# F.I.S

# 概要

本文档主要介绍了fis静态资源库的作用和使用，以及前端环境的简单配置说明，让大家了解铭飞前端的环境与资源库结构与使用。

# 使用工具

## Fis

#### Fis说明

FIS是前端静态资源库，铭飞的一些静态资源比如前端的js和api接口文件和一些插件的资源我们都放在fis中，我们使用的时候都可以用线上地址调用到fis中的文件，这样让我们可以减少大部分常用资源的本地使用，实现版本的统一管理。

#### fis使用实例

我们平时引用的这些js就是统一发布到cdn上进行资源调用的。



## Gulp

#### Gulp说明

Gulp是前端开发过程中对代码进行构建的工具，是自动化项目的构建利器，基于node.js，她不仅能对网站资源进行优化，而且在开发过程中很多重复的任务能够使用正确的工具自动完成；使用她，我们不仅可以很愉快的编写代码，而且大大提高我们的工作效率。我们在编写的过程中通常都是团队协作，这里我们就可以通过Gulp提交代码，实现代码的统一。

## Node

#### Node说明

Node 是一个基于Chrome V8 JavaScript引擎构建的JavaScript运行时。Node.js使用事件驱动的非阻塞I / O模型，使其轻便且高效。它既是开发平台, 也是运行环境, 也是个新的语言，它本身是基于google的javascript v8引擎开发的, 因此在编写基于它的代码的时候使用javascript语言. 但是又不同于传统概念的javascript，它的服务端功能以及部分客户端功能必须在服务端运行, 所以它实际上是一种在服务端的开发+运行的javascript语言. 有一点类似于Perl + PHP或者Python的概念。我们使用的Gulp就是基于Node.js构建的。

#### Node.js下载

下载地址：<https://nodejs.org/>

下载方式：点击绿色的Download按钮，它会根据系统信息选择对应版本（.msi文件），下载完成之后直接运行程序安装（安装路径随意）。Npm会随安装包一起安装。

## Npm

#### Npm说明

NPM是随同NodeJS一起安装的包管理工具，能解决NodeJS代码部署上的很多问题。

由于新版的nodejs已经集成了npm，所以之前npm也一并安装好了。同样可以通过输入 **"npm -v"**来测试是否成功安装。命令如下，出现版本提示表示安装成功:

npm -v

2.3.0

#### Npm升级

如果你安装的是旧版本的 npm，可以很容易得通过 npm 命令来升级，命令如下

npm install npm -g

#### 使用Npm

使用npm安装插件：命令提示符执行**npm install <name> [-g] [--save-dev]， <name>**：node插件名称。例：**npm install gulp-less --save-dev**

**-g**：全局安装C:\Users\Administrator\AppData\Roaming\npm，并且写入系统环境变量；非全局安装：将会安装在当前定位目录； 全局安装可以通过命令行在任何地方调用它，本地安装将安装在定位目录的node\_modules文件夹下，通过require()调用；

**--save**：将保存配置信息至package.json（package.json是nodejs项目配置文件）；

**-dev**：保存至package.json的devDependencies节点，不指定-dev将保存至dependencies节点；一般保存在dependencies的像这些express/ejs/body-parser等等。

为什么要保存至package.json？因为node插件包相对来说非常庞大，所以不加入版本管理，将配置信息写入package.json并将其加入版本管理，其他开发者对应下载即可（命令提示符执行**npm install**，则会根据package.json下载所有需要的包，**npm install --production**只下载dependencies节点的包）。

使用npm卸载插件：npm uninstall <name> [-g] [--save-dev]  PS：不要直接删除本地插件包

删除全部插件：**npm uninstall gulp-less gulp-uglify gulp-concat ……**太麻烦

借助**rimraf**：**npm install rimraf -g** 用法：**rimraf node\_modules**

使用npm更新插件：**npm update <name> [-g]** [--save-dev]

更新全部插件：**npm update [--save-dev]**

查看npm帮助：**npm help**

当前目录已安装插件：**npm list**

## 命令行

#### 命令行说明

命令提示符是在操作系统中，提示进行命令输入的一种工作提示符。在不同的操作系统环境下，命令提示符各不相同。命令行在OSX是终端（Terminal），在windows是命令提示符（Command Prompt）。在windows环境下，命令行程序为cmd.exe，是一个32位的命令行程序，微软Windows系统基于Windows上的命令解释程序，类似于微软的DOS操作系统。输入一些命令，cmd.exe可以执行，比如输入shutdown -s就会在30秒后关机。总之，它非常有用。打开方法：开始-所有程序-附件 或 开始-寻找-输入：cmd/cmd.exe 回车。它也可以执行BAT文件。

#### 简单的命令行

简单介绍gulp在使用过程中常用命令，打开命令提示符执行下列命令（打开方式：window + r 输入cmd回车）

node -v，查看安装的nodejs版本，出现版本号，说明刚刚已正确安装nodejs。PS：未能出现版本号，请尝试注销电脑重试。

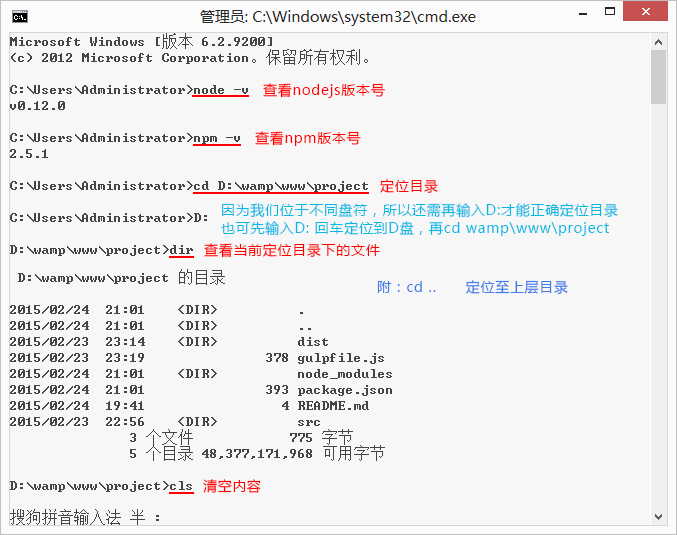
npm -v，查看npm的版本号，npm是在安装nodejs时一同安装的nodejs包管理器

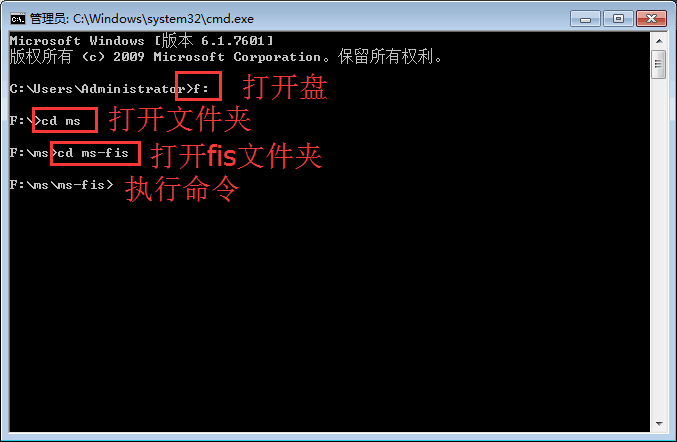
cd，定位到目录，用法：cd + 路径。

ls，列出文件列表。

dir，列出文件列表。

Cls**，**清空命令提示符窗口内容。



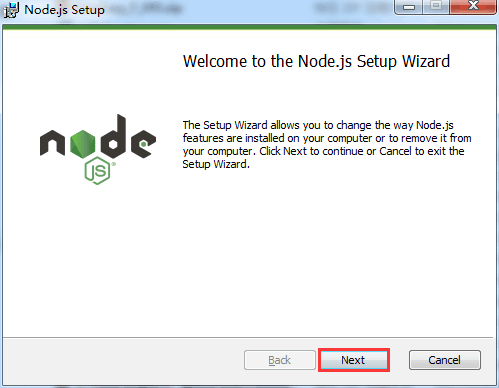


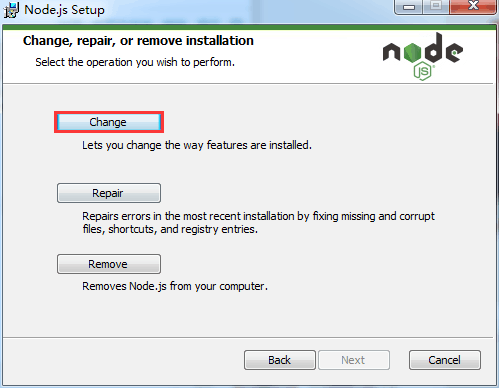
# 安装步骤

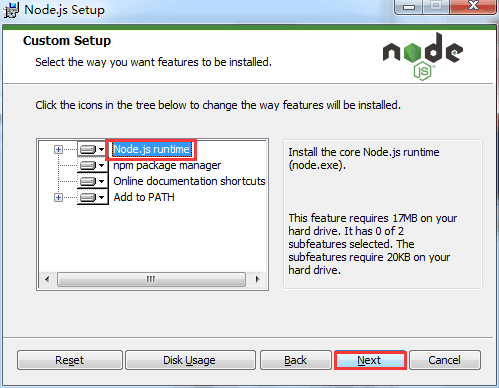
## 安装Node

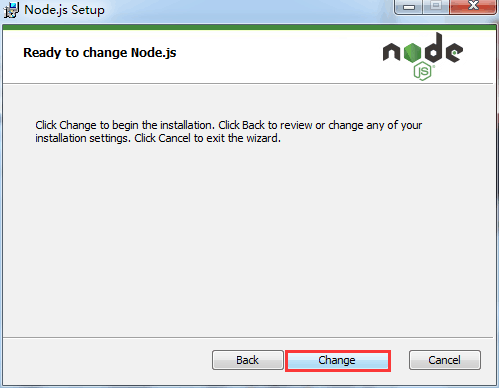
#### Node.js安装步骤

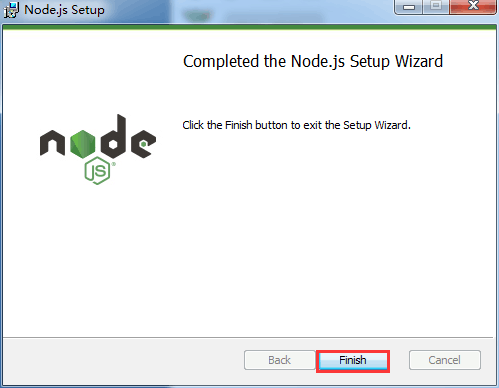
首先，最基本也最重要的是，我们需要搭建node环境。访问[http://nodejs.org](http://nodejs.org/)，然后点击大大的绿色的install按钮，下载完成后直接运行程序，就一切准备就绪。[npm](https://npmjs.org/)会随着安装包一起安装，稍后会用到它。











## 使用命令行

也许现在你还不是很了解什么是命令行——OSX中的终端（Terminal），windows中的命令提示符（Command Prompt），但很快你就会知道。它看起来没那么简单，但一旦掌握了它的窍门，就可以很方便的执行很多命令行程序，比如Sass，Yeoman和Git等，这些都是非常有用的工具。

为了确保Node已经正确安装，我们执行几个简单的命令。

node -v

回车（Enter），如果正确安装的话，你会看到所安装的Node的版本号，接下来看看npm。

npm –v

这同样能得到npm的版本号。

如果这两行命令没有得到返回，可能node就没有安装正确，尝试重启下命令行工具，如果还不行的话，只能回到第一步进行重装。

## 定位到项目

用命令就能定位到文件目录并看看目录里都有些什么文件。

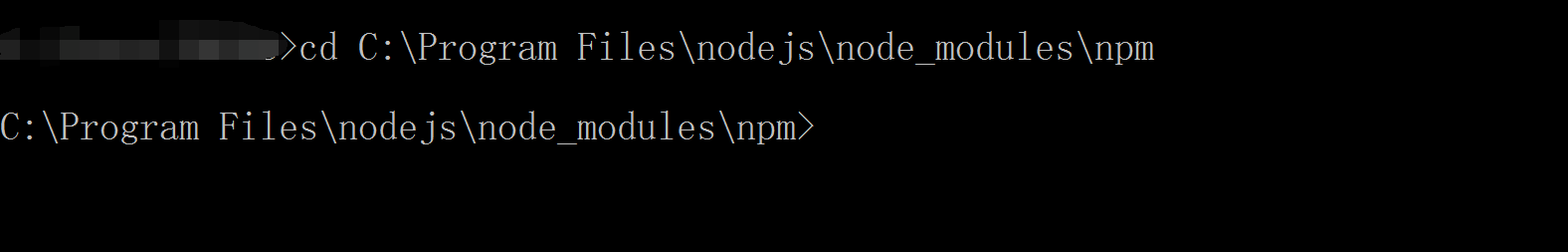
cd，定位到目录

ls，列出文件列表

建议多敲敲这两个命令，了解文件系统并知道文件都在哪里。

定位进入我们的项目目录，这个目录各不相同，举个例子，这是我进入我项目目录的命令：

cd C:\Program Files\nodejs\node\_modules\npm

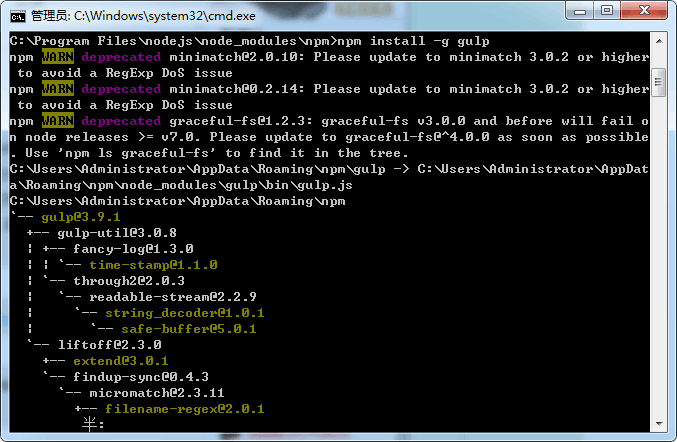


成功进入项目目录后，我们开始安装gulp。

## 安装gulp

1.在命令行输入（注意用管理员权限运行）：

npm install -g gulp



说明：

（1）npm是安装node模块的工具，执行install命令

（2）-g表示在全局环境安装，以便任何项目都能使用它

（3）gulp是将要安装的node模块的名字

2.运行时注意查看命令行有没有错误信息，安装完成后，你可以使用下面的命令查看gulp的版本号以确保gulp已经被正确安装。

gulp –v

这同样能得到npm的版本号。

然后将gulp安装到项目本地

npm install --save-dev gulp

这里，我们使用--save-dev来更新package.json文件，更新devDependencies值，以表明项目需要依赖gulp。

Dependencies可以向其他参与项目的人指明项目在开发环境和生产环境中的node模块依懒关系。

## 新建Gulpfile文件，运行gulp

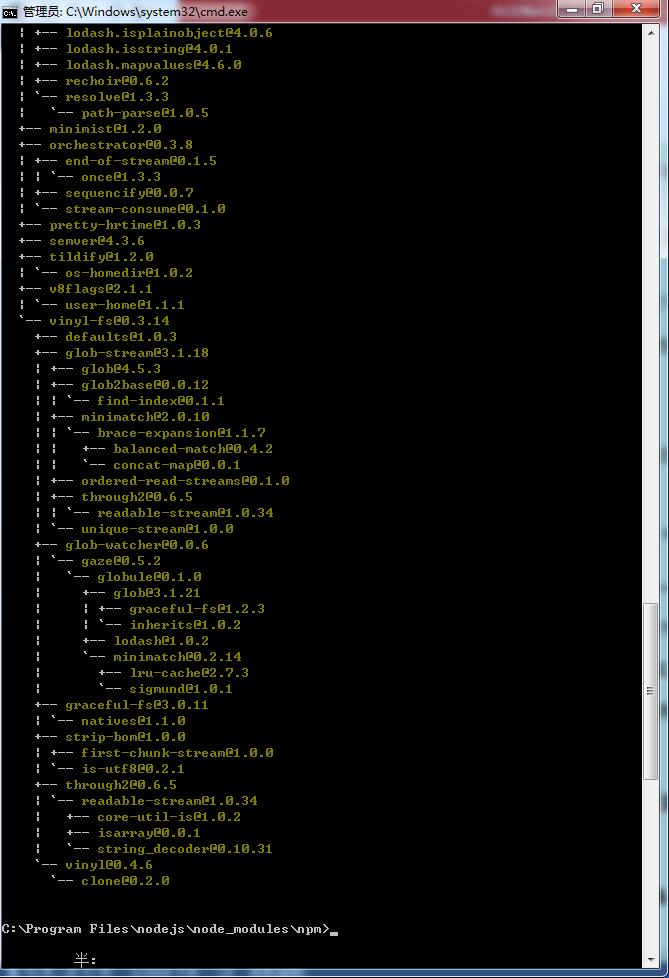
安装好gulp后我们需要告诉它要为我们执行哪些任务，需要弄清楚项目需要哪些任务，再安装相应的插件

#### 安装依赖

1.打开命令行输入下面代码

npm install gulp-jshint gulp-sass gulp-concat gulp-uglify gulp-rename --save-dev

成功结果如下图



#### 新建gulpfile.js

1.说明：gulpfile.js是gulp项目的配置文件，是位于项目根目录的普通js文件（其实将gulpfile.js放入其他文件夹下亦可）

gulp只有五个方法： task，run，watch，src，和dest，在项目根目录新建一个js文件并命名为gulpfile.js，把下面的代码粘贴进去：

**gulpfile.js**

// 引入 gulp

**var** gulp = require('gulp');

// 引入组件

**var** jshint = require('gulp-jshint');

**var** sass = require('gulp-sass');

**var** concat = require('gulp-concat');

**var** uglify = require('gulp-uglify');

**var** rename = require('gulp-rename');

// 检查脚本

gulp.task('lint', **function**() {

gulp.src('./js/\*.js')

.pipe(jshint())

.pipe(jshint.reporter('default'));

});

// 编译Sass

gulp.task('sass', **function**() {

gulp.src('./scss/\*.scss')

.pipe(sass())

.pipe(gulp.dest('./css'));

});

// 合并，压缩文件

gulp.task('scripts', **function**() {

gulp.src('./js/\*.js')

.pipe(concat('all.js'))

.pipe(gulp.dest('./dist'))

.pipe(rename('all.min.js'))

.pipe(uglify())

.pipe(gulp.dest('./dist'));

});

// 默认任务

gulp.task('default', **function**(){

gulp.run('lint', 'sass', 'scripts');

// 监听文件变化

gulp.watch('./js/\*.js', **function**(){

gulp.run('lint', 'sass', 'scripts');

});

});



##### 说明

###### 引入组件

**var** gulp = **require**('gulp');

**var** jshint = **require**('gulp-jshint');

**var** sass = **require**('gulp-sass');

**var** **concat** = **require**('gulp-concat');

**var** uglify = **require**('gulp-uglify');

**var** rename = **require**('gulp-rename');

引入了核心的gulp和其他依赖组件，然后分开创建lint, sass, scripts 和 default这四个不同的任务。

###### Lint任务

gulp.task('lint', **function**() {

gulp.src('./js/\*.js')

.pipe(jshint())

.pipe(jshint.reporter('default'));

});

Link任务会检查js/目录下得js文件有没有报错或警告。

###### Sass任务

gulp.task('sass', **function**() {

gulp.src('./scss/\*.scss')

.pipe(sass())

.pipe(gulp.dest('./css'));

});

Sass任务会编译scss/目录下的scss文件，并把编译完成的css文件保存到/css目录中。

###### Scripts任务

**gulp**.task('scripts', function() {

**gulp**.src('./js/\*.js')

.pipe(concat('all.js'))

.pipe(gulp.dest('./dist'))

.pipe(rename('all.min.js'))

.pipe(uglify())

.pipe(gulp.dest('./dist'));

});

scripts任务会合并js/目录下得所有得js文件并输出到dist/目录，然后gulp会重命名、压缩合并的文件，也输出到dist/目录。

###### Default任务

gulp.task('default', **function**(){

gulp.run('lint', 'sass', 'scripts');

gulp.watch('./js/\*.js', **function**(){

gulp.run('lint', 'sass', 'scripts');

});

});

创建了一个基于其他任务的default任务。使用.run()方法关联和运行我们上面定义的任务，使用.watch()方法去监听指定目录的文件变化，当有文件变化时，会运行回调定义的其他任务。

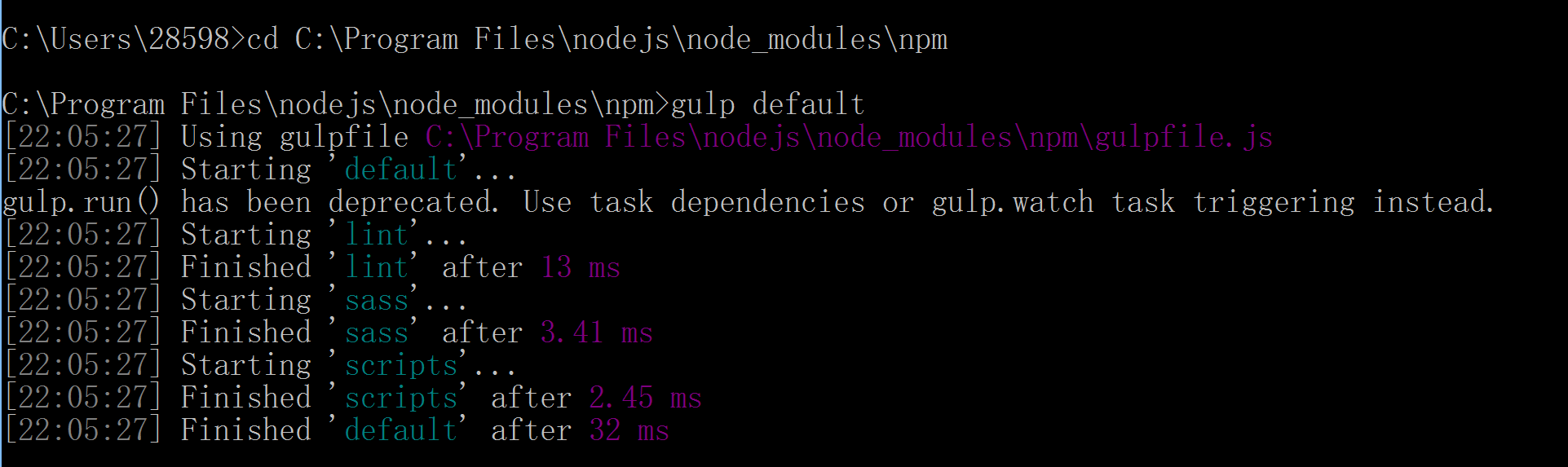
#### 结果验证

命令行输入

Gulp

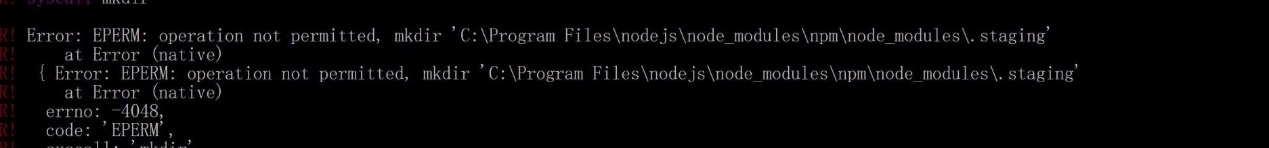
这将执行定义的default任务，换言之，这和以下的命令式同一个意思。

gulp default



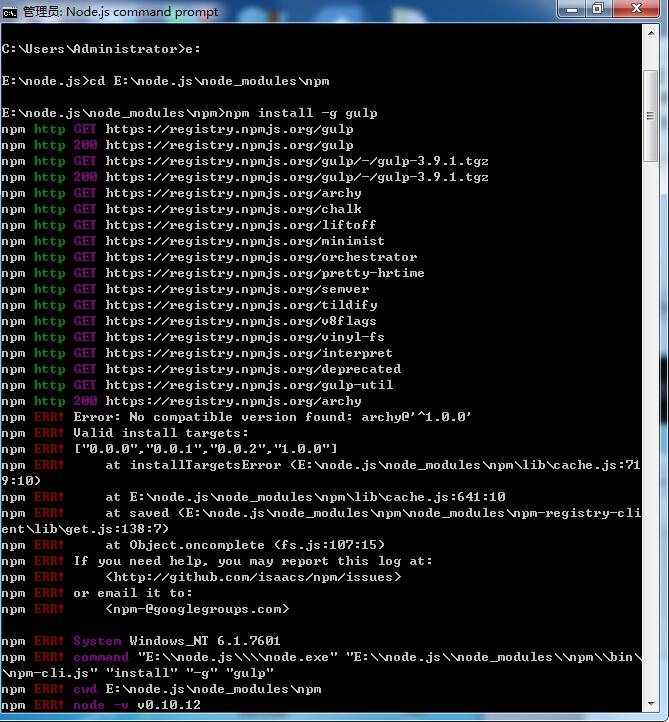
# 常见错误

## 没有用管理员权限打开



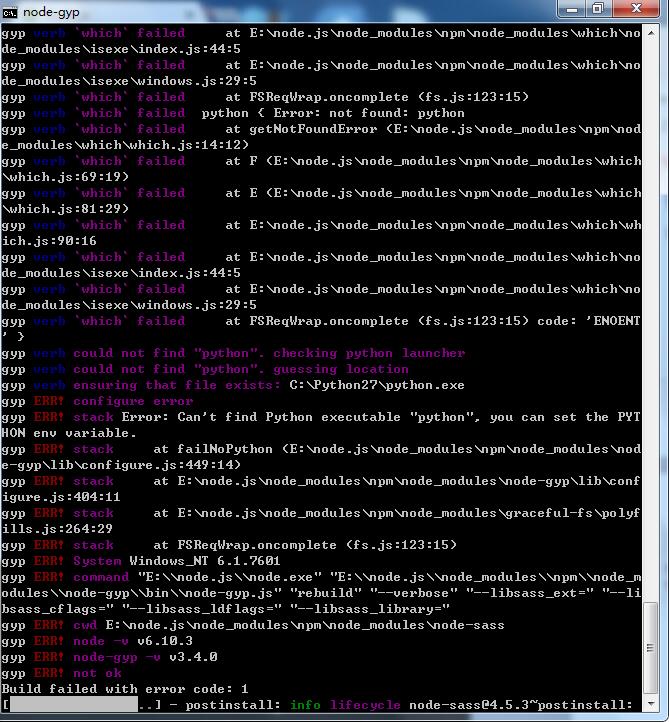
解决方案：用管理员权限打开命令行。

## 版本过低



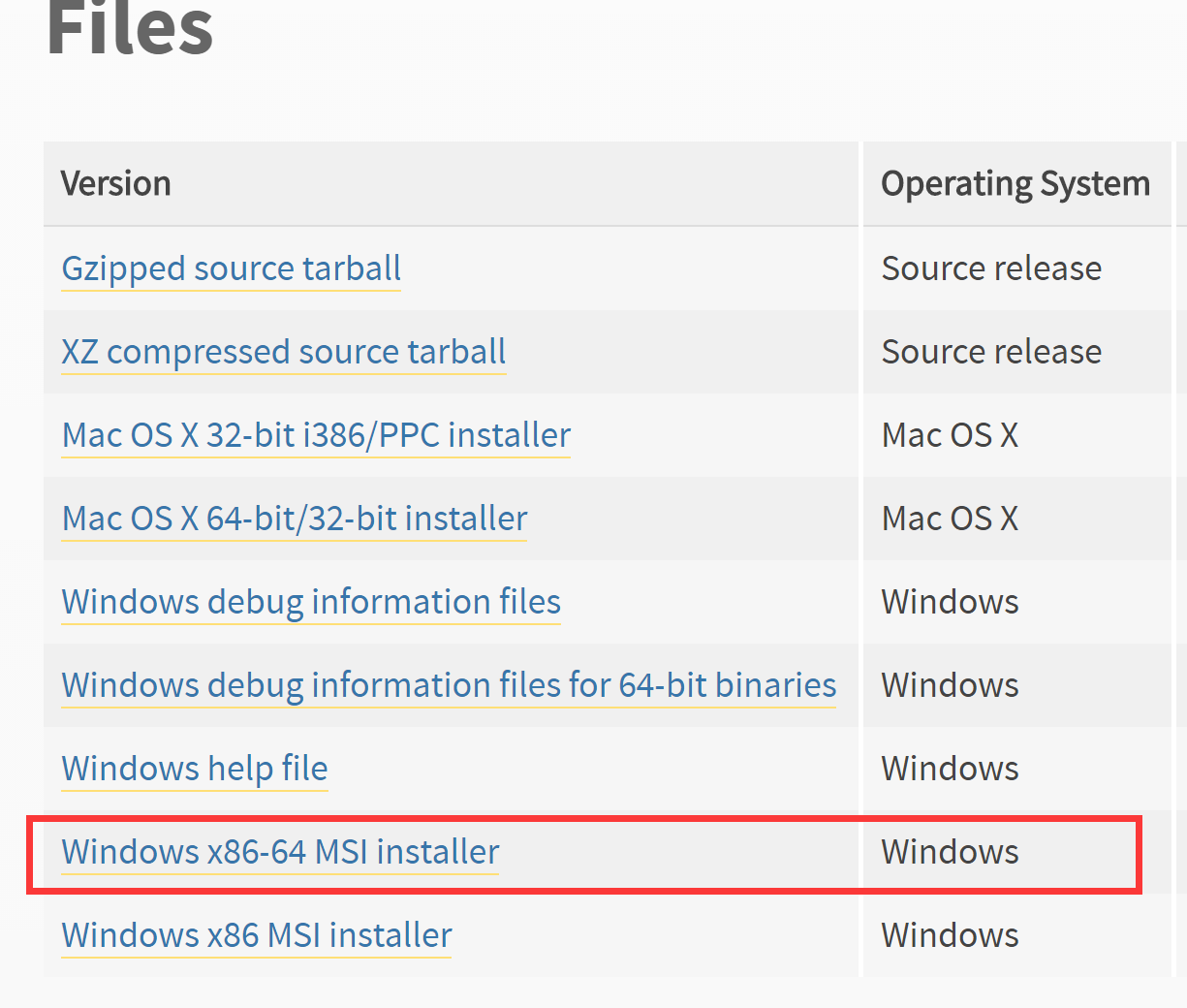
解决方案：安装高版本node.js

## Could not find “python”

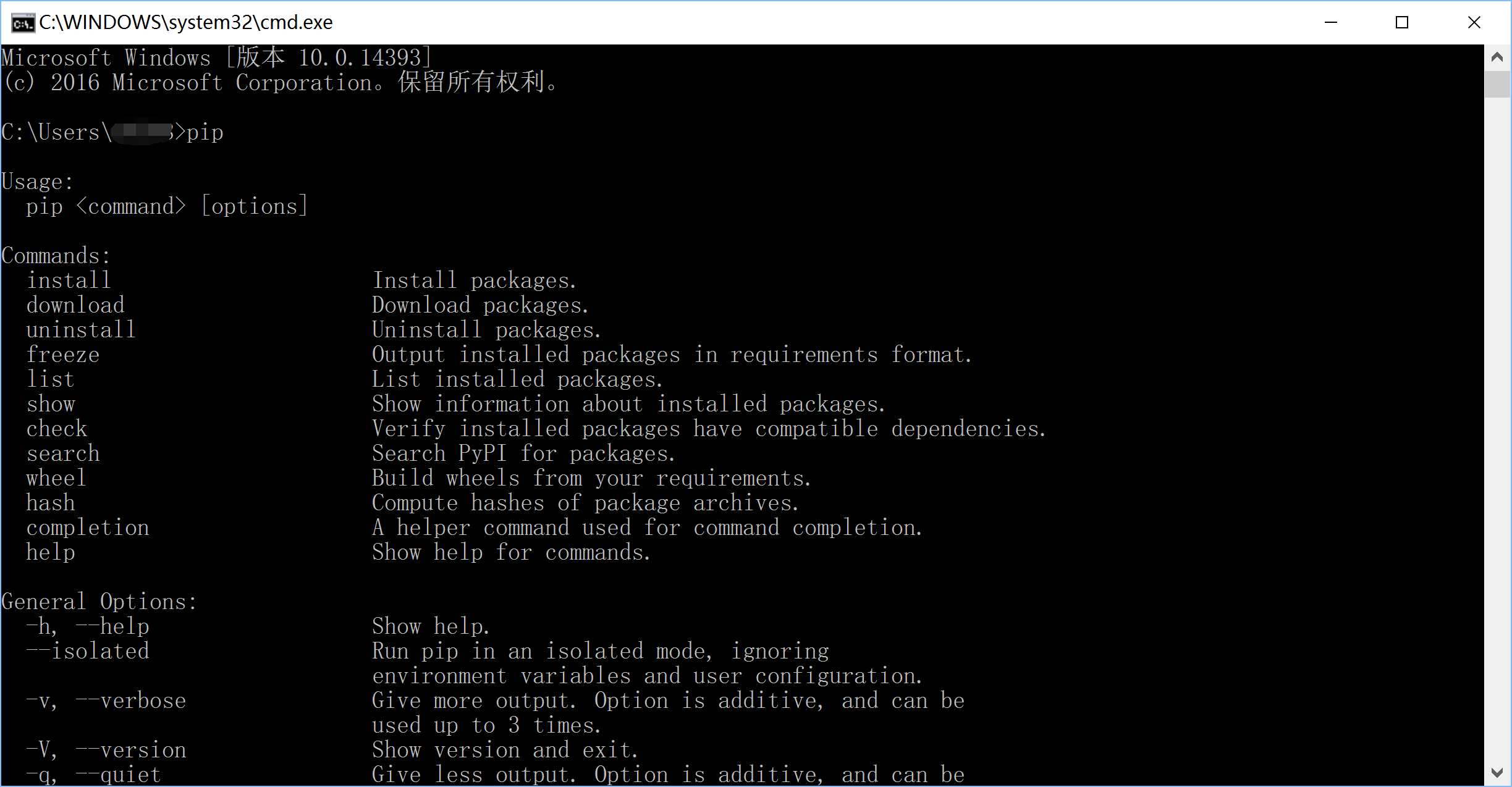


下载python

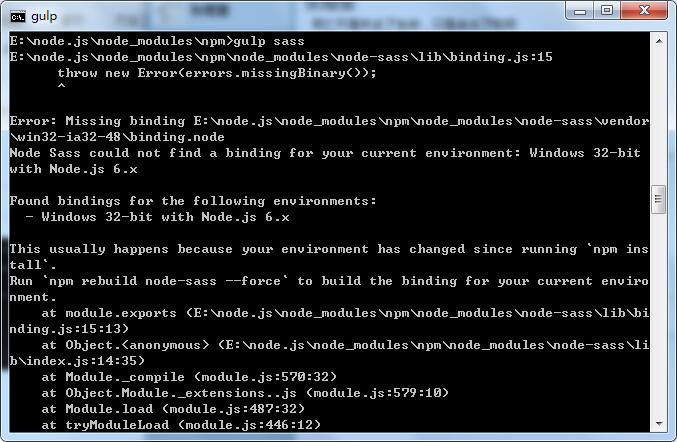
下载地址址：https://www.python.org/downloads/release/python-2713/



安装成功之后验证



## 32位版本报错



解决方案：npm rebuild node-sass

因为以前使用Node.js安过装node-sass 这通常是因为您的环境自运行以来发生变化，运行npm rebuild node-sass以构建您当前环境的绑定