Developing Apps with the BlackBerry HTML5 WebWorks SDK

Using Third Party Libraries in Application Development

Objective:

This tutorial is designed to demonstrate the advantages of using the HTML5 WebWorks SDK, in conjunction with third party libraries. Third party libraries can greatly enhance the productivity of any developer, cutting down on development time and costs. Unfortunately, most libraries (and frameworks) take time to learn how to use; it can be difficult learn how to use them initially. This tutorial aims to reduce that learning curve, and to introduce you to two great libraries which you can use to enhance any mobile application that you are developing. These libraries are: jQuery Mobile (*JQM*), a library which builds on jQuery's already impressive functionality, gearing it for use with mobile devices, and RGraph, a free HTML5-based graphing solution.

BMI App:

The app you will develop in this lab will be an application which accepts a user's height and weight as input, and then calculates and presents the user with the Body Mass Index (or *BMI*). BMI is a number which is used to give a rough indication as to whether a person's weight is healthy, relative to their height. The formula for calculating an individual's BMI is as follows:

$$\frac{(weight\ in\ kilograms)}{(height\ in\ metres)^2}$$

Getting Started:

Open up the example ***index.html*** provided with the lab. Take note that much of the preparatory work (i.e. the necessary HTML structure and initial application design) have already been done for you. This tutorial will not focus on these aspects, but on how to implement JQM and RGraph to enhance already existing web content.

Examine the top section of the page. Here you can see **<*script*>** tags, which include two of the libraries JQM requires to function: ***"jquery.mobile.1.1.0.js"*** and "***jquery-1.7.2.min.js."*** In addition, JQM takes one style sheet (see the link in the header of the example file). Once these are included, JQM will automatically format all pages in the document according to the rules that it has been given in the style sheet.

The first task in this lab is to specify which elements JQM should format, and how it should do it. In order to accomplish this, we are going to add a special attribute to many of the HTML elements present in the example file. This is an attribute which you likely have never seen before: the ***data-role*** attribute. It is part of the HTML5 standard, and it is now a common attribute which is now shared by most elements. Please note that all the work that you will be doing will take within the body of the html document.

Step 1:

Start by examining the first section, encapsulated by the element <***div id="startPage"***>**.** Give this div the following attribute/value pair: ***"data-role=page."*** This tells JQM to apply its formatting and configuration rules to this section. JQM will take any <***div***> with this attribute, and displays each of them as a separate section of the user interface . Only one page is displayed at a time on the screen. Other <***div***> elements with this attribute will be hidden initially. It is important to note that the first <***div***> with the "page role will be the first thing that will be displayed on the screen when your application is started,. This page based system allows developers to write multiple screens for their application, within one html file. By labeling with the links with the ***data-role="button"*** value/pair, local links (i.e <a ***href="#...***">…<***/a***>) will cause the screen to transition from one of these pages to another.

Figure 1: A *screenshot of the entry page of the final application*

Step 2:

Examine the screenshot given in Figure 1. Notice the black title bar at the top of the screen. It is formatted differently from the rest of the page. This is because a page consists of three sections (as specified in the data-role attribute of a div): The header, the content, and the footer (the same thing as the header only at the bottom of the page).

Look inside the <***div***> with the ***id="startPage."*** Inside it is another div with an empty <***h2***> element. Give the div the attribute ***data-role="header"***, and set the header element

to display "BMI Calculator." Giving the *** *** the header role will make JQM render it as the aforementioned title bar on the top of the page.
Examine the next *** *** in the startPage section. Give it attribute of ***"data-role=content."*** This will indicate to JQM that the material in this div is what is supposed to be displayed on the main area of the screen when the "startPage" section is being viewed. In the content section, there is a form, with several child elements.

Step 3:

Form elements can sometimes be very difficult to position correctly, given the plethora of screen sizes that exist across the many models of mobile devices available today. This is why you are going to surround every element within this form with a new type of role. This is going to be a *** *** which is given the attribute ***data-role="fieldcontain".*** This is a special type of element that JQM provides us which allows developers to specify that a groups of form elements, such as a text box and label, should be placed together, regardless of device size. Giving form elements this attribute helps to structure and ensure that your application is rendered consistently across multiple devices. Give every div within the ***<form>*** the same data-role attribute. This will ensure that our controls stay grouped in the proper order.

Lastly, examine the final div in the "startPage" section. It already has the attribute ***data-role***="footer." With just a ***
*** element, it creates a dark bar at the bottom of the page. This ensures that your application has a consistent look, when combined with the title bar, which we created earlier.

Before moving on to the next section of the lab, format the *** *** elements in the "resultsPage" and the "graphsPage" the same way as you did the "startPage" section, using the work you have done in the "startPage" section as a template.

Step 4:

The last task to take care of is the transitions between pages. Examine the links provided in the different "page" sections. Each of them have links with hashTags and an id which specify a different page in the document. On your own, give them the role "button." Please make note of the link with the id "calcButton," it has a new attribute, ***data-rel="dialog"***. See if, by experimentation using the Ripple Emulator, you can figure out what this changes! Alternatively, any link which has the text "Back" should have a value/pair: ***data-rel="back".*** This will cause JQM to automatically load the page which was previously viewed. This is just another way in which JQM makes the life of a developer such as yourself much easier. Please also make note of note the fact that certain links will not only transition between pages, but can act like a normal ***<button>*** element, triggering JavaScript functions and events as well.

You'll likely notice that there is a **<*button*>** element present in in the graphsPage section. It will open a link to a site with more nutritional information. This will be covered in the JavaScript section of the lab.

Step 5: On your own

Now it is time for you to make a page which who you are and what this app is about. At the bottom of the **<*body*>** element is a **<*div*>**. Create an image, using MS Paint or another graphics application, and place it in as a title. The link in the "graphsPage" section with the which says "About" should be formatted as a button using the data-role attribute. It should also create a transition to the page which you are creating. The about page should also link back to the "startPage" section. Go ahead and try this now!

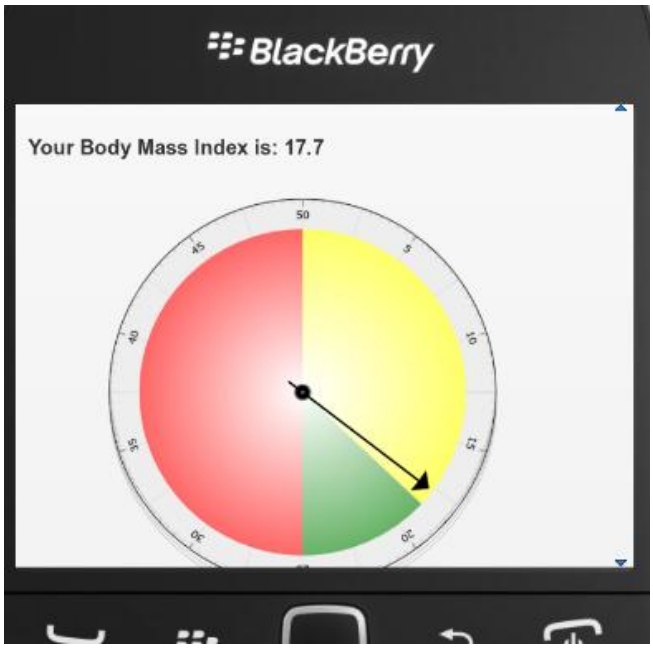

Figure 2 *The graph that your application will generate when displaying a user's calculated BMI*

Calculate BMI:

Open up the ***script.js*** file, and go to the ***getBMI()*** skeleton function provided at the top of the page. Note how on the first line we obtain the weight from the first text box on the starting screen, using a jQuery selector. Now, set the variable ***cm*** to the value in the ***heightInput*** textbox, using the example given above it (which sets the weight to the value in it's corresponding textbox) as a template (hint: Select the input by id). Make note of the use of the ***.val()*** function. This allows the application to retrieve the option has been selected in the drop-down menu which the user has selected. In this case, it controls the

system of measurement that the height and weight inputs will be given in. This brings up an interesting side note, namely that it is important to keep in mind that your applications could be distributed to an international audience, and that individuals in other countries may not use the same units of measurement as yourself. Good application design accounts for this and will prevent you from falling into mistakes which can cut off a large portion of your target audience from your application.

The last portion of the ***getBMI()*** function controls the display of the calculated BMI. The important part to note in this section of code is the use of the ***.toFixed(x)*** function. This function takes a number and limits it to the specified number of significant decimal digits. When looking at data on a screen, most people don't want to look at a huge string of numbers. Unless a high level of precision is required, limit the number of decimal digits a user has to view appropriately.

The last function to examine is the ***startBrowser()*** function. This function uses javascript to invoke the native browser on the BlackBerry device. It directs the browser to a government page with nutrition information. ***blackberry.invoke.BrowserArguments***, giving the URL for the website as a parameter. It then uses the ***blackberry.invoke.invoke*** function, which is called by passing the parameters: ***blackberry.invoke.APP_BROWSER, browserargs*.**

HTML5 Canvas:

For the next section, we are going to be using the RGraph library, found at: http://www.rgraph.net/ . RGraph is a library which uses the ***<canvas>*** element to graph a user's BMI. The ***<canvas>*** tag is a way of drawing high-performance web graphics using JavaScript. It is a new and exciting part of the HTML5 standard, and it is starting to become well supported by most major browsers. RGraph will display results to the user in the form of a circular dial. You will make use of a relatively simple and easy to use graph, which will greatly enhance the quality of your application.

The ***testGraph()*** function will execute the application's graphing functions. It starts by creating a new *Odometer* object. It specifies the following parameters, from first to last: the id of the ***<canvas>***, the minimum and maximum values, and the value that the "needle " of the odometer should point at. The set functions change the locations of the yellow and green colors on the graph, as the order of the colors displayed is normally green, yellow, red. The *Odometer* object then sets the title of the graph to tell the user what their BMI is. Lastly, it draws the graph to the screen using the ***.Draw()*** function.

After having completed all this, load your application into a simulator or onto a device, and test it. You should be able to draw some incredible looking graphs based on a user's input, as shown in Figure 2. The interface should also be animated and have smooth transitions. This would have been a very difficult, and long, application to create from scratch. Using third party libraries, it was relatively easy to develop. Congratulations on completing this lab!

Testing:

In order to test the lab, please refer to the procedures from Lab 1.