Performance of Hybrid Mobile Application UI Frameworks

Radek Vala, Roman Jasek

Tomas Bata University in Zlin, Faculty of Applied Informatics, nám. T.G.Masaryka 5555, CZECH REPUBLIC
{vala, jasek}@fai.utb.cz

Abstract. The choice of right hybrid mobile application UI framework is not elementary these days, because there is lot of possibilities and on the other hand there are no comparative studies, which can help to solve this problem. This paper is focused on HTML5, CSS a JavaScript Hybrid Mobile Application UI frameworks and comparative tests of these frameworks are conducted. The comparison focuses on both the subjective (documentation quality, learning speed, implementation simplicity) and objective parameters, influencing performance of the final application (size of source codes, complexity of DOM structure or scripts optimization).

Keywords: hybrid mobile application, performance test, mobile UI frameworks

1 Introduction

With the increasing use of mobile phones, mobile applications market is rapidly growing. Developers are facing the problem how to produce the mobile application in the shortest time and with minimal costs. To meet these objectives, developers are finding the easiest way to solve the “cross-platformity” in the development process. A lower-cost alternative to native development seems to be a hybrid mobile application approach. Using this approach the need to have different development teams for each mobile platform is eliminated. Moreover the time of the development process could be reduced. Therefore this approach becomes in recent years very popular, but it is not possible to mark it as the only correct. Conversely, if there is not selected ideal hybrid mobile applications development framework (FW), the developers are likely to face a number of issues. The selection of appropriate FW is difficult due to the number of new products appearing on the scene and due to the fact, that there is no comparative study, which could help with the decision.

This paper focuses on the most commonly used hybrid mobile application UI FWs and provides the comparison which should bring relevant data, which are necessary for the right selection.

ISBN: 978-1-61804-246-0

2 Related Work

Although the hybrid mobile application development is a very interesting area in recent years, there is lack of complex comparative study in this field. The situation is very similar to general mobile development area few year ago and the existing body of knowledge is highly pragmatic, with lots of guidelines and many pieces of sample code as examples. [1]

It is probably due the fact, that the research in this field is highly relevant only in short term and for specific FW development version or in context of specific mobile platform version.

Currently, there is possible to find research papers focusing on the basic comparison of native and hybrid development [2] [3] or the challenges of hybrid approach [4][5][6] and lot of the overall statements are well known within the community of developers. However, the specific comparison of the most used hybrid FWs could be a very important information for a huge number of developers which are focusing in non-native mobile application development. The developers are currently honing their knowledge from different on-line sources, such as professional forums, developer groups or other projects which brings comparative information. On-line professional discussion forums with answer quality voting such as StackOverflow, can be considered as a relevant source of information [7], but only in form of partial question/answers. More condensed information can be found on web portals http://mobile-frameworks-comparison-chart.com/ or http://propertycross.com/. However, these sources do not offer any complex FWs feature comparison resulting in some conclusions. It is basically an overview of FW features, or database of example implementations without any performance testing.

3 Candidates of Comparative Tests

For the comparison, following hybrid mobile application UI FW were selected. In the tested group there are both open-source and commercial software tools. Selected FWs are listed below:

1) Intel App Framework [http://app-framework-software.intel.com/]
2) Emy [http://www.emy-library.org/]
3) ChocolateChip-UI [http://chocolatechip-ui.com/]
4) jQTouch [http://jqtjs.com/]
5) jQuery Mobile [http://jquerymobile.com/]
6) PhoneJS [http://phonejs.devexpress.com/]
7) TopCoat [http://topcoat.io/]

All research data were taken in May 2014 and the latest FWs release were used. Table 1 provides the version numbers.

Table 1. Version numbers of tested FWs.

FW Name	version
Emy	v1.0
ChocolateChip-UI	v3.5.5
Intel App FW	2.1.0
jQTouch	v0.99.4rc9
jQuery Mobile	1.4.2
PhoneJS	13.2.9
TopCoat	v0.8.0

4 Weighted Multi Criteria Matrix Comparisonof Frameworks' Features

To obtain comparative results of FWs, following FWs' features were evaluated as important criteria using weighted multi criteria matrix.

4.1 Suitability for Mobile Applications Development (MA)

The criteria of suitability for mobile applications development is subjectively rated from 1 to 5, where 1 means the least appropriate and 5 means the most suitable. In this criterion following parameters are considered: FW contains common GUI objects for mobile platform, layout is responsive and the primary target is the mobile platform. The universal desktop/mobile UI FWs are less suitable because usually offers worse user experience. The DOM structure of HTML elements is usually more complicated and performance issues can be observed.

4.2 Suitability for Desktop Applications Development (DA)

The criterion of suitability for desktop applications development is subjectively rated from 1 to 5, where 1 means the least appropriate and 5 means the most suitable. Desktop development suitable FWs should contain especially common desktop GUI objects (user input dialogs, information dialogs, buttons etc.). In other hand, there should be available also the mobile version of these components. However, this universality can cause performance and user experience issues especially in mobile applications. From the mobile development point of view, there is no need to provide desktop browser support.

4.3 Actuality (A)

Actuality is one of the most important selection criteria of development tools in general. Within this criterion, the frequency of updates per month and date of last commit were evaluated. The frequency usually reflects the usability of the FW in future, when new versions of mobile platforms are issued and some fixes of FW core are needed. These parameters were gathered from the Git accounts [8] in case of open-source projects. According the two parameters above, the FWs were ordered and rated as follows: 7 points – the best and 1 point the worst result. In case of commercial project PhoneJS, average value was chosen, because of lack of public information.

4.4 License (L)

The license policy may be also one of the important feature, which indicates, if there is the possibility of a commercial use without restrictions or it is necessary to buy a commercial license. The rating is as follows: The FW is possible to use without restrictions – 3 points; there is dual license for commercial or non-commercial use – 2 points; only commercial license available – 1 point.

4.5 Documentation (D)

Availability and quality of documentation is directly influencing the learning curve of the new technology. The rating is subjective and based on empirical knowledge and experiences gathered from practical use of tested FWs. The worst evaluation is 1 point and the best is 5 points.

4.6 Size (S)

This factor means the minimal size of FW's source code, which is necessary to import to the project of mobile application. For the evaluation purpose, 6 size classes were set as follows: < 100 kB – 6 points, > 100 kB – 5 points, > 200 kB – 4 points, > 500 kB – 3 points, > 1 MB – 2 points, > 2 MB – 1 point.

4.7 Native look (NL)

The support of native look for different platform is desired property, but it is not a standard. The native looking applications provide better user experience, because the user is familiar with provided GUI patterns and overall look of the GUI objects. The rating is as follows: Support of the newest versions of at least 3 main mobile platforms (Android, iOS, Windows Phone) – 4 points; support of oldest versions of at least 3 main mobile platforms (Android, iOS, Windows Phone) – 3 points; Basic color themes for different mobile platforms – 2 points; universal look – 1 point.

ISBN: 978-1-61804-246-0

4.8 Community (C)

Especially in case of open-source products, the size and quality of community around the project is very important factor which indicates future development of the whole project. The information about the community size was taken from Git accounts. Especially these parameters were evaluated: number of contributors with at least 50 commits and number of issued opened and closed in last 30 days. The FWs were ordered and rated as follows: 7 points – the best and 1 point the worst result. In case of commercial project PhoneJS, average value was chosen, because of lack of public information.

From all of the criteria listed above, the criteria matrix shown in table 2 was created and the normalized version is available in table 3.

Table 2. Criteria matrix for FWs comparison

FW Name	MA	DA	A	L	D	S	NL	C
Emy	5	1	2	3	4	6	4	2
ChocolateChip-UI	5	1	7	3	4	5	5	5
Intel App FW	5	1	5	3	4	4	4	6
jQTouch	5	1	4	3	3	6	2	5
jQuery Mobile	4	2	6	3	5	3	3	7
PhoneJS	5	1	3.5	2	4	1	5	3.5
TopCoat	4	4	1	3	3	5	1	3

Table 3. Normalized criteria matrix

FW Name	MA	DA	A	L	D	S	NL	C
Emy	1.00	0.00	0.17	1.00	0.75	1.00	0.75	0.00
ChocolateChip-UI	1.00	0.00	1.00	1.00	0.75	0.80	1.00	0.60
Intel App FW	1.00	0.00	0.67	1.00	0.75	0.60	0.75	0.80
jQTouch	1.00	0.00	0.50	1.00	0.50	1.00	0.25	0.60
jQuery Mobile	0.75	0.25	0.83	1.00	1.00	0.40	0.50	1.00
PhoneJS	1.00	0.00	0.42	0.50	0.75	0.00	1.00	0.30
TopCoat	0.75	0.75	0.00	1.00	0.50	0.80	0.00	0.20

In context of the mobile application development process with use of some development framework, not all criteria are the same importance. The importance of the criteria differs in each specific project and it is possible to change its weights according subjective preferences. In the case of this research, 4 experts from mobile development area were addresed to compile the expert-reated weights to obtain the

ordinal ranking. There is *p* criteria and *q* experts. The criteria are ordered by assigning the rating *p*, *p* – 1, …, 1. The most important criterion is rated by number *p*, and the less important by number 1. Table 4 shows the resulting expert criteria ratings.

Table 4. Expert criteria rating.

	MA	DA	A	L	D	S	NL	C
Expert 1	7	1	6	2	4	5	8	3
Expert 2	8	1	3	7	6	2	5	4
Expert 3	7	6	5	8	4	2	1	3
Expert 4	7	6	8	2	4	1	3	5

When a_{ij} be the *i*-th criterion rating of *j*-th epert, then the weight of *i*-th criterion by *j*-th expert is calculated using (1). The weight of *i*-th criterion is calculated using (2).

$$w_{ij} = \frac{a_{ij}}{\sum_{i=1}^{p} a_{ij}} = \frac{a_{ij}}{\frac{p(p+1)}{2}} \tag{1}$$

$$w_i = \frac{\sum_{j=1}^{q} v_{ij}}{q} = \frac{\sum_{i=1}^{p} a_{ij}}{\frac{p(p+1)q}{2}} \tag{2}$$

Final results of (2) – expert-rated weights are shown in table 5. As can be seen from the results, the most imporant criteria are the suitability for mobile application development (MA), then the native look (NL) and the size (S).

Table 5. Weights for criteria matrix.

Criterion	MA	DA	A	L	D	S	NL	C
Weight	0.20	0.10	0.15	0.13	0.13	0.07	0.12	0.10

Advantage of this evaluation is the possibility of changing the proposed weights (table 5) to cover own subjective priorities, which could differ in different projects.

According to the result of weighted multi criteria matrix (table 6 or figure 1), the most successful candidate FW is Chocolate-Chip UI and the less successful one is TopCoat. Although PhoneJSis very interesting FW, it has two areas, which were highly penalized. The first one is the license – only commercial use is possible and the second one, more problematic in most of use cases, is the size. PhoneJS contains an iOS theme CSS file which has 1112 kilobytes (due the inserted graphics). But this amount of kilobytes could cause performance issue by initial run of the application.

Table 6. Comparison of hybrid mobile application UI FWs

FW Name	points	percent
ChocolateChip-UI	0.82	100%
jQuery Mobile	0.75	92%
Intel App FW	0.74	91%
jQTouch	0.63	78%
Emy	0.61	75%
PhoneJS	0.57	70%
TopCoat	0.49	61%

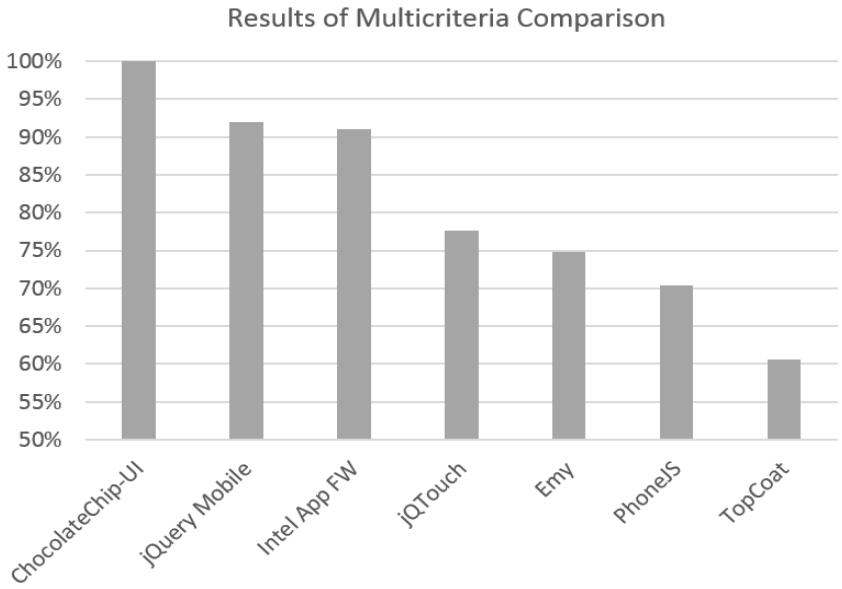


Fig. 1. Results of hybrid mobile UI FWs criteria matrix comparison

5 Hybrid Mobile Testing Application

For performance testing purpose, simple hybrid mobile application were implemented using each of selected FWs.

The application uses typical list view and detail pattern, because it is one of the most used mobile application structure. Moreover, the list view component, allows performance testing of application with very rich and complex DOM structure. The data for the list view component are loaded from the JSON file [9].

In the detail page, there are used the most common form fields, such as labels, text fields, switch fields, radio buttons and buttons. If the detail page is loaded, the form field values are prefilled using the data from the JSON object. Most of the UI FWs are creating some type of form field using DOM Element transformation with JavaScript. Especially switch fields and radio buttons are often generated using this way. Therefore this types of fields are included in testing application to address potential performance issues in different approaches.

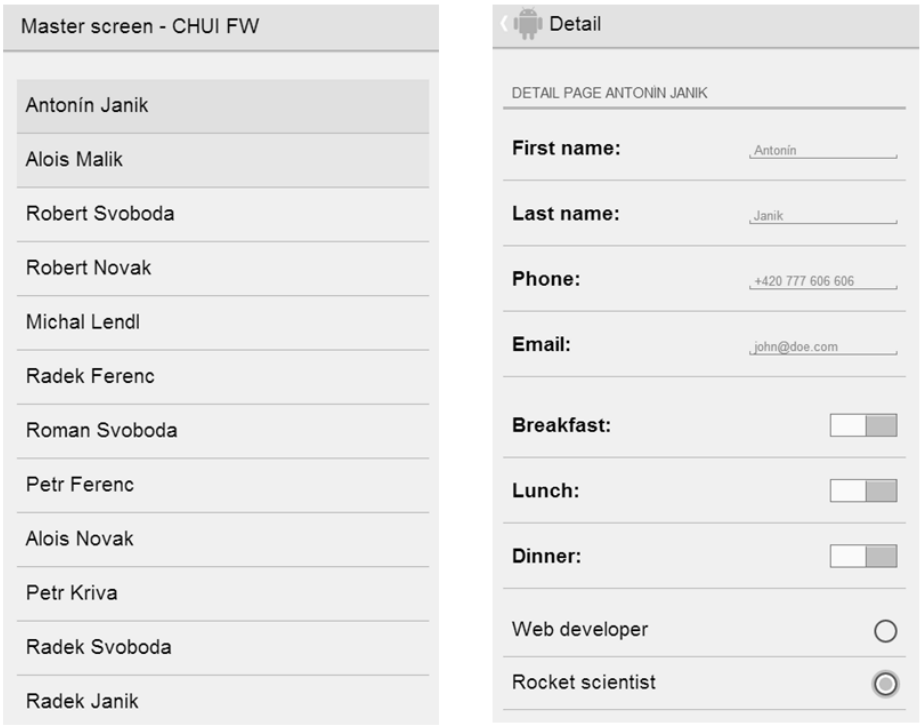


Fig. 2. List view and detail view screen with form components.

6 Performance Testing

The performance testing of selected FWs is focused into the most critical areas such as loading time, scrolling smoothness and page transition smoothness.

The tested applications were run on the Samsung Galaxy Note 10.1 (GT-N8010), with Android version 4.1.2 (in factory settings), within a mobile Chrome browser application (version 35.0.1916.138). The measurements were realized using the Chrome Remote Debugging [10] and Chrome Developer Tools [11].

6.1 Loading time

The loading time of an application could be one of the key factor of application success. According to Compuware research, the median time of user expectation of mobile application load time is about 2 seconds. If this time is exceeded, there is the risk, that some of the users turn the application off.

Loading time measurement methodology.
The time of mobile application load were measured using Chrome Developer Tools Timeline and the goal was to capture the time of DOM Load Event [12] occurrence. Average value of 10 measurement were taken. Between each measurement, the cache memory of the browser were cleared and garbage collector were run.

The second approach was the time measurement of different browser activities, such as Loading, Scripting, Rendering, Painting, Other and Idle.

Loading time results.
In the pictures below are shown the results from the DOM Load measurement (Fig. 3) and particular loading activities times (Fig. 4). As can be seen from this results, the document can be loaded in half the time of particular loading activities. This is thanks to asynchronous loading of external resources, such as Cascading Style Sheets, Java Scripts or Fonts. The overall load time is most influenced by the Scripting time.

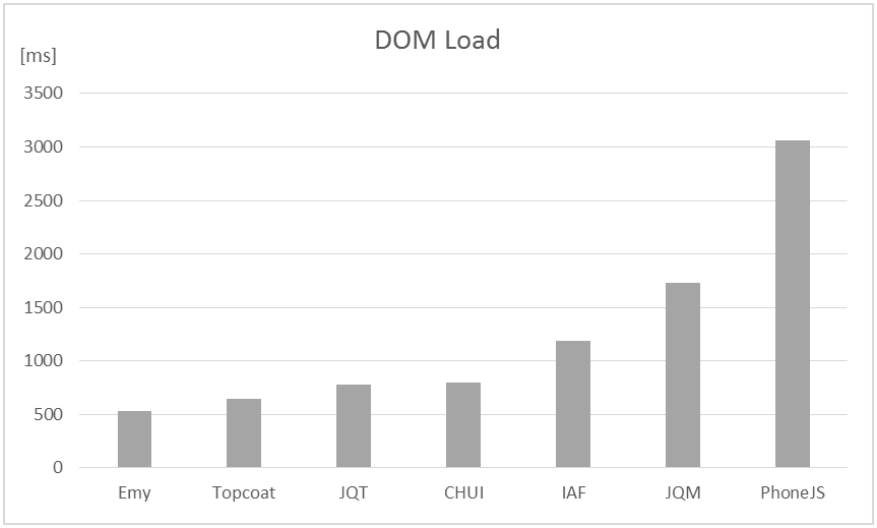


Fig. 3. DOM Load comparison

According the results in Fig. 3, all tested FWs loads within less than 2 seconds (except PhoneJS), therefore it can be stated, that from the application loading time point of view these FWs are suitable for real use. However, it is desirable to have the load time less than one second, because in real scenario, the application could be more complex and other external resources could be loaded.

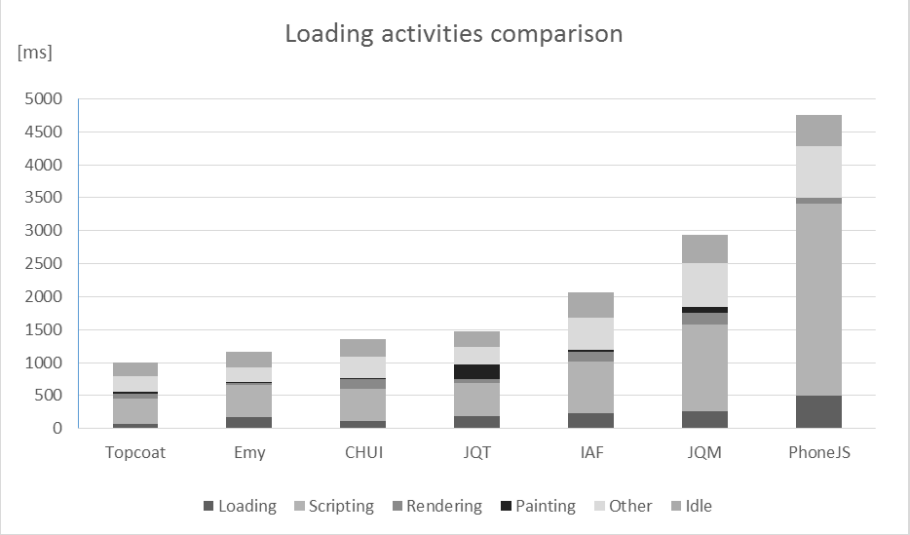


Fig. 4. Loading activities comparison

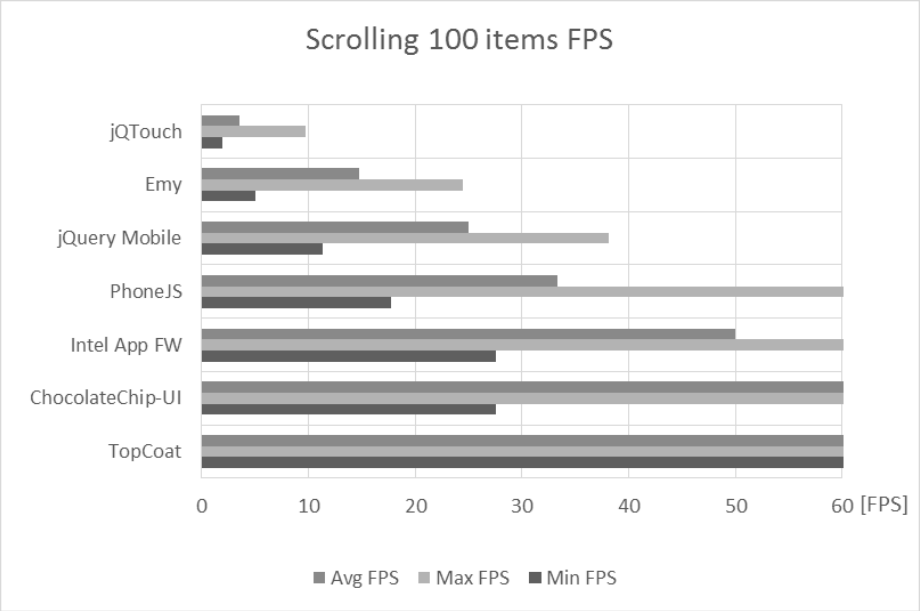
6.2 Scrolling Smoothness

The user experience is not build only during the application lunch, but especially by using the application. Because of limited screen dimensions one of the most often application task is content scrolling. The scrolling should be smooth and the feel should be if possible the same like in case of native application. Especially this area should suffer from the non-fluency, which can be caused by the memory-intensive manipulation with complex DOM structure. If the value of frames per second (FPS) is less than 30 FPS, users are starting to recognize animation plucking.

Scrolling Smoothness Measurement Methodology.
Scrolling smoothness was measured using Chrome Developer Tools FPS meter and continuous page repaint tool. [13] The tested page with 100 items (list-item elements within HTML element ul) were continuously scrolled to obtain the average FPS, minimal FPS and maximal FPS. If the minimal FPS is considerably lower than 30 FPS, users are able to recognize occasionally worst user experience. If there is FPS less than 15, scrolling is obviously not fluent. The maximal FPS of today's browsers is 60 FPS.

Scrolling Smoothness Results.
The most important value from the results in Fig. 7 is the Avg FPS (average FPS). If this value is higher or slightly lower than 30 FPS, the scrolling can be considered as fluent, like in native application. It is necessary to consider also the Min FPS (minimal FPS), because if this value is significantly lower than 30 FPS (under 20 – 15 FPS), it can cause recognizable tearing in particular moment of the animation.

As can be seen from Fig. 5, the most successful FW in scrolling test was TopCoat, where nor average neither minimal FPS was under 60 FPS. It is thank to very simple DOM structure and only CSS formatting with no Java Script transformations. Also very good performance meets ChocolateChip UI FW, Intel App FW and PhoneJS. JQuery Mobile's result is not very satisfactory with 100 items in the list and framework Emy and especially jQTouch was very slow. In real world it is recommended to preserve the item number count under 30 in list views to maintain



the smooth user experience. [14]

Fig. 5. Scolling 100 items FPS results.

6.3 Page Transition Smoothness

User's orientation within mobile application is ensured using proper page transitions. This transitions improve user's idea of mobile screen context. Therefore transitions are highly used among mobile applications on different platforms. Also hybrid mobile application should use this transitions, but there can be often the performance issue caused by complex DOM structure of manipulated content. Choppy transitions are affecting the user experience in a very negative way.

Page Transition Smoothness Measurement Methodology.
FPS of transitions between list view screen to detail screen of the application and back was continuously measured using Chrome Developer Tools FPS meter and continuous page repaint tool.

Page Transition Smoothness Results.
The results of page transition test are relatively satisfactory, taking into account the complex DOM structure (100 items in the list). According to test results in the Fig. 6, only FWs Emy, jQuery Mobile and jQTouch have significant problems with transition animations. But it has to be stated, that the minimal FPS which is under 30 FPS in case of all tested frameworks could be causing little worse user-experience than in case of native applications.

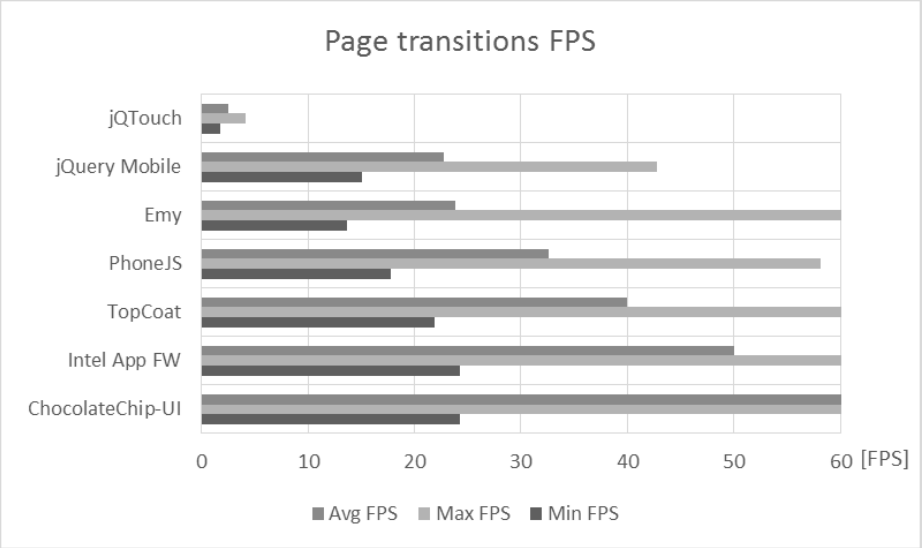


Fig. 6. Page transitions master – detail FPS.

7 Conclusion

Mobile application programming is one of the most developing area in IT world today. Developers are trying to lower their time and money cost per line of code and the cross-platform development is the promising way. In recent years few hybrid mobile application frameworks appeared on the scene and the offer of this type of developer tools is nowadays varied. It leads to the problem of proper choice, because there are no published comparative studies within these FWs.

This paper aims to create a comparative study focused in performance and other selected criteria within 7 widely used hybrid mobile application development FWs.

The selected candidates are firstly compared using these criteria: Suitability for mobile applications development, suitability for desktop applications development, actuality, license, documentation, size, native look and community. Criteria were evaluated using weighted multi criteria matrix in 4. Weightedmulti criteria matrix comparison of frameworks ‘features.

From the performance point of view, the most exposed area such as application load time, scrolling performance and page transition performance were measured and

evaluated in 6. Performance testing. The results showed, that performance issues are very common by using hybrid mobile FWs, and are connected especially with slow scripting and rendering and manipulating a complex DOM structure. There is a direct correlation between FW's simplicity (simple DOM, CSS, no JavaScript) and performance. The simplerthe FW is, the fasterit is, but in this case, there is a lack of tools and widgets often used by developers. The correct choice should be a compromise between power and feature richness. The most successful candidates in this point of view, were Chocolate Chip UI and Intel App Framework.

Acknowledgment

The research work was performed to financial support by the European Regional Development Fund under the Project CEBIA-Tech No. CZ.1.05/2.1.00/03.0089.

References

1. Tony Wasserman. "Software Engineering Issues for Mobile Application Development" *FoSER2010* (2010).
   Available at: http://works.bepress.com/tony_wasserman/4.
2. CHARLAND, SuyeshaArianit KURTI. Cross-Platform Mobile Development: Challenges and Opportunities. [online]. 2011-05-01, issue 5, s. 219 [cit. 2014-05-20]. DOI: 10.1145/1941487.1941504. Available at: http://link.springer.com/10.1007/978-3-319-01466-1_21.
3. *Techniques for Surviving the Mobile Data Explosion* [online]. Hoboken, NJ: John Wiley, 2014-03-17, vol. 54, issue 5 [cit. 2014-05-20]. ISSN 00010782. Dostupné z: http://doi.wiley.com/10.1002/9781118834404.ch10.
4. Sin, D.; Lawson, E.; Kannoorpatti, K., "Mobile Web Apps - The Non-programmer's Alternative to Native Applications," *Human System Interactions (HSI), 2012 5th International Conference on* , vol., no., pp.8,15, 6-8 June 2012
   doi: 10.1109/HSI.2012.11.
5. Ng Moon Hui; Liu Ban Chieng; Wen Yin Ting; Mohamed, H.H.; RafieHjMohd Arshad, M., "Cross-platform mobile applications for android and iOS," *Wireless and Mobile Networking Conference (WMNC), 2013 6th Joint IFIP* , vol., no., pp.1,4, 23-25 April 2013
   doi: 10.1109/WMNC.2013.6548969.
6. AMATYA, SuyeshaArianit KURTI. Cross-Platform Mobile Development: Challenges and Opportunities. [online]. s. 219 [cit. 2014-05-20]. DOI: 10.1007/978-3-319-01466-1_21. Dostupné z: http://link.springer.com/10.1007/978-3-319-01466-1_21.
7. Nasehi, S.M.; Sillito, J.; Maurer, F.; Burns, C., "What makes a good code example?: A study of programming Q&A in StackOverflow," *Software Maintenance (ICSM), 2012 28th IEEE International Conference on* , vol., no., pp.25,34, 23-28 Sept. 2012
   doi: 10.1109/ICSM.2012.6405249.

8. GitHub - Build software better, together (2014). Retrieved May 14, 2014, from https://github.com.
9. Introducing JSON (2014). Retrieved May 16, 2014, from http://json.org/.
10. Remote Debugging on Android with Chrome (2014). Retrieved May 16, 2014, from https://developer.chrome.com/devtools/docs/remote-debugging.
11. Chrome Overview (2014). Retrieved May 16, 2014, from https://developer.chrome.com/devtools/index.
12. DOMContentLoaded (2014). Retrieved May 17, 2014, from https://developer.mozilla.org/en-US/docs/Web/Reference/Events/DOMContentLoaded.
13. Chrome Rendering Settings (2014). Retrieved May 17, 2014, from https://developer.chrome.com/devtools/docs/rendering-settings.
14. Secrets of aGoodJQueryMobilePageArchitecture (2014). Retrieved May 17, 2014, from http://www.gajotres.net/secrets-of-a-good-jquery-mobile-page-architecture/.