1. Sass

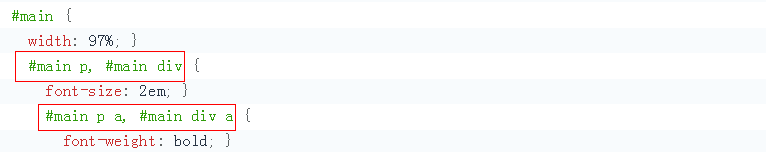
Sass有SCSS和Sass两种语法格式。两者唯一区别在于前者使用花括号，后者使用缩进来表示选择器的嵌套；前者使用分号，后者使用换行来分隔属性。两者可以使用sass-convert命令行工具相互转换。SCSS以.scss作为文件扩展名。Sass以.sass作为文件扩展名。

SCSS完全兼容CSS3语法（当然也支持CSS2）。即，每个CSS样式表也是一个同等的SCSS文件。

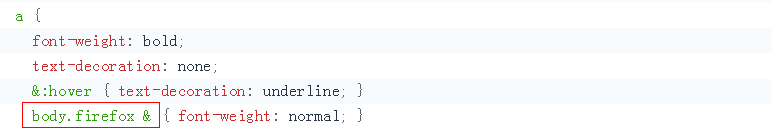
* 1. CSS扩展
     1. 嵌套规则

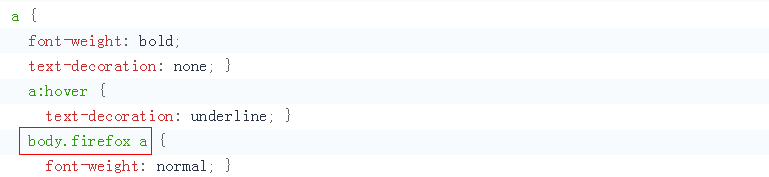
（1）多层嵌套



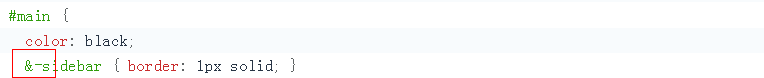


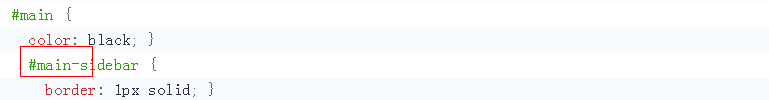
（2）引用父选择器:&



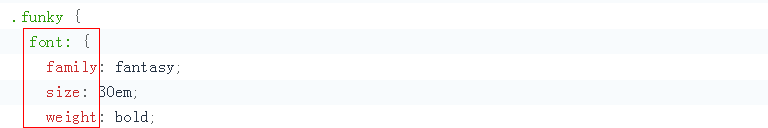


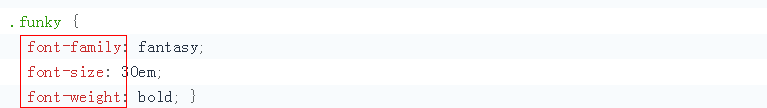
（3）后缀（只能接后缀，不能接前缀或在中间）



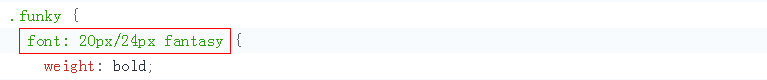


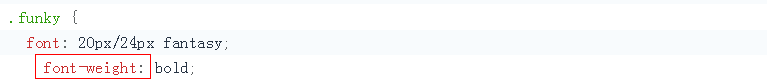
（4）嵌套属性





还可以如下：





* + 1. 注释

多行注释/\*\*/会保留。单行注释//会去掉。

* 1. SassScript

除了普通的CSS语法，Sass还支持一些扩展，名为SassScript。

* + 1. 变量

**1.概念**

（1）以$开头。

（2）变量名中的连字符（-）和下划线（\_）等同，如$main-width和$main\_width等同。

【注】html不区分大小写。css除标签选择器外，是区分大小写的。

**2.变量的作用域**

（1）定义在选择器内的变量为局部变量，只在选择器内，及其嵌套层级内可用（可以理解为块级作用域）。

（2）定义在选择器外，或定义在选择器内但在变量后加上!global标志，则全局可见。



（3）局部变量名可以和全部变量名相同，局部变量在本块级作用域内覆盖全局变量（但不会改变全局变量的值）。

* + 1. 数据类型

一共支持7种主要的数据类型：

**1.数字**

如：1.2,13,10px。

**2.文本字符串**

（1）可以为带引号字符串和不带引号字符串，如:＂FOO＂、＇BAR＇、BAZ。

（2）对于带!important的声明等，都被视为文本字符串。通常在在编译时，不会改变Sass文档中使用的字符串类型（带引号的也会原样输出）。只有#{}会去掉引号：







**3.颜色**

如：BLUE、#04A3F9、RGBA（255,0,0,0.5）。

**4.布尔值**

true、false。

**5.空值**

null。

**6.值列表（****list）**

（1）用空格或逗号分隔，如：





（2）列表可以嵌套。如，1px 2px, 5px 6px和(1px 2px) (5px 6px)都是包含1px 2px列表和5px 6px列表两个项的列表。当列表被编译为CSS时，Sass不会添加任何圆括号，即两者完全一样，最后都是1px 2px 5px 6px，没有圆括号。虽然最终同1px 2px 5px 6px，但在scss中作用却不同。

（3）()表示空列表。不能使用空列表（直接作为变量值），否则报错。当列表中包括空列表或空值，将会被去除。如，将会把空值去掉。

**7.Maps**

（1）键和值对集合。



和list不同，Maps必须始终使用括号括起来，并且必须用逗号分隔。键和值可以是任意的SassScript对象

（2）Maps不能转换为纯CSS。作为变量的值或参数传递给CSS函数将会导致错误。

（3）Maps嵌套方式同list。

（4）()既可以表示1个空的键/值对，也可以被视为1个没有元素的列表。

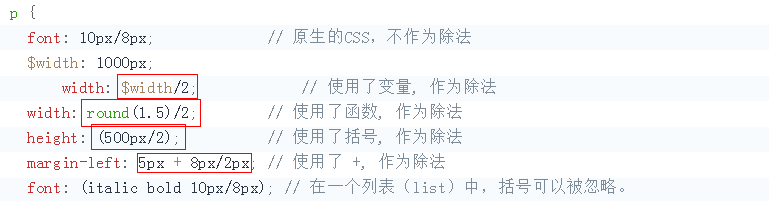
* + 1. 数据运算

所有数据类型都支持相等运算（==和!=）。除此之外，每种类型还有自己特殊的运算方式。

**1.数字运算**

（1）会保留单位，所以不能如10px×10px这样的运算方式，而应该使用如10×10px这样。

（2）由于CSS允许/分割属性值，所以通常/不会当作除号，除非以下3种情况：变量参与运算或通过函数返回，括号括起来的运算，算术表达式的一部分。



（3）如果想/同变量一起，但又不做除法运算，可以使用#{}插值语法。



（3）由于-可以如font-size这样作为标识符，还可以在变量中使用。所以使用时注意：

1）数字开头意味着不是标识符，而是数学运算，如5px-3px。

2）字面数字前-表示负数，如1 -5表示1个列表。

3）减法的时候，应该始终在-前后有空格。

4）-作为负号，-后面不要加空格。

5）用空格隔开的列表，应该使用()把负数和运算括起来。

**2.颜色运算**

（1）颜色之间，颜色和数字之间是分段计算的。如#010203+#040506=#050709，#010203\*2=#020406。

（2）包含alpha通道的颜色运算，必须具有相同的alpha值才能进行，且alpha值不变。如rgba(255,0,0,0.75)+rgba(0,255,0,0.75)=rgba(255,255,0,0.75)。

**3.字符串运算**

（1）+运算用于连接字符串（两边都是字符串）。

（2）如果左侧字符串带引号，右侧不带，返回的字符串为带引号的。同样，左侧不带引号，返回也是一个不带引号的字符串。

（3）通常，与其他类型值连用时，用空格做连接符。



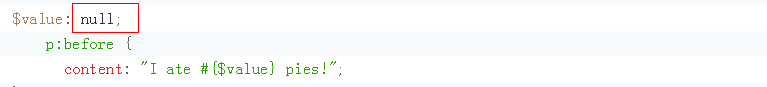


（4）在文本字符串中，#{}式插值可以用来在字符串中放置动态值。





（5）空值会在#{}式插值中去掉。





**4.布尔运算**

支持布尔值的and、or和not运算。

**5.列表运算**

不支持任何特殊运算，只能使用list函数控制。

* + 1. 圆括号

圆括号可以用来影响运算的顺序。

* + 1. 函数

可以像普通CSS函数语法一样调用。函数允许指定明确的关键词参数，可以让Sass代码更易阅读。

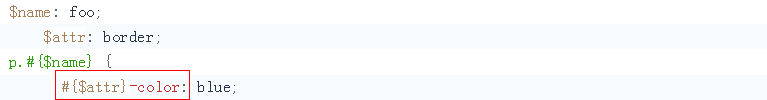




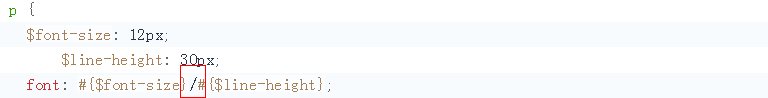
命名参数可以按任意顺序传入，并且具有默认值的参数可以省略掉。

* + 1. 插值

可以通过#{}插值语法在选择器和属性名中使用SassScript变量。

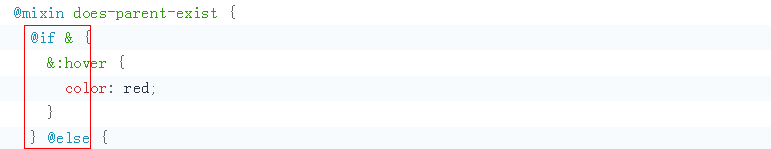


注意，使用#{}插值语法意味着它两侧运算符都将被视为纯CSS，而不会参与数据运算。



* + 1. &

SassScript中的&也指向当前父选择器（也可以指向列表）。如果没有父选择器，&的值将是空。可以在一个mixin中使用它来检测父选择是否存在。



* + 1. !default

如果变量值后有!default，表示该变量如果已经赋值，那么它不会被重新赋值；如果尚未赋值，则可以被赋予新值。

赋值为null视为没有赋值。

* 1. @规则和指令

除CSS3的@规则外，Sass还包含一些自定义指令。

* + 1. @import

（1）默认，Sass中@import会将其导入到引入文件中。一下情况例外：

* 文件扩展名是.css。
* 文件名以http://开始或url()。
* @import中包含任何的媒体查询。

如果没有扩展名，会尝试按.scss或.sass扩展名导入。

（2）@import支持同时导入多个文件。



（3）#{}插值适用于url()导入。

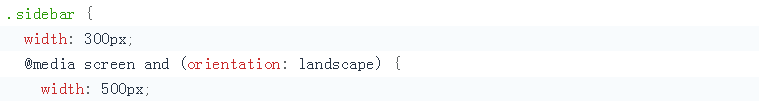


（4）如果不希望将1个.scss或.sass文件编译到1个正常的CSS文件中，可以在文件名开头添加一个下划线，但导入语句中不需要添加下划线。

（5）如果有1个命名为如\_colors.scss的文件，但不希望编译成\_colors.css文件，可以在@import时不写下划线，则会将该文件导入。

* + 1. @media

如果一个@media指令出现在CSS规则中，它将被冒泡到样式表的顶层，并且包含规则内所有的选择器。



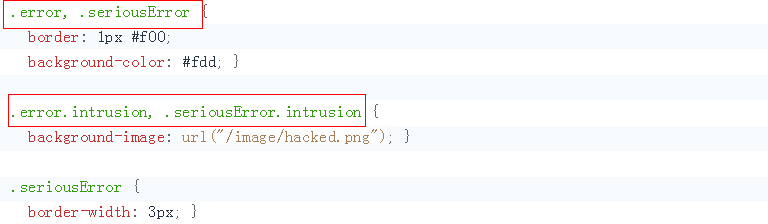


@media也可以相互嵌套，会以and操作符号结合。

* + 1. @extend

在被扩展选择器出现的地方，插入扩展选择器，可以避免不必要的重复。

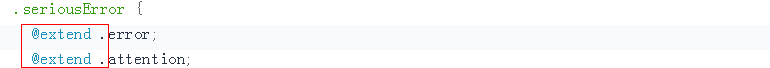




（1）还可以扩展伪类。



（2）可以多重扩展，可以用逗号分隔的选择器列表。

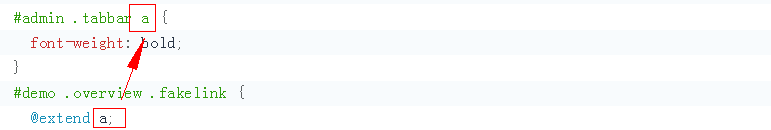


（3）还支持链式扩展。





（4）不支持扩展选择器序列，如。但支持合并选择器序列（有些复杂）。



（5）@extend-Only

使用%占位选择器,不会最终编译到css文件中的扩展。





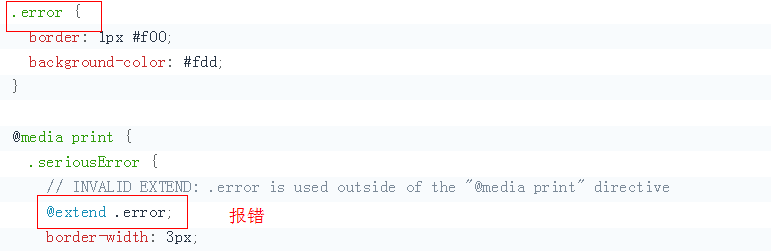
（6）!optional

如果扩展1个不存在，或语法有问题的选择器，会报错。如果在@extend后添加!optional标志，则这种情况不会编译。



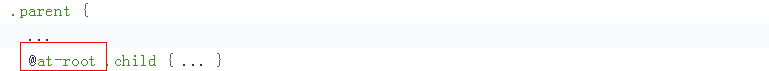
（7）@media相关

如果在@media层级内，没有问题；之外，会报错。



* + 1. @at-root

将规则输出在文档的根层级上，而不是被嵌套在其父选择器下。





默认只升到选择器外，可以使用without语法如升到media等外面。

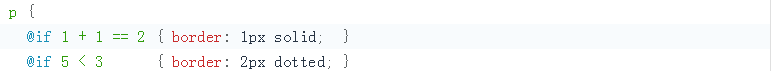
* 1. 控制指令和表达式

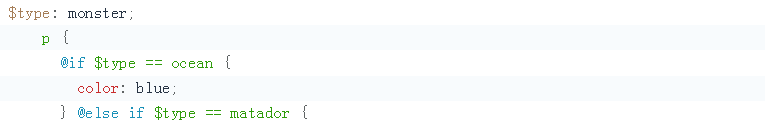
**1.if()**

处理分支并返回两种可能结果。



**2.@if**





**3.@for**

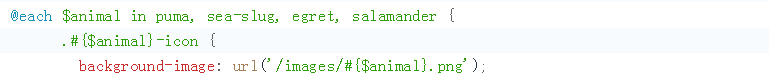


（1）区间必须是数字。

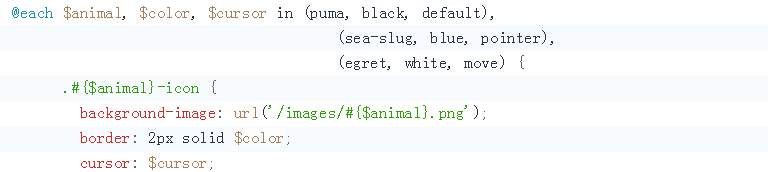
（2）from-through包括<start>和<end>，from-to包含<start>，不包括<end>。

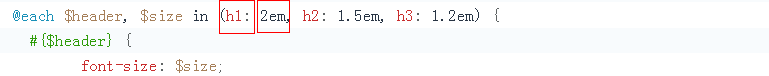
**4.@each**

（1）格式。

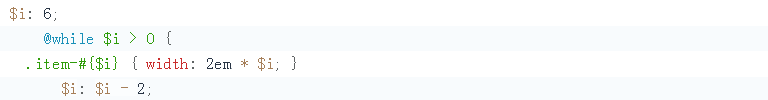


（2）多重赋值。当列表中有列表时，可以使用多个变量。



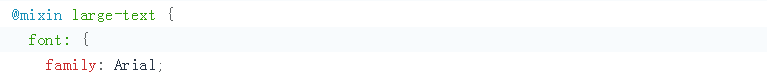


**5.@while**



* 1. Mixin
     1. 概念

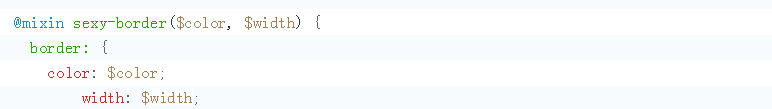
1.通过@mixin指令定义。



2.使用@include引用混合。



* + 1. 参数



（1）可以为参数指定默认值。



（2）在引入时，可以不写关键字，也可以写关键字。



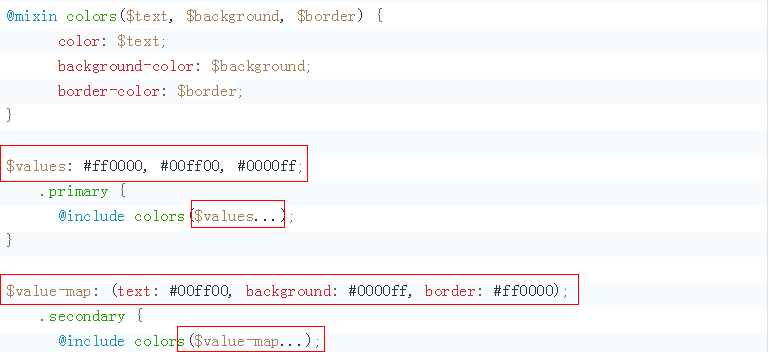


（3）可变参数

不能确定混入或者函数（function）使用多少个参数，如，创建box-shadow时。可以在参数名后加...，后面的参数将会被视为1个列表传入。

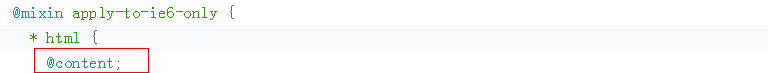


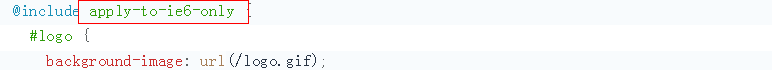
在调用时也可以使用可变参数。



* + 1. 传递内容块到混入

传递样式内容块到混入，样式内容块将出现在混入内任何@content指令的位置。



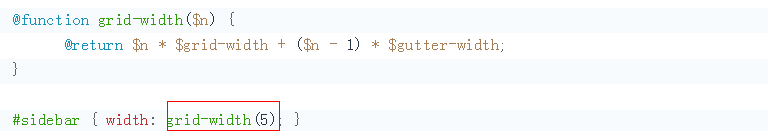




传递给混入的内容块在其被定义的作用域中进行运算，而不是混入的作用域。

* 1. 函数
     1. 函数指令（自定义函数）

可以自定义函数，并能在任何上下文中使用。



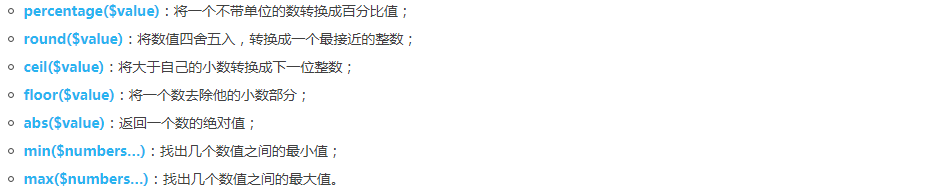
由于名称与混入类似，建议加上前缀以避免命名冲突。

* + 1. 内置函数

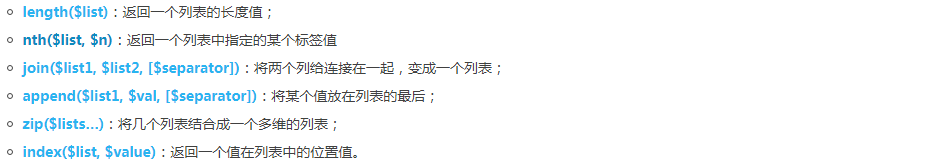
**1.字符串函数**



**2.数学函数**



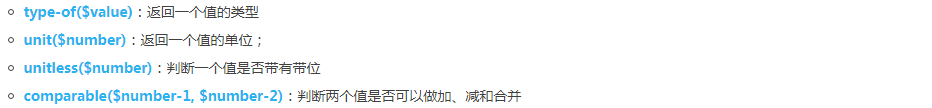
**3.List函数**



**4.map函数**



**5.Introspection函数**



**6.RGB颜色函数**







* 1. 使用总结

（1）sass中，可以使用如下方式，让&可以作为其他样式后台。



（2）可以使用@extend，复用其他样式（其他样式还存在）。



如果不想让其他样式存在，可以使用%placeholders占位。%不像.classes或者#ids，只要不通过@extend调用，不会产生任何代码量。

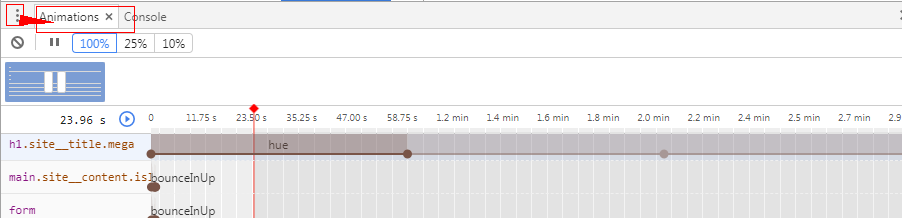


而mixin不同，因为它必须通过@mixin声明，默认不会编译，不会产生代码量。

（3）颜色使用：可以设置页面1个基础颜色，然后根据lighten、darken处理颜色。

1. Chrome开发者工具

**1.查看动画**



（1）在console更多中找到。

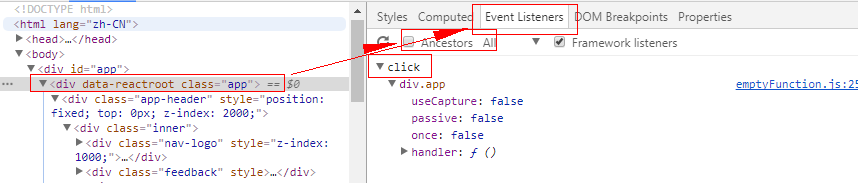
（2）web中组动画概念，chrome是根据开始时间判断。所谓动画只是animation，不包括其他。

（3）深色部分是其定义。右侧的浅色部分表示重复。

（4）拖动轴可以观察动画。

（5）鼠标悬停可以观看动画微缩图。

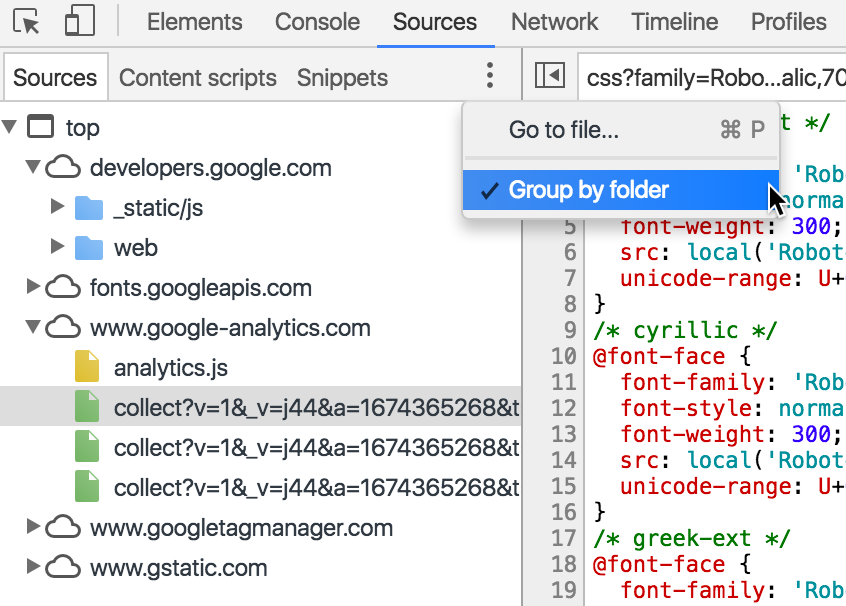
**2.元素事件观察**



（1）ancestors为所有祖先元素绑定的事件。

（2）Framework listeners可以查看自定义事件。

**3.资源**

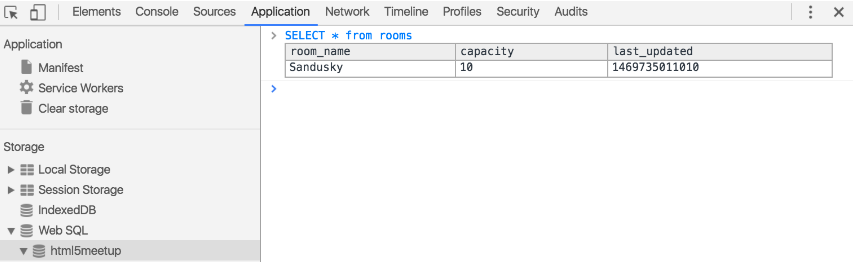


可以选择是否按文件夹查看或仅按框架查看。

**4.应用（Application）**

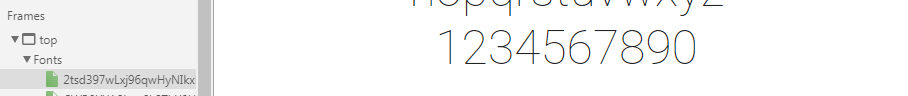
（1）本地存储按k-v存储，可以修改删除等。

（2）Web SQL可以使用sql语句。



（3）可以使用Cookies窗格查看、删除Cookie，但无法修改Cookie值。

（4）使用Frames窗格可以按框架组织页面的资源（跟source中的一样）。



**5.调试js代码**

（1）发现错误。

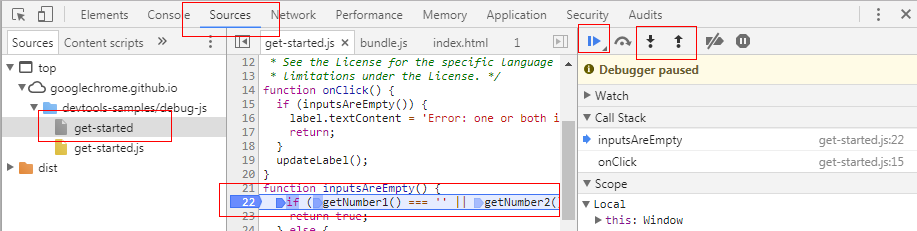
（2）设置断点：

1）点击Sources标签。

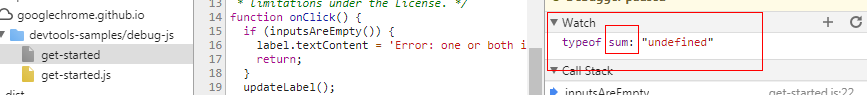
2）如果是事件，点击EventListenerBreakpoints找到事件。如果不是则估计代码错误部分设置。

3）设置断点：点击行号。

（3）逐行调试代码

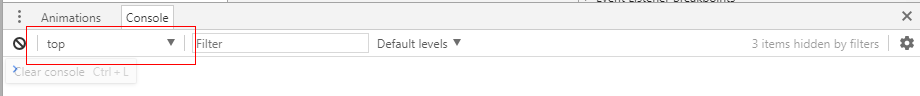


（4）检查变量值。在watch中获取变量值。该值会随着代码运行变化。

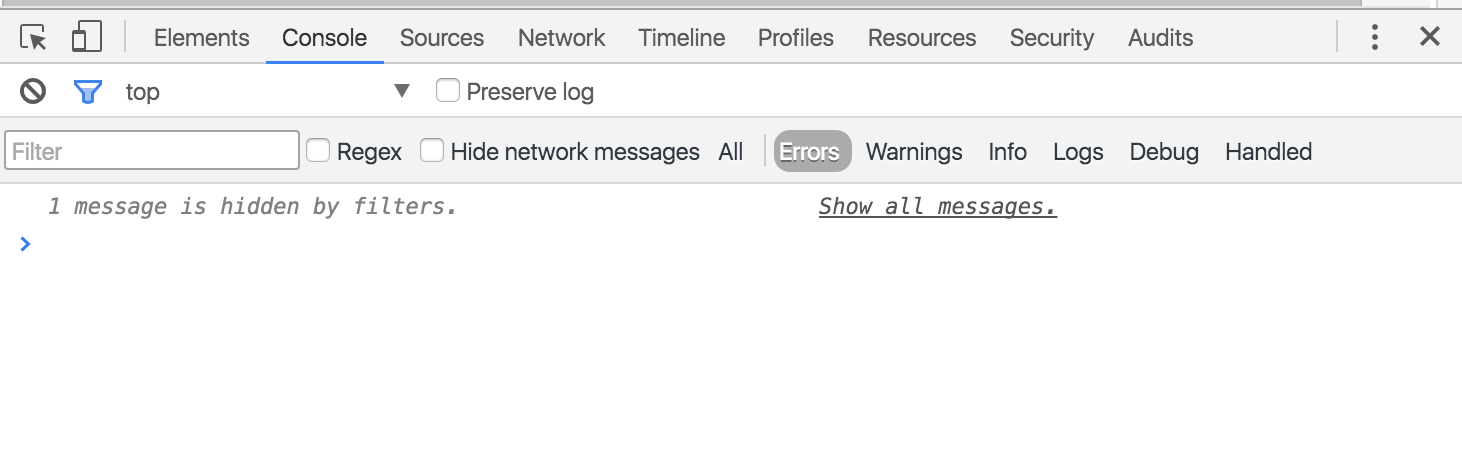


**6.控制台**

（1）选择执行环境。



（2）过滤控制台输出。

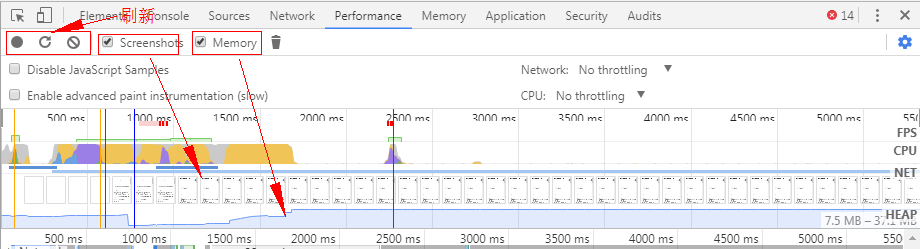


**7.性能分析**

（1）提示

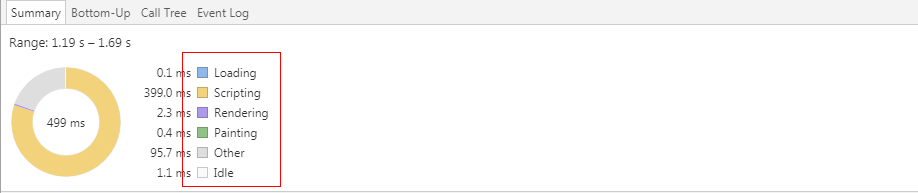
* 减少使用CSS并保持CSS选择器简洁。
* 选择根本不会触发布局的CSS。
* 不要编写会强制浏览器重新计算布局的JavaScript。
* 将JavaScript读取和写入功能分开，并首先执行读取。

（2）进入performance，点击refresh刷新（比较快）。

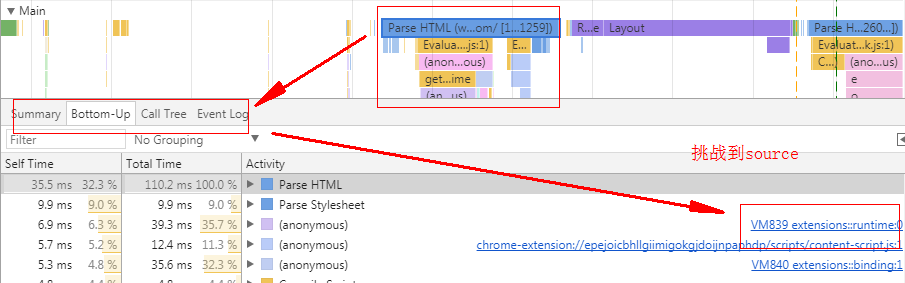


（3）查看GPU分析

* 不同颜色代表不同浏览器流程。

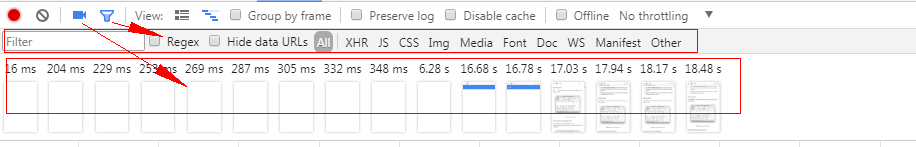


* 可以查看不同函数执行时间，并可以查看函数详情。

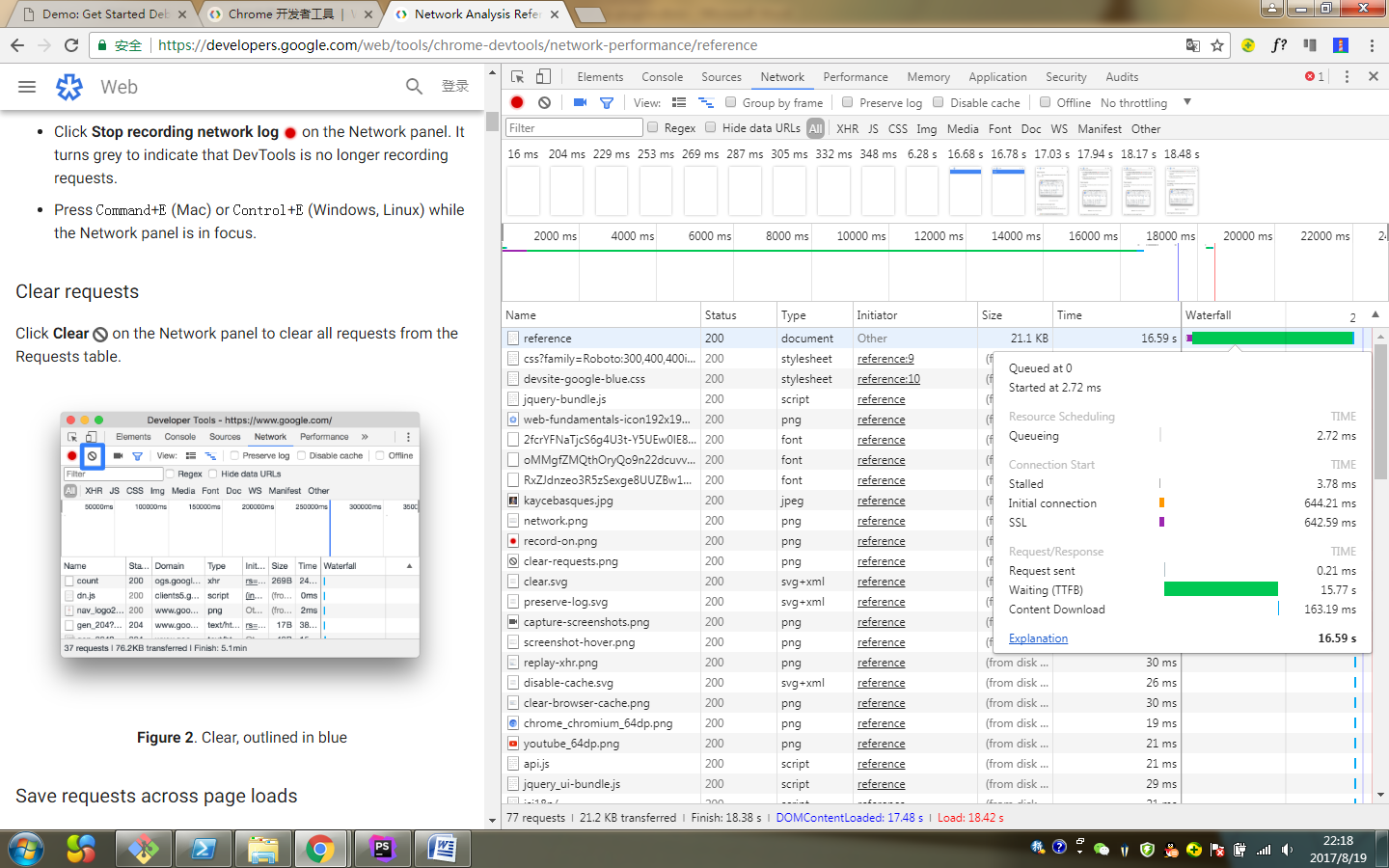


**8.网络**

（1）幻灯片和过滤



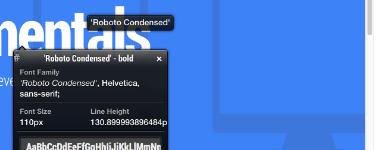
（2）查看网络时序：分为Queuing(排队)、Stalled(停滞)、DNSlookup(DNS查找)，initialconnection(初始连接)，SSLhandshake(SSL握手)、Requestsent(请求发送)、Waiting(等待)（到开始下载第一个字节的时间（TTFB））、ContentDownload(内容下载)。



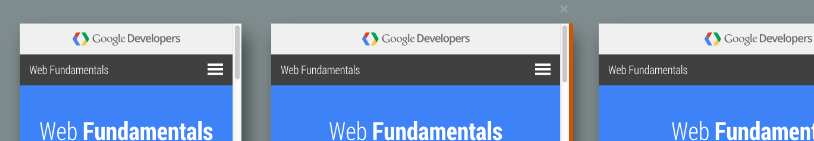
（3）按住Shift并移动鼠标到资源上可查看它的发起者和依赖关系。

**9.插件**

（1）What Font?



（2）Emmet Re:View

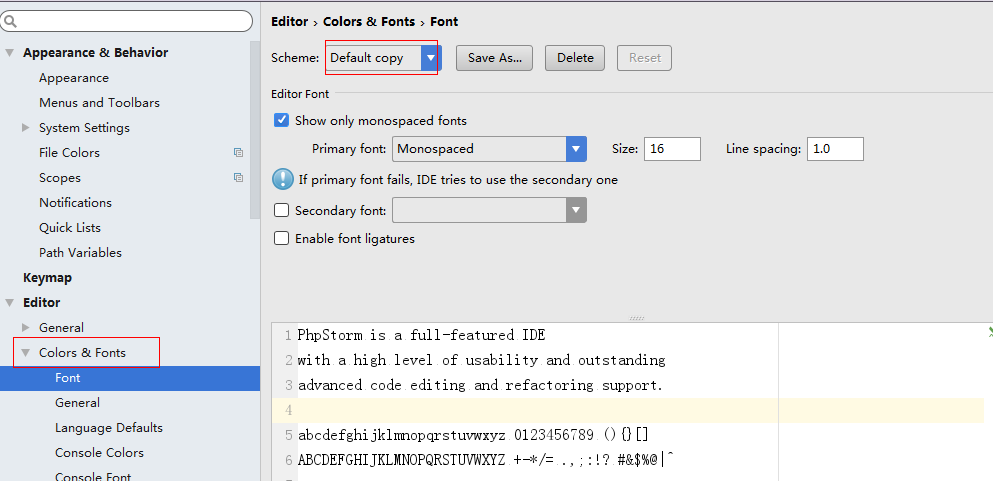


1. phpstorm
   1. ****外观****

有2个外观设置，1个是ide本身，在Appearance&Behabor下。1个是编辑器，在Editor->general下。

（1）代码风格

字体、颜色等涉及众多代码类型，因此按风格设置。



（2）显示行号

settings->Editor->Appearance->Show line numbers

（3）去掉右上角浏览器图标

settings->tools->WebBrowsers

* 1. 功能
     1. 原生支持zen coding（emmet）

（1）初始化

* html:5或!：用于HTML5文档类型
* html:xt：用于XHTML过渡文档类型
* html:4s：用于HTML4严格文档类型

（2）类、id、文本和属性

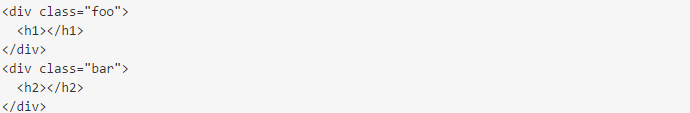
类（.）id（#）文本（{}）属性（[]）。

（3）嵌套

* >：子元素符号，表示嵌套的元素。
* +：同级标签符号。
* ^：可以使该符号前的标签提升一行，可以多次提升。

（4）分组

如输入(.foo>h1)+(.bar>h2)，生成：



（5）隐式标签

如果只输入.item，则Emmet会根据父标签进行判定。

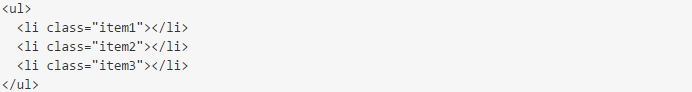
* li：用于ul和ol中。
* tr：用于table、tbody、thead和tfoot中。
* td：用于tr中。
* option：用于select和optgroup中。

（6）多个元素

\*，如ul>li\*3。

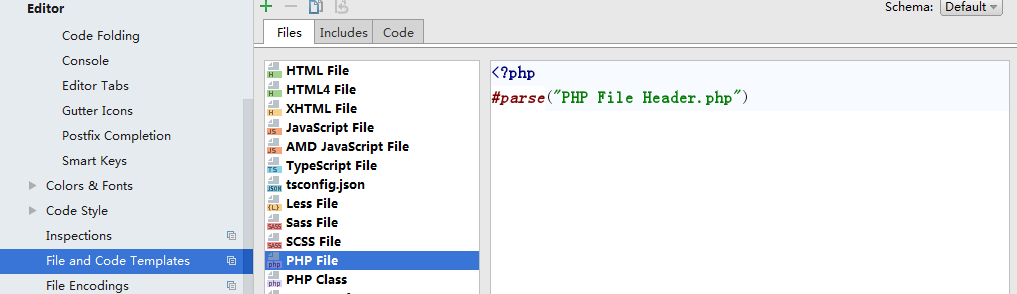
（7）定义多个带属性的元素

如果输入ul>li.item$\*3（$符号可以使用多个）生成：



* + 1. 注释

初始化文件时的模板：



* + 1. 快捷键

（1）查询

|  |  |
| --- | --- |
| CTRL+F | 在当前窗口查找文本 |
| CTRL+SHIFT+F | 在指定路径查找文本字符 |
| CTRL+R | 当前窗口替换文本 |
| CTRL+SHIFT+R | 在指定路径替换文本 |
| 双击SHIFT | 全局搜索文件 |
| CTRL+E | 最近打开的文件 |
| CTRL+N | 查找类 |
| CTRL+SHIFT+ALT+N | 查找php类名/变量名、js方法名/变量名、css选择器 |
| CIRL+B | 查找变量的来源，跳到变量申明处(CTRL+鼠标单击也可以) |

（2）编辑

|  |  |
| --- | --- |
| ALT+←/→ | 标签切换 |
| ALT+↑/↓ | 方法间快速移动定位 |
| CTRL+ALT+←/→ | 返回上次编辑的位置 |
| CTRL+G | 定位行，跳转行 |
| CTRL+/ | 单行注释/取消注释 |
| CTRL+SHIFT+/ | 块状注释/取消块状注释 |
| CTRL+[] | 光标移动到{}[]开头或结尾位置 |
| CTRL+SHIFT+[] | 选中块代码（光标到代码块开始或结尾位置） |
| CTRL+W | 块状选中代码，可以连续按 |
| ctrl+'-/+' | 折叠代码块 |
| CTRL+ALT+L | 格式化代码 |
| CTRL+D | 复制行 |
| Ctrl+Y | 删除行 |
| Ctrl+Backspace/delete | 删除整个字符串（而不再是单个字符） |
| F5 | 复制文件/文件夹 |
| ALT+INSERT | 自动生成代码（如GET/SET方法/构造函数等），光标在类中才生效 |
| Ctrl+O | 覆盖方法（父类中的） |
| Ctrl+I | 实现方法（魔术方法） |
| ctrl+shift+enter | 智能完善代码，如if() |
| CTRL+P | 方法参数提示，显示默认参数 |
| Ctrl+Q | 快速文档查询 |
| Alt+Enter | 生成注释 |

（3）运行

|  |  |
| --- | --- |
| Alt+[0-9] | 打开相应的工具窗口 |
| ALT+7 | 文件结构图 |
| CTRL+H | 显示类层级关系图，继承/实现关系 |
| CTRL+O | 显示所有继承的方法 |
| Alt+Shift+I | 检查当前文件与当前的配置文件 |
| CTRL+D | 选择连个文件，按CTRL+D，可以对比两者的不同 |

（4）本地历史VCS/SVN

|  |  |
| --- | --- |
| Ctrl+K | 提交项目VCS |
| Ctrl+T | 更新项目从VCS |
| Alt+Shift+C | 查看项目最近文件版本变化文件 |
| CTRL+E | 只是查看修改过的文件 |
| Alt+` | 快速弹出VCS菜单 |

* 1. MySQL

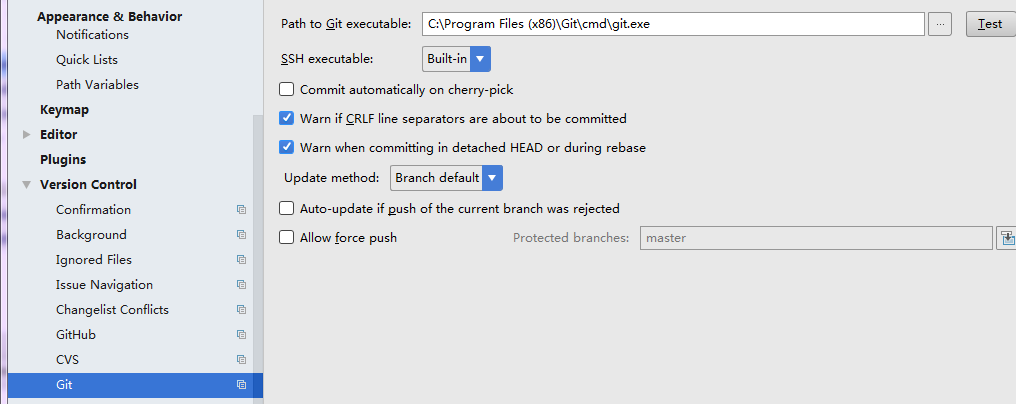
View—>Tool Windows—>Database打开database。选择“+”—>Data Source—>MySQL新建数据库。第一次打开会提示MySQL驱动未安装，点击下载，会自动安装。

之后就可以像使用phpMyAdmin在phpstorm中操作数据库。

* 1. Git

1.修改配置

Files->settings->version control->Git



2.远程克隆文件到PHPSTORM

VCS->Checkout from Version Control->Git

对于已有文件，在Files->settings->version control中添加一下即可，如果之前已在本地设置密码的话，貌似不用重新设置。

* 1. 本地历史

在文件中右键选择Local History，有show history，可进入查看、恢复至该文件之前的版本。有Put Label，可将当前状态保存为一个版本。

也可以在VCS->Local History下查看恢复。

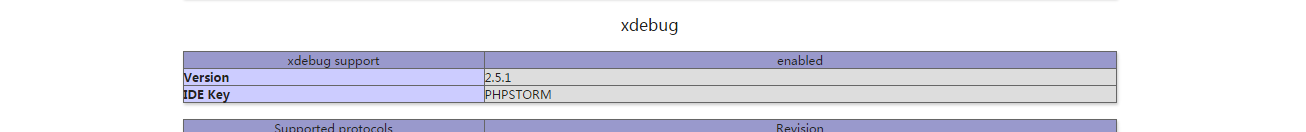
* 1. XDebug

1.登录https://xdebug.org下载与php版本对应的Xdebug，5.6版本win32下载php\_xdebug-2.5.1-5.6-vc11.dll。

2.配置php.ini

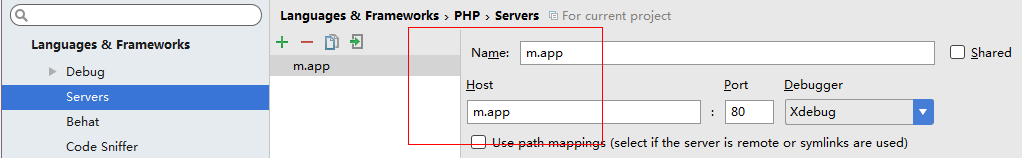
|  |
| --- |
| ;这三行是为了让IDE与XDebug协作  xdebug.remote\_enable = 1  xdebug.remote\_handler = "dbgp"  xdebug.remote\_host = "127.0.0.1"  ;.dll文件的路径  zend\_extension="C:/xampp/php/ext/php\_xdebug-2.5.1-5.6-vc11.dll"  xdebug.idekey="PHPSTORM" |

重启apache，在命令行下输入php，查看是否成功加载php\_xdebug-2.5.1-5.6-vc11.dll。在phpinfo输出中也有xdebug信息。

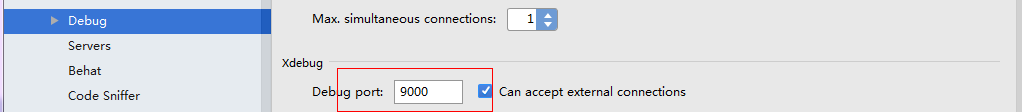


3.设置phpstorm

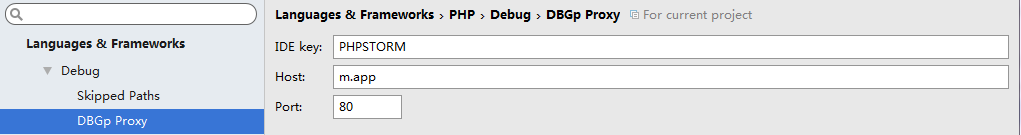
（1）进入File>Settings>PHP>Servers。



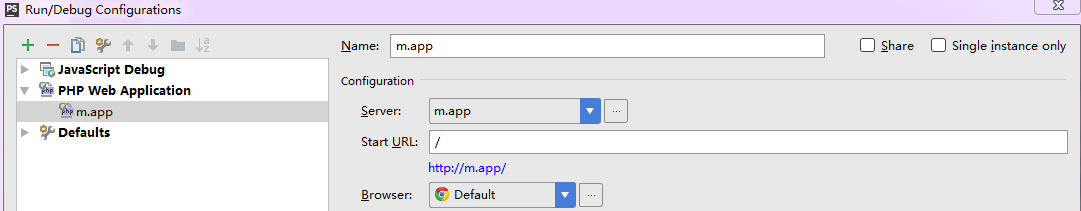
（2）进入File>Settings>PHP>Debug，找到XDebug选项卡。



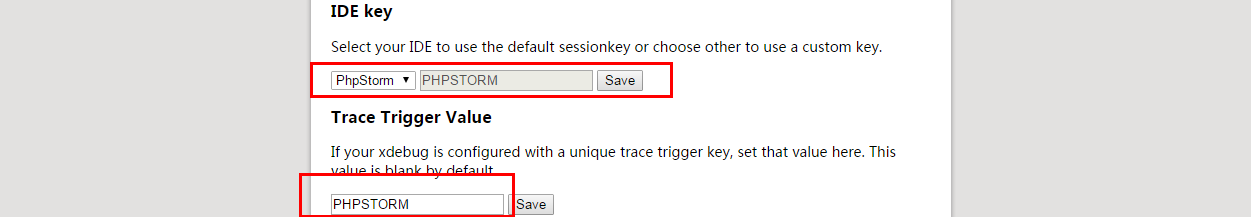
（3）进入File>Settings>PHP>Debug>DBGp Proxy。



（4）点菜单栏的Run>Edit Configurations…在弹出的窗口，点击左上角加号，选择PHP Web Application，添加1个调试配置。

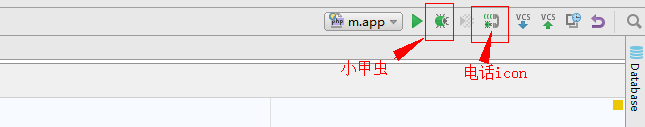


（5）在chrome浏览器下载xdebug helper插件，IDE选PHPSTORM，



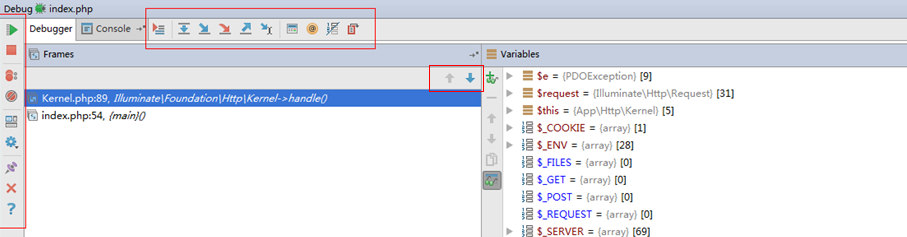
因为设置了idekey，这里加上。

（6）填写完毕后，在代码里设置断点，再开启电话icno监听，点小甲壳虫启动调试。



（7）功能说明

打断点—>点击臭虫—>点击浏览器页面触发断点—>自动跳转回PhpStorm—>查看携带的数据（调试的目的）—>可按步执行查找问题点—>点击运行（或者F5）—>浏览器页面继续执行—>调试完成。



左侧绿色三角形Resume Program，表示將继续执行，直到下一个中断点停止。

左侧红色方形Stop，表示中断当前程序调试。

上方第一个图形示Step Over，跳过当前函数。

上方第二个图形示Step Into，进入当前函数內部的程序（相当于观察程序一步一步执行）。

上方第三个图形示Force Step Into，強制进入当前函数內部的程序。

上方第四个图形示Step Out，跳出当前函数內部的程式。

上方第五个图形示Run to Cursor，定位到当前光标。

Variables：可以观察到所有全局变量、当前局部变量的数值

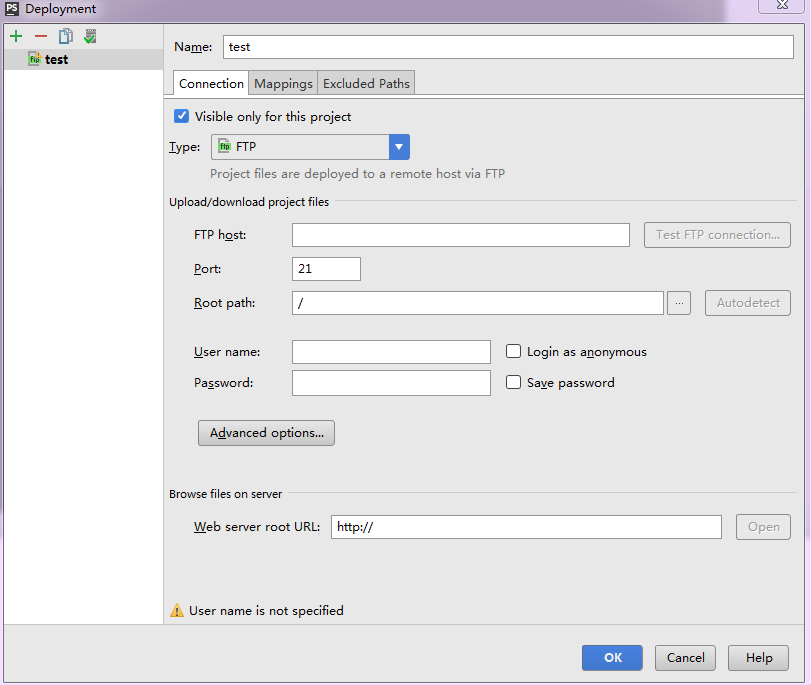
Watches可以新增变量，观察变量随着程序执行的变化。

http://www.cnblogs.com/xujian2016/p/5548921.html

* 1. FTP

1.打开一个本地项目。依次找到Tools—>Deployment—>Cofiguration。

2.新建1个ftp，然后设置。



* 1. 插件

**“file”-“Settings”-“Plugins”。**

* + 1. ****liveedit****

一款即写即浏览的插件，需要配合Chrome使用。安装完成后需要重启使配置生效。还需要按提示在chrome中安装插件。

* + 1. CodeGlance

代码地图插件，在编辑器右侧生成代码地图。

* 1. 其他
     1. 配置的导入和导出

“File”-“Export Setting/Import”

* + 1. 设置文件编码

“File”-“File Encoding”

* + 1. 新建远程项目

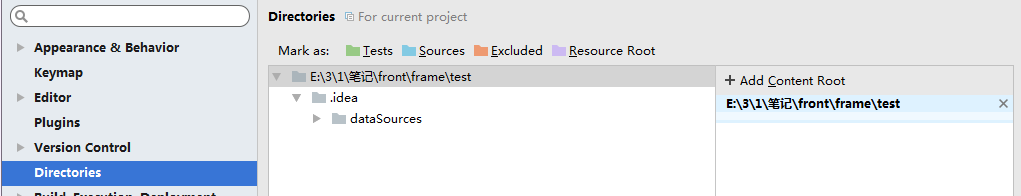
“Tools”-“Deployment”-“Brower Remote Host”

* + 1. 内置服务器

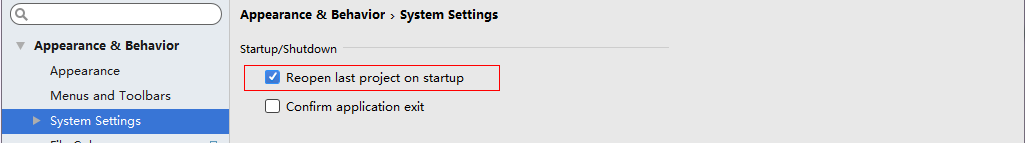
PhpStorm内置的网络服务器地址：http://localhost:63342/projectName/\*.html

* + 1. 多项目并存

File->settings->Directories->Add Content Root中添加工程目录。



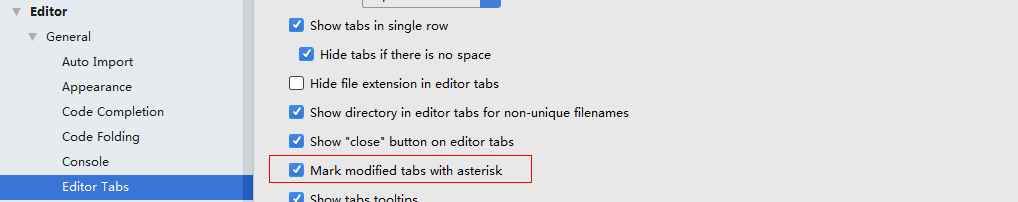
* + 1. 启动的时候不打开工程文件



* + 1. 取消自动保存

在File->settings->Apperance&Behavior->System Setting->Synchronization下设置。

* + 1. 将编辑过的文件tab加星号标识



1. cordova
   1. 命令行
      1. 基本



**1.全局命令**

（1）create：创建1个项目。

（2）help：获取帮助。

（3）telemetry：打开或关闭遥测收集。

**2.项目命令**

只在有效的cordova项目目录下有效。

（1）info：生成项目信息。

（2）requirements：检查项目平台环境依赖安装情况。

（3）platform：管理项目平台。

（4）plugin：管理项目插件。

（5）prepare：复制文件到平台中以备building。

（6）compile：Build平台。

（7）clean：清理构建工件。

（8）run：等价于prepare+compile。

（9）serve：在本地服务器上运行。

**3.通用options**

适用于所有命令。

（1）-d/--verbose：输出更详细信息。

（2）-v/--version：显示cordova-cli版本。

（3）--no-update-notifier：禁用更新检查。

（4）--nohooks：禁用钩子（不执行）。

（5）--no-telemetry：禁用遥测。

**4.平台相关options**

属于某些平台相关的option，可以通过--分隔符传给cordova，可以停止cordova-lib模块内的命令解析，并使用后面的选项来进行解析。



* + 1. 创建命令

**1.语法**



（1）path：项目所在目录，由cordava自动创建，之前不应该存在。

（2）id：映射到config.xml中的widget元素的id属性的反向域样式标识符。默认值为io.cordova.hellocordova。

（3）name：应用程序的显示标题，映射到config.xml文件中。默认值为HelloCordova。

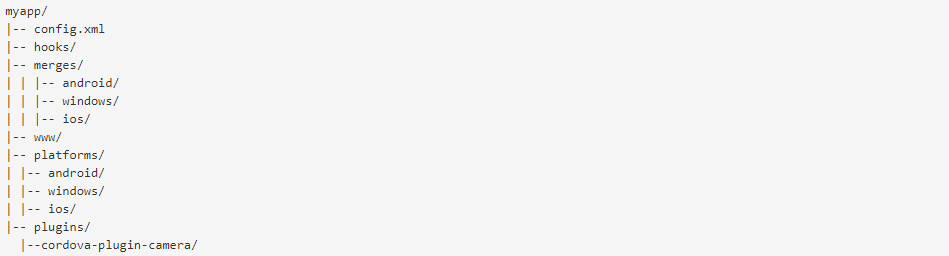
（4）config：json形式配置，将被保存到<path>/.cordova/config.json中。

**2.Options**

（1）--template：使用自定义模板。

（2）--link-to：www目录的符号链接。

**3.目录结构**



（1）config.xml：配置文件。

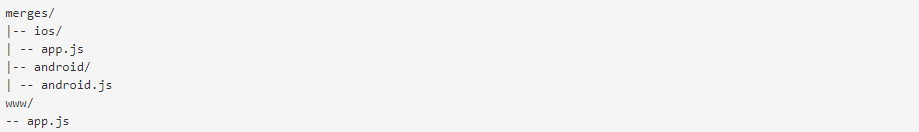
（2）www：项目的Web组件。将prepare命令复制到每个平台的www目录下，如果platforms/ios/www、platforms/android/assets/www等。

（3）platforms：添加的平台所有源代码和构建脚本，切勿手动修改。

（4）plugins：添加的插件将被提取或复制到此目录中。

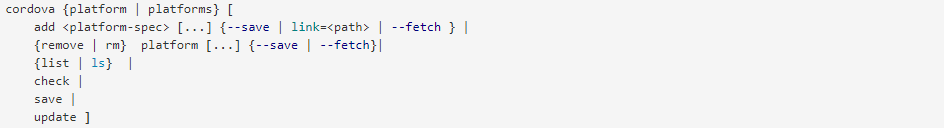
（5）hooks：用于定制cordova-cli命令的脚本。

（6）merges：特定平台的Web组件，将替换www下匹配文件。



* + 1. 平台相关命令

用于添加、删除、更新、列出和检查更新cordova平台。



（1）Platform-spec

平台特定值。



platform：如ios、andriod。

version：版本。

path/url：包含平台的路径。

commit-ish：默认为master分支，这里可以设置特定分支。

（2）add/remove中的命令参数

--save：将<platform-spec>保存到config.xml中。

--fetch：使用npm install并将平台保存到node\_modules目录下。

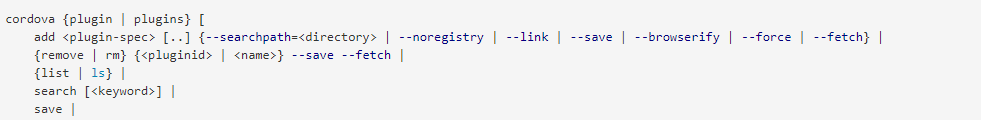






* + 1. 插件命令

管理项目插件。



（1）Plugin-spec



scope：作为范围npm包发布的插件的范围。

directory/url：包含plugin.xml的目录或git仓库。

（2）add/remove中的命令参数

--searchpath：当通过ID查找插件时，在登录注册表之前查看此目录及其每个子目录。

--noregistry：不要在注册表中搜索插件。

--link：从本地路径安装时，创建一个符号链接，而不是复制文件，用于插件开发。

--save：将<plugin-spec>保存到config.xml中。

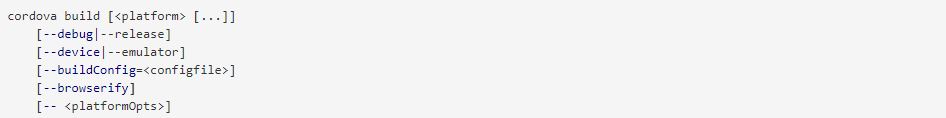


* + 1. prepare命令



* + 1. build命令

build命令运行使用钩子。



--debug：执行调试版本。

--release：执行发布构建。

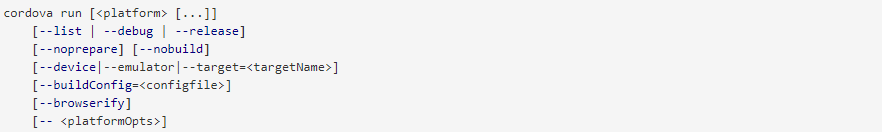
--device/--emulator：构建1个设备/模拟器。

--buildConfig=<configFile>：使用指定的构建配置文件。默认为根目录下的build.json。

<platformOpts>：平台相关的特定命令，应在--后使用。



* + 1. run命令



* + 1. emulate命令



* + 1. clean命令



* 1. Config.xml

全局配置文件。当使用CLI构建项目后，该文件的版本会复制到各个platforms/子目录中。



id：应用程序的反向域标识符。

name：应用程序的正式名称，显示在设备的主屏幕和应用商店界面中。

content：应用程序的起始页。默认值为index.html，通常在项目的顶级www目录中。。



access：访问域：定义应用程序允许与之通信的一组外部域。上面显示的默认值允许它访问任何服务器。

allow-intent：控制应用程序允许哪些URL要求系统打开。默认情况下，不允许使用外部URL。

plugin：使用--save添加插件时，该元素将自动添加到项目中。

* 1. 事件

cordova提供的各种事件，可以在js中给这些事件添加侦听器。



提供了如下事件：

deviceready：应用被转载。至关重要。

pause：当本地平台将应用程序置于后台。

resume：当本地平台将应用从后台拉出时。

下面的只有安卓支持：

backbutton：当用户按下后退按钮时，事件触发。

menubutton：当用户按下菜单按钮时，事件触发。

searchbutton：当用户按下Android上的搜索按钮时，该事件触发。

volumedownbutton：当用户按下音量按钮时，该事件触发。

volumeupbutton：当用户按音量调高按钮时，该事件触发。

* 1. 插件API

**1.电池状态**

当电池充电百分比变化至少1％时，或者当设备插入或拔下电源时触发batterystatus事件。