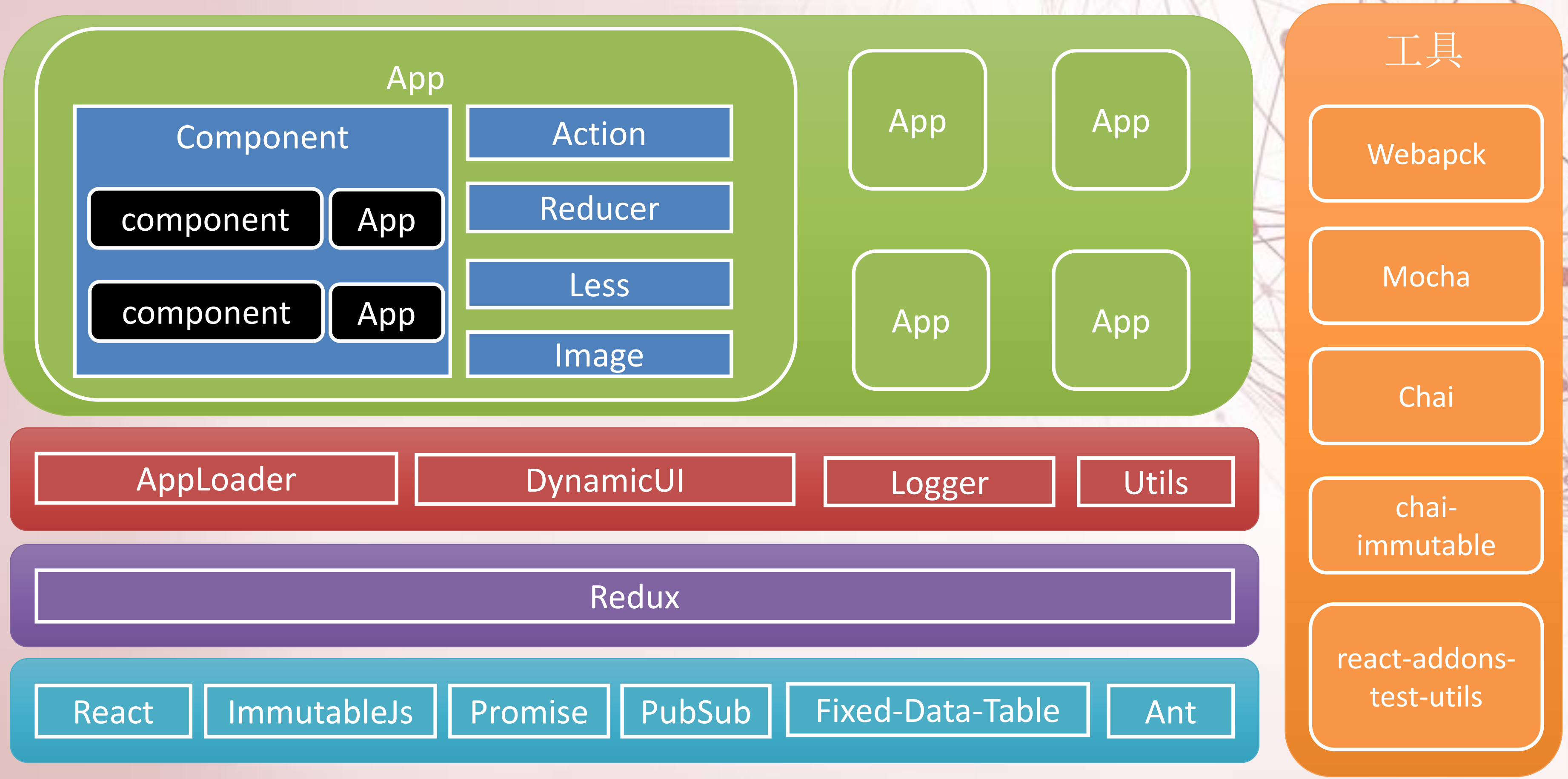
1

基于React前端框架

Monkey King

开发框架



App

Component

component

App

component

App

Action

Reducer

Less

Image

App

App

App

App

AppLoader

DynamicUI

Logger

Utils

Redux

React

ImmutableJs

Promise

PubSub

Fixed-Data-Table

Ant

工具

Webapck

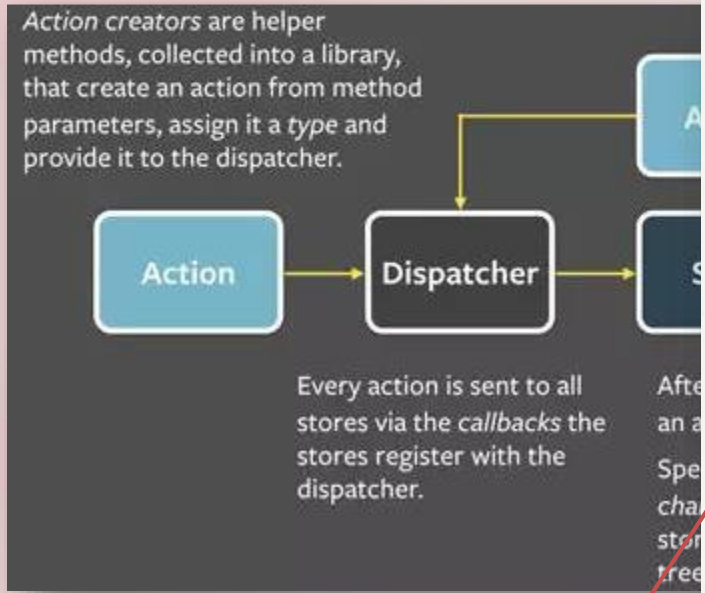
Mocha

Chai

chai-immutable

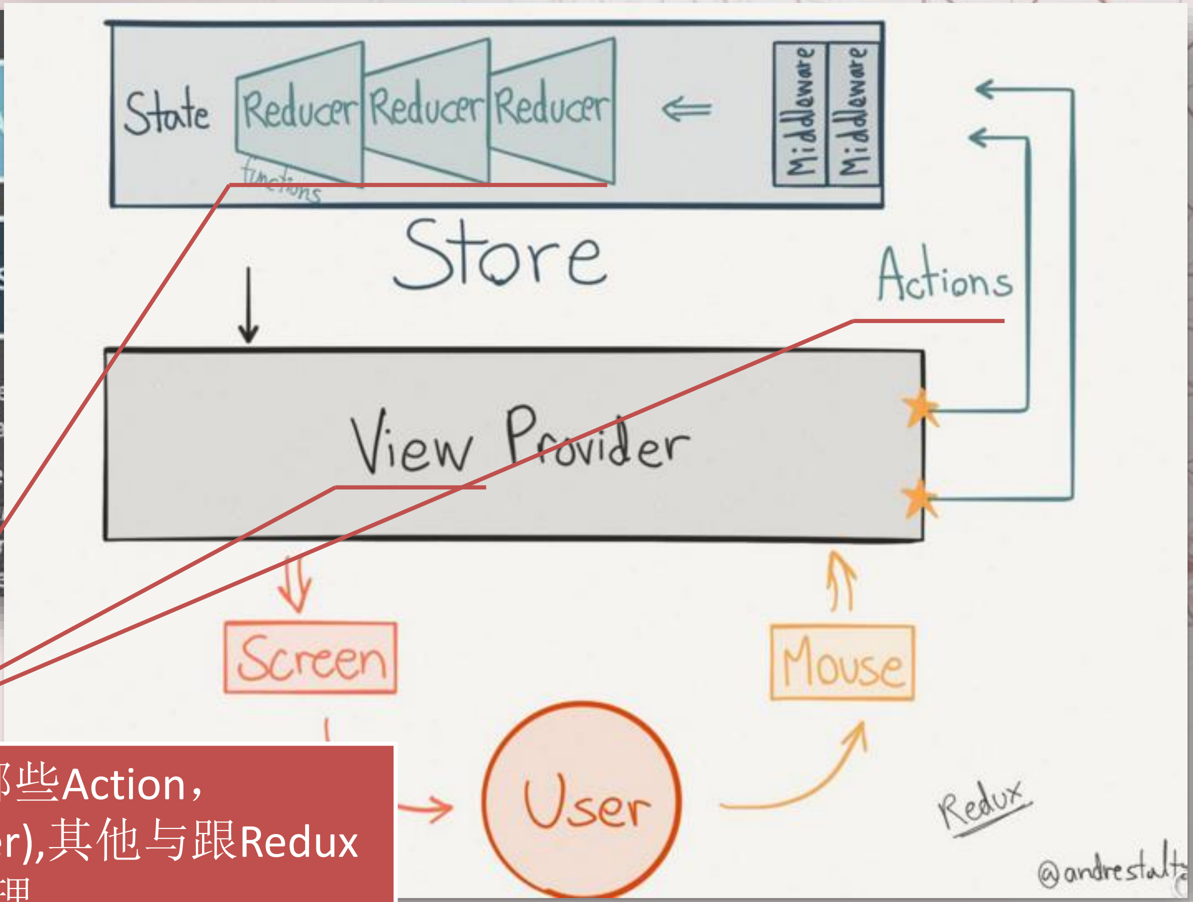
react-addons-test-utils

开发框架-Flux， Redux



Action creators are helper methods, collected into a library, that create an action from method parameters, assign it a *type* and provide it to the dispatcher.

Every action is sent to all stores via the *callbacks* the stores register with the dispatcher.



领域开发只需关心纯组件，有哪些Action，
以及Action对State的影响(reducer),其他与跟Redux
相关的东西有AppLoader黑盒处理

开发框架-特点

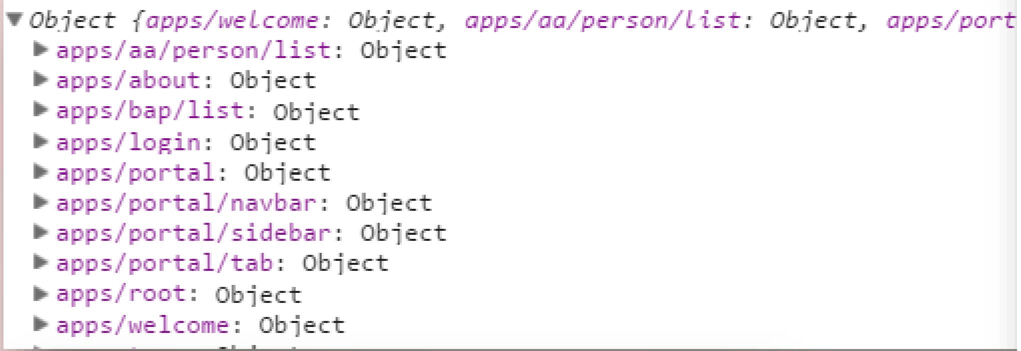
1、单页面
2 、app=component + action + reducer,Redux框架不可见
3、app path隔离state
4、state采用immutable数据类型
5、支持app方式扩展、二次开发
6、app间消息通信，支持事件回调或者发布订阅
7、按需加载app
8、可测试驱动开发TDD

开发框架-状态

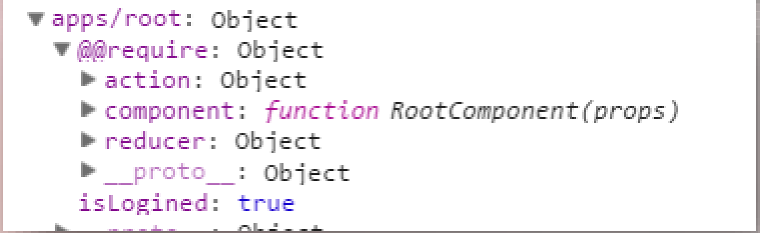
- State用于存储前端的数据
- 用App的path隔离
- Path格式有两部分path+query

<AppLoader path='apps/portal' />

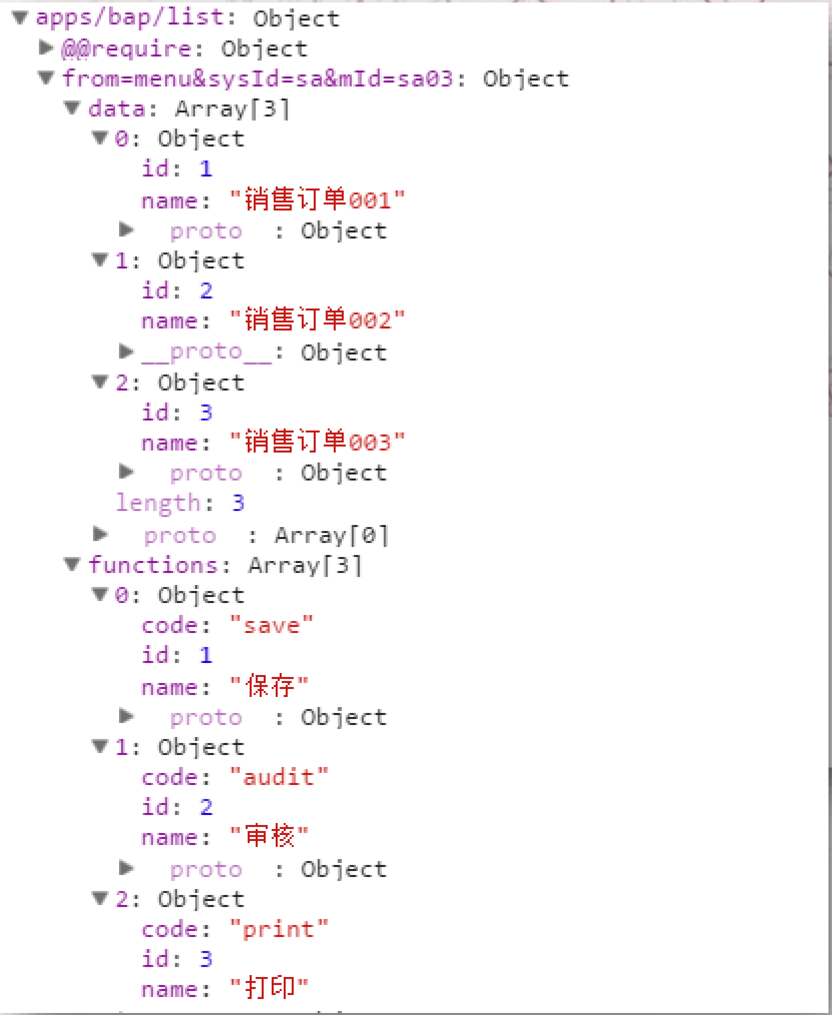
<AppLoader path=‘apps/bap/list?from=menu&s



```
▼ Object {apps/welcome: Object, apps/aa/person/list: Object, apps/port
  ▶ apps/aa/person/list: Object
  ▶ apps/about: Object
  ▶ apps/bap/list: Object
  ▶ apps/login: Object
  ▶ apps/portal: Object
  ▶ apps/portal/navbar: Object
  ▶ apps/portal/sidebar: Object
  ▶ apps/portal/tab: Object
  ▶ apps/root: Object
  ▶ apps/welcome: Object
```

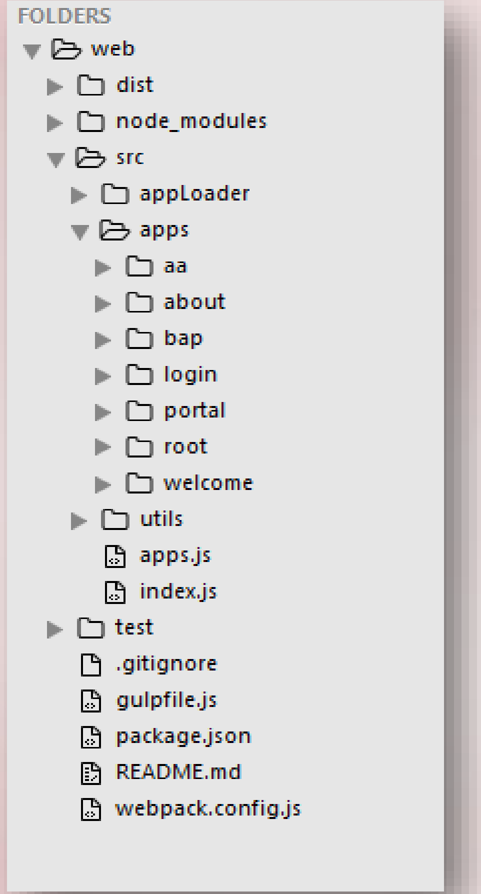


```
  ▼ apps/root: Object
    ▼ @@require: Object
      ▶ action: Object
      ▶ component: function RootComponent(props)
      ▶ reducer: Object
      ▶ __proto__: Object
      isLogined: true
```

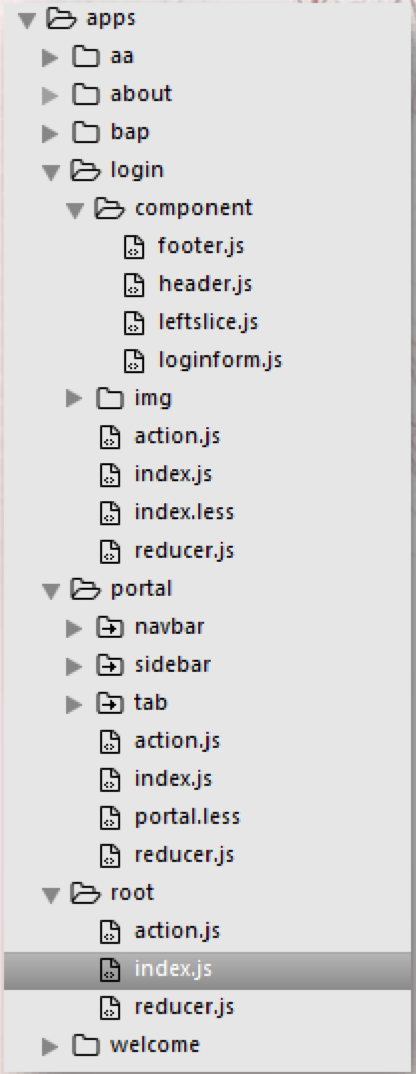


```
▼ apps/bap/list: Object
  ▶ @@require: Object
  ▼ from=menu&sysId=sa&mId=sa03: Object
    ▼ data: Array[3]
      ▼ 0: Object
          id: 1
          name: "销售订单001"
        ▶ __proto__: Object
      ▼ 1: Object
          id: 2
          name: "销售订单002"
        ▶ __proto__: Object
      ▼ 2: Object
          id: 3
          name: "销售订单003"
        ▶ __proto__: Object
        length: 3
      ▶ __proto__: Array[0]
    ▼ functions: Array[3]
      ▼ 0: Object
          code: "save"
          id: 1
          name: "保存"
        ▶ __proto__: Object
      ▼ 1: Object
          code: "audit"
          id: 2
          name: "审核"
        ▶ __proto__: Object
      ▼ 2: Object
          code: "print"
          id: 3
          name: "打印"
```

开发框架-目录结构

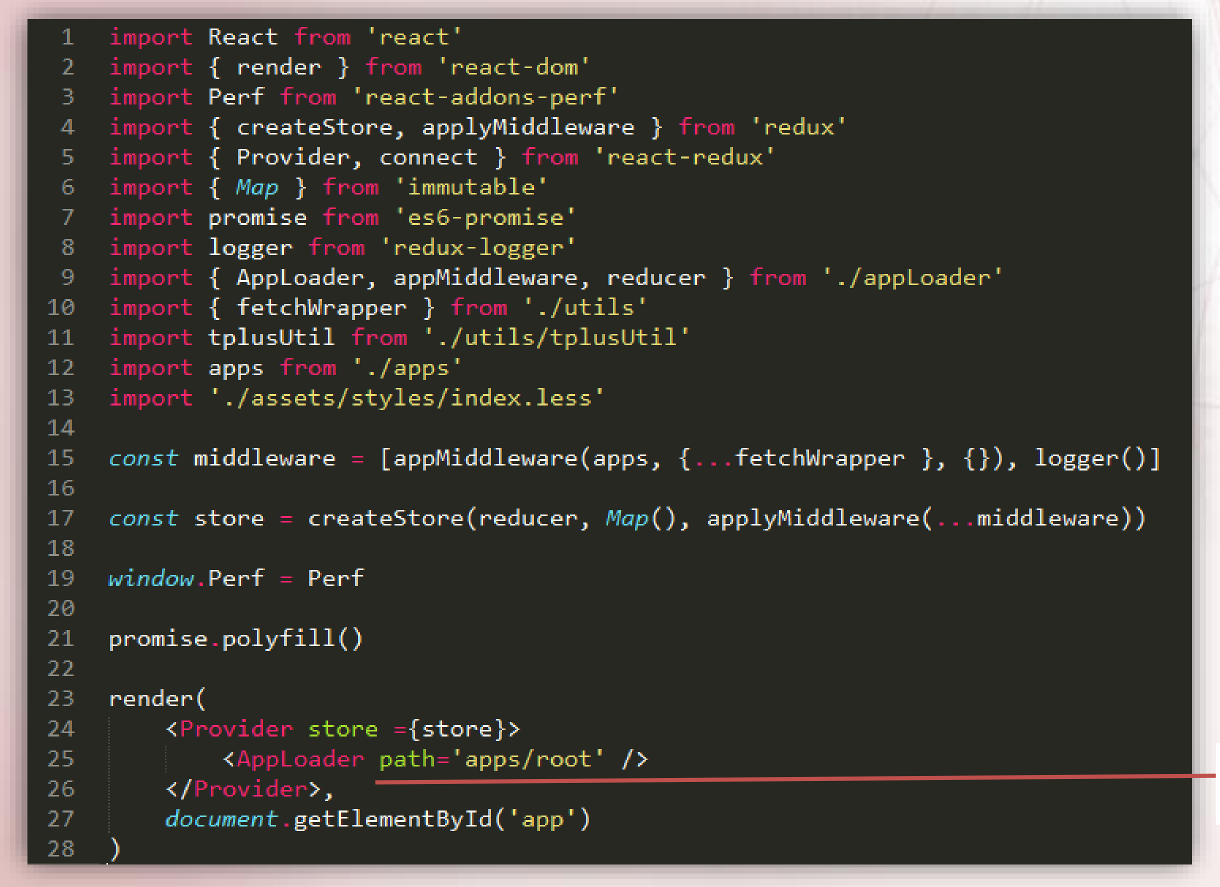


```
FOLDERS
▼ web
  ▶ dist
  ▶ node_modules
  ▼ src
    ▶ appLoader
    ▼ apps
      ▶ aa
      ▶ about
      ▶ bap
      ▶ login
      ▶ portal
      ▶ root
      ▶ welcome
    ▶ utils
      apps.js
      index.js
  ▶ test
    .gitignore
    gulpfile.js
    package.json
    README.md
    webpack.config.js
```

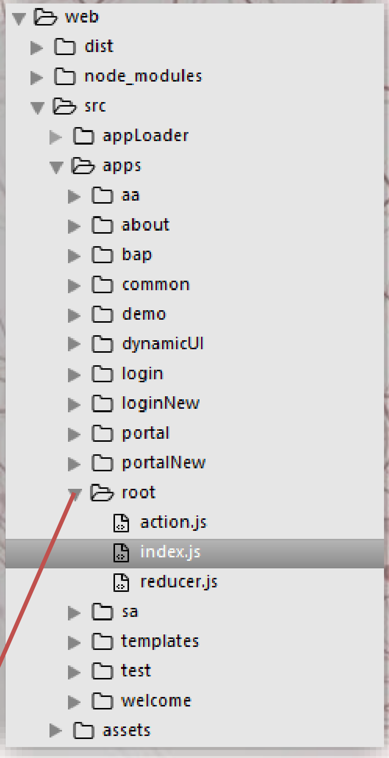


```
▼ apps
  ▶ aa
  ▶ about
  ▶ bap
  ▼ login
    ▼ component
        footer.js
        header.js
        leftslice.js
        loginform.js
    ▶ img
      action.js
      index.js
      index.less
      reducer.js
  ▼ portal
    ▶ navbar
    ▶ sidebar
    ▶ tab
      action.js
      index.js
      portal.less
      reducer.js
  ▼ root
      action.js
      index.js
      reducer.js
  ▶ welcome
```

开发框架-index代码



```
import React from 'react'
import { render } from 'react-dom'
import Perf from 'react-addons-perf'
import { createStore, applyMiddleware } from 'redux'
import { Provider, connect } from 'react-redux'
import { Map } from 'immutable'
import promise from 'es6-promise'
import logger from 'redux-logger'
import { AppLoader, appMiddleware, reducer } from './appLoader'
import { fetchWrapper } from './utils'
import tplusUtil from './utils/tplusUtil'
import apps from './apps'
import './assets/styles/index.less'

const middleware = [appMiddleware(apps, {...fetchWrapper }, {}), logger()]

const store = createStore(reducer, Map(), applyMiddleware(...middleware))

window.Perf = Perf

promise.polyfill()

render(
    <Provider store ={store}>
        <AppLoader path='apps/root' />
    </Provider>,
    document.getElementById('app')
)
```

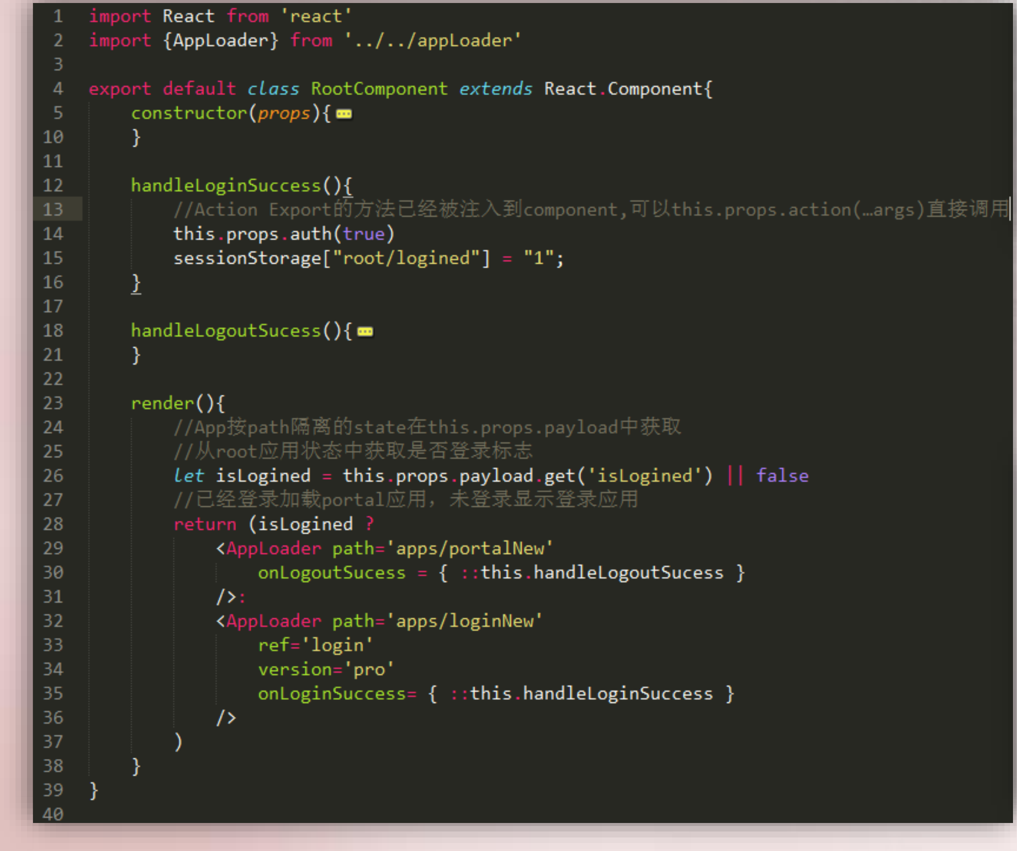


```
▼ web
  ▶ dist
  ▶ node_modules
  ▼ src
    ▶ appLoader
    ▼ apps
      ▶ aa
      ▶ about
      ▶ bap
      ▶ common
      ▶ demo
      ▶ dynamicUI
      ▶ login
      ▶ loginNew
      ▶ portal
      ▶ portalNew
      ▼ root
          action.js
          index.js
          reducer.js
      ▶ sa
      ▶ templates
      ▶ test
      ▶ welcome
    ▶ assets
```

App需要指定路径

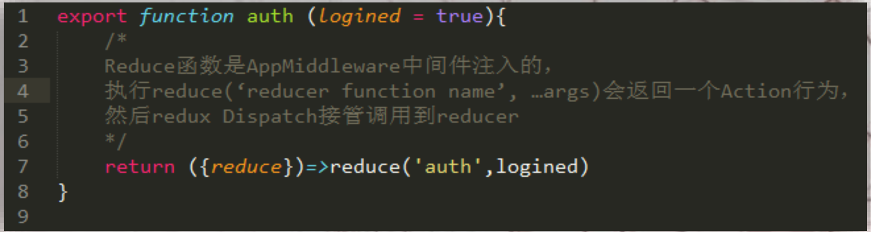
开发框架-App(root）

Component



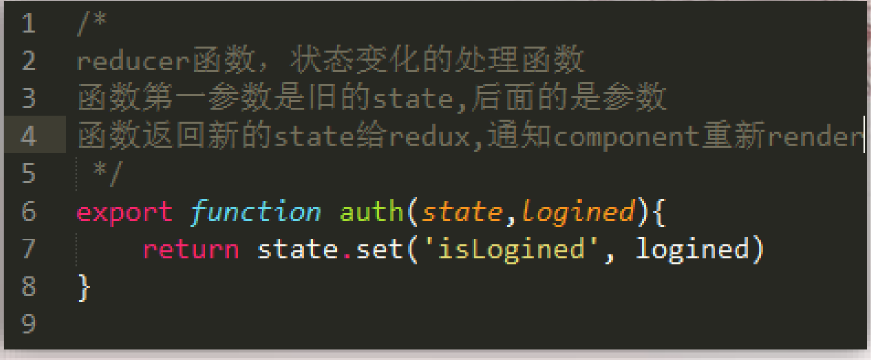
```
import React from 'react'
import {AppLoader} from '../../appLoader'

export default class RootComponent extends React.Component{
    constructor(props){⋯
    }

    handleLoginSuccess(){
        //Action Export的方法已经被注入到component,可以this.props.action(…args)直接调用
        this.props.auth(true)
        sessionStorage["root/logined"] = "1";
    }

    handleLogoutSucess(){⋯
    }

    render(){
        //App按path隔离的state在this.props.payload中获取
        //从root应用状态中获取是否登录标志
        let isLogined = this.props.payload.get('isLogined') || false
        //已经登录加载portal应用，未登录显示登录应用
        return (isLogined ?
            <AppLoader path='apps/portalNew'
                onLogoutSucess = { ::this.handleLogoutSucess }
            />:
            <AppLoader path='apps/loginNew'
                ref='login'
                version='pro'
                onLoginSuccess= { ::this.handleLoginSuccess }
            />
        )
    }
}
```

Action



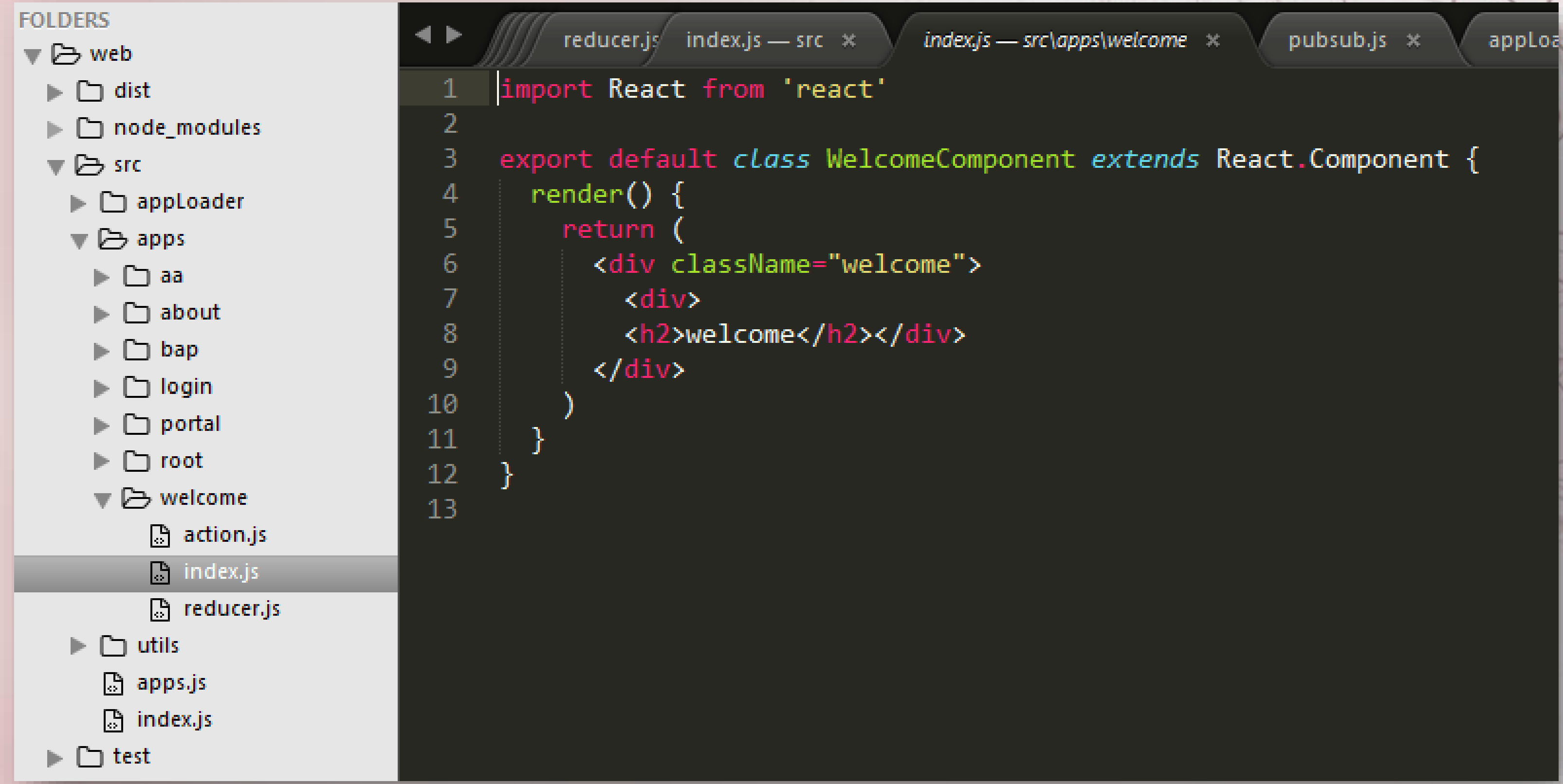
```
export function auth (logined = true){
    /*
    Reduce函数是AppMiddleware中间件注入的，
    执行reduce(‘reducer function name’, …args)会返回一个Action行为，
    然后redux Dispatch接管调用到reducer
    */
    return ({reduce})=>reduce('auth',logined)
}
```

Reducer



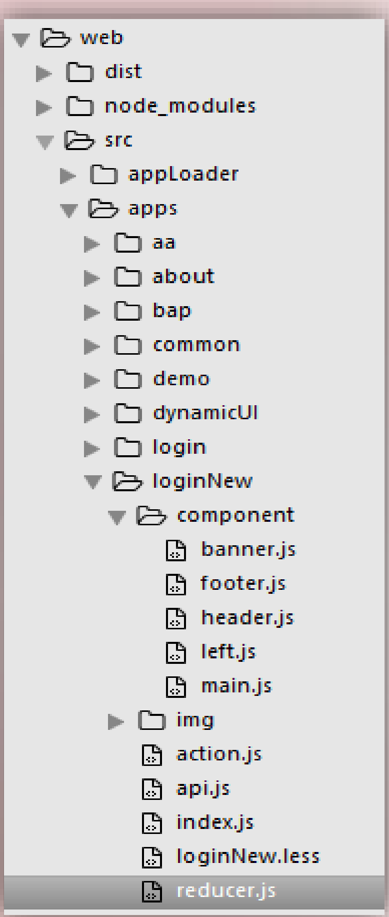
```
/*
reducer函数，状态变化的处理函数
函数第一参数是旧的state,后面的是参数
函数返回新的state给redux,通知component重新render
 */
export function auth(state,logined){
    return state.set('isLogined', logined)
}
```

开发框架-App（welcome）

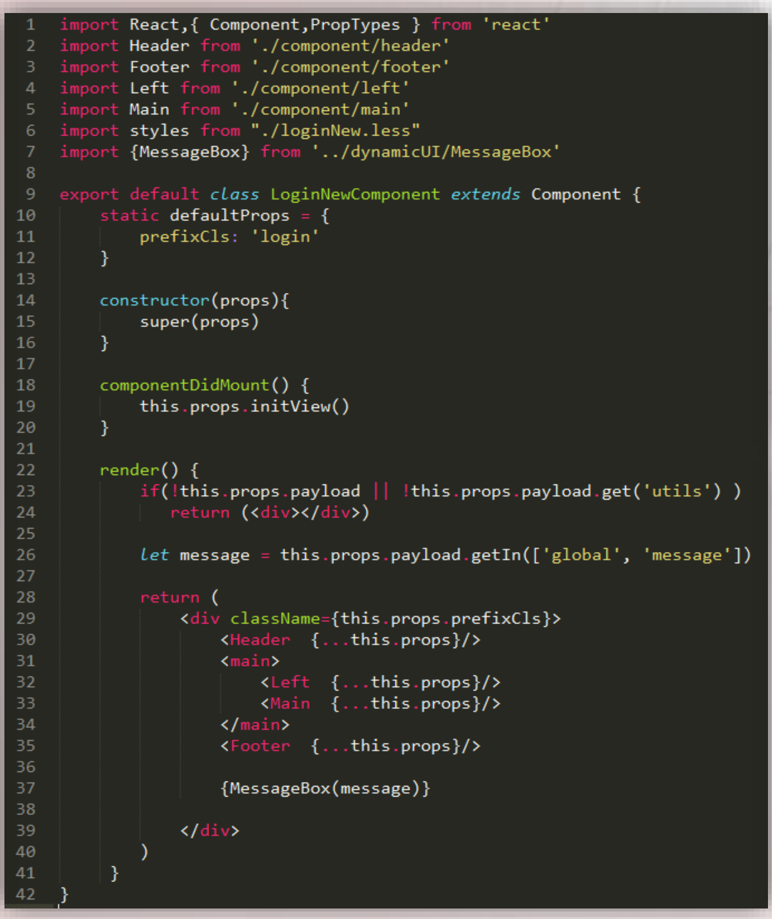


```
import React from 'react'

export default class WelcomeComponent extends React.Component {
  render() {
    return (
      <div className="welcome">
        <div>
        <h2>welcome</h2></div>
      </div>
    )
  }
}

```

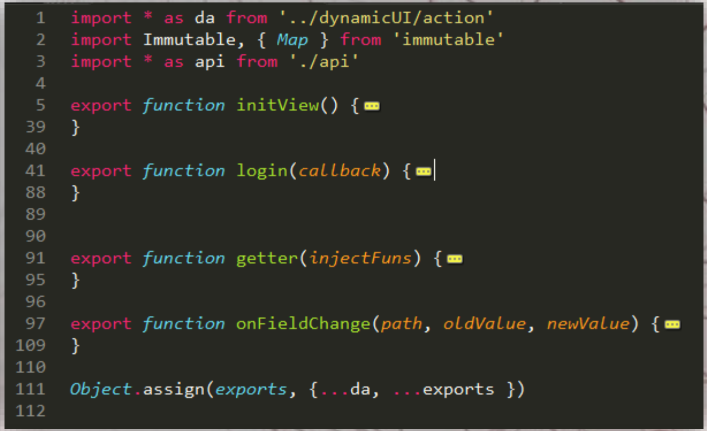
开发框架-App（login）



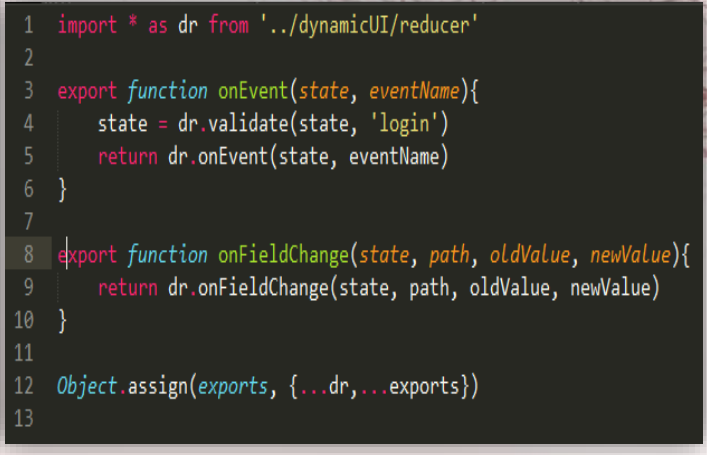
- web
  - dist
  - node_modules
  - src
    - appLoader
    - apps
      - aa
      - about
      - bap
      - common
      - demo
      - dynamicUI
      - login
      - loginNew
        - component
          - banner.js
          - footer.js
          - header.js
          - left.js
          - main.js
        - img
        - action.js
        - api.js
        - index.js
        - loginNew.less
        - reducer.js



```
import React,{ Component,PropTypes } from 'react'
import Header from './component/header'
import Footer from './component/footer'
import Left from './component/left'
import Main from './component/main'
import styles from "./loginNew.less"
import {MessageBox} from '../dynamicUI/MessageBox'

export default class LoginNewComponent extends Component {
    static defaultProps = {
        prefixCls: 'login'
    }

    constructor(props){
        super(props)
    }

    componentDidMount() {
        this.props.initView()
    }

    render() {
        if(!this.props.payload || !this.props.payload.get('utils') )
           return (<div></div>)

        let message = this.props.payload.getIn(['global', 'message'])

        return (
            <div className={this.props.prefixCls}>
                <Header  {...this.props}/>
                <main>
                    <Left  {...this.props}/>
                    <Main  {...this.props}/>
                </main>
                <Footer  {...this.props}/>

                {MessageBox(message)}

            </div>
        )
    }
}
```

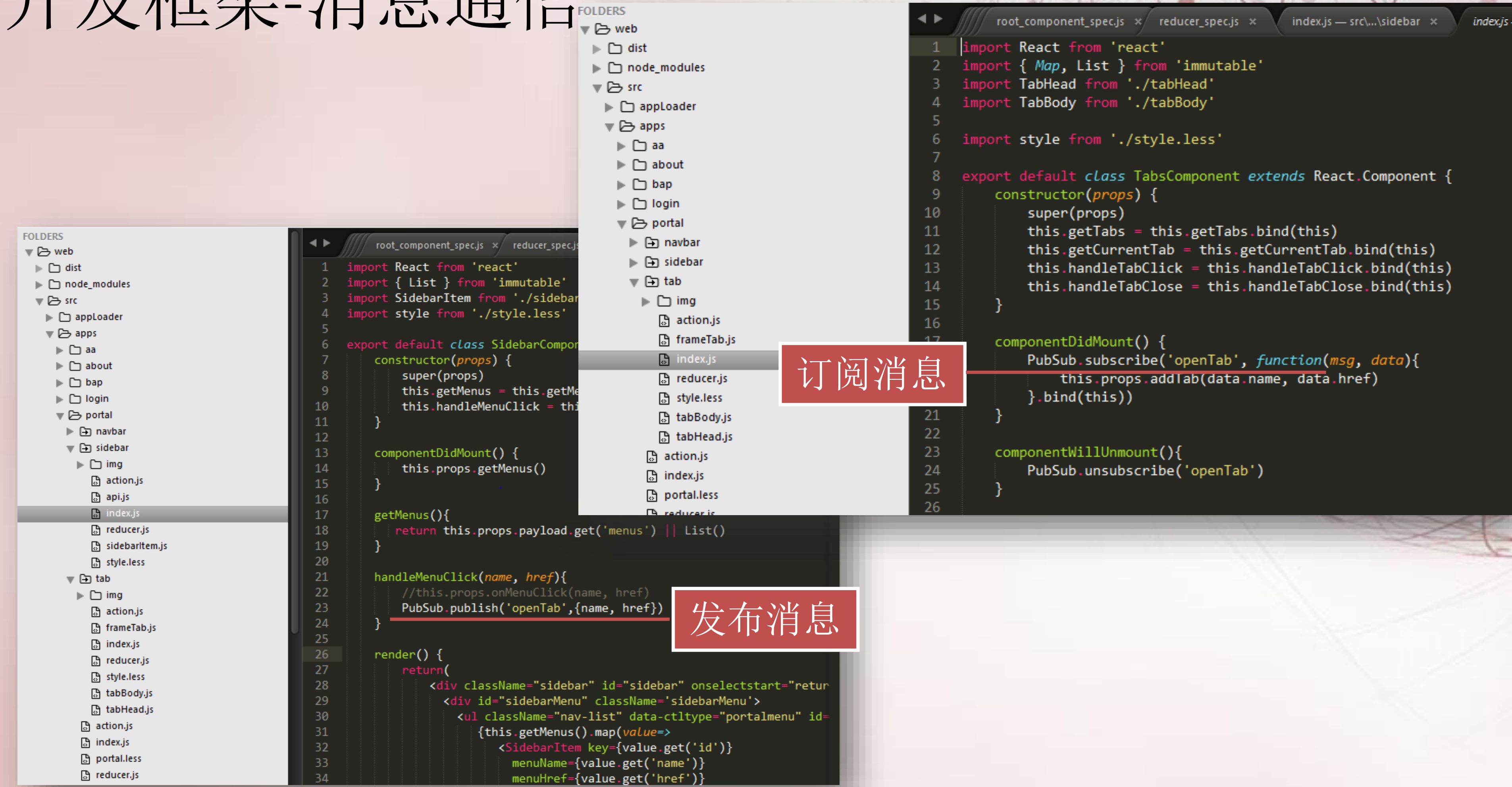


```
import * as da from '../dynamicUI/action'
import Immutable, { Map } from 'immutable'
import * as api from './api'

export function initView() {⋯
}

export function login(callback) {⋯
}


export function getter(injectFuns) {⋯
}

export function onFieldChange(path, oldValue, newValue) {⋯
}

Object.assign(exports, {...da, ...exports })

```



```
import * as dr from '../dynamicUI/reducer'

export function onEvent(state, eventName){
    state = dr.validate(state, 'login')
    return dr.onEvent(state, eventName)
}

export function onFieldChange(state, path, oldValue, newValue){
    return dr.onFieldChange(state, path, oldValue, newValue)
}

Object.assign(exports, {...dr,...exports})

```

开发框架-消息通信



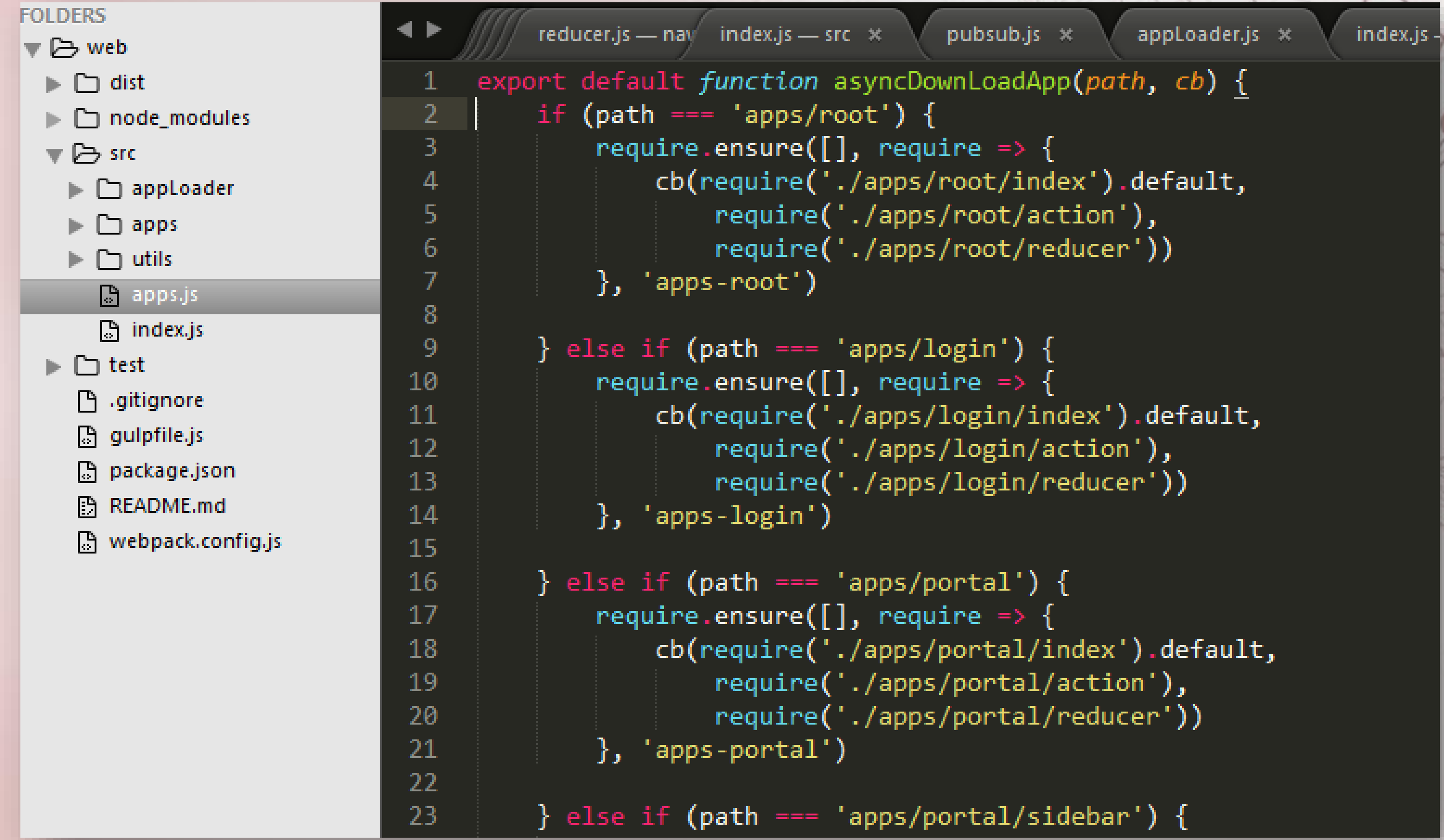
```
import React from 'react'
import { List } from 'immutable'
import SidebarItem from './sidebar
import style from './style.less'

export default class SidebarCompon
    constructor(props) {
        super(props)
        this.getMenus = this.getMe
        this.handleMenuClick = thi
    }

    componentDidMount() {
        this.props.getMenus()
    }

    getMenus(){
        return this.props.payload.get('menus') || List()
    }

    handleMenuClick(name, href){
        //this.props.onMenuClick(name, href)
        PubSub.publish('openTab',{name, href})
    }

    render() {
        return(
            <div className="sidebar" id="sidebar" onselectstart="retur
              <div id="sidebarMenu" className='sidebarMenu'>
                <ul className="nav-list" data-ctltype="portalmenu" id=
                  {this.getMenus().map(value=>
                    <SidebarItem key={value.get('id')}
                      menuName={value.get('name')}
                      menuHref={value.get('href')}
```

发布消息

```
import React from 'react'
import { Map, List } from 'immutable'
import TabHead from './tabHead'
import TabBody from './tabBody'

import style from './style.less'

export default class TabsComponent extends React.Component {
    constructor(props) {
        super(props)
        this.getTabs = this.getTabs.bind(this)
        this.getCurrentTab = this.getCurrentTab.bind(this)
        this.handleTabClick = this.handleTabClick.bind(this)
        this.handleTabClose = this.handleTabClose.bind(this)
    }

    componentDidMount() {
        PubSub.subscribe('openTab', function(msg, data){
            this.props.addTab(data.name, data.href)
        }.bind(this))
    }

    componentWillUnmount(){
        PubSub.unsubscribe('openTab')
    }
```

订阅消息

开发框架-异步按需加载配置

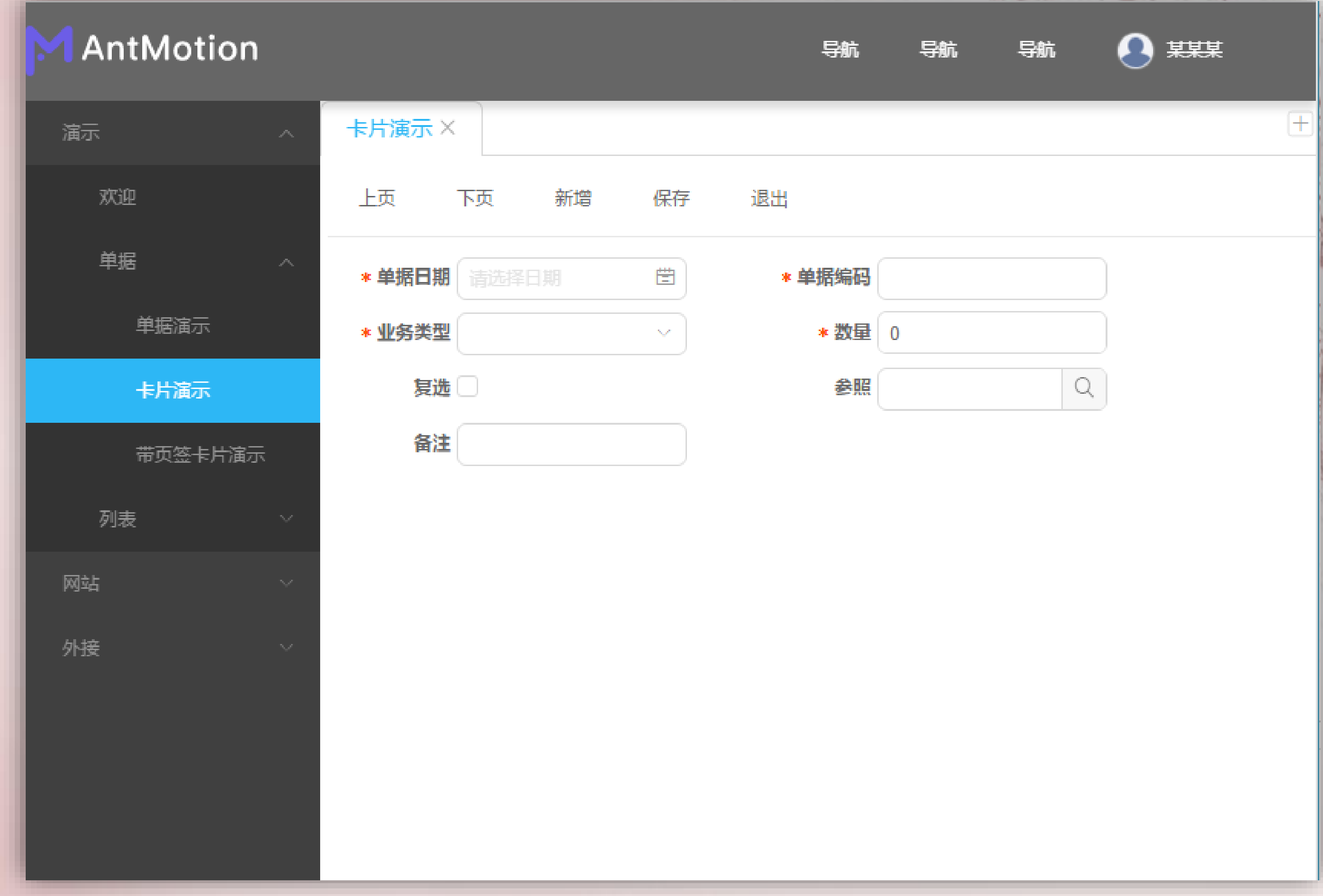


```
export default function asyncDownLoadApp(path, cb) {
    if (path === 'apps/root') {
        require.ensure([], require => {
            cb(require('./apps/root/index').default,
                require('./apps/root/action'),
                require('./apps/root/reducer'))
        }, 'apps-root')

    } else if (path === 'apps/login') {
        require.ensure([], require => {
            cb(require('./apps/login/index').default,
                require('./apps/login/action'),
                require('./apps/login/reducer'))
        }, 'apps-login')

    } else if (path === 'apps/portal') {
        require.ensure([], require => {
            cb(require('./apps/portal/index').default,
                require('./apps/portal/action'),
                require('./apps/portal/reducer'))
        }, 'apps-portal')

    } else if (path === 'apps/portal/sidebar') {
```

开发框架-动态UI

- 前端UI=控件meta+数据data+程序app
- Meta、Data格式采用json结构
- 可定义Component和Data的绑定关系
- 领域开发不接触dom
- 控件值修改自动同步state
- 程序可以截获每次取值，重写逻辑
- 程序可以截获值变化事件，增加联动逻辑

开发框架-动态UI-Card demo-效果



* 单据日期 请选择日期

* 单据编码

* 业务类型

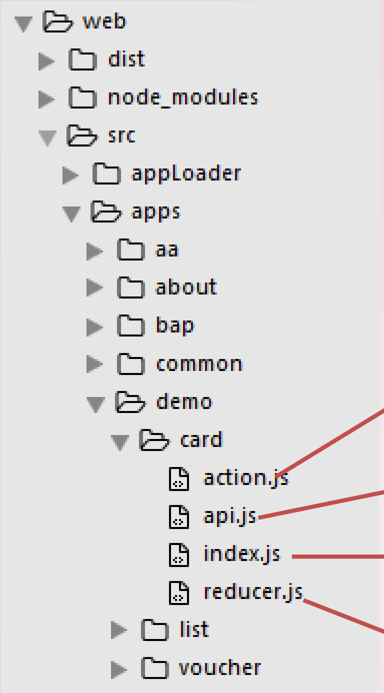
* 数量 0

复选 □

参照

备注

开发框架-动态UI-Card demo-代码结构



```
▼ web
  ▶ dist
  ▶ node_modules
  ▼ src
    ▶ appLoader
    ▼ apps
      ▶ aa
      ▶ about
      ▶ bap
      ▶ common
      ▼ demo
        ▼ card
            action.js
            api.js
            index.js
            reducer.js
        ▶ list
        ▶ voucher
```

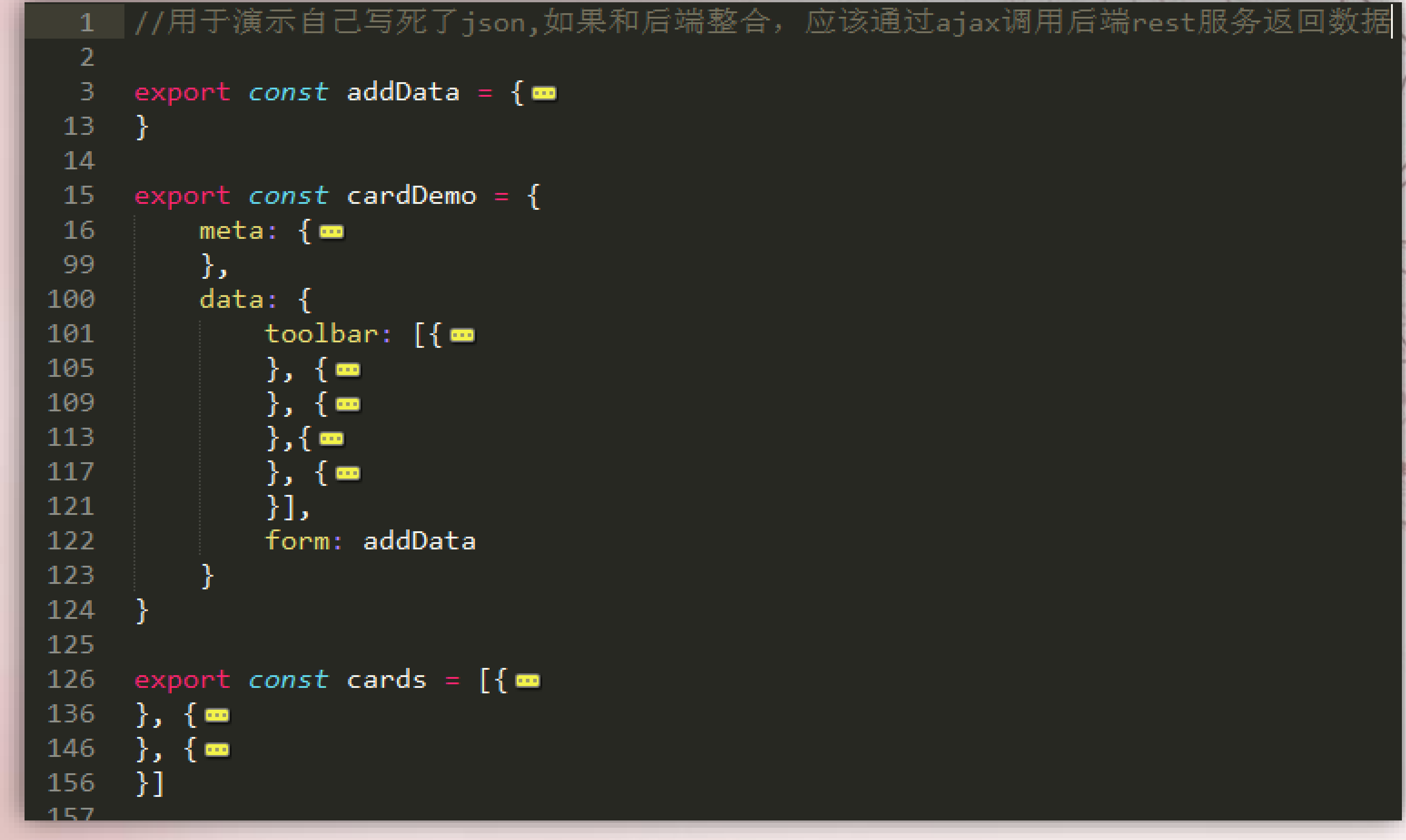
Action，App行为，component可以通过this.props.[action]调用；ajax调用后台通过import api.js调用方法实现

集中Ajax调用后台的函数，ajax调用采用promise

Component，组件包含react component

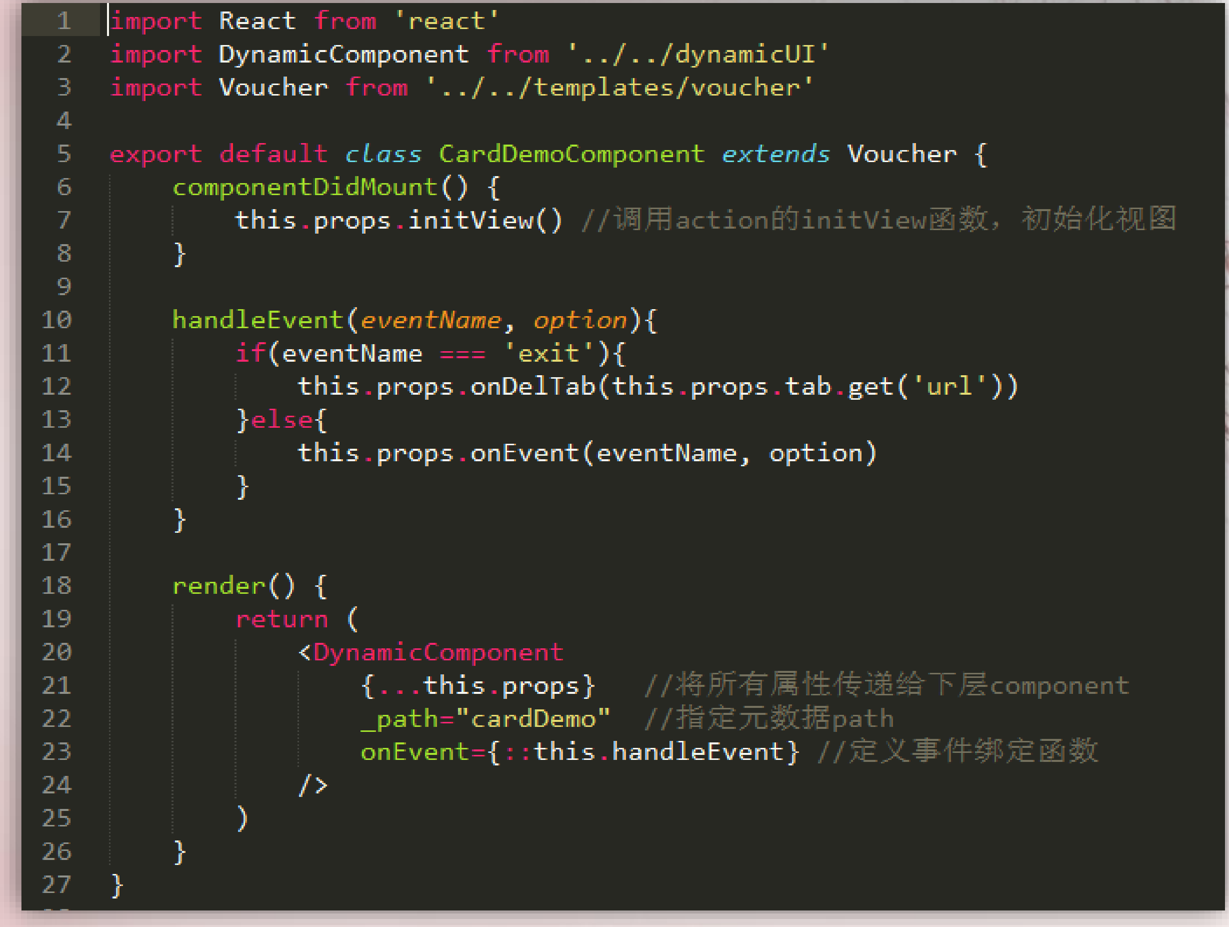
Reducer，包含state计算的多个函数，每个函数会接收旧的state通过计算返回新的state,通知component重新render

开发框架-动态UI-Card demo-Api



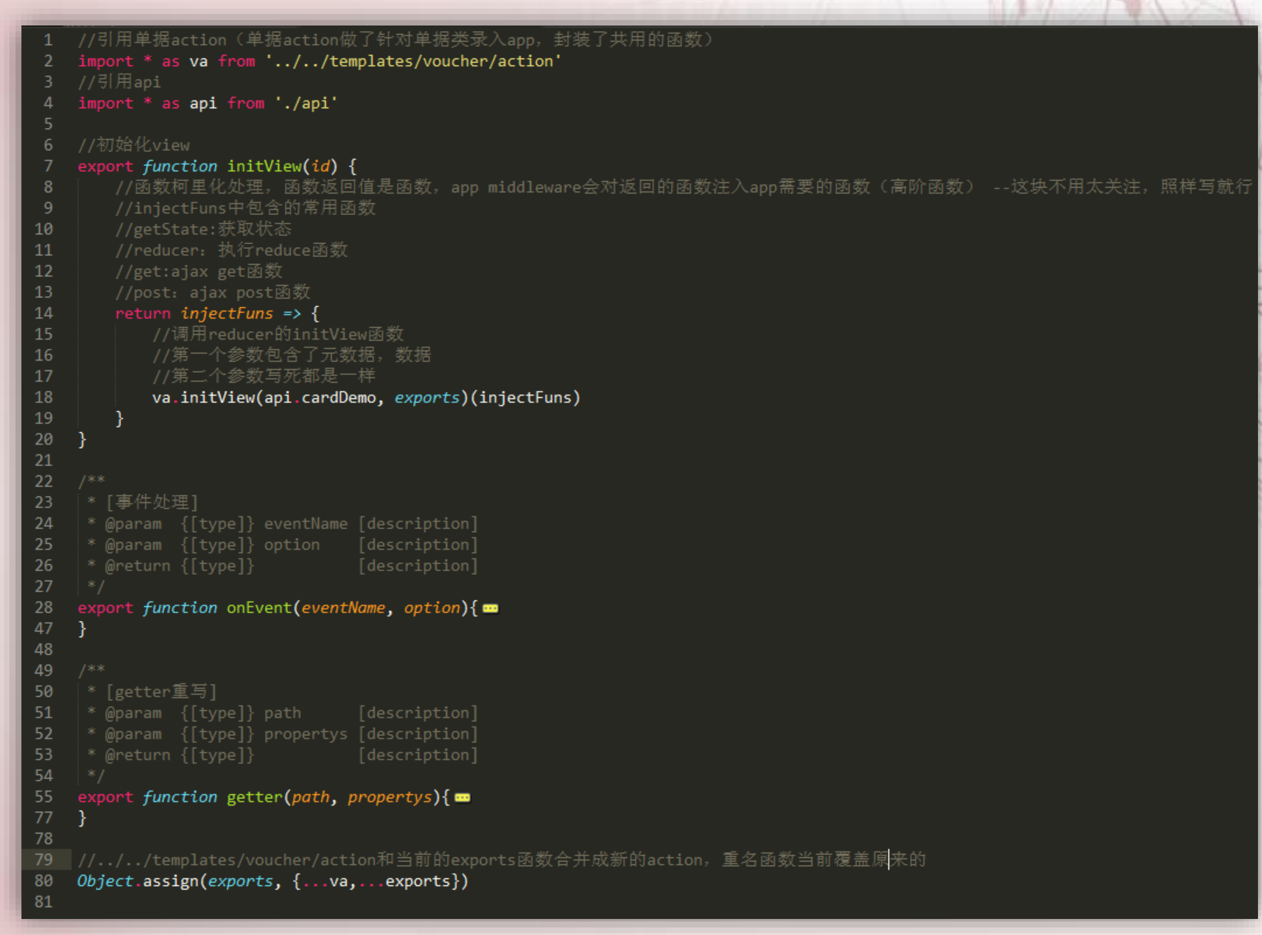
```
//用于演示自己写死了json,如果和后端整合，应该通过ajax调用后端rest服务返回数据

export const addData = {⋯
}

export const cardDemo = {
    meta: {⋯
    },
    data: {
        toolbar: [{⋯
        }, {⋯
        }, {⋯
        },{⋯
        }, {⋯
        }],
        form: addData
    }
}

export const cards = [{⋯
}, {⋯
}, {⋯
}]
```

开发框架-动态UI-Card demo-Component



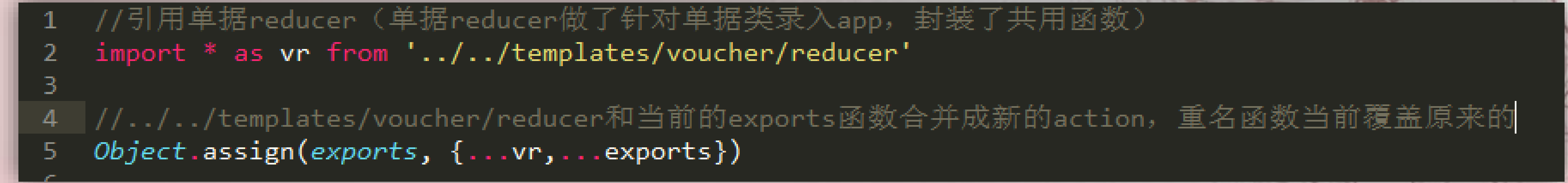
```
import React from 'react'
import DynamicComponent from '../../dynamicUI'
import Voucher from '../../templates/voucher'

export default class CardDemoComponent extends Voucher {
    componentDidMount() {
        this.props.initView() //调用action的initView函数，初始化视图
    }

    handleEvent(eventName, option){
        if(eventName === 'exit'){
            this.props.onDelTab(this.props.tab.get('url'))
        }else{
            this.props.onEvent(eventName, option)
        }
    }

    render() {
        return (
            <DynamicComponent
                {...this.props}   //将所有属性传递给下层component
                _path="cardDemo"  //指定元数据path
                onEvent={::this.handleEvent} //定义事件绑定函数
            />
        )
    }
}
```

开发框架-动态UI-Card demo-Action



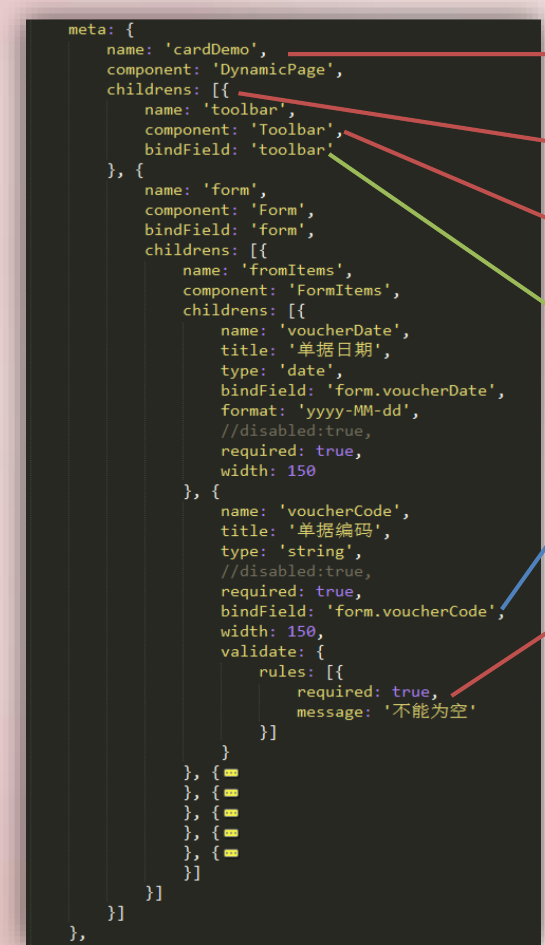
```javascript
//引用单据action（单据action做了针对单据类录入app，封装了共用的函数）
import * as va from '../../templates/voucher/action'
//引用api
import * as api from './api'

//初始化view
export function initView(id) {
    //函数柯里化处理，函数返回值是函数，app middleware会对返回的函数注入app需要的函数（高阶函数） --这块不用太关注，照样写就行
    //injectFuns中包含的常用函数
    //getState:获取状态
    //reducer：执行reduce函数
    //get:ajax get函数
    //post：ajax post函数
    return injectFuns => {
        //调用reducer的initView函数
        //第一个参数包含了元数据，数据
        //第二个参数写死都是一样
        va.initView(api.cardDemo, exports)(injectFuns)
    }
}

/**
 * [事件处理]
 * @param  {[type]} eventName [description]
 * @param  {[type]} option    [description]
 * @return {[type]}           [description]
 */
export function onEvent(eventName, option){⋯
}

/**
 * [getter重写]
 * @param  {[type]} path      [description]
 * @param  {[type]} propertys [description]
 * @return {[type]}           [description]
 */
export function getter(path, propertys){⋯
}

//../../templates/voucher/action和当前的exports函数合并成新的action，重名函数当前覆盖原来的
Object.assign(exports, {...va,...exports})
```

开发框架-动态UI-Card demo-Reducer



```
//引用单据reducer（单据reducer做了针对单据类录入app，封装了共用函数）
import * as vr from '../../templates/voucher/reducer'

//../../templates/voucher/reducer和当前的exports函数合并成新的action，重名函数当前覆盖原来的
Object.assign(exports, {...vr,...exports})
```

开发框架-动态UI-meta、 data



```
meta: {
    name: 'cardDemo',
    component: 'DynamicPage',
    childrens: [{
        name: 'toolbar',
        component: 'Toolbar',
        bindField: 'toolbar'
    }, {
        name: 'form',
        component: 'Form',
        bindField: 'form',
        childrens: [{
            name: 'fromItems',
            component: 'FormItems',
            childrens: [{
                name: 'voucherDate',
                title: '单据日期',
                type: 'date',
                bindField: 'form.voucherDate',
                format: 'yyyy-MM-dd',
                //disabled:true,
                required: true,
                width: 150
            }, {
                name: 'voucherCode',
                title: '单据编码',
                type: 'string',
                //disabled:true,
                required: true,
                bindField: 'form.voucherCode',
                width: 150,
                validate: {
                    rules: [{
                        required: true,
                        message: '不能为空'
                    }]
                }
            }, {⋯
            }, {⋯
            }, {⋯
            }, {⋯
            }, {⋯
            }]
        }]
    }]
},
```

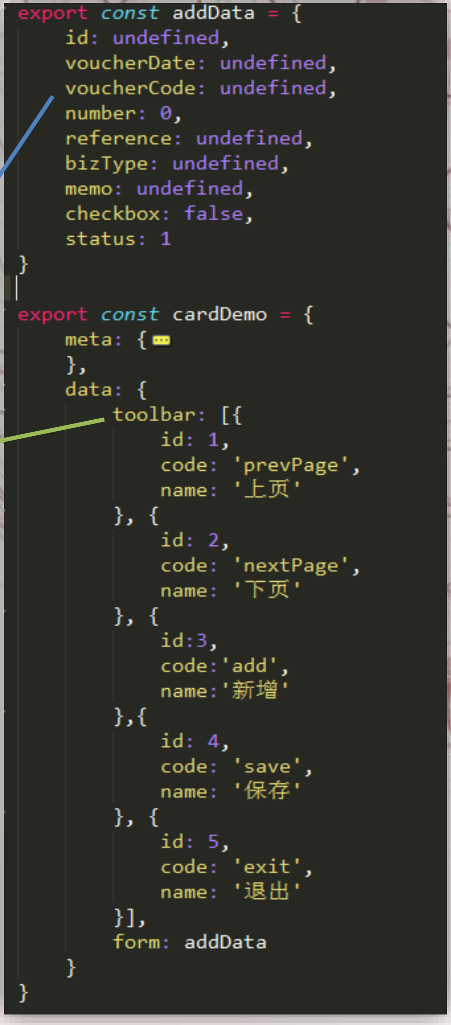
App名，随意定义

子控件数组

使用控件名

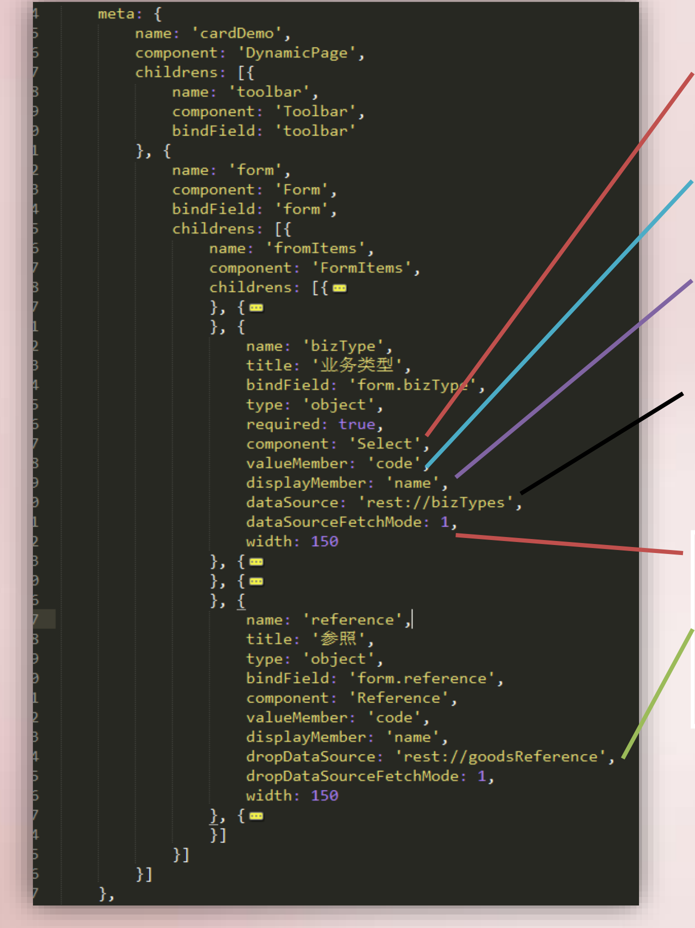
绑定Data属性

定义非空校验规则



```
export const addData = {
    id: undefined,
    voucherDate: undefined,
    voucherCode: undefined,
    number: 0,
    reference: undefined,
    bizType: undefined,
    memo: undefined,
    checkbox: false,
    status: 1
}

export const cardDemo = {
    meta: {⋯
    },
    data: {
        toolbar: [{
            id: 1,
            code: 'prevPage',
            name: '上页'
        }, {
            id: 2,
            code: 'nextPage',
            name: '下页'
        }, {
            id:3,
            code:'add',
            name:'新增'
        },{
            id: 4,
            code: 'save',
            name: '保存'
        }, {
            id: 5,
            code: 'exit',
            name: '退出'
        }],
        form: addData
    }
}
```

开发框架-动态UI-复杂Component



```
meta: {
    name: 'cardDemo',
    component: 'DynamicPage',
    childrens: [{
        name: 'toolbar',
        component: 'Toolbar',
        bindField: 'toolbar'
    }, {
        name: 'form',
        component: 'Form',
        bindField: 'form',
        childrens: [{
            name: 'fromItems',
            component: 'FormItems',
            childrens: [{⋯
            }, {⋯
            }, {
                name: 'bizType',
                title: '业务类型',
                bindField: 'form.bizType',
                type: 'object',
                required: true,
                component: 'Select',
                valueMember: 'code',
                displayMember: 'name',
                dataSource: 'rest://bizTypes',
                dataSourceFetchMode: 1,
                width: 150
            }, {⋯
            }, {⋯
            }, {
                name: 'reference',
                title: '参照',
                type: 'object',
                bindField: 'form.reference',
                component: 'Reference',
                valueMember: 'code',
                displayMember: 'name',
                dropDataSource: 'rest://goodsReference',
                dropDataSourceFetchMode: 1,
                width: 150
            }, {⋯
            }]
        }]
    }]
},
```

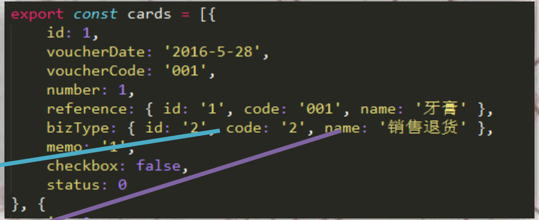
使用控件名

值对应成员属性

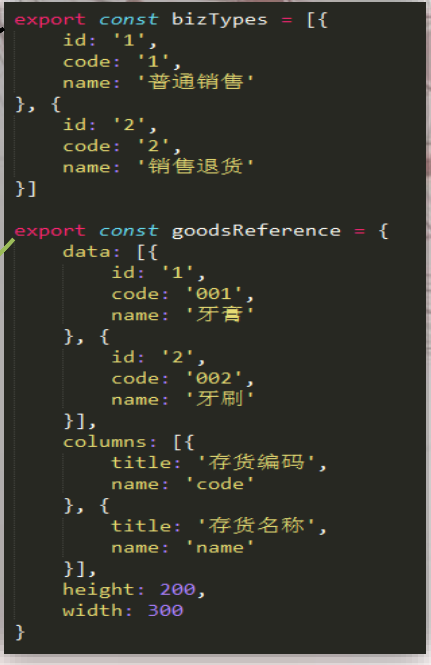
显示对应成员属性

数据源，值或者rest服务

数据源获取方式，仅一次或者每次都获取等

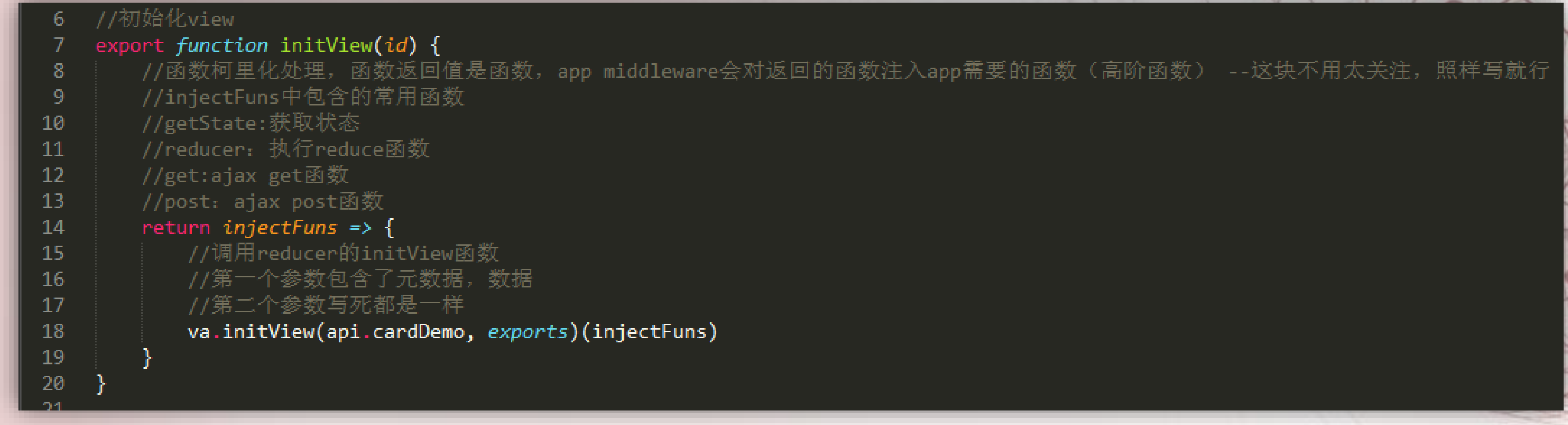


```
export const cards = [{
    id: 1,
    voucherDate: '2016-5-28',
    voucherCode: '001',
    number: 1,
    reference: { id: '1', code: '001', name: '牙膏' },
    bizType: { id: '2', code: '2', name: '销售退货' },
    memo: '1',
    checkbox: false,
    status: 0
}, {
```



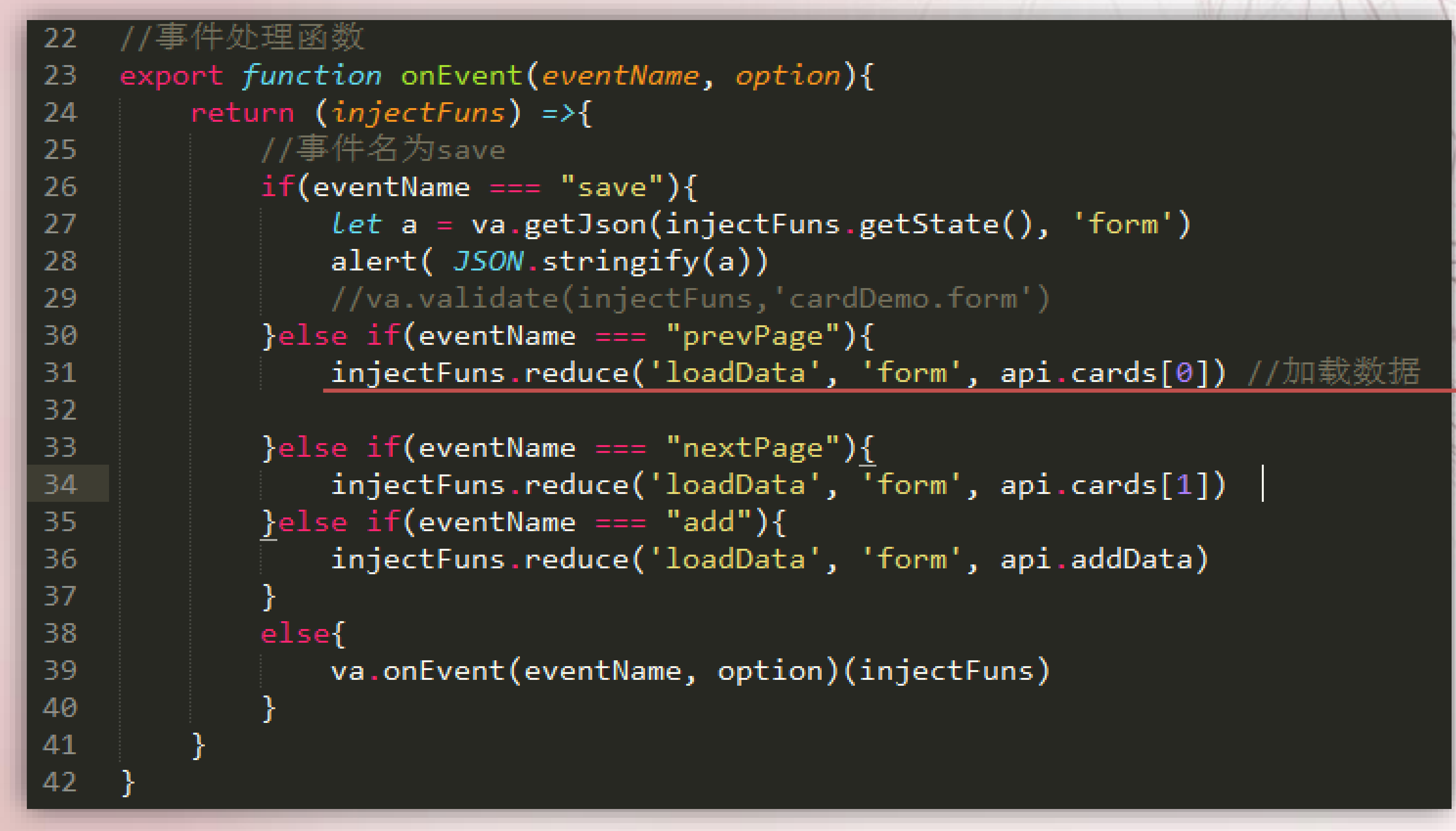
```
export const bizTypes = [{
    id: '1',
    code: '1',
    name: '普通销售'
}, {
    id: '2',
    code: '2',
    name: '销售退货'
}]

export const goodsReference = {
    data: [{
        id: '1',
        code: '001',
        name: '牙膏'
    }, {
        id: '2',
        code: '002',
        name: '牙刷'
    }],
    columns: [{
        title: '存货编码',
        name: 'code'
    }, {
        title: '存货名称',
        name: 'name'
    }],
    height: 200,
    width: 300
}
```

开发框架-动态UI-初始化视图



```
//初始化view
export function initView(id) {
    //函数柯里化处理，函数返回值是函数，app middleware会对返回的函数注入app需要的函数（高阶函数） --这块不用太关注，照样写就行
    //injectFuns中包含的常用函数
    //getState:获取状态
    //reducer：执行reduce函数
    //get:ajax get函数
    //post：ajax post函数
    return injectFuns => {
        //调用reducer的initView函数
        //第一个参数包含了元数据，数据
        //第二个参数写死都是一样
        va.initView(api.cardDemo, exports)(injectFuns)
    }
}
```

开发框架-动态UI-加载数据



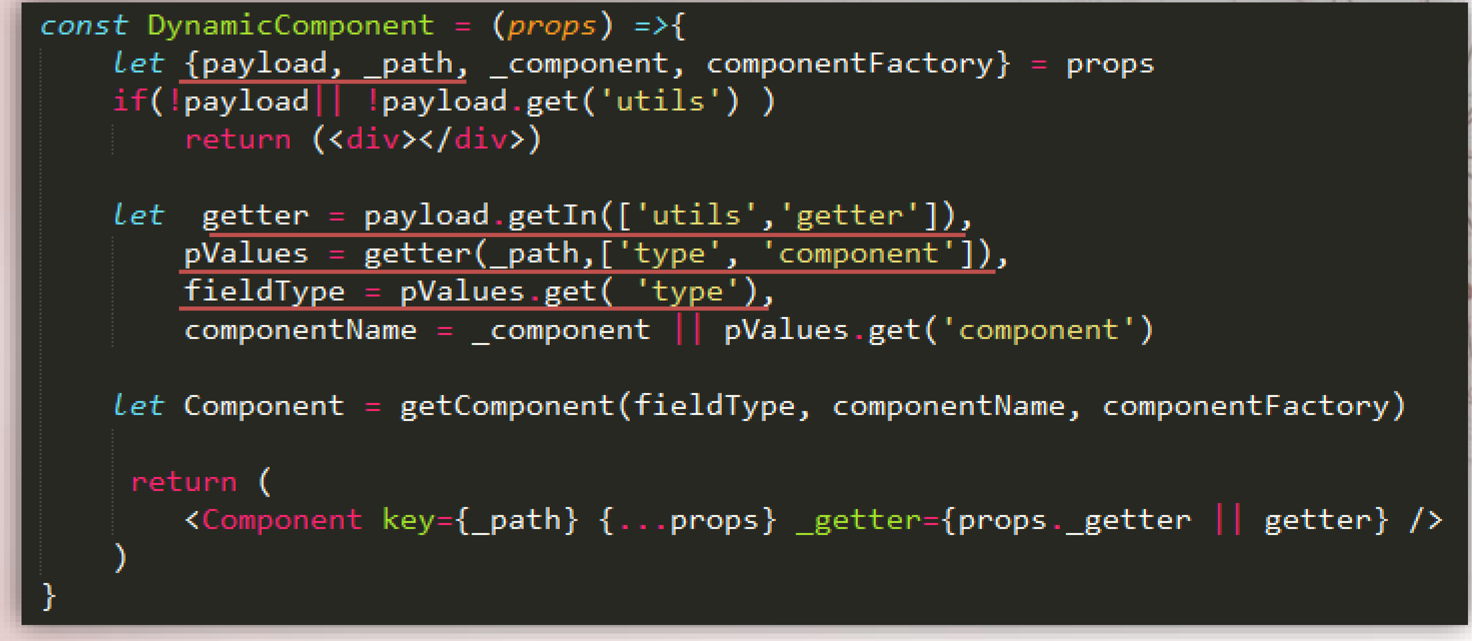
```
//事件处理函数
export function onEvent(eventName, option){
    return (injectFuns) =>{
        //事件名为save
        if(eventName === "save"){
            let a = va.getJson(injectFuns.getState(), 'form')
            alert( JSON.stringify(a))
            //va.validate(injectFuns,'cardDemo.form')
        }else if(eventName === "prevPage"){
            injectFuns.reduce('loadData', 'form', api.cards[0]) //加载数据

        }else if(eventName === "nextPage"){
            injectFuns.reduce('loadData', 'form', api.cards[1])
        }else if(eventName === "add"){
            injectFuns.reduce('loadData', 'form', api.addData)
        }
        else{
            va.onEvent(eventName, option)(injectFuns)
        }
    }
}
```

‘loadData’：reducer的函数名，引用单据action已经默认实现

‘form’:field path,确定要更新哪个路径数据

最后一个参数：value

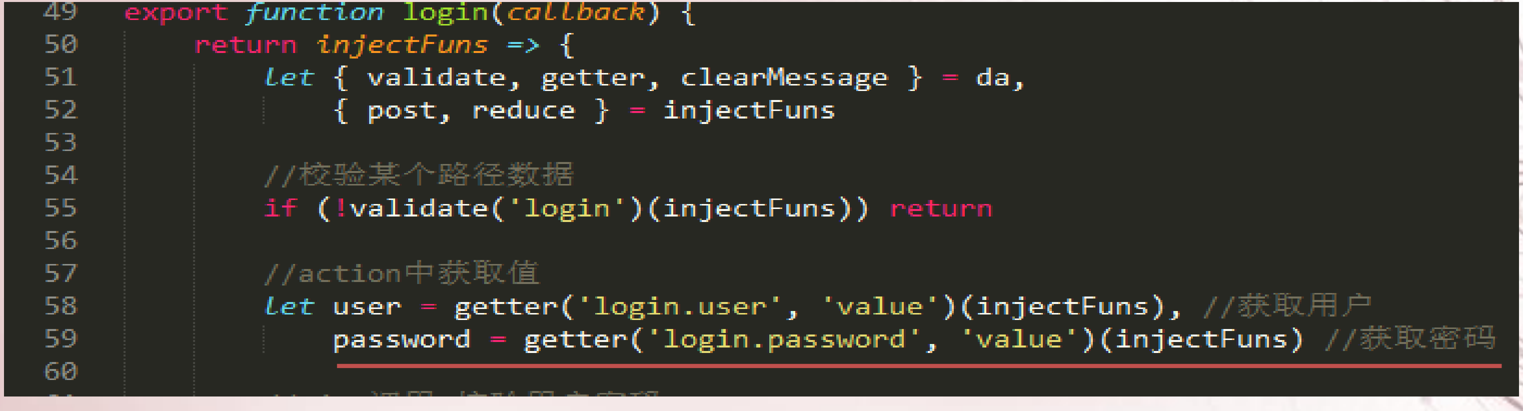
开发框架-动态UI-Component中获取值



```
const DynamicComponent = (props) =>{
    let {payload, _path, _component, componentFactory} = props
    if(!payload|| !payload.get('utils') )
        return (<div></div>)

    let  getter = payload.getIn(['utils','getter']),
        pValues = getter(_path,['type', 'component']),
        fieldType = pValues.get( 'type'),
        componentName = _component || pValues.get('component')

    let Component = getComponent(fieldType, componentName, componentFactory)

     return (
        <Component key={_path} {...props} _getter={props._getter || getter} />
    )
}
```

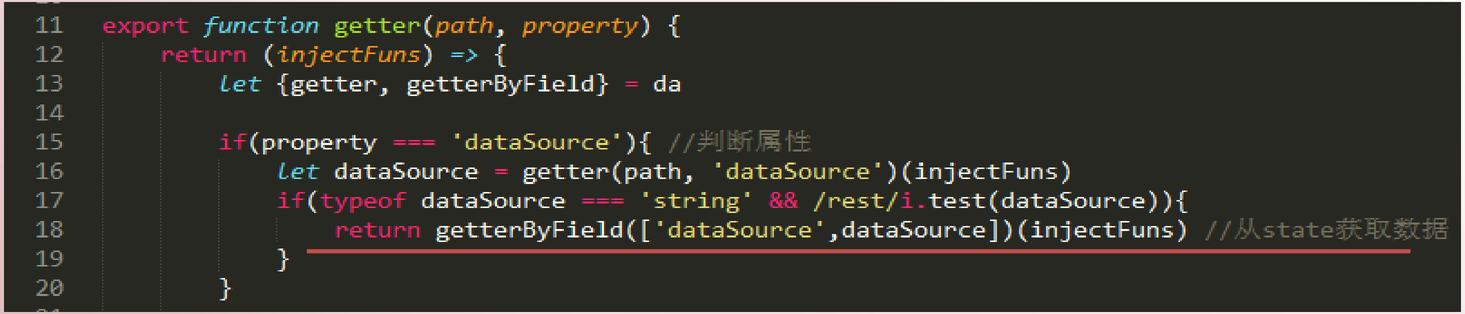
开发框架-动态UI-Action中获取值

通过控件Path



```
export function login(callback) {
    return injectFuns => {
        let { validate, getter, clearMessage } = da,
            { post, reduce } = injectFuns

        //校验某个路径数据
        if (!validate('login')(injectFuns)) return

        //action中获取值
        let user = getter('login.user', 'value')(injectFuns), //获取用户
            password = getter('login.password', 'value')(injectFuns) //获取密码

```

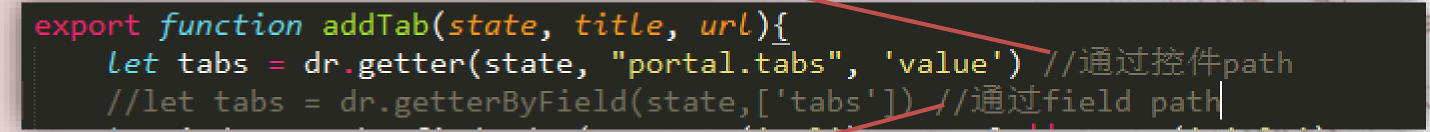
通过Field Path



```
export function getter(path, property) {
    return (injectFuns) => {
        let {getter, getterByField} = da

        if(property === 'dataSource'){ //判断属性
            let dataSource = getter(path, 'dataSource')(injectFuns)
            if(typeof dataSource === 'string' && /rest/i.test(dataSource)){
                return getterByField(['dataSource',dataSource])(injectFuns) //从state获取数据
            }
        }
```

开发框架-动态UI-Reducer中获取值

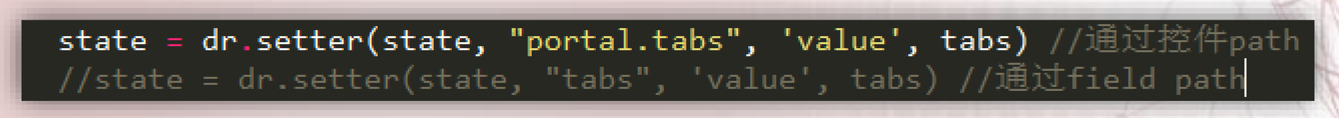
通过控件Path



```
export function addTab(state, title, url){
    let tabs = dr.getter(state, "portal.tabs", 'value') //通过控件path
    //let tabs = dr.getterByField(state,['tabs']) //通过field path
```

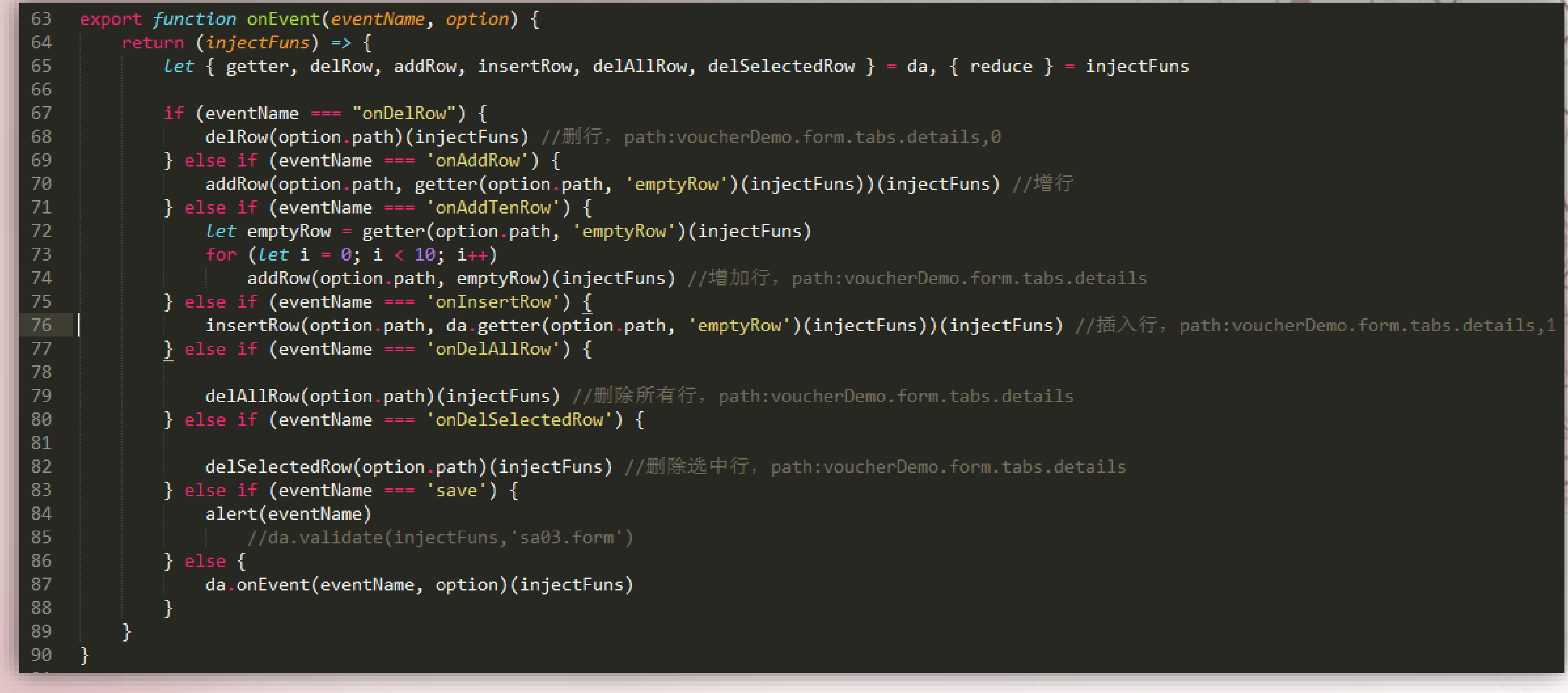
通过Field Path

开发框架-动态UI-Reducer中设置值



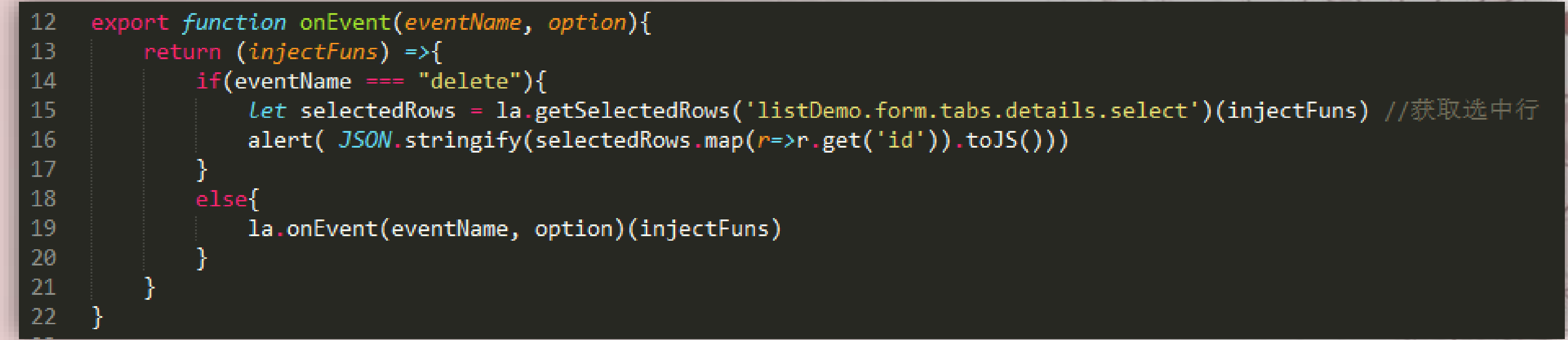
```
state = dr.setter(state, "portal.tabs", 'value', tabs) //通过控件path
//state = dr.setter(state, "tabs", 'value', tabs) //通过field path
```

开发框架-动态UI-明细行操作



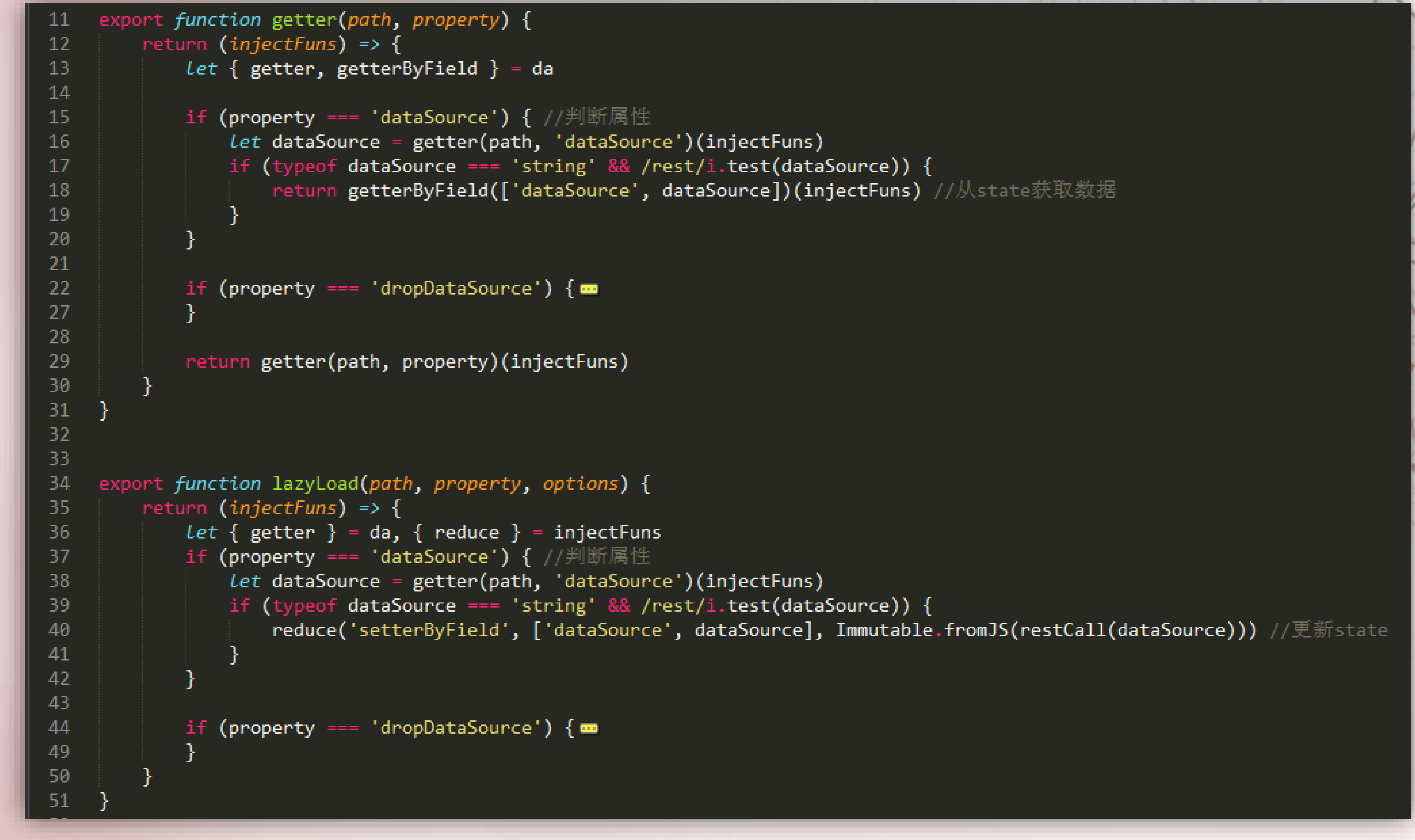
```
export function onEvent(eventName, option) {
    return (injectFuns) => {
        let { getter, delRow, addRow, insertRow, delAllRow, delSelectedRow } = da, { reduce } = injectFuns

        if (eventName === "onDelRow") {
            delRow(option.path)(injectFuns) //删行, path:voucherDemo.form.tabs.details,0
        } else if (eventName === 'onAddRow') {
            addRow(option.path, getter(option.path, 'emptyRow')(injectFuns))(injectFuns) //增行
        } else if (eventName === 'onAddTenRow') {
            let emptyRow = getter(option.path, 'emptyRow')(injectFuns)
            for (let i = 0; i < 10; i++)
                addRow(option.path, emptyRow)(injectFuns) //增加行, path:voucherDemo.form.tabs.details
        } else if (eventName === 'onInsertRow') {
            insertRow(option.path, da.getter(option.path, 'emptyRow')(injectFuns))(injectFuns) //插入行, path:voucherDemo.form.tabs.details,1
        } else if (eventName === 'onDelAllRow') {

            delAllRow(option.path)(injectFuns) //删除所有行, path:voucherDemo.form.tabs.details
        } else if (eventName === 'onDelSelectedRow') {

            delSelectedRow(option.path)(injectFuns) //删除选中行, path:voucherDemo.form.tabs.details
        } else if (eventName === 'save') {
            alert(eventName)
                //da.validate(injectFuns,'sa03.form')
        } else {
            da.onEvent(eventName, option)(injectFuns)
        }
    }
}
```

开发框架-动态UI-获取明细选中行



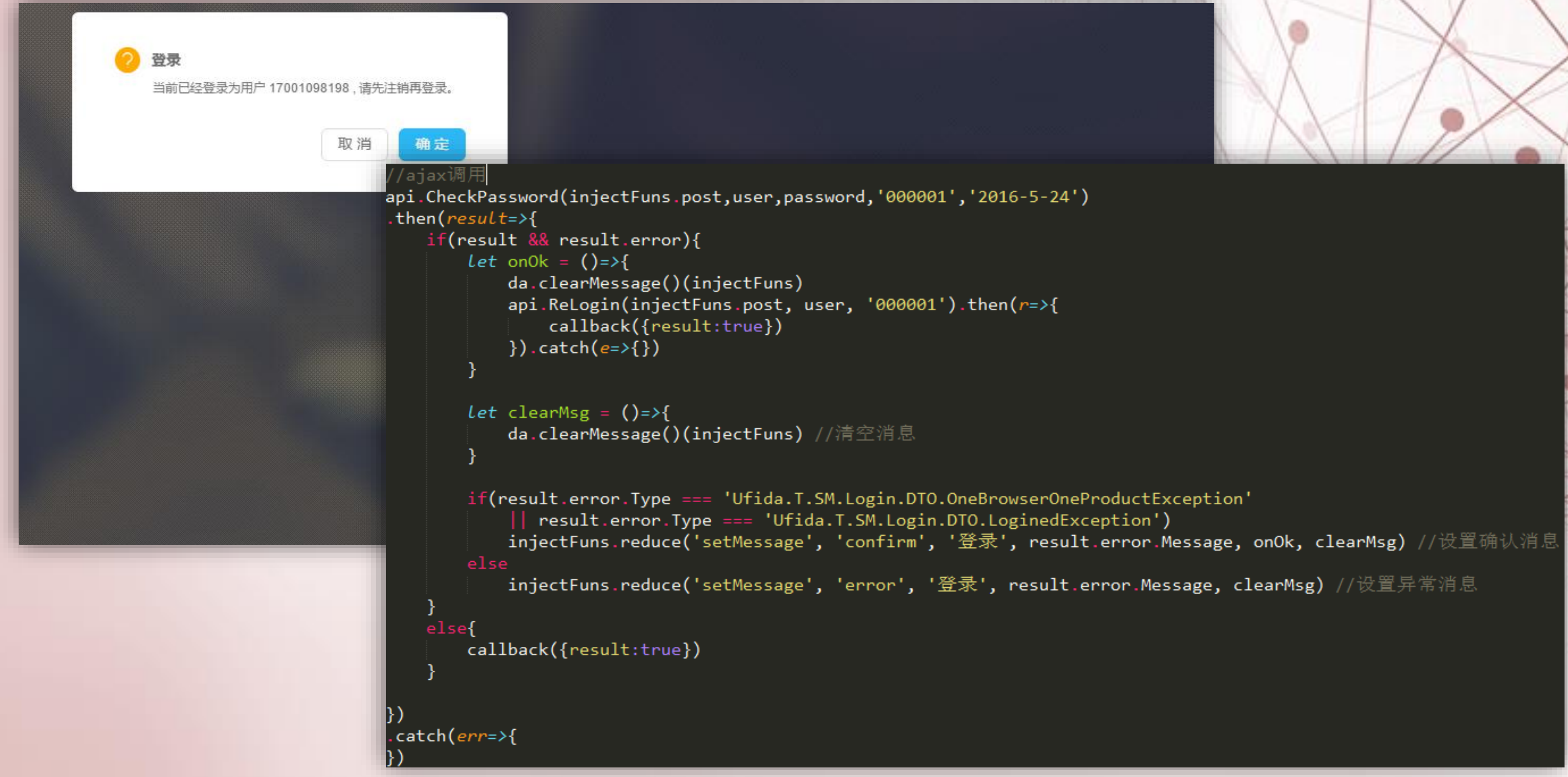
```
export function onEvent(eventName, option){
    return (injectFuns) =>{
        if(eventName === "delete"){
            let selectedRows = la.getSelectedRows('listDemo.form.tabs.details.select')(injectFuns) //获取选中行
            alert( JSON.stringify(selectedRows.map(r=>r.get('id')).toJS()))
        }
        else{
            la.onEvent(eventName, option)(injectFuns)
        }
    }
}
```

开发框架-动态UI-懒加载（例如参照下拉）



```
export function getter(path, property) {
    return (injectFuns) => {
        let { getter, getterByField } = da

        if (property === 'dataSource') { //判断属性
            let dataSource = getter(path, 'dataSource')(injectFuns)
            if (typeof dataSource === 'string' && /rest/i.test(dataSource)) {
                return getterByField(['dataSource', dataSource])(injectFuns) //从state获取数据
            }
        }

        if (property === 'dropDataSource') {⋯
        }

        return getter(path, property)(injectFuns)
    }
}


export function lazyLoad(path, property, options) {
    return (injectFuns) => {
        let { getter } = da, { reduce } = injectFuns
        if (property === 'dataSource') { //判断属性
            let dataSource = getter(path, 'dataSource')(injectFuns)
            if (typeof dataSource === 'string' && /rest/i.test(dataSource)) {
                reduce('setterByField', ['dataSource', dataSource], Immutable.fromJS(restCall(dataSource))) //更新state
            }
        }

        if (property === 'dropDataSource') {⋯
        }
    }
}
```

开发框架-动态UI-设置app消息

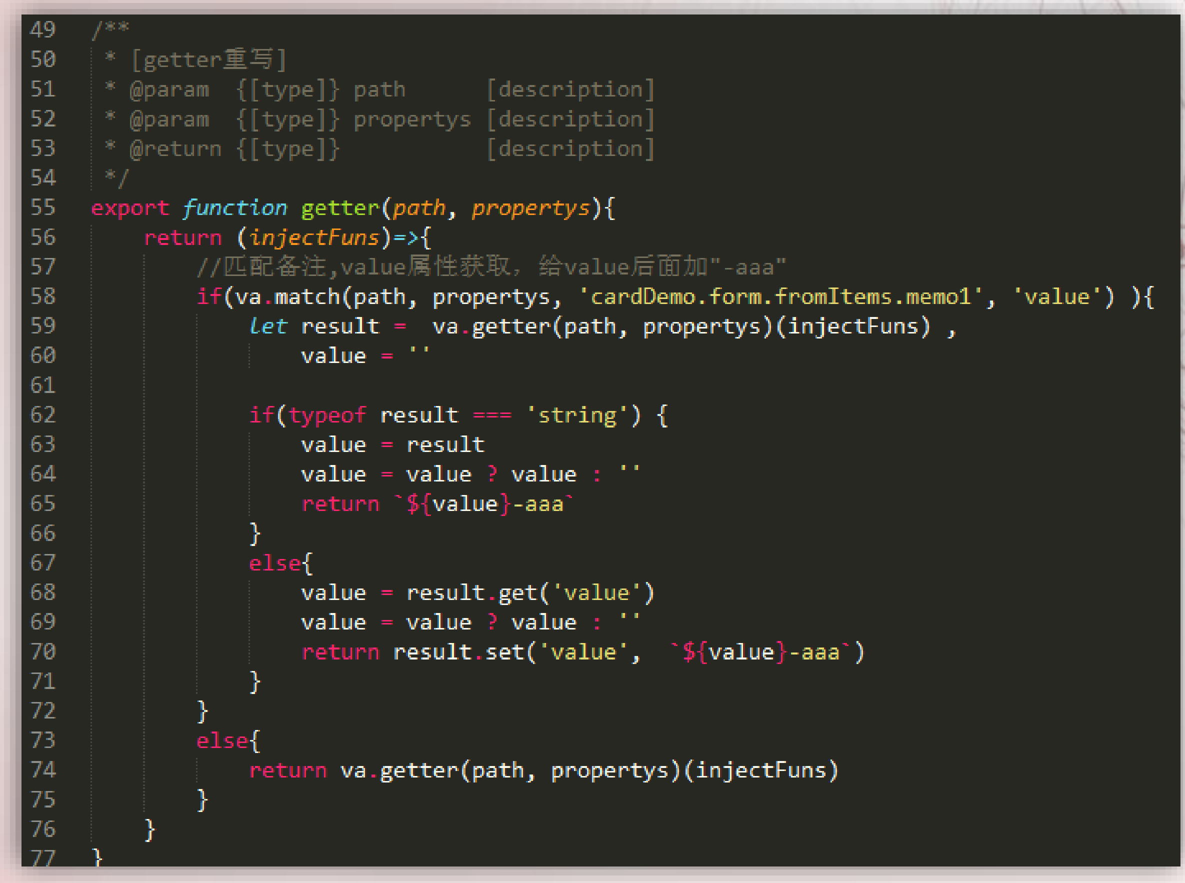


登录

当前已经登录为用户 17001098198，请先注销再登录。

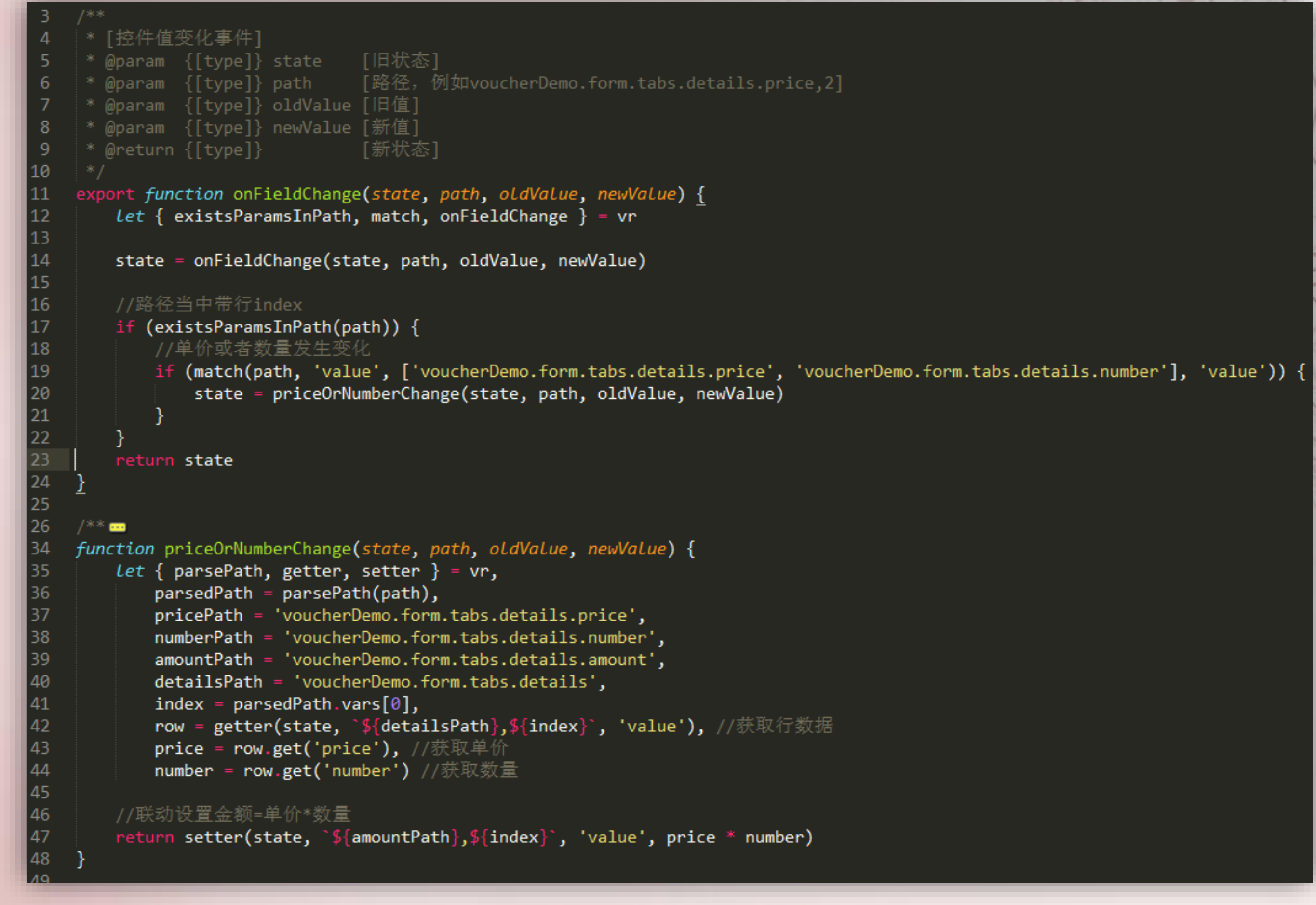
```
//ajax调用
api.CheckPassword(injectFuns.post,user,password,'000001','2016-5-24')
.then(result=>{
    if(result && result.error){
        let onOk = ()=>{
            da.clearMessage()(injectFuns)
            api.ReLogin(injectFuns.post, user, '000001').then(r=>{
                callback({result:true})
            }).catch(e=>{})
        }

        let clearMsg = ()=>{
            da.clearMessage()(injectFuns) //清空消息
        }

        if(result.error.Type === 'Ufida.T.SM.Login.DTO.OneBrowserOneProductException'
            || result.error.Type === 'Ufida.T.SM.Login.DTO.LoginedException')
            injectFuns.reduce('setMessage', 'confirm', '登录', result.error.Message, onOk, clearMsg) //设置确认消息
        else
            injectFuns.reduce('setMessage', 'error', '登录', result.error.Message, clearMsg) //设置异常消息
    }
    else{
        callback({result:true})
    }

})
.catch(err=>{
})
```

开发框架-动态UI-重写getter



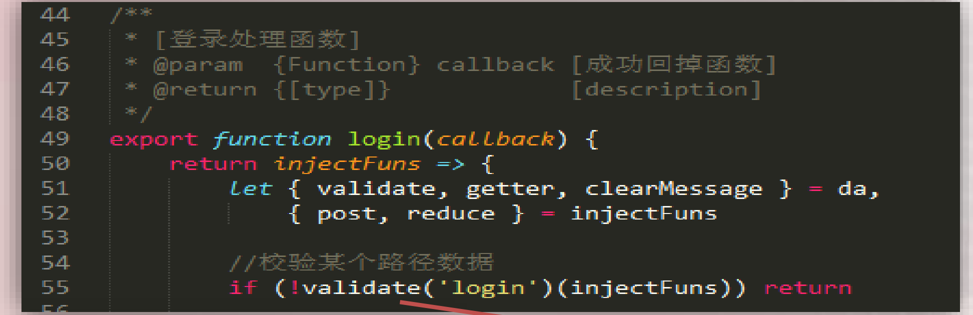
```
/**
 * [getter重写]
 * @param  {[type]} path      [description]
 * @param  {[type]} propertys [description]
 * @return {[type]}           [description]
 */
export function getter(path, propertys){
    return (injectFuns)=>{
        //匹配备注,value属性获取，给value后面加"-aaa"
        if(va.match(path, propertys, 'cardDemo.form.fromItems.memo1', 'value') ){
            let result =  va.getter(path, propertys)(injectFuns) ,
                value = ''

            if(typeof result === 'string') {
                value = result
                value = value ? value : ''
                return `${value}-aaa`
            }
            else{
                value = result.get('value')
                value = value ? value : ''
                return result.set('value',  `${value}-aaa`)
            }
        }
        else{
            return va.getter(path, propertys)(injectFuns)
        }
    }
}
```

开发框架-动态UI-值变化事件

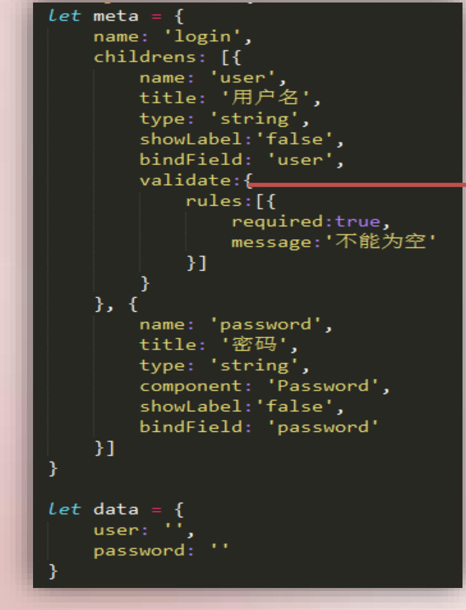


```
/**
 * [控件值变化事件]
 * @param  {[type]} state    [旧状态]
 * @param  {[type]} path     [路径, 例如voucherDemo.form.tabs.details.price,2]
 * @param  {[type]} oldValue [旧值]
 * @param  {[type]} newValue [新值]
 * @return {[type]}          [新状态]
 */
export function onFieldChange(state, path, oldValue, newValue) {
    let { existsParamsInPath, match, onFieldChange } = vr

    state = onFieldChange(state, path, oldValue, newValue)

    //路径当中带行index
    if (existsParamsInPath(path)) {
        //单价或者数量发生变化
        if (match(path, 'value', ['voucherDemo.form.tabs.details.price', 'voucherDemo.form.tabs.details.number'], 'value')) {
            state = priceOrNumberChange(state, path, oldValue, newValue)
        }
    }
    return state
}

/**
function priceOrNumberChange(state, path, oldValue, newValue) {
    let { parsePath, getter, setter } = vr,
        parsedPath = parsePath(path),
        pricePath = 'voucherDemo.form.tabs.details.price',
        numberPath = 'voucherDemo.form.tabs.details.number',
        amountPath = 'voucherDemo.form.tabs.details.amount',
        detailsPath = 'voucherDemo.form.tabs.details',
        index = parsedPath.vars[0],
        row = getter(state, `${detailsPath},${index}`, 'value'), //获取行数据
        price = row.get('price'), //获取单价
        number = row.get('number') //获取数量

    //联动设置金额=单价*数量
    return setter(state, `${amountPath},${index}`, 'value', price * number)
}
```

开发框架-动态UI-校验



```
/**
 * [登录处理函数]
 * @param  {Function} callback [成功回掉函数]
 * @return {[type]}            [description]
 */
export function login(callback) {
    return injectFuns => {
        let { validate, getter, clearMessage } = da,
            { post, reduce } = injectFuns

        //校验某个路径数据
        if (!validate('login')(injectFuns)) return
```

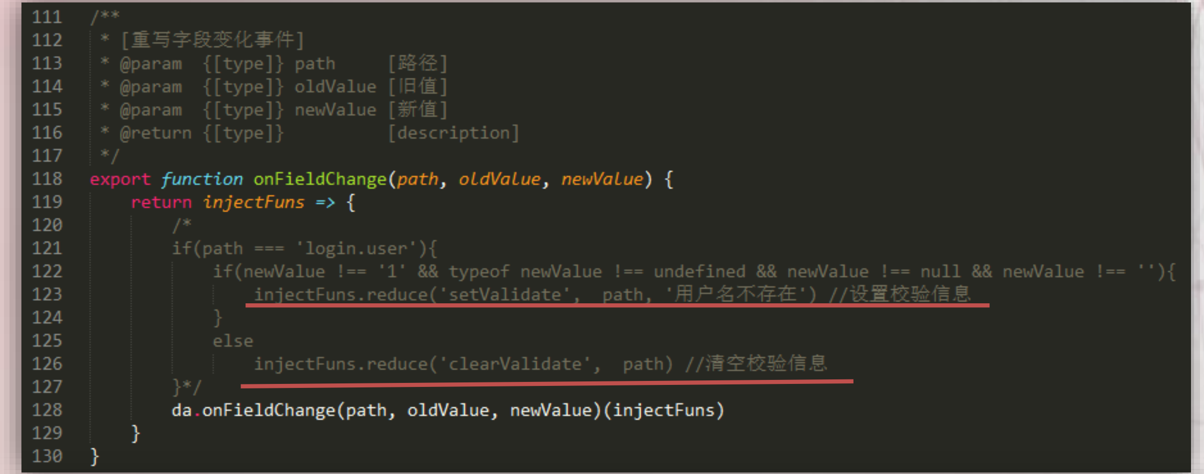


```
let meta = {
    name: 'login',
    childrens: [{
        name: 'user',
        title: '用户名',
        type: 'string',
        showLabel:'false',
        bindField: 'user',
        validate:{
            rules:[{
                required:true,
                message:'不能为空'
            }]
        }
    }, {
        name: 'password',
        title: '密码',
        type: 'string',
        component: 'Password',
        showLabel:'false',
        bindField: 'password'
    }]
}

let data = {
    user: '',
    password: ''
}
```

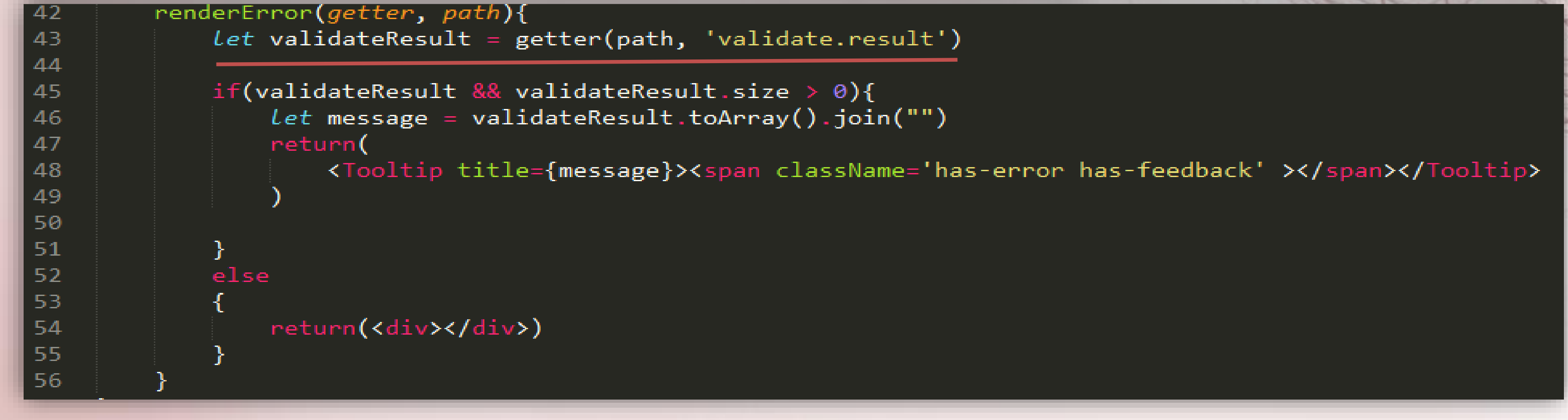
上面的代码会校验用户名不能为空

会根据当前路径，逐级往下校验子元素

开发框架-动态UI-代码设置，获取校验信息

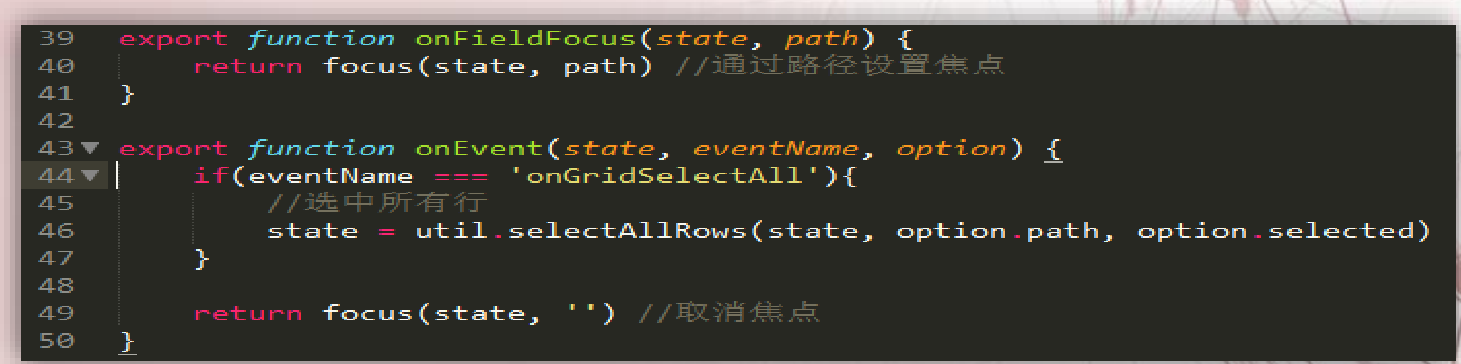


```
/**
 * [重写字段变化事件]
 * @param  {[type]} path     [路径]
 * @param  {[type]} oldValue [旧值]
 * @param  {[type]} newValue [新值]
 * @return {[type]}          [description]
 */
export function onFieldChange(path, oldValue, newValue) {
    return injectFuns => {
        /*
        if(path === 'login.user'){
            if(newValue !== '1' && typeof newValue !== undefined && newValue !== null && newValue !== ''){
                injectFuns.reduce('setValidate',  path, '用户名不存在') //设置校验信息
            }
            else
                injectFuns.reduce('clearValidate',  path) //清空校验信息
        }*/
        da.onFieldChange(path, oldValue, newValue)(injectFuns)
    }
}
```

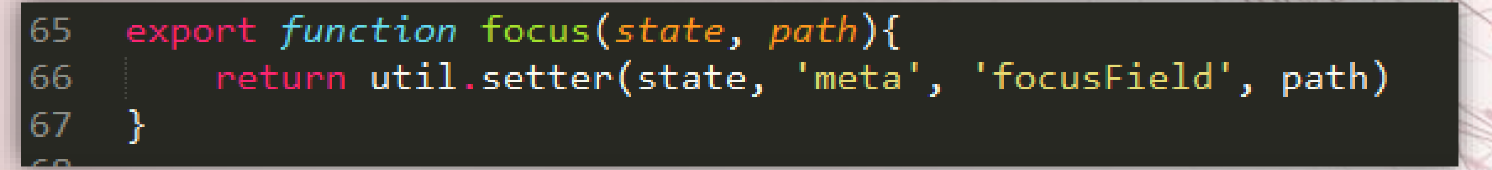


```
    renderError(getter, path){
        let validateResult = getter(path, 'validate.result')

        if(validateResult && validateResult.size > 0){
            let message = validateResult.toArray().join("")
            return(
                <Tooltip title={message}><span className='has-error has-feedback' ></span></Tooltip>
            )

        }
        else
        {
            return(<div></div>)
        }
    }
```

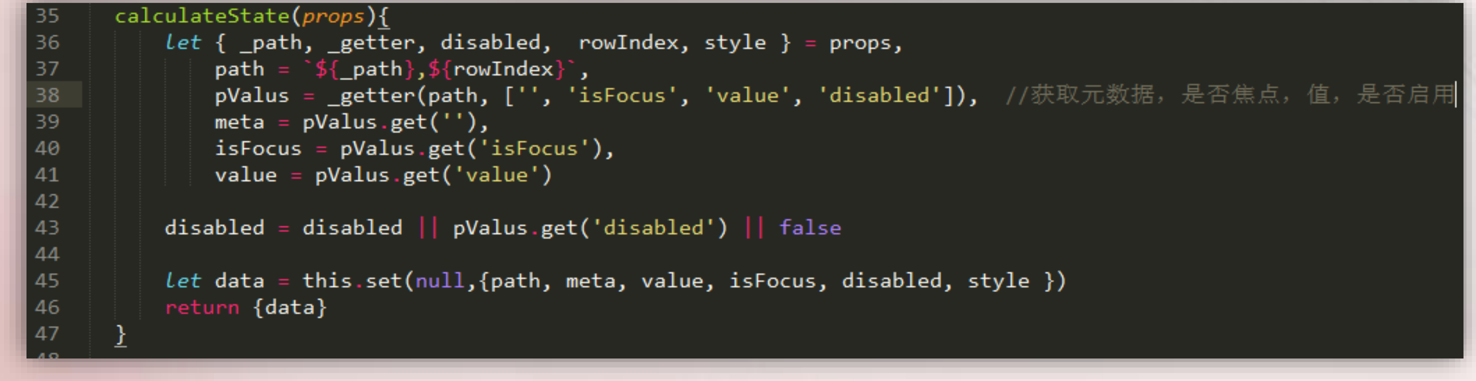
开发框架-动态UI-设置焦点



```
export function onFieldFocus(state, path) {
    return focus(state, path) //通过路径设置焦点
}

export function onEvent(state, eventName, option) {
    if(eventName === 'onGridSelectAll'){
        //选中所有行
        state = util.selectAllRows(state, option.path, option.selected)
    }

    return focus(state, '') //取消焦点
}
```

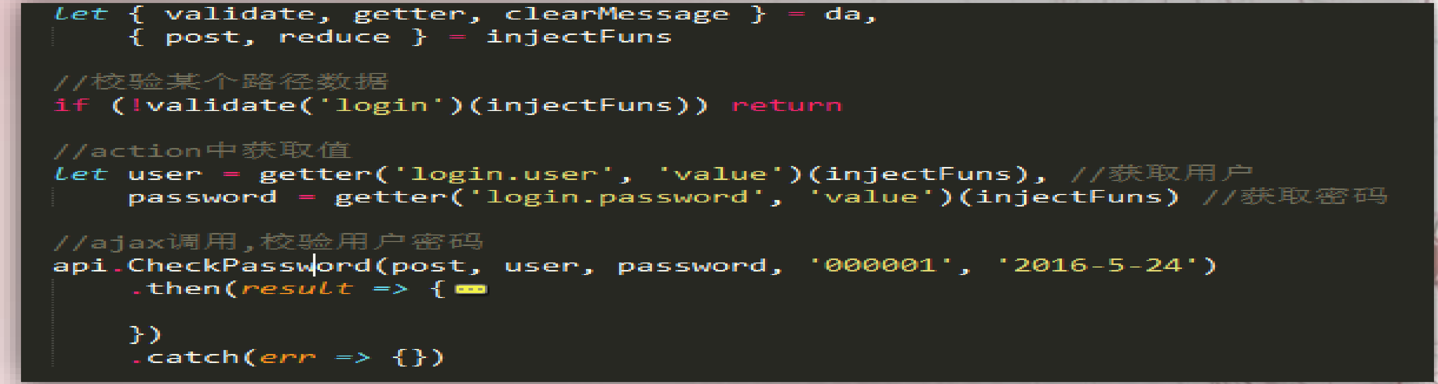


```
export function focus(state, path){
    return util.setter(state, 'meta', 'focusField', path)
}
```

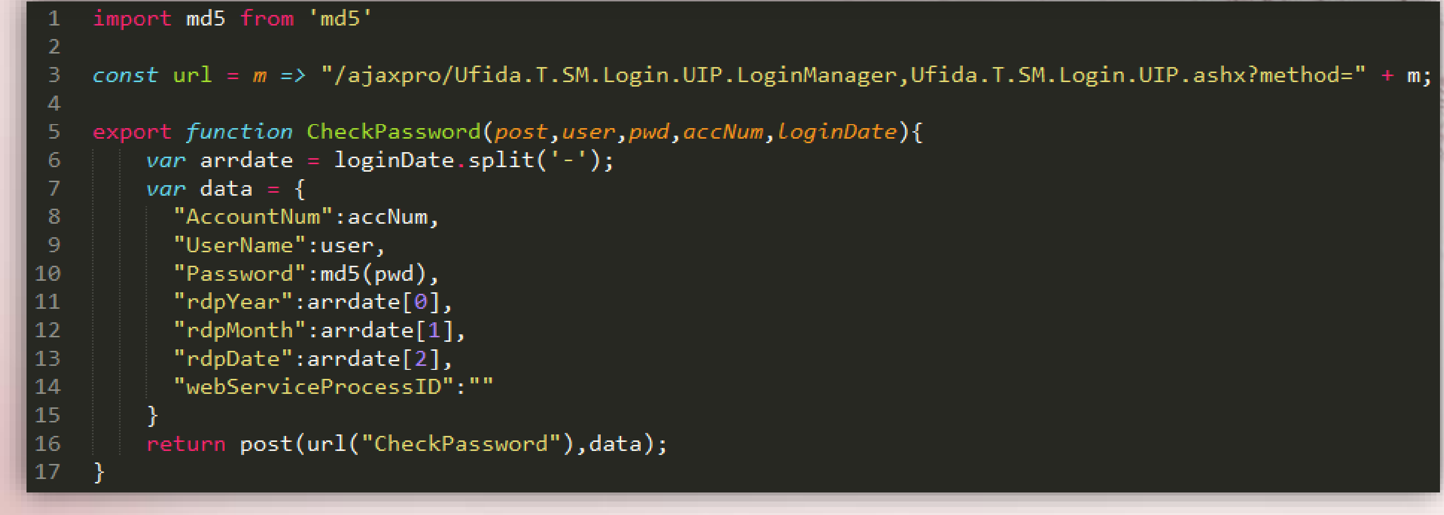


```
calculateState(props){
    let { _path, _getter, disabled,  rowIndex, style } = props,
        path = `${_path},${rowIndex}`,
        pValus = _getter(path, ['', 'isFocus', 'value', 'disabled']),  //获取元数据，是否焦点，值，是否启用
        meta = pValus.get(''),
        isFocus = pValus.get('isFocus'),
        value = pValus.get('value')

    disabled = disabled || pValus.get('disabled') || false

    let data = this.set(null,{path, meta, value, isFocus, disabled, style })
    return {data}
}
```

开发框架-动态UI-Ajax

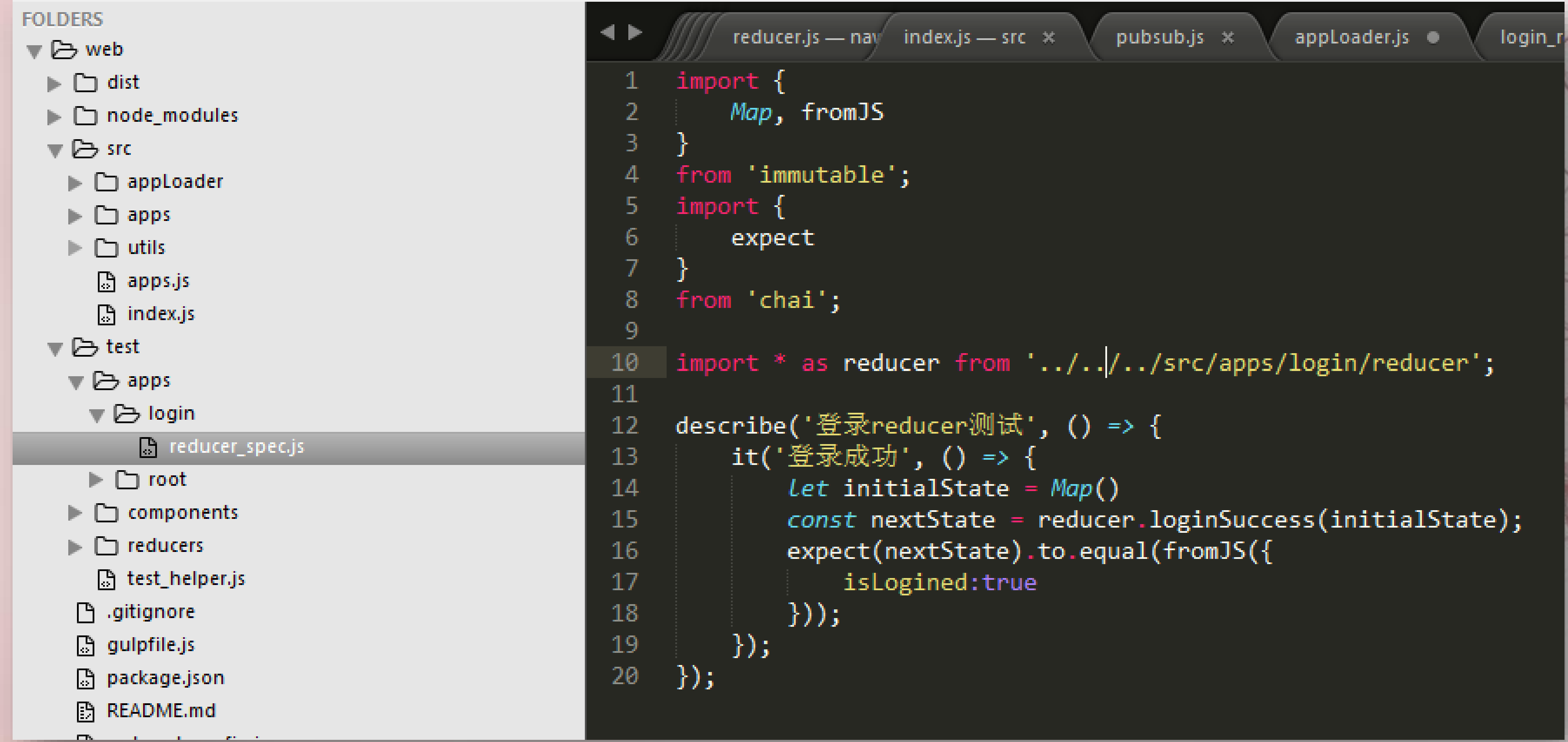


```
let { validate, getter, clearMessage } = da,
    { post, reduce } = injectFuns

//校验某个路径数据
if (!validate('login')(injectFuns)) return

//action中获取值
let user = getter('login.user', 'value')(injectFuns), //获取用户
    password = getter('login.password', 'value')(injectFuns) //获取密码

//ajax调用,校验用户密码
api.CheckPassword(post, user, password, '000001', '2016-5-24')
    .then(result => { …

    })
    .catch(err => {})
```



```
import md5 from 'md5'

const url = m => "/ajaxpro/Ufida.T.SM.Login.UIP.LoginManager,Ufida.T.SM.Login.UIP.ashx?method=" + m;

export function CheckPassword(post,user,pwd,accNum,loginDate){
    var arrdate = loginDate.split('-');
    var data = {
      "AccountNum":accNum,
      "UserName":user,
      "Password":md5(pwd),
      "rdpYear":arrdate[0],
      "rdpMonth":arrdate[1],
      "rdpDate":arrdate[2],
      "webServiceProcessID":""
    }
    return post(url("CheckPassword"),data);
}
```

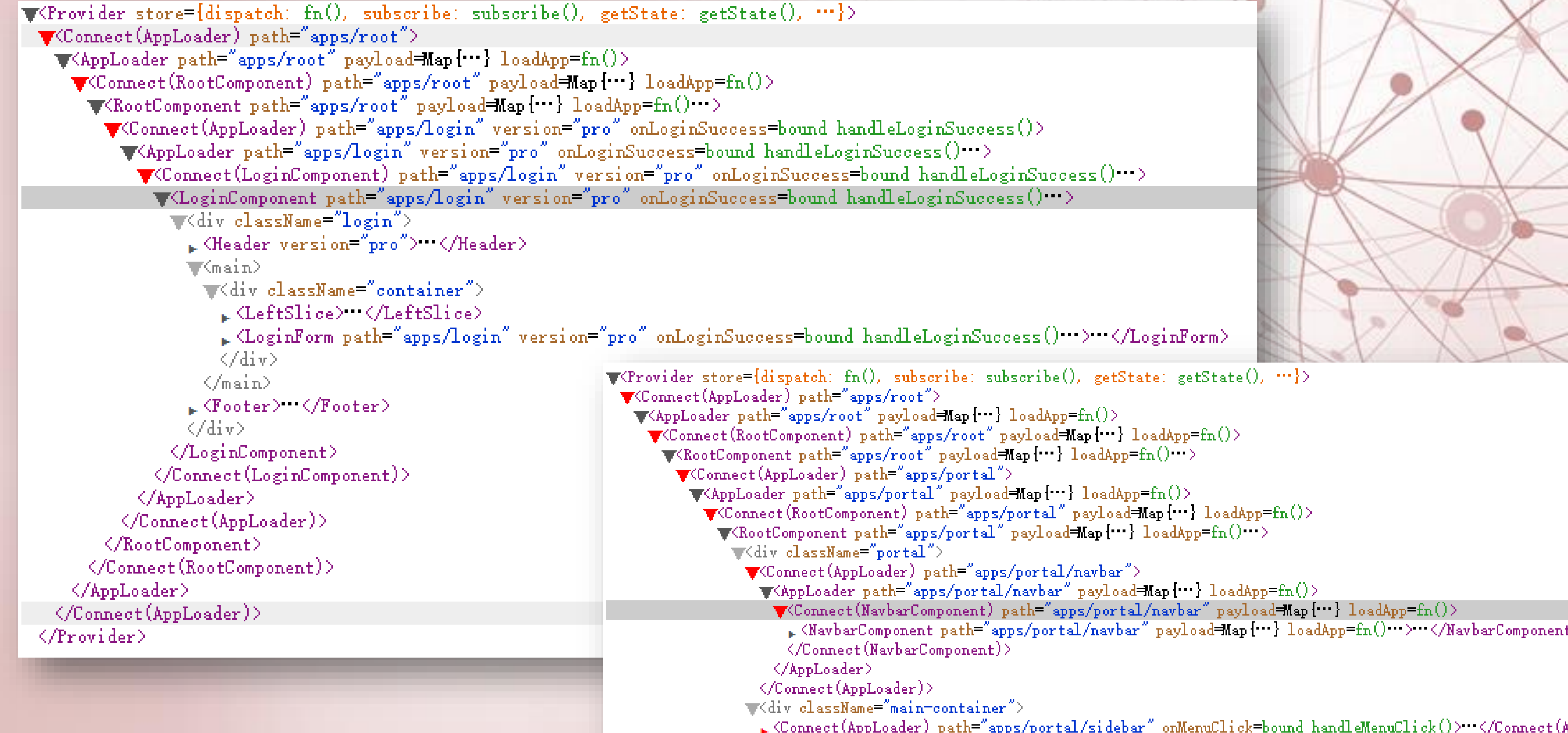
开发框架-单元测试



```
FOLDERS
▼ web
  ▶ dist
  ▶ node_modules
  ▼ src
    ▶ appLoader
    ▶ apps
    ▶ utils
      apps.js
      index.js
  ▼ test
    ▼ apps
      ▼ login
          reducer_spec.js
      ▶ root
    ▶ components
    ▶ reducers
      test_helper.js
    .gitignore
    gulpfile.js
    package.json
    README.md
```

```
import {
    Map, fromJS
}
from 'immutable';
import {
    expect
}
from 'chai';

import * as reducer from '../../../src/apps/login/reducer';

describe('登录reducer测试', () => {
    it('登录成功', () => {
        let initialState = Map()
        const nextState = reducer.loginSuccess(initialState);
        expect(nextState).to.equal(fromJS({
            isLogined:true
        }));
    });
});
```

运行态



```
<Provider store={dispatch: fn(), subscribe: subscribe(), getState: getState(), …}>
  <Connect(AppLoader) path="apps/root">
    <AppLoader path="apps/root" payload=Map{…} loadApp=fn()>
      <Connect(RootComponent) path="apps/root" payload=Map{…} loadApp=fn()>
        <RootComponent path="apps/root" payload=Map{…} loadApp=fn()…>
          <Connect(AppLoader) path="apps/login" version="pro" onLoginSuccess=bound handleLoginSuccess()>
            <AppLoader path="apps/login" version="pro" onLoginSuccess=bound handleLoginSuccess()…>
              <Connect(LoginComponent) path="apps/login" version="pro" onLoginSuccess=bound handleLoginSuccess()…>
                <LoginComponent path="apps/login" version="pro" onLoginSuccess=bound handleLoginSuccess()…>
                  <div className="login">
                    <Header version="pro">…</Header>
                    <main>
                      <div className="container">
                        <LeftSlice>…</LeftSlice>
                        <LoginForm path="apps/login" version="pro" onLoginSuccess=bound handleLoginSuccess()…>…</LoginForm>
                      </div>
                    </main>
                    <Footer>…</Footer>
                  </div>
                </LoginComponent>
              </Connect(LoginComponent)>
            </AppLoader>
          </Connect(AppLoader)>
        </RootComponent>
      </Connect(RootComponent)>
    </AppLoader>
  </Connect(AppLoader)>
</Provider>
```

```
<Provider store={dispatch: fn(), subscribe: subscribe(), getState: getState(), …}>
  <Connect(AppLoader) path="apps/root">
    <AppLoader path="apps/root" payload=Map{…} loadApp=fn()>
      <Connect(RootComponent) path="apps/root" payload=Map{…} loadApp=fn()>
        <RootComponent path="apps/root" payload=Map{…} loadApp=fn()…>
          <Connect(AppLoader) path="apps/portal">
            <AppLoader path="apps/portal" payload=Map{…} loadApp=fn()>
              <Connect(RootComponent) path="apps/portal" payload=Map{…} loadApp=fn()>
                <RootComponent path="apps/portal" payload=Map{…} loadApp=fn()…>
                  <div className="portal">
                    <Connect(AppLoader) path="apps/portal/navbar">
                      <AppLoader path="apps/portal/navbar" payload=Map{…} loadApp=fn()>
                        <Connect(NavbarComponent) path="apps/portal/navbar" payload=Map{…} loadApp=fn()>
                          <NavbarComponent path="apps/portal/navbar" payload=Map{…} loadApp=fn()…>…</NavbarComponent
                        </Connect(NavbarComponent)>
                      </AppLoader>
                    </Connect(AppLoader)>
                    <div className="main-container">
                      <Connect(AppLoader) path="apps/portal/sidebar" onMenuClick=bound handleMenuClick()>…</Connect(A
```