1. gulp

gulp使用vinyl-fs虚拟文件系统，即，依赖glob匹配文件，使用vinyl虚拟文件描述类，将文件流转为文件集。

* 1. 语法
     1. node-glob

glob(pattern, option, fn)，匹配pattern的文件，将会被传入fn处理，如果没有匹配到时传入空。

pattern的规则有：

|  |  |
| --- | --- |
| \* | 匹配0或多个字符，但不包括/，即不能跨路径匹配。 |
| \*\* | 跨路径匹配，即能匹配到/，但匹配不到文件。 |
| ? | 单个字符。 |
| [] | 任意范围。 |
| \*(p|p|p) | 0或多个p。 |
| !(p|p|p) | 不包括任何p。 |
| ?(p|p|p) | 0或1个p。 |
| +(p|p|p) | 1个或多个p。 |
| @(p|p|p) | 任意1个p。 |
| {} | 展开语法。 |

如下，\*\*匹配不到文件，必须还得用\*。



* + 1. gulp.src(globs[, options])

将匹配到的文件返回1个文件流，有多个模式时使用数组。

* + 1. gulp.dest(path[, options])

将stream写入path，如果文件不存在则新建。path可以是function，应返回1个path。文件默认权限是0777。写入文件后，stream仍然可用。

如果是相对路径，则相对本目录。且dest中的path会加上src中的path中通配符部分，如：

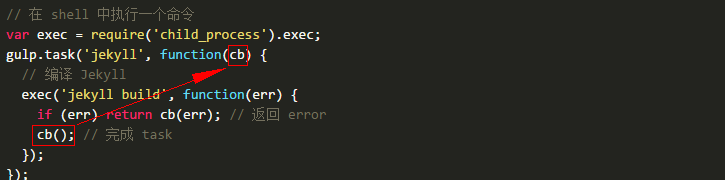
|  |  |  |
| --- | --- | --- |
| src(a/b.js) | Dest(dist/fo.js) | Dist/fo.js/b.js |
| Src(a/\*\*/b.js) | Dest(dist) | Dist/\*\*/b.js |

* + 1. gulp.task(name[, deps], fn)

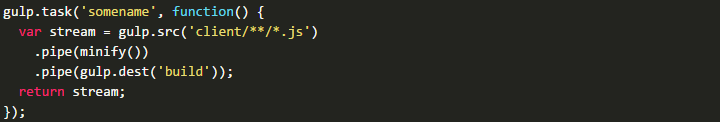
定义1个任务。name不能为空。deps为任务依赖。fn为操作。

如果没有依赖，则按定义顺序执行。如果为异步，则不会等待依赖完成之后再执行（依赖只是执行顺序，而异步则是执行的完成时间，因此依赖和异步都需要考虑），需要：

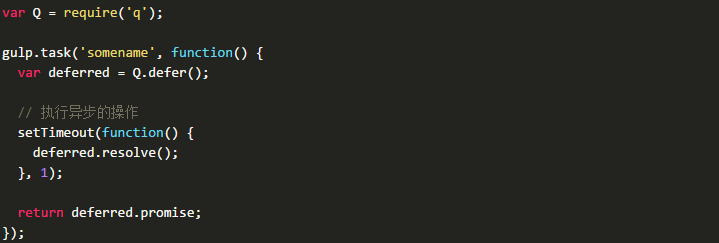
（1）在异步执行完成后，执行1个回调通知gulp这个异步已执行完。此时，该回调应为fn的第1个参数。



（2）任务返回1个stream。



（3）任务返回1个promise对象。



* + 1. gulp.watch

有2种形式：

（2）gulp.watch(glob[, opts], tasks)

glob为监视那些文件变动，可以为数组。tasks为任务，为1个数组。



（2）gulp.watch(glob[, opts, cb])

cb(event)为回调函数。event.type为added、changed、deleted。event.path为触发了该事件的文件的路径。

* 1. 常用插件

（1）gulp-load-plugins

自动加载插件，用于在gulpfile.js文件中引入过多插件时使用。将原在require()中的插件放入package.json中。

（2）gulp-rename

重命名stream中的文件名。

（3）gulp-uglyfy/minify-css/minify-html

（4）gulp-jshint

代码检查。

（5）gulp-concat

文件合并。

（6）gulp-less/sass

（7）gulp-imagemin

压缩图片。

（8）gulp-liveload

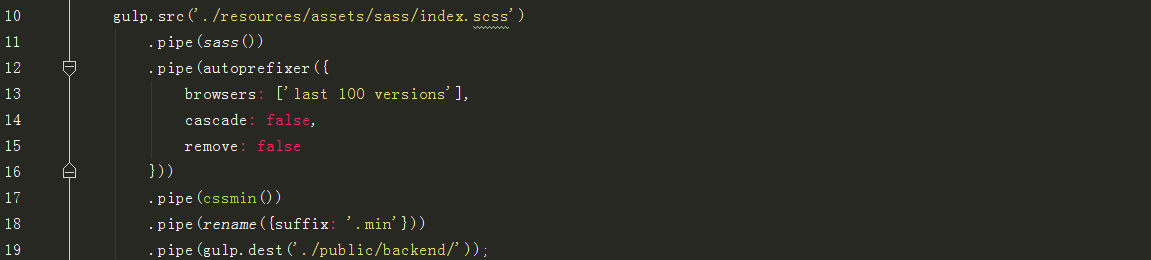
自动刷新页面，chrome需要装插件。

（9）gulp-newer

可用于增量build。

（10）gulp-remember/cached

用于还原stream。



* 1. gulp4和gulp3的不同

（1）4不在task中写依赖，而是用series和parallel两个新的api实现。

（2）由于多个串行不会对相同代码部分抽出而导致重复，如：

gulp.task('a', gulp.series('x', fn))

gulp.task('b', gulp.series('x', fn))

gulp.task('c', gulp.parallel('a', 'b'))，此时，任务x将会被执行两次，需要人为抽出x，如gulp.task('c', gulp.series('x', gulp.parallel('a', 'b')))，此时任务a、b中将不再写对x的串行依赖。

（3）gulp.series/parallel返回的是函数，可以嵌套使用。

（4）watch返回1个观察对象，可以用on来监听，但3中的方法仍然可用。

（5）fn可以是函数变量，而不一定是回调函数了。

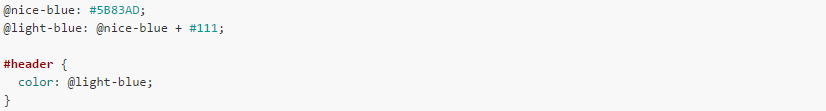
1. less

参考网址：http://www.css88.com/doc/less/features/。

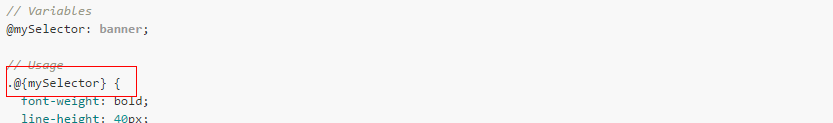
* 1. 变量

变量提供一种在一个地方管理这些重复多次值的方法，让代码变得更容易维护。

less中的变量其实是常量，每个变量只能定义1次。



变量不仅可用于css中，还可以作为属性名、url、引入文件。





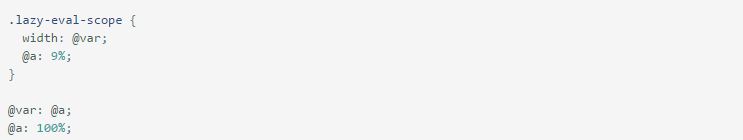




对变量名可以再次解析，如@a='b'，@c='a'，可以使用d:@@c。



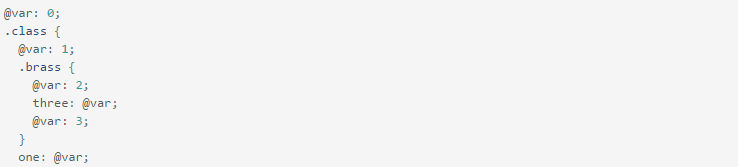
变量是延迟加载的，在使用前不一定要预先声明。



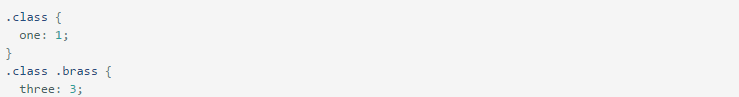
最终都会编译为：



在定义一个变量两次时，只会使用最后定义的变量，Less会从当前作用域中向上搜索。这个行为类似于CSS的定义中始终使用最后定义的属性值。

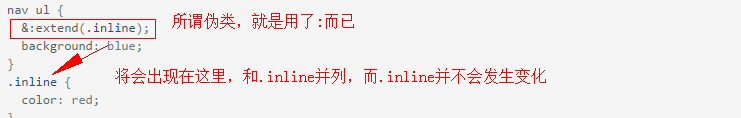


会被编译为：

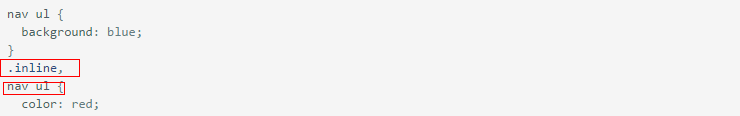


* 1. extend

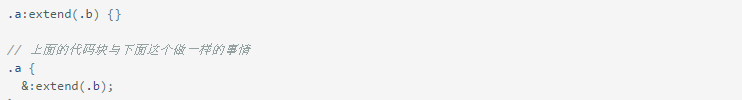
extend是一个Less伪类，它会在将在它引用的类出现的地方，将使用它的类并列在一起（用“,”，即，和引用的那个类一样）。如，



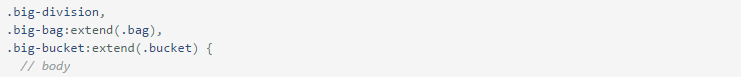
将会编译为：



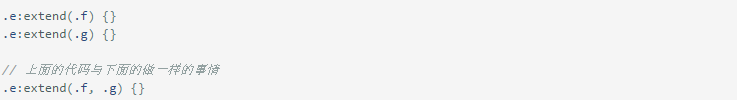
* + 1. 语法



如果extend()中为空，将被忽略。:extend和前面的选择器可以带空格。存在多个选择器时，都可以带extend。



可以有多个扩展。



多个扩展可以连写，但必须放在选择器尾部。也可以对css伪类扩展。如：pre:hover:extend(div pre):extend(.bucket tr)，等价于pre:hover:extend(div pre, .bucket tr)。但不允许pre:hover:extend(div pre).nth-child(odd)，因为扩展必须放在最后。

* + 1. 规则集内的扩展

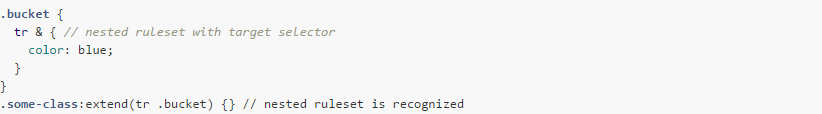
可以使用&:extend(selector)语法，在规则集内置入extend。



等价于：



* + 1. 嵌套选择



本质上，extend查找的是编译后的CSS，而不是原始的less。

* + 1. 精确匹配

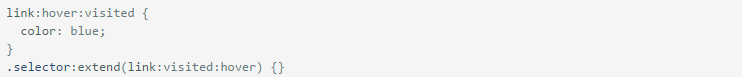
extend默认会精确匹配引用的选择器。



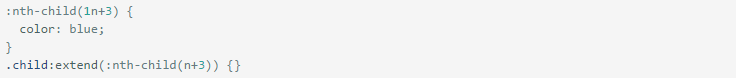
即使extend引用的选择器在css中是否等价，如.class和\*.class在css中等价，但less无法根据1个匹配另外1个。



伪类的顺序必须严格一致，否则也无法匹配。



Nth表达式1n+3和n+3是等价的，但extend也并不能匹配它们。



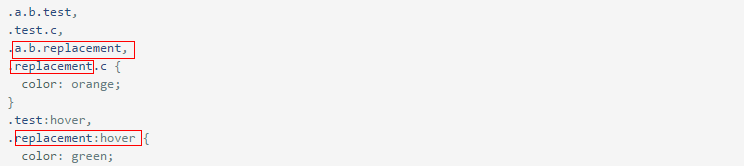
引号类型也是严格匹配，如下面3个伪类在css中等价，但extend仍然会分别匹配。



* + 1. 关键字all

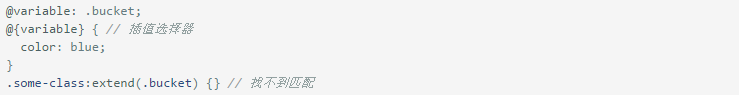
在extend参数尾部使用关键字all，使用它的选择器将会在引入的选择器的位置完全复制1份。

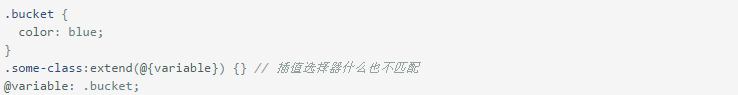




* + 1. 变量选择器

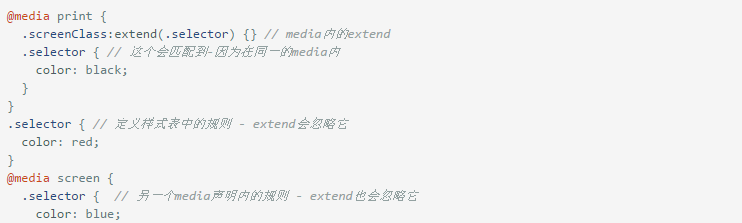
Extend不能匹配变量选择器。如果选择器包含变量，extend会忽略它。





* + 1. 作用域/@media内的扩展

编写在media声明内的extend也应该只匹配同一media声明内的选择器。



在media声明内的extend也不会匹配嵌套声明内的选择器。

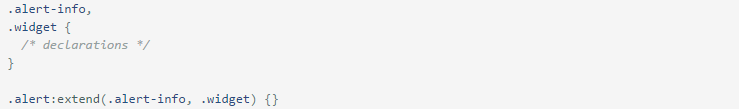


顶级extend匹配一切，包括media嵌套内的选择器。

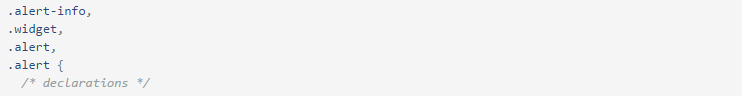


* + 1. 不会检测重复

extend不会检测重复。



会编译为：



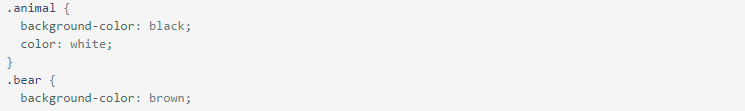
* + 1. 使用场景

（1）避免添加基础类

如有1个基础的animal类，有background-color和color两个属性。要新定义1个bear类，只改其中的background-color属性。

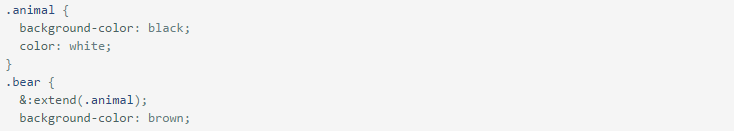
方法1，定义2个类：



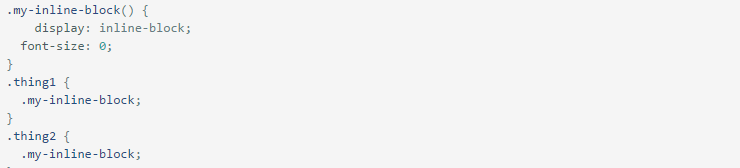


方法2，使用1个bear类：

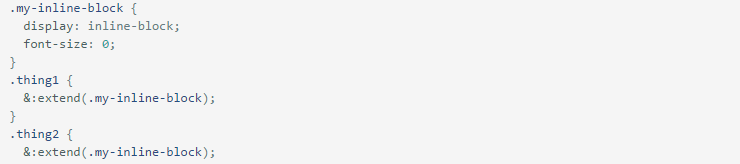




（2）减少Mixins复制导致的不必要的重复

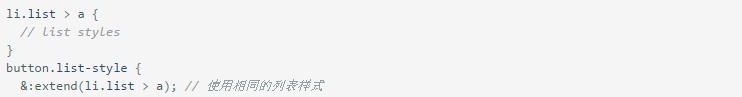


thing1和thing2类中的内容将完全一样。



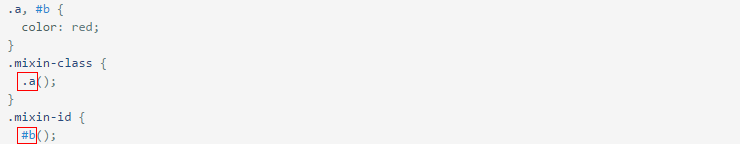
（3）合并样式

如果有两个不同的块，但这两个块需要应用相同的样式，可以使用extend来关联这两块。

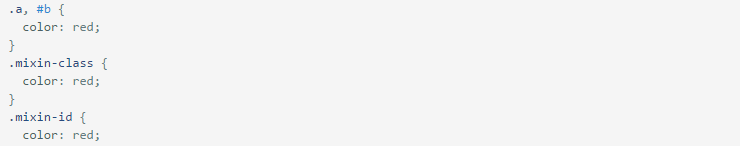


* 1. 混合（Mixin）

可以将现有的样式混合到任何class、id、元素样式中。



混合可以是类选择器或者id选择器，混合与extend不同，它将复制到使用它的地方。



使用混合集的时候，括号可加可不加。

* + 1. 不输出混合集

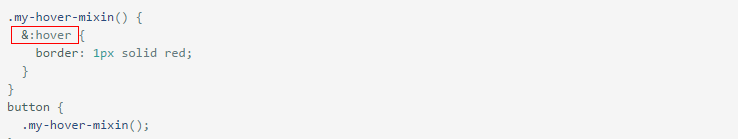
如果想要创建一个混合集，但不想让它输出到样式中，可以在混合集的名字后面加上一个括号。



my-other-mixin本身将不会出现在最终的样式表中。

* + 1. 带选择器的混合集

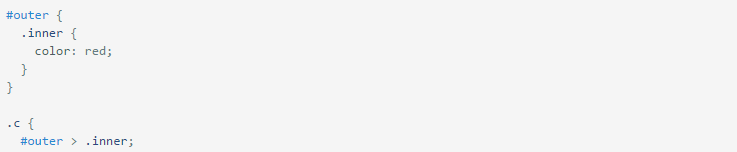
混合集内除了可以包含各种属性，还可以包括各种选择器。



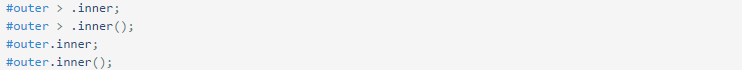


* + 1. 命名空间

如果你想要将属性混合到比较复杂的选择器中，可以通过嵌套多层id或者class。



>和.等价。



* + 1. !important关键字

在混合后面加上!important关键字表示将混合带来的所有属性标记为!important。

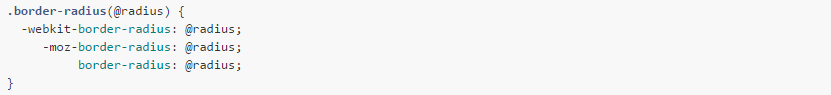






* + 1. 带参数的混合

定义通用的class时，可以像函数一样带参数（可以设置默认值）。





* + 1. 带多个参数的混合

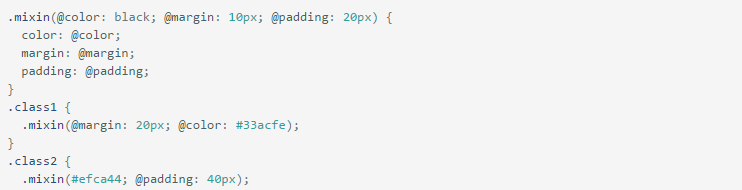
传参时，如果有多个参数，如果都是单属性值css样式，可以用,隔开。如果参数中有css列表，要用;隔开：

* 2个参数，每个参数都含有通过逗号分隔的一组值：@name(1,2,3;4,5)。
* 3个参数，每个参数都是单属性值，用,分开：@name(1,2,3)。
* 1个参数，参数是1组值，末尾用;象征性分割：@name(1,2,3;)。
* 1个参数，参数是1组默认值：.name(@param1:1,2;)。
  + 1. 同名混合

定义同名且参数数量相同的混合是合法的。Less会使用每个mixin中可以使用的属性。即，满足强制个数（即剔除默认值）的混合都将被使用。

* + 1. 命名参数

使用mixin时可以通过参数名称而不是参数的位置来为mixin提供参数值。



* + 1. @arguments变量

@arguments表示全部参数。

* + 1. 高级参数和@rest变量

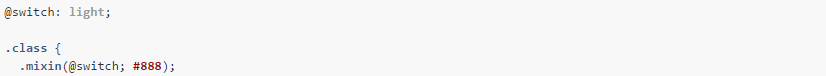
如果希望mixin接受数量不定的参数，可以使用...。在变量名后面使用它，它会将这些参数分配给变量。

* + 1. 模式匹配

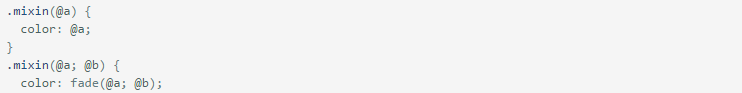
即，调用mixin时，使用变量作为参数,通过参数值控制mixin行为。



@\_代表所有mixin。使用mixin时，先定义1个变量，然后将变量传入mixin。



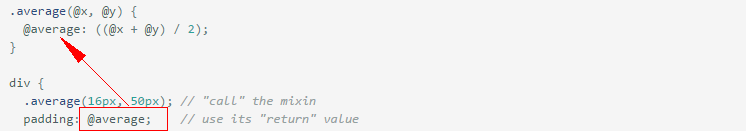
也可以基于参数数量来匹配mixin。



此时，如果用一个参数来调用.mixin，这将会输出第一个定义，如果使用两个参数调用它，这回获取第二个定义。

* + 1. 作为函数使用的混合

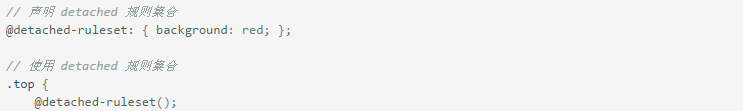
利用定义在一个mixin中的变量都是可见的这一特性，在调用它的作用域中使用这些变量，达到类似函数返回值的效果。





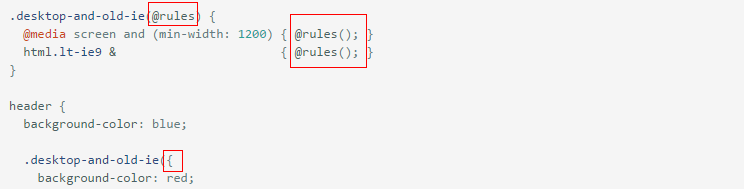
* + 1. 传递规则集给混合

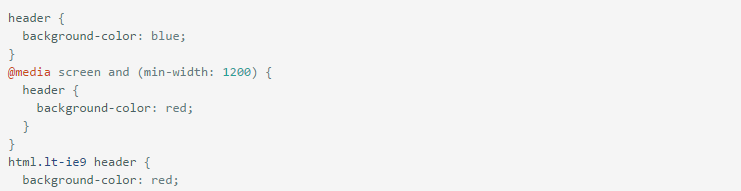
即，将规则集保存到1个变量中，再将其作为mixin的1个参数（当然也可以用在其他规则集中，将会被复制过去）。



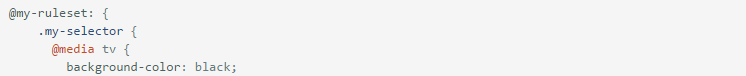
调用时分离（detached）规则集合后面的圆括号是必须的，@detached-ruleset;这样调用无效。

分离规则集合在mixin下的媒体查询用处：





分离规则集合内可以有任何less语法，包括mixin和媒体查询。

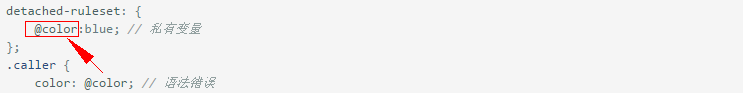


分离规则集合中如有有mixin，它也可以像作为函数的混合那样返回混合。





但如果是分离规则集合内的私有变量，则不会返回。



* + 1. 作用域

跟其他编程语言非常类似，首先会从本地查找变量或者混合模块，如果没找到的话会去父级作用域中查找，直到找到为止（且变量声明可以在使用之后）。



* 1. 导入

如果导入的文件是.less扩展名，则可以将扩展名省略掉。

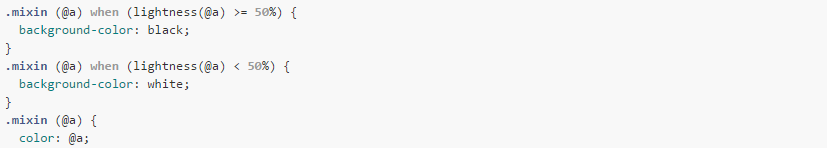


* 1. 引用

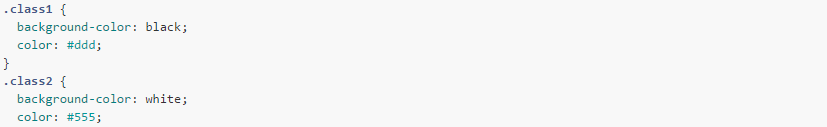
不添加导入的样式到编译输出中，只作为mixins或者extended的样式被使用。对于extend，它插入引用@import语句的位置。对于mixins，正常出现在引用它的地方。



* 1. 引导（Guards）







通过关键词when，引入了一个guard条件。true关键字是唯一被判断为真的值，即，下面两个相等。



Guards支持的运算符包括有>/>=/=/=</<。

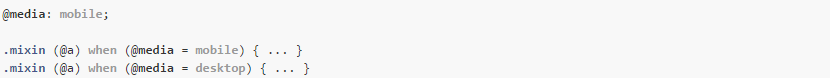
多个Guards可以通过逗号表示分隔，如果其中任意一个结果为true，则匹配成功：



不同的参数之间也可以比较。



比较条件也可以不是参数。



如果需要根据值的类型匹配混合，可以使用iscolor、isnumber、isstring、iskeyword、isurl、ispixel、ispercentage、isem函数。



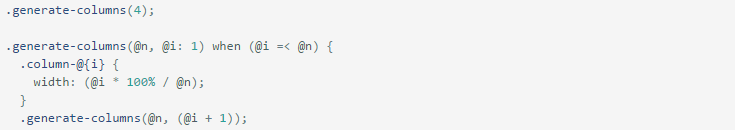
还可以使用and、not关键字。



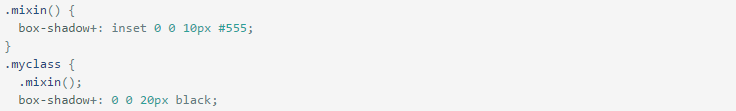


* 1. 循环

当一个混合递归调用自己，再结合Guard表达式和模式匹配这两个特性，就可以写出循环结构。使用递归循环最常见的情况就是生成栅格系统的CSS：



* 1. 合并

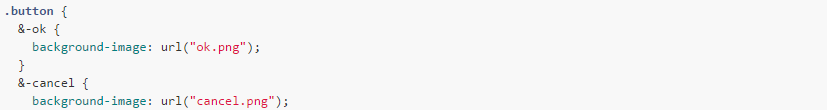


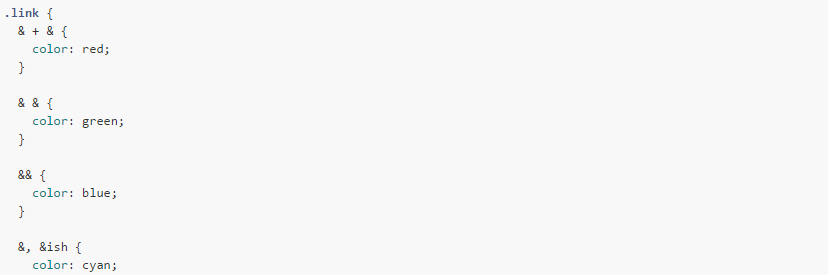


为避免任何非有意的添加，merge需要在每个待加入的声明中显示的设置一个+或者+\_标记。

* 1. &的高级用法

&本质为父选择器，如下面这些用法：





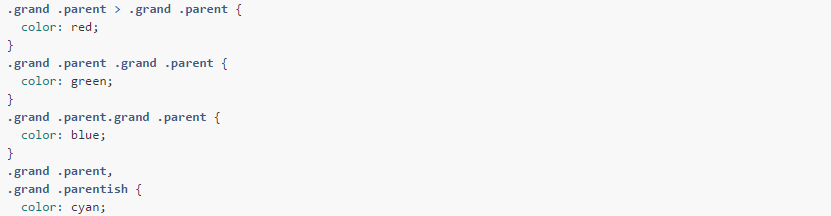
编译后为（外层的没有了）：



在多级嵌套中，&代表所有层级的父级。



此时的&代表.grand .parent（同样，最外层的没有了）。

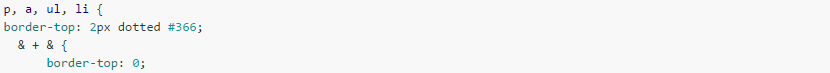


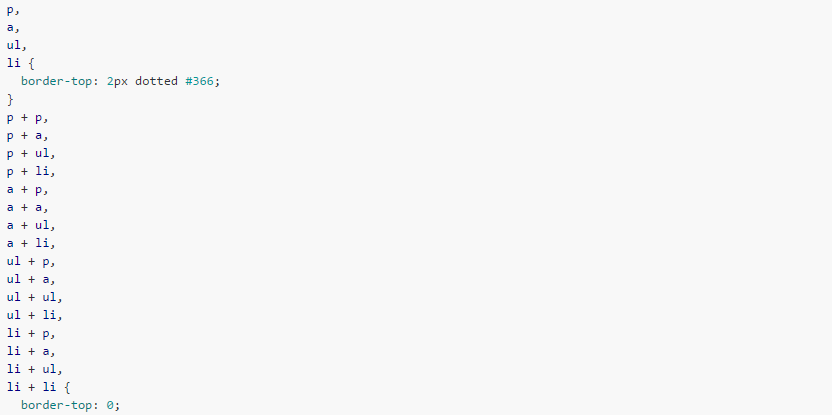
&还可以用来改变嵌套顺序：





&还可用于选择器的笛卡尔积：





* 1. 嵌套

类似dom嵌套。&符号代表串行而不是后代，对于伪类非常有用。



* + 1. 媒体查询嵌套

对于Media query也可以使用同样的方式进行嵌套。



* 1. 运算

任何数字、颜色或者变量都可以参与运算，运算应该被包裹在括号中。



less能够分辨颜色和数字单位，如下，结果将为6px。



* 1. 函数

LESS提供了多种函数用于控制颜色变化、处理字符串、算术运算等。

http://www.css88.com/doc/less/functions/

* 1. 其他
     1. 转义

有时候当需要引入一个值，但它是无效的CSS语法或者LESS不能识别，通常是一些IE的Hack，为避免抛出异常并破坏LESS，需要对他们转义。

less中可用~"value"对value转义，编译完成后在css文件里就是value，没有引号。

1. backbone
   1. 入门

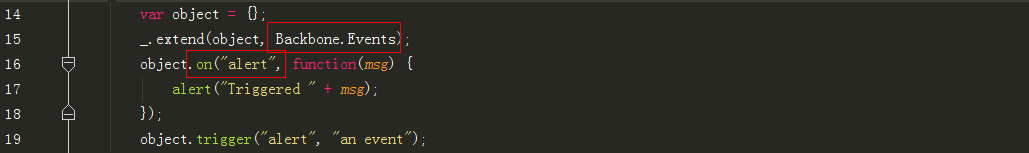
Backbone主要提供查询、操作数据的有用方法，而不是HTML小组件。与传统MVC对应关系如下：

* Backbone.Model：包裹一行数据的业务逻辑。
* Backbone.Collection：客户端上的一组Model，具有排序、过滤、聚合逻辑。
* Backbone.View：可重复使用的UI，但并非总是与Model相关联。
* Backbone.Router：路由+控制器，将URL映射到操作函数中。

Backbone依赖Underscore.js。基于RESTful的约束，histroy依赖于Backbone.Router。DOM操作依赖于Backbone.View，并进一步依赖于jQuery。Backbone不强制使用某个模板引擎。

* 1. 事件

事件是可以混合到任何对象的模块，不必在绑定之前被声明。可以传递{silent:true}作为参数，阻止执行回调函数。



**1.事件方法**

（1）on

给对象绑定回调函数。可以使用冒号作为命名空间，使用空格分隔多个事件。



可以使用关键字all，当任何事件触发时，都会触发all绑定的回调。



（2）off

移除事件绑定的回调函数。如果没有指定事件，所有事件的回调将被移除。

（3）trigger

（4）once

（5）listenTo

侦听另一个对象上的特定事件。



（6）stopListening

（7）listenToOnce

**2.事件类型**

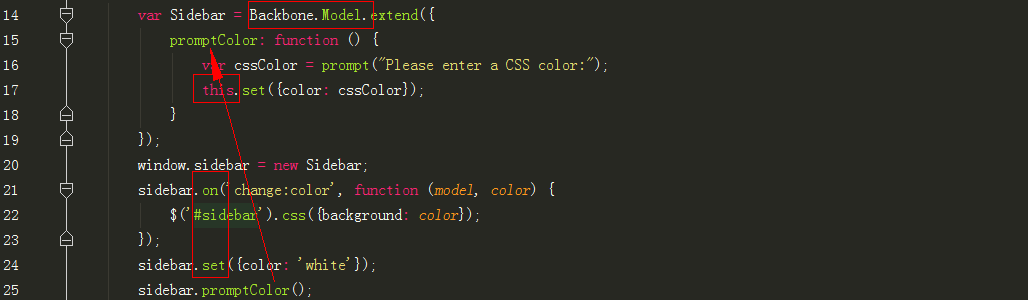
对于模型、集合、视图，有如下事件类型：add、remove、reset、sort、change、sync等。

* 1. Model

模型是任何JS应用的核心，包括数据交互及相关的逻辑。

**1.创建**

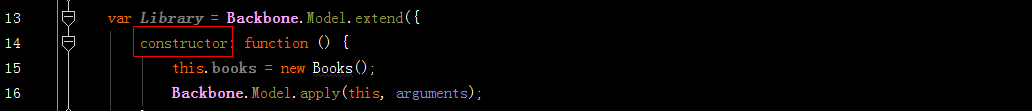
使用extend方法要创建Model类。



还可以使用extend设置原型链，创建子类。

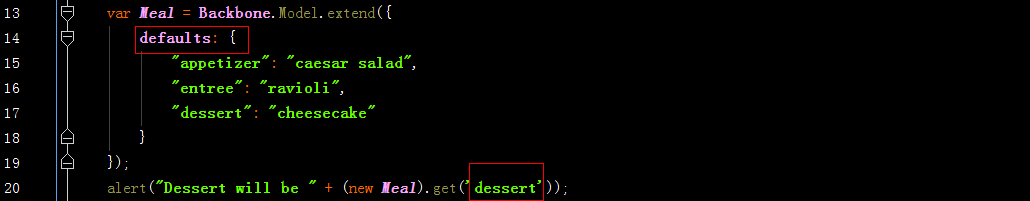


如果在创建时，传入initialize属性，该函数会在model创建后执行。如果定义constructor，可以替换model默认构造函数。



如果传入collection属性，将指定这个model属于哪个collection，否则将在你第一次添加model到一个collection的时候被自动添加。

如果传入defaults属性，在创建模型实例时，任何未指定的属性会被设置为defaults中设置的默认值。



如果传入parse属性，通过fetch获取模型数据，或save时执行。默认实现是自动进行的，仅简单传入JSON响应。传入原始response对象，应当返回可以直接set到模型的属性散列表。

如果传入validate属性，自定义模型验证逻辑。默认情况下validate在save之前调用，但如果传递了{validate:true}，也可以在set之前调用。如果属性是有效的，validate不返回验证任何东西;如果它们是无效的，返回一个你选择的错误。

以上值也可以通过实例获取。

**2.属性**

（1）id

特殊属性，可通过id从集合中获取model，还用于生成model的URLs。

（2）idAttribute

model的唯一标示符，储存在id属性下。

（3）cid

model创建时自动产生的唯一标识符。

（4）attributes

JSON对象形式表示的模型数据。采用set更新而不要直接修改。使用\_.clone(model.attributes)获取模型属性副本，。

（5）validationError

用validate最后验证失败时返回的值。

**3.实例方法**

（1）get

获取属性。

（2）escape

与get类似，返回转义HTML后的model属性值。

（3）set

设置一个或多个属性（被映射到内部属性散列表中）。如果没有传入{silent:true}，会触发change事件。

（4）unset

删除指定属性。

（5）clear

删除所有属性，包括id属性。

（6）has

属性值为非null或非undefined时返回true。

（7）sync

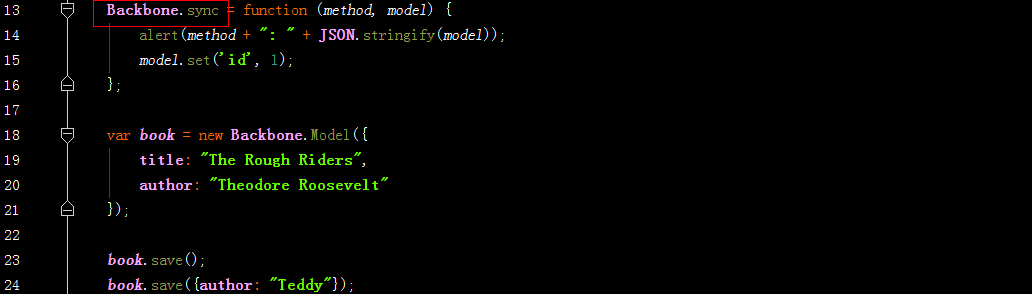
使用Backbone.sync将一个模型的状态发送到服务器。

（8）fetch

从服务器获取并重置模型的状态。返回jqXHR。

（9）save

保存模型到服务器。如下，在模型初次保存时发送create，第2次发送update请求。



（10）destroy

通过Backbone.sync发送DELETE请求，销毁服务器上的模型。

（11）url

返回模型资源在服务器上位置的相对URL。默认形式为/[collection.url]/[id]，如果模型不是集合的一部分，可以通过指定明确的urlRoot覆盖。

（12）urlRoot

对于集合外的模型，可用于生成基于模型id的默认url。

（13）toJSON

返回一个模型attributes浅拷贝副本的JSON字符串化形式，用于模型的持久化、序列化。

（14）isNew

检查模型是否已经保存到服务器。如果模型没有id，则被视为新的。

（16）hasChanged

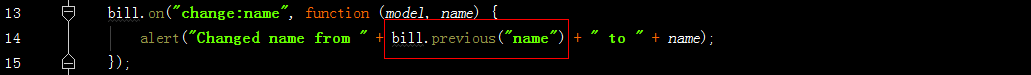
检查模型自上次set事件后是否改变过。

（17）changedAttributes

检查从最后一次set后改变的模型属性，如果没有，返回false。用于获取视图哪些部分应该更新，或者需要与服务器进行同步。

（18）previous

在change事件发生的过程中，获取已改变属性的旧值。



（19）previousAttributes

获取属性上一个值副本。

（20）isValid

运行validate来检查模型状态。

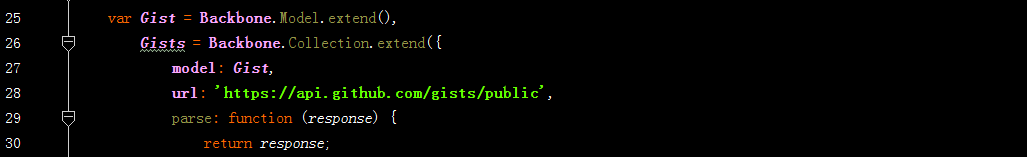
（21）clone

返回具有相同属性新的模型实例。

另外，还混入Underscore的6个方法：keys、values、pairs、invert、pick、omit。

* 1. Collection

集合是模型的有序组合。可以在集合上绑定如change事件，从而当集合中的模型发生变化时获得通知，监听集合中模型的变化。



**1.创建**

使用extend方法创建一个Collection类。初始化时，传入model属性的值作为集合中包含的模型类。如果定义了initialize函数，会在集合创建时被调用。

**2.属性/方法**

（1）models

访问集合中模型。

（2）length

返回该集合包含的模型数量。

（3）get

可通过id、cid，或model来获得集合中的模型。

（4）at

获得指定索引的模型。不论是否重新排序，at始终返回其在集合中插入时的索引值。

（5）add

向集合中增加一个模型（或一个模型数组）。

（6）remove

删除一个模型（或一个模型数组），并且返回他们。

（7）set

更新。如果模型不在集合中，将被添加；如果已在集合中，将被合并；如果集合包含不存在于列表中的任何模型，将被删除。

（8）reset

替换集合，触发reset事件。可以传入null清空集合。之前的模型列表可通过previousModels获取。

（9）push、pop、unshift、shift

（10）slice

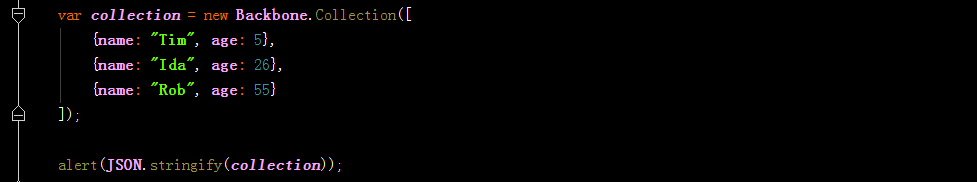
返回一个集合的模型的浅拷贝副本，使用与原生Array#slice相同选项。

（11）clone

返回一个模型列表完全相同的集合新实例。

（12）toJSON

返回所有模型的属性数组。



（13）sync

将一个集合状态持久化到服务器。

（14）url

以指定集合对应的服务器位置。

（15）parse

每次调用fetch从服务器拉取集合模型数据时，parse都会被调用。

（16）fetch

从服务器拉取集合的默认模型，获取从服务器返回的数据后会set集合。jQuery.ajax选项也可以直接传递作为fetch选项。需要传入{reset:true}，否则模型更新但集合不会更新。



（17）comparator

默认没有定义。如果定义了该函数，集合中的模型会按照指定的算法进行排序，插入到指定位置。

（18）sort

强制对集合进行重排序。一般情况下不需要调用本函数，因为可以使用comparator。

（19）pluck

获取集合中的每个模型的attribute。

（20）where

返回集合中所有匹配attributes的模型数组。



（21）findWhere

返回匹配attributes的第一个模型。

（22）create

在集合中创建1个模型的新实例。

* 1. Router

通过url的hash和History API，为应用位置提供可链接、收藏、分享的URLs。在应用已创建所有路由后，需要调用Backbone.history.start来初始化URL的路由。

（1）extend

创建一个自定义的路由类。

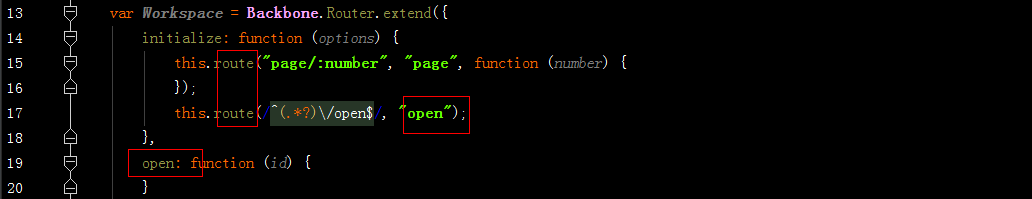
（2）routes

定义带参数的URLs映射到路由实例的方法。路由可以包含参数:param，并支持通配符，如\*splat；可将路由的可选部分放在括号中(/:optional)。路由结尾的斜杠会被当作URL的一部分，如docs和docs/将触发不同的回调。当匹配到一个路由时，会触发一个基于动作名称的event，其它对象可以监听这个路由并接收到通知。

（3）constructor/initialize

（4）route

为路由对象手动创建路由。



（5）navigate

更新的URL。设置trigger选项为true，可调用路由功能。设置replace选项为true，将不会在历史记录创建条目。



（6）execute

匹配任何路由都将执行。

* 1. history

作为全局路由服务，用于处理pushState，hashchange事件，匹配适合的路由，并触发回调函数。

Backbone会自动判断浏览器对pushState的支持。对于不支持pushState的浏览器，将继续使用基于锚点的URL片段。如果在兼容pushState的浏览器访问了某个URL锚点，将会被透明的转换为真实的URL。注意，使用真实的URLs需要web服务器支持直接渲染那些页面，因此后端程序也需要做修改。例如，如果浏览器直接访问路由“/document/100”，web服务器必须能够处理该页面。

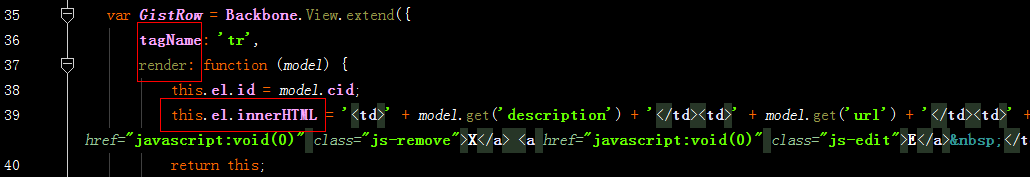
Routers创建后，需要调用Backbone.history.start监控hashchange事件并分配路由。如果需要在应用中使用HTML5支持的pushState，可以使用Backbone.history.start({pushState:true})。对于不支持pushState的浏览器，可以传入{hashChange:false}，则使用整页刷新代替。

如果应用不是基于域名的根路径/，可传入如root:"/public/search/"告诉History基于什么路径。如果服务器已经渲染了整个页面，但又不希望开始History时触发初始路由，可传入silent:true。

因IE中基于hash的历史记录依赖于<iframe>，因此需要确定DOM已准备就绪后再调用start。

* 1. View

使用view，当模型变化时，可以使页面各个部分独立地进行更新，而不必重绘整个页面；也不必通过查找DOM元素，手动更新HTML。

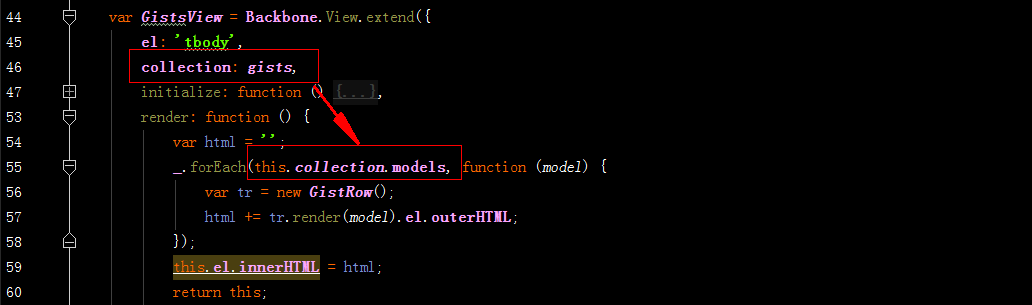


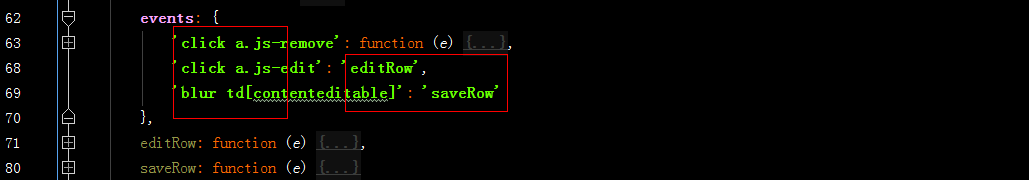
（1）extend

（2）constructor/initialize

（3）model、collection、el、id、className、tagName、attributes、events

如果在传入这些特殊选项，则将直接注册到视图中去。





（4）el

每个视图都拥有1个对应DOM元素，即使该元素尚未插入页面中。

（5）$el

el对应的jQuery对象。

（6）setElement

创建对另1个$el的引用。视图的委托事件也将从旧元素移动到新元素上。

（7）attributes

为视图el元素的属性。

（8）$(selector)

如果页面引入了jQuery，则每个视图都将拥有$函数，可以在视图元素查询作用域内运行。



（9）template

定义template模板。

（10）render

可以实现从模型数据渲染视图模板，并可用新的HTML更新this.el（并不是返回html，而是在里面渲染html）。

（11）remove

从DOM中移除一个视图。

（12）delegateEvents

采用jQuery的on函数来为视图内的DOM事件提供回调函数声明。

（13）undelegateEvents

删除视图所有委托事件。

* 1. sync

（1）sync

Backbone每次向服务器读取或保存模型时，都将调用执行sync函数。默认使用jQuery.ajax方法发送RESTful json请求，并且返回1个jqXHR。可以重载该函数以实现其他方法。

当Backbone.sync发送请求以保存模型时，模型属性会被序列化为JSON，并以application/json的内容类型发送。当接收到来自服务器的JSON响应后，客户端会更新模型。

（2）ajax

使用自定义的AJAX方法。

（3）emulateHTTP

使用X-HTTP-Method-Override头伪造PUT、PATCH和DELETE请求。

（4）emulateJSON

对不支持application/json编码请求的Web服务器进行处理。

1. 小程序

小程序主体部分的3个文件必须放在根目录下。

**1.app.js**

文件内调用App函数，用来注册一个小程序。接受一个object参数。



**（1）object参数****说明**



前台、后台定义：离开微信，小程序并没有直接销毁，而是进入了后台；当再次进入微信或再次打开小程序，又会从后台进入前台。

**（2）onLaunch、onShow参数说明**



**（3）getApp**

全局getApp()函数可以获取到小程序实例。



**2.app.json**

全局配置，决定页面文件的路径、窗口表现、设置网络超时时间、设置多tab等。



**（1）pages**

接受一个数组，每一项都是字符串，来指定小程序由哪些页面组成。数组的第一项代表小程序的初始页面。文件名不需要写文件后缀，因为框架会自动去寻找路径.json,.js,.wxml,.wxss的四个文件进行整合。



**（2）window**

用于设置小程序的状态栏、导航条、标题、窗口背景色。



**（3）tabBar**

如果一个多tab应用，那么可以通过tabBar配置项指定tab栏的表现，及切换时显示的对应页面。

**（4）networkTimeout**

可以设置各种网络请求的超时时间。

**（5）debug**

开启debug模式时，在开发者工具的控制台面板，调试信息以info的形式给出。

**（6）page.json**

每个页面也可以使用.json文件对本页面的窗口表现进行配置。只是设置app.json中的window配置项的内容，页面中配置项会覆盖app.json的window中相同的配置项。

**3.WXSS**

WXSS用来决定WXML的组件应该怎么显示。定义在app.wxss中的样式为全局样式，作用于每一个页面。在page的wxss文件中定义的样式为局部样式，只作用在对应的页面，并会覆盖app.wxss中相同的选择器。

（1）尺寸单位

rpx：规定屏幕宽为750rpx。

（2）样式导入

使用@import语句可以导入外联样式表。

（3）内联样式

支持使用style、class属性来控制组件的样式。

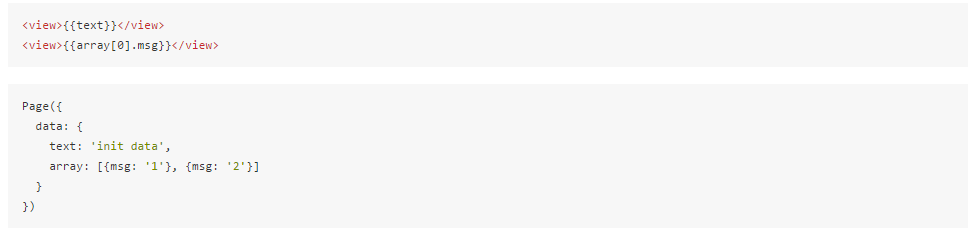
**4.页面**

在app.json的pages会定义页面路径。页面路径是1个文件夹，包含4类文件：.json,.js,.wxml,.wxss。微信会自动加载它们。其中，.js中为页面逻辑，里面会使用Page()函数注册一个页面。

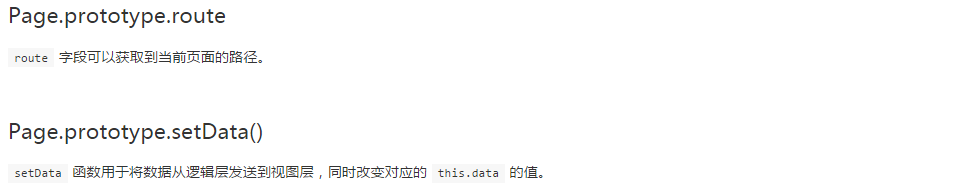
（1）Page

Page接受1个object参数，可以在其中定义初始数据、生命周期函数、动作处理函数等。

* 初始化数据



* 原型方法



（2）页面路由

由微信自己管理。路由的触发方式以及页面生命周期函数如下：



getCurrentPages()函数用于获取当前页面栈的实例，以数组形式按栈的顺序给出，第一个元素为首页，最后一个元素为当前页面。

**5.模块化**

在js文件中声明的变量和函数只在该文件中有效。通过全局函数 getApp() 可以获取全局的应用实例，如果需要全局的数据可以在 App() 中设置。



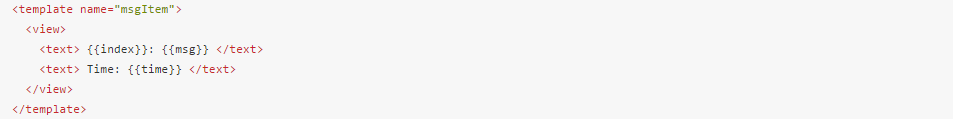


可以将一些公共的代码抽离成为一个单独的 js 文件，作为一个模块。模块只有通过 module.exports 或者 exports 才能对外暴露接口。​在需要使用这些模块的文件中，使用 require(path) 将公共代码引入。require 暂时不支持绝对路径。

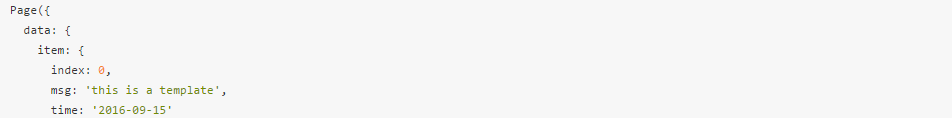
**6.视图**

（1）模板

在模板中定义代码片段，然后在不同的地方调用。使用name属性，作为模板的名字。然后在<template/>内定义代码片段。







（2）组件

组件自带一些功能与微信风格的样式。一个组件通常包括开始标签和结束标签，属性用来修饰这个组件，内容在两个标签之内。所有组件与属性都是小写，以连字符-连接

视图容器

view：视图容器。scroll-view：可滚动视图区域。swiper：滑块视图容器。

基础内容

icon：图标。text：文字。progress：进度条。

表单

导航

多媒体

地图

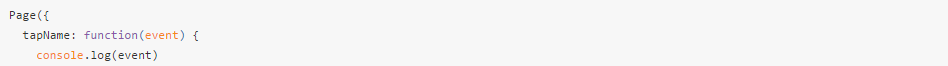
画布

客服会话

（3）事件

在组件中绑定一个事件处理函数。在相应的Page定义中写上相应的事件处理函数，参数是event。

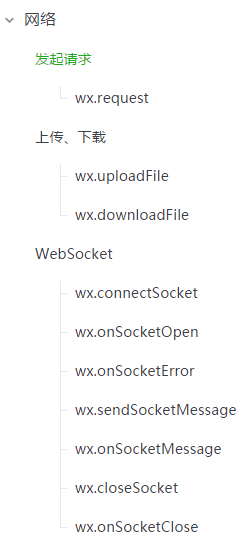
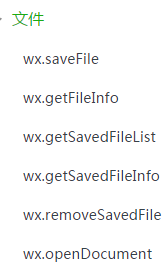
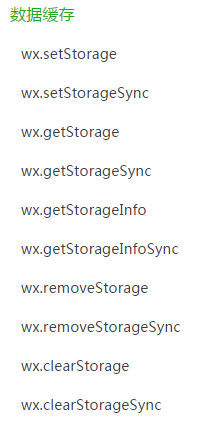


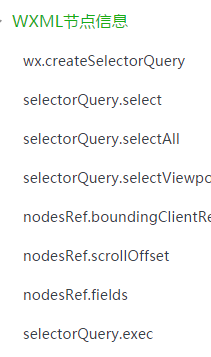


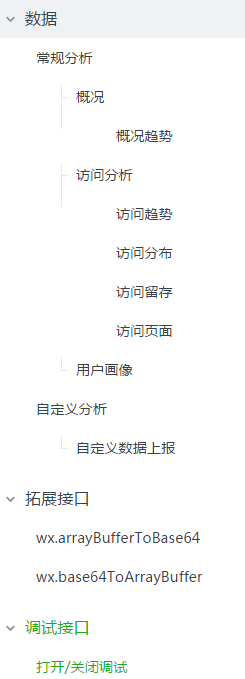
（4）引用

WXML 提供两种文件引用方式import和include。import 有作用域的概念，即只会 import 目标文件中定义的 template，而不会 import 目标文件 import 的 template。include可以将目标文件除了<template/>的整个代码引入，相当于是拷贝到include位置。

**7.API**

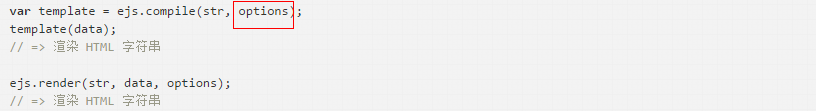
 

1. EJS

**1.安装**



**2.用法**



其中，compile返回解析好的Function函数，接收数据之后即可渲染html字符串。render直接返回经过解析的字符串。

options参数可以为：

（1）cache：是否缓存解析后的模版，需要filename作为key。

（2）filename：被cache用做缓存的键，模版文件名。

（3）context：complile后的Function执行所在的上下文环境。

（4）compileDebug：为true则会生成解析过程中的跟踪信息，用于调试。

（5）client：标识是否用于浏览器客户端运行，为true则返回解析后的可以单独运行的Function函数。

（6）delimiter：开启或者闭合尖括号所用标记。

（7）\_with：是否使用with(){}结构。如果为false则局部数据会储存在locals对象中。

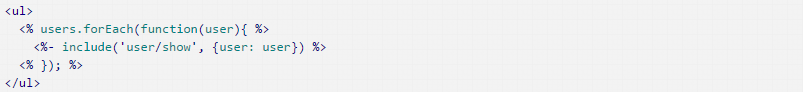
（8）rmWhitespace：移除所有可以安全移除的空白字符，包含前导和尾后的空白字符。同时会为所有scriptlet标签开启-%>换行截断的更加安全的模式。

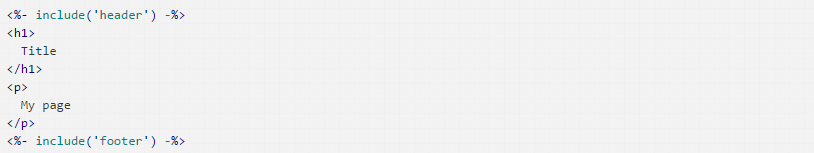
**3.标签**

* <%：用于控制流，没有输出。
* <%=：向模板输出值（带有转义）。
* <%-：向模板输出没有转义的值。
* <%#：注释标签，不执行，也没有输出。
* <%%：输出字面的'<%'。
* %>：普通的结束标签。
* -%>：Trim-mode标签，移除随后的换行符。

**4.包含**

如果不使用绝对路径，则为相对于调用include的模板的路径。如在./views/users.ejs中包含./views/user/show.ejs，应该使用。





**5.辅助函数**

（1）first：返回数组的第一个元素。

（2）last：返回数组的最后一个元素。

（3）capitalize：返回首字母大写的字符串。

（4）downcase：返回字符串的小写。

（5）upcase：返回字符串的大写。

（6）sort：排序。

（7）sort\_by:'prop'：按照指定的prop属性进行升序排序。

（8）size：返回长度，即length属性，不一定非是数组才行。

（9）plus:n：加上n，将转化为Number进行运算。

（10）minus:n：减去n，将转化为Number进行运算。

（11）times:n：乘以n，将转化为Number进行运算。

（12）divided\_by:n：除以n，将转化为Number进行运算。

（13）join:'val'：将数组用val作为分隔符，进行合并成一个字符串。

（14）truncate:n：截取前n个字符，超过长度时，将返回一个副本

（15）truncate\_words:n：取得字符串中的前n个word，word以空格进行分割。

（16）replace:pattern,substitution：字符串替换，substitution不提供将删除匹配的子串。

（17）prepend:val：如果操作数为数组，则进行合并，为字符串则添加val在前面。

（18）append:val：如果操作数为数组，则进行合并，为字符串则添加val在后面。

（19）map:'prop'：返回对象数组中属性为prop的值组成的数组。

（20）reverse：翻转数组或字符串。

（21）get:'prop'：取得属性为prop的值。

（22）json：转化为json格式字符串。