Reply to comment

An Overview of Mobile Debugging Techniques – Part Two

Raymond Camden
01/14/14
4326 views
0 replies

The Mobile Zone is presented by New Relic and Progress Software. New Relic is a performance monitoring tool and has a Data Nerd T-Shirt with your name on it if you sign up. Progress Software allows you to create powerful data-driven mobile and web apps with minimal coding.

Welcome to part two of my series on mobile debugging techniques. In my first article, I discussed how you can use Safari and Chrome remote debugging to help you with your mobile sites. In this article I'll be discussing two other tools that can help you - Adobe Edge Inspect and Weinre.

Adobe Edge Inspect (with a disclaimer)

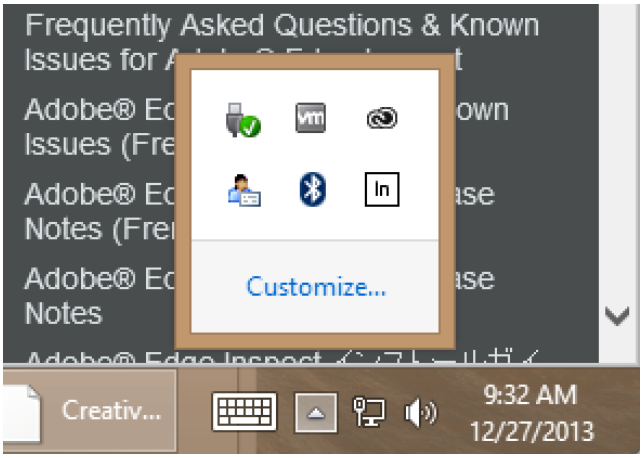
Before I say anything else, let me be sure folks know where I'm coming from. I work for Adobe. I'm talking about one of our products. It has both a free and a paid tier. But to be clear, I work for this company. I also *honestly* think it is a pretty darn cool tool. If you can't get past the fact that I'm talking about something from my own company, I understand. Check out the next section. I promise it is just as interesting and has a funny sounding name.

Edge Inspect gives you synchronized browsing between your desktop browser and your mobile browser. You can open up a URL with Chrome and watch as it loads on your mobile device. As you click around the site, your mobile device will follow along. What makes this even more powerful is that you can set up Edge Inspect on multiple devices. Imagine multiple iPhones running different versions of iOS. Next add a couple of Android devices. Oh, and how about multiple seven and ten inch tablets? It isn't unusual for mobile testing to involve large array of devices. Edge Inspect will work with all of them, at the same time.

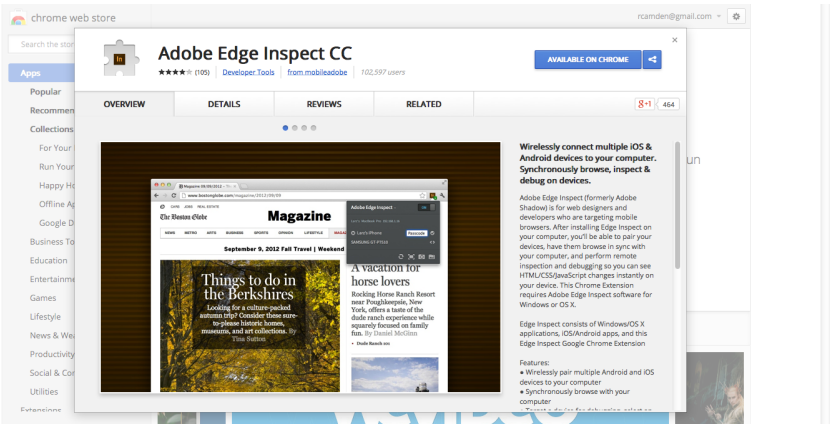
Edge Inspect is part of Adobe's Creative Cloud offering and therefore you have to sign up before you can download it. You do not have to pay for Creative Cloud though. The free tier will let you test out Edge Inspect on one app for thirty days. After you've signed up and logged in, go to the Edge Inspect page and begin the download process. Edge Inspect consists of three parts.

The first is a simple application that you will run on your computer. Download it, install it, and fire it up. In OSX it hangs out in the top bar, out of the way. In the screen shot below you can see the Edge Inspect icon on the left.

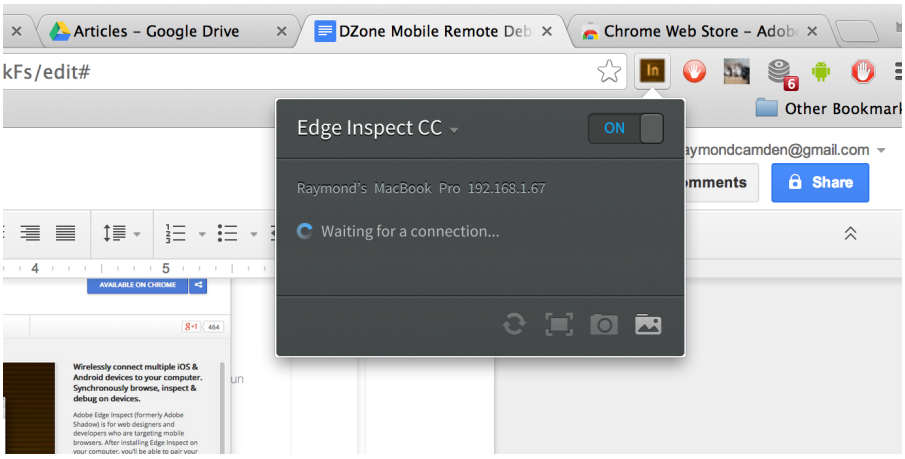
On a Windows machine you can find it in the task bar.



Once running though you can just ignore it. The second item you have to install is a Chrome extension. The extension is linked to from the Edge Inspect download page, but you can also hop over to the Chrome Web Store and just search for Edge Inspect.



Once the Chrome extension is installed you will see an icon in the UI next to the address bar. If you have the desktop app running as well it will be a bright brown color. If you don't have it running it will gray and disabled.



Getting Started with Responsive Web Design Development Techniques

An Overview of Mobile Debugging Techniques – Part One

Dev Tech That Will be HOT in 2014

PhoneGap Architectural Considerations

POPULAR AT DZONE

Why Software Testing Needs Revolution?

A close look at typography in minimalist web design

10 Free HTML Video Conversion Tools

Webix 2.0 Preview: Multi-text Inputs, Extended Localization and Other

Visual Studio 2013 Tips & Tricks – CSS Editor Color Picker

Bootstrap Layout And Wrapper

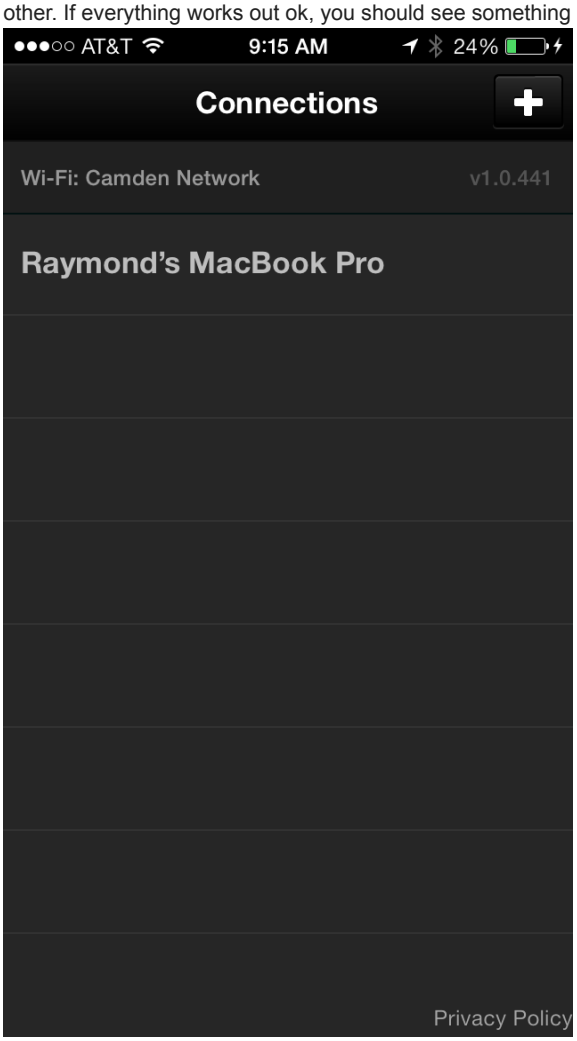
Apache Hadoop Setup

See more popular at DZone

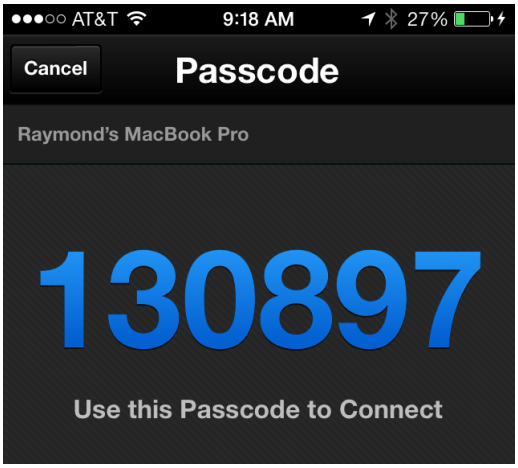
Subscribe to the RSS feed

At this point, Edge Inspect is up and running on your desktop and waiting for connections. The final part of the puzzle is downloading and running the Edge Inspect mobile app. This is currently available for Android, Amazon Kindle devices, and iOS.

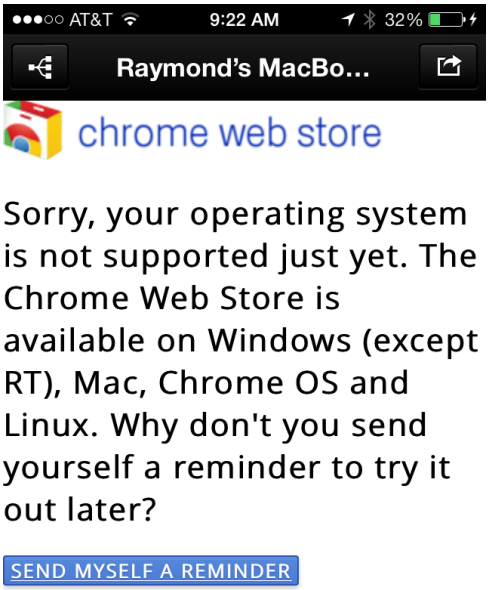
The first time you run Edge Inspect on your mobile device, it will attempt to find local machines running the Edge Inspect desktop application. This is probably the only area where you may run into trouble. On some WiFi networks, devices are not allowed to directly connect to each other. This is fairly common in large companies or at a conference. If you do not see your machine listed in the Edge Inspect app, try switching WiFi connections to see if it helps. In a worst case scenario, I've used personal WiFi devices (or shared WiFi on the mobile device) as a way to get the devices to "see" each other. If everything works out ok, you should see something like this:



Select your machine, and a passcode will show up in the app:

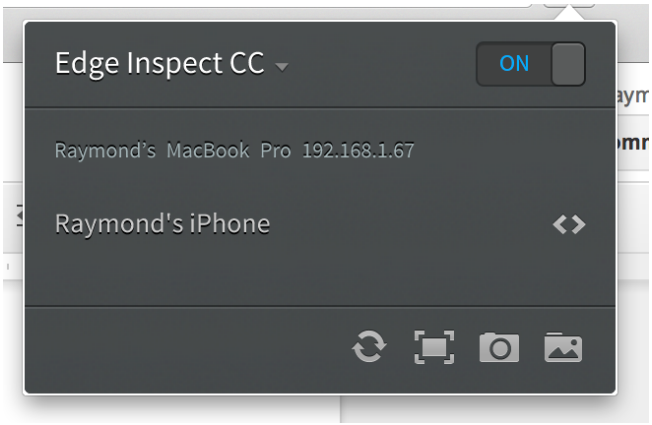


As you can probably guess, this is a security measure to ensure the two sides know each other. You should see a green plus in the Edge Inspect Chrome icon. Click it and enter the passcode. Once you do, the device is now connected to your desktop. You should immediately see the mobile device load whatever page you currently have open in Chrome.



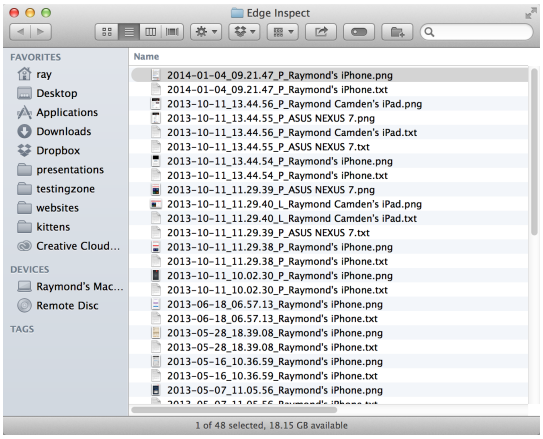
Now for the fun part. You can click around on the desktop and as you do, the device follows along. For the most part this "just works." As you navigate a site the device loads the appropriate pages. And of course, if you had twenty different devices connected they would all follow along with the desktop. One part that will not work are forms. If you enter form values on a page in your desktop and hit submit, the Edge Inspect app will load the second URL (the target of the form) but will not pass the form data you entered.

Edge Inspect also has additional features. Clicking on the Edge Inspect Chrome icon reveals this:



Let's focus on the bottom area first. As you can probably guess, the bottom icon is a refresh action. It will refresh the active page on all devices. If you've just fixed a layout bug and want to see if it helped fixed an issue, you can click this. (Of course, reloading the tab will also have the same effect.) The second icon simply removes the UI around the display on the mobile device – the black bar you saw on top of the earlier screenshot. Clicking it again returns the bar.

The third button is much more interesting. The camera icon will take a screenshot of the current page. It will do so on all devices. These screenshots are then transferred automatically to your desktop. Clicking that fourth icon will open that folder.



Notice that each image is named with a date/time stamp as well as with the name of the device. Each screenshot also includes a text file. Here's an example of what these text files contain:

device_model = iPhone 5

device_res = 1136x640

orientation = portrait

os_name = iOS

os_version = 7.0.4

pixel_density = 326 ppi

request_id = c5cfcfbb-9ac2-4663-81b6-605b799a7b43

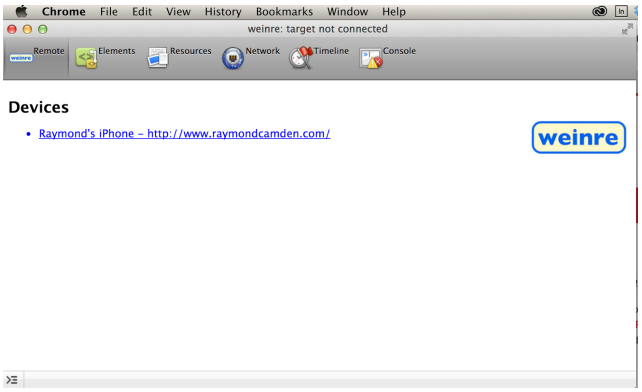
status = Viewport

url = https://chrome.google.com/webstore/detail/adobe-edge-inspect-

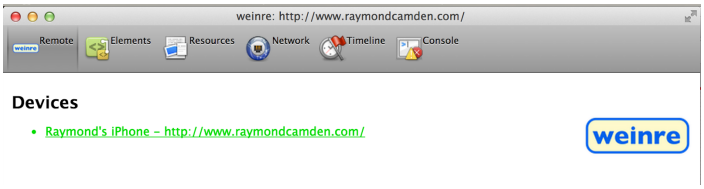
cc/ijoeapleklopieoejahbpdnhkjjgddem

Notice the level of detail. This is incredibly helpful if you are debugging an issue that may involve minor OS differences between the same type of hardware.

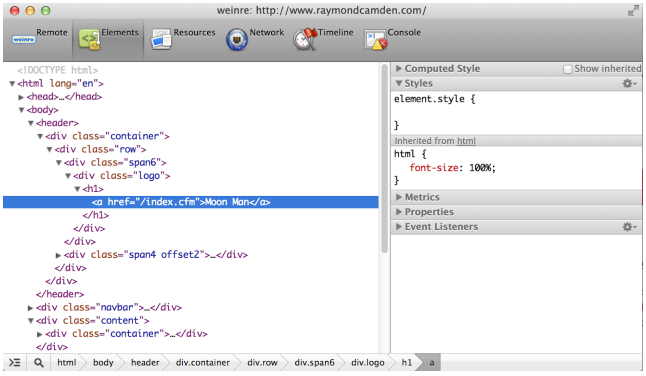
The last feature is my favorite. Notice the brackets (<>) icon? Clicking on this actually opens a Dev Tools panel for the connected device.

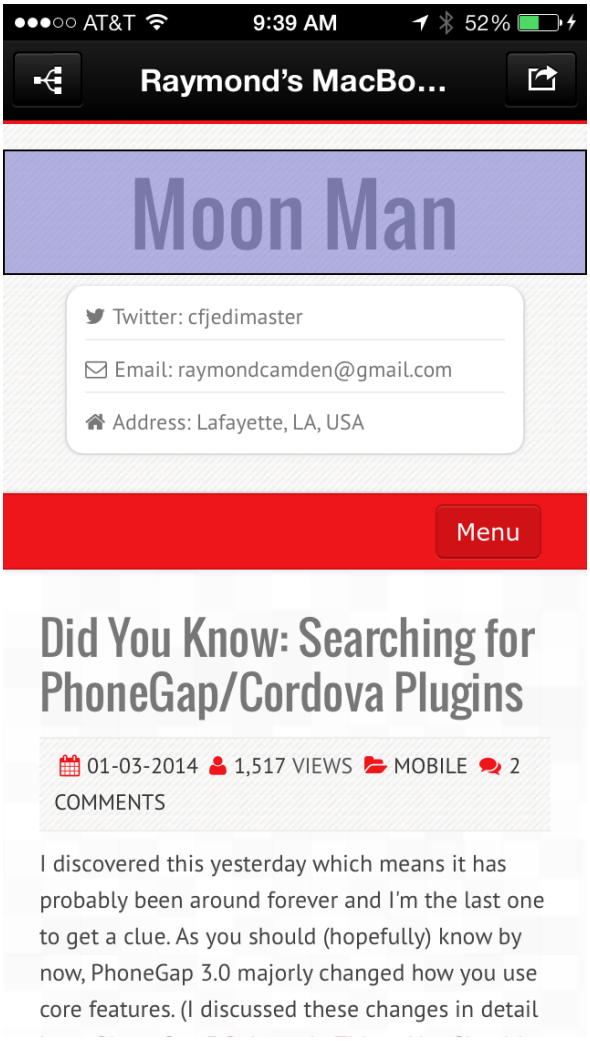


You probably notice the weinre icon on the right. Edge Inspect makes use of this code base and I'll be covering it in the next section. On the initial load, you need to tell the console to talk to the phone. This is done just by clicking it. (You will notice a similar process in the next section.)



Now that it's connected, you can switch over to either the Elements or Console tab. (Resources and Timeline do not work at this time. Network works, but only for XHR requests.) Switching to Elements gives you a live connection much like you saw in the previous article. You can highlight and modify the DOM to your hearts content.





Pretty cool, right? Again, this is fairly similar to what was in the previous article, but keep in mind that you can do this for Android and iOS (and Kindle) devices all from one Chrome browser.

Again - the free tier of Edge Inspect will allow you to test with one device but the paid tier has no limit at all. Now let's take a look at Weinre.

Working with Weinre

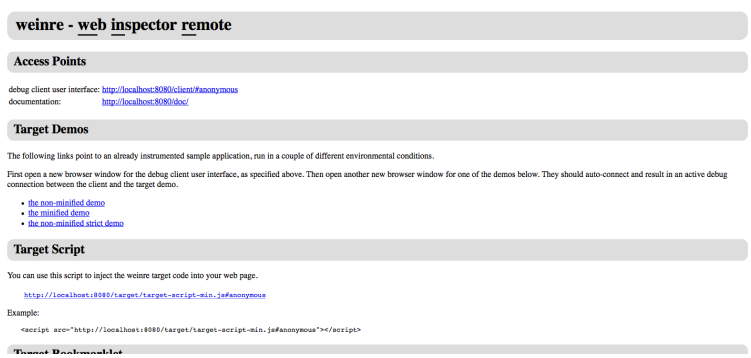
Weinre (pronounced either “winery” or “weiner,” whatever makes you happy) is an open source project created by Patrick Mueller. The home page is: http://people.apache.org/~pmuellr/weinre/docs/latest/Home.html. Weinre allows you to remotely debug browsers. It currently supports iOS Safari, Android browsers, and even PhoneGap/Cordova apps. Let's take a look at how it works.

While you could build Weinre from source, usually people just want to install the most current version. If you've got Node (and npm) installed, this can be done from the command line:

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

(Remember the sudo command only applies to OSX/Linux systems. Those of you on Windows just use npm.) Once installed, you can then simply type weinre at the command line to start it up. Weinre launches a server on your local machine.

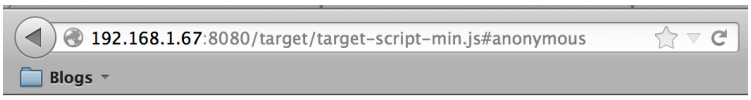
Once up and running, open up that URL in your browser.



Weinre works by providing a JavaScript source that you would add to your HTML pages. If you look at the Target Script area, you can see the script tag that you would copy into your HTML, but notice something – the URL is localhost:8080. If you were to paste in this script block and run it from your mobile device, it would attempt to load the code from localhost, as defined on the device! That's not going to work. Luckily weinre has multiple options for the server. The easiest way to correct this is to use the --boundHost argument. This lets you specify what host to use for the server. While you could specify the IP address of your development machine, the easier solution would be to just use -all- as the argument:

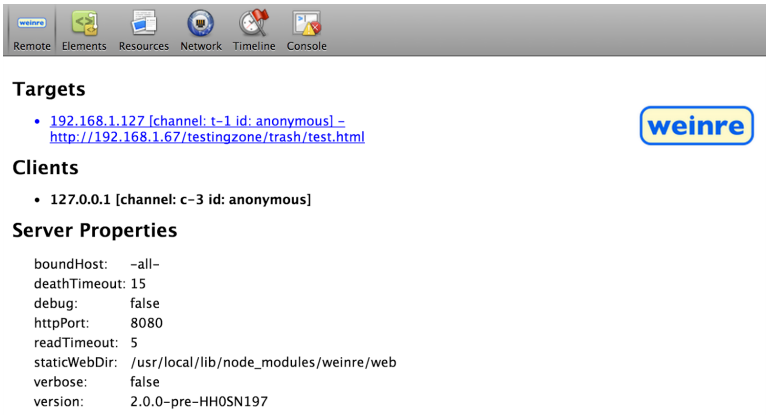
```
1. weinre --boundHost -all-
```

If you view the server page, you may notice that the script src still points to localhost, but if you get your IP address and use that instead of localhost, it will work. I recommend doing that first. Here is what you should see if you open the URL in your browser.




```
/*
 * Licensed to the Apache Software Foundation (ASF) under one
 * or more contributor license agreements.  See the NOTICE file
 * distributed with this work for additional information
 * regarding copyright ownership.  The ASF licenses this file
 * to you under the Apache License, Version 2.0 (the
 * "License"); you may not use this file except in compliance
 * with the License.  You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing,
 * software distributed under the License is distributed on an
 * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
 * KIND, either express or implied.  See the License for the
 * specific language governing permissions and limitations
 * under the License.
 */
```

Place a script tag pointing to that source in your HTML and then load up the page on your mobile device. Back in the Weinre home page, click on the debug client user interface link at the top. This brings up a screen much like you saw earlier in the Edge Inspect section.



In order to "connect", simply click on the target. It isn't terribly obvious, but you know things are working when the blue link turns green. At this point, you've got basically what I showed earlier in the Edge Inspect section. You can work with the DOM (both highlighting and modifying) and enter code in the console. Unlike Edge Inspect, the Resources tab works. It will let you see WebSQL databases, LocalStorage, and SessionStorage. (For Edge Inspect users, keep in mind that you can still see those values if you use the console.)

Like Edge Inspect, you could use Weinre with multiple clients. Just open your HTML page on multiple devices and they will show up as additional targets. Unfortunately there isn't an easy way to tell one client from the other. Their remote IP shows up and you could always check that on the device to see what is what.

Mobile Debugging - It Doesn't Suck!

Ok, so that's a bit tongue and cheek, but if you've been doing mobile web development for any amount of time you know how far we've come in such little time. While it may not be quite as easy as desktop debugging, it is pretty darn close and improving rapidly.

As our web sites turn into applications and grow in complexity, the number one skill (in my opinion) for developers will be debugging. Remember, you will never be able to fix every issue. You will never know everything. But telling the client, or your coworkers, that "it isn't working" will not be sufficient. With the debugging tools we have at our disposal now there is no excuse for that. You can always do some level of inspection to figure out what is going wrong. Fixing that problem may be more difficult of course, but knowing the problem is half the battle!

This article was commissioned by Intel and the Tizen Mobile Web Development team.

Reply

Your name:
Dilip Mistry

Comment: *

- Allowed HTML tags: <a> <i> <b>   <cite> <code>   <dl> <dt> <dd> <h2> <h3> <font> <img> <p> <pre> <table> <tr> <td> <tbody> <thead> <th>

  <blockquote> <strike>
- Images can be added to this post.
- Use to create page breaks.

More information about formatting options

Save Preview