Aalto University
School of Science
!Fixme **Set degree program** Fixme!

Kimmo Puputti

!Fixme Add English title Fixme!

!Fixme Add English subtitle Fixme!

Master's Thesis
Espoo, !Fixme **Add English date** Fixme!

DRAFT! — Wednesday 11th January, 2012 — DRAFT!

Supervisor: Professor Petri Vuorimaa, Aalto University
Instructor: Risto Sarvas D.Sc.(Tech.)

Aalto University
School of Science

Aalto University
School of Science
!Fixme **Set degree program** Fixme!

ABSTRACT OF
MASTER'S THESIS

Author:	Kimmo Puputti		
Title:	!Fixme **Add English title** Fixme! !Fixme **Add English subtitle** Fixme!		
Date:	!Fixme **Add English date** Fixme!	**Pages:**	23
Professorship:	Media Technology	**Code:**	T-110
Supervisor:	Professor Petri Vuorimaa		
Instructor:	Risto Sarvas D.Sc.(Tech.)		
!Fixme **Add English abstract** Fixme!			
Keywords:	!Fixme **Add English keywords** Fixme!		
Language:	English		

Aalto-yliopisto
Perustieteiden korkeakoulu
Tietotekniikan tutkinto-ohjelma

DIPLOMITYÖN TIIVISTELMÄ

Tekijä:	Kimmo Puputti		
Työn nimi: !Fixme **Add Finnish title** Fixme! !Fixme **Add Finnish subtitle** Fixme!			
Päiväys:	!Fixme **Add Finnish date** Fixme!	**Sivumäärä:**	23
Professuuri:	Mediatekniikka	**Koodi:**	T-110
Valvoja:	Professori Petri Vuorimaa		
Ohjaaja:	Tohtori Risto Sarvas		
!Fixme **Add Finnish abstract** Fixme!			
Asiasanat:	!Fixme **Add Finnish keywords** Fixme!		
Kieli:	Englanti		

Acknowledgements

!Fixme **Add acknowledgements** Fixme!
Thank you.

!Fixme **Decide city...** Fixme!, !Fixme **Add English date** Fixme!

Kimmo Puputti

Contents

0.1 Thesis Git repository info

Build time: **Wednesday 11th January, 2012 16:47**

Git HEAD:

```
commit 76479b7a136d75f6e35d240df5cc1a7f945c9fca
Author: Kimmo Puputti <kpuputti@gmail.com>
Date:   Wed Jan 11 16:38:34 2012 +0200

    Add HTTP RFCs.
```

Repository status:

```
# On branch master
# Your branch is ahead of 'origin/master' by 3 commits.
#
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
# modified:   methods.tex
# modified:   thesis.pdf
#
no changes added to commit (use "git add" and/or "git commit -a")
```

Mobile OS Type	**Skill Set Required**
Apple iOS	C, Objective C
Google Android	Java (Harmony flavored, Dalvik VM)
RIM BlackBerry	Java (J2ME flavored)
Symbian	C, C++, Python, HTML/CSS/JS
Windows Mobile	.NET
Windows 7 Phone	.NET
HP Palm webOS	HTML/CSS/JS
MeeGo	C, C++, HTML/CSS/JS
Samsung bada	C++

Table 1.1: Required skill sets for different mobile platforms. [2]

Chapter 1

Introduction: Smartphone Market and the Need for Cross-Platform Support

1.1 Smartphone Landscape

1.2 HTML5

1.2.1 History

1.2.2 Markup

1.2.3 CSS3

1.2.4 JavaScript APIs

1.2.5 Related APIs

1.3 Modern Mobile Web Application Architecture

1.3.1 Single-Page applications

1.3.1.1 JavaScript MVC Libraries

1.3.2 Responsive Design

1.3.3 Progressive Enhancement

1.3.4 UI Libraries

1.3.4.1 jQuery Mobile

1.3.4.2 jQTouch

1.3.4.3 Sencha Touch

1.3.5 Hybrid Applications

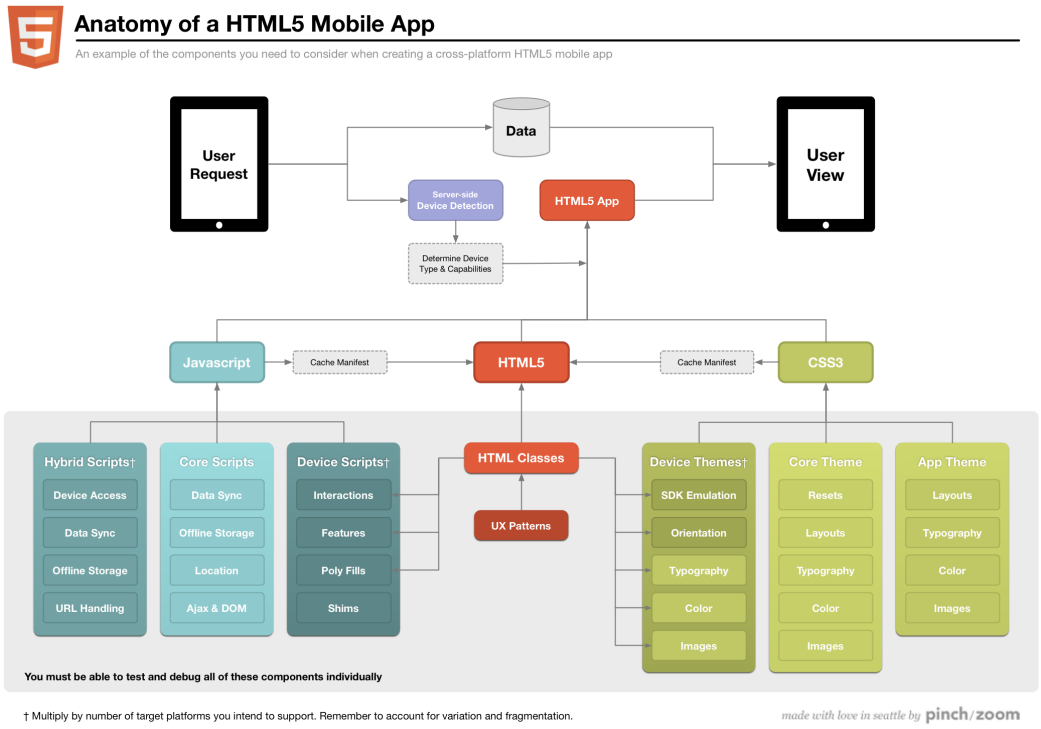

Figure 1.1: HTML5 Mobile Application Anatomy (citation needed)

1.4 Performance Guidelines

!FIXME **Add intro to sources and reasoning why frontend performance matters. [7, 8]** FIXME!

- **Make Fewer HTTP Requests**
- **Use a Content Delivery Network**
- **Add an Expires Header**
- **Gzip Components**
- **Put Stylesheets at the Top**
- **Put Scripts at the Bottom**
- **Avoid CSS Expressions**
- **Make Javascript and CSS External**
- **Reduce DNS Lookups**
- **Minify JavaScript**
- **Avoid Redirects**
- **Remove Duplicate Scripts**
- **Configure ETags**
- **Make Ajax Cacheable**
- **Splitting the Initial Payload**
- **Loading Scripts Without Blocking**
- **Coupling Asynchronous Scripts**
- **Positioning Inline Scripts**
- **Writing Efficient JavaScript**
- **Scaling with Comet**
- **Going Beyond Gzipping**
- **Optimizing Images**

- **Sharding Dominant Domains**
- **Flushing the Document Early**
- **Using Iframes Sparingly**
- **Simplifying CSS Selectors**

Chapter 2

Research Question: HTML5 - Hype versus Realities?

Chapter 3

Methods: Example Application and Library

3.1 Qt Developer Days 2011 Conference Schedule Application

The Qt Developer Days[1] is a conference for developers using the Qt cross-platform application and UI (?) framework[2]. We created a mobile web application with contextual and personalized session information and per-day schedule for the conference.

3.1.1 Application Architecture

The conference schedule[3] is a single-page application (citation needed)with a lightweight backend written in Python using the Django Web Framework[4].

The backend provides the static assets (JavaScript, CSS (?), images, etc.) and an API (?) for persisting session feedback to a MySQL[5] relational database. It also generates the HTML5 AppCache (citation needed)offline cache manifest file based on the categorized device type.

The frontend is a JavaScript application written using the Backbone[6] MVC (?) framework. Other used JavaScript libraries include Underscore[7]

[1] `http://qt.nokia.com/qtdevdays2011/`
[2] `http://qt.nokia.com/`
[3] `http://m.qtdevdays2011.qt.nokia.com/`
[4] `https://www.djangoproject.com/`
[5] `http://www.mysql.com/`
[6] `http://backbonejs.org/`
[7] `http://underscorejs.org/`

for data manipulation, jQuery[8] for DOM (?) API abstraction, Handlebars[9] for templating, and Modernizr[10] for feature detection. The HTML5 Mobile Boilerplate[11] was used as an initial markup structure for the application. The architecture of is depicted in Figure 3.1.

!FIXME **add app request flow description?** FIXME!

Wireless networks can be unreliable in conference settings, so offline support was also added using several different JavaScript techniques and HTML5 APIs.

The application was designed for touch screens on various platforms and screen sizes. The layout adjusts to the available space and provides rich interactive components. Integration to social networking services was also added as an additional functionality.

!FIXME **add screenshots on different devices (at least phone and tablet** FIXME!

3.2 JSONCache JavaScript Library

JSONCache is a lightweight JavaScript library for fetching JSON (?) data in flaky networks. The library was designed especially to handle flaky mobile networks with connection problems and short interruptions. The goal is to avoid networking as long as possible and failing gracefully if network connections are not stable.

JSONCache provides two main functionalities: data caching and attempting to fetch the data multiple times.

The caching layer uses the client side localStorage (citation needed)cache of HTML5 (?). Data requests can be done using the JSONCache API (?) which always checks the local cache first before opening any network connections. If the data is already in the cache, the cached data is checked for validity and if the data has not been expired, it is returned immediately. If the data is not in the cache or it has been expired, a new network request is made and the received data is cached and returned to the requestor. The expiration time of a data item can be configured in the library settings.

JSONCache also tries to fetch the data multiple times to handle small interruptions in network connection. !FIXME **add example and explain that it is very common** FIXME!. If a data fetch fails, a new fetch is issued after a timeout (defined in the configuration). On subsequent attempts the

[8] `http://jquery.com/`
[9] `http://handlebarsjs.com/`
[10] `http://www.modernizr.com/`
[11] `http://html5boilerplate.com/mobile`

timeout is increased, and after a defined number of attempts the fetch error is issued to the requestor.

Figure 3.2 shows an interactive demo of the JSONCache library. The demo[12] simulates the caching and fetching functionality of the library by simulating a flaky network according to the configuration.

[12] `http://kpuputti.github.com/JSONCache/demo/index.html`

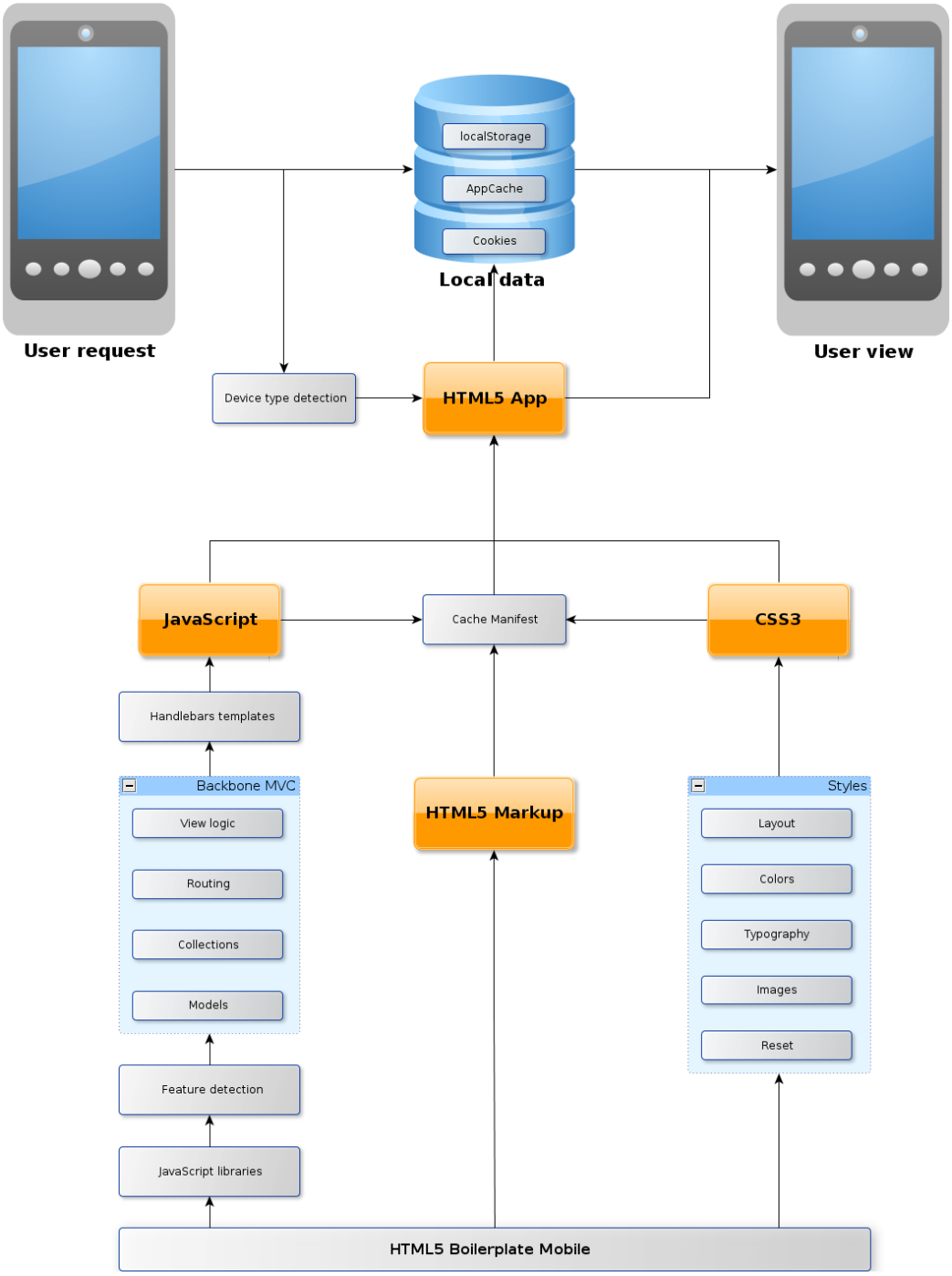

Figure 3.1: Conference schedule application architecture.

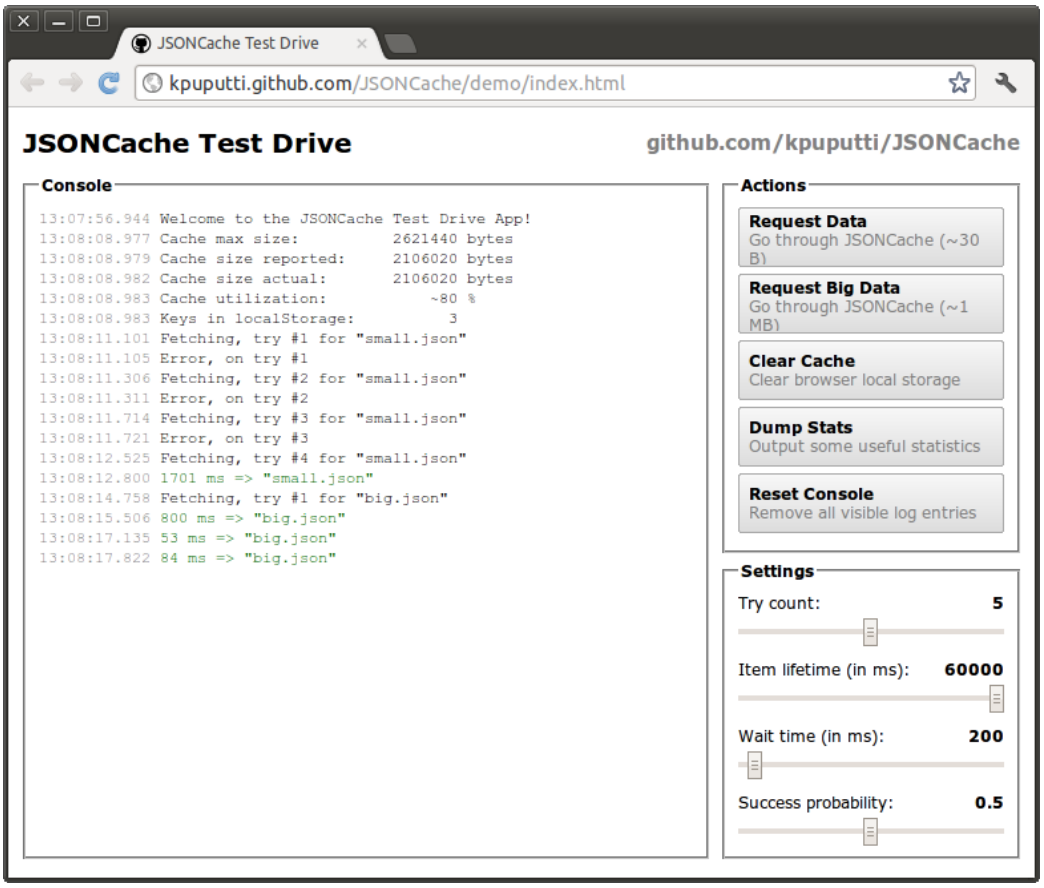

Figure 3.2: Interactive JSONCache demo.

Chapter 4

Results: What Was Good and Where Were the Compromises

4.1 Targeting Different Platforms

4.1.1 Device Detection

4.1.2 Feature Detection

4.2 Targeting Different Screens

4.3 Handling Mobile Networks

4.3.1 Minimizing Data Transfer

4.3.2 Caching

4.3.3 Preloading

4.3.4 Offline Support

4.3.5 Handling Interruptions

4.4 Graphics and Animations

4.5 Performance Analysis

4.5.1 YSlow

4.5.2 PageSpeed

Chapter 5

Discussion: Bright Future Ahead for HTML5

Chapter 6

LaTeXtest

6.1 Citing

- Berners-Lee [1]
- Mikkonen & Taivalsaari [5]
- Taivalsaari & Mikkonen [9]
- Pilgrim [6]
- Crockford [3]
- Souders [7]
- Garrett [4]
- Zakas [10]

Bibliography

[1] Berners-Lee, T. Long live the web. *Scientific American 303*, 6 (2010), 80–85.

[2] Charland, A., and Leroux, B. Mobile Application Development: Web vs. Native. *Communications of the ACM 54*, 5 (2011), 49–53.

[3] Crockford, D. *JavaScript: The Good Parts*. O'Reilly Media / Yahoo Press, 2008.

[4] Garrett, J. J. Ajax: A new approach to web applications. *Adaptive path 18* (2005). Available at: `http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications`. Accessed 5-January-2012.

[5] Mikkonen, T., and Taivalsaari, A. Apps vs. Open Web: The Battle of the Decade. In *2nd Annual Workshop on Software Engineering for Mobile Application Development* (2011).

[6] Pilgrim, M. *HTML5: Up And Running*. O'Reilly Media, 2010.

[7] Souders, S. *High Performance Web Sites*. O'Reilly Media, 2007.

[8] Souders, S. *Even Faster Web Sites*. O'Reilly Media, 2009.

[9] Taivalsaari, A., and Mikkonen, T. The Web as an Application Platform: The Saga Continues. In *Software Engineering and Advanced Applications (SEAA), 2011 37th EUROMICRO Conference on* (2011), IEEE, pp. 170–174.

[10] Zakas, N. C. *High Performance JavaScript*. O'Reilly Media / Yahoo Press, 2010.