url
http://www.patrickcatanzariti.com/2013/01/my-guide-to-developing-a-blackberry-html5-webworks-mobile-app/

I spent the weekend in BlackBerry 10 developer mode. My favourite thing about the BlackBerry 10 is their HTML5 WebWorks platform. With HTML5, CSS and JavaScript, you can develop apps for the upcoming BlackBerry device which can have the same look and feel as an app developed using native code. For a front end developer with a love of the three aforementioned languages, this was music to my ears.
As it was my first ever BlackBerry app, I learnt a lot along the way! I found some of the documentation and guidance on the web a bit scattered and thought it might help to have a step by step guide to help new players looking to get into HTML5 app development. This extremely long blog post is my first draft attempt at doing so! Feel free to provide comments and feedback if I've missed a step or a common issue which you think should be mentioned in one of the steps. I'd like to get it as detailed, yet tidy, as possible. May include screenshots in future!
Okay, here goes.

1. Download Ripple

First thing you'll want to download is the Ripple emulator for Google Chrome. It allows you to simulate what your app will look like on the phone, without having to worry about constantly deploying to a device to test.
You can find it here:
https://developer.blackberry.com/html5/download/
Keep in mind that Google have restricted how you can install extensions into their browser now. If the extension does not come from the Chrome Web Store, you need to install it manually:

1. Download the Ripple emulator .crx file
2. Open the folder you downloaded the .crx file to
3. Open the Google Chrome extensions tab (you can get to it by going to chrome://extensions/)
4. Drag and drop the .crx file into the extensions tab
5. Click add!
6. You should now have the wonders of Ripple installed. Your journey of BlackBerry 10 HTML5 development awaits!

2. Download the BlackBerry WebWorks SDK

A little bit further down on the same page as the Ripple emulator (https://developer.blackberry.com/html5/download/), there is a download link for the BlackBerry WebWorks SDK. Download it!
Once it's finished downloading, double click on it and go through the install process. Feel free to change the installation path to somewhere more convenient if you so desire.

Note: Make a note somewhere of the file path that you choose. You'll need it later!

2b. Download the simulator

You'll want to test your app on the actual device as well at some point so you can see how it works within the actual BlackBerry 10 universe. It's a pretty big download for Mac users (1.02GB… compared to only 504MB). If you're on a Mac like me, you might want to start downloading this now to be prepared in advance. You can find it even further down on the same page as the Ripple emulator and WebWorks SDK (https://developer.blackberry.com/html5/download/).
Note: You'll also need VMware Player if you're on Windows or VMware Fusion if you're on a Mac. Sadly, VMware Fusion costs money, so Mac users will either need to hand over some cash or stick with the 30 day trial version.
You don't need to install the simulator just yet in order to start app developing. So feel free to download it in the background and install it when you so desire. The instructions for installing it are below, feel free to return back and install the simulator when you're ready to start testing.
To install the simulator on Mac, double click the .dmg file that it downloads and Mac mounts it as a device. If you look on the left hand pane you should see "BlackBerry10Simulator-BB10_0_09″ (with "0_09″ possibly a slightly higher number if you're reading these steps in the future). Double click that and run the "BlackBerry10Simulator-Installer" file. Follow the prompts to install the virtual machine somewhere on your computer.
Once you've installed it, go to that folder and run the "BlackBerry 10 Simulator" file in VMware Fusion.
To install the simulator on Windows, run the .exe file that you downloaded and follow those prompts. Once you've installed it. Open the BlackBerry10Simulator.vmx file it installs in VMware Player.
Hopefully following those steps you'll have a working BlackBerry 10 simulator!

3. How a WebWorks app is structured

Now you've got Ripple installed along with the SDK, we can try making a basic app! At bare minimum, a HTML5 WebWorks app has two files in its root folder:

- config.xml – contains the settings for your app, what its name is, the author, what permissions it needs… etc
- index.htm – is the first screen of your app, our starting point!

The most useful documentation I found for this stage in the process was:

- BlackBerry's list of available config elements
- BlackBerry's sample configuration documents
- BlackBerry's Hello World tutorial – a good starting ground for understanding the basics.
- bbUI.js – This is the JavaScript toolkit for getting the native BlackBerry look and feel on HTML5 apps. I'll explain this a bit more below.

- The bbUI.js samples – very useful to get your mind around how it all works!

4. Developing using bbUI.js

If you haven’t already, download bbUI.js and its sample code from:
https://github.com/blackberry/bbUI.js
Developing using bbUI.js means you’ll be working with an AJAX style HTML5 set up. Your index.htm file is your overall template containing your CSS and JS. It will set up functions for the rest of your app and tell the app which page to load first (e.g. in the sample below we load “lafleur.htm”).
The following are super simplified examples of the important bbUI.js functions in your initial index.htm file. Most of these are based off the bbUI.js sample index.htm which I’d recommend looking at for a good overview of the structure for this file. I’ll explain the main bits:

```
<meta name="x-blackberry-defaultHoverEffect" content="false" />
```

As explained on BlackBerry’s documentation on the “Focus Visual Effect”, that meta tag turns off the default blue highlighting. Use it if you wish (I’ve included it as it is in the sample for bbUI.js).

```
<link rel="stylesheet" type="text/css" href="bbui.css"><link />
<script src="bbui.js"></script>
```

Here we are including in the bbUI CSS and JS files. Rather important for using bbUI as I’m sure you’d imagine 🙂

```
<script src="local:///chrome/webworks.js"></script>
```

This is the JS file for the BlackBerry WebWorks framework. Linking to it like this ensures it’s automatically packed into the app when you’ve finished it.

```
<script>
        var webworksreadyFired = false;
        document.addEventListener('webworksready', function(e) {
                if (webworksreadyFired) return;
                        webworksreadyFired = true;
```

This defines what will happen when the “webworksready” event is fired. This is a WebWorks API event that runs once WebWorks itself has loaded. The webworksreadyFired variable is used to ensure the event is only fired once.

```
        bb.init({actionBarDark: true,
                controlsDark: true,
                listsDark: false,
                bb10ForPlayBook: true,
```

This is the start of our declaration of bb.init. This lets you set various settings for the bbUI.js JavaScript toolkit. A list of the possible values for this can be found here:

https://github.com/blackberry/bbUI.js/wiki/Toolkit-Initialization.

```
onscreenready: function(element, id) {
    if (id == 'screenNameOne') {
        dodge(element);
    } else if (id == 'screenNameTwo') {
        dip(element);
    } else if (id == 'screenNameThree') {
        dive(element);
    } else if (id == 'screenNameFour') {
        dodge(element);
    }
},
```

The onscreenready function runs just before styling kicks in and the screen is put into the DOM. The initial code we’ve got above runs particular functions for particular screens that we’ve defined. So we’ve got dip() running for screenNameTwo.

```
    ondomready: function(element, id) {
        if (id == 'Lafleur') {
            averageJoes(element);
        }
    }
});
```

The ondomready function runs after styling kicks in and after the screen is put into the DOM. So if you’re looking to run code that uses DOM elements, make sure to place it here rather than onscreenready.

```
    bb.pushScreen('lafleur.htm', 'Lafleur', {'average' : 'joes'});
}, false);
```

bb.pushScreen is an important piece of code! This is what you run to change the .htm file currently showing in your app. It keeps track of the stack of screens you make visible so that you can use BlackBerry’s inbuilt “Back” function. Pretty useful stuff! In the example above, pushScreen() loads the “Lafleur” page.

Note: Remember the “id == ‘Lafleur’” type statements in the onscreenready and ondomready functions? We define each id in the bb.pushScreen() function.

The “false” variable is the end of the addEventListener function, you know… that one at the very start of this whole adventure.

There is more info on the way that pages are loaded and on how to structure your apps at the bbUI.js application structure page.

5. Setting up a web server to test with Ripple

So you should have a config.xml file and an index.htm file to start with. Maybe you’ve already created your first screen. In order to see how these will look, you'll want to have Ripple running.

If you have a local server running on your computer already (e.g. XAMPP, WAMP, MAMP), you can put your app files into a folder on that server to test them with no issues. I personally used MAMP on my Mac.

In order to run an app, you go to the location of your file in Chrome (e.g. http://localhost:8080/apps/BlackBerryFuntimes). Then click the "Ripple Emulator" icon that'll be next to your address bar (right where Google usually throw in extensions). That'll start up Ripple.

If you don't have a local server, Ripple can provide one for you. However you might need to adjust the permissions on the folder it creates on Mac as on my machine it was read only.

6. Putting an app together

The best place to look to get a feel for the bbUI.js components is to download the bbUI.js samples – https://github.com/blackberry/bbUI.js/tree/master/samples. You can run the samples app in Ripple to see how it all connects together and start piecing them together in your own app to see how things fit. Trial and error is your friend here!

7. Getting it onto a simulator

So you've got an app and you'd like to get it onto a simulator to see how it'll work in the BlackBerry 10 environment. Firstly, make sure you've installed the simulator as explained in 2b. Then follow the instructions from BlackBerry found here: https://developer.blackberry.com/html5/documentation/using_the_bb10_simulator_2008466_11.html.

I found it a bit tough to get the build settings just right in Ripple on my first few attempts:

- **SDK Path:** This is the full path to where you saved the BlackBerry SDK. On my Mac it is the default "/Developer/SDKs/Research In Motion/BlackBerry 10 WebWorks SDK 1.0.4.7″.
- **Project Root:** The folder location of your app (e.g. where your config.xml and index.htm files are). For me, this was "/Users/patcat/Web/Apps/BlackBerry/averagejoes"
- **Archive name:** This can be any name you choose. I just put my app name in camelcase (e.g. AverageJoes).
- **Output folder:** This is where you want the built files to be saved. It can be wherever you would like!
- **Enable remote web inspector:** Click this to have the debug version of your app running on the phone. *For running it on a device (the next section), this was necessary when I was testing it for the app to run on my alpha device. If you don't seem to be able to get the app to build, check you've got this ticked.*
- **Signing password:** This is for when you build the final version. This should be your keystore password.

- **Bundle number:** This was also a bit vague in their documentation but I believe it is the final number in your versioning. So if your version in the config.xml file is 1.0.0.1, put the bundle number as 1.
- **Target:** For the VMware simulator version, choose “Simulator”. For testing on an actual device, choose “Device”.
- **Device IP:** The IP address of the simulator. The BlackBerry documentation on where to find this in the BlackBerry 10 OS. It has pictures and is pretty straightforward so I figured there’s no need to replicate it!
- **Device Password:** Have you got a password on your device? If so, add it here so that it can access it.

Does your app not run? Does your app get stuck on the splash screen? If so, check you’ve got the latest version of the SDK installed on both your simulator and on your computer. I had this issue and it turned out my SDK on my Mac just needed to be uninstalled and the latest version reinstalled.

8. Running it on an actual device

Make sure you’ve got the latest BlackBerry software installed so that your computer picks up the device. Usually it automatically installs as you plug in the device using USB for the first time.

First, register for a debug token. You can do that here: http://developer.blackberry.com/html5/signingkey. This is used to sign your applications before they can be tested on actual devices. Fill in that form and it will email you two files.

Make sure you can remember that PIN you enter in the form here! It’s very important!

To start with, you need to register your keys with your SDK. To do this, go to “dependencies\tools\bin” inside your BlackBerry SDK folder in your command prompt (for Windows) or terminal (for Mac). Run the following command, replacing the RDK and PBDT filenames with the ones which you receive via email:

blackberry-signer -register -csjpin (the PIN you put in when registering on the signing key form) -storepass (your keystore password) client-RDK-0000000.csj client-PBDT-0000000.csj

Important for Mac users – you need to include a ./ before the command in order for it to work! So yours should be:

./blackberry-signer -register -csjpin... etc

Then you can run, from the same folder, the following command to request a debug token from the SDK which will work with your particular device:

blackberry-debugtokenrequest -storepass (Your keystore password) -devicepin (The PIN of your device) aFileNameOfYourChoosing.bar

(Remember Mac users, you may need to include a ./ before the command)

Where do you find your BlackBerry 10 device PIN? Go onto the device and go to Settings > About > Hardware.

Now you’ll want to install that debug token. Open up the development mode screen on your device by going to Settings > Security > Development Mode. Take note of the IP address it has under “Development address”. This is the IP address of your device. Now with the command prompt/terminal in the same folder location as before, run this command: blackberry-deploy -installDebugToken theFileNameYouChose.bar -device (the IP address of your device) -password (the password of your device)

(Remember Mac users, you may need to include a ./ before the command)

If your debug token comes up as invalid, try the following:

Run this command in the same folder location:

./blackberry-debugtokenrequest -verify theFileNameYouChose.bar

and check the date and time of the token's validity. Compare that to the date and time on your test device! Importantly – ***check the time zone!*** *My time zone validity was EST, yet my phone was in Sydney, Australia time. To make things easiest, I just set the time zone to EST on my test device.*

If that didn't work, try updating your device to the latest OS. I found I had issues which were only resolved by updating the phone to the latest version.

After you've got your debug token successfully accepted by your device, make a copy of the .bar file you created and place it in the root folder of your BlackBerry SDK. Give it the name of "debugtoken.bar". *Make sure you do this! Otherwise this just won't work!*

Now you've got your token all sorted out, you can go to Ripple, check that all of your build settings are correct. Make sure you have "Enable Web Developer mode" checked to enable debug mode (so it uses the debug token you worked so hard to set up!). Also make sure it now has your device's IP and password. Hit "Package & Launch" and watch the magic hopefully occur!

9. Packing it up and sending it to the App World

Open up Ripple and click "Package & Sign" rather than "Package & Launch". As long as you've got the right settings for signing your app (your keystore password and a bundle number as I pointed out earlier in this way too long guide), you should hopefully have no issues!

You'll have a .bar file ready inside the "Device" folder within the output folder you defined. This .bar file is what we want to give to BlackBerry.

In order to sell apps on the BlackBerry platform, you'll need to register as a vendor with BlackBerry. You can do that here: https://appworld.blackberry.com/isvportal. You'll need to email proof of ID or company registration, so get that ready.

Once you've been approved, go back to https://appworld.blackberry.com/isvportal and log in. From here:

1. Click "Manage Products".
2. Click "Add Product" and enter in all the details. *The SKU can be anything. I used my app name in camelcase (e.g. AverageJoes). It must be unique though on App World, so adding your vendor name or domain name to add a uniqueness to the name might be a good idea.*
3. Go through all the tabs and prompts.
4. You'll now return to the "Manage Products" screen. From here, click the + icon under "Releases" to add a release for your app.

5. Here is where you can upload your .bar file with the app. Choose “QNX” for the platform. I’ve no idea what this stands for but it appears to be where the BlackBerry 10 device is hidden. Choose the BB10 as the device you’ve designed it for and submit the form! I added the .bar file to a .zip file before uploading but you may not need to do this.
6. *Update: I’ve been advised by Neil in the comments that the BlackBerry 10 OS is a version of QNX, a Unix-like OS developed by a different company now owned by RIM. Life makes more sense now. Thanks Neil!*

It’s now in the hands of the BlackBerry powers that be.

If you have any issues or concerns, the places to look for help seem to be:

- The BlackBerry WebWorks Development Support forum
- @blackberrydev on Twitter

Amendments

Thanks to Stephen Aitchison for spotting an error in my guide for the App World submission process! You upload the .bar file or a .zip file containing the .bar file that is generated in the “Devices” folder when submitting to the App World.

Thanks to Neil for confirming this and providing further feedback which I’ve added into the article.