## Omnyiq-sc1项目

现在这个项目是可用于服务端渲染的版本。

首先，介绍一下这个项目是参考github上一个脚手架工具来写的：<https://github.com/bodyno/react-starter-kit.git>。 所以一些初始的配置说明和安装启动这里就不赘述了，读下README.md文件就好。

这里还是说一下安装启动时遇到的一些问题：

首先定位到项目目录下安装一些模块包的时候可使用的npm install ,但是这个安装的很慢，所以最好使用淘宝的镜像来安装:cnpm install , 如果没有cnpm也需要先安装这个包：npm install cnpm -g，全局安装就好。

然后，在安装package.json文件里的依赖包时，有些可能会提示没有安装成功，然后可以把这些包单独在安装一遍：cnpm install [node-sass@3.8.0](mailto:node-sass@3.8.0) 最好是要加上package.json里规定的版本好，如果不加版本号的话默认就会安装最新版本的，但是往往最新版本的如果有大的改动的话就会和现在你的使用方式不兼容，或者和其他依赖包不兼容。上面写的那个node-sass包就有这个问题，如果默认安装最新版本的直接提示就安装不成功了。

运行的话输入命令：npm start或者npm run start . Start命令默认是在开发环境（development）运行的，如果想在生产环境（production）运行的话可以输入命令：npm run compile ,先把文件编译到相应的dist目录下，然后再输入命令：npm run start --prod.

在linux环境下安装运行的问题：

。。。。。。。

文件结构：

现在的项目文件结构和它README.md里写的不太一样，这里重新写一遍：

```

.

├── bin

│ ├── compile.js # 运行编译的脚本，会把所有资源打包到dist目录下

│ ├── server.js # 启动项目的入口脚本

│=======================================================

├── build # 所有打包有关的webpack配置项

│ ├── webpack-compiler.js # 运行编译时的配置

│ ├── webpack.config.js # 公共的webpack配置文件

│ ├── webpack.config.client.js # 打包客户端资源的webpack配置文件

│ ├── webpack.config.server.js # 打包服务器端资源的webpack配置文件

│=======================================================

├── config # 项目配置文件

│ ├── environments.js # 配置运行环境的一些参数。

│ ├── index.js # 一些全局的打包和运行的配置参数

│ ├── layout.js # 在服务端构造静态html时需要用到的配置，包括title和meta的配置， 还有在head和body引入的静态.css和.js文件。

│=============================================================

├── server # Koa 程序 (使用 webpack 中间件)

│ ├── middleware # 一些webpack中间件。

│ ├── main.js # 服务端程序入口文件，应该是主要代码文件，入

│ 口是bin目录下的server.js文件。

│ ├── router.js # 这里是一些在服务端就去获取数据的配置， 当刷新页面进入时就会在这个文件里匹配当前路由并在服务端就去拿好数据存入store里。如果不是刷新进入其他页面不会运行这个文件的代码。

│==========================================================

├── src # 程序主要功能源文件

│ ├── components # 全局可复用的表现组件(Presentational Components)

│ ├── containers # 全局可复用的最外层的容器组件

│ ├── layouts # 路由里容纳其他主要功能组件的公用布局容器

│ ├── modules # 主要用来解析config/layout.js里配置的那些静态

│ css和js文件用的

│ ├── reducers # 各个功能模块的action和reducer集中到各自的一个

│ reducer文件中。

│ ├── static # 静态文件，字体、样式、图片、js(不要到处imported源文件)

│ ├── styles # 一些公用的程序样式文件

│ ├── store # Redux指定，管理数据流的

│ │ ├── createStore.js # 创建和使用redux store

│ │ └── reducers.js # 所有Reducer的注册和注入

│ └── routes # 主路由和异步分割点，也是主要功能模块的代码在这里写

│ ├── index.js # 用store启动主程序路由,注意：这里配置的路由组件名和

│ 浏览器里的url路由没太大关系。

│ └── Welcome # 独立的功能页面组件

│ ├── index.js # 真实的url路由定义和代码异步分割，注入

│ reducer

│ ├── assets # 组件引入的静态图片资源

│ ├── components # 直观React组件和只有该组件需要的css样式文

│ 件.scss文件

│ ├── container # 把主组件和actions、reducer和store进行连接

│ ├── client.js # 用客户端路由时的入口文件

│ ├── server.js # 用服务器端路由时的构造服务端渲染好的完整的html文件。

│ =======================================================================

└── tests # 单元测试

```

记录几个问题：

1. 因为这个项目是用react服务端渲染的，所以是在nodejs作为前端服务器运行的，但是在node运行环境里是没有window全局变量的。所以当在node环境下调用了某些插件，而这些插件里可能使用到了window全局变量时就会报错。

示例：该项目使用了一个react-bootstrap-slider插件，用到了bootstrap-slider.js这个插件，改插件就使用了window对象：所以我尝试着把这个插件源代码里的最后一段if语句注释掉就可以了：/node-modules/bootstrap-slider/dist/bootstrap-slider.js.

