appliness

#1 - APRIL 2012

The digital magazine for web app DEVELOPERS

TUTORIALS

HTML5, PHONEGAP, BACK-BONE.JS, jQUERY MOBILE...

INTERVIEW

MAXIMILIANO FIRTMAN, THE MOBILE WEB GODFATHER

Photography by Frank Deras

This is a PDF preview of Appliness, a digital magazine for web application developers.

Download Appliness on your iPad or your Android tablet to enjoy the best reading experience *(interactive samples, links, videos...)*.

If you want to contribute to appliness writing articles or showcasing your apps, visit our website *(www.appliness.com)* and contact us.

appliness

THE FIRST DIGITAL MAGAZINE FOR WEB APPLICATION DEVELOPERS

Welcome to appliness

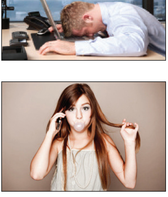
by Michaël Chaize

Welcome to appliness, the first digital magazine for web application developers. Today, developers can use web standards (HTML, JS, and CSS) to build cross-platform applications for desktop, mobile, and tablet devices. The contributors to appliness are all professional developers who have vast experience in developing such applications and are eager to share with you their favorite techniques, the tools they use most, and the libraries and frameworks they like to employ. In addition, the magazine will also highlight some of today's most talented developers. In this issue, you'll discover cool interactive tutorials, videos, articles about market trends, and links to practical information for web developers. We hope that you enjoy reading this magazine, and we encourage you to help us continue to improve it by providing feedback on appliness.com.

TABLE OF CONTENTS

DON'T WORRY, BE APPLI

BACKBONE.JS WINE CELLAR TUTORIAL: GETTING STARTED
Christophe Coenraets details the benefits of using Backbone.js, a lightweight framework to structure your JavaScript code.

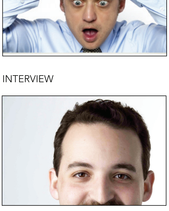


GETTING STARTED WITH JQUERY MOBILE
Discover how to code your first web mobile application with the open-source framework JQuery Mobile.

CANVAS QUIRKS
While using Canvas 2D context for drawing stuff, Mihai Corlan discovered that the drawing line API can surprise you a bit especially when drawing horizontal or vertical lines.

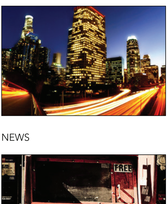
HOW I DEBUG JAVASCRIPT?
Ray Camden describes his methodology to debug Javascript code.

MOBILE WEB & PHONEGAP HTML WEB TIPS
Here are a few tips that Andy Trice has found useful for improving overall interaction and mobile HTML experiences.



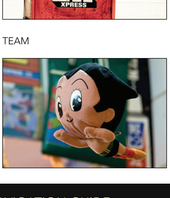
WHAT IS PHONEGAP?
Greg Wilson introduces this open-source framework to build mobile native apps with web standards.

VIDEO



HTML5 FOR DEVELOPERS: PHPSTORM/WEBSTORM
Quick review of PHPStorm and WEBStorm IDEs by Piotr.

BLEEDING EDGE



BLEEDING EDGE HTML5, WEBRTC & DEVICE ACCESS
Andrew plays with the new WebRTC features that should be soon available in our browsers.

SHOWCASE

TOP MOBILE APPLICATIONS
A selection of great mobile applications developed with web standards and PhoneGap.

CODE

WTFJS ?!*$
JavaScript is a language we love despite it giving us so much to hate.

INTERVIEW

MAXIMILIANO FIRTMAN, THE GODFATHER OF THE MOBILE WEB
Interview of the author of "Programming the Mobile Web" (O'Reilly).

TRENDS

WHAT EXACTLY IS APACHE?
Alan Greeblatt details the core values of the Apache Foundation Software and the impact on technologies such as Flex or PhoneGap.

NEWS

HELTER SKELTER NEWS
Brian Rinaldi selects for us the most innovative articles about JavaScript and HTML.

TEAM

WHO'S BEHIND THIS MAGAZINE
Presentation of the contributors and links to great online resources.

NAVIGATION GUIDE

READ

MOVE TO THE NEXT ARTICLE BY A HORIZONTAL SWIPE

READ THROUGH THE ARTICLE BY A VERTICAL SWIPE

NAVIGATE

GO BACK TO THE LIBRARY

MOVE TO THE PREVIOUS ARTICLE

DISPLAY THE TABLE OF CONTENTS

VISUALLY BROWSE ALL THE ARTICLES

Backbone.js Wine Cellar Tutorial - Part 1: Getting Started

by Christophe Coenraets

One of the challenges when building nontrivial Web applications is that JavaScript's non-directive nature can initially lead to a lack of structure in your code, or in other words, a lack of... backbone. JavaScript is often written as a litany of free-hanging and unrelated blocks of code, and it doesn't take long before it becomes hard to make sense of the logic and organization of your own code.

Backbone.js is a lightweight framework that addresses this issue by adding structure to JavaScript-heavy Web applications.

Self-contained building blocks

Backbone.js provides several classes (Model, Collection, View, Router) that you can extend to define the building blocks of your application. To build an app with Backbone.js, you first create the Models, Collections, and Views of your application. You then bring these components to life by defining a "Router" that provides the entry points of your application through a set of (deep-linkable) URLs.

With Backbone.js, your code is organized in self-contained entities (Models, Collections, Views): No more free-hanging and unrelated blocks of code.

Data Binding

With Backbone.js, you bind Views to Models so that when a Model's data changes, all the Views bound to that Model automatically re-render. No more complex UI synchronization code.

Elegant REST Integration

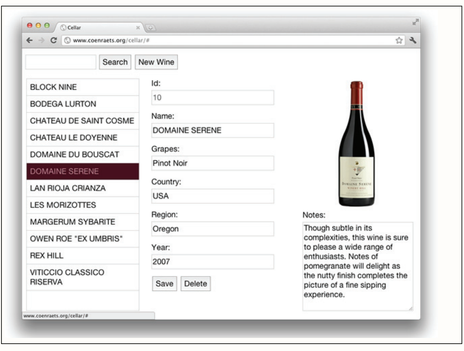
Backbone.js also provides a natural / magical / elegant integration with RESTful services. If your back-end data is exposed through a pure RESTful API, retrieving (GET), creating (POST), updating (PUT), and deleting (DELETE) models is incredibly easy using the Backbone.js simple Model API.

Sample Application

In this three-part tutorial, you'll create a Wine Cellar application. You can browse through a list of wines, as well as add, update, and delete wines.

- In Part 1 (this post), you define the basic infrastructure. You create a "read-only" version of the application: you'll be able to retrieve a list of wine and get the details of each wine.
- In Part 2, you add the code to add, update and delete wines. You leverage Backbone's powerful REST integration.
- In Part 3, you add complete support for history management and deep linking.

NOTE: I also blogged a non-Backbone version of the application here (Java back-end) and here (PHP back-end), which you can look at for comparison.
Here is a screenshot of the final application.



Part 1: The Read-Only Wine Cellar Application

You can run the application (Part 1) here.

Here is the code:

```
1  // Models
2  window.Wine = Backbone.Model.extend();
3
4  window.WineCollection = Backbone.Collection.extend({
5      model:Wine,
6      url:"../api/wines"
7  });
8
9  // Views
10 window.WineListView = Backbone.View.extend({
11
12     tagName:'ul',
13
14     initialize:function () {
15         this.model.bind("reset", this.render, this);
16     },
17
18     render:function (eventName) {
19         _.each(this.model.models, function (wine) {
20             $(this.el).append(new WineListItemView({model:wine}).render().el);
21         }, this);
22         return this;
23     }
24
25 });
26
27 window.WineListItemView = Backbone.View.extend({
28
29     tagName:"li",
30
31     template:_.template($('#tpl-wine-list-item').html()),
32
33     render:function (eventName) {
34         $(this.el).html(this.template(this.model.toJSON()));
35         return this;
36     }
37
38 });
39
40 window.WineView = Backbone.View.extend({
41
42     template:_.template($('#tpl-wine-details').html()),
43
44     render:function (eventName) {
45         $(this.el).html(this.template(this.model.toJSON()));
46         return this;
47     }
48
49 });
50
51 // Router
52 var AppRouter = Backbone.Router.extend({
53
54     routes:{
55         "":"list",
56         "wines/:id":"wineDetails"
57     },
58
59     list:function () {
60         this.wineList = new WineCollection();
61         this.wineListView = new WineListView({model:this.wineList});
62         this.wineList.fetch();
63         $('#sidebar').html(this.wineListView.render().el);
64     },
65
66     wineDetails:function (id) {
67         this.wine = this.wineList.get(id);
68         this.wineView = new WineView({model:this.wine});
69         $('#content').html(this.wineView.render().el);
70     }
71 });
72
73 var app = new AppRouter();
74 Backbone.history.start();
```

Code highlights:

1. **WineModel** (line 2): Notice that we don't need to explicitly define the attributes (name, country, year, etc). You could add validation, default values, etc. More on that in Part 2.
2. **WineCollection** (lines 4 to 7): "model" indicates the nature of the collection. "url" provides the endpoint for the RESTful API. This is all that's needed to retrieve, create, update, and delete wines with Backbone's simple Model API.
3. **WineListView** (lines 10 to 25): The render() function iterates through the collection, instantiates a WineListItemView for each wine in the collection, and adds it to the wineList.
4. **WineListItemView** (lines 27 to 38): The render() function merges the model data into the "wine-list-item" template (defined in index.html). By defining a separate View for list items, you will make it easy to update (re-render) a specific list item when the backing model changes without re-rendering the entire list. More on that in Part 2.
5. **WineView** (lines 40 to 49): The view responsible for displaying the wine details in the Wine form. The render() function merges the model data (a specific wine) into the "wine-details" template retrieved from index.html.
6. **AppRouter** (lines 52 to 71): Provides the entry points for the application through a set of (deep-linkable) URLs. Two routes are defined: The default route ("") displays the list of wine. The "wines/:id" route displays the details of a specific wine in the wine form. Note that in Part 1, this route is not deep-linkable. You have to start the application with the default route and then select a specific wine. In Part 3, you will make sure you can deep-link to a specific wine.

Download

The source code for this application is hosted on GitHub here. And here is a quick link to the download.

You will need the RESTful services to run this application. A PHP version (using the Slim framework) is available as part of the download.

UPDATE (1/11/2012): A version of this application with a Java back-end (using JAX-RS and Jersey) is also available on GitHub here. You can find more information on the Java version of this application here.

Part 2 is available on Christophe's blog here.

ABOUT THIS ARTICLE

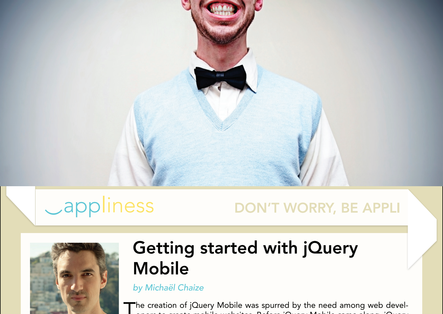
Christophe Coenraets is a Technical Evangelist for Adobe where he focuses on Mobile and Rich Internet Applications for the Enterprise. In his previous role at Macromedia, Christophe worked on JRun, the company's J2EE application server.
http://coenraets.org/
@ccoenraets

ONLINE RESOURCES

Backbone.js official website
http://documentcloud.github.com/backbone/

Wine Cellar application source files
https://github.com/ccoenraets/backbone-cellar

Slim framework
http://www.slimframework.com/



Getting started with jQuery Mobile

by Michaël Chaize

The creation of jQuery Mobile was spurred by the need among web developers to create mobile websites. Before jQuery Mobile came along, jQuery was already a widely-used JavaScript library that simplified the development of engaging experiences on the web with HTML. I believe that jQuery Mobile has a great future because it targets multiple mobile operating systems, including iOS, Android, BlackBerry, and Bada, and because it's very easy to learn.

In this article, I'll show you how to build a simple mobile web application using jQuery Mobile, and then I'll explain how to convert this mobile application into a native application using PhoneGap.

STEP 1 - Include references

The image you see below of a phone on the screen is not a screenshot. You can touch the screen of this phone and play with the application. As you can see, on the first screen, there is a custom header with the appliness logo, a picture I took in Seoul, a list, and a footer. The list is fed by an external XML file. If you tap an item in the list, you'll see the second screen of the app (page 2). A back button in the header lets you go back to the home page.

To get started building this web application, you'll need to import two JavaScript libraries and a CSS file in your HTML page. jQuery.com hosts these files so you can just add these references in the HEAD section of your HTML page:

```
<link rel="stylesheet" href="http://code.jquery.com/mobile/1.0.1/jquery.mobile-1.0.1.min.css" />
<script src="http://code.jquery.com/jquery-1.6.4.min.js"></script>
<script src="http://code.jquery.com/mobile/1.0.1/jquery.mobile-1.0.1.min.js"></script>
```

These references are links to the CDN versions hosted by jQuery. You can also choose to download the files from jQueryMobile.com and use them locally. If you take this approach, be sure to check jQueryMobile.com regularly to make sure you are using the most up-to-date versions of these files.

STEP 2 - Set up the structure of a jQuery Mobile page

Jquery Mobile expects you to declare the sections of your pages (or screens) using the <DIV> tag and the data-role attribute. You may be wondering why the new <HEADER> and <FOOTER> tags introduced with HTML5 are not used. It's simply to ensure compatibility with older smartphones. Basically, you start by declaring a DIV tag that will represent the page: . Inside a page, you can declare a header ( ), the div tag that will host the content of a page ( ), and a footer. I'm not a fan of the word "page" in this context, because in the end, you're really creating an application with screens not pages. Here is the HTML code of application shown on the mobile device to the side:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<link rel="stylesheet" href="http://code.jquery.com/mobile/1.0/jquery.mobile-1.0.min.css" />
<link rel="stylesheet" type="text/css" href="./css/site.css">
<script src="http://code.jquery.com/jquery-1.6.4.min.js"></script>
<script src="http://code.jquery.com/mobile/1.0/jquery.mobile-1.0.min.js"></script>

<meta name="viewport" content="width=device-width; user-scalable=no" />
<title>JQM tutorial</title>
</head>

<body>
<div data-role="page" class="ui-page">

    <div data-role="header" class="ui-header">
        <h1> My title </h1>
    </div>

    <div data-role="content">
        <div class="image-win">
        <p><img src="assets/astro.jpg"/><br/>
        <b2>Welcome to appliness</b2>
        </p>
    </div>

      <div class="list-questions">
        <ul data-role="listview" id="listQuestions"></ul>
      </div>
    </div>
    <div data-role="footer">
        <h3>Thanks - appliness.com<h3/>
        </div>
</div>

</body>
</html>
```

STEP 3 - Add a custom header

By default, jQuery Mobile uses attractive styles. You can, of course, change the appearance of the application and create your own unique experiences. In this section, you'll see how to create a custom header that displays the appliness logo on a custom background image.

Unless you change it, the header DIV tag uses the ui-header styles defined in the default jQuery Mobile CSS file. To use a custom header, you can simply replace the title of the page with an <img> tag. You can also extend the properties of the default ui-header style by creating your own CSS file (site.css in this example). Here is the code used to create the custom header:

In the main HTML file, reference a custom css file and decare the image:

```
<head>
...
<link rel="stylesheet" type="text/css" href="./css/site.css">
...
</head>
<body>
...
<div data-role="page" class="ui-page">

    <div data-role="header" class="ui-header">
        <div class="image-header"><img src="./assets/logo.png" /></div>
    </div>
...
```

In your CSS file, extend the styles of the default ui-header class.

```
.ui-header{
    background-image:url(../assets/header-bg.png);
    background-position:0 0px;
    background-repeat:repeat;
    height:40px;
}
```

STEP 4 - Get some data and feed the list

Now you're ready to use jQuery to launch an asynchronous HTTP request and consume some XML data. In this example, I'm using a local XML file, but the code would be the same if you were using a hosted PHP script that output some XML—you'd just need to update the URL.

When the document is ready (that is, when your application is loaded), you can use JavaScript code to access the XML file and consume the result. In the code below, notice the HTML list component, <ul>, with the id listQuestions. Thanks to jQuery, you can easily reference this list (or any other) in the DOM of the page and append items as you are parsing the content of the XML file. At the end of the loop, you just need to call the refresh method on the list to display the result. Also, jQuery Mobile will automatically apply a theme to your list. If you add the attribute data-filter="true" to your <ul> element, then jQuery will add a Filter text input field and some logic to filter your results.

This is the HTML code to declare the list with the filtering option:

```
<div class="list-questions">
      <ul data-role="listview"  data-filter="true" id="listQuestions"></ul>
</div>
```

And here is the JavaScript code to get the XLM data and feed the list:

```
<script type="text/javascript">

$(document).ready(function()
{
    $.ajax({
      type: "GET",
      url: "data/questions.xml",
      dataType: "xml",
      success: function(xml){

          $(xml).find("question").each(function()
          {
            var titleQuestion = $(this).find('title').text();
            $('#listQuestions').append("<li><a href="page2.html">"+titleQuestion+"</a></li>");
            console.log(titleQuestion);
          });

            $('#listQuestions').listview("refresh");

      }
    });
});

</script>
```

STEP 5 - Load a new page and add navigation

You can declare several pages in a single jQuery Mobile document (one HTML file). I don't care for this approach and in the sample application I've created a separate HTML page named page2.html. To load this page, just add a link using the classic <a href=""> tag. Using AJAX, jQuery Mobile will load the page in a single-page model. It will add your page to the DOM, initialize all the parts ("widget") of this new page, and launch a transition to it. On the new page, to display a back button (which is a mandatory UI design pattern in an iOS mobile application), just add an anchor/link in the header of your jQuery Mobile page and add the attribute data-rel="back". You can also add data-direction="reverse" to reverse the transition to the previous page. Here is the code of the second page, page2.html:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>Page 2</title>
<link rel="stylesheet" type="text/css" href="./css/site.css">
</head>

<body class="ui-mobile-viewport">

<div id="cameraPage" data-role="page">

    <div data-role="header" class="ui-header">
    <a href="index.html" data-icon="back" data-direction="reverse">Back</a>
        <h1>Page 2</h1>
    </div>
    <div class="section-image">
            <img src="assets/camera.jpg"/>
            <br/>
            Welcome to page 2 !!!
    </div>
</div>
</body>
</html>
```

STEP 6 - Convert from a web app to a native app

Using PhoneGap, a free and open source technology, you can now take this mobile web application and transform it as a native application targeting seven mobile platforms, including iOS, Android, Windows Mobile, and BlackBerry. If you don't want to install the PhoneGap tools on your computer, you can try the (currently free) cloud service at build.phonegap.com, where you can upload your zipped jQuery mobile project and have this hosted service automatically generate your native mobile applications. Later, you can explore the PhoneGap APIs and extend the capabilities of your application with geolocation, the vibration API, the camera, and so on. Together, jQuery Mobile and PhoneGap are a fantastic combination for building cross-platform mobile applications. These are very easy-to-learn and fun frameworks. I encourage you to download the source code for this tutorial and see how easy it is to start building your own amazing mobile apps using web standards.

GET THE SOURCE CODE
http://appliness.com/code/01.zip

ABOUT THIS ARTICLE

Michaël Chaize is a Developer Evangelist at Adobe where he focuses on Rich Internet Application and Mobile applications. Based in Paris, he works with large accounts that need to understand the benefits of rich user interfaces. He's the editor in chief of Appliness.
http://riagora.com/
@mchaize

ONLINE RESOURCES

jQuery Mobile official website
http://jquerymobile.com/

PhoneGap official website
http://www.phonegap.com

Using jQuery mobile themes
http://www.adobe.com/fr/devnet/dreamweaver/articles/themes-control-jquery-mobile.html

How I debug JavaScript

by Raymond Camden

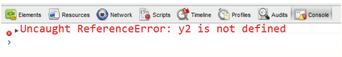
After helping a friend earlier this week with a JavaScript issue, I thought I'd quickly write up the normal ways I attack issues with JavaScript. This is not meant to be a definitive guide per se, but just how I go about dealing with problems. I'll be talking about Chrome, but pretty much everything mentioned below is doable in other browsers (some even in IE).

Check for Errors in the Console

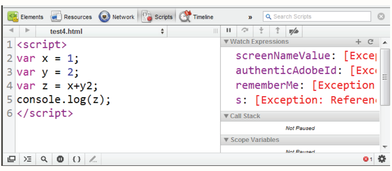
First and foremost, I check the console. Normally errors are not shown to users. Consider the follow example:

```
<script>
 var x = 1;
 var y = 2;
 var z = x+y2;
</script>
```

If I open this up in my browser, I'll see nothing wrong. Obviously if the page tried to display Z, I'd see nothing or something unexpected. But there isn't a large error alert, flashing lights, or klaxons. But as soon as I open the console I see:



You can even click the line and see the context - which would be real useful in a large file.



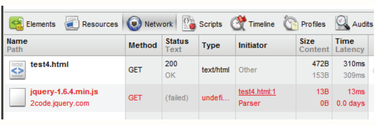
What isn't obvious in this screen shot is that Chrome will also highlight the line. Not sure why they fade it out after a few seconds, but it does make it even more obvious.

Check for Errors in the Network Panel

The other big one is the network panel. This shows all network activity and is useful in two ways. One error I see all the time (and this was the error my friend had), was simply not noticing that a library didn't load. This could be because of a web server permission issue or simply forgetting to upload a file! Consider:

```
<script src="http://2code.jquery.com/jquery-1.6.4.min.js"></script>
<script>
$(document).ready(function() {
    console.log("moo");
});
</script>
```

Notice the bad URL for the jQuery load? In the Network tab, this shows up right away (note, it also creates an error in the console since I tried to use it, but if I was not using jQuery right away, this wouldn't be obvious):



Notice that the Network panel let's you filter. That's recommended if you have a lot of stuff going on, but, you probably want to check everything first. For example, you may be using jQuery UI. If you forget to upload the CSS, things will definitely be wonky, but that wouldn't show up as a missing JavaScript library.

The other big thing to look for here is in the XHR requests area. XHR requests represent your Ajax requests. A full look into this is a bit much for this blog entry, but basically, if you are making an Ajax request for data, you want to check your network request to see what the server returned. Did the server throw an error? Did it return data in a way you didn't expect? Did your result include formatting? For example, many people will wrap all their requests in a template. While this works great for the rest of the site, including a template around your JSON data will break any code that tries to parse it. Don't forget to look at the complete response. You may have whitespace after your JSON that is a bit off screen. You can also right click on an XHR request and open it in a new window to make it clearer.

Finally - one more thing you want to pay attention to is size. If your application is working, but working slowly, look at how big those XHR requests are. Ajax isn't magic. If you return 100K of JSON data it's still going to take time for that data to transfer to the client.

Event Handlers/Selector

This is mainly jQuery based, but could apply anywhere. Another thing I check is my event handlers and jQuery selectors. For example, consider:

```
$("#mything").on("click", function() { });
```

If I notice that my click handler isn't running, I check to ensure that #mything is actually matching something in the DOM. In my startup routine I may do something like so:

```
console.dir($("#mything"));
```

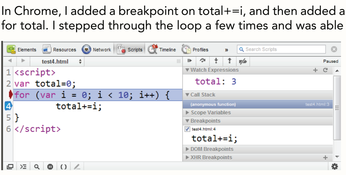
If I see that jQuery couldn't find it, then I look into my DOM and see what's up. It could be as simple as a typo.

The Debugger

And finally, if all all fails, I try out the debugger. Again, this is a bit too much to cover in detail in this blog post, and, I'm very new to this myself, but the debugger let's you set breakpoints and step through your code. If that doesn't quite make sense to you, consider this - you can pause your JavaScript application and freeze it in time. You can look at variables and their state. You can then have your application proceed one line at a time. It's like God Mode for the browser. Consider:

```
<script>
var total=0;
for (var i = 0; i < 10; i++) {
    total+=i;
}
</script>
```

In Chrome, I added a breakpoint on total+=i, and then added a watch expression (i.e., 'watch this variable') for total. I stepped through the loop a few times and was able to watch the variable increase:



ABOUT THIS ARTICLE

Meet Raymond Camden. He is a 38 year old married father of three living in beautiful Lafayette, Louisiana. Ray is a developer evangelist for Adobe where his primary technical focus is ColdFusion, jQuery, Flex, AIR and the mobile space.

http://raymondcamden.com/
@cfjedimaster

ONLINE RESOURCES

Chrome Extensions - Debugging tutorial
http://code.google.com/chrome/extensions/tut_debugging.html

Firebug for Firefox
http://getfirebug.com/

HTML5 development
http://www.adobe.com/devnet/html5.html

Canvas quirks

by Mihai Corlan

While using Canvas 2D context for drawing stuff, I discovered that the drawing line API can surprise you a bit, especially when drawing horizontal or vertical lines. Here is a live preview on your tablet with a Canvas element and 5 lines drawn using lineTo() calls. Click on the HTML button:

In case you haven't noticed, let me tell you what's wrong with this: the lines are suposed to be 1 pixel width and black. Clearly what you see on the screen is not 1 pixel and the lines are somehow grayish. It looks more like 2 pixels. The code for drawing this looks like this:

```
<input onclick="draw()" type="button" value="draw" />

<script type="text/javascript">
function draw() {
    var context, i, y;

    context = document.getElementById('canvas').getContext('2d');
    y = 20;
    context.lineWidth = 1;
    context.strokeStyle = '#000000';
    for (i = 0; i < 5; i++) {
       context.moveTo(0, y);
       context.lineTo(450, y);
       y += 10;
    }
    context.stroke();
}
</script>
```

Let's change the line width to 2 - `context.lineWidth = 2` - and check the result:

Interesting, isn't it? So the lines width is basically the same, but the color now is really black. Now, let's try something else: change the line width back to 1 and adjust the y property of the moveTo/lineTo functions with 0.5 (line 13/14):

```
context.moveTo(0, y + 0.5);
context.lineTo(450, y + 0.5);
```

And surprise, the lines are now exactly 1 pixel and black:

So what's happening? After some research I think that this is what is happening:

- When you use integer coordinates, like 10 or 15, the drawing algorithm is actually trying to draw a line in between two pixels (for example between the 9th and 10th pixels). As a result, it will actually draw two lines.
- I think the line is slightly lighter than the color set because of the antialiasing algorithm.
- When you offset the coordinates by 0.5 then you "end" up with drawing the line exactly on one pixel.
- If you draw a 1 pixel vertical line from (0,0) to (0,200) you will see that this time the line is exactly one pixel wide but the issue of lighter than defined color remains. As there is no other pixel to the left of the 0 pixel on the X axis on the screen you will see only one line.

Using fillRect() function instead of lineTo()

If you don't like adding those 0.5 to any coordinate when using the lineTo() API then you can actually use the drawing rectangle API. As you probably already guessed, the trick is to draw a rectangle of one pixel for one dimension and the length you need for the other one. So here is the script for drawing 5 horizontal lines:

```
function draw() {
    var context, i, y;

    context = document.getElementById('canvas').getContext('2d');
    y = 20;

    for (i = 0; i < 5; i++) {
       context.fillRect(0, 10 + y, 450, 1);
       y += 10;
    }
}
```

And here is the result:

If you are wondering about performance differences between lineTo() and fillRect() then you shouldn't. fillRect() is probably even faster than lineTo().

GET THE SOURCE CODE
http://appliness.com/code/01.zip

ABOUT THIS ARTICLE

Mihai Corlan has been working for Adobe since 2006. Since 2008 he has been working as a developer evangelist for Adobe. His current role is worldwide web developer evangelist. This means he writes code, he writes articles, and he speaks.
http://corlan.org/
@mcorlan

ONLINE RESOURCES

Mihai's blog
http://www.corlan.org

HTML5 and canvas
http://www.html5canvastutorials.com/

Learn to use HTML5 canvas
http://www.adobe.com/devnet/html5/html5-canvas.html

Mobile Web & PhoneGap Dev Tips

by Andy Trice

Recently I've been spending a fair amount of time working on HTML-based applications – both mobile web and mobile applications using PhoneGap. Regardless of whether you are targeting a mobile web browser or a mobile app using the PhoneGap container, you are still targeting a mobile web browser instance. If you haven't noticed, mobile web browsers can often have peculiarities with how content is rendered, or how you interact with that content. This happens regardless of platform – iOS, Android, BlackBerry, etc… All have quirks. Here are a few tips that I have found useful for improving overall interaction and mobile HTML experiences.

Disclaimer: I've been targeting iOS and Android primarily, with BlackBerry support on some applications. I don't have a Windows Phone device to test with, so I can't comment on support for the Windows platform.

Autocorrect and AutoCapitalize

First things first: autocorrect and autocapitalize on Apple's iOS can sometimes drive you to the brink of insanity. This is especially the case if you have a text input where you are typing in a username, and it keeps "correcting" it for you (next thing you know, you are locked out of the app). You can disable these features in web experiences by setting the "autocorrect" and "autocapitalize" attributes of an <input> instance.

Disabled AutoCorrect

```
<input type="text" autocorrect="off"
autocapitalize="on">
```

Disabled AutoCapitalize

```
<input type="text" autocorrect="on"
autocapitalize="off">
```

Managing the Keyboard

Have you ever experienced an app or web site on a mobile device where you have to enter numeric data, and the default keyboard pops up. Before entering any text, you have you switch to the numeric input. Repeat that for 100 form inputs, and try to tell me that you aren't frustrated… Luckily, you can manage the keyboard in mobile HTML experiences very easily using HTML5 Form elements.

Default Keyboard:

```
<input style="width: 400px;" type="text" value="default">
```

Numeric Keyboard:

```
<input style="width: 400px;" type="number" value="numeric">
```

Numeric Keyboard:

```
<input style="width: 400px;" type="text" pattern="[0-9]*"
value="numeric">
```

Phone Keyboard:

```
<input style="width: 400px;" type="tel" value="telephone">
```

URL Keyboard:

```
<input style="width: 400px;" type="url" value="url">
```

Email Keyboard:

```
<input style="width: 400px;" type="email" value="email">
```

Disable User Selection

One way to easily determine that an application is really HTML is that everything on the UI is selectable and can be copied/pasted – Every single piece of text, every image, every link, etc… Not only is this annoying in some scenarios (and very useful in others), but there may be instances where you explicitly don't want the user to be able to easily copy/paste content. You can disable user selection by applying the following CSS styles. Note: This works on iOS, and partially works on BlackBerry/QNX for the PlayBook. It did not work on Android in my testing.

```
<style>
* {
-webkit-touch-callout: none;
-webkit-user-select: none;
}
</style>
```

The -webkit-touch-callout css rule disables the callout, and the -webkit-user-select rule disables the ability to select content within an element. More details on webkit css rules from the Mobile Safari CSS Reference. More detail about disabling copy/paste on iOS is available at StackOverflow.com.

Disable Zoom

If you want your content to feel like an app instead of a web page, then I strongly suggest that you disable gestures for pinch/zoom and panning for all use cases where pinch/zoom is not required. The easiest way to do this is to set the viewport size to device-width and and disable user scaling through the HTML metadata tag.

```
<meta name="viewport" content="width=device-width, user-scalable=no"/>
```

You can read further detail on the viewport metadata tag from the Apple Safari HTML Reference, or the Mozilla reference.

On a Phone? Integrate With It

Your application can dial phone numbers very easily. Just use a standard web location, but use the "tel:<phonenumber>" URI format. Test it with Apple Customer Support: 800-275-2273

```
<a href="tel:800-275-2273">800-275-2273</a/>
```

This technique works on both Android and iOS devices, and I assume other platforms. However, I don't have the devices to test all of them.

Touch Based Scrolling

Touch-based scrolling is critical to having an application that feels native. I dont mean that the whole page should be able to scroll… Your browser will be able to take care of that alone. Instead I mean that you should be able to scroll individual elements so that they mimic clipped views, lists, or large blocks of content. You should be able to scroll content where it is, and not have to scroll an entire page to reveal something in only one area of the screen. You should minimize scrolling when it may cause poor UX scenarios. This is especially the case in tablet-based applications which have a larger UI than phone-based applications.

Luckily, this is also really easy. I personally prefer the open source iScroll JavaScript library from cubiq.org. iScroll works really well on iOS, Android and BlackBerry – I haven't tested other platforms, but you can test them out yourself: http://code.google.com/p/iscroll-js/source/browse/#hg%2Fexamples%2Fcarousel

Remove "click" Delays

"Click" events on HTML elements on mobile devices generally have a delay that is caused by the operating system logic used to capture gestural input based on touch events. Depending on the device, this could be 300-500 MS. While this doesn't sound like much, it is very noticeable. The workaround is to use touch events instead of mouse events: touchStart, touchMove, touchEnd. You can learn more about touch events from html5rocks.com. There's also a great script from cubiq that adds touch events for you to optimize the experience for onClick event handlers on iOS devices.

Add To Home Screen

If you want your web app to fee like a real app and take up the full screen without using PhoneGap as an application container, then you can always add it to the device's home screen. Although this can only be done manually through the mobile browser, there are a few open source scripts to guide the user through this processs: cubiq.org or mobile-bookmark-bubble should get you started.

Use Hardware Acceleration

Animations will generally be smoother and faster if your content is hardware accelerated (and the device supports hardware acceleration). You can make html elements hardware accelerated just by adding the translate3d(x,y,z) css style to the element (be sure to set all three x, y, and z attributes otherwise hardware acceleration may not be applied. If you don't want any translation changes, you can use the translate3d CSS rule with all zero values: translate3d(0,0,0).

```
transform: translate3d(0,0,0);
-webkit-transform: translate3d(0,0,0);
```

In your development/testing, you can even visualize which content is hardware accelerated in both desktop and mobile Safari using the technique shown at http://mir.aculo.us/.

Make Your Apps Fast

Last, but certainly not least, make your apps fast. Follow best practices, and be efficient in code execution and the loading of assets (both local and remote). Here are a few links to get you going in the right direction:

- http://mir.aculo.us/2010/06/04/making-an-ipad-html5-app-making-it-really-fast/
- http://www.html5rocks.com/en/tutorials/speed/html5/
- http://www.html5rocks.com/en/features/performance
- http://www.html5rocks.com/en/tutorials/canvas/performance/

I hope these get you moving in the right direction! If you have read this, and aren't sure what it all means, check out the Adobe Developer Connection to ramp up on HTML5, or theexpressiveweb.com to see what HTML5 & CSS3 can do.

GET THE SOURCE CODE
http://appliness.com/code/01.zip

ABOUT THIS ARTICLE

Andy Trice is a Technical Evangelist for Adobe Systems. Andrew brings to the table more than a decade of experience designing, implementing, and delivering rich applications for the web, desktop, and mobile devices.

http://tricedesigns.com/
@andytrice

ONLINE RESOURCES

HTML5rocks.com website
http://www.html5rocks.com

PhoneGap official website
http://www.phonegap.com

Mobile and Tablet development
http://www.adobe.com/devnet/devices.html

What is PhoneGap?

by Greg Wilson

Last year, in October, Adobe acquired Nitobi, the makers of PhoneGap. After this announcement, I had multiple conversations with conference attendees and found that several of them really had no idea what PhoneGap is. Some thought it was a JavaScript framework that competes with JQuery or Sencha; others thought it was something that converted JavaScript to native Objective C or Java. Both of these are incorrect – not even close… so I decided to write a quick article to explain.

PhoneGap and Android

Since I'm more familiar with Android than iOS, I'll explain how it works for Android.

First, you create a new Android project in Eclipse (requires the Android SDK), add the phonegap.jar file to the lib folder, and make a few tweaks to the manifest and other files (details here). The main java file is modified as follows:

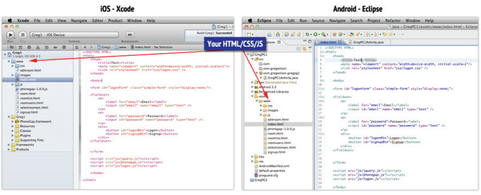
```
package com.gregwilson.gregpgl;

import com.phonegap.*;
import android.os.Bundle;

public class GregPGlActivity extends DroidGap {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        super.loadUrl("file:///android_asset/www/index.html");
    }
}
```

Notice the line in red. We've created a native Android app, and this native app loads android_assets/www/index.html in WebView when launched. WebView is a class in the Android SDK that allows you to display web pages as a part of your layout. It's like having a web browser inside of your app and uses the device's existing implementation of WebKit. In iOS, it's UIWebView. Other mobile OSes use similar techniques.

Below is my project in both Eclipse (Android) and Xcode (iOS). The web application is in the www folder indicated by the red arrows.



Now you simply compile, build, deploy like any other Android project.

Basically, PhoneGap apps are HTML/JS/CSS apps that run within the WebView (or equiv) component. If you are like me, at this point you are thinking, "Uh – that's lame – is that it? I can do that today without any additional software".

But there's more… PhoneGap extends the WebView class to give it hooks back to the device itself and exposed them as JavaScript. Remember the jar file that is in the project? The project also includes a phonegap.js file, which exposes many new functions that make this much more than simply displaying a web page in a WebView component.

Check out the code below for accessing the device GPS *(copied from one of the many great examples from http://docs.phonegap.com)*. If you create an PhoneGap project and copy the code below to your index.html file, you can see it run.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Device Properties Example</title>

    <script type="text/javascript" charset="utf-8" src="phonegap.js"></script>
    <script type="text/javascript" charset="utf-8">

    // Wait for PhoneGap to load
    //
    document.addEventListener("deviceready", onDeviceReady, false);

    // PhoneGap is ready
    //
    function onDeviceReady() {
        navigator.geolocation.getCurrentPosition(onSuccess, onError);
    }

    // onSuccess Geolocation
    //
    function onSuccess(position) {
        var element = document.getElementById('geolocation');
        element.innerHTML = 'Latitude: '+ position.coords.latitude+ '<br />' +
                            'Longitude: '+ position.coords.longitude+ '<br />' +
                            'Altitude: '+ position.coords.altitude+ '<br />' +
                            'Accuracy: '+ position.coords.accuracy+ '<br />' +
                            'Altitude Accuracy:'+position.coords.altitudeAccuracy+'<br/>' +
                            'Heading: '+ position.coords.heading+ '<br />' +
                            'Speed: '+ position.coords.speed+ '<br />' +
                            'Timestamp: '+ new Date(position.timestamp)+ '<br />';

    }

    // onError Callback receives a PositionError object
    //
    function onError(error) {
        alert('code: '    + error.code    + '\n' +
              'message: ' + error.message + '\n');
    }

    </script>
  </head>
  <body>
    <p id="geolocation">Finding geolocation...</p>
  </body>
</html>
```

Now your "web app" has access to Accelerometer, Camera, Compass, Contacts, and many other device capabilities.

Some observations

- Since the app itself is super small, **it loads crazy fast** – less than one second on my HTC Inspire and iPad 2. You probably won't even need a splash screen.
- The framework is **simple and lightweight,** so the resulting app has a very small memory footprint. This allows apps to work on older and slower hardware like the original iPhone and older BlackBerry phones.
- The PhoneGap **docs and examples are fantastic** – clear and concise. I was able to copy/paste every sample and see the results. (one note — the phonegap-1.1.0.js for Android is different than the phonegap-1.1.0.js for iOS – I initially made the mistake of assuming they were identical and killed a few hours trying to figure out why certain samples wouldn't run)
- **The platform support is broad**. As of this article, PhoneGap supports iOS (iPhone, iPhone 3G/3Gs/4/4S, iPad 1/2, Android (all versions), BlackBerry OS 4.6 and newer, WebOS, Symbian, Bada and they have recently started to support Windows Mobile. See http://www.phonegap.com/about/features for a list of what's supported on each.
- Your HTML/JS/CSS run in the native WebView so you are free to use any frameworks you desire such as JQuery Mobile, Sencha, whatever. PhoneGap gives you a WebView/UIWebView with hooks to the device. The rest is up to you.
- **PhoneGap is free, open-source** and is an Apache project (Cordova)
- PhoneGap is **extendable via a plugin model** giving you a bridge between native code capabilities and JavaScript.
- There is already a great collection of community-created plugins at https://github.com/phonegap/phonegap-plugins.
- I haven't figured out a good way to debug these types of apps yet. I suspect that this will be a bit of a challenge. There is a console.log() function that will send messages back to your Eclipse console. When the camera sample wasn't working, I really had no clues as to why. It's likely that I'm missing something, so it's too early to judge.

PhoneGap Build

As you start building mobile apps in any technology, you soon discover that each platform has its own deployment steps and tooling requirments (some of which are amazingly tedious and error-prone). For example, to build a native iOS app, you use Xcode, which only runs on Mac OS X. To build for BlackBerry phones, you have to use their tooling that only runs on Windows. To build for both platforms, you'll need two machines (or use virtualization).

This is why http://build.phonegap.com was created. The service allows you to upload your www project folder (or give it a GitHub URL) and it will package your app for the supported platforms. The specific steps are documented at https://build.phonegap.com/docs/start.

NOTE: Adobe has stated that this service will continue as part of the recently announced "Adobe Creative Cloud" but no details have been provided nor has any pricing been announced.

What's next?

If you're intrigued, I encourage you to go to http://phonegap.com, where you'll find everything you need to build an app. I was able to get a Hello World app running on both Android and iOS in under 30 minutes (not counting the downloading of Xcode). The "Getting Started" section is excellent.

ABOUT THIS ARTICLE

I've been lucky enough to have been involved in all sorts of technologies. For the past 6 years I've been with Adobe speaking at events, blogging, building tools, helping developers and learning something new every single day. I also manage the Adobe Enterprise Platform Evangelist Team.

http://gregsramblings.com/
@gregsramblings

ONLINE RESOURCES

PhoneGap website and documentation
http://phonegap.com

Build in the cloud
http://build.phonegap.com

Video: Getting started with PhoneGap
http://tv.adobe.com/watch/adc-presents-phonegap/getting-started-with-phonegap/

appliness

VIDEO TUTORIAL

HTML5 for App Developers: PhpStorm/WebStorm

by Piotr Walczyszyn

ABOUT THIS ARTICLE

In the video above you will find a quick review of PhpStorm/WebStorm IDEs. This is one of those tools that I will stick with at least for now. It has really great debugging support. Also because it is built on top of IntelliJ IDEA platform it has great code completion, rich set of refactoring features, and many others.

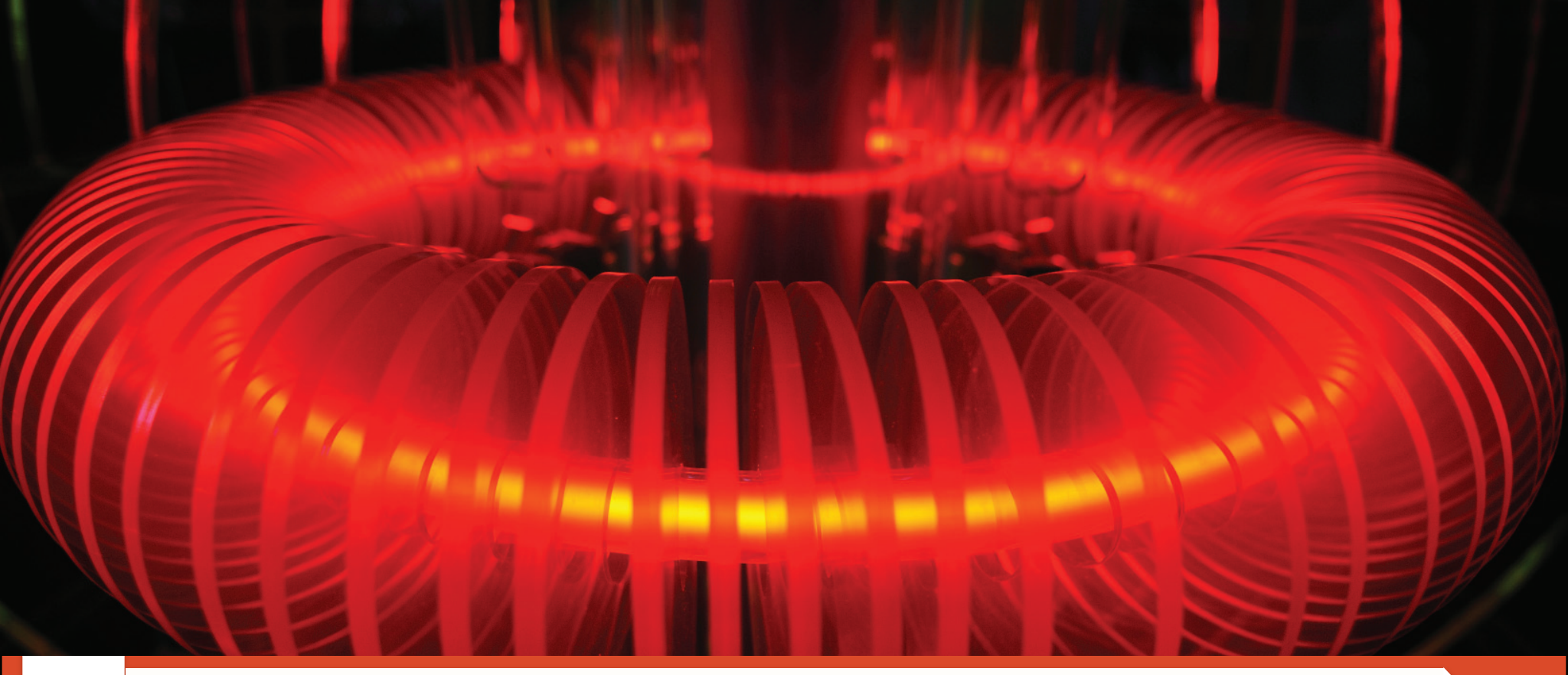
http://riaspace.net/
@pwalczyszyn

ONLINE RESOURCES

Official page of WebStorm
http://www.jetbrains.com/webstorm/

Jetbrains blog
http://blog.jetbrains.com/

HTML5 development
http://www.adobe.com/devnet/html5.html



Bleeding Edge HTML5, WebRTC & Device Access

by Andrew Trice

The world is changing… and oh my, it is changing fast. In the not-too-distant future, many capabilities that were exclusive to plugin-based content will be accessible to the HTML/JavaScript world without any plugin dependencies. This includes access to media devices (microphone and camera), as well as real time communications. You might be reading this thinking "no way, that is still years off", but it's not.

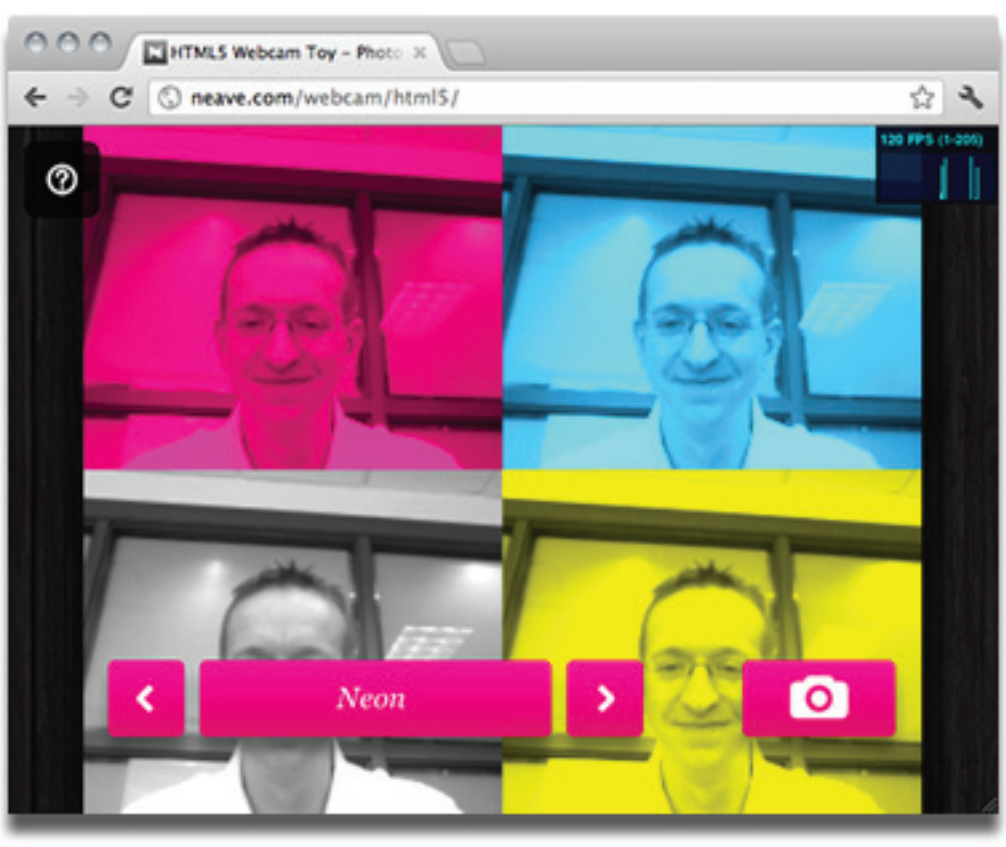
Just last night I was looking at the new webRTC capabilities that were introduced in the Google Chrome Canary build in January, and I was experimenting with the new `getUserMedia` API. WebRTC is an open source realtime communications API that was recently included in Chrome (Canary, the latest dev build), the latest version of Opera, and soon FireFox (if not already), and is built on top of the getUserMedia APIs. Device access & user media APIs aren't commonly available in most users' browsers yet, but you can be sure that they will be commonplace in the not-so-distant future.

Below you'll see a screenshot of a simple example demonstrating camera access.

The beauty of this example is that the entire experience is delivered in a whopping total of 17 lines of code. It uses the `webkitGetUserMedia` API to grab a media stream from the local webcam and display it within a HTML5 `<video>` element.

```
<html>
    <script>
    function load() {
        var video = document.getElementById('myVideo');
        if (navigator.webkitGetUserMedia) {
            navigator.webkitGetUserMedia('video',
                function(stream) { video.src = webkitURL.createObjectURL(stream); },
                function(error) { alert('ERROR: ' + error.toString()); } );
        } else {
            alert('webkitGetUserMedia not supported');
        }
    }
    </script>
    <body onload="load()">
        <video autoplay="autoplay" id="myVideo" />
    </body>
</html>
```

While this example is really basic, it is a foundational building block for more complicated operations, including realtime video enhancement and streaming/communications. Check out this more advanced example from http://neave.com/webcam/html5/, which applies effects to the camera stream in real time:



You can read more about WebRTC, get demos, and get sample code at http://www.webrtc.org

If you want to read more about some of the new "Bleeding Edge" features coming to the web, check out this slide deck by Google's Paul Kinlan. You can also read more about the getUserMedia API from Opera's developer site.

ABOUT THIS ARTICLE

Andy Trice is a Technical Evangelist for Adobe Systems. Andrew brings to the table more than a decade of experience designing, implementing, and delivering rich applications for the web, desktop, and mobile devices. He is an experienced architect, team leader, accomplished speaker, and published author, specializing in object oriented principles, mobile development and data visualization.
http://tricedesigns.com/
@andytrice

ONLINE RESOURCES

More about WebRTC
http://www.webrtc.org

About the getUserMedia API
http://dev.opera.com/

Advanced example
http://neave.com/webcam/html5/

appliness

IS A WARM GUN

We want to showcase in this section the best mobile applications built with web standards. If you want to showcase your application, contact us by email - contact@appliness.net. This time, we are promoting three mobile applications available on iOS and Android devices.

JustOneMore

by Ribot Limited

Designed to help you discover inspiring video content, Just One More provides a simple, addictive interface that brings the very best of Vimeo to the iPad/iPhone. Launched on the App Store in February and built using web-only technologies this free app shows that you really can make immersive, content-rich mobile apps using the latest in HTML5, CSS3, and JavaScript. This application was built with Sencha Touch and is available as a native app thanks to PhoneGap. You can swipe through the latest videos via a smooth user experience on iPad devices. A must for video fans and especially for Vimeo addicts.

Piotr Christophe Michaël

Wikipedia

by Wikimedia Foundation

Wikipedia, of course, is the free, online encyclopedia containing more than 20 million articles in 280 languages, and is the most comprehensive and widely used reference work humans have ever compiled.

The official Wikipedia app for Android was developed with web standards (HTML, CSS, and JavaScript) but runs as a native app thanks to PhoneGap. The developers used some of the new, advanced APIs provided by PhoneGap to implement features such as saving articles to read offline, searching articles on topics that are nearby with geolocation, sharing articles using the native Share function of Android, and reading articles in different languages.

Holly Greg Michaël

Census brower

by Andy Trice

The US Census Browser is an open source application for browsing data from the 2010 US Census. The app was written entirely using HTML and JavaScript, even the charting and data visualization components. The data visualization is implemented completely on the client side, using the Highcharts JavaScript library, which renders vector graphics based upon the data that is passed into it. The fluid scrolling and swiping between views is implemented using the iScroll JavaScript library. It's built as a native app on several platforms using PhoneGap. The source code of the app is available on Andy's blog (tricedesigns.com).

Greg Andy Michaël

appliness

WTFJS?!*%

JavaScript is a language we love despite it giving us so much to hate. This is a collection of those very special irregularities, inconsistencies and just plain painfully unintuitive moments for the language of the web. Brian LeRoux, PhoneGap bla bla bla, is listing on his blog WTFSJ.com funny facts about this language.

“All your commas are belong to Array”

This installment of wtfjs has to do with the Abstract Equality Comparison Algorithm (as most do), Array’s constructor, and expressions.

Let’s take the following example:

```
new Array([],null,undefined,null) == “,,,”; // true
```

WTF? Why does this work?

Firstly, the == causes type coersion (pretty common). From the ECMAScript Specification, 5th edition (final draft), 11.9.3 The Abstract Equality Comparison Algorithm:

The comparison x == y, where x and y are values produces true or false. Such a comparison is performed as follows:

- If Type(x) is either String or Number and Type(y) is Object, return the result of the comparison x == ToPrimitive(y).
- If Type(x) is Object and Type(y) is either String or Number, return the result of the comparison ToPrimitive(x) == y.

So both uses of Array are run through ToPrimitive. But what does ToPrimitive do, exactly? Well, according to another part of the Final final final final draft Standard ECMA-262 5th edition (the document seriously has this title...), 9.1. ToPrimitive:

> Object Return a default value for the Object. The default value of an object is retrieved by calling the [[DefaultValue]] internal method of the object, passing the optional hint PreferredType. The behavior of the [[DefaultValue]] internal method is defined by this specification for all native ECMAScript objects in 8.12.8.

So we’re hinting to the [[DefaultValue]] method within Array with the type of String, so according (again) to the spec, 8.12.8 [[DefaultValue]] (hint):

> Let toString be the results of calling the [[Get]] internal method of object O with argument “toString”.

“JavaScript is a language we love despite it giving us so much to hate.”

Unless of course, IsCallable(toString) (i.e. the object has a .toString method on it’s prototype).

> If IsCallable(toString) is true, then, a. Let str be the results of calling the [[Call]] internal metho of toString with O as the this value and an empty argument list.

And according to 15.4.4.2 Array.prototype.toString ():

> When the toString method is called, the following steps are taken:
>
> Let array be the result of calling ToObject on the this value.
> Let func be the result of calling the [[Get]] internal method of array with argument “join”.

Oh, but we’re not done yet!

Stay with me - we’re type-coersing to a string, and Array.prototype.toString calls Array.prototype.join with no arguments, so we’re joining all the internal members of the array with the default separator is the single-character String “,” (again, according to the spec). When an Array calls join on itself, it’s going from 1 .. len (all it’s members) and calling ToString on these members and concatenating them together. Essentially doing this:

```
Array.prototype.join = function (separator) {
    var result = “”;
    if (“undefined” === typeof separator) {
        separator = “,”;
    }
    for (var k = 0, len = this.length; k < len; ++k && result += separa-
tor) {
        var isToS = this[k] !== null && this[k] !== undefined && “func-
tion” === typeof this[k].toString
        result += isToS ? this[k].toString() : String(this[k]);
    }
    return result;
};
```

So in the end, we end up with weird stuff like this actually working, as [], null, and undefined all result in “” when their respective ToPrimitive methods ask for [[DefaultValue]] with String as the type hint.

Another similar WTF on the same topic:

```
“,,,” == new Array(4); // true
```

This is similar, but not quite the same. When you call Array’s constructor, if there are multiple arguments, they’re interpretted as being members of the Array. If you’ve only put 1 Integer (n) as the argument, an Array object is initiatilized with (n) undefined items. Again, from the spec 15.4.2.2 new Array (len):

> If the argument len is a Number and ToUint32(len) is equal to len, then the length property of the newly constructed object is set to ToUint32(len).

So essentially end up with

```
[undefined,undefined,undefined,undefined].join(),
```

Which yields something like:

```
“” + String(undefined) + “,” + String(undefined) + “,” + String(undefined)
+ “,” + String(undefined)
```

Which ends up being “,,,” (which evaluates to true, as it matches).

Lastly, adding just one more level of WTF to this post, you can also accidentally (or intetionally?) add an expression within Array’s constructor function (and you can also omit new, as the spec also says: “When Array is called as a function rather than as a constructor, it creates and initialises a new Array object.”).

So we can finally end up with the weirdest rendition of this WTF as so:

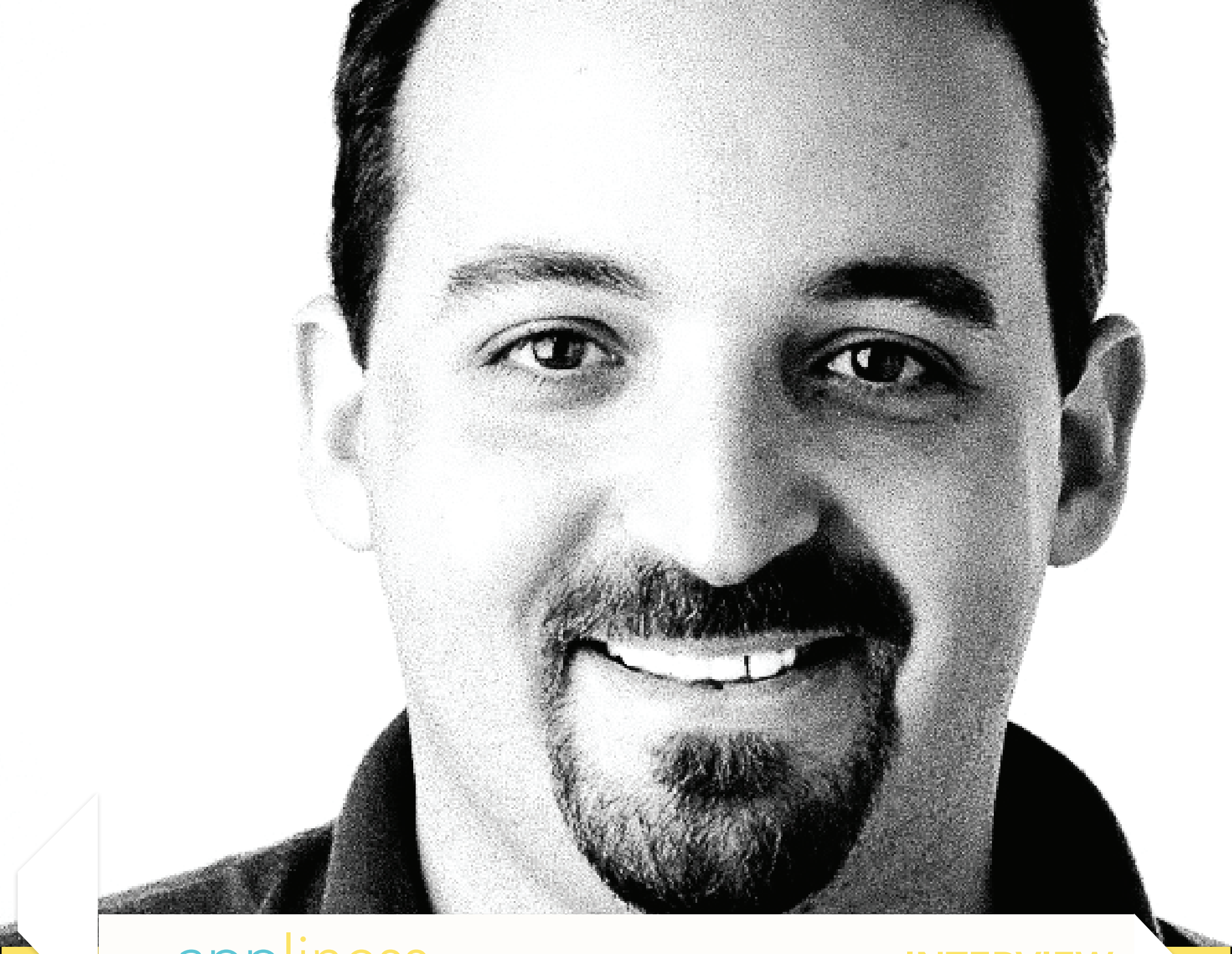
```
“,,,” == Array((null,’cool’,false,NaN,4)); // true
```

If this doesn’t make you WTF, I’m not sure what will.

ABOUT THIS ARTICLE

Brian Leroux is a free/open source software developer at Adobe, formerly of Nitobi, working on PhoneGap, XUI, Lawnchair and WTFJS. Suffice to say, Brian believes that the future of the Web is mobile and will depend on web standards, open source and hackers. Check his crazy blog about JS:
http://wtfjs.com/
@brianleroux

Dan Beam has worked as a front-end engineer for Yahoo! on the Yahoo! Web player and Yahoo! News. He also worked as a Software Engineer for the infamous Ticketmaster on the London 2012 Olympics.
Check his website:
http://danbeam.org/
@danbeam



MAXIMILIANO FIRTMAN

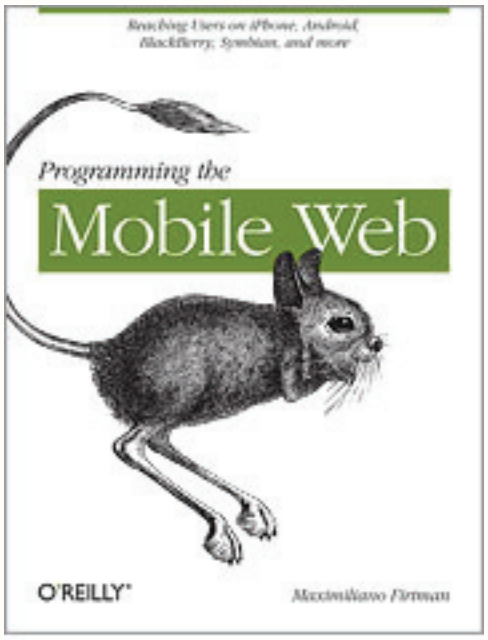
“UI design matters in my coffeemaker, I believe it’s important everywhere.”

Meet Maximiliano Firtman. With 17 years in the field, 8 books and 2,922 pages written, 112 talks, 252 trainings, 158 posts and articles - and counting...it’s no wonder we call him the Godfather of Appliness magazine. In this interview, Mr. Firtman has given us extensive insight into the world of mobile web development, HTML5 and related technologies.

Appliness: Hi, it's a great pleasure to have you interviewed for the first issue of Appliness. You will be the official godfather of this magazine, "el Padrino". Can you introduce yourself to our readers?

Maximiliano: It's an honor and a responsibility to be the first interview. I want to define myself as a mobile+web developer. I'm not just a mobile web developer. I've started doing web development in early 1996 (yes, there was a web 16 years ago) and then I've added web and native mobile development in 2001 (again? Yes, there was a mobile space 11 years ago).

My first 15 years in the area gave me experience in HTML, CSS, JavaScript, PHP, ASP, ASP.NET, C#, Visual Basic, Java, Android, Objective-C, ActionScript (for Flex), Java ME, Qt and probably something else I forgot about. The last year I was more focused on HTML5 and mobile web development.



Programming the Mobile Web

I've written some books, including "Programming the Mobile Web" and "jQuery Mobile Up and Running" both from O'Reilly Media. I love to dig into mobile browsers, so right now I have 35 devices including tablets, e-readers and phones. They allow me to test and re-test every HTML5 feature and post my results on mobilehtml5.org and my blog. I've also created some tools, such as iWebInspector, a free web debugger for iOS simulator.

In the meantime, I travel a lot doing conferences, workshops and trainings around the world.

Appliness: Can you highlight a favorite project you have worked on? How was this satisfying?

Maximiliano: I like writing, so the book "Programming the Mobile Web", 512 pages of full research on everything you need to know about mobile web is a project I loved to do. It was translated to Italian, French and Turkish and I've received messages from all over the world. It's even being used as the official material in some universities.

I can also mention iWebInspector. It is a tool that took me maybe 5 hours of one Saturday (my apologies again to my wife, Ani) and I've received tweets from web designers and developers literally saying I've saved their lives. This kind of little help that you can bring for free to the community is really satisfying.

Appliness: A lot of classic web developers are tempted to extend their skills to coding mobile apps. Do you think that HTML5 can easily answer their needs in terms of mobile application development? (you can include some framework recommendations to get started with mobile apps development)

Maximiliano: If you are a web developer, then you will always be more confortable with Dreamweaver and Firebug than working with Eclipse or Xcode. I have good experience in both worlds. The reality is that you will hate native tools if you are a web developer, so HTML5 seems to be an answer. The problem is for each question HTML5 answers; it leads to ten new questions. HTML5, specifically in the mobile space, is not something concrete, with known limits. Classic web developers need to be prepared for a new way of thinking. Even platforms such as PhoneGap seem so easy to use… until you get into it and try to make it work. I'm not saying it's difficult or impossible; I'm saying you need to understand, learn and embrace new practices and new ideas.

> *"The problem is, for each question HTML5 answers; it leads to 10 new questions."*

Appliness: When should a developer target a web application (in the browser), a hybrid app or a native application?

Maximiliano: I'm sorry to give the worst answer that someone can receive: it depends on each project. Every solution has pros and cons. The goal is to understand all of them and make the best decision on every project. If you want to be in the stores, then my first answer is: "if you can do it with a HTML5-hybrid solution, then do it".

If you have a website, then you must have a mobile version and, maybe, an app. I hate when I enter a desktop website on my mobile. It's a wrong decision from my point of view. It doesn't matter if my browser can render a big page on my small screen.

As a mobile user, I want something fast, easy to read, easy to use and if you are delivering a desktop site you are failing.

Appliness: What solutions would you recommend to develop a multi-platform hybrid app ?

Mobile HTML5

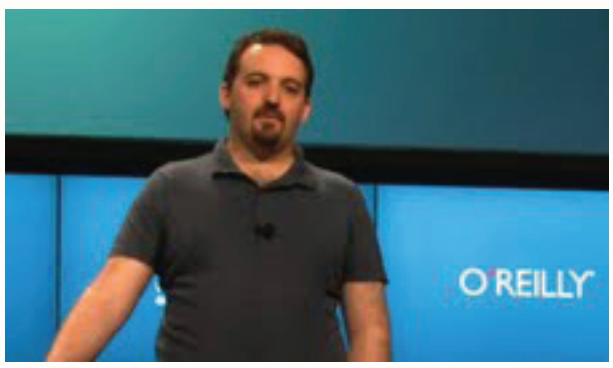
Maximiliano: I personally don't like closed solutions because I'm not sure about their future support. There are lots of multi-platform hybrid solutions out there and I believe every day ten new solutions are born. I like control; so usually in my projects I create a full hybrid environment on my own. If I need some other APIs, then PhoneGap is my recommendation. Usually my multi-platform applications include a data layer, a logic layer and a UI layer shared between every platform. Then I have some kind of a "tweak layer" per platform that adapts the UI or behavior to specific platforms, such as contextual menus on Android or the back button support on Android and Windows Phone.

I've done at least 15 projects in the last year using HTML5 and hybrid solutions for multiple platforms, including iOS, Android, BlackBerry, Nokia and Windows Phone.

"Even if you do not have 35 devices like me, debugging is painful."

Appliness: What are the limitations and the challenges of multi-platform mobile app development?

Maximiliano: One of the most important challenges is debugging on every platform. Even if you do not have 35 devices like me, debugging is painful. Not everything works exactly the same on every platform and finding bugs is a real problem. The other challenge is getting the best UI experience for each platform, and for each screen size and resolution on the same platform. Performance on Android – thanks Chrome for saving our future – and HTML5 API differences complete the limitation list.



Appliness: Do you think that hybrid applications are a temporary solution? (one day, mobile browsers may have all the native APIs… but when?)

Maximiliano: Very good question. If you are using a hybrid because of device APIs, then the answer should be yes. New APIs are appearing in the browser every day. I'm sure we are going to see web-based APIs augmented as a reality this year for example. However, the native side is not only about device API. It's also about discoverability. Right now – we can like it or not, we can discuss this apart – most users use the store as the place to find apps, meaning "things to do with your phone". And the payment process is other problem.

If this idea stays, then we will still need stores and we will still need a way to package our app to be distributed. There are some initiatives to standardize this process for the future, such as the Native Web group in the W3C.

If we solve the discoverability problem, we go forward with APIs and with offline package installation of our webapps, then hybrids are a temporary solution. The major problem I see is how all the big platforms are going to embrace one standard. I can't see a future where Apple, Android, Microsoft and BlackBerry are all using the same package standard and you can share your apps easily. I'm not seeing that. Every try failed in the past – such as the WAC standard.

Appliness: Which actors of the IT industry are the most active to enable first-class mobile web developments?

Maximiliano: Google apps and Facebook. Apple has an important role too in enabling the first good web platform to run on, but I don't see Apple using it for their developments.

Appliness: What is missing today? (a tool? a specific library? some APIs? design stuff? coding experience?)

Maximiliano: We really need remote debugging tools on every platform. We also need better packaging tools: you need to see the face of a web designer trying to deal with Eclipse, Android Developer Tools and XML files to compile his hybrid HTML5 'hello world'.

In terms of API, we need: Full Screen, Shortcut installation, Orientation Lock, Background Notifications, Web Intents and integration with Push technologies.

Appliness: What programs or tools do you use to build mobile web applications?

Maximiliano: If I'm working with native, then it's between Eclipse, NetBeans, Xcode and Visual Studio (yes, all of them installed on the same machine). If I'm working with a hybrid, I use Dreamweaver, lots of simulators and emulators, iWebInspector, Weinre and real devices. Usually I also use Apache to deliver web files in development mode, so I can easily change and refresh the app on real devices connected to my LAN to avoid recompilation and reinstallation on every CSS change.

Appliness: Do you think UI design matters more with mobile than with desktop?

Maximiliano: UI design matters in my coffee maker. I believe it's important everywhere. On mobile is different. It's not more important, it's just different and you need to understand it.

Appliness: Do you think good UI designers are hard to find?

Maximiliano: Specialized in mobile, yes. They are hard to find.

Appliness: Do you approach a tablet project differently than a mobile phone project? What if it's both?

Maximiliano: A tablet project is more mobile than desktop. Why? Because we use the same operating systems, the same HTML5 engines, the same distribution channels and the same pointer: our fingers. That's why a tablet project is treated as a mobile one to me.

Appliness: Do you prefer to work alone or on a team? What would be your recommendations to build a mobile app with other developers?

Maximiliano: This is personal. I prefer to work alone, but I know that's a failure in my motherboard. I don't think teams are bad. My recommendation is to keep the team small. Mobile projects are small. If you have a big project for mobile, then think it again. Done? Think it one more time. Unless you are creating the whole operating system, a mobile project must be small.

"In terms of breaking the rules of mobile web, I like the Gmail webapp."

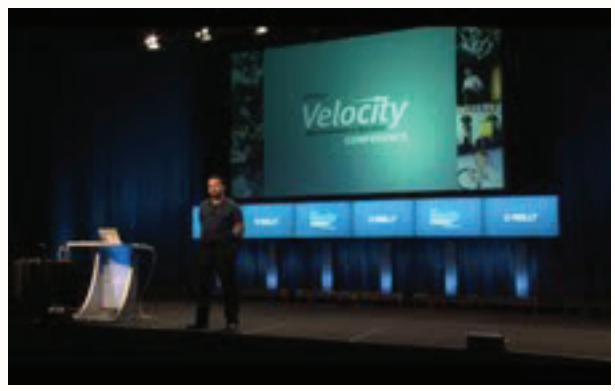
Appliness: Your books are a great source of knowledge for thousands of developers. In your case, what industry sites or blogs do you read regularly?

Maximiliano: Two years ago I made a decision. I can only maintain my reading in one place. Therefore, I've replaced my feed reading with Twitter. And today Twitter is my source of knowledge. That means that you need to choose carefully who you follow. I trust whom I follow so my readings are basically what the network I've created recommends from plenty of sites and blogs.

For my book and my blog… I'm still waiting to find a source. Usually, I have no choice than dig in to the problem myself, test, test and retest.

Appliness: Do you have some mobile applications in mind that you consider as references?

Maximiliano: Not really. In terms of breaking the rules of mobile web, I like the Gmail webapp. They are always trying to push the boundaries and we have a couple of techniques that were created by them, such as some performance hacks. The Financial Times webapp for iPad (app.ft.com) is a really good example of the power of HTML5 to replace a native app.



Appliness: Before starting a new web application development and writing your first line of code, what is your methodology? How do you get inspired? Do you prototype your ideas?

Maximiliano: I don't prototype my ideas. Wait… my mistake; I do prototype, in my head. Usually, I think on a project and I prototype it on my head. When I write my first line of code, I know what I want to do and how. My first line of code is usually a framework. I don't start with the UI. I start with the data layer or a view framework and after I have the data, and a basic navigation framework, I start to add the views. That's not because I'm following a rule or a methodology. That's because it's the fastest way to get the final product. It's incredible how fast you can finish your app when you need only to plug some wires and everything works smoothly after a couple of hard code. I believe the experience with lots of projects gives you this way of thinking.

Appliness: You have trained a lot of developers on mobile development? Have you identified special skills to be successful with mobile development? What's the most difficult concept to learn for a classic developer?

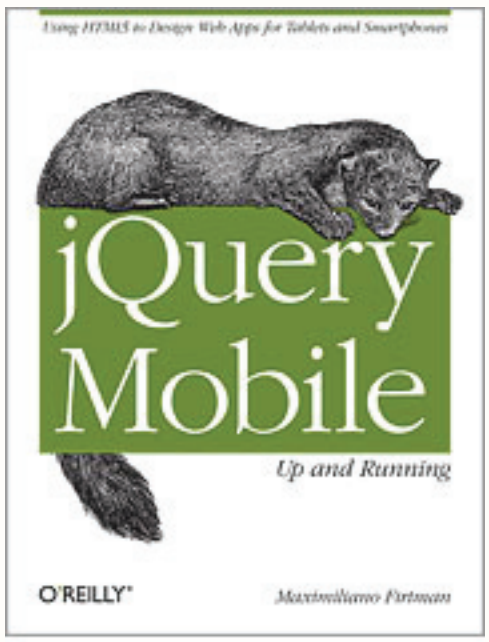
Maximiliano: The most difficult concept is to understand that your result will not be the same on every device. Your design will not be the same; the typography will not be the same; the API compat-

ibility will not be the same. You need to understand the idea of "the best experience for each platform" and not "the same experience for each platform".

If you understand Progressive Enhancement, or if you understand the power of JavaScript and how to write good-layered code, then you will be a good mobile web developer.

Appliness: What are the top 3 pieces of advice you always give to mobile developers?

Maximiliano: Don't be fanatic. Be multi-platform. If you like iOS, good for you… but don't create an app just for iOS (replace "iOS" with any other platform name). People in the world don't use the same device and it will not change in the future.



jQuery Mobile: Up and Running

Deliver the best (not the same) experience for each context. What is context? The platform, the user's network speed, if the user is in roaming or not and lots of other attributes you can get.

Keep performance at the top of your priorities. Performance is a key issue on mobile applications. Understand how to hack and feel the fastest application on earth.

Appliness: What can we wish you for 2012?

Maximiliano: A decent end of the world maybe? Seriously, I wish the mobile platform creators would embrace web developers, as we deserve. Today, we are second-class citizens. There is no – or outdated – information on HTML5 compatibility, best practices or sample code on the web. A new version of the browser appears and there is no "what's new" information for developers. I'm used to digging into the DOM to see what I find. I usually discover good values: Accelerometer API in iOS 4.2, Full-screen API in Chrome for Android, Notifications API on BlackBerry PlayBook. None of these APIs were documented by the vendors.

I wish also to have better testing tools so we don't need to have 35 devices. Ah! Wireless charging. Please! I don't have more plugs for my chargers.

Photographer: Frank Deras

What exactly is Apache?

by Alan Greenblatt

Alan works at Adobe Systems, a company which recently donated PhoneGap and Flex to the Apache Software Foundation. He explains in this article the benefits of such an organization for an IT editor and for the community of developers.

I've been using Apache software in one form or another for about 15 years. But it's only recently, with the advent of Apache Flex that I've started to actually understand what Apache is all about. I'm not referring to the Apache web server here, which in some circles is synonymous with 'Apache'. **I'm talking about the Apache Software Foundation** (ASF), its history and how it works, as well as the Apache-specific projects, and how those projects are developed and maintained.

If you're completely new to Apache software and are wondering about the commercial viability of Apache software, take a deep breath and digest this fact. As of March 2012, approximately **420 million websites use the Apache web server**. That's approximately 65% of all websites. For such ubiquitous software, you'd think there must have been a massive team of developers hard at work, writing specifications, developing the software, testing and supporting it. Yet, truth be told, on average there were no more than 15 developers from different organizations collaborating on the Apache web server at any given time

> *"65% of all websites use the Apache Web server."*

When I say they were collaborating on the software, that gets to the root mission of the ASF. **The ASF provides the legal, financial and organizational support for a broad range of open source, or rather, open development projects.** The ASF only supports collaborative projects that need an infrastructure for a community of developers. This is not a repository of one person code dumps, or master's thesis experiments. These are long-lived projects supported by a community of developers.

Apache projects work because it's all about the community. The actual code is important, but only because it brings the community together. Even the best software, without a supportive community, will actually be shunned by the ASF. If enough people care about that software and want to keep a community thriving, well that's a different story.

Joining that community is simple enough. **All you have to do is subscribe and participate.** Pick a project you're interested in, subscribe to the appropriate mailing lists to stay informed, and start participating in whatever way you can, whenever you can. To get started, some people take a look at the issues list, find a simple bug they can fix and provide a patch to the community. Simple enough. The more you get involved, the more exposure and respect you'll get in the community and more you'll be able to influence that community.

And that gets to one of the prevalent tenets of the ASF, **the notion of the meritocracy**. This applies to Apache projects and the people working on those projects (the community). You might start out as a user who simply uses some piece of Apache software. The success of that project, for whatever reason, may motivate you or your employer enough that you graduate to becoming a developer or contributor on that project. You might fix bugs or write new features. It doesn't matter as long as you are part of the community. If you get involved enough and show your commitment to the project, the project's management committee (PMC) can decide to promote you to committer so you can have write access to the source control system to make direct changes to the code.

Projects are also promoted through a meritocracy. When projects are first proposed to and accepted by the ASF, they are put in the incubation stage and are referred to as podlings. Each podling, with the help and guidance of ASF mentors, needs to prove that they are and will be a successful meritocratic community before the ASF promotes them to becoming a top level project, or TLP.

I recently read one of the status reports for the Apache Flex podling, where it listed the top four items to resolve before graduation to TLP:

- Resolve trademark donation or licensing
- Complete code and bug database donation
- Make at least one release
- Add new committers

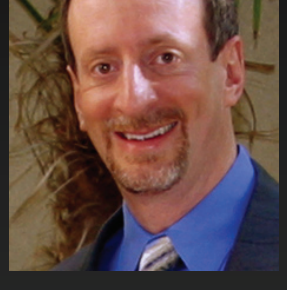
The first two are logistical matters I can understand. And of course the project needs to show they can make a release. If they can't do that, what's the point? But it's actually subtler than that. The Apache Flex community needs to show that they can all come together and agree on a release. That's a big step as a community.

As for new committers, my first reaction was truly confusion. That should be very easy I thought. I'm sure we can get a bunch of people from Adobe to sign up as committers to make this a success. But then I realized that we're talking new committers, not just new contributors. The Apache Flex community needs to prove that they have new people coming in who are contributing enough to the community that they should be promoted to committer, and that the Apache Flex PMC is organized enough to vote them in as committers. The Apache Flex community has to show that they are and will be a successful meritocratic community.

> *"The community needs to show that they can agree on a release".*

Given that there have been **over 2000 mails** on the Apache Flex mailing list in the past month alone; I think we're off to a very successful start.

ABOUT THIS ARTICLE



Alan Greenblatt brings 25 years of software development and technical management expertise to his role as Flex Partner Solutions Architect at Adobe. He has been building and deploying large enterprise data integration applications using semantic web technologies on the backend and Adobe Flex on the front end since Flex was first introduced, and holds patents associated with that work.
http://blattchat.com/
@agreenblatt

ONLINE RESOURCES

The Apache Software Foundation
http://www.apache.org/

The Apache Flex homepage
http://incubator.apache.org/flex/

Apache Cordova (PhoneGap)
http://incubator.apache.org/projects/callback.html

Fresh news about HTML and Javascript collected by Brian Rinaldi.

3D rendered Tron disk with a animated lights.
via Thibault Despoulain

How to achieve functional inheritance through using mixins in JavaScript.

Leverage functional constructs in JavaScript libraries in Node.js using CoffeeScript
via Andrew Glover

Create content areas within an accordion using CSS3.

Real-time editable ByteBeat audio generator using HTML5 and JavaScript
via Gregg Tavares

A great comparison of the three leading HTML5/JavaScript drawing libraries

Chai is a TDD/BDD assertion library for JavaScript

Building a mobile application using jQuery Mobile
via Mihai Corlan

How to save images, and other filetypes, using the browser's localStorage API

Google's Dart programming language integrated in Chromium

Geometry.js is a JavaScript geometry class for developing games

Sam Saffron shows how to stop paying your "jQuery tax" by loading the framework in the footer.

how to Transform HTML using JavaScript on the server using Node.JS

Determining page visibility with JavaScript using the Page Visibility API in HTML5

A version of Weinre ported to Node.js

Gordon Hempton's anatomy of a complex Ember application.

An interesting (and slightly controversial) comparison of the Backbone.js and Ember

This is a nice overview of callback functions in JavaScript by Louis Lazaris.

This is a PDF preview of Appliness, a digital magazine for web application developers.

Download Appliness on your iPad or your Android tablet to enjoy the best reading experience *(interactive samples, links, videos...)*.

If you want to contribute to appliness writing articles or showcasing your apps, visit our website *(www.appliness.com)* and contact us.