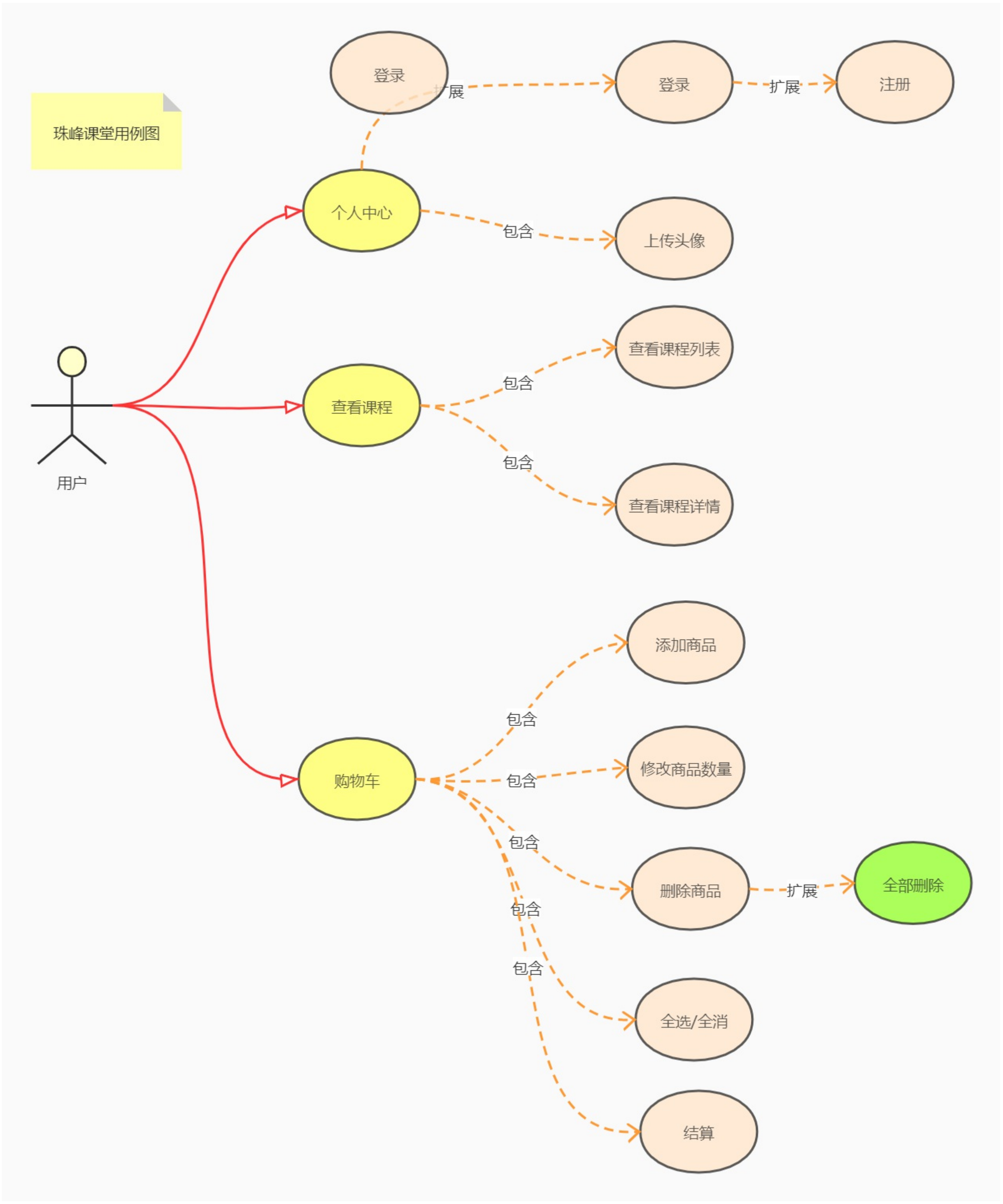
link: null
title: 珠峰架构师成长计划
description: null
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=597 sentences=2535, words=16603

1.项目介绍

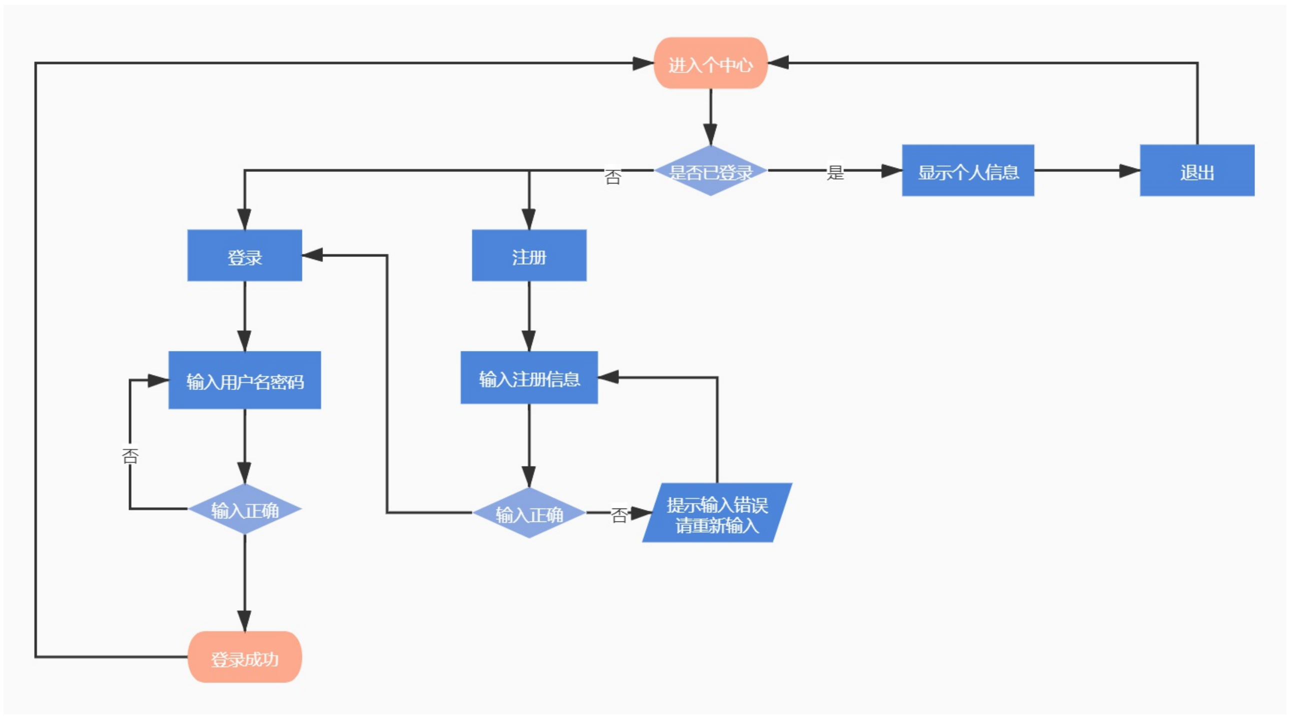
1.1 用例图

- 用例图(use case diagram)是用户与系统交互的最简表示形式，展现了用户和与他相关的用例之间的关系

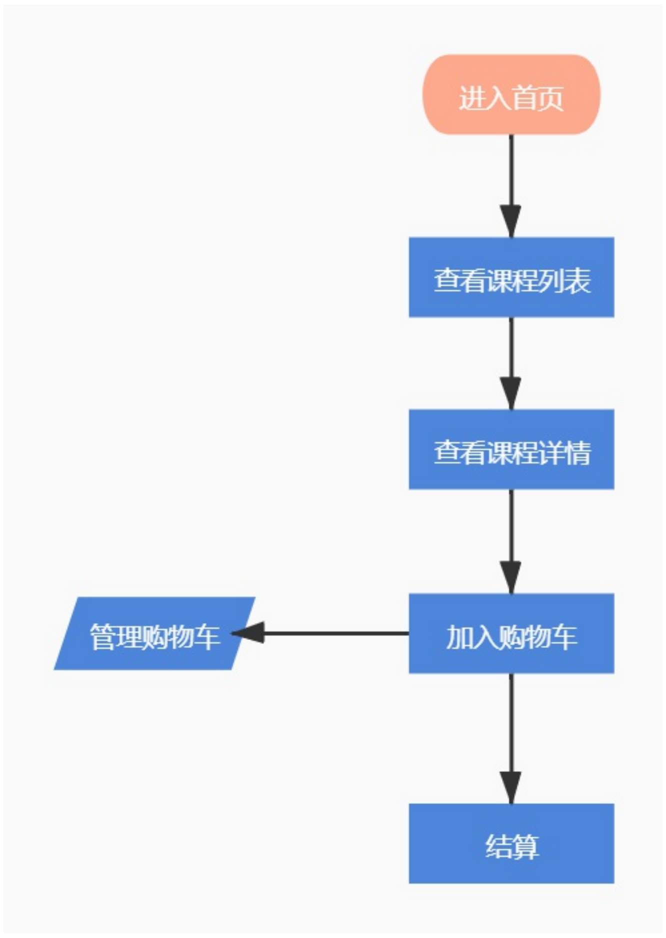


1.2 流程图

1.2.1 注册登录



1.2.2 购买课程



2. 搭建开发环境

2.1 本节目录

```
-- client
    |-- package.json
    |-- public
    |   |-- index.html
    |-- src
    |   |-- index.tsx
    |-- tsconfig.json
    |-- webpack.config.js
```

2.2 初始化项目

```
mkdir client
cd client
cnpm init -y
touch .gitignore
```

2.3 安装依赖

- @types开头的包都是typeScript的声明文件，可以进入node_modules/@types/XX/index.d.ts进行查看
- 常见的声明文件 (https://github.com/DefinitelyTyped/DefinitelyTyped)
- ant.design (https://ant.design/components/overview-cn/)

```
cnpm install react react-dom @types/react @types/react-dom react-router-dom @types/react-router-dom @ant-design/icons antd redux react-redux @types/react-redux redux-thunk  redux-logger @types/redux-logger redux-promise @types/redux-promise redux-first-history  classnames @types/classnames react-transition-group @types/react-transition-group express express-session body-parser cors axios  redux-persist immer redux-immer --save

cnpm install webpack webpack-cli webpack-dev-server copy-webpack-plugin html-webpack-plugin babel-loader typescript @babel/core @babel/preset-env @babel/preset-react @babel/preset-typescript  babel-plugin-import style-loader css-loader postcss-loader less-loader less autoprefixer px2rem-loader  lib-flexible eslint @types/eslint file-loader url-loader --save-dev
```

模块名 英文 中文 react React is a JavaScript library for creating user interfaces. React 是一个用于创建用户界面的 JavaScript 库 @types/react This package contains type definitions for React 包含 React 的类型定义 react-dom This package serves as the entry point to the DOM and server renderers for React. It is intended to be paired with the generic React package, which is shipped as react to npm 把 React 渲染到 DOM 上 @types/react-dom This package contains type definitions for React (react-dom) 包含 React (react-dom)的类型定义 react-router-dom DOM bindings for React Router React 路由的 DOM 渲染 @types/react-router-dom This package contains type definitions for React Router React Router 的类型定义 react-transition-group A set of components for managing component states (including mounting and unmounting) over time, specifically designed with animation in mind 一组用于随时间管理组件状态（包括安装和卸载）的组件，特别设计时考虑了动画 @types/react-transition-group This package contains type definitions for react-transition-group react-transition-group 的类型定义 react-swipe Brad Birdsall's Swipe.js as a React component React 轮播图组件 @types/react-swipe This package contains type definitions for react-swipe React 轮播图组件的类型定义 antd An enterprise-class UI design language and React UI library 企业级 UI 设计语言和 React UI 库 qs A querystring parsing and stringifying library with some added security 一个带有一些附加安全性的 querystring 解析和字符串化库 @types/qs This package contains type definitions for qs 该软件包包含 qs 的类型定义 webpack webpack is a module bundler. Its main purpose is to bundle JavaScript files for usage in a browser, yet it is also capable of transforming, bundling, or packaging just about any resource or asset. webpack 是一个模块打包器。它的主要目的是打包 JavaScript 文件以在浏览器中使用，但它也能够转换或打包几乎任何资源 webpack-cli webpack CLI provides a flexible set of commands for developers to increase speed when setting up a custom webpack project. As of webpack v4, webpack is not expecting a configuration file, but often developers want to create a more custom webpack configuration based on their use-cases and needs. webpack CLI addresses these needs by providing a set of tools to improve the setup of custom webpack configuration. webpack cli 提供了一组灵活的命令，供开发人员在设置自定义 webpack 项目时提高速度 webpack-dev-server Use webpack with a development server that provides live reloading. This should be used for development only 将 webpack 与提供实时重载的开发服务器一起使用。 这应该仅用于开发 html-webpack-plugin Plugin that simplifies creation of HTML files to serve your bundles 简化 HTML 文件的创建插件 ts-import-plugin Modular import plugin for TypeScript, compatible with antd, antd-mobile and so on 用于 TypeScript 的模块化导入插件，与 antd，antd-mobile 等兼容 typescript TypeScript is a language for application-scale JavaScript TypeScript 是用于应用程序级 JavaScript 的语言 ts-loader TypeScript loader for webpack 用于 Webpack 的 TypeScript 加载器 source-map-loader Extracts source maps from existing source files (from their sourceMappingURL) 从现有源文件(从其 sourceMappingURL)中提取源映射 style-loader Inject CSS into the DOM 将 CSS 注入 DOM css-loader The css-loader interprets @import and url() like import/require() and will resolve them css-loader 会像 importt()/require()一样解释@import 和 url 并将解析它们 把 less 编译成 CSS less-loader A Less loader for webpack. Compiles Less to CSS 把 less 编译成 CSS less This is the JavaScript, official, stable version of Less 这是 Less 的 JavaScript 官方稳定版本 autoprefixer PostCSS plugin to parse CSS and add vendor prefixes to CSS rules using values from Can I Use. It is recommended by Google and used in Twitter and Alibaba 根据 `can i use`

网站的 CSS 规则给 CSS 规则添加厂商前缀 px2rem-loader a webpack loader for px2rem px2rem 的 Webpack 加载器 postcss-loader Loader for webpack to process CSS with PostCSS 用于 webpack 的 Loader 以使用 PostCSS 处理 CSS lib-flexible 可伸缩布局解决方案 redux Redux is a predictable state container for JavaScript apps Redux 是 JavaScript 应用程序的可预测状态容器 react-redux Official React bindings for Redux Redux 的官方 React 绑定 @types/react-redux his package contains type definitions for react-redux 该软件包包含 react-redux 的类型定义 redux-thunk Thunk middleware for Redux 用于 Redux 的 Thunk 中间件 redux-logger Logger for Redux 用于 Redux 的 logger 中间件 @types/redux-logger This package contains type definitions for redux-logger 该软件包包含 redux-logger 的类型定义 redux-promise FSA-compliant promise middleware for Redux. 符合 FSA 的 Redux 的 promise 中间件 @types/redux-promise This package contains type definitions for redux-promise 该软件包包含 redux-promise 的类型定义 immer Create the next immutable state tree by simply modifying the current tree 通过简单地修改当前树来创建下一个不可变状态树 redux-immer redux-immer is used to create an equivalent function of Redux combineReducers that works with immer state. redux-immer 用于创建 `Redux combineReducers`

的等效功能，该功能可与 `immer`

状态一起使用 redux-first-history A Redux binding for React Router v4 and v5 用于 React Router v4 和 v5 的 Redux 绑定

2.4 支持 typescript

- 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": {
    "moduleResolution":"Node",
    "outDir": "./dist",
    "sourceMap": true,
    "noImplicitAny": true,
    "module": "ESNext",
    "target": "es5",
    "jsx": "react",
    "esModuleInterop":true,
    "baseUrl": ".",
    "paths": {
      "@/*": ["./src/*" ]
    }
  },
  "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";
```

2.5 支持图片

src\typings\images.d.ts

```
declare module '*.svg';
declare module '*.png';
declare module '*.jpg';
declare module '*.jpeg';
declare module '*.gif';
declare module '*.bmp';
declare module '*.tiff';
```

2.6 编写 webpack 配置文件

webpack.config.js

```
const webpack = require("webpack");
const HtmlWebpackPlugin = require("html-webpack-plugin");
const path = require("path");
const CopyWebpackPlugin = require("copy-webpack-plugin");
module.exports = {
    mode: process.env.NODE_ENV == "production" ? "production" : "development",
    entry: "./src/index.tsx",
    output: {
        path: path.join(__dirname, "dist"),
        filename: "main.js",
        outputPath:'/'
    },
    devtool: "source-map",
    devServer: {
        hot: true,
        static: path.join(__dirname, "static"),
        historyApiFallback:true
    },
    resolve: {
        alias: {
            "@": path.resolve(__dirname, "src"),
            "~": path.resolve(__dirname, "node_modules"),
        },

        extensions: [".ts", ".tsx", ".js", ".json"],
    },
    module: {
        rules: [
            {
                test: /\.(j|t)sx?$/,
                loader: "babel-loader",
                options: {
                    presets: [
                        '@babel/preset-env',
                        '@babel/preset-react',
                        '@babel/preset-typescript'
                    ],
                    plugins: [
                        ['import', { libraryName: 'antd', style: 'css' }]
                    ]
                },
                include: path.resolve('src'),
                exclude: /node_modules/
            },
            {
                test: /\.css$/,
                use: [
                    "style-loader",
                    {
                        loader: "css-loader",
                        options: { importLoaders: 0 },
                    }
                ]
            },
            {
                test: /\.less$/,
                use: [
                    "style-loader",
                    {
                        loader: "css-loader",
                        options: { importLoaders: 3 },
                    },
                    {
                        loader: "postcss-loader",
                        options: {
                            postcssOptions: {
                                plugins: [
                                    "autoprefixer"
                                ],
                            }
                        },
                    },
                    {
                        loader: "px2rem-loader",
                        options: {
                            remUnit: 75,
                            remPrecesion: 8,
                        },
                    },
                    "less-loader",
                ],
            },
            {
                test: /\.(jpg|png|gif|svg|jpeg)$/,
                type: 'asset'
            },
        ],
    },
    plugins: [
        new HtmlWebpackPlugin({
            template: "./public/index.html",
        }),
        new CopyWebpackPlugin({
            patterns: [
                { from: path.resolve(__dirname, 'static'), to: path.resolve(__dirname, 'dist') }
            ]
        })
    ],
};
```

2.7 src\index.tsx

src\index.tsx

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

2.8 public\index.html

public\index.html

```

<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <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>
```

2.9 package.json

```
  "scripts": {
    "build": "webpack",
    "dev": "webpack serve",
    "lint": "eslint --ext .tsx src --fix",
  }
```

2.10 移动端适配

2.10.1 设备物理像素

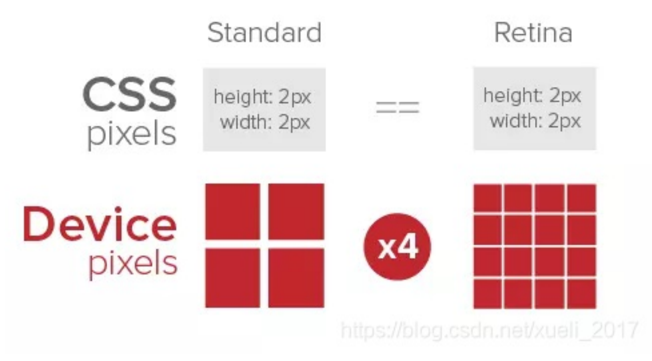
- 是一个物理概念，是显示器显示的最小物理单位
- iPhone6的像素分辨率是750*1334`
- px是一个相对单位，相对的是设备像素(device pixel)

2.10.2 设备独立像素

- 是一个逻辑概念,用于向CSS中的宽度、高度等提供信息
- iPhone6的逻辑分辨率是 375*667

2.10.3 设备像素比

- DPR(设备像素比) = 设备像素/CSS像素
- 设备像素比 window.devicePixelRatio



2.10.4 移动端适配

- 一般由设计师按照 iPhone6的像素分辨率是 750*1334制作设计稿
- 然后由前端工程师参进行换算

2.10.5 rem

- 参照根元素的字体大小
- 适配就是让根元素的字体大小根据分辨率进行动态改变
- px2rem-loader (https://www.npmjs.com/package/px2rem-loader)

2.10.6 vw和vh

- 参照的是viewport视口
- vw参照的是视口的宽度(1vw=视口宽度/100)
- vh参照的是视口的高度(1vh=视口高度/100)
- iPhone6 1vw=3.75px
- postcss-px-to-viewport (https://www.npmjs.com/package/postcss-px-to-viewport)

3.实现底部路由

3.1 参考

3.1.1 介绍

- 这一章我们开始配置路由,我们的应用在尾部有三个页签,分别对应首页,购物车和个人中心三个页面

- 在本章节我们实践以下内容

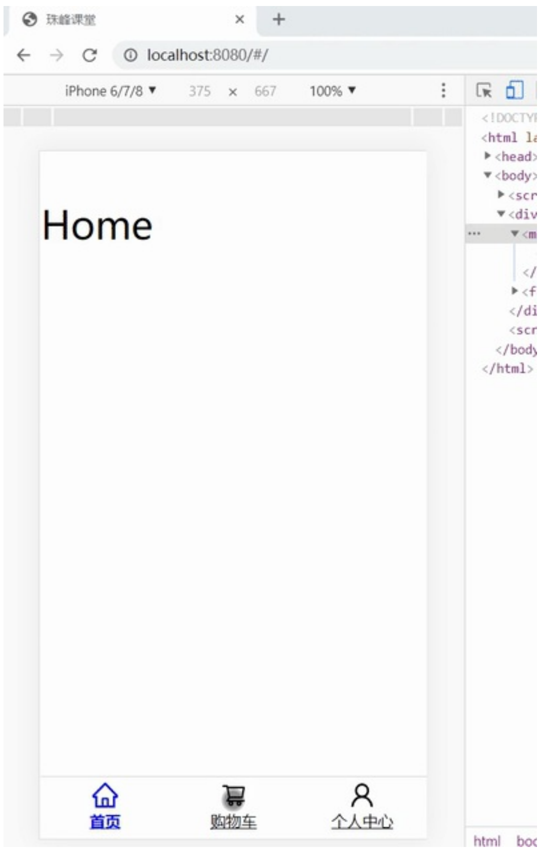
  1. 如何使用 react 全家桶配置路由

  - 1. 如何按需加载 antd并使用图标组件
  - 1. 如何在 react 样式中使用 less编写样式
  - 1. 如何在移动端中使用 rem 实现布局以及如何使用 flex布局
  - 1. 如何使用 typescript 编写 react 代码

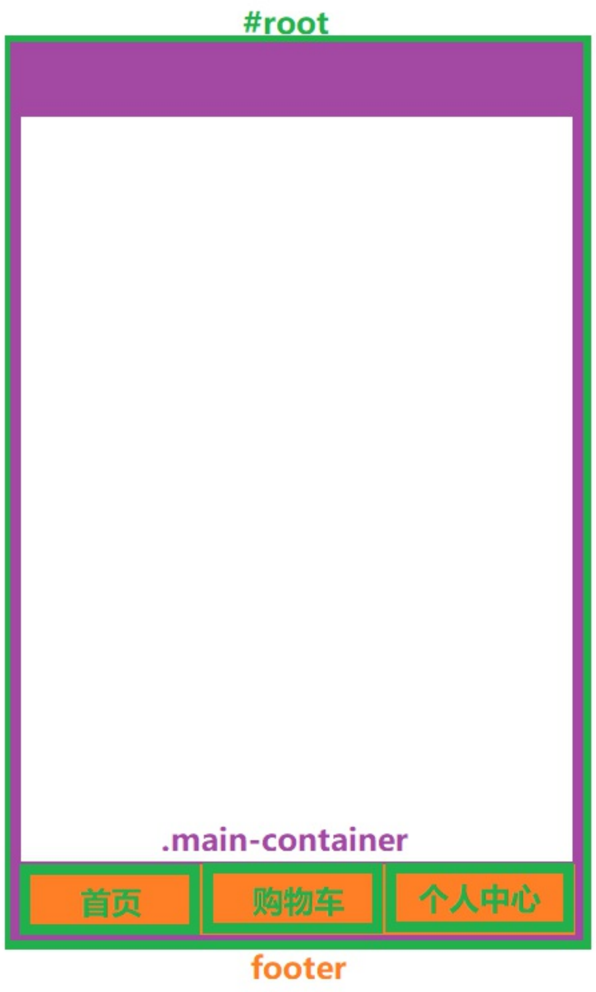
3.1.2 目录

```
├── package.json
├── public
│   └── index.html
├── src
│   ├── style
│   │   └── common.less
│   ├── components
│   │   └── Tabs
│   │       ├── index.less
│   │       └── index.tsx
│   ├── index.tsx
│   ├── routes
│   │   ├── Home
│   │   │   └── index.tsx
│   │   ├── Cart
│   │   │   └── index.tsx
│   │   └── Profile
│   │       └── index.tsx
│   └── store
│       ├── action-types.tsx
│       ├── history.tsx
│       ├── index.tsx
│       └── reducers
│           ├── home.tsx
│           ├── index.tsx
│           ├── cart.tsx
│           └── profile.tsx
├── tsconfig.json
├── webpack.config.js
```

3.1.3 效果预览



3.1.4 页面布局



3.2 src\index.tsx

src\index.tsx

```
import React from "react";
import ReactDOM from "react-dom";
import { Routes, Route } from "react-router-dom";
import { Provider } from "react-redux";
import { store, history } from "./store";
import "./style/common.less";
import Tabs from "./components/Tabs";
import Home from "./routes/Home";
import Cart from "./routes/Cart";
import Profile from "./routes/Profile";
import { HistoryRouter } from "redux-first-history/rr6";
ReactDOM.render(
    <Provider store={store}>
        <HistoryRouter history={history}>
            <main className="main-container">
                <Routes>
                    <Route path="/" element={<Home />} />
                    <Route path="/Cart" element={<Cart />} />
                    <Route path="/profile" element={<Profile />} />
                Routes>
            main>
            <Tabs />
        HistoryRouter>
    Provider>,
    document.getElementById("root")
);
```

3.3 common.less

src\style\common.less

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

3.4 Tabs\index.tsx

src\components\Tabs\index.tsx

```
import React from "react";
import { NavLink } from "react-router-dom";
import { HomeOutlined, ShoppingCartOutlined, UserOutlined } from "@ant-design/icons";
import "./index.less";
function Tabs() {
    return (
        <footer>
            <NavLink to="/" >
                <HomeOutlined />
                <span>首页span>
            NavLink>
            <NavLink to="/cart">
                <ShoppingCartOutlined />
                <span>购物车span>
            NavLink>
            <NavLink to="/profile">
                <UserOutlined />
                <span>个人中心span>
            NavLink>
        footer>
    );
}
export default Tabs;
```

3.5 Tabs\index.less

src\components\Tabs\index.less

```
footer {
  position: fixed;
  left: 0;
  bottom: 0;
  width: 100%;
  height: 120px;
  z-index: 1000;
  background-color: #fff;
  border-top: 1px 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;
    span {
      font-size: 30px;
      line-height: 50px;
      &.anticon {
        font-size: 50px;
      }
    }
    &.active {
      color: orangered;
      font-weight: bold;
    }
  }
}
```

3.6 history.tsx

src\store\history.tsx

```
import { createBrowserHistory } from 'history';
import { createReduxHistoryContext } from "redux-first-history";
const history = createBrowserHistory();
const { routerReducer, routerMiddleware, createReduxHistory } = createReduxHistoryContext({ history });
export {
    routerReducer,
    routerMiddleware,
    createReduxHistory
}
```

3.7 action-types.tsx

src\store\action-types.tsx

3.8 reducers\home.tsx

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;
    }
}
```

3.9 reducers\cart.tsx

src\store\reducers\cart.tsx

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

3.10 reducers\profile.tsx

src\store\reducers\profile.tsx

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

3.11 reducers\index.tsx

src\store\reducers\index.tsx

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

export { CombinedState }
export default reducer;
```

3.12 store\index.tsx

src\store\index.tsx

```
import { createStore, applyMiddleware } from 'redux';
import reducers from './reducers';
import logger from 'redux-logger';
import thunk from 'redux-thunk';
import promise from 'redux-promise';
import { routerMiddleware, createReduxHistory } from '../history';
export const store = applyMiddleware(thunk, routerMiddleware, promise, logger)(createStore)(reducers);
export const history = createReduxHistory(store);
```

3.13 Home\index.tsx

src\routes\Home\index.tsx

```
import React from "react";
interface Props { }
function Home(props: Props) {
    return <div>Homediv>;
}
export default Home;
```

3.14 Cart\index.tsx

src\routes\Cart\index.tsx

```
import React from "react";
interface Props { }
function Cart(props: Props) {
    return <div>Cartdiv>;
}
export default Cart;
```

3.15 Profile\index.tsx

src\routes\Profile\index.tsx

```
import React from "react";
import { connect } from "react-redux";
interface Props { }
function Profile(props: Props) {
    return <div>Profilediv>;
}
export default Profile;
```

4. 实现首页头部导航

4.1 参考

4.1.1 文档

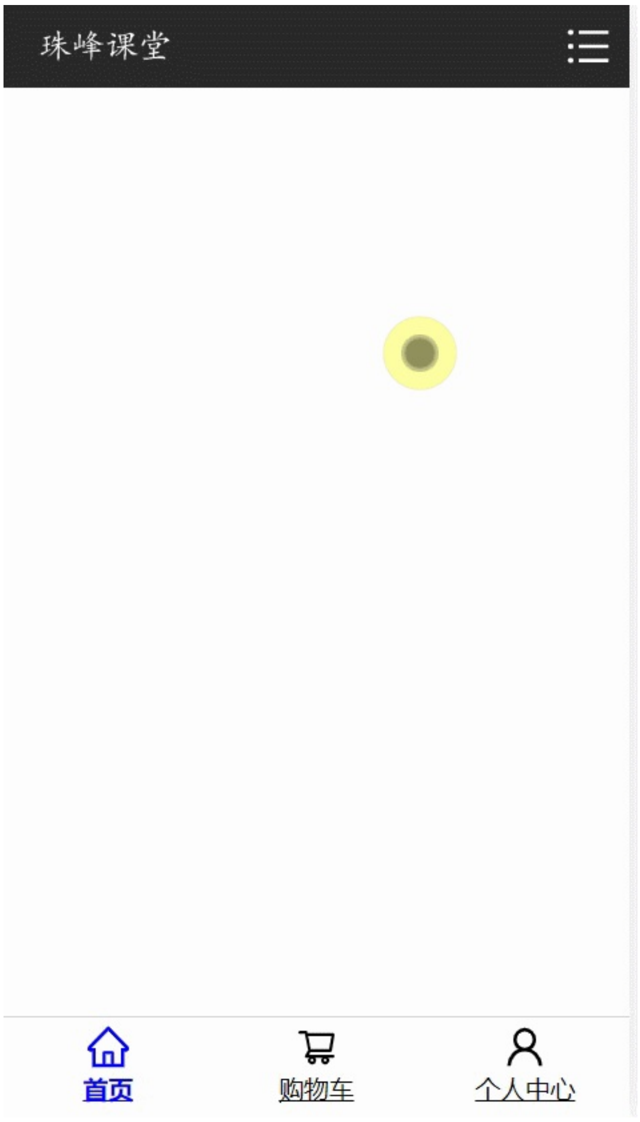
- 本章我们将要实现首页的头部导航
- 本章我们要掌握的知识点
    - React 动画库的使用
    - 如何创建 redux 仓库以及如何关联组件
- react类型声明 (https://github.com/DefinitelyTyped/DefinitelyTyped/blob/master/types/react/index.d.ts)

4.1.2 logo图

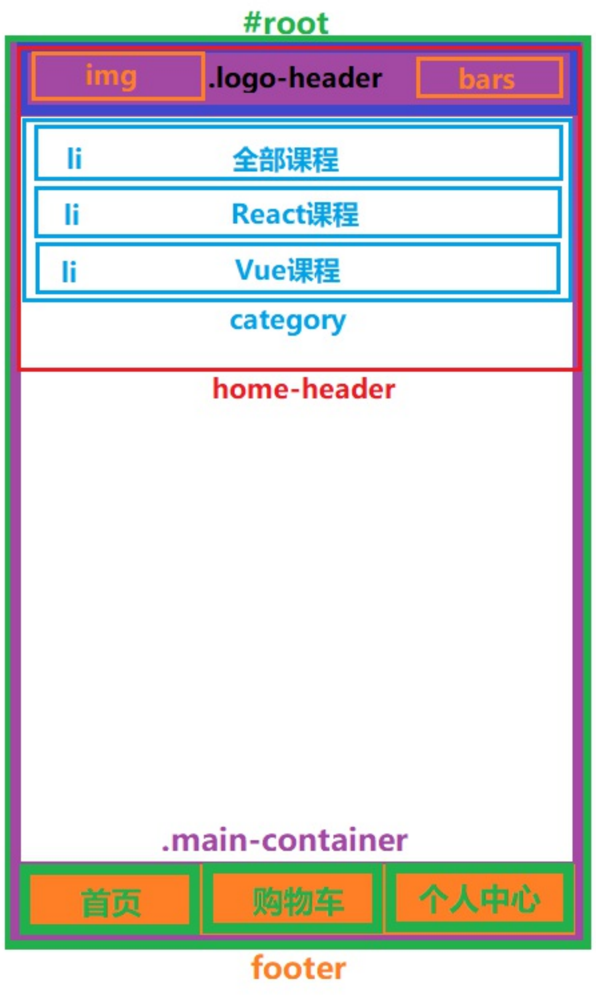
4.1.3 本章代码

```
├── package.json
├── public
│   └── index.html
├── src
│   ├── style
│   │   │   └── common.less
│   ├── assets
│   │   └── images
│   │       └── logo.png
│   ├── components
│   │   └── Tabs
│   │       ├── index.less
│   │       └── index.tsx
│   ├── index.tsx
│   ├── routes
│   │   ├── Home
│   │   │   ├── components
│   │   │   │   └── HomeHeader
│   │   │   │       ├── index.less
│   │   │   │       └── index.tsx
│   │   │   ├── index.less
│   │   │   └── index.tsx
│   │   ├── Cart
│   │   │   └── index.tsx
│   │   └── Profile
│   │       └── index.tsx
│   ├── store
│   │   ├── actionCreators
│   │   │   └── home.tsx
│   │   ├── action-types.tsx
│   │   ├── history.tsx
│   │   ├── index.tsx
│   │   └── reducers
│   │       ├── home.tsx
│   │       ├── index.tsx
│   │       ├── cart.tsx
│   │       └── profile.tsx
│   └── typings
│       └── images.d.ts
├── tsconfig.json
├── webpack.config.js
```

4.1.4 效果预览



4.1.5 本章布局



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

```
import React, { useState, CSSProperties } from 'react';
import { BarsOutlined } from '@ant-design/icons';
import classnames from 'classnames';
import { Transition } from 'react-transition-group';

import logo from '@/assets/images/logo.png';
import './index.less';
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;
}
function HomeHeader(props: Props) {
    let [isMenuVisible, setIsMenuVisible] = useState(false);

    const setCurrentCategory = (event: React.MouseEvent) => {
        let target: HTMLLIElement = event.target as HTMLLIElement;
        let category = target.dataset.category;
        props.setCurrentCategory(category);
        setIsMenuVisible(false);
    }
    return (
        <header className="home-header">
            <div className="logo-header">
                <img src={logo} />
                <BarsOutlined 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: 100px;
    background: @BG;
    color: #fff;
    display: flex;
    justify-content: space-between;
    align-items: center;
    img {
      width: 200px;
      margin-left: 20px;
    }
    span.anticon.anticon-bars {
      font-size: 60px;
      margin-right: 20px;
    }
  }
  .category {
    position: absolute;
    width: 100%;
    top: 100px;
    left: 0;
    background: @BG;
    li {
      line-height: 60px;
      text-align: center;
      color: #fff;
      font-size: 30px;
      border-top: 1px solid lighten(@BG, 20%);
      &.active {
        color: red;
      }
    }
  }
}
```

4.4 action-types.tsx

src\store\action-types.tsx

```
//设置当前分类的名称
+export const SET_CURRENT_CATEGORY = 'SET_CURRENT_CATEGORY';
```

4.5 reducers\home.tsx

src\store\reducers\home.tsx

```
import { AnyAction } from 'redux';
+import * as actionTypes from "../action-types";
export interface HomeState {
+    currentCategory: string;
}
let initialState: HomeState = {
+    currentCategory: 'all'//默认当前的分类是显示全部类型的课程
};
export default function (state: HomeState = initialState, action: AnyAction): HomeState {
    switch (action.type) {
+        case actionTypes.SET_CURRENT_CATEGORY://修改当前分类
+            return { ...state, currentCategory: action.payload };
        default:
            return state;
    }
}
```

4.6 actionCreators\home.tsx

src\store\actionCreators\home.tsx

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

4.7 Home\index.tsx

src\routes\Home\index.tsx

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

4.8 Home\index.less

src\routes\Home\index.less

5. 个人中心

- 本章主要编写 `Profile`组件,就是切换到个人中心页的时候,先发起一个 ajax 请求判断此用户是否登录,如果已经登录的话显示用户信息,如果未登录的请提示跳转到登录和注册页
- 本章实践的内容
    - 路由的切换
    - 如何在 hooks 中发起 ajax 请求
    - 如何保存及发送时携带 jwt 和 token
    - 如何根据环境不同加载不同的接口地址
    - 如何编写 `axios`拦截器

5.1 参考

5.1.1 本章目录

```
.
├── package.json
├── public
│   └── index.html
├── src
│   ├── api
│   │   ├── index.tsx
│   │   └── profile.tsx
│   ├── assets
│   │   ├── css
│   │   │   └── common.less
│   │   └── images
│   │       └── logo.png
│   ├── components
│   │   ├── NavHeader
│   │   │   ├── index.less
│   │   │   └── index.tsx
│   │   └── Tabs
│   │       ├── index.less
│   │       └── index.tsx
│   ├── index.tsx
│   ├── routes
│   │   ├── Home
│   │   │   ├── components
│   │   │   │   └── HomeHeader
│   │   │   │       ├── index.less
│   │   │   │       └── index.tsx
│   │   │   ├── index.less
│   │   │   └── index.tsx
│   │   ├── Cart
│   │   │   └── index.tsx
│   │   └── Profile
│   │       ├── index.less
│   │       └── index.tsx
│   ├── store
│   │   ├── actionCreators
│   │   │   ├── home.tsx
│   │   │   └── profile.tsx
│   │   ├── action-types.tsx
│   │   ├── history.tsx
│   │   ├── index.tsx
│   │   └── reducers
│   │       ├── home.tsx
│   │       ├── index.tsx
│   │       ├── cart.tsx
│   │       └── profile.tsx
│   └── typings
│       ├── images.d.ts
│       └── login-types.tsx
├── tsconfig.json
├── webpack.config.js
```

5.1.2 本章效果

5.2 Profile\index.tsx

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 actionCreators from "../../store/actionCreators/profile";
import LOGIN_TYPES from "../../typings/login-types";
import { Descriptions, Button, Alert, message } from "antd";
import NavHeader from "../../components/NavHeader";
import { AxiosError } from "axios";
import { useNavigate } from 'react-router-dom';
import "./index.less";

type StateProps = ReturnType<typeof mapStateToProps>;
type DispatchProps = typeof actionCreators;
interface Params { }
type Props = PropsWithChildren;
function Profile(props: Props) {
    const navigate = useNavigate();

    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="primary">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: "50px" }}>
                    <Button type="dashed" onClick={() => navigate("/login")}>
                        登录
                    Button>
                    <Button
                        type="dashed"
                        style={{ marginLeft: "50px" }}
                        onClick={() => navigate("/register")}
                    >
                        注册
                    Button>
                div>
            </>
        );
    }
    return (
        <section>
            <NavHeader >个人中心NavHeader>
            {content}
        section>
    );
}

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

5.3 Profile\index.less

src\routes\Profile\index.less

```
.user-info {
  padding: 20px;
}
```

5.4 action-types.tsx

src\store\action-types.tsx

```
export const ADD = 'ADD';
//设置当前分类的名称
export const SET_CURRENT_CATEGORY = 'SET_CURRENT_CATEGORY';
//发起验证用户是否登录的请求
+export const VALIDATE = 'VALIDATE';
```

5.5 typings\login-types.tsx

src\typings\login-types.tsx

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

5.6 reducers\profile.tsx

src\store\reducers\profile.tsx

```
import { AnyAction } from "redux";
import * as actionTypes 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 actionTypes.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,
        };
      }
    default:
      return state;
  }
}
```

5.7 actionCreators\profile.tsx

src\store\actionCreators\profile.tsx

```
import { AnyAction } from "redux";
import * as actionTypes from "../action-types";
import { validate } from "../../api/profile";
export default {

  validate(): AnyAction {

    return {
      type: actionTypes.VALIDATE,
      payload: validate(),
    };
  },
};
```

5.8 api\index.tsx

src\api\index.tsx

```
import axios from "axios";
axios.defaults.baseURL = "http://ketang.zhufengpeixun.com";

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;
```

5.10 api\profile.tsx

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 { LeftOutlined } from "@ant-design/icons";
import { UNSAFE_NavigationContext as NavigationContext } from 'react-router-dom';
interface Props {
    children: any;
}
export default function NavHeader(props: Props) {
    const { navigator } = React.useContext(NavigationContext);
    return (
        <div className="nav-header">
            <LeftOutlined onClick={() => (navigator as unknown as History).back()} />
            {props.children}
        div>
    );
}
```

src\components\NavHeader\index.less

```
.nav-header {
  position: fixed;
  left: 0;
  top: 0;
  height: 100px;
  z-index: 1000;
  width: 100%;
  box-sizing: border-box;
  text-align: center;
  line-height: 100px;
  background-color: #2a2a2a;
  color: #fff;
  font-size:35px;
  span {
    position: absolute;
    left: 20px;
    line-height: 100px;
  }
}
```

6. 注册登陆

- 本章我们主要是实现注册登录的功能
- 本章主要要实践的内容
    - 如何使用 `antd4`中的表单功能

6.1 参考

6.1.1 目录结构

```
.
├── package.json
├── public
│   └── index.html
├── src
│   ├── api
│   │   ├── index.tsx
│   │   └── profile.tsx
│   ├── assets
│   │   ├── css
│   │   │   └── common.less
│   │   └── images
│   │       └── logo.png
│   ├── components
│   │   ├── NavHeader
│   │   │   ├── index.less
│   │   │   └── index.tsx
│   │   └── Tabs
│   │       ├── index.less
│   │       └── index.tsx
│   ├── index.tsx
│   ├── routes
│   │   ├── Home
│   │   │   ├── components
│   │   │   │   └── HomeHeader
│   │   │   │       ├── index.less
│   │   │   │       └── index.tsx
│   │   │   ├── index.less
│   │   │   └── index.tsx
│   │   ├── Login
│   │   │   ├── index.less
│   │   │   └── index.tsx
│   │   ├── Cart
│   │   │   └── index.tsx
│   │   ├── Profile
│   │   │   ├── index.less
│   │   │   └── index.tsx
│   │   └── Register
│   │       ├── index.less
│   │       └── index.tsx
│   ├── store
│   │   ├── actionCreators
│   │   │   ├── home.tsx
│   │   │   └── profile.tsx
│   │   ├── action-types.tsx
│   │   ├── history.tsx
│   │   ├── index.tsx
│   │   └── reducers
│   │       ├── home.tsx
│   │       ├── index.tsx
│   │       ├── cart.tsx
│   │       └── profile.tsx
│   └── typings
│       ├── images.d.ts
│       ├── login-types.tsx
│       └── user.tsx
├── tsconfig.json
├── webpack.config.js
```

6.1.2 本章效果

6.2 src\index.tsx

src\index.tsx

```
import React from "react";
import ReactDOM from "react-dom";
import { Switch, Route, Redirect } from "react-router-dom";//三个路由组件
import { Provider } from "react-redux";//负责把属性中的store传递给子组件
import store from "./store";//引入仓库
import { ConfigProvider } from "antd";//配置
import zh_CN from "antd/lib/locale-provider/zh_CN";//国际化中文
import "./assets/css/common.less";//通用的样式
import Tabs from "./components/Tabs";//引入底部的页签导航
import Home from "./routes/Home";//首页
import Cart from "./routes/Cart";//我的课程
import Profile from "./routes/Profile";//个人中心
+import Register from "./routes/Register";
+import Login from "./routes/Login";
import { ConnectedRouter } from 'redux-first-history';//redux绑定路由
import history from './store/history';
ReactDOM.render(

+
+


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

6.3 api\profile.tsx

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);
+}
```

6.4 Profile\index.tsx

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 actionCreators from '../../store/actionCreators/profile';
import LOGIN_TYPES from '../../typings/login-types';
import { Descriptions, Button, Alert, message } from 'antd';
import NavHeader from '../../components/NavHeader';
import { useNavigate } from 'react-router-dom';
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 actionCreators;
type Props = PropsWithChildren
function Profile(props: Props) {
   const navigate = useNavigate();
    //组件加载后直接 发起验证请求,查看此用户是否已经登录过了,如果没有登录则提示错误
    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) {//如果未验证则内容为null
        content = null;
    } else if (props.loginState == LOGIN_TYPES.LOGINED) {//如果已经登录显示用户信息
        content = (

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

+                 props.logout() }>退出登录

        )
    } else {//如果没有登录,则显示注册和登录按钮
        content = (
            <>

                    navigate('/login')}>登录
                    navigate('/register')}>注册

            </>
        )
    }
    return (
        (

                个人中心
                {content}

        )
    )
}

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

6.5 action-types.tsx

src\store\action-types.tsx

```
export const ADD = 'ADD';
//设置当前分类的名称
export const SET_CURRENT_CATEGORY = 'SET_CURRENT_CATEGORY';
//发起验证用户是否登录的请求
export const VALIDATE = 'VALIDATE';

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

6.6 actionCreators\profile.tsx

src\store\actionCreators\profile.tsx

```
import { AnyAction } from 'redux';
import * as actionTypes from '../action-types';
+import { validate, register, login } from '@/api/profile';
+import { push } from 'redux-first-history';
+import { RegisterPayload, LoginPayload, RegisterResult, LoginResult } from '@/typings/user';
+import { message } from "antd";
export default {
    //https://github.com/redux-utilities/redux-promise/blob/master/src/index.js
    validate(): AnyAction {//发起判断当前用户是否登录的请求
        return {
            type: actionTypes.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: actionTypes.LOGOUT });
+                dispatch(push('/login'));
+            }
+    }
+}
```

6.7 src\typings\user.tsx

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
}
```

6.8 Register\index.tsx

src\routes\Register\index.tsx

```
import React from "react";
import { connect } from "react-redux";
import actionCreators from "../../store/actionCreators/profile";
import { Link } from "react-router-dom";
import NavHeader from "../../components/NavHeader";
import { Form, Input, Button, message } from "antd";
import { CombinedState } from "../../store/reducers";
import { ProfileState } from "../../store/reducers/profile";
import { UserAddOutlined, LockOutlined, MailOutlined } from "@ant-design/icons";
import "./index.less";
type StateProps = ReturnType<typeof mapStateToProps>;
type DispatchProps = typeof actionCreators;
type Props = StateProps & DispatchProps;

function Register(props: Props) {
    const onFinish = (values: any) => {
        props.register(values);
    };
    const onFinishFailed = (errorInfo: any) => {
        message.error("表单验证失败! " + errorInfo);
    };
    return (
        <>
            <NavHeader>用户注册NavHeader>
            <Form
                onFinish={onFinish}
                onFinishFailed={onFinishFailed}
                className="login-form"
            >
                <Form.Item
                    label="用户名"
                    name="username"
                    rules={[{ required: true, message: "请输入你的用户名!" }]}
                >
                    <Input prefix={<UserAddOutlined />} placeholder="用户名" />
                Form.Item>
                <Form.Item
                    label="密码"
                    name="password"
                    rules={[{ required: true, message: "请输入你的密码!" }]}
                >
                    <Input prefix={<LockOutlined />} type="password" placeholder="密码" />
                Form.Item>
                <Form.Item
                    label="确认密码"
                    name="confirmPassword"
                    rules={[{ required: true, message: "请输入你的确认密码!" }]}
                >
                    <Input
                        prefix={<LockOutlined />}
                        type="password"
                        placeholder="确认密码"
                    />
                Form.Item>
                <Form.Item
                    label="邮箱"
                    name="email"
                    rules={[{ required: true, message: "请输入你的邮箱!" }]}
                >
                    <Input prefix={<MailOutlined />} 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>
        </>
    );
}

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

routes\Register\index.less

```
.login-form {
  padding: 20px;
}
```

6.9 Login\index.tsx

src\routes\Login\index.tsx

```
import React from "react";
import { connect } from "react-redux";
import actionCreators from "@/store/actionCreators/profile";
import { Link } from "react-router-dom";
import NavHeader from "@/components/NavHeader";
import { Form, Input, Button, message } from "antd";
import "./index.less";
import { CombinedState } from "@/store/reducers";
import { ProfileState } from "@/store/reducers/profile";
import { UserAddOutlined, LockOutlined, MailOutlined } from "@ant-design/icons";
type StateProps = ReturnType<typeof mapStateToProps>;
type DispatchProps = typeof actionCreators;
type Props = StateProps & DispatchProps;

function Register(props: Props) {
    const onFinish = (values: any) => {
        props.login(values);
    };
    const onFinishFailed = (errorInfo: any) => {
        message.error("表单验证失败! " + errorInfo);
    };
    return (
        <>
            <NavHeader>用户登录NavHeader>
            <Form
                onFinish={onFinish}
                onFinishFailed={onFinishFailed}
                className="login-form"
            >
                <Form.Item
                    label="用户名"
                    name="username"
                    rules={[{ required: true, message: "请输入你的用户名!" }]}
                >
                    <Input prefix={<UserAddOutlined />} placeholder="用户名" />
                Form.Item>
                <Form.Item
                    label="密码"
                    name="password"
                    rules={[{ required: true, message: "请输入你的密码!" }]}
                >
                    <Input prefix={<LockOutlined />} 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 mapStateToProps = (state: CombinedState): ProfileState => state.profile;
export default connect(mapStateToProps, actionCreators)(Register);
```

6.10 Login\index.less

src\routes\Login\index.less

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

7.上传头像

- 本章节我们学习如何向服务器端上传头像
- 本章学习如下内容
    - 如何使用 antdesign 的图片上传组件

7.1 参考

7.1.1 本章目录

```
.
├── package.json
├── public
│   └── index.html
├── src
│   ├── api
│   │   ├── index.tsx
│   │   └── profile.tsx
│   ├── assets
│   │   ├── css
│   │   │   └── common.less
│   │   └── images
│   │       └── logo.png
│   ├── components
│   │   ├── NavHeader
│   │   │   ├── index.less
│   │   │   └── index.tsx
│   │   └── Tabs
│   │       ├── index.less
│   │       └── index.tsx
│   ├── index.tsx
│   ├── routes
│   │   ├── Home
│   │   │   ├── components
│   │   │   │   └── HomeHeader
│   │   │   │       ├── index.less
│   │   │   │       └── index.tsx
│   │   │   ├── index.less
│   │   │   └── index.tsx
│   │   ├── Login
│   │   │   ├── index.less
│   │   │   └── index.tsx
│   │   ├── Cart
│   │   │   └── index.tsx
│   │   ├── Profile
│   │   │   ├── index.less
│   │   │   └── index.tsx
│   │   └── Register
│   │       ├── index.less
│   │       └── index.tsx
│   ├── store
│   │   ├── actionCreators
│   │   │   ├── home.tsx
│   │   │   └── profile.tsx
│   │   ├── action-types.tsx
│   │   ├── history.tsx
│   │   ├── index.tsx
│   │   └── reducers
│   │       ├── home.tsx
│   │       ├── index.tsx
│   │       ├── cart.tsx
│   │       └── profile.tsx
│   └── typings
│       ├── images.d.ts
│       ├── login-types.tsx
│       └── user.tsx
├── tsconfig.json
├── webpack.config.js
```

7.1.2 本章效果

7.2 Profile\index.tsx

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 actionCreators from "../../store/actionCreators/profile";
import LOGIN_TYPES from "../../typings/login-types";
+import { Descriptions, Button, Alert, message, Upload } from "antd";
import NavHeader from "../../components/NavHeader";
import { AxiosError } from "axios";
import "./index.less";
+import { LoadingOutlined, UploadOutlined } from "@ant-design/icons";
//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 actionCreators;
type Props = StateProps & DispatchProps
+const mapStateToProps = (initialState: CombinedState): ProfileState =>initialState.profile;
function Profile(props: Props) {
+  let [loading, setLoading] = useState(false);
  //组件加载后直接 发起验证请求,查看此用户是否已经登录过了,如果没有登录则提示错误
  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) {
    //如果未验证则内容为null
    content = null;
  } else if (props.loginState == LOGIN_TYPES.LOGINED) {
+    const uploadButton = (
+
+        {loading ?  : }
+        上传
+
+    );
    //如果已经登录显示用户信息
    content = (

            {props.user.username}

          {props.user.email}
+
+
+              name="avatar"
+              listType="picture-card"
+              className="avatar-uploader"
+              showUploadList={false}
+              action="http://ketang.zhufengpeixun.com/user/uploadAvatar"
+              beforeUpload={beforeUpload}
+              data={{ userId: props.user.id }}
+              onChange={handleChange}
+            >
+              {props.user.avatar ? (
+
+                  src={props.user.avatar}
+                  alt="avatar"
+                  style={{ width: "100%" }}
+                />
+              ) : (
+                uploadButton
+              )}
+
+


         {
            await props.logout();
            navigate("/login");
          }}
        >
          退出登录

    );
  } else {
    //如果没有登录,则显示注册和登录按钮
    content = (
      <>

           navigate("/login")}>
            登录

           navigate("/register")}
          >
            注册

      </>
    );
  }
  return (

      个人中心
      {content}

  );
}

+export default connect(mapStateToProps, actionCreators)(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;
+}
```

7.3 action-types.tsx

src\store\action-types.tsx

```
export const ADD = "ADD";
//设置当前分类的名称
export const SET_CURRENT_CATEGORY = "SET_CURRENT_CATEGORY";
//发起验证用户是否登录的请求
export const VALIDATE = "VALIDATE";

export const LOGOUT = "LOGOUT";
+//上传头像
+export const CHANGE_AVATAR = "CHANGE_AVATAR";
```

7.4 reducers\profile.tsx

src\store\reducers\profile.tsx

```
import { AnyAction } from "redux";
import * as actionTypes from "../action-types";
import LOGIN_TYPES from "../../typings/login-types";
export interface ProfileState {
  loginState: LOGIN_TYPES; //当前用户的登录状态
  user: any; //当前已经登录的用户信息
  error: string | null; //错误信息
}
let initialState: ProfileState = {
  //初始状态
  loginState: LOGIN_TYPES.UN_VALIDATE, //当前用户的登录状态
  user: null, //当前已经登录的用户信息
  error: null, //错误信息
};
export default function (
  state: ProfileState = initialState,
  action: AnyAction
): ProfileState {
  switch (action.type) {
    case actionTypes.VALIDATE:
      if (action.payload.success) {
        //如果此用户已经登录了
        return {
          ...state,
          loginState: LOGIN_TYPES.LOGINED,
          user: action.payload.data, //设置用户名
          error: null, //没有错误
        };
      } else {
        return {
          ...state,
          loginState: LOGIN_TYPES.UNLOGIN,
          user: null, //用户名为空
          error: action.payload, //错误对象赋值
        };
      }
+    case actionTypes.LOGOUT:
+      return {
+        ...state,
+        loginState: LOGIN_TYPES.UN_VALIDATE,
+        user: null,
+        error: null,
+      };
+    case actionTypes.CHANGE_AVATAR:
+      return { ...state, user: { ...state.user, avatar: action.payload } };
    default:
      return state;
  }
}
```

7.5 actionCreators\profile.tsx

src\store\actionCreators\profile.tsx

```
import { AnyAction } from "redux";
import * as actionTypes from "../action-types";
import { validate, register, login } from "@/api/profile";
import { push } from "redux-first-history";
import {
  RegisterPayload,
  LoginPayload,
  RegisterResult,
  LoginResult,
} from "@/typings/user";
import { message } from "antd";
export default {
  //https://github.com/redux-utilities/redux-promise/blob/master/src/index.js
  validate(): AnyAction {
    //发起判断当前用户是否登录的请求
    return {
      type: actionTypes.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: actionTypes.LOGOUT });
      dispatch(push("/login"));
    };
  },
+  changeAvatar(avatar: string) {
+    return {
+      type: actionTypes.CHANGE_AVATAR,
+      payload: avatar,
+    };
+  },
};
```

8. 前台轮播图

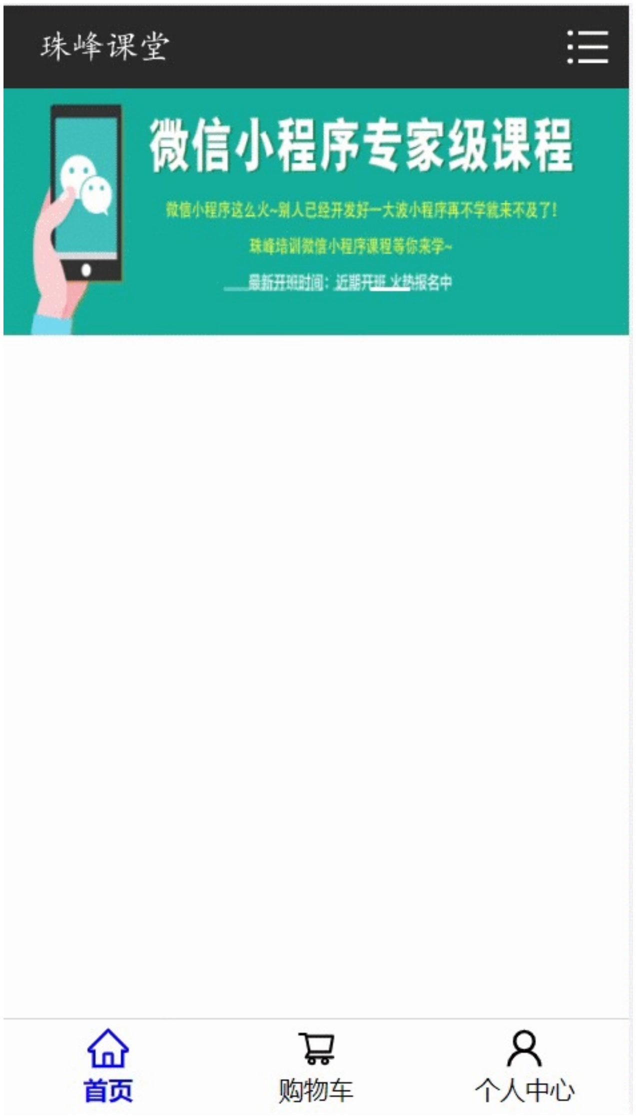
- 本章实践前台轮播图
- 本章需要实践的内容
    - 使用 `useRef`取得 DOM 元素
    - 使用 `antdesign` 的轮播图组件

8.1 参考

8.1.1 本章目录

```
.
├── package.json
├── public
│   └── index.html
├── README.md
├── src
│   ├── api
│   │   ├── home.tsx
│   │   ├── index.tsx
│   │   └── profile.tsx
│   ├── assets
│   │   ├── css
│   │   │   └── common.less
│   │   └── images
│   │       └── logo.png
│   ├── components
│   │   ├── NavHeader
│   │   │   ├── index.less
│   │   │   └── index.tsx
│   │   └── Tabs
│   │       ├── index.less
│   │       └── index.tsx
│   ├── index.tsx
│   ├── routes
│   │   ├── Home
│   │   │   ├── components
│   │   │   │   ├── HomeHeader
│   │   │   │   │   ├── index.less
│   │   │   │   │   └── index.tsx
│   │   │   │   └── HomeSliders
│   │   │   │       ├── index.less
│   │   │   │       └── index.tsx
│   │   │   ├── index.less
│   │   │   └── index.tsx
│   │   ├── Login
│   │   │   ├── index.less
│   │   │   └── index.tsx
│   │   ├── Cart
│   │   │   └── index.tsx
│   │   ├── Profile
│   │   │   ├── index.less
│   │   │   └── index.tsx
│   │   └── Register
│   │       ├── index.less
│   │       └── index.tsx
│   ├── store
│   │   ├── actionCreators
│   │   │   ├── home.tsx
│   │   │   └── profile.tsx
│   │   ├── action-types.tsx
│   │   ├── history.tsx
│   │   ├── index.tsx
│   │   └── reducers
│   │       ├── home.tsx
│   │       ├── index.tsx
│   │       ├── cart.tsx
│   │       └── profile.tsx
│   └── typings
│       ├── images.d.ts
│       ├── login-types.tsx
│       ├── slider.tsx
│       └── user.tsx
├── tsconfig.json
├── webpack.config.js
```

8.1.2 本章效果


珠峰课堂
微信小程序专家级课程
微信小程序这么火~别人已经开发好一大波小程序再不学就来不及了!
珠峰培训微信小程序课程等你来学~
最新开班时间：近期开班 火热报名中
首页
购物车
个人中心

8.2 action-types.tsx

src\store\action-types.tsx

```
export const ADD = "ADD";
//设置当前分类的名称
export const SET_CURRENT_CATEGORY = "SET_CURRENT_CATEGORY";
//发起验证用户是否登录的请求
export const VALIDATE = "VALIDATE";

export const LOGOUT = "LOGOUT";
//上传头像
export const CHANGE_AVATAR = "CHANGE_AVATAR";
+export const GET_SLIDERS = "GET_SLIDERS";
```

8.3 actionCreators\home.tsx

src\store\actionCreators\home.tsx

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

8.4 typings\slider.tsx

src\typings\slider.tsx

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

8.5 reducers\home.tsx

src\store\reducers\home.tsx

```
import { AnyAction } from "redux";
import * as actionTypes from "../action-types";
+import Slider from "@/typings/slider";
export interface HomeState {
  currentCategory: string;
+  sliders: Slider[];
}
let initialState: HomeState = {
  currentCategory: "all", //默认当前的分类是显示全部类型的课程
+ sliders: [],
};
export default function (
  state: HomeState = initialState,
  action: AnyAction
): HomeState {
  switch (action.type) {
    case actionTypes.SET_CURRENT_CATEGORY: //修改当前分类
      return { ...state, currentCategory: action.payload };
+    case actionTypes.GET_SLIDERS:
+      return { ...state, sliders: action.payload.data };
    default:
      return state;
  }
}
```

8.6 api\home.tsx

src\api\home.tsx

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

8.7 HomeSliders\index.tsx

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/slider";
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;
```

8.8 HomeSliders\index.less

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

```
.ant-carousel .slick-slide {
  text-align: center;
  height: 320px;
  line-height: 320px;
  background: #364d79;
  overflow: hidden;
  color: #fff;
  img {
    width: 100%;
    height: 320px;
  }
}
```

8.9 Home\index.tsx

src\routes\Home\index.tsx

```
import React, { PropsWithChildren, useRef } from "react";
import { connect } from "react-redux";
import actionCreators from "@/store/actionCreators/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 actionCreators;
type Props = PropsWithChildren<
StateProps & DispatchProps
>;
function Home(props: Props) {
+  const homeContainerRef = useRef(null);
  return (
    <>

+
+
+
    </>
  );
}
let mapStateToProps = (state: CombinedState): HomeState => state.home;
export default connect(mapStateToProps, actionCreators)(Home);
```

9. 课程列表

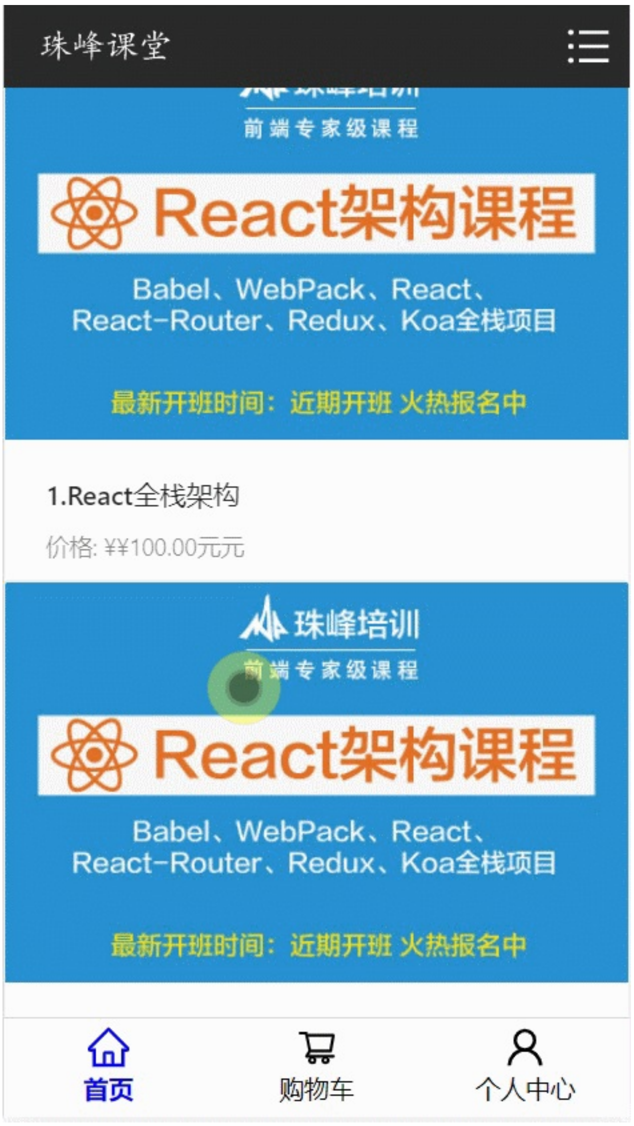
- 本章实践的内容
    - 加载课程列表
    - 上拉加载

9.1 参考

9.1.1 目录结构

```
.
├── package.json
├── public
│   └── index.html
├── src
│   ├── api
│   │   ├── home.tsx
│   │   ├── index.tsx
│   │   └── profile.tsx
│   ├── assets
│   │   ├── css
│   │   │   └── common.less
│   │   └── images
│   │       └── logo.png
│   ├── components
│   │   ├── NavHeader
│   │   │   ├── index.less
│   │   │   └── index.tsx
│   │   └── Tabs
│   │       ├── index.less
│   │       └── index.tsx
│   ├── index.tsx
│   ├── routes
│   │   ├── Home
│   │   │   ├── components
│   │   │   │   ├── HomeHeader
│   │   │   │   │   ├── index.less
│   │   │   │   │   └── index.tsx
│   │   │   │   ├── HomeSliders
│   │   │   │   │   ├── index.less
│   │   │   │   │   └── index.tsx
│   │   │   │   └── LessonList
│   │   │   │       ├── index.less
│   │   │   │       └── index.tsx
│   │   │   ├── index.less
│   │   │   └── index.tsx
│   │   ├── Login
│   │   │   ├── index.less
│   │   │   └── index.tsx
│   │   ├── Cart
│   │   │   └── index.tsx
│   │   ├── Profile
│   │   │   ├── index.less
│   │   │   └── index.tsx
│   │   └── Register
│   │       ├── index.less
│   │       └── index.tsx
│   ├── store
│   │   ├── actionCreators
│   │   │   ├── home.tsx
│   │   │   └── profile.tsx
│   │   ├── action-types.tsx
│   │   ├── history.tsx
│   │   ├── index.tsx
│   │   └── reducers
│   │       ├── home.tsx
│   │       ├── index.tsx
│   │       ├── cart.tsx
│   │       └── profile.tsx
│   ├── typings
│   │   ├── images.d.ts
│   │   ├── lesson.tsx
│   │   ├── login-types.tsx
│   │   ├── slider.tsx
│   │   └── user.tsx
│   └── utils.tsx
├── tsconfig.json
├── webpack.config.js
```

9.1.2 页面效果



9.2 api\home.tsx #

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}`
+  );
+}
```

9.3 action-types.tsx #

src\store\action-types.tsx

```
export const ADD = "ADD";
//设置当前分类的名称
export const SET_CURRENT_CATEGORY = "SET_CURRENT_CATEGORY";
//发起验证用户是否登录的请求
export const VALIDATE = "VALIDATE";

export const LOGOUT = "LOGOUT";
//上传头像
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";
```

9.4 typings\lesson.tsx #

src\typings\lesson.tsx

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

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

9.5 reducers\home.tsx

src\store\reducers\home.tsx

```
import { AnyAction } from "redux";
import * as actionTypes from "../action-types";
import Slider from "@/typings/slider";
+import Lesson from "@/typings/Lesson";

+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", //默认当前的分类是显示全部类型的课程
  sliders: [],
+  lessons: {
+    loading: false,
+    list: [],
+    hasMore: true,
+    offset: 0,
+    limit: 5,
+  },
};
export default function (
  state: HomeState = initialState,
  action: AnyAction
): HomeState {
  switch (action.type) {
    case actionTypes.SET_CURRENT_CATEGORY: //修改当前分类
      return { ...state, currentCategory: action.payload };
    case actionTypes.GET_SLIDERS:
      return { ...state, sliders: action.payload.data };
+    case actionTypes.SET_LESSONS_LOADING:
+      return {
+        ...state,
+        lessons: { ...state.lessons, loading: action.payload },
+      };
+    case actionTypes.SET_LESSONS:
+      return {
+        ...state,
+        lessons: {
+          ...state.lessons,
+          loading: false,
+          hasMore: action.payload.hasMore,
+          list: [...state.lessons.list, ...action.payload.list],
+          offset: state.lessons.offset + action.payload.list.length,
+        },
+      };
    default:
      return state;
  }
}
```

9.6 actionCreators\home.tsx

src\store\actionCreators\home.tsx

```
import * as actionTypes from "../action-types";
+import { getSliders, getLessons } from "@/api/home";
export default {
  setCurrentCategory(currentCategory: string) {
    return { type: actionTypes.SET_CURRENT_CATEGORY, payload: currentCategory };
  },
  getSliders() {
    return {
      type: actionTypes.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: actionTypes.SET_LESSONS_LOADING, payload: true });
+          let result = await getLessons(currentCategory, offset, limit);
+          dispatch({ type: actionTypes.SET_LESSONS, payload: result.data });
+        }
+      })();
+    };
+  }
};
```

9.7 src\utils.tsx

src\utils.tsx

```
export function loadMore(element: HTMLElement, callback: Function) {
    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 debounce(fn: any, wait: number) {
  var timeout: any = null;
  return function () {
    if (timeout !== null) clearTimeout(timeout);
    timeout = setTimeout(fn, wait);
  };
}
```

9.8 LessonList\index.tsx

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

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

function LessonList(props: Props) {
  useEffect(() => {
    if (props.lessons.list.length == 0) {
      props.getLessons();
    }
  }, []);
  return (
    <section className="lesson-list">
      <h2>
        <MenuOutlined /> 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) =>(
            <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>
        ))}
        {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 LessonList;
```

9.9 LessonList\index.less

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

```
.lesson-list {
  h2 {
    line-height: 100px;
    i {
      margin: 0 10px;
    }
  }
  .ant-card{
    height: 650px;
    overflow: hidden;
  }
}
```

9.10 Home\index.tsx

src\routes\Home\index.tsx

```
+import React, { PropsWithChildren, useRef, useEffect } from "react";
import { connect } from "react-redux";
import actionCreators from "@/store/actionCreators/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} from "@/utils";
type StateProps = ReturnType;
type DispatchProps = typeof actionCreators;
type Props = PropsWithChildren<
  StateProps & DispatchProps
>;
function Home(props: Props) {
  const homeContainerRef = useRef(null);
  useEffect(() => {
        loadMore(homeContainerRef.current, props.getLessons);
  }, []);
  return (
    <>


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

    </>
  );
}
let mapStateToProps = (state: CombinedState): HomeState => state.home;
export default connect(mapStateToProps, actionCreators)(Home);
```

9.11 Home\index.less

src\routes\Home\index.less

```
+.home-container {
+  position: fixed;
+  top: 100px;
+  left: 0;
+  width: 100%;
+  overflow-y: auto;
+  height: calc(100vh - 220px);
+  background-color: #FFF;
+}
```

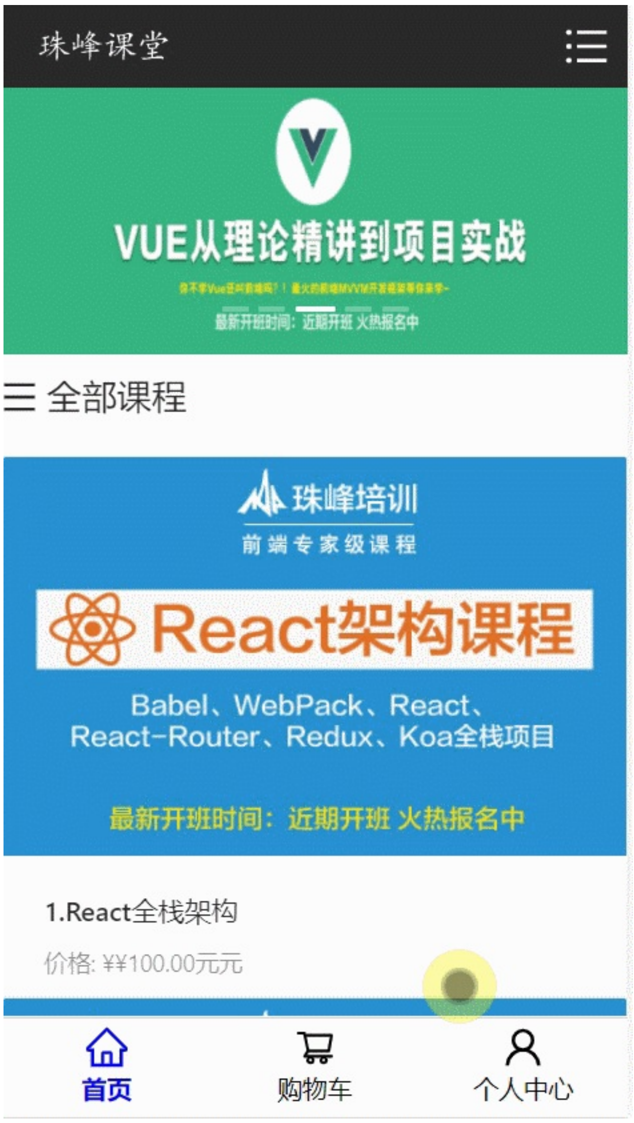
10. 课程详情

10.1 参考

10.1.1 目录结构

```
.
├── package.json
├── public
│   └── index.html
├── README.md
├── src
│   ├── api
│   │   ├── home.tsx
│   │   ├── index.tsx
│   │   └── profile.tsx
│   ├── assets
│   │   ├── css
│   │   │   └── common.less
│   │   └── images
│   │       └── logo.png
│   ├── components
│   │   ├── NavHeader
│   │   │   ├── index.less
│   │   │   └── index.tsx
│   │   └── Tabs
│   │       ├── index.less
│   │       └── index.tsx
│   ├── index.tsx
│   ├── routes
│   │   ├── Detail
│   │   │   └── index.tsx
│   │   ├── Home
│   │   │   ├── components
│   │   │   │   ├── HomeHeader
│   │   │   │   │   ├── index.less
│   │   │   │   │   └── index.tsx
│   │   │   │   ├── HomeSliders
│   │   │   │   │   ├── index.less
│   │   │   │   │   └── index.tsx
│   │   │   │   └── LessonList
│   │   │   │       ├── index.less
│   │   │   │       └── index.tsx
│   │   │   ├── index.less
│   │   │   └── index.tsx
│   │   ├── Login
│   │   │   ├── index.less
│   │   │   └── index.tsx
│   │   ├── Cart
│   │   │   └── index.tsx
│   │   ├── Profile
│   │   │   ├── index.less
│   │   │   └── index.tsx
│   │   └── Register
│   │       ├── index.less
│   │       └── index.tsx
│   ├── store
│   │   ├── actionCreators
│   │   │   ├── home.tsx
│   │   │   └── profile.tsx
│   │   ├── action-types.tsx
│   │   ├── history.tsx
│   │   ├── index.tsx
│   │   └── reducers
│   │       ├── home.tsx
│   │       ├── index.tsx
│   │       ├── cart.tsx
│   │       └── profile.tsx
│   ├── typings
│   │   ├── images.d.ts
│   │   ├── lesson.tsx
│   │   ├── login-types.tsx
│   │   ├── slider.tsx
│   │   └── user.tsx
│   └── utils.tsx
├── tsconfig.json
├── webpack.config.js
```

10.1.2 页面效果



10.2 src\index.tsx #

src\index.tsx

```
import React from "react";
import ReactDOM from "react-dom";
import { Switch, Route, Redirect } from "react-router-dom"; //三个路由组件
import { Provider } from "react-redux"; //负责把属性中的store传递给子组件
import store from "./store"; //引入仓库
import { ConfigProvider } from "antd"; //配置
import zh_CN from "antd/lib/locale-provider/zh_CN"; //国际化中文
import "./assets/css/common.less"; //通用的样式
import Tabs from "./components/Tabs"; //引入底部的页签导航
import Home from "./routes/Home"; //首页
import Cart from "./routes/Cart"; //我的课程
import Profile from "./routes/Profile"; //个人中心
import Register from "./routes/Register";
import Login from "./routes/Login";
+import Detail from "./routes/Detail";
import { ConnectedRouter } from "redux-first-history"; //redux绑定路由
import history from "./store/history";
ReactDOM.render(

+

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

10.3 api\home.tsx #

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}`);
+}
```

10.4 routes\Detail\index.tsx

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 Lesson from "@/typings/lesson";
import { StaticContext } from "react-router";
import { LessonResult } from "@/typings/lesson";
const { Meta } = Card;
interface Props {

}

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>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);
```

11. 下拉刷新

11.1 参考

11.1.1 目录结构

```
.
├── package.json
├── public
│   └── index.html
├── README.md
├── src
│   ├── api
│   │   ├── home.tsx
│   │   ├── index.tsx
│   │   └── profile.tsx
│   ├── assets
│   │   ├── css
│   │   │   └── common.less
│   │   └── images
│   │       └── logo.png
│   ├── components
│   │   ├── NavHeader
│   │   │   ├── index.less
│   │   │   └── index.tsx
│   │   └── Tabs
│   │       ├── index.less
│   │       └── index.tsx
│   ├── index.tsx
│   ├── routes
│   │   ├── Detail
│   │   │   └── index.tsx
│   │   ├── Home
│   │   │   ├── components
│   │   │   │   ├── HomeHeader
│   │   │   │   │   ├── index.less
│   │   │   │   │   └── index.tsx
│   │   │   │   ├── HomeSliders
│   │   │   │   │   ├── index.less
│   │   │   │   │   └── index.tsx
│   │   │   │   └── LessonList
│   │   │   │       ├── index.less
│   │   │   │       └── index.tsx
│   │   │   ├── index.less
│   │   │   └── index.tsx
│   │   ├── Login
│   │   │   ├── index.less
│   │   │   └── index.tsx
│   │   ├── Cart
│   │   │   └── index.tsx
│   │   ├── Profile
│   │   │   ├── index.less
│   │   │   └── index.tsx
│   │   └── Register
│   │       ├── index.less
│   │       └── index.tsx
│   ├── store
│   │   ├── actionCreators
│   │   │   ├── home.tsx
│   │   │   └── profile.tsx
│   │   ├── action-types.tsx
│   │   ├── history.tsx
│   │   ├── index.tsx
│   │   └── reducers
│   │       ├── home.tsx
│   │       ├── index.tsx
│   │       ├── cart.tsx
│   │       └── profile.tsx
│   ├── typings
│   │   ├── images.d.ts
│   │   ├── lesson.tsx
│   │   ├── login-types.tsx
│   │   ├── slider.tsx
│   │   └── user.tsx
│   └── utils.tsx
├── tsconfig.json
├── webpack.config.js
```

11.2 src\utils.tsx

src\utils.tsx

```
//element 要实现此功能DOM对象 callback加载更多的方法
export function loadMore(element: HTMLElement, callback: Function) {
    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 debounce(fn: any, wait: number) {
  var timeout: any = null;
  return function () {
    if (timeout !== null) clearTimeout(timeout);
    timeout = setTimeout(fn, wait);
  };
}

+export function downRefresh(element: HTMLDivElement, callback: Function) {
+  let startY: number; //变量，存储接下时候的纵坐标
+  let distance: number; //本次下拉的距离
+  let originalTop = element.offsetTop; //最初此元素距离顶部的距离 top=50
+  let startTop: number;
+  let $timer: any = null;
+  element.addEventListener("touchstart", function (event: TouchEvent) {
+    if ($timer) clearInterval($timer);
+    let touchMove = throttle(_touchMove, 30);
+    //只有当此元素处于原始位置才能下拉，如果处于回弹的过程则不能拉了.并且此元素向上卷去的高度==0
+    if (element.scrollTop === 0) {
+      startTop = element.offsetTop;
+      startY = event.touches[0].pageY; //记录当前点击的纵坐标
+      element.addEventListener("touchmove", touchMove);
+      element.addEventListener("touchend", touchEnd);
+    }
+
+    function _touchMove(event: TouchEvent) {
+      let pageY = event.touches[0].pageY; //拿到最新的纵坐标
+      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 = domElement.offsetTop;
+         if (currentTop - originalTop >= 1) {
+             //如果距离最原始的顶部多于1个像素，回弹一个像素
+             domElement.style.top = currentTop - 1 + 'px';
+         } else {
+             backTimer && clearInterval(backTimer)
+             domElement.style.top = originalTop + 'px';
+         }
+       }, 16);
+    }
+  });
+}
+
+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 = now;
+    }
+  };
+}
```

11.3 action-types.tsx

src\store\action-types.tsx

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

11.4 reducers\home.tsx

src\store\reducers\home.tsx

```
import { AnyAction } from "redux";
import * as actionTypes from "../action-types";
import Slider from "@/typings/slider";
import Lesson from "@/typings/Lesson";
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 actionTypes.SET_CURRENT_CATEGORY:
      return { ...state, currentCategory: action.payload };
    case actionTypes.GET_SLIDERS:
      return { ...state, sliders: action.payload.data };
    case actionTypes.SET_LESSONS_LOADING:
      return {
        ...state,
        lessons: { ...state.lessons, loading: action.payload },
      };
    case actionTypes.SET_LESSONS:
      return {
        ...state,
        lessons: {
          ...state.lessons,
          loading: false,
          hasMore: action.payload.hasMore,
          list: [...state.lessons.list, ...action.payload.list],
          offset: state.lessons.offset + action.payload.list.length,
        },
      };
+   case actionTypes.REFRESH_LESSONS:
+       return {
+         ...state,
+         lessons: {
+           ...state.lessons,
+           loading: false,
+           hasMore: action.payload.hasMore,
+           list: action.payload.list,
+           offset: action.payload.list.length,
+         },
+       };
    default:
      return state;
  }
}
```

11.5 Home\index.tsx

src\routes\Home\index.tsx

```
import React, { PropsWithChildren, useRef, useEffect } from "react";
import { connect } from 'react-redux';
import actionCreators from '@/store/actionCreators/home';
import HomeHeader from './components/HomeHeader';
import { CombinedState } from '@/store/reducers';
import { HomeState } from '@/store/reducers/home';
import './index.less';
import HomeSliders from "./components/HomeSliders";
import LessonList from "./components/LessonList";
+import { loadMore,downRefresh} from "@/utils";
+import {Spin} from 'antd';
type StateProps = ReturnType;
type DispatchProps = typeof actionCreators;
type Props = PropsWithChildren;
function Home(props: Props) {
    const homeContainerRef = useRef(null);
    useEffect(() => {
        loadMore(homeContainerRef.current, props.getLessons);
+       downRefresh(homeContainerRef.current, props.refreshLessons);
    }, []);
    return (
        <>
+

        </>
    )
}
let mapStateToProps = (state: CombinedState): HomeState => state.home;
export default connect(
    mapStateToProps,
    actionCreators
)(Home);
```

11.6 Home\index.less

src\routes\Home\index.less

```
+.ant-spin-spinning{
+  margin-top:20px;
+  margin-left:360px;
+}
.home-container {
  position: fixed;
  top: 100px;
  left: 0;
  width: 100%;
  overflow-y: auto;
  height: calc(100vh - 222px);
}
```

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

```
interface Props {
    currentCategory: string;//当前选中的分类 此数据会放在redux仓库中
    setCurrentCategory: (currentCategory: string) => any;// 改变仓库中的分类
+   refreshLessons: Function;
}
function HomeHeader(props: Props) {
    let [isMenuVisible, setIsMenuVisible] = useState(false);//设定标识位表示菜单是否显示
    //设置当前分类,把当前选中的分类传递给redux仓库
    const setCurrentCategory = (event: React.MouseEvent) => {
        let target: HTMLUListElement = event.target as HTMLUListElement;
        let category = target.dataset.category;//获取用户选择的分类名称
        props.setCurrentCategory(category);//设置分类名称
+       props.refreshLessons();
        setIsMenuVisible(false);//关闭分类选择层
    }
}
export default HomeHeader;
```

12. 虚拟列表

12.1 参考

12.1.1 目录结构

```
.
├── package.json
├── public
│   └── index.html
├── README.md
├── src
│   ├── api
│   │   ├── home.tsx
│   │   ├── index.tsx
│   │   └── profile.tsx
│   ├── assets
│   │   ├── css
│   │   │   └── common.less
│   │   └── images
│   │       └── logo.png
│   ├── components
│   │   ├── NavHeader
│   │   │   ├── index.less
│   │   │   └── index.tsx
│   │   └── Tabs
│   │       ├── index.less
│   │       └── index.tsx
│   ├── index.tsx
│   ├── routes
│   │   ├── Detail
│   │   │   └── index.tsx
│   │   ├── Home
│   │   │   ├── components
│   │   │   │   ├── HomeHeader
│   │   │   │   │   ├── index.less
│   │   │   │   │   └── index.tsx
│   │   │   │   ├── HomeSliders
│   │   │   │   │   ├── index.less
│   │   │   │   │   └── index.tsx
│   │   │   │   └── LessonList
│   │   │   │       ├── index.less
│   │   │   │       └── index.tsx
│   │   │   ├── index.less
│   │   │   └── index.tsx
│   │   ├── Login
│   │   │   ├── index.less
│   │   │   └── index.tsx
│   │   ├── Cart
│   │   │   └── index.tsx
│   │   ├── Profile
│   │   │   ├── index.less
│   │   │   └── index.tsx
│   │   └── Register
│   │       ├── index.less
│   │       └── index.tsx
│   ├── store
│   │   ├── actionCreators
│   │   │   ├── home.tsx
│   │   │   └── profile.tsx
│   │   ├── action-types.tsx
│   │   ├── history.tsx
│   │   ├── index.tsx
│   │   └── reducers
│   │       ├── home.tsx
│   │       ├── index.tsx
│   │       ├── cart.tsx
│   │       └── profile.tsx
│   ├── typings
│   │   ├── images.d.ts
│   │   ├── lesson.tsx
│   │   ├── login-types.tsx
│   │   ├── slider.tsx
│   │   └── user.tsx
│   └── utils.tsx
├── tsconfig.json
├── webpack.config.js
```

12.2 Home\index.tsx

src\routes\Home\index.tsx

```
import React, { PropsWithChildren, useRef, useEffect } from "react";
import { connect } from 'react-redux';
import actionCreators from '@/store/actionCreators/home';
import HomeHeader from './components/HomeHeader';
import { CombinedState } from '@/store/reducers';
import { HomeState } from '@/store/reducers/home';
import './index.less';
import HomeSliders from "./components/HomeSliders";
import LessonList from "./components/LessonList";
+import { loadMore,downRefresh,debounce, throttle} from "@/utils";
import {Spin} from 'antd';
type StateProps = ReturnType;
type DispatchProps = typeof actionCreators;
type Props = PropsWithChildren;
function Home(props: Props) {
    const homeContainerRef = useRef(null);
+   const lessonListRef = useRef(null);
    useEffect(() => {
        loadMore(homeContainerRef.current, props.getLessons);
        downRefresh(homeContainerRef.current, props.refreshLessons);
+       lessonListRef.current();
+       homeContainerRef.current.addEventListener("scroll", throttle(lessonListRef.current,13));
+       homeContainerRef.current.addEventListener('scroll', () => {
+           sessionStorage.setItem('scrollTop', homeContainerRef.current.scrollTop);
+       });
    }, []);
+    useEffect(() => {
+       //保持滚动条的位置
+       let scrollTop = sessionStorage.getItem('scrollTop');
+       if (scrollTop) {
+           homeContainerRef.current.scrollTop = scrollTop;
+       }
+   });
    return (
        <>

             +                  ref={lessonListRef}
+                 homeContainerRef={homeContainerRef}
             />

        </>
    )
}
let mapStateToProps = (state: CombinedState): HomeState => state.home;
export default connect(
    mapStateToProps,
    actionCreators
)(Home);
```

12.3 LessonList\index.tsx

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

```
import React, { useEffect, forwardRef, useState } from "react";
import "./index.less";
import { Card, Skeleton, Button, Alert, Menu } from "antd";
import { Link } from "react-router-dom";
import Lesson from "@/typings/lesson";
import { MenuOutlined } from "@ant-design/icons";
interface Props {
  children?: any;
  lessons?: any;
  getLessons?: any;
+ homeContainerRef: any
}
+interface VisibleLesson extends Lesson {
+  index: number
+}
function LessonList(props: Props, lessonListRef: any) {
+  const [, forceUpdate] = React.useReducer(x => x + 1, 0);
  useEffect(() => {
    if (props.lessons.list.length == 0) {
      props.getLessons();
    }
+   lessonListRef.current = forceUpdate;
  }, []);
+  const remSize: number = parseFloat(document.documentElement.style.fontSize);
+  const itemSize: number = (650 / 75) * remSize;
+  const screenHeight = window.innerHeight - (222 / 75) * remSize;
+  const homeContainer = props.homeContainerRef.current;
+  let start = 0, end = 0;
+  if (homeContainer) {
+    //卷去的高度要减去 轮播图和全部课程这个H1标签的高度
+    const scrollTop = homeContainer.scrollTop - ((320+65)/75)*remSize;;
+    start = Math.floor(scrollTop / itemSize);
+    end = start + Math.floor(screenHeight / itemSize);
+    start-=2,end+=2;
+    start = start < 0 ? 0 : start;
+    end = end > props.lessons.list.length ? props.lessons.list.length : end;
+  }
+  const visibleList: Array = props.lessons.list.map((item: Lesson, index: number) => ({ ...item, index })).slice(start, end);
+  const style: React.CSSProperties = { position: 'absolute', top: 0, left: 0, width: '100%', height: itemSize };
+  const bottomTop = (props.lessons.list.length)*itemSize;
+  return (
+

+
+         全部课程
+
+
+       {
+         visibleList.map((lesson: VisibleLesson) => (
+
+             key={lesson.id}
+             style={{ ...style, top: `${itemSize * lesson.index}px` }}
+             to={{ pathname: `/detail/${lesson.id}`, state: lesson }}
+           >
+
+               hoverable={true}
+               cover={}
+             >
+
+                 title={lesson.title}
+                 description={`价格: ¥${lesson.price}元`}
+               />
+
+
+         ))
+       }
       {props.lessons.hasMore ? (
         +          style={{ textAlign: "center" ,top: `${bottomTop}px`}}
            onClick={props.getLessons}
            loading={props.lessons.loading}
            type="primary"
            block
          >
            {props.lessons.loading ? "" : "加载更多"}

        ) : (
            +             style={{ textAlign: "center",top: `${bottomTop}px` }}
              message="到底了"
              type="warning"
            />
          )}

  );
}
+export default React.forwardRef(LessonList);
```

13. 路由懒加载

13.1 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 {Spin} from 'antd';
+const Home = React.lazy(() => import("./routes/Home"));
+const Cart = React.lazy(() => import("./routes/Cart"));
+const Profile = React.lazy(() => import("./routes/Profile"));
+const Register = React.lazy(() => import("./routes/Register"));
+const Login = React.lazy(() => import("./routes/Login"));
+const Detail = React.lazy(() => import("./routes/Detail"));
import { ConnectedRouter } from "redux-first-history";
import history from "./store/history";
ReactDOM.render(

+      }>

+


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

14. 购物车

- redux-immer (https://github.com/salvoravida/redux-immer)is used to create an equivalent function of Redux combineReducers that works with immer state.
- redux-persist (https://github.com/rt2zz/redux-persist)Persist and rehydrate a redux store.

14.1 src\index.tsx

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 "./assets/css/common.less";
import Tabs from "./components/Tabs";
import {Spin} from 'antd';
const Home = React.lazy(() => import("./routes/Home"));
const Profile = React.lazy(() => import("./routes/Profile"));
const Register = React.lazy(() => import("./routes/Register"));
const Login = React.lazy(() => import("./routes/Login"));
const Detail = React.lazy(() => import("./routes/Detail"));
+const Cart = React.lazy(() => import("./routes/Cart"));
+import { PersistGate } from 'redux-persist/integration/react'
import { ConnectedRouter } from "redux-first-history";

import history from "./store/history";
ReactDOM.render(

+    } persistor={persistor}>

       }>

+


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

14.2 Detail\index.tsx

src\routes\Detail\index.tsx

```
+import React, { useState, useEffect,PropsWithChildren } from "react";
import { connect } from "react-redux";
import { Card, Button } from "antd";
import NavHeader from "@/components/NavHeader";
import { getLesson } from "@/api/home";
import Lesson from "@/typings/lesson";
import { StaticContext } from "react-router";
import { LessonResult } from "@/typings/lesson";
+import { CombinedState } from '@/store/reducers';
+import actionCreators from '@/store/actionCreators/cart';
const { Meta } = Card;
interface Params {
  id: string;
}
+type StateProps = ReturnType;
+type DispatchProps = typeof actionCreators;
+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-shopping-cart');
+  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={}
      >
        +          <>
+           价格: {lesson.price}
+
+
+               className="add-cart"
+               onClick={() => addCartItem(lesson)}
+             >加入购物车
+         </>
+       } />
+
+   </>
  );
}
+let mapStateToProps = (state: CombinedState): CombinedState => state;
export default connect(
+  mapStateToProps,
+  actionCreators
)(Detail);
```

14.3 action-types.tsx

src\store\action-types.tsx

```
export const ADD = "ADD";

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
```

14.4 typings\cart.tsx

src\typings\cart.tsx

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

14.5 cart.tsx

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) {
        state.push({
          checked: false,
          count: 1,
          lesson: action.payload,
        });
      } else {
        state[oldIndex].count +=1;
      }
      break;
    case actionTypes.REMOVE_CART_ITEM:
      let removeIndex = state.findIndex(
        (item) => item.lesson.id === action.payload
      );
      state.splice(removeIndex,1);
      break;
    case actionTypes.CLEAR_CART_ITEMS:
      state.length = 0;
      break;
    case actionTypes.CHANGE_CART_ITEM_COUNT:
      let index = state.findIndex(
        (item) => item.lesson.id === action.payload.id
      );
      state[index].count=action.payload.count;
      break;
    case actionTypes.CHANGE_CHECKED_CART_ITEMS:
      let checkedIds = action.payload;
      state.forEach((item:any)=>{
        if(checkedIds.includes(item.lesson.id)){
          item.checked =true;
        }
      });
      break;
    case actionTypes.SETTLE:
      state = state.filter((item) => !item.checked);
      break;
    default:
      break;
  }
    return state;
}
```

14.6 reducers\home.tsx

src\store\reducers\home.tsx

```
import { AnyAction } from "redux";
import * as actionTypes from "../action-types";
import Slider from "@/typings/slider";
import Lesson from "@/typings/Lesson";
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 actionTypes.SET_CURRENT_CATEGORY:
+     state.currentCategory=action.payload;
+     break;
+   case actionTypes.GET_SLIDERS:
+     state.sliders = action.payload.data;
+     break;
+   case actionTypes.SET_LESSONS_LOADING:
+     state.lessons.loading = action.payload;
+     break;
+   case actionTypes.SET_LESSONS:
+     state.lessons.loading = false;
+     state.lessons.hasMore =  action.payload.hasMore;
+     state.lessons.list=[...state.lessons.list,...action.payload.list];
+     state.lessons.offset += action.payload.list.length;
+     break;
+   case actionTypes.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;
+     break;
+   default:
+     break;
+  }
+   return state;
}
```

14.7 reducers\index.tsx

src\store\reducers\index.tsx

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

export { CombinedState }
export default reducer;
```

14.8 store\index.tsx

src\store\index.tsx

```
import { createStore, applyMiddleware} from 'redux';
import reducers from './reducers';
import logger from 'redux-logger';
import thunk from 'redux-thunk';
import promise from 'redux-promise';
import { routerMiddleware } from 'redux-first-history';
+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 = applyMiddleware(thunk, routerMiddleware(history), promise, logger)(createStore)(persistedReducer);
+let persistor = persistStore(store);
+export  { store, persistor };
```

14.9 actionCreators\cart.tsx

src\store\actionCreators\cart.tsx

```
import * as actionTypes from "../action-types";
import { Lesson } from "@/typings/lesson";
import { message } from "antd";
import { push } from "redux-first-history";
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("/"));
    };
  },
};
```

14.10 Cart\index.tsx

src\routes\Cart\index.tsx

```
import React, { PropsWithChildren, useState } from "react";
import { connect } from "react-redux";
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 actionCreators from "@/store/actionCreators/cart";
import { CartItem } from "@/typings/cart";
type StateProps = ReturnType<typeof mapStateToProps>;
type DispatchProps = typeof actionCreators;
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:any) => 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 + parseFloat(item.lesson.price.replace(/[^0-9\.]/g,'')) * item.count,
      0
    );
  return (
    <>
      <NavHeader>购物车NavHeader>
      <Table
        rowKey={(row:any) => 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, actionCreators)(Cart);
```

zhufengketang-client (https://gitee.com/zhufengpeixun/zhufengketang-client) 复盘 (https://www.yuque.com/ninesean/blog/gh5x7f)

1. 创建服务器端项目

- 安装客户端工具 Robo 3T (https://robomongo.org/download)
- 安装 api 测试工具 Insomnia Core (https://insomnia.rest/download/#windows)

1.1 目录结构

```
.
├── package.json
├── src
│   └── index.ts
└── tsconfig.json
```

1.2 生成项目

```
mkdir zfkt2020-api
cd zfkt2020-api
cnpm init -y
```

1.3 安装依赖

```
cnpm i express mongoose body-parser bcryptjs jsonwebtoken morgan cors validator helmet dotenv multer http-status-codes -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 cross-env -D
```

模块名 英文 中文 express Fast, unopinionated, minimalist web framework for node. 基于 Node.js 平台，快速、开放、极简的 Web 开发框架 @types/express This package contains type definitions for Express express 的类型声明 mongoose Mongoose is a MongoDB object modeling tool designed to work in an asynchronous environment. Mongoose supports both promises and callbacks. Mongoose 为模型提供了一种直接的，基于 scheme 结构去定义你的数据模型。它内置数据验证， 查询构建，业务逻辑钩子等，开箱即用 @types/mongoose This package contains type definitions for Mongoose mongoose 的类型声明 body-parser Node.js body parsing middleware. body-parser 是一个 HTTP 请求体解析中间件，使用这个模块可以解析 JSON、Raw、文本、URL-encoded 格式的请求体，Express 框架中就是使用这个模块做为请求体解析中间件 bcryptjs Optimized bcrypt in JavaScript with zero dependencies. Compatible to the C++ bcrypt binding on node.js and also working in the browser. bcryptjs 是一个第三方加密库，用来实现在 Node 环境下的 bcrypt 加密 @types/bcryptjs jsonwebtoken An implementation of JSON Web Tokens JSON Web Token（JWT）是一个非常轻巧的规范。这个规范允许我们使用 JWT 在用户和服务器之间传递安全可靠的信息 @types/jsonwebtoken This package contains type definitions for jsonwebtoken jsonwebtoken 的类型声明 morgan HTTP request logger middleware for node.js morgan 是 express 默认的日志中间件，也可以脱离 express，作为 node.js 的日志组件单独使用 @types/morgan This package contains type definitions for morgan morgan 的类型声明 cors CORS is a node.js package for providing a Connect/Express middleware that can be used to enable CORS with various options. CORS 是用于提供 Connect / Express 中间件的 node.js 程序包，可用于启用具有各种选项的 CORS @types/cors This package contains type definitions for cors cors 的类型声明 validator A library of string validators and sanitizers 一个用于字符串验证和净化的库 @types/validator This package contains type definitions for validator.js validator 的类型声明 helmet Helmet helps you secure your Express apps by setting various HTTP headers. It's not a silver bullet, but it can help! Helmet 可通过设置各种 HTTP 标头来帮助您保护 Express 应用程序 @types/helmet This package contains type definitions for helmet helmet 的类型声明 dotenv Dotenv is a zero-dependency module that loads environment variables from a .env file into process.env. Storing configuration in the environment separate from code is based on The Twelve-Factor App methodology. Dotenv 是一个零依赖模块，可将环境变量从.env 文件加载到 process.env 中 multer Multer is a node.js middleware for handling multipart/form-data, which is primarily used for uploading files. It is written on top of busboy for maximum efficiency. Multer 是用于处理 `multyparty/formdata`

类型请求体的 node.js 中间件，主要用于上传文件。 它是在 busboy 之上编写的，以实现最大效率。 @types/multer This package contains type definitions for multer multer 的类型声明 typescript TypeScript is a language for application-scale JavaScript ypeScript 是用于应用程序级 JavaScript 的语言 @types/node This package contains type definitions for Node.js 该软件包包含 Node.js 的类型定义 ts-node-dev Tweaked version of node-dev that uses ts-node under the hood. 调整后的版本，在后台使用 ts-node nodemon nodemon is a tool that helps develop node.js based applications by automatically restarting the node application when file changes in the directory are detected. nodemon 是一种工具，可在检测到目录中的文件更改时通过自动重新启动应用程序来帮助开发基于 node.js 的应用程序。

1.4 初始化 tsconfig.json

- 执行以下命令并选择 `node`

```
npx tsconfig.json
```

1.5 .gitignore

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

1.6 .env

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

1.7 src\index.ts

```
import express, { Express } from "express";
const app: Express = express();
const PORT: number = (process.env.PORT && parseInt(process.env.PORT)) || 8000;
app.listen(PORT, () => {
    console.log(`Running on http://localhost:${PORT}`);
});
```

1.8 package.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"
+  }
```

1.9 测试

```
npm run build
npm run dev
npm run start
```

2. 用户管理

2.1 参考

2.1.1 介绍

- 本章的是编写用户管理相关的接口
- 本章需要编写以下接口
  - 用户注册
  - 用户登录
  - 验证用户是否登录
  - 上传头像

2.1.2 本章目录

```
.
├── package.json
├── src
│   ├── controller
│   │   ├── slider.ts
│   │   └── user.ts
│   ├── exceptions
│   │   └── HttpException.ts
│   ├── index.ts
│   ├── middlewares
│   │   └── errorMiddleware.ts
│   ├── models
│   │   ├── index.ts
│   │   ├── slider.ts
│   │   └── user.ts
│   ├── public
│   ├── typings
│   │   ├── express.d.ts
│   │   └── jwt.ts
│   └── utils
│       └── validator.ts
└── tsconfig.json
```

2.1.3 本章效果

2.2 src/index.ts

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}`);
  });
})();
```

2.3 HttpException.ts

src/exceptions/HttpException.ts

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

2.4 errorMiddleware.ts

src/middlewares/errorMiddleware.ts

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

2.5 src/utils/validator.ts

src/utils/validator.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(email)) {
    errors.email = "邮箱格式必须合法";
  }
  return { errors, valid: Object.keys(errors).length == 0 };
};
```

2.6 src/typings/jwt.ts

src/typings/jwt.ts

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

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

2.7 src\models\index.ts

src\models\index.ts

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

2.8 src/models/user.ts

src/models/user.ts

```
import mongoose, { Schema, Model, Document } 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
}
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,toJSON:{
    transform(_doc,ret){
        ret.id=ret._id;
        delete ret._id;
        delete ret.__v;
        delete ret.password;
        return ret;
    }
} });

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: any) {
    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.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>('User', UserSchema);
```

2.9 src\controller\user.ts

src\controller\user.ts

```
import { Request, Response, NextFunction } from 'express';
import { validateRegisterInput } from '../utils/validator';
import HttpException from '../exceptions/HttpException';
import StatusCodes 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) {
                    res.json({
                        success: true,
                        data: user.toJSON()
                    });
                } else {
                    next(new HttpException(StatusCodes.UNAUTHORIZED, `用户不合法!`));
                }
            } catch (error) {
                next(new HttpException(StatusCodes.UNAUTHORIZED, `token不合法!`));
            }

        } else {
            next(new HttpException(StatusCodes.UNAUTHORIZED, `token未提供!`));
        }
    } else {
        next(new HttpException(StatusCodes.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(StatusCodes.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(StatusCodes.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(StatusCodes.UNAUTHORIZED, `登录失败`);
        }
    } catch (error) {
        next(error);
    }
}
export const uploadAvatar = async (req: Request, res: Response, _next: NextFunction) => {
    let { userId } = req.body;
    let domain = process.env.DOMAIN || `${req.protocol}://${req.headers.host}`;
    let avatar = `${domain}/uploads/${req.file.filename}`;
    await User.updateOne({ _id: userId }, { avatar });
    res.send({ success: true, data: avatar });
}
```

2.10 typings\express.d.ts

src\typings\express.d.ts

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

2.11 .env

.env

```
JWT_SECRET_KEY=zhufeng
MONGODB_URL=mongodb:
PORT=8899
DOMAIN=http:
```

3. 后台轮播图轮播图接口

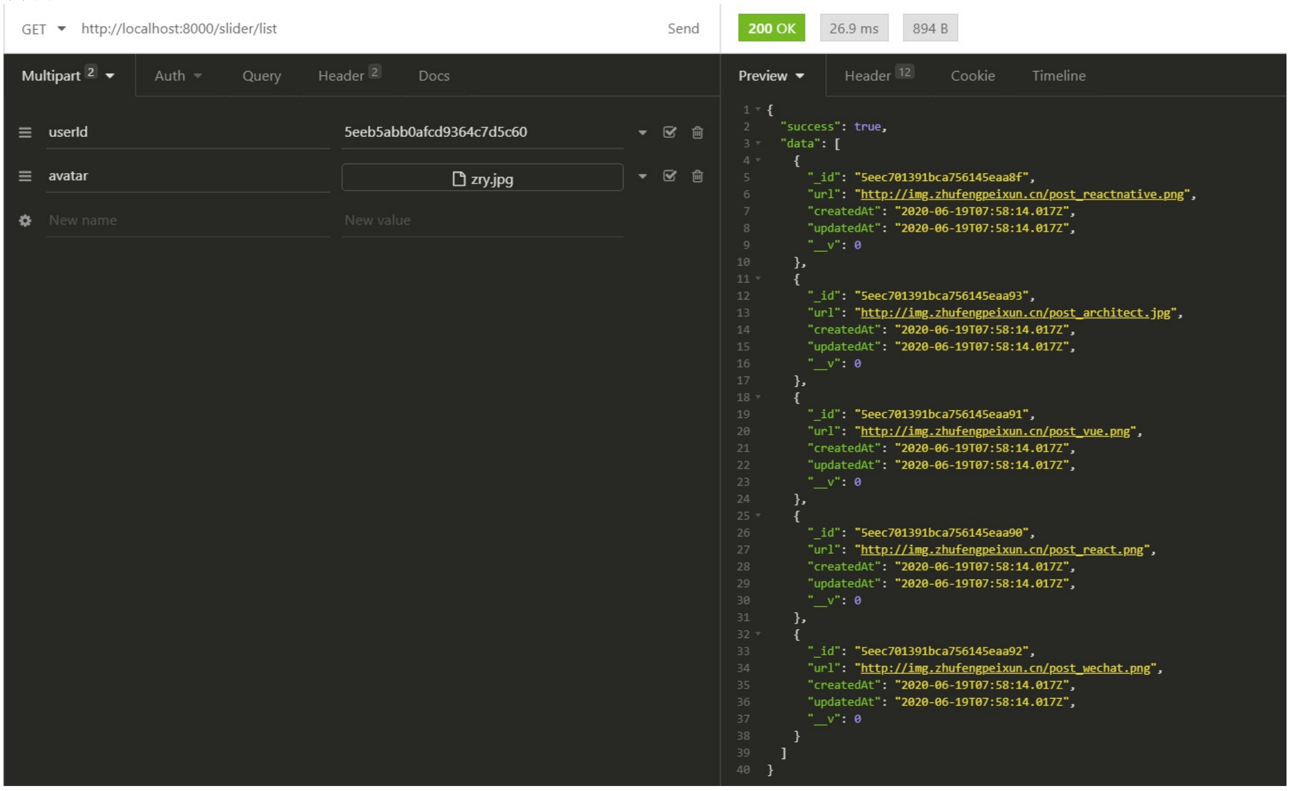
- 本章是编写轮播图接口

3.1 参考

3.1.1 本章目录

```
.
├── package.json
├── src
│   ├── controller
│   │   ├── slider.ts
│   │   └── user.ts
│   ├── exceptions
│   │   └── HttpException.ts
│   ├── index.ts
│   ├── middlewares
│   │   └── errorMiddleware.ts
│   ├── models
│   │   ├── index.ts
│   │   ├── slider.ts
│   │   └── user.ts
│   ├── public
│   ├── typings
│   │   ├── express.d.ts
│   │   └── jwt.ts
│   └── utils
│       └── validator.ts
└── tsconfig.json
```

本章效果



```
http://localhost:8000/slider/list

userId    5eeb5abb0afcd9364c7d5c60
avatar    zry.jpg
```

```
{
  "success": true,
  "data": [
    {
      "_id": "5eec701391bca756145eaa8f",
      "url": "http://img.zhufengpeixun.cn/post_reactnative.png",
      "createdAt": "2020-06-19T07:58:14.017Z",
      "updatedAt": "2020-06-19T07:58:14.017Z",
      "__v": 0
    },
    {
      "_id": "5eec701391bca756145eaa93",
      "url": "http://img.zhufengpeixun.cn/post_architect.jpg",
      "createdAt": "2020-06-19T07:58:14.017Z",
      "updatedAt": "2020-06-19T07:58:14.017Z",
      "__v": 0
    },
    {
      "_id": "5eec701391bca756145eaa91",
      "url": "http://img.zhufengpeixun.cn/post_vue.png",
      "createdAt": "2020-06-19T07:58:14.017Z",
      "updatedAt": "2020-06-19T07:58:14.017Z",
      "__v": 0
    },
    {
      "_id": "5eec701391bca756145eaa90",
      "url": "http://img.zhufengpeixun.cn/post_react.png",
      "createdAt": "2020-06-19T07:58:14.017Z",
      "updatedAt": "2020-06-19T07:58:14.017Z",
      "__v": 0
    },
    {
      "_id": "5eec701391bca756145eaa92",
      "url": "http://img.zhufengpeixun.cn/post_wechat.png",
      "createdAt": "2020-06-19T07:58:14.017Z",
      "updatedAt": "2020-06-19T07:58:14.017Z",
      "__v": 0
    }
  ]
}
```

3.2 src\index.ts

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:any = [
+            { url: 'http://img.zhufengpeixun.cn/post_reactnative.png' },
+            { url: 'http://img.zhufengpeixun.cn/post_react.png' },
+            { url: 'http://img.zhufengpeixun.cn/post_vue.png' },
+            { url: 'http://img.zhufengpeixun.cn/post_wechat.png' },
+            { url: 'http://img.zhufengpeixun.cn/post_architect.jpg' }
+        ];
+        Slider.create(sliders);
+    }
+}
```

3.3 controller\slider.ts

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 });
};
```

3.4 models\slider.ts

src\models\slider.ts

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

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

3.5 src\models\index.ts

src\models\index.ts

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

4. 课程管理后端接口

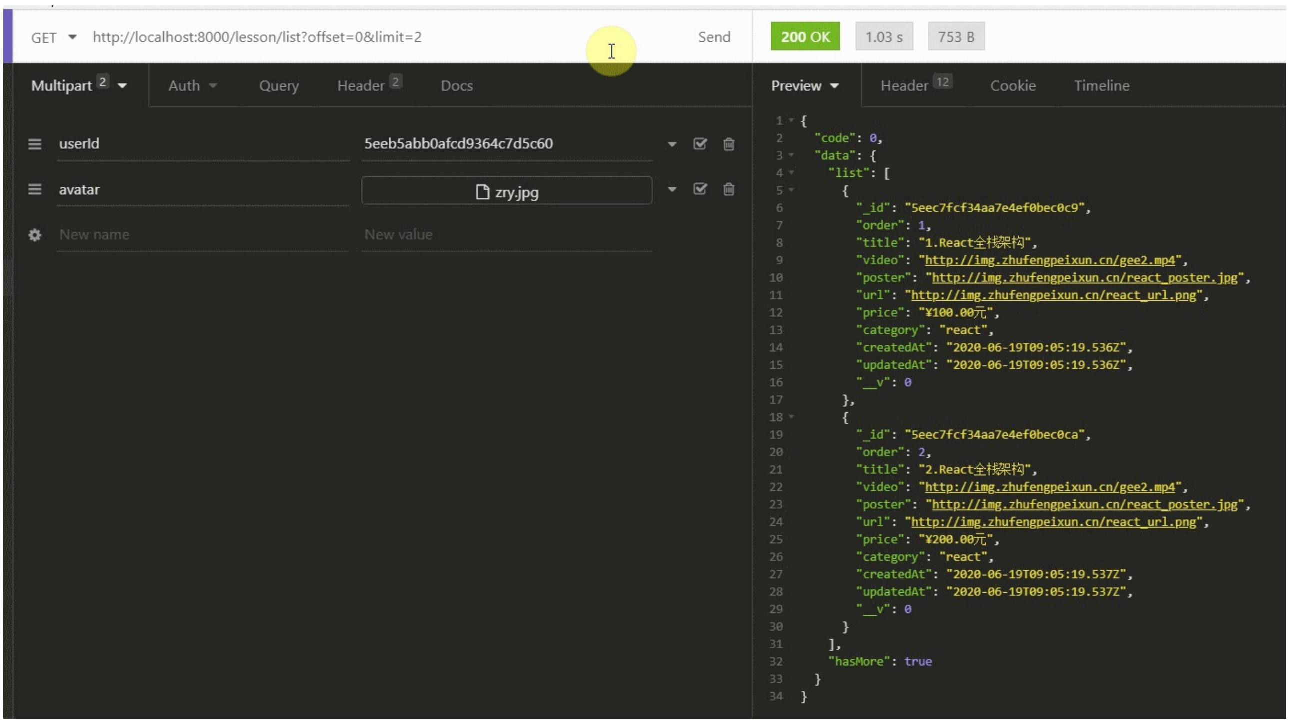
- 本章主要编写课程管理的后台接口

4.1 参考

4.1.1 本章目录

```
.
├── package.json
├── src
│   ├── controller
│   │   ├── lesson.ts
│   │   ├── slider.ts
│   │   └── user.ts
│   ├── exceptions
│   │   └── HttpException.ts
│   ├── index.ts
│   ├── middlewares
│   │   └── errorMiddleware.ts
│   ├── models
│   │   ├── index.ts
│   │   ├── lesson.ts
│   │   ├── slider.ts
│   │   └── user.ts
│   ├── public
│   ├── typings
│   │   ├── express.d.ts
│   │   └── jwt.ts
│   └── utils
│       └── validator.ts
└── tsconfig.json
```

4.1.2 本章效果



http://localhost:8000/lesson/list?offset=0&limit=2

```
userId    5eeb5abb0afcd9364c7d5c60
avatar    zry.jpg
```

```
{
  "code": 0,
  "data": {
    "list": [
      {
        "_id": "5eec7fcf34aa7e4ef0bec0c9",
        "order": 1,
        "title": "1.React全栈架构",
        "video": "http://img.zhufengpeixun.cn/gee2.mp4",
        "poster": "http://img.zhufengpeixun.cn/react_poster.jpg",
        "url": "http://img.zhufengpeixun.cn/react_url.png",
        "price": "¥100.00元",
        "category": "react",
        "createdAt": "2020-06-19T09:05:19.536Z",
        "updatedAt": "2020-06-19T09:05:19.536Z",
        "__v": 0
      },
      {
        "_id": "5eec7fcf34aa7e4ef0bec0ca",
        "order": 2,
        "title": "2.React全栈架构",
        "video": "http://img.zhufengpeixun.cn/gee2.mp4",
        "poster": "http://img.zhufengpeixun.cn/react_poster.jpg",
        "url": "http://img.zhufengpeixun.cn/react_url.png",
        "price": "¥200.00元",
        "category": "react",
        "createdAt": "2020-06-19T09:05:19.537Z",
        "updatedAt": "2020-06-19T09:05:19.537Z",
        "__v": 0
      }
    ],
    "hasMore": true
  }
}
```

4.2 src\index.ts

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("mongodb://localhost/zhufengketang");
  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 initSliders: any = [
      { url: "http://img.zhufengpeixun.cn/post_reactnative.png" },
      { url: "http://img.zhufengpeixun.cn/post_react.png" },
      { url: "http://img.zhufengpeixun.cn/post_vue.png" },
      { url: "http://img.zhufengpeixun.cn/post_wechat.png" },
      { url: "http://img.zhufengpeixun.cn/post_architect.jpg" },
    ];
    Slider.create(initSliders);
  }
}

+async function createLessons() {
+  const lessons = await Lesson.find();
+  if (lessons.length == 0) {
+    const lessons: any = [
+      {
+        order: 1,
+        title: "1.React全栈架构",
+        video: "http://img.zhufengpeixun.cn/gee2.mp4",
+        poster: "http://img.zhufengpeixun.cn/react_poster.jpg",
+        url: "http://img.zhufengpeixun.cn/react_url.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://img.zhufengpeixun.cn/react_poster.jpg",
+        url: "http://img.zhufengpeixun.cn/react_url.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://img.zhufengpeixun.cn/react_poster.jpg",
+        url: "http://img.zhufengpeixun.cn/react_url.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://img.zhufengpeixun.cn/react_poster.jpg",
+        url: "http://img.zhufengpeixun.cn/react_url.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://img.zhufengpeixun.cn/react_poster.jpg",
+        url: "http://img.zhufengpeixun.cn/react_url.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://img.zhufengpeixun.cn/vue_poster.png",
+        url: "http://img.zhufengpeixun.cn/vue_url.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://img.zhufengpeixun.cn/vue_poster.png",
+        url: "http://img.zhufengpeixun.cn/vue_url.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://img.zhufengpeixun.cn/vue_poster.png",
+        url: "http://img.zhufengpeixun.cn/vue_url.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://img.zhufengpeixun.cn/vue_poster.png",
+        url: "http://img.zhufengpeixun.cn/vue_url.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://img.zhufengpeixun.cn/vue_poster.png",
+        url: "http://img.zhufengpeixun.cn/vue_url.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://img.zhufengpeixun.cn/react_poster.jpg",
+        url: "http://img.zhufengpeixun.cn/react_url.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://img.zhufengpeixun.cn/react_poster.jpg",
+        url: "http://img.zhufengpeixun.cn/react_url.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://img.zhufengpeixun.cn/react_poster.jpg",
+        url: "http://img.zhufengpeixun.cn/react_url.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://img.zhufengpeixun.cn/react_poster.jpg",
+        url: "http://img.zhufengpeixun.cn/react_url.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://img.zhufengpeixun.cn/react_poster.jpg",
+        url: "http://img.zhufengpeixun.cn/react_url.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://img.zhufengpeixun.cn/vue_poster.png",
+        url: "http://img.zhufengpeixun.cn/vue_url.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://img.zhufengpeixun.cn/vue_poster.png",
+        url: "http://img.zhufengpeixun.cn/vue_url.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://img.zhufengpeixun.cn/vue_poster.png",
+        url: "http://img.zhufengpeixun.cn/vue_url.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://img.zhufengpeixun.cn/vue_poster.png",
+        url: "http://img.zhufengpeixun.cn/vue_url.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://img.zhufengpeixun.cn/vue_poster.png",
+        url: "http://img.zhufengpeixun.cn/vue_url.png",
+        price: "¥1000.00元",
+        category: "vue",
+      },
+    ];
+    Lesson.create(lessons);
+  }
+}
```

4.3 src\models\index.ts

src\models\index.ts

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

4.4 controller\lesson.ts

src\controller\lesson.ts

```
import { Request, Response } from "express";
import { ILessonDocument, Lesson } from "../models";
import {FilterQuery} from 'mongoose';
export const list = async (req: Request, res: Response) => {
  let { category } = req.query;
  let offset: any = req.query.offset;
  let limit: any = req.query.limit;
  offset = isNaN(offset) ? 0 : parseInt(offset);
  limit = isNaN(limit) ? 5 : parseInt(limit);
  let query: FilterQuery = {};
  if (category && category != "all") query.category = category as string;
  let total = await Lesson.count(query);
  let list = await Lesson.find(query)
    .sort({ order: 1 })
    .skip(offset)
    .limit(limit);
  list = list.map((item:ILessonDocument)=>item.toJSON());
  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 });
};
```

4.5 models\lesson.ts

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;
}
const LessonSchema: Schema = new Schema(
  {
    order: Number,
    title: String,
    video: String,
    poster: String,
    url: String,
    price: String,
    category: String,
  },
  { timestamps: true,toJSON:{
    transform(_doc,ret){
        ret.id=ret._id;
        delete ret._id;
        delete ret.__v;
        delete ret.password;
        return ret;
    }
} }
);

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

5.immer

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

5.1 produce

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

```
let { produce } = require('immer');
let baseState = {}

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

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

```
let { produce } = require('immer');
let baseState = {
  ids: [1],
  pos: {
    x: 1,
    y: 1
  }
}

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

```
let { produce } = require('immer');

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