link: null
title: 珠峰架构师成长计划
description: null
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=39 sentences=91, words=676

1. 共享可变状态

```
let objA = { name: 'zhufeng' };
let objB = objA;
objB.name = 'jiagou';
console.log(objA.name);
```

2. 解决方案

- 深度拷贝
- immutable-js (https://github.com/facebook/immutable-js)
- immutable (http://www.zhufengpeixun.com/grow/html/62.6.react-immutable.html)

3.immer

- immer (https://github.com/immerjs/immer) 是 mobx 的作者写的一个 immutable 库
- 核心实现是利用 ES6 的 proxy,几乎以最小的成本实现了 js 的不可变数据结构

```
cnpm i --save immer
```

4.produce

- 对 draftState 的修改都会反应到 nextState 上
- 而 Immer 使用的结构是共享的，nextState 在结构上又与 currentState 共享未修改的部分

4.1 基本使用

```
import { produce } from 'immer';
let baseState = {}

let nextState = produce(baseState, (draft) => {

})
console.log(baseState===nextState);
```

```
import { produce } from 'immer';
let baseState = {
  ids: [1],
  pos: {
    x: 1,
    y: 1
  }
}

let nextState = produce(baseState, (draft) => {
  draft.ids.push(2);
})
console.log(baseState.ids === nextState.ids);
console.log(baseState.pos === nextState.pos);
```

```
import { produce } from 'immer'

const baseState = {
  list: ['1', '2']
}
const result = produce(baseState, (draft) => {
  draft.list.push('3')
})
console.log(baseState);
console.log(result);
```

4.2 实现

4.2.1 immer\index.js

src\immer\index.js

```
export {produce} from './core';
```

4.2.2 immer\is.js

src\immer\is.js

```
export const isObject = (val) => Object.prototype.toString.call(val) === '[object Object]';
export const isArray = (val) => Array.isArray(val);
export const isFunction = (val) => typeof val === 'function';
```

4.2.3 src\immer\core.js

src\immer\core.js

```
import * as is  from './is';
export  const INTERNAL = Symbol('INTERNAL');
export function produce(baseState, producer) {
    const proxy = toProxy(baseState);
    producer(proxy);
    const internal = proxy[INTERNAL];debugger
    return internal.mutated ? internal.draftState : baseState;
}

export function toProxy(baseState, valueChange) {
    let keyToProxy = {};
    let internal = {
        draftState: createDraftState(baseState),
        keyToProxy,
        mutated: false
    }
    return new Proxy(baseState, {
        get(target, key) {
            if (key === INTERNAL) {
                return internal;
            }
            const value = target[key];
            if (is.isObject(value) || is.isArray(value)) {
                if (key in keyToProxy) {
                    return keyToProxy[key];
                } else {
                    keyToProxy[key] = toProxy(value, () => {
                        internal.mutated = true;
                        const proxyOfChild = keyToProxy[key];
                        const { draftState } = proxyOfChild[INTERNAL];
                        internal.draftState[key] = draftState;
                        valueChange && valueChange();
                    })
                    return keyToProxy[key];
                }
            } else if (is.isFunction(value)) {
                internal.mutated = true;
                valueChange && valueChange();
                return value.bind(internal.draftState);
            }
            return internal.mutated ? internal.draftState[key] : baseState[key];
        },
        set(target, key, value) {
            internal.mutated = true;
            let {draftState}= internal;
            for (const key in target) {
                draftState[key] = key in draftState ? draftState[key] : target[key];
            }
            draftState[key] = value;
            valueChange && valueChange();
            return true;
        }
    });
    function createDraftState(baseState) {
        if (is.isArray(baseState)) {
            return [...baseState];
        } else if (is.isObject(baseState)) {
            return Object.assign({}, baseState);
        } else {
            return baseState;
        }
    }
}
```

5.useImmerState.js

5.1 基本使用

```
import React from 'react';
import ReactDOM from 'react-dom';
import {useImmerState} from './immer'
let id = 1;
function Todos() {
  const [todos, setTodos] = useImmerState({
    list: []
  })
  const addTodo = () => setTodos((draft) => {
    draft.list.push(id++)
  })
  return (
    <>
      <button onClick={addTodo}>增加button>
      <ul>
        {
          todos.list.map((item, index) => <li key={index}>{item}li>)
        }
      ul>
    </>
  )
}
ReactDOM.render(
  <Todos />,
  document.getElementById('root')
);
```

5.2 useImmerState.js

src\immer\useImmerState.js

```
import { useState, useRef } from 'react';
import { toProxy, INTERNAL } from './core';
import * as is from './is';
function useImmerState(baseState) {
    const [state, setState] = useState(baseState);
    const draftRef = useRef(toProxy(baseState, () => {
        queueMicrotask(()=>{
            const internalState = draftRef.current[INTERNAL];
            const newState = internalState.draftState;
            setState(() => {
                return (is.isArray(newState) ? [...newState] : Object.assign({}, newState));
            });
        })
    }));
    const updateDraft = (producer) => producer(draftRef.current);
    return [state, updateDraft];
}
export default useImmerState;
```

5.3 immer\index.js

src\immer\index.js

```
export {produce} from './core';
+export {default as useImmerState} from './useImmerState';
```