link: null
title: 珠峰架构师成长计划
description: src\index.html
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=189 sentences=886, words=5517

1.搭建开发环境

```
mkdir 2019zfkt
cd 2019zfkt
cnpm init -y
touch .gitignore
```

```
cnpm i react react-dom @types/react @types/react-dom react-router-dom @types/react-router-dom react-transition-group @types/react-transition-group react-swipe
@types/react-swipe antd qs @types/qs -S
cnpm i webpack webpack-cli webpack-dev-server html-webpack-plugin -D
cnpm i typescript ts-loader source-map-loader style-loader css-loader less-loader less url-loader file-loader -D
cnpm i redux react-redux @types/react-redux redux-thunk  redux-logger @types/redux-logger redux-promise @types/redux-promise -S
cnpm i connected-react-router -S
cnpm i express express-session body-parser cors axios -S
```

- `ts-loader`可以让Webpack使用TypeScript的标准配置文件 `tsconfig.json`编译TypeScript代码。
- `source-map-loader`使用任意来自Typescript的 `sourcemap`输出，以此通知webpack何时生成自己的sourcemaps,这让你在调试最终生成的文件时就好像在调试TypeScript源码一样。
- 需要生成一个 `tsconfig.json`文件来告诉 `ts-loader`如何编译代码 `TypeScript`代码

```
tsc --init
```

```
{
  "compilerOptions": {
    "outDir": "./dist",
    "sourceMap": true,
    "noImplicitAny": true,
    "module": "commonjs",
    "target": "es5",
    "jsx": "react",
    "esModuleInterop":true
  },
  "include": [
    "./src/**/*"
  ]
}
```

项目 含义 outDir 指定输出目录 sourceMap 把ts 文件编译成 js 文件的时候，同时生成对应的sourceMap文件 noImplicitAny 如果为true的话，TypeScript 编译器无法推断出类型时，它仍然会生成 JavaScript 文件，但是它也会报告一个错误 module：代码规范 target：转换成es5 jsx react模式会生成React.createElement，在使用前不需要再进行转换操作了，输出文件的扩展名为.js include 需要编译的目录 allowSyntheticDefaultImports 允许从没有设置默认导出的模块中默认导入。这并不影响代码的输出，仅为了类型检查。 esModuleInterop 设置 esModuleInterop: true 使 typescript 来兼容所有模块方案的导入

> 在 TypeScript 中，有多种 import 的方式，分别对应了 JavaScript 中不同的 export

```
import * as xx from 'xx';

import xx from 'xx';
```

- webpack.config.js

```
const webpack=require('webpack');
const HtmlWebpackPlugin=require('html-webpack-plugin');
const path=require('path');
module.exports={
    mode: 'development',
    entry: "./src/index.tsx",
    output: {
        filename: "bundle.js",
        path: path.join(__dirname,'dist')
    },
    devtool: "source-map",
    devServer: {
        hot: true,
        contentBase: path.join(__dirname,'dist'),
        historyApiFallback: {
            index:'./index.html'
        }
    },
    resolve: {
        alias:{
            "~":path.resolve(__dirname,'node_modules')
        },
        extensions: [".ts", ".tsx", ".js", ".json"]
    },

    module: {
        rules: [{
                test: /\.(j|t)sx?$/,
                loader: "ts-loader",
                exclude:/node_modules/
            },
            {
                enforce: "pre",
                test: /\.js$/,
                loader: "source-map-loader"
            },
            {
                test: /\.css$/,
                use:['style-loader','css-loader']
            },
            {
                test: /\.less$/,
                use:['style-loader','css-loader','less-loader']
            },
            {
                test: /\.(gif|svg|png|jpg|jpeg)$/,
                use:['url-loader']
            }
        ]
    },

    plugins: [
        new HtmlWebpackPlugin({
            template:'./src/index.html'
        }),
        new webpack.HotModuleReplacementPlugin()
    ],
};
```

```
import React from 'react';
import ReactDOM from 'react-dom';
ReactDOM.render((
    <h1>helloh1>
),document.getElementById('root'));
```

src\index.html

```

<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <link rel="stylesheet" href="http://img.zhufengpeixun.cn/reset.min.css">
    <title>珠峰课堂title>
head>
<body>
<script>
(function(){
  function refreshRem(){
    document.documentElement.style.fontSize = document.documentElement.clientWidth/750*100+'px';
  }
  refreshRem();
  window.addEventListener('resize',refreshRem,false);
})();
script>
    <div id="root"></div>
body>
html>
```

2.跑通路由

src\index.tsx

```
import React from "react";
import ReactDOM from "react-dom";
import { Switch, Route, Redirect } from "react-router-dom";
import { Provider } from "react-redux";
import store from "./store";
import { ConfigProvider } from "antd";
import zh_CN from "antd/lib/locale-provider/zh_CN";
import "./assets/css/common.less";
import Tabs from "./components/Tabs";
import Home from "./routes/Home";
import Mime from "./routes/Mime";
import Profile from "./routes/Profile";
import { ConnectedRouter } from 'connected-react-router';
import history from './store/history';
ReactDOM.render(
    <Provider store={store}>
        <ConnectedRouter history={history}>
            <ConfigProvider locale={zh_CN}>
                <main className="main-container">
                    <Switch>
                        <Route path="/" exact component={Home} />
                        <Route path="/mime" component={Mime} />
                        <Route path="/profile" component={Profile} />
                        <Redirect to="/" />
                    Switch>
                main>
                <Tabs />
            ConfigProvider>
        ConnectedRouter>
    Provider>,
    document.getElementById("root")
);
```

src\store\index.tsx

```
import { createStore, applyMiddleware, Store, AnyAction } from 'redux';
import reducers from './reducers';
import logger from 'redux-logger';
import thunk from 'redux-thunk';
import promise from 'redux-promise';
import { routerMiddleware } from 'connected-react-router';
import history from './history';
let store: Store = createStore(reducers, applyMiddleware(routerMiddleware(history), promise, thunk));
export default store;
```

src\store\history.tsx

```
import { createHashHistory } from 'history';
export default createHashHistory();
```

src\store\action-types.tsx

```
export const DEMO = 'DEMO';
```

src\store\reducers\index.tsx

```
import { combineReducers,ReducersMapObject } from 'redux';
import home from './home';
import { connectRouter } from 'connected-react-router';
import history from '../history';
let reducers:ReducersMapObject = {
    router: connectRouter(history),
    home
};
type TypeRootState = {
    [key in keyof typeof reducers]: ReturnType<typeof reducers[key]>
}
let reducer: Reducer = combineReducers(reducers);

export { TypeRootState }
export default reducer;
```

src\store\reducers\home.tsx

```
import { AnyAction } from 'redux';
export interface TypeHome {

}
let initialState: TypeHome = {
};
export default function (state: TypeHome = initialState, action: AnyAction): TypeHome {
    switch (action.type) {
        default:
            return state;
    }
}
```

src\typing\common.tsx

```
import { Dispatch, Store } from 'redux';
export interface TypeObject {
    [propName: string]: any
}
export interface TypeAction {
    type: string;
    payload?: any
}
export interface TypeThunkFunction {
    (dispatch: Dispatch, getState: Store['getState']): void
}
```

src\routes\Home\index.tsx

```
import React from 'react';
import { connect } from 'react-redux';
import { RouteComponentProps } from 'react-router-dom';
interface State { }
interface PathParamsType { }
type Props = RouteComponentProps & {
    children?: any
}
class Home extends React.Component<Props, State> {
    render() {
        return (
            <div>
                Home
            div>
        )
    }
}
export default connect(
)(Home);
```

src\routes\Mime\index.tsx

```
import React from 'react';
import { connect } from 'react-redux';
import { RouteComponentProps } from 'react-router-dom';
interface State { }
interface PathParamsType { }
type Props = RouteComponentProps & {
    children?: any
}
class Mime extends React.Component<Props, State> {
    render() {
        return (
            <div>
                Mime
            div>
        )
    }
}
export default connect(

)(Mime);
```

src\routes\Profile\index.tsx

```
import React from 'react';
import { connect } from 'react-redux';
class Profile extends React.Component {
    render() {
        return (
            <div>
                Profile
            div>
        )
    }
}
export default connect()(Profile);
```

src\components\Tabs\index.tsx

```
import React from "react";
import { RouteComponentProps, withRouter, NavLink } from 'react-router-dom';
import { Icon } from 'antd';
import './index.less';
interface State { }
interface PathParamsType { }
type Props = RouteComponentProps & {
    children?: any
}
class Tabs extends React.Component<Props, State> {
    constructor(props: Props) {
        super(props);
    }
    render() {
        return (
            <footer>
                <NavLink exact to="/"><Icon type="home" /><span>首页span>NavLink>
                <NavLink to="/mime"><Icon type="solution" /><span>我的课程span>NavLink>
                <NavLink to="/profile"><Icon type="user" /><span>个人中心span>NavLink>
            footer>
        );
    }
}
export default withRouter(Tabs);
```

src\assets\css\common.less

```
@import "~antd/dist/antd.css";
html{
    font-size:100px;
}
#root{
    margin: 0 auto;
    max-width:750px;
    box-sizing: border-box;
}
.main-container{
    padding:1rem 0 1.2rem 0;
}
```

3.首页头部导航

routes\Home\index.tsx

```
import React from 'react';
import { connect } from 'react-redux';
import HomeHeader from './HomeHeader';
import actions from '../../store/actions/home';
import { TypeRootState } from '../../store/reducers';
import { TypeHome } from '../../store/reducers/home';
import { RouteComponentProps } from 'react-router-dom';
import './index.less';
type StateProps = ReturnType<typeof mapStateToProps>;
type DispatchProps = typeof actions;
interface PathParamsType { }
type Props = RouteComponentProps & StateProps & DispatchProps & {
    children: any
};
class Home extends React.Component<Props>{
    constructor(props: Props) {
        super(props);
    }
    render() {
        return (
            <>
                <HomeHeader currentCategory={this.props.currentCategory}
                    setCurrentCategory={this.props.setCurrentCategory}
                />
            </>
        )
    }
}
let mapStateToProps = (state: TypeRootState): TypeHome => state.home;
export default connect(
    mapStateToProps,
    actions
)(Home);
```

src\store\action-types.tsx

```
export const SET_CURRENT_CATEGORY = 'SET_CURRENT_CATEGORY';
```

store\reducers\home.tsx

```
import * as TYPES from "../action-types";
import { TypeAction } from '../../typing/common';
export interface TypeHome {
    currentCategory: string;
}
let initialState: TypeHome = {
    currentCategory: 'all'
};
export default function (state: TypeHome = initialState, action: TypeAction): TypeHome {
    switch (action.type) {
        case TYPES.SET_CURRENT_CATEGORY:
            return { ...state, currentCategory: action.payload };
        default:
            return state;
    }
}
```

routes\Home\index.less

```
.home-container{
    position: fixed;
    top:1rem;
    left:0;
    width:100%;
    overflow-y: auto;
    height:calc(100vh - 2.22rem);
}
```

store\actions\home.tsx

```
import * as TYPES from '../action-types';
import { TypeAction } from '../../typing/common';
export default {
    setCurrentCategory(currentCategory: string): TypeAction {
        return { type: TYPES.SET_CURRENT_CATEGORY, payload: currentCategory };
    }
}
```

routes\Home\HomeHeader\index.tsx

```
import React from "react";
import { RouteComponentProps, withRouter } from 'react-router-dom';
import { Icon } from 'antd';
import { Transition } from 'react-transition-group';
import './index.less';
const duration = 300;
const defaultStyle: React.CSSProperties = {
  transition: `all ${duration}ms ease-in-out`,
  opacity: 0,
  display: 'none'
}
interface TypeTransitionStyles {
  entering: React.CSSProperties;
  entered: React.CSSProperties;
  exiting: React.CSSProperties;
  exited: React.CSSProperties;
  unmounted: React.CSSProperties;
}
const transitionStyles: TypeTransitionStyles = {
  entering: { opacity: 0, display: 'none' },
  entered: { opacity: 1, display: 'block' },
  exiting: { opacity: 0, display: 'none' },
  exited: { opacity: 0, display: 'none' },
  unmounted: { opacity: 0, display: 'none' }
}
interface PathParamsType { }
type Props = RouteComponentProps & {
  children?: any,
  currentCategory?: any,
  setCurrentCategory?: any;
  refreshLessons?: any
}
interface State {
  in?: boolean
}
class HomeHeader extends React.Component<Props, State>{
  state = { in: false }
  constructor(props: Props) {
    super(props);
  }
  setCurrentCategory = (event: React.MouseEvent) => {
    let target: EventTarget = event.target;
    let category = (target as HTMLUListElement).dataset.category;
    this.props.setCurrentCategory(category);
    this.setState({ in: false }, this.props.refreshLessons);
  }
  render() {
    return (
      <header className="home-header">
        <div className="logo-header">
          <img src="http://img.zhufengpeixun.cn/zfkelogo.png" />
          <Icon type="bars" onClick={event => this.setState({ in: !this.state.in })} />
        div>
        <Transition in={this.state.in} timeout={300}>
          {
            state => (
              <ul className="category" style={{
                ...defaultStyle,
                ...transitionStyles[state]
              }}
                onClick={this.setCurrentCategory}
              >
                <li data-category="all" className={this.props.currentCategory == 'all' ? 'active' : ''}>全部课程li>
                <li data-category="react" className={this.props.currentCategory == 'react' ? 'active' : ''}>React课程li>
                <li data-category="vue" className={this.props.currentCategory == 'vue' ? 'active' : ''}>Vue课程li>
              ul>
            )
          }
        Transition>
      header>
    );
  }
}
export default withRouter(HomeHeader);
```

src\routes\Home\HomeHeader\index.less

```
@BG: #2a2a2a;
.home-header {
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  z-index: 999;
  .logo-header {
    height: 1rem;
    background: @BG;
    color: #fff;
    display: flex;
    justify-content: space-between;
    align-items: center;
    img {
      width: 2rem;
      margin-left: 0.2rem;
    }
    i {
      font-size: 0.6rem;
      margin-right: 0.2rem;
    }
  }
  .category {
    position: absolute;
    width: 100%;
    top: 1rem;
    left: 0;
    padding: 0.1rem 0.5rem;
    background: @BG;
    li {
      line-height: 0.6rem;
      text-align: center;
      color: #fff;
      font-size: 0.3rem;
      border-top: 0.02rem solid lighten(@BG, 20%);
      &.active{
        color: red;
      }
    }
  }
}
```

4.个人中心

src\routes\Profile\index.tsx

```
import React from 'react';
import { connect } from 'react-redux';
import { TypeRootState } from '../../store/reducers';
import { TypeProfile } from '../../store/reducers/profile';
import actions from '../../store/actions/profile';
import LOGIN_TYPES from '../../typings/login-types';
import { RouteComponentProps } from 'react-router';
import { Descriptions, Button, Alert } from 'antd';
import NavHeader from '../../components/NavHeader';
import './index.less';
interface State {

}

type StateProps = ReturnType<typeof mapStateToProps>;
type DispatchProps = typeof actions;
interface IParams { }
type RouteProps = RouteComponentProps;
type Props = StateProps & DispatchProps & RouteProps & {
    children?: any
}
class Profile extends React.Component<Props, State> {
    async componentDidMount() {
        if (!this.props.user) {
            await this.props.validate();
        }
    }
    render() {
        let content;
        if (this.props.loginState == LOGIN_TYPES.UN_VALIDATE) {
            content = null;
        } else if (this.props.loginState == LOGIN_TYPES.LOGINED) {
            content = (
                <div className="user-info">
                    <Descriptions title="当前登录用户">
                        <Descriptions.Item label="用户名">珠峰架构Descriptions.Item>
                        <Descriptions.Item label="手机号">15718856132Descriptions.Item>
                        <Descriptions.Item label="邮箱">zhangsan@qq.comDescriptions.Item>
                    Descriptions>
                    <Button type="danger">退出登录Button>
                div>
            )
        } else {
            content = (
                <>
                    <Alert type="warning" message="当前未登录" description="亲爱的用户你好，你当前尚未登录，请你选择注册或者登录" />
                    <div style={{ textAlign: 'center', padding: '.5rem' }}>
                        <Button type="dashed" onClick={() => this.props.history.push('/login')}>登录Button>
                        <Button type="dashed" style={{ marginLeft: '.5rem' }} onClick={() => this.props.history.push('/register')}>注册Button>
                    div>
                </>
            )
        }
        return (
            (
                <section>
                    <NavHeader history={this.props.history}>个人中心NavHeader>
                    {content}
                section>
            )
        )
    }
}
let mapStateToProps = (state: TypeRootState): TypeProfile => state.profile
export default connect(
    mapStateToProps,
    actions
)(Profile);
```

src\routes\Profile\index.less

```
.user-info{
    padding:.2rem;
}
```

src\store\action-types.tsx

```
export const SET_CURRENT_CATEGORY = 'SET_CURRENT_CATEGORY';

+export const VALIDATE = 'VALIDATE';
+export const LOGOUT = 'LOGOUT';
```

src\store\reducers\index.tsx

```
import { combineReducers, Reducer, ReducersMapObject, AnyAction } from 'redux';
import { connectRouter } from 'connected-react-router';
import history from '../history';
import home from './home';
import mine from './mine';
+import profile from './profile';
+let reducers: ReducersMapObject = {
    home,
    mine,
+    profile,
    router: connectRouter(history)
}
export type TypeRootState = {
    [key in keyof typeof reducers]: ReturnType
}
+let reducer: Reducer = combineReducers(reducers);

export default reducer;
```

src\api\index.tsx

```
import axios from "axios";
import qs from "qs";
axios.defaults.baseURL = "http://localhost:8000";
axios.defaults.withCredentials = true;
axios.defaults.transformRequest = (data = {}) => qs.stringify(data);
axios.interceptors.response.use(result => result.data);
export default axios;
```

src\api\profile.tsx

```
import axios from './index';
export function validate() {
    return axios.get('/validate');
}
```

components\NavHeader\index.tsx

```
import React from 'react';
import './index.less';
import { Icon } from 'antd';
interface Props {
    history: any;
    children: any
}
export default function NavHeader(props: Props) {
    return (
        <div className="nav-header">
            <Icon type="left" onClick={() => props.history.goBack()} />
            {props.children}
        div>
    )
}
```

components\NavHeader\index.less

```
.nav-header{
    position: fixed;
    left:0;
    top:0;
    height:1rem;
    z-index: 1000;
    width:100%;
    box-sizing: border-box;
    text-align: center;
    line-height: 1rem;
    background-color: #2A2A2A;
    color:#FFF;
    i{
        position: absolute;
        left:.2rem;
        line-height: 1rem;
    }
}
```

src\typings\login-types.tsx

```
enum LOGIN_TYPES {
    UN_VALIDATE,
    LOGINED,
    UNLOGIN
}
export default LOGIN_TYPES;
```

store\actions\profile.tsx

```
import { AnyAction } from 'redux';
import * as TYPES from '../action-types';
import { validate } from '../../api/profile';
export default {
    validate(): AnyAction {
        return {
            type: TYPES.VALIDATE,
            payload: validate()
        }
    },
}
```

src\store\reducers\profile.tsx

```
import { AnyAction } from 'redux';
import * as TYPES from "../action-types";
import LOGIN_TYPES from '../../typings/login-types';
export interface TypeProfile {
    loginState: LOGIN_TYPES,
    user: any,
    error: string | null
}
let initialState: TypeProfile = {
    loginState: LOGIN_TYPES.UN_VALIDATE,
    user: null,
    error: null
}
export default function (state: TypeProfile = initialState, action: AnyAction): TypeProfile {
    switch (action.type) {
        case TYPES.VALIDATE:
            let { code, data, error } = action.payload;
            if (code == 0) {
                return {
                    ...state,
                    loginState: LOGIN_TYPES.LOGINED,
                    user: data,
                    error: null
                };
            } else {
                return { ...state, loginState: LOGIN_TYPES.UNLOGIN, user: null, error };
            }
        default:
            return state;
    }
}
```

5.注册登录

src\routes\Register\index.tsx

```
import React from "react";
import { connect } from "react-redux";
import actions from "../../store/actions/profile";
import { RouteComponentProps, Link } from "react-router-dom";
import NavHeader from "../../components/NavHeader";
import { Form, Icon, Input, Button, message } from "antd";
import { TypeRootState } from '../../store/reducers';
import { TypeProfile } from '../../store/reducers/profile';
import "./index.less";
type StateProps = ReturnType<typeof mapStateToProps>;
type DispatchProps = typeof actions;
interface PathParamsType { }
type Props = RouteComponentProps & StateProps & DispatchProps & {
    children?: any;
    form: any;
};

class Register extends React.Component<Props> {
    handleSubmit = (event: React.FormEvent) => {
        event.preventDefault();
        this.props.form.validateFields(async (err: any, values: any) => {
            if (err) {
                message.error('表单验证失败!');
            } else {
                this.props.register(values);
            }
        });
    };
    render() {
        const { getFieldDecorator } = this.props.form;
        return (
            <>
                <NavHeader history={this.props.history}>用户注册NavHeader>
                <Form onSubmit={this.handleSubmit} className="login-form">
                    <Form.Item>
                        {getFieldDecorator("username", {
                            rules: [{ required: true, message: "请输入你的用户名!" }]
                        })(
                            <Input
                                prefix={
                                    <Icon type="user" style={{ color: "rgba(0,0,0,.25)" }} />
                                }
                                placeholder="用户名"
                            />
                        )}
                    Form.Item>
                    <Form.Item>
                        {getFieldDecorator("password", {
                            rules: [{ required: true, message: "请输入你的密码!" }]
                        })(
                            <Input
                                prefix={
                                    <Icon type="lock" style={{ color: "rgba(0,0,0,.25)" }} />
                                }
                                type="password"
                                placeholder="密码"
                            />
                        )}
                    Form.Item>
                    <Form.Item>
                        {getFieldDecorator("email", {
                            rules: [{ required: true, message: "请输入你的邮箱!" }]
                        })(
                            <Input
                                prefix={
                                    <Icon type="mail" style={{ color: "rgba(0,0,0,.25)" }} />
                                }
                                type="email"
                                placeholder="邮箱"
                            />
                        )}
                    Form.Item>
                    <Form.Item>
                        {getFieldDecorator("phone", {
                            rules: [{ required: true, message: "请输入你的手机号!" }]
                        })(
                            <Input
                                prefix={
                                    <Icon type="phone" style={{ color: "rgba(0,0,0,.25)" }} />
                                }
                                type="text"
                                placeholder="手机号"
                            />
                        )}
                    Form.Item>
                    <Form.Item>
                        <Button
                            type="primary"
                            htmlType="submit"
                            className="login-form-button"
                        >
                            注册
            Button>
                        或者 <Link to="/login">立刻登录!Link>
                    Form.Item>
                Form>
            </>
        );
    }
}
const WrappedRegister = Form.create({ name: "login" })(Register);
let mapStateToProps = (state: TypeRootState): TypeProfile => state.profile;
export default connect(mapStateToProps, actions)(WrappedRegister);
```

index.less

```
.login-form{
    padding:.2rem;
}
```

routes\Login\index.tsx

```
import React from "react";
import { connect } from "react-redux";
import actions from "../../store/actions/profile";
import { Link, RouteComponentProps } from "react-router-dom";
import NavHeader from "../../components/NavHeader";
import { Form, Icon, Input, Button, message } from "antd";
import "./index.less";
import { TypeRootState } from '../../store/reducers';
import { TypeProfile } from '../../store/reducers/profile';
type StateProps = ReturnType<typeof mapStateToProps>;
type DispatchProps = typeof actions;
interface PathParamsType { }
type Props = RouteComponentProps & StateProps & DispatchProps & {
    children?: any;
    form: any;
};
class Register extends React.Component<Props> {
    handleSubmit = (event: React.FormEvent) => {
        event.preventDefault();
        this.props.form.validateFields(async (err: any, values: any) => {
            if (err) {
                message.error('表单验证失败!');
            } else {
                this.props.login(values);
            }
        });
    };
    render() {
        const { getFieldDecorator } = this.props.form;
        return (
            <>
                <NavHeader history={this.props.history}>用户登录NavHeader>
                <Form onSubmit={this.handleSubmit} className="login-form">
                    <Form.Item>
                        {getFieldDecorator("username", {
                            rules: [{ required: true, message: "请输入你的用户名!" }]
                        })(
                            <Input
                                prefix={
                                    <Icon type="user" style={{ color: "rgba(0,0,0,.25)" }} />
                                }
                                placeholder="用户名"
                            />
                        )}
                    Form.Item>
                    <Form.Item>
                        {getFieldDecorator("password", {
                            rules: [{ required: true, message: "请输入你的密码!" }]
                        })(
                            <Input
                                prefix={
                                    <Icon type="lock" style={{ color: "rgba(0,0,0,.25)" }} />
                                }
                                type="password"
                                placeholder="密码"
                            />
                        )}
                    Form.Item>
                    <Form.Item>
                        <Button
                            type="primary"
                            htmlType="submit"
                            className="login-form-button"
                        >
                            登录
            Button>
                        或者 <Link to="/register">立刻注册!Link>
                    Form.Item>
                Form>
            </>
        );
    }
}
const WrappedRegister = Form.create({ name: "login" })(Register);
const mapStateToProps = (state: TypeRootState): TypeProfile => state.profile;
export default connect(mapStateToProps, actions)(WrappedRegister);
```

index.less

```
.login-form{
    padding:.2rem;
}
```

src\routes\Profile\index.tsx

```
import React from 'react';
import { connect } from 'react-redux';
import { TypeRootState } from '../../store/reducers';
import { TypeProfile } from '../../store/reducers/profile';
import actions from '../../store/actions/profile';
import LOGIN_TYPES from '../../typings/login-types';
import { RouteComponentProps } from 'react-router';
+import { Descriptions, Button, Alert, Upload, Icon, message } from 'antd';
import NavHeader from '../../components/NavHeader';
import './index.less';
interface State {
+    loading: boolean,
+    imageUrl?: string
}
//当前的组件有三个属性来源
//1.mapStateToProps的返回值 2.actions对象类型 3. 来自路由 4.用户传入进来的其它属性
type StateProps = ReturnType;
type DispatchProps = typeof actions;
interface IParams { }
type RouteProps = RouteComponentProps;
type Props = StateProps & DispatchProps & RouteProps & {
    children?: any
}
class Profile extends React.Component {
    async componentDidMount() {
        if (!this.props.user) {
            await this.props.validate();
        }
    }
+    state = {
+        loading: false,
+        imageUrl: ''
+    };
+
+    handleChange = (info: any) => {
+        if (info.file.status === 'uploading') {
+            this.setState({ loading: true });
+        } else if (info.file.status === 'done') {
+            let { code, data, error } = info.file.response;
+            if (code == 0) {
+                this.setState({
+                    imageUrl: data,
+                    loading: false,
+                }, () => this.props.changeAvatar(data))
+            } else {
+                message.error(error);
+            }
+        }
+    };
    render() {
        let content;//里存放着要渲染的内容
        if (this.props.loginState == LOGIN_TYPES.UN_VALIDATE) {
            content = null;
        } else if (this.props.loginState == LOGIN_TYPES.LOGINED) {
+            let { user } = this.props;
+            let { imageUrl } = this.state;
+            if (!imageUrl)
+                imageUrl = user.avatar;
+            content = (
+
+
+
+
+
+                                name="avatar"
+                                listType="picture-card"
+                                className="avatar-uploader"
+                                showUploadList={false}
+                                action="http://localhost:8000/uploadAvatar"
+                                beforeUpload={beforeUpload}
+                                data={{ userId: user._id }}
+                                onChange={this.handleChange}
+                            >
+                                {
+                                    this.state.loading ?  :
+                                }
+
+
                        {}
                        15718856132
                        zhangsan@qq.com

+                      {
+                         await this.props.logout();
+                         this.props.history.push('/login');
+                     }}>退出登录

            )
        } else {
            content = (
                <>

                         this.props.history.push('/login')}>登录
                         this.props.history.push('/register')}>注册

                </>
            )
        }
        return (
            (

                    个人中心
                    {content}

            )
        )
    }
}
let mapStateToProps = (state: TypeRootState): TypeProfile => state.profile
export default connect(
    mapStateToProps,
    actions
)(Profile);

+function getBase64(img: any, callback: any) {
+    const reader = new FileReader();
+    reader.addEventListener('load', () => callback(reader.result));
+    reader.readAsDataURL(img);
+}

+function beforeUpload(file: any) {
+    const isJpgOrPng = file.type === 'image/jpeg' || file.type === 'image/png';
+    if (!isJpgOrPng) {
+        message.error('你只能上传JPG/PNG 文件!');
+    }
+    const isLessThan2M = file.size / 1024 / 1024 < 2;
+    if (!isLessThan2M) {
+        message.error('图片必须小于2MB!');
+    }
+    return isJpgOrPng && isLessThan2M;
+}
```

src\store\action-types.tsx

```
export const SET_CURRENT_CATEGORY = 'SET_CURRENT_CATEGORY';

export const VALIDATE = 'VALIDATE';
export const LOGOUT = 'LOGOUT';
+export const CHANGE_AVATAR = 'CHANGE_AVATAR';
```

src\store\actions\profile.tsx

```
+import { AnyAction, Dispatch } from 'redux';
import * as TYPES from '../action-types';
+import { validate, logout, register, login } from '../../api/profile';
+import { push } from 'connected-react-router';
+import { message } from "antd";
+import { TypeAction, TypeObject, TypeThunkFunction } from '../../typings/common';
export default {
    validate(): AnyAction {
        return {
            type: TYPES.VALIDATE,
            payload: validate()
        }
    },
+    register(values: TypeObject): TypeThunkFunction {
+        return async function (dispatch: Dispatch) {
+            let result: any = await register(values);
+            if (result.code === 0) {
+                dispatch(push('/login'));
+            } else {
+                message.error(result.error);
+            }
+        }
+    },
+    login(values: TypeObject): TypeThunkFunction {
+        return async function (dispatch) {
+            let result: any = await login(values);
+            if (result.code === 0) {
+                dispatch(push('/profile'));
+            } else {
+                message.error(result.error);
+            }
+        }
+    },
+    logout(): TypeAction {
+        return {
+            type: TYPES.LOGOUT,
+            payload: logout()
+        }
+    },
+    changeAvatar(avatar: string) {
+        return {
+            type: TYPES.CHANGE_AVATAR,
+            payload: avatar
+        }
+    }
}
```

src\store\reducers\profile.tsx

```
import { AnyAction } from 'redux';
import * as TYPES from "../action-types";
import LOGIN_TYPES from '../../typings/login-types';
export interface TypeProfile {
    loginState: LOGIN_TYPES,
    user: any,
    error: string | null
}
let initialState: TypeProfile = {
    loginState: LOGIN_TYPES.UN_VALIDATE,
    user: null,
    error: null
}
export default function (state: TypeProfile = initialState, action: AnyAction): TypeProfile {
    switch (action.type) {
        case TYPES.VALIDATE:
            let { code, data, error } = action.payload;
            if (code == 0) {
                return {
                    ...state,
                    loginState: LOGIN_TYPES.LOGINED,
                    user: data,
                    error: null
                };
            } else {
                return { ...state, loginState: LOGIN_TYPES.UNLOGIN, user: null, error };
            }
+        case TYPES.LOGOUT:
+            return { ...state, loginState: LOGIN_TYPES.UN_VALIDATE, user: null, error: null };
+        case TYPES.CHANGE_AVATAR:
+            return { ...state, user: { ...state.user, avatar: action.payload } };
        default:
            return state;
    }
}
```

src\api\profile.tsx

```
import axios from './index';
+import { TypeObject } from '../typings/common';
export function validate() {
    return axios.get('/validate');
}
+export function logout() {
+    return axios.get('/logout');
+}
+export function login(values: TypeObject) {
+    return axios.post('/login', values);
+}
+export function register(values: TypeObject) {
+    return axios.post('/register', values);
+}
```

6.轮播图

src\routes\Home\index.tsx

```
import React from 'react';
import { connect } from 'react-redux';
import { TypeRootState } from '../../store/reducers';
import { TypeHome } from '../../store/reducers/home';
import actions from '../../store/actions/home';
import { RouteComponentProps } from 'react-router';
import HomeHeader from './HomeHeader';
+import HomeSliders from './HomeSliders';
import './index.less';
interface State {

}
//当前的组件有三个属性来源
//1.mapStateToProps的返回值 2.actions对象类型 3. 来自路由 4.用户传入进来的其它属性
type StateProps = ReturnType;
type DispatchProps = typeof actions;
interface IParams { }
type RouteProps = RouteComponentProps;
type Props = StateProps & DispatchProps & RouteProps & {
    children?: any
}
class Home extends React.Component {
+    homeContainerRef: any
    render() {
        return (
+            <>

+
+
+                        sliders={this.props.sliders}
+                        getSliders={this.props.getSliders} />
+
+            </>
        )
    }
}
let mapStateToProps = (state: TypeRootState): TypeHome => state.home
export default connect(
    mapStateToProps,
    actions
)(Home);
```

src\routes\Home\HomeSliders\index.tsx

```
import React from "react";
import { Carousel } from "antd";
import "./index.less";
import { TypeSlider } from '../../../store/reducers/home';
interface Props {
    children?: any;
    sliders?: any,
    getSliders?: any
}
interface State { }
class HomeSliders extends React.Component<Props, State> {
    constructor(props: Props) {
        super(props);
    }
    componentDidMount() {
        if (this.props.sliders.length == 0) {
            this.props.getSliders();
        }
    }
    render() {
        return (
            <Carousel effect="scrollx" autoplay>
                {
                    this.props.sliders.map((item: TypeSlider, index: number) => (
                        <div key={index}>
                            <img src={item.url} />
                        div>
                    ))
                }
            Carousel>
        );
    }
}
export default HomeSliders;
```

routes\Home\HomeSliders\index.less

```
.ant-carousel .slick-slide {
    text-align: center;
    height: 3.2rem;
    line-height: 3.2rem;
    background: #364d79;
    overflow: hidden;
  }

.ant-carousel .slick-slide {
    color: #fff;
    img{
      width:100%;
      height:3.2rem;
    }
}
```

src\store\actions\home.tsx

```
import * as TYPES from '../action-types';
import { TypeAction } from '../../typings/common';
+import { getSliders } from '../../api/home';
export default {
    setCurrentCategory(payload: string): TypeAction {
        return { type: TYPES.SET_CURRENT_CATEGORY, payload }
    },
+    getSliders(): TypeAction {
+        return {
+            type: TYPES.GET_SLIDERS,
+            payload: getSliders()
+        }
+    }
}
```

src\store\action-types.tsx

```

export const SET_CURRENT_CATEGORY = 'SET_CURRENT_CATEGORY';
+export const GET_SLIDERS = 'GET_SLIDERS';

export const VALIDATE = 'VALIDATE';
export const LOGOUT = 'LOGOUT';
export const CHANGE_AVATAR = 'CHANGE_AVATAR';

```

src\api\home.tsx

```
import axios from './index';
export function getSliders() {
    return axios.get('/sliders');
}
```

src\store\reducers\home.tsx

```
import { AnyAction } from 'redux';
import * as TYPES from '../action-types';
export interface TypeSlider {
    url: string
}
export interface TypeHome {
    currentCategory: string;
    sliders: Array;
}
let initialState: TypeHome = {
    currentCategory: 'all',
    sliders: []
}
export default function (state: TypeHome = initialState, action: AnyAction) {
    switch (action.type) {
        case TYPES.SET_CURRENT_CATEGORY:
            return { ...state, currentCategory: action.payload };
        case TYPES.GET_SLIDERS:
            return { ...state, sliders: action.payload.data };
        default:
            return state;
    }
}
```

7.课程列表

src\routes\Home\index.tsx

```
import React from 'react';
import { connect } from 'react-redux';
import { TypeRootState } from '../../store/reducers';
import { TypeHome } from '../../store/reducers/home';
import actions from '../../store/actions/home';
import { RouteComponentProps } from 'react-router';
import HomeHeader from './HomeHeader';
import HomeSliders from './HomeSliders';
+import LessonList from './LessonList';
+import { loadMore, downReferesh } from '../../utils';
import './index.less';
interface State {

}
//当前的组件有三个属性来源
//1.mapStateToProps的返回值 2.actions对象类型 3. 来自路由 4.用户传入进来的其它属性
type StateProps = ReturnType;
type DispatchProps = typeof actions;
interface IParams { }
type RouteProps = RouteComponentProps;
type Props = StateProps & DispatchProps & RouteProps & {
    children?: any
}
class Home extends React.Component {
    homeContainerRef: any
+    lessonListRef: any
+    constructor(props: Props) {
+        super(props);
+       this.homeContainerRef = React.createRef();
+        this.lessonListRef = React.createRef();
+    }
+    componentDidMount() {
+        loadMore(this.homeContainerRef.current, this.props.getLessons);
+        downReferesh(this.homeContainerRef.current, this.props.refreshLessons);
+        this.homeContainerRef.current.addEventListener('scroll', () => {
+            this.lessonListRef.current.forceUpdate();
+        });
    }
    render() {
        return (
            <>

+
+                         ref={this.lessonListRef}
+                         container={this.homeContainerRef}
+                         lessons={this.props.lessons}
+                         getLessons={this.props.getLessons} />

            </>
        )
    }
}
let mapStateToProps = (state: TypeRootState): TypeHome => state.home
export default connect(
    mapStateToProps,
    actions
)(Home);
```

src\routes\Home\index.less

```
+.home-container{
+    position: fixed;
+    top:1rem;
+    left:0;
+    width:100%;
+    overflow-y: auto;
+    height:calc(100vh - 2.22rem);
}
```

src\routes\Home\LessonList\index.tsx

```
import React from "react";
import "./index.less";
import { Icon, Card, Skeleton, Button, Alert } from 'antd';
import { Link } from 'react-router-dom';
import { TypeLesson } from '../../../store/reducers/home';
interface Props {
    children?: any;
    lessons?: any;
    getLessons?: any;
    container?: any
}
class LessonList extends React.Component<Props> {
    constructor(props: Props) {
        super(props);
    }
    componentDidMount() {
        if (this.props.lessons.list.length == 0) {
            this.props.getLessons();
        }
    }
    render() {
        let start = 0;
        let rem = parseInt(document.documentElement.style.fontSize);
        if (this.props.container.current) {
            let scrollTop = this.props.container.current.scrollTop;

            if (scrollTop - 4.2 * rem > 0) {
                start = Math.floor((scrollTop - 4.2 * rem) / (6.5 * rem));
            }
        }
        return (
            <section className="lesson-list">
                <h2><Icon type="menu" />全部课程h2>
                <Skeleton loading={this.props.lessons.list.length == 0 && this.props.lessons.loading} active paragraph={{ rows: 8 }}>
                    {
                        this.props.lessons.list.map((lesson: TypeLesson, index: number) => (
                            index >= start && index < start + 5 ? <Link key={lesson._id} to={{ pathname: `/detail/${lesson._id}`, state: lesson }}>
                                <Card hoverable={true} style={{ width: '100%' }} cover={<img alt={lesson.title} src={lesson.poster} />} >
                                    <Card.Meta title={lesson.title} description={`价格: ${lesson.price}`} />
                                Card>
                            Link> : <div key={index} style={{ height: `${6.5 * rem}px` }}>div>
                        ))
                    }
                    {
                        this.props.lessons.hasMore ? <Button onClick={this.props.getLessons} loading={this.props.lessons.loading} type="primary" block >
{this.props.lessons.loading ? '' : '加载更多'}Button> : <Alert style={{ textAlign: 'center' }} message="到底了" type="warning" />
                    }
                Skeleton>
            section>
        );
    }
}
export default LessonList
```

src\routes\Home\LessonList\index.less

```
.lesson-list{
  h2{
    line-height: 1rem;
    i{
      margin:0 .1rem;
    }
  }
  .ant-card.ant-card-bordered.ant-card-hoverable{
    height:6.5rem;
    overflow:hidden;
  }
}
```

src\store\actions\home.tsx

```
import * as TYPES from '../action-types';
import { TypeAction, TypeThunkFunction } from '../../typings/common';
import { getSliders, getLessons } from '../../api/home';
export default {
    setCurrentCategory(payload: string): TypeAction {
        return { type: TYPES.SET_CURRENT_CATEGORY, payload }
    },
    getSliders(): TypeAction {
        return {
            type: TYPES.GET_SLIDERS,
            payload: getSliders()
        }
    },
+    getLessons(): TypeThunkFunction {
+        return async (dispatch, getState) => {
+            let { currentCategory, lessons: { hasMore, offset, limit, loading } } = getState().home;
+            if (hasMore && !loading) {
+                dispatch({ type: TYPES.SET_LESSONS_LOADING, payload: true });
+                let result = await getLessons(currentCategory, offset, limit);
+                dispatch({ type: TYPES.SET_LESSONS, payload: result.data });
+            }
+        }
+    },
+    refreshLessons(): TypeThunkFunction {
+        return async (dispatch, getState) => {
+            let { currentCategory, lessons: { limit, loading } } = getState().home;
+            if (!loading) {
+                dispatch({ type: TYPES.SET_LESSONS_LOADING, payload: true });
+                let result = await getLessons(currentCategory, 0, limit);
+                dispatch({ type: TYPES.REFRESH_LESSONS, payload: result.data });
+            }
+        }
+    }
}
```

src\store\action-types.tsx

```

export const SET_CURRENT_CATEGORY = 'SET_CURRENT_CATEGORY';
export const GET_SLIDERS = 'GET_SLIDERS';

export const VALIDATE = 'VALIDATE';
export const LOGOUT = 'LOGOUT';
export const CHANGE_AVATAR = 'CHANGE_AVATAR';

+export const GET_LESSONS = 'GET_LESSONS';
+export const SET_LESSONS_LOADING = 'SET_LESSONS_LOADING';
+export const SET_LESSONS = 'SET_LESSONS';
+export const REFRESH_LESSONS = 'REFRESH_LESSONS';
```

src\api\home.tsx

```
import axios from './index';
export function getSliders() {
    return axios.get('/sliders');
}
+export function getLessons(currentCategory: string = 'all', offset: number, limit: number) {
+    return axios.get(`/getLessons/${currentCategory}?offset=${offset}&limit=${limit}`);
+}
```

src\store\reducers\home.tsx

```
import { AnyAction } from 'redux';
import * as TYPES from '../action-types';
export interface TypeSlider {
    url: string
}

+export interface TypeLesson {
+    _id: string;
+    id: number;
+    title: string;
+    video: string;
+    poster: string;
+    url: string;
+    price: string;
+    category: string;
+}

+export interface TypeLessons {
+    loading: boolean;
+    list: Array;
+    hasMore: boolean;
+    offset: number;
+    limit: number;
+}

export interface TypeHome {
    currentCategory: string;
    sliders: Array;
+    lessons: TypeLessons;
}
let initialState: TypeHome = {
    currentCategory: 'all',
    sliders: [],
+    lessons: {
+        loading: false,
+        list: [],
+        hasMore: true,
+        offset: 0,
+        limit: 5
+    }
}
export default function (state: TypeHome = initialState, action: AnyAction) {
    switch (action.type) {
        case TYPES.SET_CURRENT_CATEGORY:
            return { ...state, currentCategory: action.payload };
        case TYPES.GET_SLIDERS:
            return { ...state, sliders: action.payload.data };
+        case TYPES.SET_LESSONS_LOADING:
+            return {
+                ...state, lessons: {
+                    ...state.lessons,
+                    loading: action.payload
+                }
+            };
+        case TYPES.SET_LESSONS:
+            return {
+                ...state, lessons: {
+                    ...state.lessons,
+                    loading: false,
+                    hasMore: action.payload.hasMore,
+                    list: [...state.lessons.list, ...action.payload.list],
+                    offset: state.lessons.offset + action.payload.list.length
+                }
+            };
+        case TYPES.REFRESH_LESSONS:
+            return {
+                ...state, lessons: {
+                    ...state.lessons,
+                    loading: false,
+                    hasMore: action.payload.hasMore,
+                    list: action.payload.list,
+                    offset: action.payload.list.length
+                }
+            };
        default:
            return state;
    }
}
```

src\utils.tsx

```
export function loadMore(element: any, callback: any) {
    function _loadMore() {
        let clientHeight = element.clientHeight;
        let scrollTop = element.scrollTop;
        let scrollHeight = element.scrollHeight;
        if (clientHeight + scrollTop + 10 >= scrollHeight) {
            callback();
        }
    }
    element.addEventListener('scroll', debounce(_loadMore, 300));
}

export function downReferesh(element: any, callback: any) {
    console.log('downReferesh', new Date());
    let startY: number;
    let distance: number;
    let originTop = element.offsetTop;
    element.addEventListener('touchstart', function (event: any) {
        let touchMove = throttle(_touchMove, 70);
        if (element.offsetTop == originTop && element.scrollTop == 0) {
            startY = event.touches[0].pageY;
            element.addEventListener('touchmove', touchMove);
            element.addEventListener('touchend', touchEnd);
        }

        function _touchMove(event: any) {
            let pageY = event.touches[0].pageY;
            if (pageY > startY) {
                distance = pageY - startY;
                element.style.top = originTop + distance + 'px';
            } else {
                element.removeEventListener('touchmove', touchMove);
                element.removeEventListener('touchend', touchEnd);
            }
        }
        function touchEnd() {
            element.removeEventListener('touchmove', touchMove);
            element.removeEventListener('touchend', touchEnd);
            let timer = setInterval(function () {
                if (distance < 1) {
                    element.style.top = originTop + 'px';
                    clearInterval(timer);
                } else {
                    element.style.top = originTop + (--distance) + 'px';
                }
            }, 13);
            if (distance > 30) {
                callback();
            }
        }
    });
}

export function debounce(fn: any, wait: number) {
    var timeout: any = null;
    return function () {
        if (timeout !== null) clearTimeout(timeout);
        timeout = setTimeout(fn, wait);
    }
}
export function throttle(func: any, delay: number) {
    var prev = Date.now();
    return function () {
        var context = this;
        var args = arguments;
        var now = Date.now();
        if (now - prev >= delay) {
            func.apply(context, args);
            prev = Date.now();
        }
    }
}
```

8.课程详情

src\index.tsx

```
import React from 'react';
import ReactDOM from 'react-dom';
import { Switch, Route, Redirect } from 'react-router-dom';
import { Provider } from 'react-redux';
import store from './store';
import './assets/common.less';//这里放公共的样式
import { ConnectedRouter } from 'connected-react-router';
import history from './store/history';
import Home from './routes/Home';
import Profile from './routes/Profile';
import Register from './routes/Register';
import Login from './routes/Login';
import Mine from './routes/Mine';
import Tabs from './components/Tabs';
+import Detail from "./routes/Detail";
ReactDOM.render((

            <>

+

            </>

), document.getElementById('root'));
```

src\routes\Detail\index.tsx

```
import React from 'react';
import { connect } from 'react-redux';
import { Card, Button } from 'antd';
import NavHeader from "../../components/NavHeader";
import { getLesson } from '../../api/detail';
import { RouteComponentProps } from 'react-router';
import { TypeLesson } from '../../store/reducers/home';
const { Meta } = Card;
interface IParams { id: string }
type RouteProps = RouteComponentProps;
type Props = RouteProps & {
    children?: any
}
interface State {
    lesson: TypeLesson;
}
class Detail extends React.Component<Props, State> {
    state: State = { lesson: { _id: '', id: 1, title: '', video: '', poster: '', url: '', price: '', category: '' } }
    async componentDidMount() {
        let lesson = this.props.location.state;
        if (!lesson) {
            let id = this.props.match.params.id;
            let result: any = await getLesson(id);
            if (result.code == 0)
                lesson = result.data;
        }
        this.setState({ lesson });
    }
    render() {
        let { lesson } = this.state;
        return (
            <>
                <NavHeader history={this.props.history}>课程详情NavHeader>
                <Card
                    hoverable
                    style={{ width: '100%' }}
                    cover={<video src={lesson.video} controls autoPlay={false} />}
                >
                    <Meta title={lesson.title} description={<p>价格: {lesson.price}p>} />
                Card>
            </>
        )
    }
}
export default connect(

)(Detail);
```

src\api\detail.tsx

```
import axios from './index';
export function getLesson(id: string) {
    return axios.get(`/getLesson?id=${id}`);
}
```