CHOOSING THE RIGHT HTML5 FRAMEWORK

To Build Your Mobile Web Application

A RapidValue Solutions Whitepaper

Author: Pooja Prasad, Technical Lead, RapidValue Solutions

Contents

Executive Summary

The mobility sector was mainly dominated by native technologies (Android and iOS applications) until the rise of HTML5. For a web developer to build mobile applications can be challenging with variety of platforms to choose from and technologies to learn. HTML is known to be an easy to learn and fast to implement technology, and has the maximum number of web applications to its credit. However, when HTML5 was released by the World Wide Web consortium, it came along with an added advantage in the mobility domain i.e. cross-platform capability with a single code base. The native technologies demand a higher cost to market, since they consist of SDKs and IDEs, and require a higher learning curve for each device platform. HTML is a well-known technology to most of the in-house developers and the enterprises can start entering the mobility sector right away.

HTML5 technology though gaining momentum quickly is still not quite ready to be used for complex web applications, particularly line-of-business applications. Many frameworks built on HTML and JavaScript are available to enable easier development. However, the web/desktop applications differ from the mobile applications. The device capabilities and usability are a major factor while developing a mobile application. The common questions which most of the enterprises have in mind, before building mobile applications are which technology framework to choose to build their first mobile app and what factors to consider in making the right choice.

This paper provides a guide for developers and solution architects to understand the different software architecture patterns, HTML5 frameworks available to build mobile apps, pros and cons of these application development frameworks and elements to consider for selecting the right framework, while making a decision to build mobile web apps.

If you are a web/desktop developer and looking to build your first mobile application, please refer to our whitepaper on Making a transition from web/desktop application development to mobile application development. http://rapidvaluesolutions.com/whitepapers/making-the-transition-from.html

Software Architecture Patterns

Most development communities are familiar with the MV* architectural patterns such as MVC (Model -View-Controller), MVP (Model-View-Presenter) and MVVM (Model View-ViewModel). MVC is the most widely used pattern. MVP is similar to the MVC pattern. However, MVVM is different from MVC and MVP patterns in many aspects.

Common Parameters

Before we get into more details about each of these patterns, listed below are some of the common components in the three architectural patterns.

Model

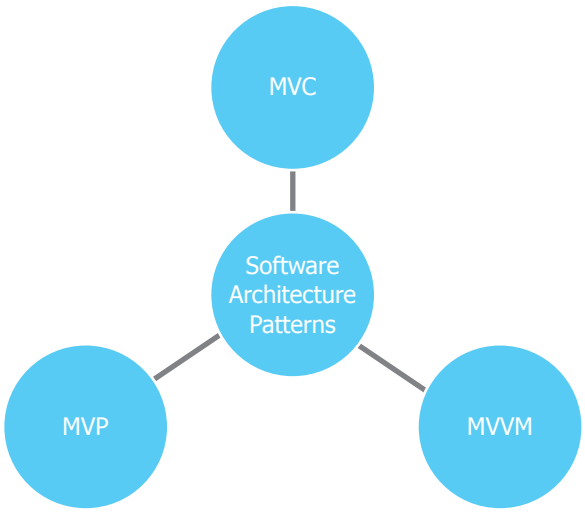
The 'model' component holds data together. It consists of application data, business rules, logic, and functions. It is also responsible for managing the data access layer and communicating with the database. This component mainly consists of model classes and data access objects. In case of mobile apps, in which web service calls are common, the model could contain the service layer as well.

View

The 'view' component is purely the user interface part. This consists of the UI elements. In terms of HTML, the html tags that form the page constitute the view.

Definitions of Architecture Patterns

The term software architecture intuitively denotes the high level structures of a software system. The definitions of the three architectural patterns are as follows:



1. Model-View-Controller

Model–View–Controller (MVC) is a software architecture pattern which separates the representation of information from the user's interaction with it[1]. The controller component in MVC acts as a mediator between the view and the model. Any change in the view that calls for a change in the data is updated in the model by the controller and any change in the model is notified to the controller to update the view. The MVC pattern is one of the first architectural patterns to emerge and is extensively used in Java platforms.

2. Model-View-Presenter

Model–View–Presenter (MVP) is a derivative of the Model–View–Controller (MVC) software pattern, also used mostly for building user interfaces[1]. The presenter component in MVP is closely associated with the view component because it holds a reference to the view. Apart from the role that a controller plays in MVC, a presenter is responsible to make changes which are needed to take place for specific elements of the view based on the changes made to other elements in the view. This kind of view logic based on user interaction is handled through the presenter and these changes are unaware to the model component.

3. Model-View-ViewModel

MVVM is targeted at UI development platforms which support event-driven programming such as HTML5, Windows Presentation Foundation (WPF), Silverlight and the ZK framework. MVVM facilitates a clear separation of the development of the graphical user interface (either as markup language or GUI code) from the development of the business logic or back-end logic known as the model (also known as the data model to distinguish it from the view model)[1].

The view-model acts as a model for the view. This means, the view-model holds data for the view. This is established through data-binding. The view is completely unaware of the changes made to the data that it is bound to, and the view-model is completely unaware of view structure.

[1]Source: Wikipedia definitions

Choosing the Right Architecture for Mobile Web App Development

Implementing a MVC architecture pattern would be the easiest approach, since this is the most common pattern known to the developer community. Considering the way mobile web apps are created, if we try to break-down an application into a model, view and controller, the view would consist of a set of html elements created in a defined structure. The model would take care of any data fetched from the database or through the web services. The controller would be responsible for handling the changes based on the user interaction at the view as well as the responses obtained from the model.

If we look into the above structuring in more detail, handling the user interactions at the view requires that the controller has a reference to the view. There would also be cases where the data that is entered by the user or displayed from the server requires presentation logic. This calls for addition of presenter component and therefore relates more to a Model-View-Presenter pattern than MVC.

The benefit of using these patterns comes from the familiarity and years of experience in architecting solutions using MVC/MVP. These two are more popular and commonly used patterns than MVVM.

MVVM architecture requires a paradigm shift in the way we perceive the application. This pattern establishes separation of concerns and the open-close principle well. This basic capability takes MVVM to next level (ahead in competition). MVVM has been in use in Microsoft applications and therefore is not completely new. Adopting MVVM to build mobile web application gives developers the added benefit of enabling independent testing of different architectural units.

Frameworks to Build Mobile Web App

There are several application development frameworks which are popular among developers. In this paper we will address pros and cons of some of the popular frameworks used by developers which include Knockout.js, Kendo UI, Sencha Touch and Backbone.js.

Knockout.js

Knockout.js (Knockout) is a standalone JavaScript library. The library works on top of jQuery and is not a supplementary to jQuery. It is based on MVVM architectural pattern. This framework is used to build many complex applications. Although it may be possible to develop complex applications using jQuery alone; maintaining, adding new features and troubleshooting the app becomes time consuming and expensive. This is where Knockout.js is most useful. It provides the application a structure making the app more scalable and easily maintainable.

What's in Knockout?

Knockout is purely JavaScript which enables it to be used with any other web framework. Therefore Knockout can be used with most JavaScript based UI frameworks. This makes the UI highly customizable and provides developers with a large library of external plugins to work with.

Knockout adopts declarative binding (two way binding) which allows developers to associate DOM elements to particular model data and its properties. All this is achieved using concise and readable syntax. Even though Knockout is library agnostic, it works particularly well with jQuery. Knockout has minimal external dependencies which makes it robust and supported across most modern browsers such as IE 6+, Firefox 2+, Chrome, Opera, Safari (Desktop/Mobile).

What's missing in Knockout?

Knockout does not enforce that a particular folder structure be followed. Developer can write the code in any file and follow any folder structure. This feature becomes important when multiple developers are working on a complex app. Also it does not provide an out-of-box solution for app development since it is a framework which depends on other UI frameworks like jQuery mobile for UI generation. Integrating Knockout with other jQuery frameworks (although possible) is not fully smooth and requires some amount of coding.

When should you use Knockout?

Developers can select Knockout when you require:

- **High level of UI customization** – this framework includes variety of widgets. Also number of plugins available for jQuery is large compared to other frameworks and this can be leveraged with Knockout.js.
- **Moderate learning curve** - this framework does not have a UI support, therefore developers needs to use a UI framework that they are familiar with. Since Knockout is built on jQuery, it is easy to learn. The only new thing to learn is data binding.
- MVVM architecture is preferred.
- **An open source library** – this helps developers to avail all the associated benefits provided in open source.

Kendo UI

Kendo UI is a jQuery based HTML5 commercial framework developed by Telerik. This framework includes MVVM architectural pattern. It allows developers to build HTML5 apps without hand-coding JavaScript.

What's in Kendo UI Mobile?

Kendo UI Mobile allows developers to build native-like apps and sites for mobile devices. It consists of rich HTML5 widgets and a single codebase to target multiple phone and tablet platforms. Kendo UI Mobile provides UI widgets for iOS, Android, BlackBerry and Windows Phone, and a complete application framework to handle app navigation, views, layout templates, and more. Kendo UI does not deform the DOM. This helps us to do a lot more with less number of supporting elements and with less coding.

Kendo UI supports seamless integration between the model data and the view. It also supports Unit testing frameworks like QUnit and Jasmine.

What's missing in Kendo UI?

Kendo UI is paid and it does not have the same kind of backing of an open source community as Knockout does. This increases the cost of overall development. Kendo also lacks variety in UI widgets. Therefore, it is difficult to use for building complex mobile apps and requires high-level of customization. As the UI is coupled with the framework developing new widgets is difficult too.

When should you use Kendo UI?

Developers can select Kendo UI when you require:

- An out-of-box and affordable solution - Since Kendo UI is a commercial development framework it provides paid support.
- Native-like app experience - Kendo UI implements native packaging using PhoneGap. This framework has its own IDE called Icenium. The IDE defines a folder structure for every app in case the application is complex and provides a facility to publish iOS and Android apps directly to their respective marketplaces.
- Faster development time - Kendo UI consists of easy-to-use jQuery-based widgets, in-built rich components and JavaScript library which enhances mobile application development. The learning curve involved is moderate and developers who have worked on Knockout could easily adapt to Kendo UI.

Sencha

Sencha is a JavaScript framework which allows building cross-platform applications for web and mobile. Sencha Touch is a JavaScript framework used to build apps for mobile devices.

What's in Sencha Touch?

Sencha Touch has built-in MVC support with the folder structure specified. This ensures an organized approach to app building and helps in maintaining and scaling a particular app. Sencha Touch is an open source library, therefore it has large number of developer communities supporting it. This framework's key strength includes graphical representation of data; making it a most preferred framework.

Sencha is consistent across most modern web browsers (IE6+, Chrome), Apple iOS and Google Android platforms. It also supports unit testing with frameworks like QUnit and Jasmine.

What's missing in Sencha Touch?

Although it is an open library it cannot be used easily with other frameworks. It is not backward compatible. If a newer version of Sencha is released, to include those new features into the app requires features to be rewritten.

When should you use Sencha Touch?

For developers to start using Sencha Touch, you need to have working knowledge of the JavaScript language and CSS.

Developers can select Sencha Touch when you require:

- A highly scalable mobile app, which mostly includes graphical representation of data. Building complex mobile apps is easier with Sencha Touch as it enforces a folder structure making it easily scalable and maintainable app.
- Better user experience app - It includes native packaging for rich user experience.
- MVC architectural pattern is preferred.
- Learning curve is moderate.

Backbone.js

Backbone.js is an open source light weight JavaScript library (component of DocumentCloud) which provides structure to web applications by providing models with key-value binding and custom events, collections. It consists of a rich API of enumerable functions, views with declarative event handling, and also connects it all to your existing API through RESTful JSON interface[2].

This framework includes MVC architectural pattern. However, in Backbone there is no such thing as a typical controller. The role of a controller is partly played by the views which contain UI logic along with representing data.

What's in Backbone.js?

Backbone.js is another framework which is ideal for writing complex multi-screened apps as it lessens the problem of code getting cluttered by providing an event-driven communication between views and models. Developers can attach event listeners to any attribute of a model. This gives you nuanced control over what you change in the view. Another feature of backbone is that the models in Backbone.js can be easily tied to back-end. It provides excellent support for RESTful API in which models can map to a RESTful endpoint directly. It enforces maintainability i.e. if conventions are followed; less code needs to be written. This helps to maintain a clean code base, inspite of having multiple people collaborating on the code. Backbone is a good alternative for apps looking for an MVC based solution which needs to be scalable and maintainable.

What's missing in Backbone.js?

Backbone.js does not have a UI framework built into it. It depends on other UI frameworks for the actual UI rendering .Therefore it does not offer an out-of-box solution.

When should you use Backbone.js?

Developers can select Backbone.js when you require:

- Single-page applications (SPA).
- MVC architectural pattern is preferred.
- A scalable and easily maintainable apps which needs wealth of plugins and extensions.

[2]Source: http://backbonejs.org/

Comparison Summary

The following table summarizes some of the features supported by the four HTML5 application development frameworks.

Framework	KnockOut.js	Kendo UI	Sencha Touch	Backbone.js
MVVM Supported	Yes	Yes	No	No
MVC Supported	No	No	Yes	Yes
Open Source	Yes	No	Yes	Yes
Offline Supported	Yes	Yes	Yes	Yes
Ability to Customize UI Widgets	Not Applicable	Medium	Low	Not Applicable
Browsers Supported	IE 6+, Firefox 2+, Chrome, Opera, Safari	Internet Explorer 7+, Google Chrome, Firefox ESR+, Safari 5+ (OS X), Opera 11+	IE 10 for Windows phone 8, Chrome, Safari	IE 6+, Firefox 2+, Chrome, Opera, Safari
Device Platforms	Any device in which the chosen UI framework is supported	Android 2.2+, iOS 4.0+, BlackBerry OS 7.0+	Android 2.0+, iOS 3+, BB 6+, BB 10	Any device in which the chosen UI framework is supported
Folder Structure Specified	No	No	Yes	Yes
User Experience	Good	Rich	Rich	Good
Native Packaging	Not Available	Yes (With cordova using Icenium IDE)	Yes	Not Available
PhoneGap Support	Yes	Yes	Yes	Yes
Templating	Yes	Yes	Yes	Yes
Integrated UI framework	No	Yes	Yes	No
Unit Testing Framework Support	Yes	Yes	Yes	Yes

Advantages	– Any JavaScript based UI framework can be used which makes it highly customizable – IDE supported – Provides guidelines for structure – Good UX/UI, based on Knockout – Separation of view and data	– Any UI possible – Easy Unit testing – Separation of view and data	– Free version available – Rich UX/UI – Partly structured – Easy to maintain Apps – Unit testing supported	– Any UI possible – Easy Unit testing – Separation of view and data
Limitations	– No folder structure enforced – Cannot be used without the help of another UI framework	Paid model of licens-ing	Difficult to integrate with other frameworks	– No folder structure enforced – Cannot be used without the help of another UI framework

Conclusion

There are two technology trends which are increasingly growing; they are mobile application development and standards-based HTML5 web development. Developing a native mobile application requires knowledge of specific platforms and skills, which include Java for Android and Objective-C for iPhone. HTML5 development has gained traction lately because it is standards-based. Many app development framework vendors are working to incorporate and comply with these early specifications.

Different frameworks work for different developers. When adopting a framework for mobile web app development, developers should look for two main parameters – a framework which is developer-friendly, and which has a wide reach.

Among all the frameworks discussed in this paper, if you need a framework to build complex mobile web app with highly customizable features, then you should select Knockout.js. If you need to develop an application which is more native-like and you require an out-of-box solution which is affordable, you should choose Kendo UI. Both of these frameworks are based on MVVM architectural patterns.

Sencha Touch is the right choice when you need to develop native-like app with graphical representation of data. Whereas, Backbone.js is recommended when you want to develop a scalable and easily maintainable app and which needs wealth of plugins and extensions. Both of these frameworks are based on MVC architectural pattern.

RapidValue has a team of domain experts and mobility consultants to help you build innovative and comprehensive mobile applications for your enterprise. If you'd like more information on this topic or need guidance on building your first mobile application, please write to marketing@rapidvaluesolutions.com, we'll be happy to hear from you.

About RapidValue

RapidValue is a leading provider of mobility solutions to enterprises worldwide. Armed with a team of 175+ experts in mobility consulting and application development, along with experience delivering over 200 mobility projects, we offer a range of mobility services across industry verticals. RapidValue delivers its services to the world's top brands and Fortune 1000 companies, and has offices in the United States and India.

www.rapidvaluesolutions.com

www.rapidvaluesolutions.com/blog

+1 877.690.4844

contactus@rapidvaluesolutions.com