[神器——Chrome开发者工具(一)](https://segmentfault.com/a/1190000000683599)

* [chrome](https://segmentfault.com/t/chrome/blogs)

* [chrome开发者工具](https://segmentfault.com/t/chrome%E5%BC%80%E5%8F%91%E8%80%85%E5%B7%A5%E5%85%B7/blogs)

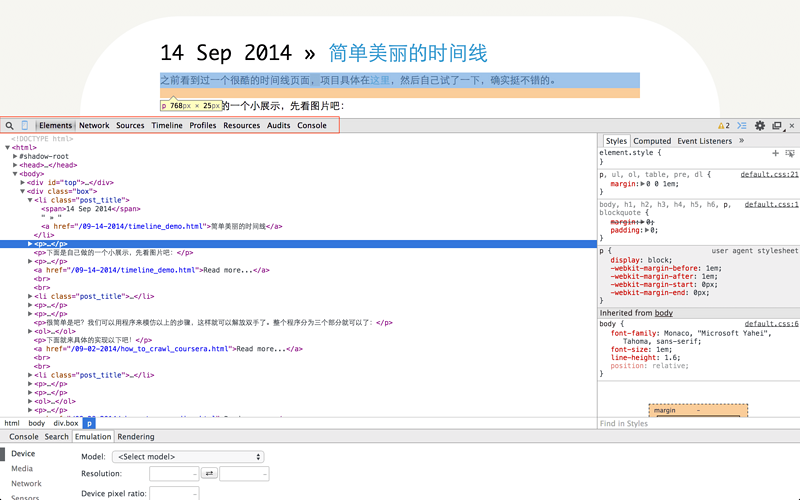
[**selfboot**](https://segmentfault.com/u/selfboot) 2014年09月19日发布

* 推荐 **13** 推荐
* 收藏 **171** 收藏，43.9k 浏览

[原文地址](http://xuelangzf.github.io/2014/09/19/chrome_DevTools_1/)

这里我假设你用的是Chrome浏览器，如果恰好你做web开发，或者是比较好奇网页中的一些渲染效果并且喜欢折腾，那么你一定知道Chrome的开发者工具了。其实其他浏览器也有类似工具，比如Firefox下的firebug。本文将会详细讲述如何使用Chrome开发者工具，希望里面有些让你感到惊艳的东西！即使你不用Chrome，那么文中的某些内容也会相当有用。

首先啰嗦一下如何打开开发者工具吧。可以直接在页面上点击右键，然后选择审查元素；或者是打开Tools--Developer Tools；或者是用快捷键 Command Option + I 打开。打开后我们看到的界面大概如下：



<!-- more -->

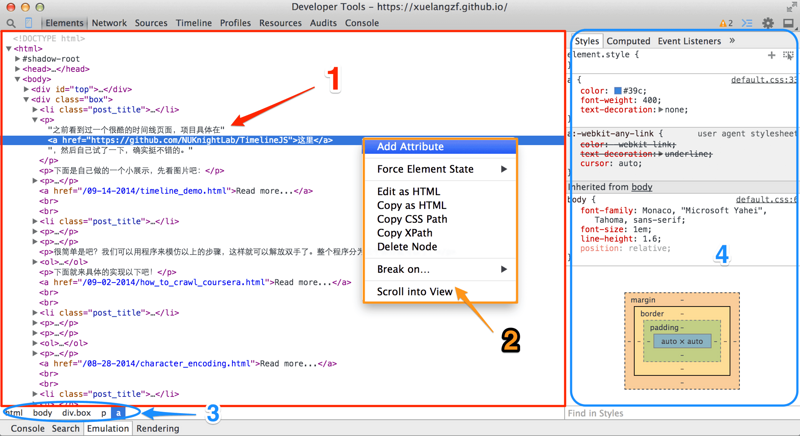
Google Chrome一共提供了8大组工具：

1. Elements: 允许我们从浏览器的角度看页面，也就是说我们可以看到chrome渲染页面所需要的的HTML、CSS和DOM(Document Object Model)对象。此外，还可以编辑这些内容更改页面显示效果；
2. Network: 可以看到页面向服务器请求了哪些资源、资源的大小以及加载资源花费的时间，当然也能看到哪些资源不能成功加载。此外，还可以查看HTTP的请求头，返回内容等；
3. Sources: 主要用来调试js；
4. Timeline: 提供了加载页面时花费时间的完整分析，所有事件，从下载资源到处理Javascript，计算CSS样式等花费的时间都展示在Timeline中；
5. Profiles: 分析web应用或者页面的执行时间以及内存使用情况；
6. Resources: 对本地缓存（IndexedDB、Web SQL、Cookie、应用程序缓存、Web Storage）中的数据进行确认及编辑；
7. Audits: 分析页面加载的过程，进而提供减少页面加载时间、提升响应速度的方案；
8. Console: 显示各种警告与错误信息，并且提供了shell用来和文档、开发者工具交互。

强大的Chrome开发者工具提供了很棒的提示功能，当我们把鼠标悬停在某些项时，会显示一些很有用的提示信息，有时候我们可以得到意想不到的收获。此外，开发者工具还提供了Emulation功能，做移动开发时特别有用。下面先详细研究一下8大工具的使用方法。

[Elements](https://developer.chrome.com/devtools/docs/dom-and-styles)

Elements工具像一把手术刀一样“解剖”了当前页面，我们看到的Elements页面一般像这样子：



图中标记为1的红色区块为页面HTML文件，HTML中的每个元素比如<body>、<p>都是一个DOM节点，所有的DOM节点组成了DOM树。我们完全可以把红色区块1当做是DOM树，把HTML元素标签看做DOM节点。

当我们在这里选中某一DOM对象时，网页中相应元素也会被阴影覆盖。我们可以对DOM对象进行修改，修改后结果会在页面实时显示出来。此外，还可以用 Command + f 搜索DOM树中指定的内容，或者是以HTML形式更改页面元素，如下图：



选中DOM对象之后右键即可以看到一些辅助的功能，如图中标记为2的区块所示：

1. Add Attribute: 在标签中增加新的属性；
2. Force Element State: 有时候我们为页面元素添加一些动态的样式，比如当鼠标悬停在元素上时的样式，这种动态样式很难调试。我们可以使用Force Element State强制元素状态，便于调试，如下图：

![图4. 强制元素状态][4]

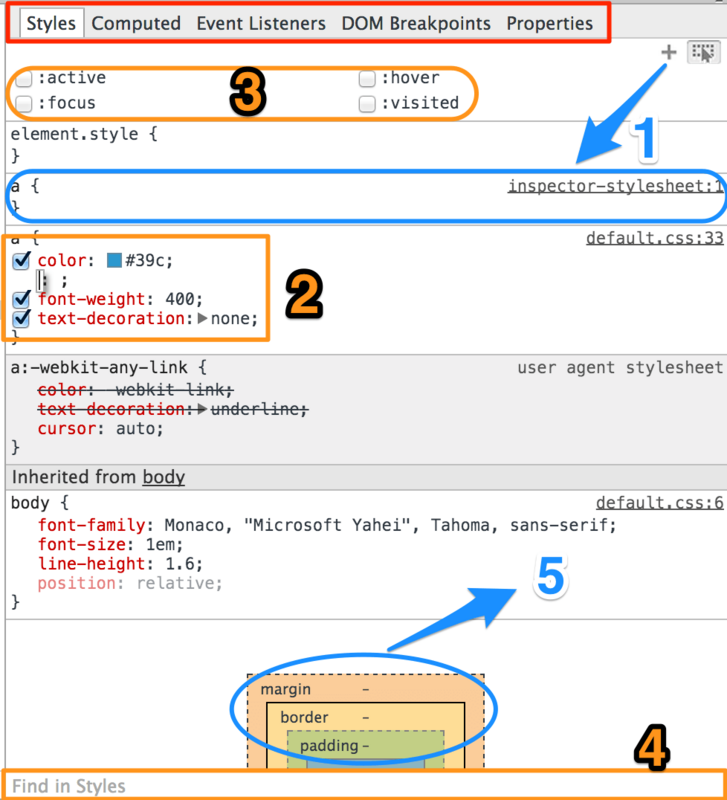
1. Edit as HTML: 以HTML形式更改页面元素；
2. Copy XPath: 复制XPath；
3. Delete Node: 删除DOM节点；
4. Break On: 设置DOM 断点。

图中被标记为3的蓝色区块显示当前标签的路径：从html开始一直到当前位置，我们单击路径中任何一个标签，即可以跳转到相应标签内容中去。

图中标记为4的蓝色区块实时显示当前选中标签的CSS样式、属性等，一共有以下5小部分：

* Styles: 显示用户定义的样式，比如请求的default.css中的样式，和通过Javasript生成的样式，还有开发者工具添加的样式；
* Computed: 显示开发者工具计算好的元素样式；
* Event Listeners: 显示当前HTML DOM节点和其祖先节点的所有JavaScript事件监听器，这里的监听脚本可以来自Chrome的插件。可以点击右边小漏斗形状(filter)选择只显示当前节点的事件监听器。
* DOM Breakpoints: 列出所有的DOM 断点；
* Properties: 超级全面地列出当前选中内容的属性，不过基本很少用到。

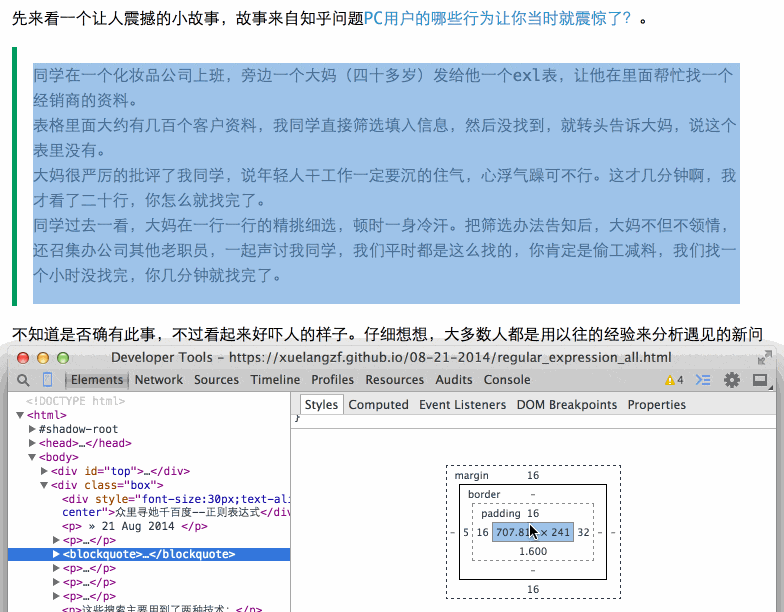
实际应用中我们经常会用到Styles，如下图：



图中标记为1的+号为New style rule，可以为当前标签添加新的选择器，新建立的样式为inspector-stylesheet。此外，也可以直接在原有的样式上增加、修改、禁用样式属性，如图中标记2显示。

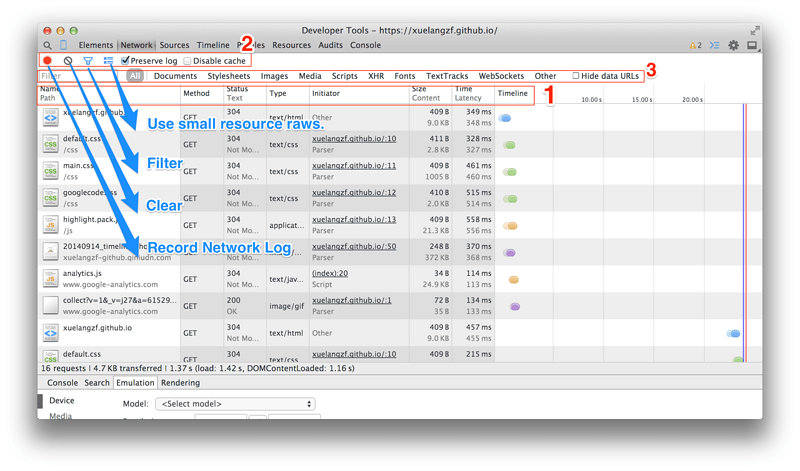
在New style rule右边为Toggle Element State，选择后会出现标记3显示的选择框，如果选中:hover后，即可以看到鼠标悬停在页面元素上时的CSS样式了，作用类似于前面的Force Element Satte，更多内容可以看[:hover state in Chrome Developer Tools](http://stackoverflow.com/questions/4515124/see-hover-state-in-chrome-developer-tools) 。

更强大的是，开发者工具以直观的图形展示了盒子模型的margin、border、padding部分，如标记5所示。下面动态图给出了盒子模型的一个示例：



[Network](https://developer.chrome.com/devtools/docs/network)

有时候我们的网页加载的很慢，而相同网速下，其他网页加载速度并不慢。这时候就得考虑优化网页，优化前我们必须知道加载速度的瓶颈在哪里，这个时候可以考虑使用Network工具。下图为我的博客首页加载时的Network情况：



请求的每个资源在Network表格中显示为一行，每个资源都有许多列的内容(如红色区块1)，不过默认情况下不是所有列都显示出来。

* Name/Path: 资源名称以及URL路径；
* Method: HTTP请求方法；
* Status/Text: HTTP状态码/文字解释；
* Type: 请求资源的MIME类型；
* Initiator解释请求是怎么发起的，有四种可能的值：
  1. Parser：请求是由页面的HTML解析时发送的；
  2. Redirect：请求是由页面重定向发送的；
  3. Script：请求是由script脚本处理发送的；
  4. Other：请求是由其他过程发送的，比如页面里的link链接点击，在地址栏输入URL地址。
* Size/Content: Size是响应头部和响应体结合起来的大小，Content是请求内容解码后的大小。进一步了解可以看这里[Chrome Dev Tools - “Size” vs “Content”](http://stackoverflow.com/questions/8072921/chrome-dev-tools-size-vs-content)；
* Time/Latency: Time是从请求开始到接收到最后一个字节的总时长，Latency是从请求开始到接收到第一个字节的时间；
* Timeline: 显示网络请求的可视化瀑布流，鼠标悬停在某一个时间线上，可以显示整个请求各部分花费的时间。

以上是默认显示的列，不过我们可以在瀑布流的顶部右键，这样就可以选择显示或者隐藏更多的列，比如Cache-Control, Connection, Cookies, Domain等。

我们可以按照上面任意一项来给资源请求排序，只需要单击相应的名字即可。Timeline排序比较复杂，单击Timeline后，需要选择根据Start Time、Response Time、End Time、Duration、Latency中的一项来排序。

红色区块2中，一共有6个小功能：

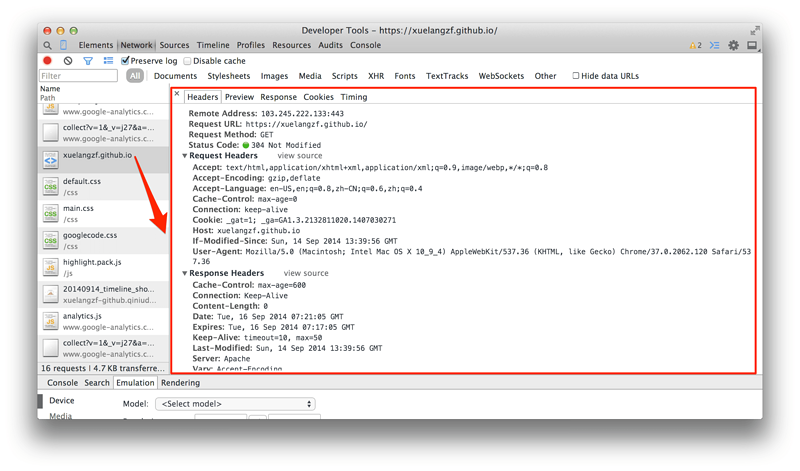
1. Record Network Log: 红色表示此时正在记录资源请求信息；
2. Clear: 清空所有的资源请求信息；
3. Filter: 过滤资源请求信息；
4. Use small resource raws: 每一行显示更少的内容；
5. Perserve Log: 再次记录请求的信息时不擦出之前的资源信息；
6. Disable cache: 不允许缓存的话，所有资源均重新加载。

选择Filter后，就会出现如红色区块3所显示的过滤条件，当我们点击某一内容类型(可以是Documents, Stylesheets, Images, Scripts, XHR, Fonts, WebSockets, Other)后，只显示该特定类型的资源。在XHR请求中，可以在一个请求上右键选择“Replay XHR”来重新运行一个XHR请求。

有时候我们需要把Network里面内容传给别人，这时候可以在资源请求行的空白处右键然后选择Save as HAR with Content保存为一个HAR文件。然后可以在一些第三方工具网站，比如[这里](http://ericduran.github.io/chromeHAR/)重现网络请求信息。

选定某一资源后，我们还可以Copy as cURL，也就是复制网络请求作为curl命令的参数，详细内容看[Copying requests as cURL commands](https://developer.chrome.com/devtools/docs/network#copying-requests-as-curl-commands)

此外，我们还可以查看网络请求的请求头，响应头，已经返回的内容等信息，如下图：



资源的详细内容有以下几个：

* HTTP request and response headers
* Resource preview: 可行时进行资源预览；
* HTTP response: 未处理过的资源内容；
* Cookie names and values: HTTP请求以及返回中传输的所有Cookies；
* WebSocket messages: 通过WebSocket发送和接收到的信息；
* Resource network timing: 图形化显示资源加载过程中各阶段花费的时间。

补充解释

**[XPath]**

XPath 是一门在 XML 文档中查找信息的语言。XPath 用于在 XML 文档中通过元素和属性进行导航。比如在图2中，

<a href="https://github.com/NUKnightLab/TimelineJS">这里</a>

这里a标签的Xpath为：/html/body/div[2]/p[1]/a，解读为：html里面body标签的第二个div标签的第一个p标签下的a标签。

**[HTML DOM事件]**

HTML DOM允许我们在某一事件发生时执行特定的JavaScript脚本，这里的事件可以是：

* 鼠标移到某元素之上；
* 窗口或框架被重新调整大小；
* 文本被选中；
* 用户在HTML元素上单击；
* 某个键盘按键被按下...

这里事件发生时执行的javascript脚本就是事件监听器。

**[DOM 断点]**

DOM 断点（DOM Breakpoints）可以监听某个 DOM 被修改情况，在Elements中某个元素上右键可以设置三种不同情况的断点：

* 子节点修改
* 自身属性修改
* 自身节点被删除

设置后 DOM Breakpoints 列表中就会出现该 DOM 断点。

一旦监听的DOM被修改时，断点就会定位到执行修改操作的代码，这对于绑定了多个事件的 DOM 调试有很大的帮助。

**[XMLHttpRequest]**

XMLHttpRequest是一个浏览器接口，使得Javascript可以进行HTTP(S)通信。XMLHttpRequest 对象用于在后台与服务器交换数据，这样就可以

* 在不重新加载页面的情况下更新网页
* 在页面已加载后从服务器请求数据
* 在页面已加载后从服务器接收数据
* 在后台向服务器发送数据

更多关于XHR的内容可以看阮一峰老师的[XMLHttpRequest Level 2 使用指南](http://www.ruanyifeng.com/blog/2012/09/xmlhttprequest_level_2.html)。

**[WebSocket]**

[WebSocket规范](http://dev.w3.org/html5/websockets/)定义了一种 API，可在网络浏览器和服务器之间建立“套接字”连接。简单地说：客户端和服务器之间存在持久的连接，而且双方都可以随时开始发送数据。这样就解决了长期以来只能由客户端发起请求才能从服务器获取信息这一问题。

更多关于WebSocket的内容可以看[WebSockets 简介：将套接字引入网络]()

**[本文环境]**

* 操作系统：OS X 10.9.4
* Chrome版本：Version 37.0.2062.120