# 1.起步

1. 此自动化工具是基于nodejs的，所以需要在[nodejs官网](http://nodejs.cn/download/)下载nodejs，安装路径按照默认路径装在c盘即可。

2. 安装完以后在命令行（git bash也可以）输入npm i -g gulp（使用npm可能会比较慢，推荐使用[cnpm](https://npm.taobao.org/)）

3. 新建一个文件夹，将gulpfile.js和package.json复制到文件夹根目录下，shift+右键在此处打开命令后（安装有git bash的直接右键git bash here）在命令行中输入npm（或cnpm） i。

4. 待所有东西安装完成后命令行输入gulp dev进入开发模式；开发完毕后即项目即将发布以后退出开发模式在命令行中输入gulp build将代码和图片进行压缩。

**推荐：在每次退出开发模式时运行一次gulp build，养成压缩代码的习惯。**

**注意：**

1. **gulp build命令会把dist文件夹删除，然后重新输出新的dist目录，所以不要手动在dist目录里修改增加删除文件。**
2. **不要把项目下的node\_moudles发布到服务器上**
3. **如需修改gulpfile.js文件建议备份一份，然后修改。**

# 目录结构

目录结构统一以一下结构作为标准

Project（你的项目）

|-dist（工具生成的文件夹，**不需要手动创建**）

|-node\_moudles（npm安装的依赖包，**不需要手动创建**）

|-src（项目的源码，也就是你写的代码都放到这里）

|-html（html文件存放的位置）

|-css（css或者scss或者less文件存放的位置）

|-js（js文件存放的位置）

|-images（图片存放的位置）

|-sprite（雪碧图切片存放位置）

|-static（静态文件存放位置，例如网站头像，字体库，阿里图标库）

|-gulpfile.js（自动化构建工具配置文件，**复制进来的**）

|-package.json（依赖包配置文件，**复制进来的**）

# 自动化工具任务说明

## Html

在开发模式下html会直接被复制到dist的html文件夹里，生产模式（也就是gulp build）下会被压缩放到dist的html文件夹里。

**注意：外联css只允许以下书写格式（双引号可换成单引号，推荐第三种书写方式）：**

**<link rel=“stylesheet” href=”../css/css文件” />**

**<link rel=“stylesheet” href=”../css/css文件”/>**

**<link rel=“stylesheet” href=”../css/css文件”>**

**<link rel=“stylesheet” type=”text/css” href=”../css/css文件” />**

**<link rel=“stylesheet” type=”text/css” href=”../css/css文件”/>**

**<link rel=“stylesheet” type=”text/css” href=”../css/css文件”>**

## Css

一些css3属性（例如display:flex）会自动添加浏览器前缀，所以在源码书写时无需多余添加浏览器前缀。其他的功能与html的类似，不再赘述。

## Scss/Less等一些css预处理语言（没学过的同学请直接跳过）

本自动化工具目前只支持scss的预编译，如果学习less的同学或者想用less的同学请自行配置gulpfile.js。使用scss作为css预编译的同学只需将gulpfile.js文件里28-30行和105行代码注释去掉即可。

## Js

正常功能与html相似，不多赘述

**注意：在外联外部js时只允许以下书写格式（双引号可换成单引号，推荐第二种书写方式）：**

**<script type="text/javascript" src=”../js/index.js”></script>**

**<script src=”../js/index.js”></script>**

## Js es6/es7/es8...语法（没学过的同学请跳过）

本自动化工具内置es6/es7语法编译，如有需要的同学请将gulpfile.js文件里41-43和119-121行代码注释去掉即可。

如果需要用到类似Promise等这类新的构造函数请在html文件里适当位置粘贴如下代码：<script src="https://cdn.bootcss.com/babel-core/5.8.38/browser-polyfill.js"></script>，当然这个方法不是最佳的，有能力的同学可以去研究研究[babel-runtime](http://babeljs.io/docs/plugins/transform-runtime/)。

## Image

大致功能与html相似，不多赘述。

另外，使用雪碧图（精灵图）的同学请把雪碧图（精灵图）碎片放到src/images/sprite目录下，css输出输出到dist/css/sprite文件下（如使用scss或者less预处理语言的将对应文件输出到src/css/sprite目录下，文件名命名以\_开头）并自行配置gulpfile.js文件。

## Static

这个任务只会将src/static文件夹复制到dist/static，没什么可说的。

## Browser

这个任务只会在开发模式下运行，就是只要有文件改动就会实时地体现在浏览器上。这个任务运行时会在本地3000端口开启一个服务器，并自动打开浏览器，运行前，请务必保证3000端口没被占用。