[介绍 3](#_Toc375394766)

[破解 3](#_Toc375394767)

[显示行号 4](#_Toc375394768)

[代码格式化 4](#_Toc375394769)

[自定义快捷键 4](#_Toc375394770)

[全局搜索 5](#_Toc375394771)

[基本快捷键 5](#_Toc375394772)

[垂直查看 6](#_Toc375394773)

[编码 6](#_Toc375394774)

[Zen coding 7](#_Toc375394775)

[调试 7](#_Toc375394776)

[调试js 7](#_Toc375394777)

[调试typescript 7](#_Toc375394778)

[文件模板 9](#_Toc375394779)

[额外给模版注入变量 10](#_Toc375394780)

[优秀的代码片段模版 12](#_Toc375394781)

[如何插入代码片段呢？ 13](#_Toc375394782)

[如何在模版中加入变量呢？ 13](#_Toc375394783)

[使用JsTestDriver和jasmine测试js 15](#_Toc375394784)

[1.在你的根目录中插入JsTestDriver的配置文件jsTestDriver.conf 15](#_Toc375394785)

[2.在工程内引入JsTestDriver 16](#_Toc375394786)

[3.使用jasmine 17](#_Toc375394787)

[4.运行JsTestDriver 18](#_Toc375394788)

[Coffeescript 19](#_Toc375394789)

[预览coffeescript编译后的js代码 19](#_Toc375394790)

[支持TypeScript 20](#_Toc375394791)

[搭建环境 20](#_Toc375394792)

[已解决的问题 21](#_Toc375394793)

[1. 编译ts文件时报错：CreateProcess error=193, %1 ²»ÊÇÓÐЧµÄ Win32 ӦÓó（注：node 不是内部或外部命令） 21](#_Toc375394794)

[问题描述 22](#_Toc375394795)

[原因分析 22](#_Toc375394796)

[解决方案 22](#_Toc375394797)

[搭建node.js环境 23](#_Toc375394798)

[运行node.js 24](#_Toc375394799)

[安装模块 27](#_Toc375394800)

[注释 30](#_Toc375394801)

[代码导航和用法查询 30](#_Toc375394802)

[安装插件 31](#_Toc375394803)

[通过IDE直接安装插件 31](#_Toc375394804)

[从网页插件库下载插件 32](#_Toc375394805)

[插件 33](#_Toc375394806)

[ AltN8 33](#_Toc375394807)

[将界面风格换成sublime text风格 34](#_Toc375394808)

[参考资料 34](#_Toc375394809)

## 介绍

WebStorm是JetBrains的一个专门为Web开发人员设计的IDE。

JetBrains给WebStorm下的定义是：The smartest Javascript IDE。

## 破解

注册码：  
User Name:

ninja911.com

Licsense Key:

===== LICENSE BEGIN =====

53931-12042010

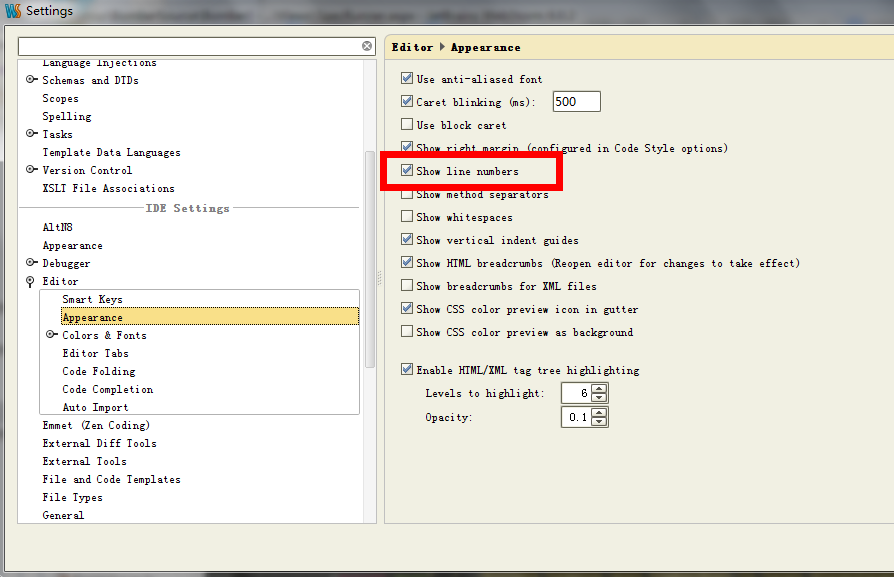
00001VQROnNeYqp6yrImItwg0Opa7U

68hDtCnR1uaFA7"k6Ufzkppi9dSwAv

8UahXQ7XHY3AHOZN6XSI"PUZch6Ji3

===== LICENSE END =====

## 显示行号



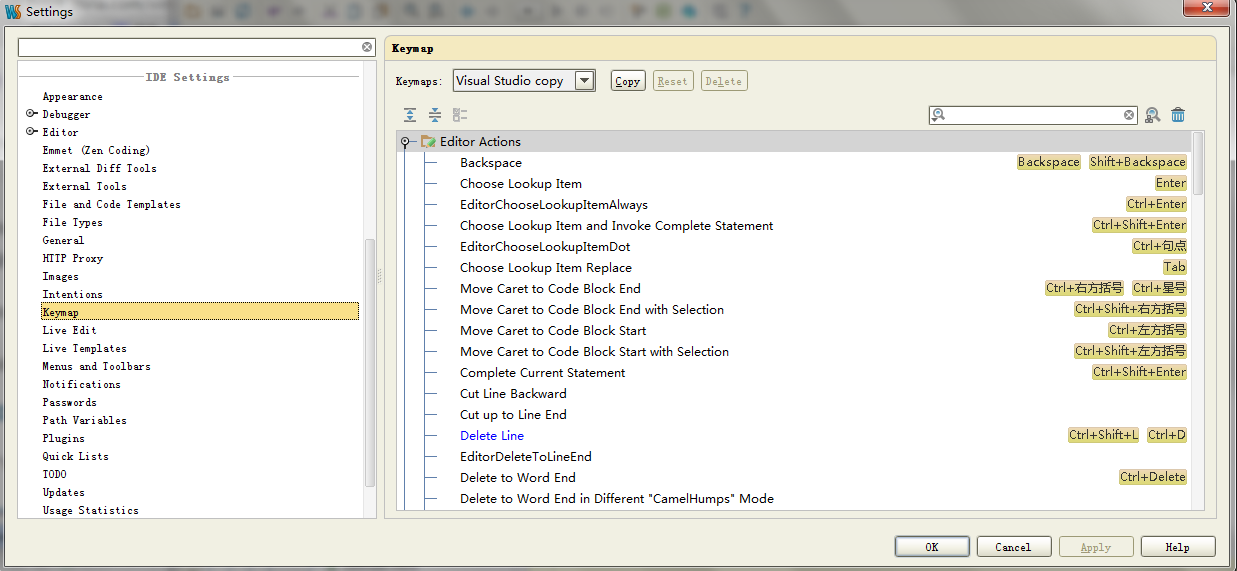
## 代码格式化

默认快捷键为：ctrl + shift+ f

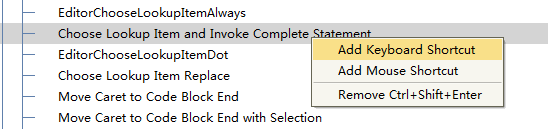
我设置为ctrl+q

## 自定义快捷键

File -> setting：



选中要更改的快捷键，鼠标右击，选择“Keyboard”：



更改代码格式化快捷键（搜索 format）

括号匹配

## 全局搜索

**ctrl+shift+A 全局搜索，如可以搜索工程名、setting选项等！**

|  |  |
| --- | --- |
| **ctrl+shift+R** | 指定目录内代码批量替换 |
| **ctrl+shift+F** | 指定目录内代码批量查找 |

## 基本快捷键

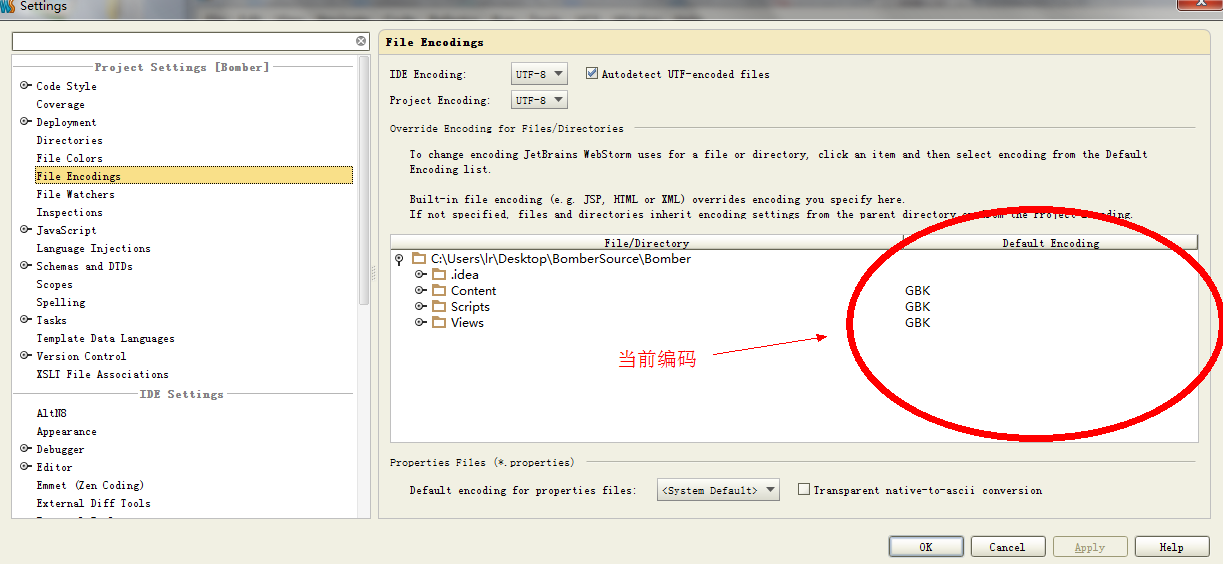
## 垂直查看

与vs2012相同

## 编码

设置全局编码：

File -> setting：

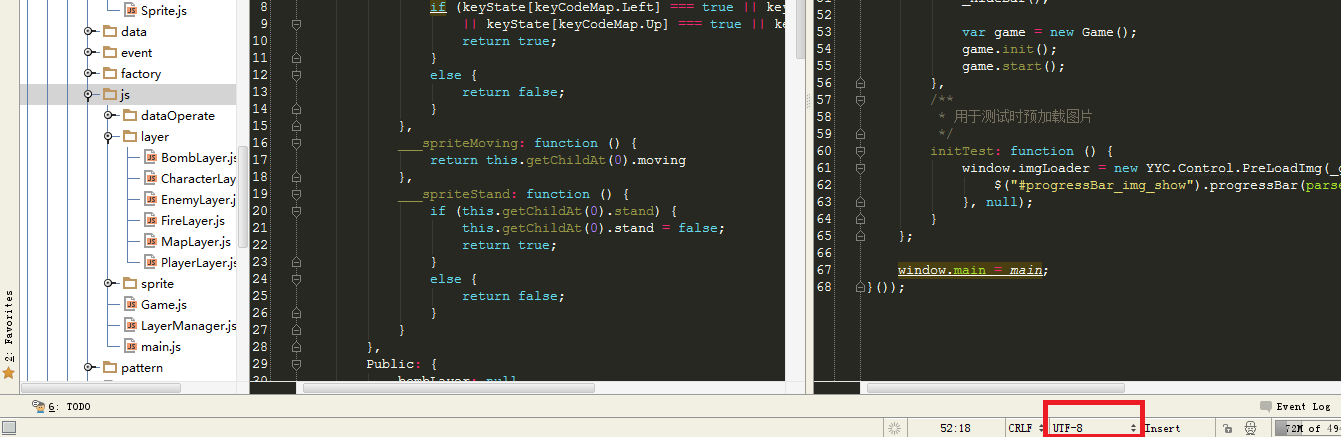


可以更改“当前编码”

也可以设置每个文件的编码

可以在file -> remove BOM或鼠标右键 -> remove BOM，来去掉utf-8的签名

还可以在右下角直接查看和修改当前文件编码：



## Zen coding

Ide原生支持zen coding编写方式。

经过我的实验，

在html页面的body外，可以正常使用zen coding；

在body和div中，只有部分zen coding可使用（如ul+）

## 调试

### 调试js

运行html页面的调试，提示并安装插件（chrome插件安装不起！因此我在ff下调试）

### 调试typescript

步骤：

1. 在html页面上引用js文件
2. 在ts文件中加入断点
3. 运行html页面的调试，提示并安装插件（chrome插件安装不起！因此我在ff下调试）
4. 进行调试

实际上我在ts中加入断点没有作用！！！！？？？

只有在js中加入断点，才有作用！！

怎么回事？

网上资料：

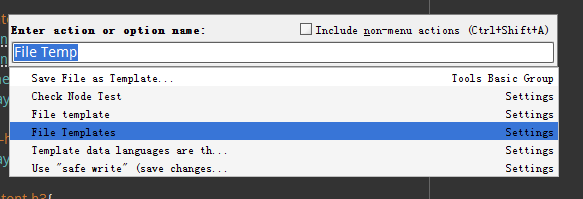
**To debug TypeScript code, perform these general steps:**

1. [Configure the JavaScript debugger.](http://www.jetbrains.com/webstorm/webhelp/configuring-javascript-debugger.html)
2. [Transpile the TypeScript code into Javascript](http://www.jetbrains.com/webstorm/webhelp/transpiling-typescript-to-javascript.html).
3. In the editor, open the HTML files with the TypeScript references.
4. Manually replace the references to the TypeScript files with the references to the JavaScript files that were generated through the**transpilation**.
5. [Configure and set breakpoints](http://www.jetbrains.com/webstorm/webhelp/using-breakpoints.html) in the TypeScript code. Although actually debugging will be performed against the generated JavaScript, WebStorm will emulate TypeScript debugging using the generated source-maps that set correspondence between lines in the the TypeScript and in the JavaScript code.

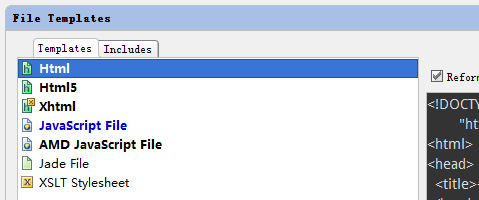
## 文件模板

提高编码效率，离不开代码模版，Webstorm的模版功能颇为强大和全面。

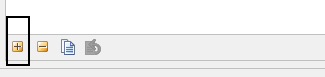
使用“ctrl+shift+a”，搜索File Templates：

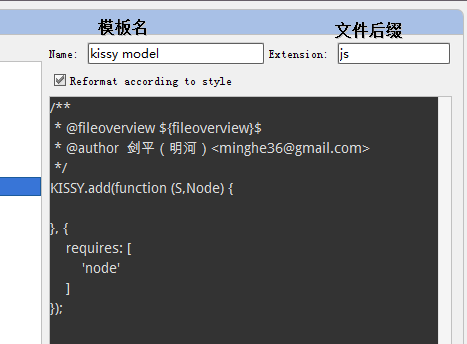
[](http://www.36ria.com/5822/template-1)

会打开如下窗口：

[](http://www.36ria.com/5822/template-2)

接下来以新建个kissy模块文件为例：

[](http://www.36ria.com/5822/template-3)

[](http://www.36ria.com/5822/template-4)

### 额外给模版注入变量

A [file template](http://www.jetbrains.com/webstorm/webhelp/file-and-code-templates.html) can contain template variables. When a template is applied, the variables are replaced with their values.

A file template variable is a string that starts with a dollar sign which is followed by the variable name. The variable name may be enclosed in braces. For example: $MyVariable or ${MyVariable}.

In this section:

* [Predefined template variables](http://www.jetbrains.com/webstorm/webhelp/file-template-variables.html#d127700e733)
* [Custom template variables](http://www.jetbrains.com/webstorm/webhelp/file-template-variables.html#d127700e818)

1. **Predefined template variables**

WebStorm comes with a set of predefined template variables.

The available predefined file template variables are:

* ${PROJECT\_NAME} - the name of the current project.
* ${NAME} - the name of the new file which you specify in the **New File** dialog box during the file creation.
* ${USER} - the login name of the current user.
* ${DATE} - the current system date.
* ${TIME} - the current system time.
* ${YEAR} - the current year.
* ${MONTH} - the current month.
* ${DAY} - the current day of the month.
* ${HOUR} - the current hour.
* ${MINUTE} - the current minute.
* ${PRODUCT\_NAME} - the name of the IDE in which the file will be created.
* ${MONTH\_NAME\_SHORT} - the first 3 letters of the month name. Example: Jan, Feb, etc.
* ${MONTH\_NAME\_FULL} - full name of a month. Example: January, February, etc.

1. **Custom template variables**

可以自定义一个变量，该变量需要指定一个值（不能在创建时输入值！）。如：

#set( $lowerCaseName = "$NAME.substring(0,1).toLowerCase()$NAME.substring(1)" )

可以在模板中使用如${ lowerCaseName }或$lowerCaseName来引用自定义的变量：

/\*\*#set( $lowerCaseName = "$NAME.substring(0,1).toLowerCase()$NAME.substring(1)" )

$lowerCaseName

\*/

In addition to the predefined template variables, it is possible to specify custom variables. If necessary, you can define the values of custom variables right in a template using the #set VTL directive.

For example, if you want to use your full name instead of your login name defined through the ${USER}, write the following construct:

#set( $MyName = "John Smith" )

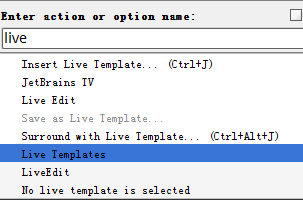
If the value of a variable is not defined in the template, WebStorm will ask you to specify it when the template is applied.

You can prevent treating dollar characters ($) in template variables as prefixes. If you need a dollar character ($ inserted as is, use the ${DS}file template variable instead. When the template is applied, this variable evaluates to a plain dollar character ($).

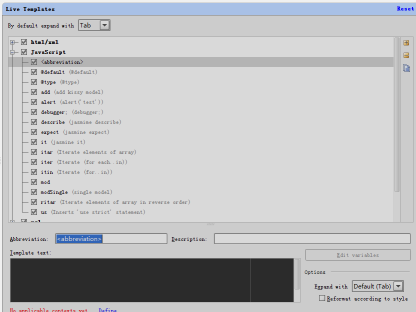
## 优秀的代码片段模版

Live Templates是非常实用的功能，严重推荐大家在日常工作中用起来。

使用“ctrl+shift+a”，搜索“live”：

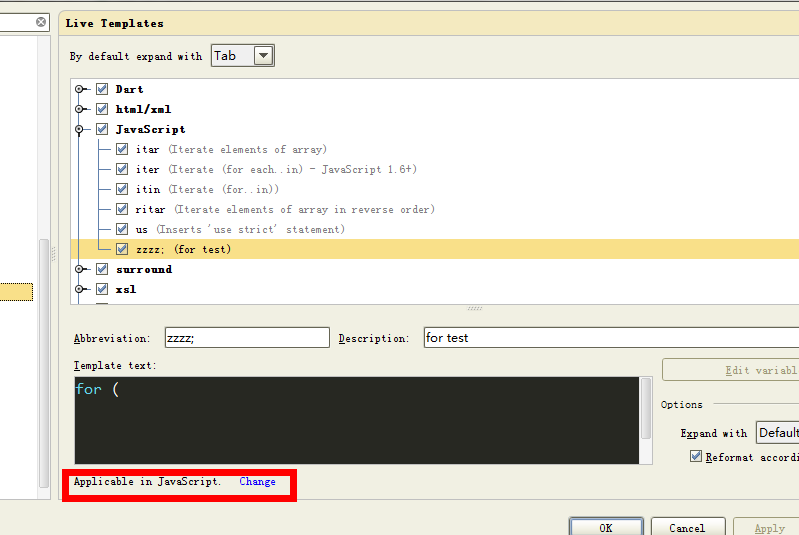
[](http://www.36ria.com/5822/template-8)

会打开如下界面：

[](http://www.36ria.com/5822/template-9)

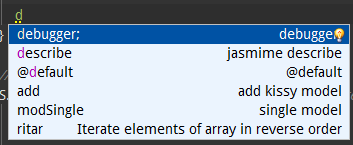
左侧会有个新建按钮，接下来我们来新建个常用的**debugger;**片段试试。

特别留意底部的有个live template出现的文件类型设置，是必须选择的。



### 如何插入代码片段呢？

在代码区域，输入“d”（不需要输入完成的模版名），然后使用“ctrl+J”：

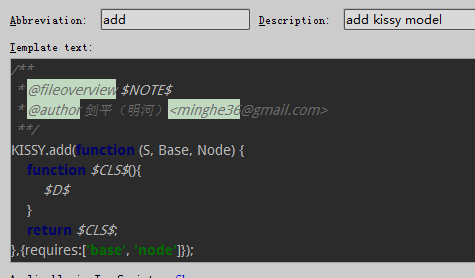
[](http://www.36ria.com/5822/template-11)

就可以找到我们之前新建的**debugger**模版。

### 如何在模版中加入变量呢？

webstorm的live template是可以加入变量的，来体会下这个功能。

以新建一个kissy的模块模版为例。

[](http://www.36ria.com/5822/template-12)

模版代码如下：

|  |
| --- |
| \*\*   \* @fileoverview $NOTE$   \* @author 剑平（明河）<minghe36@gmail.com>   \*\*/  KISSY.add(function (S, Base, Node) {      function $CLS$(){          $D$      }      return $CLS$;  },{requires:['base', 'node']});  </minghe36@gmail.com> |

变量以$变量名$这样的形式出现。

当你插入模版的使用，可以使用“tab”键切光标到变量位置，而且如果模版内这个变量出现多次，你只要输入一次变量值，所有地方都改变，非常的方便。

## 使用JsTestDriver和jasmine测试js

明河曾在年初发了[《使用JsTestDriver插件—idea使用技巧》](http://www.36ria.com/4823)，讲解idea中的JsTestDriver的使用，Webstorm内置JsTestDriver，用于自动化集成测试，JsTestDriver可以快速地在本地各个浏览器上帮你回归单元测试代码，功能颇为实用（但有些蛋疼的缺点，后面明河会论述）。

（PS：js的单元测试火过一段时间，但因为在实践过程中存在种种困难，比如代码的可测试性差，缺少最佳实践，缺少回归工具，缺少覆盖率检测、占用工时的误解等等一系列问题，导致现在js的单元测试很少出现在我们的业务中。明河日后会总结更多这方面的经验分享给大家。）

#### 1.在你的根目录中插入JsTestDriver的配置文件jsTestDriver.conf

内容可能如下：（根据本机目录的实际情况配置）

|  |
| --- |
| server: http://localhost:9876  load:    - assets/3.0/tools/jasmine/jasmine.js    - assets/3.0/tools/jasmine/jasmineAdapter.js    - assets/3.0/tools/jasmine/jasmine-html.js  test:  - assets/3.0/apply/1.0/test/\*.js |

* server：服务器路径
* load：依赖脚本，会在test指定的脚本执行前执行
* test：测试脚本

JsTestDriver简单原理说明：将load和test的脚本移到启动的server服务器上，然后在各个浏览器中运行服务器的脚本，最后将单测得结果集返回给Webstorm。

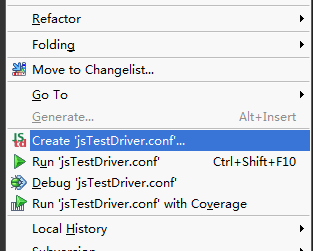
[](http://www.36ria.com/5855/overview)

使用JsTestDriver的注意点：

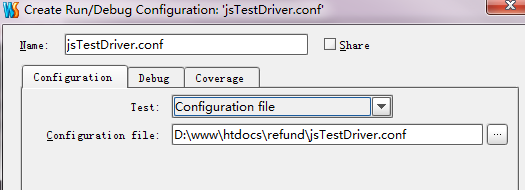
* 测试脚本所依赖的所有文件，必须指定；
* JsTestDriver的测试运行页面是自己创建的，没办法自己指定（这点蛋疼，会造成很多困难）；
* 在ajax的测试中要特别注意，你需要mock方法，不然会出现无法获取数据，导致单测无法进行的问题；
* JsTestDriver如果使用其他单测类库比如jasmine，需要适配器

#### 2.在工程内引入JsTestDriver

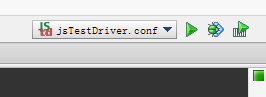
在代码界面右击，寻找到下面的菜单项：

[](http://www.36ria.com/5855/jstestdriver-1)

打开创建界面后，指定下配置文件路径：

[](http://www.36ria.com/5855/jstestdriver-2)

成功后，右上角会出现JsTestDriver的信息：

[](http://www.36ria.com/5855/jstestdriver-3)

#### 3.使用jasmine

jasmine应该是最被广泛使用的js单测类库，所以明河以jasmine的单测运行为例，而不是JsTestDriver（这里指的是JsTestDriver内置的单测类库）。

配置load，引入jasmine和jasmineAdapter：

|  |
| --- |
| server: http://localhost:9876  load:    - assets/3.0/tools/jasmine/jasmine.js    - assets/3.0/tools/jasmine/jasmineAdapter.js    - assets/3.0/tools/jasmine/jasmine-html.js |

这三个文件可以在[明河的代码库](https://github.com/minghe/jasmine-kissy)中找到。

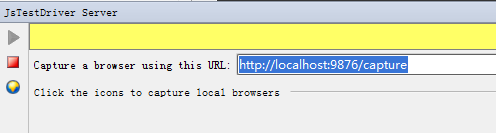
随便写个测试代码：

|  |
| --- |
| server: http://localhost:9876  test:  - assets/3.0/apply/1.0/test/\*.js |
| describe('this is a demo',function(){       it('demo it',function(){          expect(false).toBeTruthy();       })  }); |

#### 4.运行JsTestDriver

使用快捷键“shift+F10”

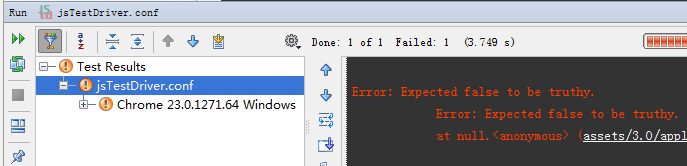
有可能会因为缺少服务器路径配置，导致运行失败，没关系，在右下角打开JsTestDriver的界面，填写上url（http://localhost:9876）

[](http://www.36ria.com/5855/jstestdriver-4)

点亮浏览器（点击灰色的浏览器图标）：

[](http://www.36ria.com/5855/jstestdriver-5)

再按“shift+F10”运行之，如果出现类似以下界面，说明运行成功：

[](http://www.36ria.com/5855/jstestdriver-6)

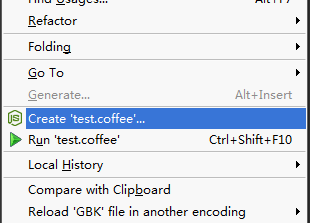
Webstorm的确强大，给我们提供了众多便利，就单元测试而言JsTestDriver只是第一步，关键还是开发者将单元测试写起来！以后会给大家分享单测代码层面的最佳实践。

## Coffeescript

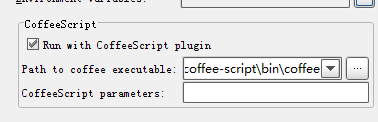
**Webstorm对coffeescript的支持**

在demo工程中新建一个test.coffee,

在代码区域右键点击：

[](http://www.36ria.com/5869/coffeescript-1)

需要指定下coffeescript的插件：

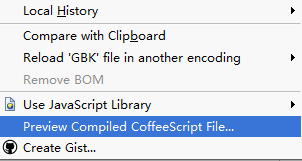
[](http://www.36ria.com/5869/coffeescript-2)

接下来就可以使用coffeescript来写nodejs的代码。

|  |
| --- |
| square = (x) -> x \* x  console.log square(10) |

点击运行，控制台会打印出100。

##### 预览coffeescript编译后的js代码

[](http://www.36ria.com/5869/coffeescript-3)

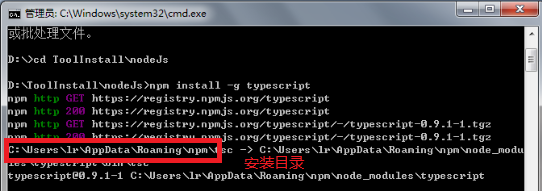
## 支持TypeScript

### 搭建环境

1. 首先安装node.js
2. 安装tsc编译器

在cmd中，进入nodeJs安装目录

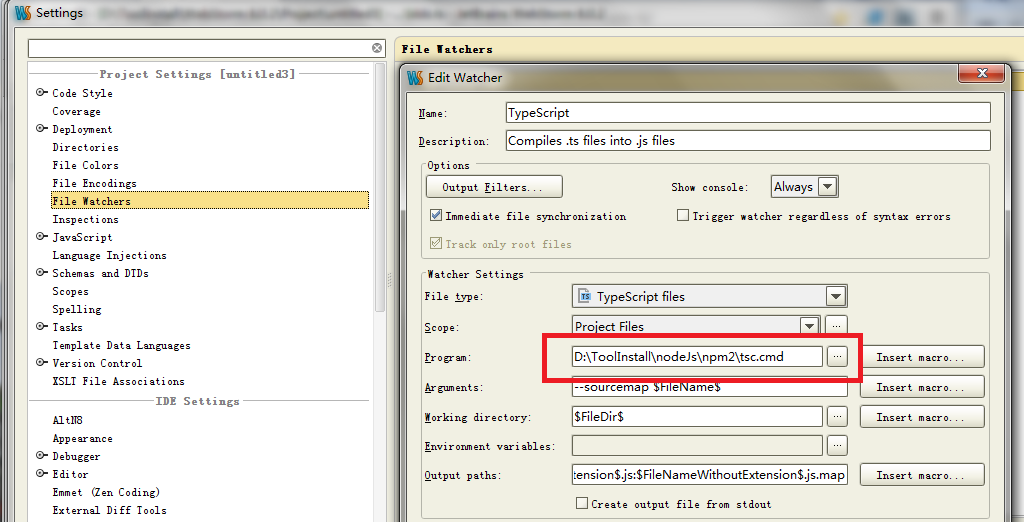
输入npm install -g typescript，安装编译器：



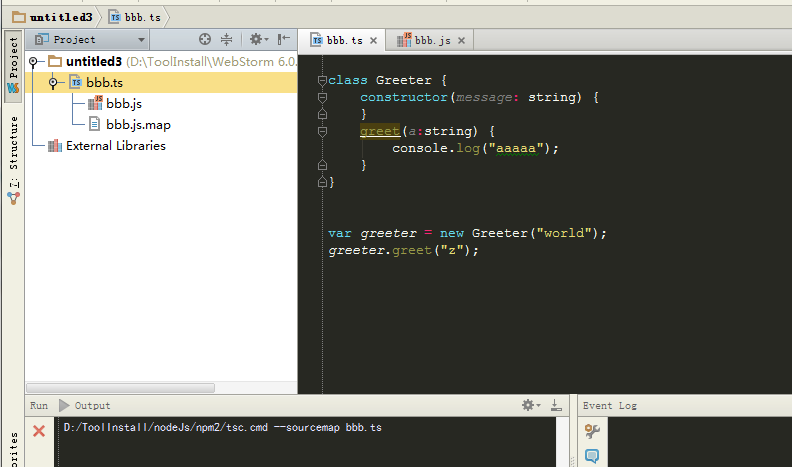
此处安装目录为C:\Users\lr\AppData\Roaming\npm

注意！要把该文件夹重命名为npm2，移到nodeJs的安装目录（如D:\ToolInstal\nodeJs\）。详见“已解决的问题”

1. 然后配置typescript的编译器tsc。该编译器可以在ts文件（typescript文件）更改时，自动编译检查，生成js和js.map文件。  
     
   在file -> setting -> file watchers中新建个watcher，注意Program要选择编译器目录中的tsc.cmd！（编译器目录已经移到nodeJs安装目录中了）



1. 然后在项目中新建一个typescript文件，编写代码后会自动编译，生成js和js.map文件



### 已解决的问题

1. 编译ts文件时报错：CreateProcess error=193, %1 ²»ÊÇÓÐЧµÄ Win32 ӦÓó（注：node 不是内部或外部命令）

##### 问题描述

在配置了file watcher后，系统自动编译ts文件时报错。

这里报错信息有中文乱码！所以我不知道具体是什么错！

所以我首先是想解决中文乱码的问题！结果网上的解决方案都用了，还是中文乱码！

所以我就直接在cmd中，进入tsc的目录，运行tsc D:\bbb.ts，尝试手动编译。结果报错：node 不是内部或外部命令

至此，我终于搞清楚原来是这个错误！

网上查了下，说是要配置环境变量Path。

我把用户变量和系统变量的Path都增加了nodeJs的安装目录，还是不行！

##### 原因分析

##### 解决方案

将tsc的目录文件（npm）拷贝到nodeJs安装目录中！（npm重命名为npm2）

然后将File watcher的program路径设为新目录的路径（如：D:\ToolInstal\nodeJs\npm2\tsc.cmd）

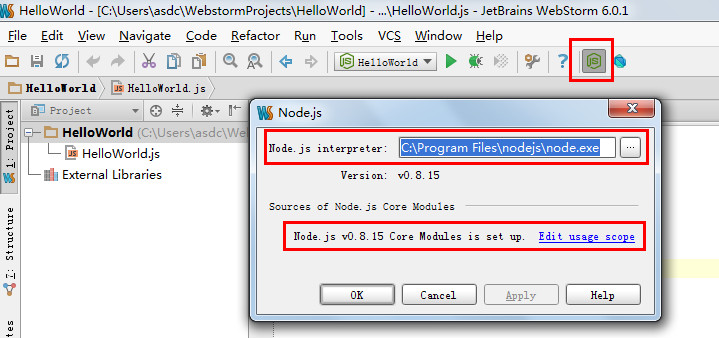
然后重启webstorm

## 搭建node.js环境

    1.根据个人情况[下载](http://nodejs.org/download/)不同版本的安装包，并安装好Node.js环境。安装步骤很简单，不需要配置和修改什么，一切都走默认路线即可。

      2.在安装完Node.js并没有什么异常提示之后，在cmd下，进入nodeJs安装目录，敲入 node 命令，测试一下，看看node.js是否安装成功。

      3.安装WebStorm软件，安装成功之后，需要在其中配置一下Node.js：如下图



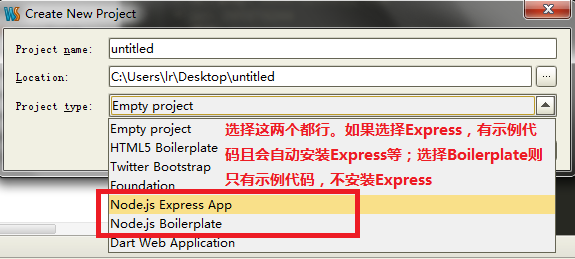
       1）点击 [JS] 图标，在弹出的配置框中默认检索到本地安装的Node.js，也可手动选择到相应的文件目录下。

       2）Sources of Node.js Core Modules: 指的是配置源文件路径，把Node.js源码解压后的路径配置到此即可。

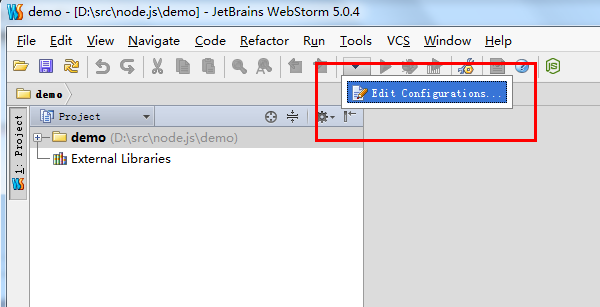
我配置的是：D:\ToolInstall\nodeJs\node\_modules\npm

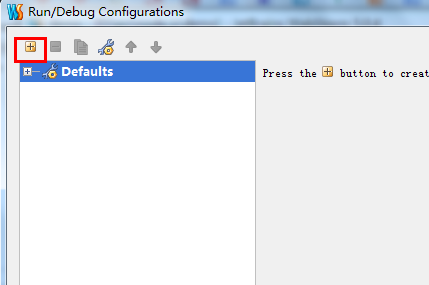
### 运行node.js

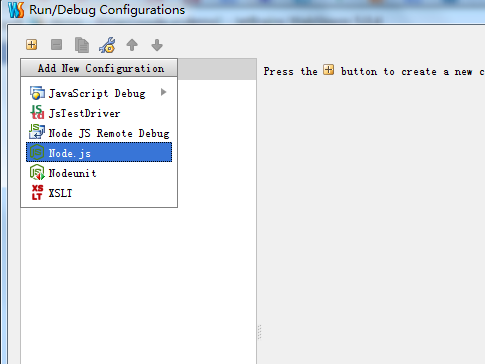
1. 创建一个Node.js项目（如果不是Node.js项目，也可以运行，但没有智能提示）。

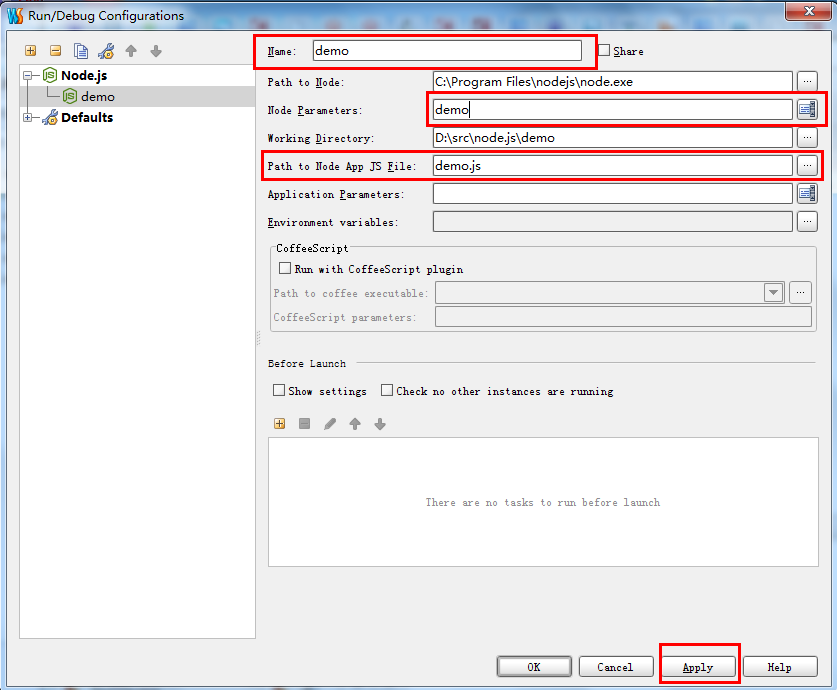


1. 配置编译环境





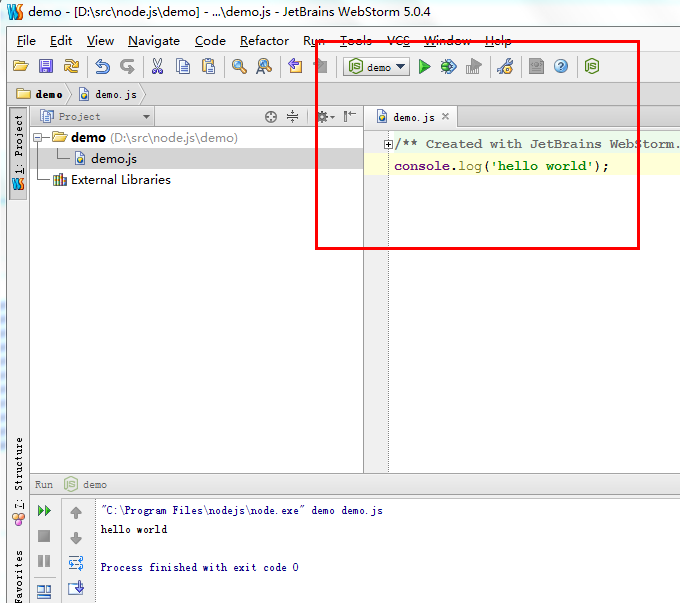




Working Directory为node.js文件所在目录，需要选择。

Path to Node App Js File：准备运行的node.js文件，需要选择

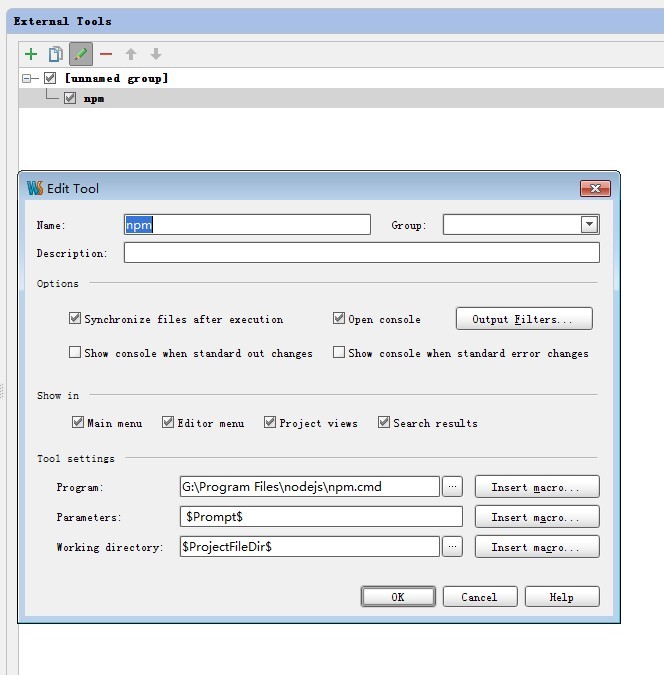
1. 运行node.js工程

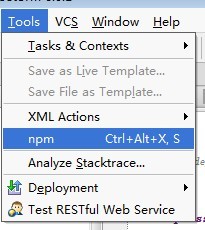
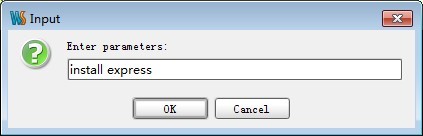


点击箭头执行。

### 安装模块

1. 添加自定义的工具npm

ctrl+alt+s打开**setting**菜单，找到**external tools**  
点+号，在tool setting里填空：  
program -> 外部命令所在位置，填入npm的完整路径  
parameters -> 参数，这里设为 $Prompt$运行命令时表示弹个窗口让你输入  
Working Directory -> 在哪个目录下运行这条命令，输入$ProjectFileDir$表示在当前项目的根目录下运行。  
点右边的insert macro可以看到更多的选项，除了macro的说明，webstorm还会告诉你对应macro在当前环境下是个什么值的，不怕看不懂e文了。[](http://ww2.sinaimg.cn/large/7327f6d5gw1e4sips6984j20ig0irtag.jpg)  
最后给这个命令起个名字，叫npm保存完事。

1. 在顶部菜单找到tools就可以看到刚保存的npm。点击它就会弹出个窗口，输入install express试试吧。  
   [](http://ww2.sinaimg.cn/large/7327f6d5gw1e4siqh4em2j205p06ewep.jpg)  
   [](http://ww4.sinaimg.cn/large/7327f6d5gw1e4siqz9rkgj20br03s0st.jpg)

## 注释

支持jsdoc

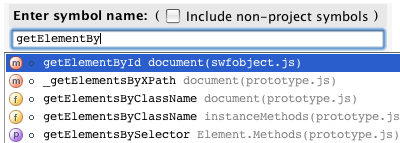
注释快捷键已修改为：alt + z

## ****代码导航和用法查询****

只需要按着Ctrl键点击函数或者变量等，就能直接跳转到定义：

http://www.jetbrains.com/webstorm/features/screenshots/JS_go_to_definition.png

可以全项目查找函数或者变量，还可以查找使用并高亮：



## 安装插件

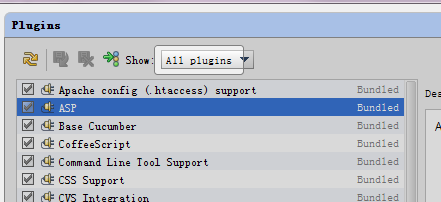
Webstorm的插件丰富程度无法跟eclipse、sublime text、vim相比，但Webstorm本身就足够强大，并不大需要安装太多插件。

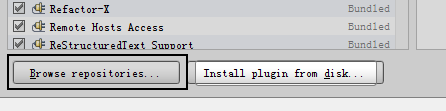
Webstorm的插件库中有些还是很有意思，用户可以多挖掘下。

（PS:idea的插件并不一定通用于webstorm）

### 通过IDE直接安装插件

进入“settings”，搜索“plugins”，进入插件页面

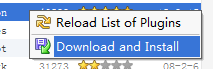
[](http://www.36ria.com/5708/plugin)

[](http://www.36ria.com/5708/webstorm-plugins-1)

试试安装个“css-X-fire”插件（用于当使用firebug修改css属性时，编辑器内的css代码也会发生变化。）

[http://www.36ria.com/wp-content/uploads/2012/10/webstorm-plugins-2.png](http://www.36ria.com/5708/webstorm-plugins-2)

右击选择下载：

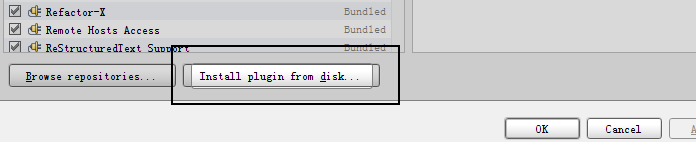
[](http://www.36ria.com/5708/webstorm-plugins-3)

安装成功后自动重启下IDE即可。

### 从网页插件库下载插件

以前使用IDE下载插件，经常出现下载不动的情况。

你还可以进入[插件库页面](http://plugins.intellij.net/?webide)，将插件下载到本地后。

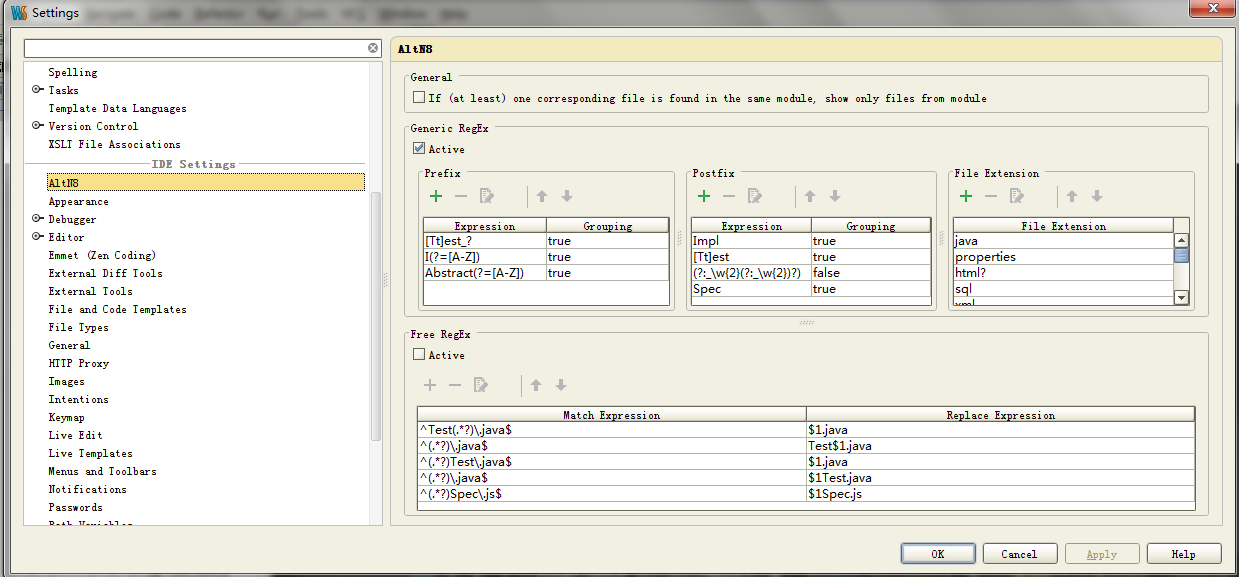
[](http://www.36ria.com/5708/plugins-5)

## 插件

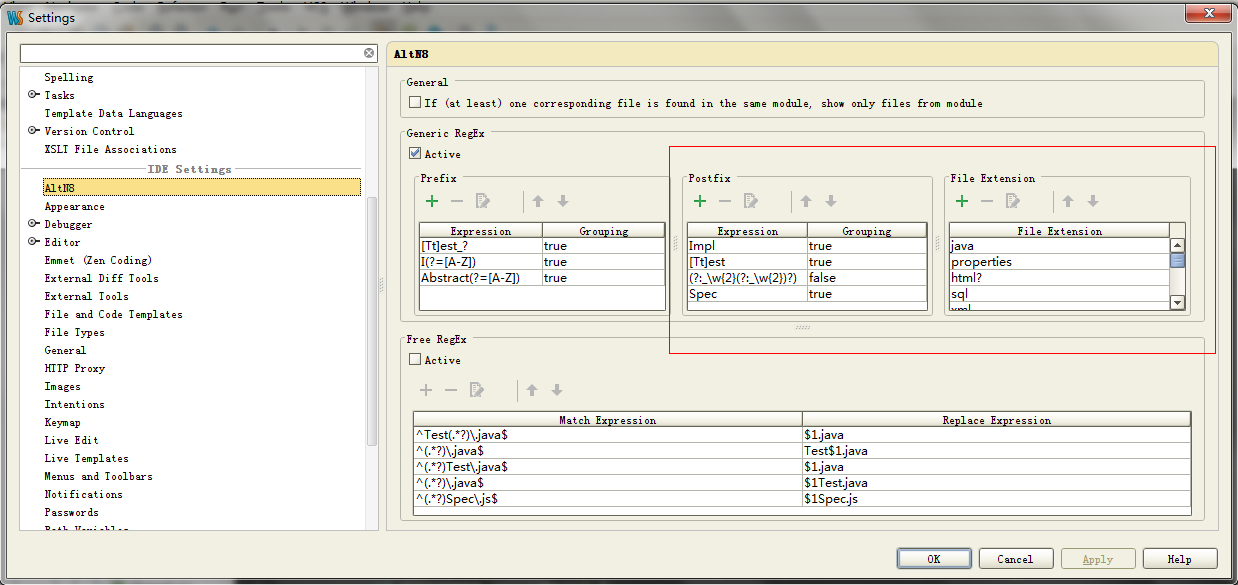
* AltN8

该插件可以设置正则表达式，来找到与当前文件匹配的文件。

安装插件后，在File -> setting -> AltN8设置匹配规则：



在第一行的中间那个prefix中设置匹配的文件包含的文件名，第一行最右边设置文件后缀名：



设置完毕后，在文件中按alt + 8 即可查看相关文件列表并可跳转。

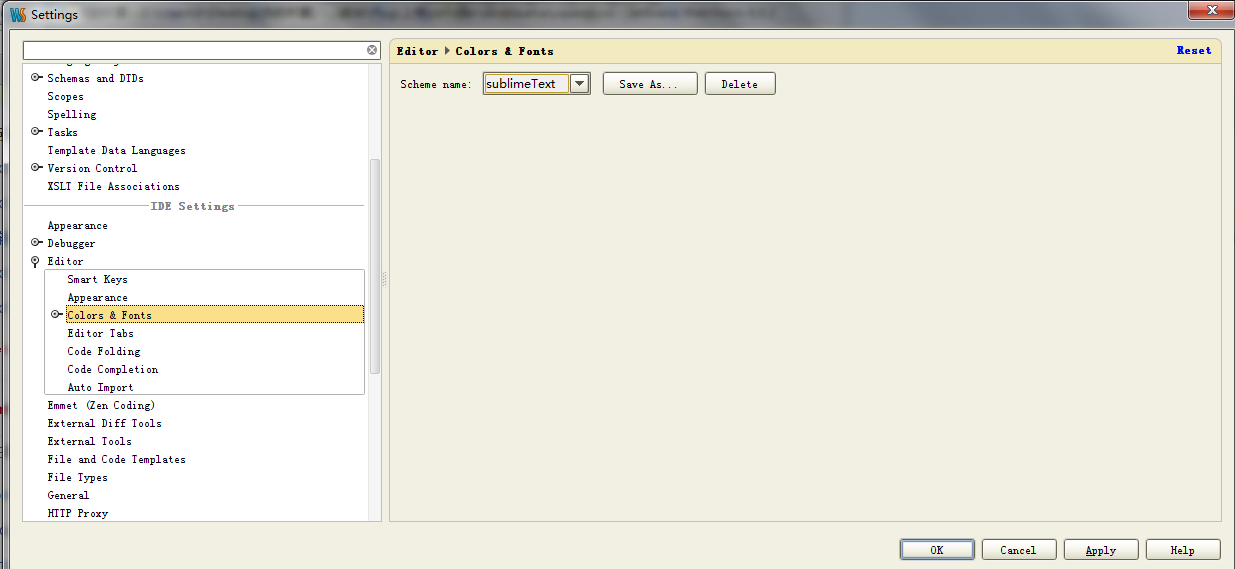
我设置快捷键为alt + 1.

## 将界面风格换成sublime text风格

下载sublimeText.zip，解压文件（我已下载到插件中）

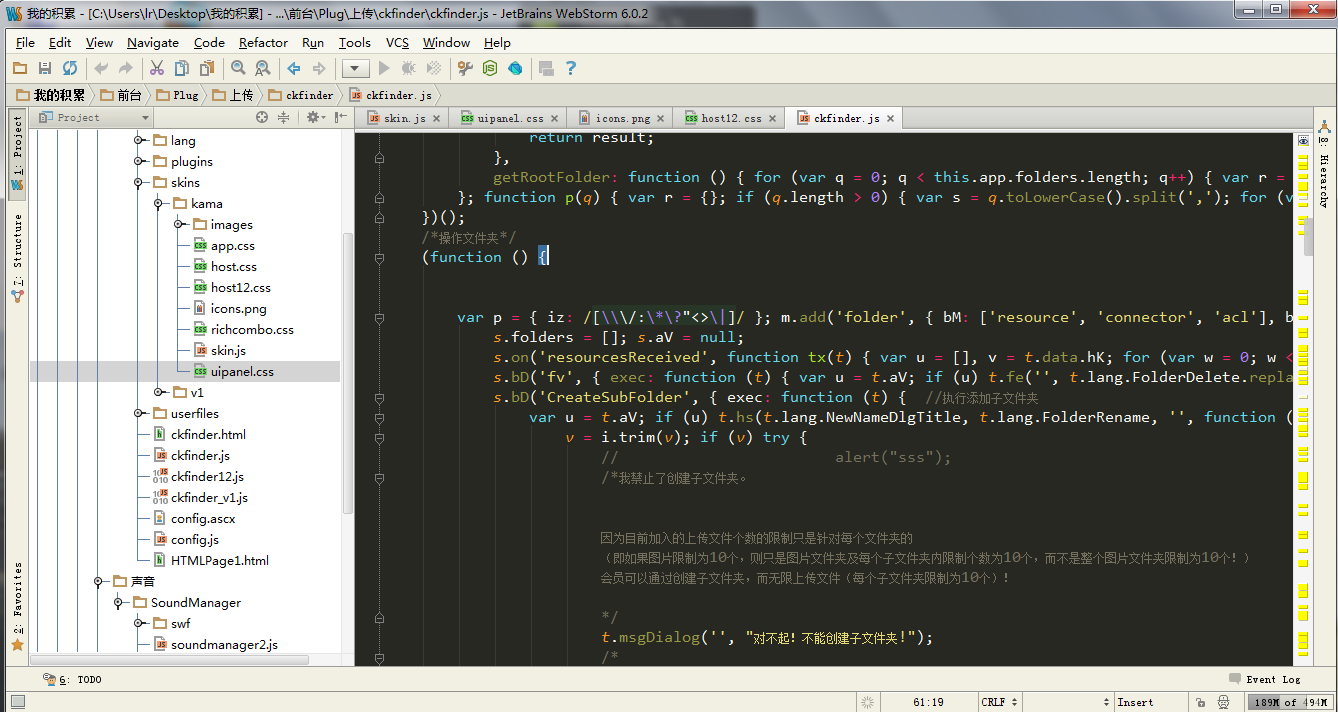
将文件放到**C:\Users\lr\.WebStorm6\config\colors中。**

然后在file -> setting -> editor -> Colors & Fonts选择scheme name：



即成功了！

效果图：



# 参考资料

[WebStorm 7.0.0 Web Help](http://www.jetbrains.com/webstorm/webhelp/webstorm.html)