TECHNOLOGY

Scaling Into the Future With Smartlinks

Jason Bengtson

Jason Bengtson is the Emerging Technologies/R&D Librarian for the University of New Mexico's Health Sciences Library and Informatics Center in Albuquerque, NM.(E-mail: jbengtson@salud.unm.edu).

Comments and suggestions should be sent to the Column Editors: Hope Leman (hleman@samhealth.org), Nicholle Mitchell (nmitchell79@gmail.com), Jason Bengtson (jbengtson@salud.unm.edu) and Ellie Bushhousen (ellieb@ufl.edu).

INTRODUCTION

The mobile revolution has changed the digital landscape in profound ways and it can be a struggle to keep up. A proliferation of websites have sprung up designed around the special needs of mobile devices; needs which include small screens, touch interaction, and limited bandwidth and processing power. By far the most commonly seen solution is for organizations or individuals seeking to be "mobile friendly" to create a "mobile site" mirroring their regular website. These mobile sites often showcase resources and features that would (or so the designers believe) be of particular interest to mobile users, packaged in Web design built around mobile device features. A persistent challenge is getting visitors to those sites, as anyone who has read my article about mobile redirect scripts knows. The approach to mobile, however, is evolving.

One of the most promising Web technologies for the mobile era is Responsive Web Design. Employing a few different potential approaches, from grid based layouts to multiple style sheets, Responsive Web Design uses CSS and HTML to cleverly resize, reformat, and recontextualize a single site based upon the screen size of a visitor's device. This revolutionary approach would heal the desktop/mobile rift that has bifurcated the World Wide Web. Yet it is not without problems. Again, as anyone who read my article in last quarter's JHL is aware, the new apple retina displays on mobile devices create challenges for detecting mobile devices by pixel based screen size (which seems to be the main mechanism for mobile detection when creating media types in CSS). Responsive Web Design also removes a certain element of choice from visitors and requires Web designers to think carefully about how to categorize their content when deciding what will not be included on the mobile screen. But, perhaps most importantly for libraries, developers will be in the position of trying to direct users to sometime bifurcated resources from a single set of links.

As an example, AccessMedicine offers a mobile-friendly Web portal, as well as a regular website designed for more powerful desktop computers. On our website at the University of New Mexico's Health Sciences Library and Informatics Center, this division in AccessMedicine is reflected by a similar division in our own website. Our regular, desktop friendly site has links to the desktop-friendly portal for resources like AccessMedicine. Visitors on a mobile device are automatically redirected to our mobile site, which links to the mobile versions of AccessMedicine and other mobile-friendly database portals. But what happens when we have only one site?

One possible solution to this issue has been implemented experimentally on a small scale on the HSLIC homepage. I call this solution "smart linking", and it has the potential to be employed for a variety of uses beyond the scope of the mobility issues it was originally designed to address.

Smart linking uses an external Javascript/jQuery file to "hijack" specified links and change their destination based upon set criteria. While this solution does require that some changes be made to the links, none of those changes affect the basic functionality of the link. This is of vital importance, allowing the solution to "degrade gracefully" in situations where Javascript may be turned off on a device, or in which other circumstances interfere with the functionality of the script. Rather than breaking the links, any failure of the Javascript would simply mean that the links will lead to their default destinations only, making this a particularly safe implementation.

There is currently one smart link in operation on the homepage of the University of New Mexico's Health Sciences Library and Informatics Center. The link that leads to the Groupwise e-mail Web application is governed by a script which functions much like the library's mobile redirect script; detecting the presence of a mobile device, then changing the destination of the link as needed.

The benefits to responsive Web design don't end with the responsive nature of smart links. In order to specify which links should be affected, those links need to have a class attribute added. If you already apply classes to your links for some purpose, don't worry; remember that you can apply as many classes to an element as you like. Since the content of a page often needs to be scaled down as part of the formatting changes needed for a mobile device, this class name gives the style sheet a rather simple and effective indicator to specifically retain those links while turning off other content items.

FORMATTING LINKS

A normal HTML link is comprised of the anchor (<a>) element, and smart links are no exception. Here is an ordinary version of the link to HSLIC's Groupwise Web app:

```
<a title="GroupWise" style="text-decoration: none"
href="http://hsc.unm.edu/mail">GroupWise</a>
```

The "title" attribute here aids in accessibility, but has little effect for most conventional browsing tools (in most browsers it will cause a small balloon, or tooltip, with the word "Groupwise" to appear when you hover over the link). The important part of the link for our purposes is the "href" attribute, which holds the link destination. To make this into a smart link, we’re going to add a couple of attributes . . . but notice how we never change the "href" attribute. Here is our new link:

```
<a title="GroupWise" style="text-decoration: none" rel="http://hsc.unm.edu/mail/mobile/"
href="http://hsc.unm.edu/mail" class="mobilev">
```

We have added two attributes. The first is the class "mobilev". This can, of course, be any class name you choose, as long as it is the class name you use in the external smart link Javascript file. While elements should only have one id value, they can have multiple class memberships, like so: class="newlink, homepage, mobilev". Just like that, you’ve added the "mobilev" class to your element.

The second attribute we’ve added is "rel", which is an attribute set aside for describing the relationship between the linking page and the linked resource. The URL for the mobile version of the application has been added as the value for this attribute. We’re misusing the rel attribute a little here, but if that makes you uncomfortable HTML 5 gives us another option that we’ll discuss later. For now, even though we’ve made some significant changes to our link, you’ll notice that nothing we’ve done will keep the link from doing what it was originally intended to do.

THE Javascript FILE

Now let's look at our Javascript:

```
// Bengtson-Fu
// created by Jason Bengtson, MLIS, AHIP : Available under Creative Commons
//non-commercial share alike open source license
//this sweet little number can be used to send mobile devices to a mobile
//link instead of them going to the
//regular link. If shut down, links still work normally
//must add class="mobilev" to affected links and add rel="mobile url" attribute
//to the links as well
//add beneath jquery library
$(document).ready(function(){
$("a.mobilev").click(function(e){
e.preventDefault();
var linkit=$(this).attr("rel");
var reglink=$(this).attr("href");
mobileswitch(linkit, reglink);
});

 function mobileswitch(linkit, reglink)
{
 if ((screen.width <= 799)||(navigator.userAgent.indexOf('Mobile') != -1))
 {
window.location = linkit;
}
else
{
window.location = reglink;
 }
}
});
```

I've included the script comments not just because I hope that anyone using this script will include the attribution, but also because they describe a lot of the salient features of this technique that we need to be aware of. We've already discussed the changes that need to be made to the link, but notice, as stated in the comments, that we need to add the jQuery library to our page as well, since this file uses some jQuery selectors and methods. This file must be added to the page after the jQuery file is added in order for it to work properly. We'll talk a little bit more about adding these files to a webpage in a moment.

If you aren't terribly conversant with Javascript or jQuery, you don't really need to know how this file works, only that you need your link's class name to match the text in red above. For those of you who do some Web coding, you'll notice that this isn't a terribly complex piece of script. Basically it first identifies the links to target when they are clicked ($("a.mobilerev")), attaching the instructions that follow that selector to the link as an event handler. When the link is clicked, the script disables that link's normal functionality (e.preventDefault();) then extracts the value of the "href" and "rel" attributes, which are then assigned to variables. The script then checks the device screen size (along with a backup check of the User Agent string to catch those pesky retina displays) and, based on what it finds, the user is either sent to the mobile link (from the rel attribute) or the regular link (from the href attribute).

ADDING THE EXTERNAL FILES TO AN HTML PAGE

Adding the file to our page is simple, too, and works just like adding any external file (be it Javascript, CSS, or whatever) to a webpage. First, the file (created in a simple text editor, like notepad) must be saved with a ".js" extension and copied into the folder on your server that holds the various .html and related files that comprise your website. In this case I've named the file "linkreplace.js". As such I needed to add the following line to my webpage, preferably in the header:

```
<script type="text/Javascript"
src="http://hsc.unm.edu/library/scriptfiles/linkreplace.js"></script>
```

The "src" attribute is the URL of the file on your Web server. If you place the file in the same folder as your webpage, you only need src="linkreplace.js" as your src attribute. As my comments within the Javascript file stated, this line must appear after a similar line adding the jQuery library to the page. If you don’t know how to add jQuery you can either visit jQuery.com to learn more about it, or simply add the jQuery library that Google hosts with the line:

```
<script type="text/Javascript" src="
https://ajax.googleapis.com/ajax/libs/jquery/1.7.2/jquery.min.js">
</script>
```

Now you should have your first smart links in place.

USING THE DATA- CUSTOM ATTRIBUTE

As I mentioned earlier, we ended up misusing the rel attribute a bit, but there is another way for us to provide alternative destinations for our links as an attribute in the HTML of the page. HTML 5 provides for custom element attributes prefixed with "data-". Be aware, however, that if you are, for some reason, concerned about your page validating as XHTML, those data attributes will keep it from doing so. Your page should, however, validate to the evolving and somewhat sketchy XHTML 5 standard(when such a validator becomes available). The additional advantage that using data- attributes provides us with is the ability to add an unlimited number of alternative destinations to a link. As such, data- attributes allow us to scale this solution up into more complex applications. An example of an HTML link employing data- attributes is shown below:

```
<a title="LibGuides" style="text-decoration: none;" href="http://libguides.health.unm.edu/" data-
rad="http://libguides.health.unm.edu/content.php?pid=187265" data-
pub="http://libguides.health.unm.edu/content.php?pid=240384" data-
cell="http://libguides.health.unm.edu/content.php?pid=201937" class="mobilev”>Subject
Guides</a>
```

A more complex application of this principle is employed in a proof of concept that I recently built. Upon their accessing this page, visitors are asked to choose between being a member of one of two departments or a member of the public. Using script very similar to the script for the Groupwise smart link, the page dynamically changes a subject guides link to point to a relevant libguide, instead of the general libguides homepage, based upon the value of the variable holding the visitor's response. If the visitor indicated that they were a member of the public, the script also altered the attributes of the HSLIC Quick Search box, replacing the Ebsco Discovery Service search with a search box for Medline Plus. In this way the smart links were scaled into more general smart attributes, all running out of a relatively small external file that can fail without torpedoing the entire page.

CONCLUSION

Smart links and their more general application, smart attributes, provide a potentially flexible solution to challenges posed by the Responsive Web Design model, as well as providing the basis for powerful site customization and dynamic adaptability. Like nearly all of my Web code, I encourage my colleagues to use this approach and further develop it as they see fit to further their library's mission. The Javascript that powers this functionality is relatively simple, and the solution itself is highly scalable.