NTNU – Trondheim
Norwegian University of
Science and Technology

HTML5 for the Development of Personalized Multi-Platform Applications

Christian Gjerde Jensen

Master of Science in Computer Science
Submission date: June 2013
Supervisor: John Krogstie, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Abstract

As mobile devices grow more integrated into the everyday life of the general public, there is an increased benefit to be gained from utilizing personalized applications. One of the biggest challenges in developing for these mobile devices is the variation between the supported tools and languages for each platform. The HTML5 standard is supported by every modern platforms, which allows these problems to be circumvented. HTML5 includes several new features and APIs that allow for development of complex web applications, including functionality for personalization. These factors provide great incentive for using HTML5 in the development of multi-platform personalized applications.

A prototype application was developed in order to examine the process and product of developing a personalized multi-platform application with HTML5. The results show that although there are many benefits to choosing HTML5 for this purpose, extensive testing is still required in order to prevent platform-specific issues and bugs, as well as to ensure a consistent user experience on all devices. While HTML5 lacks access to certain device features, it is a great choice when these are not required. However, HTML5 is still in development, and the W3C has plans for adding APIs for these features, which bodes well for the future of personalized web applications.

Sammendrag

Mobile enheter blir stadig viktigere i det daglige livet til folk flest, noe som medfører økte fordeler ved bruk av personaliserte applikasjoner. En av de største utfordringene ved utvikling for slike mobile enheter er variasjonen mellom støttede verktøy og språk for hver plattform. HTML5 er en standard som støttes av alle moderne plattformer, noe som gjør det mulig å unngå dette problemet. HTML5 inkluderer også flere nye funksjoner og APIer som muliggjør utvikling av komplekse webapplikasjoner, inkludert funksjonalitet for personalisering. Dette gjør det fordelaktig å bruke HTML5 i utvikling av multi-plattform personaliserte applikasjoner.

En applikasjonsprototype ble utviklet for å undersøke både prosessen og produktet ved utviklingen av en personalisert multi-plattform applikasjon med HTML5. Resultatene viser at selv om det er mange fordeler med å velge HTML5 til dette formålet, kreves det fortsatt grundig testing både for å forhindre plattform-spesifikke programvarefeil så vel som å sikre en konsistent brukeropplevelse på alle enheter. Selv om HTML5 mangler tilgang til visse enhetsfunksjonaliteter er det et godt valg når disse ikke kreves. HTML5 er fortsatt under utvikling, og W3C har planer om å legge til flere APIer for å støtte slik funksjonalitet, noe som lover godt for fremtiden til personaliserte webapplikasjoner.

Preface

This master's thesis was written during the final semester of the 5-year integrated master's programme in Computer Science at the Norwegian University of Science and Technology, Department of Computer and Information Science. The assigned task was to assess the applicability of HTML5 for the development of multi-platform personalized applications. This was to be accomplished by creating and evaluating a prototype application, a personalized news service, in accordance with the design science research method.

I would like to thank the following people for their assistance and advice.

- John Krogstie, for his advice and guidance.
- Muhammad Asif, for his insight into the field of personalization research.
- Alf Inge Wang, John Krogstie, and Simone Mora for providing devices for testing.

Contents

List of Figures

List of Tables

Abbreviations

ACM	**A**ssociation for **C**omputing **M**achinery
AJAX	**A**synchronous **J**avascript **A**nd **X**ML
API	**A**pplication **P**rogramming **I**nterface
CSS	**C**ascading **S**tyle **S**heets
DOM	**D**ocument **O**bject **M**odel
GPS	**G**lobal **P**ositioning **S**ystem
HTML	**H**yper**T**ext **M**arkup **L**anguage
IDE	**I**ntegrated **D**evelopment **E**nvironment
IEEE	**I**nstitute of **E**lectrical and **E**lectronics **E**ngineers
JS	**J**ava**S**cript
JSON	**J**ava**S**cript **O**bject **N**otation
NTNU	**N**orwegian **U**niversity of **S**cience and **T**echnology
MVVM	**M**odel **V**iew **V**iew**M**odel
LAN	**L**ocal **A**rea **N**etwork
W3C	**W**orld **W**ide **W**eb **C**onsortium

Chapter 1

Introduction

Personalization is the concept of adapting a service based on the preference of the individual user. Companies like Facebook and Google provide their services free of charge, earning their revenue from personalized advertisements tailored to each user based on the personal data they have gathered. In e-commerce, personalization is used to recommend suitable products based on the browsing history and previous purchases of users. Entertainment providers like Netflix and YouTube use personalization to recommend content to each individual user based on their preferences.

HTML5 enables the development of ubiquitous applications which can be used on virtually every device with a web browser. This is ideal for personalized applications and services, both for the convenience the users gain by always having access to their personalized service, and by giving the service provider more data points which can be utilized to improve the personalized experience on each device.

The purpose of this thesis is to investigate the suitability of HTML5 as a development platform for personalized applications. This involves examining the features of HTML5 which would be beneficial to personalized applications, as well as investigating the restrictions HTML5 imposes on the development of such applications.

1.1 Rationale

The task was originally suggested by the advisor of the thesis, after the author expressed his interest in working with web technology and HTML5. The author has previously worked with HTML5 and related technology, both at NTNU and during summer internships. It is a fascinating technology due to its nearly ubiquitous platform availability, and is looking to be an important part in shaping the future of software development. Personalization is a concept which is used to great effect by many successful companies, with Google being the most notable example.

1.2 Research Questions

As personalization is a multi-faceted concept, the research questions have been designed to each concern one particular aspect.

RQ1: *"How suitable is HTML5 for developing a context-aware personalized application?"*
RQ2: *"How suitable is HTML5 for developing applications with personalization based on a persistent user profile?"*
RQ3: *"How suitable is HTML5 for developing applications with personalized user interfaces?"*

1.3 Contribution

As HTML5 is new technology and still in development, there has not yet been a great amount of academic research on the topic. The intent is to provide insight into the use of HTML5 for developing personalized multi-platform applications. This includes determining which personalization requirements HTML5 is capable of fulfilling, as well as the benefits and drawbacks of choosing HTML5 as the

development platform for personalized applications. The prototype application used to investigate the features and suitability of HTML5 is available under the Apache 2.0 licence, and can be used freely for any future works on the subject.

1.4 Reader's guide

This thesis is structured similarly to the chronological order of the execution of the practical work, from research to planning, execution and conclusion. First, chapter 2 explains the theories of personalization, along with the technology to be examined. The research method and plans for the development and testing of the prototype application are specified in chapter 3. Chapter 4 contains the details of the completed prototype application, including its architecture and the implementation process. The results of the system tests performed on the prototype application can be found in chapter 5. Chapter 6 evaluates the results with discussions of their implications for each of the research questions. Finally, the overall conclusion is made in chapter 7, along with a brief explanation of the limitations of the thesis as well as suggestions for further work. Appendix A provides code samples and detailed explanations of some key concepts and technologies used in the prototype application.

Chapter 2

Background

The purpose of this chapter is to explore the basis of the thesis. The concept of personalization is explained, as is the terminology and classifications needed to relate the work to the overall field of personalization. Previous studies on the topic of personalization and web development are examined in order to learn from previous work in the field, and to ensure the originality of the thesis. Finally, the state of the art of HTML5 and related web technologies is described, with an emphasis on new features that may benefit the development of personalized applications.

2.1 Literature Study

A literature search was performed in order to further examine the existing body of work in the field of personalization. Two search engines were used in the literature search, Sciverse's Scopus and Google Scholar. Both of these search online archives such as the ACM library and IEEE Xplore and aggregate the results, and provide functionality for filtering and sorting the results.

Broad searches consisting of combinations of terms such as "HTML5", "Web", "Personalization" and related topics were performed to get an overview of existing

research and knowledge on the topic. The results of these queries were filtered based on their recency and received citations from later work. The recency was weighted heavily for the technological and web-related works, as web technology today is very different than it was only a decade ago. This resulted in a set of articles on personalization concerning a wide span of topics, including data mining, web technology, metrics, privacy, personalization algorithms, and so on.

These articles were then examined in order to determine which one were the most relevant to the task at hand. After excluding several articles that were deemed to not be significantly relevant, the articles that best represented the remaining topics were chosen for further examination, discarding those that overlapped with more suitable alternatives. The results of this process are the articles discussed in this chapter, each concerning a different aspect of the topic of personalization and the web.

2.2 Personalization

Broadly speaking, personalization can be defined as the use of technology to accommodate the individual needs of the users. Personalization can be utilized for a wide variety of purposes, and there are a plethora of means to personalize services. In order to ensure clarity and relate the work to the overall field, the thesis will utilize the classification scheme for implementations of personalization defined by Fan and Poole (Fan and Poole, 2006). Fan and Poole divide their scheme into the following three dimensions:

1. The first dimension concerns which parts of the system are personalized. There are four aspects in information systems that can be personalized; the content, the user interface, the access channel, and the functionality of the system.

2. The second dimension is the target of personalization, which can either be groups of people or individual users. For groups, the personalization is aimed

Type	**Means**	**Goal**
Architectural	Altering the environment based on the cognitive, affective and social-cultural aspects of the user	Provide an experience tailored to the personal style of the user
Relational	Building social interactions from the social context of the user	Enhance interpersonal interactions and maintain social networks
Instrumental	Tailoring the tools and functionality to the needs of the user	Improve user productivity
Commercial	Present services and products of relevance to the consumer	Increase sales and improve user loyalty

TABLE 2.1: Ideal types of personalization

at people who fit into certain categories, such as parents, students, or football enthusiasts. Individualized personalization targets each specific individual based on their personal data and preferences.

3. The third dimension is the degree of automation of the personalization. With *explicit personalization*, the user controls the personalization, whereas with *implicit personalization*, the system controls an automated personalization process.

Additionally, Fan and Poole define four ideal types of personalization, each of which represents a motivation to incorporate personalization along with its goals and the means accomplish them. The ideal types are summarized in table 2.1. These ideal types are also distinct in two other dimensions. The instrumental and commercial types are *utilitarian*, as they focus on improving the productivity of the user, while the architectural and relational perspectives are *affective* due to their emphasis on the emotions of the users. The second dimension describes how the types interact with the users. Architectural and instrumental personalization are *individual*, and are concerned with the individual's interaction with the system, whereas commercial and relational personalization are considered *interactional* because they focus on the relations between multiple entities.

2.3 Personalization research

2.3.1 Web mining for personalization

Acquiring data points to improve the experience of each individual user is an important aspect of personalization. Eirinaki and Vazirgiannis examine a variety of methods for web mining in personalization in their work, reviewing the most popular methods (Eirinaki and Vazirgiannis, 2003). What follows is a brief summary of their findings. While some of the technology-specific descriptions of their work no longer applies, Eirinaki and Vazirgiannis highlight some of the most important modules of personalization and discuss common issue which may be encountered when developing personalized services. This information will be used to identify points of interest for the construction and evaluation of the artefact in this thesis.

User profiling is the process of gathering data about the individual viewer of a website. There are two ways of collecting this data, either explicitly or implicitly. Explicit user profiling is provided to the system by the user through means such as user registration forms and questionnaires, while implicit user profiling collects data without asking, for example by storing usage logs. Eirinaki and Vazirgiannis claim that the most important issue when it comes to user profiling is handling privacy concerns.

Web usage mining is used to identify trends and patterns in the user's interaction with the website, and is performed by applying statistical analysis to usage logs. This is often utilized in combination with user profiling for more effective personalization. Important issues linked to web usage mining are choice of data mining methods, how to process and filter data, as well as the method used to handle user identification.

2.3.2 The personalization process

Personalization can be performed cyclically in order to improve the personalized offering with each cycle. In their work, Adomavicius and Tuzhilin describe a process-oriented approach to personalization and identify the most important issues when developing this type of personalized application (Adomavicius and Tuzhilin, 2005).

Adomavicius and Thuzhilin divide the personalization cycle into three parts; *understanding*, *delivering* and *measuring*. Understanding the user is done by gathering information about the user and converting this knowledge into user profiles. In the delivery part of the cycle, each user profile is matched to the appropriate personalized offerings. Finally, measuring involves determining the effectiveness of the delivered offerings, which helps improve the user profile for the next cycle. An illustration of this personalization cycle can be found in figure 2.1.

When each part of the cycle successfully assists in the improvement of the next part, we achieve what Adomavicius and Thuzhilin call a *virtuous cycle*. In the virtuous cycle, the personalization improves over time by learning the preferences of the users and adjusting over time to reflect the current trends and changes in the market. The antithesis of the virtuous cycle is *depersonalization*, in which the effectiveness of the personalization decreases over time, or if the personalization cycle fails to adapt to a changing environment. Depersonalization is one of the biggest issues to consider when developing personalized systems, as users will not use the system if the personalized offering no longer reflects their personal preferences.

2.3.3 Personalization and HTML5

It is also important to examine similar previous works, both to ensure the originality of the research as well as to learn from what has been done before. Zhang et al. designed and developed a prototype of a context-aware mobile web application

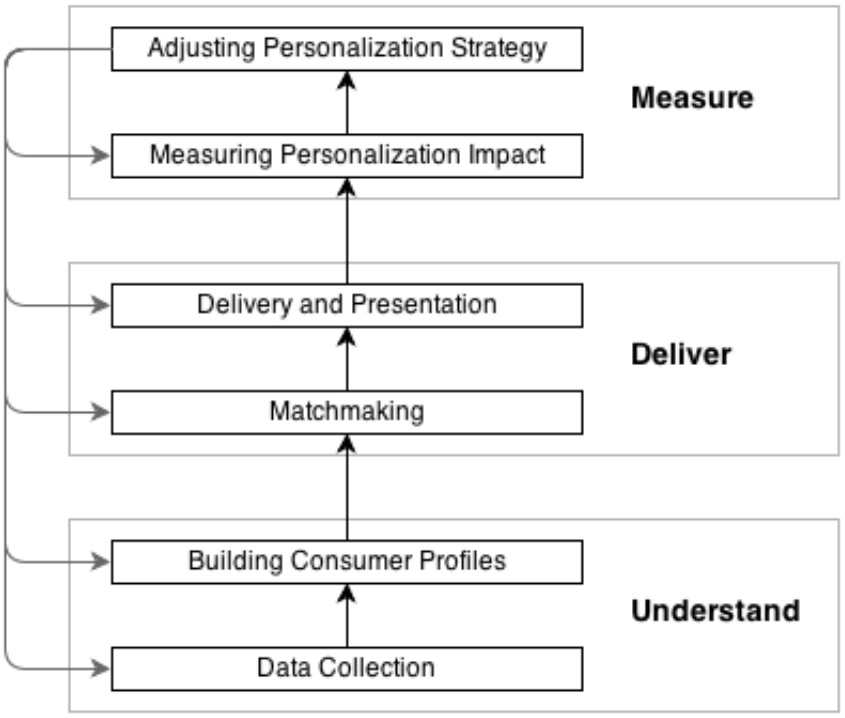

FIGURE 2.1: The personalization cycle of Adomavicius and Thuzhilin

(Zhang et al., 2012). Like the artefact of this thesis, their prototype was a personalized news service, although their application was developed solely for mobile devices. The application used time, location, usage logs, and current activity as the data points for the personalization, although the latter was not actually implemented into the prototype. The extent of the personalization included presenting the appropriate news content based on preferences, rearranging the order of the structure based on trends in usage logs, and recommending related content.

The extent of their use of HTML5 features such as geolocation and storage is ambiguous, as the PhoneGap framework (Adobe Systems Inc, 2008) is stated as being used to access these features natively on the mobile devices. Zhang et al. provide no evaluation or review of their usage of HTML5, but their work does seem to indicate that HTML5 satisfied the requirements of their prototype. Their work also shows that there is interest in the academic community to investigate the use of HTML5 with personalization.

2.4 The HTML5 stack

While HTML5 is the name of the newest version of the HTML language, it is also commonly used as an umbrella term that encompasses the three cornerstones of the newest generation of web technology, namely HTML5, CSS3 and JavaScript. Discussion on the capabilities of HTML5 generally refer to the combined use of these three technologies. In order to avoid ambiguity, the term *HTML5 stack* will be used to refer to the umbrella term.

2.4.1 HTML5

HTML (HyperText Markup Language) is used to define content structure on the World Wide Web, and is used in most websites today. HTML tags annotate the various component in a website, such as titles, paragraphs and images. This markup is then interpreted by the visitors' browsers, and displayed accordingly. While there are other technologies for defining website structure and layout, such as Microsoft Silverlight or Adobe Flash, most that use one of these opt for a hybrid solution with HTML.

Among the new features in HTML5 (Kesteren and Pieters, 2012) are several that are of particular interest for the purpose of personalization. The Web Storage API allows client-side storage, both persistent (localStorage) and session-based (sessionStorage). The Geolocation API allows applications to find the geographical coordinates of the device in use. As there are a multitude of methods to determining the position of the device, the Geolocation API can determine this based several different sources, including IP address, wireless network, cell towers, and dedicated GPS hardware. HTML5 is compatible with all modern desktop and mobile browsers, enabling multi-platform development with a single codebase.

2.4.2 CSS

CSS (Cascading Style Sheets) defines the presentation of websites and web applications. The most prominent new feature of CSS3 is its media queries, which enable responsive web design by allowing websites to change their appearance depending on the capabilities of the device viewing it (Meyer and Bos, 2011). New additions in CSS3 also includes several new features to improve the visual presentation of websites, such as rounded corners, box-shadows and webfonts.

LESS (Sellier, 2009) is a CSS pre-processor, written in JavaScript. It extends the functionality of CSS by adding variables, operators and functions to the language. This allows for much simplified modifiability in the stylesheet by enabling the definition of properties that are used in multiple elements, such as colours and dimensions, in a single variable. Another feature of LESS are mixins, which allow for the reuse of groups of defined properties. LESS is compiled to regular CSS, making it compatible with all CSS-compliant browsers. Several other CSS pre-processors with similar features exist, most notably Sass (Catlin, Weizenbaum, and Eppstein, 2006). LESS was chosen for this thesis due to the authors familiarity with it.

2.4.3 Javascript

JavaScript is a programming language used in websites for client-side scripting. Common usage includes dynamic page loading, form validation, and even interactive content such as games. It is an interpreted language with weak typing which supports the imperative, functional and object-oriented (prototype-based) programming paradigms.

jQuery (The jQuery Foundation, 2006) is the most widely used JavaScript library on the Internet, present in more than half of the most popular websites on the World Wide Web (W3Techs, 2013). Its main purpose is to enable rapid web

development by simplifying the client-side scripting of HTML. To accomplish this, it provides methods to select elements from the HTML DOM (Document Object Model), as well as the ability to alter the CSS style of web pages, and handling AJAX (Asynchronous Javascript And XML) for data transactions with servers.

Chapter 3

Methodology

This chapter describes the methodologies that were used in the creation of the prototype application. Design science was chosen as the research method as it was required by the task, and due to its suitability for prototype-based research. What follows is a description of how the work adheres to the Design science methodology, a description and reasoning for the prototype application requirements, along with the test plan for said application.

3.1 Design science research

The core concept of design science research is the creation and evaluation of artefacts. There are four types of artefacts in design science (Krogstie, 2012).

- *Constructs* form the vocabulary of a domain, and are used to describe the problems and solutions within it.
- *Models* are sets of statements that express the relationship between constructs.
- *Methods* are guidelines on how to perform a task, and are used with constructs and models to solve problems.

- *Instantiations* are realizations of artefacts, used to demonstrate that they can be implemented in working systems.

The focus of this project will be the creation and evaluation of an instantiated artefact, as well as evaluating the methods used in the creation of this artefact. Both the method and the instantiation will be evaluated, as both the quality of the end result and the ease of development is important when choosing a development platform.

In their seminal work, Hevner et al. suggest seven guidelines to help information science researchers conduct, evaluate and present design science research (Hevner et al., 2004). While Hevner et al. discourage rote use of the guidelines, as their application should vary based on the work in question, they state that each guideline should be addressed in order to ensure proper use of design science. The following describes how each guideline will be followed in this thesis.

Design science research must produce a viable artefact in the form of a construct, a model, a method, or an instantiation

This thesis will involve the creation and evaluation of two artefacts, an instantiation and a method. The instantiation will be a fully functional web application which will provide personalized news to users, and the method artefact will be the procedure used to create this instantiation.

The objective of design science research is to develop technology-based solutions to important and relevant business problems

The objective of this thesis is to examine solutions for personalized applications. Personalization has become an important concept for modern online businesses, with technology giants such as Google and Facebook gaining significant revenue from personalized advertisements. Other companies, like Netflix and Amazon, use personalization to advise their customers on what to watch or purchase, which

increases customer satisfaction and sales. Personalization is already a relevant business problem, and seems likely to gain even more importance in the future.

The utility, quality, and efficacy of a design artefact must be rigorously demonstrated via well-executed evaluation methods

Two artefacts will be produced and evaluated; an instantiation in the form of a personalized application, and the method used to create it. System testing will be performed on the prototype application to assess the utility of the artefact and uncover possible errors or problems. These tests are detailed later in this chapter.

The method will be evaluated through a qualitative review of the development process with an emphasis on the ease of use of the technology, the utility of the features relevant to personalization, as well as the complexity needed to fulfil the requirements of the case application. The code quality of the instantiated artefact will be verified by the use of static analysis with JSHint (Kovalyov, 2010).

Effective design science research must provide clear and verifiable contributions in the areas of the design artefact, design foundations, and/or design methodologies

The contributions will be the examination of the functionality and capabilities of HTML5 for development of personalized multi-platform applications, as well as an evaluation of the resulting instantiation. In other words, this thesis is primarily concerned with contributing to the methodologies of the field of personalization in information science research.

Design science research relies upon the application of rigorous methods in both the construction and evaluation of the design artefact

The requirements of the artefact should be representative of common use cases for personalization. To accomplish this, the requirements will be based on the features

of similar existing applications. Best practices will be used where appropriate in the construction of the artefact to ensure its rigour.

The search for an effective artefact requires utilizing available means to reach desired ends while satisfying laws in the problem environment

The creation of the artefact will be an iterative development process, with the architecture utilizing loosely coupled components to ensure high modifiability. This will enable the search for satisfactory solutions by allowing for easy replacement of functionality or technology which is found to not fulfil the requirements of the artefact.

Design science research must be presented effectively both to technology-oriented as well as management-oriented audiences

While the emphasis of this thesis is on the technology, it shall be presented as to advise both developers as well as management. The goal is to provide both parties with insight which will assist in the decision of whether or not to use HTML5 for their personalized application.

3.2 Case application

The artefact to be created for this thesis will be a multi-platform personalized news service. The reasoning behind this decision is that there is another project at NTNU on developing a news aggregation server, which will be utilized in the prototype application. This means that the work can be focused on the client-side aspect of development, which is the main objective of the thesis.

First, the planned artefact will be classified using Fan and Poole's scheme (Fan and Poole, 2006), in order to clearly relate the artefact to the overall field of personalization. The first dimension describes the aspects of the artefact to be

Personalized aspects	Content, user interface
Target	Individual
Automation	Implicit, explicit
Ideal types	Architectural, commercial

TABLE 3.1: Classification of the prototype application using Fan and Poole's scheme

personalized, where the content and the user interface would be the most pertinent aspects for a multi-platform news service. Personalization of content would be achieved by presenting news articles relevant to the user's personal preferences, and the personalized user interface would adapt to fit the device in use. The second dimension, the target of personalization, would be the individual user, as this adds some complexity when compared to targeting groups. In the third dimension, which is the degree of automation, the initial configuration shall be done implicitly based on the profile of the user, but allow for the user to explicitly alter their preferences. Additionally, personalization algorithms should be used to adjust the user's content over time based on their browsing activity, in order to achieve a virtuous personalization cycle in such a way that the content will adapt to changes in the user's interests. This combination of implicit and explicit personalization allows the user to get started quickly with a personalized system, while still providing transparency to the user as well as allowing him or her to correct mistakes made by the implicit personalization algorithm.

In terms of Fan and Poole's ideal types, this would be architectural and commercial personalization with an emphasis on the commercial aspect. The main purpose of a personalized news service is to present news content which is relevant to the interests of consumer, in order to gain user loyalty. Monetizing such a service would most likely be done by displaying personalized advertisement, similar to how companies such as Google and Facebook earn a significant portion of their revenue. The other aspect is the architectural personalization, which is primarily meant to ensure the usability of the user interface on all compatible devices, regardless of screen size and other hardware capabilities. However, architectural personalization could also be used to make the user more comfortable by altering

the visual appearance to suit the preferences of the user, for example by changing the colour scheme. The consolidated classification of the planned prototype application in Fan and Poole's scheme can be found in table 3.1.

Based on this description, a set of requirements have been compiled in table 3.2. The requirements have been categorized based on which research question they relate to, where requirements R1.X relate to RQ1, R2.X to RQ2, and so on. R1.X describes the requirements for context-based personalization, the most important of which is using the current location of the user to present local news. Requirements 2.X concern the personalization of the news content based on the preferences of the user. The most notable part of these requirements is the inclusion of both explicit and implicit personalization, which allows the users to tweak their personalized content beyond what an implicit personalization algorithm would be capable of, while still using the implicit personalization to provide both a baseline profile as well as adapting the profile over time based on user behaviour. This combination of implicit and explicit personalization is intended to reduce the risk of depersonalization. The final set of requirements is the architectural personalization, which is used to ensure compatibility and efficiency on all of the supported devices, from smartphones to desktop computers.

3.3 Evaluation of results

3.3.1 Test plan

System testing will be performed in order to validate the functional requirements of the prototype application. The system test consists of black box tests that validate the entire application, and should be performed with the application running in its required environment (Braude, 2001). While it is not feasible to test every possible configuration, the application should be tested on a variety of browsers and devices. Tables 3.3 through 3.9 contain the test cases for the prototype application.

R1.1	The application shall present relevant news based on the current local time of the user.
R1.2	The application shall present relevant news based on the current location of the user.
R2.1	The application shall present relevant news based on the available user profile, which includes personal data such as age and sex, as well as their personal preferences in news categories.
R2.2	The application shall be able to acquire personal information from social networks such as Facebook and Google+.
R2.3	The application shall store the personalized preferences of each user persistently.
R2.3	The application shall automatically update the personalized preferences of each user based on their browsing activity in the application.
R2.4	The application shall allow the user to manually add or remove desired news categories in their stored profile.
R3.1	The user interface shall be adjusted to the device used to view it, including computers, tablets and smartphones.
R3.2	The user interface shall be suited to both touch and mouse input.

Table 3.2: Requirements of the case application

Test ID	T1.1
Requirements	R1.1
Procedure	1. Start application 2. Navigate to personalized news 3. Verify that time-based news are displayed 4. Navigate to configuration panel 5. Disable time-based news 6. Navigate to personalized news 7. Verify that time-based news are no longer displayed 8. Close application

Table 3.3: Test case T1.1

Test ID	T1.2
Requirements	R1.2
Procedure	1. Start application 2. Navigate to personalized news 3. Verify that location-based news are displayed 4. Navigate to configuration panel 5. Disable location-based news 6. Go to personalized news 7. Verify that location-based news are no longer displayed 8. Close application
Notes	This test needs to be run multiple times on each device due to the various means available for the HTML5 Geolocation API to determine location. This includes any of the following: LAN connection, Wi-Fi connection, 3G connection, and GPS.

Table 3.4: Test case T1.2

Test ID	T2.2.1
Requirements	R2.1, R2.2
Procedure	1. Start application 2. Navigate to configuration panel 3. Connect to Facebook profile 4. Verify that categories were added to favourites 5. Go to personalized news 6. Verify that the new categories are displayed 7. Close application

Table 3.5: Test case T2.2.1

Test ID	T2.2.2
Requirements	R2.1, R2.2
Procedure	1. Start application 2. Navigate to configuration panel 3. Connect to Google+ profile 4. Verify that categories were added to favourites 5. Go to personalized news 6. Verify that the new categories are displayed 7. Close application

Table 3.6: Test case T2.2.2

Test ID	T2.3
Requirements	R2.3
Procedure	1. Start application 2. Navigate to configuration panel 3. Add all categories to favourites 4. Toggle off location and time based news 5. Restart application 6. Verify that the configuration persisted through the restart 7. Close application

Table 3.7: Test case T2.3

Test ID	T2.4
Requirements	R2.4
Procedure	1. Start application 2. Navigate to configuration panel 3. Add all news categories to favourites 4. Navigate to personalized news 5. Verify that added categories are displayed 6. Navigate to config panel 7. Remove all news categories from favourites 8. Navigate to personalized news 9. Verify that removed categories are not displayed 10. Close application

Table 3.8: Test case T2.4

Test ID	T3.1
Requirements	R3.1, R3.2
Procedure	1. Start application 2. Verify that news articles are displayed correctly 3. If the device is a PC, resize the window down until only 1 news article is displayed per row 4. If the device is a smartphone or tablet, flip it to landscape view 5. Verify that news articles are displayed correctly 6. Close application

Table 3.9: Test case T3.1

3.4 Anticipated results

The design science research method will be used to create and evaluate two artefacts. The result of this effort is expected to be the following three items.

- A open-source personalized web application.
- The method used in the creation of personalized applications using HTML5.
- An evaluation of the application and method used to create it.

The personalized web application will be a news service with the primary of objective of providing its users with news tailored to their profile and preferences. The source code of this application will be made open source, for the benefit of anyone who wishes to utilize it in their research. The method is provided so that anyone may reproduce this work in order to confirm its validity, should this be desired. The evaluation of the application and method is intended to help with the decision of whether or not to utilize HTML5 and related technologies for the creation of a personalized application, by informing the reader of the strengths and weaknesses of this approach.

3.5 Ethical issues

While this project takes advantage of the news aggregation and search server of a third party, there is no direct involvement which could cause a bias. Relevant ethical issues are the use of references and sources for the report, along with the use of frameworks and other third party software in the creation of the prototype application. To ensure clarity, everything which originates from a third party, both text and code, will be attributed to its original creators with proper reference and credit.

Chapter 4

Architecture and implementation

This chapter contains the bulk of the knowledge and results gathered from the development process and resulting application. It provides a basic explanation of the features of the prototype application, along with descriptions of its architecture and implementation process. Also included are the details and results of the testing process.

4.1 Prototype application

4.1.1 Single-page application

In order to provide a fluid user experience similar to what one would expect from a native mobile or desktop application, the prototype was created as a single-page application. A single-page application is a web application that only needs to load a single HTML document, and does not require page refreshes during use. Without the need to load pages during interaction, response time is vastly decreased and network latency is less noticeable. This was accomplished by making the application data-driven, with the news data being retrieved from the news aggregation server with AJAX.

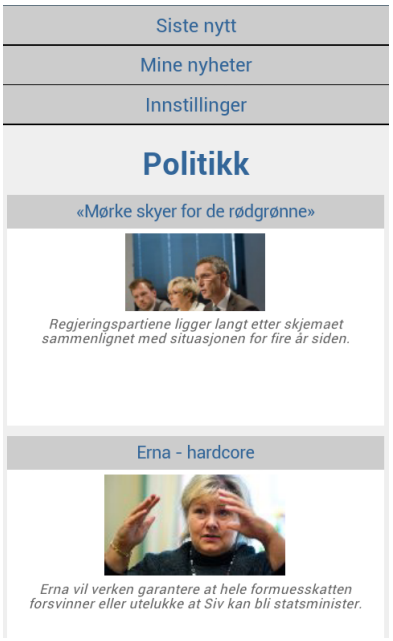

FIGURE 4.1: Screenshot of the prototype application (smartphone portrait)

4.1.2 Personalized user interface

Screenshots of the application can be found in figures 4.1 through 4.4. In terms of the personalized interface, there are two components which change based on the capabilities of the device being used. The first component is the top menu, which is presented as a bar with horizontally aligned buttons on tablets and workstation PCs, but changes to vertically aligned buttons on smaller screens such as those of smartphones in order to ensure that the buttons are easily clickable regardless of the device being used.

The second component which is subject to architectural personalization is the display of news articles. Here, the amount of articles in each row adapts to the width of the screen. This varies all the way from 4 articles per row on desktop PCs and landscape tablets, down to 1 article per row on smartphones in portrait mode.

FIGURE 4.2: Screenshot of the prototype application (smartphone landscape)

FIGURE 4.3: Screenshot of the prototype application (tablet portrait)

4.1.3 Personalized news

The news personalization is the core feature of the prototype application, and is achieved through several means. Firstly, the users' actions in the application are tracked in order to determine which categories of news articles they prefer

FIGURE 4.4: Screenshot of the prototype application (tablet landscape)

reading. Each news category is assigned a numerical weight that increases as the user browses and reads related news, but these weights also decay over time. The intention of this system is to ensure that the personalized news are up to date on the user's current interests, as categories will be added to the personalized news section as the user finds new categories of interest, or be removed if the user loses interest in said category. While this is a simple algorithm, it is still a functional implementation of personalization based on the virtuous cycle.

Secondly, the application will present context-appropriate news personalized to the user. This includes time-based news, which display news based on the current local time of the user. The time-based news in the prototype application are not particularly sophisticated, but were included to test the technological capabilities of the platform and verify that it would indeed be possible to use the built-in *Date* objects for this purpose. This built-in functionality does indeed seem apt for use in more complex algorithms to not only find what kind of news the user wants, but also when they want a certain type of news articles. Location-based news is the other method used to present context appropriate news. The HTML5 Geolocation API is used alongside a third party geographical database in order to present the user with news from their current location.

The prototype application can also tap into social networks Google+ and Facebook. The benefit from this is two-fold. It allows the application to get a reasonable starting point to determine which categories are of interest to the user based on their profile, and also acquire some additional points of interest, such as the users' home town.

Finally, the user may also add or remove news categories manually, in order to set a starting point for their preferences or to remedy any possible mistakes made by the personalization algorithms. This means that the users themselves can prevent depersonalization, should it occur.

4.2 Architecture

4.2.1 Revealing Module pattern

Unlike certain languages like Java and C#, JavaScript places very few restrictions on the structure of the code. One stated requirement for the application was that the functional components should be loosely coupled, which would allow for easy replacement of technology which is not deemed suitable, as well as making it reusable for future work. The Revealing Module pattern was implemented to accomplish a modular structure, while keeping in line with the goal of using best practices for the implementation of the artefact. An improved version of the Module pattern, the Revealing Module pattern emulates concept of classes as used in other programming languages, but allows for a more consistent syntax (Osmani, 2012).

Simply put, a module in the Revealing Module pattern is a variable containing a self-executing function, where said function contains the variables and functions of the module. To expose functions and variables to the public scope, it returns an anonymous object containing pointers to the functions and variables that should

be revealed. A more detailed explanation of the Revealing Module pattern complete with a code sample can be found in Appendix A.1.

4.2.2 Application structure

Figure 4.5 illustrates the structure of the system by the use of a class diagram. While there are obviously no actual classes in the JavaScript code, the classes represent the previously described modules. Similarly, the keywords *public* and *private* do not exist in JavaScript, but are used in the diagram describe whether or not the functions and variables are exposed and thus available outside its module. The arrows illustrate the call hierarchy, showing which modules call on each other. The arrows show calls from one module to another, so an arrow from *viewModel* to *newsHandler* means that *viewModel* uses the exposed functions of *newsHandler*.

The prototype application is connected to several third party services in order to provide personalized news. Figure 4.6 shows the communication between the prototype application and the external services it utilizes. The news aggregation server is responsible for gathering news from Norwegian newspapers, and provides the desired news to the prototype application. Facebook and Google+ APIs allow users to connect to their social media profiles to improve personalization. GeoNames WebServices are used to enable local news by converting geographical coordinates to place names.

4.3 Implementation

4.3.1 Development environment

Sublime Text (Skinner, 2008) was used as the IDE for working with JavaScript, HTML and LESS. It was chosen for its customizability and useful community-created packages, with Sublime Package Control (Bond, 2011) being used to install

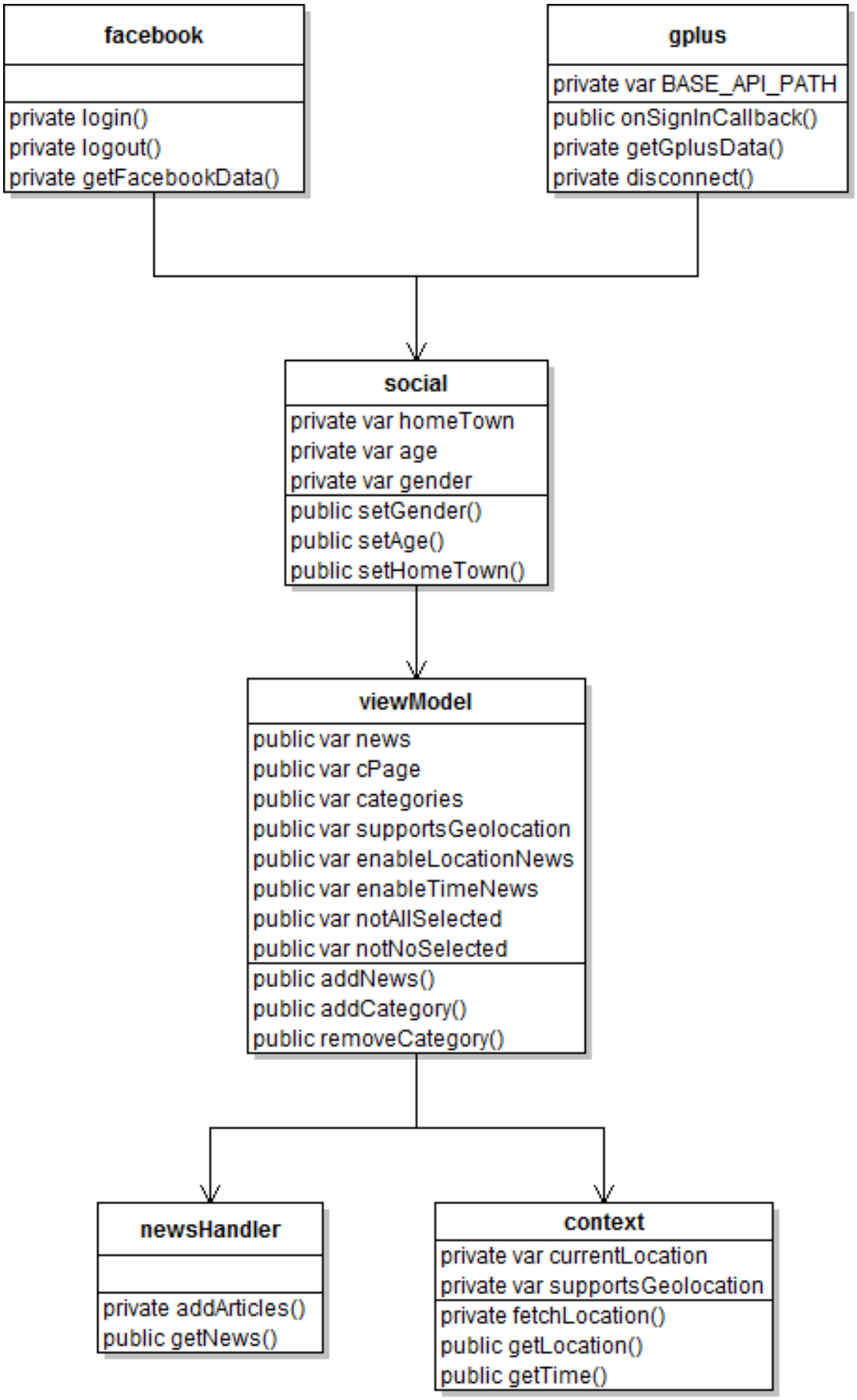

FIGURE 4.5: Class diagram illustrating the modules of the prototype application and the relations between them.

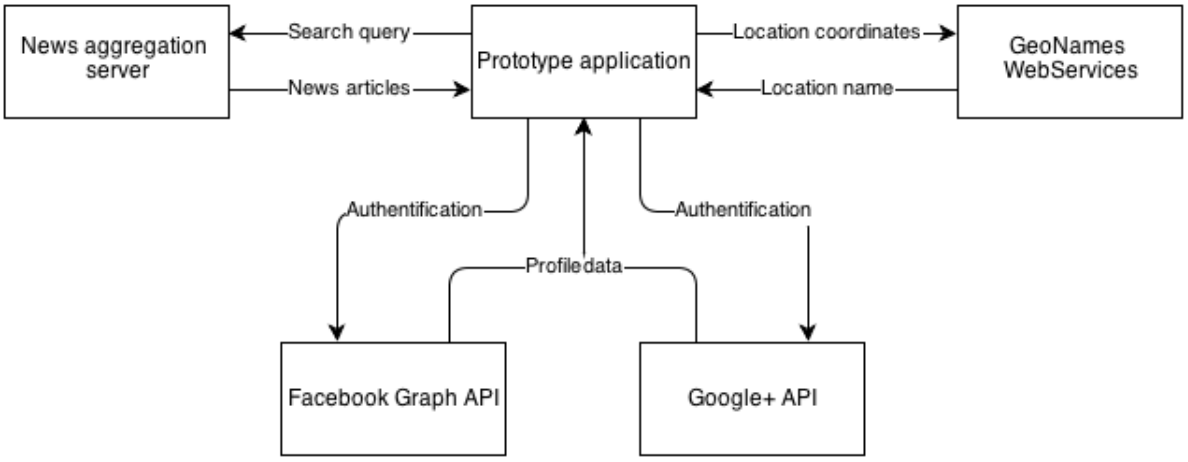

FIGURE 4.6: Illustration of communication between the prototype application and external services.

and update said packages. The most important packages used were LESS and LESS-build, for syntax highlighting and compilation of the LESS code, as well as SublimeLinter for automatic linting of JavaScript code using JSHint. The development server was a Raspberry Pi (The Raspberry Pi Foundation, 2008) running a lighttpd (Kneschke, 2003) webserver on the Arch Linux OS.

4.3.2 Single-page application

Two JavaScript frameworks were used to assist in the development of the single-page aspect of the application. Knockout.js (Sanderson, 2010) implements the MVVM (Model View ViewModel), allowing developers to link HTML DOM elements to data properties and automatically updating them whenever the model changes. In the prototype, it handles the display of the news articles fetched from the server, as well as allowing the user to change between the configuration menu and news display without reloading the page. Sammy.js (Quint, 2009) handles the routing of the web application, mapping URLs to view model states. This means that the application follows standard web site behaviour even though the page is only loaded initially, so that the back and forward buttons work as expected and the users can link to specific parts of the application or save certain states as bookmarks if desired.

4.3.3 Revealing Module pattern

The Revealing Module pattern was implemented as described in the Architecture section, with the application prototype consisting of four main modules, namely *viewModel*, *context*, *newsHandler* and *social*. The *viewModel* module contains the view model used by Knockout. The next module, *context*, contains the functionality used to determine the context, i.e. the current time and location. *newsHandler* is responsible for the interaction with the news aggregation server, retrieving news articles and structuring the objects to be used by the view model. Finally, the *social* module handles personal information acquired from social networks and uses this data to determine which news categories are of interest to the user. Two additional modules were added, *facebook* and *gplus*, which handle connections and API calls to Facebook and Google Plus respectively. This includes connecting to their respective APIs, authorizing user login, and retrieving personal information.

4.3.4 User interface

In order to keep the structure of the application as clean as possible, the interface should only be defined and altered in the style sheets. The Semantic Grid System (Tate, 2011) for LESS was used to implement a grid system cleanly, without the need for special markup in the HTML structure or manipulating the DOM in the application scripts. CSS media queries are used to make the grid responsive, and to adapt the rest of the user interface to the device in use. The following four layouts were defined with the help of media queries.

- Desktop PC and Landscape tablet
- Portrait tablet
- Landscape smartphone
- Portrait smartphone

CSS provides information on the orientation, resolution, pixel density and many more properties of the device in use. This means that there is a large variety of means available to distinguish between devices using media queries. However, due to inconsistencies with the reported values as well as compatibility issues, it became apparent that using only the *width* property provided the best overall results. That is not to say that this simple approach is not without its issues, as differences in resolutions and reported values vary significantly between the devices currently on the market.

In order to efficiently develop and test media queries, the web application The Responsinator (Pugsley and Hovey, 2012) was used to simulate the appearance of the application on multiple devices. This provided a good approximation that saved a lot of time, but does not necessarily show device-specific bugs and issues. As such, it is not a replacement for proper testing on actual devices, which will be done as part of the system tests. Despite of this, it proved to be a helpful tool for finding good baselines for the media queries.

As mentioned in Chapter 2, one of the most appealing features of HTML5 is its wide compatibility, allowing it to run on most modern devices with a browser. However, as all web browsers do not use the same engine, they sometimes interpret JavaScript and CSS different from one another, which makes it difficult to get the web application to render identically on each platform. This also means that browser-specific bugs can appear, causing the application not to work properly on some platforms or browsers. This was one of the most significant challenges in the implementation of the prototype applications, with CSS inconsistencies being particularly tricky to debug.

4.3.5 News

The news aggregation and search server provides JSON as one of its data formats, which makes it ideal for use with a HTML5 web application. The application

sends AJAX requests to the server, which asynchronously returns the requested articles. jQuery provides functions for sending these AJAX requests, which saves time when implementing this kind of functionality.

The usage of the news server is fairly simple, with the prototype application sending a query string and the amount of desired articles, and the news server returning the newest articles matching the search query. This means that the selection of articles is fairly simple, as the server only supports selection by a single search string. The main weakness of this simple selection is that some ambiguous queries return results that were not intentional.

An example of such an ambiguous query would occur if the application determines that news from the municipality of Ski in Akershus is of interest to the user, which will prompt the application to request news with the search query of *ski*. Such a query is not likely to find local news from Ski, as most of the search results would be articles on cross-country skiing. This could be prevented by filtering search results on the client-side, but the choice was made to not do so in order to not slow down the application. The news server is a work in progress, and when category-based filtering is added this will no longer be an issue.

4.3.6 News personalization

As the emphasis of this project is on examining the web technology surrounding the HTML5 stack, the implemented personalization algorithms are very basic. Each news category is assigned a weight which represents its relevance to the interests of the user. On the personal news page of the application, news above a certain weight threshold are shown along with the context-appropriate news which are described later in this chapter. The user's actions in the application are tracked in order to increase the accuracy of these weights, by increasing the weight of related news whenever the users opens a certain category or article, while simultaneously decaying the value of unrelated weights. This means that

Storage	**Data**
Memory	Home town, age, gender
Session storage	Location
Local storage	Personalized news categories, configuration preferences

TABLE 4.1: Storage methods used in the prototype application

the weights will grow more accurate over time. Additionally, users may manually add or remove categories from their favourites list through the configuration panel.

Persistence of settings and favourite categories is handled using the *localStorage* and *sessionStorage* functionality of the HTML5 Web Storage API, which offers a method to persistently store data locally on the device. This was simple to implement as only two functions are needed to use the API, one for storing data and one for loading it. Data is stored in key/value pairs, where the values are always stored as strings. Serialization in the prototype application is done by storing data objects as JSON strings. JavaScript provides functions to parse from and to JSON format, and Knockout.js does the same for its special observable objects. Appendix A.2 contains a brief example showing how the serialization with HTML5 Web Storage is handled in the prototype application. One important point to note is that Web Storage is not secure, and as such should not be used to store any personal information. Table 4.1 shows method of storage is used for each kind of storage.

4.3.7 Context awareness

As expected, it was trivial to implement the time-specific aspect of context awareness. JavaScript provides build-in functionality to find the current local time through the use of *Date* objects. Similarly, HTML5 provides the Geolocation API to determine the geographical position of the user. This results in a set of coordinates with a longitude and latitude that corresponds to the location of the device in use. In order to provide meaningful semantic data which can be used to find relevant news, reverse geopositioning is used to find the name of the location. This requires the use of a geographical database. The GeoNames (Wick and Boutreux,

2005) geographical database was chosen for the prototype application because it is a free API available under the Creative Commons licence. GeoNames' web service provides JSON responses, which can be easily handled in HTML5 via the use of jQuery. The other geographical databases that were considered are Google Maps and Microsoft's Bing Maps. See Appendix A.3 for a detailed explanation of how HTML5 Geolocation was used with GeoNames to find semantic location data.

4.3.8 Integrating social networks

Facebook and Google both provide JavaScript APIs for accessing their social networks, making them readily accessible for HTML5 applications. Facebook integration is more beginner-friendly to implement, as a variety of tutorials are provided on how to access the functionality of the API, whereas Google only provides sparsely commented code samples. Both Facebook and Google require developer accounts to access their APIs, but sign-up and usage of both is free of charge.

The only major concern that was the discovered in terms of the social network integration was that Google Plus only provides access to public data, with no available method for the application to ask the user for permission to access specific data fields from their Google Plus account, as is the case with the Facebook API. This means that if a user wishes to share any of their personal details with an application, they must make that information available to anyone on the Internet, which will raise privacy concerns for many. Overall, this means that it will be more difficult to gather relevant personal data from Google Plus accounts than to get the same information from Facebook.

Chapter 5

Test results

This chapter describes the executions of the system tests along with their results. The issues that were discovered will be used as part of the discussion and evaluation of the prototype application.

5.1 System testing

5.1.1 Test execution

The devices used for testing are listed in table 5.1. The devices cover all the most popular modern mobile operating systems, and both tablets and smartphones are represented. Additionally, both low resolution and high resolution devices are included, to ensure that the user interface is compatible with both newer and older hardware.

5.1.2 Test results

Tables 5.2 through 5.8 list the result of the system tests, with each table containing one issue that was discovered during system testing. These tables describe the issue

	Tablet	**Smartphone**
Android	Samsung Galaxy Tab 10.1	Samsung Galaxy S2
iOS	iPad 2	iPhone 4
Windows Phone	-	Nokia Lumia 920

Table 5.1: Devices used in tests

Issue ID	I1
Test ID	T2.2.1
Device	Desktop PC
Description	Personalized categories derived from the personal data from Facebook are not immediately added to the configuration screen upon logging in to Facebook, but are added once the web application is restarted.
Cause	The application was relying on a function to retrieve personal data and add the categories which was no longer being used due to a last minute change to the Facebook login implementation.
Solution	Added an event listener which retrieves personal data and add categories when the Facebook login event is received.

Table 5.2: System test report I1

Issue ID	I2
Test ID	T2.2.1, T2.2.2
Device	Nokia Lumia 920
Description	Login windows for Google+ and Facebook open in the application window instead of in a new window as they should. The login still works, but the application will have to be reopened, as the login effectively closes the web application.
Cause	This seems to be an issue with the Internet Explorer browser on Windows Phone 8. The standard methods for forcing pages to open in new tabs or windows do not seem to work, and as such the news articles would not open in new tabs either.
Solution	No solution for this issue was found. This is an issue that should be solved by the browser developers eventually. As Windows Phone 8 is one of the newest mobile platforms, issues should be expected, but it should hopefully be resolved in the near future.

Table 5.3: System test report I2

and what caused it, along with any discovered solutions to said issue, specifying which solution was used for the prototype application.

Issue ID	I3
Test ID	T1.1
Device	Apple iPad 2
Description	The checkboxes in the configuration menu are not clickable.
Cause	This is a bug specific to older version of iOS, in this case version 5.1.1. The bug simply causes form labels to not be clickable.
Solution	This issue was solved by updating the iPad 2 to the newest version of iOS. There is also a workaround for this issue which involves adding an empty *onclick* attribute to the labels, which will make them clickable in the old versions of iOS.

Table 5.4: System test report I3

Issue ID	I4
Test ID	T2.2.2
Device	Galaxy Tab
Description	Google+ authorization access denied due to origin mismatch.
Cause	The application was opened with the URL *http://cgj.me* instead of *http://www.cgj.me*, which was not authorized.
Solution	Added *http://cgj.me* to permitted origins in the Google API console for the prototype application.

Table 5.5: System test report I4

Issue ID	I5
Test ID	T1.1
Device	Apple iPhone 4
Description	No personalized news are displayed in the *My news* section.
Cause	This issue was caused by an erroneous conditional statement in the view model.
Solution	The conditional statement in the view model was corrected.

Table 5.6: System test report I5

Issue ID	I6
Test ID	T1.2
Device	Apple iPhone 4
Description	Location-based news are not displayed in the *My news* section.
Cause	Location Services were disabled on the iPhone 4 used in the test.
Solution	Enabling Location Services through the settings menu of the iPhone 4 solved this issue.

Table 5.7: System test report I6

Issue ID	I7
Test ID	T3.1
Device	Apple iPhone 4
Description	The text in the top menu is larger than expected when the device is in landscape mode.
Cause	The scaling of font size seems to be an intentional feature of Safari on iOS.
Solution	It is possible to alter this behaviour through the use of the vendor-specific CSS property *-webkit-text-size-adjust*. However, this is not desired for the prototype application both because vendor-specific CSS is contrary to the standards, and because overriding default platform behaviour could mean that the application will not behave as expected by the users of that platform.

Table 5.8: System test report I7

Chapter 6

Evaluation and discussion

In this chapter, the results are discussed and elaborated upon. The chapter is divided into three parts, one for each research question. The intent is to discuss both the positive and negative aspects of the findings for each research question in order to provide a complete evaluation of the results.

6.1 RQ1: Context-aware personalization

The prototype application implemented two forms of context-awareness, namely time and location. No issues were found to affect the time functionality during system testing. Issue I6 was the only negative impairment on the location functionality, and was caused by factors outside the control of the prototype application. However, regardless of being outside of the applications control, it must still be handled in order to provide the optimal experience to as many users as possible. One way of accomplishing this would be to inform the users that the device and/or browser in use needs to have Location Services enabled in order to allow the application to provide location-based news. Overall, the HTML5 Geolocation API is a powerful and easy to use feature that should be useful to many context-aware personalized applications. It embraces the strength of web applications, the vast array of supported platforms, by utilizing the best method of determining

position available to the device, such as GPS, cell tower positioning, IP and Wi-Fi.

However, modern mobile devices have hardware capabilities other than GPS which could provide useful data points for personalization, but these are not readily accessible for use in web applications. For example, the camera found on most phones could be useful in many personalized applications, by reading QR codes or using image recognition to identify landmarks. There could also be a use for the accelerometer, for example to identify that the user is currently performing a physical activity such as jogging.

The problem for HTML5-based web applications is that they may not access device functionalities such as the ones mentioned, whereas native mobile applications may do so. However, the W3C is working on amending this. The DeviceOrientation Event Specification provides events for obtaining information about the movement and orientation of the device in use, obtaining this information from sources such as gyroscopes, compasses and accelerometers (Block and Popescu, 2012). HTML Media Capture will enable user access to the media capture functionality of devices, including cameras and microphones (Kostiainen, Oksanen, and Hazaël-Massieux, 2013). These working drafts and many like them are still being developed by the W3C, which shows that while the pure HTML5 approach may not yet be a viable options for all personalized applications, there is the promise of even better support for personalized applications in the future.

One option for developers that require device functionality not accessible to HTML5, is to create a hybrid application to attain the best of both worlds. Hybrid applications fuse the HTML5 core with a native layer for each platform, enabling the wide compatibility of a web application with access to all the features of a native application, although this does come at a slight cost in terms of development time and additional maintenance. There are also hybrid frameworks such as PhoneGap (Adobe Systems Inc, 2008) which provide their own native wrappers to simplify

the development of hybrid applications, although these also come with an attached cost in the form of restrictions on the code structure an architecture of the application.

6.2 RQ2: Persistent user profile

In terms of its purpose, HTML5 Web Storage was found to perform admirably. It is simple to use and requires a minimal amount of code, which makes it very quick to implement. While it may only be capable of storing strings, serialization is made convenient by the built-in functions provided by JavaScript and jQuery. The downside is that it does not provide any form of security, which means that no sensitive information should be stored using the HTML5 Web Storage, as it could potentially be accessed by malicious parties. Another point of note is that it could be desirable to store user profiles online so that users may access their profile across all of their devices, and as HTML5 Web Storage only supports storage locally on the device, it will not be of any use when implementing server-side storage.

There is also functionality which was not included in the prototype application that could be of interest to other personalized applications. Mobile devices have more functionality that could provide data points for personalization. The contact list of phones and tablets could be used to tie profiles together, allowing for an easily accessible social element sharing news and the like. The calendar could be used to find news related to travelling and appointments in the near future. These are just a few examples, there will be more possibilities for such features as smartphones and other devices become ingrained into our daily lives. While it may not be possible currently, the W3C does intend for this functionality to be available in the future, as evidenced by their Device APIs and Policy Working Group Charter (Hazael-Massieux and Roessler, 2011). If this functionality is desired for current development, the solution is yet again to create a native or hybrid application, and it should be noted that the previously mentioned hybrid development framework PhoneGap also supports these features (Adobe Systems Inc, 2008).

As for acquiring personal data, both Facebook and Google+ support the HTML5 platform well by providing their own respective JavaScript APIs, with a wealth of functionality and data available to be used for the purpose of personalization. The sole concern discovered during the social media integration was that the Google+ API only provides access to the publicly accessible data of the user's profile. This means that users are unable to grant applications access to any given part of their Google+ without also making it visible to the public. This is not the case with Facebook, which allows users to grant third party applications permission to access their profile after asking for specific data.

During system testing, only one issue was found to affect the social network integration. Issue I2 arose from how the Internet Explorer browser on Windows Phone 8 treats hyperlinks which are supposed to open in a new window, and was not specific to the Facebook, Google+ APIs or their implementation in the prototype application.

6.3 RQ3: Personalized presentation

While CSS does its job for the most part, it can get clunky when working with larger and more complex systems. LESS proved to be very helpful in that regard, as it adds tremendous convenience and modifiability in allowing for the of use variables as well as the reuse of defined sets of properties. There are still issues with creating the presentation in some cases, as certain layouts can be tricky to implement using CSS, with vertical centring of some types of elements being a particularly notorious example of this. Apart from these minor issues, the main challenge in creating the personalized presentation of a multi-platform application is ensuring that the user interface works correctly on all platforms, as well as making the presentation consistent on all supported devices.

Issues I3 and I7 from the system tests are examples of the challenge involved

in creating multi-platform applications with HTML5. The issue found in I3 was that the styled checkboxes in the configuration screen did not work correctly on the iPad 2 due a bug in iOS which prevented form labels from being clickable. There will often be bugs and issues specific to one platform or browser due to many of them having differing implementations of the standards. Issue I7 provides an example of the other side of the same coin, where one browser intentionally behaves differently than the others, as opposed to Issue I3 which was caused by a bug that has been fixed in newer versions of the OS.

When a browser intentionally alters a certain aspect of the browsing experience in this way, it presents the developer with an interesting conundrum; should the default behaviour of the browser be altered in order to achieve consistent behaviour across all platforms, or should it be left unchanged to conform with the intended browser experience. There really is no right or wrong in these situations, as both providing the same experience on all platforms and conforming to platform standards is done with the purpose of making application behaviour follow user expectations. A decision would have to be made on a case-by-case basis, although leaning towards not altering default platform behaviour would wise, as vendor prefixes and platform-specific code will increase the size and complexity of the application codebase, increasing the cost of testing and maintenance of the application.

Media queries provide a method to adapt the user interface of an application to the capabilities of the device in use. For the most part, the standard method of using the width of the device in pixels works well, although there are some exceptions. First of all, it should be noted that pixels in CSS do not necessarily map to the physical pixels of the display device (Bos, 2013). As an example, take the original Apple iPhone, which has a resolution of 320 by 480. In CSS, the original iPhone has a width of 320px in portrait and a width of 480px in landscape mode. For the iPhone 4, Apple doubled the resolution of the screen to 640 by 960, but it still has the same pixel dimensions in CSS, but with double the pixel density.

This is a good thing, as both the original iPhone and the iPhone 4 have 3.5 inch screens, so for most purposes they should use the same layout.

Not everything is as consistent as this, however, as there is a plethora of devices and manufacturers on the market, and there seems to be no overall agreement on any form of standardized ratio between CSS pixel dimension, display resolution and physical display size. This can make it troublesome to distinguish between certain devices, even though their actual specifications may be very similar. As an example, the Nokia Lumia 920 is a modern flagship smartphone with a large high resolution screen, yet it has fewer CSS pixels than smaller and lower resolution devices such as the Samsung Galaxy SII. However, another modern flagship device, the Samsung Galaxy SIII, sports more CSS pixels than its previous incarnation, and consequently also far more than the Lumia 920 even though the two have very similar specifications.

The media queries of the prototype application were found to utilize the correct layout for all of the test devices, including portrait and landscape modes where applicable, so it would seem that it is possible to properly categorize most devices based on common CSS pixel widths. However, there is no guarantee that these media queries work flawlessly, as it is not feasible to test the application on every device on the market. Unless efforts are made by the handset and tablet manufacturers to standardize a ratio between physical dimensions and CSS pixel dimensions, media queries will have to grow more and more complex over time to account for all the devices which do not follow the norm. While CSS provides numerous methods for distinguishing devices, such as pixel density, orientation and aspect ratio, there is still a need for the device manufacturers to provide consistent and meaningful values for these to be of any use.

Chapter 7

Conclusion and further work

This chapter concludes the thesis, summarizing the most notable findings. As with the previous chapter, the conclusion is divided into one section per research question. This is followed by a conclusion on the benefits and challenges involved when developing multi-platform personalized applications with HTML5. Lastly, the limitations of the thesis are described, along with suggestions for further work based on the findings of the thesis.

7.1 Conclusion

7.1.1 RQ1: Context-aware personalization

The main feature of HTML5 for context-aware personalization was the Geolocation API, which was implemented in the prototype application and found to perform admirably. The only issue related to the usage of the Geolocation API was that device-wide preferences can prevent its usage without notification of the user. However, as long as the user is informed that the device need to permit the browser to use location-based services in order to take advantage of this feature, there seems to be no reason not to utilize the Geolocation API for personalized web applications. The time-based personalization based on the built-in Date objects

of JavaScript also worked as expected, and no issues were discovered during the system testing of this functionality.

7.1.2 RQ2: Persistent user profile

The HTML5 Web Storage API provides basic functionality for client-side persistent storage. It is extremely simple to use and implement, but its main drawback is the lack of secure storage. Due to the lack of security, it is crucial not to use it to store any form of sensitive information, as it could be misused by malicious websites or software. While HTML5 Web Storage may not be as useful if personalization data needs to be synchronized across clients or devices, it remains a simple to use tool for persisting device-specific configuration and the like. This is a feature that should be useful to most developers intending to develop a personalized multi-platform application with HTML5.

7.1.3 RQ3: Personalized presentation

The media queries of CSS3 proved to be a useful feature to enhance the inherent ease of developing multi-platform applications with HTML5, by providing the means to adapt the user interface to suit the capabilities of the device in use. There may be some potential challenges in unambiguously determining the actual capabilities of every device in the future, as there is no standardized ratio between the physical and CSS dimensions yet. For now, the capabilities of the media queries seem sufficient for their purpose, and were found to work as planned in the development and testing of the prototype application.

7.1.4 Developing multi-platform personalized applications with HTML5

While one of the most notable advantages of HTML5 is its wide platform compatibility, this does not mean that it allows for effortless development of multi-platform

applications. The issues found during the system testing of the prototype application clearly show that extensive testing is required to ensure full compatibility, due to the variations between the JavaScript and CSS interpretations of the different web browser engines currently on the market. There is also the simple fact that HTML5 web applications do not yet have the wide support for native device functionality that native applications do. As has been mentioned previously, HTML5 is still in development, and the W3C is working on bridging the gap between native and web by creating a host of APIs which enable access to native functionality.

Overall, HTML5 is a great tool for creating multi-platform personalized applications, but it should be kept in mind that it is not a magic bullet solution to every project. However, if the requirements do not go beyond the capabilities of HTML5, it can provide an accelerated development process with cheaper maintenance when developing multi-platform personalized applications, compared to the alternative of creating and maintaining native applications for each target platform.

7.2 Limitations

The prototype application was specific to a domain, that being news service providers. While no other domains were considered during the development of the prototype, the findings and results of the thesis should be applicable to most context-aware personalization work, as none of the technology is domain-specific.

As the emphasis of the thesis was to investigate the capabilities of HTML5, the personalization algorithms that were implemented are only rudimentary in function. This limitation had to be imposed due to the limited time frame of the thesis. However, this should not have had any impact on the answers to the research questions.

7.3 Further work

The thesis and its findings provide several avenues for further work. First, there is the topic that was examined. While a thorough examination of the relevant technological capabilities of HTML5 for the purpose of personalization was performed, there are still some details that were left out which could be of interest. Personalized presentation was used in the prototype application to adapt the user interface to the device being used, but it could also be interesting to investigate the capabilities of HTML5 for adapting the aesthetics to the preferences of the user.

There is also the matter of usability. While single page web applications such as the prototype may emulate the behaviour and aesthetics of native applications, they're not the same. Depending on the platform in question, users may or may not react well to this. It would be pertinent to investigate the user experience of personalized web applications compared to that of native applications by performing usability and technology acceptance tests.

Additionally, there is the matter of the hybrid development approach. While pure HTML5 applications lose out on certain device accesses and functionalities, it is possible to create hybrid applications that have the best of both worlds, although it does require a little more effort than plain web applications. PhoneGap, which has been described earlier, is one of the most popular ways of creating such hybrid applications, and it could be of interest to investigate its suitability for personalized applications.

The prototype application used in this project has been released as open source under the Apache 2.0 licence. It was created with modifiability in mind, so it could provide a useful starting point for further personalization research. It provides a complete solution for the front-end of a personalized application, and could potentially save a significant amount of time for anyone intending to focus on a different aspect of personalization research.

Appendix A

Code samples

A.1 Revealing Module Pattern

```
1  var myRevealingModule = function () {
2      var privateVar = "Ben Cherry",
3          publicVar  = "Hey there!";
4
5      function privateFunction() {
6          console.log("Name:" + privateVar);
7      }
8      function publicSetName(strName) {
9          privateVar = strName;
10     }
11     function publicGetName() {
12         privateFunction();
13     }
14     return {
15         setName: publicSetName,
16         greeting: publicVar,
17         getName: publicGetName
18     };
19 }();
20 myRevealingModule.setName("Paul Kinlan");
```

Listing A.1: Revealing Module Pattern sample

This sample code was taken from Addy Osmani's book, *Learning JavaScript Design Patterns* (Osmani, 2012). It is used in place of actual code from the prototype application in order to illustrate the workings of the Revealing Module pattern with a brief example that shows the full functionality of the pattern without any distractions.

The basis of the Revealing Module pattern is that the variables and functions of each module are contained within the private scope of the surrounding function. This provides the benefit of preventing naming conflicts, as well as ensuring that variables and functions intended solely for internal use are not accessed externally. In order to expose the functions and variables that should be accessed publicly, the module returns an object containing pointers to said functions and variables, as seen on line 14-17. The revealed functions and variables can then be accessed as seen on line 20.

A.2 HTML5 Web Storage

```
1  localStorage.setItem("categories", ko.toJSON(categories));
2
3  if (localStorage.getItem("categories") !== null) {
4    var savedCats = JSON.parse(localStorage.getItem("categories"));
5    for (var i in savedCats) {
6      categories.push({name: savedCats[i].name, weight: ko.observable(
       savedCats[i].weight)});
7    }
8  } else {
9    var defaultCats = ["Innenriks", "Utenriks", "Politikk", "Sport", "
       Bil", "Finans", "Mote", "Teknologi", "Dataspill"];
10   for (var i in defaultCats) {
11     categories.push({name: defaultCats[i], weight: ko.observable(0)})
       ;
12   }
13 }
```

Listing A.2: HTML5 Web Storage sample

The code above illustrates how to use HTML5 Web Storage. Line 1 contains the most common form of serialization used in the prototype. It converts the current category preferences from a Knockout ObservableArray to a JSON string and stores it in the *localStorage*. This is done whenever the categories are updated, to ensure that the stored data is always current in case of an unexpected closure of the application.

Lines 3 through 13 shows how the stored configuration is loaded. When the application is started, it will check *localStorage* for stored preferences and load them if they are found. Otherwise, the categories are populated with the default values. As shown in this example, HTML5 requires very little code in order to add client-side persistence via the use of the Web Storage API.

A.3 HTML5 Geolocation

```
1  function fetchLocation() {
2    if (navigator.geolocation) {
3      navigator.geolocation.getCurrentPosition(function(position) {
4        $.ajax({
5          'url': 'http://api.geonames.org/findNearbyPlaceNameJSON',
6          'data': {'username' : 'cgjme',
7                   'lat' : position.coords.latitude,
8                   'lng' : position.coords.longitude},
9          'success': function(data) {
10                  currentLocation = data.geonames[0].name;
11                  sessionStorage.setItem("currentLocation",
12                  data.geonames[0].name); },
13         'dataType': 'json',
14         'jsonp': 'json.wrf'
15       });
16     });
17   }
18   else {
19     supportsGeolocation = false;
20   }
21 }
```

Listing A.3: HTML5 Geolocation sample

This sample code shows how the HTML5 Geolocation API is used to find semantic information about the current location of the user in the prototype application. When running the function above, the Geolocation API uses device knowledge such as GPS or IP address to find the longitude and latitude of the user. This information is then sent to the GeoNames Web Service with a jQuery AJAX call, as seen on lines 4 through 16. If the AJAX call is successful, the name of the user's location is sent to the *viewModel*. Additionally, the name of the location is also stored HTML5 Web Storage, but this time *sessionStorage* instead of *localStorage*. This is done to reduce the amount of calls sent to the GeoNames Web Service, but

the storage is limited to a single session as there is no way to guarantee that the user will be in the same location when the next session is started.

References

Adobe Systems Inc (2008). *PhoneGap*. Available at: http://phonegap.com/. Accessed: 18.02.2013.

Adomavicius, G. and A. Tuzhilin (2005). "Personalization technologies: A process-oriented perspective". *Commun. ACM* 48.10, pp. 83–90. ISSN: 0001-0782. DOI: 10.1145/1089107.1089109.

Block, S. and A. Popescu (2012). *DeviceOrientation Event Specification*. Available at: http://dev.w3.org/geo/api/spec-source-orientation.html. Accessed: 05.05.2013.

Bond, W. (2011). *Sublime Package Control*. Available at: http://wbond.net/sublime_packages/package_control. Accessed: 20.05.2013.

Bos, B. (2013). *Web Style Sheets CSS tips & tricks*. Available at: http://www.w3.org/Style/Examples/007/units.en.html. Accessed: 05.05.2013.

Braude, E. J. (2001). *Software Engineering, An Object-Oriented Perspective*. John Wiley & Sons.

Catlin, H., N. Weizenbaum, and C. Eppstein (2006). *Sass*. Available at: http://sass-lang.com/. Accessed: 18.02.2013.

Eirinaki, M. and M. Vazirgiannis (2003). "Web mining for web personalization". *ACM Trans. Internet Technol.* 3.1, pp. 1–27. ISSN: 1533-5399. DOI: 10.1145/643477.643478.

Fan, H. and M. S. Poole (2006). "What is personalization? Perspectives on the design and implementation of personalization in information systems". *Journal of Organizational Computing and Electronic Commerce* 16.3-4, pp. 179–202.

Hazael-Massieux, D. and T. Roessler (2011). *Device APIs and Policy Working Group Charter*. Available at: `http://www.w3.org/2009/05/DeviceAPICharter.html`. Accessed: 05.05.2013.

Hevner, A. R. et al. (2004). "Design science in information systems research". *MIS Q.* 28.1, pp. 75–105. ISSN: 0276-7783.

Kesteren, A. van and S. Pieters (2012). *HTML5 differences from HTML4*. Available at: `http://www.w3.org/TR/html5-diff/`. Accessed: 18.02.2013.

Kneschke, J. (2003). *LIGHTTPD, fly light.* Available at: `http://www.lighttpd.net/`. Accessed: 20.05.2013.

Kostiainen, A., I. Oksanen, and D. Hazaël-Massieux (2013). *HTML Media Capture*. Available at: `http://www.w3.org/TR/html-media-capture`. Accessed: 05.05.2013.

Kovalyov, A. (2010). *JSHint, A JavaScript Code Quality Tool*. Available at: `http://www.jshint.com/`. Accessed: 18.02.2013.

Krogstie, J. (2012). "Bridging Research and Innovation by Applying Living Labs for Design Science Research". *Nordic Contributions in IS Research*. Ed. by C. Keller et al. Vol. 124. Lecture Notes in Business Information Processing. Springer Berlin Heidelberg, pp. 161–176. ISBN: 978-3-642-32269-3. DOI: `10.1007/978-3-642-32270-9_10`.

Meyer, E. A. and B. Bos (2011). *Introduction to CSS3*. Available at: `http://www.w3.org/TR/2001/WD-css3-roadmap-20010406/`. Accessed: 18.02.2013.

Osmani, A. (2012). *Learning JavaScript Design Patterns, A JavaScript and jQuery Developer's Guide*. O'Reilly Media.

Pugsley, T. and A. Hovey (2012). *The Responsinator*. Available at: `http://www.responsinator.com/`. Accessed: 22.04.2013.

Quint, A. (2009). *Sammy.js, RESTful Evented JavaScript*. Available at: `http://sammyjs.org/`. Accessed: 13.03.2013.

Sanderson, S. (2010). *Knockout. Simplify dynamic JavaScript UIs by applying the MVVM pattern*. Available at: `http://knockoutjs.com/`. Accessed: 13.03.2013.

Sellier, A. (2009). *LESS - The dynamic stylesheet language*. Available at: `http://lesscss.org/`. Accessed: 18.02.2013.

Skinner, J. (2008). *Sublime Text*. Available at: `http://www.sublimetext.com/`. Accessed: 20.05.2013.

Tate, T. (2011). *The Semantic Grid System*. Available at: `http://semantic.gs/`. Accessed: 24.05.2013.

The jQuery Foundation (2006). *jQuery - write less, do more*. Available at: `http://jquery.com/`. Accessed: 18.02.2013.

The Raspberry Pi Foundation (2008). *Raspberry Pi*. Available at: `http://www.raspberrypi.org/`. Accessed: 20.05.2013.

W3Techs (2013). *Usage of JavaScript libraries for websites*. Available at: `http://w3techs.com/technologies/overview/javascript_library/all`. Accessed: 18.02.2013.

Wick, M. and C. Boutreux (2005). *GeoNames*. Available at: `http://www.geonames.org/`. Accessed: 24.05.2013.

Zhang, X. et al. (2012). "Context-Aware Mobile Web Browsing Based on HTML5". *Ubiquitous Intelligence Computing and 9th International Conference on Autonomic Trusted Computing (UIC/ATC), 2012 9th International Conference on*, pp. 945–950. DOI: `10.1109/UIC-ATC.2012.107`.