**[超完整的Chrome浏览器客户端调试大全](http://blog.csdn.net/xueer767/article/details/65936204)**

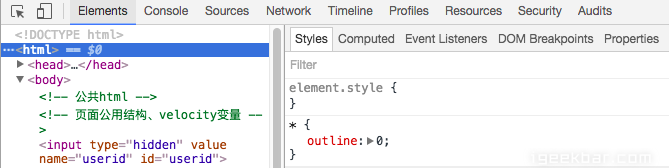
**引言**

“工欲善其事，必先利其器”

没错，这句话个人觉得说的特别有道理，举个例子来说吧，厉害的化妆师都有一套非常专业的刷子，散粉刷负责定妆，眼影刷负责打眼影，各司其职，有了专业的工具才能干专业的事，这个灵感要来源于之前我想买化妆品时，店里的海报标语，由此联想到，如果你想在某个事情上做好，并且做的专业，那么你一定要把你的工具用好，这样才能事半功倍，我见过很多师兄师姐，写了很多代码，能够很快的完成工作，能够处理很多复杂的业务逻辑，但是对于浏览器的调试掌握的并不全面和深入，说说本姑娘吧，我的编程调试起源于自学滴前端课程，因为学习的时候看的都是基础的教学视频，对于调试也只是掌握了alert和console, 请大家别笑话，认真看完再说话，试问谁当初入门时候不是走的这条路呢，当你不再限于做静态页面，古老而经典的调试方式肯定不能帮你完成日常调试，日后我进入到了大公司去实习，才真正开始接触专业开发业务，开始跟着师兄和师傅一起上路，那时我才有了“js断点调试“的意识，开始一步步调试我的js代码~

下面总结一下一些常用调试方法，这些方法能让开发的工作顺利并且高效，这里小女子拿出来总结一下，与各位程序猿同仁分享一下 ~ （此处应有掌声…… ^\_^）

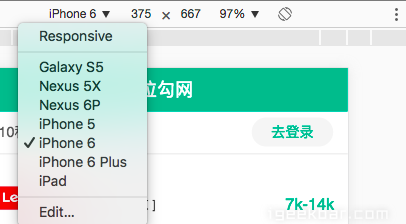
**一. 先来认识一下这些按钮**

[](http://igeekbar.com/igeekbar/networks/uploadimg/52af2165-aa3b-41ea-acdf-4c3fb7ba3b27.png)

先来看这张图最上头的一行是一个功能菜单，每一个菜单都有它相应的功能和使用方法，依次从左往右来看

1.箭头按钮：用于在页面选择一个元素来审查和查看它的相关信息，当我们在Elements这个按钮页面下点击某个Dom元素时，箭头按钮会变成选择状态

2.设备图标：点击它可以切换到不同的终端进行开发模式，移动端和pc端的一个切换，可以选择不同的移动终端设备，同时可以选择不同的尺寸比例，chrome浏览器的模拟移动设备和真实的设备相差不大，是非常好的选择

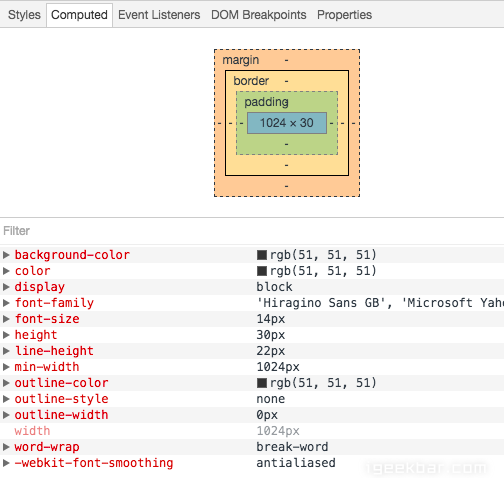
[](http://igeekbar.com/igeekbar/networks/uploadimg/51d40eb1-3e74-4e27-9583-b3b50f332510.png)

可选择的适配

3.Elements 功能标签页：用来查看，修改页面上的元素，包括DOM标签，以及css样式的查看，修改，还有相关盒模型的图形信息，下图我们可以看到当我鼠标选择id 为lg\_tar的div元素时，右侧的css样式对应的会展示出此id 的样式信息，此时可以在右侧进行一个修改，修改即可在页面上生效， 灰色的element.style样式同样可以进行添加和书写，唯一的区别是，在这里添加的样式是添加到了该元素内部，实现方式即：该div元素的style属性，这个页面的功能很强大，在我们做了相关的页面后，修改样式是一块很重要的工作，细微的差距都需要调整，但是不可能说做到每修改一点即编译一遍代码，再刷新浏览器查看效果，这样很低效，一次性在浏览器中修改之后，再到代码中进行修改

[](http://igeekbar.com/igeekbar/networks/uploadimg/b548b239-eb00-447d-be60-29c1c3b3ce9b.png)

* 对应的样式

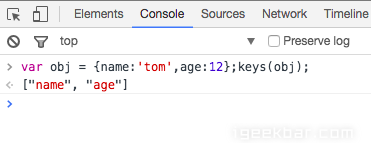
[](http://igeekbar.com/igeekbar/networks/uploadimg/54802412-70df-4c69-8adb-cea9c52b2b09.png)

* 盒模型信息

同时，当我们浏览网站看到某些特别炫酷的效果和难做的样式时候，打开这个功能，我们即可看到别人是如何实现的，学会它这知识就是你的了，仔细钻研也会有意想不到的收获

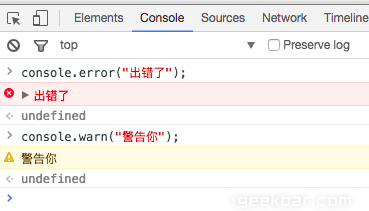
4.Console控制台：用于打印和输出相关的命令信息，其实console控制台除了我们熟知的报错，打印console.log信息外，还有很多相关的功能，下面简单介绍几个：

a: 一些对页面数据的指令操作，比如打断点正好执行到获取的数据上，由于数据都是层层嵌套的对象，这个时候查看里面的key/value不是很方便，即可用这个指令开查看，obj的json string 格式的key/value，我们对于数据里面有哪些字段和属性即可一目了然

[](http://igeekbar.com/igeekbar/networks/uploadimg/ac4fc806-f29a-44e2-bcf0-95b28c401790.png)

其他功能

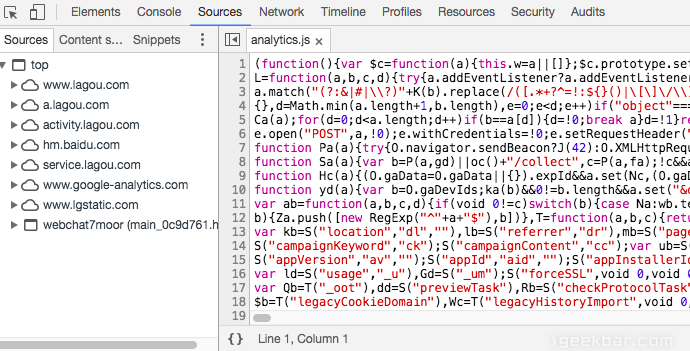
b: 除了console.log还有其他相关的指令可用

[](http://igeekbar.com/igeekbar/networks/uploadimg/6e7f7867-9422-4238-a23b-d463e1f799d5.png)

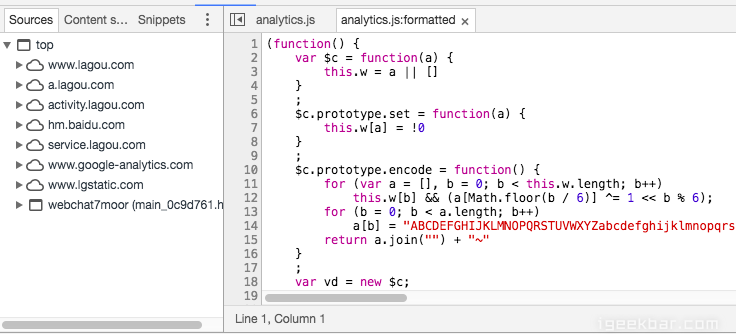
console也有相关的API

5.Sources js资源页面：这个页面内我们可以找到当然浏览器页面中的js 源文件，方便我们查看和调试，在我还没有走出校园时候，我经常看一些大站的js代码，那时候其实基本都看不懂，但是最起码可以看看人家的代码风格，人家的命名方式，所有的代码都是压缩之后的代码，我们可以点击下面的{}大括号按钮将代码转成可读格式

Sources Panel 的左侧分别是 Sources 和 Content scripts和Snippets

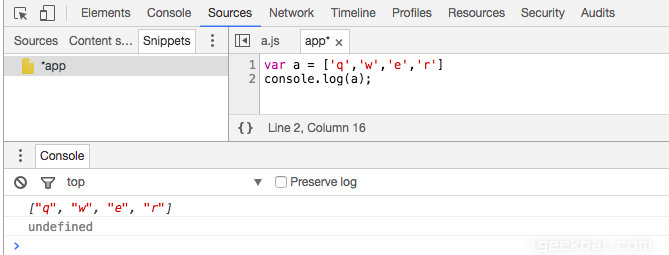
[](http://igeekbar.com/igeekbar/networks/uploadimg/a147d491-68bd-45d7-8403-6c25ce99201e.png)

* 对应的源代码

[](http://igeekbar.com/igeekbar/networks/uploadimg/a8e61566-e44e-4a92-95e0-c872cf9a2cbb.png)

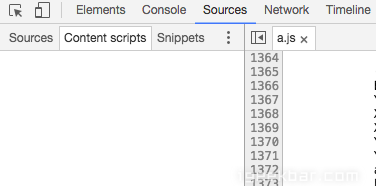
* 格式化后的代码

关于打断点调试的内容，下面介绍，先来说一些，其他平时基本没人用但是很有用的小点，比如当我们想不起某个方法的具体使用时候，会打开控制台随意写一些**[测试](http://lib.csdn.net/base/softwaretest" \o "软件测试知识库" \t "_blank)**代码，或者想测试一下刚刚写的方法是否会出现期待的样子，但是控制台一打回车本想换行但是却执行刚写的半截代码，所以推荐使用Sources下面的左侧的Sinppets代码片段按钮，这时候点击创建一个新的片段文件，写完测试代码后把鼠标放在新建文件上run，再结合控制台查看相关信息（新建了一个名叫：app.js的片段代码，在你的项目环境页面内，该片段可执行项目内的方法）

[](http://igeekbar.com/igeekbar/networks/uploadimg/f56570d0-2ec4-4970-8ba5-c1bf2b8abf2d.png)

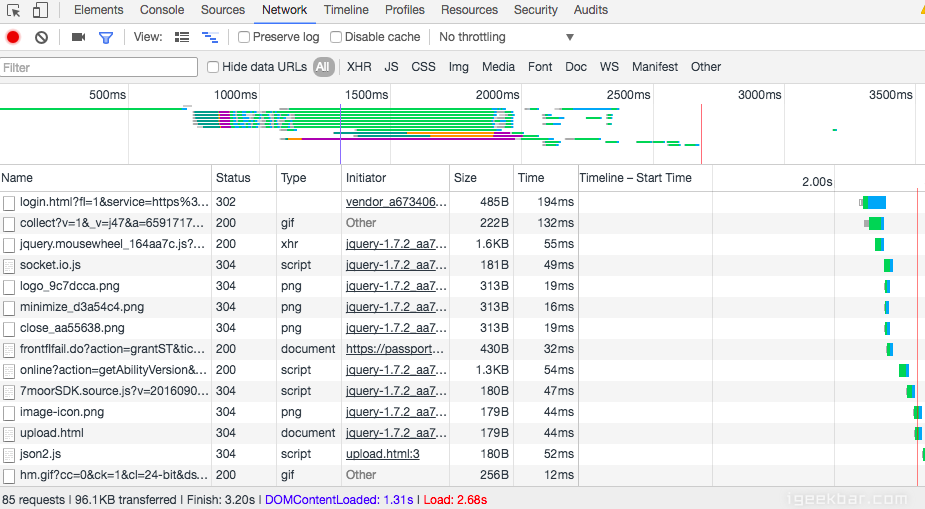
* 自己书写的片段

Content scripts 是 Chrome 的一种扩展程序，它是按照扩展的ID来组织的，这些文件也是嵌入在页面中的资源，这类文件可以读写和操作我们的资源，需要调试这些扩展文件，则可以在这个目录下打开相关文件调试，但是几乎我们的项目还没有相关的扩展文件，所以啥也看不到，平时也不需要关心这块

[](http://igeekbar.com/igeekbar/networks/uploadimg/e93f193c-3bf2-41ef-b300-e199a8a60d27.png)

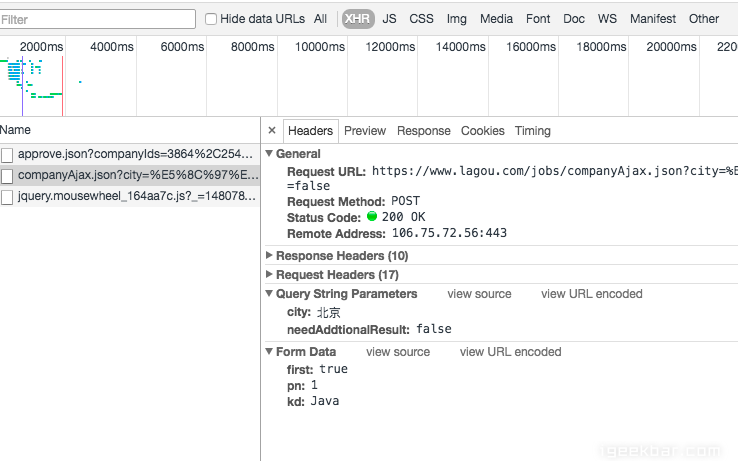
无结果

6.Network 网络请求标签页：可以看到所有的资源请求，包括网络请求，图片资源，html,css，js文件等请求，可以根据需求筛选请求项，一般多用于网络请求的查看和分析，分析后端接口是否正确传输，获取的数据是否准确，请求头，请求参数的查看

[](http://igeekbar.com/igeekbar/networks/uploadimg/83a9003a-434c-4f11-a7d6-6ce2f5965106.png)

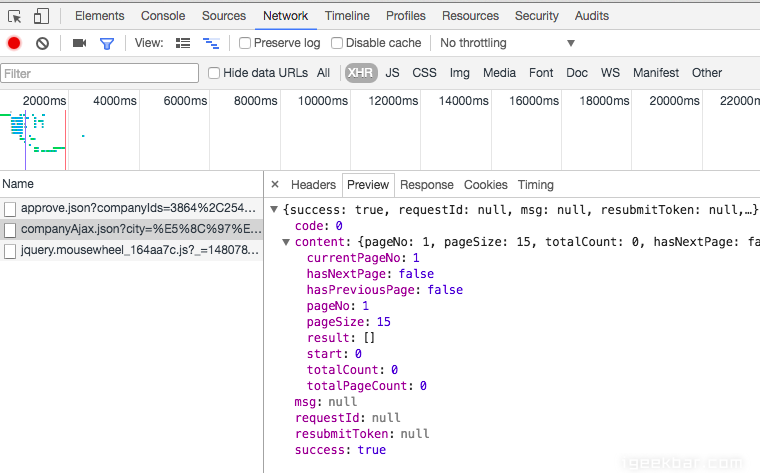
* 所有的资源

以上我选择了All，就会把该页面所有资源文件请求下来，如果只选择XHR 异步请求资源，则我们可以分析相关的请求信息

[](http://igeekbar.com/igeekbar/networks/uploadimg/a36f5b9e-2e06-4593-9c77-a50d798bc8ea.png)

* 请求的相关信息

打开一个Ajax异步请求，可以看到它的请求头信息，是一个POST请求，参数有哪些，还可以预览它的返回的结果数据，这些数据的使用和查看有利于我们很好的和后端工程师们联调数据，也方便我们前端更直观的分析数据

[](http://igeekbar.com/igeekbar/networks/uploadimg/90ebfadd-e79a-4837-9873-5cbb0cd1b0f2.png)

* 预览请求的数据

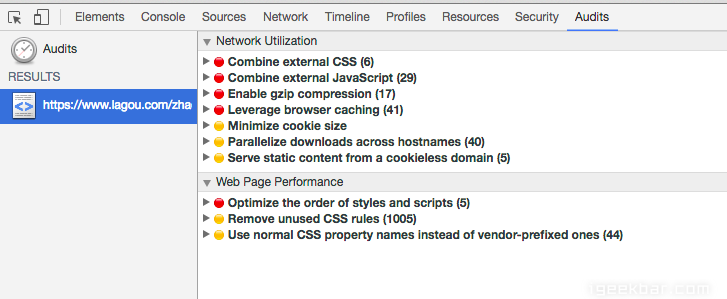
7.Timeline标签页可以显示JS执行时间、页面元素渲染时间，不做过多介绍

8.Profiles标签页可以查看CPU执行时间与内存占用，不做过多介绍

9.Resources标签页会列出所有的资源，以及**[HTML5](http://lib.csdn.net/base/html5" \o "HTML5知识库" \t "_blank)**的Database和LocalStore等，你可以对存储的内容编辑和删除 不做过多介绍

10.Security标签页 可以告诉你这个网站的安全性，查看有效的证书等

11.Audits标签页 可以帮你分析页面性能，有助于优化前端页面，分析后得到的报告

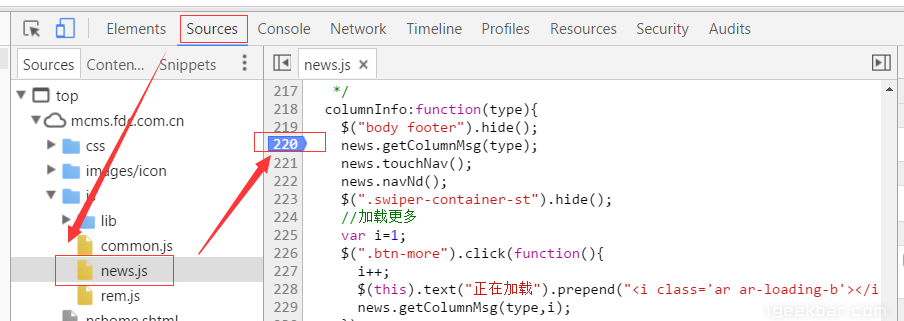
[](http://igeekbar.com/igeekbar/networks/uploadimg/2e0393c9-cc86-4a20-905e-3f80154f4f2f.png)

* 分析结果

**二.Sources资源页面的断点调试**

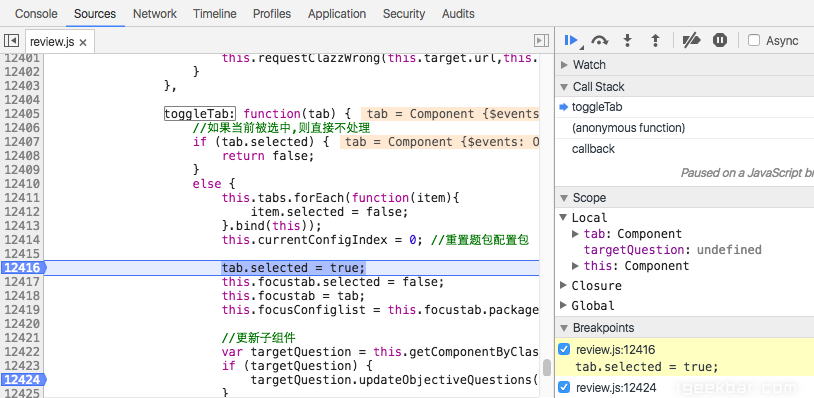
1.如何调试：

调试js代码，肯定是我们常用的功能，那么如何打断点，找到要调试的文件，然后在内容源代码左侧的代码标记行处点击即可打上一个断点

[](http://igeekbar.com/igeekbar/networks/uploadimg/0d2066b5-e10e-4562-b5f7-eef9ff2e9a02.png)

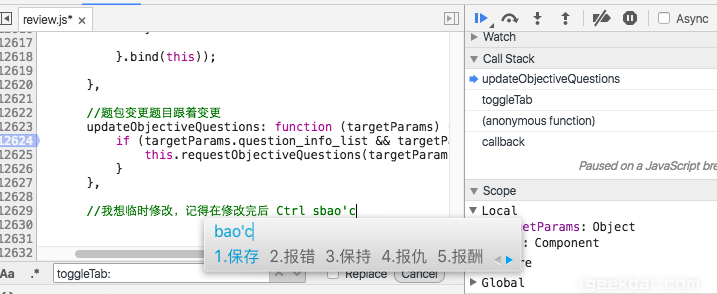
2.断点与 js代码修改

看下面这张图，我在一个名为toggleTab的方法下打了两个断点，当开始执行我们的点击切换tab行为后，代码会在执行的断点出停下来，并把相关的数据展示一部分，此时可以在已经执行过得代码处，把鼠标放上去，即可查看相关的具体数据信息，同时我们可以使用右侧的功能键进行调试，右侧最上面一排分别是：暂停/继续、单步执行(F10快捷键)、单步跳入此执行块(F11快捷键)、单步跳出此执行块、禁用/启用所有断点。下面是各种具体的功能区

[](http://igeekbar.com/igeekbar/networks/uploadimg/980c6de1-f378-48c3-90a5-6d865b46881f.png)

* 在代码中打断点

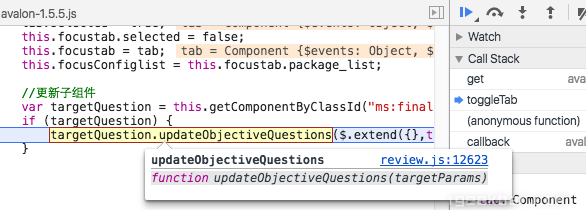
在当前的代码执行区域，在调试中如果发现需要修改的地方，也是可以立即修改的，修改后保存即可生效，这样就免去了再到代码中去书写，再刷新回看了

[](http://igeekbar.com/igeekbar/networks/uploadimg/cce6a3d6-e055-43ea-9687-e3f1fa6854cf.png)

临时修改

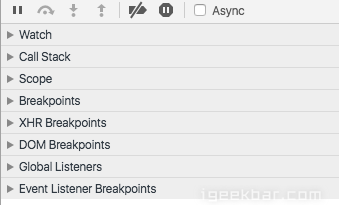
3.快速进入调试的方法

当我们的代码执行到某个程序块方法处，这个方法上可能你并没有设置相关的断点，此时你可以F11进入此程序块，但是往往我们的项目都是经过很多源代码封装好的方法，有时候进入后，会走很多底层的封装方法，需要很多步骤才能真正进入这个函数块，此时将鼠标放在此函数上，会出现相关提示，会告诉你在该文件的哪一行代码处，点击即可直接看到这个函数，然后临时打上断点，按F10或者点击右上角的第二个按钮即可直接进入此函数的断点处

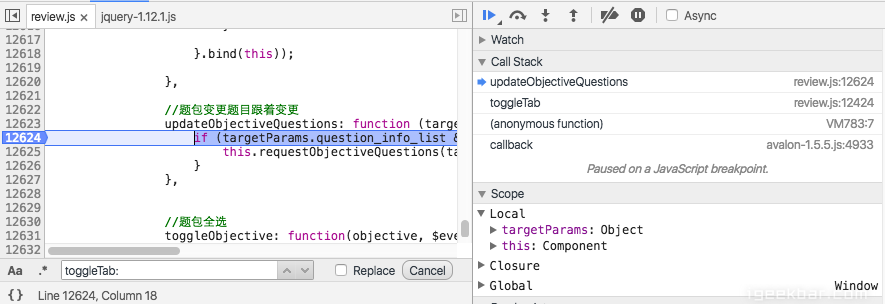
[](http://igeekbar.com/igeekbar/networks/uploadimg/5130e68e-74c7-4b1a-9a12-7f5e7f5ceb6d.png)

4.调试的功能区域

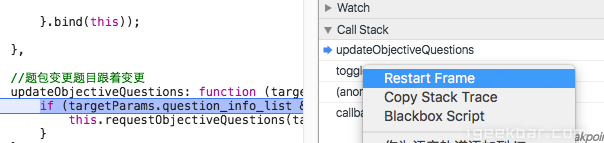
每一个功能区，都有它相关的左右，先来看一张图，它都有哪些功能

[](http://igeekbar.com/igeekbar/networks/uploadimg/bdb50caf-0484-47b9-a3ed-ec0aa9d38e67.png)

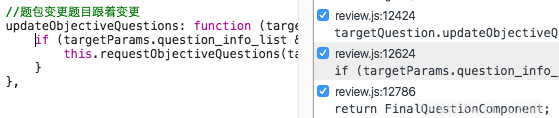
Call Stack调用栈：当断点执行到某一程序块处停下来后，右侧调试区的 Call Stack 会显示当前断点所处的方法调用栈，从上到下由最新调用处依次往下排列，Call Stack 列表的下方是Scope Variables列表可以查看此时局部变量和全局变量的值。图中可以看出，我们最先走了toggleTab这个方法，然后走到了一个更新对象的方法上，当前调用在哪里，箭头会帮你指向哪里，同时我们可以点击，调用栈列表上的任意一处，即可回头再去看看代码

[](http://www.igeekbar.com/igeekbar/networks/uploadimg/8dd4193f-f723-4597-9633-1c789fbf474b.png)

但是若你想从新从某个调用方法出执行，可以右键Restart Frame， 断点就会跳到此处开头重新执行，Scope 中的变量值也会依据代码从新更改，这样就可以回退来从新调试，错过的调试也可以回过头来反复查看

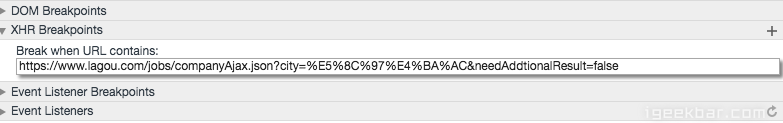
[](http://igeekbar.com/igeekbar/networks/uploadimg/c7ba0acc-7305-4a55-9790-6bc637922989.png)

Breakpoints关于断点：所有当前js的断点都会展示在这个区域，你可以点击按钮用来“去掉/加上”此处断点，也可以点击下方的代码表达式，调到相应的程序代码处，来查看

[](http://igeekbar.com/igeekbar/networks/uploadimg/1cd4281f-4170-40e4-8bfa-cc0050af9a53.png)

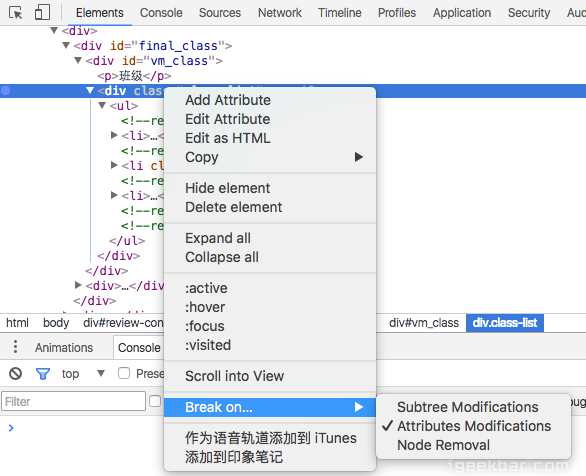
XHR Breakpoints

在XHR Breakpoints处，点击右侧的+号，可以添加请求的URL，一旦 XHR 调用触发时就会在 request.send() 的地方中断

[](http://igeekbar.com/igeekbar/networks/uploadimg/238a0267-dbd8-4dd3-8598-b09895b5694e.png)

DOM Breakpoints:

可以给你的DOM元素设置断点，有时候真的需要监听和查看某个元素的变化情况，赋值情况，但是我们并是不太关心哪一段代码对它做的修改，只想看看它的变化情况，那么可以给它来个监听事件，这个时候DOM Breakpoints中会如图

[](http://igeekbar.com/igeekbar/networks/uploadimg/da1c5b48-42db-4d5f-8a24-19389fa858f0.png)

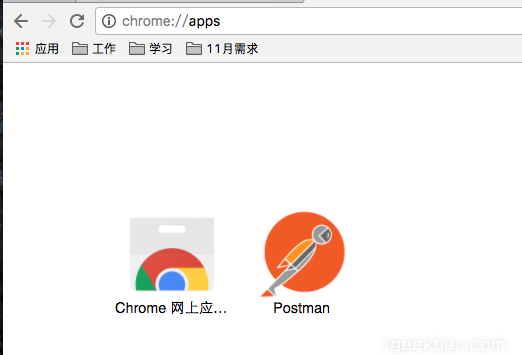
当要给DOM添加断点的时候，会出现选择项分别是如下三种修改1.子节点修改2.自身属性修改3.自身节点被删除。选中之后，Sources Panel 中右侧的 DOM Breakpoints 列表中就会出现该 DOM 断点。一旦执行到要对该 DOM 做相应修改时，代码就会在那里停下来

Event listener Breakpoints

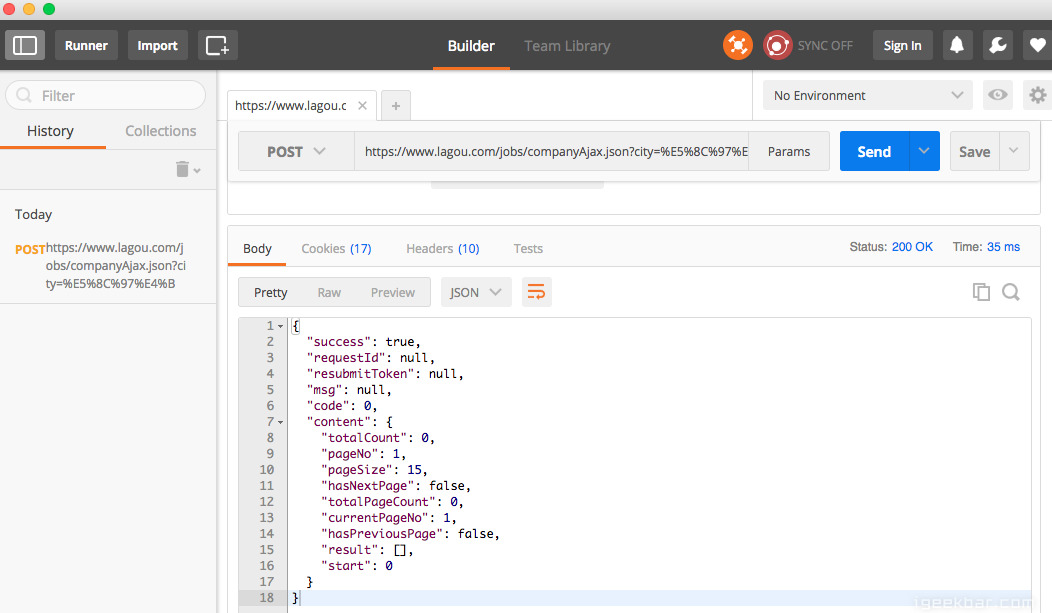
最后Event Listener 列表，这里列出了各种可能的事件类型。勾选对应的事件类型，当触发了该类型的事件的 [**JavaScript**](http://lib.csdn.net/base/javascript) 代码时就会自动中断

**三.Post man你值得拥有的网络请求神器**

在我们的开发过程中，后端的接口都是由发起AJAX请求而获取到的相关数据，但是很多情况是我们的业务还没有做到那块时，后端的同学接口都已经准备好了，但是为了便于后期的工作，将接口请求的数据模拟访问，然后对接口联调很重要，也很方便，因为我们不可能把每个请求代码都写到文件里编译好了再去浏览器内查看，这时候可以安装一个post man网络请求插件，在谷歌应用商店下载，需要翻墙

[](http://igeekbar.com/igeekbar/networks/uploadimg/eea9dc12-f18e-4ccc-9998-b34f4f367ce1.png)

该扩展程序使用非常简单，功能同时也非常强大，输入你的请求，选择好请求的method，需要请求参数的挨个填好，send之后，就可以看到返回的数据，这个小工具很利于我们的开发

[](http://igeekbar.com/igeekbar/networks/uploadimg/48b702eb-7a24-4fa9-aa52-bcc8b97c8fbd.png)

**完结**

写到这里这篇总结就结束了，也许有很多写的不到位的地方，也有一些专业用词不严谨的地方，希望看到的读者可以和我一起交流，我也非常乐意我的总结可以给 刚刚学会编程需要调试的同学受用，再此之前我一直在寻找一篇从头到尾的调试教学文章，我一直没有找到，要么是一点点的片段讲解，要么是专讲js断点调试，所以索性后来就直接看了 Chrome Developer Tools 的英文官方文档，并结合自己的一些小使用心得总结出此文，希望受到指点和修正 也希望可以帮助一些同学~

Cayley 一个不断努力学习的文艺女程序员

**原文版权说明：**

版权说明：

本文配图/文字版权属于[igeekbar.com](http://www.igeekbar.com/" \t "_blank)及作者联合所有，如需转载请务必注明iG客吧原文来源，感谢阅读支持~

原文链接：https://note.wiz.cn/pages/manage/biz/payReadNote.html?kb=3fe9d146-6498-4882-b75c-f533442aba5b&dc=857ef34b-af65-423c-b2e2-2aedf734825f