Debugging Mobile Javascript with WEINRE

2.5
94e7fded-7162-445e-8ceb-97a2140866a9 Blog
Expanded section

▼ Tags ?

Expanded section

▼ Archive

Expanded section

▼ Blog Authors

Clicking the button causes a full page refresh. The user could go to the "Entry list" region to view the new content.) Entry list

Debugging Mobile Javascript with WEINRE

Debugging javascript on a mobile device can be maddening. As a novice, I usually have no idea what has gone wrong, and I am often frustrated with a blank white screen and no clues. I could learn to use the debug tools available on each platform, but that would become onerous quickly.

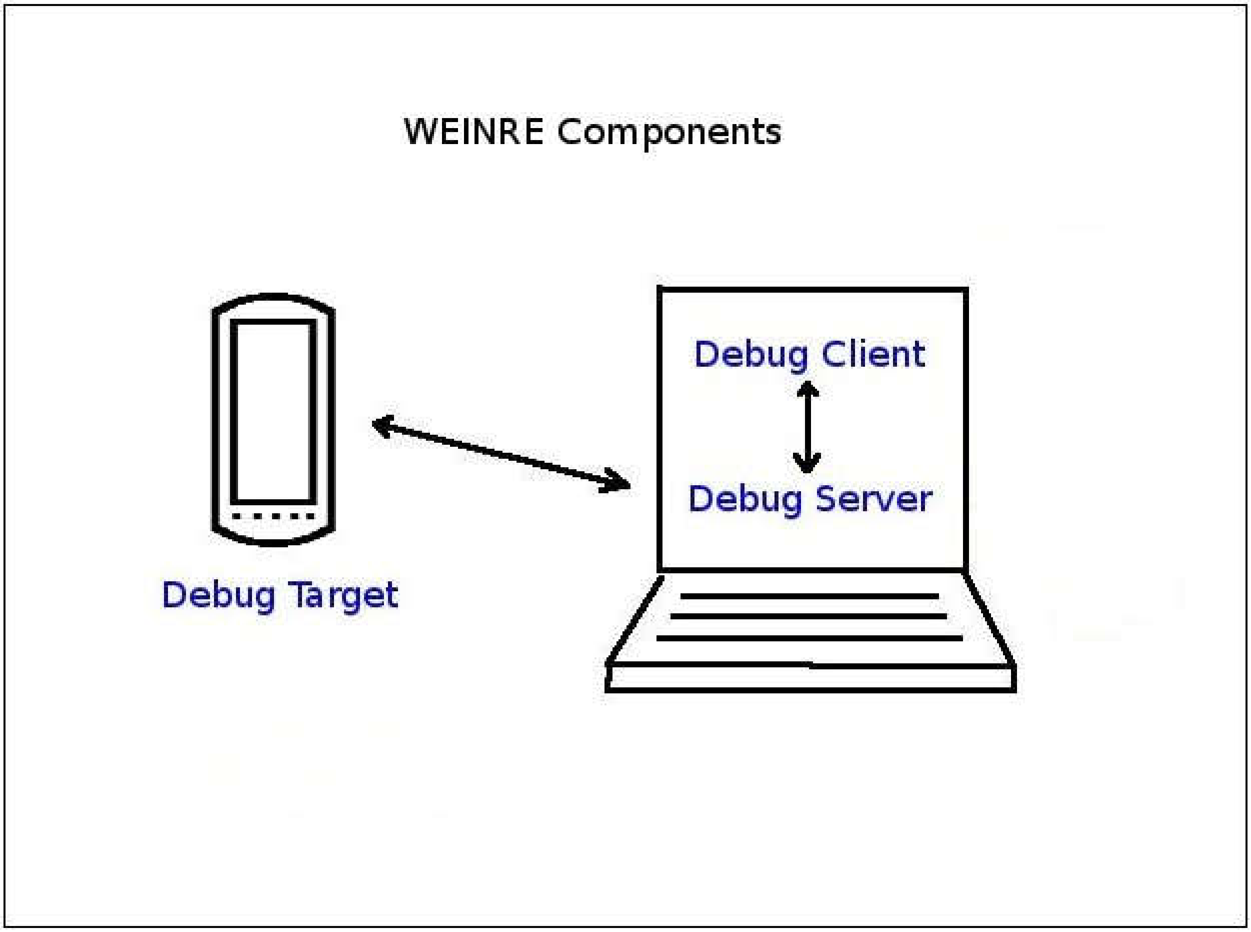
I recently discovered an open-source package called WEINRE, or Web Inspector Remote, written by fellow IBMer Patrick Mueller. WEINRE interacts with the running javascript. It has views for elements, resources, and the console, which let you view and modify the DOM, see console.log messages, and much more.

I was previously familiar with debuggers like Web Inspector (and Firebug) which explore the code running in the co-located browser. But WEINRE

does it remotely. And since it works with any webkit browser, like those in Android and iOS mobile devices, it is a perfect match for mobile debugging. Continue reading for a brief explanation of how WEINRE works, and a quick demonstration of how I started using it...

How it works

Figure 1. WEINRE Components



WEINRE consists of three pieces...

The first piece is a server called the “Debug Server”. It is the core component, and runs on a desktop machine or laptop, and communicates with the other two pieces.

The second piece is the “Debug Client”. This is a traditional webkit browser running on the same or different machine as the server. It connects to the server and presents the user with the familiar debugger view (like Web Inspector), where you can see what is going on and change the DOM.

The third and final piece is the “Debug Target”, which is the webkit-browser-under-test. It consists of a javascript library which runs in your mobile browser along with your app. It communicates with the server via XHR.

With a little linkage to your code, this library exposes your javascript to the server for inspection and modification.

Where to get it

I found lots of good information, documentation, and instructions at the WEINRE home page:
http://phonegap.github.com/weinre/Home.html

I downloaded WEINRE here. I went to Downloads-> weinre-jar-1.4.0.zip
https://github.com/phonegap/weinre

Starting the Debug Server
First, I fetched the WEINRE jar from the link above to my laptop, unzipped it, and found the jar file. I started the jar file, specifying a concrete IP address on my laptop. I did not accept the default localhost, because my mobile phone would not have been able to connect to it. I did accept the server's default listen port of 8080.

```
    java -jar weinre.jar --boundHost 192.168.2.4
```

Aside: The author of WEINRE publishes executables for the Debug Server in two formats: a java jar, and a MAC executable. I tested the java version only.

Starting the Debug Client

I started Chrome on my laptop, and browsed to the server:

```
    http://192.168.2.4:8080/client/#anonymous
```

Aside: One WEINRE debug server can support multiple clients and targets. As a single client, I used “anonymous”.

This brought up a screen which looks much like a web inspector screen (see Figure 2 below).

Instrumenting the HTML/Javascript in my Debug Target

I selected one of my sandbox programs which uses the PhoneGap and Dojo Toolkit to play an mp3 music stream. (You can get a copy here: (weinre.sample.2011-0719-1209.html). I instrumented it by adding three lines of code:
In the head section, following the dojo and phonegap scripts, I added this statement. This one line is the only instrumentation which is required to start debugging and inspect and modify the DOM using WEINRE.

```
    <script src="http://192.168.2.4:8080/target/target-script-min.js#anonymous"></script>
```

Note: This link is hard-coded directly to the explicit IP address and port of my Debug Server. Yours should link to your Debug Server.
To indulge my preferred style of debugging, I also added two console.log statements to the two javascript methods which start and stop the music:

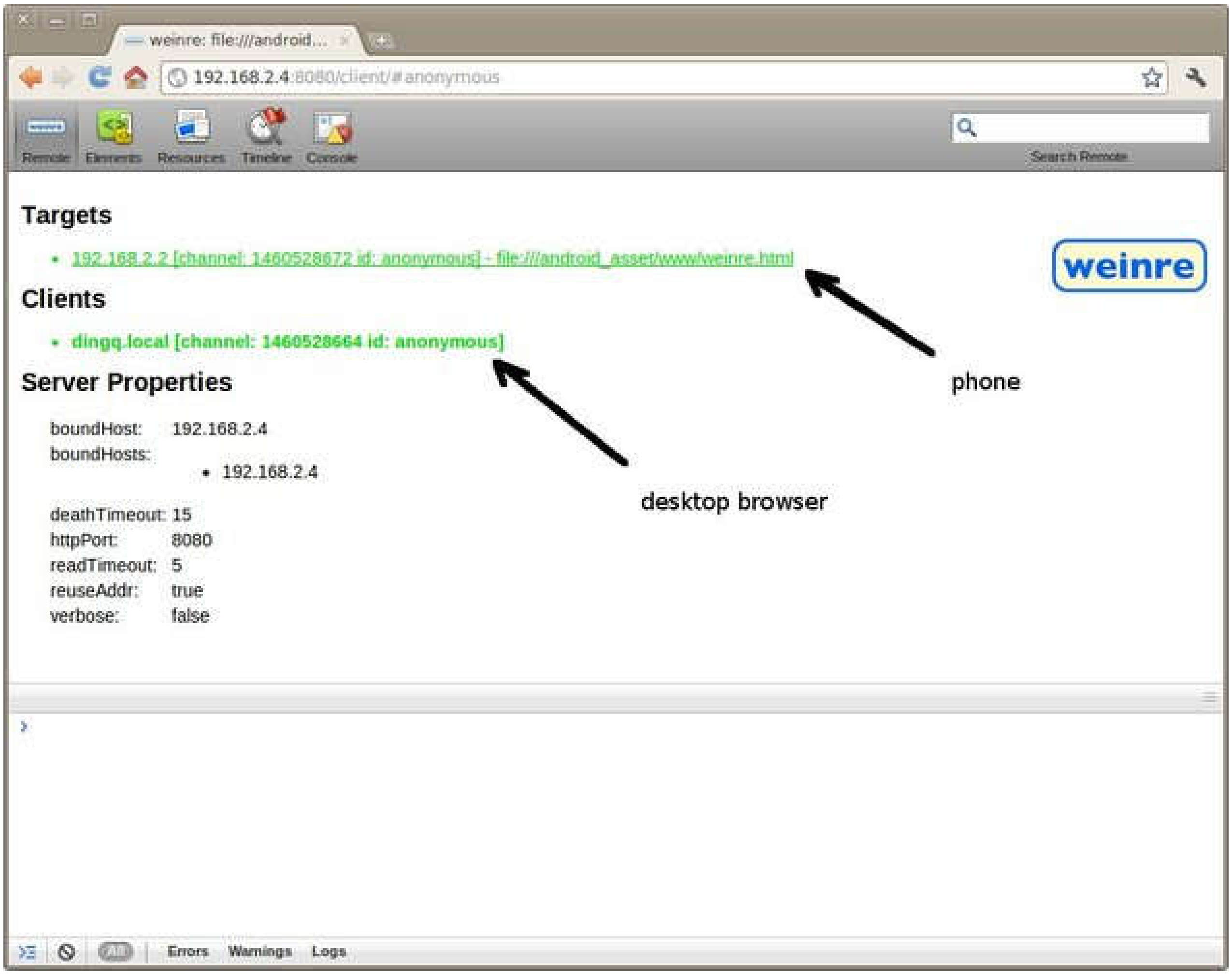
```
    console.log("playAudio");
    console.log("stopAudio");
```

I rebuilt the app in my PhoneGap/Eclipse sandbox, and installed the app on my phone.
After starting the Debug Server and installing the mobile app on my Android phone, as described in the previous sections, I started the mobile app.

A few seconds after starting the app on my phone, the Debug Client on my Chrome browser listed the phone as a new Debug Target. Green means good contact. Under the covers, this means the phone had fetched the WEINRE javascript from the server, executed it, and made good XHR contact back to the server.

Aside: The Debug Client also showed my Chrome Browser (ie, itself) in its list of Debug Clients. This should be helpful information with multiple users.

Figure 2. Connected Target



Targets

- 192.168.2.2 [channel: 1460528672 id: anonymous] - file:///android_asset/www/weinre.html

Clients

- dingq.local [channel: 1460528664 id: anonymous]

Server Properties

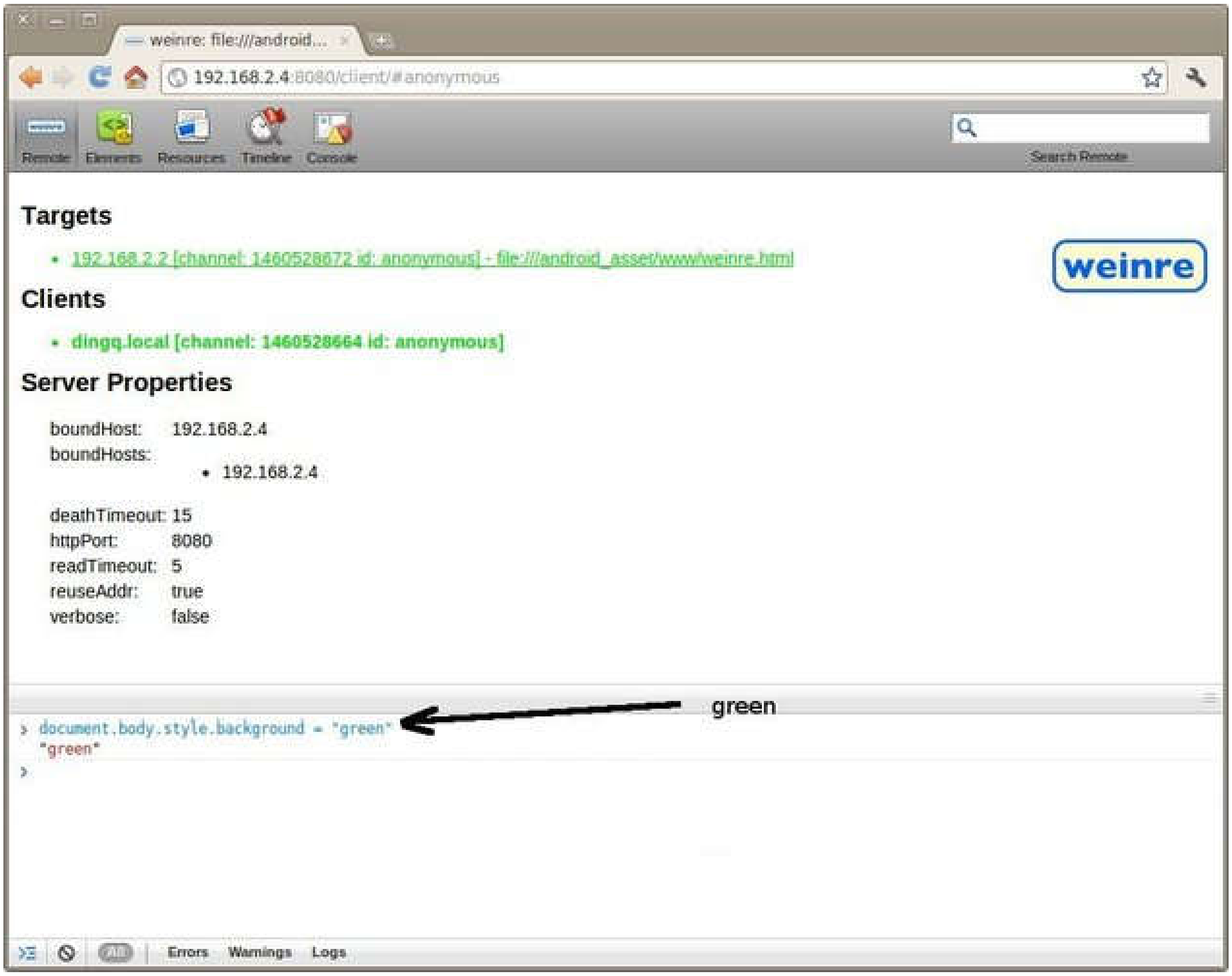
```
boundHost:     192.168.2.4
boundHosts:
                 • 192.168.2.4

deathTimeout: 15
httpPort:      8080
readTimeout:  5
reuseAddr:     true
verbose:       false
```

Changing the DOM

As a simple experiment in modifying the DOM, I followed the examples in the WEINRE instructions and changed the background color on my phone. In the bottom window of the Debug Client, I typed:

```
document.body.style.background = "green"
```

Figure 3. WEINRE Background Green



Targets

- 192.168.2.2 [channel: 1460528672 id: anonymous] - file:///android_asset/www/weinre.html

Clients

- dingq.local [channel: 1460528664 id: anonymous]

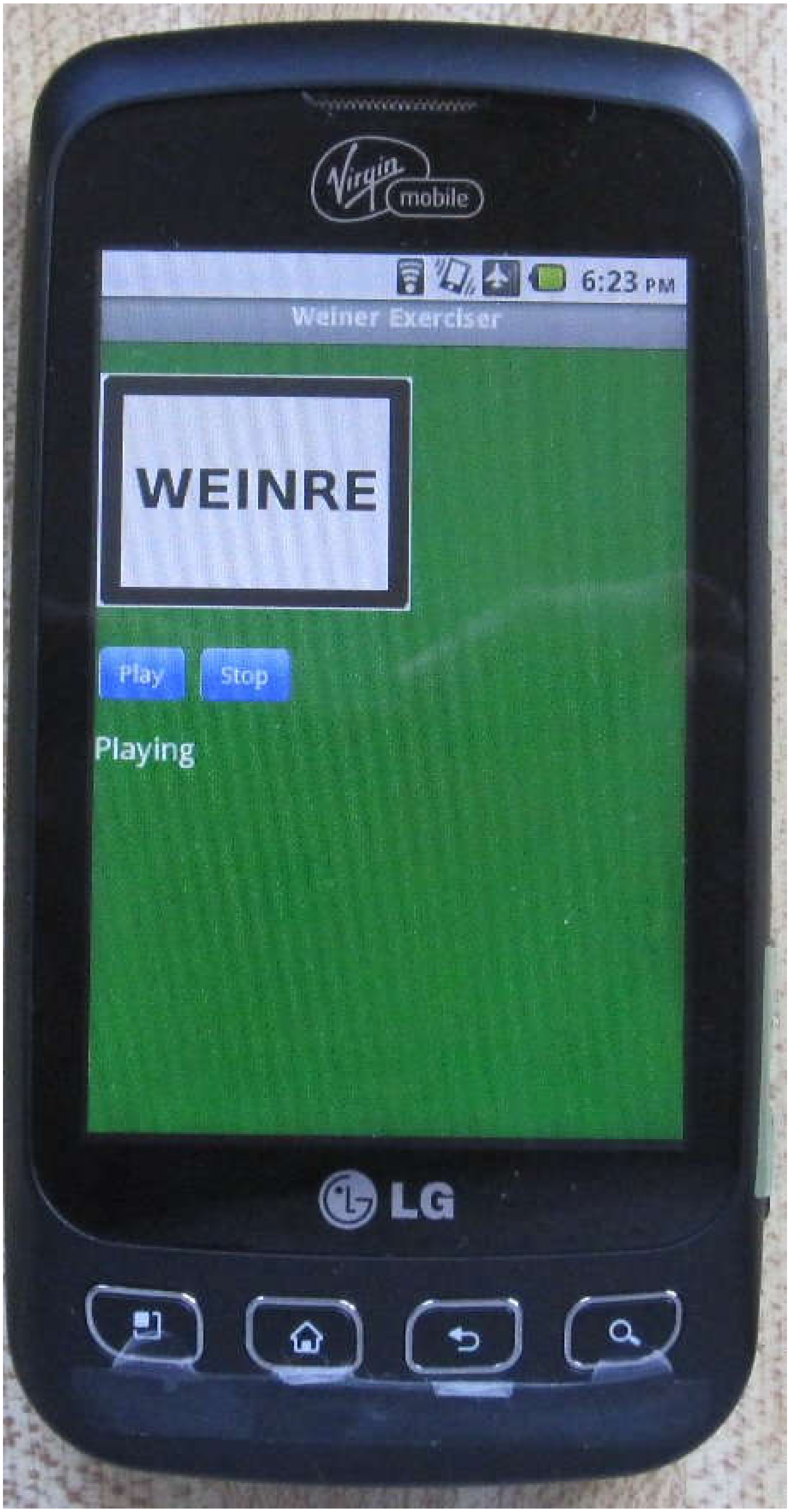
Server Properties

```
boundHost:     192.168.2.4
boundHosts:
                 • 192.168.2.4

deathTimeout: 15
httpPort:      8080
readTimeout:   5
reuseAddr:     true
verbose:       false
```

```
> document.body.style.background = "green"
  "green"
>
```

The background color on my phone immediately changed to green.

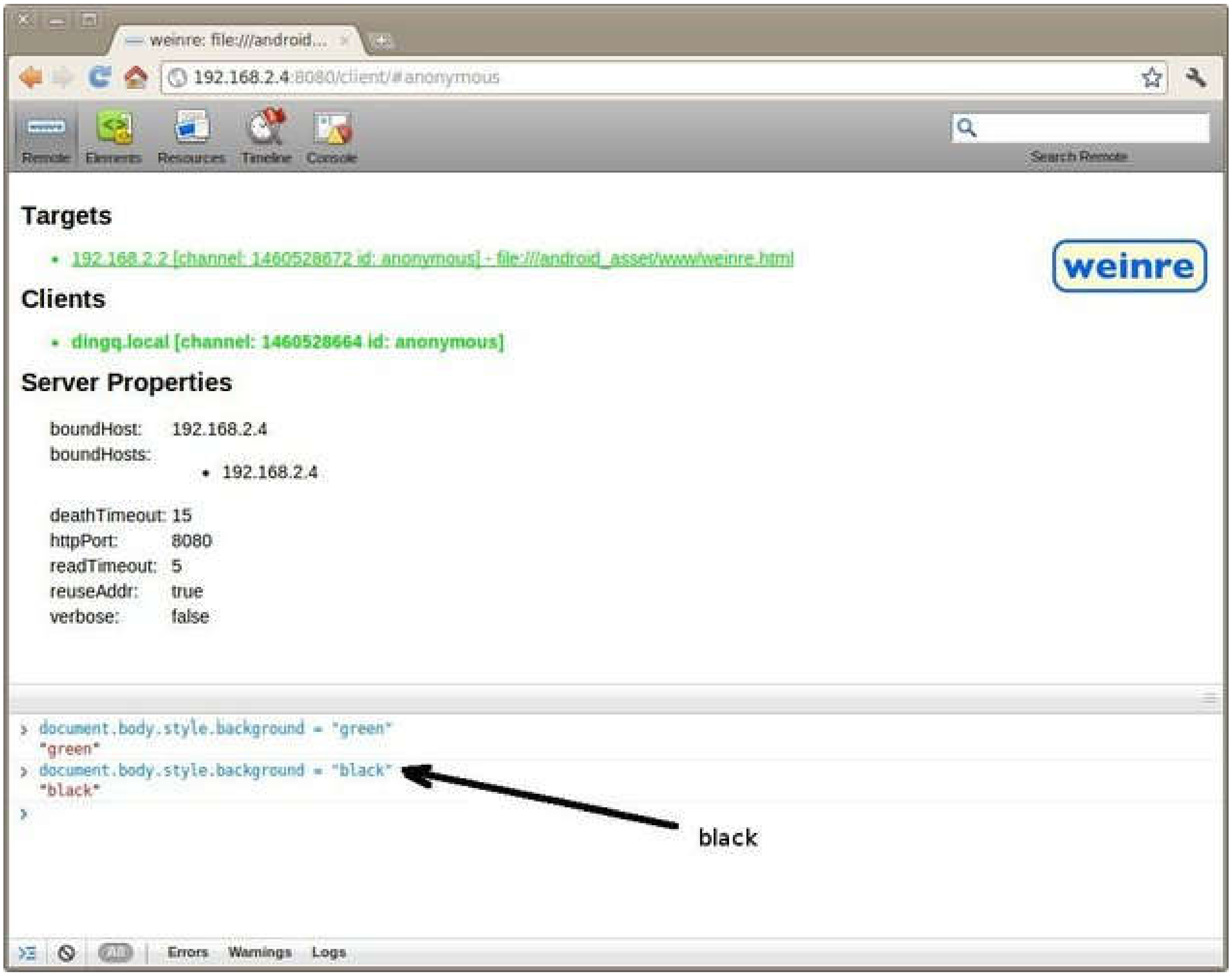
Figure 4. Phone Background Green



Then I set the background back to black:

```
document.body.style.background = "black"
```

Figure 5: WEINRE Background Black



Targets

- 192.168.2.2 [channel: 1460528672 id: anonymous] - file:///android_asset/www/weinre.html

Clients

- dingq.local [channel: 1460528664 id: anonymous]

Server Properties

boundHost: 192.168.2.4
boundHosts:
- 192.168.2.4

deathTimeout: 15
httpPort: 8080
readTimeout: 5
reuseAddr: true
verbose: false

```
> document.body.style.background = "green"
"green"
> document.body.style.background = "black"
"black"
>
```

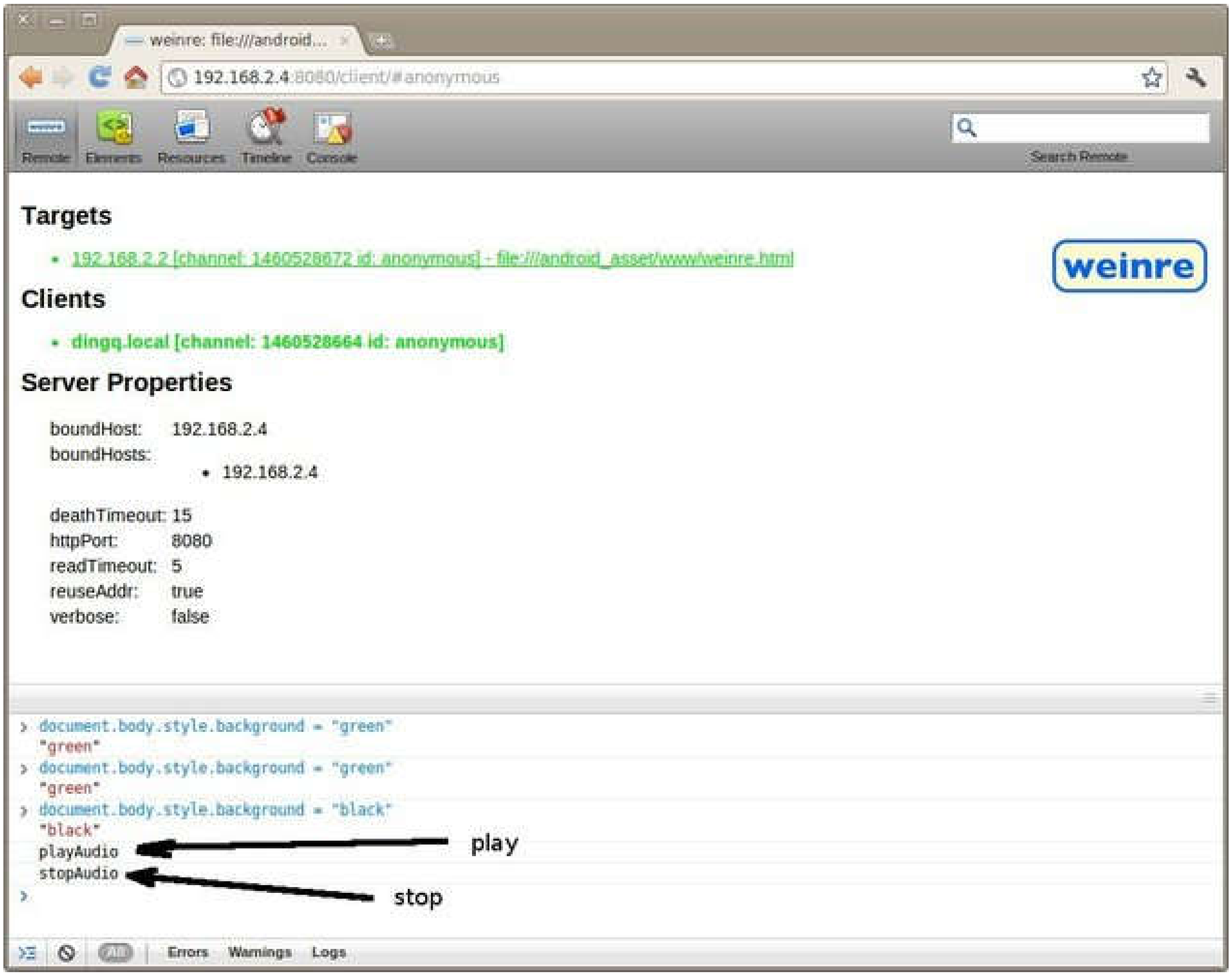
And the phone immediately was restored back to black:

Figure 6: Phone Background Black

Console.log messages

To test console.log messages, I clicked the play and stop buttons. I immediately saw these messages appear in the bottom panel of the Debug Client.

Figure 7: Console.log messages



Targets

- 192.168.2.2 [channel: 1460528672 id: anonymous] - file:///android_asset/www/weinre.html

Clients

- dingq.local [channel: 1460528664 id: anonymous]

Server Properties

boundHost: 192.168.2.4
boundHosts:
- 192.168.2.4

deathTimeout: 15
httpPort: 8080
readTimeout: 5
reuseAddr: true
verbose: false

```
> document.body.style.background = "green"
  "green"
> document.body.style.background = "green"
  "green"
> document.body.style.background = "black"
  "black"
  playAudio
  stopAudio
>
```

Conclusions and Recommendations
WEINRE worked amazingly well for me right out of the box.

For the first time, I can painlessly debug my apps written in HTML and Javascript within the PhoneGap framework. Modifying the DOM and seeing console.log statements are killer debuggers for me. Changing the DOM is useful to test javascript fixes on the fly. Generating console.log messages is useful to follow code flow and look for sequence errors.

As an added benefit, WEINRE also works when testing my apps in an Android simulator. It works just like the phone.

I look forward to trying the other features of WEINRE, and encourage you to give it a try.

Kudos to fellow IBMer Patrick Mueller for developing WEINRE

Attachments

Sample HTML: weinre.sample.2011-0719-1209.html

References

WEINRE home page: http://phonegap.github.com/weinre/Home.html
WEINRE downloads: https://github.com/phonegap/weinre
PhoneGap framework: http://www.phonegap.com
Dojo Toolkit: http://www.dojotoolkit.org