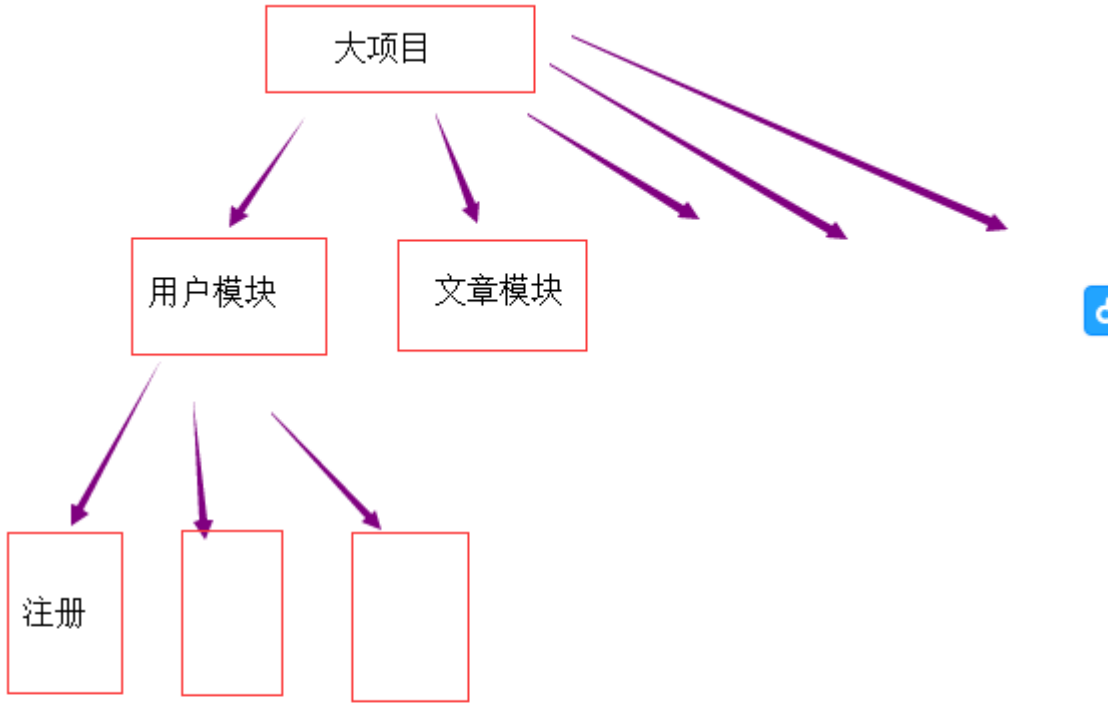


5. 组件化开发

5.1 组件[component]

组件（Component）是自定义封装的功能。在前端开发过程中，经常出现多个网页的功能是重复的，而且很多不同的网站之间，也存在同样的功能。

而在网页中实现一个功能，需要使用html定义功能的内容结构，使用css声明功能的外观样式，还要使用js来定义功能的特效，因此就产生了把一个功能相关的[HTML、css和javascript]代码封装在一起组成一个整体的代码块封装模式，我们称之为“组件”。

所以，组件就是一个html网页中的功能，一般就是一个标签，标签中有自己的内容结构，样式和特效。

这样，前端人员就可以在开发时，只需要书写一次代码，随处引入即可使用。

组件有两种：默认组件[全局组件] 和 单文件组件

5.1.1 默认组件

```
<div id="app">
    <addnum></addnum>
    <addnum></addnum>
    <addnum></addnum>
    <addnum></addnum>
    <addnum></addnum>
</div>
<script>
    Vue.component("addnum",{
        template:'<div><input type="text" v-model="num"><button @click="num+=1">点击
</button></div>',
        data: function(){
            // 写在这里的数据只有当前组件可以使用
            return {
                num:1,
            }
        }
    });


    var vm = new Vue({
        el:"#app",
        // 这里写的数据是全局公用的，整个文件共享
        data:{

        }
    })
</script>
```

6. Vue自动化工具（Vue-CLI）

前面学习了普通组件以后，接下来我们继续学习单文件组件则需要提前先安装准备一些组件开发工具。否则无法使用和学习单文件组件。

一般情况下，单文件组件，我们运行在 自动化工具vue-CLI中，可以帮我们编译单文件组件。所以我们需要在系统中先搭建vue-CLI工具，

官网：https://cli.vuejs.org/zh/

Vue CLI 需要 Node.js 8.9 或更高版本 (推荐 8.11.0+)。你可以使用 nvm 或 nvm-windows在同一台电脑中管理多个 Node 版本。

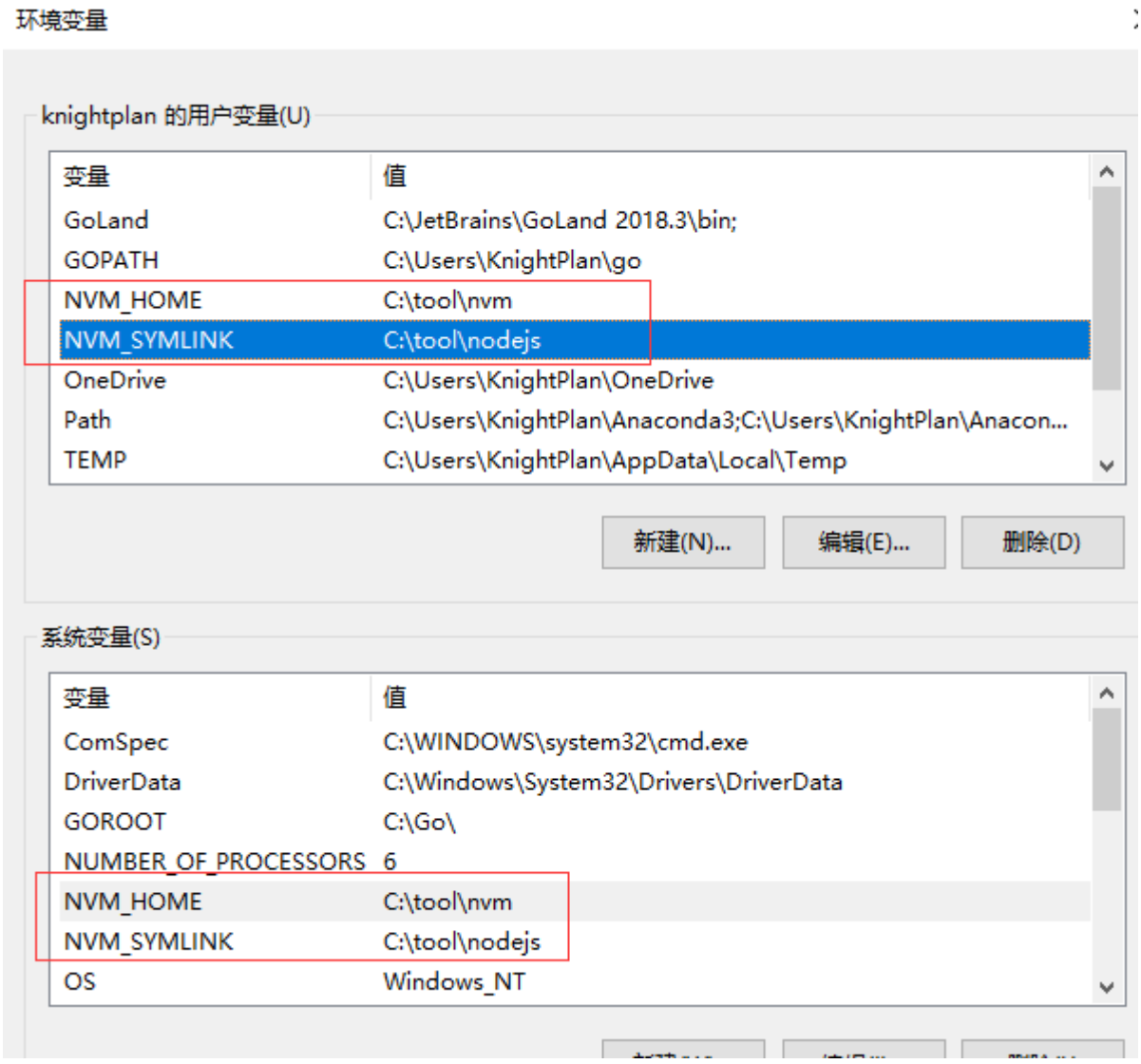
nvm工具的下载和安装： https://www.jianshu.com/p/d0e0935b150a

https://www.jianshu.com/p/622ad36ee020

安装记录:

打开:https://github.com/coreybutler/nvm-windows/releases

安装完成以后,先查看环境变量是否设置好了.



常用的nvm命令

nvm list # 列出目前在nvm里面安装的所有node版本 nvm install node版本号 # 安装指定版本的node.js nvm uninstall node版本号 # 卸载指定版本的node.js nvm use node版本号 # 切换当前使用的node.js版本

如果使用nvm工具，则直接可以不用自己手动下载，如果使用nvm下载安装 node的npm比较慢的时候，可以修改nvm的配置文件(在安装根目录下)

```
# settings.txt
root: C:\tool\nvm    [这里的目录地址是安装nvm时自己设置的地址,要根据实际修改]
path: C:\tool\nodejs
arch: 64
proxy: none
node_mirror: http://npm.taobao.org/mirrors/node/
npm_mirror: https://npm.taobao.org/mirrors/npm/
```

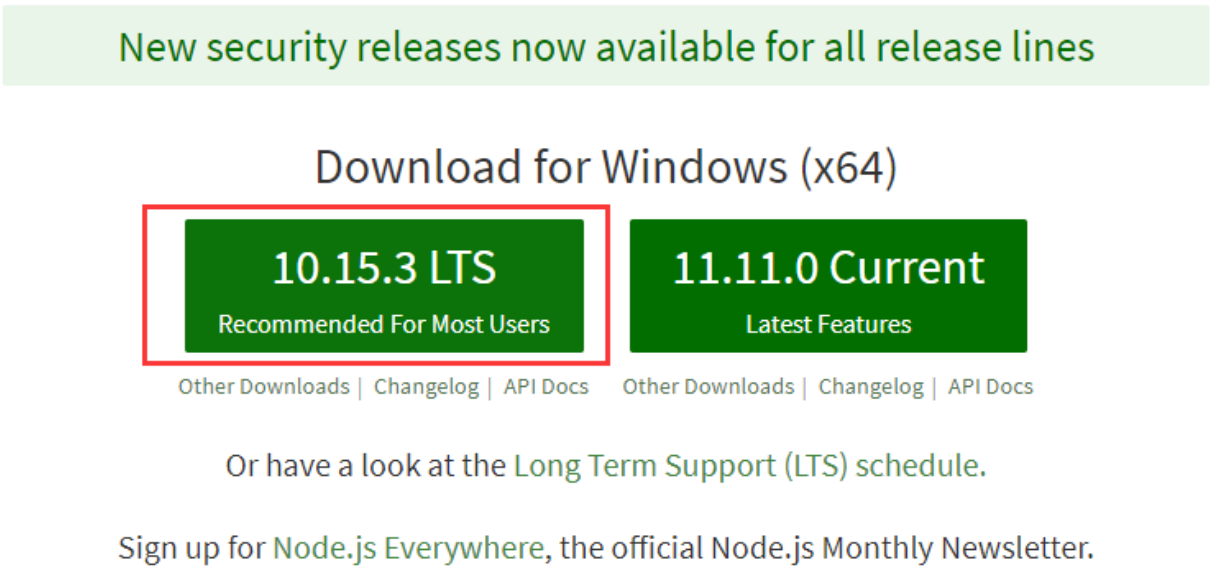
6.1 安装node.js

Node.js是一个新的后端(后台)语言，它的语法和JavaScript类似，所以可以说它是属于前端的后端语言，后端语言和前端语言的区别：

- 运行环境：后端语言一般运行在服务器端，前端语言运行在客户端的浏览器上
- 功能：后端语言可以操作文件，可以读写数据库，前端语言不能操作文件，不能读写数据库。

我们一般安装LTS(长线支持版本)：

下载地址：https://nodejs.org/en/download/【上面已经安装了nvm，那么这里不用手动安装了】



node.js的版本有两大分支：

官方发布的node.js版本：0.xx.xx 这种版本号就是官方发布的版本

社区发布的node.js版本：xx.xx.x 就是社区开发的版本

Node.js如果安装成功，可以查看Node.js的版本,在终端输入如下命令：

```
node -v
```

6.2 npm

在安装node.js完成后，在node.js中会同时帮我们安装一个npm包管理器npm。我们可以借助npm命令来安装node.js的包。这个工具相当于python的pip管理器。

```
npm install -g 包名                 # 安装模块    -g表示全局安装，如果没有-g，则表示在当前项目安装
npm list                           # 查看当前目录下已安装的node包
npm view 包名 engines               # 查看包所依赖的Node的版本
npm outdated                       # 检查包是否已经过时，命令会列出所有已过时的包
npm update 包名                     # 更新node包
npm uninstall 包名                  # 卸载node包
npm 命令 -h                         # 查看指定命令的帮助文档
```

6.3 安装Vue-CLI

```
npm install -g vue-cli
```

如果安装速度过慢，一直超时，可以考虑切换npm镜像源：http://npm.taobao.org/

6.4 使用Vue-CLI初始化创建项目

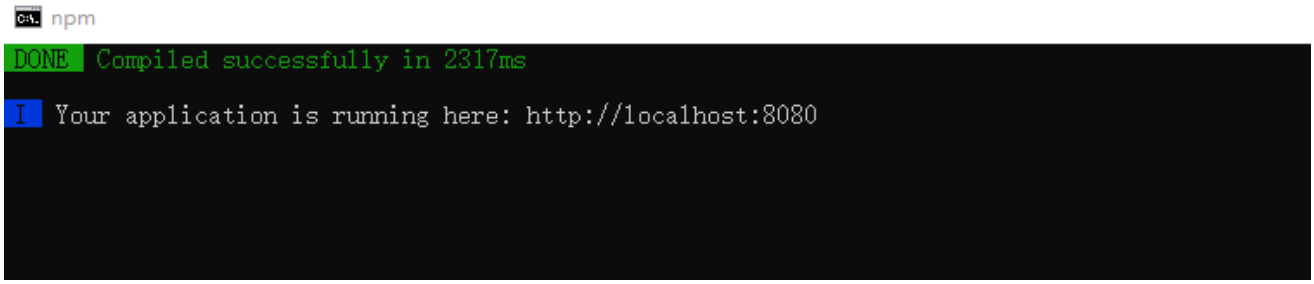
6.4.1 生成项目目录

使用vue自动化工具可以快速搭建单页应用项目目录。

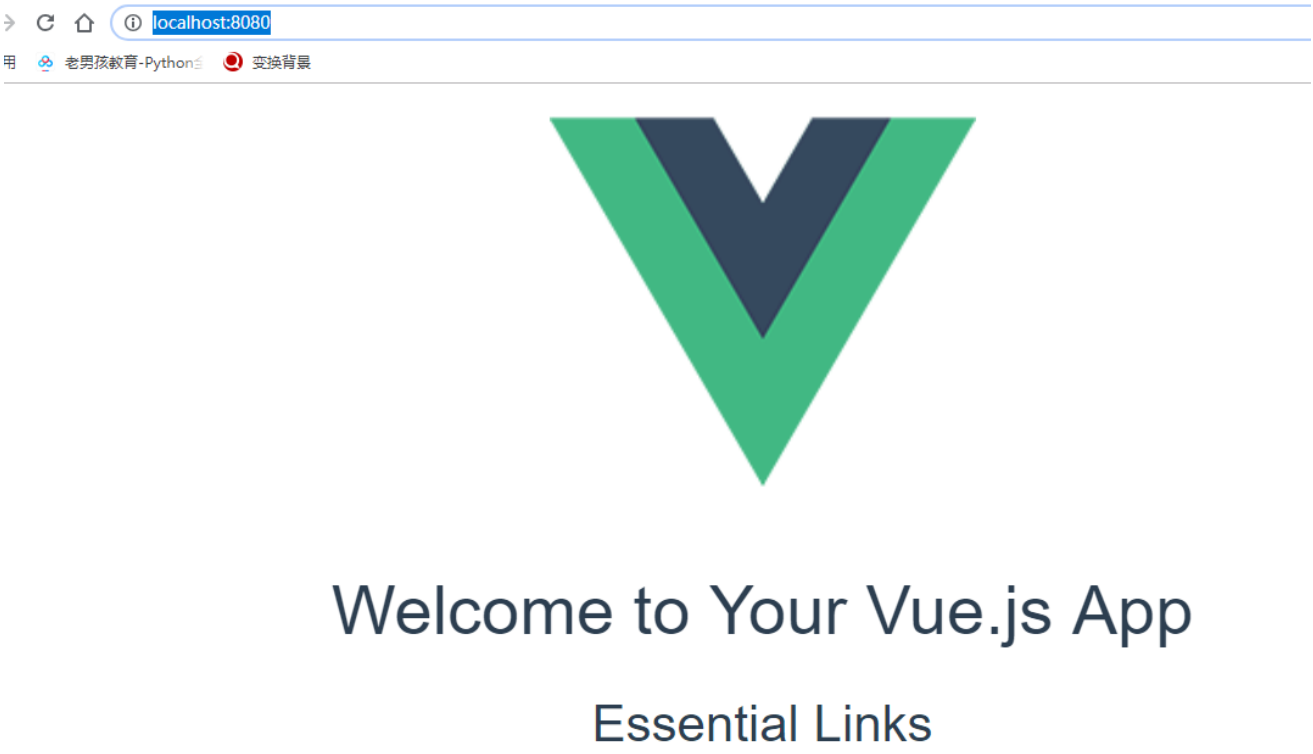
该工具为现代化的前端开发工作流提供了开箱即用的构建配置。只需几分钟即可创建并启动一个带热重载、保存时静态检查以及可用于生产环境的构建配置的项目：

```
// 生成一个基于 webpack 模板的新项目
vue init webpack 项目名
例如:
vue init webpack myproject


// 启动开发服务器 ctrl+c 停止服务
cd myproject
npm run dev            # 运行这个命令就可以启动node提供的测试http服务器
```

运行了上面代码以后,终端下会出现以下效果提示：



那么访问：http://localhost:8080/



6.4.2 项目目录结构

src 主开发目录，要开发的单文件组件全部在这个目录下的components目录下

static 静态资源目录，所有的css，js文件放在这个文件夹

dist项目打包发布文件夹，最后要上线单文件项目文件都在这个文件夹中[后面打包项目,让项目中的vue组件经过编译变成js 代码以后,dist就出现了]

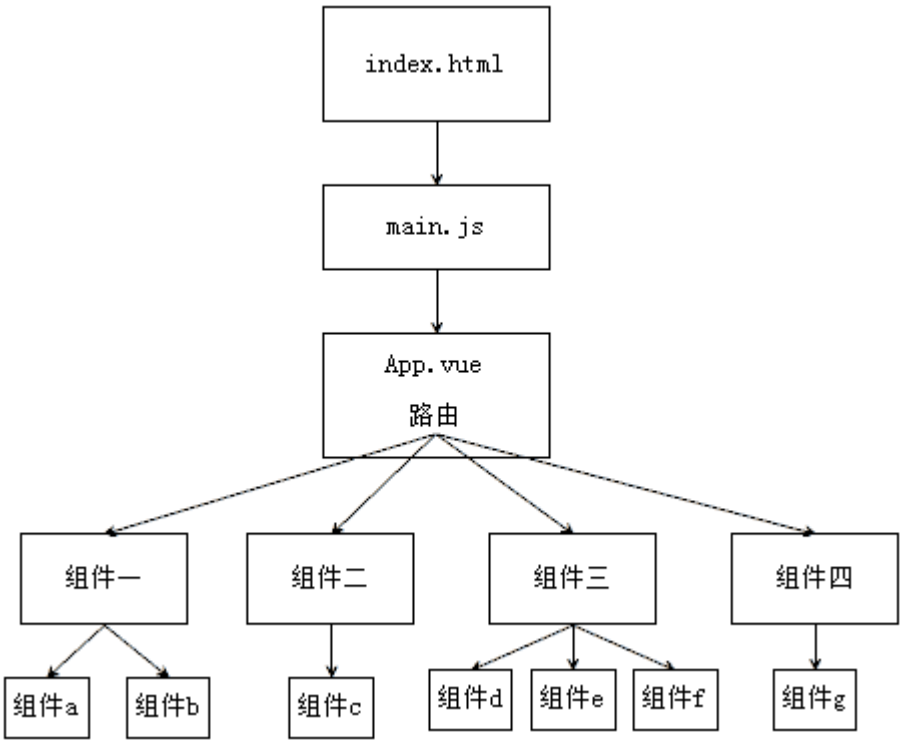
node_modules目录是node的包目录，

config是配置目录，

build是项目打包时依赖的目录

src/router 路由,后面需要我们在使用Router路由的时候,自己声明.

6.4.3 项目执行流程图



整个项目是一个主文件index.html,index.html中会引入src文件夹中的main.js,main.js中会导入顶级单文件组件App.vue,App.vue中会通过组件嵌套或者路由来引用components文件夹中的其他单文件组件。

7. 单文件组件的使用

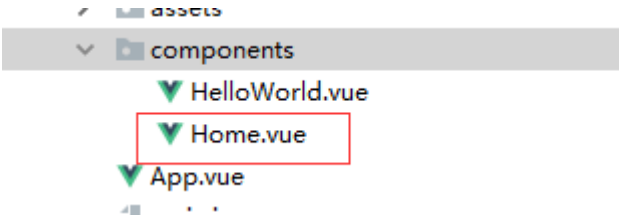
组件有两种：普通组件、单文件组件

普通组件的缺点：

1. html代码是作为js的字符串进行编写，所以组装和开发的时候不易理解，而且没有高亮效果。
2. 普通组件用在小项目中非常合适，但是复杂的大项目中，如果把更多的组件放在html文件中，那么维护成本就会变得非常昂贵。
3. 普通组件只是整合了js和html，但是css代码被剥离出去了。使用的时候的时候不好处理。

将一个组件相关的html结构，css样式，以及交互的JavaScript代码从html文件中剥离出来，合成一个文件，这种文件就是单文件组件，相当于一个组件具有了结构、表现和行为的完整功能，方便组件之间随意组合以及组件的重用，这种文件的扩展名为“.vue”，比如："Home.vue"。

1. **创建组件**



在组件中编辑三个标签，编写视图、vm对象和css样式代码。

7.1 template 编写html代码的地方

```
<template>
  <div id="Home">
    <span @click="num--" class="sub">-</span>
    <input type="text" size="1" v-model="num">
    <span @click="num++" class="add">+</span>
  </div>
</template>
```

7.2 script编写vue.js代码

```
<script>
  export default {
    name:"Home",
    data: function(){
      return {
        num:0,
      }
    }
  }
</script>
```

7.3 style编写当前组件的样式代码

```
<style scoped>
  .sub,.add{
    border: 1px solid red;
    padding: 4px 7px;
  }
</style>
```

7.4 完成案例-点击加减数字

创建Homes.vue

```
<template>
  <div class="add_num">
      <span @click="num++">+</span>
      <input type="text" size="2" v-model="num">
      <span @click="num--">-</span>
  </div>
</template>

<script>
  export default{
    name:"AddNum",
    data: function(){
      return {
        num: 0,
```

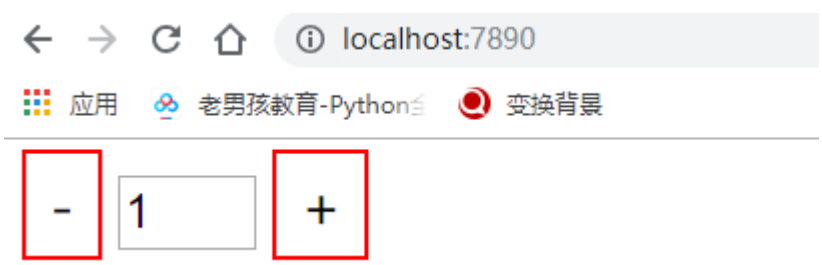
```
      }
    }
  }
</script>

<style scoped>
   .add_num{
      font-size: 32px;
   }
</style>
```

在App.vue组件中调用上面的组件

```
<template>
  <div id="Home">
    <span @click="num--" class="sub">-</span>
    <input type="text" size="1" v-model="num">
    <span @click="num++" class="add">+</span>
  </div>
</template>


<script>
  export default {
    name:"Home",
    data: function(){
      return {
        num:0,
      }
    }
  }
</script>

<style scoped>
  .sub,.add{
    border: 1px solid red;
    padding: 4px 7px;
  }
</style>
```

在开发vue项目之前，需要手动把 App.vue的HelloWorld组件代码以及默认的css样式，清楚。

上面的代码效果：



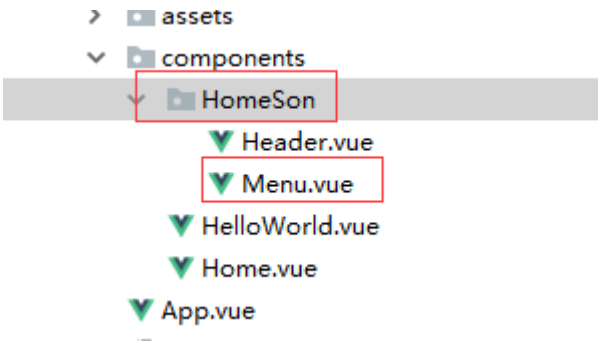
7.4 组件的嵌套

有时候开发vue项目时,页面也可以算是一个大组件,同时页面也可以分成多个子组件.

因为,产生了父组件调用子组件的情况.

例如,我们可以声明一个组件,作为父组件

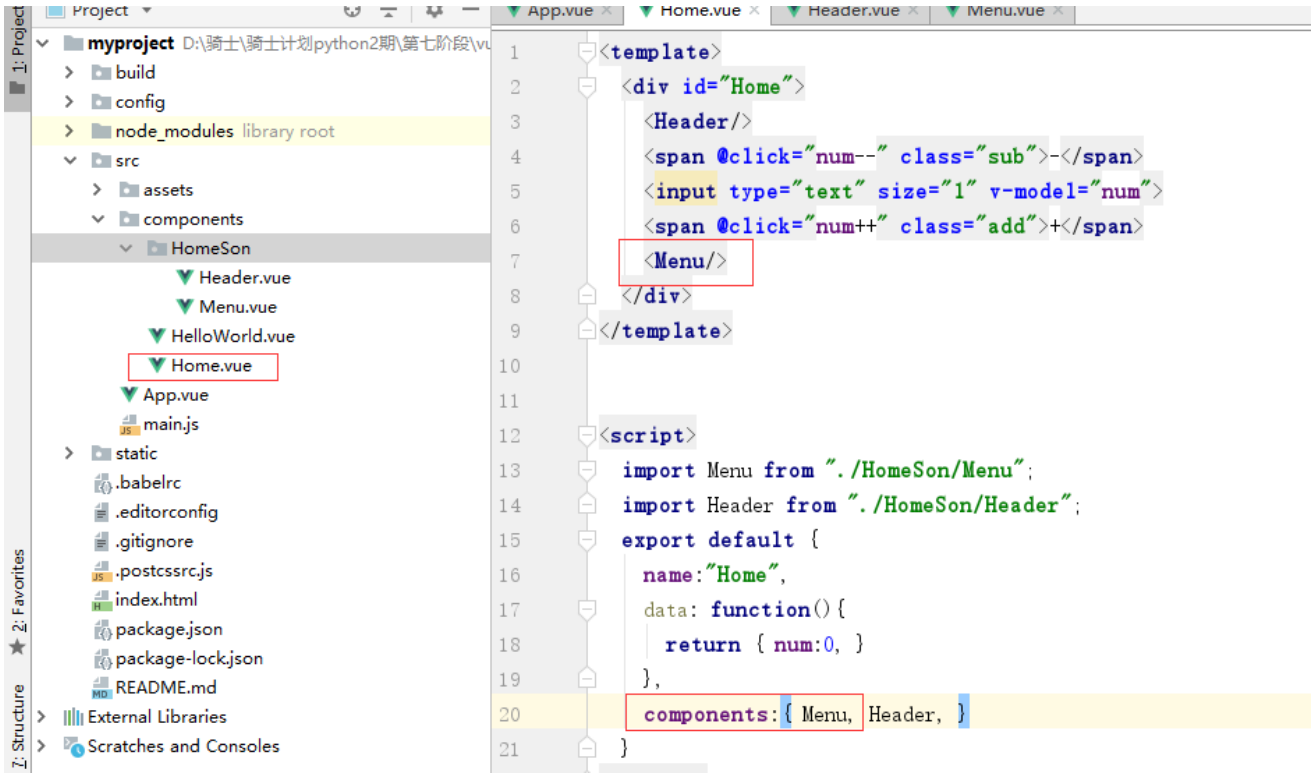
在components/创建一个保存子组件的目录HomeSon



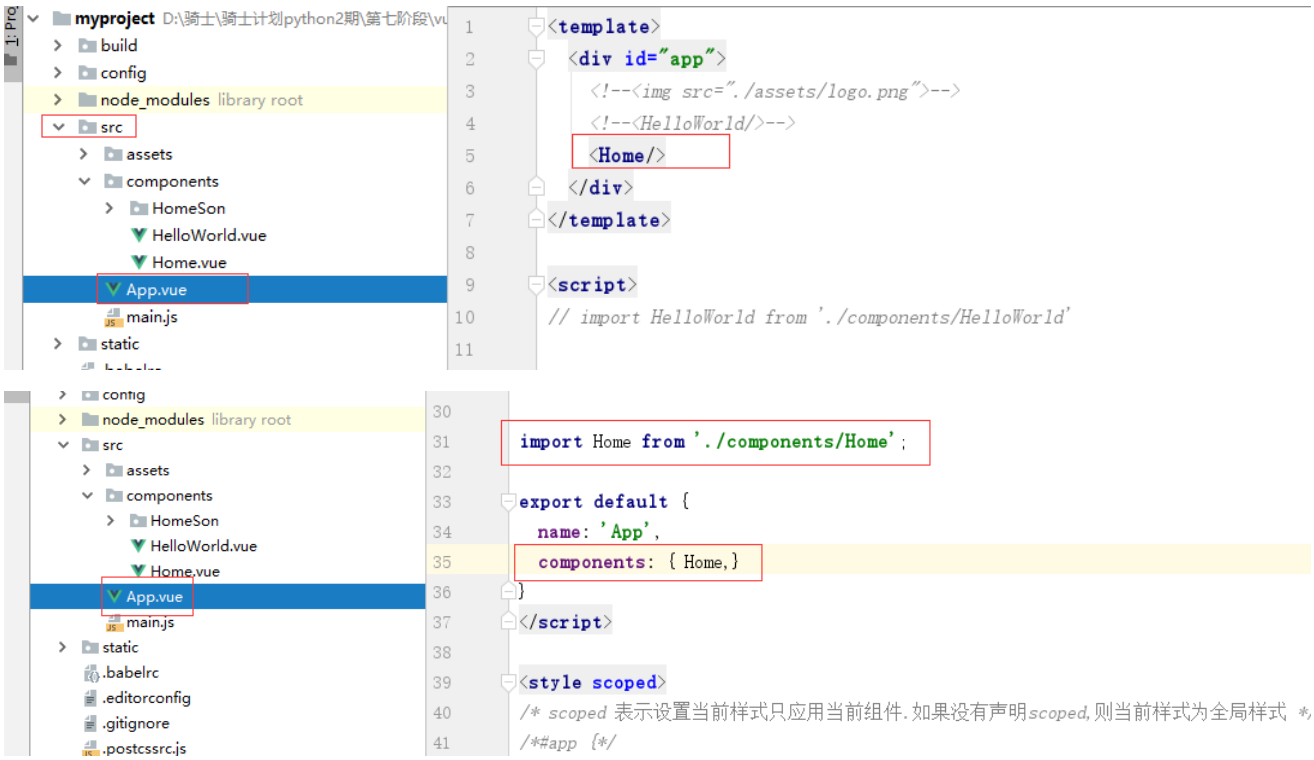
在HomeSon目录下,可以创建当前页面的子组件,例如,是Menu.vue

```
//  组件中代码必须写在同一个标签中
<template>
    <div id="menu">
      <span>{{msg}}</span>
      <div>hello</div>
  </div>
</template>

<script>
  export default {
    name:"Menu",
    data: function(){
      return {
        msg:"这是Menu组件里面的菜单",
      }
    }
  }
</script>
```

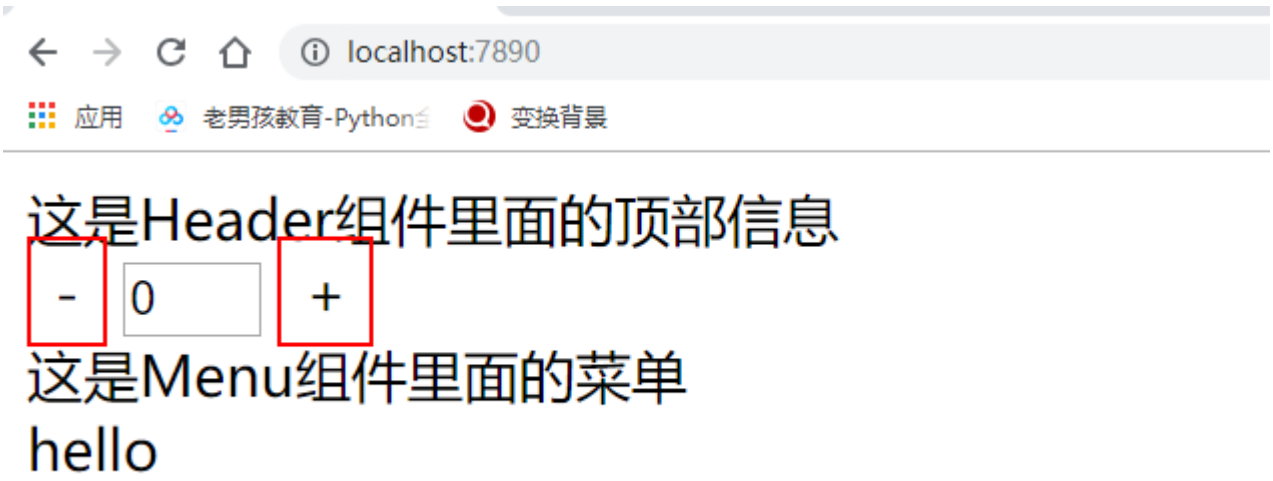
然后，在父组件中调用上面声明的子组件。



最后,父组件被App.vue调用.就可以看到页面效果.



效果:



7.5 传递数据

例如,我们希望把父组件的数据传递给子组件.

可以通过props属性来进行传递.

传递数据三个步骤:

1. 在父组件中，调用子组件的组名处，使用属性值的方式往下传递数据

```
<Menu :mynum="num" title="home里面写的数据"/>

# 上面表示在父组件调用Menu子组件的时候传递了2个数据:
  如果要传递变量[变量可以各种类型的数据]，属性名左边必须加上冒号:，同时，属性名是自定义的，会在子组件中使用。
  如果要传递普通字符串数据，则不需要加上冒号:
```

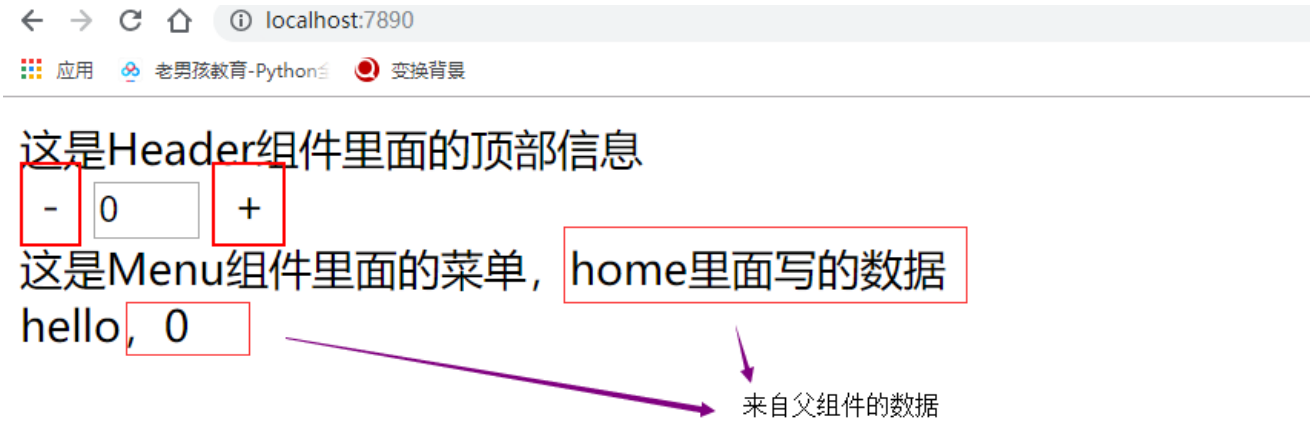
2. 在子组件中接受上面父组件传递的数据，需要在vm组件对象中，使用props属性类接受。

```
<script>
  export default {
    name:"Menu",
    props:["mynum","title"],
    data: function(){
      return {
        msg:"这是Menu组件里面的菜单",
      }
    }
  }
</script>

// 上面 props属性中表示接受了两个数据。
```

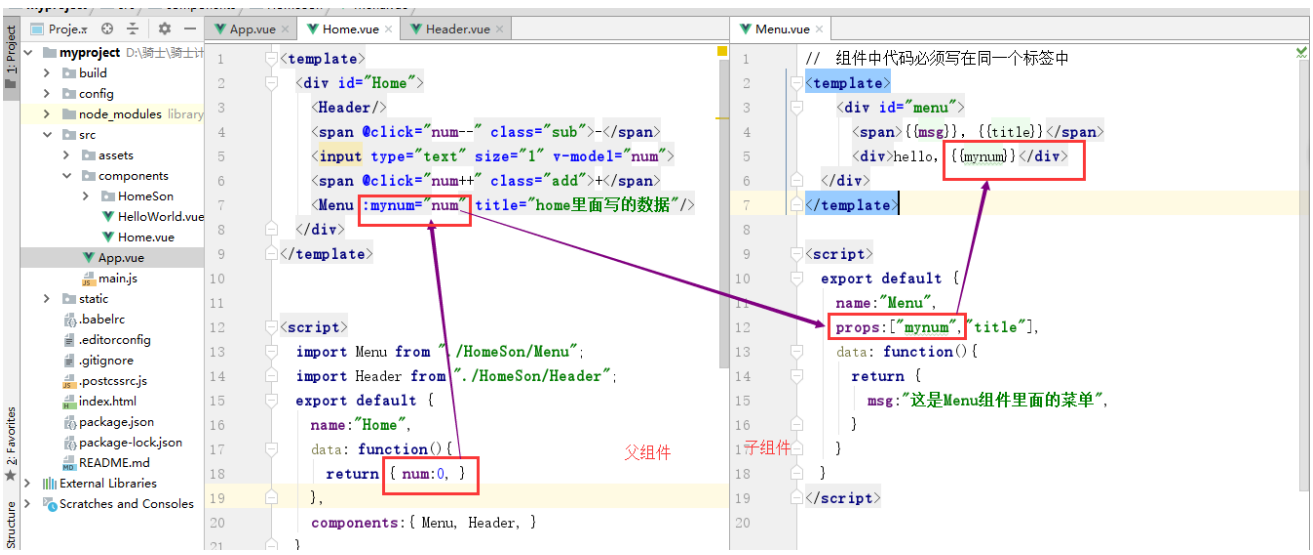
3. 在子组件中的template中使用父组件传递过来的数据.

```
<template>
    <div id="menu">
      <span>{{msg}}, {{title}}</span>
      <div>hello, {{mynum}}</div>
  </div>
</template>
```

效果:



步骤流程:



使用父组件传递数据给子组件时, 注意一下几点:

1. 传递数据是变量,则需要在属性左边添加冒号.

   传递数据是变量,这种数据称之为"动态数据传递"

   传递数据不是变量,这种数据称之为"静态数据传递"

2. 父组件中修改了数据,在子组件中会被同步修改,但是,子组件中的数据修改了,是不是影响到父组件中的数据.

   这种情况,在开发时,也被称为"单向数据流"

8. 在组件中使用axios获取数据

默认情况下，我们的项目中并没有对axios包的支持，所以我们需要下载安装。

在项目根目录中使用 npm安装包

```
npm install axios
```

接着在main.js文件中，导入axios并把axios对象 挂载到vue属性中多为一个子对象，这样我们才能在组件中使用。

```
// The Vue build version to load with the `import` command
// (runtime-only or standalone) has been set in webpack.base.conf with an alias.
import Vue from 'vue'
import App from './App' // 这里表示从别的目录下导入 单文件组件
import axios from 'axios'; // 从node_modules目录中导入包
Vue.config.productionTip = false

Vue.prototype.$axios = axios; // 把对象挂载vue中

/* eslint-disable no-new */
new Vue({
  el: '#app',
  components: { App },
  template: '<App/>'
});
```

8.1 在组建中使用axios获取数据

```
<script>
  export default{
    。。。
    methods:{
      get_data:function(){
         // 使用axios请求数据
        this.$axios.get("http://wthrcdn.etouch.cn/weather_mini?city=深
圳").then((response)=>{
            console.log(response);
        }).catch(error=>{
            console.log(error);
        })
      }
    }
```

```
    }
</script>
```

效果:

使用的时候，因为本质上来说，我们还是原来的axios，所以也会收到同源策略的影响。

9. 路由（Router）

安装:

```
npm install --save vue-router
```

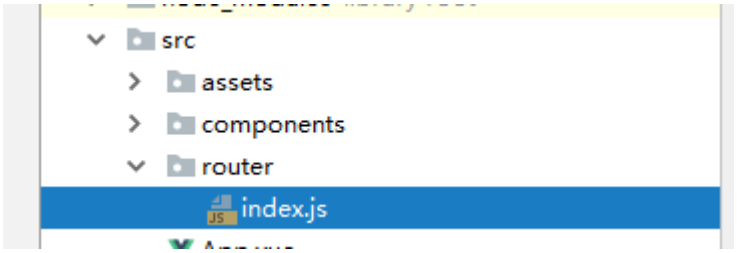
路由：把组件和对应的 uri地址进行一一映射的关系。

9.1 创建路由文件

路由文件可以直接创建在src目录下，但是如果项目大了， 分成多个不同的大平台或者大的子项目，可以选择分目录保存路由，

src/router/index.js // 前台路由

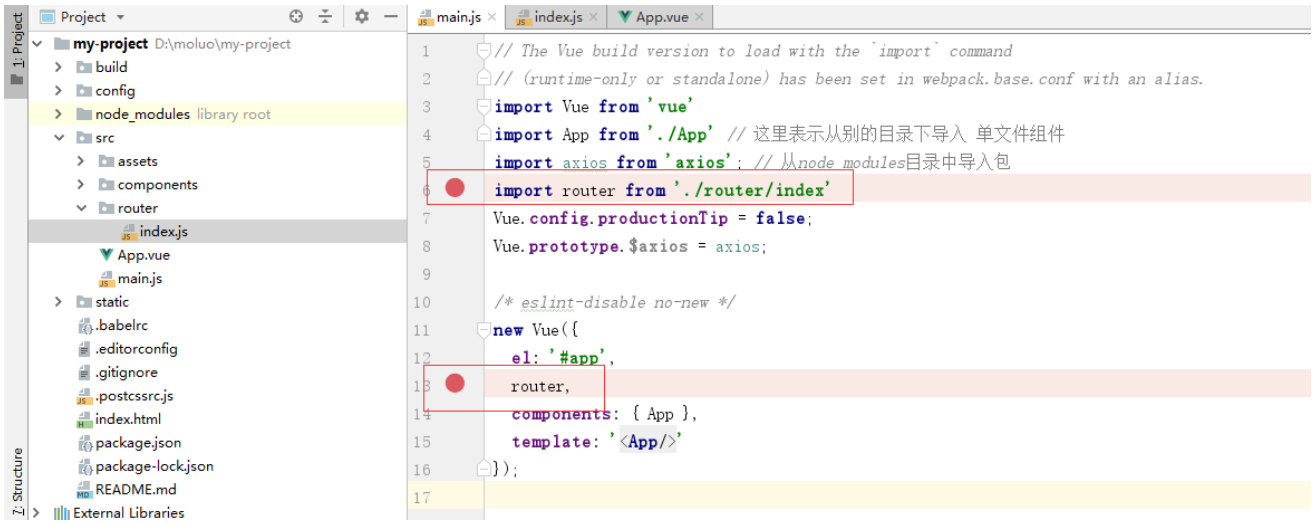
src/router/backend.js // 后台路由



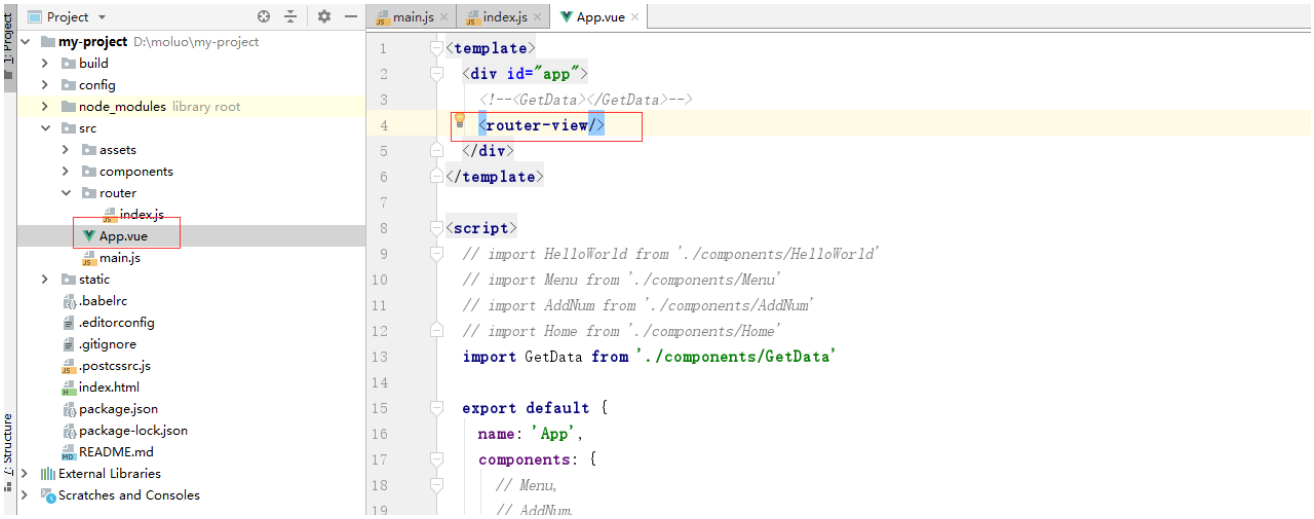
在index.js文件中编写初始化路由信息,以及绑定地址和组件的映射关系

```
import Vue from 'vue'
import Router from 'vue-router'

// 导入路由中需要使用的组件
import Home from '@/components/Home'
import AddNum from '@/components/AddNum'
import HelloWorld from '@/components/HelloWorld'

Vue.use(Router);

export default new Router({
  routes:[
    // 路由信息
    {
      path:"/",
      name: "Home",
      component: Home
    },
    {
      path:"/num",
      name: "AddNum",
      component: AddNum
    },
    {
      path:"/hw",
      name: "HelloWorld",
      component: HelloWorld
    },
  ]
})
```

index.js路由信息要被main.js加载,所以需要在src/main.js中导入路由对象



在main.js中的Vue对象中注册了路由以后,那么直接在`App.vue文件中的html代码里面,显示当前uri路径对应的组件内容.

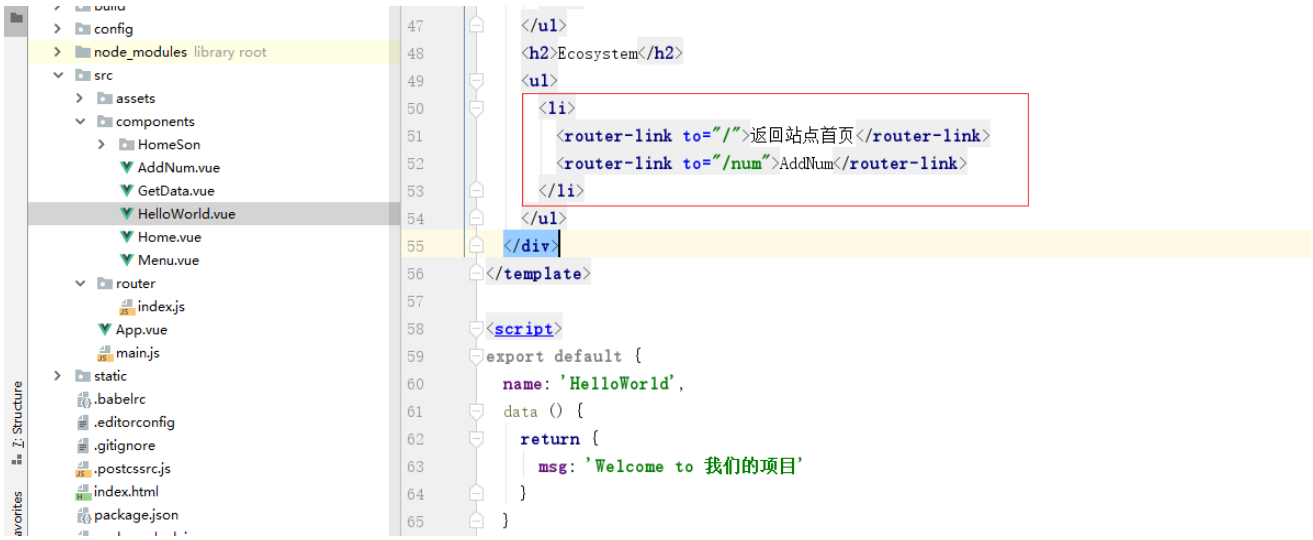


实现生成站内连接.可以使用vue-router提供的路由标签也可以使用vue-router提供的this.$router

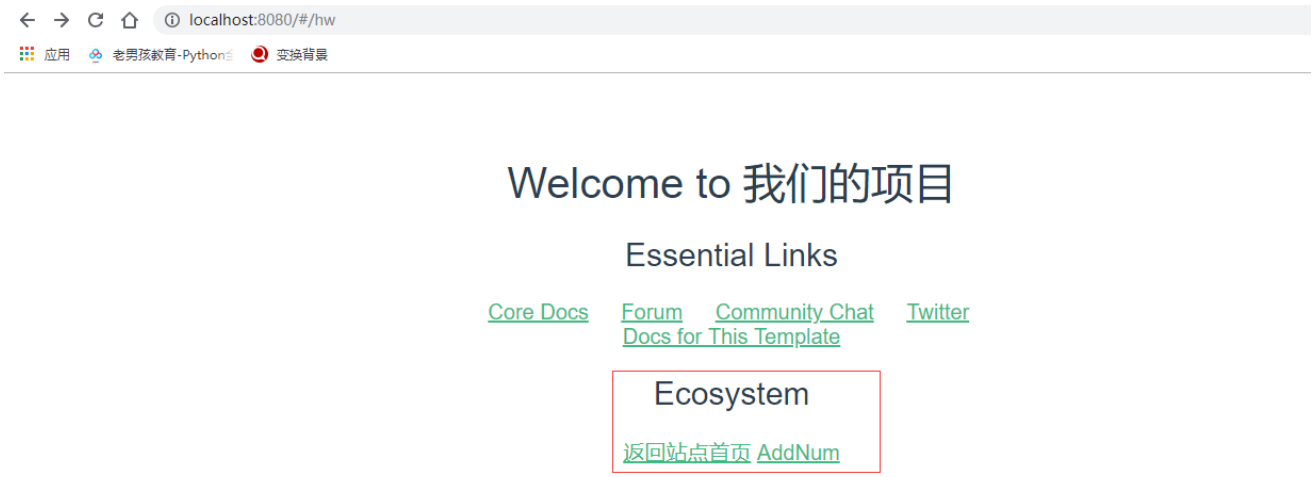
通过链接可以切换路由标签里面对应的组件，链接的地址是上面index.js文件中定义的path值，不过链接标签是"router-link",链接地址用'to'来定义：

```
<router-link to="/">站点首页</router-link>
<router-link to="/num">AddNum</router-link>
```

代码编写:



效果:



链接地址中可以传递参数，格式如下：

```
// name对应的是路由中定义的一个path对应的name属性
    <router-link :to='{name:"UpDate",params:{code:item.code}}'>
```

有时候需要在组件的js中跳转页面，也就是改变路由，改变路由有下面这些方式：

```
// 当前页面重新加载
this.$router.go('/user');

// 跳转到另外一个路由
this.$router.push({path:'/user'});

// 获取当前的路由地址
var sPath = this.$route.path;
```