在项目开发过程中，总会牵扯到接口文档的设计与编写，之前使用的都是office工具，写一个文档，总也是不够漂亮和直观。好在git上的开源大神提供了生成文档的工具，so来介绍一下！
该工具是Nodejs的模块，请务必在使用前安装好nodejs环境！

工具名称：apiDoc
Git地址：https://github.com/apidoc/apidoc
项目地址：http://apidocjs.com/
样例项目：http://apidocjs.com/example_basic/
apoDoc是从源码的注释中生成RestFul api 文档，样子还是蛮漂亮的……

> 自己写了的ApiDoc文档自动生成工具，避免每次写完后手动执行生成代码
> http://blog.csdn.net/soslinken/article/details/50476384

支持的注释样式:
JavaDoc-Style

```
/**
 * This is a comment.
 */
```

CoffeeScript

```
###
This is a comment.
###
```

Elixir

```
@apidoc """
This is a comment.
"""
```

Erlang（%不是必须的）

```
%{
% This is a comment.
%}
```

Perl (Doxygen)

```
#**
# This is a comment.
#*
```

Python

```
"""
This is a comment.
"""
```

Ruby

```
=begin
This is a comment.
=end
```

安装apiDoc

```
npm install apidoc -g
```

使用npm命令安装apidoc即可!

使用方式:
在命令行中输入

```
apidoc -f ".*\\.js$" -f ".*\\.java$" -i myapp/ -o apidoc/ -t mytemplate/
```

参数说明:
-f 文件过滤
使用正则表达式，表示哪些文件需要本转换，不设置的情况下，默认为.cs .dart .erl .go .java .js .php .py .rb .ts 后缀的文件。

-i 代码文件夹

-o 输出Api文档的路径

-t 使用模板文件的路径，可以自定义输出的模板

常用的命令格式如下:

```
apidoc -i myapp/ -o apidoc/
```

配置
无package.json文件时，需要在代码文件夹的根目录下，创建apidoc.json文件。

```
{
  "name": "example",
  "version": "0.1.0",
  "description": "apiDoc basic example",
  "title": "Custom apiDoc browser title",
  "url" : "https://api.github.com/v1"
}
```

在该文件中即可配置apidoc

有package.json文件时，在package.json文件中添加”apidoc”: { }属性即可。

```
{
  "name": "example",
  "version": "0.1.0",
  "description": "apiDoc basic example",
  "apidoc": {
    "title": "Custom apiDoc browser title",
    "url" : "https://api.github.com/v1"
  }
}
```

配置属性如下:
name：项目名称
version：项目版本
description：项目介绍
title：浏览器显示的标题内容
url：endpoints的前缀，例如https://api.github.com/v1
sampleUrl：如果设置了，则在api文档中出现一个测试用的from表单
header
title：导航文字包含header.md文件
filename：markdown-file 文件名
footer
title：导航文字包含header.md文件
filename：markdown-file 文件名
order：用于配置输出 api-names/group-names 排序，在列表中的将按照列表中的顺序排序，不在列表中的名称将自动显示。

模板的配置:
在apidoc.json中或在package.json中添加template属性，将对模板进行特殊设置
forceLanguage： 生成指定语言的文档，简体中文仅需设置”zh-cn”，支持的语

言：https://github.com/apidoc/apidoc/tree/master/template/locales
withCompare：是否启用与旧版本的比较功能，默认为true
withGenerator：是否输出生成信息，默认为true
jQueryAjaxSetup：设置Ajax请求的默认值,参见http://api.jquery.com/jquery.ajaxsetup/

我使用的配置如下:

```
{
  "name": "测试",
  "version": "0.0.1",
  "description": "API文档测试",
  "title": "API文档测试",
  "url" : "http://xxxxxxx",
  "sampleUrl" : "http://xxxxxxxx",
  "template":{
    "forceLanguage":"zh-cn"
  }
}
```

先来个demo试试：
在myapp文件夹下创建example.java

```
/**
 * @api {get} /user/:id Request User information
 * @apiName GetUser
 * @apiGroup User
 *
 * @apiParam {Number} id Users unique ID.
 *
 * @apiSuccess {String} firstname Firstname of the User.
 * @apiSuccess {String} lastname  Lastname of the User.
 *
 * @apiSuccessExample Success-Response:
 *     HTTP/1.1 200 OK
 *     {
 *       "firstname": "John",
 *       "lastname": "Doe"
 *     }
 *
 * @apiError UserNotFound The id of the User was not found.
 *
 * @apiErrorExample Error-Response:
 *     HTTP/1.1 404 Not Found
 *     {
 *       "error": "UserNotFound"
 *     }
 */
```

之后执行

```
apidoc -i myapp/ -o apidoc/
```

打开apidoc文件夹下的index.html即可看到效果。赞赞哒……

重头戏来了，下面要讲解一下apiDoc的注释都是什么含义

@api

```
@api {method} path [title]
```

使用该注释的才能被识别成为文档的一块内容

> method：请求方法，参见https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol#Request_methods
> path：请求路径
> title(可选)：标题

Example:

```
/**
 * @api {get} /user/:id
 */
```

@apiDefine

```
@apiDefine name [title]
                    [description]
```

定义重复内容，供其他模块引用

> name：重复模块名
> title（可选）：标题
> description（可选）：说明

Example1:

```
/**
 * @apiDefine MyError
```

```
 * @apiError UserNotFound The <code>id</code> of the User was not found.
 */

/**
 * @api {get} /user/:id
 * @apiUse MyError
 */
```

Example2:

```
/**
 * @apiDefine admin User access only
 * This optional description belong to to the group admin.
 */

/**
 * @api {get} /user/:id
 * @apiPermission admin
 */
```

@apiDescription

```
@apiDescription text
```

设置多行说明

> text：多行说明内容

Example

```
/**
 * @apiDescription This is the Description.
 * It is multiline capable.
 *
 * Last line of Description.
 */
```

@apiError

```
@apiError [(group)] [{type}] field [description]
```

请求错误参数

（group）（可选）：参数将以这个名称分组，不设置的话，默认是Error 4xx
{type}（可选）：返回值类型，例如：{Boolean}, {Number}, {String}, {Object}, {String[]}
field：返回值字段名称
descriptionoptional（可选）：返回值字段说明

Example:

```
/**
 * @api {get} /user/:id
 * @apiError UserNotFound The <code>id</code> of the User was not found.
 */
```

@apiErrorExample

```
@apiErrorExample [{type}] [title]
                 example
```

错误返回信息样例。

type（可选）：返回内容的格式
title（可选）：标题
example：样例，可以是多行文本

Example:

```
/**
 * @api {get} /user/:id
 * @apiErrorExample {json} Error-Response:
 *     HTTP/1.1 404 Not Found
 *     {
 *       "error": "UserNotFound"
 *     }
 */
```

@apiExample

```
@apiExample [{type}] title
            example
```

请求*Api*的测试样例

{type}（可选）：请求方式类型

> title：标题
> example：样例，可以是多行文本

Example:

```
/**
 * @api {get} /user/:id
 * @apiExample {curl} Example usage:
 *     curl -i http://localhost/user/4711
 */
```

@apiGroup

```
@apiGroup name
```

定义*Api*的分组

> name：组名称，也是导航的标题

Example:

```
/**
 * @api {get} /user/:id
 * @apiGroup User
 */
```

@apiHeader

```
@apiHeader [(group)] [{type}] [field=defaultValue] [description]
```

定义*head*参数

> (group)（可选）：分组
> {type}（可选）：类型，例如：{Boolean}, {Number}, {String}, {Object}, {String[]}
> field：变量名
> [field]：定义变量，并标记变量是可选项
> =defaultValue（可选）：默认值
> description（optional）：变量说明

Examples:

```
/**
 * @api {get} /user/:id
 * @apiHeader {String} access-key Users unique access-key.
 */
```

@apiHeaderExample

```
@apiHeaderExample [{type}] [title]
                  example
```

*head*参数样例

> {type}（可选）：请求方式类型
> title：标题
> example：样例，可以是多行文本

Example:

```
/**
 * @api {get} /user/:id
 * @apiHeaderExample {json} Header-Example:
 *     {
 *       "Accept-Encoding": "Accept-Encoding: gzip, deflate"
 *     }
 */
```

@apiIgnore

```
@apiIgnore [hint]
```

忽略不发布到文档

> hint（可选）：不发布的原因

Example:

```
/**
 * @apiIgnore Not finished Method
 * @api {get} /user/:id
 */
```

@apiName

```
@apiName name
```

定义*api*名称

> name：api名称

Example:

```
/**
 * @api {get} /user/:id
 * @apiName GetUser
 */
```

@apiParam

```
@apiParam [(group)] [{type}] [field=defaultValue] [description]
```

定义请求*Api*的参数

> (group)（可选）：分组
> {type}（可选）：类型，例如：{Boolean}, {Number}, {String}, {Object}, {String[]}
> {type{size}}（可选）：类型限定长度，例如：{string{..5}} 限定最大长度为5个字符
> {string{2..5}} 限定最短2个字符，最长5个字符
> {number{100-999}} 限定数字在100-999之间
> {type=allowedValues}（可选）：类型限定值，例如：{string="small"}限定只允许传递small字符，
> {string="small","huge"} 限定只允许传递small或huge字符，
> {number=1,2,3,99} 限定只允许传1，2，3，99这四个数字
> field：变量名
> [field]：定义变量，并标记变量是可选项
> =defaultValue（可选）：默认值
> description（optional）：变量说明

Examples:

```
/**
 * @api {get} /user/:id
 * @apiParam {Number} id Users unique ID.
 */

/**
 * @api {post} /user/
```

```
 * @apiParam {String} [firstname]  Optional Firstname of the User.
 * @apiParam {String} lastname     Mandatory Lastname.
 * @apiParam {String} country="DE" Mandatory with default value "DE".
 * @apiParam {Number} [age=18]     Optional Age with default 18.
 *
 * @apiParam (Login) {String} pass Only logged in users can post this.
 *                                 In generated documentation a separate
 *                                 "Login" Block will be generated.
 */
```

@apiParamExample

```
@apiParamExample [{type}] [title]
                 example
```

请求参数样例

> {type}（可选）：请求方式类型
> title：标题
> example：样例，可以是多行文本

Example:

```
/**
 * @api {get} /user/:id
 * @apiParamExample {json} Request-Example:
 *     {
 *       "id": 4711
 *     }
 */
```

@apiPermission

```
@apiPermission name
```

定义接口权限

> name：权限名称

Example:

```
/**
 * @api {get} /user/:id
```

```
 * @apiPermission none
 */
```

@apiSampleRequest

```
@apiSampleRequest url
```

设置测试请求*form*，如果在*apidoc.json*或*package.json*中设置过了*sampleUrl*，则默认请求地址为，*sampleUrl+apipath*，设置这个标签则重写测试请求路径

> url：测试地址，
> 样例
> @apiSampleRequest http://www.example.com 改变测试地址
> @apiSampleRequest /my_test_path 在apipath中加前缀
> @apiSampleRequest off 不显示测试请求from

Examples:
默认请求的地址是http://api.github.com/user/:id

```
Configuration parameter sampleUrl: "http://api.github.com"
/**
 * @api {get} /user/:id
 */
```

需要将测试地址修改为http://test.github.com/some_path/user/:id，则

```
Configuration parameter sampleUrl: "http://api.github.com"
/**
 * @api {get} /user/:id
 * @apiSampleRequest http://test.github.com/some_path/
 */
```

当地址需要请求http://api.github.com/test/user/:id则

```
Configuration parameter sampleUrl: "http://api.github.com"
/**
 * @api {get} /user/:id
 * @apiSampleRequest /test
 */
```

不需要测试请求from

```
Configuration parameter sampleUrl: "http://api.github.com"
/**
 * @api {get} /user/:id
 * @apiSampleRequest off
 */
```

没有设置sampleUrl时需要显示测试请求from，并设置请求http://api.github.com/some_path/user/:id

```
Configuration parameter sampleUrl is not set
/**
 * @api {get} /user/:id
 * @apiSampleRequest http://api.github.com/some_path/
 */
```

@apiSuccess

```
@apiSuccess [(group)] [{type}] field [description]
```

请求成功返回的参数

> （group）（可选）：参数将以这个名称分组，不设置的话，默认是Error 4xx
> {type}（可选）：返回值类型，例如：{Boolean}, {Number}, {String}, {Object}, {String[]}
> field：返回值字段名称
> descriptionoptional（可选）：返回值字段说明

Example:

```
/**
 * @api {get} /user/:id
 * @apiSuccess (200) {String} firstname Firstname of the User.
 * @apiSuccess (200) {String} lastname  Lastname of the User.
 */
```

@apiSuccessExample

```
@apiSuccessExample [{type}] [title]
                   example
```

请求成功返回数据样例

> {type}（可选）：请求方式类型
> title：标题

> example：样例，可以是多行文本

Example:

```
/**
 * @api {get} /user/:id
 * @apiSuccessExample {json} Success-Response:
 *     HTTP/1.1 200 OK
 *     {
 *       "firstname": "John",
 *       "lastname": "Doe"
 *     }
 */
```

@apiUse

```
@apiUse name
```

使用在*@apiDefine*中定义的内容

> name：在@apiDefine定义的name

Example:

```
/**
 * @apiDefine MySuccess
 * @apiSuccess {string} firstname The users firstname.
 * @apiSuccess {number} age The users age.
 */

/**
 * @api {get} /user/:id
 * @apiUse MySuccess
 */
```

@apiVersion

```
@apiVersion version
```

设定*Api*的版本号

> version：版本号，格式(major.minor.patch)

Example:

```
/**
 * @api {get} /user/:id
 * @apiVersion 1.6.2
 */
```

以上就是apiDoc的介绍！谢谢各位看官……

附上一个样例

user.java

```
/**
 * @api {POST} /register 注册用户
 * @apiGroup Users
 * @apiVersion 0.0.1
 * @apiDescription 用于注册用户
 * @apiParam {String} account 用户账户名
 * @apiParam {String} password 密码
 * @apiParam {String} mobile 手机号
 * @apiParam {int} vip = 0  是否注册Vip身份 0 普通用户 1 Vip用户
 * @apiParam {String} [recommend] 邀请码
 * @apiParamExample {json} 请求样例:
 *                ?account=sodlinken&password=11223344&mobile=13739554137&vip=0&recommend=
 * @apiSuccess (200) {String} msg 信息
 * @apiSuccess (200) {int} code 0 代表无错误 1代表有错误
 * @apiSuccessExample {json} 返回样例:
 *                {"code":"0","msg":"注册成功"}
 */

 /**
 * @api {POST} /login 用户登录
 * @apiGroup Users
 * @apiVersion 0.0.1
 * @apiDescription 用于用户登录
 * @apiParam {String} userName 用户名
 * @apiParam {String} password 密码
 * @apiParamExample {json} 请求样例:
 *                ?userName=张三&password=11223344
 * @apiSuccess (200) {String} msg 信息
 * @apiSuccess (200) {String} code 0 代表无错误 1代表有错误
 * @apiSuccess (200) {String} user 用户信息
 * @apiSuccess (200) {String} userId 用户id
 * @apiSuccessExample {json} 返回样例:
 *                {"code":"0","msg":"登录成功","userId":"fe6386d550bd434b8cd994b58c3f8075"}
 */

 /**
```

```
     * @api {GET} /users/:id 获取用户信息
     * @apiGroup Users
     * @apiVersion 0.0.1
     * @apiDescription 获取用户信息
     * @apiSuccess (200) {String} msg 信息
     * @apiSuccess (200) {int} code 0 代表无错误 1代表有错误
     * @apiSuccess (200) {String} name 真实姓名
     * @apiSuccess (200) {String} mobile 手机号
     * @apiSuccess (200) {String} birthday 生日
     * @apiSuccess (200) {String} email 邮箱
     * @apiSuccess (200) {String} summary 简介
     * @apiSuccess (200) {String} recommendCode 我的推荐码
     * @apiSuccess (200) {String} idCardNo 身份证号
     * @apiSuccess (200) {String} memberState 会员状态 0普通用户 1VIP 2账户冻结
     * @apiSuccess (200) {String} address 家庭住址
     * @apiSuccess (200) {String} money 账户现金
     * @apiSuccessExample {json} 返回样例:
     * {
     *   "code": 0,
     *   "msg": "",
     *   "name": "真实姓名",
     *   "mobile": 15808544477,
     *   "birthday": "1990-03-05",
     *   "email": "slocn@gamil.com",
     *   "summary": "简介",
     *   "recommendCode": "我的推荐码",
     *   "idCardNo": "身份证号",
     *   "memberState": 1,
     *   "address": "家庭住址",
     *   "money": "30.65"
     * }
     */


    /**
     * @api {POST} /users/:id 修改(完善)用户信息
     * @apiGroup Users
     * @apiVersion 0.0.1
     * @apiDescription 修改(完善)用户信息
     * @apiParam (200) {String} [name] 真实姓名
     * @apiParam (200) {String} [mobile] 手机号
     * @apiParam (200) {String} [birthday] 生日
     * @apiParam (200) {String} [email] 邮箱
     * @apiParam (200) {String} [summary] 简介
     * @apiParam (200) {String} [idCardNo] 身份证号
     * @apiParam (200) {String} [address] 家庭住址
     * @apiSuccess (200) {String} msg 信息
     * @apiSuccess (200) {int} code 0 代表无错误 1代表有错误
     * @apiSuccessExample {json} 返回样例:
     *                {"code":"0","msg":"修改成功"}
```

```
     */
```

这个样例可以直接生成，生成后即可看到apidoc的效果

2018年1月22日 更新

看到有很多人说中文乱码的事情，今天特别更新说明一下

主要是 @apiGroup 不支持utf-8 字符串。仅支持ascii码。所以很多使用者要么只能用英文说明，要么就放弃这个工具。

在官方有个办法可以实现utf-8字符串放置在@apiGoup 中。
代码如下

```
    /**
     * @apiDefine userApiStr 用户接口文档
     */

    /**
     * @api {get} /user/:id Request User information
     * @apiName GetUser
     * @apiGroup userApiStr
    */
```

说明
1、@apiDefine 必须放置在 /* ...../中，也必须与引用变量的地方分开。
2、@apiGroup 后 放置的是 @apiDefine 定义的 变量名

其实本质上是定义了一个引用变量，这个变量支持utf-8

来源： http://blog.csdn.net/soslinken/article/details/50468896