# E宠商城项目记录

# 第一章 启动项目

* 1. create-react-app hello-react(hello-react为项目根目录的名字)
  2. 删除src的所有东西(个人习惯)
  3. 将public下的index.html的东西整理一下，删除不要的干扰项

|  |
| --- |
| <!DOCTYPE **html**> <**html lang="en"**>  <**head**>  <**meta charset="utf-8"**>  <**meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no"**>  <**meta name="description" content="E宠商城是专业宠物用品商城,网站提供宠物狗、宠物猫、宠物用品、狗粮、猫粮、宠物领养等,宠物市场近万种狗狗，猫咪用品任你选,上千知名宠物品牌【正牌保障】货到付款30天无条件退换货。品质养宠,尽在E宠！"**>   <**meta name="theme-color" content="#000000"**>  *<!--添加小图标-->* <**link rel="icon" href="./favicon-E.ico"**>  <**title**>【E宠商城】宠物商城\_宠物用品\_宠物狗\_宠物店\_宠物网</**title**>  </**head**>  <**body**>  <**div id="root"**></**div**>  </**body**> </**html**> |

# 预测试

## 2.1 创建index.js入口文件

|  |
| --- |
| **import** React **from 'react'**; **import** ReactDOM **from 'react-dom'**; **import** App **from './components/App'**; ReactDOM.render(<**App** />, **document**.getElementById(**'root'**)); |

## 2.2 创建components文件用于放置组件

|  |
| --- |
| **import** React **from 'react'**; **export default class** App **extends** React.Component{  render(){  **return** (  <**div**>  app组件内容。。。  </**div**>  )  } } |

# 下载依赖包注册路由

## 3.1 下载包

3.1.1 npm install react-router-dom@4

3.1.2 npm install axios

3.1.3 npm install better-scroll

3.1.4 npm install mockjs

3.1.5 npm install postcss-px2rem

3.1.6 npm install prop-types

3.1.7 npm install redux

3.1.8 npm install react-redux

3.1.9 npm install react-router-dom

3.1.10 npm install react-lazyimg-component

3.1.11 npm install redux-devtools-extension

3.1.12 npm install react-thunk

3.1.13 npm install swiper

package.json说明

|  |
| --- |
| {  **"name"**: **"hello-react"**,  **"version"**: **"0.1.0"**,  **"private"**: **true**,  **"dependencies"**: {  **"autoprefixer"**: **"7.1.6"**,  **"axios"**: **"^0.18.0"**, *// 发送ajax请求* **"babel-core"**: **"6.26.0"**,  **"babel-eslint"**: **"7.2.3"**,  **"babel-jest"**: **"20.0.3"**,  **"babel-loader"**: **"7.1.2"**,  **"babel-preset-react-app"**: **"^3.1.1"**,  **"babel-runtime"**: **"6.26.0"**,  **"better-scroll"**: **"^1.10.2"**, *// 设置滑动的组件* **"case-sensitive-paths-webpack-plugin"**: **"2.1.1"**,  **"chalk"**: **"1.1.3"**,  **"css-loader"**: **"0.28.7"**,  **"dotenv"**: **"4.0.0"**,  **"dotenv-expand"**: **"4.2.0"**,  **"eslint"**: **"4.10.0"**,  **"eslint-config-react-app"**: **"^2.1.0"**,  **"eslint-loader"**: **"1.9.0"**,  **"eslint-plugin-flowtype"**: **"2.39.1"**,  **"eslint-plugin-import"**: **"2.8.0"**,  **"eslint-plugin-jsx-a11y"**: **"5.1.1"**,  **"eslint-plugin-react"**: **"7.4.0"**,  **"extract-text-webpack-plugin"**: **"3.0.2"**,  **"file-loader"**: **"1.1.5"**,  **"fs-extra"**: **"3.0.1"**,  **"html-webpack-plugin"**: **"2.29.0"**,  **"jest"**: **"20.0.4"**,  **"jquery"**: **"^1.12.4"**,  **"mockjs"**: **"^1.0.1-beta3"**, *// 模拟后台返回数据* **"object-assign"**: **"4.1.1"**,  **"postcss-flexbugs-fixes"**: **"3.2.0"**,  **"postcss-loader"**: **"2.0.8"**,  **"postcss-px2rem"**: **"^0.3.0"**, *// 自动转换px为rem的插件* **"promise"**: **"8.0.1"**,  **"prop-types"**: **"^15.6.1"**, *// 设置react组件传入props数据的数据类型和必要性，可见官网* **"px2rem-loader"**: **"^0.1.9"**,  **"raf"**: **"3.4.0"**,  **"react"**: **"^16.3.2"**,  **"react-dev-utils"**: **"^5.0.1"**,  **"react-dom"**: **"^16.3.2"**,  **"react-lazyimg-component"**: **"0.0.2"**, *// 图片懒加载插件* **"react-redux"**: **"^5.0.7"**, *// redux：管理状态的react插件需配合redux舒勇* **"react-router-dom"**: **"^4.2.2"**, *// 管理react路由* **"react-validate-framework"**: **"^0.14.5"**,  **"redux"**: **"^3.7.2"**, *// 管理状态* **"redux-devtools-extension"**: **"^2.13.2"**, *// 开发者工具：主要用来在控制台查看redux状态的，需配合redux-thunk* **"redux-thunk"**: **"^2.2.0"**, *// 使用中间键* **"resolve"**: **"1.6.0"**,  **"style-loader"**: **"0.19.0"**,  **"sw-precache-webpack-plugin"**: **"0.11.4"**,  **"swiper"**: **"^4.2.2"**, *// 轮播图插件，网址：http://www.swiper.com.cn/* **"url-loader"**: **"0.6.2"**,  **"webpack"**: **"3.8.1"**,  **"webpack-dev-server"**: **"2.9.4"**,  **"webpack-manifest-plugin"**: **"1.3.2"**,  **"whatwg-fetch"**: **"2.0.3"** },  **"scripts"**: {  **"start"**: **"node scripts/start.js"**,  **"build"**: **"node scripts/build.js"**,  **"test"**: **"node scripts/test.js --env=jsdom"** },  **"jest"**: {  **"collectCoverageFrom"**: [  **"src/\*\*/\*.{js,jsx,mjs}"** ],  **"setupFiles"**: [  **"<rootDir>/config/polyfills.js"** ],  **"testMatch"**: [  **"<rootDir>/src/\*\*/\_\_tests\_\_/\*\*/\*.{js,jsx,mjs}"**,  **"<rootDir>/src/\*\*/?(\*.)(spec|test).{js,jsx,mjs}"** ],  **"testEnvironment"**: **"node"**,  **"testURL"**: **"http://localhost"**,  **"transform"**: {  **"^.+\\.(js|jsx|mjs)$"**: **"<rootDir>/node\_modules/babel-jest"**,  **"^.+\\.css$"**: **"<rootDir>/config/jest/cssTransform.js"**,  **"^(?!.\*\\.(js|jsx|mjs|css|json)$)"**: **"<rootDir>/config/jest/fileTransform.js"** },  **"transformIgnorePatterns"**: [  **"[/\\\\]node\_modules[/\\\\].+\\.(js|jsx|mjs)$"** ],  **"moduleNameMapper"**: {  **"^react-native$"**: **"react-native-web"** },  **"moduleFileExtensions"**: [  **"web.js"**,  **"js"**,  **"json"**,  **"web.jsx"**,  **"jsx"**,  **"node"**,  **"mjs"** ]  },  **"babel"**: {  **"presets"**: [  **"react-app"** ]  },  **"eslintConfig"**: {  **"extends"**: **"react-app"** } } |

## 3.2. react-router相关API

## 3.3 组件

1. <BrowserRouter>
2. <HashRouter>
3. <Route>
4. <Redirect>
5. <Link>
6. <NavLink>
7. <Switch>

## 3.4 其它

1. history对象
2. match对象
3. withRouter函数

## 3.5 注册路由

### 3.5.1 Link和NavLink路由链接

推荐用NavLink，在点击的时候会自动添加activeClassName类默认值为‘active’

### 3.5.2 Route路由组件容器

1. 相对于3的版本4版本中的Route更为灵活，可放在任何需要路由组件的地方

2. 扩展：如何向路由组件传递参数？

|  |
| --- |
| <**Route path='/main' render=**{() => <**Main data='传递的数据' path='main'**/>}></**Route**> |

### 3.5.3 踩坑

1. 问题描述：

当使用BrowserRouter时候，一旦有二级路由，刷新页面会报错。

1. 解决办法：

目前看来使用 HashRouter是最快的解决方式，关于BrowserRouter和HashRouter的区别，可自行上网查看

1. 相关文章：

<https://blog.csdn.net/weixin_39168678/article/details/79756305>

# 适配

## 4.1 需要掌握知识点

1. 布局视口，设备视口，分辨率，设备像素比(dpr)

2. viewport，meta设置完美视口

3. 常用的适配方案

## 4.2 本项目适配方案：

1. 在iphone6下设置rem = 100px，其他的机型根据手机屏幕的设备宽度比例动态设置

2. 以下适配方案可拿走直接使用

|  |
| --- |
| (**function** (doc, win) {  **var** docEl = doc.**documentElement**,  *//orientationchange 苹果公司为移动 Safari中添加了 orientationchange 事件，以便开发人员能够确定用户何时将设 备由横向查看模式切换为纵向查看模式。移动 Safari的 window.orientation 属性中可能包含 3个值： 0 表示肖像模式，90 表示向左旋转的横向模式（“主屏幕”按钮在右侧），-90 表示向右旋转的横向模 式（“主屏幕”按钮在左侧）。* resizeEvt = **'orientationchange' in window** ? **'orientationchange'** : **'resize'**,  *recalc* = **function** () {  **var** clientWidth = docEl.**clientWidth**;  **var** dpr = **window**.devicePixelRatio; *// 思路： 如果公司的设计稿是按照iphone6的分辨率来设定的(750 \* 1334)  // 问题：当前获取的clientWidth是视觉视口宽度(设备逻辑像素), 如果不设定viewport=device-width，获取的恒定980  // 如何获取设备的真实分辨率呢？ 设备逻辑像素 \* dpr* **if** (!clientWidth) **return**;  **if**(clientWidth>=375){  *// 根据clientWidth & 阿里的适配方案考虑rem适配* **fontSize** = 100;  *// 如果设计稿是750的话，再考虑视网膜屏幕的dpr值，在高分辨率下显示的内容样式也是等比清晰* docEl.**style**.**fontSize** = **fontSize** + **'px'**;  }**else**{  **fontSize** = 100 \* (clientWidth / 375) + **'px'**;  docEl.**style**.**fontSize** = **fontSize**;  }  };   **if** (!doc.addEventListener) **return**;  *// 绑定事件 浏览器窗口大小发生改变* win.addEventListener(resizeEvt, *recalc*, **false**);  *// 文档内容加载完毕* doc.addEventListener(**'DOMContentLoaded'**, *recalc*, **false**); })(**document**, **window**); |

## 4.3 相关库：postcss-px2rem

1. 库说明：依赖当前库帮助开发者自动将px单位的值根据比例转换成rem

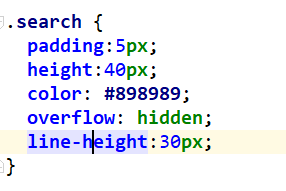
2. 如果rem设置为100px的话，当设置14px的时候转换为rem为0.14rem，在书写和 换算上较为麻烦，可使用less,stylus等解决

3. postcss-px2rem可以设置计算比例自动计算rem的值。如：比例为100，当书写100px 的时候在页面反应出来的是100/100rem正好相应根标签的100px;

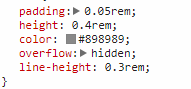
4. webpack.config.dev.js配置：



5. 使用方式，正常编写



1. 页面反馈



# CSS modules

## 5.1 前言

可能大家都熟知的js模块化，一谈到js模块化可能张口就来的是ES，commonjs等等，而且也都知道js模块化的好处。

当react兴起的时候引出一个概念叫css in js，熟悉react的人都知道react组件化编程的思想，并且提出的JSX语法，就是在js里可以直接写html当然也可以写css，但是带来的问题是结构，样式耦合度高，不方便管理

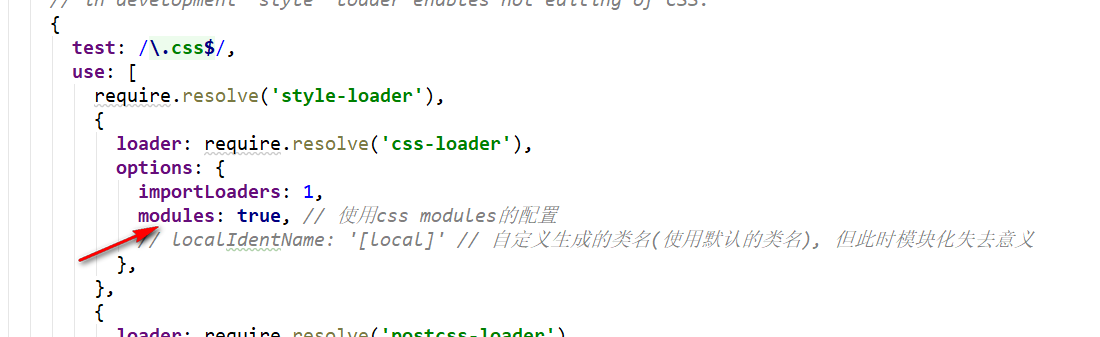
以上的问题可以利用css modules的思想解决，就是同js一样，将css也模块化，并且同js分离，这样的话样式类就是私有的，不会互相影响，且方便管理。

## 5.2 css modules工作机制

1. 都使用类定义样式

2. 页面加载的时候css-loader会将类名转换为哈希字符串(这样类名就是独一无二的)

## 5.3 webpack.config配置



## 5.4 可能遇到的问题

1. 使用其他插件的时候需要指定的类名，如：swiper插件的swiper-container类名，但使用css modules之后类名转换为哈希值，swiper插件无法识别

2. 有时候需要根据类名操作虚拟DOM

## 5.5 解决方案

1. css modules提供了定义全局样式的方式： :global(类名){}，当这样设置的时候类名不会转换

|  |
| --- |
| :**global**(.**swiper-container**){  **height**:160**px**;  **width**:100%;  **position**: **relative**; } |

# 踩坑系列

## swiper插件

1. swiper插件需要在componentDidMount中使用，vue中在mounted
2. 有时候在componentDidMount中也无法获取容器对象，此时需要使用延时定时器获取
3. Swiper插件的类的固定的，设置样式的时候需要使用全局样式 :global(类名){}
4. 在componentWillUnmount中关闭定时器，建议养成该习惯，自己开的定时器在不需要的时候一定要手动关掉

|  |
| --- |
| componentDidMount(){  **this**.**timeoutId** = setTimeout(() => {  *// swiper插件 设置轮播图效果* **this**.**swiper** = **new** Swiper(**'.swiper-container'**, {  **direction**: **'horizontal'**,  **loop**: **true**,  **autoplay**: **true**,  *// 如果需要分页器* **pagination**: {  **el**: **'.swiper-pagination'**,  }  })  }, 1000) } componentWillUnmount(){  clearTimeout(**this**.**timeoutId**); } |

## better-scroll插件

1. 子元素宽度/高度一定要大于父元素
2. 在componentDidMount中使用，vue中在mounted
3. 同swiper一样也需要延时定时器
4. 在componentWillUnmount中关闭定时器

## Mockjs

### 注册地址

|  |
| --- |
| *// 指定监听的url和对应的json数据模板* Mock.mock(**'/getMain'**, {  **code**: 0,  **data**: mainData.menus }) |

### 请求地址容易出错

发送ajax请求的时候地址就写注册的地址：/getMain，否则容易出错

### 温馨提示

工作中想要使用mockjs的小伙伴在后台开发接口后正式对接的时候一定要把mockjs的东西删除掉，因为mockjs会拦截所有的ajax请求，如果不删除，会导致ajax请求失效。

第七章

### 手机动态验证登录的图片验证

使用的是jQuery的一个插件，各种坑，各种修改，最后勉强可以，但感觉意义不大。