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! — Monday 9th 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:**	18
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

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:**	18
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: **Monday 9th January, 2012 14:12**

Git HEAD:

```
commit 5c2c1716284916d490fedd9fa0db9c991506fc86
Author: Kimmo Puputti <kpuputti@gmail.com>
Date:   Mon Jan 9 13:59:45 2012 +0200

    Add JSONCache section.
```

Repository status:

```
# On branch master
# 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:   introduction.tex
#
no changes added to commit (use "git add" and/or "git commit -a")
```

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

- 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

3.2 JSONCache JavaScript Library

JSONCache is a lightweight JavaScript library for fetching JSON (!Fixme **abbreviation definition** Fixme!) 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.

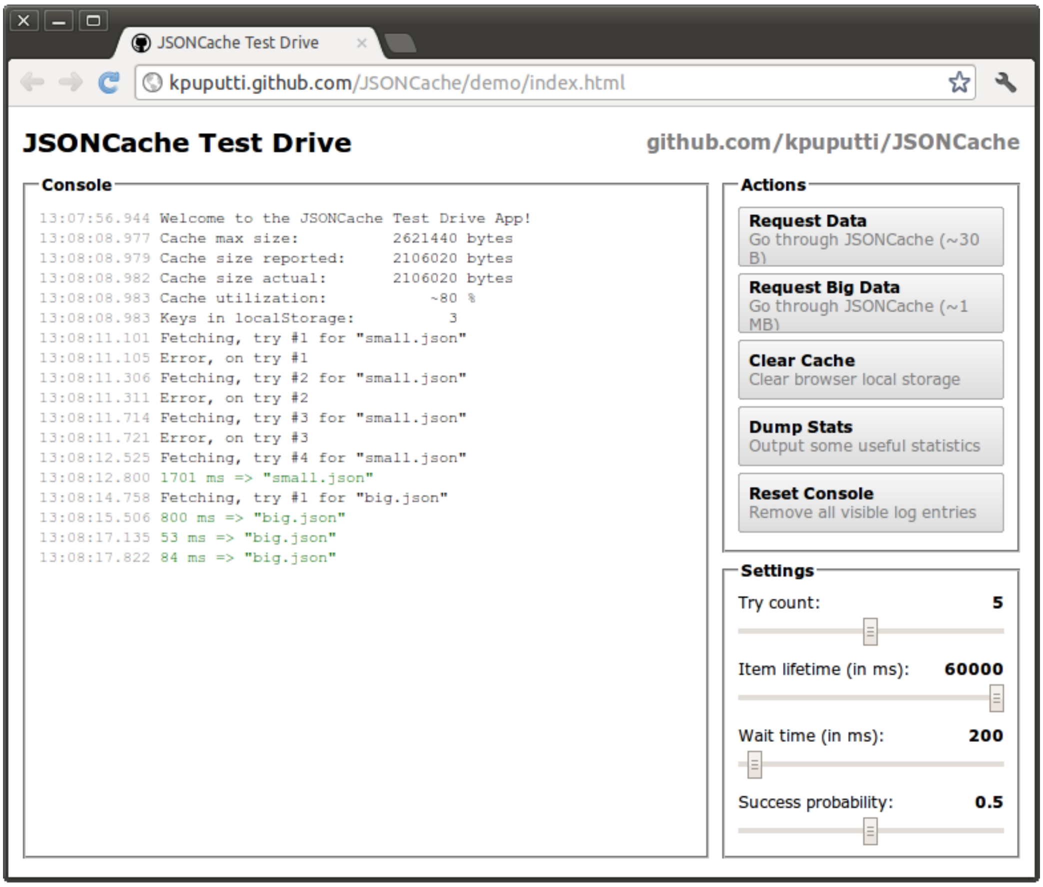
JSONCache provides two main functionalities: data caching and attempting to fetch the data multiple times.

The caching layer uses the client side localStorage (!Fixme **citation needed** Fixme!) cache of HTML5 (!Fixme **abbreviation definition** Fixme!). Data requests can be done using the JSONCache API (!Fixme **abbreviation definition** Fixme!) 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 timeout is increased, and after a defined number of attempts the fetch error is issued to the requestor.

Figure 3.1 shows an interactive demo of the JSONCache library. The demo[1] simulates the caching and fetching functionality of the library by simulating a flaky network according to the configuration.


JSONCache Test Drive
kpuputti.github.com/JSONCache/demo/index.html
JSONCache Test Drive
github.com/kpuputti/JSONCache
Console
Actions
Request Data
Go through JSONCache (~30 B)
Request Big Data
Go through JSONCache (~1 MB)
Clear Cache
Clear browser local storage
Dump Stats
Output some useful statistics
Reset Console
Remove all visible log entries
Settings

```
13:07:56.944 Welcome to the JSONCache Test Drive App!
13:08:08.977 Cache max size:          2621440 bytes
13:08:08.979 Cache size reported:     2106020 bytes
13:08:08.982 Cache size actual:       2106020 bytes
13:08:08.983 Cache utilization:           ~80 %
13:08:08.983 Keys in localStorage:          3
13:08:11.101 Fetching, try #1 for "small.json"
13:08:11.105 Error, on try #1
13:08:11.306 Fetching, try #2 for "small.json"
13:08:11.311 Error, on try #2
13:08:11.714 Fetching, try #3 for "small.json"
13:08:11.721 Error, on try #3
13:08:12.525 Fetching, try #4 for "small.json"
13:08:12.800 1701 ms => "small.json"
13:08:14.758 Fetching, try #1 for "big.json"
13:08:15.506 800 ms => "big.json"
13:08:17.135 53 ms => "big.json"
13:08:17.822 84 ms => "big.json"
```

Try count: 5

Item lifetime (in ms): 60000

Wait time (in ms): 200

Success probability: 0.5

Figure 3.1: Interactive JSONCache demo.

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

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 [4]
- Taivalsaari & Mikkonen [7]
- Pilgrim [5]
- Crockford [2]
- Souders [6]
- Garrett [3]
- Zakas [8]

Bibliography

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

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

[3] 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.

[4] 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).

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

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

[7] 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.

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