

Vergelijkende studie van raamwerken voor de ontwikkeling van mobiele HTML5 applicaties

Tim Ameye
Sander Van Loock

Thesis voorgedragen tot het behalen
van de graad van Master of Science
in de ingenieurswetenschappen:
computerwetenschappen

Promotor:

Prof. dr. ir. E. Duval

Assessoren:

Ir. W. Eetveel

W. Eetrest

Begeleiders:

Ir. A. Assistent

D. Vriend

© Copyright KU Leuven

Zonder voorafgaande schriftelijke toestemming van zowel de promotor als de auteurs is overnemen, kopiëren, gebruiken of realiseren van deze uitgave of gedeelten ervan verboden. Voor aanvragen tot of informatie i.v.m. het overnemen en/of gebruik en/of realisatie van gedeelten uit deze publicatie, wend u tot het Departement Computerwetenschappen, Celestijnenlaan 200A bus 2402, B-3001 Heverlee, +32-16-327700 of via e-mail info@cs.kuleuven.be.

Voorafgaande schriftelijke toestemming van de promotor is eveneens vereist voor het aanwenden van de in deze masterproef beschreven (originele) methoden, producten, schakelingen en programma's voor industrieel of commercieel nut en voor de inzending van deze publicatie ter deelname aan wetenschappelijke prijzen of wedstrijden.



Voorwoord

Dit is mijn dankwoord om iedereen te danken die mij bezig gehouden heeft. Hierbij dank ik mijn promotor, mijn begeleider en de voltallige jury. Ook mijn familie heeft mij erg gesteund natuurlijk.

Tim Ameye
Sander Van Loock

Inhoudsopgave

Voorwoord	i
Samenvatting	iv
Lijst van figuren	v
Lijst van tabellen	vi
Lijst van afkortingen	vii
1 Inleiding	1
2 Literatuurstudie	3
2.1 Mobiele apparaten	3
<i>Soorten</i> 3, <i>Kenmerken</i> 4	
2.2 Mobiele besturingssystemen	6
<i>iOS</i> 6, <i>Android</i> 6, <i>Windows Phone</i> 7	
2.3 Mobiele applicaties	7
<i>Webapplicaties</i> 8, <i>Native applicaties</i> 8, <i>Hybride applicaties</i> 8	
2.4 Mobiele webbrowsers	8
<i>Mobile Safari</i> 9, <i>Android browser</i> 9, <i>Internet Explorer Mobile</i> 10, <i>Opera Mobile/Mini</i> 10	
2.5 HTML5, CSS3 en JavaScript	10
<i>HTML5</i> 10, <i>CSS3</i> 12, <i>JavaScript</i> 14	
2.6 Mobiele HTML5 raamwerken	14
2.7 Vergelijken van raamwerken	14
<i>ISO 25010</i> 15, <i>Bestaande use cases</i> 15, <i>Vergelijkingstabellen</i> 16	
3 Vergelijking	17
3.1 POC	17
3.2 Vergelijkingscriteria	17
<i>Omkadering</i> 17, <i>Productiviteit</i> 17, <i>Gebruik</i> 17, <i>Ondersteuning</i> 17, <i>Performantie</i> 17	

4	Mobiele HTML5 raamwerken	19
4.1	jQuery Mobile	19
	<i>Omkadering</i> 19, <i>Code en ontwikkeling</i> 20, <i>Functionele kenmerken</i> 20, <i>Niet-functionele kenmerken</i> 21	
4.2	Sencha Touch	22
	<i>Omkadering</i> 22, <i>Code en ontwikkeling</i> 24, <i>Functionele kenmerken</i> 24, <i>Niet-functionele kenmerken</i> 25	
4.3	Kendo UI	27
	<i>Omkadering</i> 27, <i>Code en ontwikkeling</i> 27, <i>Functionele kenmerken</i> 27, <i>Niet-functionele kenmerken</i> 27	
4.4	Lungo	27
	<i>Omkadering</i> 27, <i>Code en ontwikkeling</i> 27, <i>Functionele kenmerken</i> 27, <i>Niet-functionele kenmerken</i> 27	
5	Evaluatie	29
5.1	Omkadering	29
5.2	Productiviteit	29
5.3	Gebruik	29
5.4	Ondersteuning	35
5.5	Performantie	35
6	Besluit	37
	Bibliografie	39




Samenvatting

In dit **abstract** environment wordt een al dan niet uitgebreide samenvatting van het werk gegeven. De bedoeling is wel dat dit tot 1 bladzijde beperkt blijft.



Lijst van figuren

2.1	Resoluties van bekende mobiele apparaten [53].	5
2.2	Marktaandeel iOS-besturingssystemen op 14 november 2012 [49].	6
2.3	Marktaandeel Android besturingssystemen op 1 november 2012 [2].	7
2.4	HTML5e mobiele ondersteuning [11]	13
4.1	Sencha Touch Kitchen Sink opstarttijden [45].	26



Lijst van tabellen

2.1	Marktaandeel mobiele webbrowsers op november 2012 [28].	9
-----	---	---



Lijst van afkortingen

Afkortingen

AJAX	Asynchronous JavaScript And XML
API	Application Programming Interface
CSS	Cascading Style Sheets
DOM	Document Object Model
GWT	Google Web Toolkit
GPU	Graphics Processing Unit
(G)UI	(Graphical) User Interface
HTML	HyperText Markup Language
IDE	Integrated Development Environment
JSON	JavaScript Object Notation
PDF	Portable Document Format
PPI	Pixels Per Inch
RIA	Rich Internet Application
SASS	Syntactically Awesome Stylesheets
SDK	Software Development Kit
SEO	Search Engine Optimization
XML	Extensible Markup Language

Inleiding

In dit hoofdstuk wordt het werk ingeleid. Het doel wordt gedefinieerd en er wordt uitgelegd wat de te volgen weg is (beter bekend als de rode draad).

Literatuurstudie

Eerst zullen we kijken naar welke mobiele apparaten er allemaal bestaan (2.1). Vervolgens zullen we kijken wat er onder de motorkap van deze apparaten zit, namelijk welke mobiele besturingssystemen (2.2), welke mobiele applicaties (2.3) en welke mobiele webbrowsers (2.4) er bestaan. Daarna zullen we het hebben over de drie bouwblokken van het web (2.5): HTML, CSS en JavaScript. Hierna zullen we het hebben over mobiele HTML5 raamwerken (2.6). Ten slotte kijken we naar verschillende manieren om raamwerken te vergelijken (2.7).

2.1 Mobiele apparaten

Mobiele apparaten vind je in alle soorten en maten, met weinig of veel opties, voor weinig of veel geld. Het verdient daarom de aandacht om deze diversiteit onder de loep te nemen. Eerst zullen we de soorten mobiele apparaten bekijken volgens [?] en daarna zullen we ingaan op de kenmerken volgens [35].

2.1.1 Soorten

Sinds de voorstelling van de Apple iPhone in 2007 [5], stijgt het gebruik van de *smartphone* ontzettend snel in onze samenleving. Momenteel zijn er al meer dan 1 miljard toestellen in gebruik [54]. Foto's of video's nemen, navigeren naar het dichtstbijzijnde restaurant of nog snel het weer voor de komende dagen opzoeken, het is allemaal mogelijk. Hoewel Apple de lat hoog heeft gelegd met het uitbrengen van de iPhone, zijn er ook nog andere spelers op de markt. Zo hebben we bijvoorbeeld ook de op Google's Android gebaseerde *smartphones* zoals de Nexus 4 en de op Windows Phone gebaseerde *smartphones* zoals de Nokia Lumia 800.

Niet enkel de *smartphone* behoort tot de categorie van mobiele apparaten, maar ook de *tablet*. Ook hier kan terug gedacht worden aan één van Apple's succesvolle producten, namelijk de in 2010 uitgebrachte iPad [?]. Er dient echter wel opgemerkt te worden dat tien jaar voordien, Microsoft al eerder een *tablet* uitbracht met veel minder succes [?].

De *e-reader* behoort tot de laatste categorie van mobiele apparaten. Deze wordt hoofdzakelijk gebruikt om digitale boeken te lezen, maar betere modellen laten

bijvoorbeeld ook toe om te surfen op het Internet. Ook hier bestaat er een variëteit aan modellen zoals de Kindle van Amazon en de Reader van Sony.

2.1.2 Kenmerken

Door de vele verschillende soorten en modellen aan mobiele apparaten, is het nodig om op een hoog niveau te bekijken over welke kenmerken deze allemaal (kunnen) beschikken. Bij deze bespreking zullen we ingaan op de voornaamste kenmerken van *smartphones* en *tablets*. De kenmerken en tekst zijn gebaseerd op [35].

Resolutie en PPI

Een eerste kenmerk, waar vooral Apple met haar Retina graag mee uitpakt, is de resolutie. Dit is het aantal pixels getoond op het beeldscherm en wordt uitgedrukt in breedte \times hoogte. Hoe kleiner, hoe minder er op het scherm kan worden getoond. Dit is vooral belangrijk wanneer veel informatie op het scherm wordt getoond. Indien men maar over een kleine resolutie beschikt, zal men moet scrollen om te rest van de informatie te kunnen zien. Een overzicht van resoluties van bekende mobiele apparaten wordt getoond op de figuur 2.1.

Als men naast de resolutie ook nog eens gaat rekening houden met de fysieke grootte van het scherm, dan kunnen we spreken van over pixels per inch (PPI). De eerste iPhone had een resolutie van 320×480 en een 3,5" scherm, wat neerkomt op 163 PPI. De iPhone4 (Retina) daarentegen heeft een resolutie van 640×960 en een 3,5" scherm, wat neerkomt op 326 PPI. Met andere woorden zijn er meer pixels op dezelfde fysieke grootte geplaatst, wat een scherper beeld tot resultaat heeft.

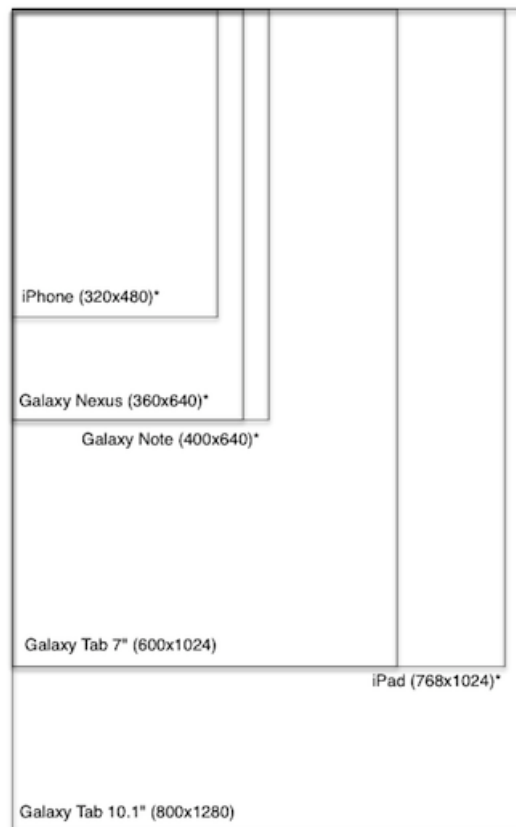
Aanraakscherm

De populaire soorten schermen zijn resistieve en capacitieve aanraakschermen. De eerste soort maakt gebruik van twee lagen die gescheiden worden door een tussenruimte. Door druk ontstaat er contact tussen de twee lagen. Meestal wordt bij deze soort schermen een stylus meegeleverd.

De tweede soort maakt gebruik van veranderingen in frequentie. Door het scherm aan te raken met je vinger, dat een geleider is, ontstaat er een kleine verandering in frequentie die gedetecteerd wordt. Niet-geleidende materialen zullen geen frequentieverandering veroorzaken, wat verklaart dat zo'n scherm niet reageert als je het aanraakt met een handschoen.

GPS

Met het *global positioning system* (GPS) kan de gebruiker zijn locatie opvragen en doorgeven aan een applicatie om zo bijvoorbeeld het dichtstbijzijnde restaurant te vinden. Doordat het wat kan duren vooraleer de locatie is vastgesteld via GPS, kan het mobiel apparaat ook gebruik maken van mobiele masten of het Internet om zo, hetzij minder nauwkeurig, sneller de locatie te bepalen.



FIGUUR 2.1: Resoluties van bekende mobiele apparaten [53].

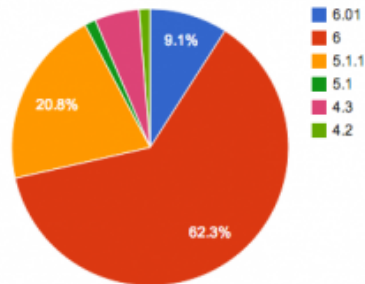
Camera

Praktisch ieder recent mobiele apparaat is uitgerust met een camera. Sommige bevatten zelfs twee camera's. De camera vooraan is veelal van mindere kwaliteit en wordt gebruikt om videogesprekken te voeren. Achteraan het apparaat zit dan een camera met hogere resolutie om mooie foto's te kunnen maken.

Twee andere toepassingen van de camera zijn toegevoegde realiteit en het inscannen van barcodes. Bij het eerstgenoemde wordt informatie toegevoegd aan het beeld dat door de camera wordt geregistreerd. Het laatstgenoemde wordt gebruikt om de populaire QR-code in te scannen en te zien wat ze betekent. Zo'n code kan tekst bevatten, een link naar een website, een telefoonnummer, enzovoort.

Verbinding

In deze periode wil iedereen met elkaar verbonden zijn, dus ook op mobiele apparaten. We bespreken kort Wifi, 3G, Bluetooth en infrarood. Het mobiel apparaat kan meerdere mogelijkheden voorzien om verbinding te maken. Enerzijds kan men verbinden via Wi-Fi. Daarnaast zijn er ook nog andere technologieën zoals 3G



FIGUUR 2.2: Marktaandeel iOS-besturingssystemen op 14 november 2012 [49].

mogelijk.

2.2 Mobiele besturingssystemen

Net zoals er brede waaier bestaat aan besturingssystemen voor computers, geldt dit ook zo voor mobiele apparaten. We geven hier een overzicht van mobiele besturingssystemen met een significant marktaandeel [5, 11] zoals iOS en Android, maar ook een nieuwkomer op de markt, namelijk Windows Phone.

2.2.1 iOS

Het iPhone besturingssysteem is voor het eerst uitgekomen in juni 2007 tezamen met de iPhone. Later werd het hernoemd naar iPhone OS en uiteindelijk werd het iOS. Het is duidelijk dat iOS gebonden is aan de hardware van Apple. Verschillende versies volgden elkaar op: iOS 2 (juli 2008), iOS 3 (juni 2009), iOS 4 (juni 2010) en iOS 5 (oktober 2011) [6, 35].

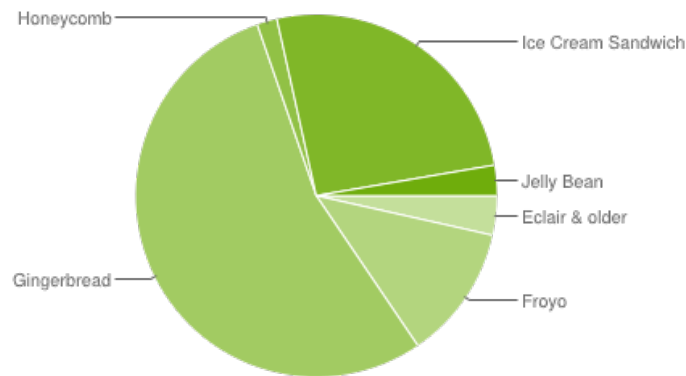
De nieuwste versie, iOS 6, werd uitgegeven in september 2012. Nieuwigheden zijn onder andere hun eigen Maps (in plaats van Google Maps) en een Pass Kit (de vervanging van het traditionele trein-, cinematicket, enz.). Daarnaast zijn er ook ander andere verbeteringen uitgevoerd met betrekking tot sociale media en spraakcommando's [6].

Op figuur 2.2 is te zien dat bijna twee derde van de iOS-gebruikers al iOS 6 gebruikt.

Browsen op het web gebeurt met de geïnstalleerde Mobile Safari webbrowser (zie 2.4.1). Applicaties kunnen gedownload worden in de App Store, die sinds iOS 2 aanwezig is [6].

2.2.2 Android

Android Inc. werd opgericht in 2003 en werd in 2005 overgekocht door Google Inc [39]. Het is net zoals iOS een mobiel besturingssysteem, maar in tegenstelling tot



FIGUUR 2.3: Marktaandeel Android besturingssystemen op 1 november 2012 [2].

iOS is het open [5]. De eerste stabiele versie, Android 1.0, kwam uit in september 2008. Ook hier volgden verschillende versies elkaar op: Android 2.0 (oktober 2009), Android 3.0 (februari 2011) en Android 4.0 (oktober 2011) [39]. Hun nieuwste versie, Android 4.2, werd aangekondigd in oktober 2012 [40].

Op figuur 2.3 is het marktaandeel te zien van de verschillende Android besturingssystemen, waargenomen over een periode van 14 dagen. Het is duidelijk dat Gingerbread (Android 2.3) meer dan de helft van het marktaandeel inneemt. Applicaties worden gedownload in Google Play. Android bevat ook een standaard browser (zie 2.4.2).

2.2.3 Windows Phone

Windows Phone van Microsoft werd aangekondigd in oktober 2010 als vervanging voor Windows Mobile [41, 23]. Dit is duidelijk te zien als we kijken naar de versies: de laatste versie was Windows Mobile 6.5.3 en de eerste versie is Windows Phone 7. In 2011 ging Microsoft een partnerovereenkomst aan met Nokia om zo snel de markt te kunnen overwinnen [26]. De nieuwste versie, Windows Phone 8, werd aangekondigd in oktober 2012 [36].

2.3 Mobiele applicaties

Er zijn drie mogelijkheden om mobiele applicaties te maken [1, 11]. Eén aanpak is het maken van een webapplicatie. Zo'n applicatie wordt geopend vanuit de webbrowser. Een andere aanpak is een *native* applicatie. Hierbij zal de gebruiker de applicatie installeren op zijn apparaat. Als laatste kan een mix van de vorige gemaakt worden en dat wordt een hybride applicatie genoemd.

2.3.1 Webapplicaties

In het rapport 'The (Not So) Future Web' [34] uit juni 2011 wordt gesteld dat tegen 2015 60% van alle mobiele bedrijfsapplicaties en 40% van alle mobiele consumentenapplicaties, webapplicaties zullen zijn. Er zijn namelijk veel voordelen [1] verbonden aan webapplicaties.

Ten eerste heeft iedereen die een webbrowser heeft op zijn mobiel apparaat, toegang tot de applicatie. Dit voordeel gaat niet op voor een native applicatie dat enkel voor een specifiek platform is geschreven.

Ten tweede, aansluitend bij het bovenstaande voordeel, moet de code slechts eenmaal worden geschreven. Een vaak voorkomende term die dit samenvat is WORA: *write once, run anywhere* [11]. Dit is in tegenstelling tot een native applicatie die specifiek geschreven is voor bijvoorbeeld iOS, Android en Windows Phone. Daar dient de code driemaal te worden geschreven én te worden onderhouden.

Ten derde moeten webapplicaties niet worden geverifieerd vooraleer ze worden uitgebracht. Dit is wel zo bij native applicaties. Hierdoor kan in een webapplicatie een belangrijke update snel doorgevoerd worden, terwijl de native applicatie nogmaals het verificatieproces moet doorlopen.

2.3.2 Native applicaties

Een andere mogelijkheid is om een native applicatie te schrijven. Voordelen [1] hier zijn onder meer de snelheidswinst doordat de applicatie rechtstreeks met het besturingssysteem kan werken. Aansluitend bij het vorige kan ook worden geargumenteed dat het over het algemeen een native applicatie gemakkelijker de kenmerken van het mobiel apparaat, zoals de camera of GPS, aan kan spreken. Ten derde blijft beveiliging nog altijd een knelpunt bij webapplicaties. Een native applicatie heeft hier minder problemen. Als laatste kan opgemerkt worden dat het gebruik van een winkel (*store*) voor het aanbieden van een applicatie als voordeel kan gezien worden, afgezien van het verificatieproces. De applicatiewinkel zorgt namelijk voor reclame en correcte uitbetaling bij gebruik van de applicatie.

2.3.3 Hybride applicaties

Er bestaat een mix tussen de twee voorgaande soorten van mobiele applicaties, namelijk een hybride applicatie [1]. Hierbij wordt de webapplicatie verpakt in een native applicatie. Hierdoor kan men specifieke kenmerken van het mobiel apparaat benaderen die men vanuit een pure webapplicatie niet kon benaderen.

2.4 Mobiele webbrowsers

Sinds 2008 spreken we van het mobiele web [11]. Vanuit mobiele webbrowsers op tablets en smartphones wordt het web meer en meer aangesproken. Deze mobiele webbrowsers vormen als het ware kleine besturingssystemen, waardoor de browser zelf een platform wordt [11]. Ze geven namelijk toegang tot allerlei kenmerken van het

Mobile webbrowser	Marktaandeel (%)
Mobile Safari	61.50
Android browser	26.09
Opera Mini	7.02
Chrome	1.14
Opera Mobile	0.53
Internet Explorer Mobile	
Andere	

TABEL 2.1: Marktaandeel mobiele webbrowsers op november 2012 [28].

mobiele apparaat zoals camera en GPS. Denk maar aan het heel concreet voorbeeld van Google die het besturingssysteem Chrome OS maakte op basis van de Chrome webbrowser [11].

Vanuit het standpunt om webapplicaties te maken, is het dan ook zeer belangrijk om deze evolutie op te volgen. Een webbrowser haalt namelijk webpagina's op die geschreven zijn in HTML en andere technologieën. Doordat deze technologieën evolueren (zie 2.5), zullen de webbrowsers zelf ook (moeten) mee evolueren. Niet iedere browser zal dit op dezelfde manier doen, waardoor er verschillen zullen ontstaan waar men rekening mee zal moeten houden. Het is namelijk ongewenst dat een webapplicatie enkel op Mobile Safari werkt als men een zo breed mogelijk publiek wenst te bereiken.

Hieronder bespreken we enkele mobiele webbrowsers. Eerst halen we de twee meest populaire browsers aan, namelijk Mobile Safari en de native Android browser [11]. Ze zijn beide op de WebKit browser *engine* gebaseerd [29]. Zo'n *engine* zorgt ervoor dat de code van de opgehaalde webpagina wordt omgezet naar de webpagina die de gebruiker te zien krijgt. We bekijken ook kort Internet Explorer Mobile en Opera Mobile. Het marktaandeel van de genoemde browsers kunt u zien in tabel 2.1.

2.4.1 Mobile Safari

Deze webbrowser van Apple zit standaard bij iOS en kan ook enkel op dit besturingssysteem worden gebruikt. Apple heeft veel moeite gedaan om telkens de laatste nieuwe specificaties van HTML5 in zijn webbrowsers te implementeren [11]. Natuurlijk zal dit ook te maken hebben met het feit dat ze geen Flash meer ondersteunen op hun iPods, iPhones en iPads [15].

2.4.2 Android browser

Android biedt de native Android browser aan. Implementatie van de HTML5 specificaties hebben wat aangesleept, maar vanaf Android 4.0 gaat dit een stuk beter [11]. Daarnaast is het nu ook mogelijk om de Chrome webbrowser op mobiele apparaten te installeren.

2.4.3 Internet Explorer Mobile

Net zoals je bij Windows ook Internet Explorer krijgt, geldt dit ook voor hun mobiel besturingssysteem. Bij de nieuwe Windows Phone 8 zal Internet Explorer Mobile 10 worden meegeleverd. Deze gebruikt dezelfde *engine* als Internet Explorer 10. *WebSockets*, *Web Workers*, *Application Cache* en *IndexedDB* worden hierin ondersteund [11], meer daarover in 2.5.

2.4.4 Opera Mobile/Mini

Op het moment van schrijven is Opera Mobile 12.10 de beste mobiele HTML5 browser [46]. Opera heeft eigenlijk twee aparte browsers, namelijk Opera Mobile en Opera Mini. Bij deze laatste staat de browser *engine* op servers van Opera, waardoor het niet het mobiel apparaat is die de webpagina verwerkt. De server zal, na verwerking, deze webpagina op een gecomprimeerde manier doorsturen naar de browser op het apparaat [35].

2.5 HTML5, CSS3 en JavaScript

De drie bouwstenen voor webontwikkeling zijn HTML5, CSS3 en JavaScript. HTML5 is verantwoordelijk voor de inhoud, CSS3 voor de presentatie en JavaScript voor de functionaliteit [35]. Hieronder zullen we dan ook deze bouwstenen toelichten.

2.5.1 HTML5

Zoals uitgelegd in [24] stopte in 1998 het W3C (World Web Consortium) met het werken aan de HTML standaard en alle energie ging uit naar zijn opvolger: XHTML 1.0, een verbeterde HTML versie die XML-gedreven is. XHTML kwam in grote mate overeen met HTML, maar de syntax was veel strikter. In het begin kon het zijn naam waarmaken en webontwerpers helpen betere resultaten te boeken doordat ze slechte gewoontes moesten opgeven. Jammer genoeg bleven de beloofde voordelen uit. Wat veel erger was voor de nieuwe standaard, was dat geen enkele browser klaagde indien deze strikte syntax niet werd gevolgd.

In [24] staat ook de reactie die hierop kwam van het W3C. Ze brachten een nieuwe versie uit, namelijk XHTML 2. De manier waarop webpagina's werden geschreven veranderde doordat vele tags waren veranderd of verwijderd. Daarenboven sleepte deze nieuwe standaard maar aan en aan, wat ook niet in hun voordeel was.

In plaats van te onderzoeken wat er mis was met HTML, wat XHTML probeerde te doen, werd in 2004 onderzocht wat er ontbrak. Opera Software, Mozilla Foundation en Apple vormden de WHATWG (Web Hypertext Application Technology Working Group). Ze wilden HTML niet vervangen, maar uitbreiden en die manier moest achterwaarts compatibel zijn. Na reflectie geloofde ook het W3C in deze aanpak, weliswaar op hun eigen manier. Zo werd HTML5 geboren, waarbij versie 5 refereert naar waar de vorige versie, HTML 4.01, gestopt was.

HTML5 is volgens [24] nog altijd in ontwerp. Hierdoor kunnen nieuwe kenmerken op ieder momenten worden toegevoegd. Er is ook nog steeds onduidelijkheid waar HTML5 ons zal brengen. Het W3C focust op een unieke HTML5 standaard (verwacht rond 2014) terwijl WHATWG de nieuwe markup-taal ziet als levende taal waarbij voortdurend nieuwe dingen kunnen worden toegevoegd. Een belangrijke opmerking hierbij is dat het laatste woord altijd bij de webbrowserfabrikant ligt, net zoals dat het geval was met de strikte syntax in XHTML. Als een kenmerk niet in de browser wordt ondersteund, heeft het ook geen kans op overleven.

Drie basisprincipes

Achter HTML5 zit een filosofie die in drie basisprincipes kan worden samengevat [24]. De eerste is achterwaartse comptabiliteit. De standaard mag geen veranderingen invoeren die oudere pagina's zou doen breken. Ten tweede moet de standaard geen nieuwe specificaties afdwingen die door de meerderheid op een andere manier worden gedaan. Als laatste moeten de specificaties ook een praktisch nut hebben. Dit betekent dat daar waar veel vraag naar is, ook het beste opweegt om in de specificaties op te nemen.

Acht technologieklassen

HTML5 kan ook bekeken worden als de volgende acht technologieklassen [50]. Iedere klasse wordt met enkele concrete voorbeelden aangehaald.

Multimedia De nieuwe video- en audiotags maken het mogelijk om video- en geluidsfragmenten toe te voegen zonder gebruik te maken van plug-ins van derden zoals Adobe Flash en Microsoft Silverlight.

Offline en opslag Mobiele apparaten zijn onstabiel in hun verbinding met het Internet. HTML5 voorziet het offline werken in de *cache*, lokale opslag (vroeger kon men enkel gebruik maken van de zogenaamde cookies) en een API om bestanden te manipuleren.

Performantie en integratie *Web Workers* maken het mogelijk om langdurige JavaScript taken in de achtergrond uit te voeren zodat webapplicaties dynamisch en snel blijven.

Semantiek Een hele hoop nieuwe tags zorgen voor meer semantiek binnen webpagina's. Waar voorheen de webpagina bestond uit een verzameling `<div>`-elementen, kan nu veel concreter worden aangegeven wat er precies binnen die tags staat. Dit kan voor *search engine optimization* (SEO) een grote impact hebben. Daarnaast biedt dit ook mogelijkheden voor *e-readers* die nu beter de pagina kunnen analyseren.

CSS3 Hand in hand met HTML5 gaat CSS3 (zie 2.5.2). Het laat toe om webpagina's op te maken afhankelijk van het formaat van het mobiele apparaat. Ook kunnen webpagina's met effecten worden uitgebreid.

3D, grafieken en effecten De nieuwe `<canvas>`-tag in samenwerking met enkele lijnen JavaScript zijn enorm krachtig om eenvoudig tekeningen en animaties zelf te programmeren.

Verbinding *Events* aan server zijde kunnen data naar *WebSockets* pushen. Hierdoor moet de webpagina niet meer voortdurend de server raadplegen, wat veel efficiënter is.

Toegang tot het apparaat Webapplicaties kunnen meer en meer kenmerken zoals camera en GPS aanspreken net zoals native applicaties dat kunnen.

Er dient opgemerkt te worden dat aangehaalde klassen zoals CSS3 en geolocatie niet tot de specificaties van HTML5 behoren. Toch worden ze onder de koepel van HTML5 gezien [24].

Kenmerken detecteren en opvullen

KENMERKEN DETECTEREN Door enerzijds de levendigheid van HTML5 en anderzijds het verdeelde landschap van browsers en besturingssystemen, worden niet alle kenmerken van HTML5 overal ondersteund. Een eerste mogelijkheid is om zelf op te zoeken welke kenmerken op welke apparaten werken. Dat kan je bijvoorbeeld controleren op www.caniuse.com en www.mobilehtml5.org [24].

Wat nog handiger is, is om op het apparaat zelf te detecteren of het gewenste kenmerk beschikbaar is. Een erg handige tool hiervoor is Modernizr [27]. Het toevoegen van dit JavaScript-bestand creëert een JavaScript-object dat voor elk kenmerk teruggeeft of het al dan niet in de gebruikte browser wordt ondersteund.

KENMERKEN OPVULLEN Wanneer eenmaal gedetecteerd is dat een kenmerk niet aanwezig is, zijn er twee mogelijkheden: ofwel terugvallen op een alternatief of simuleren van dat kenmerk. Een voorbeeld van dit eerste kan gebeuren bij het gebruiken van de `<video>`-tag. Indien dit niet wordt ondersteund, kan men terugvallen op de Adobe Flash plug-in. Voor het simuleren van een kenmerk maakt men gebruik van *polyfills*. Dit zijn alternatieven op basis van JavaScript waarbij de native functionaliteit die normaal moet aanwezig zijn, geëmuleerd wordt [24, 51].

HTML5e

Een bedrijf wil enerzijds een stabiele webapplicatie en wil anderzijds ook van deze nieuwe kenmerken zoveel mogelijk gebruik gaan maken. De term HTML5e [11] omvat de vijf meest ondersteunde HTML5 kenmerken in browsers. Op figuur 2.4 vind je een tabel die voor mobiele webbrowsers van toepassing is.

2.5.2 CSS3

Hand in hand met HTML5 gaat CSS3, dat zorgt voor de presentatie. Het is namelijk het hart van webdesign. CSS3 heeft hetzelfde probleem zoals HTML5 als het aankomt op de ondersteuning bij browsers [24]. Ook hier is er dus een brede waaier

OS/API	Geolocation	WebSocket	Web Storage	Device Orientation	Web Workers
Mobile Safari	Yes	Yes	Yes	Yes	Yes
Android	Yes	No	Yes	Yes	No
Mobile IE	Yes	No	Yes	No	No
Opera Mobile	Yes	Mixed**	Yes	Mixed*	Yes
Mobile Firefox	Yes	Mixed**	Yes	Yes	Yes

FIGUUR 2.4: HTML5e mobiele ondersteuning [11]

aan kenmerken die nog niet overal worden ondersteund. Kenmerken die enkel in een bepaalde browser ondersteund worden, worden voorafgaan door een browserprefix (zoals `-webkit-` voor WebKit gebaseerde browsers en `-o-` voor Opera).

In deze sectie zullen we kort belangrijke eigenschappen bespreken zoals *media queries*, effecten en lettertypes aan de hand van [24].

Media queries

Zoals al aangehaald, hebben we verschillende apparaten met verschillende schermen en resoluties. Een goeie webpagina bestaat erin deze elementen zo goed mogelijk te benutten. Dit kan vanaf nu door gebruik te maken van *media queries* in CSS3. De website kan zich hiermee aanpassen aan het apparaat waarop het wordt getoond. Dit wordt in het Engels omschreven als *responsive design*.

Ook CSS3 volgt het principe van achterwaartse compatibiliteit. Browsers die deze *media queries* niet ondersteunen, zullen deze negeren en enkel de gewone lay-out toepassen ongeacht het toestel.

Effecten

Transparantie, afgeronde hoeken, schaduw en kleurenverloop zijn maar enkele van de nieuwe kenmerken in CSS3. Voorheen moest de webdesigner deze dingen vaak met afbeeldingen oplossen, maar nu kan dit allemaal gebeuren met CSS3. Daarnaast hebben we ook effecten als transformaties en transitities. Zo is het mogelijk wanneer men over een afbeelding gaat, deze ingezoomd en geroteerd kan worden.

Dit is zeer vooruitstrevend om wille van twee zaken. Enerzijds schrijf men dingen makkelijker in CSS dan met JavaScript-code. Anderzijds komt er ook meer en meer ondersteuning vanuit de hardware. Zo worden 3D transformaties in CSS3 versneld door de *graphics processing unit* (GPU) [11, 22].

Lettertypes

Een laatste kenmerk in CSS3 is de betere ondersteuning van lettertypes. Waar vroeger enkel gewerkt kon worden met veilige lettertypes voor het web, is het nu mogelijk om eigen lettertypes op te laden en te gebruiken op je website.

2.5.3 JavaScript

JavaScript gaat terug tot in 1995, toen LiveScript [25]. Het heeft een lange weg afgelegd tot nu en is niet altijd even ernstig genomen. Dit kwam omdat men niet inzag wat er allemaal mee kon worden gedaan.

Op dit moment is het maar al te duidelijk waar JavaScript in uitblinkt: het aanpassen van het *document object model* of kortweg DOM [35]. Dit is een API voor HTML-documenten [12]. Hierdoor kunnen dynamische interfaces gecreëerd worden, kan op gebeurtenissen - zoals ergens op klikken - onmiddellijk gereageerd worden en is de website dan ook meer bruikbaar geworden door deze directe feedback [25].

Het schrijven van JavaScript is niet gemakkelijk om twee redenen [25]. Ten eerste, vergelijkbaar met HTML5 en CSS3, kunnen browsers JavaScript op verschillende manieren interpreteren. Gelukkig is er de laatste tijd veel gestandaardiseerd, maar toch blijven er nog verschillen. De ontwikkelaar dient dus tijdens het programmeren met deze verschillen rekening te houden. Ten tweede vergt het schrijven van simpele, veel voorkomende taken soms veel code.

Een oplossing voor de bovenstaande pijnpunten is gebruik maken van een bibliotheek. Een voorbeeld hiervan is de populaire jQuery Core bibliotheek. Het is ook mogelijk om jQuery uit te breiden met verscheidene plug-ins om de functionaliteit nog te vergroten [25].

2.6 Mobiele HTML5 raamwerken

jQuery Mobile
ST

Kendo UI
Lungo
TMP
Moobile

Davinci
jQTouch

2.7 Vergelijken van raamwerken

Om de verschillende mobiele HTML5 raamwerken te kunnen vergelijken hebben we een consistente manier nodig om dit te doen. Op deze manier worden alle raamwerken

op dezelfde manier getest.

2.7.1 ISO 25010

HTML5 raamwerken zijn software en om software te vergelijken bestaat er de ISO 25010 standaard [48]. Hieronder vallen twee modellen: de productkwaliteit en de kwaliteit van het product in gebruik. Beide modellen beschrijven de kwaliteit van software op basis van een aantal categorieën met specifieke kwaliteitseigenschappen. Het beoordelen van de categorieën kan gebeuren op basis van een checklist.

Productkwaliteit

De acht karakteristieken die horen bij dit model zijn: functionele geschiktheid, betrouwbaarheid, performantie, efficiëntie, uitwisselbaarheid, bruikbaarheid, betrouwbaarheid, beveiligbaarheid, onderhoudbaarheid en overdraagbaarheid. Vanzelfsprekend zijn niet alle categorieën even toepasbaar op HTML5 raamwerken. Beveiligbaarheid is bijvoorbeeld niet zo belangrijk bij mobiele HTML5 raamwerken. Performantie en overdraagbaarheid dan weer wel.

Kwaliteit in gebruik

De vijf karakteristieken voor dit model zijn: effectiviteit, efficiëntie, voldoening, vrijheid van risico en context dekking. Elke karakteristiek kan toegewezen worden aan verschillende activiteiten van belanghebbenden. Weer zijn alle categorieën niet even toepasbaar. De risico die een mobiele webapplicatie meebrengt is niet van belang, het moet vooral efficiënt zijn en voldoening scheppen.

De kwaliteit voor een systeem in gebruik wordt bepaald door de kwaliteit van de software, de hardware en het besturingssysteem samen met de gebruikers, hun taken en de sociale omgeving. De belanghebbenden worden opgedeeld in primaire en secundaire gebruikers. De eerste zijn de personen die het systeem gebruiken. De laatste zijn diegene die zorgen voor het onderhoud.

2.7.2 Bestaande use cases

Op het web en in de literatuur kunnen we ook *use cases* terugvinden waar de proef op de som wordt genomen en twee of meer raamwerken met elkaar worden vergeleken. Deze werkwijze verschilt van *use case* tot *use case*

Codefessions

Op een blogpost van Codefessions wordt een vergelijking gemaakt tussen jQuery Mobile, Sencha Touch, jQTouch en Kendo UI [38]. Als referentiesysteem gebruiken ze zeven criteria. De eerste drie zijn de native *look-and-feel*, performantie en platform-onafhankelijke capaciteiten. Deze worden gequoteerd met een cijfer van 0 tot 5 waarbij 5 staat voor de maximale score. Kenmerken worden gequoteerd door de raamwerk met elkaar te vergelijken. Het raamwerk met de meeste kenmerken krijgt

een 5, het tweede beste een 4, enzovoort. Op een analoge manier wordt code efficiëntie en gebruiksgemak gequoteerd. Het raamwerk dat de minste lijnen code vereist, krijgt de perfecte score. Hierbij moeten wel alle bestanden gerekend worden die het raamwerk nodig heeft om functioneel te zijn. Licenties krijgen een score van 0 tot 5 waarbij 0 betekent dat het niet beschikbaar is voor een individuele ontwikkelaar en 5 dat het raamwerk *open-source* en gratis te gebruiken is. Andere factoren zoals omkadering en uitbreidbaarheid worden niet in de vergelijkingstabel opgenomen omdat ze afhangen van de interesse van de gebruiker. Ze worden echter wel bekeken.

2.7.3 Vergelijkingstabellen

Naast ISO standaarden of al bestaande use cases, kunnen we ook tabellen raadplegen die zoveel mogelijk raamwerken en zoveel mogelijk kenmerken naast elkaar zetten. Op Wikipedia creëerde men bijvoorbeeld zo'n tabel voor JavaScript raamwerken [52].

Specifiek voor mobiele HTML5 raamwerken bestaat er ook zo'n tabel, te vinden op www.markus-falk.com/mobile-frameworks-comparison-chart [10]. We zien er een matrix met als rijen de verschillende raamwerken en in de kolommen de vergelijkingscriteria. Deze laatste worden opgedeeld in compatibiliteit met het besturingssysteem, doel van de applicatie, taal voor ontwikkeling, hardware interactie, UI, licenties en andere. Deze laatste categorie bevat de criteria of er al-dan-niet een SDK beschikbaar is, encryptie ondersteund wordt en of advertenties worden ondersteund. Handig hierbij is dat de webpagina een stappenplan voorziet waarin je per categorie al je vereisten moet invullen. De resultaten zijn dan de raamwerken die compatibel zijn met je vereisten.

Vergelijking

3.1 POC

3.2 Vergelijkingscriteria

3.2.1 Omkadering

3.2.2 Productiviteit

3.2.3 Gebruik

3.2.4 Ondersteuning

3.2.5 Performantie

Mobiele HTML5 raamwerken

In dit hoofdstuk gaan we inzoomen op bestaande mobiele HTML5 raamwerken die gebruik maken van de laatste nieuwe technologieën zoals HTML5, CSS3 en JavaScript. Deze raamwerken worden gecategoriseerd volgens twee courante aanpakken [29]: opmaak-up gedreven en JavaScript gedreven. Bij een opmaak-up gedreven aanpak wordt de webapplicatie voornamelijk in HTML code geschreven. Daarentegen wordt bij een JavaScript gedreven aanpak hoofdzakelijk in JavaScript geprogrammeerd.

4.1 jQuery Mobile

jQuery Mobile is een mobiel HTML5 *user interface* (UI) raamwerk dat werd aangekondigd in 2010 [37]. In november 2011 werd versie 1.0 uitgebracht [31] en een jaar later werd in oktober versie 1.2 uitgebracht [32]. Op het moment van schrijven kwam versie 1.3 uit [?]. Het raamwerk wordt beheerd door het jQuery Project dat onder andere jQuery Core beheert en waar jQuery Mobile afhankelijk van is [19]. jQuery Mobile wordt door onder andere Adobe, BlackBerry en Mozilla gesponsord [17].

4.1.1 Omkadering

PROGRAMMEERTAAL Om met jQuery Mobile aan de slag te kunnen, heb je niets meer nodig dan kennis over HTML, CSS en JavaScript. Alle UI elementen worden geschreven in HTML en aangeduid met `data-*` attributen.

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 [3] of WebStorm [14] 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 [47].

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 initiële 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 [17].

LICENTIES Vanaf september 2012 is het enkel nog mogelijk om jQuery Mobile onder de Massachusetts Institute of Technology (MIT) licentie te verkrijgen [7]. Dit betekent dat de code wordt vrijgegeven als *open-source* en dat deze tegelijkertijd kan worden gebruikt in propriëtaire projecten en applicaties [35].

4.1.2 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 4.1.4).

Er zijn drie strategieën om webapplicaties te maken in jQuery Mobile [4]. Een eerste is om de volledige applicatie in één 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 één bestand wordt opgehaald. Dit geldt ook zo voor de geïmporteerde 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 één webpagina plaatsen. De schermen die de gebruiker zelden nodig heeft, plaats men dan op aparte webpagina's.

4.1.3 Functionele kenmerken

jQuery Mobile is een raamwerk dat voornamelijk UI elementen aanbied, met name pagina's en dialoogvensters, werkbalken, knoppen, inhoud vormgeven, elementen voor formulieren en lijsten [18].

PAGINA'S EN DIALOOGVENSTERS De basisstructuur van een pagina bestaat uit een koptekst, inhoud en voettekst. Bij het overgaan naar een andere pagina kan men kiezen uit tien overgangseffecten. Voordat deze overgang gebeurt, zal jQuery Mobile altijd eerst die pagina ophalen via AJAX en inladen in het DOM. Zo kan een soepel overgangseffect worden getoond aan de gebruiker. Daarnaast is het ook mogelijk om gelinkte pagina's op voorhand op te halen. Als laatste biedt jQuery Mobile ook dialoogvensters en pop-ups aan.

WERKBALKEN Het is mogelijk om zowel knoppen bij de koptekst als bij de voettekst te plaatsen. Bij deze laatste kunnen typisch meer knoppen geplaatst worden, bij de koptekst slechts twee. Daarnaast is het ook mogelijk om navigatiebalken te maken. Aan zowel de werk- als navigatiebalken kunnen iconen worden toegevoegd.

KNOPPEN Het is ook mogelijk om knoppen te plaatsen in het inhoud gedeelte. Ook hier is er terug een variëteit aan mogelijkheden: grote of kleine, met iconen of zonder, gegroepeerd of niet.

INHOUD VORMGEVEN De inhoud van de pagina kan worden vormgegeven door gebruik te maken van een rooster. jQuery Mobile laat roosters tot vijf kolommen toe. Daarnaast zijn er ook nog opklapbare blokken ter beschikking. Als laatste kunnen deze blokken ook samengevoegd worden tot een accordeon.

ELEMENTEN VOOR FORMULIEREN jQuery Mobile biedt alle gangbare elementen voor formulieren aan zoals textinvoer, een selectie uit een lijst, een zoekveld, een *slider* en een *switch*. Het raamwerk verplicht zelf om de `<label>`-tag te gebruiken. Zo wordt de applicatie toegankelijker gemaakt voor bijvoorbeeld mensen met een *e-reader*.

LIJSTEN Een laatste categorie UI elementen die jQuery Mobile aanbiedt, zijn lijsten. Deze gaan van standaard ongeordende lijsten tot lijsten met alle soorten decoraties als iconen, afbeeldingen, telbubbels en verdelers. Ook is het mogelijk om in deze lijsten te zoeken. Hiervoor dient de gebruiker enkel één data attribuut toe te voegen, waarna het raamwerk de implementatie voorziet.

4.1.4 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 één 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 [21]. 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 ontwerp patroon 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 ontwerp patroon af.

Een ander raamwerk, genaamd The-M-Project [30], dwingt het Model-View-Controller (MVC) echter wel af. Met dit raamwerk is het mogelijk om webapplicaties te maken die het jQuery Mobile raamwerk gebruiken. In plaats van HTML5-code te schrijven, zoals dat bij jQuery Mobile gebeurt, schrijft je JavaScript-code. The-M-Project zal dan zelf intern deze JavaScript-code omzetten naar de desbetreffende jQuery Mobile HTML5-code.

BROWSERONDERSTEUNING jQuery Mobile deelt browsers op in drie verschillende klassen: A, B en C [20]. 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* [13]. 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 ondersteunt.

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.

4.2 Sencha Touch

Sencha Touch is een relatief verschillend raamwerk in vergelijking met jQuery Mobile. Het wordt ontwikkeld door Sencha, een bedrijf dat in 2010 is ontstaan als een samensmelting van Ext JS, jQuery Touch en Raphaël. Ext JS is een JavaScript raamwerk voor de ontwikkeling van web applicaties. jQuery Touch is een jQuery plugin voor mobiele web ontwikkeling. Het steunt op WebKit en voegt *touch events* toe aan jQuery. Raphaël, ten slotte, is een JavaScript bibliotheek voor vector tekeningen. Op het moment van schrijven is Sencha Touch aan versie 2.1.1 [42].

4.2.1 Omkadering

PROGRAMMEERTAAL Sencha Touch is JavaScript gedreven dus all functionaliteiten worden in JavaScript geïmplementeerd. Het aanroepen van het raamwerk gebeurt door het invoeren van de Sencha Touch bibliotheek binnen `<script>`-elementen. Alle HTML code wordt bij het bekijken van de pagina gegenereerd.

TOOLS Naast Sencha Touch levert Sencha nog producten die Sencha Touch uitbreiden of het leven van de ontwikkelaar makkelijker maken. Deze worden hieronder opgelijst [42].

Sencha Animator Dit is een desktop applicatie om CSS3 animaties te ontwerpen. Deze animaties worden enkel in WebKit browsers ondersteund.

Sencha Architect Dit is een andere desktop applicatie waarmee je makkelijk een UI kan ontwikkelen met behulp van *drag-and-drop* commando's.

Sencha GXT Sencha GXT is een uitbreiding op Google Web Toolkit (GWT). De compiler van GWT laat toe applicaties in Java te schrijven en ze te compileren naar geoptimaliseerde, *cross-browser* HTML5 en JavaScript. Sencha GXT voegt grafieken, widgets, etc. toe aan GWT.

Sencha.IO Deze uitbreiding zorgt voor *cloud* services binnen mobiele applicaties.

DOCUMENTATIE Alle documentatie voor Sencha Touch 2.1.1 is te vinden op docs.sencha.com/touch/2-0. Een zoekfunctie voor objecten, eigenschappen en methoden is aanwezig om snel zaken op te zoeken. De meeste functionaliteiten zijn voorzien van codevoorbeelden samen met het resultaat hoe de browser de code rendert. Verder biedt de Sencha website ook een groot aanbod om Sencha te leren gebruiken www.sencha.com/learn/touch/. Hier staan handleidingen, introductie video etc..

Een ander handig raadslagwerk is de 'Kitchen Sink' [44]. Dit is een webapplicatie, geschreven in Sencha Touch, die de belangrijkste functionaliteiten bevat samen met de bijhorende code.

MARKTADOPTATIE Volgens de Sencha website is 50% van de Fortune 100 - een lijst van de grootste Amerikaanse bedrijven gerangschikt op jaaromzet - een Sencha klant [42]. Enkele van hun grootste klanten zijn CNN, Samsung, Cisco en Visa.

LICENTIES Sencha Touch is gratis binnen een commerciële context waarbij het bedrijf in kwestie de broncode niet deelt voor zijn gebruikers. Wanneer je dit wel wil doen bestaat er ook een gratis *open-source* versie van Sencha Touch. Deze komt met een GNU GPL v3 *open-source* licentie wat wil zeggen dat je de vrijheid hebt om aanpassingen aan de broncode te maken en te verspreiden, zolang je zelf je code maar gratis verspreid voor alle gebruikers.

Voor de ontwikkeling van eigen raamwerken of SDKs betaal je een *original equipment manufacturer* (OEM) licentie. Dit wil zeggen dat bedrijven hun producten gaan verkopen onder hun eigen merk en naam, maar gebruik maken van Sencha. Omdat het gebruik hiervan per gebruiker verschilt, worden OEM licenties op maat gemaakt [42].

4.2.2 Code en ontwikkeling

Zoals reeds vermeld moet alle code in JavaScript worden geschreven en dient één HTML bestand slechts als container om de bestanden in te laden. Sencha valt dus onder JavaScript gebaseerde raamwerken. De keuze voor deze aanpak heeft twee belangrijke motivaties. Enerzijds is Sencha Touch gebouwd op Ext JS, wat op zich een JavaScript raamwerk is. Anderzijds zorgt het voor een betere ondersteuning voor toestellen met verschillende resoluties. Samen met SASS en Compass kan Sencha lay-outs definiëren per device (zie sectie 4.2.4). De `Ext.env.Browser` en `Ext.env.OS` eigenschappen en `Ext.Viewport.getOrientation` en `Ext.feature.has` methoden kunnen de vereisten bepalen en de juiste lay-out kiezen [16].

Om het de ontwikkelaars makkelijker te maken biedt Sencha ook SDK tools aan. Momenteel bevinden deze zich nog in bèta. Concreet zijn deze tools commando's voor de terminal die onder andere nieuwe projecten kunnen aanmaken, JavaScript bestanden kunnen optimaliseren maar vooral de webapplicatie kunnen omzetten naar native applicaties voor iOS en Android.

DEBUGGING Het debuggen van je code gebeurt voornamelijk in de browser zelf. Tools als de Safari Web Inspector, Chrome Developer Tools of Firebug moeten de fouten kunnen opsporen. De broncode van Sencha Touch kan ook ingeladen worden met `sencha-touch-debug.js` als bibliotheek. Deze versie is niet gecomprimeerd en bevat commentaar en documentatie om makkelijker te zoeken waar in de code de fout zich juist bevond.

4.2.3 Functionele kenmerken

Net zoals jQuery Mobile heeft Sencha Touch ook een hele hoop functionaliteiten om eenvoudig UI elementen te genereren. Sencha Touch bevat alle elementen van de UI als JavaScript objecten. Net zoals alle objectgerichte programmeertalen maken deze objecten gebruik van een klassensysteem, iets wat slechts vanaf Sencha Touch 2 werd ingevoerd. Op die manier kunnen klassen worden gedefinieerd (`Ext.define`) en aangemaakt (`Ext.create`). Hierbij is ook overerving mogelijk. De basisklasse van alle objecten is `Ext.Component`. Componenten kunnen gerenderd worden, zichzelf tonen of verbergen, centreren op het scherm en zichzelf aan- of uitzetten. Het aanmaken van componenten kan compacter door het gewenste component als `xtype` te definiëren.

Een andere belangrijke component is `Ext.Container`. Containers kunnen sub-componenten bevatten en een lay-out speciëren. Alle componenten krijgen een naam die verwijst naar een namespace. Dit is handig om conflicten te vermijden tussen je eigen objecten en de standaard objecten van het raamwerk.

Voor een opsomming van alle raamwerk componenten verwijzen we naar de documentatie [43].

MODEL Data kan intern worden voorgesteld met models. Dit is iets wat hoort bij het MVC patroon (zie sectie 4.2.4). Een model specificeert een lijst van velden die bij het model horen waarbij een veld een naam en een type heeft. Optioneel kunnen validaties bij de velden worden toegevoegd om data consistent te houden.

STORE `Ext.data.Store` is de klasse om instanties van een model op te slaan. Een *store* wordt voorzien van een *proxy*. Deze kan data aan de client of server zijde opslaan. Een *proxy* voor opslag aan client zijde kan zowel in het RAM geheugen als in de *local storage* van de browser opslaan. Een *proxy* voor server opslag kan data verzenden via AJAX (zelfde domein) of JSONP (verschillende domeinen). Een *proxy* kan ook nog voorzien worden van een *reader* die aangeeft hoe de ontvangen data gelezen moet worden.

VIEW Een *view* is de benaming voor objecten die aan de gebruiker kunnen getoond worden. Een voorbeeld hiervan zijn lijsten, waar vaak de data van een *store* wordt in weergegeven. Zo'n lijst kan makkelijk gefilterd of gesorteerd worden op basis van velden uit het model. Hiervoor moeten we *filters* of *sorters* aan de *store* toevoegen. De lay-out van één lijstitem bepalen kan via een `XTemplate`. Het sjabloon bepaalt de HTML structuur van elk item. Alle gedefinieerde velden van het model kunnen in de template worden opgeroepen of gemanipuleerd.

4.2.4 Niet-functionele kenmerken

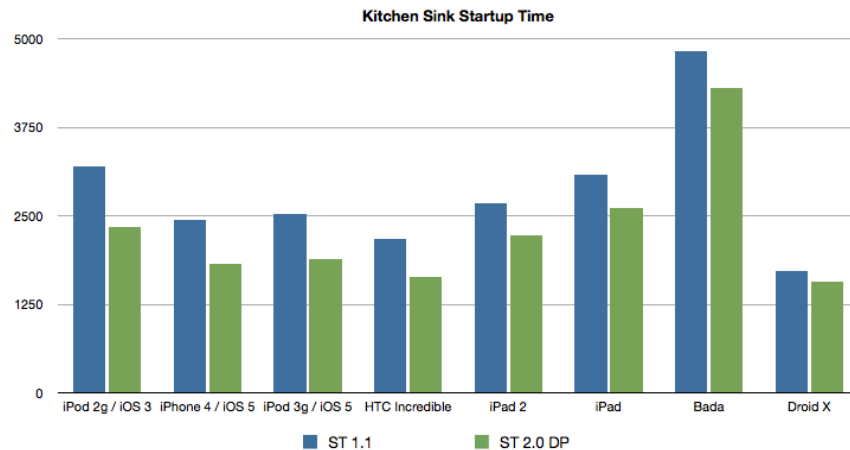
PERFORMANTIE In vergelijking met versie 1.1 van Sencha Touch is de performantie gestegen om wille van verschillende factoren. De introductie van het klasse systeem, zoals besproken in de vorige sectie, laat toe objecten dynamisch te laden. Het grote verschil tussen `Ext.define` en `Ext.create` is dat objecten enkel in het geheugen worden geladen na creatie. Het is dus de taak van de programmeur om objecten enkel te construeren wanneer ze nodig zijn.

Verder kwam versie 2.0 met een nieuwe lay-out *engine* die vooral het verwisselen van oriëntatie van het toestel versnelde. Ook een verbetering in performantie op Android toestellen, voornamelijk bij scrollen en animaties, werd ingevoerd [42].

Een benchmark voor deze verbeteringen zijn de opstarttijden van de Kitchen Sink applicatie. Het opstarten gebeurde met de verschillende Sencha Touch versies en op verschillende toestellen. De resultaten zijn terug te vinden op figuur 4.1. Op bijna elk toestel blijkt Sencha Touch 2.0 ongeveer één seconde sneller te werken [45].

AANPASBAARHEID Elke component binnen het raamwerk moet overerven van `Ext.Component`. Deze voorziet een attribuut `ui`. De waarde hiervan is een CSS klasse die bepaald hoe de component er zal uitzien. Sencha heeft al twee CSS klassen voorzien: `light` en `dark`. Andere componenten kunnen deze lijst uitbreiden. Een knop kan bijvoorbeeld `normal`, `back`, `round`, `small`, `action` of `forward` als `ui` waarde hebben.

Het is ook mogelijk om eigen waarden voor `ui` te definiëren of de standaarden van Sencha aan te passen. Hiervoor moet je gebruik maken van SASS en Compass om je CSS bestanden aan te maken. SASS staat voor Syntactically Awesome Stylesheets en breidt CSS uit met variabelen, geneste structuren, mixins en overerving [9]. Mixins groeperen enkele CSS eigenschappen en kunnen worden herbruikt. Compass is een raamwerk bovenop SASS en CSS. Het compileert SCSS (Sassy CSS) naar CSS bestanden [8].



FIGUUR 4.1: Sencha Touch Kitchen Sink opstarttijden [45].

Sencha thema's bestaan allemaal uit een set van mixins. Door zelf mixins te creëren of reeds bestaande te manipuleren kunnen we eigen thema's creëren en ze aan de ui-waarde van een component toekennen.

PROGRAMMEERBAARHEID Zoals reeds aangehaald ondersteund Sencha Touch het MVC (Model-View-Controller) patroon. Dit patroon vermijdt lange JavaScript bestanden door ze logisch op te delen. Modellen groeperen velden tot een beschrijving van data-objecten, views definiëren de weergave van componenten en controllers verbinden beide op basis van events.

In theorie zou het verschil tussen mobiele websites en applicaties enkel in de views terug te vinden zijn. Echter, dit wordt nog niet volledig ondersteund en raadt men dus aan om hiervoor aparte projecten te voorzien.

ONDERSTEUNING BROWSER Sencha Touch steunt op de WebKit browser *engine* dus moet de browser deze bevatten. Hoewel dit bij de meeste browsers geen probleem meer vormt vallen toch enkele populaire browsers uit de boot. Sencha Touch is bijvoorbeeld niet compatibel met Firefox Mobile en Opera Mobile [16].

Zoals reeds vermeld zijn er ook methoden voorzien om informatie op te vragen over de context die gehanteerd wordt (browser, OS, toestel, etc.). Verder kan Sencha Touch ook vragen naar de ondersteuning van specifieke kenmerken (audio, canvas, CSS3, ...), analoog als Modernizr.

Op de Sencha website zijn voor sommige browsers en bijhorend besturingssystemen scorecards voorzien om hun compatibiliteit met HTML5 en Sencha Touch te bespreken [42].

4.3 Kendo UI

4.3.1 Omkadering

4.3.2 Code en ontwikkeling

4.3.3 Functionele kenmerken

4.3.4 Niet-functionele kenmerken

4.4 Lungo

4.4.1 Omkadering

4.4.2 Code en ontwikkeling

4.4.3 Functionele kenmerken

4.4.4 Niet-functionele kenmerken

Evaluatie

In dit hoofdstuk voeren we de vergelijking uit en bekijken we de bekomen resultaten. Enerzijds vergelijken we in ?? de implementatie van de POC in de verschillende raamwerken. Anderzijds vergelijken we in ?? de raamwerken op basis van de vergelijkscriteria opgesteld in hoofdstuk 3.

5.1 Omkadering

5.2 Productiviteit

5.3 Gebruik

Formulieren

JQUERY MOBILE Voor het toevoegen van *placeholders* in de formulervelden kon beroep worden gedaan op het **placeholder** attribuut in HTML5. Er dienden geen labels te worden gezet voor de velden. Deze labels zijn echter wel verplicht in jQuery Mobile, maar kunnen onzichtbaar worden gemaakt met de **ui-hide-label** CSS klasse [?]. Wat wel opmerkelijk is wanneer men een formulier invult, daarna verstuurt en dan terugkeert, het formulier nog alle waarden bevat. Men moet na het formulier te hebben verstuurd, zelf het formulier altijd leegmaken. Dit kan met behulp van JavaScript via de **reset()**-functie op het formulier.

Voor de types van de formulervelden werd beroep gedaan op de volgende types: **text**, **number** en **email**. Deze zorgen ervoor dat op de mobiele apparaten aangepaste toetsenborden te voorschijn komen. Het **date** type werd echter niet gebruikt. Enerzijds was hiervoor een slechte ondersteuning naar mobiele browsers toe [?]. Dit betekende onder andere dat Android 2.3 dit niet ondersteunde. Een ander probleem was het ontbreken van een *placeholder* in het veld op iOS 6 en Android 4.2. Hierdoor weet de gebruiker in eerste instantie niet wat hij hier moet invullen, omdat er ook geen labels aanwezig moesten zijn. Daarnaast is een **placeholder** instellen onmogelijk voor een **date** type [?]. Anderzijds moest het ook mogelijk zijn om de gebruiker maar een bepaald bereik van data op te leggen, wat met het **date** type onmogelijk is. Hierdoor werd gebruik gemaakt van de Date & Time Picker van Mobiscroll [?]

die ook aangepaste lay-out heeft conform met die van jQuery Mobile. Het veld heeft dan het type `text`. Het is dus in principe mogelijk om iets anders dan een datum in te geven. Dit wordt belet door ook nog eens een datum validatie te doen op dit tekstveld mocht de plugin het al niet afgedwongen hebben.

Het was ook nodig om enkel de maand en jaar in te geven. Ook hier kon niet het `date` type gebruikt worden, omdat daar ook een dag voor nodig is. Daardoor werden de maanden handmatig geprogrammeerd als vaste lijstitems. De jaren zijn dynamisch en zijn telkens dit jaar, het volgende en het vorige jaar. Deze functionaliteit had ook met de Mobiscroll plugin kunnen worden verwezenlijkt.

Validatie

JQUERY MOBILE Sommige formulervelden waren verplicht in te vullen, terwijl andere niet. Hiervoor werd eerst gedacht om het `required` attribuut in HTML5 te gebruiken. Het probleem is echter dat er geen ondersteuning is voor mobiele browsers [?]. Daarnaast was het ook nodig om de velden te valideren op hun waarde. Validatie is echter niet standaard aanwezig in jQuery Mobile. Als oplossing werd de plugin van Jörn Zaefferer gebruikt [?]. Deze plugin loste ook het probleem met de verplichte velden op. Deze plugin kan op twee manieren gebruikt worden: enerzijds annoteren van de formulervelden met speciale CSS klassen ofwel anderzijds door programmatie met JavaScript. Beide aanpakken werden getest doorheen de POC.

De plugin bevatte de volgende ingebakken validatieregels nodig in de POC: `required`, `number`, `email` en `date`. Daarnaast was het nodig dat een veld verplicht was enkel indien een bepaalde optie aangevinkt was. Zo'n afhankelijkheidsrelatie is standaard aanwezig in de plugin.

Bij fouten tegen validatie moest een dialoogvenster worden weergegeven met daarin een beschrijving van alle foute velden. Aangezien de plugin standaard onder het foute formulerveld een foutboodschap toont, diende de plugin te worden aangepast. Door de uitgebreide API van de plugin die ook uitvoerig gedocumenteerd is, konden alle foutboodschappen samen in een dialoogvenster worden weergegeven.

Een specifiek mobiel probleem was bij het tonen van het dialoogvenster, waarbij de plugin op de achtergrond de cursor op het eerste veld zette. Hierdoor verscheen het toetsenbord op het scherm van het mobiele apparaat wanneer het dialoogvenster tevoorschijn kwam, wat niet de bedoeling is. Dit werd opgelost door `focusInvalid:false` in te stellen in de plugin.

Bij het sluiten van het dialoogvenster dienden de foute velden met een rode rand te worden gemarkeerd. Aangezien de validatie plugin de foute velden annoteert met de `error` CSS klasse, kon de rode rand in CSS worden geprogrammeerd. Dit ging voor `input` en `textarea`, maar gaf problemen voor `select` en `fieldset`. Door de extra code die jQuery Mobile genereert rond deze velden, moest via de DOM de omringende code geannoteerd worden om de rode rand te bekomen. Deze functie kon aangehaakt worden op de `highlight` en `unhighlight` functies van de plugin.

Opladen van bewijs

JQUERY MOBILE Het opladen van een bestand kan gebeuren door `file` als invoertype van het formulierveld te gebruiken. In versie 1.2 wordt dit veld nog niet opgemaakt met lay-out, maar dit gebeurt wel in versie 1.3 [?]. Voor het kan worden doorgestuurd naar de backend, moet het bewijs eerst lokaal worden omgevormd naar base64. Dit werd geïmplementeerd met de FileReaderAPI en het canvas, wat beide HTML5 specificaties zijn. Het aangeklikte bestand wordt gelezen door middel van de FileReaderAPI, waarna het als afbeelding wordt opgeslagen en geïmporteerd wordt op het canvas. Eenmaal geïmporteerd, kan men de `.toDataURL()` oproepen op het canvas om de geïmporteerde afbeelding om te vormen naar base64. Deze aanpak werkt correct op recente mobiele apparaten. De FileReaderAPI wordt echter niet ondersteund op Android versies 2.3 en lager of iOS versies lager dan 6.0 [?] waardoor het opladen van een bewijs niet werkt.

Het voorvertonen van het geüploade bestand hangt af van het mobiele besturings-systeem. Zo wordt op iOS 6 een miniatuurafbeelding getoond, terwijl op Android de bestandsnaam wordt getoond. Het is natuurlijk ook mogelijk om de preview na conversie zelf te tonen op het scherm. Bij iOS zouden er dan twee voorvertoningen te zien zijn op hetzelfde scherm.

Handtekening

JQUERY MOBILE Er werd gezocht naar een plugin om deze functionaliteit te bekomen, doordat jQuery Mobile dit niet standaard aanbiedt. Eerst werd gewerkt met Signature Pad van Thomas Bradley [?]. Door de lange tijd die werd besteed aan het aanpassen van layout, werd overgestapt naar jSignature van Willow Systems [?]. Deze laatste gaf ook het voordeel dat de breedte van het gebied om te handtekening in te zetten, zich automatisch naar 100% schaalde. De plugin maakt gebruik van het HTML5 canvas element en de `.toDataURL()` methode. Deze wordt echter niet ondersteund op Android versies 2.3 en lager [?] waardoor de functionaliteit op die toestellen niet werkt.

AJAX

JQUERY MOBILE Het maken van oproepen via AJAX gebeurt via de jQuery bibliotheek waar jQuery Mobile op steunt. Dit gebeurt met de functie `$.ajax` waar onder andere kan ingesteld worden wat het te verwachten antwoord is (zoals tekst, JSON of XML). Bij het succesvol uitvoeren van de oproep wordt de **succes** functie opgeroepen, bij falen de **error** functie waarna een relevante foutboodschap wordt getoond. Het afmelden zonder antwoord, het aanmelden voor het bekomen van het token (tekst), het ophalen van de gebruikersgegevens (JSON), het ophalen van de uitgaveformulieren (XML) en het ophalen van de omwisselingskoersen (XML) ging zonder enig probleem.

JSON

JQUERY MOBILE In jQuery is er de functie `parseJSON` aanwezig, maar aangezien we in de AJAX oproep instellen dat we JSON verwachten, parst jQuery al automatisch het antwoord. Hierdoor hebben we `parseJSON` niet nodig en kunnen we direct omgaan met het antwoord.

Het is ook nodig om JSON te versturen als oproep naar de backend. Dit wordt gedaan vanuit JavaScript zonder een jQuery nodig te hebben. Eerst wordt een object met de nodige inhoud aangemaakt, waarop daarna de functie `JSON.stringify` opgeroepen wordt die het object in een string omzet. Deze is daarna klaar om te worden verstuurd als data via een AJAX oproep met behulp van jQuery.

XML

JQUERY MOBILE Net zoals bij JSON het geval was, was het ook niet nodig om expliciet de `parseXML` functie te gebruiken. Het doorlopen en opvragen van gegevens uit het XML-bestand vraagt meer werk. Waar je bij JSON direct aan de data kon, moet je bij XML de data ophalen net zoals je dat zou doen uit een HTML-pagina. Dit betekent dus met selectoren aan de hand van de jQuery bibliotheek.

PDF

JQUERY MOBILE Het is niet aangeraden om ruwe data, zoals een PDF, op te halen via AJAX. Hierdoor werd gebruik gemaakt van een verborgen formulier met de nodige parameters die de PDF ophaalt bij de backend. Bij het klikken op een lijstitem in het overzicht, wordt dit verborgen formulier opgestuurd naar de backend die dan een PDF teruggeeft in de browser. Het weergeven van de PDF wordt overgelaten aan het mobiel apparaat dat de correcte applicatie hiervoor opstart.

Automatische aanvulling

JQUERY MOBILE Hoewel versie 1.3 automatische aanvulling ter beschikking heeft [?], werd tijdens de implementatie gebruik gemaakt van versie 1.2 die dit niet had. Daarom werd de plugin van Andy Matthews gebruikt [?]. Dit is een zeer gemakkelijk te integreren plugin die zowel met lokale data als data op afstand kan werken. Daarnaast dienden enkel vijf suggesties te worden getoond. Deze functionaliteit zat niet in de plugin, maar werd geïmplementeerd met de JavaScript `slice` functie.

Tabbalk

JQUERY MOBILE Standaard is er een tabbalk aanwezig in jQuery Mobile, maar de POC impliceerde een tabbalk die niet de volledige breedte innam. Daarom werd gekozen voor `fieldset` met twee opties.

Inlogschermb indien niet aangemeld

JQUERY MOBILE Indien men een applicatie maakt met meerdere schermen op eenzelfde pagina, laadt jQuery Mobile altijd het eerste scherm in de code in. Het startschermb werd als eerste scherm gekozen. Indien gemerkt wordt dat de gebruiker niet aangemeld was, dan wordt hij doorverwezen naar het inlogschermb.

Detail van toegevoegde uitgave

JQUERY MOBILE Na het toevoegen van een uitgave, is het mogelijk om deze opnieuw te bekijken (maar niet aan te passen). Hiervoor wordt hetzelfde formulier (dat om een uitgave toe te voegen wordt gebruikt) gekopieerd, waarna alle elementen op enkel lezen worden gezet. Dit was geen probleem voor velden van het type `input` en `textarea`. Dit kon echter niet bij `fieldset`. Daar moesten via `disabled` de andere opties onmogelijk worden gemaakt. Eenzelfde probleem gold voor het `select` formuliertype bij een buitenlandse uitgave. Daar werd enkel de geselecteerde optie in het lijstje getoond en alle andere opties eruit verwijderd.

Het invullen van het formulier zelf werd bekomen door de `id`'s van de velden op te vragen en hun waarde in te stellen volgens de JSON voorstelling van die uitgave. Er is met andere woorden geen automatische mapping van de JSON data naar de formulervelden.

Laadschermb

JQUERY MOBILE Het standaard laadschermb is enkel een *spinner* die niet opvallend aanwezig is en ook zonder een tekst eronder ronddraait. Door de opties in de API te gebruiken, komt de *spinner* duidelijk naar voor en staat er ook een tekst onder.

Dialoogvenster

JQUERY MOBILE Eerst werd gebruik gemaakt van `DateBox` [?] als plugin om op een gemakkelijke manier een dialoogvenster te tonen. Uiteindelijk bleek de plugin niet zo gemakkelijk aanpasbaar en daarenboven zijn dialoogvensters standaard in jQuery Mobile aanwezig. Het is dan ook helemaal niet nodig om hiervoor een plugin te gebruiken. Door zelf de dialoogvenster met jQuery Mobile aan te maken, kon de layout minimier aangepast worden.

Omvormen van valuta

JQUERY MOBILE De omvorming bij een buitenlandse uitgave dient automatisch te gebeuren bij het ingeven van bedrag en munteenheid. Hiervoor wordt aangehaakt op het veranderingsevenement `.change` dat jQuery aanbiedt, waarna na omvorming het bedrag direct getoond wordt aan de gebruiker.

Sorteren

JQUERY MOBILE Het sorteren van data werd geïmplementeerd door eerst in JavaScript een vergelijkingsfunctie te schrijven. Daarna wordt deze functie meegegeven aan de sorteerfunctie die ook in JavaScript aanwezig is. Er komt dus geen functionaliteit van het raamwerk om data te sorteren.

Offline

TODO

Tablet en smartphone

JQUERY MOBILE In jQuery Mobile is er standaard geen splitview aanwezig om een menu te tonen voor tablets, maar niet voor smartphones. Eerst werd hiervoor gezocht naar plugins aan de hand van [?], wat leidde tot: Splitview [?], SimpleSplitView [?] en Multiview [?]. Deze drie mogelijke kanshebbers hadden elk hun tekorten. Zo was de eerste destructief ten opzichte van het raamwerk. Dit betekent dat de bestanden van het raamwerk zelf werden aangepast, wat het moeilijker maakt als men wil updaten naar een nieuwe versie. De tweede plugin werkte enkel tot versie 1.0.1 van jQuery Mobile. De laatste plugin had moeite met het zich aanpassen aan veranderende afmetingen van de browser.

Uiteindelijk werd van een plugin afgestapt door [?] waarbij werd aangetoond hoe men via CSS3 media queries hetzelfde kan bereiken. Daarnaast gebruikt de documentatie van jQuery Mobile 1.2 een gelijkaardige layout [18]. De uiteindelijke oplossing voor het probleem kwam uit te combinatie van deze twee voorgaande oplossingen. Ook uit de documentatie van versie 1.3 [?] blijkt dat dit de correcte manier is om hiermee om te gaan.

Navigatie op een smartphone gebeurt door te klikken op de extra titel onder de koptekst. Hierdoor ga je naar een smartphone vriendelijk menu om naar andere stappen te gaan.

Koptekst en voettekst

JQUERY MOBILE Het toevoegen van een koptekst en voettekst ging zonder enig probleem door gebruik te maken van `data-role="header"` en `data-role="footer"`. Wel moest dezelfde code op ieder scherm worden herhaald. Dit kan worden vermeden door gebruik te maken van eenzelfde `data-id` attribuut. Daarnaast werd de voettekst gefixeerd aan de onderkant van het scherm en de bijhorende logo's links en rechts uitgelijnd. Voor dit laatste werd gebruik gemaakt van de zogenaamde *grid* die jQuery Mobile aanbiedt. Deze voettekst wordt niet getoond op een smartphone, wat wordt bekomen door gebruik te maken van de CSS3 media queries.

Bij het toevoegen van een uitgave dient er een extra titel onder de koptekst te komen. Eerst werd geprobeerd om bovenaan een lijstdeeler te plaatsen, maar dan schoof de inhoud van de pagina niet mee naar onder. De uiteindelijke oplossing kwam vanuit de documentatie [?] om dit met behulp van de `ui-bar` CSS klasse te

implementeren. Deze extra titel wordt ook gebruikt om op de smartphone naar de speciale smartphone navigatie te gaan (zie ook vorige sectie).

Knoppen

JQUERY MOBILE De kleur van de knoppen aanpassen kan op twee manieren. Ofwel schrijft men zelf de CSS-code ofwel gebruikt men ThemeRoller [21]. Deze laatste manier werd gebruikt om de knoppen groen te maken. Men sleept dan eenvoudigweg in die webinterface de groene kleur op de knop en daarna kan de bijhorende CSS-code worden gedownload. Door daarna de knop te annoteren met het **data-theme** attribuut activeert men het betreffende thema. Om de knop blauw te maken was er geen nood aan een aanpassing, doordat blauw al één van de standaard thema's was en men die direct kan gebruiken.

Knoppen toevoegen aan de koptekst gaat ook op een zeer eenvoudige manier.

Lijsten

JQUERY MOBILE Het aanmaken van lijsten gebeurt met de *listview* widget.

Lijsten

JQUERY MOBILE

5.4 Ondersteuning

5.5 Performantie

Besluit

De masterproeftekst wordt afgesloten met een hoofdstuk waarin alle besluiten nog eens samengevat worden. Dit is ook de plaats voor suggesties naar het verder gebruik van de resultaten, zowel industriële toepassingen als verder onderzoek.



Bibliografie

- [1] Accenture. HTML5: The Path to Cross-Platform Mobile Development. 2012.
- [2] Android. Platform Versions, 2012.
- [3] Aptana. Aptana Studio 3, 2012.
- [4] B. Broulik. *Pro jQuery Mobile*. Apress, 2012.
- [5] M. David. *HTML5 Mobile Websites: Turbocharging HTML5 with jQuery Mobile, Sencha Touch, and Other Frameworks*. Focal Press, 2011.
- [6] P. Deitel, H. Deitel, A. Deitel, and E. Kern. *iOS 6 for Programmers: An App-Driven Approach, Second Edition*. Prentice Hall, 2012.
- [7] Dmethvin. jQuery Licensing Changes, 2012.
- [8] C. M. Eppstein. Compass, 2013.
- [9] H. C. . N. W. . C. Eppstein. SASS, 2013.
- [10] M. Falk. Mobile Framework Comparison Chart, 2011.
- [11] W. Hales. *HTML5 and JavaScript Web Apps*. O'Reilly Media, Inc., 2012.
- [12] P. L. Hégaret, L. Wood, and J. Robie. What is the Document Object Model?, 2004.
- [13] J. Hens. Progressive Enhancement versus Graceful Degradation, 2012.
- [14] JetBrains. WebStorm, 2012.
- [15] S. Jobs. Thoughts on Flash, 2010.
- [16] B. P. J. John E Clark. *Sencha Touch Mobile JavaScript Framework*. Packt Publishing, 2012.
- [17] JQuery. jQuery Mobile, 2012.

- [18] JQuery. JQuery Mobile: Demos and Documentation, 2012.
- [19] JQuery. JQuery Project, 2012.
- [20] JQuery. Mobile Graded Browser Support, 2012.
- [21] JQuery. ThemeRoller for JQuery Mobile, 2012.
- [22] M. Kool. Let's Play With Hardware-Accelerated CSS, 2012.
- [23] D. Lieberman. Microsoft's Windows Phone 7 to replace Windows Mobile, 2010.
- [24] M. MacDonald. *HTML5: The Missing Manual*. O'Reilly Media, Inc., 2011.
- [25] D. S. McFarland. *JavaScript & jQuery: The Missing Manual, Second Edition*. O'Reilly Media, Inc., 2011.
- [26] Microsoft. Nokia and Microsoft Announce Plans for a Broad Strategic Partnership to Build a New Global Mobile Ecosystem, 2011.
- [27] Modernizr. Modernizr: the feature detection library for HTML5/CSS3, 2012.
- [28] NetApplications. Mobile/Tablet Browser Market Share, 2012.
- [29] D. Oehlman and S. Blanc. *Pro Android Web Apps: Develop for Android using HTML5, CSS3 & JavaScript*. Apress, 2011.
- [30] Panacoda. The-M-Project, 2012.
- [31] T. Parker. Announcing JQuery Mobile 1.0, 2011.
- [32] T. Parker. Announcing JQuery Mobile 1.2.0 Final, 2012.
- [33] T. Parker. JQuery Mobile Team Meeting December 20, 2012, 2012.
- [34] G. Phifer and D. M. Smith. The (Not So) Future Web. Technical report, Gartner, 2011.
- [35] Phil Dutson. *Sams Teach Yourself jQuery Mobile in 24 Hours*. Sams, 2012.
- [36] B. Reed. Microsoft show off new key Windows Phone 8 features, 2012.
- [37] J. Resig. Announcing the JQuery Mobile Project, 2010.
- [38] A. Sarrafi. Mobile JavaScript frameworks, Evaluation and Comparison, 2012.
- [39] Satyesh. Android Versions History, 2012.
- [40] P. Sawers. Google announces Android 4.2, a "new flavor of Jelly Bean" with gesture typing and slick photo sphere camera, 2012.
- [41] B. Seitz. Windows Phone 7 Global Portfolio Unveiled in New York, 2010.

- [42] Sencha Inc. Sencha, 2013.
- [43] Sencha Inc. Sencha Touch Documentatie, 2013.
- [44] Sencha Inc. Sencha Touch Kitchen Sink, 2013.
- [45] Sencha Inc. What's New in Sencha Touch 2.0, 2013.
- [46] Sights. The HTML5 test, 2012.
- [47] B. Sperry and M. Lynch. Codiqa, 2012.
- [48] I. Standard. ISO/IEC 25010. Technical report, 2010.
- [49] Sylvain. iOS Version Statistics – November 14th 2012, 2012.
- [50] W3C. HTML5 Logo, 2012.
- [51] E. Weyl, L. Lazaris, and A. Goldstein. *HTML5 & CSS3 For The Real World*. SitePoint, 2011.
- [52] Wikipedia. Comparison of JavaScript frameworks.
- [53] S. Wolfermann. The Responsive Webdesign Process, 2012.
- [54] J. Yang. Smartphones in Use Surpass 1 Billion, Will Double by 2015, 2012.

Fiche masterproef

Studenten: Tim Ameye
Sander Van Loock

Titel: Vergelijkende studie van raamwerken voor de ontwikkeling van mobiele HTML5 applicaties

Engelse titel: The best master thesis ever

UDC: 621.3

Korte inhoud:

Hier komt een heel bondig abstract van hooguit 500 woorden. \LaTeX commando's mogen hier gebruikt worden. Blanco lijnen (of het commando `\par`) zijn wel niet toegelaten!

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Thesis voorgedragen tot het behalen van de graad van Master of Science in de ingenieurswetenschappen: computerwetenschappen

Promotor: Prof. dr. ir. E. Duval

Assessoren: Ir. W. Eetveel
W. Eetrest

Begeleiders: Ir. A. Assistent
D. Vriend