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 Frameworks

2.1 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, BlackBerry and Mozilla [**?**].

Omkadering

Programmeertaal jQM is markup driven framework

Tools Een basis teksteditor voldoet om met jQuery Mobile aan de slag te kunnen. Natuurlijk kan het gemakkelijk zijn om van *integrated development environments* (IDE's) zoals Aptana Studio [**?**] of WebStorm [**?**] gebruik te maken, waardoor je handige kenmerken krijgt zoals *code completion*.

Je kan ook gebruiken maken van Codiqua om via *drag-and-drop* UI elementen op je scherm te slepen. Codiqua zal automatisch op de achtergrond de HTML code voorzien [**?**].

Documentatie Documentatie is te vinden op www.jquerymobile.com/demos/1.2.0. Hierop is een catalogus te vinden van alle mogelijke elementen waarover jQuery Mobile beschikt. Door de broncode van een voorbeeld te bekijken, kan je zien welke code je moet schrijven om tot dat resultaat te komen.

Naast de UI elementen is er ook documentatie over de API. Deze gaat over initile configuraties, events en methodes die kunnen worden gebruikt.

Marktadoptatie Als we kijken op de website van jQuery Mobile zien we een reeks applicaties gemaakt met hun raamwerk. Enkele voorbeelden zijn webapplicaties voor Ikea, Disney World, Stanford University en Moulin Rouge [**?**].

Licenties Vanaf september 2012 is het enkel nog mogelijk om jQuery Mobile onder de Massachusetts Institute of Technology (MIT) licentie te verkrijgen [**?**]. Dit betekent dat de code wordt vrijgegeven als *open-source* en dat deze tegelijkertijd kan worden gebruikt in propritaire projecten en applicaties [**?**].

Code en ontwikkeling
Zoals werd aangehaald, schrijft men vooral HTML5 code voorzien van `data-*` attributen. Daarna zal het raamwerk door middel van *progressive enhancement* allerhande code toevoegen om de beoogde UI elementen correct te tonen in de browser. Dit wordt verder uitgelegd in de sectie browserondersteuning (zie 2.1).

Er zijn drie strategien om webapplicaties te maken in jQuery Mobile [**?**]. Een eerste is om de volledige applicatie in n webpagina te schrijven. Met andere woorden, de vele schermen van de webapplicatie zijn dan allemaal samengebracht op eenzelfde webpagina. Het voordeel bij deze aanpak is dat er initieel minder verzoeken zijn naar de server omdat alles in n bestand wordt opgehaald. Dit geldt ook zo voor de gemporteerde CSS en JavaScript-bestanden.

Een tweede strategie is om voor ieder scherm een aparte webpagina aan te maken. Het voordeel hierbij is dat de eerste pagina waar de gebruiker op terecht komt, sneller wordt gedownload. Bij iedere navigatie naar een ander scherm, moet dit scherm via AJAX worden opgehaald, waardoor dit vertragend kan werken.

Een laatste strategie is om een mix tussen beide te maken. Men kan bijvoorbeeld alle schermen die de gebruiker vaak nodig heeft op n webpagina plaatsen. De schermen die de gebruiker zelden nodig heeft, plaats men dan op aparte webpagina's.

Functionele kenmerken
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 [**?**].

Niet-functionele kenmerken

Performantie Zoals gezegd schrijft de ontwikkelaar HTML5 code met specifieke data attributen en zal het raamwerk daarna de code verder aanvullen. Dit gebeurt enkel op de pagina die de gebruiker op dat moment bekijkt. Dit gaat dus ook op voor een webapplicatie waarbij alle schermen op n webpagina zijn geschreven. Deze webpagina bevat allemaal `<div>`-verpakkingen voor ieder scherm. jQuery Mobile zal enkel die `<div>` verder aanvullen die op dat moment getoond wordt aan de gebruiker.

Aanpasbaarheid Als je jQuery Mobile *out-of-the-box* gebruikt, zit alles al goed qua kleur en design. Je hebt de keuze uit vijf kleurenthema's die je kan toepassen op de gehele applicatie of enkel op bepaalde elementen. Om je applicatie echt te laten onderscheiden van de andere, zal je natuurlijk graag je eigen kleurthema willen toepassen. Hier is jQuery Mobile op voorzien door hun *stylesheet* op te delen in twee delen: thema's en structuur. Je kan als ontwikkelaar ook enkel de structuur downloaden en zelf de thema CSS schrijven. Daar dit laatste heel wat inspanning vraagt, hebben de ontwikkelaars van jQuery Mobile ook een tool ter beschikking, namelijk ThemeRoller [**?**]. Hier kan je zeer eenvoudig kleuren slepen naar een voorbeeldapplicatie. Eenmaal tevreden kan je de overeenkomstige *stylesheet* downloaden en toevoegen aan je project.

Programmeerbaarheid Bij het programmeren in jQuery Mobile wordt geen enkel ontwerppatroon afgedwongen. De code voor de UI elementen wordt tenslotte als HTML5 code geschreven. Voor de echte functionaliteit wordt beroep gedaan op JavaScript en meer bepaald op de jQuery Core bibliotheek. Ook deze dwingt geen ontwerppatroon af.

Browserondersteuning jQuery Mobile deelt browsers op in drie verschillende klassen: A, B en C [**?**]. Hierbij ondersteunt een klasse A browser alles, terwijl een klasse C browser enkel de basis HTML ondersteunt (en dus bijvoorbeeld geen hippe CCS3 overgangen).

Er dient een onderscheid te worden gemaakt tussen de begrippen *progressive enhancement* en *graceful degradation* [**?**]. Het eerste is wat jQuery Mobile toepast, namelijk starten met de basis HTML. Deze code wordt door iedere browser, dus ook deze uit de C klasse, op een goede manier weergegeven. Daarna zal het iteratief elementen toevoegen tot het op een moment komt dat de betreffende browser een bepaald kenmerk niet meer ondersteund.

De tegenhanger is *graceful degradation*. Hierbij wordt eerst een versie ontwikkeld die enkel in de meest recentste browser kan worden getoond. Daarna, als de ontwikkelaar nog tijd heeft, gaat hij *fallbacks* implementeren waardoor minder recente browser de applicatie ook kunnen weergeven.