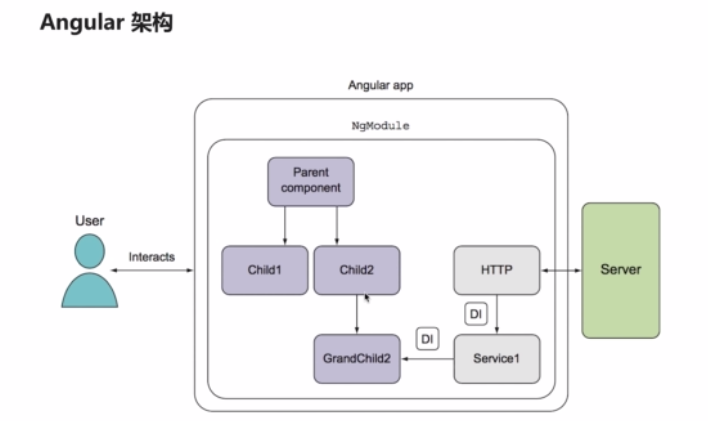
Angular程序架构 构建块

搭建angular开发环境 angular CLI 命令行工具

1. angular程序架构



组件: angular应用的基本构建块,可以理解为一段带有业务逻辑和数据的HTML

服务: 用来封装可重用的业务逻辑

指令: 允许向HTML元素添加自定义行为

模块: 用来将应用中不同的部分组织成一个angular框架可以理解的单元

前三个用于完成功能, 最后一个模块负责打包

1. 搭建angular的环境
2. 安装node.js
3. 安装angular CLI

$ npm install -g angular-cli

安装时间很长,安装结束后可以进行查看

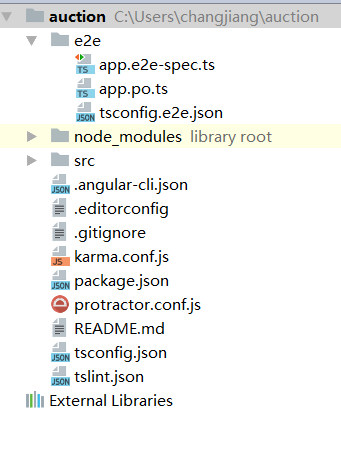
$ ng -v (以后所有的ng命令都是angular的)

1. 安装webstorm

试用期,30天

1. 创建新项目

$ ng new auction (命令格式:ng new 项目名)



目录情况(千万不要更改或者重命名)

e2e: 自动化测试的文件

src: 代码路径

Node\_modules: 第三方的安装包,自己写的不要放在这赫尔

.editorconfig: 编译器的环境

.gitignore: git命令

.angular-cli.json: angular命令行的文件(后期可能会修改)

Karma.conf.js : 自动化测试的配置文件

Package.json : 所需要的工具包 下载后,自动放在

Tslint.json : ts语言检查的wenjian

Src: Assets:资源目录

Environments: 环境配置文件

Index.html ; 入口

Main.ts : js执行入口,启动项目

Polyfills.ts : 第三方库,支持老版本

Styles.ts : 全局样式

app文件目录

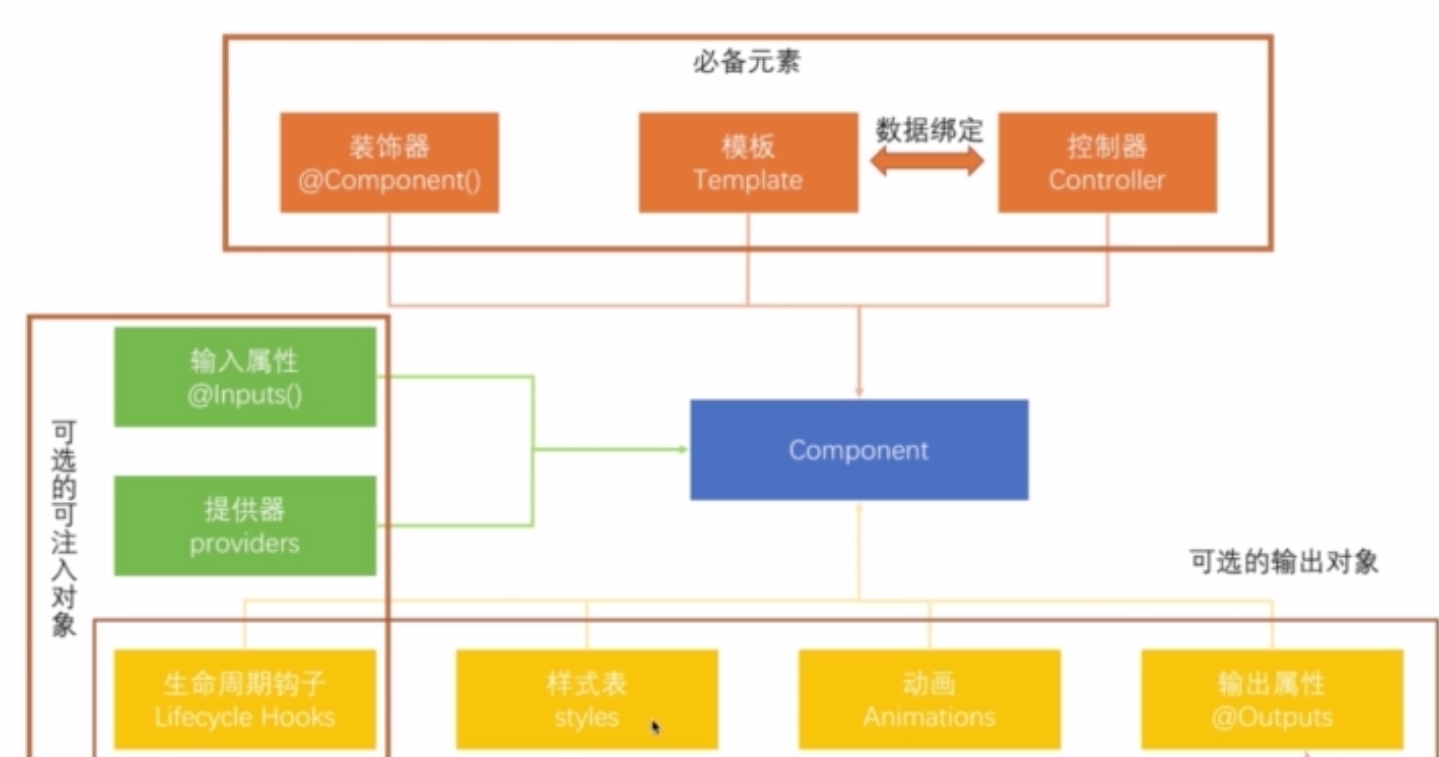
app.conmoent.ts

app.module.ts

装饰器: (组件源数据装饰器)告知angular框架处理类, 根据源数据渲染组件

模板: 渲染组件,

控制器: 包含组件的逻辑



1. 启动angular应用

1.首先明确,启动时加载哪个页面?加载哪个脚本?这些脚本做了什么事?

.angular-cli.json文件



进入main.ts文件

**import** { *enableProdMode* } **from '@angular/core'**;//angular的核心模块  
**import** { platformBrowserDynamic } **from '@angular/platform-browser-dynamic'**;//调用平台种方法  
  
**import** { AppModule } **from './app/app.module'**;//angular的主模块  
**import** { environment } **from './environments/environment'**;  
  
**if** (environment.**production**) {  
 *enableProdMode*();  
}  
  
platformBrowserDynamic().bootstrapModule(AppModule);//angular应用起点

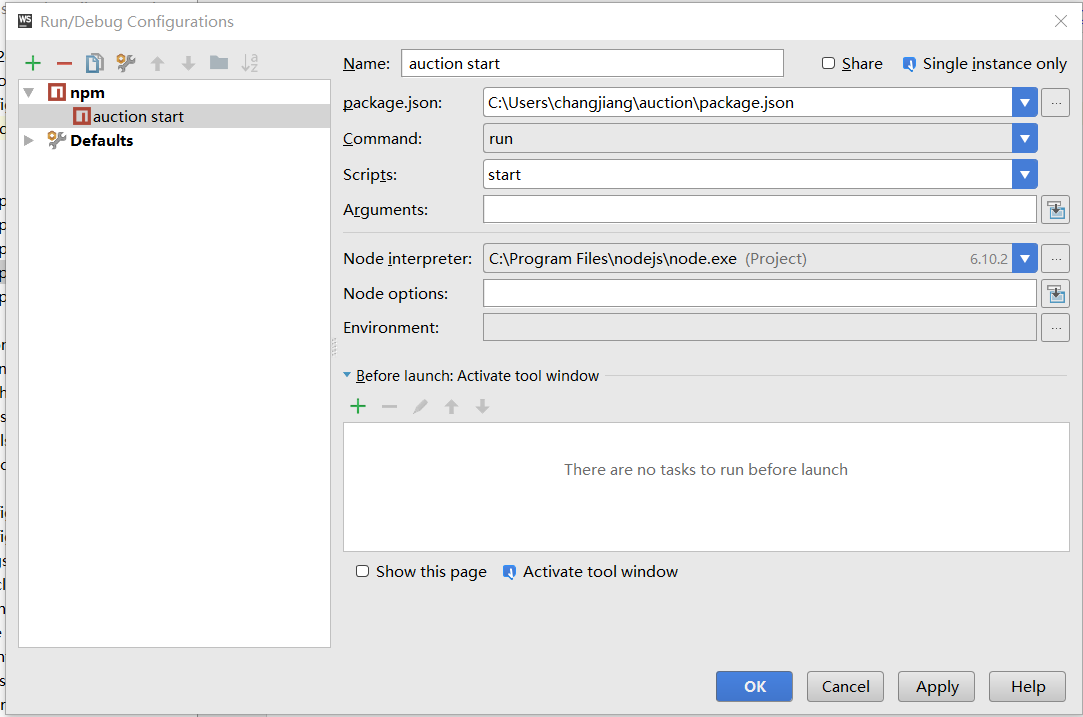
angular会在index.html文件中寻找启动模块指定的主组件对应的css选择器

也就是说寻找启动启动模块AppModule中指定的主组件AppComponent对应的CSS选择器**app-root**

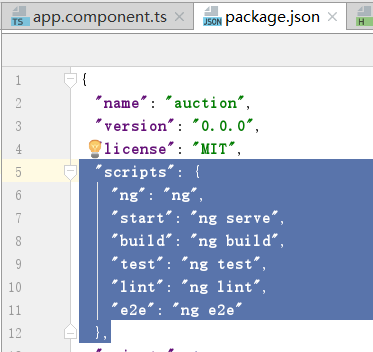
App.module.ts

**import** { BrowserModule } **from '@angular/platform-browser'**;  
**import** { NgModule } **from '@angular/core'**;  
**import** { FormsModule } **from '@angular/forms'**;  
**import** { HttpModule } **from '@angular/http'**;  
  
**import** { AppComponent } **from './app.component'**;  
  
@NgModule({  
 **declarations**: [  
 AppComponent  
 ],  
 **imports**: [  
 BrowserModule,  
 FormsModule,  
 HttpModule  
 ],  
 **providers**: [],  
 **bootstrap**: [AppComponent]  
})  
**export class** AppModule { }

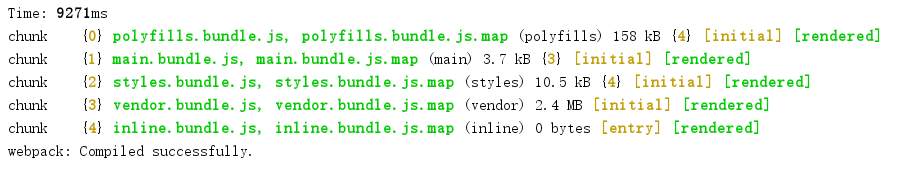
2.创建启动命令



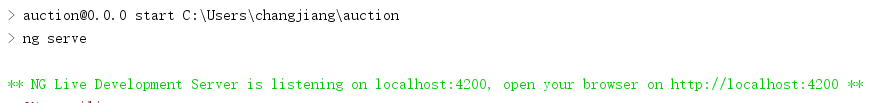
这里添加scripts为start, 这些所有的命令都来自于



执行之后

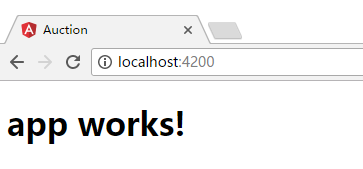


代表成功运行.在上面会出现



此时可以在<http://localhost:4200>进行访问

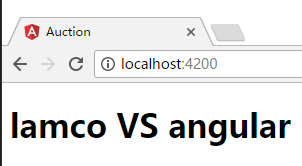
此时浏览器中出现



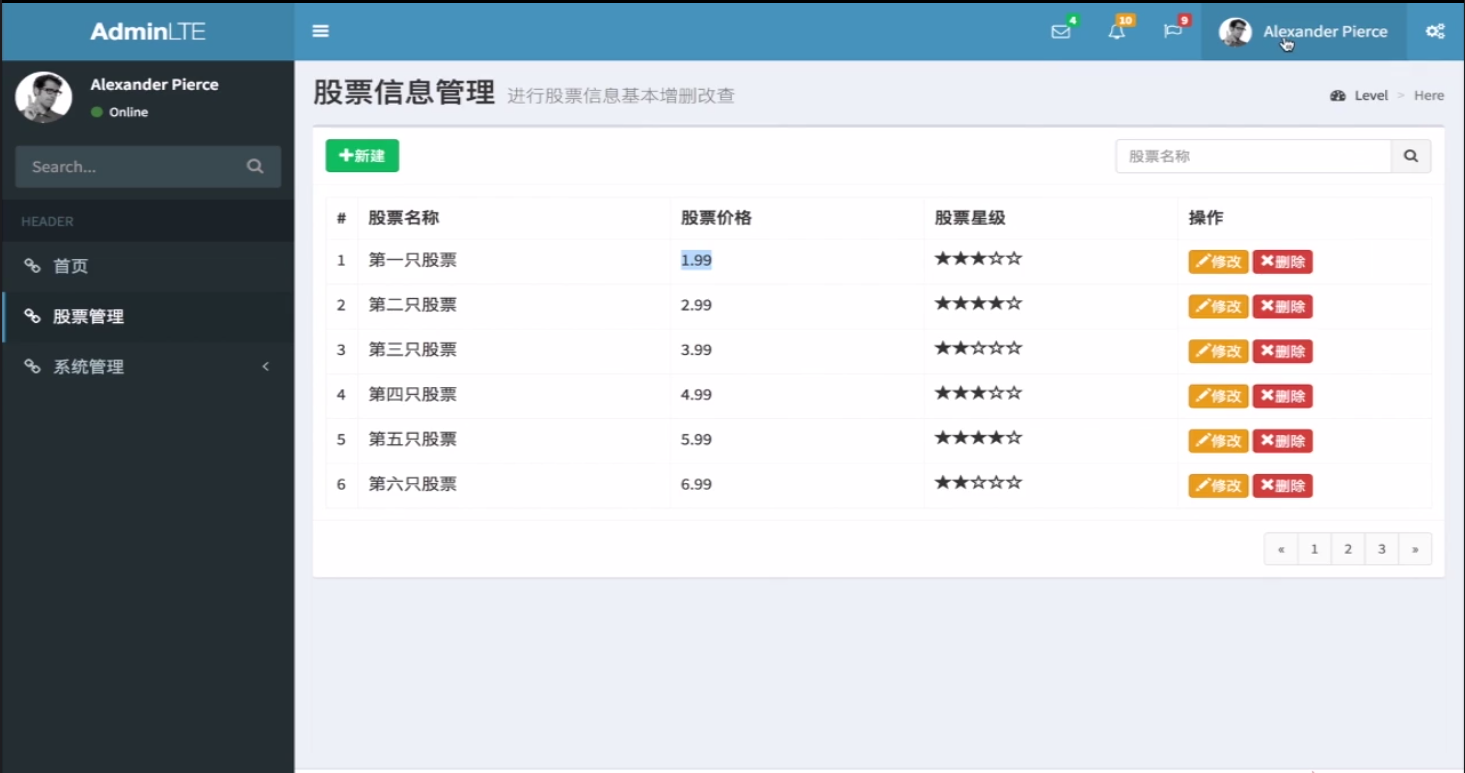
此时这个路径会自动检测src里面的内容

更改**export class** AppComponent {  
 **title** = **'lamco VS angular'**;  
}

页面会自动更新



1. 开发准备

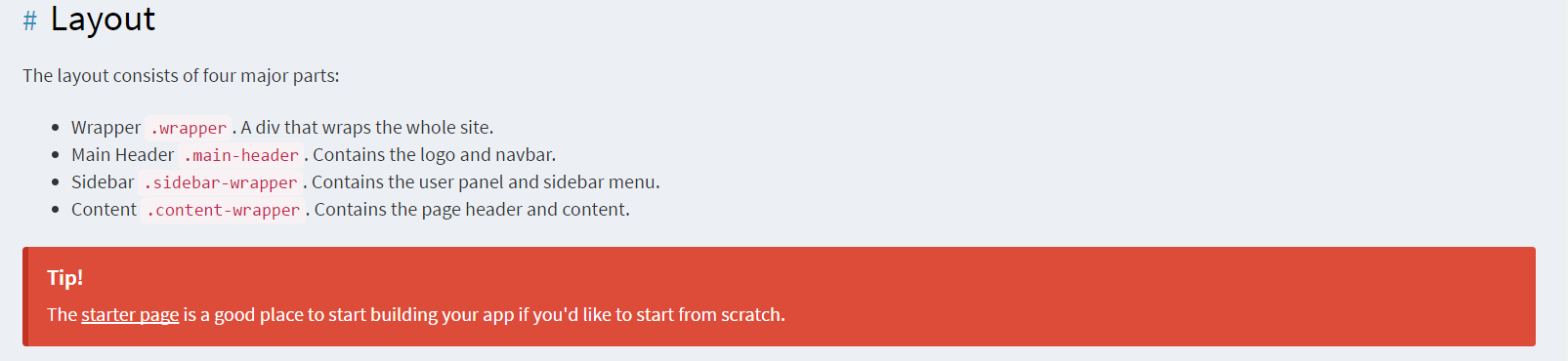


完成此效果, 拆成相应的组件.

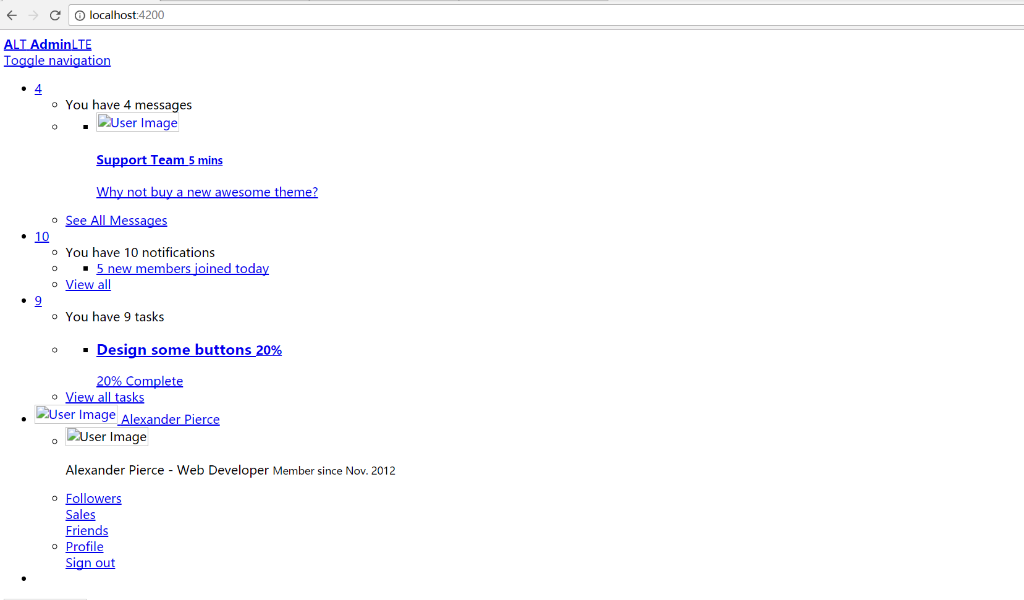
1. 创建新项目 stock

Ng new stock

1. Github找开源项目模板, adminLTE 基于bs的CSS模板
2. 使用命令 npm install --save admin-lte 进行安装
3. 进入<https://adminlte.io/> 进行预览 查看相应的网页内容
4. 在文档块中找到, 进去start page页面



1. 查看项目源码 , 从div起复制代码到 app.conponent.html中

此时页面会变成

1. 接下来添加样式

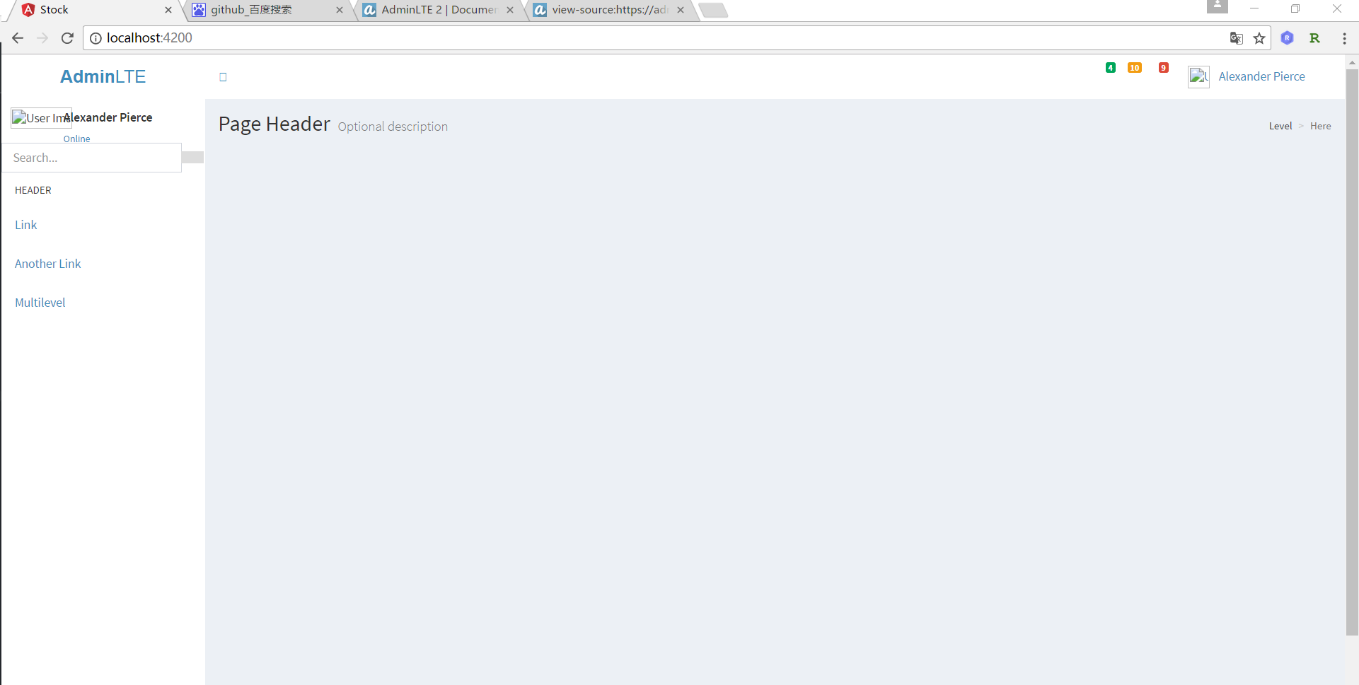
npm install --save bootstrap 安装bootstrap

安装完成之后,.angular-cli.json文件中红styles.css后添加bootstrap的样式

之后添加adminlte样式,添加结束后

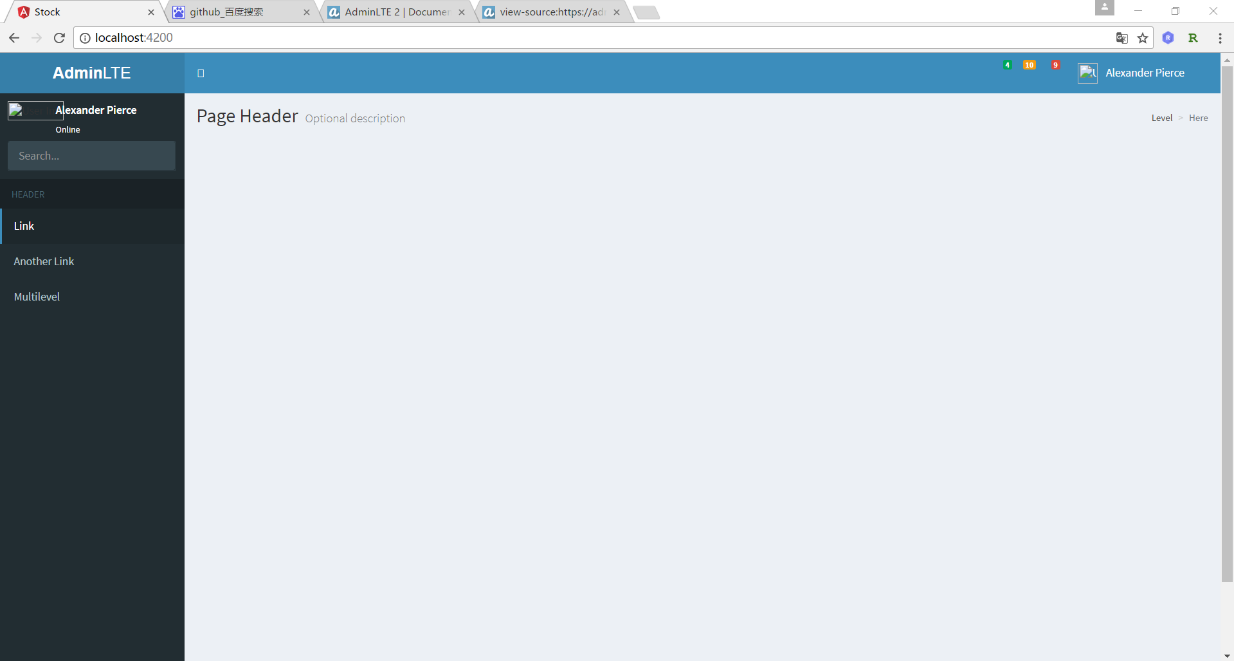
**"styles.css"**,  
 **"../node\_modules/bootstrap/dist/css/bootstrap.min.css"**,  
**"../node\_modules/admin-lte/dist/css/AdminLTE.min.css"**,  
 **"../node\_modules/admin-lte/dist/css/skins/skin-blue.min.css"**

重启服务器 npm run start



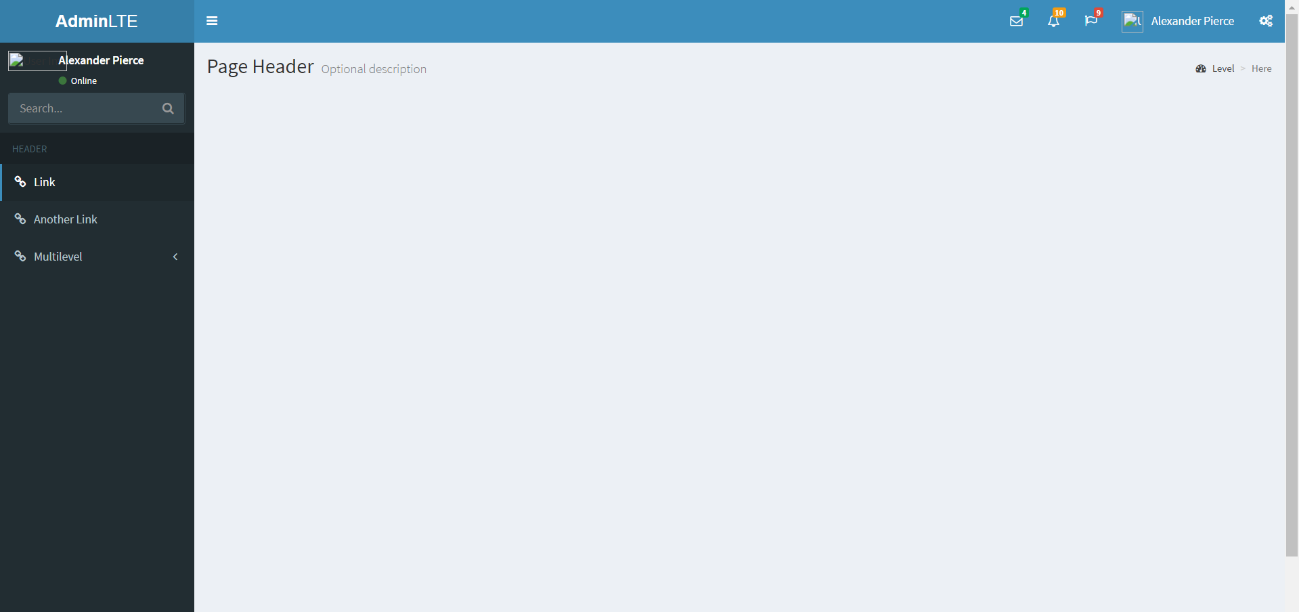
将 class="hold-transition skin-blue sidebar-mini" 放到index.html的body中

此时



图标不显示, 在style.css文件中

**@import "~https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.5.0/css/font-awesome.min.css"**;  
**@import "~https://cdnjs.cloudflare.com/ajax/libs/ionicons/2.0.1/css/ionicons.min.css"**;



1. 接下来引入js, 完成事件

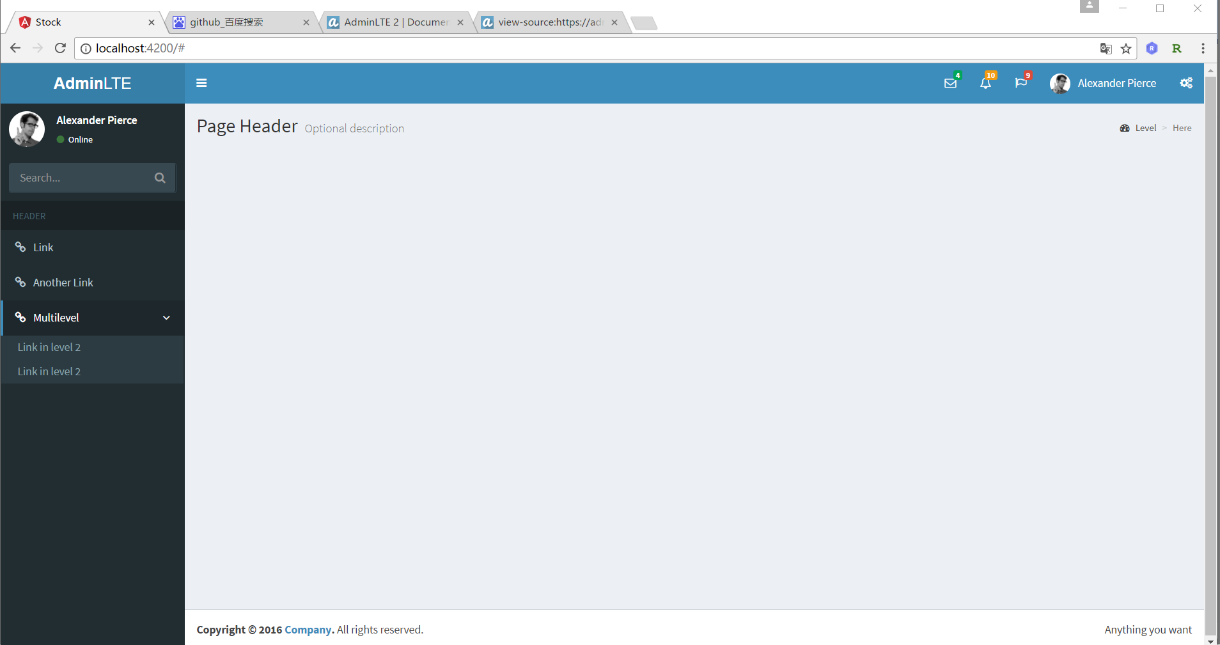
安装jquery npm install --save jquery

引入

**"scripts"**: [  
 **"../node\_modules/jquery/dist/jquery.min.js"**,  
 **"../node\_modules/bootstrap/dist/js/bootstrap.min.js"**,  
 **"../node\_modules/admin-lte/dist/js/app.min.css"**],

1. 解决图片

更改路径



1. 开发组件
2. 头部组件

命令 ng g component header

将app.component.html文件中header部分复制到header. component.html中

改成 <**app-header**></**app-header**>

1. 左侧组件

ng g component menu

把adide中代码移走

1. 右边边栏

ng g component sidebar

1. 页脚

ng g component footer

1. 内容

ng g component content