ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing Group

Beat Gebistorf

Enhancement of website usability for mobile phones

http://thebuzz.kazowie.com/tag/top/

Master Thesis
November 2010 until May 2011

Professor: Prof. Dr. Roger Wattenhofer
Advisor ETH: Samuel Welten
Advisors Namics AG: Johannes Waibel, Damian Amherd & Roman Zollet

Abstract

2009 more than half of the Internet users accessed the web with a mobile phone [6]. Nonetheless, in March 2011 79% of Google's largest advertisers did not provide a mobile optimised website [31] which would be necessary to be useful for mobile phones. Desktop versions of websites are not useful for mobile phones as they are developed for a different context. Mobile phones have smaller screens, worse input methods, and a lower bandwidth than desktop PCs. Additionally, mobile phones are used on the go.
Hence, mobile phones need websites which are optimised for them to get the best usability.
To face the huge potential of optimising websites for mobile phones this master thesis provides two approaches: one approach addresses the development of mobile websites which can be assisted with a mobile JavaScript framework whereas the other approach modifies existing websites according to the user's need with a proxy. Both approaches enhance the usability of websites for mobile phones at two different points.
The first approach concerns the development of websites. Therefore five mobile JavaScript frameworks were evaluated which support mobile optimised website development. jQuery Mobile as the best one was taken to carry out a reference implementation. The reference implementation showed that one can imitate the native *Helvetia Notfall-App* application with a mobile website without many constraints.
The second approach concerns the delivery of an existing website between the server and the client device. Hereby a proxy modifies websites on behalf of the mobile phone and delivers the optimised content to the client afterwards. A prototype of a mobile proxy called MOCUSI was implemented in this thesis. It allows mobile phone users to view an arbitrary website with customised modifications applied. Finally a usability test was conducted to verify the usability of the proxy solution. It showed that 50% of the tested persons would use a system like MOCUSI in the future. 25% would even use it daily.

Kurzbeschrieb

Im Jahre 2009 griffen mehr als die Hälfte der Internetnutzer via Mobiltelefon auf das Web zu [6]. Trotzdem verfügten im März 2011 79% der grössten Werbekunden von Google nicht über eine mobil optimierte Webseite [31], welche nötig wäre, um auf Mobiltelefonen nützlich zu sein. Desktop Versionen von Webseiten sind nicht nützlich für Mobiltelefone da sie für einen anderen Kontext entwickelt wurden. Mobiltelefone besitzen kleinere Bildschirme, schlechtere Eingabemethoden und weniger Bandbreite als Desktop PCs. Zudem werden Mobiltelefone unterwegs eingesetzt.
Daher benötigen Mobiltelefone optimierte Webseiten um die beste Benutzerfreundlichkeit zu erreichen.
Um das grosse Potenzial von optimierten Webseiten für Mobiltelefone zu nutzen, untersucht diese Masterarbeit zwei Vorgehensweisen: die eine betrifft die Entwicklung von Webseiten für Mobiltelefone, welche durch JavaScript Programmierumgebungen für Mobiltelefone untersützt werden kann, die andere verändert vorhandene Webseiten gemäss den Bedürfnissen des Benutzers mit einem Proxy. Beide Ansätze verbessern die Benutzerfreundlichkeit von Webseiten für Mobiltelefone an unterschiedlichen Stellen.
Der erste Ansatz betrifft die Entwicklung von Webseiten. Dazu wurden in dieser Arbeit fünf JavaScript Programmierumgebungen für Mobiltelefone evaluiert, welche die Entwicklung von mobil optimierten Webseiten unterstützen. jQuery Mobile, der Evaluationssieger, wurde gewählt um eine Referenzimplementation durchzuführen. Die Referenzimplementation zeigte, dass man mit wenigen Einschränkungen die native *Helvetia Notfall-App* Applikation mithilfe einer Webseite für Mobiltelefone imitieren kann.
Der zweite Ansatz betrifft die Auslieferung existierender Webseiten zwischen dem Server und dem Mobiltelefon. Dabei verändert ein Proxy Webseiten stellvertretend für ein Mobiltelefon und liefert den optimierten Inhalt an den Benutzer aus. MOCUSI, ein Proxy Prototyp, wurde in dieser Arbeit implementiert. MOCUSI erlaubt es Nutzern von Mobiltelefonen beliebige Webseiten mit benutzerdefinierten Modifikationen anzuzeigen. Zum Schluss wurde eine Nutzerstudie durchgeführt um die Akzeptanz von MOCUSI zu bestätigen. Die Studie zeigte dass 50% der Testpersonen ein System wie MOCUSI in Zukunft nutzen würden. 25% gar täglich.

Acknowledgements

I will thank first and foremost my wife, family and friends for their support. Furthermore, I thank my advisors for their helpful advices, constructive discussions, and prompt revisions.
I will also thank all tested persons who agreed to attend the usability test of MOCUSI:
Irina Gebistorf, Hendrik Grahl, Alex Maier, Manuela Lauchenauer, Martin Schilliger, Ernst Ammann, Fredi Wiedmer, Denis Skeledzic, Prof. Dr. Wattenhofer, Jasmin Smula, Claude Hohl, and Marlis Gebistorf.

Contents

List of Tables

List of Figures

1 Introduction

"... 625 million people are exclusively using only a mobile phone to access internet content." (Tomi Ahonen, 2011) [6]

"Only 21% of Google's largest advertisers have a website that is optimized for mobile." (Google Mobile Ads Blog, 2011) [31]

9% of the world population use exclusively mobile phones to access Internet content. Nonetheless, many websites do not provide a mobile version to enhance the usability of the website for mobile phones. Just one in five of Google's largest advertisers does so. This thesis will address this problem and will show solutions to enhance the usability of websites for mobile phones.

1.1 Internet on Mobile Phones

"The Internet is by far the most popular source of information..." [51]. The Internet is the 6th mass media after print, recordings, cinema, radio, and tv. Mobile is labelled as the 7th mass media. Amazingly, the mobile media grows by far faster than the first six mass media did [5].
"54% of all people alive on the planet have a mobile phone" [6]. Tomi Ahonen stated that there are more mobile phone subscriptions than people who use toothbrushes.
"Today in US and Western Europe, 90 percent of mobile subscribers have an Internet-ready phone" [44]. Mobile phones with Internet access combine the

last two mass media to achieve mobility in conjunction with the popularity of information fetching from the web.

1.2 Mobile Optimised Websites

Despite the technological improvements and the growing mobile market many websites provide still just a desktop optimised version.
Before 2007 this was the most meaningful way to go as mobile devices had significant limitations and were at most able to display extremely reduced versions of a website.
This changed in 2007 as the iPhone was released and allowed to browse websites in a feasible way by allowing zooming and scrolling of websites. Hence, websites became useable although not really useful.
Nonetheless, desktop versions of websites are not useful for mobile phones as they are developed for a different context. Mobile phones in contrast to desktop PCs are mobile, have a small screen, limited input methods and a lower bandwidth. Much of the provided content and layout of desktop websites constrain mobile phone users more than they help. Especially by stressing the network connection.

In the years after the first iPhone was released mobile phones with Internet access gained importance [44]. At the same time native applications became very popular. Native applications were developed specifically for a mobile phone and its context. They were not just useable but also useful.
Consequently, users preferred using native applications and not websites if they had to choose. But native applications have an important drawback. They are platform dependent.

1.3 Thesis Foundation

We think that mobile optimised websites bear a huge potential if they combine the cross platform compatibility of websites with the usability of native application's user interface and general mobile usability considerations. Hence this master thesis focuses on mobile websites and to enhance their usability.

Goals

The goals of the thesis are as follows:

- Providing guidance for mobile website developers to enhance the usability of websites for mobile phones.
- Providing a solution for users to enhance the usability of existing websites on mobile phones.

Approaches

There are two possible approaches to enhance the usability of websites for mobile phones. Either to generate mobile websites from the beginning or to modify existing websites.

Development The first approach handles the development of a website and starts at the source where the website is generated. The enhancement of usability can hereby be achieved by following guidelines at developing the website. Such guidelines are described in Section 3.1.2.
Furthermore, frameworks are available which help developing websites to enhanced usability for mobile phones (see Chapter 4).
The reference implementation at the end of Chapter 4 shows how far a specific native application can be imitated with the help of a framework.

Delivery The second approach handles the delivery and rendering of an existing website. Websites can be modified on their way from the server to the client by a proxy or by the client itself. These modifications of the website are applied to enhance the usability for the client mobile phone.
As heavy website modifications are computing intensive and proxies have enough computation power one should prefer proxy solutions. A prototype proxy solution called MOCUSI[1] was implemented within this thesis and is explained in Chapter 5.

Expected results

We expect that this thesis will provide the following results:

- A listing of guidelines to enhance the usability of websites for mobile phones.

[1] MOCUSI stands for MObile CUstomised webSItes.

- An evaluation of state of the art frameworks which is reusable for future projects.
- A reference implementation with the best evaluated framework showing that a specific native application can be imitated as a web application without many constraints.
- A proxy implementation which allows users of mobile phones to customise existing websites according to their needs.
- A usability test which verifies the usefulness of the proxy implementation for mobile phone users.

1.4 Outline

In the following this thesis provides details and facts concerning the world of mobile phones in **Chapter 2** .
Chapter 3 discusses related work. It introduces the usability guidelines used within the thesis, lists a categorisation of website adaptation approaches and describes the improvements this thesis provides compared to related work.
In **Chapter 4** the most popular Mobile JavaScript Frameworks (MJF) are evaluated. The framework with the best score is taken to realise a reference implementation. The reference implementation mimics the existing *Helvetia Notfall-App*[2] and identifies how well a website can mimic a specific native application.
Chapter 5 explains the mobile proxy implementation MOCUSI which enhances website usability for mobile phones by modifying existing websites. The chapter furthermore describes the conducted usability test of MOCUSI.
Finally **Chapter 6** concludes the thesis by summarising the achievements and insights and list possible future work at the end.

[2] *Helvetia* is a Swiss insurance agency. The app helps in emergency situations. More information on http://www.helvetia.ch/service/mobile/iphone-app.html.

2 Mobile World

"Fundamentally, 'mobile' refers to the user, and not the device or the application." (Barbara Ballard, Designing the Mobile User Experience, 2007)

The following chapter will role up the world of mobile phones and introduce the most important facts related to the thesis.

2.1 Mobile Phones

This thesis discusses the enhancement of mobile website usability. The topic can be applied to various mobile devices from Internet capable wristwatches to full featured laptops which would be out of scope of a master thesis. Therefore the scope is narrowed to mobile phones.
In this thesis mobile phones are defined as portable cell phones which fit in a trouser pocket.
Mobile phones are chosen because more than half of all people have one [6] and because of the limitations they nonetheless have compared to desktop PCs. Hence tablets and netbooks are less challenging because they are more similar to desktop PCs.
Although tablets and netbooks are not covered in this thesis the approaches to enhance usability of websites for them can also be derived from this thesis. The main differences are the fewer constraints, e.g. tablet screen sizes are generally not much smaller than of PC screens.

The capabilities and features of modern mobile phones enable their user to

handle appointments, tasks, messages, take photos, navigate through cities, play games, make phone calls, and browse the Internet. 90% of mobile subscribers in the US and Western Europe have an Internet ready phone [44].
The awareness of users for the help mobile phones provide for daily life was enforced by the appearance of the iPhone in the year 2007 [22].
The importance of mobile phones gets confirmed by some impressive numbers from Tomi Ahonen in [6] this year:

- About 6.9 billion people are alive.
- 5.2 billion active mobile phone accounts exist.
- *"54% of all people alive on the planet have a mobile phone."*
- In 2010 the mobile telecommunication industry generated $1.2 trillion.
- According to Ahonen, 80% of all 2.0 billion Internet users access Internet content exclusively with their mobile phone or in combination with a PC. 31% use exclusively a mobile phone to access the Internet.

2.2 Mobile Phone History

The mobile phone history started in the year 1973 with the first handheld cellular phone. In the following, the eras and important dates are listed according to [22].

- **1973-1988:** In the **brick era** the first cordless and portable telephones appeared. They had to be carried in suitcases.
- **1979:** The first generation (**1G**) network was launched in Japan by NTT as the first commercially automated cellular network. It based on analog transmissions.
- **1988-1998:** In the **candy bar era** long, thin, rectangular mobile phones appeared which were small enough to fit in a pocket. Candy bar phones were capable of using the Short Message Service (SMS). These phones are often associated with 2G and Global System for Mobile Communications (GSM) networks.
- **1991:** The **2G** network introduced a new way to communicate with SMS text messages. It is based on digital transmissions.
- **1996:** Nokia 9000 Communicator was the first mobile phone with the capability to access the Internet.

- **1998:** The first Wireless Application Protocol (**WAP**) 1.0 standard was released. WAP browsers are browsers for small mobile devices. WAP 1.x bases on the Wireless Markup Language (WML) (see Figure 2.2). Standard Internet websites need to be "translated" by a proxy in WML to be viewable on WAP browsers.

- **1998-2008:** In the **feature phone era** mobile devices got new features like listening to music, taking photos, and using the Internet. In this era GSM was extended with General Packet Radio Service (GPRS), which is most often referred to as 2.5G. **With the feature phones the Internet reached mobile devices.** But no one used it due to high prices, poor marketing, and inconsistent rendering.

- **2001:** The **3G** network can mainly be distinguished from 2G by its use of packet switching rather than circuit switching for data transmission. The higher connection speed of 3G allows media streaming of radio or even television content.

- **2002:** Release of **WAP 2.0** which uses XHTML Mobile Profile, a subset of the Extensible Hypertext Markup Language (XHTML).

- **2002-today:** The **smartphone era** overlaps the previous and following era. Smartphones are similar to feature phones but are equipped with a common operating system, a larger screen size, a QWERTY keyboard or stylus for input, and Wi-Fi or another form of high-speed wireless connectivity.

- **2007-today:** The **touch era** started with the introduction of the iPhone. The phones of the touch era changed the everyday perception of the mobile phone usability and capabilities entirely. Mobile phones got an own identity and separated themselves from phones and computers. Applications and services appeared which would neither for landline phones nor for desktop computers make sense, e.g. location based phone call to the next restaurant. *"In less than a year, more than 2,000 mobile web applications were made freely available specifically for the iPhone"* [22].

- **future:** The **4G** network will be introduced to provide a higher data rate. No commercial implementation has yet achieved the target speed of 100 Mbps to qualify for the 4G label.

Touch phones Mobile phones featured with touchscreens are in the following called touch phones. Gartner predicts that in the year 2013 58% of the sold mobile devices will be equipped with a touchscreen [17]. The growing request for touch phones can be seen in the continuous growing market share

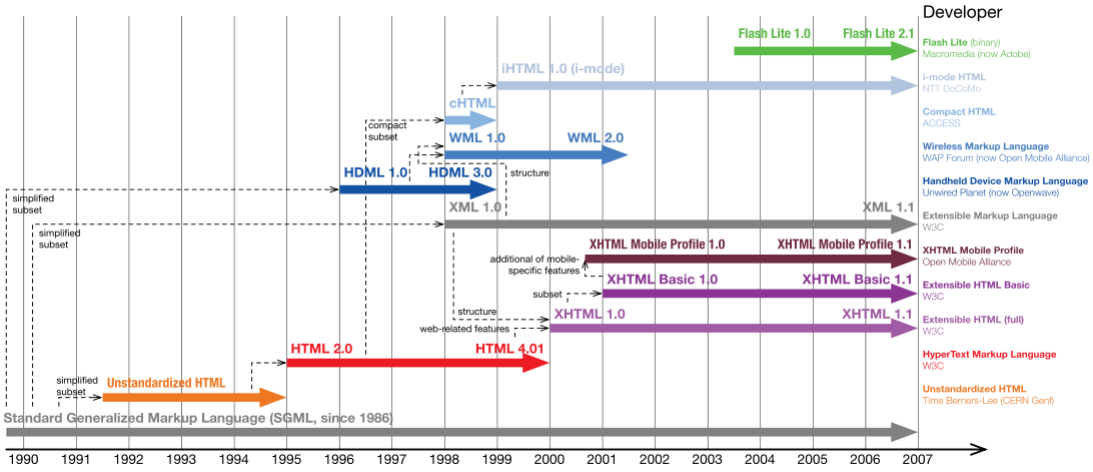

Figure 2.1: Evolution of mobile web-related markup languages (from David Höffer, GNU licence). Since 2007 HTML5 gained importance although it still has draft status.

of iPhones and Android phones which are all equipped with touchscreens. [44]

2.3 Mobile vs. PC

The benefits of mobile phones compared to desktop PCs are their

- size and weight which make them portable,
- context aware usage on the go, e.g. location detection on a mountain hike,
- short or no boot time which facilitates quick searches, and
- feature connections, e.g. searching the nearest Hotel and start a phone call to it.

Mobile phones also have some important drawbacks compared to desktop PCs. They have

- a smaller screen size,
- only a few websites optimised for mobile phones,
- reduced input methods (small or no QWERTY keyboard etc.),

- less bandwidth,
- smaller storage, and
- less processing power.

Summarising the pros and cons mobile phones benefit from mobility and quick context aware usage with the drawback of worse information processing and visualisation capabilities. Information is most useful for mobile users if the previously mentioned pros and cons are considered. If information does not utilise the benefits and treat the drawback of the mobile context many users will wait until they have access to a desktop device.

2.4 Mobile Internet

2.4.1 The Internet

A main service of the Internet is to provide websites. The words *web* and *Internet* are used exchangeable in the following.
Websites base on data traffic handled by the Hypertext Transfer Protocol (HTTP). The data can contain different Internet Media Types (MIME-Type) whereas the most important are text, image, video, audio, and applications. The implementation of a website consists mainly of a Hypertext Markup Language (HTML) which structures the content. To additionally provide style or behaviour definitions Cascading Style Sheet (CSS) and JavaScript (JS) statements are used respectively.
HTML5 and CSS3 are drafts of the World Wide Web Consortium (W3C). The drafts propose extensions and updates to the previous versions. They alleviate the development of website for different platforms and browsers.

2.4.2 Mobile Internet Terms

Websites optimised for modern mobile phones base on the same languages (HTML, CSS, JS) as "normal" websites. Hence mobile optimised websites differ not in the technology but in the way they behave and they are styled. This does not hold for older mobile phones which based on WAP 1.x and WML. Websites optimised for modern mobile phones are in the following called **mobile websites**.
The meaning of websites and web pages differs within this thesis. **A website refers to a web appearance of a domain which may contain various web pages**. A web page is constricted by visually connected content. After a link navigation a page load leads to a new web page which may be on the

same website.
Optimising a website means to modify or to adapt it to a device or a context. Desktop optimised websites are websites which were developed with desktop PCs in mind.
Website usability refers to the user friendliness of a website under the given circumstances. E.g. mobile website usability has to consider small screens and limited input methods. Website usability is described in more detail in Section 3.1.
A web application (web app) is a form of a website. A web app has an application nature and has to interact with the user. A website in contrast can only show content without interaction elements. The terms website and web app are used exchangeable in the following.

2.4.3 Native Application vs. Mobile Website

Beside browsing websites modern mobile phones allow to install native applications (native apps).

Native apps are compiled programs which have to be downloaded and installed on a device to use them. They are developed for a specific platform like iOS or Android and use the platform specific API.

Benefits:

- Able to access device sensors and interfaces like a camera, a gyrosensor[1] or a light sensor.
- Can run offline if no new data has to be fetched.
- Their look and feel is generally snappier than on websites.
- New content can be loaded via the Internet.

Drawbacks:

- Not cross platform compatible.
- New applications are controlled by app market authorities.
- Application updates are controlled by app market authorities.
- Some app markets even dictate an approval process for new applications or updates of applications.

[1] Orientation sensor to measure direction changes.

Mobile websites are platform independent and can be used by all mobile phones containing a browser.

Benefits:

- Cross platform compatible.
- Able to access a subset of device's sensors and interfaces. They are to detect orientation changes between landscape and portrait, location detection if the phone is featured with GPS[2], and using the local storage.
- With the HTML5 application cache it is possible to run a website offline.

Drawbacks:

- Limited access to sensors and interfaces of devices.
- Slower animations and transitions.
- Slower execution.

The focus of the thesis will be on mobile websites based on the following reasons:

- Mobile websites allow to browse the whole web without installation processes in between.
- Mobile websites are cross platform compatible and only needs to be implemented once for all devices. Native apps in contrast are mainly based on specific programming languages like Java (Android) or Objective-C (iOS).
- Mobile websites can easily be updated without running through an acceptance process in contrast to native apps in app store markets.
- Mobile website are useful to fetch a one time information and do not require to download and install a native app. This can be illustrated with the airport flight information. Users will not install an airport flight information app if they fly twice a year. They will rather visit the website.
- Mobile websites cover all devices containing a web browser.

[2]Global Positioning System

2.4.4 Mobile Optimised Websites

Information is spread over the Internet in various forms. Websites, blogs, RSS feeds and many more. Gartners reviews let assume that in the future mobile phones will be a main audience of this information. Despite this trend one can observe that only a small amount of websites is optimised for mobile phones.
Unfortunately even some of the seemingly mobile optimised websites are hard to read on small screens or hard to control with the limited input methods of mobile devices. In the following pictures one can see the differences between a page optimised for mobile phones in Figure 2.2, one with a medium optimisation in Figure 2.3 and one that is not optimised in Figure 2.4.

Figure 2.2: Optimised website

Figure 2.3: Medium optimised website

Figure 2.4: Not optimised website

Most websites optimised for desktop screens are not well usable on mobile phones because they were designed for large size screens, full featured browsers e.g. with flash support, and advanced input methods.
Too largely sized websites generate much data traffic and will visually either be scaled down by mobile browsers or one has to scroll horizontally and vertically or part of the content are cut away. Unsupported content resources like flash or other formats will just be ignored by most mobile browsers or cause an error. To many form fields will upset the user because of awkward input methods.
Although many users are used to the situation of websites not being optimised for mobile phones, it constrains the usability on mobile phones.

2.4.5 Mobile Website Development

There exist several approaches to develop a mobile optimised website.

Manually

The W3C drafts in progress CSS3 and HTML5 alleviate the development of mobile websites as they help to overcome cross platform compatibility problems. Previous compatibility problems of animated or interactive elements like flash can now be implemented with CSS3 animations or HTML5 canvas.

Content Management Systems (CMS)

Mobile CMS extensions exist for many CMS systems (e.g. Typo3, Day CQ 5.4, Drupal, Wordpress). Some of these extensions base on device detection to know which kind of device accesses the page and which capabilities this device provides. The most popular two are the open source WURFL database [48] and the proprietary DeviceAtlas [45]. Mobile CMS extensions provide an abstraction layer for groups of devices. One can define a template for a device group. Templates will be applied according to the accessing device.

Mobile JavaScript Framework (MJF)

Mobile JavaScript Frameworks (MJF) are code libraries which alleviate the development of mobile websites. These libraries mainly consist of JavaScript code which provide an abstraction layer for browser specific implementations. Despite specifications of W3C and other associations browser implement specifications differently or even ignore specifications. Hence MJFs provide a unique programming interface for the different browser implementations. Additionally, MJFs facilitate framework specific components, layouts, events and animations.
MJF follow different approaches to deliver a unique interface. Some require to program the website's content and behaviour completely in JavaScript (e.g. Sencha Touch) whereas others rely on predefined tag classes or ids in the HTML code (e.g. jQuery Mobile).
Additionally, many MJF provide a CSS style sheet which defines a default layout and theme suitable for mobile device screens.

CMS extensions imply that the website bases on a CMS. MJFs in contrast are deployable more flexibly. MJFs focus on the user interface (front end) and are compatible with various back-end technologies including CMS. MJFs strongly affect the usability of a website because the usability heavily depends on the displayed content.
Technically, most MJFs rely mainly on their JavaScript implementation.

For about two years MJFs have been springing up like mushrooms. Lists of

different MJFs are provided by [56], [40], and [41]. Many of the MJFs focus mainly on imitating the look and feel of native iPhone apps in a rigid way (WebApp.net, iUI, UiUIKit, iWebKit, Magic Framework, CiUI, DynamicX). Other MJFs are more flexible and apply their own style (jQuery Mobile, Sencha Touch, Wink ToolKit, jQTouch, XUI, zepto.js).
Nonetheless, all of them look and behave more or less the same. That has most likely to do with the following two facts:

- Apple revolutionised the market with the iPhone and it's easy to use interfaces following a strong interface guideline. MJF tend to follow these layouts and guidelines.
- Users prefer interfaces they are familiar with [14].

Hence Apple set the unofficial standard for the layout and behaviour of mobile web apps and native apps (a.k.a. dominant design).

Hybrid solutions are on the market besides the MJFs which allow to develop native apps in a website development manner. The hybrid applications run as a wrapper of the content applications on a mobile device.
The advantage of hybrid solutions is that the content applications can be programmed in HTML/CSS/JS like websites and can access interfaces of the device like native apps. Furthermore this solution is cross platform compatible in case the wrapper application exists for the platforms. A drawback is that the wrapper application has to be installed. QuickConnect, PhoneGap, LiquidGear, and safire belong to the hybrid solutions.

For the sake of completeness Appcelerator Titanium and similar frameworks have to be mentioned which can be programmed with JavaScript but compile the code to native apps.

As this thesis focuses on pure MJFs which are evaluated in Chapter 4 the hybrid and native app solutions will henceforward be skipped.
To reduce the number of MJFs to evaluate in Chapter 4 we tried to select MJFs which follow different approaches and represents the respective approach best.
jQuery Mobile and Sencha Touch are the most popular MJFs beside jQTouch. jQTouch will not be evaluated because it is a predecessor of the other two. Furthermore jQTouch bases on jQuery and is written by the same person who later on helped to create Sencha Touch.
We chose jQuery Mobile, Sencha Touch, webapp.net, joHTML5, and wink toolkit to be evaluated based on the facts listed in the following:

- **jQuery Mobile:** jQuery Mobile is built on top of the very popular jQuery. Furthermore jQuery Mobile's primary goal is cross platform

compatibility which forms the main advantage of mobile websites in contrast to native apps.

- **Sencha Touch:** Sencha Touch bases on the very popular ExtJS. It relies much more on JavaScript programming than jQuery Mobile. It provides many features and components.
- **wink:** Wink provides a wide range of User Interface (UI) components and animated transitions. It centres on the look and feel although it provides no default layout.
- **webapp.net:** webapp.net is a lightweight MJF. It bases heavily on HTML tag classes and identifiers.
- **JoHTML5:** JoHTML5 is a lightweight MJF. In contrast to webapp.net it relies heavily on JavaScript programming. It is a relatively unpopular framework and helps to give a relation in the evaluation.

2.4.6 Web Application Features

Application cache

The draft of HTML5 specifies the application cache. It enables developers to instruct supported browsers to save websites for offline usage. This is done by providing a reference to a configuration file in the manifest attribute of the html tag.

```
<!DOCTYPE HTML>
  <html manifest="config.php">
  ...
```

The configuration file has to be delivered by the server with the mime-type `text/cache-manifest`.
The configuration file defines which files will be downloaded to the cache, which files have to be loaded from the server anyway if the browser is online and how the fallback variants look like if the browser is offline.
If the browser is online it checks on page load whether the configuration file changed. If it did all the defined contents will be downloaded according to the configuration file including the new configuration file. This means that changes to the configuration file are only applied after a second reload.
By implementing the configuration file dynamically (e.g. with php) one is able include the sum of the hashes of all downloaded files. Therewith one can ensure that the configuration file changes if any of the downloaded files changed. Consequently all files are downloaded if one file changed.
The "*" is the only allowed regular expression within the configuration file

and means wildcard entries.
An example of `config.php` is shown in the following:

```
<?php
  // Define the mime-type
  header('Content-Type: text/cache-manifest');
  // Start of the configuration file
  echo "CACHE MANIFEST\n";
  // Directive to start the list of all files
  // that have to be downloaded
  echo "CACHE:\n";
  $hashes = "";

  // Iterate through all files except the defined ones and
  // add them to the list
  $dir = new RecursiveDirectoryIterator(".");
  foreach(new RecursiveIteratorIterator($dir) as $file) {
    if ($file->IsFile() &&
      !preg_match('/.php/',$file) &&
      !preg_match('/.txt/',$file) &&
      !preg_match('/.svn/',$file) &&
      !preg_match('/maps/',dirname($file)) &&
      !preg_match('/mobile/',dirname($file)) &&
      !preg_match('/checkout/',dirname($file)) &&
      !preg_match('/content/',dirname($file)) &&
      !preg_match('/pages/',dirname($file)) &&
      substr($file->getFilename(), 0, 1) != ".")
    {
      echo $file . "\n";
      // compute the hash of the file and sum it up
      $hashes .= md5_file($file);
    }
  }
  // Define files which have to be requested from the server
  echo "NETWORK: \n";
  echo "*\n";
  // Define fallback variants
  echo "FALLBACK:\n";
  // Inject the sum of all hashes
  echo "# Hash: " . md5($hashes) . "\n";
?>
```

Although HTML5 application cache enables offline web app development one has to be careful because of the limited storage. If the cache is full

no additional data can be stored. More details can be found in the book *Building Android Apps with HTML, CSS, and JavaScript* [55].

iOS web app

Apple provides a set of meta tags to instruct iOS devices to handle websites specially. The meta tags allow to save and run websites visually like native apps. This means that the browser context menus (navigation bar and URL field) disappear.

3
Related Work

"The phrase 'mobile usability' is pretty much an oxymoron. It's neither easy nor pleasant to use the Web on mobile devices." (Jakob Nielsen, 2009) [46]

Since the introduction of WAP in 1998 many approaches and enhancements of implementations were proposed to provide mobile usability of websites. Relevant ones for this thesis are mentioned in this chapter. The first section handles usability basics and lists related work to mobile website usability. Section two and three discuss two approaches to enhance mobile website usability—section two by highlighting related work on JavaScript frameworks; section three by describing related work on the topic of mobile proxies.

3.1 Mobile Website Usability

Browsing the web on mobile phones forms an own world in case of usability. Websites which are usable on desktop PCs are not by default well usable on mobile phones. This section helps to define the term usability for websites on mobile phones. The following subsections will list the key problems of mobile phone usability, summarise guidelines to create useful websites for mobile phones and finally categorise possible website adaptations for mobile phones to enhance the overall usability.

3.1.1 Key Problems

In spite of the time gap of eight years and the technology gap from WAP to HTML/CSS/JS websites the studies of Buchanan et al. [14](2001) and Nielsen [46](2009) list similar usability hurdles for using websites on mobile phones. Combining the results of the two studies the following usability problems can be identified:

- The small screen size of mobile phones.
- Awkward input methods of mobile phones.
- Long download delays.
- Websites which are not optimised for mobile phones on a conceptual level (e.g. complicated navigation and inappropriate website structures).

Although screens are larger than several years ago and input methods are better on newer phones the screen size and input problems remain an issue [46] and will also do so in the near future. The problems of long download delays and websites not optimised for mobile phones are easier solvable and are hopefully mitigated soon, amongst others with the contribution of this thesis.

The main learning out of the two studies by Buchanan et al. and Nielsen is that **mobile phones are not the same as desktop PCs an never will be** [46].

3.1.2 Guidelines

Many guidelines can be found for mobile website development. The following list summarises the most popular ones of [8][14][32][43][46][54][65] beginning with the top then of the *W3C Mobile Web Best Practises* guidelines [61]:

- **Design for One Web:** Content found under a URI should be the same independent of the accessing device. Restructuring the content is legit.
- **Rely on Web standards:** Standards help to guarantee interoperability regarding the high device fragmentation.
- **Stay away from known hazards:** Pop ups, nested tables, table layouts, graphics for spaces, frames, and image maps should not be used.

- **Be cautious of device limitations:** Consider that there are great capability differences between devices when you choose a web technology.
- **Optimise navigation:** Keep the navigation simple, short and consistent, and reduce typing effort. Hence avoid heavy interactions. Additionally, clarify where a link leads to.
- **Check graphics & colours:** Consider low-contrast screens, and various colour and format support of devices.
- **Keep it small:** Small websites in case of data volume costs less time and money for the user.
- **Use the network sparingly:** Benefit from web protocol features (e.g. caching) to reduce network latencies and to cope with network bottlenecks.
- **Help & guide user input:** Minimise the need of the keypad or other input methods.
- **Think of users on the go:** Users on the go need compact information if they have little time and are distracted.
- **Structure content wisely:** On small screens every pixel counts. Mobile websites should therefore be very focused. A small screen size has less negative influence if the shown text is appropriate for the screen size.
- **Information first:** Avoid navigation links at the top as far as possible. Furthermore put the most used content at the top and less frequented content below.
- **Desktop version:** Provide a link to the desktop version. *"As for link labels, we recommend 'Mobile Site' and 'Full Site', respectively"* [46].
- **Consider screen sizes:** Screen sizes vary from 128x128 to 960x640 pixels.
- **Avoid zooming:** Zooming is not ideal because it adds an extra step and is not easy to do on all devices.
- **Avoid horizontal scrolling:** Vertical scrolling is enough, additional horizontal scrolling complicates orientation.
- **Decide on the number of mobile sites:** To best cover differences of device capabilities and screen sizes you may provide several mobile sites (see 0.facebook.com / m.facebook.com / touch.facebook.com).

- **Consider touchscreens:** If touchscreens belong to your target audience provide large enough selection elements (e.g. links). Fingers tend to be too thick to hit small elements.
- **Take advantage of inbuilt functionality:** All phones can make phone calls. Hence phone numbers can be linked to directly trigger a phone call. Many phones support further features, e.g. location-detection which helps to return context aware information.
- **Auto forward:** Auto-forward mobile devices to the mobile version of your website. Otherwise users will hardly find your mobile optimised website.
- **Testing:** Emulators provide a good help at developing mobile websites but differ from the behaviour of real devices.

Additional to the guidelines you have to consider the following three points:

- **End user development (EUD):** The EUD idea [28] proclaims that applications and tools should be customisable for users without programming knowledge to meet their needs. According to EUD web applications should provide settings to set preferences and allow to extend the application. Hence this can only be applied to websites with an application nature or for website adaption solutions like proxies (see Section 3.3).
- **Context:** Mobile websites have to be designed with clear concepts in mind. The target audience and the circumstances in which they will use the website are extremely important [63].
- **Brand recognition:** Mobile optimised pages have to allow users to be able to associate it with the original page [49][52].

The additional guidelines are all met by the proxy implementation MOCUSI because it enables users to customise websites themselves according to their needs. Consequently users can "develop" context awareness and brand recognition in their own way. MOCUSI is described in detail in Chapter 5.

3.1.3 Website Adaptation

Soon after smartphones were able to access the Internet various solutions were proposed to enhance website usability by adapting websites for mobile devices.
To be able to compare approaches with each other, it is helpful to provide

categories for all possible website adaptation approaches.
Bickmore et al. [13] categorises in his paper adaptation approaches to display websites on small screens in the following five categories (see Figure 3.1).

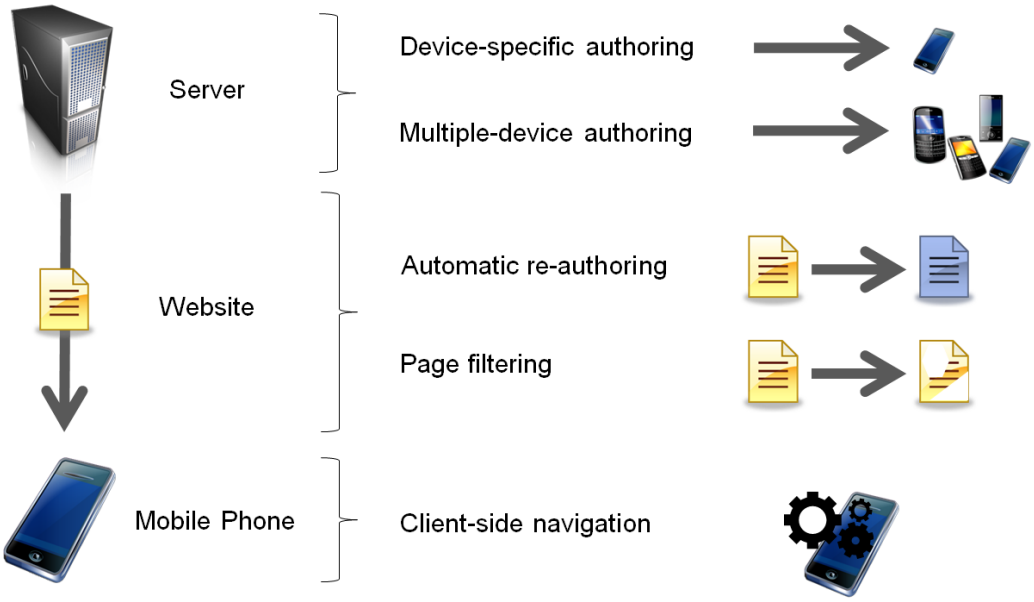

Figure 3.1: Website adaptation categories according to Bickmore et al.

Device-specific authoring Within this approach websites are adapted or newly created with one specific device in mind. Hence only a small user group can take advantage of the optimisation. E.g. programming web apps for Apple's iPhone and iPod touch follow this approach with special tags ignored by other devices.

Multiple-device authoring Multiple-device authoring means to optimise a website for multiple devices. This can be achieved with several style sheets and providing different devices with the appropriate one. Cutting edge implementations of this approach are fluid layouts which provide a *"CSS grid system for designing adaptive websites"* (http://lessframework.com/).

The previous two approaches have to be initiated and applied by website authors whereas the following three can be applied on websites by the user. Hence the following three approaches provide solutions a user can apply but is not forced to.

Client-side navigation The client-side navigation approach enables the user to interactively navigate within a website. Cutting edge mobile browsers enable client side navigation by default with the tools to zoom web pages and to scroll. P. Baudisch et al. proposed in [10] an Internet Explorer (IE) extension giving users the ability to collapse elements and zoom elements to extract elements containing relevant information. This solution provides a good usability enhancement but runs only on IE. An important issue is that the gestures used by Baudisch et al. would conflict with the native gestures used by modern mobile browsers. Furthermore the collapsible elements are predefined by a page analysis algorithm and can not be user defined.

Automatic re-authoring Automatic re-authoring describes the automatic adaption of an arbitrary web page depending on the target device capabilities. This approach relies on transformations of the content and associated resources. HTTP proxy solutions belong to this approach.

Page filtering This approach filters elements out of a web page to show only parts which are interesting for the user. Many times this approach is combined with automatic re-authoring. A representative proposal of this approach comes from Xinyi Yin and Wee Sun Lee [66].

We agree with the statement of Bickmore et al.: *"If automatic re-authoring can be made to produce legible, navigable and aesthetically pleasing re-authored documents without loss of information, it is superior to the other approaches."*

Nonetheless, the approaches of automatic re-authoring or page filtering have to be used carefully. One should not completely destroy the original structure. According to the additional guideline *Brand recognition* mobile optimised web pages have to allow users to be able to associate it with the original page.

This thesis provides solutions in the category *Multiple-device authoring* by addressing mobile JavaScript frameworks (see Chapter 4) and the category *Automatic re-authoring* with the proxy solution MOCUSI (see Chapter 5).

3.2 Mobile JavaScript Framework (MJF)

The approach of recent MJFs belong to the multiple-device authoring approach (see Section 3.1). Little related work can be found regarding MJFs or regarding evaluations of MJFs. The only findable evaluation by [21] is relatively unscientific. It compares - beside the mobile application framework Titanium Mobile - the MJFs Sencha Touch, jQuery Mobile, and jQTouch.

The comparison is very subjective and founded on implementation experiences for a specific project rather than predefined criteria. The provided conclusion of the comparison is to choose Sencha Touch because it provides a good performance and breadth although the development style and the documentation were tedious. The evaluation done in this thesis funds on predefined criteria and gathered property lists for five MJFs which provides a more objective and broader evaluation.
More related work exists for non-mobile JavaScript frameworks which are similar to MJFs or even are connected to them (e.g. jQuery Mobile is built upon jQuery).
A. Langer [37] evaluated in his thesis JavaScript frameworks which provide Asynchronous JavaScript and XML (AJAX) handling. G. Janisch [34] evaluated in his thesis Rich Internet Application (RIA) frameworks. Both theses provide useful criteria for the MJF evaluation found in this thesis. Further useful criteria to evaluate JavaScript frameworks are listed in [39][47][58]. Given the spare related work on MJF evaluations Chapter 4 will provide helpful information to choose an appropriate MJF for future projects.

3.3 Mobile Proxy

Mobile proxies belong to the automatic re-authoring approach which transforms websites. Transformations can be split as described in the following.

Basic transformation Basic transformation means adaption of resources like images which can for example be scaled down or discoloured to consider the small screens and the low bandwidth of mobile phones. Another approach is to use lossy image compressions [50].
The proxy solution of this thesis does apply image scaling.

General transformation General transformation can be achieved by summarising text and forms [15] or reducing large tables [57] to save valuable place on the screen. Similar approaches reduce pages to a navigation list [15][59]. Because these transformations restrict the content they are not employed in this thesis.

The main drawbacks of the basic and general transformation approaches are the lack of flexibility. User preferences are not respected. In other words the approaches do not allow users to configure the degree of transformation or to select adaption methods.

Thumbnail transformation Another transformation method applied by [10] and [42] shows a preview image (thumbnail) of the visited website. The user afterwards can click on the interesting part of the website thumbnail. The clicked part will be displayed in more detail.

The drawback of this method is its static nature. Dynamic content or interaction elements cannot be captured with a thumbnail. Therefore it is not satisfying enough to be used in this thesis.

Customised transformations MOWSER of Harini Bharadvaj et al. [12] transforms the upstream and downstream data according to user preferences. Additionally, MOWSER delivers only supported file formats with a format negotiating system. SWEEZER [53], Google mobile proxy [25] and SmartWeb [19] provide beside many others a similar solution. Fabio Paternò and Giuseppe Zichittella [49] proposed a more configurable and flexible proxy solution where user are able to set various preferences concerning the layout. Paternò and Zichitella's solution provides the starting point for the proxy solution within this thesis.
The previously listed mobile proxies resolve the drawbacks of the earlier mentioned solutions. But they do not allow the end users to adapt a web page with own methods according to EUD. Users are restricted to the given methods.

MOCUSI The step of allowing users to define their own methods is provided by MOCUSI (MObile CUstomised webSItes), the proxy solution proposed in this thesis. MOCUSI is described in detail in Chapter 5. The main achievement of MOCUSI is the integrated EUD idea which increases the degree of freedom for the user.

4

MJF Evaluation and Reference Implementation

When developing a mobile website or web app, it is currently state of the art to take advantage of MJFs and other JavaScript frameworks (e.g. jQuery or ExtJS). JavaScript frameworks provide an abstraction layer for browser specific implementations and facilitate framework specific features. The challenge for developers is to find the frameworks which apply best to their project needs.
This part of the thesis would not help other developers by only providing a project specific MJF evaluation. Therefore the evaluation in this thesis is split in three parts whereas the first two are project independent and reusable.
The first part lists the evaluation criteria (see Section 4.1). The second part provides the properties according to the evaluation criteria of the five MJFs jQuery Mobile, Sencha Touch, webapp.net, joHTML5, and wink toolkit (see Appendix A). The third part evaluates the five previously mentioned MJFs for a specific project (see Section 4.2).

Beside the criteria definition in Section 4.1 and the project specific evaluation in Section 4.2 this chapter provides a reference implementation in Section 4.3. The reference implementation is realised with the winner MJF of the project specific evaluation.

4.1 MJF Evaluation Criteria

The definition of the evaluation criteria including the unweighted evaluation are listed in this section.

4.1.1 Criteria Definition

Evaluation criteria have to describe requirements similar to project goals. Therefore we can apply the SMART [26] definition. Hence the criteria should be as far as possible:

- **specific** (well defined and clear to anyone that has a basic knowledge of the subject),
- **measurable** (include a measurement to know whether the criteria is fulfilled, or to what degree it is fulfilled),
- **achievable** (reachable by at least one of the evaluated objects),
- **relevant** (related to usability), and
- **tangible** (concretely formulated).

Additionally, criteria for JavaScript frameworks can be split in the four categories

- **basics** (basic framework properties),
- **supported features** (technically provided functionality),
- **programming** (circumstances at programming) and
- **maintenance** (support and persistence of the framework).

The four categories address criteria of all steps from development to redesign. Additionally, the categories help a specific audience to identify relevant criteria for them, e.g. a developer is most interested in the programming and maintenance properties whereas a project leader is rather interested in the basic framework properties and supported features.

The following list holds the most meaningful criteria for JavaScript frameworks [39] [47] [58] divided in the four before mentioned categories. Six of the criteria are mobile device specific and are marked with  . The list is completed with parameters enabling to measure the criteria.
The score of the evaluation is mainly divided in achieved (+) and failed (-)

regarding the criteria measurements. These two are extended by the increments well achieved (++) and totally failed (- -).

Basics:

- **Price / Licence:** Absolute price per year in USD is smaller than 100 and the type of licence belongs to the open source category.
- **Code size:** Size of the minified[1] and compressed (GZIP) JavaScript core code in kilobytes (KB) is smaller than 15 KB.
- **Future oriented:** Existence of a release plan. Update cycles are continuous (longest period is smaller than 3 months). Core development team consists of more than one member.
- **Compatibility:** Provisions exist to alleviate compatibility with other JavaScript frameworks. Preventions for name space conflicts are provided. Conventional development methods are applied (HTML → markup, CSS → style, JavaScript → behaviour).

Supported features:

- **Security:** At least one encryption method and https connections are supported.
- **Cross browser compatibility:** More than one browser engine is supported (for example WebKit and another one).
- **Cross platform compatibility :** More than two platforms are supported (browser implementations differ depending on the platform).
- **AJAX[2]:** Ability to handle AJAX.
- **Event handling :** Ability to bind events. Basic touch events are supported (tap, taphold and swipe). Gestures are supported.
- **Device hardware access :** Support for geolocation and local storage. Optional support for additional device hardware.
- **Animations / Behaviour :** Basic page transition animation (slide) and native application behaviour available (one page behaviour: no entire page reloads).
- **W3C validity :** Score of the W3C mobileOK[3] validation for the main demo page is above 50

[1] Optimised by deleting white spaces and unnecessary symbols.
[2] AJAX: Asynchronous JavaScript and XML[23]

- **CSS3 / HTML5:** The usage of CSS3 and HTML5 has to be leveraged.
- **Native application imitation:** Support for native application look and feel is available (mainly style sheet).

Programming:

- **Fast learning curve:** No new concepts of web page development are used (HTML → markup, CSS → style, JavaScript → behaviour). Only few framework specific coding conventions and namings are used. Tutorial is available.
- **Code style:** Well-defined and logical code structure, coding conventions and namings (good readability and maintainability) are specified.
- **Debugging:** Uncompressed source code is available. Debugging tools are available.
- **Documentation quality:** More than 90% of all core functions are explained and more than 50% of the most important functions are illustrated with examples.
- **Modularity:** Plugins and functions can be modified or exchanged with a small effort. Modular code structure is given in the meaning of object oriented code (→ OOP[3]).
- **Plugin support:** Plugin writing is supported. Plugins are available.
- **Quality of the code:** At testing the core JavaScript code with jslint[18] → no severe errors occur. No new global variables are defined. An own name space is provided.

Maintenance:

- **Size of the community:** More than 500 forum users are subscribed. More than 1000 posts are listed.
- **Quality of the community:** Core development team is active in the forum. Helpful answers are provided instead of redirects.
- **Maturity of the framework:** The framework is no longer in alpha or beta phase. Public version repository exists.
- **Frequency of public updates and releases:** Continuous and periodic updates are available. Updates are not more often than weekly (mostly caused by bug fixes) provided.

[3]OOP: Object Oriented Programming[11]

- **Commercial support:** One or more commercial supporter exist for the framework development.

Further:

- **Addition:** Miscellaneous remarks to the framework.
- For companies it might additionally be important whether knowledge of the framework already is present. Though this will not be considered in this thesis.

4.1.2 Unweighted Evaluation

Table 4.1 shows the general MJF evaluation which is reusable in future projects. Because it is not project specific the criteria are unweighted.
The evaluation handles the five MJF jQuery Mobile, Sencha Touch, webapp.net, joHTML5, and wink toolkit. The evaluation is done based on the property lists provided in Appendix A.

4.2 Project Specific Evaluation

The project which forms the base of the reference implementation in Section 4.3 is described in the first part of this section. In a next step the project specifications are listed. The specifications are afterwards used to weight the criteria. Finally the weighed evaluation is shown.

4.2.1 Specification

Project description

Customer The customer is the Swiss insurance agency *Helvetia*. *Helvetia* already provides a native iPhone app called *Helvetia Notfall-App* [1].

Customer needs The customer wants to reach a broader audience than just iPhone users. Additionally, the customer wants to provide the same functionality as the *Helvetia Notfall-App* provides.

Solution We decided to meet the two primary needs (broader audience and functionality preservation) of the customer with a web app. The functionality preservation implies that the target user group remains mobile and

Table 4.1: Unweighted MJF Evaluation

Criteria	Framework				
	JQueryMobile	SenchaTouch	webapp.net	wink	JoHTML5
Price / Licence	++	++	++	++	++
Code size	-	- -	++	++	+
Future oriented	+	-	- -	++	-
Compatibility	++	+	++	++	-
Security	++	++	- -	- -	- -
Cross browser compatibility	++	-	-	-	-
Cross platform compatibility	++	-	-	-	+
AJAX	++	++	-	+	- -
Event handling	++	+	-	+	-
Device hardware access	++	++	- -	++	-
Animations / Behaviour	++	++	+	++	-
W3C validity	+	+	- -	+	-
CSS3 / HTML5	++	++	-	-	+
Native application imitation	+	+	++	- -	-
Fast learning curve	++	+	-	+	-
Code style	+	++	-	+	-
Debugging	++	-	-	+	-
Documentation quality	+	++	-	++	-
Modularity	++	++	-	++	+
Plugin support	++	++	-	+	-
Quality of the code	+	-	-	+	+
Size of the community	+	++	++	- -	-
Quality of the community	++	++	+	+	++
Maturity of the framework	+	+	-	++	+
Frequency of public updates and releases	+	+	-	+	+
Commercial support	++	++	- -	+	- -
Addition	++	++	+	-	+
Unweighted sum	42	28	-11	19	-5
Rank	1.	2.	5.	3.	4.

the broader audience can best be achieved with a web solution. A first technical review assured that all important functionality can be realised with web technologies.

Specifications

In the following the specifications are listed for the reference implementation described in Section 4.3. The priority of the specification items are given in parentheses. The range of the priority reaches from 5 (very important) to 1 (nice to have). Specifications:

1. The application is used in emergency situations and has to work also if the network connection is interrupted. The map is an exception and only has to work if network connection is available. (5)

2. The application has to preserve the functionality to list the emergency information. (5)

3. The application has to preserve the functionality to trigger phone calls from listed phone numbers. (5)

4. The application has to run on as many different devices as possible. Required are iPhone and Android phones. (5)

5. The application has to preserve the map functionality including to display the current location with address information. (4)

6. The application has to preserve the functionality to store card and contact information. (4)

7. The application should cost as little as possible. (4)

8. The application should be available as soon as possible. (4)

9. The application should perform fluid on the device and use a small amount of data. (4)

10. The application has to imitate the look and feel of the native iPhone applications *Helvetia Notfall-App*[1] (3)

11. The application has to preserve the functionality to retrieve contact information form the address book on the mobile phone. (2)

12. The application is low security sensitive. Nonetheless, contact and card information must not be sent to the Internet. (2)

13. The application has to preserve the morse lamp functionality. (1)

4.2.2 Criteria Weighting

The specifications can be translated into technical terms and requirements related to the MJF criteria. These requirements together with the prioritisation allows weighting the criteria.

Technical requirements

The following list contains the technical requirements and affected criteria derived from the specifications. The affected criteria are provided in parentheses before the given priority. Hence, each requirement has the following form: requirement (criteria)(priority value)

1. The application has to be stored on the device using the application cache. (CSS3/HTML5)(5)

2. The application has to show text, pictures and navigable list elements. Mainly a question of how to achieve it best. (Quality of the community; Maturity of the framework; Documentation quality)(5)

3. The application has to use anchor tags with the `href="tel:...."` attribute or a similar solution. (Device hardware access; Maturity of the framework)(5)

4. The application has to be cross platform and cross browser compatible. W3C validity helps to be interoperable. (Cross browser compatibility; Cross platform compatibility; W3C validity)(5)

5. The contact and card information have to be stored with HTML5 local storage or cookies. (Device hardware access; CSS3/HTML5)(4)

6. The application has to contain map handling, location detection with GPS and reverse address fetching. Google maps provide JavaScript libraries and APIs to do this. To prevent page reloads AJAX handling is beneficial. (Plugin support; Compatibility; Device hardware access; Modularity; AJAX)(4)

7. The application has to run with open source code to reduce licence fees and maintenance costs. (Price/Licence)(4)

8. The application has to base on frameworks with a steep learning curve, helpful debugging tools, a good documentation, and a helpful community. (Fast learning curve; Code style; Debugging; Documentation quality; Quality of the community)(4)

9. The application should use frameworks with a small code size. Page changes should include fluid transitions enabled by loading following pages with AJAX. (Code size; AJAX; Animation/Behaviour)(4)

10. The application has to use a native iPhone theme or use special style sheets definitions. (Native application imitation; CSS3/HTML5; Animations/Behaviour)(3)

11. At the moment, address book contacts cannot be retrieved by web apps from mobile phones. (-)(2)

12. The application has to support basic security features. Contact and card information has to be on the device with HTML5 local storage or cookies. (Security; Device hardware access; CSS3/HTML5)(2)

13. The application has to handle touch events[4]. (Event handling)(1)

Weight calculation The weights are calculated with the priority values. All criteria start with the weight of one. Each time a criterion is listed in the technical requirements list the respective priority value is added to the criterion weight.
In other words the weight of a criterion is the sum of all priority values given to it plus one. E.g. weight calculation for criterion *CSS3/HTML5*: 1 (start value) + 5 (priority value of requirement 1.) + 4 (value of 5.) + 3 (value of 10.) + 2 (value of 12.) = 15.

Table 4.2: Weighted MJF Evaluation

Criteria (Weight)	Framework				
	JQueryMobile	SenchaTouch	webapp.net	wink	JoHTML5
Price / Licence (5)	10	10	10	10	10
Code size (5)	-5	-10	10	10	5
Future oriented (1)	1	-1	-2	2	-1
Compatibility (5)	10	5	10	10	-5
Security (3)	6	6	-6	-6	-6
Cross browser compatibility (6)	12	-6	-6	-6	-6
Cross platform compatibility (6)	12	-6	-6	-6	6
AJAX (9)	18	18	-9	9	-18
Event handling (2)	4	2	-2	2	-2
Device hardware access (17)	34	34	-34	34	-17
Animations / Behaviour (8)	16	16	8	16	-8
W3C validity (6)	6	6	-12	6	-6
CSS3 / HTML5 (15)	30	30	-15	-15	15
Native application imitation (4)	4	4	8	-8	-4
Fast learning curve (5)	10	5	-5	5	-5
Code style (5)	5	10	-5	5	-5
Debugging (5)	10	-5	-5	5	-5
Documentation quality (10)	10	20	-10	20	-10
Modularity (5)	10	10	-5	10	5
Plugin support (5)	10	10	-5	5	-5
Quality of the code (1)	1	-1	-1	1	1
Size of the community (1)	1	2	2	-2	-1
Quality of the community (10)	20	20	10	10	20
Maturity of the framework (11)	11	11	-11	22	11
Frequency of public updates and releases (1)	1	1	-1	1	1
Commercial support (1)	2	2	-2	1	-2
Addition (1)	2	2	1	-1	1
Weighted sum (max. 308)	251	195	-83	140	-31
Rank	1.	2.	5.	3.	4.

[4]Any mobile browser that does not support touch events by the end of 2010 will be out of the race. (http://www.quirksmode.org/mobile/advisoryTouch.html). Hence also MJFs have to support touch events!

4.2.3 Weighted Evaluation

To get the weighted evaluation the calculated weights for the criteria are multiplied with the values given in the unweighted evaluation ("++" = 2, "+" = 1, "-" = -1, and "- -" = -2). The resulting weighted MJF evaluation is illustrated in Table 4.2.

The weighted evaluation clearly shows that jQuery Mobile best matches the project specification needs. Therefore the reference implementation will be realised with jQuery Mobile.

4.3 Reference Implementation

The project description and specifications of the reference implementation can be found in Section 4.2. The goal of the reference implementation is to test whether the evaluation criteria, the MJF evaluation and the project specific evaluation can be proved as useful.

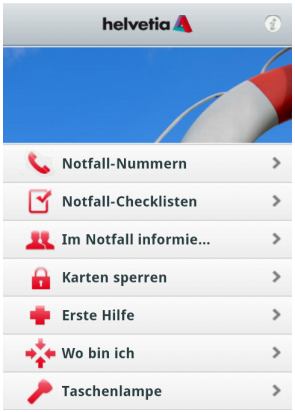

Figure 4.1: *Helvetia* start screen

4.3.1 Application Concept

The reference implementation includes the following challenges and solutions:

Offline availability Thanks to HTML5's application cache web apps can be stored on a device and run independent of a network connection.

Features (map, card and contact information handling, morse lamp, and phone calls) The features posed the biggest challenges. The **phone calls** can be handled by using anchor tags with a `href="tel:..."` attribute. The **morse lamp** can be implemented by toggling the background image (a white and black one) upon touching or clicking the image. The **card and contact information handling** can be realised using HTML5's local storage. The **map** can be implemented with the help of various JavaScript

libraries and APIs (jQuery UI Map [38], YQL Geo Library [27], and Google's Map API [24]).

Theming and behaviour The implementation has to imitate the iPhone app. From the visual aspect jQuery Mobile's theme can be used to achieve it as far as possible. The rest has to be solved with CSS definitions (e.g. with CSS definitions from http://imgless.com). The behaviour of transitions is no challenge as the standard slide transition is supported by default in jQuery Mobile.

Content The content was provided by the source code of the native iPhone app which *Namics AG* offered in agreement with *Helvetia*. Given that the content was mainly written in HTML and many styles were defined with CSS the adaptation for the web app was minimal.

4.3.2 Application Implementation

In the following, main information about the implementation is provided. Details about the implementation can be found in Appendix B.

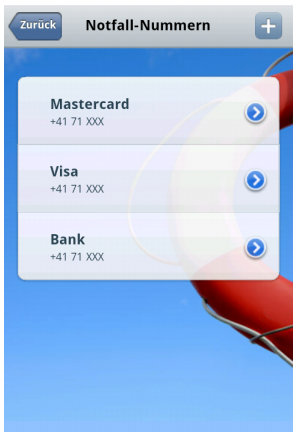

Figure 4.2: *Helvetia* card information

HTML The HTML part consists only of the `index.html` file. The index file holds the whole text information.
For normal websites or web apps it would be better to split the content in different files to reduce the data traffic by only loading content that is needed. As the *Helvetia* web app is implemented with application cache (see Section 2.4.6) it reduces network traffic as all files will be downloaded anyway. This is the case because all content in one file reduces the number of file requests and therefore the network traffic.

JavaScript The JavaScript part consists of five local JavaScript files and one remote JavaScript file. Four of them are only used to handle the map feature. The `helvetia.js` is the only one written specifically for the project.

CSS The CSS part consists of three local files. One is provided by jQuery Mobile, `styles.css` is imported from the native *Helvetia Notfall-App* and the last one (`helvetia.css`) is written specifically for the project. `helvetia.css` overwrites styles from jQuery Mobile and defines new styles needed to imitate the native *Helvetia Notfall-App*.

Images Many images and icons are used within the reference implementation. Most of them origin from jQuery Mobile and from the native *Helvetia Notfall-App*. Only the "info icon" and the "add button" for the contact and card information pages were additionally important.

4.3.3 Result

The reference implementation showed, that jQuery Mobile allows to achieve a reasonable solution within 2 weeks. Therefore, the project specific evaluation chose an adequate MJF.
During the writing of this thesis jQuery Mobile advanced to the fourth alpha version. Many bugs where fixed and new devices are supported. Unfortunately the alpha status is still perceivable in special cases or by looking at the few qualified answers in the forum.
While implementing the reference implementation we observed divergences between the evaluated properties of jQuery Mobile and the made experiences for the following criteria:

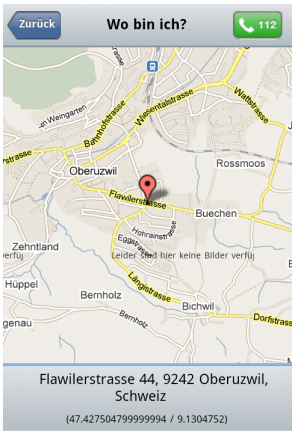

Figure 4.3: *Helvetia* map

- **Future oriented:** The promised 1.0 release in January 2011 could not be hold. Till now jQuery Mobile is available in version 1.0 alpha 4.1.
- **Device hardware access:** The geolocation plugin jQuery UI Map is very useful. But additional JavaScript files have to be included with a total size of 200Kb. Consequently plugins have to be considered in future evaluations.
- **Animation/Behaviour:** If all pages (`div` with `data-role="page"`) are contained in one file the page transition is no problem. But to preserve the overview it is preferable to locate different pages in different files. Unfortunately it is either hard to achieve or not satisfying to link

pages in external files. All JavaScript files and CSS definitions have to be redefined in the external files.

- **W3C validity:** The reference implementation achieved 0% with mobileOK. Maybe page enhancements would lead to a better result.
- **Native application imitation:** Some buttons, icons and styles had to be manually included to mimic native iPhone style.
- **Quality of the community:** Although the jQuery community is large it seems that in the current alpha version questions tend to remain unanswered for a while. This may be the case as many core developers are merely busy with bug fixing and implementing change requests than answering questions.
- **Frequency of public updates and releases:** Same as for the criteria **Future oriented**.

4.4 Recapitulation

To reach the goal of enhancing mobile website usability five MJF were analysed and evaluated in the first part of this chapter. In the second part a reference implementation was described. In the following the achievements and insights of this chapter are described.

4.4.1 Achievements

The MJF evaluation within this chapter made an important contribution by allowing developers to compare various MJF to answer the frequently asked question which MJF one should choose.
The chapter provides not only evaluation criteria to compare MJF but also an unweighted evaluation of five representative MJF.
An additional contribution is made by the property lists of the five MJF in Appendix A.
The MJF evaluations (unweighted and weighted) showed that jQuery Mobile is a front player amongst the MJF although it still has alpha status.
With the reference implementation at the end of the chapter we showed that MJFs allow to imitate a native app reasonable with a relatively small effort.
The reference implementation allows *Helvetia* to maintain the content out of their CMS, to update content without the acceptance process needed for native apps, and to be platform independent to provide a wider range of target devices.

4.4.2 Insights

MJFs are characterised by their support of user interface (UI) components, animations, event handling, or layouts which enhance mobile website usability.
The MJF jQuery Mobile demonstrated in use that it enhances website usability by default. The given layout and components help to provide usability for mobile users but in turn limits the design of developers. Customising the layout or components is possible but awkward.

An important point regarding the usability guideline in Section 3.1.2 that jQuery Mobile does not consider, is to keep the implementation small, especially when plugins are needed. Future project specific evaluations have to consider the size of needed plugins.
Some of the guidelines are out of the scope of MJF in general like the amount of displayed content or the auto-forwarding which should be defined by the markup and not by the framework.

The observed divergence between the jQuery Mobile property list and our experiences taught us that the maturity of a framework has much to do with its behaviour in extreme cases. jQuery Mobile is in alpha status and does not yet handle special cases, e.g. offline support for application cache. The reference implementation verified that jQuery Mobile is adequate to create a useful mobile web app and to enhance mobile website usability.

5

Mobile Proxy Solution MOCUSI

MOCUSI is a prototype proxy solution developed within this thesis. It consists of a proxy and a client part. The goal of MOCUSI is to enhance website usability by allowing users to configure transformations for arbitrary websites according to their needs. This chapter describes the concept of MOCUSI in the first section and the implementation in the second one. The third section explains the conducted usability test for MOCUSI and lists the results.

5.1 Concept

MOCUSI has to obtain enhancements of the mobile usability for already existing websites. In contrast to MJFs (see Chapter 4) which enhance the usability for mobile phones in the creation phase of a website.
To achieve the goal of enhancing website usability for mobile phones a proxy solution like MOCUSI has to modify websites. The modification happens at the proxy part of MOCUSI between the web server and the user. With MOCUSI, users are able to display existing websites which are not optimised for mobile phones in a modified version. The modified version is customisable and enhances the usability for mobile phones depending on the customised settings.

Basic concept

Basically, MOCUSI is a proxy solution consisting of a website as a client part and a proxy. The website serves as a wrapper application to bookmark websites, configure modifications for bookmarked websites and display the resulting websites. The proxy on the other hand fetches, modifies, and caches websites.

Website commonalities

To ensure that a proxy has the ability to modify arbitrary websites, it has to base on commonalities amongst websites. Existing websites have the following in common:

- **Protocols:** Websites are based on HTML/CSS/JavaScript although CSS an JavaScript are optional.
- **Media types:** Websites mainly consist of the media types text, image, video, and embedded elements like flash or applets.
- **Resource identifier:** Websites are related to an Uniform Resource Identifier (URI) or Uniform Resource Locator (URL).
- **Interactivity:** Most websites are interactive. Interactive websites contain interaction elements like navigation links, buttons, input fields, and selection elements.

Transformation

As explained in Section 3.3 a proxy can transform websites. Hence a proxy can apply arbitrary changes to a delivered web page. But it has no influence on the server which creates and delivers the page. Proxies are middle ware between the server delivering a web page and the client browser which renders it.

Transformations applicable on arbitrary websites have to respect commonalities of websites. Out of the previously listed commonalities MOCUSI considers the URL, images, and the protocols HTML, CSS, and JavaScript. The remaining commonalities will pose no difficulty to be implemented and are similar to the considered ones. The implementation of these remaining commonalities is left open for future work as MOCUSI only is a prototype.

The MOCUSI prototype is based on transformations enhance the usability of websites profoundly. The transformations of MOCUSI include to enable or disable images, CSS files, or JavaScript files. These profound settings have most influence on the style, behaviour and data size of a website. It enables the user to change general characteristics of a website.

EUD

Next to the previously mentioned transformations MOCUSI integrates the EUD idea. EUD allows users to provide their own modifications to websites by providing CSS style sheets or JavaScript files which will be included in the source code of the displayed website. Hence users can define how the layout should look like, how the website has to behave, or which elements should be hidden.
However this EUD solution requires programming knowledge of either CSS or JavaScript. Nevertheless users with no programming skills are able to use this EUD approach because the programming can be done by the community. Users only have to specify the resource URL for the CSS or JavaScript file which should be injected. It does not matter where the file comes from.
The user is responsible for the resource selection to prevent malicious code injection. Particularly because malicious code only affects the client device.

Target devices

Considering the cutting edge progress in the mobile phone market MOCUSI focuses on mobile phones with touchscreens and JavaScript enabled browsers. This can be reasoned with the fact that most smartphones on the market are equipped with this features. Android phones and iPhones have a fast growing market share and can be taken as example. In Q1 2011 they had a combined smartphone market share of over 50% worldwide [16].
All Android phones and iPhones are already equipped with a touchscreen. They also provide a JavaScript enabled browser according to the *Mobile Graded Browser Support* list of jQuery Mobile (`http://jquerymobile.com/gbs/`).

Used technology

The proxy part of MOCUSI is built on PHP 5.3 which provides useful tools and approaches to fetch and edit source code of websites. The wrapping application at the client side is built on jQuery Mobile. jQuery Mobile ensures a good mobile usability with its default interface and verified to be

useful in Chapter 4.
Most of the client side MOCUSI data has only to be downloaded once because it is implemented using the HTML5 application cache.

Computation power of the proxy

To take advantage of the computation power of the proxy and to relieve the computation power of mobile clients the modifications of websites are done on the proxy side as far as possible. This division of work load also reduces data traffic as only needed data is sent to the client. E.g. if images, CSS files, and JavaScript files are disabled the proxy does not have to send them and can send merely a text file.

Workflow

The workflow of MOCUSI works like depicted in Figure 5.1. A mobile client visits the MOCUSI website (wrapper application). From the start page of the wrapper application the user sends a URL request to the proxy. The proxy checks whether the requested web page is already in the cache. If it is, the cached web page will be processed. Otherwise the proxy will fetch the web page from the web server defined by the URL and saves the web page in the cache. Afterwards the proxy modifies the web page according to the configurations provided by the client. Finally the transformed web page is delivered to the client and rendered by the browser.

Interface

The user interface of MOCUSI basically provides the following:

- An input field to define the website which has to be adapted (see Figure 5.2)
- Predefined modes which describe sets of configurations which can be applied to websites (see Figure 5.3)
- An interface to set own configurations (see Figure 5.4)
- Preview of made configurations (see Figure 5.5)
- Possibility to save websites with the made configuration (bookmark) (see Figure 5.6)
- A list of all stored bookmarks (see Figure 5.7)

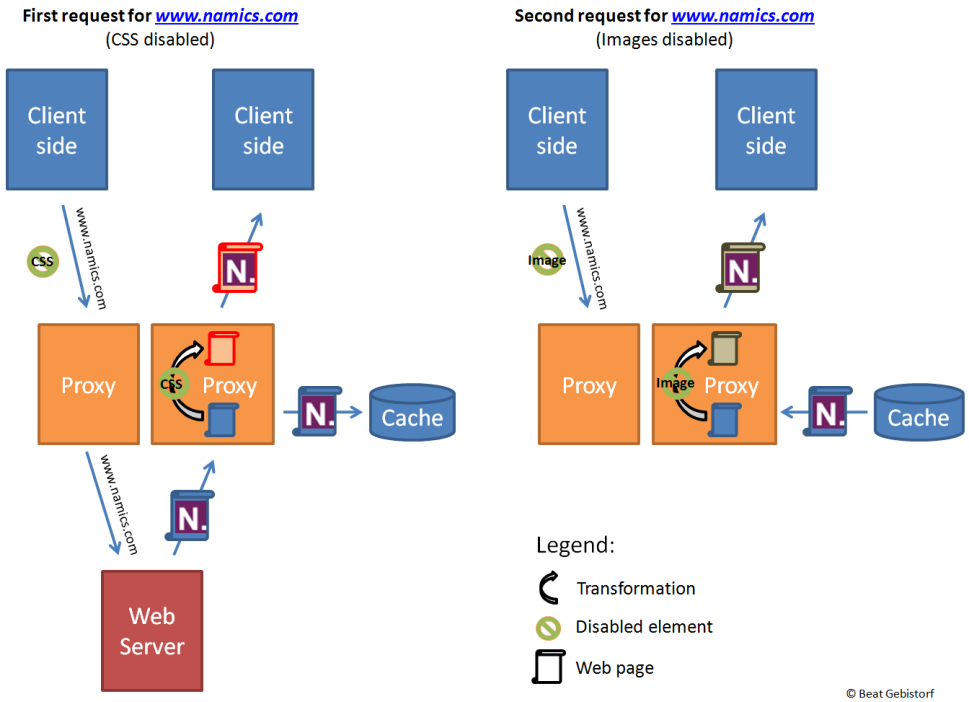

Figure 5.1: MOCUSI workflow from a website request over the transformation until the rendering.

Figure 5.2: MOCUSI start screen

Figure 5.3: Adaptation modes

Figure 5.4: Custom configurations

Figure 5.5: Website preview

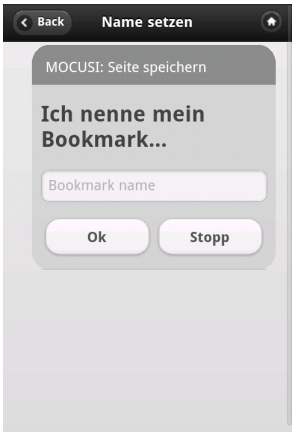

Figure 5.6: Save bookmark

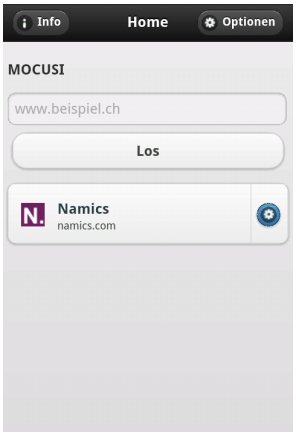

Figure 5.7: Bookmark list

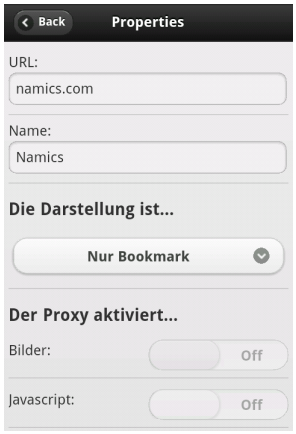

Figure 5.8: Edit bookmark

Figure 5.9: Add CSS or JS files

Figure 5.10: Information screen

- An edit mode to change configurations of bookmarks (see Figure 5.8)
- An interface to include own CSS or JavaScript files by specifying the resource URI (see Figure 5.9)
- An information page (see Figure 5.10)

Usability

MOCUSI follows the usability guidelines in Section 3.1.2 in most instances. The guidelines were ignored in some points to achieve a better usability

for cutting edge phones like iPhones or Android phones. The guideline are ignored by

- using an iframe which is necessary to remain in the wrapper application context at displaying the modified website,
- relying on JavaScript and either HTML5 local storage or cookies to save websites with the respective configurations as bookmarks,
- relying on JavaScript to handle events and trigger actions,
- using absolute layout sizes to reduce development time, and
- using overlay elements to keep navigation simple and well-arranged.

The above listed guideline disregard will cause problems on mobile phones which have a small screen, a browser which does not support JavaScript, or a browser that does not support up to date HTML and CSS features. The disregards could be eliminated with a huge effort which would be out of scope of this thesis.

In consequence, MOCUSI defines the following requirements: To be able to use MOCUSI mobile phones have to support

- JavaScript,
- either HTML5 local storage or cookies,
- iframes, and
- overlay elements.

To meet the usability guidelines *Design for One Web* and *Desktop version* (see Section 3.1.2) MOCUSI provides a configuration to display the unchanged desktop website.

Benefits and drawbacks

Users who use MOCUSI to browse websites could benefit from

- reduced data traffic by only fetching the selected resources of the website (regarding image, CSS, and JavaScript),
- a wrapper application which can be cached and has only to be downloaded once to cause no additional traffic (settings can even be changed offline this way),

- customisable website modifications,
- extendable website modifications by providing own CSS or JavaScript resources,
- a solution which can be applied on arbitrary websites,
- fast transformation computation on the proxy on behalf of the client device,
- the bookmarking functionality to save the favourite websites including the modification configuration, and
- the simple interface which eases to set configurations.

Users who use MOCUSI suffer from the drawback that

- they have to trust MOCUSI because all content passes through the proxy,
- Secure Sockets Layer (SSL) connections are not supported yet,
- MOCUSI adds an additional point of failure,
- the proxy might become a bottleneck in the network chain, and
- MOCUSI only works on mobile phones which fulfil the requirements described earlier.

5.2 Implementation

The implementation of MOCUSI consists of a proxy side implementation written in PHP and a client side implementation written in HTML/CSS/-JavaScript.
In the rest of this section details are provided to explain the proxy side and client side implementation.

5.2.1 Proxy Side

The proxy implementation consists of the files `proxy.php`, `addScript.php`, `getScript.php`, `checkUrl.php`, `getIcon.php`, and `manifest.php`. All except `manifest.php` work with provided parameters which can be passed to the php script with the query string.

- `proxy.php` contains the main proxy logic which modifies web pages according to the configurations of the client.

- `addScript.php` adds a script (CSS, JavaScript or PHP[1]) to the proxy cache if the passed script source is valid.
- `getScript.php` fetches all scripts stored in the proxy cache matching the passed type (CSS, JavaScript or PHP).
- `checkUrl.php` tests whether a passed URL exists.
- `getIcon.php` returns a favicon for the passed website.
- `manifest.php` contains the configuration for the application cache.

The proxy uses the two PHP methods `file_get_contents` and `curl_exec` to fetch web pages from web servers. `curl_exec` provides more settings than `file_get_contents` which is simpler. In `checkUrl.php` the method `@fopen` is used to check whether a URL exists.

The proxy is capable of

- caching a web page for one hour (duration is hardcoded),
- caching of scripts (CSS, JavaScript and PHP),
- rewriting links and URLs so that they use the proxy,
- removing JavaScript and CSS import definitions,
- removing JavaScript script sections,
- removing CSS style sections and inline style definitions, and
- adapt the used useragent to fetch the web page from the web server in a desired version (basically mobile or desktop).

5.2.2 Client Side

The client side of the prototype MOCUSI provides the wrapper application. The wrapper application allows users to specify which website the want to see (by entering a URL), configure how the seen web page should be displayed and save the web page and the made configurations as a bookmark. In other words the client side of MOCUSI is a bookmarking application which allows to save and edit modification configurations within the bookmarks.
The client implementation consists of several HTML, CSS, JavaScript, and image files.

[1] Some code sections contain PHP code injection handlers which are provided for a later extension. At the moment only CSS and JavaScript file injection are supported.

- `index.html` includes all HTML code. The German version can be found as `indexd.html`.
- `base.css` overwrites some of the jQuery Mobile styles and defines the layout of the wizard. The wizard describes the process from entering a URL until saving a bookmark.
- `base.js` contains the main logic of the wrapper application on the client side. It provides the bookmark handler, configuration handler, the data interface to the proxy and the predefined adaptation modes. The German version can be found as `basede.js`.
- As jQuery Mobile is used as MJF the corresponding CSS, JavaScript, and image files of jQuery Mobile are used.
- A few additional images are used on the information page to illustrate the adaption modes.

Configuration options

MOCUSI provides the following configuration options for website modifications:

- Disable or enable images.
- Disable or enable JavaScript code sections and JavaScript file imports.
- Disable or enable CSS style sections, inline style definitions, and CSS file imports.
- Choose whether the mobile user agent should be used or one of a desktop device.
- Disable or enable optimised scaling of the web page.
- Optionally define a font size which will be applied to the documents body.
- Selection of CSS and/or JavaScript files that have to be injected.

Predefined adaptation modes

The following predefined adaptation modes are provided by MOCUSI:

- **Minimal:** All options are disabled. Only the user agent is chosen to be a mobile one.

- **Standard:** Additional to the mobile user agent images and CSS styles are enabled.

- **Maximal:** All options are enabled. The user agent is chosen to be a desktop one. Hence this mode shows the version a desktop browser would display.

- **Bookmarked only:** The options are irrelevant because the bookmark links directly to the original web page like a web link. This means that the browser will leave the MOCUSI context.

One has to select the **customised** mode to configure the modifications oneself.

5.2.3 Remaining Issues

Some important issues could not be resolved within the prototype implementation:

- The link and URL rewriting is a sophisticated part of the proxy and does not yet work for all URLs.

- The query string of websites is not yet considered at the proxy.

- Proper removing images, CSS, and/or JavaScript definitions does not work on complicated websites, e.g. the websites of Google or Yahoo which build nearly the whole page on minified JavaScript code.

- The native Android browser contains a bug called "bleeding through". By clicking on an overlay the click event is applied to the element underneath the overlay.

5.2.4 Injection Example

An example of CSS and JavaScript file injection is provided for the website http://www.worldatlas.com/aatlas/world.htm. The two injected files are `worldatlas.js` and `worldatlas.css`. Both are about 60 lines of code long and achieve an impressive layout change (see Figure 5.11 and Figure 5.12). The two files remove ad banners and redundant navigation elements, change the layout to a one column layout, and scale to large image to fit in the screen width.

Figure 5.11: Original worldatlas website

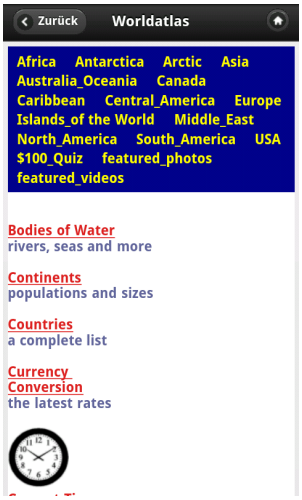

Figure 5.12: Worldatlas website with CSS and JavaScript file injection

5.3 Usability Test

The usability test was conducted to get information about the subjective usability perception of MOCUSI by users and to validate its everyday usage potential. The test bases on provided knowledge from *Namics AG* and the *Usability Test Plan Toolkit* [60]. The test structure is similar to the one in W3C's *WAI Site Usability Testing* [29].

5.3.1 Test Configuration

The test involved 12 test persons. They where chosen according to their technical background and mobile phone experience. The test persons can be categorised in the following groups (in parenthesis the number of test persons per category is marked):

- Mobile phone user and developer of mobile apps (1)
- Mobile phone user and developer of web apps (3)
- Mobile phone user and technician in a wider sense (5)
- Mobile phone user and not a technician (2)
- Not mobile phone user (1)

The test was performed on an iPhone 4.
The following goals, answers and results have to be understood in the mobile context.

The main goal of the usability test was to find out whether

- MOCUSI is intuitively usable,
- user would use MOCUSI (whether they find it valuable/useful), and
- MOCUSI enhances the usability of browsing websites.

A secondary goal was to find new bugs.

The structure of the test was the following (in parenthesis the duration in minutes of the part is marked):

1. (1') Welcome and introduction
2. (2') Introduction questions
3. (5') Introduction test cases (comparison of the original and adapted worldatlas website, see Figure 5.11 and Figure 5.12. One task for each case)
4. (1') Explanation that MOCUSI is built for mobile website usability enhancement
5. (2') Ask what the user thinks MOCUSI might look like and what it provides
6. (1') Show MOCUSI start screen
7. (2') Ask for thoughts about the start screen, what the user now thinks MOCUSI provides
8. (6') Let the user play with MOCUSI for 5 minutes (explanations by the tester: user has to think aloud including emotions and has to be honest; tester will not help the user; inform the user that he can not handle wrongly.)
9. (30') Complete the tasks (34' for advanced technicians with an additional task)
10. (10') Ask the follow up questions
11. (1') End of test

A more detailed version of the test can be found in Appendix C.

5.3.2 Result Summary

MOCUSI got a score of 5.3 on a scale from 1 (not usable) to 10 (usable very well). Various statements of the tested persons indicated that MOCUSI does not enhance their mobile Internet usability because they either do not use mobile Internet or because they became accustomed with the way mobile phones display websites which are not optimised for them.
The results regarding the main goals are described in the following. Within the results "Q" is used as abbreviation for "Question" to indicate which follow up questions led to the stated result (follow up questions can be found in Appendix C.

Find out whether MOCUSI is usable intuitively (Q13, Q14, Q16, Q17, Q18, Q19) 75% of the users think, they understood MOCUSI after using it for about an hour. Even non-touch-phone-users were able to use MOCUSI after the test. Naming or label comprehension were the biggest challenge.

Find out whether user would use MOCUSI (whether they find it valuable/useful) (Q7, Q8, Q9, Q10) 50% of the users would use MOCUSI in the future.

Find out whether MOCUSI enhances the usability of browsing websites (Q5, Q7, Q8, Q9, Q10) 42% would recommend MOCUSI. 25% would use MOCUSI daily.

The two introductory tasks with the *worldatlas* website and the respective questions served as reference to prevent biased results. These reference tasks and questions were done before the user knew what MOCUSI provides and what it looks like. Hence the users were not able to interrelate the tasks and questions with MOCUSI and were thus not biased.
In this reference part 50% of the users preferred to use the transformed *worldatlas* website with MOCUSI. It showed a similar result like the others. Hence it can be stated that the test was unbiased with a high probability.

5.4 Recapitulation

To reach the goal of enhancing mobile website usability the prototype of a proxy solution called MOCUSI was developed in this thesis. In the following the achievements and insights of this chapter are described.

5.4.1 Achievements

MOCUSI implements a representative solution for mobile proxies. MOCUSI allows users to decide which modification methods will be applied to websites and to add customised modifications. That is achieved by configuring the modification methods an injecting CSS and/or JavaScript files in a selected website. Afterwards the customised configuration can be saved together with the website for later use as bookmarks.
MOCUSI meets EUD even though CSS and JavaScript coding needs programming experience. EUD is met because the CSS and JavaScript code can be imported from foreign sources. In other words, EUD is ensured because users can benefit from provided solutions of the community.
With MOCUSI, users are empowered to see a desired distillate of a website. This increases the value of the mobile Internet usage as described by Gaddo F.Benedetti: *"What sells the mobile Web is not how it is similar to the desktop Web, but how it differs."* At the same time MOCUSI preserves the One Web (3.1 in [62]) vision because the user can easily visit the unchanged desktop version.
The conducted tests confirmed the usability of MOCUSI based on the subjective perception users gained of MOCUSI. 50% of the tested persons acknowledged that MOCUSI would enhance the usability of their mobile Internet usage. Hence, MOCUSI achieves its main purpose to enhance the mobile web usability for at least half of the tested group.

5.4.2 Insights

The implementation of MOCUSI showed, that despite the commonality of website, it is hard to find transformation methods which apply to all websites.

The usability test was very helpful for gaining further insights in the behaviour, perception, and way of thinking of mobile users.

- A majority of the test persons liked the simple start screen and navigation. More than 30% of the users were curios to use the input field and the button below. A majority of the tested persons indicated that they knew exactly what to do with the input field and the button without introduction.
- More than half of the tested persons complained about the unclear mode names. Usability and easy navigation depends heavily on clear naming concepts and UI element labels.
- MOCUSI was perceived as intuitive by the majority of the tested persons. The few configuration options and simple navigation concepts

allowed users to learn how to use MOCUSI within about half an hour of usage.

- The flat navigation hierarchy helped users to keep the overview. Nearly all tested persons confirmed that MOCUSI has a clear structure.
- Mobile users do not like to read much. About a quarter of the tested persons stated that they would appreciate a visual introduction to MOCUSI.

Because the usability test was performed at the end of the implementation no time left to counter the identified problems.
In a next project we would conduct a usability test earlier in the development cycle to get first insights of usability problems and obstacles. Hence one can counter the problems earlier and easier. Additionally, we would perform usability tests several times within the development process to get regular feedback.

6 Conclusion

Enhancing the usability of websites for mobile phones was the goal of this thesis. This chapter will summarise the achievements and gained insights beside providing possible improvement ideas for future work.

6.1 Summary

This master thesis showed various approaches to enhance the usability of websites for mobile phones. On the one hand, approaches to enhance usability at the development phase of a website and on the other hand to enhance usability by modifying existing websites during the delivery. A listing of achievements of the thesis and insights are given in the following.

6.1.1 Achievements

The usability guideline in Chapter 3 consolidates various existing guidelines and provides a useful reference for developers of mobile websites.
A further aid for mobile website developers poses the MJF evaluation. It provides reusable evaluation criteria, MJF property lists of five popular MJFs, and an example of a project specific evaluation.
The reference implementation with the MJF jQuery Mobile showed, that a specific native application can be imitated with a web app very well with little effort. Even the offline capability can be provided. It empowers *Helvetia* to maintain the content more efficiently and to reach a broader audience.

The mobile proxy prototype MOCUSI successfully enables users of mobile phones to customise websites. The usability test verified that MOCUSI is feasible and enhances mobile usability for many users.

MOCUSI provides end users a tool to optimise existing websites for mobile phones as they can not influence the website development.

The implementations and the usability test validated that the usability guideline in Section 3.1.2 is feasible.

Goal achievement

Both goals of the thesis are achieved.

- **Guidance for mobile website developers:** The usability guideline, the MJF evaluations and the reference implementation provide a guidance for mobile website developers to enhance the usability of websites for mobile phones.
- **Solution for website users:** MOCUSI provides a solution for users to enhance the usability of existing websites for mobile phones.

Achievement of the expected results

The majority of the expected results are achieved.

- **Listing of guidelines:** A usability guideline is listed in Section 3.1.2 and provides a manageable reference of important principles for mobile website developers.
- **Reusable evaluation of state of the art frameworks:** The evaluation criteria and the unweighted evaluation of MJFs in Chapter 4 consider state of the art technologies and frameworks. They are both reusable for future projects.
  In contrast to the unweighted evaluation the weighted evaluation illustrates a project specific evaluation. It cannot be reused but it can be imitated by future projects.
- **Specific native app imitation with a web app without many constrains:** The reference implementation with jQuery Mobile described in Section 4.3 imitates the native *Helvetia Notfall-App* application very well. Beside the access to the address book of mobile phones

all features are imitated by the web app. The big advantages of the web app are the cross platform compatibility and the good maintainability.

- **Proxy solution enabling users to customise existing websites:** MOCUSI which is explained in Chapter 5 enables users to customise websites according to their wishes and to bookmark websites with the respective configurations. MOCUSI did only partially achieve the expected result. It did only achieve the prototype status because of the time constraints of the thesis. A complete implementation would have been too extensive. Hence, the prototype does not offer a satisfying result for all websites and has some requirements for devices.
- **Usability test to approve the proxy solution:** The conducted usability test verified that MOCUSI is useful based on the fact that half of the tested persons stated that they would use a system like MOCUSI in the future.

6.1.2 Insights

The implementation of MOCUSI revealed, how sophisticated the handling of different websites is. This challenge of proxies and the comparable small effort of the reference implementation leads to the conclusion that enhancement of mobile usability should in favour happen when developing a website. The enhancements done in the development phase can be very content specific and have not to consider other websites. In contrast to a proxy solution which has to handle arbitrary websites.

The usability test indicated that

- the defined usability guideline provides viable points,
- one has to respect that users like what they are familiar with (new navigation concepts, styles or behaviours have to be used cautiously), and
- further versions of MOCUSI have to base on a clear use case. As the use case was not clear for the tested persons. Hence usability enhancements for websites rely on a clear idea of the use case which has to be met.

6.2 Future Work

As MJFs mature over time one has to adjust the property lists provided in Appendix A regularly.

The MJF evaluation can be extended by implementing a web app for each evaluated MJF and comparing the results. This would allow a more significant comparison. In addition, it would help to verify the property lists.
The created *Helvetia* reference implementation was developed with the iPhone and Android phones in mind. It has been left open to be tested on various devices to formulate requirements. Especially regarding the offline application cache.

MOCUSI needs further improvements to properly cover a wider range of websites. The content transformation at the proxy can hereby be improved. The proxy provides code which is foreseen to allow users to inject PHP code. This PHP code allows modification of websites at the proxy and not only at the client like with CSS or JavaScript code injection. The PHP injection would offer a valuable improvement but one has to consider the security issues this would imply.
MOCUSI can be extended with a login functionality to allow users to edit and use their bookmarks on various devices.
Finally, MOCUSI can be improved by meaningful component replacements (e.g. of the iframe or overlay) leading to lower device requirements.

We see a huge potential in the large number of websites without a mobile optimised version. This huge potential is met with the solutions presented in this thesis. **The usability guideline, MJF evaluation, MJF property lists and the reference implementation serve developers of mobile websites** as reference to enhance the usability of their website for mobile phones. **The proxy solution MOCUSI provides end users** with the opportunity to visit an arbitrary website with customised modifications applied to enhance the website usability.

A

MJF Evaluation: Property Lists

Within this appendix the criteria listed in Section 4.1 are applied to five of the most popular MJFs (JQuery Mobile, Sencha Touch, webapp.net, Wink Toolkit and joHTML5). Following the MJF properties according to the criteria can be found in five tables. One table for each MJF. The tables catch the **state on the 29th of December 2010**.
The information for the tables are mainly retrieved from the web pages of the frameworks [36][33][64][4][9]. Additional information were gained from test implementations of a simple web page with a twitter stream using sliding page transitions.

Table A.1: JQuery Mobile Properties

Criteria	JQuery Mobile
Price / Licence	open source / MIT, GPL
Code size	14 KB (jQuery Mobile) + 26 KB (jQuery) = 40 KB
Future oriented	1.0 release of the platform is planed for January 2011. JQuery developer team with more than one member. Update cycle is continuous.
Compatibility	Name space conflicts prevented with own name space and with jQuery.noConflict(). Applies conventional development methods.
Security	Yes, with jQuery plugins.
Cross browser compatibility	Like jQuery, IE 6.0+, FF 2+, Safari 3.0+, Opera 9.0+, Chrome.
Cross platform compatibility	iOS, Android, Blackberry, Palm WebOS, Nokia/Symbian, Windows Mobile, bada, MeeGo with baseline support for all devices that understand HTML
AJAX	Yes
Event handling	Yes, binding of events possible. Tap, taphold, swipe, swipeleft, swiperight, orientationchange, scrollstart, scrollstop, pagebeforeshow, pagebeforehide, pageshow, pagehide, pagebeforecreate and pagecreate event exist. Gestures are supported.
Device hardware access	Geolocation with plugins and localStorage (jStorage) supported.
Animations / Behaviour	Pagetransitions: slide, slidedown, slideup, pop, fade and flip. New pages are loaded via AJAX not with page reload.
W3C validity	mobileOK: jquerymobile demo 60%
CSS3 / HTML5	Yes, CSS3 & HTML5.
Native application imitation	Yes, mimic iPhone.
Fast learning curve	If jQuery known, yes. Have to learn the attribute and function names. Some external tutorials available.
Code style	Meaningful namings. Code structure not defined. Good coding conventions. Well-defined function interfaces. Complicated pages may be cumbersome.
Debugging	Uncompressed source code available. jQuery debug allows helpful debugging.
Documentation quality	jQuery, very good documented. jQuery Mobile, well documented. Most functions are illustrated with examples.
Modularity	Yes, OOP.
Plugin support	Yes, plugin writing supported. Plugins available.
Quality of the code	jslint: stopped at 90%. Only a few mentionable errors. 4 new global variables. Namespace: *$.** or *jQuery.**.
Size of the community	JQuery had more than 500 users in the forum. JQuery Mobile had 803 posts.
Quality of the community	The answers are good and are given fast. The jQuery development team is active in the forum.
Maturity of the framework	Based on the mature JQuery. JQuery Mobile in alpha status. No version repository for jQuery Mobile.
Frequency of public updates and releases	Alpha releases each month. Version 1.0 in January 2011 expected.
Commercial support	Media temple, mozilla, fillament group, palm, blackberry, nokia, deviceAtlas, dotMobi and adobe.
Addition	JQuery follows the progressive enhancement and graceful degradation principle. It supports wai-aria (Web Accessibility Initiative - Accessible Rich Internet Applications[30]).

Table A.2: Sencha Touch Properties

Criteria	Sencha Touch
Price / Licence	Open source & commercial licence 299$ one time fee for one developer (1395$ for five) / Dual Licencing Model: Quid Pro Quo (commercial use: closed source → buy, open source → GPL v3).
Code size	95 KB
Future oriented	No release plan for Sencha Touch. 8 technical workers for sencha. Update cycle is continuous.
Compatibility	Name space conflicts prevented with own name space. Does not apply conventional development methods. Markup, content and behaviour are all programmed in JavaScript. Styling done with Sass[2].
Security	Yes, with Ext JS features.
Cross browser compatibility	All WebKit based browsers supported.
Cross platform compatibility	iOS, Android.
AJAX	Yes
Event handling	Yes, binding of events possible. Touchend, touchstart, tap/hold, touchmove, doubletap, swipe, pinch, rotate. Gestures are not yet supported / documented.
Device hardware access	Geolocation and localStorage supported.
Animations / Behaviour	Pagetransitions: slide (normal,cover,reveal), pop, fade, flip, cube. One page behaviour.
W3C validity	mobileOk: kitchensink 65%
CSS3 / HTML5	Yes, CSS3(Sass) & HTML5
Native application imitation	Yes, mimic iPhone and Android.
Fast learning curve	One has to learn Sencha Touch functions and structure. One has to program both the behaviour and markup in JavaScript. If Ext JS known it is easy to learn. For the styling Sass knowledge is a prerequisite. Tutorials and introduction videos are available.
Ease of development	Is OOP.
Code style	Good and logically structured code. OOP with intuitive namings. Well-defined function interfaces. Sass helps to define the style.
Debugging	Uncompressed and well documented source code available. No debugging tools.
Documentation quality	Very good documentation (Ext SJ), example for most of the functions.
Modularity	Yes, OOP.
Plugin support	Plugins supported (as for Ext Js). No plugins found for Sencha Touch.
Quality of the code	jslint: stopped at 10%. Uncompressed code not properly programmed. Naming not nice (*ename*) and using *eval*. No new global variable. Namespace: *Ext.**.
Size of the community	Over 180'000 users for Sencha totally. 16'500 posts for Sencha Touch but for Ext JS.
Quality of the community	Relatively good. Fast and helpful answers. Ext JS developers and team members are active in the forum.
Maturity of the framework	Version 1.0.1a. No public version repository (download via email link). Older versions exist, but not easy accessible.
Frequency of public updates and releases	About each month or more often.
Commercial support	Investors for Sencha: Sequoia Capital, Radar Partners. Sencha offers paid support, code reviews and more.
Addition	Sencha Touch changes the overall scale of their interfaces on the fly. Buttons always have the perfect tap area, regardless of screen size or resolution. Fix position elements are possible;

Table A.3: webapp.net Properties

Criteria	webapp.net
Price / Licence	Open source / BSD License
Code size	9 KB
Future oriented	No release plan. Only one developer, Chris Apers. No continuous update cycle.
Compatibility	Name space conflicts prevented with own name space. Applies conventional development methods.
Security	No
Cross browser compatibility	All Webkit based browsers supported.
Cross platform compatibility	iOS, Android, webOS.
AJAX	Yes, but only basically.
Event handling	Yes, binding of events possible. Orientationchange and click event (tap). No gesture support.
Device hardware access	No support.
Animations / Behaviour	Pagetransition: slide. One page behaviour. One page behaviour.
W3C validity	mobileOk: webapp.net-demo 0% (a resource not found)
CSS3 / HTML5	Only CSS3.
Native application imitation	Yes, mimic IPhone and IPod Touch.
Fast learning curve	Yes, one has only to learn and follow the naming conventions. No tutorial.
Code style	No structure and conventions for the JavaScript code. Namings are ok. The few core functions are rarely or bad documented and not intuitive. It is basically a UI driven and HTML based framework.
Debugging	Uncompressed source code available. No debugging tools.
Documentation quality	Not finished, relatively small and hard to understand.
Modularity	Not OOP.
Plugin support	Possible, but not leveraged. No plugins available.
Quality of the code	jslint: stopped after 2%. Getter and setter (ES5: ECMAScript 5[20]) can not be scanned. 8 FIXMEs. More than ten new global variables. Namespace: *WA.** or *WebApp.**.
Size of the community	Small, 650 users in the forum. 2050 posts. 502 topics.
Quality of the community	More or less fast and good answers. Chris Apers is active in the forum.
Maturity of the framework	Version 0.5.2. No public version repository (direct download newest version). Release information are helpful.
Frequency of public updates and releases	Not continuous (Last release 0.5.2 was in february 2010!).
Commercial support	No
Addition	Small UI driven client side framework. Full screen mode of iPhone and iPod Touch. Is i-related. No good jsonp support. Smooth transitions and header animation.

Table A.4: Wink Toolkit Properties

Criteria	Wink Toolkit
Price / Licence	Open source / Simple BSD licence
Code size	Only *base.js* 6 KB
Future oriented	Begin of 2011 new release. Merging plan with dojox mobile and embedJS exists. Update cycle is continuous. Sylvain Lalande and Jerome Giraud are the core development team.
Compatibility	Name space conflicts prevented with own name space and one can use *wink.byId* instead of *$*. Applies conventional development methods.
Security	No
Cross browser compatibility	All Webkit based browsers supported.
Cross platform compatibility	Depending on components. Mainly iOS, Android 1.0.-2.1. and Blackberry.
AJAX	Yes, basic support with *xhr.js*.
Event handling	Event binding and management with wink.publish & wink.subscribe. Start, move, end, gesturestart, gesturemove and gestureend. Gestures are supported.
Device hardware access	Geolocation and localStorage supported.
Animations / Behaviour	Pagetransition: slide, slide cover, slide reveal, 3D cube, flippage and other 2D/3D animations. One page behaviour.
W3C validity	mobileOK: documentation of winktoolkit 63%.
CSS3 / HTML5	Two HTML5 features (video and audio).
Native application imitation	No, lack of style sheets.
Fast learning curve	One only has to learn the intuitive function interfaces. Tutorials are available.
Code style	Well defined source code structure with separated files. Intuitive conventions and namings. Well-defined function interfaces.
Debugging	Uncompressed source code available. Debugging possible with wink.error.log.
Documentation quality	Very well structured documentation with examples. Also provided in xml!
Modularity	Yes, OOP. Source code split in single modules in separated JavaScript files.
Plugin support	Yes. No plugins available.
Quality of the code	jslint: (base.js) stopped after 10%. Only few not severe errors. Misuse of the reserved word *arguments*. No new global variable. After 2 simple code corrections 100% scanned with 5 new global variables. Namespace: *wink.**.
Size of the community	Small, 88 posts.
Quality of the community	Core developers are active in the forum. Only few questions but good answers and comments.
Maturity of the framework	Version 1.2.2. A public version repository exists.
Frequency of public updates and releases	More often than each month.
Commercial support	Dojo Foundation
Addition	Merge regularly with embedJS into the dojo toolkit. Wink Toolkit is experimental & lean, embedJS most complete, dojo mobile most stable for dojo users (http://www.slideshare.net/dylanks/html5-codecamptoolkits); Wink Toolkit has no basic CSS support → UI not supported.

Table A.5: Jo HTML5 Web Apps Properties

Criteria	JoHTML5
Price / Licence	Open source / OpenBSD
Code size	11 KB
Future oriented	A road map exists. Update cycle is continuous. Only one developer, Dave Balmer.
Compatibility	Yes, claims to be compatible with many other JavaScript frameworks inclusive PhoneGap. Does not prevent name space conflicts because it has no own name space. Does not apply conventional development methods. Markup, content and behaviour are all programmed in JavaScript and own HTML tags are introduced.
Security	No
Cross browser compatibility	Safari and Chrome supported.
Cross platform compatibility	WebOS, iOS, Android, Symbian, Dashboard Widgets.
AJAX	No, under construction (joDataSource).
Event handling	Yes, binding events is possible (joSubject). Touch selection (selectEvent in joControl). Gestures not yet ready (joGesture).
Device hardware access	No, only on top of PhoneGap.
Animations / Behaviour	Pagetransitions: rotation. One page behaviour. One page behaviour. Pages in separated JavaScript files, all loaded at the beginning.
W3C validity	mobileOK: kitchensink 34% (resource not found. With source correction 65%).
CSS3 / HTML5	Yes, CSS3 & uses HTML5 as a development stack.
Native application imitation	Mimic parts of the iPhone layout.
Fast learning curve	One has to learn the complex code structure, functions and the framework specific HTML tags (the documentation is not easy to understand). One has to program both the behaviour and markup in JavaScript. Styling with CSS. No tutorial available.
Code style	The code separation is well-defined with different files but the code structure itself is not intuitive. The namings are clear as they start with jo*. The interfaces of the core functions are not well-defined by the documentation.
Debugging	Uncompressed source code available. No debugging tools.
Documentation quality	Not all available functions are documented and the documentation is in working process. The examples are not very meaningful.
Modularity	Yes, basic OOP structure.
Plugin support	Possible, but not leveraged. No plugins available.
Quality of the code	jslint: 100% scanned. No severe errors. More than ten new global variables. No namespace.
Size of the community	Very small, $\sim$ 49 users in the forum and less than 500 posts.
Quality of the community	Fast and helpful answers by Dave Balmer.
Maturity of the framework	Version 0.3.0. Public version repository on gitHub.
Frequency of public updates and releases	$\sim$weekly
Commercial support	No
Addition	Supports YQL. Uses own tags. Hence no direct browser optimisation but either no conflict with included HTML code styled with CSS.

B

Reference Implementation

Details about the reference implementation can be found in this appendix.

B.1 HTML

The `index.html` file starts with the following definitions:

```
<!-- HTML5 doctype -->
<!DOCTYPE HTML>

<!-- Application cache definition -->
<html lang="en-US" manifest="manifest.php">
```

The first tag defines the `index.html` file as a HTML5 document. The second tag defines the primary language for this web page as American English and provides the relative path of the offline application cache manifest file.
Additionally, the index file defines various `meta` and `link` tags for Apple devices [7], e.g. allowing to run the web app like a native app.

The pages of the web app are implemented with jQuery Mobile's definition e.g. `<div data-role="page" id="home">...</div>`. jQuery Mobile allows easy to use references like `<a href="#home">...</a>` which result in soft transitions between the pages.
The content of the pages is taken from the native *Helvetia Notfall-App* application.

B.2 JavaScript

All used JavaScripts are explained in the following, starting with the remotely loaded JavaScript.

http://maps.google.com/maps/api/js?sensor=true This JavaScript is used to interact with the map API of Google. The `sensor` parameter defines whether the device uses a sensor like GPS to detect the location [24]

jquery-1.5.2.min.js This JavaScript includes the minified jQuery JavaScript library in version 1.5.2. It provides Document Model (DOM) manipulation, document traversing, event handling, animating, and Ajax interactions for rapid web development[35].

yqlgeo.js YQL Geo Library [27] provides many geolocation services including location detection based on the IP address. This is used as fallback in the reference implementation if the browser service `navigator.geolocation` defined in the W3C geolocation API is not available.

jquery.mobile-1.0a4.1.min.js This JavaScript includes the minified jQuery Mobile JavaScript library in version 1.0 Alpha 4.1. It provides a unified user interface system across all popular mobile device platforms [36] and is built on jQuery. It is furthermore built with progressive enhancement.

jquery-ui.min.js This JavaScript includes the minified jQuery UI JavaScript library which is needed for the jQuery UI Map library used in the reference implementation.

jquery.ui.map.min.js This JavaScript includes the minified jQuery UI Map plugin for jQuery UI and jQuery Mobile. It alleviates the interaction with the Google Map API. It is only injected if the client device is online. Otherwise the plugin would throw an error at offline usage.

```
if(navigator.onLine){
  $("head").append("<script src='js/jquery.ui.map.min.js'
  type='text/javascript'></script>");
}
```

helvetia.js This JavaScript is project specific and uses an own name space (`helvetia`) to prevent variable name conflicts. It consists of three parts. The first part provides contact and card information handlers. The second part provides map handlers. Finally the third part loads the contact and card information, injects the jQuery UI Map plugin in online mode and binds methods to events. It is loaded as soon as the document is ready.

Most of the mentioned JavaScript code concern the map. A basic example of using maps for mobile devices can be found on `http://jquery-ui-map.googlecode.com/svn/trunk/demos/basic-example.html`.

B.3 CSS

All used CSS files are explained in the following.

jquery.mobile-1.0a4.1.min.css This style sheet includes the minified jQuery Mobile CSS definitions in version 1.0 Alpha 4.1.

styles.css This style sheet is imported from the native *Helvetia Notfall-App* application. It provides useful style definitions for the content to mimic the native application.

helvetia.css This style sheet is project specific and is split in four parts. The first is iOS specific. The second overwrites jQuery Mobile style definitions. The third provides style definitions for newly introduced classes. The fourth provides style definitions for newly introduced ids.
To disable the context menu popping up at tap holding on the lamp picture one needs to define the iOS specific definitions `-webkit-user-select: none;` and `-webkit-touch-callout: none;`. The first one disables to select the element whereas the second one disables the callout sheet.

B.4 Images

The images used in the reference implementation are mainly imported from the native *Helvetia* app. The remaining are generated manually.

C

Usability Test Sheet

Usability test MOCUSI

0' Welcome to the usability test of MOCUSI

1' Introduction questions

1. To which age group do you belong to?

   - ☐ Teeny 10-20
   - ☐ Young 20-30
   - ☐ Middle age 30-50
   - ☐ Senior 50-...

2. What is your profession?

3. Did you ever use touch phones?

   - ☐ Yes: for how long? which one?
   - ☐ No: Another mobile phone?

4. Do you know what bookmarks are?

   - ☐ Yes: Describe?
   - ☐ No. Explain!

5. Do you use your mobile phone to browse websites?

   □ Yes: What bothers you most?

   □ No

6. Language of the test?

   □ English

   □ German

7. Is it okay for you if your name is mentioned in the thesis?

   □ Yes

   □ No

3' Explanation:

During the tasks you should think aloud. Say what you think. Be open and above board. There are no wrong statements and you will not hurt anyone with your statements. I will not help you solving the tasks. Try to solve the tasks yourself.

4' Introduction test cases

1. Tasks (one for each version):

   (a) □ MOCUSI □ Normal –> Find the name and height of the highest point of the Antarctica (Vinson Massif, 4'897m)!

   (b) □ MOCUSI □ Normal –> Find the name and height of the highest point of Africa (Mt. Kilimanjaro, Tanzania, 5'895m)!

2. Questions:

   (a) Which version you would use?

   (b) Why?

   (c) Name the 3 most important differences you observed!

9' Explain: MOCUSI has been developed to enhance usability of websites on mobile devices!

10' Question: What do you think MOCUSI looks like and what will it provide?

12' Show home screen of MOCUSI

13' Question: What's your opinion regarding the home screen?

What do you now think MOCUSI provides?

15' Task: Now you have 5 minutes to play with MOCUSI. Remember to think aloud and to pronounce what you intend to do next and what you think about the behaviour of MOCUSI!

21' Test cases to be completed

1. (4') Bookmark www.namics.com! Assure fast loading and name the bookmark "Namics".
2. (4') Bookmark www.namics.com without settings (bookmark only) and name it meaningful.
3. (2') View the with fast namics site –> go back to the MOCUSI home screen.
4. (4') Bookmark www.kneedeep.ch site with any view settings you like!
5. (4') Bookmark www.kneedeep.ch enabling images only and a fontsize of 10.
   (a) (4') Bookmark worldatlas.com with the style sheet worldatlas enabled and the two JavaScript files jquery and worldatlas enabled (for advanced technicians only).
6. (2') Find the detail information about the site.
7. (2') You have a problem with MOCUSI, who will you contact and how?
8. (2') Delete the second bookmark!
9. (2') Change the name of the first bookmark!
10. (2') Delete all bookmarks!
11. (2') Go to the google website! How do you go to this site?

51' Follow up questions

Overall impression

1. What is your overall impression of MOCUSI from 1-10 (1:not usable, 10:usable very well)?
2. How did it feel to use MOCUSI?
3. Which 3 parts did you like most of MOCUSI?
4. Which 3 parts die you like least of MOCUSI?
5. Would you recommend MOCUSI to others? Why / Why not?
6. How would you explain MOCUSI to someone with equal understanding as you have?

Usability

7. Do you think MOCUSI is useful?
8. Is MOCUSI valuable to your daily browsing?
9. Would MOCUSI enhance your mobile browsing feeling? How?
10. Will you use MOCUSI in the future?
    (a) If no, what needs to be improved before you use it
    (b) If yes, how frequent
11. Would you visit Facebook or your mail account with MOCUSI?
12. Would you do online-banking with MOCUSI?

User friendliness

13. Was it unnecessary complex to use? Which parts?
14. Was it easy to control?
15. What was surprising or unexpected?
16. Do you think you know how to use MOCUSI now?
17. Will others understand MOCUSI?
18. Is additional support, information or description needed?

19. Is MOCUSI clearly structured?
20. Are the wordings chosen well?
21. Does the name MOCUSI suit to the functionality of it?

Advice / Improvements

22. Please give 2 pieces of advise to improve MOCUSI!
23. Other questions or comments?

61' Thank you for your participation!

Bibliography

[1] Helvetia unterstützt sie auch in Notsituationen, 2010. http://www.helvetia.ch/service/iphone-app.htm.

[2] Sass, 2010. http://sass-lang.com/.

[3] W3c mobileok checker, 2010. http://validator.w3.org/mobile/.

[4] wink toolkit (webapp innovation kit), 2010. http://www.winktoolkit.org/.

[5] Tomi Ahonen. *Mobile as 7th of the Mass Media.* Futuretext, October 2008.

[6] Tomi Ahonen. Mobile usability, February 2011. http://communities-dominate.blogs.com/brands/2011/02/all-the-numbers-all-the-facts-on-mobile-the-trillion-dollar-industry-why-is-google-saying-put-your-b.html.

[7] Apple. Configuring web applications, 2011. http://developer.apple.com/library/safari/#documentation/appleapplications/reference/SafariWebContent/ConfiguringWebApplications/ConfiguringWebApplications.html.

[8] C. Armbrüster, F. Voss, M. Neul, and K. Horst. Usability monitor 2010. Technical report, syzygy, 2010. http://know.namics.com/download/attachments/25788643/UM10_Mobile-Web.pdf?version=1&modificationDate=1275634426850.

[9] Dave Balmer. Jo html5 mobile app framework, 2010. http://joapp.com/.

[10] Patrick Baudisch, Xing Xie, Chong Wang, and Wei ying Ma. Collapse-to-zoom: Viewing web pages on small screen devices by interactively removing irrelevant content, 2004.

[11] Gregor Rayma Bernhard Lahres. *Objektorientierte Programmierung.* Galileo Computing, 2 edition, 2009.

[12] Harini Bharadvaj, Anupam Joshi, and Sansanee Auephanwiriyakul. An active transcoding proxy to support mobile web access. In *In Proceedings of the 17th IEEE Symposium on Reliable Distributed Systems*, pages 118–123, 1998.

[13] Timothy Bickmore, Andreas Girgensohn, and Joseph W. Sullivan. Web page filtering and re-authoring for mobile users. *The Computer Journal*, 42(6):534–546, 1999.

[14] George Buchanan, Sarah Farrant, Matt Jones, Harold Thimbleby, Gary Marsden, and Michael Pazzani. Improving mobile internet usability. In *Proceedings of the 10th international conference on World Wide Web*, WWW '01, pages 673–680, New York, NY, USA, 2001. ACM.

[15] Orkut Buyukkokten, Oliver Kaljuvee, Hector Garcia-Molina, Andreas Paepcke, and Terry Winograd. Efficient web browsing on handheld devices using page and form summarization. *ACM Trans. Inf. Syst.*, 20:82–115, January 2002.

[16] Canalys. Android increases smart phone market leadership with 35% share, May 2011. `http://www.canalys.com/pr/2011/r2011051.html`.

[17] Ben Tudor Christy Pettey. Gartner says touchscreen mobile device sales will grow 97 percent in 2010, March 2010. `http://www.gartner.com/it/page.jsp?id=1313415`.

[18] Douglas Crockford. Jslint, 2002. `http://www.jslint.com/`.

[19] Péter Cserkúti, Zoltán Szabó, and János Pál. Smartweb - web content adaptation for mobile devices. -, 2008.

[20] Ecma International. *ECMAScript Language Specification*, 5. edition, 12 2009. `http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-262.pdf`.

[21] DAVID FELDMAN. Comparing mobile web (html5) frameworks: Sencha touch, jquery mobile, jqtouch, titanium, January 2011. `http://interfacethis.com/2011/adventures-in-html5-part-one/`.

[22] Brian Fling. *Mobile Design and Development*. O'Reilly Media, August 2009.

[23] Jesse James Garrett. Ajax: A new approach to web applications, 2. 2005. `http://www.adaptivepath.com/ideas/essays/archives/000385.php`.

[24] Google. Google maps api-familie, 2011. `http://code.google.com/intl/de/apis/maps/index.html`.

[25] Google. Google mobile proxy, 2011. http://www.google.de/gwt/n.

[26] Duncan Haughey. Smart goals, 2010. http://www.projectsmart.co.uk/smart-goals.html.

[27] Christian Heilmann. Yql geo library, 2010. http://isithackday.com/geo/yql-geo-library.

[28] Lieberman Henry, Paternò Fabio, and Wulf Volker. *End User Development*, volume 9 of *Human-Computer Interaction Series*. Springer, 2006.

[29] Shawn Lawton Henry. Wai site usability testing questions, October 2003. http://www.w3.org/WAI/EO/Drafts/UCD/questions.html.

[30] Shawn Lawton Henry. Wai-aria overview, 2010. http://www.w3.org/WAI/intro/aria.

[31] Vicky Homan. Getting mobile-ready part 1: Creating a mobile-optimized website, March 2011. http://googlemobileads.blogspot.com/2011/03/getting-mobile-ready-part-1-creating.html.

[32] Apple Inc. ios human interface guidelines, 2011. http://developer.apple.com/library/ios/#documentation/userexperience/conceptual/mobilehig/Introduction/Introduction.html.

[33] Sencha Inc. Sencha touch 1.0, 2010. http://www.sencha.com/products/touch/.

[34] Gerhard Janisch. Analyse von rich internet application frameworks am beispiel einer thesaurusverwaltung. Master's thesis, Fachhochschule Hagenburg, Österreich, Juni 2008. http://staff.fh-hagenberg.at/kurschl/pubs/advisedDT/Janisch.Gerhard.2008.pdf.

[35] The jQuery Project. Jquery - write less, do more., 2010. http://jquery.com/.

[36] The jQuery Project. Jquery mobile, 2010. http://jquerymobile.com/.

[37] André Langer. Evaluierung von ajax-basierten frameworks für das web 2.0. Master's thesis, Technische Universität Chemnitz Deutschland, March 2007. http://www.qucosa.de/fileadmin/data/qucosa/documents/5366/data/Studienarbeit_Ajax_AndreLanger.pdf.

[38] Johan Sall Larsson. jquery-ui-map, 2011. http://code.google.com/p/jquery-ui-map/.

[39] Max Planck Digital Library. Java script framework, 2009. http://colab.mpdl.mpg.de/mediawiki/Java_Script_Framework.

[40] Kevin Liew. 9 mobile framework to kick start your mobile development career, February 2011. http://www.queness.com/post/6706/9-mobile-framework-to-kick-start-your-mobile-development-career.

[41] Guido Mühlwitz. Javascript goes mobile: Überblick zu den wichtigsten webapp-frameworks, October 2010. http://t3n.de/news/javascript-mobile-uberblick-wichtigsten-282544/.

[42] Natasa Milic-Frayling and Ralph Sommerer. Smartview: Enhanced document viewer for mobile devices. Technical report, Microsoft Research, November 2002.

[43] mobiThinking. Designing usable pocket rockets: A three step guide to usability on the mobile web. Technical report, dotMobi, 2008. http://mobithinking.com/sites/mobithinking.com/files/dotMobi_Mobile_Usability_Best_Practice.pdf.

[44] mobiThinking. Global mobile statistics 2011: all quality mobile marketing research, mobile web stats, subscribers, ad revenue, usage, trends..., February 2011. http://mobithinking.com/stats-corner/global-mobile-statistics-2011-all-quality-mobile-marketing-research-mobile-web-stats-su.

[45] mTLD Top Level Domain Ltd. Deviceatlas, mobile device intelligence, 2011. http://deviceatlas.com/.

[46] Jakob Nielsen. Mobile usability, July 2009. http://www.useit.com/alertbox/mobile-usability.html.

[47] Alexander Olaru. Selection criteria for javascript frameworks, 2007. http://www.infoq.com/news/2007/12/choosing-javascript-frameworks.

[48] Luca Passani. Wurfl the wireless universal resource file, 2011. http://wurfl.sourceforge.net/.

[49] Fabio Paternò and Giuseppe Zichittella. Desktop-to-mobile web adaptation through customizable two-dimensional semantic redesign. In *Proceedings of the Third international conference on Human-centred software engineering*, HCSE'10, pages 79–94, Berlin, Heidelberg, 2010. Springer-Verlag.

[50] Han R., Bhagwat P., LaMaire R., Mummert T., Perret V., and Rubas J. Dynamic adaptation in an image transcoding proxy for mobile web browsing. In *Personal Communications, IEEE*, 5 Issue:6, pages 8–17, IBM Thomas J. Watson Res. Center, Yorktown Heights, NY, 1998. IEEE.

[51] Reuters. Internet most popular information source: poll, June 2009. http://www.reuters.com/article/2009/06/17/us-media-internet-life-idUSTRE55G4XA20090617.

[52] Virpi Roto and Anne Kaikkonen. A.: Perception of narrow web pages on a mobile phone. In *Proceedings of Human Factors in Telecommunications*, 2003.

[53] Inc. Skweezer. Skweezer, 2011. http://www.skweezer.com/.

[54] Steven Snell. Mobile web design trends for 2009, Janaury 2009. http://www.smashingmagazine.com/2009/01/13/mobile-web-design-trends-2009/.

[55] Jonathan Stark. *Building Android Apps with HTML, CSS, and JavaScript*. O'Reilly Media, September 2010. http://ofps.oreilly.com/titles/9781449383268/chapOfflineApplicationCache.html.

[56] Dionysios G. Synodinos. Virtual panel: The state of the art in mobile web application development, August 2010. http://www.infoq.com/articles/mobile-web-development.

[57] Keishi Tajima and Kaori Ohnishi. Browsing large html tables on small screens. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*, UIST '08, pages 259–268, New York, NY, USA, 2008. ACM.

[58] TECHNOticles. Javascript framework comparison, 2010. http://technoticles.com/2010/01/15/javascript-framework-comparison/.

[59] Jonathan Trevor, David M. Hilbert, and Bill N. Schilit. From desktop to phonetop: A ui for web interaction on very small devices. In *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology (UIST2001*, pages 121–130, 2001.

[60] Userfocus. Usability test plan toolkit, February 2010. http://www.userfocus.co.uk/articles/testplan.html.

[61] W3C. Mobile web best practices (mwbp) flipcards, February 2007. http://www.w3.org/2007/02/mwbp_flip_cards.

[62] W3C. Mobile web best practices 1.0, 2008. http://www.w3.org/TR/mobile-bp/.

[63] Johannes Waibel. Always on. always carried. Presentation, November 2010. http://blog.namics.com/2010/SIC_MobileTrends_2010-11-25_BLOG.pdf.

[64] WebApp.Net. Webapp.net, 2010. http://webapp-net.com/.

[65] Webcredible. 7 usability guidelines for websites on mobile devices, March 2011. http://www.webcredible.co.uk/user-friendly-resources/web-usability/mobile-guidelines.shtml.

[66] Xinyi Yin and Wee Sun Lee. Using link analysis to improve layout on mobile devices. In *Proceedings of the 13th international conference on World Wide Web*, WWW '04, pages 338–344, New York, NY, USA, 2004. ACM.