link: null
title: 珠峰架构师成长计划
description: 纯函数很严格，也就是说你几乎除了计算数据以外什么都不能干，计算的时候还不能依赖除了函数参数以外的数据。
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=20 sentences=114, words=526

1. 渲染状态

```
let appState={
    title: {color: 'red',text: '标题'},
    content:{color:'green',text:'内容'}
}
function renderTitle(title) {
    let titleEle=document.querySelector('#title');
    titleEle.innerHTML=title.text;
    titleEle.style.color=title.color;
}
function renderContent(content) {
    let contentEle=document.querySelector('#content');
    contentEle.innerHTML=content.text;
    contentEle.style.color=content.color;
}
function renderApp(appState) {
    renderTitle(appState.title);
    renderContent(appState.content);
}
renderApp(appState);
```

2. 提高数据修改的门槛

- 一旦数据可以任意修改，所有对共享状态的操作都是不可预料的
- 模块之间需要共享数据和数据可能被任意修改导致不可预料的结果之间有矛盾
- 所有对数据的操作必须通过 dispatch 函数

```
let appState={
    title: {color: 'red',text: '标题'},
    content:{color:'green',text:'内容'}
}
function renderTitle(title) {
    let titleEle=document.querySelector('#title');
    titleEle.innerHTML=title.text;
    titleEle.style.color=title.color;
}
function renderContent(content) {
    let contentEle=document.querySelector('#content');
    contentEle.innerHTML=content.text;
    contentEle.style.color=content.color;
}
function renderApp(appState) {
    renderTitle(appState.title);
    renderContent(appState.content);
}
function dispatch(action) {
    switch (action.type) {
        case 'UPDATE_TITLE_COLOR':
            appState.title.color=action.color;
            break;
        case 'UPDATE_CONTENT_CONTENT':
            appState.content.text=action.text;
            break;
        default:
            break;
    }
}
dispatch({type:'UPDATE_TITLE_COLOR',color:'purple'});
dispatch({type:'UPDATE_CONTENT_CONTENT',text:'新标题'});

renderApp(appState);
```

3.封装仓库

```
function renderTitle(title) {
    let titleEle=document.querySelector('#title');
    titleEle.innerHTML=title.text;
    titleEle.style.color=title.color;
}
function renderContent(content) {
    let contentEle=document.querySelector('#content');
    contentEle.innerHTML=content.text;
    contentEle.style.color=content.color;
}
function renderApp(appState) {
    renderTitle(appState.title);
    renderContent(appState.content);
}

function createStore(reducer) {
    let state;
    function getState() {
        return state;
    }

    function dispatch(action) {
        state=reducer(state,action);
    }
    dispatch({});
    return {
        getState,
        dispatch
    }

}
let initState={
    title: {color: 'red',text: '标题'},
    content:{color:'green',text:'内容'}
}
let reducer=function (state=initState,action) {
    switch (action.type) {
        case 'UPDATE_TITLE_COLOR':
            return {...state,title: {...state.title,color:action.color}};
        case 'UPDATE_CONTENT_CONTENT':
        return {...state,content: {...state.content,text:action.text}};
            break;
        default:
            return state;
    }
}
let store=createStore(reducer);
renderApp(store.getState());
setTimeout(function () {
    store.dispatch({type:'UPDATE_TITLE_COLOR',color:'purple'});
    store.dispatch({type:'UPDATE_CONTENT_CONTENT',text:'新标题'});
    renderApp(store.getState());
},2000);
```

4. 监控数据变化

```
function renderTitle(title) {
    let titleEle=document.querySelector('#title');
    titleEle.innerHTML=title.text;
    titleEle.style.color=title.color;
}
function renderContent(content) {
    let contentEle=document.querySelector('#content');
    contentEle.innerHTML=content.text;
    contentEle.style.color=content.color;
}
function render() {
    renderTitle(store.getState().title);
    renderContent(store.getState().content);
}

function createStore(reducer) {
    let state;
    let listeners=[];
    function getState() {
        return state;
    }

    function dispatch(action) {
        state=reducer(state,action);
        listeners.forEach(l=>l());
    }

    function subscribe(listener) {
        listeners.push(listener);
        return () => {
            listeners = listeners.filter(item => item!=listener);
            console.log(listeners);
        }
    }
    dispatch({});
    return {
        getState,
        dispatch,
        subscribe
    }

}
let initState={
    title: {color: 'red',text: '标题'},
    content:{color:'green',text:'内容'}
}
let reducer=function (state=initState,action) {
    switch (action.type) {
        case 'UPDATE_TITLE_COLOR':
            return {...state,title: {...state.title,color:action.color}};
        case 'UPDATE_CONTENT_CONTENT':
        return {...state,content: {...state.content,text:action.text}};
            break;
        default:
            return state;
    }
}
let store=createStore(reducer);
render();
let unsubscribe = store.subscribe(render);
setTimeout(function () {
    store.dispatch({type:'UPDATE_TITLE_COLOR',color:'purple'});
    unsubscribe();
    store.dispatch({type:'UPDATE_CONTENT_CONTENT',text:'新标题'});
},2000);
```

5.纯函数

纯函数很严格，也就是说你几乎除了计算数据以外什么都不能干，计算的时候还不能依赖除了函数参数以外的数据。

- 函数的返回结果只依赖于它的参数
- 函数执行过程没有副作用,一个函数执行过程对产生了外部可观察的变化那么就说这个函数是有副作用的
    - 调用 DOM API 修改页面
    - 发送了 Ajax 请求