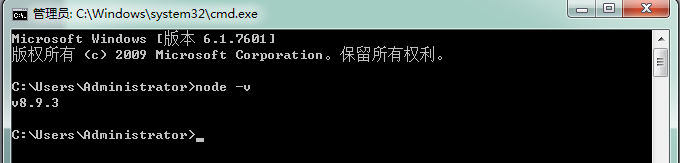
1. 安装node.js，网址 [https://nodejs.org/](https://nodejs.org/" \t "http://www.cnblogs.com/wangfupeng1988/p/_blank)，一直点击下一步安装即可；安装了nodejs之后，可以在你的控制台中输入“node -v”来查看nodejs的版本，也顺便试验nodejs是否安装成功。



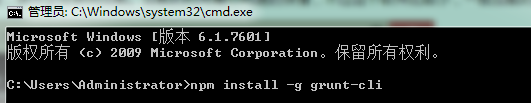
注意两点。第一，grunt依赖于nodejs的v0.8.0及以上版本；第二，奇数版本号的版本被认为是不稳定的开发版，不过从官网上下载下来的应该都是偶数的稳定版。

1. 安装 grunt-CLI

首先保证电脑联网

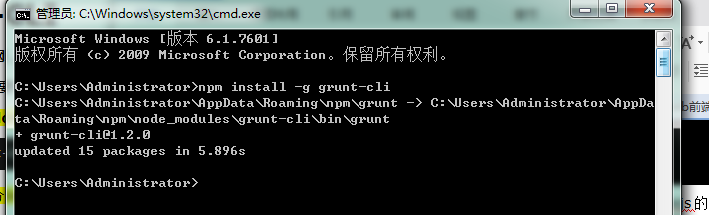
‘CLI’被翻译为 命令行。要想使用grunt，先要保证grunt-CLI安装到全局环境中，使用nodejs的npm安装软件。

打开控制台（windows系统下使用管理员权限打开），输入npm install -g grunt-cli

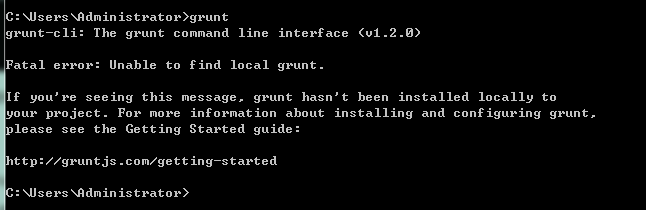


【mac os系统、部分linux系统中，在这个语句之前加 sudo 指令】

回车，命令行会出现一个转动的小横线，表示正在联网加载。加载的时间看你网速的快慢，不过这个软件比较小，一般加载时间不会很长，稍一会儿，就加载完了。你会看到以下界面。



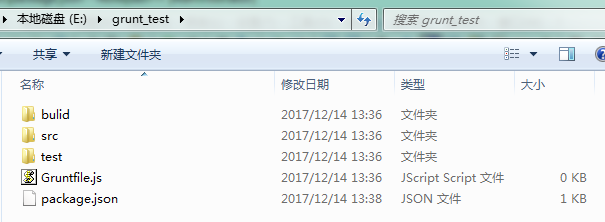
这时候要验证一下grunt-cli是否安装完成并生效，你只需要继续在命令行中输入“grunt”命令即可。如果生效，则会出现以下结果：



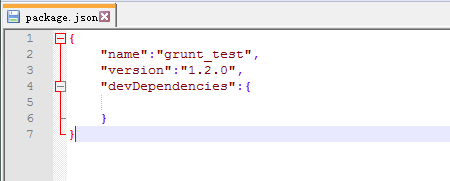
1. 创建一个简单的网站

Grunt是应用于实际项目的，所以需要创建一个简单的项目来掩饰grunt的安装、使用。

　首先，我在电脑的D盘下面建了一个“grunt\_test”文件夹，里面建了三个空文件夹、两个空文档，名称如下图。（注意 Gruntfile.js 文件的首字母大写！）



先把package.json这个文件写一些东西。记住，既然文件后缀名叫“json”，那么文件中的格式肯定是严格的json格式。内容如下：



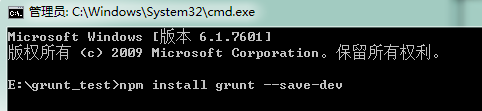
devDependencies //开发依赖项

1. 安装grunt

接下来我们会有一系列插件的安装，他们的安装过程和grunt一样。但是他们的执行都是基于grunt的，因此才能把grunt叫做一个“构建工具”。Grunt没有具体的作用，但是它能把有具体作用的一个一个插件组合起来，形成一个整体效应。

　　例如，你需要检查js语法错误，然后再去压缩js代码。如果这两步你都去手动操作，会耗费很多成本。Grunt就能让你省去这些手动操作的成本。

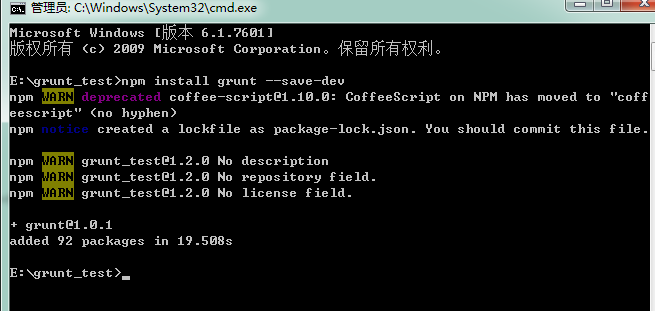
书归正传。注意，**这里安装grunt不再是全局安装了**，需要你在控制台进入到网站或系统的具体目录下。这里我们进入 E:\grunt\_test 目录下。然后输入以下命令：



这里需要解释的是，“—save-dev”的意思是，在当前目录安装grunt的同时，顺便把grunt保存为这个目录的开发依赖项。看到“开发依赖项”这一次，是不是觉得有些眼熟？上文在配置package.json时，其中的“devDependencies”中就会存储开发依赖项。

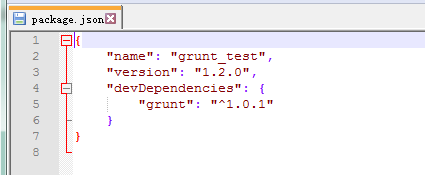
　　具体保存为开发依赖项是一个什么效果？动手安装一下就是了。首先，在运行安装grunt的命令之前，package.json中的“devDependencies”对应的是空对象。

现在我们来运行这行命令。你会看到几条warning提示，不用管它。然后接下来就是加载状态，一个旋转的小横线。稍等待一会儿，会提示你安装成功。如下图：

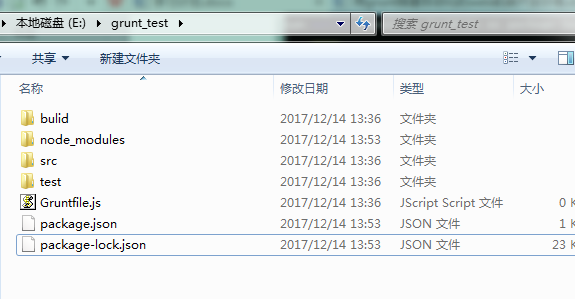


【注意 nodejs升级到8以后，自带的npm相应升级，所以命令语句有变化，可以百度，也可以参考https://www.topjishu.com/11681.html】

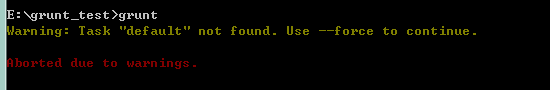
打开package.json去看看，那里的“devDependencies”发生变化。



然后你再看看文档目录中的文件或者文件夹有什么变化？我这里多了一个“node\_modules”文件夹，其中有一个“grunt”文件夹，再其中有若干文档。这里就是存储grunt源文件的地方。



　这是见证奇迹的时刻，别着急，奇迹还没完呢。然后你在控制台运行“grunt”命令。如果你得到一个warning提示，那说明grunt已经起作用了。如下图：

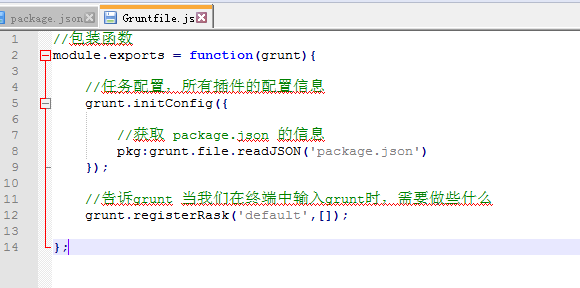


经过以上三步，说明grunt已经在这个目录下成功安装。

　　那么，为何我们在刚才执行grunt时候会有Warning提示呢？根据提示，我们得知的信息是：Task “default” not found ，如何搞定这个问题？——当然是继续往下看啊

1. 配置Gruntflie.js

Gruntfile.js 这个文件，是为了grunt做某种配置的。按照grunt的规定，我们首先把Gruntfile.js配置成如下格式。



在以上代码中，我们看到了刚才运行grunt命令，warning提示中的“default”字眼。不妨我们此时再运行一下grunt命令，看看会不会再次出现“warning”、“default”等字眼。



运行结果告诉我们“Done, without errors”。那就继续往下吧。

　　另外请注意Gruntfile.js中的一句话：

IMG_256

　　这句话是在Gruntfile.js中获取package.json中的内容。在上文配置package.json时我们说过：package.json中的内容不光是用来占位置的，还可以在其他地方获取。这里不是已经获取了package.json内容了吗？至于获取了如何使用，下文会有介绍，还是继续往下看吧。

1. Grunt插件介绍

官网插件列表[http://www.gruntjs.net/plugins](http://www.gruntjs.net/plugins" \t "http://www.cnblogs.com/wangfupeng1988/p/_blank)

* Contrib-jshint——javascript语法错误检查；
* Contrib-watch——实时监控文件变化、调用相应的任务重新执行；
* Contrib-clean——清空文件、文件夹；
* Contrib-uglify——压缩javascript代码
* Contrib-copy——复制文件、文件夹
* Contrib-concat——合并多个文件的代码到一个文件中
* karma——前端自动化测试工具
* cssmin——压缩css代码插件

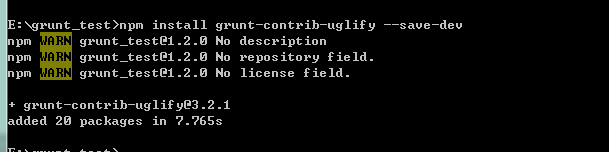
1. 使用uglify插件（压缩js代码）

安装插件时，先去官网的插件列表中找到该插件，点击查看详情，内部含有下载插件的方式，以及配置项详情内容。

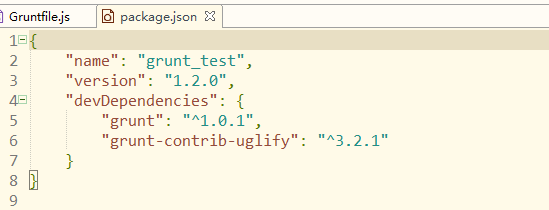
开始安装：

首先执行

npm install grunt-contrib-uglify --save-dev



此时 package.json文件中发生变化：



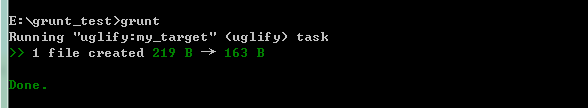
然后开始使用该插件，进行js代码的压缩工作。

需要使用Gruntflie.js配置。

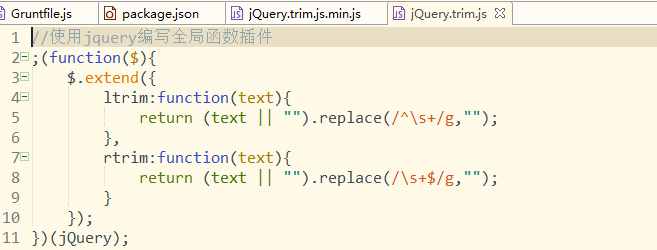
具体该插件的配置详情可查看插件官网，此处以比较基本的方式为例演示，在压缩时，使用banner option，压缩后的文件会在开始部分展示 版本号，项目名称，创建时间，具体配置如下：



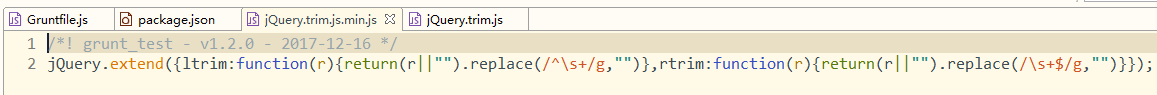
配置完成之后，就在终端，输入指令 grunt 即可，压缩结果如下：



压缩前：



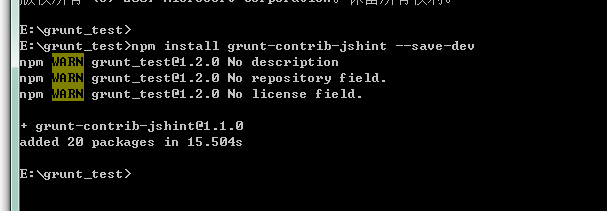
压缩后：

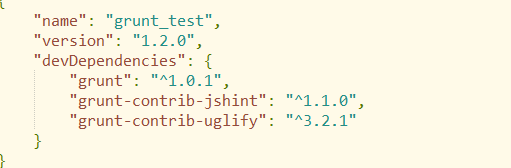


1. 使用jshint插件（检测js代码语法错误）

执行命令：

npm install grunt-contrib-jshint --save-dev





编写配置文件，其中该插件的配置项有很多，详情请看http://jshint.com/docs/options/

此处就以官网给出的几个作为示例。

options**:** {

      curly**:** true, //循环或者条件语句必须使用换括号包住

      eqeqeq**:** true, //禁止使用 ==和！==，强制使用===和！==

      eqnull**:** true,

      browser**:** true, //暴露浏览器属性的全局变量 例如 window document

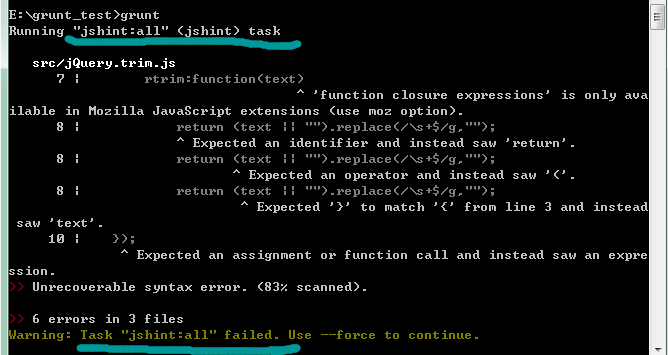
      globals**:** { //global 详情的，全局的

        jQuery**:** true

      },

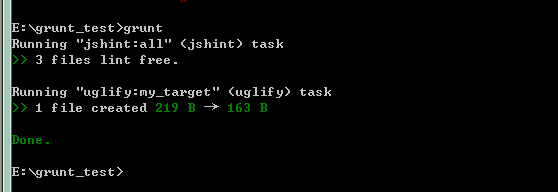
    }

没有使用配置项，使用的是jshint的all任务，检测所有的js文件



检测出问题，并且下一个任务【uglify】不执行

更改代码后再次执行grunt任务。



两个任务 全部完成。

1. 使用watch插件（实现自动化）

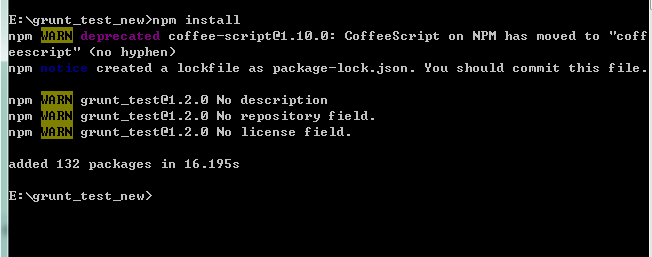
此插件其实是一个监听插件，配置好需要监听的文件，以及 终端要执行的tasks即可。

目标文件发生变化，监测到变化，执行 tasks。

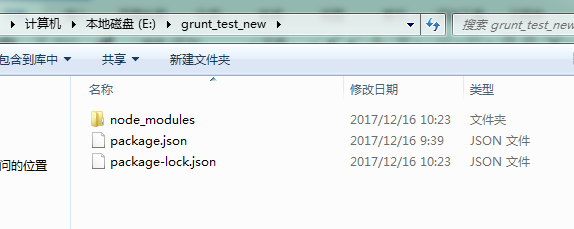
1. 批量现在插件

在实际项目开发过程中，使用版本控制软件时，不会把 依赖的插件全部push，所以只需要push一个package.json文件，里面记录了所有的插件信息，然后利用这个文件，直接在项目中执行 npm install命令，会发生什么呢？

我在e盘中新建一个文件夹，grunt\_test\_new，在此文件夹下把之前grunt\_test中的package.json文件copy进去。然后进入该文件目录下，cmd，执行npm install命令。



该目录文件下也发生变化：



1. 结束语

没有一一把插件都安装完毕，只是走一个流程，熟悉之后，在脑子里有一个脉络，知道是怎么一回事，怎么做就好。需要什么插件直接去官网的插件list中查找，然后根据说明相应的安装使用插件即可。

前提是需要安装nodejs，然后使用nodejs中的npm安装插件。也可以使用淘宝镜像安装cnpm，因为有些插件可能使用npm下载速度慢或者下载失败的问题。

其次，使用grunt中的各个插件是为了让项目更好，检测代码格式，压缩代码等问题。但是也不要为了使用而使用。

最后通过这个工具，衍生出另一个问题，就是项目的文档结构要有自己的思想，例如一个目录下是原始代码，一个目录下是grunt加工后的代码，另一个是平时测试的代码。之后每个据图的目录中也要有清晰的脉络结构。