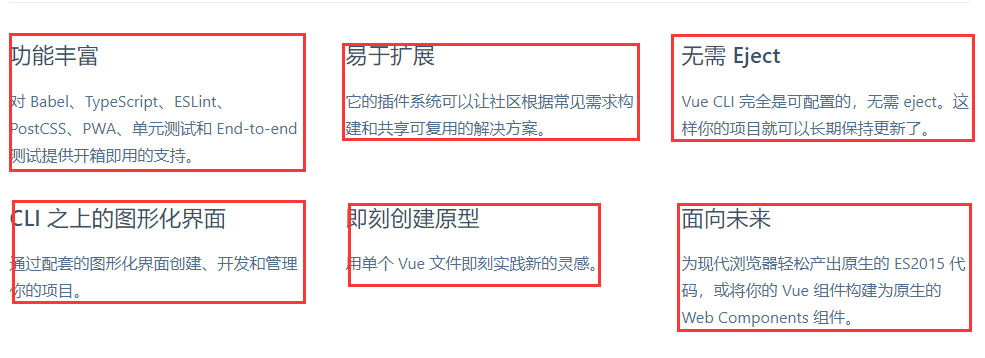
# Vue-cli脚手架工具

官网：<https://cli.vuejs.org/zh/>



特点：



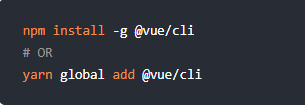
为了更加地方便我们去开发vue的应用程序，vue它提供一套工具，你可以通过vue-cli去快速地构造你的项目结构。

## Vue-cli

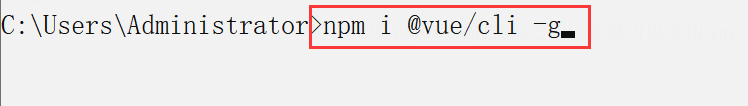
Vue.js 提供一个官方命令行工具，可用于快速搭建大型**单页（spa）应用**。该工具提供开箱即用的构建工具配置，带来现代化的前端开发流程。只需几分钟即可创建并启动一个**带热重载**、**保存时静态检查**以及可用于生产环境的构建配置的项目。

## 步骤

### 全局安装 @vue/cli

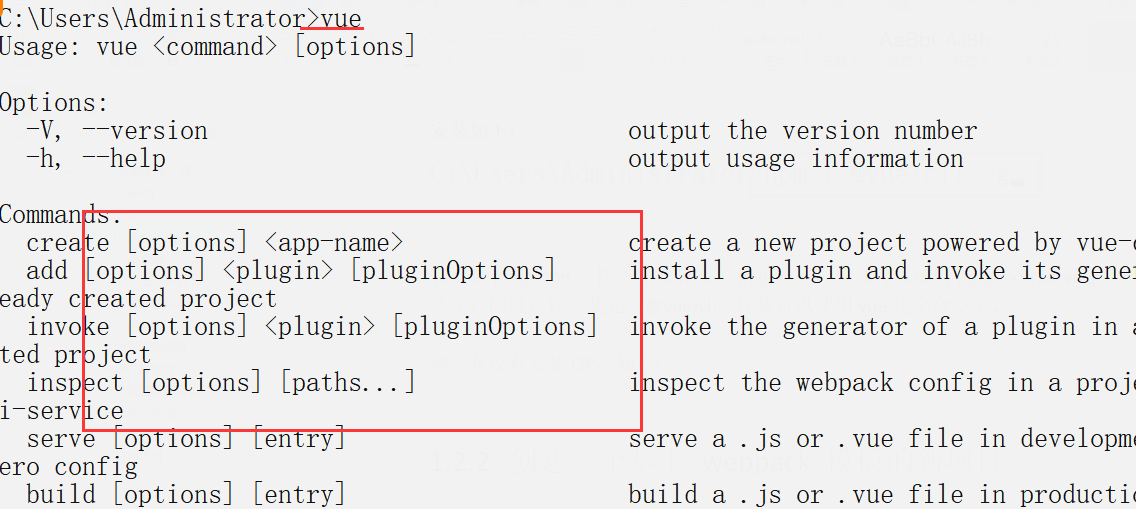


**安装如下：**



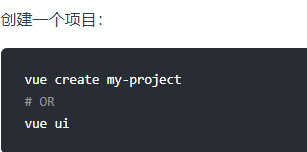
和安装nodemon一样，你安装完了nodemon，你就可以使用nodemon server.js运行这个文件了。同理，你安装完，@vue/cli，你就可以使用vue这个命令了。

**测试有没有安装OK，如下：**

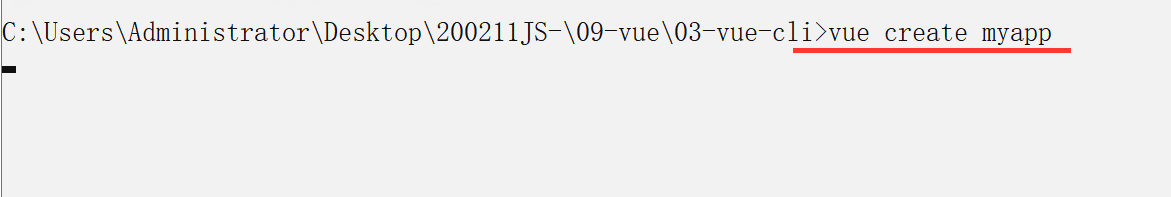


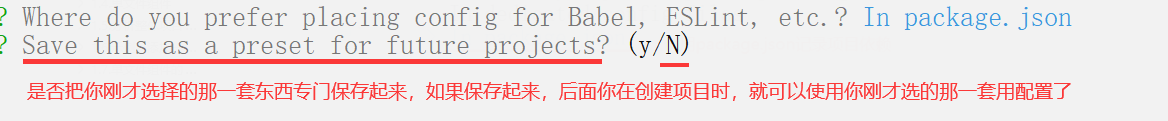
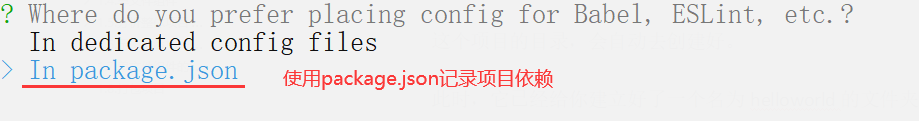
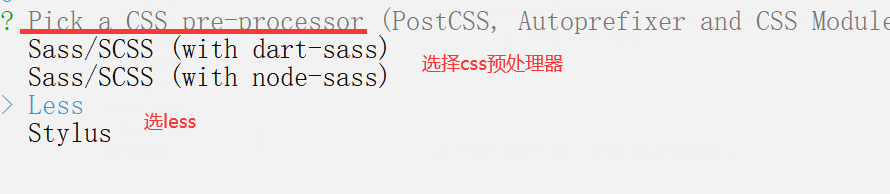
### 创建一个基于 webpack 模板的新项目。

当你完成这个脚手架之后，你就可以使用vue这个命令了。

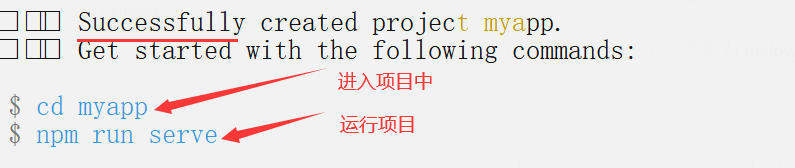


**创建项目如下：**





经过一段时间，项目就创建OK，提示如下：

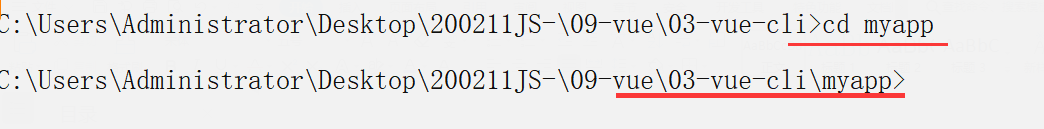


这个项目的目录，会自动去创建好。

此时，它已经给你建立好了一个名为helloworld的文件夹，其中已经搭建了项目的目录。

### 进入项目目录

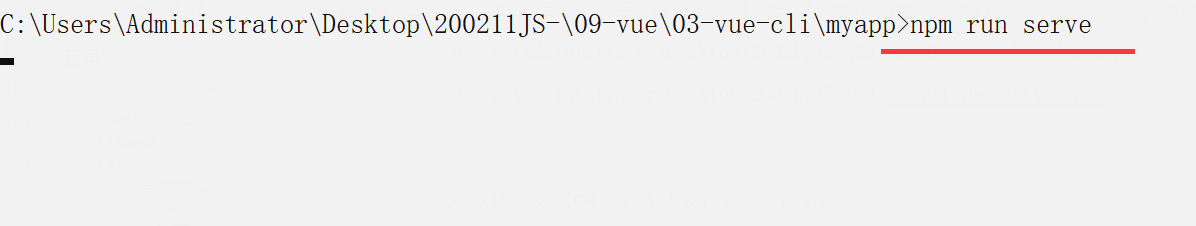
进入项目目录：



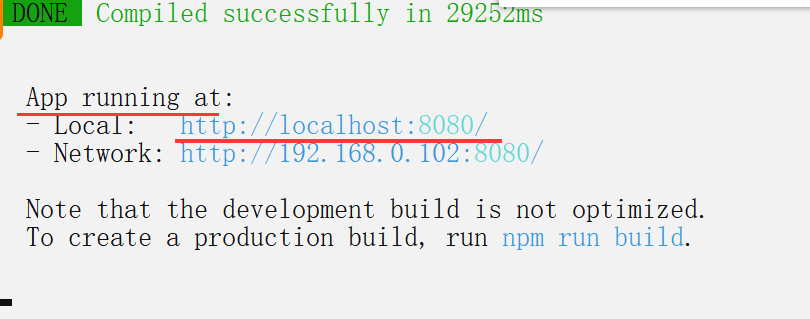
依赖默认都安装OK了，不需要再安装依赖了。

### 运行这个项目

通过npm run serve运行如下：



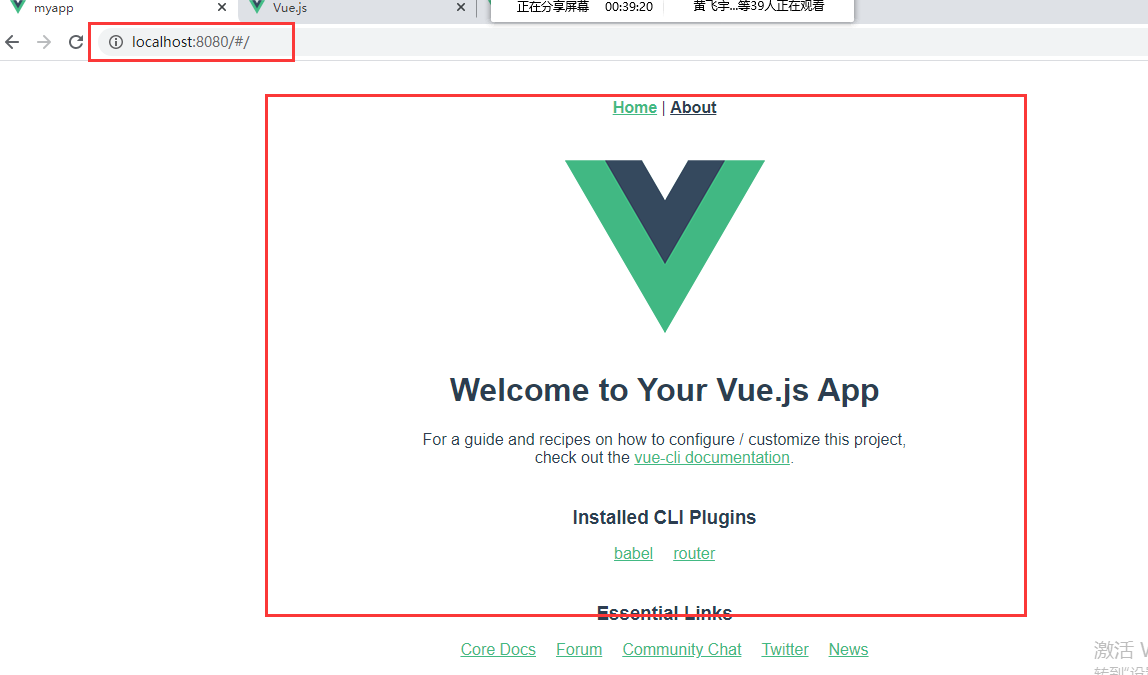
运行后，如下：



这个serve命令是脚手架工具已经写在package.json中的scripts中的命令。如下：



### 效果如下



## 目录结构说明



## 单文件组件

把一个组件所需要的代码全写在一个单独的文件中。换句话，一个文件就是一个组件，vue给这个文件取了一个特殊的后缀名:　.vue。



### 为什么需要这么做？

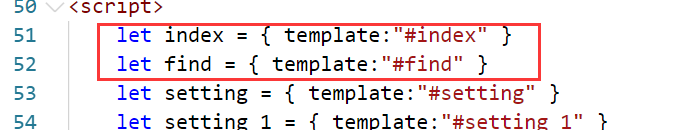
**我们之前写组件的方式比较零散：**



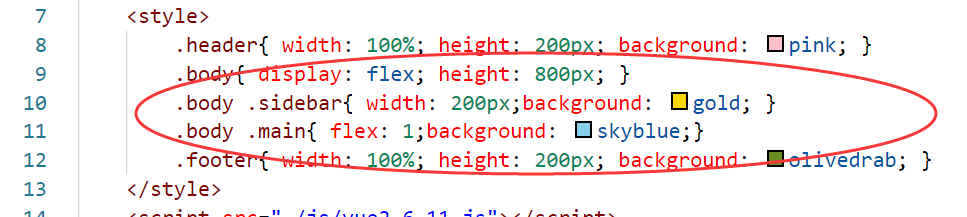
**一个组件要有模板template:**



**一个组件还需要组件对象：**



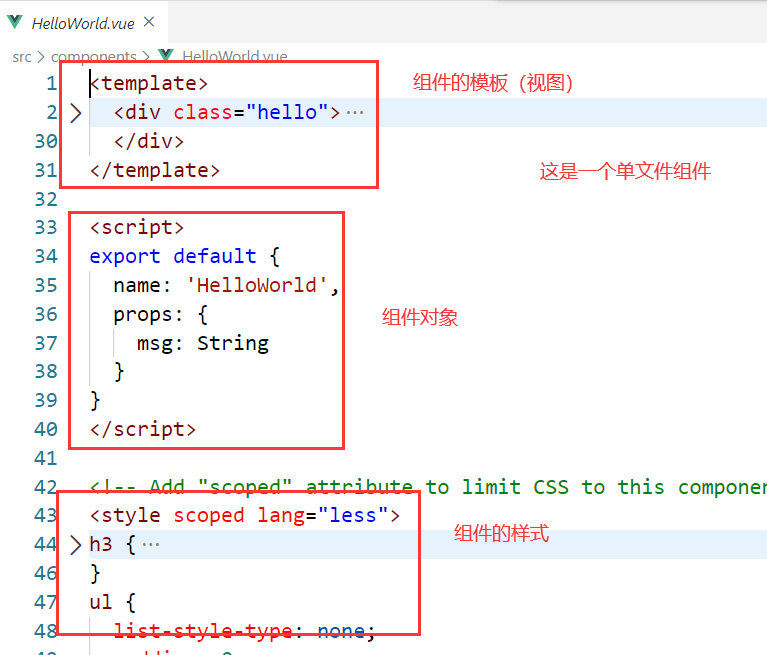
**一个组件还可以有自己的样式：**



一个组件就被拆成三个部分！！

**现在我们可以有一种新的方式去把这三个部分组合一个单独文件中！**

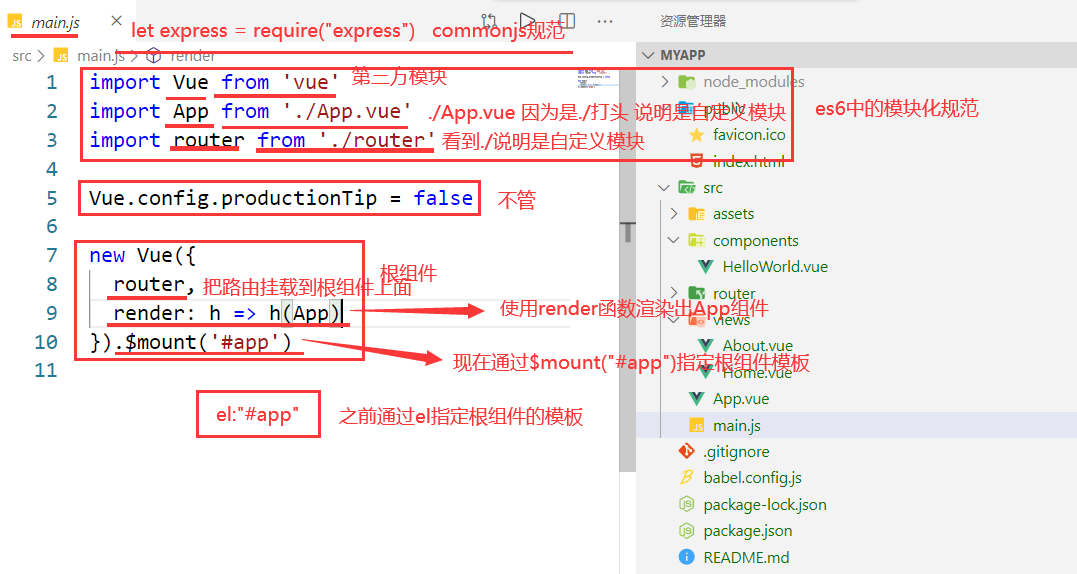
### 参观一下HelloWorld.vue组件



## 分析整体项目的工作流程

### 入口：main.js

一切的入口在这里。

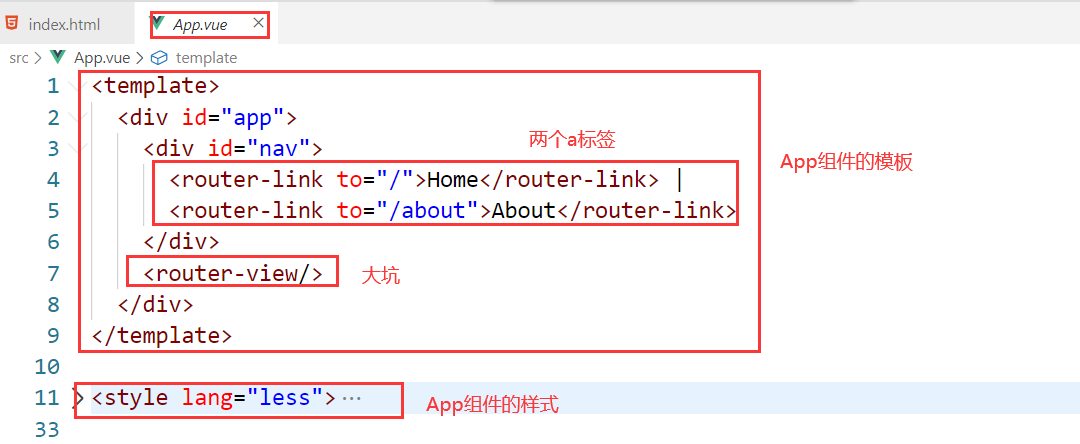


根组件的视图的在public下面的index.html中，如下：



只有一个vue实例，会挂载到index.html中的div#app上。

### 找到App.vue



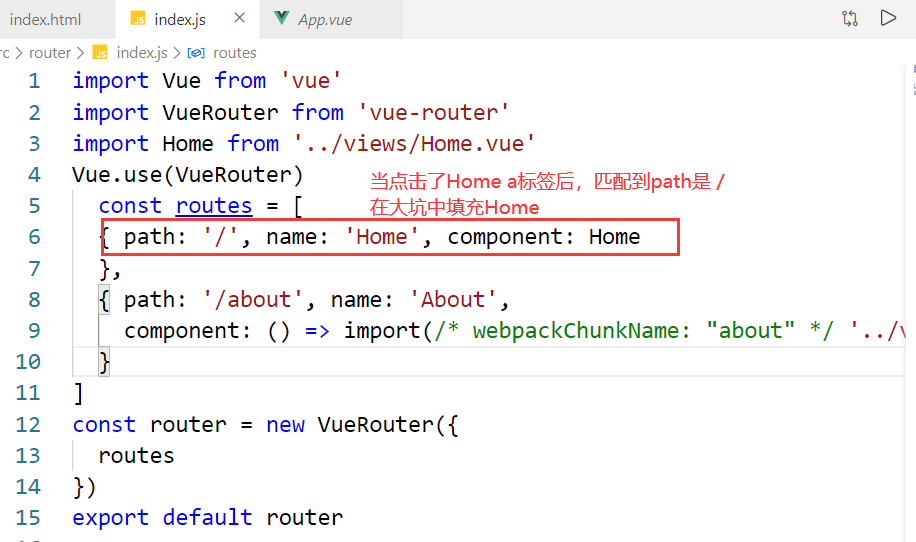
在上面的template中，可以看到不管载入哪个组件，这个图片是一直存在的。

### 路由：router/index.js

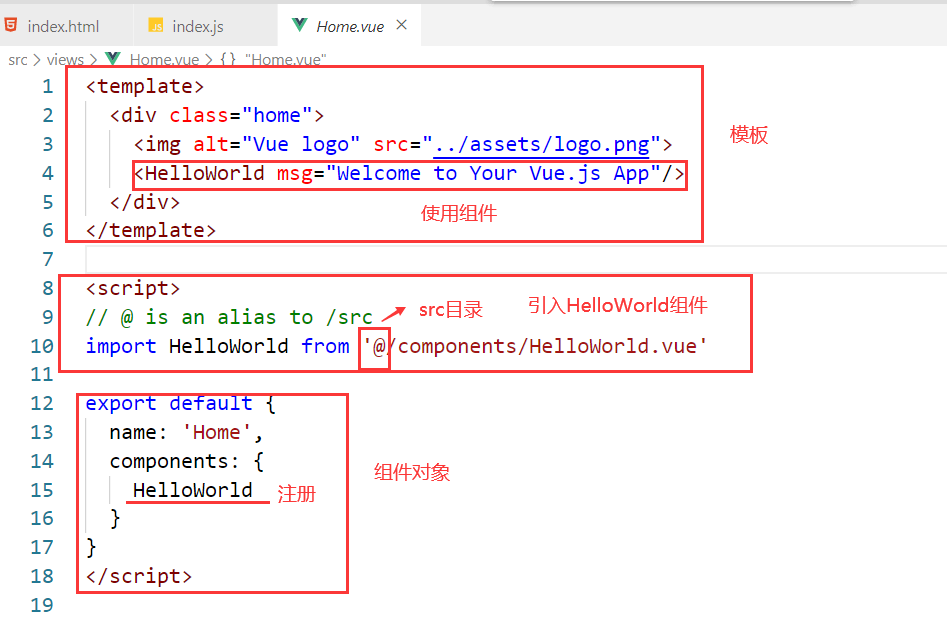
路由对象在vue的实例中被注入，具体的代码在main.js中。现在，去查看router/index.js中的设置项。如下：



### 去进一步查看Home.vue组件



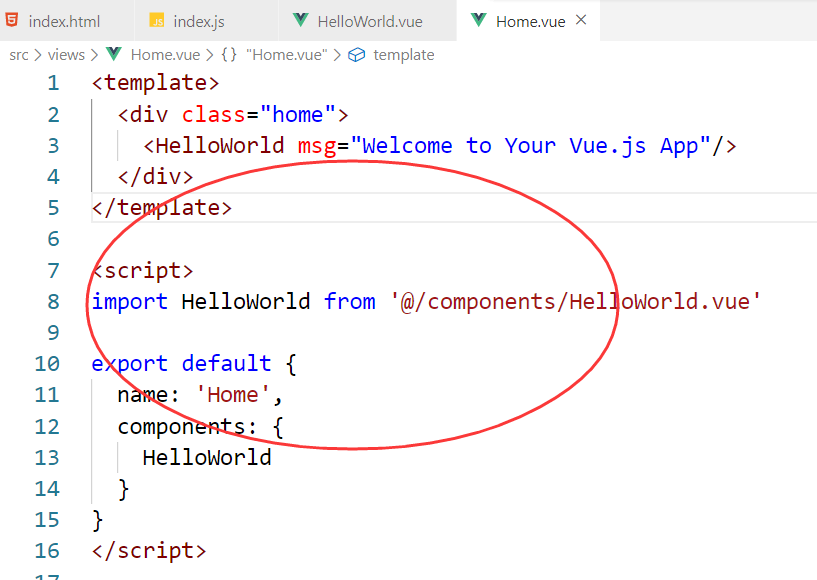
Home.vue组件如下：



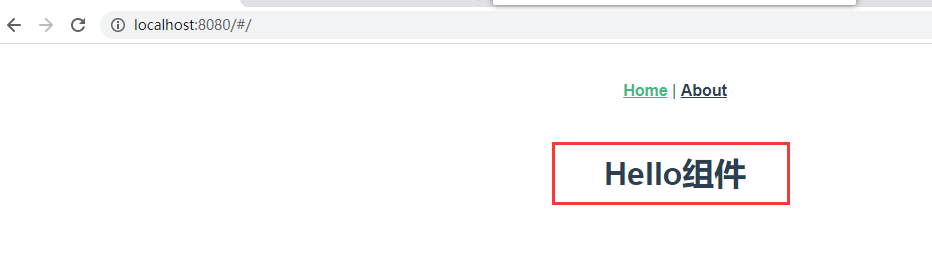
再看HelloWorld组件，如下：



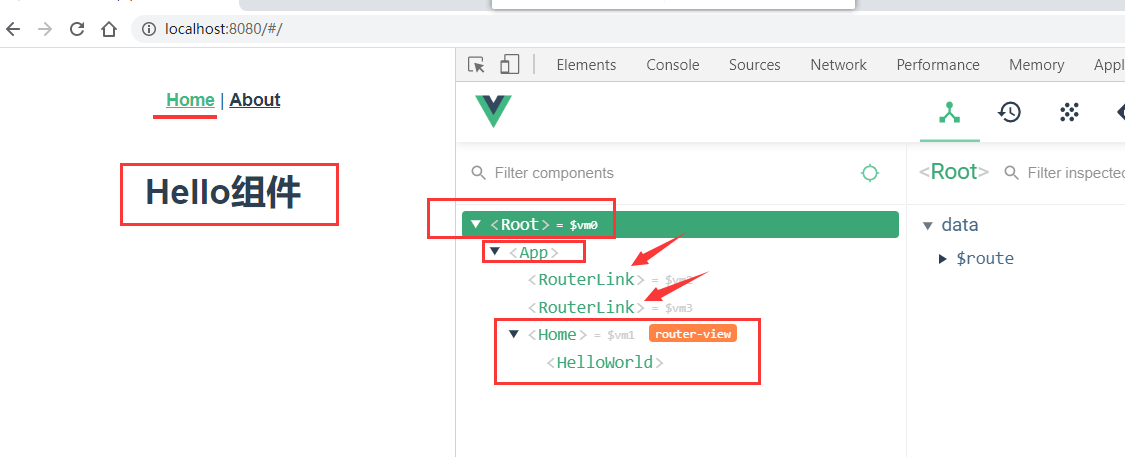
修改代码如下：



效果如下：



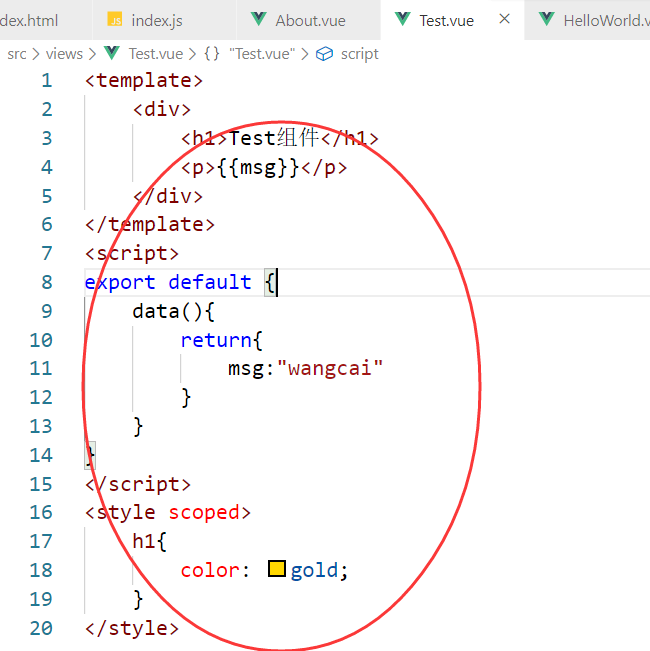
### 结构图



## 示例：添加一个组件 Test.vue

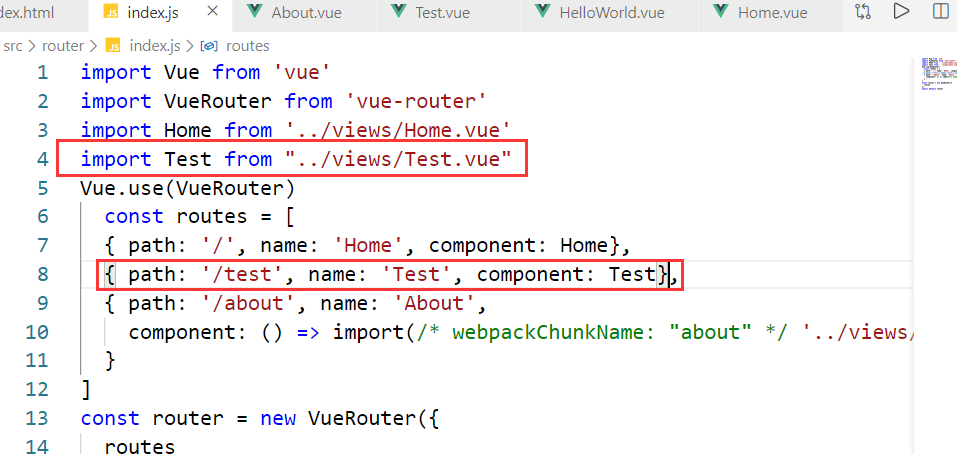
### 创建文件

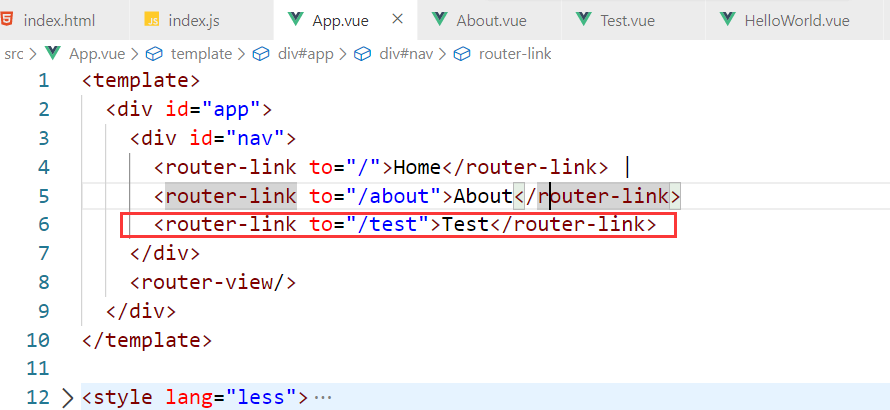
直接.vue格式，会自动分成三块。



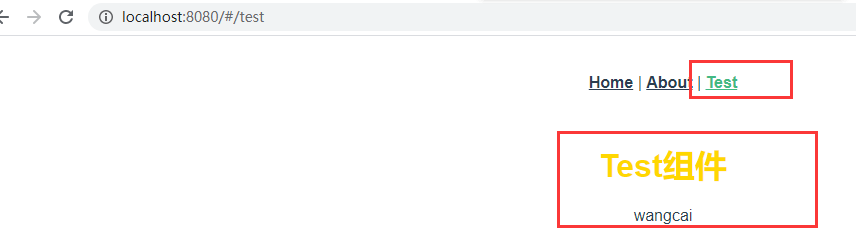
### 在路由中使用组件

配置路由对象。



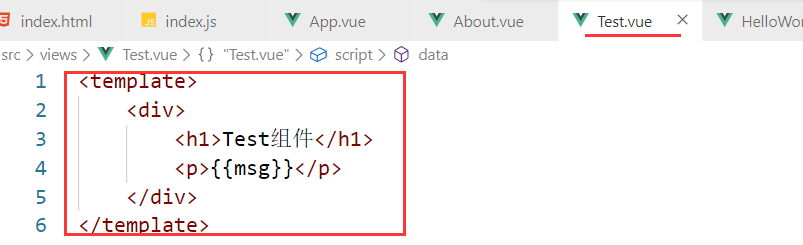


### 测试



## 单文件组件的三个部分的使用细节

### template



就是相当于是

{

template :”#app”

}

已经直接被写好了，就是template标签中的内容。它可以是：

* html标签
* 各种vue指令
* 或者是数据项



### Script

你可以写任何的js代码。

在export default {} 中，就相当于是你定义的组件对象，我们之前定义的组件的：

* data
* computed
* filters
* methods
* components

都一样可以用

和我们之前讲的一样的。

### Style

就是设置当前组件的样式。

有两点要注意：

#### Scoped (scope 作用域)

如果在style上添加scoped，则会在生成样式同时，加上一些独一无二的标记符，它们的作用就是让**这个样式会作用于当前的组件**.

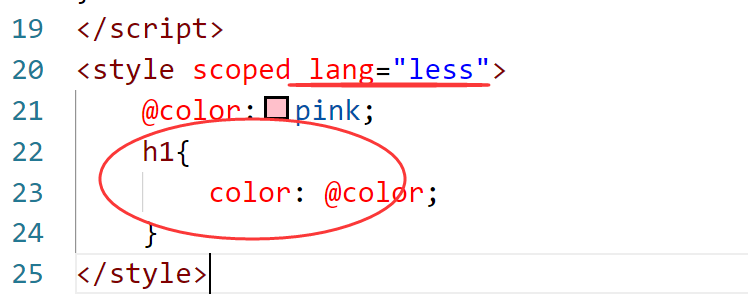


#### Lang

如果你想用其它的css的预编译语言去写样式：

如less,sass.....也可以。

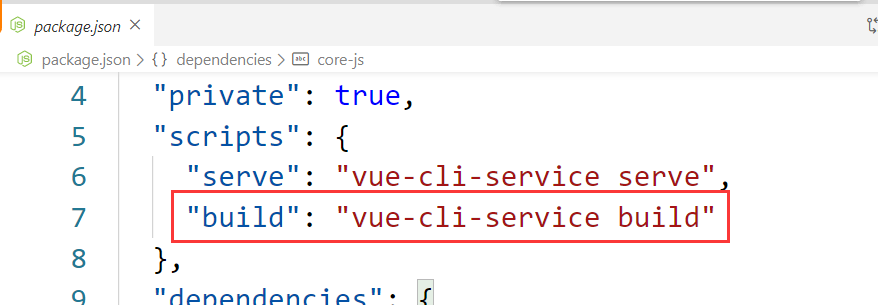
你只需要把lang设置一下：





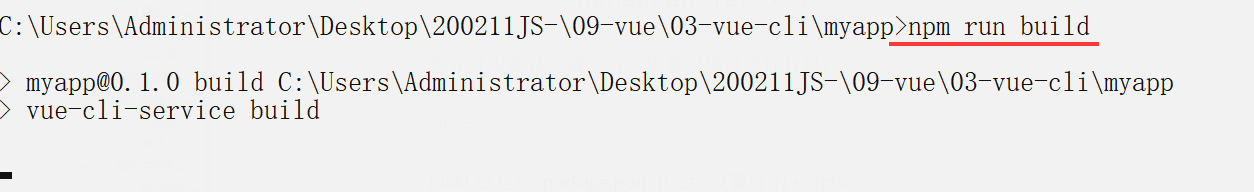
## 打包项目

.vue文件，浏览器肯定不谁，浏览器只认识html,css,js，项目开发完毕，我们需要在硬盘上打一个包，然后放到服务器，让所有人都可以访问。如何打包呢？通过一个命令，如下：



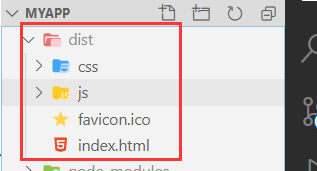
你在开发测试完成之后，则要对项目进行打包。

命令：



build也是在 package.json中已经设置好的scripts：

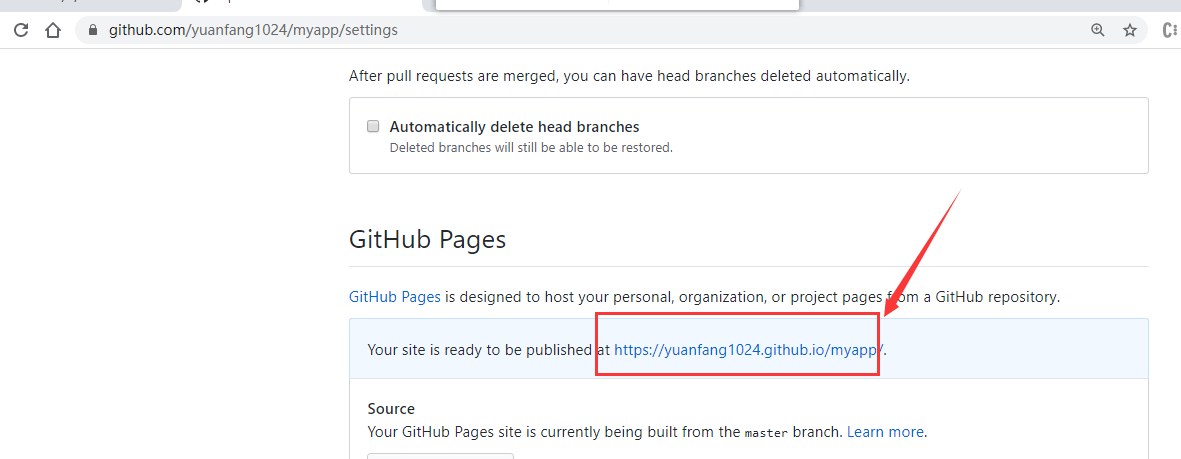
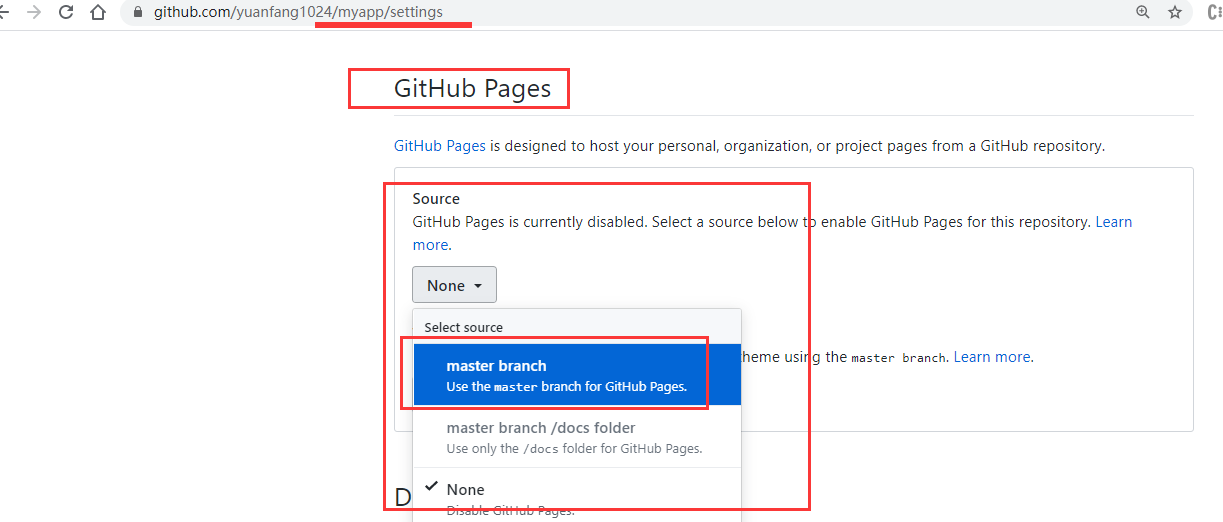
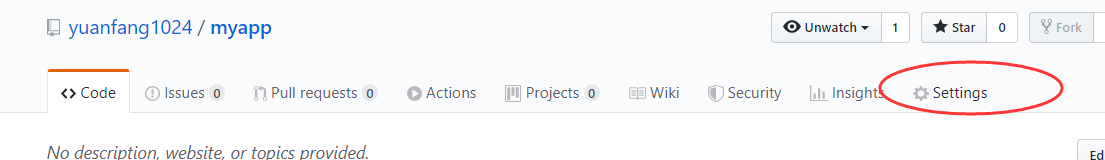
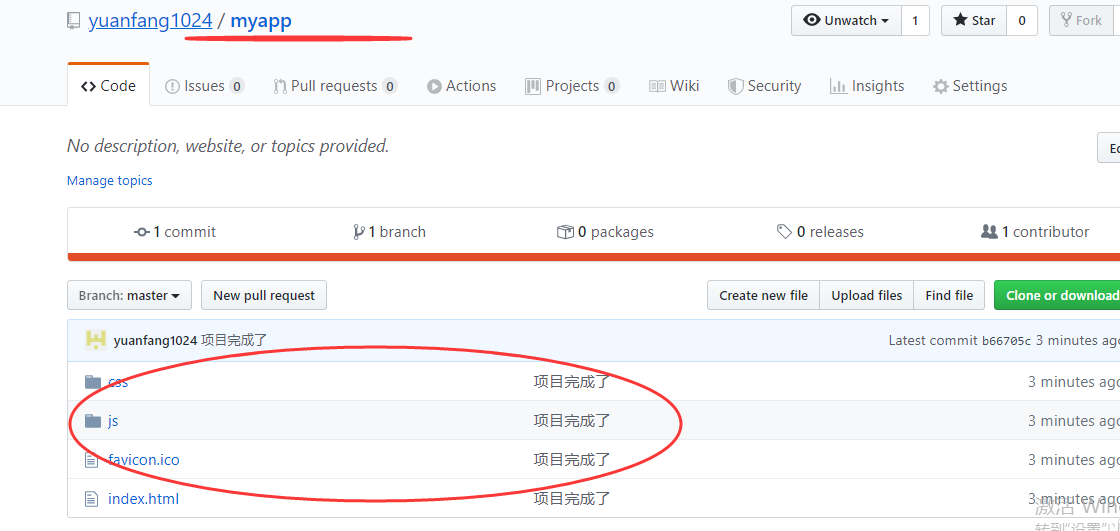
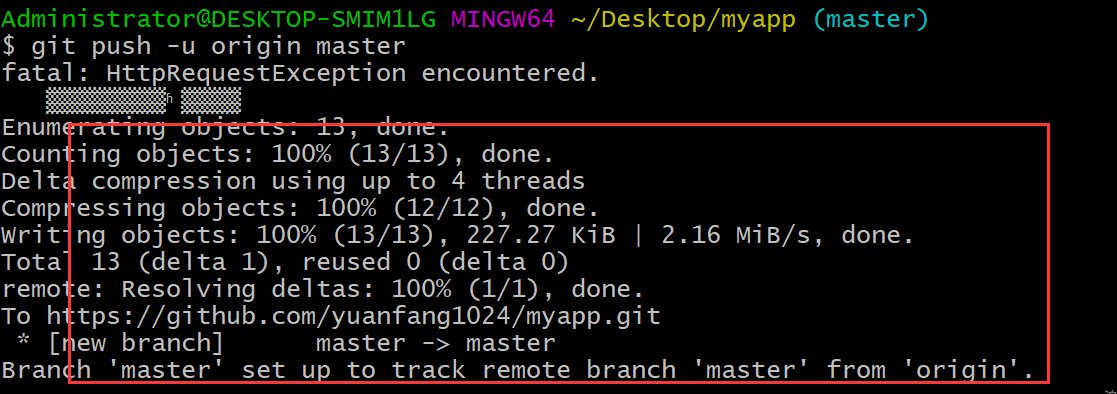
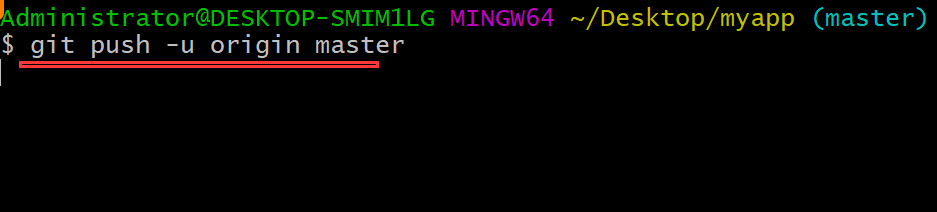
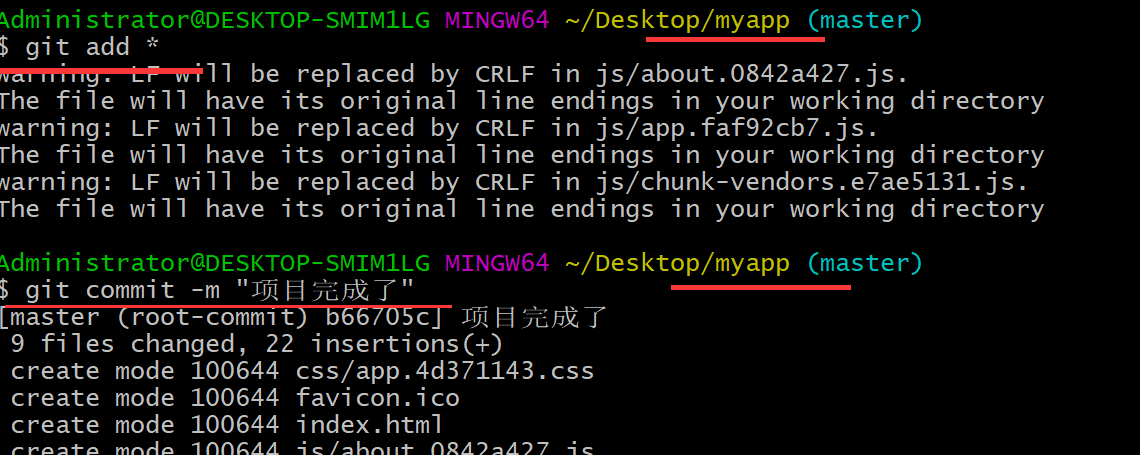
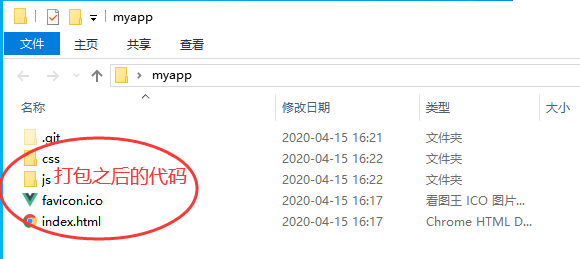
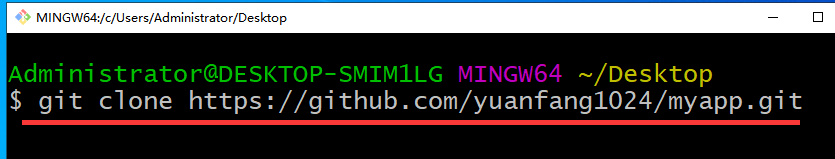
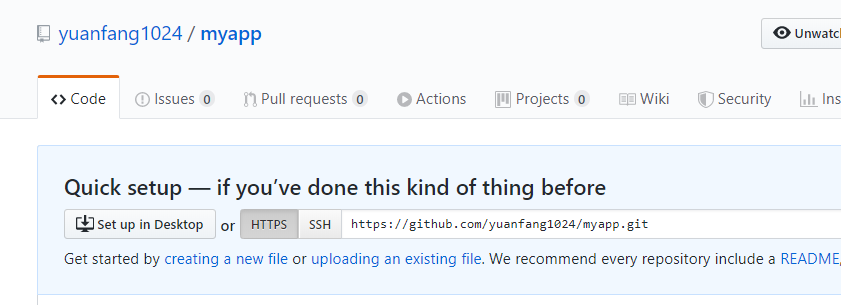
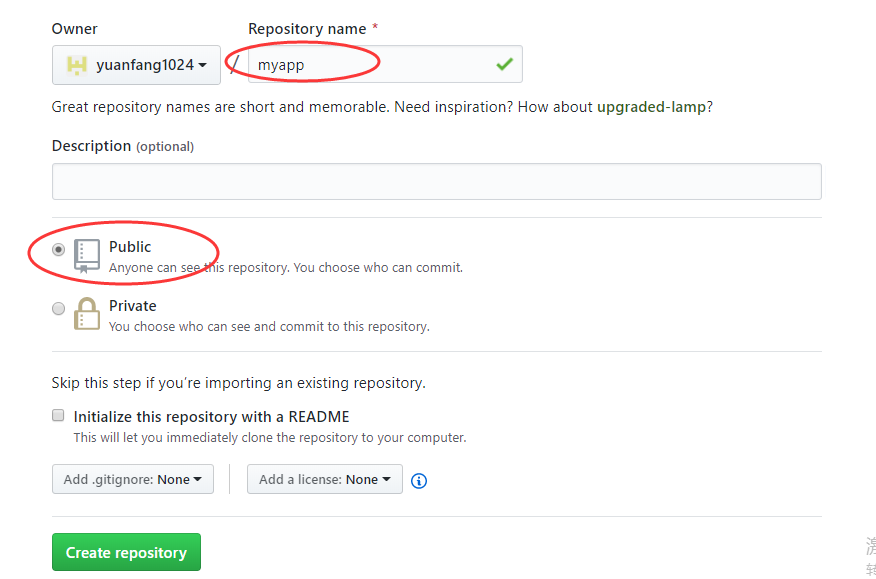
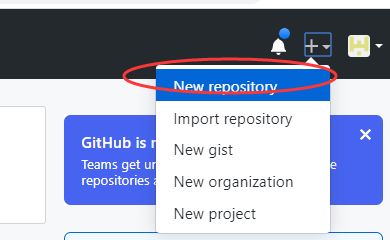
打包的结果如下：



**打包完后，把代码放到服务器上，如何放？**

答：要么我们自己学习一些服务器（linux）运维的内容，要么把代码交给运维工程师。

可以把代码放到github上面的，如下：



访问OK~