Cross platform applications with HTML5

Hampus Svedestedt

February 10, 2013
Master's Thesis in Interaction Technology and Design, 30 credits
Supervisor at TFE-UmU: Kalle Prorok
Examiner: Håkan Gullikson

Umeå University
Department of Applied Physics and Electronics
SE-901 87 UMEÅ
SWEDEN

Abstract

This thesis was made for Cybercom in Östersund. They wanted an evaluation of HTML5 cross platform applications which studied HTML5 features as well as JavaScript libraries and frameworks. The resources put into mobile application development can be reduced by making applications that can work on all platforms instead of only natively. These types of applications are called cross platform applications and can be developed with specific tools. One way to develop cross platform applications is by using HTML5, which can either be used as a web application or packed into native applications using plug-ins. The purpose of this thesis was to create a mobile web app that can save maps to be used offline. The work was done by evaluating frameworks for web applications. Frameworks that provide user interface elements and features similar to those in native applications, and libraries that render maps served by map servers. Development environments for web development were also tested and evaluated. The results of the research and development were documented experience, and a HTML5 application that shows a map, has GPS functionality and can be used offline.

Contents

List of Figures

Chapter 1

Introduction

A big problem with developing applications for mobile devices is platform fragmentation[58]. That means that there are many different mobile platforms (Android, iOS, Windows Phone and more) that are further divided by the different versions available[1][17]. Users with older hardware are left without support and updates as newer devices are put out on the market[29]. Most if not all of these platforms are based on differing programming languages and frameworks such as Objective-C for iOS, JAVA converted by Dalvik for Android and .NET for Windows Mobile. This means that the developer has the choice between limiting the solution and only aim for a minor part of the spectra or to develop for more platforms to reach as many users as possible. To maximize the amount of possible users, the developer has to create an application for each platform and make sure that they are backwards compatible so that users with older devices can use them. Because of the fragmentation some companies have turned to cross-platform solutions that work on several platforms with a minimum of adaptation[31][47]; this is done to save time and resources on development. One way to create cross-platform applications is to use HTML5 and create a mobile adapted web application that can be saved to device and used as a normal mobile application. The purpose of this thesis was to analyze the implementation of HTML5 support in the standard browsers of mobile devices, find and compare JavaScript frameworks that help development and test developer tools. This is done in order to gain information for a in-company knowledge database of web app applications. After a knowledge base was acquired, the HTML5 support on mobile devices was to be tested. The purpose of this was to see if it is mature enough that a full scale application in HTML5 can work as well on multiple platforms as an application developed for a single platform does natively. This project was done at Cybercom Group AB in Östersund. Cybercom Group is a global consultancy that has 19 offices in eight countries around the world and delivers business solutions, primarily in telecom management, connected devices, Internet services and security.

Chapter 2

Problem Description

2.1 Problem Statement

The problem was to to create a prototype for a lightweight application. This application was to be able to handle online maps and to cache map tiles locally so that a user could navigate a selected area without network connection. The prototype and study was meant to build an in-office knowledge base for HTML5 web application development. This was sought after as there have not been any major development in HTML5 thus far in the Cybercom Östersund office. The subject of this thesis is a study of the current state of HTML5 support on different platforms, frameworks to help application development, libraries for map handling and tools to streamline development. If the prototype could not be completed due to limitations in the HTML5 implementation then the reasons for why, possible workarounds and future possibilities was to be noted and studied. As the Östersund office of Cybercom is currently developing a online type of community that is based around maps, this prototype can be seen as a step towards a mobile version of their system.

2.2 Goals

The main goal was to develop a streamlined HTML5 application that acts like a native application. If this goal cannot be reached then as much as possible of the functionality should be implemented. Beyond the main goal, secondary goals are to test, implement and evaluate:

- Frameworks that provide graphical design and simplify the development
- Different methods of map handling (online/offline)
- Solutions for offline storage (database/cache)
- Scalability in form of resource handling, version handling and screen size

The last goal is to test developing tools/environments that can streamline the development.

2.3 Purposes

The purpose of the work is for Cybercom to get a knowledge database to base their future application development on. HTML5 is a pretty new technology and there have not been any major development for the technology in the Östersund office of Cybercom. The purpose of this thesis is to state if HTML5 is mature enough for major development. If it is mature enough the thesis can be used as a stepping stone into the technology, giving a starting point and showing how to avoid obstacles. If HTML5 is too immature to work with then the thesis will give information on why that is and what to expect in future implementations.

2.4 Methods

The work in this project was done through a Scrum-method[48] using two week long development periods called sprints. This is a method for managing software projects and development and was used to set up what to do between meetings and the priorities of the tasks.

Chapter 3

Method

This chapter presents the planning work and method (Scrum). The planning and method formed the way the work was laid out and made clear what tasks to do and in what order. Scrum ensured the quality of the project as the priority order was set up to make sure that the most important pieces made it into the finished application while those less important would be implemented if there was time left at the end of the project.

3.1 Planning

The planning phase was a period of time where the goals of the project were set. The roles and other parts of the Scrum setup such as milestones and timespan and sprint length was decided upon and a backlog of tasks was put up.

3.2 Scrum

Scrum is a method for managing software projects and product or application development[48]. It is an iterative and incremental agile software development that has earned a lot of praise and use since it emerged in 1986[56]. It is meant to increase the speed and flexibility of development with a process that is performed by a team of members with varying talents and expertise during overlapping phases working together towards a common goal. Scrum is basically built around roles and sprints where roles decide what your objective is in the larger scheme and sprints are the phases that divide the work. The people working in a scrum project have a backlog of tasks to do, these are ordered by priority and each person gets a list of tasks to do per sprint.

3.2.1 Roles

The Scrum method has roles with differing objectives, there are three roles in total: Product Owner, Scrum Master and Development Team.

Product Owner

The product owner represents the company or people that have put resources into the project and have something to lose if it fails. The product owner is responsible for the project's success and thus needs to present what the stakeholders and customers want. Members of the scrum team should do end-user centered tasks such as user stories, prioritize them and add them to the project backlog. In this project the supervisor at Cybercom was the product owner.

Scrum Master

The scrum master is the person clearing the way for the developers so that they can do their work and deliver at the end of each sprint. This does not mean that this person is the team leader, just that the purpose of the role is to make sure that there are no distractions that may interrupt the work process and make the developers lose their focus. The scrum master also makes sure that the scrum process is used in a correct way. In this project the supervisor at Cybercom was the scrum master.

Development Team

The development team are the people responsible for finishing the tasks of the project backlog and deliver something potentially shippable at the end of each sprint. The developers are 3-9 people with different expertise that do the actual work. They are self organized but may communicate with the project managers. In this project the author of the thesis was the development team.

3.2.2 Backlog

A study was made to create a backlog of user stories to work from during the project.

Project backlog

The project backlog is a list of tasks to do for a project to be finished. It is ordered by the product owner after individual priorities such as “what needs to be done first” and “what is most important to have in the finished product”. The backlog items are often written in story-form such as “the user wants to see a map” and have estimates of the time needed to finish them and their estimated business value. The estimates are so that the product owner has an overview of how much actual work there is left to do.

For this thesis the tasks were ordered in priority ranging from “Needed” to “Only if there is some time left at the end” and the business was the amount of knowledge gained per task.

Sprint backlog

The sprint backlog is a list of tasks that the developers need to finish during a sprint. This backlog is put together at the start of a sprint. The development team will select tasks from the top of the project backlog and ask themselves “can we do this as well?”. This continues until the team members decide that they have enough to do during the sprint. The amount of tasks selected has to be balanced against the amount of work and time that is left for the project. If the development is behind schedule then more tasks needs to be completed.

3.2.3 Sprints

A sprint is a defined period of time where development is taking place during a scrum project. A sprint can last between a day and a month depending on the format, development form and total development time. The sprint is only as long as the set up time limit. If the sprint backlog is not emptied at the end of a sprint the remaining tasks are pushed forward into the following one and the reasons for missing the delivery time are written down as lessons learned. The finished tasks are added to the total sum of finished items and must be in a usable condition even if the project owner decides not to release the product. Before each sprint the scrum team has a meeting where the goals are set up as an estimate of what the team can manage for the sprint. After the sprint there is a retrospective meeting where the progress is evaluated and positive and negative lessons are identified and used to better the work in upcoming sprints. During this thesis the length of the sprints were 2 weeks, the tasks were chosen by the supervisor and the author at the start of a sprint. At the end of every sprint there was a meeting discussing how the work had progressed and what needs to be done differently.

Chapter 4

Requirement studies

This chapter will present the work that went into creating a foundation for the development of the web application. The foundation was based on research and background studies and used to build the project backlog for the Scrum development method.

4.1 JaktAppen

To get something from which to base the work, a requirement study was done. It was mostly done by reading through design documents, looking through code and using an older in-house android application on a mobile device to find inspiration for the work on the web application. The application is called JaktAppen4.1 and was developed in-house as a study for what is possible to do with maps in applications for Android devices. It is an application that is supposed to help hunters out in the forest by giving them a map with updated information about their hunting party and provide them with means of communicating with the other people in the party. It is not finished but a lot of the planned functionality is in place. The JaktAppen application's main view is a map view that uses Google maps and

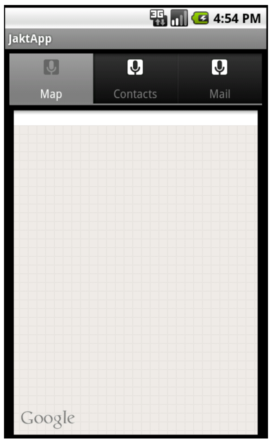


Figure 4.1: The Main screen of the JaktApp Android application.

has GPS functionality, a compass and an indicator showing the information to the user. If there is further functionality in the main view, it could not be accessed as the application crashes as soon as a user interacts with it. The other views are a contactlist supposed to contain the people in the hunting party and a list of the chat messages sent between the users. The last views are to provide means for contacting other users as the application has a focus on communication. The plan was to have a dedicated server to handle the traffic of users communicating and information sent by the application to update the map views with current information about the hunt. The communication was supposed to work through a chat channel for the hunting party. The chat channel would use contacts specific to the application. According to the design documents there were ideas for interesting features going into the application, such as: As a user I

– can watch the selection or the map while offline

– can see the members of the hunting party on the map

– can see the borders of the chase on the map

– can see the dog on the map

– can send and receive messages from other users

– want to call my dog to hear what's going on

– want to share photos and video with other members of the hunting group

– want to register my killing or damaging shots on the map

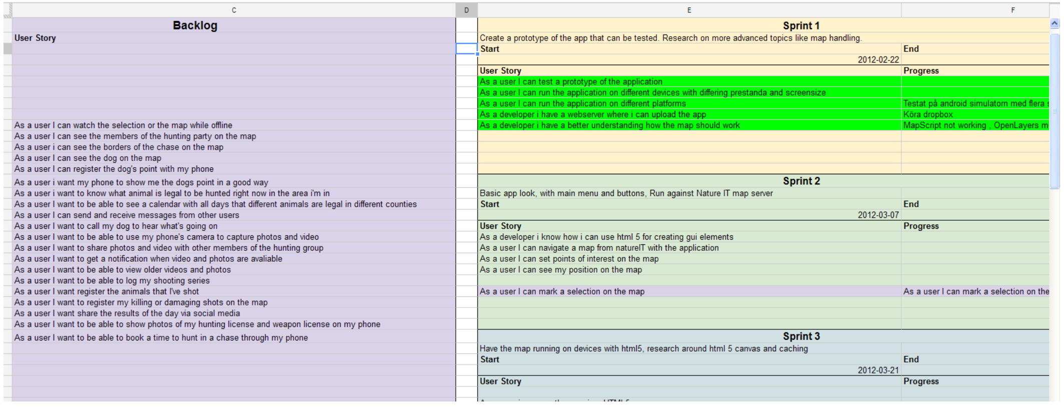
– want to share the results of the day via social media

Not all of these features are in the current version of the application and some of them may not make it into the final version when/if it is finished. The JaktAppen application is currently in development limbo and might be released in the future.

4.2 Building a backlog

The backlog was started with a brainstorming meeting between the author and the supervisor where the goals and basic features were planned out and rewritten as user/developer stories. The features suggested in the design documents and found in the application were also turned into user/developer stories and added to the project's scrum backlog no matter how unrealistic they were.

The tasks of the backlog were ordered in order of priority. The backlog list was ordered in the order that the features needed to be implemented for the application to work as soon as possible. The features that was deemed less necessary for the application to work was given less priority and put lower down on the backlog and the features that was not deemed very necessary at all were put even lower. The latter were features such as "As a user I want to call my dog to hear what's going on" which points to attaching a mobile device to the gun-dog. Lower priority meant that the feature is less likely to be added into the final version of the web application.



Backlog
User Story
As a user I can watch the selection or the map while offline
As a user I can see the members of the hunting party on the map
As a user i can see the borders of the chase on the map
As a user I can see the dog on the map
As a user I can register the dog's point with my phone
As a user i want my phone to show me the dogs point in a good way
As a user i want to know what animal is legal to be hunted right now in the area i'm in
As a user I want to be able to see a calendar with all days that different animals are legal in different counties
As a user I can send and receive messages from other users
As a user I want to call my dog to hear what's going on
As a user I want to be able to use my phone's camera to capture photos and video
As a user I want to share photos and video with other members of the hunting group
As a user I want to get a notification when video and photos are avaliable
As a user I want to be able to view older videos and photos
As a user I want to be able to log my shooting series
As a user I want register the animals that I've shot
As a user I want to register my killing or damaging shots on the map
As a user I want share the results of the day via social media
As a user I want to be able to show photos of my hunting license and weapon license on my phone
As a user I want to be able to book a time to hunt in a chase through my phone

Sprint 1	
Create a prototype of the app that can be tested. Research on more advanced topics like map handling.	
Start	**End**
2012-02-22	
User Story	**Progress**
As a user I can test a prototype of the application	
As a user I can run the application on different devices with differing prestanda and screensize	
As a user I can run the application on different platforms	Testat på android simulatorn med flera
As a developer i have a webserver where i can upload the app	Köra dropbox
As a developer i have a better understanding how the map should work	MapScript not working , OpenLayers m

Sprint 2	
Basic app look, with main menu and buttons, Run against Nature IT map server	
Start	**End**
2012-03-07	
User Story	**Progress**
As a developer i know how i can use html 5 for creating gui elements	
As a user I can navigate a map from natureIT with the application	
As a user I can set points of interest on the map	
As a user I can see my position on the map	
As a user I can mark a selection on the map	As a user I can mark a selection on the

Sprint 3	
Have the map running on devices with html5, research around html 5 canvas and caching	
Start	**End**
2012-03-21	
User Story	**Progress**

Figure 4.2: A screenshot of the Scrum Backlog and the sprints.

4.3 Discussion

JaktAppen is where the idea for this project came from. As the original application was an experiment to see if it was possible to do a functioning map application for Android, this thesis can be seen as the follow-up as it is focused on making a foundation for developing a cross platform map application. The idea of having a mobile application to compliment the map based desktop applications being developed is good but an application built for providing map support for hunters would need some user studies and interviews to find out what features are wanted and how they would be used. A chat client might not be a feature that many hunters would prioritize as they already have working communication systems. The middle of the forest where a lot of hunting is taking place in not an ideal environment for using services that are based on connectivity and access to mobile networks such as chats and dynamic updates. Features such as having the mobile device setting off alarms after picking up the sounds of dogs, which are pointing for prey is intriguing but might be hard to implement. The backlog was used to manage the project and, using the JaktAppen application for inspiration, helped filling it out with tasks to do during development. Ordering the list made sure that the less mature and non achievable tasks of the backlog were pushed back to be looked at if there was time to spare at the end of the development process. The tasks that were vital for the application to work were put in the top of the stack, to be done as soon as previous tasks were finished.

Chapter 5

Designing mobile web applications

This chapter will present a study of the cornerstones of modern web development. The study was made to have a better understanding of the technologies before development started. The study focused on getting an understanding of the basics and finding features that could help development.

5.1 Introduction

With the number of platforms around that all need applications and development specifically geared towards them, cross platform solutions are interesting to developers. This is because the platforms come with their own developing languages and API (Application Programming Interface) that connects their software to their hardware. A native application will only work on the OS it is native to, so one application has to be made for each OS. There is time and resources to save by using a cross platform solution as it only has to be made once and will function on most platforms.

HTML5 can be used as a cross platform technology as a mobile web application basically is a web page which is rendered exactly the same on every browser as long as they follow the HTML standard[36]. HTML5 is under development and scheduled for release in 2014[57] so the version available to users and developers currently is a beta of the final version. This means that there are features coming and going as they push updates to the API. Because of this, web browsers are trailing in development as they have to respond to what happens to the standard[22]. As more people are using the desktop versions of the browsers, they have a higher priority feature wise. That leaves the mobile browsers further behind for feature support. So currently, features that are in the beta API cannot be used in web pages aimed towards desktop browsers as they do not support them and that features that desktop browsers support are not supported by mobile browsers. While developing a web application for mobile use, there is a greater need to make sure that the features used actually work on the intended platforms than when developing for desktop browsers. To add to the fragmentation, mobile browsers developed by different companies are not equally far along in HTML5 API implementation and may even render CSS (Cascading Style Sheets) differently. When developing a native application there is less risk of fragmentation as long as the application is backwards compatible. Another difference between browsers is their

JavaScript engines and some browsers may process JavaScript faster than others[39]. The JavaScript engines can not compete with a native applications support for multi-threading and hardware acceleration so performance heavy applications are better of as native.

As as web application basically is a web page, access to everything a native application can access would be a security problem[55]. That means that web applications can not access hardware such as a camera or files on the device memory as long as they are not bundled as an actual native application.

5.2 HTML5

HTML (Hypertext Mark-up Language) is the language that provides the framework and information for a web page or web application and tells the browser how it should be structured and presented. The language is in constant development and the newest version (5) is still under development and brings a lot of new features. This section will present some history, the features that was considered for use in the development and the current limitations.

```
<!DOCTYPE html>
<html>
        <head>
                <title>My Page</title>
                <link rel="stylesheet" href="mycss.css" />
        </head>
<body>
<div id="page">
        <header>
                <h1>My Title</h1>
        </header><!-- /header -->
        <div id="content">
                <p>Hello World</p>
        </div><!-- /content -->
</div><!-- /page -->
</body>
</html>
```

Code 5.1: A basic HTML page

5.2.1 Background

Tim Berners-Lee invented the web in 1989 to be used as a system to link references in research papers to the referenced paper[19]. To do this he used his knowledge in document and text processing and started developing a Web browser that used hypertext. Hypertext is text displayed on a computer with links to other text that the user can access. His prototype Web browser was ready in 1990. From 1990 to 1993 development was going forward and in 1993 the first proposal for a HTML specification was presented. In 1994 the World Wide Web Consortium (W3C) was founded to try to enforce compatibility and agreement around adopting the HTML standards so that they are rendered similarly[52]. Many proposals for standards were put forward during the 90's and HTML was finally standardized in 1998 with HTML 4.0. HTML was written to not stop rendering when stumbling over an error and the

W3C thought that was a problem so a stricter version XHTML was put forward in January of 2000 as a reformulation using XML. XHTML was harsher and brought error handling that caused a faulty page to stop loading and show an error message. This was something that developers and users thought of as a hassle so they ended up still using the old HTML 4 standard but encapsulated in a XHTML structure. W3C considered themselves finished with HTML 4 and started developing a follow up to XHTML that left HTML 4 behind. During their development a group of W3C developers jumped ship and started developing a follow up to HTML and XHTML that followed HTML 4.0 as that was what the public wanted. In 2006 W3C got involved in developing HTML again and in 2007 they recognized HTML5 as the next standard of HTML. Finally in 2009 they stopped working on their own version of XHTML, XHTML 2. The purpose of HTML5 is to evolve the HTML specification by adding technologies to phase out external libraries and peripherals such as the need of using Flash to present video and also help users by incorporating existing procedures such as <div id="header">and <div id="footer">with tags <header>and <footer>[53][40]. Code 5.1 is an example of a basic HTML page. As the doctype is set to html the page is defined as a HTML5 page. The page includes the new header tag to be used to define were the header information should be placed. This section will present features of HTML5 and evaluate them. As the HTML5 specification is extensive, only features considered interesting for the development are presented. These features were interesting due to being solutions to the goals put forward for the project:

- Geolocation API
- Canvas
- Various solutions for Offline Storage

5.2.2 Geolocation API

Geolocation is an API that lets the web page get hold of the users position. The code 5.2 shows how to set Geolocation up[13]. It has three functions:

- getCurrentPosition
- watchPosition
- clearWatch

With getCurrentPosition you get the current position of the user. In a desktop solution this is usually approximated to the position of the ISPs base station. In a mobile solution the position is received from the GPS of the device and finds the actual position of the user.

The watchPosition function sets a variable that will be called at a set interval with the current position so it can be used to track the users movement. This function has no effect on a desktop solution as the position will not change as long as the user do not connect to a different ISP. On a mobile solution this function is very effective as it can be used to create GPS functionality in a web application. The last function clearWatch just clears the watched variable and stops the GPS. This API was used to add GPS functionality to the application.

```
function getLocation()
{
        //Function for getting the current position
        navigator.geolocation.getCurrentPosition(showPosition);
}
```

```
function showPosition(position)
{
  //Function for showing the current position on the map
  //position has the current position data:
  //position.coords.latitude and position.coords.longitude
}
function scrollMap(position)
{
        //Scrolls the map so that it is always
        //centered at the users position.
        //Position holds:
        //position.coords.latitude, position.coords.longitude
}
var watchId;
function watchLocation()
{
        //Request repeated updates that tracks
        //the user and calls scrollMap on each update.
        var watchId =
                navigator.geolocation.watchPosition(scrollMap);
}
function clearWatchLocation()
{
        //Cancels the position updates
        navigator.geolocation.clearWatch(watchId);
}
```

Code 5.2: An example of Geolocation implementation

5.2.3 Canvas

A canvas is an HTML5 element that contains a bitmap canvas on which graphics can be drawn. It can only be drawn on by accessing the canvas context with JavaScript, the element is just a container of the graphics and has no drawing abilities[9]. The HTML5 canvas can be used for drawing either graphics or images and the context can be turned into data URLs for offline storage. For an image to be turned into a data URL, it either has to be hosted on the same server as the site or the server being cross server linked to has to allow it. This is for security reasons as otherwise it would be very simple to write scripts that download all images from web sites automatically. If the server allows for images to be handled through cross origin they can be made available for modification by setting the image element's crossOrigin variable to “anonymous” before loading the image into it.

```
//Create a new canvas and get the context to draw upon.
var canvas = new Canvas();
var ctx = canvas.getContext('2d');
//Create an image.
var img = new Image();
//Get the url to the image
var imgurl = "example.net/picture.jpg";
//Set the image to handle its content as if it was on
//the same server
img.crossOrigin = "Anonymous";
//Load the image from the url into the Image element
img.src = imgurl;
//When the image is loaded, draw it on the canvas context.
img.onload=function()
{
        ctx.drawImage(img,0,0);
}
```

Code 5.3: How to load an image to a canvas

5.2.4 Offline Storage

As this application was to be able to run in offline mode, parts of it had to be saved locally on the device. As a map with several layers of tiles for different zoom levels takes up a lot of space there had to be a way to select an area and save just those tiles. Something to think about is the file size limit for offline storage. All of the offline storage solutions share a standard 5MB limit on the amount of data that can be saved locally[21]. Some of the technologies can have their individual file size limits increased for users using select web browsers. In such a case the user is asked to allow the increase through a prompt and has to accept for it to work. The limit for WebSQL[10] can be increased to 25MB if the user allows it and Firefox's limit for IndexDB can be extended to 50MB if the user allows it.

Web Storage

Web storage is the combination of localStorage which allows for data to be saved locally and used while the web application is offline and session storage which allows for data to be saved during a session[10]. Both of them are key-value arrays that store strings. They are not databases and thus scale badly as you put a lot of objects into them. Problems that can arise using web storage are that saved objects may be dropped because secure transactions are not implemented. Another problem is that there is no way of knowing how much of the 5MB quota is used. This is especially problematic when using Web Storage coupled with Offline Cache since the cache cuts into the available space. The only way to get an idea of how much free space there is left is to count the objects saved and make an estimate calculation or to fill the remaining space with data of known size until the storage is full to get an approximation.

```
var storage;
function initStorage()
{
        //Test if the browser supports storage
        if(typeof(Storage)!="undefined")
        {
          //The browser supports local Storage
          //so set it to a variable
          storage = window['localStorage'];
        }else
        {
                //The browser does not support local Storage
                alert("Sorry! No web storage support..");
        }
}
```

Code 5.4: How to set up Local Storage

The code 5.4 shows an example of how to set up localStorage for a web page. The check for browser support is to stop the web page from crashing if it does not support localStorage. This can be used to deactivate the parts of the page that uses localStorage to avoid errors. When the web storage is initiated it can be used with functions such as in code 5.5. To be safe it is best to check for browser availability in the functions as well as catching the exceptions that can be thrown from loading to keep the page stable.

```
function saveInStorage(key, value)
{
        //Check that the browser supports localStorage
        //If not skip the function
        if (!window['localStorage']) return;
        //If the browser supports localStorage
        //save the value at the key
        storage.setItem(key, value);
}
```

```
function loadFromStorage(key)
{
        //Check that the browser supports localStorage
        //If not skip the function
        if (!window['localStorage']) return;
        try{
                //If the browser supports localStorage
                //try to get the value with the key
                var tileData = storage.getItem(key);
                return tileData;
        }
        catch(e){
                //If the localStorage has not any value
                //at the wanted key, return null;
                return null;
        }
}
```

Code 5.5: An example of Local Storage implementation

IndexedDB

IndexedDB is a flat-file database that holds objects of any type[14]. It is not a SQL database and the queries are using the NoSQL language. It can store more data than Web Storage and is more scalable than Web Storage. The problem with IndexedDB is that it is still an experimental technology that is barely implemented for desktop browsers and not at all implemented on mobile browsers. Because of that there is no reason to use it for an mobile application.

Application Cache

```
<!DOCTYPE html>
<html manifest="myappcache.cache">
<body>
        //The content of the document
</body>
</html>
```

Code 5.6: A HTML page with an Application Cache manifest

Application cache is a way to cache a homepage by downloading all the files listed on a manifest[11]. It is added to a web page by adding “manifest=” and the name of a cache manifest file to the first html tag as in code 5.6. As shown in code 5.7 the first line of the manifest file has to be “CACHE MANIFEST” or else it will not work. There are three different tags used for the cache manifest: “CACHE”, “FALLBACK” and “NETWORK”. The “CACHE” tag is used to list the files that should be cached but can be skipped as the files can just be listed under “CACHE MANIFEST”. The “FALLBACK” tag is where the files that should be called if an online resource is requested while the site is offline are to be listed. Under the “NETWORK” tag the files that are always going to be requested from a server should be listed. Placing an asterisk under the “NETWORK” tells the browser that all files that are not cached should always be loaded from a server. A thing to have in mind is that Application Cache may work differently on different browsers. Not using putting an asterisk under the “NETWORK” and instead put a link to a folder holding data worked correctly in Google Chrome but did not work at all in Mozilla Firefox. If a homepage is to be cached using offline cache then it needs to be small enough to fit within the 5MB quota, otherwise it cannot cache all the required files. While using JavaScript frameworks such as Dojo with a lot of features and files all script files that are used needs to be cached, otherwise the site will not work as intended when used offline.

```
CACHE MANIFEST
//List the files that should be cached except the main page
mycss.css

NETWORK:
//List the files that should not be cached
*

FALLBACK:
//Fallback files if the page is not available
offline.html
```

Code 5.7: An example of a cache manifest

WebSQL

WebSQL is an HTML5 implementation of a SQLite database handler that can create manipulate webpage specific client side SQL databases[7]. The problem with WebSQL is that W3C has stopped supporting it in support for IndexedDB and Web Storage, has no support in the newest versions of Internet Explorer and Firefox and it will most certainly be phased out in future versions of browsers from other developers. Code snippet 5.8 shows how to setup a a database and execute a query.

```
//Create a function to prepare the database
function prepareDatabase(ready, error)
{
        //Returns a handle to the database
        //If the database do not exist it will be created
        return openDatabase('documents', '1.0',
        'Offline document storage', 5*1024*1024,
        function (db)
        {
                db.changeVersion('', '1.0', function (t)
                {
                //Create a database table
                t.executeSql('CREATE TABLE example (id, name)');
                }, error);
        });
}
//Call the database setup
prepareDatabase(
        function(db)
        {
                //If the database call succeed
                //execute the query
                db.readTransaction(function (t)
                {
                t.executeSql('SELECT COUNT(*) AS c FROM example',
                        [],     function (t, r)
                        {
                                //Handle the results
                        }, function (t, e)
                        {
                                // couldn't read database
                                //Handle errors
                        });
                });
        },
        function (e)
        {
                // error getting database
                //show error message
```

```
            }
);
```

Code 5.8: How to setup WebSQL

5.3 CSS3

CSS (Cascading Style Sheets) is a style sheet used to describe how the information in a HTML or XML document should be presented[8]. CSS3 is the third version of CSS and has been in development since 1998. CSS is designed to be a separation between content and design in a way that improves flexibility, allows for multiple pages to share the same design document. CSS3 brings a lot of new features which enrich the style sheets with things that needed to be done with JavaScript frameworks or different CSS hacks before. This section will present the background of CSS3 and describe features interesting to the project.

5.3.1 Background

As the development of HTML went on there arose a need for more design functionality[37]. The design functionality that existed was bound to the HTML document and had to be added for each element. To have the same design for other pages the design tags had to be transferred over manually. As more design capabilities were added to give more control over site appearance the HTML got more complex and a need emerge, to separate the document styling from the document presentation. Nine different proposals for style sheets was put forward and two were accepted and used as a foundation for what would become CSS. The first CSS specification was finalized in 1996. It took more than three years for browsers to support it fully. Two years later the second level specification was published in 1998 and and an update, level 2.1, published in 2004. The earliest CSS3 draft was published in 1999 and as of July of 2012 four drafts out of fifty have been published as formal recommendations and most of CSS3 is supported by the major browsers. There will not be any CSS4 because CSS3 refers to everything published after 2.1[35]. There are level 4 drafts of CSS features but as of CSS3, the language is split up into individual modules that can be updated individually. This means that the draft levels only refer to the age of the module, not the version of the language as a whole. It would be appropriate to just drop the version number and just refer to the language as CSS. An intriguing feature of CSS3 is that it can be used to phase out Flash and some JavaScript libraries as it has support for movement, animations and user interaction[12].

5.3.2 Media Queries

This is a powerful feature to use for mobile web applications[15]. You can specify specific CSS queries that change the design depending on different features of the sceen. One query can be done towards the screen resolution. With that the size of of HTML elements can be changed depending on the resolution of the screen as in code snippet 5.9. This is useful since there is no standard device screen resolution and the application is supposed to be shown as one page without any scrollbars. Other things that can be queried towards is screen aspect ratio, what type of projection the screen uses or what orientation the screen has and if the screen is in color or not.

```
/* height smaller than 950 */
@media screen (max-height:950px)
```

```
{
        /* if the height is smaller than 950
         set the height of the map element to 900px*/
        #map{
                height:900px;
        }
}
/* height smaller than 800 */
@media screen (max-height:800px)
{
        /* if the height is smaller than 800
         set the height of the map element to 550px*/
        #map{
                height:550px;
        }
}
```

Code 5.9: Two behaviors for different screen sizes

5.4 JavaScript

JavaScript is an objective script language that is most commonly used embedded in or included from HTML pages. It is used to build advanced functionality into web pages such as enhanced user interfaces and dynamic websites.

5.4.1 Background

JavaScript was originally developed for Netscape during their Web browser war with Microsoft as a competitor to Microsoft's Visual Basic which is a code language aimed towards nonprofessional developers[16]. It was shipped under the name LiveScript in beta releases of Netscape Navigator 2.0 in 1995 but was renamed JavaScript for the official release. The name JavaScript caused confusion as it gives the impression that the name is built from the Java programming language. The name has been interpreted as a marketing ploy due to support for Java was being added to the Netscape browser at roughly the same time. JavaScript was a widespread success and Microsoft added support for it the next year but called it JScript due to trademark issues. Since then JavaScript has been used as background code base to go to for dynamic web solutions as the industry standard for client-side scripting. Today, the JavaScript development is handled by the Mozilla foundation on license from Oracle Corporation and an ongoing standardization by Ecma International process makes sure that it is processed similarly by are browsers. The next chapter will present the features of the JavaScript libraries and frameworks that were under consideration for the development part of this project.

5.5 Discussion

This section will present the thought process behind the selection of features used in the development. The choice of including Geolocation API or not was an easy one. If there is a need for GPS functionality it is very simple to plug it in and get coordinates from the API functions, then just plug them into the map engines. The canvas element was used to draw map tiles on for the application. The tiles could have been drawn in image elements but they can not be turned into data URLs without first being loaded into a canvas object, so it was easier to work with canvas elements from the beginning. It was chosen instead of using plain images as the nature of the project was "the more HTML5 experience the better" also the most of map engines drew graphics on canvas elements layered over the map tiles as a standard so mashing them together and drawing everything on the same layer just made sense. During the development it became clear that the map server that was planned for the project would not work. The server did not support cross server image manipulation which made saving map tiles impossible as it blocked turning canvas data into data URL's. A solution for this would be to install a Node.js client made to handle the images directly on the server. This was not possible as the server was setup to deliver images for Cybercom's live web map platform and new code may cause it to be instable. Another possible solution was to download the images with the HTML5 feature Web Workers. But that feature is not supported by Android browsers, so it did not work. The offline solution chosen for the project was to use offline cache for static objects and web storage for dynamic objects. A static file limit was set up to make sure that the Web Storage did not overflow. Overflowing would throw an exception and would cause the tiles to be saved in weird patterns. IndexedDB could not be used as it was not supported for mobile browsers. WebSQL was considered for development but was finally not chosen due to not being supported by W3C. This means that WebSQL eventually will be phased out and end up not working. The only CSS3 feature that was really considered was media queries as the rest of the design was supposed to be built by design pieces brought by UI frameworks. Media queries is very a powerful asset and helps a lot as the web applications are supposed to be run on devices with varying screen sizes. They allow for the application to adapt to different screen sizes so that the users do not have to zoom the page to access the functionality. The JavaScript did not need any examples as it had been featured in the HTML5 section and as it will be featured further in the upcoming chapters.

Chapter 6

Evaluation of libraries and frameworks

This chapter will present the frameworks and libraries that are under consideration for use in the development. The frameworks were chosen due to their functionalities, which were considered useful for reaching the goals of the project.

6.1 Introduction

JavaScript frameworks and libraries are codebases that act as plug-ins of features and provide an easy way to add more advanced functionality to a web page without having to build everything from scratch. The frameworks are often freeware projects that are available for use and further development. There are a lot of libraries created for almost any functionality sought after by a developer. This chapter will present some libraries and frameworks that were considered for use in the development of this project and an evaluation to decide what part to finally include.

6.2 Mapping libraries

As the map was supposed to be the central part of the application it was important to have a good map library which renders the map quickly and takes up a small space on the web page. This is because then it can be stored within the 5MB offline file size limit while leaving space for the frameworks and map tiles. These are the map engines that were considered for this project:

– Tile5

– OpenLayers

– Leaflet.js

6.2.1 Tile5

Tile5 is a map library fully written in HTML5[42]. It is able to take maps from different map providers, style them and present the tiles as canvas layers or WebGL objects. It is fast at rendering, the rendering looks good and the files are not big. This map library is in an alpha stage of development and the documentation is sparse or simply nonexistent. As a help to get anywhere with Tile5 there are extensive examples with a lot of code[43]. The function drawing the map layer in the examples takes maps from a Cloudmade, which is a service provider that among other things provides access to different map engines, and an API-key which is needed for the maps to be rendered[41]. There are no examples where a map-provider besides Cloudmade is used so using a custom map server, which is the intent of this thesis, is plain guesswork. The standard call of creating a map layer (code 6.1) takes a pre-defined map service (generator) and an api key. This means that using a custom map server would mean a re-write of the code.

```
//Create a map within a <div> named mapContainer
map = new T5.Map('mapContainer', {
        padding: 'auto'
});

//Create a layer of tiles upon the map
map.layer('tiles', 'tile', {
        generator: 'osm.cloudmade',
    // demo api key, register for an API key
        //at http://dev.cloudmade.com/
    apikey: //insert api key here
});
```

Code 6.1: A basic Tile5 setup

6.2.2 OpenLayers

OpenLayers is the goto map library when you want a magnitude of tools and support[46]. Basically OpenLayers supports every map provider and most web frameworks have built in support of it. Basically, if a server can provide maps then OpenLayers can render them. It is a stable library that incorporates everything a user and a developer would want out of a map renderer feature wise and documentation wise[44][45]. Everything about it is well documented and easy to get working. Problems with it is that it may be too big to incorporate in a mobile web application with offline support as the whole library needs to go offline. If there is a need to render tiles as canvas elements, a rewrite of some files the current release would be needed. The last problem of OpenLayers is that it need a large amount of resources because of how big it is which can be a problem for mobile devices.

```
var map;
//A function that sets up the map with tiles
//This function should be called when the page is loaded.
function init(){
        //Creates a new map in a <div> named map
        map = new OpenLayers.Map('map');
        //Creates a layer containing the tiles
        var wms = new OpenLayers.Layer.WMS(
                "OpenLayers WMS",
                "http://vmap0.tiles.osgeo.org/wms/vmap0?",
                {layers: 'basic'}
        );
        map.addLayer(wms);
}
```

Code 6.2: A basic OpenLayers setup

6.2.3 Leaflet.js

Leaflet.js is a lightweight mapping library that only does the very basics but does them well[5]. It is small in size and renders maps quickly and beautifully. It has a good, unfinished

API documentation, that is easy to understand and has plenty available plug-ins for added functionality[4]. Problems are that it is not supported by as many systems as OpenLayers is and has not got as many features, it does not support all kinds of projections and that it is still a beta version. The version evaluated for this project does not provide drawing tools but it is probably coming in an upcoming release. It does not render map tiles as canvas elements from scratch but the functionality can be added by overwriting the drawTile function and adding configurations for drawing the images on a canvas[6].

```
//Set up a map in a <div> called map and set settings
//such as projection and allowed zoom levels.
var map = new L.Map('map',{crs: L.CRS.EPSG3857 ,
        attributionControl: false , minZoom: 6, maxZoom: 12});

//Set up a layer for the map that renders the tiles
//on canvas elements
var url =
        "[mapserver]"
var     canvaslayer =
                new L.TileLayer.Canvas(url,
                {maxZoom: 18, scheme: 'xyz'});

map.addLayer(canvaslayer);

canvasTiles.drawTile =
        function(canvas, tilePoint , zoom) {
    var ctx = canvas.getContext('2d');
    //Get the url to the tile image for the current canvas
    var imgurl = "[mapserver]/{0}/{1}/{2}.png".format(
        zoom,tilePoint.x,(tilePoint.y));

        //Create a new image
        var img = new Image();

        //Set the image to use cross origin permissions
    img.crossOrigin = "Anonymous";

        //Write the cross server image to the image
        img.src = imgurl;

        img.onload = function () {
                // draw the tiles on the canvas when the
                //image has loaded
                ctx.drawImage(img,0,0);
        }
}
```

Code 6.3: A basic Leaflet.js setup rendering on canvas elements

6.2.4 Other libraries

Some testing was made on MapsScript[38] and Kothic JS[18] but they did not provide the functionality the project needed. MapScript was more of a server script library and the Kothic map library was more geared towards map styling. After testing, no further development was done with these frameworks.

6.3 Enhanced features and graphical interfaces

This section will present frameworks and a plug-in that was considered for additional functionality and graphical styling. The frameworks and plug-in were chosen for ease of use, the amount of featured UI elements, the look of the graphical UI and size of the files due to the file limit of the offline storage. The ones that were considered for this project were:

– jQuery Mobile

– jQTouch

– Dojo

– Sencha Touch

These frameworks are built to provide a mobile-optimized code base and interface that works and looks the same on an assortment of different mobile platforms. They are mostly optimized around touch functionality with many different widgets, layouts and features that imitate their counter-parts in native applications on mobile platforms. The difference between the frameworks are which code bases they are based on and the scope the developers are aiming for. What they have in common are that they are one page solutions. That means that the whole applications is built around a single HTML page separated into different views as in code: 6.4. This means that the whole application gets loaded at the same time and the secondary views are hidden until called for. Dojo, Sencha and the jQ frameworks have their own codebases and and because of that each has new syntax to learn and get used to. As some of the frameworks are big both content wise and and file size wise there is a need to build a mimimized version of the chosen framework that is specifically adapted to suit the web page. This is done by setting up a shell script that collect the specific JavaScript files used in a web page and packs them into a single minimized file by cutting away spaces and comments.

```
<body>
  <!-- page 1 -->
  <div id="page1" data-dojo-type="dojox.mobile.View">
    <!-- page 1 content -->
  </div>

  <!-- page 2 -->
  <div id="page2" data-dojo-type="dojox.mobile.View">
        <!-- page 2 content -->
  </div>

  <!-- page 3 -->
  <div id="page3" data-dojo-type="dojox.mobile.View">
        <!-- page 3 content -->
  </div>
</body>
```

Code 6.4: A one page setup

6.3.1 jQuery Mobile

JQuery Mobile is a framework built by the jQuery foundation and adapted for mobile development[34]. It is aimed towards making mobile web application development easier for people that are used to using jQuery[33] for web development. The development is focused on optimizing touch control and being supported by as many mobile devices and browsers as possible as. JQuery Mobile uses the HTML5 data attribute to specify how interface elements should be rendered by setting them to the appropriate role. In code snippet 6.5

this is done for example by setting a headers data-role to “header”. After giving an element a role it will automatically have the specified behavior as well as look[32].

```
<body>
<div data-role="page">

        <div data-role="header">
                <h1>My Title</h1>
        </div><!-- /header -->

        <div data-role="content">
                <p>Hello world</p>
        </div><!-- /content -->

</div><!-- /page -->

</body>
</html>
```

Code 6.5: A basic jQuery Mobile page

6.3.2 jQTouch

JQTouch is a plug-in that adds functionality to a web page and adapts it for mobile web development[20]. JQTouch is open source and is aimed towards mimicking mobile platform behavior and being as light weight as possible. JQTouch supports the zepto.js[28] framework as well as JQuery[33] for browsers using WebKit. Zepto is a slimmer framework adapted for WebKit browsers (Safari, Chrome, Safari mobile, android browser). The syntax is similar to jQuery but as it is only aimed towards WebKit[54], it has a 50% smaller file size wise. JQTouch uses the class and id attributes of HTML coupled with CSS as normally done in web development to render the user interface elements. The implementation of a jQTouch page is shown in code 6.6 with the class attribute specifying how to render the UI elements. Setting the class attribute will also bind the element to the appropriate behavior.

```
<body>
        <div id="jqt">
                <div id="home">
                        <div class="toolbar">
                                <h1>Hello World</h1>
                                <a href="#info" class=
                                        "button add slideup">
                                        Settings
                                </a>
             </div>
             <div id="content">
                <p>HELLO</p>
             </div>
                </div>
        <div id="info">
                        <div class="toolbar">
```

```
                                <h1>About</h1>
                <a href="#add" class="cancel">Cancel</a>
            </div>
                </div>
        </div>
</body>
```

Code 6.6: A basic jQTouch Mobile page

6.3.3 Dojo

Dojo is a large web page framework that amongst other things have mobile browser part (dojox)[26]. Dojo is separated into three parts dijit[23], dojo[24] and dojox[27]. Dijit is the widget part of the library[23], dojo is the base library[24] and dojox is a library for graphics and the also the mobile optimized part of the library[27]. For a web application to work the whole library is still needed as the main parts of the dojo library is used to connect everything together. This means that this library will take up a lot of space on the server and will take up a lot of space in the offline storage for it to work while cached. The dojo framework changed their syntax with the last update and their API documentation is not very updated because of that. Another thing with the documentation that can cause problems for a new user is that it is just clean references to the functions and variables with just the plain names and no explanation for how they are used or any example code snippets. A basic Dojo page looks like in code: 6.7 and has custom data attributes to mark where the UI elements should go as called "data-dojo-type"[25]. For Dojo to render the user interface a parsing function needs to be implemented. The parser goes through the code and adds graphical styling to elements with the appropriate attributes. The render call looks like 6.8. If a UI element is wanted then it needs to be included within the require call as well as in the function that starts the parser. If not included in the call the element and its functionality will not be loaded and the page will not work as it should. The standard theme of dojo mobile makes the layout look like if it was a native iPhone application but if queried the renderer will choose a theme for the application that makes it looks like it was made for the currently used platform.

```
<body >
  <div id="mapview" data-dojo-type="dojox.mobile.View">
        <div id="headerdiv" style="height:45px;">
          <h1 data-dojo-type="dojox.mobile.Heading">
                <div data-dojo-type=
                  "dojox.mobile.ToolBarButton"
                        data-dojo-props="label:'Toolbar',
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣onClick:generateToolBar"
                        class="mblDomButton" style=
                        "float:right;">
                </div>
                Example
          </h1>
        </div>
        <div id="content" data-dojo-type="dojox.mobile.View"
                data-dojo-props="selected:␣true" >
        </div>
  </div>
</body>
```

Code 6.7: A basic Dojo Mobile page

```
require([
        /*This calls the files required to
        render the page */
        //the parser that parses the page
        //for it to render
        "dojox/mobile/parser",
        //The overall mobile library
        "dojox/mobile",
        //Needed to render a tab bar
        "dojox/mobile/TabBar",
        //Needed to render a button
        "dojox/mobile/Button",
        //Compatability checker
        "dojox/mobile/compat"
        //Calls the render on page load
        , "dojo/dom-attr"
        , "dojo/domReady!"
        ],
        //Calls the code within the included files
        function(mobileParser, TabBar, dm,  domAttr, compat) {
    // Parse the page!
    mobileParser.parse();

});
```

Code 6.8: The call to render a Dojo page

6.3.4 Sencha Touch

Sencha Touch is a commercial framework with two free versions available[50]. It is built specifically for mobile web applications but the files can be converted to work as a native with tools bundled with its tool kit. Sencha Touch only works for WebKit based browsers and therefore do not support Firefox or Firefox Mobile. This framework is set up with a main HTML page that is empty except for calls to the JavaScript source code. The source code as in 6.9 sets up the page by generating the ui and other graphical elements from the code and showing it in the browser[51]. The code file should also contain all the wanted functionality. This separates the framework from the others as their solutions are focused around the HTML page containing the user interface because Sencha Touch is more based around plain coding than they are. Sencha Touch can be set up to mimic the device specific themes and effects of the currently used device. Sencha has a development tool for developing Sencha web pages and web applications.

```
Ext.define('MyApp.view.MyTabPanel', {
    extend: 'Ext.tab.Panel',
    config: {
        items: [
            {
                xtype: 'container',
                title: 'Tab 1',
                items: [
                    {
                        xtype: 'panel'
                    }
                ]
            },
            {
                xtype: 'container',
                title: 'Tab 2'
            },
            {
                xtype: 'container',
                title: 'Tab 3'
            }
        ]
    }
});
```

Code 6.9: A basic Sencha Touch page

6.4 Discussion

For this project a lot of experimentation was done before finding a final solution to go with for the final development. The map engine selection was the most difficult trial. It begun with MapScript that seemed promising, was hard to start working with and in the end did not have any of the wanted functionality and went on to OpenLayers. OpenLayers is a great mapping library that has everything for a mapping web service. The problem is the size and performance as it is to heavy for a mobile mapping service. A problem with finding a new mapping library was that other libraries lacked the documentation and features that OpenLayers had. A lot of mapping libraries are not well documented so it is difficult to get started. If it takes extensive detective work or trial and error to get something working, it does not matter how great advertised features might be. The Tile5 author acknowledged the problems with his library and documentation. He explicitly told users to look for another library for these things, Leaflet.js. Leaflet.js is everything a mobile web application developer could want. It has a clean design, it is lightweight, it is fast and it is easy to work with. As soon as the testing of Leaflet began it was clear that it was the obvious choice to go with. A problem with Leaflet was the lack of support of specific map projections. This was solved by experimenting with the projections available. The Cybercom mapping server provided tiles in the opposite direction of how the Leaflet rendered them so the tiles were laid out upside down. This was fixed by inverting the renderer. For frameworks the choice was hard. All of them do the same thing but with different syntaxes. If a developer is comfortable with the jQuery syntax then a jQuery based framework is the best to work with as Dojo and Sencha Touch are harder to get started with. If a developer has no prior experience with jQuery then frameworks are equally suitable for development. The author was not used to jQuery so the Dojo and Sencha Touch was more suited for this project. The way to design the GUI in Sencha Touch is different from the other frameworks and can be a reason to not pick it for development. This is because both web development and application development is usually based around using separate files for the graphical parts that users see and handles. Applications for Android and iPhone uses XML formatted interface files that can be edited either through writing the tags manually in a text-editor or with WYSIWYG editors that is included with the developer kits. What the developer favors is left to personal taste but the author preferred having the graphical interface left mostly separate from the code. This led to the development being done with the Dojo framework even though the file size led to a compromise that limited the amount of tiles that could be stored offline. This was because most of the Dojo JavaScript files had to be cached for the page to work without an active connection. Building customized JavaScript libraries with scripts to minimize them was quite complicated. The author did not get it to work the way it was intended which turned into a problem due to the offline storage limit.

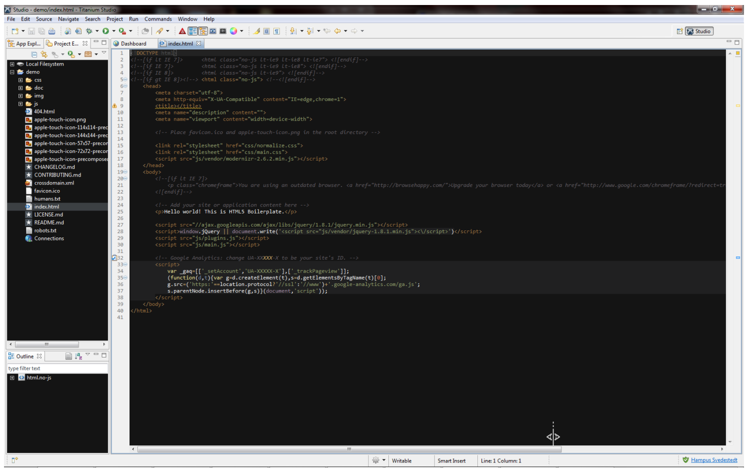
Chapter 7

Tools for development

An environment was needed for development and thus a study was done before it started to find tools suitable for HTML5 development. The author was interested in freeware development solutions before solutions costing money. These were the environments considered for the project:

- Aptana Studio 3
- Titanium Studio
- Sencha Architect

7.1 Titanium Studio/Aptana Studio 3



```
<!DOCTYPE html>
<!--[if lt IE 7]>      <html class="no-js lt-ie9 lt-ie8 lt-ie7"> <![endif]-->
<!--[if IE 7]>         <html class="no-js lt-ie9 lt-ie8"> <![endif]-->
<!--[if IE 8]>         <html class="no-js lt-ie9"> <![endif]-->
<!--[if gt IE 8]><!--> <html class="no-js"> <!--<![endif]-->
    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
        <title></title>
        <meta name="description" content="">
        <meta name="viewport" content="width=device-width">

        <!-- Place favicon.ico and apple-touch-icon.png in the root directory -->

        <link rel="stylesheet" href="css/normalize.css">
        <link rel="stylesheet" href="css/main.css">
        <script src="js/vendor/modernizr-2.6.2.min.js"></script>
    </head>
    <body>
        <!--[if lt IE 7]>
            <p class="chromeframe">You are using an outdated browser. <a href="http://browsehappy.com/">Upgrade your browser today</a> or <a href="http://www.google.com/chromeframe/?redirect=tr
        <![endif]-->

        <!-- Add your site or application content here -->
        <p>Hello world! This is HTML5 Boilerplate.</p>

        <script src="//ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js"></script>
        <script>window.jQuery || document.write('<script src="js/vendor/jquery-1.8.1.min.js"><\/script>')</script>
        <script src="js/plugins.js"></script>
        <script src="js/main.js"></script>

        <!-- Google Analytics: change UA-XXXXX-X to be your site's ID. -->
        <script>
            var _gaq=[['_setAccount','UA-XXXXX-X'],['_trackPageview']];
            (function(d,t){var g=d.createElement(t),s=d.getElementsByTagName(t)[0];
            g.src=('https:'==location.protocol?'//ssl':'//www')+'.google-analytics.com/ga.js';
            s.parentNode.insertBefore(g,s)}(document,'script'));
        </script>
    </body>
</html>
```

Figure 7.1: The Titanium Studio layout

Aptana Studio 3[2] is a web development tool built from the open source project Eclipse which is a large development studio that handles a lot of different languages and is widely used. Aptana has a support for a wide array of web development technologies such as HTML5, CSS3, JavaScript, Ruby, Rails, PHP and Python. Titanium Studio[3] is built

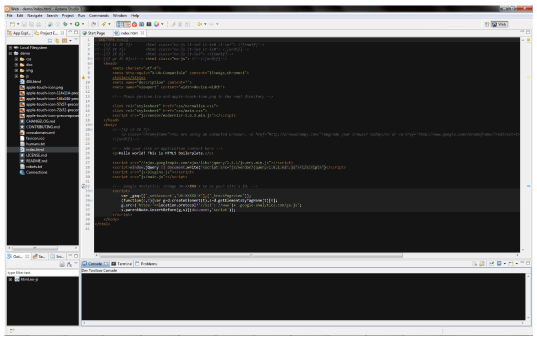


Figure 7.2: The Aptana Studio 3 layout

from Aptana as both are made by Appcelerator. It is basically the same program as Aptana in both functionality and looks (figure 7.2 and figure 7.2) but has extra support for Appcelerator's cross-platform framework that uses JavaScript to build applications. The Appcelerator cross-platform JavaScript code is compiled into code that runs natively on the most popular platforms. The addition of support for the Appcelerator specific features has caused the developers to add a login screen to Titanium Studio which forces the user to register an account on their web page to access the program. This feature is absent in Aptana Studio 3 which makes it the better choice of the two for web applications or web development.

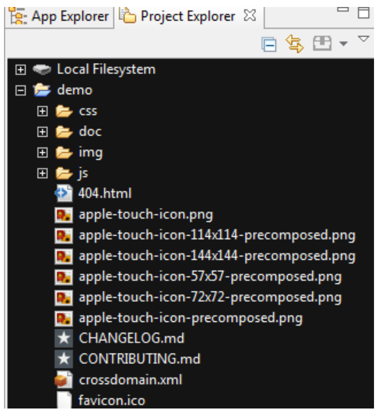
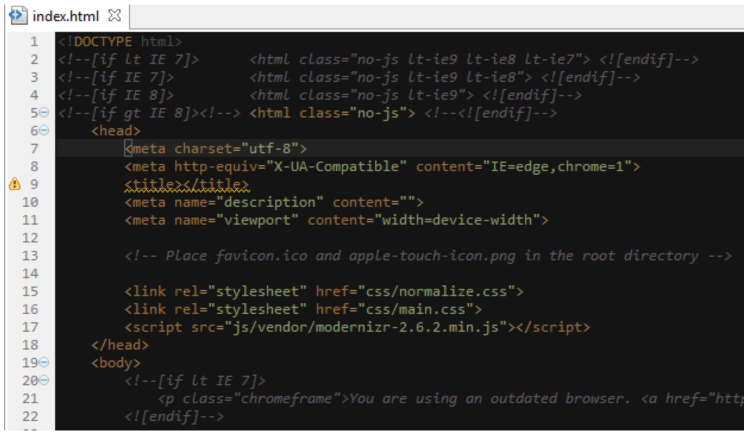

App Explorer
Project Explorer
Local Filesystem
demo
css
doc
img
js
404.html
apple-touch-icon.png
apple-touch-icon-114x114-precomposed.png
apple-touch-icon-144x144-precomposed.png
apple-touch-icon-57x57-precomposed.png
apple-touch-icon-72x72-precomposed.png
apple-touch-icon-precomposed.png
CHANGELOG.md
CONTRIBUTING.md
crossdomain.xml
favicon.ico

Figure 7.3: The Aptana Project Explorer

The layout of Aptana Studio and Titanium Studio has a Project Explorer to the left (figure 7.3) that contains all the files of the project and a tabbed Code View to the right (figure 7.4). Aptana Studio and Titanium Studio both have suggestion lists for variables, tags and functions and they warn when the users misspell things. When a user runs a web page or web applications from these developing tools they set up a dummy local server that runs them as if they were actually on a server. This is great for debugging web pages without having to upload new files to a test server. There is no way to start this debug server on a connected mobile device. Debugging the mobile web applications on a mobile device can be done by uploading it to an actual web server or to package the application as a native application with a HTML5 to native bundler.


index.html

```
<!DOCTYPE html>
<!--[if lt IE 7]>      <html class="no-js lt-ie9 lt-ie8 lt-ie7"> <![endif]-->
<!--[if IE 7]>         <html class="no-js lt-ie9 lt-ie8"> <![endif]-->
<!--[if IE 8]>         <html class="no-js lt-ie9"> <![endif]-->
<!--[if gt IE 8]><!--> <html class="no-js"> <!--<![endif]-->
    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
        <title></title>
        <meta name="description" content="">
        <meta name="viewport" content="width=device-width">

        <!-- Place favicon.ico and apple-touch-icon.png in the root directory -->

        <link rel="stylesheet" href="css/normalize.css">
        <link rel="stylesheet" href="css/main.css">
        <script src="js/vendor/modernizr-2.6.2.min.js"></script>
    </head>
    <body>
        <!--[if lt IE 7]>
            <p class="chromeframe">You are using an outdated browser. <a href="htt
        <![endif]-->
```

Figure 7.4: The Aptana Code view

7.2 Sencha Architect

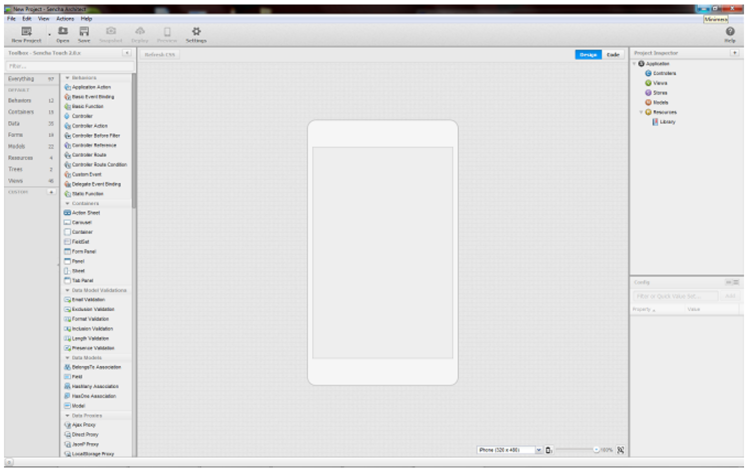


Figure 7.5: The Sencha Architect layout

Sencha Architect is an editor that is used to create Sencha web and mobile applications using their frameworks[49]. Additional functionality can be added by including other JavaScript libraries but every project built with Sencha Architect will use a Sencha library. The layout of Sencha Architect is really clean and sleek (figure 7.5).

The editor has two modes: design (figure 7.6) and code (figure 7.7). The design mode is a graphical, drag and drop, WYSIWYG editor where the user can put the wanted functionality from the tool bar (figure 7.8) directly into a board the size of a custom mobile device screen. Once the functionality is added to the design mode the code for it is automatically added to the code mode as well as showing up in the project inspector (figure 7.9). Settings for the graphical functionality can not be changed or added to through editing the project's JavaScript code. The settings can just be reached by selecting a feature and have its settings values show up in the configuration box (figure 7.11). This is because the code in code mode

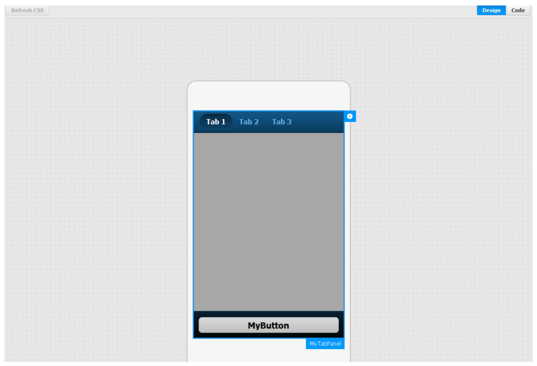
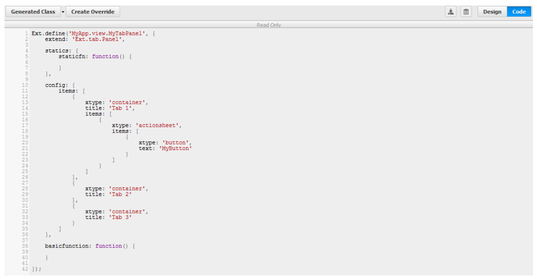


Figure 7.6: The Sencha Architect design editor



```
Ext.define('MyApp.view.MyTabPanel', {
    extend: 'Ext.tab.Panel',

    statics: {
        staticfn: function() {

        }
    },

    config: {
        items: [
            {
                xtype: 'container',
                title: 'Tab 1',
                items: [
                    {
                        xtype: 'actionsheet',
                        items: [
                            {
                                xtype: 'button',
                                text: 'MyButton'
                            }
                        ]
                    }
                ]
            },
            {
                xtype: 'container',
                title: 'Tab 2'
            },
            {
                xtype: 'container',
                title: 'Tab 3'
            }
        ]
    },

    basicfunction: function() {

    }

});
```

Figure 7.7: The Sencha Architect code editor

is read only. Code can only be edited by manually adding a function to the project (through drag-and-drop) and giving it a name through the configuration box. Once a function is added, clicking it lets the user reach the function editor (figure 7.10) and can edit the code for that single function. This takes some time to get used to and can be interpreted as the software needlessly limiting the user's access to the raw code. The function editor has functionality for reporting errors and coming with suggestions for variable and function names. Sencha Architect supports starting the project on a mobile device connected to the computer but it can be a problem previewing the project in a web browser.

7.3 Text editors

A text editor is a basic tool for developing web services. There are a lot of different text editors and some are more advanced than others. Some of them have syntax marking and help with debugging, others are just handle text plainly. An advantage of using a plain text editor is the ease of starting up the program for a quick edit or for just typing code quickly if the added functionality of a real development program is not needed. During the work of this thesis the text editor *Notepad++* (figure 7.12) has been used both for development and for documentation[30]. Notepad++ is a very flexible editor that features syntax coloring for over 50 languages and a lot of other functionality needed for basic to advanced text editing. If additional functionality is wanted it is entirely possible to add it to the program with a plug-in. A text editor can not fully replace a development studio as it normally lacks the overview and work flow that a good development tool provides.

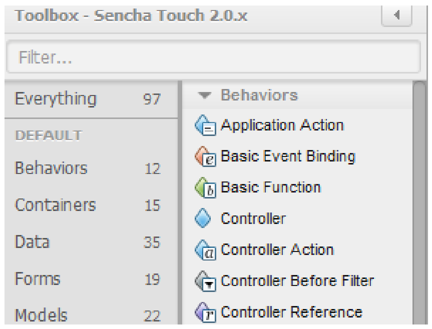


Figure 7.8: The Sencha Architect toolbar

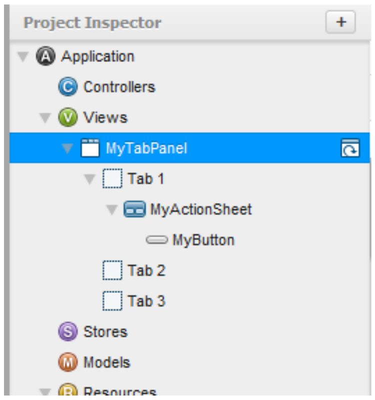
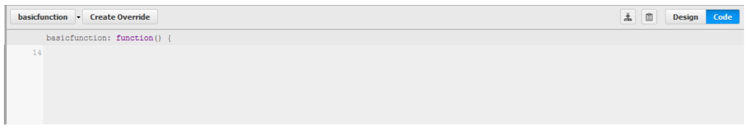


Figure 7.9: The Sencha Architect project inspector

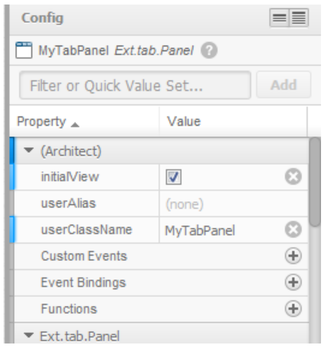
7.4 Discussion

As the project was not to use the Sencha Touch library, Sencha Architect was not considered suitable for the project. If Sencha Touch was considered for the project Sencha Architect would still not be used due to the clunky feel it gives by limiting the access to the code. The reasoning behind limiting the access to the code is to keep the interface data separate from the code so that programming functionally will not mess it up which could render the design mode unusable. The “security” gained from this decision is minimal compared to the accessibility and time lost due to it and shows limitations in the program. Another reason why limiting users from editing the code in Sencha Architect is strange is that users can develop Sencha applications with other development tools. Using another development tool would let the developer access the code at any time, which makes them the better choice and leaves Sencha Architect superfluous. The author was used to the layout of programming



```
basicfunction: function() {
```

Figure 7.10: The Sencha Architect function editor



MyTabPanel *Ext.tab.Panel*

Property	Value
(Architect)	
initialView	✓
userAlias	(none)
userClassName	MyTabPanel
Custom Events	
Event Bindings	
Functions	
Ext.tab.Panel	

Figure 7.11: The Sencha Architect configuration box

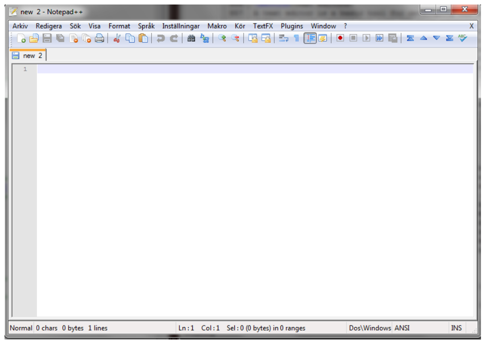

new 2 - Notepad++
Arkiv Redigera Sök Visa Format Språk Inställningar Makro Kör TextFX Plugins Window ?
new 2
Normal 0 chars 0 bytes 1 lines
Ln : 1 Col : 1 Sel : 0 (0 bytes) in 0 ranges
Dos\Windows ANSI
INS

Figure 7.12: A Notepad++ Window

environments built from Eclipse so the Aptana Studio 3 and Titanium Studio environments were preferred. Appcelerator wants to know who are using their JavaScript based cross platform solutions and thus the user needs an account on their web page to use Titanium Studio. When the program loads the user gets prompted to type in their username and password to access the environment. Because of that Aptana Studio 3 is the better choice as long as the extra functionality is not wanted. The development was done in Aptana Studio 3 and in Notepad++ and other text editors if done on other computers where Aptana was not installed.

Chapter 8

Results

This chapter will present the results of the research and show and describe the resulting web application.

8.1 Method and process

The development environments used were Titanium Studio and Notepad++. The frameworks and libraries used were Dojo and Leaflet.js

8.2 The finished application

The user interface of the finalized application 8.1 is similar to the interface of Jaktappen but has a status bar with extra buttons added to the main view and the tab bar moved to the bottom of the screen. The user interface was built using the Dojo framework and the whole page, except the map, is automatically cached on load. Once cached the page will load if there is no connection. The tab bar is for navigation purposes and lets the users

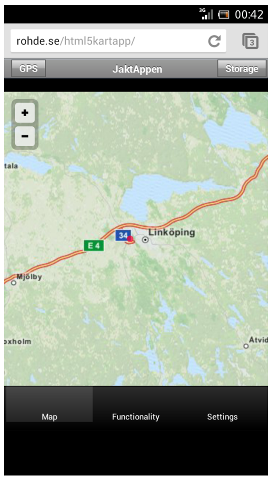


Figure 8.1: The JaktAppen Layout

reach all the views. The map view is the pre-selected standard view and will be the first view to load when accessing the application. The "Functionality" and "Settings" views are currently not implemented. The buttons on the status bars gives access to new tool bars. The GPS button gives access to a tool bar (figure 8.2) that holds GPS functionality and the storage button gives access to the tool bar (figure8.4) of saving map tiles to be used whilst offline. The map screen itself is map from Open Street Maps handled by the Leaflet.js plug-in. The GPS tool bar8.2 has two buttons; the first button is for locating the user, pressing

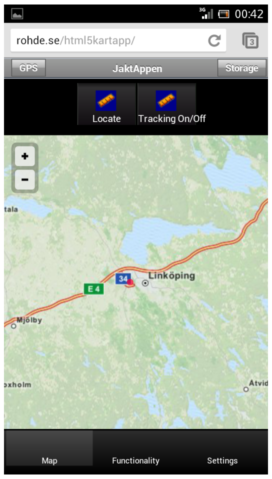


Figure 8.2: The GPS tool bar

it shows the user's location on the map with a red circle (figure 8.3). The second button is for activating/deactivating the GPS tracking which tracks user movement on the map by having the circle move over the map. The Storage tool bar (figure 8.4) has 4 buttons that

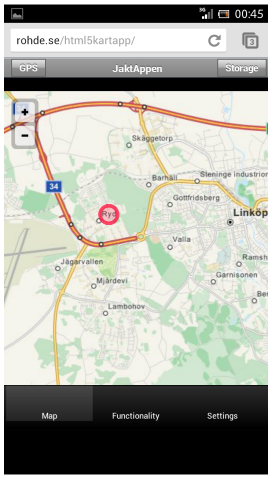


Figure 8.3: The user's location

control the functionality of the storage system. The map tile storage can save marked map tiles to be used while offline. The tiles are marked by activating the tile marking mode by pressing the "Mark On/Off" button. This will put a red border (figure 8.5) around the focused tiles to show that they are chosen for offline storage. Moving around the map will push more tiles into view to be marked. When the user has marked an appropriate amount

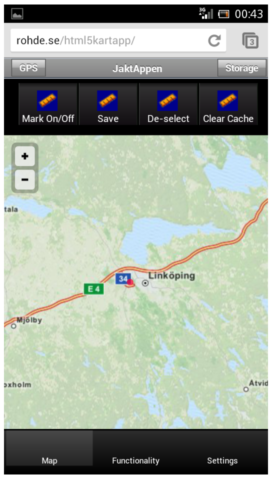


Figure 8.4: The Storage tool bar

of tiles they can be saved to the HTML5 offline storage by pressing the save button. The saved tiles will then be loaded from the offline storage instead of downloaded from a server and can thus be viewed even if there is no network connection. The remaining buttons are for clearing all the marked tiles and clearing all saved map tiles.

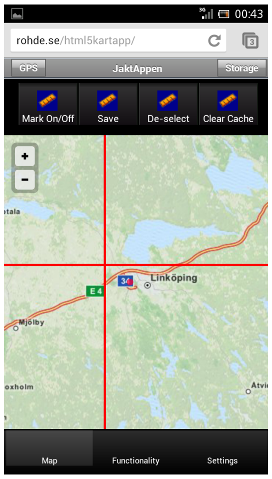


Figure 8.5: Marking map tiles

Chapter 9

Conclusions

9.1 Goals and project management

The goal of this thesis was to examine if it was possible to create a web application with a map that could be used while offline. If this was possible to do, then a prototype was to be presented. If it was not possible, a prototype showing how far it was possible to go was to be presented. A secondary purpose was to research frameworks, libraries and development tools that would help the development and add functionality. Added to this were sub-goals that appeared during development and research, such as some of the user stories in the project backlog. Using SCRUM to manage the project worked well. The project backlog was a great tool to use and the sprints and their backlogs made the work more focused. The SCRUM setup should have included time for research from the beginning as the sprints were more aimed directly towards development. This meant that the research had less focus than actual development during the beginning of the project. The research, testing and development showed that it is possible to create a web application that can both handle maps, be used offline and show cached map tiles while offline. A full implementation of a web application that uses maps that can be used offline will most likely need to prioritize what can be used offline. This is because of offline mapping needing a lot of free space available to work well.

9.2 Implementation

The current implementation is using an Open Street Maps server, which is a free to use map server, but lacks locations. For a full implementation, searchable locations is most likely needed. Searchable locations such as lakes have been added manually to the Natureit web page by the Cybercom developers. Using that server and the information the Natureit web page has available would be great for a full implementation as it would bring a lot of features to the application with a chance of reusing code from the web page. A full implementation should probably be done as a PHP HTML5 page. This is because PHP can make the web page more dynamic. A PHP Offline Cache can be used instead of Web Storage but this would make the page less dynamic as once a Cache is set for a page it is static until the page is reset and reloaded. Since Offline Cache works with image links and not data-URL's this will make for a different type of map rendering but it should work.

9.3 JavaScript

Concerning frameworks, the Dojo framework works well but in hindsight jQuery Mobile would most likely have been a better choice. This is because it is leaner in size and easier to start up. What makes Dojo harder to start working with is that their documentation and examples are outdated and that their renderer and custom data-tags takes a while to get used to. One thing with the Dojo framework that is not so good for user interactivity is that all the elements are basically differently styled <div>elements, which makes buttons feel static as they do not have a built in on-click animation. This can make it seem like the button was not clicked when it was. JavaScript libraries for mapping were interesting to research and test due to the difference in features and functionality that the developers have managed to fit in libraries of small sizes. The Tile5 library would have been very interesting to work with if there were documentation to follow but as of now the Leaflet.js library is easily the best. It is small, well documented, easy to work with, fast and rich in features.

9.4 Limitations

Using the Natureit server did not work due to limitations in cross-server accessibility and what could be installed on the server. If that server could be used, then the web application could use the map licensed by Cybercom and show the information, marked areas and points of interest that Natureit uses. A huge limit of web applications vs. native applications is the lack of support for HTML5 features in browsers made for mobile devices. Web Workers is the HTML5 way of using multi-threading and it is currently only supported by Apple mobile browsers. Multi-threading can be used to make web applications run smoother and more optimized on mobile devices. The HTML5 local file limit is an obstacle that would be neat to work around. More memory would allow for more offline capability both in the amount of tiles that can be cached and other functionality available while offline. The most severe limitation of the work was that due to time restraints and how tasks were prioritized, some tasks did not get implemented. These were tasks such as the chat client which were less prioritized due to the connectivity possibilities already available on mobile devices and that chat clients need consistent connections to work.

9.5 Future work

There is a lot that can be done with the application in the future. The first thing is to adapt the application to work with the Cybercom web solution Natureit. This means adding functionality to the application and optimizing its loading times and offline caching abilities so it acts more like a native application. Making it use the Natureit map server would be ideal. This would let users connect to the data and information, such as marked areas and points of interest, used by the Natureit web page in their mobile device. Letting users access their Natureit accounts through the web application would mean that they could set up a trip on the desktop and access it on their mobile device. Additional functionality could primarily mean implementing the other views as for example a view with settings for the application or views with ways to manipulate the map data. A search engine for points on the map and tools for adding custom markers and areas to the map that can be saved locally would be a great addition. When mobile browsers are updated to support more HTML5 features and the HTML5 API will be more complete, mobile web applications can get closer to native applications. HTML5 can not handle multitasking and threading currently, for all mobile browsers, and this makes web applications less optimized than than native applications. Optimizing the application can currently only be done through minimizing the JavaScript code and files or mimimizing the amount of objects stored in the memory. This will put less stress on the file limit and let the page load faster as well as making the page work better when loaded. If a way to accurately see what amount space there is left on the file limit can be implemented that would be great. Outside of developing, letting users test the application should be prioritized to see if the implementation and functionality is wanted. This could also bring new ideas into development.

Chapter 10

Acknowledgements

I thank my supervisor at Cybercom, Tobias Björnudd, for guiding me through the project and Kalle Prorok at UmeåUniversity for support and help with my report writing. I also thank the people at the Cybercom offices in Östersund and Linköping for the inspiration and wonderful work environment.

References

[1] Android. Platform versions, 2012. http://developer.android.com/about/dashboards/index.html, accessed 2012-10-31.

[2] Inc Appcelerator. Aptana, 2012. http://www.aptana.com/, accessed 2012-11-07.

[3] Inc Appcelerator. Titanium studio mobile application development environment, 2012. http://www.appcelerator.com/platform/titanium-studio/, accessed 2012-11-07.

[4] Vladimir Agafonkin CloudMade. Documentation - leaflet - a javascript library for mobile-friendly maps, 2012. http://leafletjs.com/reference.html, accessed 2012-11-07.

[5] Vladimir Agafonkin CloudMade. Leaflet - a javascript library for mobile-friendly maps, 2012. http://leafletjs.com/, accessed 2012-11-07.

[6] Vladimir Agafonkin CloudMade. L.tilelayer.canvas, 2012. http://leafletjs.com/reference.html#tilelayer-canvas, accessed 2012-11-07.

[7] W3 Consortium. Web sql database, 2010. http://www.w3.org/TR/webdatabase/, accessed 2012-11-06.

[8] W3 Consortium. Cascading style sheets (css) snapshot 2010, 2011. http://www.w3.org/TR/CSS/, accessed 2012-11-06.

[9] W3 Consortium. A vocabulary and associated apis for html and xhtml, 2011. http://www.w3.org/TR/html5/the-canvas-element.html, accessed 2012-11-06.

[10] W3 Consortium. Web storage, 2011. http://www.w3.org/TR/webstorage/, accessed 2012-11-06.

[11] W3 Consortium. The cache manifest syntax, 2012. http://www.w3.org/TR/html5/offline.html#manifests, accessed 2012-11-06.

[12] W3 Consortium. Css animations, 2012. http://www.w3.org/TR/css3-animations/, accessed 2012-11-06.

[13] W3 Consortium. Geolocation api specification, 2012. http://www.w3.org/TR/geolocation-API/, accessed 2012-11-06.

[14] W3 Consortium. Indexed database api, 2012. http://www.w3.org/TR/IndexedDB/, accessed 2012-11-06.

[15] W3 Consortium. Media queries, 2012. http://www.w3.org/TR/css3-mediaqueries/, accessed 2012-11-06.

[16] W3 Consortium. A short history of javascript, 2012. http://www.w3.org/community/webed/wiki/A_Short_History_of_JavaScript, accessed 2012-11-06.

[17] Sam Costello. iphone firmware & ios history, 2012. http://ipod.about.com/od/iphonesoftwareterms/a/firmw_history.htm, accessed 2012-10-31.

[18] Vladimir Agafonkin Darafei Praliaskouski and Maksim Gurtovenko. kothic/kothic-js - github, 2012. https://github.com/kothic/kothic-js, accessed 2012-11-07.

[19] Ian Alexander Dave Raggett, Jenny Lam and Michael Kmiec. A history of html, 1998. http://www.w3.org/People/Raggett/book4/ch02.html, accessed 2012-11-06.

[20] Jonathan Stark David Kaneda. jqtouch, 2012. http://jqtouch.com/, accessed 2012-11-07.

[21] Goole Developers. Managing html5 offline storage, 2012. https://developers.google.com/chrome/whitepapers/storage, accessed 2012-11-06.

[22] Alexis Deveria. Can i use, 2012. http://caniuse.com/, accessed 2012-11-05.

[23] The Dojo Foundation. Dijit overview, 2012. http://dojotoolkit.org/reference-guide/1.8/dijit/index.html, accessed 2012-11-07.

[24] The Dojo Foundation. Dojo base documentation, 2012. http://dojotoolkit.org/reference-guide/1.8/dojo/index.html#dojo-index, accessed 2012-11-07.

[25] The Dojo Foundation. Dojo html5 data-attribute support, 2012. http://dojotoolkit.org/features/1.6/html5data-attributes, accessed 2012-11-07.

[26] The Dojo Foundation. Dojo toolkit, 2012. http://dojotoolkit.org/, accessed 2012-11-07.

[27] The Dojo Foundation. Dojox documentation, 2012. https://dojotoolkit.org/reference-guide/1.8/dojox/index.html, accessed 2012-11-07.

[28] Thomas Fuchs. zepto.js, 2012. http://zeptojs.com/, accessed 2012-11-07.

[29] Ken Hess. No ios 6 for my original ipad? now, i'm an angry bird., 2012. http://www.zdnet.com/no-ios-6-for-my-original-ipad-now-im-an-angry-bird-7000004740/, accessed 2012-10-31.

[30] Don Ho. Notepad++ home, 2012. http://notepad-plus-plus.org/, accessed 2012-11-07.

[31] jQuery. jquery mobile gallery, 2012. http://www.jqmgallery.com/, accessed 2012-10-31.

[32] The jQuery Foundation. Getting started with jquery mobile, 2012. http://jquerymobile.com/demos/1.2.0/docs/about/getting-started.html, accessed 2012-11-07.

[33] The jQuery Foundation. jquery, 2012. http://zeptojs.com/, accessed 2012-11-07.

[34] The jQuery Foundation. jquery mobile, 2012. http://jquerymobile.com/, accessed 2012-11-07.

[35] Tab Atkins Jr. A word about css4, 2012. http://www.xanthir.com/b4Ko0, accessed 2012-11-06.

[36] Louis Lazaris. There is no such thing as a standards-compliant browser, 2010. http://www.impressivewebs.com/no-standards-compliant-browser/, accessed 2012-11-05.

[37] Høakon Wium Lie and Bert Bos. The css saga, 1999. http://www.w3.org/Style/LieBos2e/history/, accessed 2012-11-06.

[38] Stephen Lime. Mapscript, 2012. http://mapserver.org/mapscript/index.html, accessed 2012-11-07.

[39] Michael Mahemoff. Html5 vs native: The mobile app debate, 2011. http://www.html5rocks.com/en/mobile/nativedebate/, accessed 2012-11-05.

[40] Bernard Morris. What is the purpose of html5 in today's web design?, 2012. http://lmhawaii.com/blog/?p=936, accessed 2012-11-06.

[41] Damon Oehlman. Getting started with tile5, 2012. http://www.tile5.org/docs/tutorials/getting-started.html, accessed 2012-11-07.

[42] Damon Oehlman. Html5 mobile mapping, 2012. http://www.tile5.org/, accessed 2012-11-07.

[43] Damon Oehlman. Tile5 code playground, 2012. http://www.tile5.org/demos/playground/#basic/simple-map, accessed 2012-11-07.

[44] OSGeo. Documentation - openlayers, 2012. http://trac.osgeo.org/openlayers/wiki/Documentation, accessed 2012-11-07.

[45] OSGeo. Openlayers examples, 2012. http://openlayers.org/dev/examples/, accessed 2012-11-07.

[46] OSGeo. Openlayers: Free maps for the web, 2012. http://openlayers.org/, accessed 2012-11-07.

[47] PhoneGap. Apps created with phonegap, 2012. http://phonegap.com/app, accessed 2012-10-31.

[48] Ken Schwaber and Jeff Sutherland. The scrum guide - the definitive guide to scrum: The rules of the game. White Paper, 2011.

[49] Sencha. Sencha architect build for mobile and desktop, 2012. http://notepad-plus-plus.org/, accessed 2012-11-07.

[50] Sencha. Sencha touch build mobile web apps with html5, 2012. http://www.sencha.com/products/architect, accessed 2012-11-07.

[51] Sencha. Using components in sencha touch 2, 2012. http://www.sencha.com/learn/using-components-in-sencha-touch-2, accessed 2012-11-07.

[52] Ross Shannon. The history of html, 2012. http://www.yourhtmlsource.com/starthere/historyofhtml.html, accessed 2012-11-06.

[53] Arun Singh. What purpose does html5 serve?, 2011. `http://www.mindstick.com/Interview/1177/What%20purpose%20does%20HTML5%20serve`, accessed 2012-11-06.

[54] The WebKit Source Project. The webkit open source project, 2012. `http://www.webkit.org/`, accessed 2012-11-07.

[55] Joe Stangarone. 6 native features you can use with mobile web apps, 2012. `http://www.mrc-productivity.com/blog/2012/01/6-\%E2\%80\%9Cnative\%E2\%80\%9D-features-you-can-use-with-mobile-web-apps/`, accessed 2012-11-05.

[56] Hirotaka Takeuchi and Ikujiro Nonaka. The new new product development game. *Harvard Business Review*, 1, 1986.

[57] W3C. Plan 2014, 2012. `http://dev.w3.org/html5/decision-policy/html5-2014-plan.html`, accessed 2012-11-05.

[58] Richard Wong. In mobile, fragmentation is forever. deal with it., 2010. `http://techcrunch.com/2010/03/04/mobile-fragmentation-forever/`, accessed 2012-10-31.