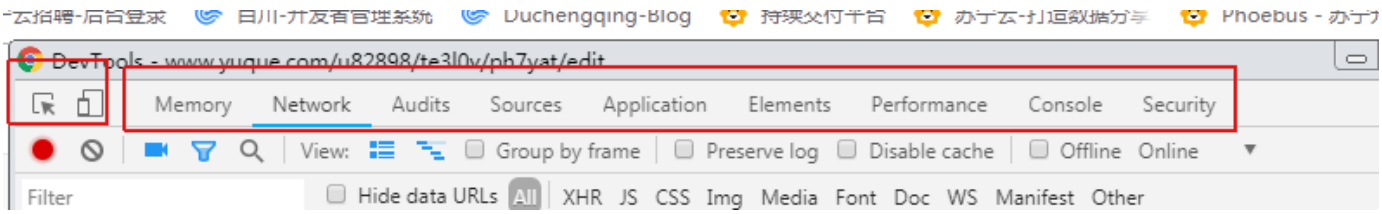
简诉

掌握一定的调试方法可以快速排查和定位问题，大幅度提高开发效率
chrome 开发者工具　　快捷键：F12

老司机

自己的代码、别人的代码、是否后端 ==== 性能大师

了解面板



- Memory：内存面板
- Network：　网络面板
- Audits：审计面板
- Source：　源代码面板
- Application：应用面板

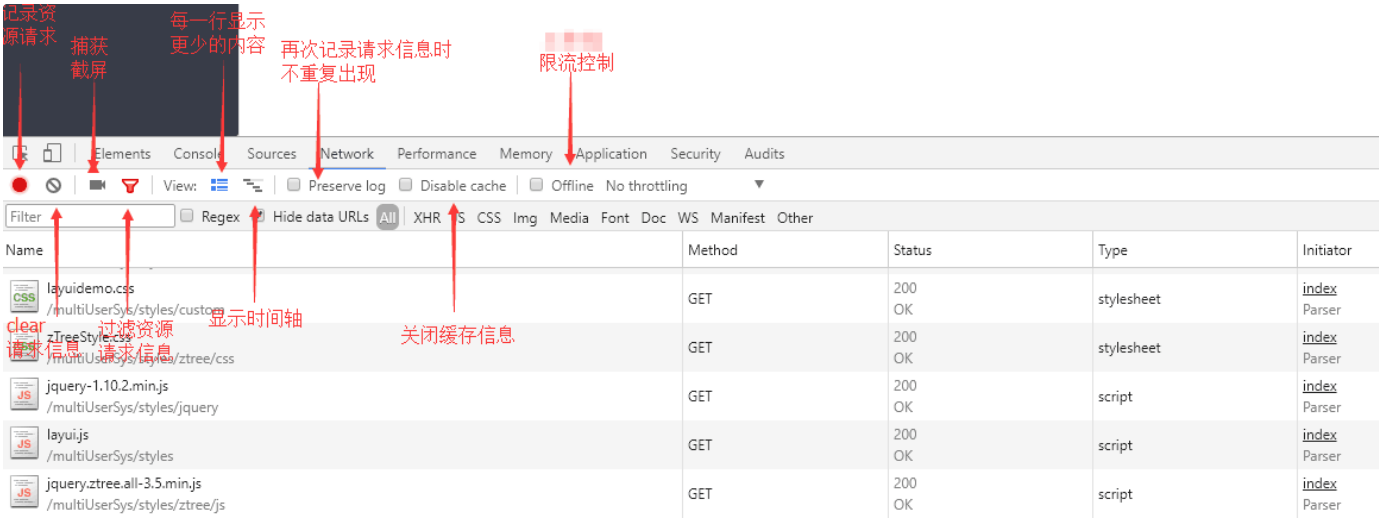
- Elements：元素面板
- Performance：性能面板
- Console：控制台面板
- Security：安全面板

Memory：

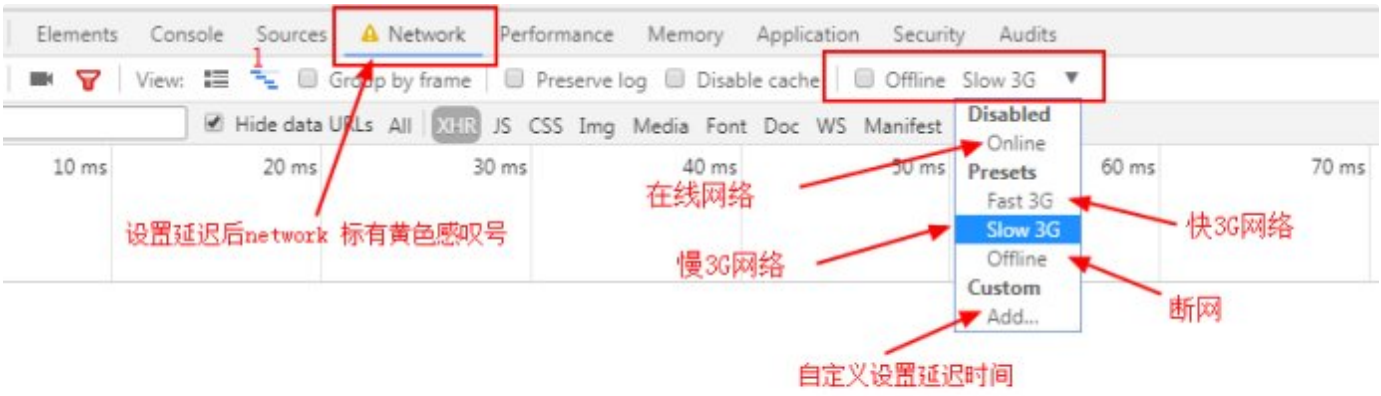
作用主要是监控网页中各种方法执行时间和内存的变化

Network：

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



Network 模拟网络延迟



1、**Cookies**

`Name`：cookie的名称。
`Value`：cookie的值。
`Domain`：cookie所属域名。
`Path`：cookie所属URL。
`Expire/Max-Age`：cookie的存活时间。
`Size`：cookie的字节大小。
`HTTP`：表示cookie只能被浏览器设置，而且JS不能修改。
`Secure`：表示cookie只能在安全连接上传输。

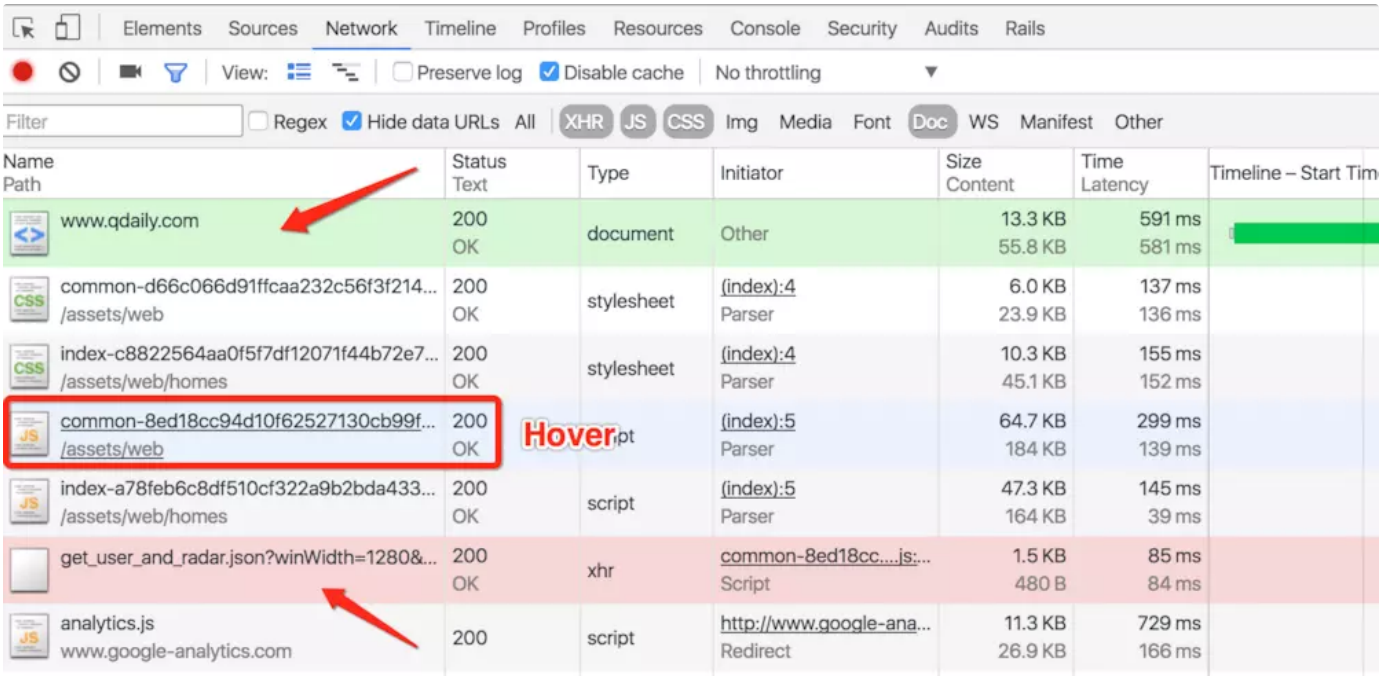
1、**Timing**

- **Queuing** 排队的时间花费。可能由于**该请求被渲染引擎认为是优先级比较低的资源（图片）、服务器不可用、超过浏览器的并发请求的最大连接数（Chrome的最大并发连接数为6）**.
- **Stalled** 从HTTP连接建立到请求能够被发出送出去(真正传输数据)之间的时间花费。包含用于处理代理的时间，如果有已经建立好的连接，这个时间还包括等待已建立连接被复用的时间。
- **Request sent** 发起请求的时间。
- **Waiting (Time to first byte (TTFB))** 是最初的网络请求被发起到从服务器接收到第一个字节这段时间，它包含了TCP连接时间，发送HTTP请求时间和获得响应消息第一个字节的时间。
- **Content Download** 获取Response响应数据的时间花费。

录制快照

如何查看资源请求的上游和下游?

按时shift键，鼠标hover在请求上，可以查看请求的上游和下游，如下图所示，hover在`common.js`上，可以看到有一个绿色请求、一个红色请求。其中绿色请求表示`common.js`的上游请求，即谁触发了`common.js`请求，红色请求表示`common.js`的下游请求，即`common.js`又触发了什么请求。



结合DOM文档加载的加载步骤，`DOMContentLoaded事件/Load事件`触发时机如下：

1. 解析HTML结构。
2. 加载外部脚本和样式表文件。
3. 解析并执行脚本代码。// 部分脚本会阻塞页面的加载
4. DOM树构建完成。//`DOMContentLoaded 事件`
5. 加载图片等外部文件。
6. 页面加载完毕。//`load 事件`

Disable cache

Audits

对当前网页进行网络利用情况、网页性能方面的诊断，并给出一些优化建议。比如列出所有没有用到的CSS文件等。

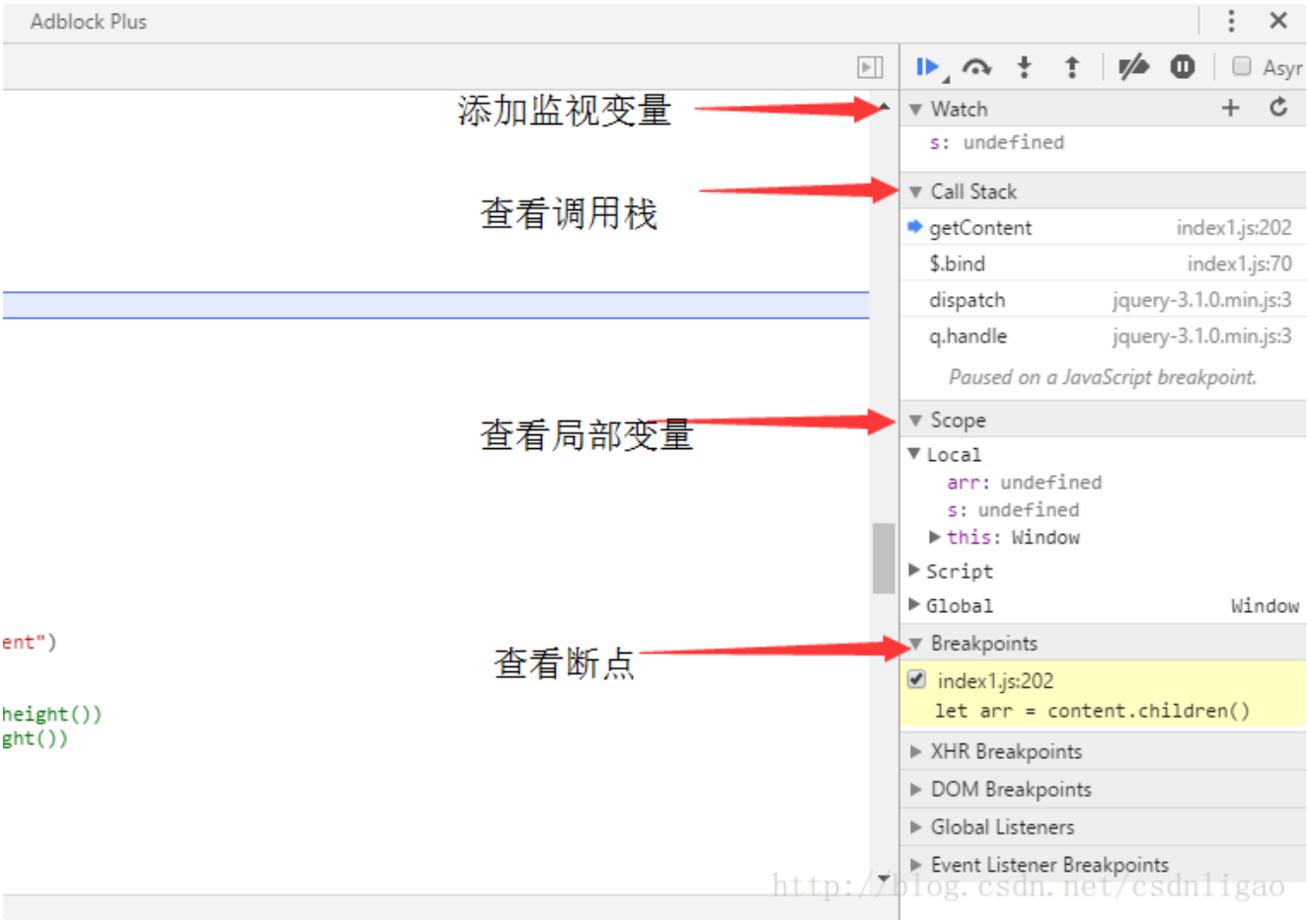
Source

断点调试JS

在js代码中写 debugger;会自动断点到该位置

可以设置断点调试，**如果当前代码经过压缩，可以点击下方的花括号{}来增强可读性**

- 也可以在右边的侧栏上查看：



- 在右侧变量上方，有继续运行、单步跳过等按钮，可以在当前断点后，逐行运行代码，或者直接让其继续运行。

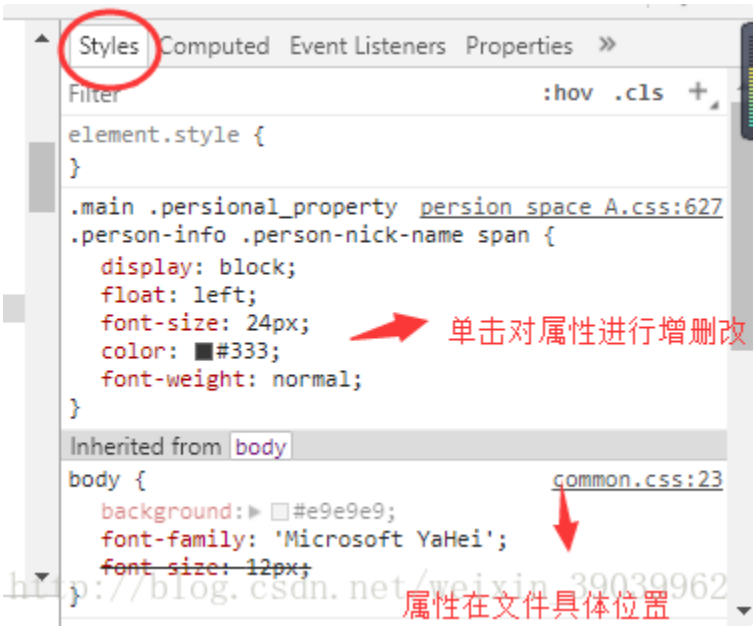
Applica77tion

记录网站加载的所有资源信息，包括存储数据555（Local Storage、Session Storage、IndexedDB、Web SQL、Cookies）、缓存数据、字体、图片、脚本、样式表等

Elements

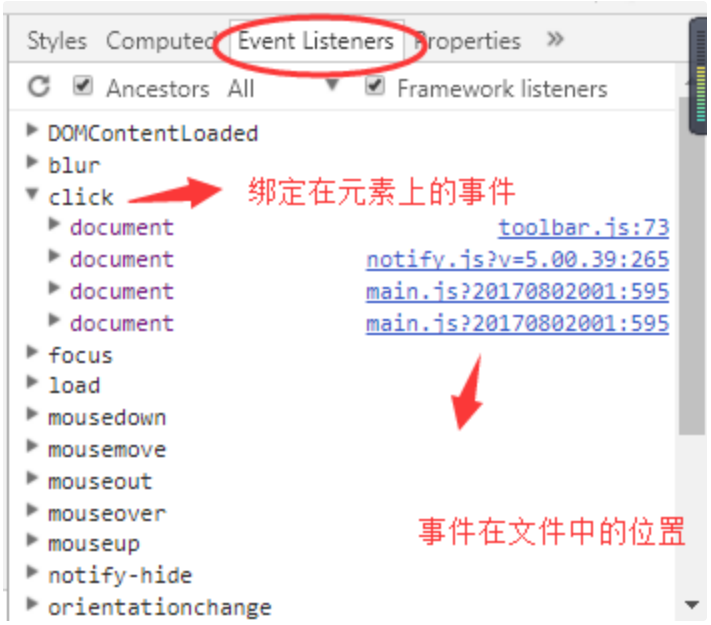
给元素添加断点：在元素的右键菜单中选择断点选项（Break on...），选中之后，当元素被修改（通常是被JS代码修改）时，页面加载会暂停，然后可以查看该元素的属性

Style css的预处理器，可见既可得。直接更改css样式在界面中可以直观的看到效果。点击具体位置之后，跳转到Sources位置，这个不影响源文件。如演示的样子，可以在Sources查看更改的历史

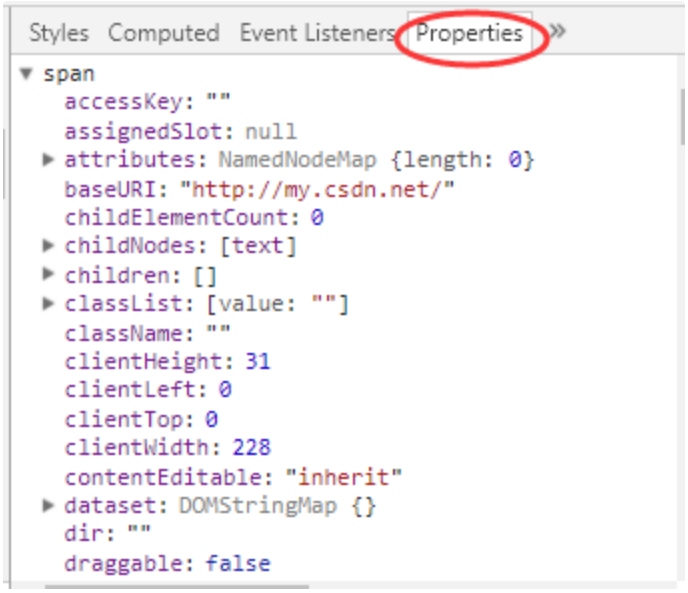


Event Listeners 元素绑定的事件

查看元素的监听事件：元素的右边栏的Event Listener页面，可以查看到该元素的所有监听事件。在开发中，尤其是维护其他人的代码时，会出现不了解元素对应的监听事件，这个时候，可以在这个页面中找到。这个页面不仅能看到对应的事件函数，还可以定位该函数所在的JS文件以及在该文件中的具体位置（行数），大大提高开发维护的效率



Properties 元素具有的属性与方法，比直接查api会更方便



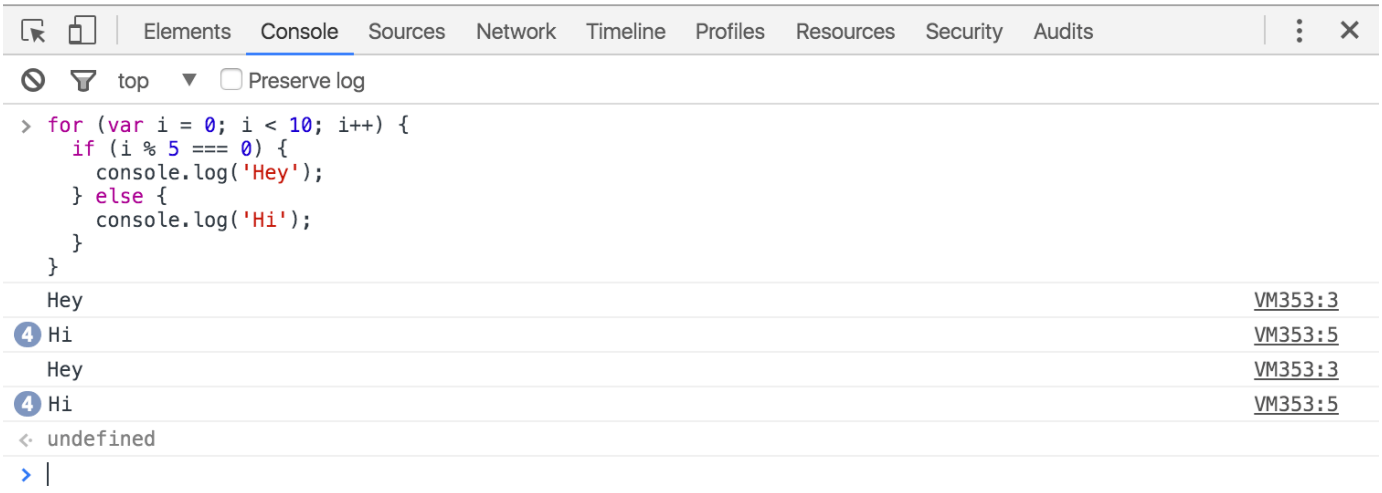
Performance

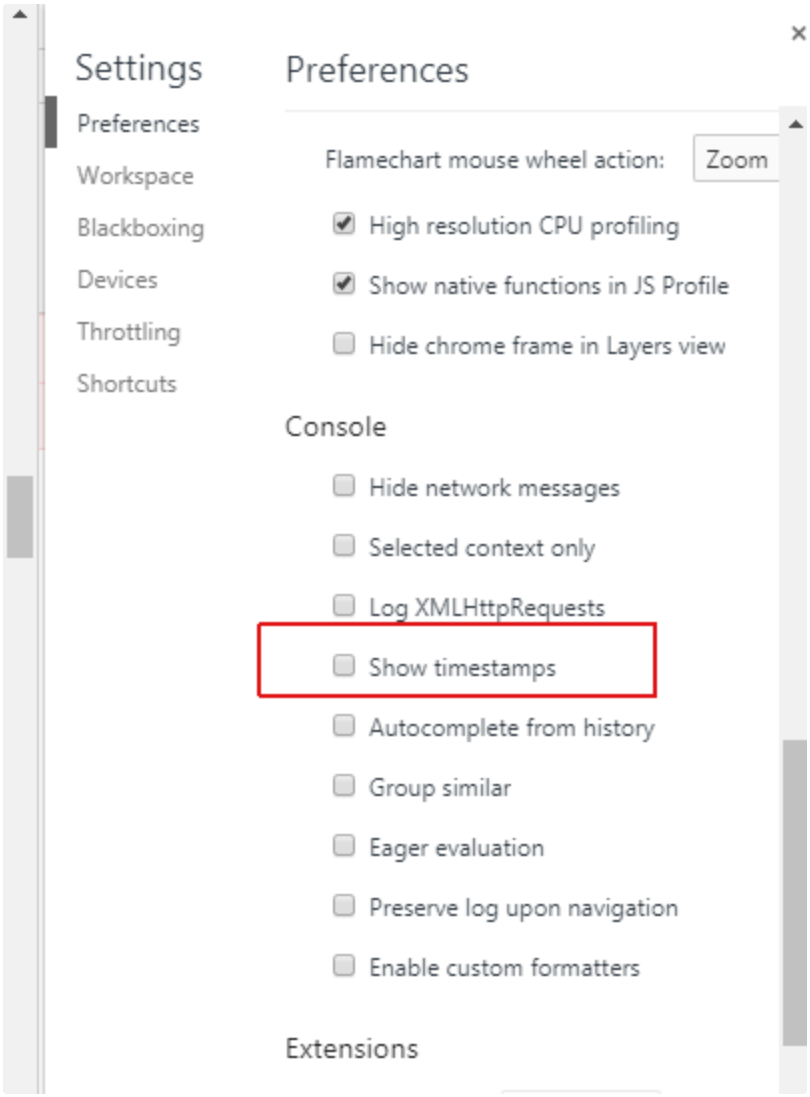
它的作用就是记录与分析应用程序运行过程中所产生的活动，更多的是用在性能优化方面

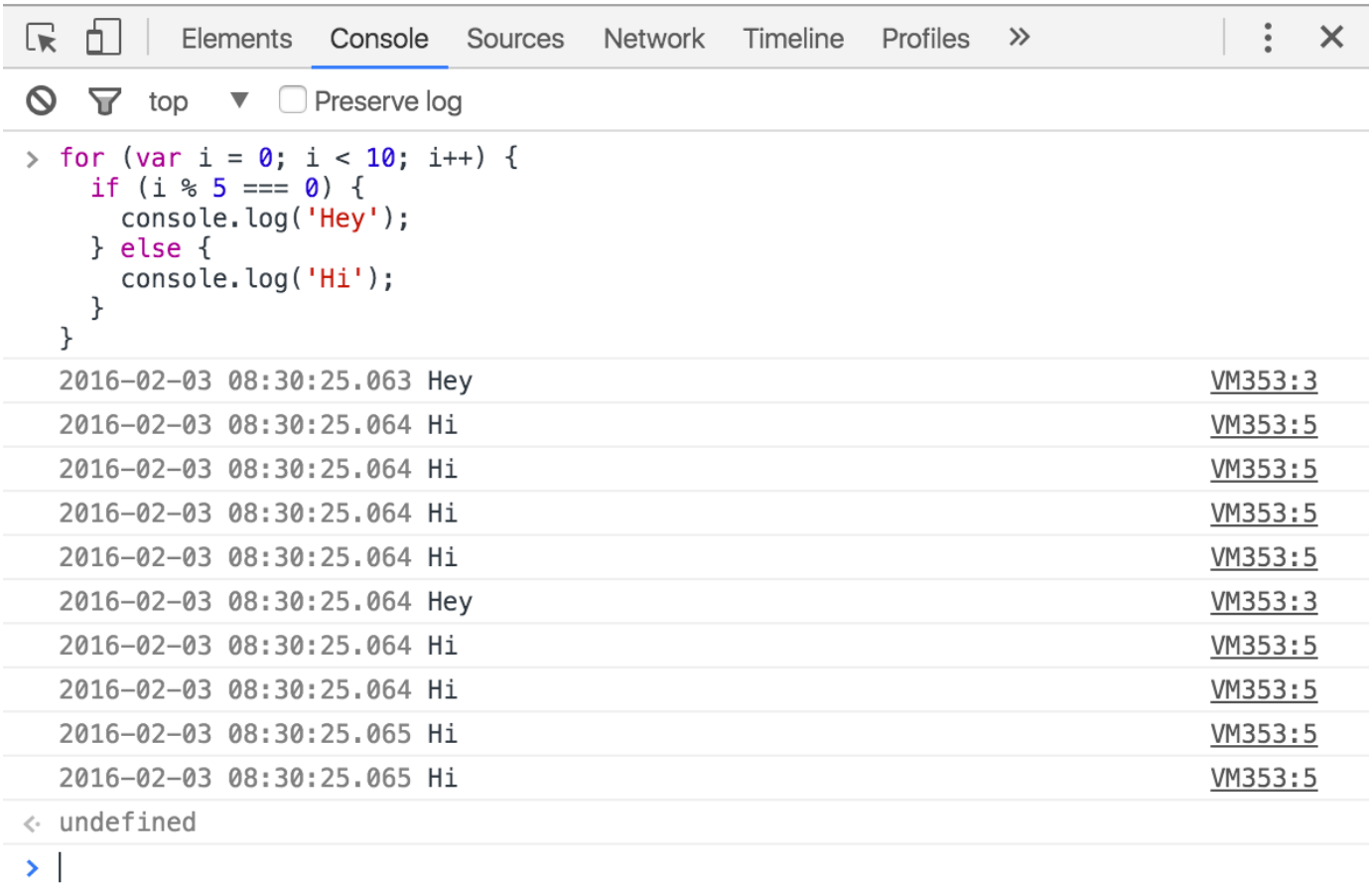
console

消息堆叠

如果一条消息连续重复，而不是在新行上输出每一个消息实例，控制台将“堆叠”消息并在左侧外边距显示一个数字。此数字表示该消息已重复的次数。




Settings
Preferences
Workspace
Blackboxing
Devices
Throttling
Shortcuts
Preferences
Flamechart mouse wheel action:
Zoom
High resolution CPU profiling
Show native functions in JS Profile
Hide chrome frame in Layers view
Console
Hide network messages
Selected context only
Log XMLHttpRequests
Show timestamps
Autocomplete from history
Group similar
Eager evaluation
Preserve log upon navigation
Enable custom formatters
Extensions



保留历史记录

启用控制台顶部的 **Preserve log** 复选框可以在页面刷新或更改之间保留控制台历史记录。 消息将一直存储，直至您清除控制台或者关闭标签。

Security

判断当前网页是否安全

HTTPS和HTTP的区别主要为以下四点：
① https协议需要到CA申请证书，一般免费证书很少，需要交费。
② http是超文本传输协议，信息是明文传输，https则是具有安全性的ssl加密传输协议。
③ http和https使用的是完全不同的连接方式，用的端口也不一样，前者是80，后者是443。
④ http的连接很简单，是无状态的；HTTPS协议是由SSL+HTTP协议构建的可进行加密传输、身份认证的网络协议，比http协议安全。

该面板可以区分两种类型的不安全的页面：

- 如果被请求的页面通过HTTP提供服务，那么这个主源就会被标记为不安全。
- 如果被请求的页面是通过HTTPS获取的，但这个页面接着通过HTTP继续从其他来源检索内容，那么这个页面仍然被标记为不安全。这就是所谓的**混合内容**页面,混合内容页面只是部分受到保护,因为HTTP内容(非加密的内容)可以被嗅探者入侵,容易受到**中间人攻击**。

Coverage

除移死代码

懒加载代码

如上图所示，Coverage 录制结果表格展示了录制过程中加载的所有 JS 和 CSS 文件，以及每个文件的大小、运行时覆盖率，汇总数据展示在页面底部的状态栏中（上面的截图没有展示）。单击单个静态资源能将其在 Sources 面板中打开，代码行号的左边用红绿色的条来标识代码是否在录制过程中被执行到。