
          render();
      }
      memoizedState =  memoizedState||initialState;
      return  [memoizedState, dispatch];
}
function useState(initialState){
    return  useReducer((oldState, newState)=>newState, initialState);
}

function Counter(){
    const [number,setNumber] = useState(0);
    return (
        <>
            <p>{number}</p>
            <button onClick={()=>setNumber(number+1)}>+</button>
        </>
    )
}
function render(){
    ReactDOM.render(<Counter/>,document.getElementById('root'));
}
render();
```

6. 多个useState

6.1 使用

```
import React,{useState} from 'react';
import ReactDOM from 'react-dom';


function Counter(){
    const [name,setName] = useState('计数器');
    const [number,setNumber] = useState(0);
    return (
        <>
            <p>{name}:{number}</p>
            <button onClick={()=>setName('计数器'+Date.now())}>修改名称</button>
            <button onClick={()=>setNumber(number+1)}>+</button>
        </>
    )
}
function render(){
    ReactDOM.render(<Counter/>,document.getElementById('root'));
}
render();
```

6.2 实现

```
import React from 'react';
import ReactDOM from 'react-dom';

let memoizedStates = [];
let index = 0;
function useState(initState){
    memoizedStates[index]=memoizedStates[index]||initState;
    const currentIndex = index;
    function setState(newState){
        memoizedStates[currentIndex] = newState;
        render();
    }
    return [memoizedStates[index++],setState];
}


function Counter(){
    const [name,setName] = useState('计数器');
    const [number,setNumber] = useState(0);
    return (
        <>
            <p>{name}:{number}</p>
            <button onClick={()=>setName('计数器'+Date.now())}>修改名称</button>
            <button onClick={()=>setNumber(number+1)}>+</button>
        </>
    )
}
function render(){
    index = 0;
    ReactDOM.render(<Counter/>,document.getElementById('root'));
}
render();
```

7. useEffect

- 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
- useEffect 就是一个 Effect Hook，给函数组件增加了操作副作用的能力。它跟 class 组件中的 componentDidMount、componentDidUpdate 和 componentWillUnmount 具有相同的用途，只不过被合并成了一个 API

`useEffect(didUpdate);`

7.1 使用useEffect

```
import React,{useState,useEffect} from 'react';
import ReactDOM from 'react-dom';

function Counter(){
    const [name,setName] = useState('计数器');
    const [number,setNumber] = useState(0);
    useEffect(() => {
        console.log(number);
     }, [number]);
    return (
        <>
            <p>{name}:{number}</p>
             <button onClick={()=>setName('计数器'+Date.now())}>修改名称</button>
            <button onClick={()=>setNumber(number+1)}>+</button>
        </>
    )
}
function render(){
    ReactDOM.render(<Counter/>,document.getElementById('root'));
}
render();
```

7.2 实现useEffect

```
import React,{useState} from 'react';
import ReactDOM from 'react-dom';
let lastDependencies;
function useEffect(callback,dependencies){
  if(!dependencies) return callback();
  let changed =
lastDependencies?!dependencies.every((item,index)=>item===lastDependencies[index
]):true;
  if(changed){
    callback();
    lastDependencies=dependencies;
  }
}
function Counter(){
    const [name,setName] = useState('计数器');
    const [number,setNumber] = useState(0);
    useEffect(() => {
```

```
        console.log(number);
     }, [number]);
    return (
        <>
            <p>{name}:{number}</p>
             <button onClick={()=>setName('计数器'+Date.now())}>修改名称</button>
            <button onClick={()=>setNumber(number+1)}>+</button>
        </>
    )
}
function render(){
    ReactDOM.render(<Counter/>,document.getElementById('root'));
}
render();
```

7.3 useState+useEffect

```
import React from 'react';
import ReactDOM from 'react-dom';
let memoizedStates = [];
let index = 0;
function useState(initState){
    memoizedStates[index]=memoizedStates[index]||initState;
    const currentIndex = index;
    function setState(newState){
        debugger;
        memoizedStates[currentIndex] = newState;
        render();
    }
    return [memoizedStates[index++],setState];
}

function useEffect(callback,dependencies){
  if(!dependencies) {
      index++;
      return callback();
  }
  const lastDependencies = memoizedStates[index];
  let changed =
lastDependencies?!dependencies.every((item,index)=>item===lastDependencies[index
]):true;
  if(changed){
    callback();
    memoizedStates[index]=dependencies;
  }
  index++;
}
function Counter(){
    const [name,setName] = useState('计数器');
    const [number,setNumber] = useState(0);
    useEffect(() => {
        console.log(number);
     }, [number]);
    return (
        <>
            <p>{name}:{number}</p>
             <button onClick={()=>setName('计数器'+Date.now())}>修改名称</button>
```

```
            <button onClick={()=>setNumber(number+1)}>+</button>
        </>
    )
}
function render(){
    index = 0;
    ReactDOM.render(<Counter/>,document.getElementById('root'));
}
render();
```

8. useState源码中的链表实现

```
import React from 'react';
import ReactDOM from 'react-dom';

let firstWorkInProgressHook={memoizedState: null,next: null};
let workInProgressHook=firstWorkInProgressHook;

function useState(initState){
    let currentHook = workInProgressHook.next?workInProgressHook.next:
{memoizedState: initState,next: null};
    function setState(newState){
        currentHook.memoizedState = newState;
        render();
    }
    if(workInProgressHook.next){
        workInProgressHook = workInProgressHook.next;
    }else{
        workInProgressHook.next = currentHook;
        workInProgressHook = currentHook;
    }
    return [currentHook.memoizedState,setState];
}

function Counter(){
    const [name,setName] = useState('计数器');
    const [number,setNumber] = useState(0);
    return (
        <>
            <p>{name}:{number}</p>
            <button onClick={()=>setName('新计数器'+Date.now())}>新计数器</button>
            <button onClick={()=>setNumber(number+1)}>+</button>
        </>
    )
}
function render(){
    workInProgressHook = firstWorkInProgressHook;
    ReactDOM.render(<Counter/>,document.getElementById('root'));
}
render();
```