mippin
→ makes it mobile

Programming Web applications for the iPhone

This document explains how to get the animated effects seen with many iPhone web applications. We explain which tools to use and how to enhance existing open source Javascript libraries to work effectively on the iPhone. But before reading the whole of this document in all its gory details consider this:-

- If you are a developer and want to create an iPhone web-site which consists of a number of stories then why not use **Mippin.com** to do it. All you have to do is pass Mippin.com an RSS feed(s) from your existing site and hey presto your site will be made available on the iPhone with all this animation and also sized exactly for nearly every other phone in the world too. It could take less than 5 minutes and your job is done –stop reading this document now ☺ and go to http://mippin.com/app/faces/jsp/faq.xhtml to find out more.

So, we didn't persuade you to stop reading this document then! By continuing to read this, it means you want to understand how to write Javascript to create these effects – basically how we did the work for Mippin.com.

First of all get the right tools. Install Firefox (http://www.firefox.com) and the Firefox Javascript debugger add-on called Firebug (https://addons.mozilla.org/en-US/firefox/addon/1843). This add-on was written by a very clever guy called Joe Hewitt and it's his open source Javascript library for the iPhone called **iui**, which we based all our programming on. Download iui from latest software from http://iui.googlecode.com/svn/tags/REL-current/ or original from http://www.joehewitt.com/iui/ - this is an amazing piece of programming which we will delve into in more detail shortly. The important files inside the download zip file are iui.js and iui.css (the iuix.js and iuix.css files are just compressed versions of these files – these can be compressed using Rhino (http://www.mozilla.org/rhino/)).

The Javascript I had been writing for years always started with a command in the document like this:

```
<img src="pic.jpg" onClick="runsomething()" />
```

The difference with the iui library is that you write no Javascript in your html document (other than the initial load statement). All the functionality is written on click events anywhere in your document. This is a very powerful technique because the library acts like a browser functionality extension rather than being specific to your web-site. This enables the iPhone to perform those horizontal scrolling transitions it is known for on every url click in your document rather than you having to program each one which would be the more traditional Javascript approach.

Of course you may not want transitions on every url click or you may want different behaviours on different types of link, so how are these achieved? The answer is to extend your HTML with additional attributes. This keeps the separation between the functionality being in the Javascript and the data instructions being in the HTML. The great thing about this is that we can extend the HTML with additional commands which other phones ignore because they do not know what it means but the Javascript in the iPhone version will recognise.

An example of this in Mippin is where we have enhanced the HTML to implement the left/right

buttons to scroll left and right. The HTML for these buttons has a new attribute called move:

```
        <a href=”pageleft.html” move=”left”><img src=”leftbutton.jpg”/></a>
```

In the Javascript we added the command

```
if (currentLink) {
                effect = currentLink.getAttribute("move");
        }
```

This populates a variable called effect which is used when we decide to scroll to the left or right.

Another area where we enhanced Joe Hewitt’s original code is in the way the horizontal scrolling actually works under the covers. Joe had used the approach of moving the top left position coordinates of the page in a loop (i.e. changing the **element.style.left** variable values for the element representing the whole page). This meant the handset had to redraw the screen on each movement. Although this generally works well the problem was if the handset was doing something else at the same time the redraw often froze and the user got the impression of a stuttering effect as the page scrolled. You can see this on all the sites that have used Joe’s code without any modification. The change we did was to change the scrolling to use the scrollTo() method instead of moving the left coordinate of a page (this is something hinted at in many comments by other developers on Joe’s development page). The reason why scrollTo() works better is that it is the hardware which is move a viewport over the page rather than redrawing the page. However, switching to use scrollTo was not as simple to do as you might expect; the problem was that when scrolling to the right the viewport moves right to 100% of the width of the screen. At this point we need to reset the port back to 0% so any future scrolling can occur. Just before switching it back to 0%, the content underneath it has to be the content currently at 100%. We discovered that in order to prevent flicker at this point we need to duplicate the content and repeat it at both 0% and 100% otherwise very briefly the wrong page appeared – this was made more difficult to diagnose by the fact that in Firefox or even desktop Safari this flicker did not happen – it only occurred on the iPhone itself. We also changed the scroll increment to follow a cosine curve (in our jelly function below) so it accelerated as the scroll started and decelerated as it finished.

```
function jelly(t){
var f=(Math.cos((t)/3.14)+1);
f = (f*50);
if (f<0) return 0;
return f;
}
```

Joe's code generally cached every page being viewed – this is a great approach which reduces the amount of download data and increases the speed of execution. However, since Mippin has access to thousands of sites and as we build this up will becomes hundreds of thousands of sites, caching every page will eventually become a memory issue for the iPhone. As a result we changed the memory caching to only cache within a Mippsite – if the user switches to another Mippsite then we empty the cache. Additionally we cache the category list pages so we can always quickly get back to see the category pages. Because the Mippin is highly structured in its navigation on all Mippsites we could do another clever thing: we fetch the next page of a Mippsite before the user actually asks for it (we call this pre-caching). This means the user does not have to wait when he selects the next page – its just scrolls into place. This is ideal when the iPhone is running on an Edge network where the data download rates are poor.

Apple have been very helpful in making suggestions for improving the look of Mippin. One great thing they gave me a tip for dealing with media content inline rather than having to create a new page. In Mippin we do this when we recognise Podcast content.

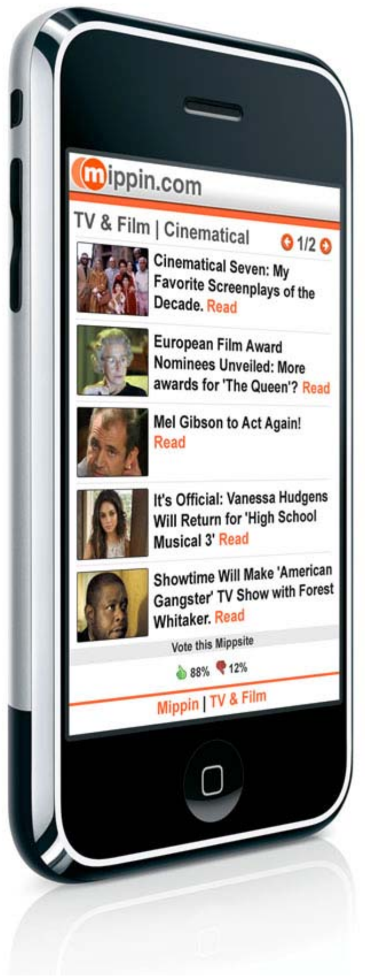
```
<embed width="100%" height="22" enablejavascript="true"
 controller="false"
                    autostart="false"
 pluginspage="www.apple.com/quicktime/download"
                    src="http://url of media"/>
                    <p class="info">Podcast</p>
```

Finally we did get all of this running on another well known handset running the Safari browser. This was even harder to do than the work described here and will be the subject of a future whitepaper if and when we release this version of the code.

This iPhone work was a much bigger undertaking than we'd expected (it must have taken more than 1 man month of effort)– it was partly because this way of using Javascript is new to us, the fact that we have single server-side code base for all phones (and this caused some constraints and interdependencies), but mainly because even though Javascript is a simple interpreted language, the problem is that when a run-time error occurs the Javascript thread just stops – that means that sometimes it difficult to know what state the system is in. We suggest you start developing in Firefox but when it works try desktop Safari, then try the iPhone (you'll find all 3 browser behave differently in the difficult areas).

In short I recommend using Mippin rather than trying to copy what we've done.

Any questions or comments please do not hesitate to contact me at robin@mippin.com.



Cinematical Seven: My Favorite Screenplays of the Decade. Read

European Film Award Nominees Unveiled: More awards for 'The Queen'? Read

Mel Gibson to Act Again! Read

It's Official: Vanessa Hudgens Will Return for 'High School Musical 3' Read

Showtime Will Make 'American Gangster' TV Show with Forest Whitaker. Read