

目 录

致谢

1. 初识Weinre

2. Weinre进阶

3. Q&A

4. References

来源

致谢

当前文档《Weinre入门手册》由 进击的皇虫 使用 书栈(BookStack.CN) 进行构建，生成于 2018-02-26。

书栈(BookStack.CN) 仅提供文档编写、整理、归类等功能，以及对文档内容的生成和导出工具。

文档内容由网友们编写和整理，书栈(BookStack.CN) 难以确认文档内容知识点是否错漏。如果您在阅读文档获取知识的时候，发现文档内容有不恰当的地方，请向我们反馈，让我们共同携手，将知识准确、高效且有效地传递给每一个人。

同时，如果您在日常生活、工作和学习中遇到有价值有营养的知识文档，欢迎分享到 书栈(BookStack.CN) ，为知识的传承献上您的一份力量！

如果当前文档生成时间太久，请到 书栈(BookStack.CN) 获取最新的文档，以跟上知识更新换代的步伐。

文档地址：<http://www.bookstack.cn/books/Weinre>

书栈官网：<http://www.bookstack.cn>

书栈开源：<https://github.com/TruthHun>

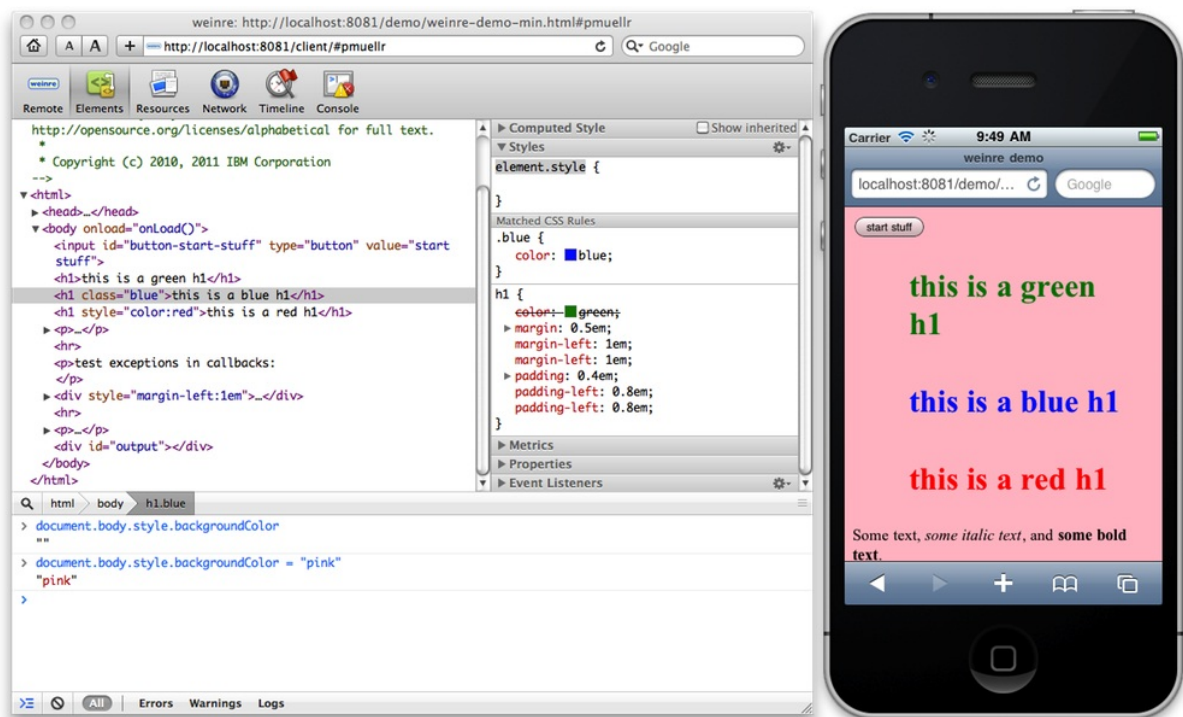
分享，让知识传承更久远！感谢知识的创造者，感谢知识的分享者，也感谢每一位阅读到此处的读者，因为我们都将成为知识的传承者。

1. 初识Weinre

- 1. 初识Weinre
 - 1.1 Weinre实验环境搭建
 - 1.1.1 安装Weinre
 - 1.1.2 运行Weinre
 - 1.1.3 搭建实验环境
 - 1.1(补充) 使用Grunt搭建环境
 - 1.2 Weinre用户接口与功能介绍

1. 初识Weinre

Weinre(Web Inspector Remote)是一款基于Web Inspector(Webkit)的远程调试工具，它使用JS编写， 可以让我们在电脑上直接调试运行在手机上的远程页面。 与传统的Web Inspector的使用场景不同， Weinre的使用场景如下图， 调试的页面在手机上， 调试工具在PC的chrome， 二者通过网络连接通信。



1.1 Weinre实验环境搭建

下面是Weinre的安装运行说明， 如果你熟悉Grunt， 那么也可以直接跳过这里， 按照下面1.1(补充)的步骤来搭建环境。

1.1.1 安装Weinre

Weinre是基于nodejs实现的，使用它必须先安装node运行环境，安装node可参考：[node官网](#)。新版的node已经集成了npm，所以直接在在命令行键入下面的命令即可安装，如果你是Mac/Linux用户，还需要在前面加入“sudo”：

```
1. [sudo] npm -g install weinre
```

安装成功，会得到下面的输出。如果你想根据自己的需要选择下载Weinre文档、源码、运行压缩包，那么可以通过这个网址下载：[点击查看](#)。

```
C:\Users\yiqi.hl>npm install -g weinre
C:\Users\yiqi.hl\AppData\Roaming\npm\weinre -> C:\Users\yiqi.hl\AppData\Roaming\npm\node_modules\weinre\weinre
weinre@2.0.0-pre-HH0SN197 C:\Users\yiqi.hl\AppData\Roaming\npm\node_modules\weinre
├── underscore@1.3.3
├── coffee-script@1.3.3
├── nopt@1.0.10 <abbrev@1.0.5>
├── express@2.5.11 <mime@1.2.4, qs@0.4.2, mkdirp@0.3.0, connect@1.9.2>
```

1.1.2 运行Weinre

在Terminal中输入weinre开启服务，

```
1. weinre
```

若运行成功，输出如下：

```
C:\Users\yiqi.hl>weinre
2014-07-28T03:05:16.749Z weinre: starting server at http://localhost:8080
```

像上面这样虽然启动成功了，但是默认boundHost为localhost，只能本地PC上用[http://localhost:8080](#)来访问，将localhost换做本地ip就无法打开Weinre调试工具，为了能在其他设备以及本地设备用ip打开Weinre调试工具，我们还需要设置boundHost为“-all-”，如下：

```
1. weinre --boundHost -all-
```

Weinre还提供了下面的启动参数：

--help : 显示Weinre的Help

--httpPort [portNumber] : 设置Weinre使用的端口号, 默认是8080

--boundHost [hostname | ip address | -all-] : 默认是'localhost', 这个参数是为了限制可以访问Weinre Server的设备, 设置为-all-或者指定ip, 那么任何设备都可以访问Weinre Server。

--verbose [true | false] : 如果想看到更多的关于Weinre运行情况的输出, 那么可以设置这个选项为true, 默认为false;

--debug [true | false] : 这个选项与--verbose类似, 会输出更多的信息。默认为false。

--readTimeout [seconds] : Server发送信息到Target/Client的超时时间, 默认为5s。

--deathTimeout [seconds] : 默认为3倍的readTimeout, 如果页面超过这个时间都没有任何响应, 那么就会断开连接。

执行完上面的命令, 在浏览器地址栏中输入下面的网址打开Weinre调试工具。

```
1. http://本地ip:8080
```

在页面中, 有两部分是我们使用的, 第一部分是Access Points, 如下图:

weinre - web inspector remote

Access Points

```
debug client user interface: http://10.68.124.159:8082/client/#anonymous
documentation: http://10.68.124.159:8082/doc/
```

红框中的地址是Debug Client(Weinre调试工具)的用户访问接口, 可以通过这个地址进入Debug Client。

第二个部分是Target Script, 如下图, 这个地址是系统根据我们启动Weinre服务时的参数设置生成的target-script.js文件的链接地址。我们需要将这个js文件嵌入到待测试的页面中。要注意的是不要使用localhost:8080打开Weinre服务, 否则生成的TargetScript链接也以localhost开头, 这样直接复制到手机, 就无法获取到文件了。

Target Script

You can use this script to inject the weinre target code into your web page.

```
http://10.68.124.159:8082/target/target-script-min.js#anonymous
```

Example:

```
<script src="http://10.68.124.159:8082/target/target-script-min.js#anonymous"></script>
```

1.1.3 搭建实验环境

有了上面的基础，我们就可以搭建实验环境了。在你的Web环境(apache/tomcat等)下创建下面的目录结构：

- `lib` //存放zepto.js的目录
 - `zepto.js`
- `data.json` //任意json格式的文件
- `index.html`

使用下面的命令运行Weinre服务：

```
1. weinre --httpPort 8082 --boundHost -all-
```

打开<http://本地ip:8082>，将TargetScript中生成的链接嵌入到待测试页中。并且通过AccessPoints中给出的用户接口进入DebugClient，编辑index.html，如下，最后的target-script.js链接需要自己修改。

```
1. <!DOCTYPE html>
2. <html><head><meta charset="UTF-8"></head>
3. <body>
4.     test
5.     <script type="text/javascript" src="lib/zepto.js"></script>
6.     <script type="text/javascript">
7.         $(function() {
8.             $.ajax({
9.                 url: 'data.json',
10.                success: function(res) {
11.                    console.log(res);
12.                }
13.            });
14.            window.localStorage.a = 'abc';
15.            console.log(window.localStorage);
16.        });
17.    </script>
```

```

18.     <script src="http://10.68.124.176:8082/target/target-script.js"></script>
19. </body></html>

```

在data.json文件中随便填入一些json格式的数据，至此实验环境已经搭建完成。

打开后的DebugClient如下图：



Targets

- none

Clients

- 10.68.124.176 [channel: c-34 id: anonymous]
- 10.68.124.176 [channel: c-33 id: anonymous]

Server Properties

```

boundHost:    -all-
deathTimeout: 15
debug:        false
httpPort:     8082
readTimeout:  5
staticWebDir: d:\work\myTest\weinreTest\node_modules\grunt-weinre\node_modules\weinre\web
verbose:      false
version:      2.0.0-pre-HHOSN197

```

首页RemoteTab由三部分组成，Targets是注册的远程设备列表，当前我们还没有访问测试页面，所以Targets列表为none，Clients是Weinre客户端，也即打开这个Weinre页面的设备列表。ServerProperties就是我们启动Weinre时的一些配置项。在手机上打开测试页就可以开始调试了。

1.1(补充) 使用Grunt搭建环境

为实验方便，这里使用到了grunt-contrib-watch与grunt-contrib-connect。这两个插件的使用请参考[watch](#)，[connect](#)，这里不做介绍，要注意的是这里假设你已经安装了node运行环境。

a. 首先使用npm安装以下插件（package.json文件中依赖如下）

```

1. "devDependencies":{
2.     "grunt":"^0.4.5",
3.     "grunt-weinre":"~0.0.2",
4.     "grunt-contrib-connect": "~0.7.1",
5.     "load-grunt-tasks": "~0.4.0",

```

```

6.     "grunt-concurrent": "~0.5.0",
7.     "grunt-contrib-watch": "~0.6.1"
8. }

```

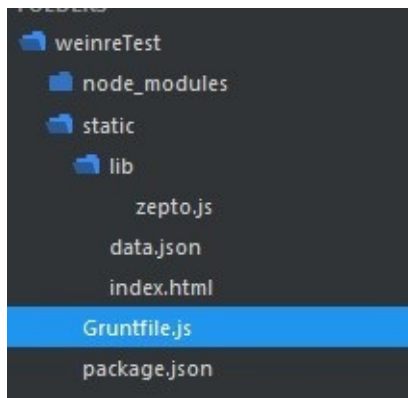
b. 安装好后，配置Gruntfile.js如下：

```

1. module.exports = function(grunt) {
2.     require('load-grunt-tasks')(grunt);
3.     grunt.initConfig({
4.         watch: {
5.             livereload:{
6.                 options:{ livereload:true },
7.                 files:[ '*.html' ]
8.             }
9.         },
10.        weinre: {
11.            dev: {
12.                options: { httpPort: 8082, boundHost: '-all-' }
13.            }
14.        },
15.        connect: {
16.            options: { port: 9000, open: true, livereload: 35729, hostname: '0.0.0.0'},
17.            livereload: {
18.                options: {
19.                    middleware: function(connect) {
20.                        return [ connect.static('static') ];
21.                    }
22.                }
23.            }
24.        },
25.        concurrent: { dist: ['weinre','watch'] }
26.    });
27.    grunt.registerTask('default', ['connect','concurrent']);
28. };

```

c. 建立如下图的目录结构，static目录是我们在Gruntfile中配置的静态文件目录，里面存放我们的测试页面以及相关的静态资源，这里为了测试ajax请求，使用到了一个data.json文件与zepto.js。



在Terminal输入下面命令，开启Weinre服务：

```
1. grunt
```

打开<http://本地ip:8082>，将TargetScript中生成的链接嵌入到待测试页中。并且通过AccessPoints中给出的用户接口进入DebugClient，编辑index.html，如下，最后的target-script.js链接需要自己修改。

```
1. <!DOCTYPE html>
2. <html><head><meta charset="UTF-8"></head>
3. <body>
4.     test
5.     <script type="text/javascript" src="lib/zepto.js"></script>
6.     <script type="text/javascript">
7.         $(function() {
8.             $.ajax({
9.                 url: 'data.json',
10.                success: function(res) {
11.                    console.log(res);
12.                }
13.            });
14.            window.localStorage.a = 'abc';
15.            console.log(window.localStorage);
16.        });
17.    </script>
18.    <script src="http://10.68.124.176:8082/target/target-script.js"></script>
19. </body></html>
```

d. 在data.json文件中随便填入一些json格式的数据，至此实验环境已经搭建完成。打开后的DebugClient如下图：



Targets

- none

Clients

- 10.68.124.176 [channel: c-34 id: anonymous]
- 10.68.124.176 [channel: c-33 id: anonymous]

Server Properties

```
boundHost:    -all-
deathTimeout: 15
debug:        false
httpPort:     8082
readTimeout:  5
staticWebDir: d:\work\myTest\weinreTest\node_modules\grunt-weinre\node_modules\weinre\web
verbose:      false
version:      2.0.0-pre-HHOSN197
```

首页RemoteTab由三部分组成，Targets是注册的远程设备列表，当前我们还没有访问测试页面，所以Targets列表为none，Clients是Weinre客户端，也即打开这个Weinre页面的设备列表。ServerProperties就是我们运行Weinre时的一些配置项。

运行grunt后，浏览器会自动打开网页<http://0.0.0.0:9000>，因为我们在Gruntfile中设置了hostname为0.0.0.0，这个设置是为了让外部ip访问这个文件服务，这时我们在浏览器地址栏将0.0.0.0改为我们电脑的ip，然后用手机访问这个链接，例如我的链接是<http://10.68.124.176:9000>，这样就可以在Targets列表中看到我们手机的ip地址，证明手机链接上了Weinre服务。如果有多个设备访问这个网页，那么Targets列表就会将这些设备的ip都列出来，默认情况是蓝色显示列表项，如果需要调试某个设备的页面，那么可以在Target列表中点击那个设备的对应的列表项，可以看到当点击后，该项变为绿色表示选中，此时测试的就是选中设备的页面。

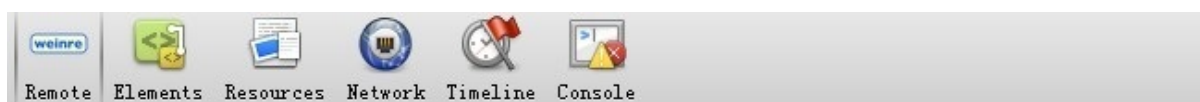
1.2 Weinre用户接口与功能介绍

接下来将从以下几个方面介绍Weinre工具的使用：调试DOM/CSS，调试Ajax请求，本地存储的使用，Console的使用，这几方面分别对应着Weinre工具的ElementsTab，NetworkTab，Resources-Tab，ConsoleTab。



在进入调试步骤前，要确保我们选中正确的Targets，如下图，选择待调试设备的ip，选中会显

示绿色，蓝色是未选中。



Targets

- 10.68.111.123 [channel: t-3 id: anonymous] - http://10.68.124.159:8080/index.html
- 10.68.124.159 [channel: t-2 id: anonymous] - http://10.68.124.159:8080/index.html

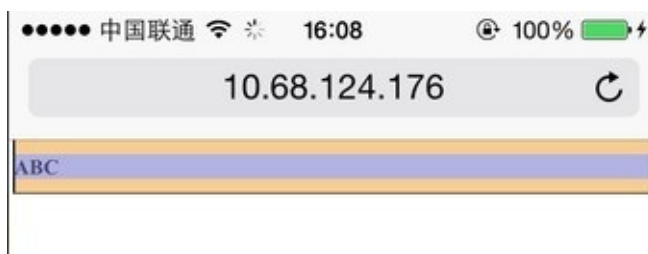
• 调试DOM/CSS

切换至ElementsTab， 可以看到调试页面的html结构， 如果看不到或者不是需要的页面，有可能是Targetslist中的目标设备选择的不正确。在这里我们就可以像调试本地页面一样的修改这个文件，可以修改DOM结构也可以修改元素样式。例如：当我们在ElementTab中选中h1标签时，手机上我们可以看到h1元素被高亮框圈了起来，此时我们给h1元素设置字体颜色为红色，就可以看到手机上的“ABC”变为了红色。

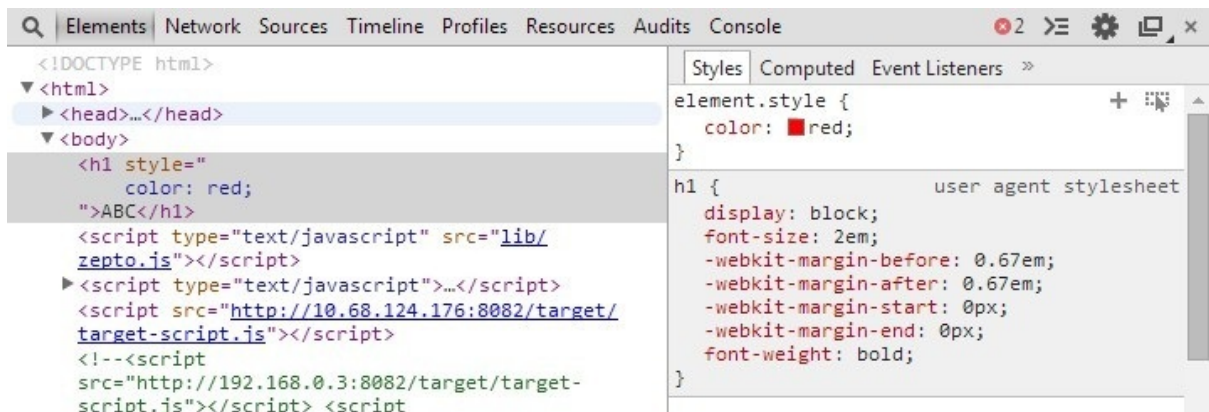
a. ElementTab选中h1标签：

```
<!DOCTYPE html>
<html>
  <head>...</head>
  <body>
    <h1>ABC</h1>
    <script type="text/javascript" src="lib/zepto.js"></script>
    <script type="text/javascript">...</script>
    <script src="http://10.68.124.176:8082/target/target-script.js"></script>
    <script type="text/javascript">...</script>
    <script src="http://10.68.124.176:35729/livereload.js?snipver=1" type="text/javascript"></script>
  </body>
</html>
```

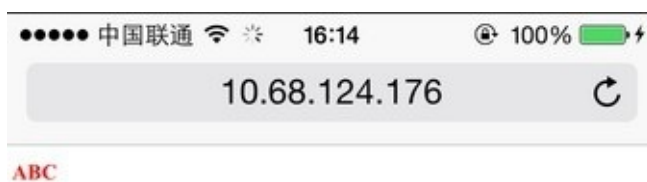
a. 选中时手机的页面的效果：



b. 将字体颜色修改为红色：



b. 修改后手机上的页面效果：



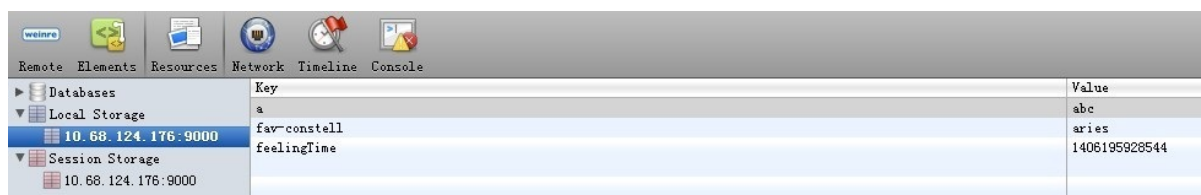
- 调试Ajax请求

切换至NetworkTab，在这里可以看到我们发送的ajax请求， 如我们搭建环境时使用到的html1页面， 在页面中， 请求了一个data.json的数据文件， 当页面加载完成后就可以看到在NetworkTab中有一条请求data.json的数据， 这个与我们平时调试本地页面一样， 不同的只是这里不能显示页面加载资源的情况， 比如页面中引入的js， css， img文件的加载情况， 所以如果资源加载错误， 在这里是无法看出来的。 值得一提的是， ajax显示的收据也仅限于远端页面已经与Weinre链接建立成功后的请求。请求结果如下图：

Remote Elements Resources Network Timeline Console						
Name	Method	Status	Type	Size	Time	
Path		Text		Transfer	Latency	
data.json	undefined	Pending	Pending		0B 0B	Pending

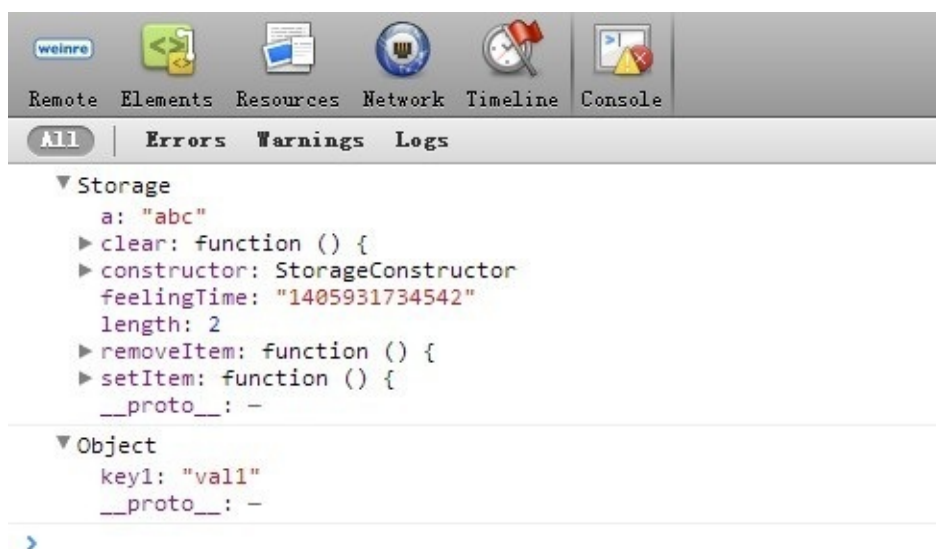
- 本地存储的使用

切换至ResourcesTab， 这是对应本地调试工具的一个缩减版， 可以看到只有Database， Local Storage与Session Storage三项， 我们的例子还是使用上面的index.html， 其中有一处使用了localStorage存储， 运行后， 我们可以切换到LocalStorage项， 在里面就可以看到我们写入的值了， 如下图：



- Console的使用

切换至ConsoleTab，这里基本与本地调试时console的功能一致，可以输入执行一些js代码，也可以显示console输出的信息。以index.html为例，可以看到如下图的输出：



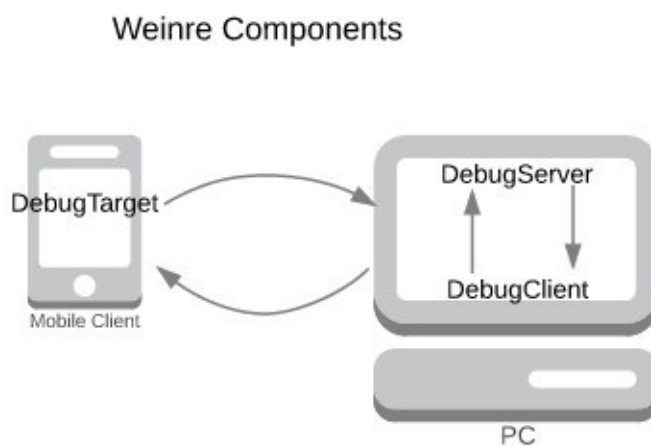
Ajax请求与Console的输出要发生在Target连接成功之后才能看到结果，所以如果上面的例子看不到结果，可以加入`setTimeout`给`console.log`一个延时。

2. Weinre进阶

- 2. Weinre进阶
 - 2.1 Weinre工作原理
 - 2.2 Multi-User
 - 2.3 Weinre安全性
 - 2.4 Weinre与其他类似工具比较
 - 2.5 远程调试未来的发展

2. Weinre进阶

2.1 Weinre工作原理



如上图，Weinre由三部分组成，第一部分是运行在PC上的Debug Server，它会与其他两部分交互，我们在测试页引入的那个target.js文件就存在于这个Server里，如果你想找到那个Target文件，可以通过下图的路径找到：

Targets


- none

Clients

- 192.168.0.3 [channel: c-1 id: anonymous]

Server Properties

```
boundHost:    -all-
deathTimeout: 15
debug:        false
httpPort:     8082
readTimeout:  5
staticWebDir: d:\work\myTest\weinreTest\node_modules\grunt-weinre\node_modules\weinre\web
verbose:      false
version:      2.0.0-pre-HHOSN197
```



第二部分是Debug Client，这个就是我们上面一直在使用的运行在chrome中的调试客户端，它与Debug Server进行连接，并提供调试接口给用户。

第三部分是Debug Target，也就是运行在我们远程设备浏览器中的target.js，它通过XHR与Server连接交互，将我们的代码暴露给Server，来实现DOM Inspection与修改。

我们实验的步骤也就是分别开启这三部分，运行grunt后，就开启了我们设置好的Debug Server，然后在浏览器中输入<http://ip:weinre端口>就打开了Debug Client，在调试页面中嵌入target.js代码，在手机中打开页面，就开启Debug Target。

2.2 Multi-User

这里所说的多用户，并不是支持多用户同时修改调试同一个页面，只是提供了DebugServer的多用户共享功能，也就是说无需每个人都开一个Weinre服务来调试自己的页面，只要运行一个Debug Server，各自在各自的Debug Client上调试自己的Debug Target。

要使用这个功能，需要给每个用户分配一个id，他们通过自己的id来链接Weinre服务，这里仅仅是一个简单的标识，任何人都可以通过别人的id调试别人的页面，造成混乱，weinre并没有提供任何安全服务来避免这种情况，按照官网的文档，这个id需要保密，通过这种方式，可以勉强解决这个问题。Weinre默认就是以多用户模式启动，只是我们没有设置id，它会自动为我们分配一个叫做anonymous的id。

按照上面的解释，可以知道，我们只要修改Debug Client的访问路径和Debug Target中嵌入的target.js的url，就可以使用这个功能了，具体的说就是在各自的url最后加入#id(id为用户分配的id)即可。下图是修改后的打开Debug Client的路径：


```
http://some.server.xyz/client/
```

```
http://some.server.xyz/client/#itsReallyMe
```

下图是修改后嵌入调试页面的Debug Target路径:

```
<script src="http://some.server.xyz/target/target-script-min.js"></script>
```

```
<script src="http://some.server.xyz/target/target-script-min.js#itsReallyMe"></script>
```

2.3 Weinre安全性

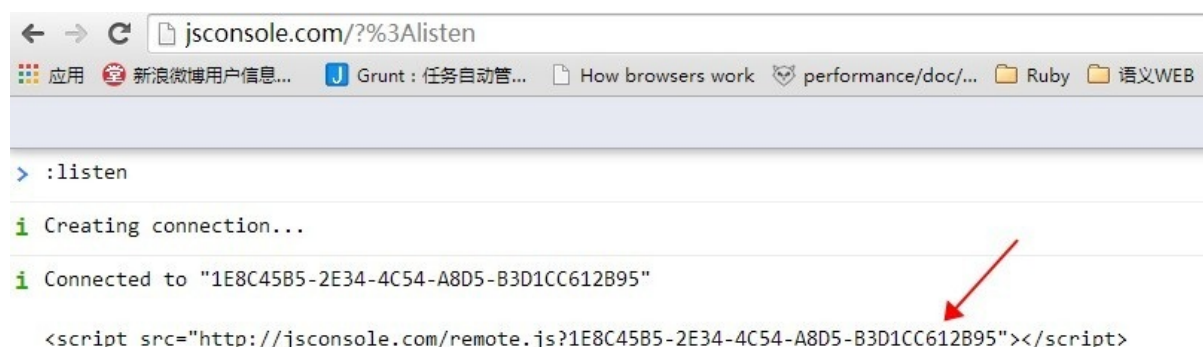
就像官网中说的: About security for weinre: there is none. Weinre解决的问题安全问题就是限制Debug Server的访问权限, 它的实现也是通过简单的设置boundHost选项来完成, 如果启动Server时设置boundHost选项为localhost, 那么只有运行在localhost上的软件能与Debug Server交互, 如果设置为"-all-"或其他的话, 那么任何能访问运行Debug Server设备的远程设备, 都可以与Debug Server交互, 这就可能造成信息泄露。

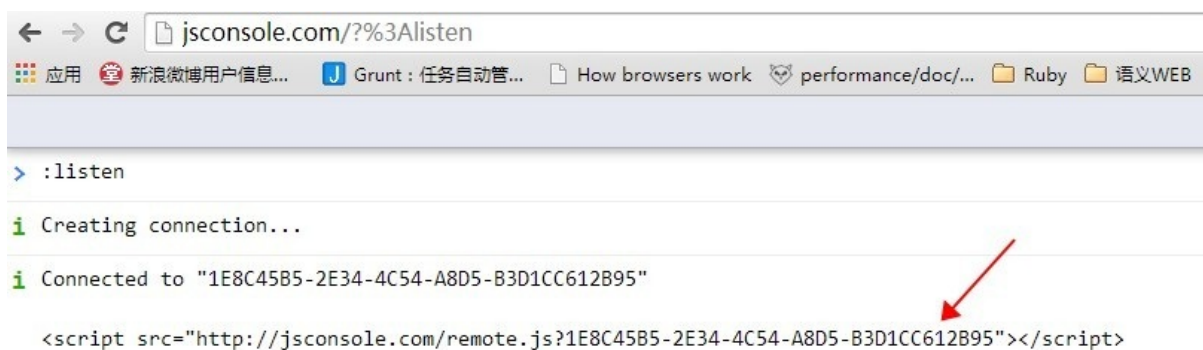
2.4 Weinre与其他类似工具比较

- JSConsole

这个工具与Weinre功能类似, 更像是简化版的Weinre, 它相对于Weinre的优点就是提供了现成了线上Debug Server与Debug Client, 无需用户在PC本地运行Debug服务, 只要在需要调试的页面像Weinre一样加入一个target库, 就可以在JSConsole官网上调试这个页面了。点击[这里](#)体验JSConsole。

使用步骤: 打开JSConsole官网, 输入":listen"后, 会得到如下的一个url, 这个就是target库, 我们要将输出的结果复制到要调试的页面中, 接下来在手机上访问这个调试页面, 如果页面中有console输出, 那么就可以在网站上看到输出的结果。





这个工具更专注于Console功能， 它还提供了一个运行在苹果设备上的软件， 下载地址如下：[下载链接](#)。与这个app类似的还有一款收费软件叫做Bugaboo， 它也是运行在苹果设备上的app， 同样是在手机上提供了一个console功能。

2.5 远程调试未来的发展

从Weinre的功能介绍已经可以看出存在很多不足了， 这些功能的欠缺就是远程调试未来要解决的问题， 例如调试javascript、 完善Resources资源的查看包括Cookie等、Network资源加载的显示的完整性等。因为Weinre是使用javascript编写的， 并不支持底层的断点调试等功能。现在的V8并不包含调试模块， 只有最新的webkit协议才拥有web调试特性， 如果手机上的浏览器具有了这样的调试特性， 那么只要通过远程协议就可以在PC上调试远程mobile页面。BlackBerry， 与Opera Mobile浏览器已经加入了远程调试特性， 通过下面的链接可以学习如何使用这样的新特性：[BlackBerry](#)， [Opera Mobile](#)

3. Q&A

3. Q&A

- Q: `console.log`无法显示在Client的ConsolePanel上。

A: 要确保webview已经完全加载并且在Weinre上注册成功，只有满足这两个条件才能看到`console.log`的内容。判断是否注册成功的方法是，在Client的Targetlist中能看到远程端设备的信息。下图不满足条件，所以在Console面板中看不到输出信息。

```
<body>
  test
  <script type="text/javascript" src="lib/zepto.js"></script>
  <script type="text/javascript">
    console.log(123);
  </script>
  <script src="http://10.68.124.176:8082/target/target-script.js"></script>
</body>
</html>
```

需要使用`setTimeout`加一个延时，保证输出信息在连接成功之后发生。

- Q: Network Panel都能看到哪些信息？

A: 如果webview已经与Client建立了链接，那么在Network Panel中可以看到ajax请求的数据。为什么看不到页面加载过程的请求的原因同上(注册成功前，无法获取到)。

- Q: 为什么不支持JS调试

A: Chrome开发者工具使用了很多native的代码来实现调试功能，而Weinre是完全基于JS的，没有包含任何native代码，JS并没有对本地的断点等的功能的支持。

- Q: 页面有时候没过一会显示的数据就不见了是怎么回事？

A: 如果出现这种情况，可能是超时造成的，在启动Weinre的时候，可以带上`deathTimeout`参数延长超时时长。

4. References

References

- [1] [Debugging Mobile Javascript with WEINRE](#)
- [2] [Debugging mobile web applications with weinre](#)
- [3] [Web移动应用调试工具-Weinre](#)
- [4] [Weinre官网](#)
- [5] [Console.log not working with weinre?](#)
- [6] [Weinre Issues问题汇总](#)
- [7] [Debugging Mobile Web Apps: Weinre and JSConsole Now, Remote WebKit Eventually](#)
- [8] [Debugging Mobile Web Apps: Weinre and JSConsole Now, Remote WebKit Eventually](#)翻译版
- [9] [JSConsole工具](#)

来源

来源

<https://github.com/nupthale/weinre>