**wn(蜗牛)**

**背景：**

小强是一位刚入职的蜗牛前端攻城狮，打完第一天的酱油后，第二天pm终于坐不住了，他找到了小强说：小强啊，现在有个紧急任务啊。。。。Balabala。。。。。你赶紧做吧，下班前要上哦^ ^，说完淫笑的走了。还没从酱油状态反应过来的小强，不得不赶紧问旁边的同事，湿胸，任务资源怎么下载啊，你们一般用的什么编辑器啊？。。。在湿胸的帮助下，小强总算是从ftp里拷贝到了任务的设计图，拿到设计图一看，小强的第一反应就是：“”



TM这么复杂的任务，下班前就上线？开什么玩笑？经过一番心里抱怨后，小强还是无奈的开始看起了设计图，额.......**轮播图片**，**弹窗**，**注册**，**登陆**，功能还真是齐全啊，这TM还是专题吗，还好都是静态的，不需要调接口。时间不多了，赶紧开始搞起来吧，不然第一天就要加班，小强一边想一边打开了他常用的编辑器。于是小强开始了在蜗牛的开发之旅，他打开ps，开始切设计图上需要是图片的地方，保存后小强停住了，项目应该新建到哪啊，于是他向旁边的湿胸求助，湿胸告诉他：。。。。Balabala。。。。。，得到湿胸的帮助，小强总算是找到了当前任务的项目所在地，于是他理所当然的在新建的项目根目录里建好了css、js、images三个文件夹，然后在根目录创建了一个空白的html，又看了看设计图后，小强开始了的敲代码过程，他在设计图和浏览器效果之间来回切换...da..da..da....da...da..da......小强的手指在键盘上飞快的敲打着，仿佛在敲打着自己的人生。终于小强在集中精力的对键盘进行一番狂轰乱炸后，把结构样式写好了，看见设计图在浏览器里的再现，小强还是颇有成就感，正在小强面带微笑的端详着浏览器里自己的作品时，只听周围开始了一阵骚动，小强很奇怪正准备转过身问旁边的湿胸时，发现旁边湿胸已经不见了，起身一看才发现，食堂已经排起了长长的队伍，而站在第一个的正是旁边的湿胸**- -。**

..............中场休息..............

总算吃完多的小强，急匆匆的又回到了自己座位，小强心想下午得抓紧啊，得赶紧把js功能写好，第一个轮播，恩......这个应该不难，网上有好多这样的插件，我去搜一个就行了，............searching...........，经过一番搜索后，小强总算是找到一个长得比较像的，倒腾了一番后，轮播功能总算是做好了，下一个弹窗，这时小强忽然灵光一现，这个功能以前好像做过啊，恩......是做过，就是想不起来是哪个项目了，不行，得好好回想一下，一旦找到就不用再写了呀，那多好，于是小强开始在他记忆的大海里开始裸泳起来............回想.............，突然他大腿一拍，终于想起是什么时候做过了，于是他开始疯狂的在他云盘寻找起来，......dir xxx不对.....dir xxx2不对......dir xxx3......dir xxxN.......不对，ding！哈哈，终于找到了！小强带着中了彩票似的心情，赶紧把找到的弹窗代码贴上去改好，弹窗也总算是好了。NEXT，注册，登陆，这时小强眉头开始皱起来了，这东西咋搞啊，感觉很麻烦的样子啊，想到这，小强用无助的眼神转向旁边带着耳机的湿胸，弱弱的问了一句：“湿胸这个注册、登陆怎么搞啊？”“奥，注册是吧，我们有现成的模块啊，你去随便找个有注册、登陆的游戏官网，扒下来就行了”，说完湿胸又带上了耳机，继续追寻他的音乐梦想去了。这答案倒是真够简约的啊，小强不禁心想，哎，算了，也不指望他给我详细讲了，去找个官网看看吧，.........研究xxx官网注册、登陆..........，经过小强大脑的高速运转后，小强总算是搞懂了注册、登陆怎么扒下来，把剩下的注册、登陆做好了。看看时间，距离下班时间已经不多了，提交给测试后，小强得到了片刻的放松，正在拿着杯子喝水的小强，正心想，要是测试过了的话说不定还能准时下班呢，嘿嘿，这时他突然看见屏幕右下方，测试发来一堆消息：小强，你这个页面在ie6全乱了啊？还有你title，description，keywords怎么都没有啊？统计代码呢，通用头尾呢，也没加？还有注册的时候gameId也不对啊，你那是另一个游戏的gameId啊？

噗。。。。小强差点没把喝的水吐到屏幕上，what the fuck！你在说什么呀？我怎么一句都听不懂啊，title，description，keywords应该怎么写啊，没人告诉我啊，统计代码，通用头尾怎么加啊，gameId又是什么东东啊，在哪查啊？小强正在想这些问题应该问谁的时候，PM终于出现了，“搞得怎么样了啊？小强，上午给你的任务下班前要上线的哦！”，“啊！下班前肯定是上不了了啊，做是做完了，不过还有一大堆问题啊！你赶紧告诉我title，des.......”，没等小强说完，PM突然打断他说：“等等，这个专题好像之前xxx做过一个类似的啊，你直接改改就完了呀！哪来这么多问题啊！”，这时的小强瞬间石化，时间仿佛永远凝固在了那一刻.............



**简介：**

wn是一套针对目前蜗牛前端业务需求的集成解决方案。主要解决目前部门在前端开发过程中，代码复用率低，开发缺乏统一规范，代码维护沟通困难，缺乏一个便捷、高效、统一的开发、分享环境，项目每次都是从0开始，工作流程不够标准化，统一化，自动化，还有上线、回滚不够顺畅等一系列问题。希望用这套解决方案，能建立一套快速应对需求的机制，让前端部门能更快更轻松的变现需求方的需求，也让大家有个更好的开发环境。

**快速开始：**

安装：

安装nodejs，http://nodejs.org/和git，http://git-scm.com/download/

打开命令行

**$ npm install -g wn**



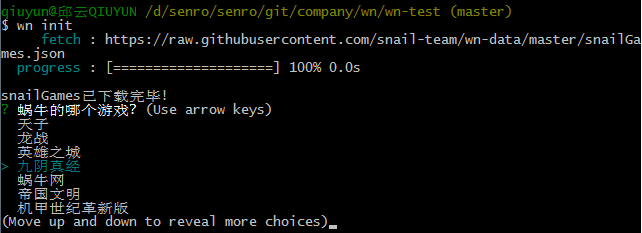
**创建项目：**

在一个空目录，右击选择git bash here，打开在该目录的命令行

**$ wn init**

根据提示选择相应的选项即可

1.用上下箭头选择游戏，默认指向《九阴真经》，如是新游戏或新项目没有被列出来，只需更新github/snail-team上，我们团队的wn-data仓库的snailGames.json即可。



2.选择项目类型，目前只做了两个项目类型的模板，以后可以根据需要无限添加



如果选择专题，则需要填写专题名称，用于seo自动拼接。



3.输入项目的英文名称，给spm打包或安装依赖模块用，默认是当前目录名



4. 提前安装需要的模块，目前为了测试默认安装jquery@1.8.3，当然可以重新填写，比如输入jquery arale-cookie@1.1.0，即代表安装jquery@stable arale-cookie@1.1.0，jquery的“stable”最新版，和arale-cookie的1.1.0版本；也可以直接敲空格，表示不安装任何模块，为空。如果项目模板里有package.json且里面的spm. dependencies属性有项目依赖模块，则会合并你输入的模块和项目本身配置的依赖模块，如果有模块冲突，如spm. dependencies里有依赖jquery@1.8.3，而你输入了一个jquery则只安装你输入的版本jquery@stable，最新版和其他不冲突模块。

注意：

用“@”符号连接版本。

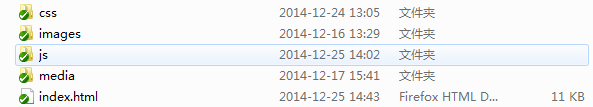
空格连接多个模块。



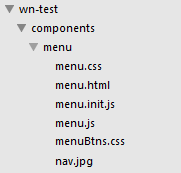
到此项目初始化完毕，如果没有报任何错误，刚才的空目录应该出现了以下文件：



看见这堆东西，估计你的第一反应是，目录这么复杂啊，哪有我原来的



目录结构清晰啊，我一看就知道哪些文件该放哪里，的确，上面的这种目录结构，只要不瞎，都知道要把文件放哪。只是我们为什么要改变这样的目录结构呢，有一天小强接到了一个任务，PM让他把xxx官网的导航条改一下，需要换个圣诞节主题的导航条。于是小强找到该官网的index的目录，然后他开始在index.html里寻找原来导航条的结构，改完后，继续去css目录找index.css里面的导航条样式，如果还有导航条还有js还得去项目的一大堆js里去找导航条对应的代码，还得去找到images里对应的导航条图片，找了半天后才找到所有要改的地方，由于目录里文件太多，代码太分散，有些地方怕改错了，还只能先注释掉，或者图片先复制备份一个之类的， 然后该项目里的注释无用的代码越来越多，无用的图片也越来越多。而如果我们的目录结构是这样的呢：



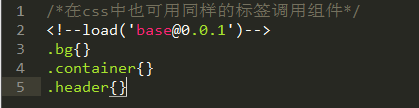
情况又会怎样。小强还是接到了刚才的任务，只是他去改导航条的时候，只需要在menu里改导航条的所有文件即可。这里的menu即是一个模块。什么是模块？我们把模块定义为一个由html、css、js、图片随机结合的文件目录。我们把模块的所有文件资源都放在一个目录里，且约定这样的模块定义规范：目录名就是该模块的名字，里面凡是和模块名同名的文件都默认为该模块的对应资源，比如menu里menu.css就是定义该menu模块的所有样式，menu.html就是定义了menu的所有结构，menu.js即定义了该模块的脚本代码，menu.init.js定义了该模块的初始化代码，图片就是该模块用到的图片。menuBtns.css是menu.css里的内部引用，这个用法下面具体会讲。也可用index定义该模块的所有资源，比如index.html、index.css、index.js、index.init.js他的调用优先级比组件名的文件更高。

然后通过里面的package.json去定义该模块的包信息，以便打包上传到包管理平台，一个包即一个模块。包管理平台目前有很多，比如npm（管理nodejs模块的），bower（google遵循component规范的包管理平台），spm（淘宝的以seajs为加载核心的静态资源包管理平台），我们这次就是采用的spm来管理所有的模块包，之所以采用spm是因为这个平台本身支持自己搭一个私服，并且配置也很方便，还有考虑到目前我们商城用的seajs加载，这样也算是一种兼容，以后商城的资源也都在这个平台，这样更好的整合了我们部门的资源，当然最重要的原因还是没时间去重新开发一个包管理平台了，但是现在的spm平台的展示方式，还不是我最初构想的样子，至少它应该有个模块缩略图的功能，这个功能在未来的不久就会完善上去。npm、bower、spm这些英文缩写看起来好像很陌生的样子，其实他们就是一个存放所有用户发布上去的包的网站而已，只是他们还得把这些包的必要信息展示出来，供大家检索。可能你会问，那我怎么使用我们定义的包呢，现在让我们看看刚才创建的项目文件的每个文件：



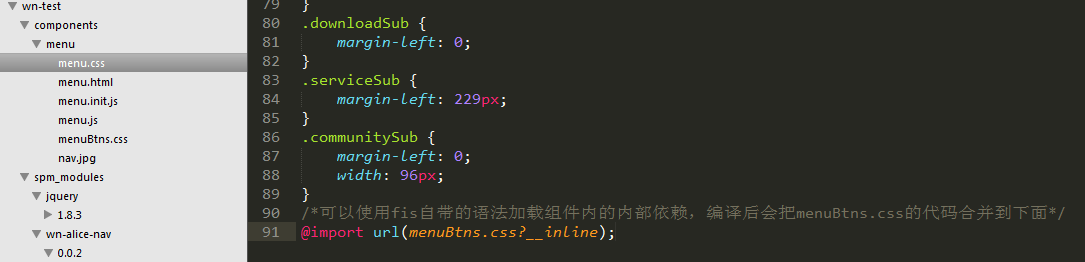
（项目文件解析图.jpg）

此外在样式里也可以用<!--load("xxx")-->调用模块，比如在static/index/index.css里



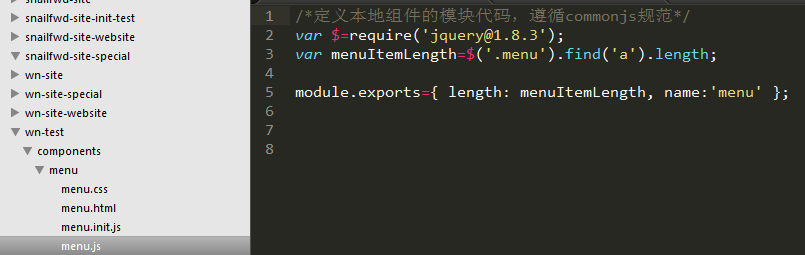
这样会把0.0.1版本的base模块的css资源调到这里来。

在模块里引用模块内部的css可以这样写：

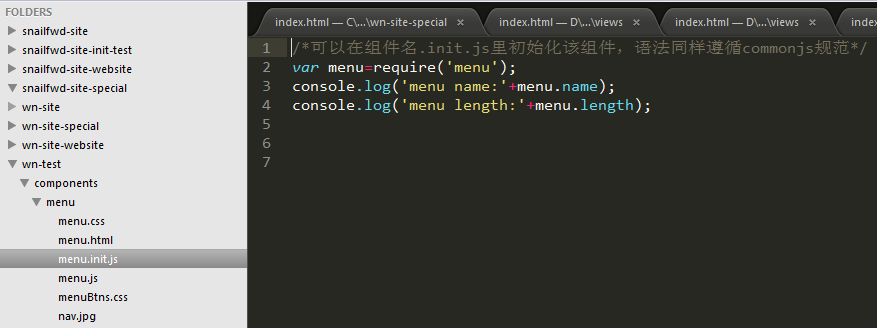


（模块内部引用css.jpg）

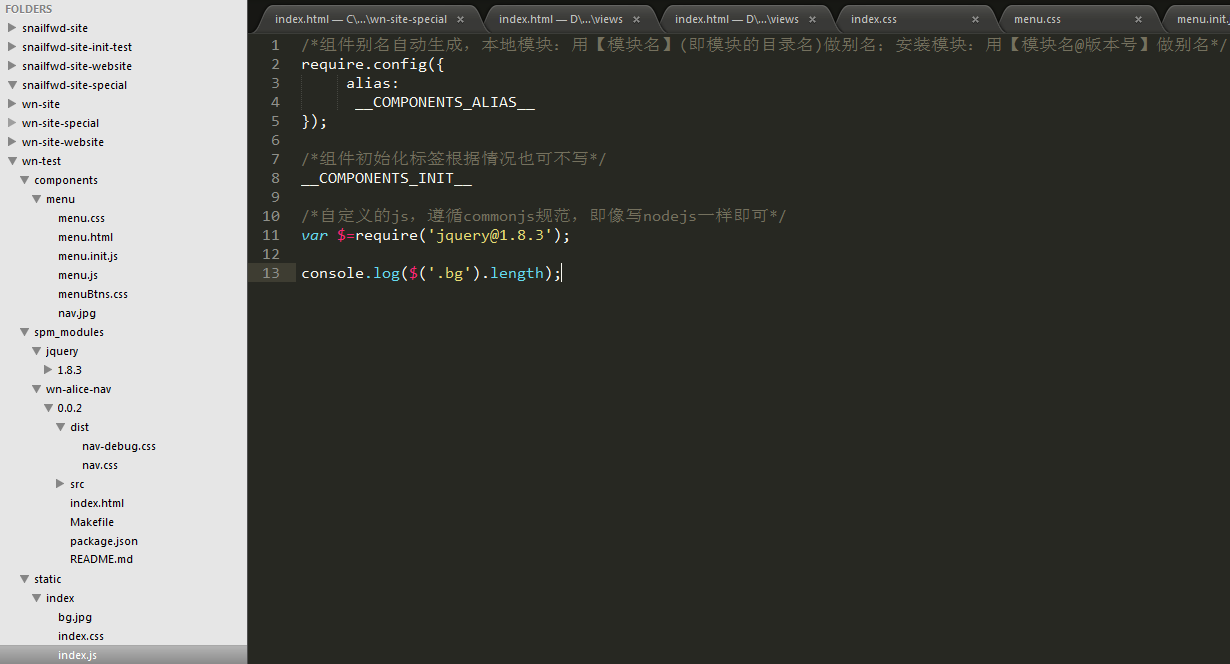
定义模块的js，比如menu.js，遵循commonjs规范，就像写nodejs一样（<http://javascript.ruanyifeng.com/nodejs/commonjs.html>）



在menu.init.js里调用menu.js，同样遵循commonjs规范，就像写nodejs一样



在static/index/index.js里，即index页面的入口文件里



（index.js.jpg）

到此你应该对wn的开发过程有个大体的了解了吧，其实说来说去就是在围绕一个模块化思想转，这也是目前比较先进、流行的思想。

下面来看看开发完项目后怎么预览呢？

很简单，只需在刚才目录命令行里输入**$ wn release** 即可



这个命令还有很多参数，通过不同参数的组合，可以实现不同的功能.

release是一个非常强大的命令，它的主要任务就是进行代码的 **编译** 与 **部署**，它的参数囊括了前端开发所需的各种基础功能：

添加 **--watch** 或 **-w** 参数，支持对项目进行增量编译，监听文件变化再触发编译

添加 **--live** 或 **-L** 参数（注意大写），支持编译后自动刷新浏览器。Liveload功能需要浏览器支持Web Socket功能，例如Chrome、Firefox、Safari等浏览器。

添加 **--dest [path|name]** 或 **-d** 参数，来指定编译后的代码部署路径，支持发布到 **本地目录、本地调试服务器目录、远程机器目录(需要配置)**，它与--watch参数配合使用，可以让你的代码保存就上传！而且--dest值支持逗号分隔，这也就意味着，你 **一次编译可以同时发布到本地以及多台远程机器上**！举几个栗子：

发布到wn server open目录下用于本地调试

wn release

# or

wn release --dest preview

发布到项目根目录的output目录下， 注意，这里的output其实是一个内置的部署配置名，而不是一个目录名。

wn release -d output

发布到相对 工作目录 的路径

wn release -d ../output

发布到绝对路径

wn release -d /home/work/ouput

# win

wn release -d d:/work/output

使用配置文件的 [deploy节点配置](http://fis.baidu.com/docs/api/fis-conf.html#deploy) 进行发布，此配置可将代码上传至远端

wn release -d remote

以上所有发布规则任意组合使用（一次编译同时上传到多台远端机器 & 项目根目录下的output & 调试服务器根目录 & 本地绝对路径）

wn releaes -d remote,qa,rd,output,preview,D:/work/output

添加 **--md5 [level]** 或 **-m [level]** 参数，在编译的时候可以对文件自动加md5戳，从此告别在静态资源url后面写?version=xxx的时代

添加 **--lint** 或 **-l** 参数，支持在编译的时候根据项目配置自动代码检查

添加 **--test** 或 **-t** 参数，支持在编译的时候对代码进行自动化测试

添加 **--pack** 或 **-p** 参数，对产出文件根据项目配置进行打包

添加 **--optimize** 或 **-o** 参数，对js、css、html进行压缩

添加 **--domains** 或 **-D** 参数，为资源添加domain域名

参数可以随意组合，没有顺序限制，比如：wn release -wLc 代表发布时文件监听，自动刷新，清楚缓存。值得一提的是，那个pack命令我们现在基本用不到，因为这次wn采用的是自动打包，同步加载模块的方式，所以暂时用不到那个pack参数，md5参数也暂时先不用，这个功能还需进一步测试，和规划，dest的远程部署，也暂时还没开放，还需要进一步的测试和规划。

当然编译后，还需要启动个服务器去查看，可以使用**wn server start**，启动一个本地服务器，服务器启动之后，它会自动检查环境，最后告诉你它监听了8080端口，这个时候，你的浏览器应该打开了一个调试服务器根目录的浏览页面，地址是**http://localhost:8080/**。并且该服务器支持php等后端语言。



用**wn server open**可以打开服务器的根目录



如果需要关闭服务器则用**$ wn server stop**即可，如果想要删除服务器那个目录，就必须先关闭服务器，不然服务器的进程会一直占用里面的server.log文件，以至于提示无法删除目录。



上面这段关于release、server命令的详细介绍是我从fis官网的拷贝，为了不产生误解，我把fis替换成了wn，因为我们这个工具是用wn命令调用的，也可去fis官网查看命令行的具体用法[http://fis.baidu.com/docs/api/cli.html#](http://fis.baidu.com/docs/api/cli.html)。

**wn install**

**用法：**

wn install [options] name[@version]

比如：wn install jquery@1.9.1 wn-9yin-nav@0.0.1

**功能：**

安装wn模块，默认从http://spmjs.io安装，可以运行$ spm config set registry http://[custom uri]改变到内网，是对spm install的封装。

**参数：**

-r, --registry <url> 设置源地址

-I, --input-directory <dir> 设置输入目录，默认当前目录

-O, --output-directory <dir>设置安装目录，默认 spm\_modules

-f, --force 强制下载一个不稳定模块

-q, --quiet 显示少量日志

-S, --save 保存依赖到package.json

-SD, --save-dev 保存开发依赖到package.json

--verbose 显示完整日志

--no-color 禁用颜色输出

**wn publish**

**用法：**

wn publish [options]

在模块的根目录运行即可，比如

wn publish -i（发布该模块到内网spm.woniu.com）

wn publish -di（只发布文档页面到内网spm.woniu.com）

**功能：**

发布wn模块，可以通过参数切换发布环境，如果模块是组件，则自动生成demo页面，以便平台显示缩略图。

**参数：**

设置customUrl，inner，outer，doc，prepare四个参数

-c --customUrl <url>可以设置publish的目标源为自定义平台

-i --inner设置发布到内网平台，spm.woniu.com

-o --outer设置发布到外网平台，spmjs.io

-d --doc设置只发布doc，不发布该模块

-p --prepare设置只生成package.json和readme

对了，还有个fis-conf.js看起来很陌生的样子，这个文件是fis的配置文件，我们这次的wn工具就是在这个工具上进行的扩展和二次封装。FIS是专为解决前端开发中自动化工具、性能优化、模块化框架、开发规范、代码部署、开发流程等问题的工具框架。当然你可能会问，不是有很多现成的基于fis 解决方案吗，为什么不直接用现成的，的确，目前基于fis 的解决方案有很多，但是在经过对各种方案进行研究、比较后，发现没有一个现成的方案是可以满足我们公司特定需求和目前现状的，所以才在fis的基础上封装了现在的解决方案，wn。

当然，最后得感谢fis团队对前端界做的无私贡献，还有各个前辈的无私思想分享，wn是一个站在巨人肩上开发出来的解决方案。由于时间仓促，wn是一个还不够完善的解决方案，使用过程难免会有还没考虑到的bug，如果遇到还请到https://github.com/snail-team/wn/issues留言，我们会及时改进，继续完善的，最后希望这个解决方案能给大家带来一点帮助，开创一条更和谐、友好的蜗牛前端开发之旅！

谢谢观看！