Comparative study of frameworks for the development of mobile HTML5 applications

Tim Ameye
tim.ameye@student.kuleuven.be

Sander Van Loock
sander.vanloock1@student.kuleuven.be

Abstract

1 Introduction

2 Comparison criteria and related work

Many in-depth comparisons of HTML5 mobile frameworks already exists today. However, none of them are scientifically published or use a large proof of concept (POC) to validate their comparison. Blog posts like [**?**; **?**; **?**] all have their own criteria and methodology to assess different mobile frameworks. The overall applications of the criteria changes from use case to use case. [**?**] presents the chosen criteria and discusses each for each framework. [**?**] presents a whole bunch of criteria but all of them are discusses at once per framework. Thereafter, advantages and disadvantages are subtracted and proposed to the reader. [**?**] finally, presents their chosen criteria together with a scorecard and explanation of scores per criteria. Each framework gets evaluated based on the scores for each criteria.

All these blog posts compare more than two frameworks where some of them mobile and some are hybrid (see table 2). Other websites like [**?**; **?**] only focus on 2 mobile HTML5 frameworks while on the other side of the spectrum [**?**] tries to compare as much as frameworks as possible in a large tabular form.

	Hybrid	Mobile
[**?**]	0	4
[**?**]	2	3
[**?**]*	1**	6

*is still in development
**a combination of Bootstrap, jQuery and Angular JS

As mentioned earlier, we will compare 4 HTML5 mobile frameworks based upon a large-scale proof of concept.

Community Productivity Usage Support Performance

3 Frameworks

3.1 jQuery Mobile

jQuery Mobile (jQM) is a mobile HTML5 user interface (UI) framework that was announced in 2010 [**?**]. In November 2011 version 1.0 was released [**?**] and one year later in October, version 1.2 was released [**?**]. As at the time of writing, jQM will be releasing version 1.3 very soon [**?**]. The framework is controlled by the jQuery Project that also manages jQuery Core. The latter is a JavaScript library where jQM is dependent on [**?**]. jQM is among other things sponsored by Adobe, Nokia, BlackBerry and Mozilla [**?**].

Licence As of September 2012 it is only possible to use jQM under the Massachusetts Institute of Technology (MIT) licence [**?**]. This means that the code is released as open source and can also be used in proprietary projects [**?**].

Documentation One can find the documentation of jQM on www.jquerymobile.com/demos/1.2.0. On the one hand it contains an overview of all possible UI components. By checking the source code, you can find out what code to write to get the same result. On the other hand it explains the API on how to configure defaults, use events, methods, utilities, data attributes and theme the framework [**?**].

Code and development jQM is a UI framework and thus provides mainly UI components. jQM provides 6 categories of components: pages and dialogs, toolbars, buttons, content formatting, form elements and litsviews [**?**]. One can obtain these components by writing HTML5 with jQM specific `data-*` attributes. When running the application, jQM will add the extra necessary code to correctly show these components by doing progressive enhancement.

There are three ways of writing a web application in jQM [**?**]. The first one is to write the full application that is composed of many screens, on one single web page. The advantage is that there are initially less requests to the server. The second option is to write a web page for each screen. The advantage here is that the first viewed screen is downloaded more quickly. However, with each transition, the next screen has to be fetched which can delay navigation. This is done with AJAX by default in jQM. Lastly, you can mix the two above to find an optimum by putting the most likely viewed

screens on one web page and the less likely viewed on separated pages.

Browser support jQM divides browsers into three grades: A, B and C. An A graded browsers supports everything of the jQM framework, where a C graded browsers only provides basic HTML experience (so for example no CSS3 transitions) [**?**].

3.2 Sencha Touch

Sencha Touch is a framework developed by Sencha, a company founded in 2010 as a composition of Ext JS, jQuery Touch and Raphaël. Ext JS is a JavaScript framework for the development of web applications. jQuery Touch is a jQuery plug in for mobile development that adds touch events to jQuery and depends on the WebKit engine. Finally, Raphaël, is a JavaScript library for vector drawings. Pieces of the first two technologies can be found in the implementation of Sencha Touch framework.

Currently, Sencha Touch is at version 2.1.1 [**?**].

Documentation All documentation for Sencha Touch 2.1.1 can be found at docs.sencha.com/touch/2-0. The most important features, are provided with code examples and an example of the code after rendered by the browser. The key concepts of Sencha Touch are explained in extensive tutorials: some texts, some videos.

Another handy tool to discover the Sencha Touch features is the 'Kitchen Sink' [**?**]. This is a web application, written in Sencha Touch, that lines up all possibilities of the framework combined with the corresponding code.

Licenses Sencha Touch is free within a commercial context in which the developer does not share the code with its users. There is also the option to use an open source version. This comes with a GNU GPL v3 license which implies a free code redistribution as most important property. More detailed licenses can be found at [**?**].

Code and development Sencha Touch is written on top of Ext JS, and can also be considered as JavaScript framework. All code needs to be written in JavaScript and loaded by one HTML container. An other important aspect of Sencha Touch is that is supports the Model-View-Controller (MVC) pattern. Models group fields to data-objects, views define how the content is presented to the user and controllers connect these based on events.

Sencha Touch contains all UI-elements as JavaScript objects. Just like object-oriented programming, those objects are part of a class system. Classes can both be defined (`Ext.define`) or created (`Ext.create`). Single-inheritance and overriding is also possible.

To enhance performance, it is the programmers task to create components before they are used. In this manner, programmers can mimic asynchronous loading of pages by creating them in advance.

Browser support Just like jQuery Touch, Sencha Touch is based upon the WebKit browser engine. This forms the major requirement for browser support. Although most mobile browsers contain this engine, some like FireFox Mobile and Opera Mobile lack behind [**?**]. Following [**?**], the next release of the Opera browser will contain this engine, a trend that most browser vendors will (have to) follow.

Sencha Touch offers the programmer methods to ask for the current context where the end-user is working in. Properties like `Ext.env.Browser` and `Ext.env.OS` or methods like `Ext.Viewport.getOrientation` and `Ext.feature.has` can determine this context [**?**]. The latter has functionalities, just like Modernizer [**?**].

3.3 Table

4 Comparison

4.1 Explanation

4.2 Community

4.3 Proof of concept

5 Future work

6 Conclusion

References