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=331 sentences=1560, words=11288

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 autoprefixer px2rem-loader postcss-loader lib-
flexible -D
cnpm i redux react-redux @types/react-redux redux-thunk  redux-logger @types/redux-logger redux-promise @types/redux-promise immer redux-immer -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": "ESNext",
    "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 tsImportPluginFactory = require("ts-import-plugin");
const path = require("path");

module.exports = {
  mode: process.env.NODE_ENV == "production" ? "production" : "development",
  entry: "./src/index.tsx",
  output: {
    path: path.join(__dirname, "dist"),
    filename: "bundle.js",
  },
  devtool: "source-map",
  devServer: {
    hot: true,
    contentBase: path.join(__dirname, "dist"),
    historyApiFallback: {

      index: "./index.html",
    },
  },
  resolve: {
    alias: {
      "@": path.resolve(__dirname, "src"),
      "~": path.resolve(__dirname, "node_modules"),
    },

    extensions: [".ts", ".tsx", ".js", ".json"],
  },
  module: {
    rules: [
      {
        test: /\.(j|t)sx?$/,
        loader: "ts-loader",
        options: {
          transpileOnly: true,
          getCustomTransformers: () => ({
            before: [
              tsImportPluginFactory({
                libraryName: "antd",
                libraryDirectory: "es",
```

```
              style: "css",
            }),
          ],
        }),
        compilerOptions: {
          module: "es2015",
        },
      },
    },
    {
      test: /\.css$/,
      use: [
        "style-loader",
        {
          loader: "css-loader",
          options: { importLoaders: 0 },
        },
        {
          loader: "postcss-loader",
          options: {
            plugins: [require("autoprefixer")],
          },
        },
        {
          loader: "px2rem-loader",
          options: {
            remUnit: 75,
            remPrecesion: 8,
          },
        },
      ],
    },
    {
      test: /\.less$/,
      use: [
        "style-loader",
        {
          loader: "css-loader",
          options: { importLoaders: 0 },
        },
        {
          loader: "postcss-loader",
          options: {
            plugins: [require("autoprefixer")],
          },
        },
        {
          loader: "px2rem-loader",
          options: {
            remUnit: 75,
            remPrecesion: 8,
          },
        },
        "less-loader",
      ],
    },
    {
      test: /\.(jpg|png|gif|svg|jpeg)$/,
      use: ["url-loader"],
    },
  ],
 },
 plugins: [
   new HtmlWebpackPlugin({
     template: "./src/index.html",
   }),

   new webpack.HotModuleReplacementPlugin(),
 ],
};
```

```
  "scripts": {
    "build": "webpack",
    "dev": "webpack-dev-server"
  },
```

```
import React from "react";
import ReactDOM from "react-dom";
ReactDOM.render(<h1>hello1>, 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" />
    <link
      rel="stylesheet"
      href="https://cdn.bootcss.com/normalize/8.0.1/normalize.min.css"
    />
    <title>珠峰课堂title>
  head>

  <body>
    <script>
      let docEle = document.documentElement;
      function setRemUnit() {
        docEle.style.fontSize = docEle.clientWidth / 10 + "px";
      }
      setRemUnit();
      window.addEventListener("resize", setRemUnit);
    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 Mine from "./routes/Mine";
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="/Mine" component={Mine} />
            <Route path="/profile" component={Profile} />
            <Redirect to="/" />
          Switch>
        main>
        <Tabs />
      ConfigProvider>
    ConnectedRouter>
  Provider>,
  document.getElementById("root")
);
```

src\assets\css\common.less

```
ul,li{
    list-style: none;
}
#root{
    margin:0 auto;
    max-width: 750px;
    box-sizing: border-box;
}
.main-container{
    padding:100px 0 120px 0;
}
```

src\components\Tabs\index.tsx

```
import React from "react";
import { withRouter, NavLink } from "react-router-dom";
import { Icon } from "antd";
import "./index.less";
function Tabs() {
  return (
    <footer>
      <NavLink exact to="/">
        <Icon type="home" />
        <span>首页span>
      NavLink>
      <NavLink to="/mine">
        <Icon type="shopping-cart" />
        <span>购物车span>
      NavLink>
      <NavLink to="/profile">
        <Icon type="user" />
        <span>个人中心span>
      NavLink>
    footer>
  );
}
export default withRouter(Tabs);
```

src\components\Tabs\index.less

```
footer {
  position: fixed;
  left: 0;
  bottom: 0;
  width: 100%;
  height: 1.2rem;
  z-index: 1000;
  background-color: #fff;
  border-top: 0.02rem solid #d5d5d5;
  display: flex;
  justify-content: center;
  align-items: center;
  a {
    display: flex;
    flex: 1;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    color: #000;
    i {
      font-size: 0.5rem;
    }
    span {
      font-size: 0.3rem;
      line-height: 0.5rem;
    }
    &.active {
      color: blue;
      font-weight: bold;
    }
  }
}
```

src\store\history.tsx

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

src\store\action-types.tsx

```
export const ADD = "ADD";
```

src\store\reducers\home.tsx

```
import { AnyAction } from "redux";
export interface HomeState {}
let initialState: HomeState = {};
export default function (
  state: HomeState = initialState,
  action: AnyAction
): HomeState {
  switch (action.type) {
    default:
      return state;
  }
}
```

src\store\reducers\index.tsx

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

export { CombinedState }
export default reducer;
```

src\store\index.tsx

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

src\routes\Home\index.tsx

```
import React, { PropsWithChildren } from "react";
import { connect } from "react-redux";
import { RouteComponentProps } from "react-router-dom";
interface Params {}
type Props = PropsWithChildren>;
function Home(props: Props) {
  return <div>Homediv>;
}
export default connect()(Home);
```

src\routes\Mine\index.tsx

```
import React, { PropsWithChildren } from "react";
import { connect } from "react-redux";
import { RouteComponentProps } from "react-router-dom";
interface Params {}
type Props = PropsWithChildren>;
function Mine(props: Props) {
  return <div>Minediv>;
}
export default connect()(Mine);
```

src\routes\Profile\index.tsx

```
import React, { PropsWithChildren } from "react";
import { connect } from "react-redux";
import { RouteComponentProps } from "react-router-dom";
interface Params {}
type Props = PropsWithChildren>;
function Profile(props: Props) {
  return <div>Profilediv>;
}
export default connect()(Profile);
```

src\store\reducers\mime.tsx

```
import { AnyAction } from "redux";
export interface MimeState {}
let initialState: MimeState = {};
export default function (
  state: MimeState = initialState,
  action: AnyAction
): MimeState {
  switch (action.type) {
    default:
      return state;
  }
}
```

3.首页头部导航

tsconfig.json

```
  "compilerOptions": {
    "baseUrl": ".",
    "paths": {
      "@/*": [
        "./src/*"
      ]
    }
  },
```

src\routes\Home\components\HomeHeader\index.tsx

```
import React, { useState, CSSProperties } from 'react';
import './index.less';
import { Icon } from 'antd';
import classnames from 'classnames';
import { Transition } from 'react-transition-group';
import logo from '@/assets/images/logo.png';

const duration = 1000;

const defaultStyle = {
    transition: `opacity ${duration}ms ease-in-out`,
    opacity: 0,
}
interface TransitionStyles {
    entering: CSSProperties;
    entered: CSSProperties;
    exiting: CSSProperties;
    exited: CSSProperties;
}
const transitionStyles: TransitionStyles = {
    entering: { opacity: 1 },
    entered: { opacity: 1 },
    exiting: { opacity: 0 },
    exited: { opacity: 0 },
};

interface Props {
    currentCategory: string;
    setCurrentCategory: (currentCategory: string) => any;
    refreshLessons: any;
}
function HomeHeader(props: Props) {
    let [isMenuVisible, setIsMenuVisible] = useState(false);
    const setCurrentCategory = (event: React.MouseEvent) => {
        let target: HTMLULListElement = event.target as HTMLULListElement;
        let category = target.dataset.category;
        props.setCurrentCategory(category);
        props.refreshLessons();
        setIsMenuVisible(false);
    }
    return (
        <header className="home-header">
            <div className="logo-header">
                <img src={logo} />
                <Icon type="bars" onClick={() => setIsMenuVisible(!isMenuVisible)} />
            div>
            <Transition in={isMenuVisible} timeout={duration}>
                {
                    (state: keyof TransitionStyles) => (
                        <ul
                            className="category"
                            onClick={setCurrentCategory}
                            style={{
                                ...defaultStyle,
                                ...transitionStyles[state]
                            }}
                        >
                            <li data-category="all" className={classnames({ active: props.currentCategory === 'all' })}>全部课程li>
                            <li data-category="react" className={classnames({ active: props.currentCategory === 'react' })}>React课程li>
                            <li data-category="vue" className={classnames({ active: props.currentCategory === 'vue' })}>Vue课程li>
                        ul>
                    )
                }
            Transition>
        header>
    )
}
export default HomeHeader;
```

src\routes\Home\components\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;
      }
    }
  }
}
```

src\store\action-types.tsx

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

src\store\reducers\home.tsx

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

src\store\actions\home.tsx

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

src\routes\Home\index.tsx

```
import React, { PropsWithChildren } from 'react';
import { connect } from 'react-redux';
import { RouteComponentProps } from 'react-router-dom';
+import actions from '@/store/actions/home';
+import HomeHeader from './components/HomeHeader';
+import { CombinedState } from '@/store/reducers';
+import { HomeState } from '@/store/reducers/home';
+import './index.less';
+type StateProps = ReturnType;
+type DispatchProps = typeof actions;
interface Params { }
+type Props = PropsWithChildren & StateProps & DispatchProps>;
function Home(props: Props) {
    return (
+        <>
+
+                currentCategory={props.currentCategory}
+                setCurrentCategory={props.setCurrentCategory}
+                refreshLessons={props.refreshLessons}
+            />
+        </>
+    )
}
+let mapStateToProps = (state: CombinedState): HomeState => state.home;
export default connect(
+    mapStateToProps,
+    actions
)(Home);
```

4.个人中心

src\routes\Profile\index.tsx

```
import React, { PropsWithChildren, useEffect } from "react";
import { connect } from "react-redux";
import { CombinedState } from "../../store/reducers";
import { ProfileState } 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, message } from "antd";
import NavHeader from "../../components/NavHeader";
import { AxiosError } from "axios";
import "./index.less";

type StateProps = ReturnType<typeof mapStateToProps>;
type DispatchProps = typeof actions;
interface Params {}
type RouteProps = RouteComponentProps;
type Props = PropsWithChildren;
function Profile(props: Props) {
  useEffect(() => {
    props.validate().catch((error: AxiosError) => message.error(error.message));
  }, []);
  let content;
  if (props.loginState == LOGIN_TYPES.UN_VALIDATE) {
    content = null;
  } else if (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={() => props.history.push("/login")}>
            登录
          Button>
          <Button
            type="dashed"
            style={{ marginLeft: ".5rem" }}
            onClick={() => props.history.push("/register")}
          >
            注册
          Button>
        div>
      </>
    );
  }
  return (
    <section>
      <NavHeader history={props.history}>个人中心NavHeader>
      {content}
    section>
  );
}

let mapStateToProps = (state: CombinedState): ProfileState => state.profile;
export default connect(mapStateToProps, actions)(Profile);
```

src\routes\Profile\index.less

```
.user-info {
  padding: 0.2rem;
}
```

src\store\action-types.tsx

```
export const SET_CURRENT_CATEGORY = 'SET_CURRENT_CATEGORY';

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

src\typings\login-types.tsx

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

src\store\reducers\profile.tsx

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

src\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\index.tsx

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

export { CombinedState }
export default reducer;
```

src\api\index.tsx

```
import axios from "axios";
import qs from "qs";
axios.defaults.baseURL = "http://localhost:8000";
axios.defaults.headers.post["Content-Type"] = "application/json;charset=UTF-8";

axios.interceptors.request.use(
  (config) => {
    let access_token = sessionStorage.getItem("access_token");
    config.headers = {
      Authorization: `Bearer ${access_token}`,
    };
    return config;
  },
  (error) => {
    return Promise.reject(error);
  }
);
axios.interceptors.response.use(
  (response) => response.data,
  (error) => Promise.reject(error)
);
export default axios;
```

src\api\profile.tsx

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

src\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>
  );
}
```

src\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: 0.2rem;
    line-height: 1rem;
  }
}
```

5.注册登陆

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 Mine from "./routes/Mine";
import Profile from "./routes/Profile";
+import Register from "./routes/Register";
+import Login from "./routes/Login";
import { ConnectedRouter } from 'connected-react-router';
import history from './store/history';
ReactDOM.render(

+
+


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

src\api\profile.tsx

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

src\routes\Profile\index.tsx

```
import React, { PropsWithChildren, useEffect } from 'react';
import { connect } from 'react-redux';
import { CombinedState } from '../../store/reducers';
import { ProfileState } 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, message } from 'antd';
import NavHeader from '../../components/NavHeader';
import { AxiosError } from 'axios';
import './index.less';
//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 Params { }
type RouteProps = RouteComponentProps;
type Props = PropsWithChildren;

function Profile(props: Props) {
    useEffect(() => {
        props.validate().catch((error: AxiosError) => message.error(error.message));
    }, []);
    let content;//里存放着要渲染的内容
    if (props.loginState == LOGIN_TYPES.UN_VALIDATE) {
        content = null;
    } else if (props.loginState == LOGIN_TYPES.LOGINED) {
        content = (

+                {props.user.username}
+                {props.user.email}

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

        )
    } else {
        content = (
            <>

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

            </>
        )
    }
    return (
        (

                个人中心
                {content}

        )
    )
}
const mapStateToProps = (initialState: CombinedState): ProfileState => initialState.profile;
export default connect(
    mapStateToProps,
    actions
)(Profile);
```

src\store\action-types.tsx

```
export const SET_CURRENT_CATEGORY = 'SET_CURRENT_CATEGORY';

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

src\store\actions\profile.tsx

```
import { AnyAction } from 'redux';
import * as TYPES from '../action-types';
+import { validate, register, login } from '@/api/profile';
+import { push } from 'connected-react-router';
+import { RegisterPayload, LoginPayload, RegisterResult, LoginResult } from '@/typings/user';
+import { message } from "antd";
export default {
    validate(): AnyAction {
        return {
            type: TYPES.VALIDATE,
            payload: validate()
        }
    },
+    register(values: RegisterPayload) {
+        return function (dispatch: any) {
+            (async function () {
+                try {
+                    let result: RegisterResult = await register(values);
+                    if (result.success) {
+                        dispatch(push('/login'));
+                    } else {
+                        message.error(result.message);
+                    }
+                } catch (error) {
+                    message.error('注册失败');
+                }
+            })();
+        }
+    },
+    login(values: LoginPayload) {
+        return function (dispatch: any) {
+            (async function () {
+                try {
+                    let result: LoginResult = await login(values);
+                    if (result.success) {
+                        sessionStorage.setItem('access_token', result.data.token);
+                        dispatch(push('/profile'));
+                    } else {
+                        message.error(result.message);
+                    }
+                } catch (error) {
+                    message.error('登录失败');
+                }
+            })();
+        }
+    },
+    logout() {
+        return function (dispatch: any) {
+            sessionStorage.removeItem('access_token');
+            dispatch({ type: TYPES.LOGOUT });
+            dispatch(push('/login'));
+        }
+    }
}
```

src\typings\user.tsx

```
export interface RegisterPayload {
    username: string,
    password: string,
    email: string;
    confirmPassword: string;
}
export interface LoginPayload {
    username: string,
    password: string,
}
export interface RegisterResult {
    data: { token: string }
    success: boolean,
    message?: any
}
export interface LoginResult {
    data: { token: string }
    success: boolean,
    message?: any
}
```

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 { FormComponentProps } from "antd/lib/form";
import { CombinedState } from "../../store/reducers";
import { ProfileState } from "../../store/reducers/profile";
import "./index.less";
import { RegisterPayload } from "@/typings/user";
type StateProps = ReturnType<typeof mapStateToProps>;
type DispatchProps = typeof actions;
interface Params {}
type Props = RouteComponentProps &
  StateProps &
  DispatchProps &
  FormComponentProps;

function Register(props: Props) {
  const handleSubmit = (event: React.FormEvent) => {
    event.preventDefault();
    props.form.validateFields(async (errors: any, values: RegisterPayload) => {
      if (errors) {
        message.error("表单验证失败!");
      } else {
        props.register(values);
      }
    });
  };
  const { getFieldDecorator } = props.form;
  return (
    <>
      <NavHeader history={props.history}>用户注册NavHeader>
      <Form onSubmit={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("confirmPassword", {
            rules: [{ required: true, message: "请输入你的确认密码!" }],
          })(
            <Input
              prefix={<Icon type="lock" style={{ color: "rgba(0,0,0,.25)" }} />}
              type="password"
              placeholder="确认密码"
            />
          )}
        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>
          <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: CombinedState): ProfileState => state.profile;
export default connect(mapStateToProps, actions)(WrappedRegister);
```

routes\Register\index.less

```
.login-form {
  padding: 0.2rem;
}
```

src\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 { FormComponentProps } from "antd/lib/form";
import "./index.less";
import { CombinedState } from "@/store/reducers";
import { ProfileState } from "@/store/reducers/profile";
import { LoginPayload } from "@/typings/user";
type StateProps = ReturnType<typeof mapStateToProps>;
type DispatchProps = typeof actions;
interface Params {}
type Props = RouteComponentProps &
  StateProps &
  DispatchProps &
  FormComponentProps;

function Register(props: Props) {
  const handleSubmit = (event: React.FormEvent) => {
    event.preventDefault();
    props.form.validateFields(async (errors: any, values: LoginPayload) => {
      if (errors) {
        message.error("表单验证失败!");
      } else {
        props.login(values);
      }
    });
  };
  const { getFieldDecorator } = props.form;
  return (
    <>
      <NavHeader history={props.history}>用户登录NavHeader>
      <Form onSubmit={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: CombinedState): ProfileState => state.profile;
export default connect(mapStateToProps, actions)(WrappedRegister);
```

src\routes\Login\index.less

```
.login-form {
  padding: 0.2rem;
}
```

6.上传头像

src\routes\Profile\index.tsx

```
+import React, { PropsWithChildren, useEffect, useState } from 'react';
import { connect } from 'react-redux';
import { CombinedState } from '../../store/reducers';
import { ProfileState } 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, message, Upload, Icon } from 'antd';
import NavHeader from '../../components/NavHeader';
import { AxiosError } from 'axios';
import './index.less';
//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 Params { }
type RouteProps = RouteComponentProps;
type Props = PropsWithChildren;
```

```
function Profile(props: Props) {
+    let [loading, setLoading] = useState(false);
    useEffect(() => {
        props.validate().catch((error: AxiosError) => message.error(error.message));
    }, []);
+    const handleChange = (info: any) => {
+        if (info.file.status === 'uploading') {
+            setLoading(true);
+        } else if (info.file.status === 'done') {
+            let { success, data, message } = info.file.response;
+            if (success) {
+                setLoading(false);
+                props.changeAvatar(data);
+            } else {
+                message.error(message);
+            }
+        }
    };
    let content;//里存放着要渲染的内容

    if (props.loginState == LOGIN_TYPES.UN_VALIDATE) {
        content = null;
    } else if (props.loginState == LOGIN_TYPES.LOGINED) {
+        const uploadButton = (
+
+
+                上传
+
+        );
        content = (

                    {props.user.username}
                    {props.user.email}
+
+
+                            name="avatar"
+                            listType="picture-card"
+                            className="avatar-uploader"
+                            showUploadList={false}
+                            action="http://localhost:8000/user/uploadAvatar"
+                            beforeUpload={beforeUpload}
+                            data={{ userId: props.user._id }}
+                            onChange={handleChange}
+                        >
+                            {
+                                props.user.avatar ?  : uploadButton
+                            }
+
+


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

        )
    } else {
        content = (
            <>

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

            </>
        )
    }
    return (
        (

                个人中心
                {content}

        )
    )
}
+const mapStateToProps = (initialState: CombinedState): ProfileState => initialState.profile;
+export default connect(
+    mapStateToProps,
+    actions
+)(Profile);

+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\reducers\profile.tsx

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

```
import { AnyAction } from 'redux';
import * as TYPES from '../action-types';
import { validate, register, login } from '@/api/profile';
import { push } from 'connected-react-router';
import { RegisterPayload, LoginPayload, RegisterResult, LoginResult } from '@/typings/user';
import { message } from "antd";
export default {
    validate(): AnyAction {
        return {
            type: TYPES.VALIDATE,
            payload: validate()
        }
    },
    register(values: RegisterPayload) {
        return function (dispatch: any) {
            (async function () {
                try {
                    let result: RegisterResult = await register(values);
                    if (result.success) {
                        dispatch(push('/login'));
                    } else {
                        message.error(result.message);
                    }
                } catch (error) {
                    message.error('注册失败');
                }
            })();
        }
    },
    login(values: LoginPayload) {
        return function (dispatch: any) {
            (async function () {
                try {
                    let result: LoginResult = await login(values);
                    if (result.success) {
                        sessionStorage.setItem('access_token', result.data.token);
                        dispatch(push('/profile'));
                    } else {
                        message.error(result.message);
                    }
                } catch (error) {
                    message.error('登录失败');
                }
            })();
        }
    },
    logout() {
        return function (dispatch: any) {
            sessionStorage.removeItem('access_token');
            dispatch({ type: TYPES.LOGOUT });
            dispatch(push('/login'));
        }
    },
+    changeAvatar(avatar: string) {
+        return {
+            type: TYPES.CHANGE_AVATAR,
+            payload: avatar
+        }
+    }
}
```

7.轮播图

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';

+export const GET_SLIDERS = 'GET_SLIDERS';
```

src\store\actions\home.tsx

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

src\typings\slider.tsx

```
export interface Slider {
  url: string;
}
```

src\store\reducers\home.tsx

```
import { AnyAction } from 'redux';
import * as TYPES from "../action-types";
+import Slider from '@/typings/slider';
export interface HomeState {
    currentCategory: string;
+    sliders: Slider[];
}
let initialState: HomeState = {
    currentCategory: 'all',
+    sliders: []
};
export default function (state: HomeState = initialState, action: AnyAction): HomeState {
    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;
    }
}
```

src\api\home.tsx

```
import axios from "./index";
export function getSliders() {
  return axios.get("/slider/list");
}
```

src\routes\Home\components\HomeSliders\index.tsx

```
import React, { PropsWithChildren, useRef, useEffect } from "react";
import { Carousel } from "antd";
import "./index.less";
import { Slider } from "@/typings/lesson";
type Props = PropsWithChildren;
function HomeSliders(props: Props) {
  useEffect(() => {
    if (props.sliders.length == 0) {
      props.getSliders();
    }
  }, []);
  return (
    <Carousel effect="scrollx" autoplay>
      {props.sliders.map((item: Slider, index: number) => (
        <div key={index}>
          <img src={item.url} />
        div>
      ))}
    Carousel>
  );
}

export default HomeSliders;
```

src\routes\Home\components\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\routes\Home\index.tsx

```
+import React, { PropsWithChildren, useRef } from 'react';
import { connect } from 'react-redux';
import { RouteComponentProps } from 'react-router-dom';
import actions from '@/store/actions/home';
import HomeHeader from './components/HomeHeader';
import { CombinedState } from '@/store/reducers';
import { HomeState } from '@/store/reducers/home';
import HomeSliders from './components/HomeSliders';
import './index.less';
type StateProps = ReturnType;
type DispatchProps = typeof actions;
interface Params { }
type Props = PropsWithChildren & StateProps & DispatchProps>;
function Home(props: Props) {
+    const homeContainerRef = useRef(null);
    return (
        <>

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

8.课程列表

- 记住滚动条的位置

src\api\home.tsx

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

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';

+export const GET_SLIDERS = 'GET_SLIDERS';

+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\typings\lesson.tsx

```
export interface Lesson {
  id: string;
  title: string;
  video: string;
  poster: string;
  url: string;
  price: string;
  category: string;
}

export interface LessonResult {
  data: Lesson;
  success: boolean;
}
```

src\store\reducers\home.tsx

```
import { AnyAction } from 'redux';
import * as TYPES from "../action-types";
import Slider from '@/typings/slider';
import Lesson from '@/typings/Lesson';
+export interface Lesson {
+    id: string;
+    title: string;
+    video: string;
+    poster: string;
+    url: string;
+    price: string;
+    category: string;
+}

+export interface Lessons {
+    loading: boolean;
+    list: Lesson[];
+    hasMore: boolean;
+    offset: number;
+    limit: number;
+}

export interface HomeState {
    currentCategory: string;
    sliders: Slider[];
+    lessons: Lessons;
}

let initialState: HomeState = {
    currentCategory: 'all',
    sliders: [],
+    lessons: {
+        loading: false,
+        list: [],
+        hasMore: true,
+        offset: 0,
+        limit: 5
+    }
};
export default function (state: HomeState = initialState, action: AnyAction): HomeState {
    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:
+            state.lessons.loading = action.payload;
+            return state;
+        case TYPES.SET_LESSONS:
+            state.lessons.loading = false;
+            state.lessons.hasMore = action.payload.hasMore;
+            state.lessons.list = [...state.lessons.list, ...action.payload.list];
+            state.lessons.offset = state.lessons.offset + action.payload.list.length;
+            return state;
+        case TYPES.REFRESH_LESSONS:
+            state.lessons.loading = false;
+            state.lessons.hasMore = action.payload.hasMore;
+            state.lessons.list = action.payload.list;
+            state.lessons.offset = action.payload.list.length;
+            return state;
        default:
            return state;
    }
}
```

src\store\actions\home.tsx

```
import * as TYPES from '../action-types';
+import { getSliders, getLessons } from '@/api/home';
export default {
    setCurrentCategory(currentCategory: string) {
        return { type: TYPES.SET_CURRENT_CATEGORY, payload: currentCategory };
    },
    getSliders() {
        return {
            type: TYPES.GET_SLIDERS,
            payload: getSliders()
        }
    },
+    getLessons() {
+        return (dispatch: any, getState: any) => {
+            (async function () {
+                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() {
+        return (dispatch: any, getState: any) => {
+            (async function () {
+                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\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 downRefresh(element: HTMLDivElement, callback: Function) {
  let startY: number;
  let distance: number;
  let originalTop = element.offsetTop;
  let startTop: number;
  let $timer: any = null;
  element.addEventListener("touchstart", function (event: TouchEvent) {
    if ($timer) clearInterval($timer);
    let touchMove = throttle(_touchMove, 30);

    if (element.scrollTop === 0) {
      startTop = element.offsetTop;
      startY = event.touches[0].pageY;
      element.addEventListener("touchmove", touchMove);
      element.addEventListener("touchend", touchEnd);
    }

    function _touchMove(event: TouchEvent) {
      let pageY = event.touches[0].pageY;
      if (pageY > startY) {
        distance = pageY - startY;
        element.style.top = startTop + distance + "px";
      } else {
        element.removeEventListener("touchmove", touchMove);
        element.removeEventListener("touchend", touchEnd);
      }
    }

    function touchEnd(_event: TouchEvent) {
      element.removeEventListener("touchmove", touchMove);
      element.removeEventListener("touchend", touchEnd);
      if (distance > 30) {
        callback();
      }
      $timer = setInterval(() => {
        let currentTop = element.offsetTop;
        if (currentTop - originalTop > 1) {
          element.style.top = currentTop - 1 + "px";
        } else {
          element.style.top = originalTop + "px";
        }
      }, 13);
    }
  });
}

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();
    }
  };
}

export const store = {
  set(key: string, val: string) {
    sessionStorage.setItem(key, val);
  },
  get(key: string) {
    return sessionStorage.getItem(key);
  },
};
```

src\routes\Home\components\LessonList\index.tsx

```
import React, { useEffect, forwardRef, useState } from "react";
import "./index.less";
import { Icon, Card, Skeleton, Button, Alert } from "antd";
import { Link } from "react-router-dom";
import { Lesson } from "@/typings/lesson";
interface Props {
  children?: any;
  lessons?: any;
  getLessons?: any;
  container?: any;
}

function LessonList(props: Props, lessonListRef: any) {
  const [_, forceUpdate] = useState(0);
  useEffect(() => {
    if (props.lessons.list.length == 0) {
      props.getLessons();
    }
    lessonListRef.current = () => forceUpdate((x) => x + 1);
  }, []);

  let start = 0;
  let rem = parseInt(document.documentElement.style.fontSize);
  if (props.container.current) {
    let scrollTop = 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={props.lessons.list.length == 0 && props.lessons.loading}
        active
        paragraph={{ rows: 8 }}
      >
        {props.lessons.list.map((lesson: Lesson, 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}元`}
                />
              Card>
            Link>
          ) : (
            <div key={index} style={{ height: `${6.5 * rem}px` }}>div>
          )
        )}
        {props.lessons.hasMore ? (
          <Button
            onClick={props.getLessons}
            loading={props.lessons.loading}
            type="primary"
            block
          >
            {props.lessons.loading ? "" : "加载更多"}
          Button>
        ) : (
          <Alert
            style={{ textAlign: "center" }}
            message="到底了"
            type="warning"
          />
        )}
      Skeleton>
    section>
  );
}
export default forwardRef(LessonList);
```

src\routes\Home\components\LessonList\index.less

```
.lesson-list {
  h2 {
    line-height: 1rem;
    i {
      margin: 0 0.1rem;
    }
  }
  .ant-card.ant-card-bordered.ant-card-hoverable {
    height: 6.5rem;
    overflow: hidden;
  }
}
```

src\routes\Home\index.tsx

```
+import React, { PropsWithChildren, useRef, useEffect } from 'react';
import { connect } from 'react-redux';
import { RouteComponentProps } from 'react-router-dom';
import actions from '@/store/actions/home';
import HomeHeader from './components/HomeHeader';
import { CombinedState } from '@/store/reducers';
import { HomeState } from '@/store/reducers/home';
import HomeSliders from './components/HomeSliders';
import './index.less';
+import LessonList from './components/LessonList';
+import { loadMore, downReferesh,store } from '@/utils';
type StateProps = ReturnType;
type DispatchProps = typeof actions;
interface Params { }
type Props = PropsWithChildren & StateProps & DispatchProps>;
function Home(props: Props) {
    const homeContainerRef = useRef(null);
+    const lessonListRef = useRef(null);
+    useEffect(() => {
+        loadMore(homeContainerRef.current, props.getLessons);
+        downReferesh(homeContainerRef.current, props.refreshLessons);
+        homeContainerRef.current.addEventListener('scroll', () => {
+            lessonListRef.current();
+        });
+       if (props.lessons) {
+            homeContainerRef.current.scrollTop = store.get('homeScrollTop');
+        }
+        return () => {
+            store.set('homeScrollTop', homeContainerRef.current.scrollTop);
+        }
+    }, []);
    return (
        <>

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

        </>
    )
}
let mapStateToProps = (state: CombinedState): HomeState => 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);
+}
```

9.详情页

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 Mine from "./routes/Mine";
import Profile from "./routes/Profile";
import Register from "./routes/Register";
import Login from "./routes/Login";
+import Detail from "./routes/Detail";
import { ConnectedRouter } from 'connected-react-router';
import history from './store/history';
ReactDOM.render(

+


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

src\api\home.tsx

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

src\typings\lesson.tsx

```
import { Lesson } from "@/typings/lesson";
export interface LessonResult {
  data: Lesson;
  success: boolean;
}
```

src\routes\Detail\index.tsx

```
import React, { useState, useEffect } from 'react';
import { connect } from 'react-redux';
import { Card, Button } from 'antd';
import NavHeader from "@/components/NavHeader";
import { getLesson } from '@/api/home';
import { RouteComponentProps } from 'react-router';
import { Lesson } from '@/typings/lesson';
import { StaticContext } from 'react-router';
import { LessonResult } from '@/typings/lesson';
const { Meta } = Card;
interface Params { id: string }
type RouteProps = RouteComponentProps;
type Props = RouteProps & {
    children?: any
}

function Detail(props: Props) {
    let [lesson, setLesson] = useState({} as Lesson);
    useEffect(() => {
        (async () => {
            let lesson: Lesson = props.location.state;
            if (!lesson) {
                let id = props.match.params.id;
                let result: LessonResult = await getLesson(id);
                if (result.success)
                    lesson = result.data;
            }
            setLesson(lesson);
        })();
    }, []);
    return (
        <>
            <NavHeader history={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);
```

10.购物车

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, persistor } from "./store";
import { ConfigProvider ,Spin} 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 Mine from "./routes/Mine";
+import Cart from "./routes/Cart";
import Profile from "./routes/Profile";
import Register from "./routes/Register";
import Login from "./routes/Login";
import Detail from "./routes/Detail";
import { ConnectedRouter } from 'connected-react-router';
+import { PersistGate } from 'redux-persist/integration/react'
import history from './store/history';
ReactDOM.render(

+

+

+
    ,
    document.getElementById("root")
);
```

src\routes\Detail\index.tsx

```
import React, { useState, useEffect, PropsWithChildren } from 'react';
import { connect } from 'react-redux';
import { Card, Button, Icon } from 'antd';
import NavHeader from "@/components/NavHeader";
import { getLesson } from '@/api/home';
import { RouteComponentProps } from 'react-router';
import { Lesson } from '@/typings/lesson';
import { StaticContext } from 'react-router';
import { LessonResult } from '@/typings/lesson';
import actions from '@/store/actions/cart';
import { CombinedState } from '@/store/reducers';
```

```
import './index.less';
const { Meta } = Card;
interface Params { id: string }
+type RouteProps = RouteComponentProps;
+type StateProps = ReturnType;
+type DispatchProps = typeof actions;
+type Props = PropsWithChildren;
function Detail(props: Props) {
    let [lesson, setLesson] = useState({} as Lesson);
    useEffect(() => {
        (async () => {
            let lesson: Lesson = props.location.state;
            if (!lesson) {
                let id = props.match.params.id;
                let result: LessonResult = await getLesson(id);
                if (result.success)
                    lesson = result.data;
            }
            setLesson(lesson);
        })();
    }, []);
+    const addCartItem = (lesson: Lesson) => {
+        //https://developer.mozilla.org/zh-CN/docs/Web/API/Element/getBoundingClientRect
+        let video: HTMLVideoElement = document.querySelector('#lesson-video');
+
+        let cart: HTMLSpanElement = document.querySelector('.anticon.anticon-solution');

+        let clonedVideo: HTMLVideoElement = video.cloneNode(true) as HTMLVideoElement;
+        let videoWith = video.offsetWidth;
+        let videoHeight = video.offsetHeight;
+        let cartWith = cart.offsetWidth;
+        let cartHeight = cart.offsetHeight;
+        let videoLeft = video.getBoundingClientRect().left;
+        let videoTop = video.getBoundingClientRect().top;
+        let cartRight = cart.getBoundingClientRect().right;
+        let cartBottom = cart.getBoundingClientRect().bottom;
+        clonedVideo.style.cssText = `
+          z-index: 1000;
+          opacity:0.8;
+          position:fixed;
+          width:${videoWith}px;
+          height:${videoHeight}px;
+          top:${videoTop}px;
+          left:${videoLeft}px;
+          transition: all 2s ease-in-out;
+        `;
+        document.body.appendChild(clonedVideo);
+        setTimeout(function () {
+            clonedVideo.style.left = (cartRight - (cartWith / 2)) + 'px';
+            clonedVideo.style.top = (cartBottom - (cartHeight / 2)) + 'px';
+            clonedVideo.style.width = `0px`;
+            clonedVideo.style.height = `0px`;
+            clonedVideo.style.opacity = '50';
+        }, 0);
+        props.addCartItem(lesson);
+    }
    return (
        <>
            课程详情
            +                /*  cover={} */
+                cover={}
            >
+
+                    <>
+                        价格: ¥{lesson.price}元
+
+
+                                    className="add-cart"
+                                    icon="shopping-cart"
+                                    onClick={() => addCartItem(lesson)}
+                                >加入购物车
+                    </>
+                } />
+
        </>
    )
}
+let mapStateToProps = (state: CombinedState): CombinedState => state;
export default connect(
+    mapStateToProps,
+    actions
)(Detail);

/**
 * https://cubic-bezier.com/#0,0,1,1
 * linear:cubic-bezier(0,0,1,1)             匀速运动
 * ease:cubic-bezier(0.25,0.1,0.25,1)       先慢后快再慢
 * ease-in:cubic-bezier(0.42,0,1,1)         先慢后快
 * ease-out:cubic-bezier(0,0,0.58,1)        先快后慢
 * ease-in-out:cubic-bezier(0.42,0,0.58,1)  先慢后快再慢
 */
```

src\routes\Detail\index.less

```less
button.add-cart{
    &:hover{
        background-color: #F71F40;
        color:#FFF;
    }
}
```

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';

export const GET_SLIDERS = 'GET_SLIDERS';

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';

+export const ADD_CART_ITEM = 'ADD_CART_ITEM';//向购物车中增一个商品
+export const REMOVE_CART_ITEM = 'REMOVE_CART_ITEM';//从购物车中删除一个商品
+export const CLEAR_CART_ITEMS = 'CLEAR_CART_ITEMS';//清空购物车

+export const CHANGE_CART_ITEM_COUNT = 'CHANGE_CART_ITEM_COUNT';//直接修改购物车商品的数量减1

+export const CHANGE_CHECKED_CART_ITEMS = 'CHANGE_CHECKED_CART_ITEMS';//选中商品
+export const SETTLE = 'SETTLE';//结算
```

src\store\index.tsx

```
import { createStore, applyMiddleware, Store, AnyAction, $CombinedState } from 'redux';
import reducers, { CombinedState } from './reducers';
import logger from 'redux-logger';
import { Dispatch } from 'redux';
import thunk, { ThunkDispatch } from 'redux-thunk';
import promise from 'redux-promise';
import { routerMiddleware } from 'connected-react-router';
+import { persistStore, persistReducer } from 'redux-persist';
+import storage from 'redux-persist/lib/storage';
import history from './history';
+const persistConfig = {
+    key: 'root',
+    storage,
+    whitelist: ['cart']
+}
+const persistedReducer = persistReducer(persistConfig, reducers)
+let store: Store = createStore(persistedReducer, applyMiddleware(thunk, routerMiddleware(history), promise, logger));
+let persistor = persistStore(store);
+export type StoreGetState = () => CombinedState;
+export type StoreDispatch = Dispatch & ThunkDispatch;
+export { store, persistor };
```

src\store\actions\home.tsx

```
import * as TYPES from '../action-types';
import { getSliders, getLessons } from '@/api/home';
+import { StoreGetState, StoreDispatch } from '../index';
export default {
    setCurrentCategory(currentCategory: string) {
        return { type: TYPES.SET_CURRENT_CATEGORY, payload: currentCategory };
    },
    getSliders() {
        return {
            type: TYPES.GET_SLIDERS,
            payload: getSliders()
        }
    },
    getLessons() {
+        return (dispatch: StoreDispatch, getState: StoreGetState) => {
            (async function () {
                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() {
+       return (dispatch: StoreDispatch, getState: StoreGetState) => {
            (async function () {
                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\reducers\index.tsx

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

export { CombinedState }
export default reducer;
```

src\typings\cart.tsx

```
import { Lesson } from "./lesson";
export interface CartItem {
  lesson: Lesson;
  count: number;
  checked: boolean;
}
export type CartState = CartItem[];
```

src\store\reducers\cart.tsx

```
import { AnyAction } from "redux";
import { CartState } from "@/typings/cart";
import * as actionTypes from "@/store/action-types";
let initialState: CartState = [];
export default function (
  state: CartState = initialState,
  action: AnyAction
): CartState {
  switch (action.type) {
    case actionTypes.ADD_CART_ITEM:
      let oldIndex = state.findIndex(
        (item) => item.lesson.id === action.payload.id
      );
      if (oldIndex == -1) {
        return [
          ...state,
          {
            checked: false,
            count: 1,
            lesson: action.payload,
          },
        ];
      } else {
        let lesson = state[oldIndex];
        return [
          ...state.slice(0, oldIndex),
          { ...lesson, count: lesson.count + 1 },
          ...state.slice(oldIndex + 1),
        ];
      }
    case actionTypes.REMOVE_CART_ITEM:
      let removeIndex = state.findIndex(
        (item) => item.lesson.id === action.payload
      );
      return [...state.slice(0, removeIndex), ...state.slice(removeIndex + 1)];
    case actionTypes.CLEAR_CART_ITEMS:
      return [];
    case actionTypes.CHANGE_CART_ITEM_COUNT:
      return state.map((item) => {
        if (item.lesson.id === action.payload.id) {
          item.count = action.payload.count;
        }
        return item;
      });
    case actionTypes.CHANGE_CHECKED_CART_ITEMS:
      let checkedIds = action.payload;
      return state.map((item) => {
        if (checkedIds.includes(item.lesson.id)) {
          item.checked = true;
        } else {
          item.checked = false;
        }
        return item;
      });
    case actionTypes.SETTLE:
      return state.filter((item) => !item.checked);
    default:
      return state;
  }
}
```

src\store\actions\cart.tsx

```
import * as actionTypes from "../action-types";
import { Lesson } from "@/typings/lesson";
import { message } from "antd";
import { push } from "connected-react-router";
import { StoreGetState, StoreDispatch } from "../index";
export default {
  addCartItem(lesson: Lesson) {
    return function (dispatch: StoreDispatch) {
      dispatch({
        type: actionTypes.ADD_CART_ITEM,
        payload: lesson,
      });
      message.info("添加课程成功");
    };
  },
  removeCartItem(id: string) {
    return {
      type: actionTypes.REMOVE_CART_ITEM,
      payload: id,
    };
  },
  clearCartItems() {
    return {
      type: actionTypes.CLEAR_CART_ITEMS,
    };
  },
  changeCartItemCount(id: string, count: number) {
    return {
      type: actionTypes.CHANGE_CART_ITEM_COUNT,
      payload: {
        id,
        count,
      },
    };
  },
  changeCheckedCartItems(checkedIds: string[]) {
    return {
      type: actionTypes.CHANGE_CHECKED_CART_ITEMS,
      payload: checkedIds,
    };
  },
  settle() {
    return function (dispatch: StoreDispatch, getState: StoreGetState) {
      dispatch({
        type: actionTypes.SETTLE,
      });
      dispatch(push("/"));
    };
  },
};
```

src\routes\Cart\index.tsx

```
import React, { PropsWithChildren, useState } from "react";
import { connect } from "react-redux";
import { RouteComponentProps } from "react-router-dom";
import {
  Table,
  Button,
  InputNumber,
  Popconfirm,
  Icon,
  Row,
  Col,
  Badge,
  Modal,
} from "antd";
import { CombinedState } from "@/store/reducers";
import NavHeader from "@/components/NavHeader";
import { Lesson } from "@/typings/lesson";
import { StaticContext } from "react-router";
import actions from "@/store/actions/cart";
import { CartItem } from "@/typings/cart";
interface Params {
  id: string;
}
type RouteProps = RouteComponentProps;
interface Params {
  id: string;
}
type StateProps = ReturnType<typeof mapStateToProps>;
type DispatchProps = typeof actions;
type Props = PropsWithChildren;
function Cart(props: Props) {
  let [settleVisible, setSettleVisible] = useState(false);
  const confirmSettle = () => {
    setSettleVisible(true);
  };
  const handleOk = () => {
    setSettleVisible(false);
    props.settle();
  };
  const handleCancel = () => {
    setSettleVisible(false);
  };
  const columns = [
    {
      title: "商品",
      dataIndex: "lesson",
      render: (val: Lesson, row: CartItem) => (
        <>
          <p>{val.title}p>
          <p>单价:{val.price}p>
        </>
```

```
      ),
    },
    {
      title: "数量",
      dataIndex: "count",
      render: (val: number, row: CartItem) => (
        <InputNumber
          size="small"
          min={1}
          max={10}
          value={val}
          onChange={(value) => props.changeCartItemCount(row.lesson.id, value)}
        />
      ),
    },
    {
      title: "操作",
      render: (val: any, row: CartItem) => (
        <Popconfirm
          title="是否要删除商品?"
          onConfirm={() => props.removeCartItem(row.lesson.id)}
          okText="是"
          cancelText="否"
        >
          <Button size="small" type="danger">
            删除
          </Button>
        </Popconfirm>
      ),
    },
  ];
  const rowSelection = {
    selectedRowKeys: props.cart
      .filter((item: CartItem) => item.checked)
      .map((item: CartItem) => item.lesson.id),
    onChange: (selectedRowKeys: string[]) => {
      props.changeCheckedCartItems(selectedRowKeys);
    },
  };
  let totalCount: number = props.cart
    .filter((item: CartItem) => item.checked)
    .reduce((total: number, item: CartItem) => total + item.count, 0);
  let totalPrice = props.cart
    .filter((item: CartItem) => item.checked)
    .reduce(
      (total: number, item: CartItem) =>
        total + Number(item.lesson.price) * item.count,
      0
    );
  return (
    <>
      <NavHeader history={props.history}>购物车NavHeader>
      <Table
        rowKey={(row) => row.lesson.id}
        rowSelection={rowSelection}
        columns={columns}
        dataSource={props.cart}
        pagination={false}
        size="small"
      />
      <Row style={{ padding: "5px" }}>
        <Col span={4}>
          <Button type="danger" size="small" onClick={props.clearCartItems}>
            清空
          </Button>
        </Col>
        <Col span={9}>
          已经选择{totalCount > 0 ? <Badge count={totalCount} /> : 0}件商品
        </Col>
        <Col span={7}>总价: ¥{totalPrice}元Col>
        <Col span={4}>
          <Button type="danger" size="small" onClick={confirmSettle}>
            去结算
          </Button>
        </Col>
      </Row>
      <Modal
        title="去结算"
        visible={settleVisible}
        onOk={handleOk}
        onCancel={handleCancel}
      >
        <p>请问你是否要结算?p>
      </Modal>
    </>
  );
}
let mapStateToProps = (state: CombinedState): CombinedState => state;
export default connect(mapStateToProps, actions)(Cart);
```

1.项目初始化

```
mkdir server
cd server
cnpm init -y
```

```
cnpm i express mongoose body-parser bcryptjs jsonwebtoken morgan cors validator helmet dotenv multer -S
cnpm i typescript  @types/node @types/express @types/mongoose @types/bcryptjs @types/jsonwebtoken  @types/morgan @types/cors @types/validator ts-node-dev
nodemon  @types/helmet @types/multer -D
```

模块名 用途

加载到环境变量

```
npx tsconfig.json
```

```
+  "scripts": {
+    "build": "tsc",
+    "start": "cross-env PORT=8000  ts-node-dev --respawn src/index.ts",
+    "dev": "cross-env PORT=8000 nodemon --exec ts-node --files src/index.ts"
+  }
```

```
node_modules
src/public/upload/
.env
```

```
JWT_SECRET_KEY=zhufeng
MONGODB_URL=mongodb:
```

2.用户管理

src/index.ts

```
import express, { Express, Request, Response, NextFunction } from "express";
import mongoose from "mongoose";
import HttpException from "./exceptions/HttpException";
import cors from "cors";
import morgan from "morgan";
import helmet from "helmet";
import errorMiddleware from "./middlewares/errorMiddleware";
import * as userController from "./controller/user";
import "dotenv/config";
import multer from "multer";
import path from "path";
const storage = multer.diskStorage({
  destination: path.join(__dirname, "public", "uploads"),
  filename(_req: Request, file: Express.Multer.File, cb) {
    cb(null, Date.now() + path.extname(file.originalname));
  },
});
const upload = multer({ storage });
const app: Express = express();
app.use(morgan("dev"));
app.use(cors());
app.use(helmet());
app.use(express.static(path.resolve(__dirname, "public")));
app.use(express.json());
app.use(express.urlencoded({ extended: false }));
app.get("/", (_req: Request, res: Response) => {
  res.json({ success: true, message: "hello world" });
});
app.get("/user/validate", userController.validate);
app.post("/user/register", userController.register);
app.post("/user/login", userController.login);
app.post(
  "/user/uploadAvatar",
  upload.single("avatar"),
  userController.uploadAvatar
);
app.use((_req: Request, _res: Response, next: NextFunction) => {
  const error: HttpException = new HttpException(404, "Route not found");
  next(error);
});
app.use(errorMiddleware);
const PORT: number = (process.env.PORT && parseInt(process.env.PORT)) || 8000;
(async function () {
  mongoose.set("useNewUrlParser", true);
  mongoose.set("useUnifiedTopology", true);
  await mongoose.connect("mongodb://localhost/zhufengketang");
  app.listen(PORT, () => {
    console.log(`Running on http://localhost:${PORT}`);
  });
})();
```

src/exceptions/HttpException.ts

```
class HttpException extends Error {
    constructor(public status: number, public message: string, public errors?: any) {
        super(message);
    }
}
export default HttpException;
```

src/middlewares/errorMiddleware.ts

```
import HttpException from "../exceptions/HttpException";
import { Request, Response, NextFunction } from "express";
import { INTERNAL_SERVER_ERROR } from "http-status-codes";
const errorMiddleware = (
  error: HttpException,
  _request: Request,
  response: Response,
  _next: NextFunction
) => {
  response.status(error.status || INTERNAL_SERVER_ERROR).send({
    success: false,
    message: error.message,
    errors: error.errors,
  });
};
export default errorMiddleware;
```

src/utils/validator.ts

```ts
import validator from "validator";
import { IUserDocument } from "../models/user";

export interface RegisterInput extends Partial {
  confirmPassword?: string;
}

export interface RegisterInputValidateResult {
  errors: RegisterInput;
  valid: boolean;
}

export const validateRegisterInput = (
  username: string,
  password: string,
  confirmPassword: string,
  email: string
): RegisterInputValidateResult => {
  let errors: RegisterInput = {};
  if (username == undefined || validator.isEmpty(username)) {
    errors.username = "用户名不能为空";
  }
  if (password == undefined || validator.isEmpty(password)) {
    errors.password = "密码不能为空";
  }
  if (confirmPassword == undefined || validator.isEmpty(confirmPassword)) {
    errors.password = "确认密码不能为空";
  }
  if (!validator.equals(password, confirmPassword)) {
    errors.confirmPassword = "确认密码和密码不相等";
  }
  if (email == undefined || validator.isEmpty(password)) {
    errors.email = "邮箱不能为空";
  }
  if (validator.isEmail(password)) {
    errors.email = "邮箱格式必须合法";
  }
  return { errors, valid: Object.keys(errors).length == 0 };
};
```

src/typings/jwt.ts

```ts
import { IUserDocument } from "../models/user";

export interface UserPayload {
    id: IUserDocument['_id']
}
```

src\models\index.ts

```ts
export * from "./user";
```

src/models/user.ts

```ts
import mongoose, { Schema, Model, Document, HookNextFunction } from 'mongoose';
import validator from 'validator';
import jwt from 'jsonwebtoken';
import { UserPayload } from '../typings/jwt';
import bcrypt from 'bcryptjs';
export interface IUserDocument extends Document {
    username: string,
    password: string,
    email: string;
    avatar: string;
    generateToken: () => string,
    _doc: IUserDocument
}
const UserSchema: Schema = new Schema({
    username: {
        type: String,
        required: [true, '用户名不能为空'],
        minlength: [6, '最小长度不能少于6位'],
        maxlength: [12, '最大长度不能大于12位']
    },
    password: String,
    avatar: String,
    email: {
        type: String,
        validate: {
            validator: validator.isEmail
        },
        trim: true,
    }
}, { timestamps: true });

UserSchema.methods.generateToken = function (): string {
    let payload: UserPayload = ({ id: this._id });
    return jwt.sign(payload, process.env.JWT_SECRET_KEY!, { expiresIn: '1h' });
}
UserSchema.pre('save', async function (next: HookNextFunction) {
    if (!this.isModified('password')) {
        return next();
    }
    try {
        this.password = await bcrypt.hash(this.password, 10);
        next();
    } catch (error) {
        next(error);
    }
});
UserSchema.static('login', async function (this: any, username: string, password: string): Promise<IUserDocument | null> {
    let user: IUserDocument | null = await this.model('User').findOne({ username });
    if (user) {
        const matched = await bcrypt.compare(password, user.password);
        if (matched) {
            return user;
        } else {
            return null;
        }
    }
    return user;
});
interface IUserModel extends Model {
    login: (username: string, password: string) => IUserDocument | null
}
export const User: IUserModel = mongoose.model<IUserModel>('User', UserSchema);
```

src\controller\user.ts

```ts
import { Request, Response, NextFunction } from 'express';
import { validateRegisterInput } from '../utils/validator';
import HttpException from '../exceptions/HttpException';
import { UNPROCESSABLE_ENTITY, UNAUTHORIZED } from 'http-status-codes';
import { IUserDocument, User } from '../models/user';
import { UserPayload } from '../typings/jwt';
import jwt from 'jsonwebtoken';
export const validate = async (req: Request, res: Response, next: NextFunction) => {
    const authorization = req.headers['authorization'];
    if (authorization) {
        const token = authorization.split(' ')[1];
        if (token) {
            try {
                const payload: UserPayload = jwt.verify(token, process.env.JWT_SECRET_KEY!) as UserPayload;
                const user = await User.findById(payload.id);
                if (user) {
                    delete user.password;
                    res.json({
                        success: true,
                        data: user
                    });
                } else {
                    next(new HttpException(UNAUTHORIZED, `用户不合法!`));
                }
            } catch (error) {
                next(new HttpException(UNAUTHORIZED, `token不合法!`));
            }

        } else {
            next(new HttpException(UNAUTHORIZED, `token未提供!`));
        }
    } else {
        next(new HttpException(UNAUTHORIZED, `authorization未提供!`));
    }
}
export const register = async (req: Request, res: Response, next: NextFunction) => {
    try {
        let { username, password, confirmPassword, email, addresses } = req.body;
        const { valid, errors } = validateRegisterInput(username, password, confirmPassword, email);
        if (!valid) {
            throw new HttpException(UNPROCESSABLE_ENTITY, `参数验证失败!`, errors);
        }
        let user: IUserDocument = new User({
            username,
            email,
            password,
            addresses
        });
        let oldUser: IUserDocument | null = await User.findOne({ username: user.username });
        if (oldUser) {
            throw new HttpException(UNPROCESSABLE_ENTITY, `用户名重复!`);
        }
        await user.save();
        let token = user.generateToken();
        res.json({
            success: true,
            data: { token }
        });
    } catch (error) {
        next(error);
    }
}

export const login = async (req: Request, res: Response, next: NextFunction) => {
    try {
        let { username, password } = req.body;
        let user = await User.login(username, password);
        if (user) {
            let token = user.generateToken();
            res.json({
                success: true,
                data: {
                    token
                }
            });
        } else {
            throw new HttpException(UNAUTHORIZED, `登录失败`);
        }
    } catch (error) {
        next(error);
    }
}
export const uploadAvatar = async (req: Request, res: Response, next: NextFunction) => {
    let { userId } = req.body;
    let avatar = `${req.protocol}://${req.headers.host}/uploads/${req.file.filename}`;
    await User.updateOne({ _id: userId }, { avatar });
    res.send({ success: true, data: avatar });
}
```

src\typings\express.d.ts

```ts
import { IUserDocument } from "../models/user";
declare global {
    namespace Express {
        export interface Request {
            currentUser?: IUserDocument | null;
            file: Multer.File
        }
    }
}
```

3.轮播图

src\index.ts

```
import express, { Express, Request, Response, NextFunction } from 'express';
import mongoose from 'mongoose';
import HttpException from './exceptions/HttpException';
import cors from 'cors';
import morgan from 'morgan';
import helmet from 'helmet';
import errorMiddleware from './middlewares/errorMiddleware';
import *  as userController from './controller/user';
+import *  as sliderController from './controller/slider';
import "dotenv/config";
import multer from 'multer';
import path from 'path';
+import { Slider } from './models';
const storage = multer.diskStorage({
    destination: path.join(__dirname, 'public', 'uploads'),
    filename(_req: Request, file: Express.Multer.File, cb) {
        cb(null, Date.now() + path.extname(file.originalname));
    }
});
const upload = multer({ storage });
const app: Express = express();
app.use(morgan("dev"));
app.use(cors());
app.use(helmet());
app.use(express.static(path.resolve(__dirname, 'public')));
app.use(express.json());
app.use(express.urlencoded({ extended: false }));
app.get('/', (_req: Request, res: Response) => {
    res.json({ success: true, message: 'hello world' });
});
app.get('/user/validate', userController.validate);
app.post('/user/register', userController.register);
app.post('/user/login', userController.login);
app.post('/user/uploadAvatar', upload.single('avatar'), userController.uploadAvatar);
+app.get('/slider/list', sliderController.list);
app.use((_req: Request, _res: Response, next: NextFunction) => {
    const error: HttpException = new HttpException(404, 'Route not found');
    next(error);
});
app.use(errorMiddleware);
const PORT: number = (process.env.PORT && parseInt(process.env.PORT)) || 8000;
(async function () {
    mongoose.set('useNewUrlParser', true);
    mongoose.set('useUnifiedTopology', true);
    await mongoose.connect(process.env.MONGODB_URL!);
+    await createSliders();
    app.listen(PORT, () => {
        console.log(`Running on http://localhost:${PORT}`);
    });
})();

+async function createSliders() {
+    const sliders = await Slider.find();
+    if (sliders.length == 0) {
+        const sliders = [
+            { url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/reactnative.png' },
+            { url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/react.png' },
+            { url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/vue.png' },
+            { url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/wechat.png' },
+            { url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/architect.jpg' }
+        ];
+        Slider.create(sliders);
+    }
+}
```

src\controller\slider.ts

```
import { Request, Response } from "express";
import { ISliderDocument, Slider } from "../models";
export const list = async (_req: Request, res: Response) => {
  let sliders: ISliderDocument[] = await Slider.find();
  res.json({ success: true, data: sliders });
};
```

src\models\slider.ts

```
import mongoose, { Schema, Document } from "mongoose";
export interface ISliderDocument extends Document {
  url: string;
  _doc: ISliderDocument;
}
const SliderSchema: Schema = new Schema(
  {
    url: String,
  },
  { timestamps: true }
);

export const Slider =
  mongoose.model < ISliderDocument > ("Slider", SliderSchema);
```

src\models\index.ts

```
export * from './user';
+export * from './slider';
```

4.课程管理

src\index.ts

```
import express, { Express, Request, Response, NextFunction } from 'express';
import mongoose from 'mongoose';
import HttpException from './exceptions/HttpException';
import cors from 'cors';
import morgan from 'morgan';
```

```
import helmet from 'helmet';
import errorMiddleware from './middlewares/errorMiddleware';
import *  as userController from './controller/user';
import *  as sliderController from './controller/slider';
+import *  as lessonController from './controller/lesson';
import "dotenv/config";
import multer from 'multer';
import path from 'path';
+import { Slider, Lesson } from './models';
const storage = multer.diskStorage({
    destination: path.join(__dirname, 'public', 'uploads'),
    filename(_req: Request, file: Express.Multer.File, cb) {
        cb(null, Date.now() + path.extname(file.originalname));
    }
});
const upload = multer({ storage });
const app: Express = express();
app.use(morgan("dev"));
app.use(cors());
app.use(helmet());
app.use(express.static(path.resolve(__dirname, 'public')));
app.use(express.json());
app.use(express.urlencoded({ extended: false }));
app.get('/', (_req: Request, res: Response) => {
    res.json({ success: true, message: 'hello world' });
});
app.get('/user/validate', userController.validate);
app.post('/user/register', userController.register);
app.post('/user/login', userController.login);
app.post('/user/uploadAvatar', upload.single('avatar'), userController.uploadAvatar);
app.get('/slider/list', sliderController.list);
+app.get('/lesson/list', lessonController.list);
+app.get('/lesson/:id', lessonController.get);
app.use((_req: Request, _res: Response, next: NextFunction) => {
    const error: HttpException = new HttpException(404, 'Route not found');
    next(error);
});
app.use(errorMiddleware);
const PORT: number = (process.env.PORT && parseInt(process.env.PORT)) || 8000;
(async function () {
    mongoose.set('useNewUrlParser', true);
    mongoose.set('useUnifiedTopology', true);
    await mongoose.connect(process.env.MONGODB_URL!);
    await createSliders();
+    await createLessons();
    app.listen(PORT, () => {
        console.log(`Running on http://localhost:${PORT}`);
    });
})();

async function createSliders() {
    const sliders = await Slider.find();
    if (sliders.length == 0) {
        const sliders = [
            { url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/reactnative.png' },
            { url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/react.png' },
            { url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/vue.png' },
            { url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/wechat.png' },
            { url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/architect.jpg' }
        ];
        Slider.create(sliders);
    }
}

+async function createLessons() {
+    const lessons = await Lesson.find();
+    if (lessons.length == 0) {
        const lessons = [
            {
                order: 1,
                title: '1.React全栈架构',
                video: "http://img.zhufengpeixun.cn/gee2.mp4",
                poster: "http://www.zhufengpeixun.cn/react/img/react.jpg",
                url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/react.png',
                price: '¥100.00元',
                category: 'react'
            },
            {
                order: 2,
                title: '2.React全栈架构',
                video: "http://img.zhufengpeixun.cn/gee2.mp4",
                poster: "http://www.zhufengpeixun.cn/react/img/react.jpg",
                url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/react.png',
                price: '¥200.00元',
                category: 'react'
            },
            {
                order: 3,
                title: '3.React全栈架构',
                video: "http://img.zhufengpeixun.cn/gee2.mp4",
                poster: "http://www.zhufengpeixun.cn/react/img/react.jpg",
                url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/react.png',
                price: '¥300.00元',
                category: 'react'
            },
            {
                order: 4,
                title: '4.React全栈架构',
                video: "http://img.zhufengpeixun.cn/gee2.mp4",
                poster: "http://www.zhufengpeixun.cn/react/img/react.jpg",
                url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/react.png',
                price: '¥400.00元',
                category: 'react'
            },
```

```
{
    order: 5,
    title: '5.React全栈架构',
    video: "http://img.zhufengpeixun.cn/gee2.mp4",
    poster: "http://www.zhufengpeixun.cn/react/img/react.jpg",
    url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/react.png',
    price: '¥500.00元',
    category: 'react'
},
{
    order: 6,
    title: '6.Vue从入门到项目实战',
    video: "http://img.zhufengpeixun.cn/gee2.mp4",
    poster: "http://www.zhufengpeixun.cn/vue/img/vue.png",
    url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/vue.png',
    price: '¥100.00元',
    category: 'vue'
},
{
    order: 7,
    title: '7.Vue从入门到项目实战',
    video: "http://img.zhufengpeixun.cn/gee2.mp4",
    poster: "http://www.zhufengpeixun.cn/vue/img/vue.png",
    url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/vue.png',
    price: '¥200.00元',
    category: 'vue'
},
{
    order: 8,
    title: '8.Vue从入门到项目实战',
    video: "http://img.zhufengpeixun.cn/gee2.mp4",
    poster: "http://www.zhufengpeixun.cn/vue/img/vue.png",
    url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/vue.png',
    price: '¥300.00元',
    category: 'vue'
},
{
    order: 9,
    title: '9.Vue从入门到项目实战',
    video: "http://img.zhufengpeixun.cn/gee2.mp4",
    poster: "http://www.zhufengpeixun.cn/vue/img/vue.png",
    url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/vue.png',
    price: '¥400.00元',
    category: 'vue'
},
{
    order: 10,
    title: '10.Vue从入门到项目实战',
    video: "http://img.zhufengpeixun.cn/gee2.mp4",
    poster: "http://www.zhufengpeixun.cn/vue/img/vue.png",
    url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/vue.png',
    price: '¥500.00元',
    category: 'vue'
},
{
    order: 11,
    title: '11.React全栈架构',
    "video": "http://img.zhufengpeixun.cn/gee2.mp4",
    poster: "http://www.zhufengpeixun.cn/react/img/react.jpg",
    url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/react.png',
    price: '¥600.00元',
    category: 'react'
},
{
    order: 12,
    title: '12.React全栈架构',
    video: "http://img.zhufengpeixun.cn/gee2.mp4",
    poster: "http://www.zhufengpeixun.cn/react/img/react.jpg",
    url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/react.png',
    price: '¥700.00元',
    category: 'react'
},
{
    order: 13,
    title: '13.React全栈架构',
    video: "http://img.zhufengpeixun.cn/gee2.mp4",
    poster: "http://www.zhufengpeixun.cn/react/img/react.jpg",
    url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/react.png',
    price: '¥800.00元',
    category: 'react'
},
{
    order: 14,
    title: '14.React全栈架构',
    video: "http://img.zhufengpeixun.cn/gee2.mp4",
    poster: "http://www.zhufengpeixun.cn/react/img/react.jpg",
    url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/react.png',
    price: '¥900.00元',
    category: 'react'
},
{
    order: 15,
    title: '15.React全栈架构',
    video: "http://img.zhufengpeixun.cn/gee2.mp4",
    poster: "http://www.zhufengpeixun.cn/react/img/react.jpg",
    url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/react.png',
    price: '¥1000.00元',
    category: 'react'
},
{
    order: 16,
    title: '16.Vue从入门到项目实战',
    video: "http://img.zhufengpeixun.cn/gee2.mp4",
    poster: "http://www.zhufengpeixun.cn/vue/img/vue.png",
```

```
                url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/vue.png',
                price: '¥600.00元',
                category: 'vue'
            },
            {
                order: 17,
                title: '17.Vue从入门到项目实战',
                video: "http://img.zhufengpeixun.cn/gee2.mp4",
                poster: "http://www.zhufengpeixun.cn/vue/img/vue.png",
                url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/vue.png',
                price: '¥700.00元',
                category: 'vue'
            },
            {
                order: 18,
                title: '18.Vue从入门到项目实战',
                video: "http://img.zhufengpeixun.cn/gee2.mp4",
                poster: "http://www.zhufengpeixun.cn/vue/img/vue.png",
                url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/vue.png',
                price: '¥800.00元',
                category: 'vue'
            },
            {
                order: 19,
                title: '19.Vue从入门到项目实战',
                video: "http://img.zhufengpeixun.cn/gee2.mp4",
                poster: "http://www.zhufengpeixun.cn/vue/img/vue.png",
                url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/vue.png',
                price: '¥900.00元',
                category: 'vue'
            },
            {
                order: 20,
                title: '20.Vue从入门到项目实战',
                video: "http://img.zhufengpeixun.cn/gee2.mp4",
                poster: "http://www.zhufengpeixun.cn/vue/img/vue.png",
                url: 'http://www.zhufengpeixun.cn/themes/jianmo2/images/vue.png',
                price: '¥1000.00元',
                category: 'vue'
            }
        ];
+       Lesson.create(lessons);
    }
}
```

src\models\index.ts

```
export * from './user';
export * from './slider';
+export * from './lesson';
```

src\controller\lesson.ts

```
import { Request, Response } from 'express';
import { ILessonDocument, Lesson } from '../models';
export const list = async (req: Request, res: Response) => {
    let { offset, limit, category } = req.query;
    offset = isNaN(offset) ? 0 : parseInt(offset);
    limit = isNaN(limit) ? 5 : parseInt(limit);
    let query: Partial = {} as ILessonDocument;
    if (category && category != 'all')
        query.category = category;
    let total = await Lesson.count(query);
    let list = await Lesson.find(query).sort({ order: 1 }).skip(offset).limit(limit);
    setTimeout(function () {
        res.json({ code: 0, data: { list, hasMore: total > offset + limit } });
    }, 1000);
}
export const get = async (req: Request, res: Response) => {
    let id = req.params.id;
    let lesson = await Lesson.findById(id);
    res.json({ success: true, data: lesson });
}
```

src\models\lesson.ts

```
import mongoose, { Schema, Document } from "mongoose";
export interface ILessonDocument extends Document {
  order: number;
  title: string;
  video: string;
  poster: string;
  url: string;
  price: string;
  category: string;
  _doc: ILessonDocument;
}
const LessonSchema: Schema = new Schema(
  {
    order: Number,
    title: String,
    video: String,
    poster: String,
    url: String,
    price: String,
    category: String,
  },
  { timestamps: true }
);

export const Lesson =
  mongoose.model < ILessonDocument > ("Lesson", LessonSchema);
```