VAASAN AMMATTIKORKEAKOULU
VASA YRKESHÖGSKOLA
UNIVERSITY OF APPLIED SCIENCES

Sofia Mattsson

HTML5:n hyödyntäminen mobiililaitteissa

Liiketalous ja matkailu
2012

VAASAN AMMATTIKORKEAKOULU
Tietojenkäsittely

TIIVISTELMÄ

Tekijä	Sofia Mattsson
Opinnäytetyön nimi	HTML5:n hyödyntäminen mobiililaitteissa
Vuosi	2012
Kieli	suomi
Sivumäärä	45
Ohjaaja	Mika Tamminen

Tässä opinnäytetyössä olen perehtynyt HTML5:n uusiin ominaisuuksiin ja siihen, miten näitä on mahdollista hyödyntää mobiililaitteissa. HTML-merkintäkieli on kehitetty 1990-luvun alussa World Wide Webin sisällön tuottamiseen. HTML5 toi mahdollisuuden kehittää mobiilisovelluksia ilman mobiililaitteiden varsinaisten ohjelmointikielten osaamista. HTML5:n avulla on mahdollista tehdä selainpohjaisia sovelluksia, jotka toimivat kaikissa mobiililaitteissa.

Perehdyin tutkimaan, miten mobiilisovelluksia tehdään. Toteutin selainpohjaisen sovelluksen HTML-kielellä, JavaScriptillä ja CSS-tyylitiedostolla, jonka avulla testasin HTML5:n ominaisuuksia. Sovelluksesta on mahdollista tehdä natiivi versio käyttämällä PhoneGap-sovelluskehystä.

Selainpohjainen mobiilisovellus on yksinkertainen toteuttaa jQTouch-sovelluskehyksen avulla lyhyen tutustumisen jälkeen. jQTouch-sovelluskehys koostuu JavaScript-tiedostosta. Selainpohjainen sovellus toimii mobiililaitteissa samalla tavalla kuin natiivit sovellukset. Sovellusta voidaan käyttää suoraan selaimessa, mutta myös kokonäyttötilassa, jolloin käytön yhteydessä ei huomaa, että kyse on selaimessa toimivasta sovelluksesta.

Avainsanat sovelluskehykset, HTML, mobiililaitteet, ohjelmointi

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Tietojenkäsittely

ABSTRACT

Author	Sofia Mattsson
Title	The Use of HTML5 in Mobile Devices
Year	2012
Language	Finnish
Pages	45
Name of Supervisor	Mika Tamminen

This thesis studied the new features of HTML5 and how they are usable in mobile devices. HTML was developed in the early 90's for the production of content in the World Wide Web. HTML5 gave opportunity to develop native mobile applications without having the skills to develop mobile applications. With HTML5 it is possible to develop web applications that run on all mobile devices.

The aim was to study how applications are made and to develop a web application that represents the key features of HTML5. The web application was developed with HTML, JavaScript and CSS. It is possible to run the application with PhoneGap to make it native for several mobile platforms.

Web applications are easy to develop after a short study in jQTouch. jQTouch is needed in applications and consists of a JavaScript file. Web applications work in the same way as native applications. It is possible to use the application in a mobile browser, but also in full screen, when it acts like a native application.

Keywords Mobile Device, Application Framework, HTML, Programming

SISÄLLYS

KUVA- JA TAULUKKOLUETTELO

LIITELUETTELO

1 JOHDANTO

Työni perehtyy HTML5:n käyttämiseen mobiiliohjelmoinnissa. HTML5 on uusin versio HTML-kielestä, jota on käytetty 1990-luvun alusta internetsivujen rakentamisessa. HTML5-versioon on tullut monia uusia ominaisuuksia, joiden avulla on mahdollista tehdä selainpohjaisia sovelluksia ja HTML5:n kehitysalustan avulla myös natiiveja.

Käsittelen aihetta sen ajankohtaisuuden vuoksi. Käyn työssäni läpi teoriaa HTML5:stä, selaimista ja internetin kasvusta 1990-luvun alussa. Teorian jälkeen siirryn esittelemään toteuttamaani selainpohjaista sovellusta, sen ominaisuuksia ja toimivuutta.

HTML5:n avulla tehdyt sovellukset kehitetään yhdessä CSS:n ja JavaScriptin avulla. Sovelluksessa käytin jQTouch-sovelluspohjaa, johon lisäsin HTML5:n eri ominaisuuksia. HTML5:n uusia ominaisuuksia ovat tähän mennessä: videon toisto, musiikkisoitin, sijainnin määrittäminen ja <canvas>-tunnisteen avulla tehdyt erilaiset animaatiot ja kuvagalleriat.

2 HTML-KIELEN HISTORIA

2.1 HTML-merkintäkieli

HTML (lyhenne sanoista Hypertext Markup Language) on sovelluskehys, joka yleisesti tunnetaan internetsivustojen merkintäkielenä. Merkintäkieli on kuvauskieli, jota käytetään ulkoasun osoittamiseen ja sen avulla voidaan varmistaa, että sivuston rakenne säilyy halutunlaisena (Tietotekniikan termitalkoot 2010). HTML-kieltä käytetään tiedon, kuvien, tekstien ja multimedian visuaalisen ulkoasun esittämiseen WWW:ssä (lyhenne sanoista World Wide Web). WWW on kansainvälinen käsite puhuttaessa internetistä. HTML pohjautuu SGML-metakieleen (lyhenne sanoista Standard Generalized Markup Language), joka määrittää, miten merkintäkielet toimivat rakenteeltaan sekä niiden käyttämät tunnisteet (SearchSOA 2000). SGML julkaistiin vuonna 1986.

HTML on laiteriippumaton merkintäkieli, kuten myös XML. HTML:n ja XML:n ero on niiden toimintamallit. HTML-kieltä käytetään määrittelemään, miltä tieto näyttää, kun taas XML keskittyy tiedon siirtoon ja tallentamiseen. Näistä kahdesta merkintäkielestä on kehitetty XHTML-merkintäkieli (engl. eXtensible HyperText Markup Language), jonka tarkoituksena oli korvata HTML (W3C 2002). XHTML kehitettiin yhdistäen sekä HTML:n että XML:n parhaimmat ominaisuudet (w3school 2012). XHTML:llä on tiukemmat koodaussäännöt kuin HTML:llä (w3school 2012). XHTML on siitä parempi, että kehittäjät saavat määritellä lisää tarvittavia tunnisteita, kun taas HTML-kielessä tunnisteita on rajallinen määrä.

HTML syntyi vuonna 1989 ja se julkaistiin ensimmäisen kerran NeXT-tietokoneessa vuonna 1990. HTML kehitettiin WWW:iä, eli internetiä varten, jotta dokumentteja pystyttiin esittämään visuaalisessa muodossa. Ensimmäinen virallinen versio HTML-kielestä julkaistiin vuonna 1994.

2.2 HTML-kielen kehitys

Tim Berners-Lee on yksi HTML-kielen aktiivisista kehittäjistä ja innoittajista. Berners-Lee kehitti HTML-kielen ollessaan töissä CERN:ssä. CERN oli aikoinaan eurooppalainen laboratorio atomifyysikoille, jossa fyysikot ympäri maailmaa tapasivat ja työskentelivät yhdessä miettien ilmiöitä ajassa ja avaruudessa. Nykyään CERN:ssä keskitytään kestävään kehitykseen sekä ydinfysiikan tutkimustyöhön. Berners-Lee halusi luoda ohjelmointikielen, joka ymmärtäisi hypertekstiä, eli linkkejä dokumenttien välillä. Näin fyysikoiden olisi helpompaa tehdä yhteistyötä jakamalla tekstit ja diagrammit kätevästi omien näyttöjen avulla. Tämä tarkoitti sitä, että linkit syntyivät ja graafinen ulkoasu alkoi muodostua.

Lisävaatimuksia syntyi, kun fyysikot alkoivat aktiivisesti käyttää internetiä, sillä heillä oli erityistarpeita jakaessaan tutkimustuloksiaan. Esitellessään tutkimuksia muille fyysikot halusivat lisätä dokumentteihin muutakin kuin vain tekstiä. Erityistarpeita olivat nimenomaan kuvien ja kaavioiden lisääminen dokumentteihin. Tämä johti siihen, että internetsivut kehittyivät nopeasti vuosina 1992–1994. Internetsivujen käytettävyys helpottui graafisen ulkoasun myötä.

2.3 HTML-versiot

HTML-versioiden julkaisuvuodet:

- HTML2: v. 1994
- HTML3: v. 1995
- HTML3.2: v. 1997
- HTML4: v. 1998
- HTML5: v. 2008

HTML:n ensimmäistä versiota ei ole, sillä HTML2 pohjautuu SGML-kieleen, joka tarkoittaa sitä, että virallista HTML1-versiota ei ole. HTML2 julkaistiin, jotta ohjelmointi yleistyisi virallisten määritysten myötä. HTML3 päätettiin kuitenkin julkaista piakkoin HTML2:n jälkeen, sillä todettiin, että HTML2:ssa ei ollut vielä kaikkia tarvittavia ominaisuuksia. HTML-kieli kehittyi huimaa vauhtia ja sitä oli

kehittämässä The World Wide Web Consortium, lyhennettynä W3C. Myös monet ihmiset kotonaan halusivat tehdä omia kotisivuja ja internet kasvoi. (Raggett, Lam, Alexander & Kmiec 1998).

HTML5 julkaistiin ensimmäisen kerran 22. tammikuuta 2008 (W3C 2008). Tämä julkaisu oli ensimmäinen vedos HTML5:tä. Sen jälkeen HTML5:tä on kehitetty useamman vuoden aikana ennen lopullisen version julkistamista (W3C 2012). Vuonna 2011 W3C ilmoitti HTML5-version lopullisen julkaisupäivämäärän olevan kesäkuussa 2014 (Jackson 2011).

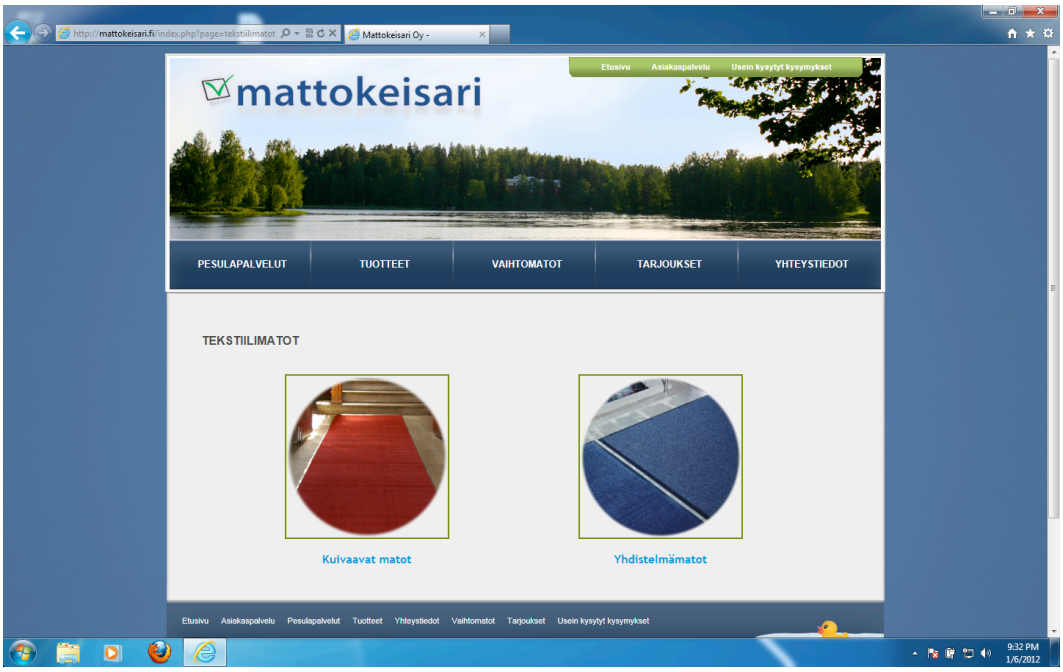
2.4 WWW-selainten kehitys

Ennen HTML2-kielen julkaisua oli olemassa useampia selaimia. Jopa videoiden lisääminen sivuille oli jo silloin käytössä. Mosaic oli yksi käytetyimmistä selaimista 1990-luvulla. Myös suomalainen selain nimeltä Erwise julkaistiin vuonna 1992. Sitä oli kehittämässä neljä nuorta yliopisto-opiskelijaa. Huonon taloudellisen tilanteen takia Erwise ei kuitenkaan lyönyt kunnolla läpi (Lasar 2011).

Vuonna 1994 julkaistun virallisen HTML2-kielen jälkeen syntyi Netscape-selain (Raggett, ym. 1998). Netscape korvasi suurilta osin aikaisemman Mosaic-selaimen. Mosaic oli kuitenkin selain, joka aiheutti internetin nopean kasvun, mutta Netscape saavutti lopulta suuremman suosion. Tämä johtui siitä, että ihmiset eivät olleet vielä hankkineet omaa päätelaitetta 90-luvun alussa, kun Mosaic julkaistiin.

Selaimet ovat olleet erilaisia käytettävyydessä. On ollut huomattavissa varsinkin internetsivustoja kehitettäessä, että sivustot ovat näyttäneet erilaisilta eri selaimilla. Joitakin asioita on ilmennyt Internet Explorer –selaimessa, kun niitä ei näkynyt Applen selaimessa nimeltä Safari. Alla vertailukuvat Safarista (kuva 1) ja Internet Explorer -selaimesta (kuva 2). Sivusto (www.mattokeisari.fi) on minun toteuttamani.

Kuva 1. Safari 5.1.2.



Kuva 2. Internet Explorer 9, jossa vihreät kehykset pyöreiden kuvien ympärillä.

Safari on jo jonkin aikaa tukenut uusia ominaisuuksia, kuten WebKit-sovellusta, joka on avoimen lähdekoodin sovelluslisäosa selaimiin. Puhuttaessa mobiili- ja tablettilaitteista Apple ei ole hyväksynyt Adoben Flashiä laitteidensa selaimiin, sillä se pitää Flashiä liian vanhana teknologiana.

Adoben Flash on lisäosa, jonka avulla on ollut mahdollista lisätä videosisältöä internetsivuille. Myös Microsoft kehitti Flashin tilalle oman lisäosan nimeltä Silverlight. Silverlightin avulla on myös mahdollista lisätä videosisältöä internetsivuille. Selaimissa on kuitenkin nykyään muitakin eroja kuin vain HTML-kielen tuettavuus, kuten nopeus ja käytettävyys.

HTML5:n tuettavuus vaihtelee suuresti tänä päivänä. IE 9:n tuettavuus HTML5:lle on puolta pienempi kuin esimerkiksi Firefoxin tai Safarin. HTML5:tä eniten tukeva selain on tällä hetkellä Maxthon 3.3, joka on saatavilla Windows-käyttöjärjestelmille ja Android-mobiililaitteille (Leenheer 2011).

3 HTML5

3.1 HTML4 vs. HTML5

On tapahtunut suuria muutoksia edellisen HTML:n ja uuden version välillä. Osa elementeistä ja attribuuteista on kokonaan poistettu suositeltavien käskyjen joukosta. Tilalle on tullut kokonaan uusia elementtejä ja attribuutteja. Tunnisteiden toimivuutta ja käyttötarkoitusta on myös uudistettu ja muutettu.

HTML5:tä kehitettäessä on otettu hyvin huomioon se, mitä käskyjä on aikaisemmin käytetty ja mitkä ovat jääneet vähemmälle käytölle. HTML-kielessä on myös korjailtu joitakin epäselviä käskyjä ja niiden käyttötarkoitusta sekä samaa tarkoittavia käskyjä, jotta HTML:n käyttö olisi selkeämpää ja toimivuus olisi parempi. Kehittäjät ovat myös halunneet parantaa HTML:n toimivuutta uudelle tasolle, jotta se aiheuttaisi vähemmän ohjelmavirheitä. HTML-kieltä on selkeästi uudistettu kehittämällä uusia ominaisuuksia, kuten videoiden lisäämisen internet-sivulle ilman tarvetta erilliselle lisäosalle.

Koodia tutkiessa voi huomata, että kehittäjät ovat erityisesti halunneet panostaa yksinkertaisuuteen ja siten helpottaa koodin kirjoittamista. Esimerkkinä toimii jokaisen HTML-koodin ensimmäinen rivi, eli DOCTYPE. HTML4:ää kirjoitettaessa DOCTYPE kirjoitettiin esimerkiksi näin: `<!DOCTYPE HTML PUBLIG "-//W3C//DTD 4.01 Transitional//EN">`. Nykyään HTML5-versiossa riittää, että kirjoittaa vain `<!DOCTYPE html>`. HTML-kieltä voi kirjoittaa noudattamatta tiukkoja sääntöjä, sulkematta tunnisteita, isoilla ja pienillä kirjaimilla (taulukko 1). XHTML-kielessä on puolestaan erittäin tarkat koodausohjeet.

Taulukko 1. HTML-kielen rakenne ja kirjoittaminen ilman tiukkoja sääntöjä.

```
<HMTL>
<body>

<DIV>
<h1>Otsikko
<p>Kappale
<p>Toinen kappale</p>
```

```
</div>

</body>
</HTML>
```

3.2 Elementit ja attribuutit

Monet elementit ja attribuutit ovat muuttuneet, osa on poistettu käytöstä ja tilalle on tullut uusia. Esimerkiksi elementtejä, joita ei enää käytetä HTML-kielessä, ovat <basefont>, <big>, ja <font>, sillä nämä elementit haittaavat esitystapaa ja toimivat paremmin CSS-tyylitiedostossa (W3C 2011).

HTML5-kielessä on mahdollista tehdä erilaisia kuvioita, kuten ympyröitä ja neliöitä, erivärisiä ja -kokoisia käyttäen <canvas>-elementtiä. Kuvioille on mahdollista tehdä animaatioita, jotka suorittavat erilaista liikettä ja värimuunnosta. Animaatio tapahtuu JavaScriptin avulla. Animaation ei tarvitse olla vain videomuodossa tapahtuvaa liikettä, vaan liikkeen voi toteuttaa HTML:n elementteihin, jolloin onnistuu esimerkiksi internetsivun latautuminen esille osissa.

Kaksi HTML5:n uusista ominaisuuksista ovat videon ja audion upottaminen internetsivustoon <video>- ja <audio>-tunnisteiden avulla (taulukko 2).

Taulukko 2. Videon lisääminen HTML-kielen avulla.

```
<div class="movie">
       <h4>Elli syö luuta</h4>
       <video src="pyry.m4v" poster="kuva/kuva.png" controls="controls"></video>
</div>
```

Attribuuttiin *src* (engl. source) kirjoitetaan, mistä lähteestä elokuva tulee, *poster* on elokuvan esikatselukuva, joka näkyy, kun sovellus aukeaa. *Controls*-attribuutin avulla ohjaimet, toisto ja tauko, näkyvät elokuvatiedoston alareunassa.

Yllämainitut elementit tarkoittavat sitä, että HTML5:llä tulee olemaan samat ominaisuudet kuin mitä Adoben Flashillä on tähän asti ollut. Flash-sovelluksen avulla

kehittäjät ovat median toiston lisäksi kehittäneet pelejä ja interaktiivisia internet-sivustoja.

Muita uusia elementtejä ovat mm. <header>, <article>, <section> ja <footer>. Nämä elementit ovat tunnisteita, joiden avulla varataan alueita internetsivulta. Varaamisella tarkoitetaan tilan varaamista siten, että esimerkiksi alkutunnisteen <header> ja lopputunnisteen </header> väliin pystyy lisäämään muita elementtejä, jotka pysyvät samassa paikassa sivulla ilman erillisiä pakotteita sijainnin määrittämiseen.

HTML5 yhdessä JavaScript-kielen kanssa kykenee tallentamaan tietoja paikalliseen selaimeen, eli välimuistiin, <html>-tunnisteen avulla. <html>-tunnisteeseen lisätään *manifest*-attribuutti, jonka avulla selain tallentaa site.manifest-tiedostossa määritetyt tiedostot, jotka ovat olennaisia sivuston toimivuudessa. *Manifest*-attribuutti on mahdollistanut selainpohjaisten sovellusten käytön yhteydettömässä tilassa. Yhteydettömästä tilasta on hyötyä, kun sivustoa haluaa käyttää ilman internetyhteyttä. Selainpohjaisista sovelluksista lisää kappaleessa 4.4 ja yhteydettömän tilan toimivuudesta kappaleessa 5.2.4.

HTML5 on mahdollistanut viisaiden lomakkeiden teon <form>-tunnisteella. Lomakeriveille määritetään erikseen attribuuttien avulla sisällön muoto, esimerkiksi sähköposti, päivämäärä tai puhelinnumero. Riveille on mahdollista suorittaa varmistuksia rivien oikeellisuudesta ja vaatia tietojen täyttöä PHP-kielen avulla. PHP (lyhenne sanoista Hypertext Preprocessor) on laajasti käytetty yleiskäyttöinen komentosarjakieli, joka sopii erityisesti WWW-kehitykseen ja voidaan upottaa HTML-kieleen. Vaihtoehtoisesti lomakkeiden teossa voidaan hyödyntää *required*-attribuuttia vaadittaessa rivien täyttöä.

Uusia attribuutteja ovat mm. *autofocus, required, placeholder* ja *form*. *Autofocus*-attribuutin avulla osoitin on valmiina sille osoitetulla lomakerivillä. *Placeholder*-attribuutin avulla on mahdollista laatia ohjetekstit riveille, jotka katoavat, kun käyttäjä osoittaa hiirellä riville ja alkaa kirjoittaa. Attribuutit, kuten *email* ja *number* (suomennettuna sähköposti ja numero), ovat erityisen käteviä mobiilisovellusta tehtäessä. Niiden avulla näppäimistö muuttuu oikeanlaiseksi, eli käyttäjän

osoittaessa mainittua riviä, puhelin näyttää joko qwerty- tai numeronäppäimistön (kuva 3).

Kuva 1. Näyttökuvat iPhone-puhelimesta.

Vasemman puoleisessa kuvassa on esitetty, miten näppäimistö muuttuu, kun käyttäjä valitsee sähköpostille tarkoitetun rivin. Oikean puoleisessa kuvassa käyttäjä on valinnut rivin puhelinnumerolle.

4 HTML5 JA MOBIILILAITTEET

4.1 Mobiilisovellukset

Mobiilisovellus on ohjelma, joka on suunniteltu varta vasten puhelimen käyttöjärjestelmälle. Sovelluksia on tähän mennessä kehitetty iPhonelle 500 000 kpl, Androidille 400 000 kpl ja Windows Phonelle 70 000 kpl (Apple Inc. 2012; I. Paul 2012; Blandford 2012). Monet sovellukset vaativat internetyhteyden, kuten Facebook ja sähköpostiohjelma. Mobiilisovelluksista on tullut jo niin suosittuja, että internetin käyttö mobiililaitteilla on ohittanut tietokoneella käytetyn ajan (Talouselämä 2011).

Mobiililaitteille tarkoitetut selainpohjaiset sovellukset ohjelmoidaan käyttäen HTML5:tä, JavaScriptiä ja CSS-tyylitiedostoa. HTML5 on mahdollistanut helpomman tavan kehittää sovelluksia, eikä ainoastaan selainpohjaisia vaan myös natiiveja sovelluksia. Kappaleessa 4.3 esitellään, mikä on natiivi sovellus, ja kappaleessa 4.4 on tietoa selainpohjaisista sovelluksista.

4.2 Mobiilialustat

HTML5 toimii jokaisella suositulla mobiilialustalla, esimerkiksi iOS:llä, Androidilla ja Windows Phonella. HTML5 tulee olemaan erittäin tärkeä mobiililaitteiden kehityksessä. Älypuhelimia yhdistää sama ominaisuus, nimittäin selain. Jokainen selain tukee HTML-kieltä, joka taas mahdollistaa selainpohjaiset mobiilisovellukset. Tämän takia HTML5 on niin yksinkertainen ja varmasti yksi tulevaisuuden käytetyimmistä ohjelmointikielistä mobiililaitteille.

Haittoja toki myös löytyy, kuten alustojen puutteellinen tuki HTML5:lle. Käyttöjärjestelmien takaamat ominaisuudet eivät vielä toimi selainpohjaisissa sovelluksissa yhtä hyvin kuin natiiveissa sovelluksissa. iPhonelle ja Androidille on tarjolla mallikoodeja eri ominaisuuksista, jotka ovat kirjoitettuna niiden omilla ohjelmointikielillä, eli Objective-C:llä ja Javalla, joita ei voi käyttää HTML5-kieltä kirjoitettaessa.

HTML5:tä joutuu itse opiskelemaan, jotta sen käytön etuja pystyy hyödyntämään täysin mobiilisovelluksia ohjelmoitaessa. Mobiilialustat ovat erittäin kehittyneitä, sillä ohjelmointirajapinnan avulla voidaan hyödyntää eri natiiveja ominaisuuksia. (Firtman 2011). Ohjelmointirajapinta on selitettynä tarkemmin kappaleessa 4.3.

4.3 Natiivi sovellus

Natiivi sovellus tarkoittaa sovellusta, joka on kehitetty tietylle mobiilialustalle. Natiivi sovellus on myös ohjelmoitu mobiilialustan omalla ohjelmointikielellä, esimerkiksi iPhonelle kehitetään sovelluksia käyttäen Objective-C:tä, Androidille käyttäen Javaa sekä Windows Phonelle C#:lla. Natiivit sovellukset ostetaan useimmiten palveluntarjoajan verkkokaupasta, kuten AppStoresta, Google Play:stä tai Marketplacesta ja ne asennetaan puhelimeen.

Natiiveissa sovelluksissa on käytettävissä Applen, Googlen ja Microsoftin kehittämät koodinosat, jotka ovat hyödyllisiä ja sisältävät erilaisia ominaisuuksia. Näitä ominaisuuksia ovat mm., miten sovelluksen avulla saa GPS-yhteyden tai miten kameran saa käynnistettyä. Edellä mainitut ominaisuudet liittyvät ohjelmointirajapintaan (engl. API, joka on lyhenne sanoista Application Programming Interface).

Ohjelmointirajapinnan avulla eri ohjelmat voivat keskustella keskenään lähettämällä pyyntöjä ja vaihtamalla tietoja. Ohjelmointirajapintojen avulla on mahdollista kehittää pelejä ja hyötysovelluksia, jotka hyödyntävät puhelimen eri ominaisuuksia, kuten kiihtyvyysanturia, gyroskooppia ja GPS:ää ilman, että sovellus häiriintyy rajapinnan alla toimivasta tietokannasta.

4.4 Selainpohjainen sovellus

Selainpohjainen sovellus (engl. Web Application) on selaimeen kehitetty sovellus. Aiemmin selainpohjaiset sovellukset, kuten internetsivustot, tallennettiin puhelimeen ja ne toimivat kirjanmerkkeinä eli oikoteinä sivustoille. Niitä ei ollut mahdollista muuttaa kokonäyttötilaan ja sovelluksilla ei ollut omia kuvakkeita. HTML5:n uusien ominaisuuksien takia nämä kirjanmerkit muuttuivat enemmän natiivin sovelluksen kaltaisiksi.

HTML5:n avulla kuvakkeita on mahdollista tehdä selainpohjaisiin sovelluksiin, näin niitä ei erota päältäpäin muiden sovellusten joukosta. Näytölle ilmestyvä aloitusruutu on mahdollista erikseen määrittää. Itse sivuston/palvelun saa tehtyä kokonäyttötilassa toimivaksi, jolloin sitä ei huomaa käytössä, että se on selaimen kautta toimiva.

Selainpohjaiset sovellukset ovat mobiililaitteiden selaimessa toimivia sivustoja, palveluja ja hyötysovelluksia. Niissä toimivat melkein kaikki samat ominaisuudet kuin natiiveissa sovelluksissa. iOS 5:n ilmoituskeskus on esimerkki ominaisuudesta, joka ei toimi selainpohjaisissa sovelluksissa. Muita esimerkkejä ovat Siri ja Twitter-integraatio. iOS 5 on iPhonen ja iPadin uusin käyttöjärjestelmäversio tähän mennessä.

Mikä on natiivin ja selainpohjaisen sovelluksen ero? Sisältö ei niinkään ole ratkaiseva niiden välillä. Selainpohjaisten sovellusten tärkeimmät määritelmät ovat, että ne on kehitetty käyttäen verkkostandardeja, saatavilla WWW-osoitteesta ja optimoitu erityisesti mobiililaitteille.

Sitä voisi luulla, että selainpohjaista sovellusta käytettäessä tarvitaan aina internetyhteys, mutta näin ei ole. HTML5:ssä on ominaisuus, joka mahdollistaa sovelluksen käytön offline-tilassa. Offline-tila tarkoittaa tilaa, jolloin ei ole internetyhteyttä. Tämä on hyödyllistä, kun kyseessä on esimerkiksi peli, mutta jos kyseessä on uutissivusto, niin uutiset eivät päivity, vaan sen sijaan vanhoja uutisia on mahdollista lukea. Itse sovellusta ei ole mahdollista tehdä maksulliseksi, mutta palvelut sovelluksessa sen sijaan voivat olla. Esimerkiksi lehtitalon tekemä sivusto näköislehden lukemiseen on helppo tehdä sovellukseksi, jolloin näköislehden lukemisesta veloitetaan.

4.5 PhoneGap

PhoneGap on sovelluskehys, jonka avulla on mahdollista tehdä hybridisovelluksia miltei kaikille mobiilialustoille: iOS, Android, Windows Phone, BlackBerry, WebOS, Symbian ja Bada (PhoneGap 2012). Hybridisovellukset näyttävät ulospäin natiivien sovellusten kaltaisilta, mutta käyttävät kuitenkin selainta toimiakseen.

Hybridisovellukset eroavat selainpohjaisista sovelluksista siten, että ne ovat nykyään hyväksyttyjä sovelluskaupoissa. Saadakseen sovelluksen kaikille yllä mainituille mobiilialustoille toimimaan, tarvitsee tehdä vain yksi sovellus HTML-kielellä, CSS:llä ja JavaScriptillä. Tämän jälkeen kooditiedostot käännetään PhoneGapillä jokaisen mobiililaitteen alustaan sopivaksi. Vastaava vaihtoehtoinen sovelluskehys PhoneGapille on Sencha Touch. Sencha Touchin avulla on mahdollista kääntää selainpohjaiset sovellukset iOS:lle ja Androidille. Käytän PhoneGapia esimerkkinä, sillä se on tällä hetkellä kehittynein sovelluskehys.

Käännös tapahtuu kirjautumalla PhoneGapin palvelimelle osoitteessa build.phonegap.com, johon käyttäjä lataa sovelluksensa joko pakattuna tiedostona tai pelkästään index.html-tiedoston. Takaisin käyttäjä saa valmiit natiivit sovellukset seitsemälle eri alustalle. PhoneGap on ilmainen, kun sillä tekee yleisiä sovelluksia, mutta muuttuu maksulliseksi, jos sovelluksia haluaa tehdä yksityiseen käyttöön enemmän kuin yhden (PhoneGap 2012). Yleisillä sovelluksilla tarkoitetaan tässä yhteydessä sovelluksia, jotka ovat kaikille ilmaiseksi ladattavissa käännöksen jälkeen. Jos haluaa käyttää PhoneGapia hybridisovellusten tekoon yksityiseen käyttöön, on siitä maksettava PhoneGapille. PhoneGapin avulla on mahdollista hyödyntää mobiililaitteiden omia natiiveja ominaisuuksia, kuten esimerkiksi kamera- ja karttaohjelmaa.

Apple ja Google olivat alussa erittäin tarkkoja julkaistujen sovellusten laadusta ja käytettävyydestä. Aikaisemmin oli luvallista kehittää sovelluksia iOS-alustoille käyttäen ainoastaan Objective-C-ohjelmointikieltä ja Androidille käyttäen Java-ohjelmointikieltä. Vuonna 2009 Android julkaisi mahdollisuuden kehittää sovelluksia käyttäen C- ja C++-ohjelmointikieliä ja täten myös PhoneGapia (R. Paul 2009). Apple antoi hyväksyntänsä PhoneGapilla tehdyille sovelluksille vuonna 2009 (Claburn 2009). Windows Phonelle sovellusten kehittäminen kehittäjien osalta alkoi marraskuussa 2010, kun Microsoft hyväksyi kehittäjien julkaista sovelluksia Marketplaceen (Finley 2010).

4.6 HTML5:n hyödyntäminen

HTML5:n avulla säästää sekä aikaa että rahaa sovelluksen yksinkertaisen julkaisun, kehittämisen ja päivittämisen ansiosta. Selainpohjaisten sovellusten kehittämisestä ja julkaisemisesta ei tarvitse maksaa kuluja. Tämä on merkittävä asia, sillä jos haluaa kehittää usealle eri mobiilialustalle, on jokaisella palveluntarjoajalla omat maksut, jotka pitää suorittaa, kun haluaa tehdä natiiveja sovelluksia. Selainpohjaisia sovelluksia varten tarvitsee vain olla palvelin, jolle tallentaa sovellukset.

Kun halutaan kehittää natiiveja sovelluksia, on jokaiselle yritykselle, esimerkiksi Applelle, Googlelle ja Microsoftille maksettava rekisteröitymismaksu ja jokaisesta myydystä sovelluksesta osa menee palvelun tarjoajalle. Applella ja Microsoftilla ovat samat hinnat ja kulut: vuosittainen kehittäjämaksu on 99 dollaria ja maksullisista ohjelmista menee 30 prosenttia palvelumaksua (Apple Inc. 2012; Microsoft 2012). Googlella maksaa 25 dollaria rekisteröityä Android Marketiin ainiaaksi ja jokaisesta myydystä maksullisesta ohjelmasta menee 30 prosenttia Googlelle (Google 2012).

HTML5:n avulla tehdyt selainpohjaiset sovellukset päivitetään lataamalla päivitetty versio palvelimelle, jolloin päivittyvät myös käyttäjien puhelimissa sijaitsevat sovellukset. Tämä mahdollistaa nopeammat ohjelmavirheiden korjaukset ja uusien päivitysten julkaisut verrattuna natiiveihin sovelluksiin.

Natiivit sovellukset päivitetään lähettämällä sovellus uudelleen hyväksyttäväksi. Vaikka kyse on vain päivityksestä, on se joka kerta hyväksyttävä Applen, Googlen tai Microsoftin kautta. Sovellusten hyväksyntä AppStoreen ja Android Marketiin kestää yleensä noin 14 päivää, kun taas Windows Phone Marketplaceen kestää vain 3–5 päivää (Finley 2010).

5 SOVELLUS

5.1 Esittely

Käytännön osuus työssäni keskittyi selainpohjaisen mobiilisovelluksen kehittämiseen. Sovelluksen avulla esittelen yleisimmät HTML5-kieleen liittyvät ominaisuudet, kuten videoiden käyttäminen sovelluksissa, sijainnin paikallistaminen ja sovelluksen käyttö yhteydettömässä tilassa, eli offline-tilassa. Sovelluksen toteuttamisessa käytin Coda-kehitystyökaluohjelmaa ja jQTouch-sovelluspohjaa. Coda on kehitystyökalu, jolla voi ohjelmoida esimerkiksi HTML:llä, CSS:llä ja JavaScriptillä. Valitsin Codan, sillä käytän Macintoshia ja Codassa on useita käteviä ominaisuuksia, jotka helpottavat koodin kirjoittamista.

jQTouch on ohjelmistokomponenttikirjasto, joka sisältää tarvittavat tiedostot mobiilisovelluksen kehittämiseen, kuten JavaScriptit, CSS-tyylit ja HTML-tiedostot. Kirjaston sisältämiä koodeja saa käyttää omaan sovellukseensa apuna. jQTouch sisältää erilaisia efektejä, joiden avulla välilehdet aukeavat oikealta vasemmalle, ylhäältä alas tai pyörähtäen esiin sekä kolme erilaista tyylitiedostoa. Vastaavia ominaisuuksia on Sencha Touchilla ja jQuery Mobile:lla. Valitsin kuitenkin jQTouchin, sillä olen aikaisemmin tutustunut siihen.

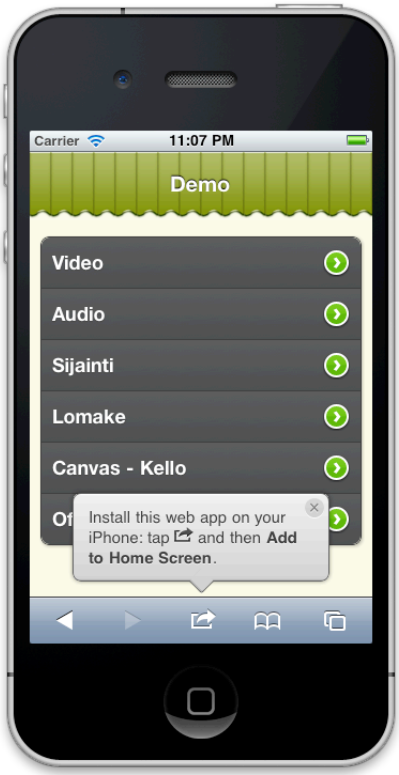
Sovellus on nimeltä Demo ja sovelluksen kuvake on valkoinen D mustalla pohjalla (kuva 4). Sovelluksen avulla esittelen HTML5:n ominaisuudet käytännössä ja vertailen, mikä toimi hyvin ja mikä vähemmän hyvin.

Kuva 2. Tämä on sovelluksen kuvake, joka näkyy puhelimen työpöydällä.

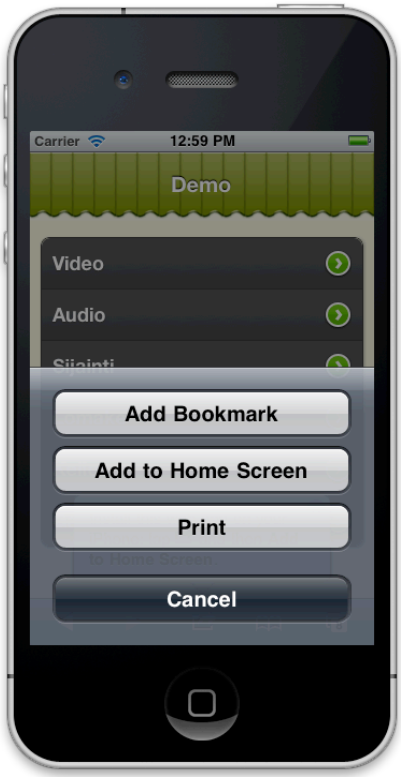
Kuvake on toteutettu Adobe Photoshop -kuvankäsittelyohjelmalla. Kuvakkeen mitat ovat 114 * 114 pixeliä. Puhelin lisää kuvakkeelle automaattisesti kulmien pyöristykset ja varjon.

5.2 Käyttöliittymät

Sovellus ladataan osoitteesta sofliini.lumies.fi. Sovelluksen etusivulle ilmestyy ohje, miten sovellus tallennetaan puhelimeen (kuva 5). Ohje katoaa itsestään 10 sekunnin kuluttua, ellei käyttäjä poista sitä ylänurkassa sijaitsevasta rastista. Osoittamalla näytön alareunassa sijaitsevaa kuvaketta, pongahtaa auki lista, josta valitaan *Add to Home Screen,* suomennettuna ”lisää kotivalikkoon” (kuva 6). Tämä tarkoittaa sitä, että sovellus tullaan tallentamaan kirjanmerkkinä puhelimeen, jonka avulla pääsee suoraan käyttämään sovellusta www-osoitteessa. Sovellusta on myös mahdollista käyttää ilman, että sitä tallentaa puhelimeen. Selaimessa käytettynä tilaa on vähemmän johtuen alareunan valikosta. Käyttöä voi myös häiritä osoitekenttä, joka välillä ilmestyy yläreunasta esiin selatessa sivua ylöspäin. Jos sovellus on tallennettu puhelimeen, sitä ei voi päivittää käytön aikana, joten voi olla hyödyllistä käyttää sovellusta selaimessa. Tallennettu sovellus päivittyy ainoastaan sen latautuessa auki.

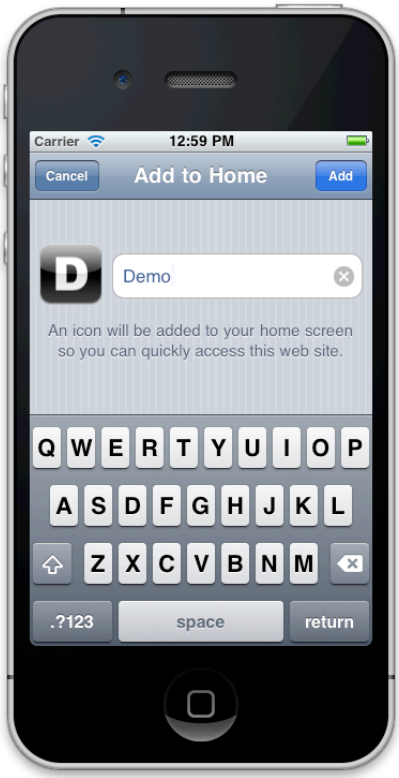


Kuva 5. Selaimessa ilmestyy ohje, miten tallentaa sovellus.



Kuva 3. Listasta valitaan *Add to Home Screen*.

Seuraavassa vaiheessa tallennetaan sovellus osoittamalla ylänurkasta *Add* eli ”lisää” (kuva 7). HTML5 on mahdollistanut kuvakkeen lisäämisen sovellukselle, jolloin kuvake ja nimi ovat valmiiksi sovelluksella. Käyttäjä voi vapaasti vaihtaa nimen, ennen kuin tallentaa sovelluksen. Tämän jälkeen sovellus tallentuu puhelimen kotivalikkoon (kuva 8). Sovellus avataan osoittamalla sovelluksen kuvaketta.



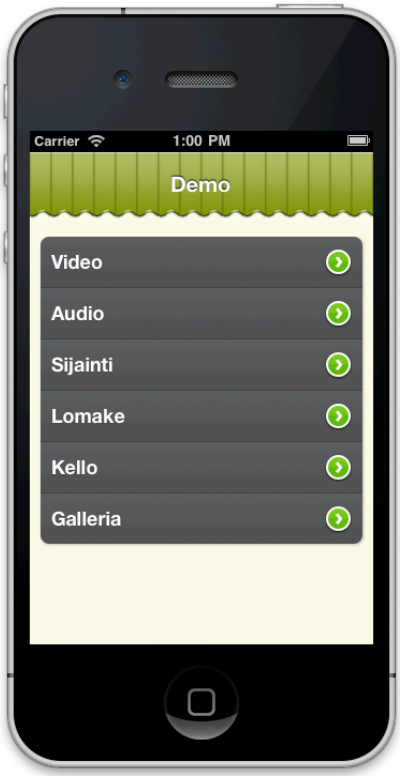
Kuva 4. Sovellus tallennetaan ylänurkasta osoittamalla *Add*.

Kuva 8. Sovellus kotivalikossa.

Sovelluksen latautuessa auki näkyy aloitusruutu (kuva 9). Selainpohjainen sovellus voi olla hidas aukeamaan verrattuna natiiveihin sovelluksiin. Syynä on se, että selainpohjainen sovellus avaa internetyhteyden joka kerta latautuessaan auki, jolloin internetyhteyden nopeus on ratkaisevassa roolissa. Jos sovellus on ollut aikaisemmin yhteydessä internetiin, sovellus voi toimia myös ilman internetyhteyttä. Selainpohjainen sovellus näyttää kotivalikosta avattuna natiivin sovelluksen kal-

taiselta (kuva 10). Eroa ei huomaa ulospäin, sillä sovellus toimii kokonäyttötilassa ilman selaimen osoiteriviä ja alareunan valikkoa.

Kuva 5. Sovellus latautuu auki.



Kuva 10. Selainpohjainen sovellus kokonäyttötilassa.

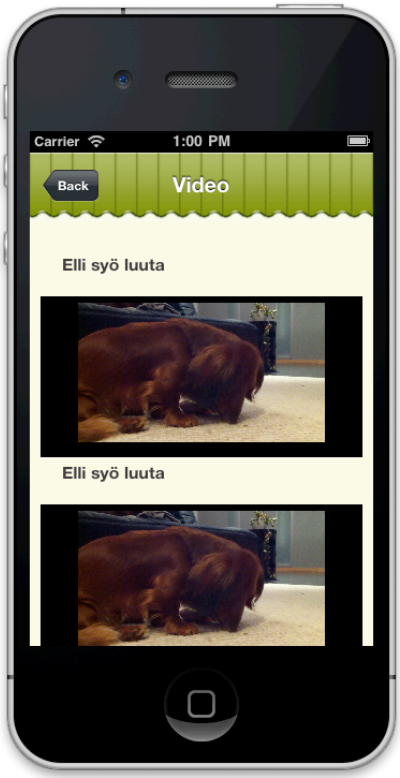
Sovelluksen etusivulla näkyy listaus sovelluksessa olevista ominaisuuksista. Listan rivejä osoittamalla pääsee seuraavalle sivulle. Sivut aukeavat oikealta vasemmalle ja suljettaessa vasemmalta oikealle.

5.3 Ominaisuudet

Sovelluksessa esittelen videokuvan toimivuutta, musiikkisoitinta, oman sijainnin määrittämistä kartalla, canvas-elementillä toteutettua kelloa, lomaketta ja galleriaa. Ominaisuudet ovat valmiita kokonaisuuksia, jotka olen löytänyt eri lähteistä internetistä, testannut ja lisännyt sovellukseeni, jotta voin esitellä HTML5:n uusimmat ominaisuudet yhdessä.

5.3.1 Video ja musiikki

Sovelluksen etusivulla ensimmäisenä listalla on Video-sivu, johon on lisätty kaksi videota (kuva 11). Videoilla esiintyy koirani Elli syömässä luuta. Video on lisätty <video>-tunnisteen avulla. Tunnisteeseen lisätään attribuutti *src*, johon annetaan arvoksi polku videotiedostoon (taulukko 3). Tiedoston on hyvä sijaita samassa kansiossa kuin sivusto.



Kuva 11. Video-sivu.

Taulukko 3. Videon lisäys.

```
<video src="pyry.m4v" poster="kuva.png" controls="controls"></video>
```

Seuraava sivu on nimeltä Audio, jossa on musiikkisoitin (kuva 12). Osoittamalla Toista-painiketta soitin toistaa musiikkia. Seuraava-painikkeesta soitin toistaa seuraavan kappaleen listalta. Myös osoittamalla suoraan kappaletta listalta, kappale

alkaa soimaan. Audion esittäminen onnistuu samalla tavalla kuin videon. Tähän käytettävä <audio>-tunnistetta (taulukko 4).

Kuva 12. Audio-musiikkisoitin.

Taulukko 4. Musiikin lisääminen <audio>-tunnisteella.

```
<audio id="musiikki" preload="auto">
    <source src="Halo.mp3" type="audio/mpeg" />
</audio>
```

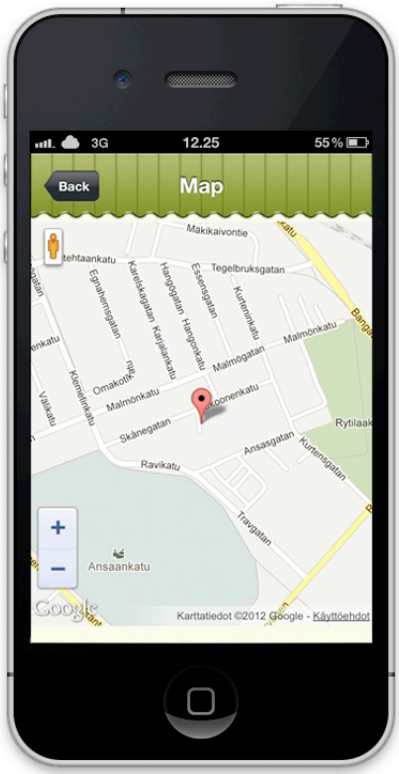
Preload-attribuutti lataa musiikin etukäteen, jotta musiikki toistuisi saman tien käyttäjän osoittaessa toista-painiketta. Lähde asetetaan <audio>-tunnisteiden sisään ja suljetaan saman tien /> -merkkijonolla.

5.3.2 Sijainti kartalla

Sijainnin määrittäminen ei varsinaisesti ole HTML5:n ominaisuus, mutta mainitaan usein HTML5:n yhteydessä. HTML5:llä voi käyttää ohjelmointirajapintaa,

jonka avulla on mahdollista hyödyntää mobiililaitteen GPS-yhteyttä, jolloin sijainnin määrittäminen on tavallaan osa HTML5:tä.

Oman sijainnin määrittämisessä käytin Googlen tarjoamaa valmista koodia. Karttaa pystyy liikuttamaan, lähentämään ja loitontamaan (kuva 13). Siinä toimii myös katunäkymä (engl. Streetview). Katunäkymä aukeaa vetämällä oranssia ukkoa vasemmasta reunasta kartan päälle siihen kohtaan, jonka haluaa nähdä (kuva 14). Katunäkymässä voi liikkua eteenpäin, takaisin ja sivuille, myös katujen ja teiden nimet näkyvät.



Kuva 13. Karttanäkymä.

Kuva 14. Katunäkymä.

Googlen tarjoamaa koodia ohjelmointirajapintaa varten kutsuttiin yhdellä komentorivillä, johon lisättiin lähteeksi www-osoite (taulukko 5). Koodi on suurimmaksi osaksi JavaScriptiä, jossa määritellään Google Maps -toiminnon avulla leveys- ja pituusasteet (liite 1). Näiden avulla sijainti määräytyy kartalle. HTML5-kieleen lisättiin ainoastaan ohjelmointirajapintaa kutsuva komentorivi ja -tunniste, johon kartta tulee näkyviin (taulukko 6).

Taulukko 5. Ohjelmointirajapintaa kutsuva komentosarja.

```
<script src="http://maps.google.com/maps/api/js?sensor=false"></script>
```

Taulukko 6. Sijainnin rakenne HTML-koodissa.

```
<body>
    <div id="getlocation">
         <div id="mapholder"></div>
         <script src="http://maps.google.com/maps/api/js?sensor=false">
</script>
    </div>
</body>
```

5.3.3 Offline Cache

Offline Cache on välimuistiin tallentuva metadata. Toteutin yhteydettömän tilan toimivuuden, jotta sovellusta on mahdollista käyttää myös ilman internetyhteyttä. Tämä toteutettiin kolmella askeleella:

1. <html>-tunnisteeseen lisätään seuraava rivi `manifest="site.manifest"`

2. Luodaan uusi tiedosto nimeltä `site.manifest`. Tiedostoon lisätään niiden tiedostojen nimet, jotka ovat olennaisia, jotta sovellus toimii, esimerkiksi index.html (taulukko 7).

3. Luodaan uusi tiedosto, jonka nimeksi laitetaan `.htaccess`. Tiedostoon kirjoitetaan ainoastaan `AddType text/cache-manifest manifest`.

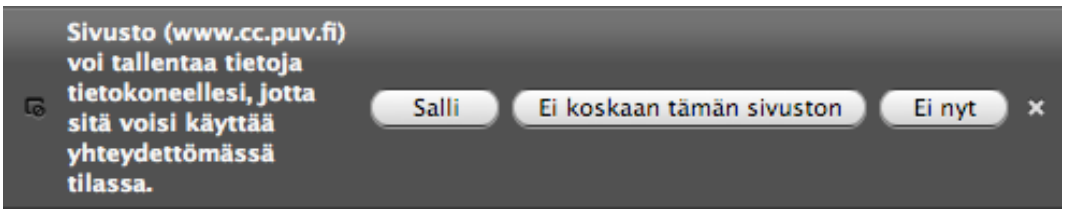
Taulukko 7. Sisältö site.manifest-tiedostossa.

```
CACHE MANIFEST

# version 0.1

index.html
style.css
script.js
preview.jpg
```

Ilman internetyhteyttä sovellus toimii seuraavasti: Käyttäjä käyttää selainpohjaista sovellusta, jolloin selain lataa ja tallentaa tiedot puhelimen välimuistiin, jotka on määritettynä site.manifest-tiedostossa. On erittäin tärkeää, että tiedostojen nimet on kirjoitettu oikein, jotta sivusto toimii yhteydettömässä tilassa. Muutoin yhteydettömän tilan käyttö ei onnistu. Seuraavalla kerralla kun käyttäjä käyttää selainpohjaista sovellusta, selain käyttää välimuistin tietoja sen sijaan, että lataisi sivuston uudestaan internetin kautta. Jos jokin on muuttunut sovelluksessa viime käytön jälkeen, selain lataa ja päivittää muuttuneet tiedot sovellukseen.

Testasin sovelluksen käyttöä yhteydettömässä tilassa ja se osoittautui toimivaksi. Sovellus aukesi huomattavasti nopeammin, kun selaimen ei tarvinnut ladata sovelluksen tietoja uudelleen internetyhteyden avulla. Videoita, musiikkisoitinta, kuvia ja kelloa oli mahdollista käyttää. Sijaintia ei ollut mahdollista paikantaa eikä lomaketta ollut mahdollista lähettää.

Kokeilin myös sovellusta tietokoneella käyttäen Firefox-selainta, joka osoittautui myös toimivaksi, sillä Firefox tukee välimuistiin tallentamista. Sovelluksen auettua selain ilmoitti mahdollisuudesta käyttää sivustoa yhteydettömässä tilassa (kuva 15).

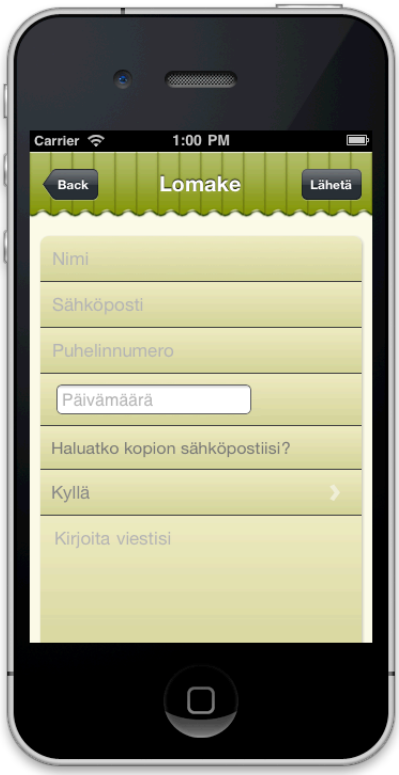


Kuva 15. Testaus Firefox-selaimessa tietokoneella.

5.3.4 Älykäs lomake

HTML5-kielen <form>-elementissä on uusia elementtejä ja attribuutteja. Sovelluksen lomakkeessa on nimi, sähköposti, puhelinnumero, päivämäärä, pudotusvalikko ja viestikenttä (kuva 16). Lomakeriveillä on kuvailevat tekstit valmiina, jotka on tehty *placeholder*-attribuutilla. Kun käyttäjä osoittaa riviä, teksti katoaa alta pois ilman, että sitä tarvitsee erikseen poistaa. Näppäimistö muuttuu aina kentän

sisällön mukaan. Jos kenttään on tarkoitus kirjoittaa numeroita, puhelin näyttää numeronäppäimistön.



Kuva 16. Sovelluksen lomake.

En kuitenkaan voinut hyödyntää kaikkia elementtejä sovelluksessa johtuen tämänhetkisestä HTML5:n tuettavuudesta, esimerkiksi <datalist>-elementtiä. Tämä olisi ollut hyödyllinen, sillä sen avulla on mahdollista tehdä pudotusvalikkoja lisäämällä valikon arvot *option*-attribuutteihin (taulukko 8).

Taulukko 8. Esimerkki <datalist>-elementin käytöstä.

```
<input list="browsers" />

<datalist id="browsers">
  <option value="Internet Explorer">
  <option value="Firefox">
  <option value="Chrome">
  <option value="Opera">
  <option value="Safari">
</datalist>
```

Käyttämäni pudotusvalikko toteutettiin <select>-elementillä (taulukko 9). PHP-tiedosto hyödyntää <value>-attribuuttien arvoja 1 ja 2, jonka avulla se tietää, kumman vaihtoehdon käyttäjä valitsi.

Taulukko 9. Pudotusvalikon toteutus.

```
<li><select name="kysymys">
        <option value="1">Kyllä</option>
        <option value="2">Ei</option>
</select></li>
```

Sähköpostin lähettämisessä käytin PHP-ohjelmointikieltä, jotta sain lomakkeen lähettämään sähköpostia. HTML5-kielen <form>-elementtiin lisätään attribuutti *action*, johon laitetaan arvoksi lähde PHP-tiedostoon ja *method*-attribuuttiin laitetaan arvoksi post (taulukko 10). Lähetä-napista sovellus lähettää lomakkeen sähköpostilla osoitteeseen, joka on määritetty PHP-koodissa. Valintaikkunasta on mahdollista valita, lähettääkö lomake kopion käyttäjälle vai ei. Jos on valittuna ”Kyllä”, lomake lähettää kopion siihen osoitteeseen, joka on täytetty sähköpostiriville, ja kun valittuna on ”Ei”, niin lomake ei lähetä kopiota, vaikka sähköpostiosoite on annettu.

Taulukko 10. HTML-kielen <form>-elementti.

```
<form class="form" action="mail.php" method="post" type="text/plain" auto-
complete="on" >
<ul class="rounded">
       <li><input name="nimi" type="text" placeholder="Nimi" /></li>
       <li><input name="email" type="email" placeholder="Sähköposti" /></li>
       <li><input name="puhelin" type="tel" placeholder="Puhelinnumero" /></li>
       <li><input name="date" type="date" placeholder="Päivämäärä" </li>
       <li><label class="teksti">Haluatko kopion sähköpostiisi?</label></li>
       <li><select name="kysymys">
              <option value="1">Kyllä</option>
              <option value="2">Ei</option>
       </select></li>
       <li><textarea name="txt" placeholder="Kirjoita viestisi"></textarea>
       </li>
       <button class="button" type="submit" value="Submit">Lähetä</button>
</ul>
```

```
</form>
```

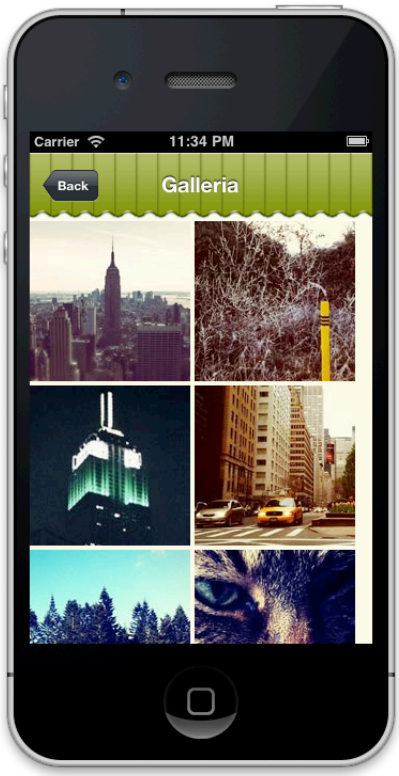
5.3.5 Canvas ja animaatio

HTML5-kielessä on uusi elementti nimeltä <canvas>. <Canvas> on alue, jolle voi luoda grafiikkaa, liikettä ja säilöä valokuvia. JavaScriptin avulla luodaan sisältö. Sovelluksen kello on tehty <canvas>-tunnisteen avulla (kuva 17). HTML-koodiin on lisätty linkki JavaScript-tiedostoon, jonka avulla on määritetty värit, koot ja animaatiossa tapahtuvat liikkeet. Kellon viisarit liikkuvat ajan kanssa eteenpäin ja näyttävät jatkuvasti ajan reaaliajassa.

Kuva 17. Kello on toteutettu <canvas>-tunnisteen avulla.

Muita animaatiomahdollisuuksia on kehittää interaktiivista kanssakäymistä käyttäjän kanssa, jossa käyttäjä pystyy muokkaamaan liikkeen suuntaa, kokoa ja väriä. Canvas-tunnisteen avulla olisi ollut mahdollista kehittää kuvagalleria, mutta en onnistunut saamaan sitä toimimaan. Jouduin toteuttamaan gallerian perinteisem-

mällä tavalla (kuva 18). Tein gallerian linkkien avulla. Pikkukuvia osoittamalla kuvat aukeavat kokonäyttötilassa (kuva 19). Kuvaesitys tapahtuu JavaScriptin avulla, jossa on määritettynä sormen liikkeen minimipituus, jolloin kuva vaihtuu.



Kuva 18. Galleria.

Kuva 19. Kuvaesitys.

6 YHTEENVETO

6.1 Toimivuus

Sovellus toimii kokonaisuudessaan hyvin. Sovellusta tehdessä ilmeni ongelmia sivujen selattavuudessa, hitaudessa ja ominaisuuksien toimivuudessa. Kun lisäilin ominaisuuksia ja koodirivien määrä kasvoi, sovelluksen käyttö hankaloitui. Sovellus avautui hitaasti ja sovelluksen selailu oli hankalaa. Käymällä läpi koodirivejä ja ulkoistamalla JavaScript-rivejä linkeiksi sovellusta käsitteli koodia nopeammin. Myös yleinen siisteys ja käyttämättömien koodirivien poistaminen oli hyväksi.

Sovellus avautuu riippuen internetyhteyden nopeudesta välillä nopeammin ja välillä hitaammin. Toimivuus on parempaa, kun sovellusta käyttää yhteydettömässä tilassa. Nopeuteen vaikuttaa myös sisällön määrä, kun sovellus lataa avautuessaan myös video- ja audiosisällöt. Etukäteen ladattavat toiminnot ovat määritettynä *preload*-attribuutissa, jotta sovellusta käytettäessä eri toiminnot, kuten videon toisto, alkavat saman tien.

Ominaisuudet eivät alussa toimineet lainkaan halutulla tavalla. Eniten ongelmia aiheutti sijainnin näkyminen kartalla. Aluksi käytin HTML Demos-sivuston antamaa valmista koodia, mutta sijainti ei näkynyt kartalla, vaan rajautui pois vasemmalta puolelta. En onnistunut korjaamaan koodia, joten etsin toisen ratkaisun. Löysin Googlen antaman valmiin koodin sijainnin määrittämiseen. Googlen tarjoamalla koodilla sijainnin määrittäminen onneksi onnistui vaivattomasti. Jäljelle jäi kartan koon määrittäminen sopivaksi puhelimen selaimessa ja kokonäyttötilassa.

Muita ominaisuuksia, jotka aiheuttivat ongelmia, olivat videon toisto ja kuvagalleria. Aluksi minulla oli video väärässä muodossa, .mov, mutta vaihdoin .m4v-muotoon, jolloin video alkoi toimimaan. Kuvagalleria ei lopputuloksessakaan toiminut halutulla tavalla. Toivomuksena olisi ollut liukuva kuvan selaus, kun taas lopputuloksessa kuvat vaihtuvat ilman liikettä. Muutoin kuvagallerian lopputulos

on hyvä. Kello oli helpoin ominaisuus lisätä sovellukseen. Kello toimi alusta alkaen oikein ja muokkaaminen oli helppoa.

6.2 Ohjelmoinnin helppous

HTML-kieli oli tuttu ennestään. Mobiilisovellusta ohjelmoitaessa HTML-kielen osaamisesta oli apua, mutta opiskella piti kuitenkin lisää, sillä HTML5-versioon oli tullut paljon muutoksia ja uudistuksia. Mobiilille ohjelmoitaessa HTML-koodin rakenne toimi eri tavalla kuin internetsivustoa tehdessä. Internetsivustojen välilehdet tehdään erillisille .html-tiedostoille, kun taas mobiilisovellusta tehdessä jokainen sivu on samassa koodissa. Eri sivut määritettiin id-tunnisteiden avulla, jotka listattiin -elementtiin <a>-tunnisteiden sisään, jolloin niistä muodostui linkkejä sivuille (taulukko 11).

Taulukko 11. Linkit välilehtiin id-tunnisteen avulla.

```
<ul class="rounded">
      <li class="arrow"><a href="#video">Video</a></li>
      <li class="arrow"><a href="#audio">Audio</a></li>
      <li class="arrow"><a href="#getlocation" onclick="getLocation()">
      Sijainti</a></li>
      <li class="arrow"><a href="#formi">Lomake</a></li>
      <li class="arrow"><a href="#canvaasi">Kello</a></li>
      <li class="arrow"><a href="#galleria">Galleria</a></li>
</ul>
```

Suurin puute osaamisessani oli JavaScriptissä, jonka avulla sovellus suurimmaksi osaksi toimi. Sovelluksen rakenne määriteltiin HTML-kielellä, mutta sovelluksen toimivuus perustui jQTouchin JavaScript-tiedostoon, joka kokosi id-tunnisteista sovelluksen mahdollistaen selaamisen eri välilehtien välillä. Ilman jQTouchia sovellus olisi vain yhdellä sivulla toimiva, jolloin eri sivut näkyisivät allekkain.

HTML-kielen osuus sovelluksessa oli rakentaa koodin runko, jossa määritettiin elementtien sijainnit, koot, attribuutit ja niiden arvot. HTML-kieleen lisättiin myös JavaScript-koodia sekä linkkejä ulkoisiin JavaScript-tiedostoihin.

6.3 Natiivi vs. selainpohjainen sovellus

Kun haluaa julkaista yleisimmille mobiilialustoille natiivin sovelluksen, ohjelmoinnin osaaminen seitsemälle eri alustalle on välttämätöntä. Sovellusten päivittäminen virallista kautta vaatii aina sovelluksen hyväksymisen, jotta sen saa myyntiin ja jakoon. Siinä voi mennä useampia viikkoja. Jos haluaa vain julkaista yhdelle tai kahdelle eri mobiilialustalle, voi natiivin sovelluksen tekeminen hyödyttää, sillä silloin kehittäjä saa käyttöönsä mobiilialustoille tarkoitettuja valmiita koodipaketteja eri toimintoja varten ja toimivuus on varmempaa. Natiivia sovellusta kehitettäessä on mahdollista käyttää puhelimen omia natiiveja ominaisuuksia, kuten kameraa ja karttaohjelmaa.

Selainpohjaisissa sovelluksissa on se etu, että tarvitsee tehdä vain yhdellä ohjelmointikielellä yhden sovelluksen, joka toimii seitsemässä eri mobiilialustassa. Tarvittaessa sovelluksesta on mahdollista tehdä myös hybridisovellus, jolloin se pitää hyväksyttää, jonka jälkeen sen saa myyntiin sovellusmarketeille. Selainpohjaisen sovelluksen päivittäminen käy helposti tallentamalla uuden version palvelimelle, jonka jälkeen jokaisen käyttäjän sovellus päivittyy, kun he seuraavan kerran avaavat sovelluksen.

6.4 Lopputulos

Kannattaako perehtyä selainpohjaisten sovellusten kehittämiseen? Ovatko ne käytettäviä ja tarpeellisia, jos on jo ennestään internetsivustot, joita käyttäjä voi lähennellä ja loitonnella lukiessa mobiililaitteella? Miksi siis tehdä erikseen mobiiliverkkosivut ja edelleen mobiilisovellus?

Kaiken oppimani jälkeen olen sitä mieltä, että selainpohjainen sovellus on hyödyllinen, sillä käyttökokemus on paljon parempi, kun sivuston koko on optimoitu puhelimen näytölle eikä käyttäjän tarvitse asettaa sivua näytölle nähdäkseen lukea. Myös käyttäjän tallentaessa sivuston puhelimeensa sivusto tallentuu sovelluksen kaltaiseksi, jolloin on helppoa avata sivusto uudestaan.

Mobiiliverkkosivun ja mobiilisovelluksen ero ei ole suuri. Mobiiliverkkosivu on selaimessa toimiva, jolloin käytettävissä on osoiterivi ja näytön alareunassa on valikko, mutta verkkosivua ei voi käyttää kokonäyttötilassa. Mobiilisovellus toimii myös selaimessa, mutta sivustoa on mahdollista käyttää kokonäyttötilassa ilman osoiteriviä ja alareunan valikkoa. Mobiilisivuston tekeminen mobiilisovellukseksi vaatii vain yhden koodiriviä lisää (taulukko 12).

Taulukko 12. Meta-linkki, jonka avulla internetsivusto tallentuu sovellukseksi.

```
<meta name="apple-mobile-web-app-capable" content="yes" />
```

HTML-koodiin lisätty meta-linkki mahdollistaa mobiiliverkkosivun tallentamisen puhelimeen, jolloin verkkosivu muuttuu sovellukseksi. Kun verkkosivun seuraavan kerran avaa sovelluksena, käytettävissä ei ole osoiteriviä eikä alareunassa valikkoa. Tästä on hyötyä, sillä käyttäjän on helppo selata esimerkiksi uutissivustoa tai verkkokauppaa eksymättä muille sivustoille.

LÄHTEET

Kirjat

Powers, D. 2011. *Adobe Dreamweaver CS5.5 Studio Techniques: Designing and Developing for Mobile with jQuery, HTML5, and CSS3*. Pearson Education (US).

Korpela, J. 08/2011. *HTML5 - uudet ominaisuudet*. WSOY.

Artikkelit

Blandford, R. (25. 2 2012). *Windows Phone Marketplace passes 70,000 apps*. Noudettu osoitteesta All About Windows Phone: http://allaboutwindowsphone.com/news/item/14316_Windows_Marketplace_passes_700.php

Claburn, T. (25. 11 2009). *Apple Accepts PhoneGap For iPhone Development*. Noudettu osoitteesta InformationWeek: http://www.informationweek.com/news/personal-tech/smart-phones/221901204

Finley, K. (4. 11 2010). *Microsoft Opens Windows Phone 7 Market to All Developers*. Noudettu osoitteesta ReadWrite: http://www.readwriteweb.com/hack/2010/11/microsoft-opens-windows-phone.php

Firtman, M. (14. 10 2011). *Safari on iOS 5, HTML5 evolution for iPhone and iPad*. Noudettu osoitteesta Mobile Web Programming: http://www.mobilexweb.com/blog/ios-5-iphone-and-ipad-html5

Jackson, J. (14. 02 2011). *W3C: HTML5 will be finished in 2014*. Noudettu osoitteesta Computerworld: http://www.computerworld.com/s/article/9209322/W3C_HTML5_will_be_finished_in_2014

Lasar, M. (2011). *ars technica*. Noudettu osoitteesta http://arstechnica.com/web/news/2011/10/before-netscape-forgotten-web-browsers-of-the-early-1990s.ars

Paul, I. (2012). *Android Market Tops 400,000 Apps*. Noudettu osoitteesta PCWorld: http://www.pcworld.com/article/247247/android_market_tops_400000_apps.html

Paul, R. (2009). *Android goes beyond Java, gains native C/C++ dev kit*. Noudettu osoitteesta ars technica: http://arstechnica.com/open-source/news/2009/06/android-goes-beyond-java-gains-native-cc-dev-kit.ars

Talouselämä. (20. 06 2011). *Mitä? Mobiilisovellukset ohittivat jo pc-surfailun*. Noudettu osoitteesta Talouselämä:

http://www.talouselama.fi/uutiset/mita+mobiilisovellukset+ohittivat+jo+pcsurfailun/a645733

Elektroniset julkaisut

Apple Inc. (2012). *iOS Developer Program*. Noudettu osoitteesta Developer: http://developer.apple.com/programs/ios/

Apple Inc. (2012). *iPhone*. Noudettu osoitteesta Apple: http://www.apple.com/iphone/built-in-apps/app-store.html

Google. (2012). *Android Market*. Noudettu osoitteesta Google: https://support.google.com/androidmarket/developer/bin/answer.py?hl=fi&answer=113468&topic=2365624&ctx=topic

Leenheer, N. (2011). *Desktop browsers*. Noudettu osoitteesta The HTML5 Test: http://html5test.com/results.html

Microsoft. (2012). *faq: answers at a glance*. Noudettu osoitteesta App Hub: http://create.msdn.com/en-us/home/faq

PhoneGap. (2012). Noudettu osoitteesta PhoneGap:Build: https://build.phonegap.com/

PhoneGap. (2012). *Supported Features*. Noudettu osoitteesta PhoneGap: http://phonegap.com/about/features

Raggett, D.;Lam, J.;Alexander, I.;& Kmiec, M. (1998). *A history of HTML*. Noudettu osoitteesta W3C: http://www.w3.org/People/Raggett/book4/ch02.html

SearchSOA. (2000). *SGML*. Noudettu osoitteesta SearchSOA: http://searchsoa.techtarget.com/definition/SGML

Tietotekniikan termitalkoot. (28. 12 2010). *Merkintäkieli*. Noudettu osoitteesta Tietotekniikan termitalkoot: http://www.tsk.fi/tsk/termitalkoot/fi/node/266

W3C. (12. 03 2012). *HTML » HTML Working Group*. Noudettu osoitteesta W3C: http://www.w3.org/html/wg/

W3C. (25. 05 2011). *HTML5 differences from HTML4*. Noudettu osoitteesta W3C: http://www.w3.org/TR/html5-diff/

W3C. (22. 01 2008). *W3C Publishes HTML 5 Draft, Future of Web Content*. Noudettu osoitteesta W3C: http://www.w3.org/2008/02/html5-pressrelease.html.en

W3C. (01. 08 2002). *XHTML 1.0*. Noudettu osoitteesta W3C: http://www.w3.org/TR/2002/REC-xhtml1-20020801/

w3school. (2012). *HTML vs. XHTML*. Noudettu osoitteesta w3school: http://www.w3schools.com/html/html_xhtml.asp

w3school. (2012). *Introduction to XML*. Noudettu osoitteesta w3school: http://www.w3schools.com/xml/xml_whatis.asp

Webkit. (ei pvm). Noudettu osoitteesta The WebKit Open Source Project: http://www.webkit.org/

OMAN SIJAINNIN MÄÄRITTÄMINEN

```
<script type="text/javascript">
     var x=document.getElementById("demo");
     function getLocation()
     {
        if (navigator.geolocation)
     {
     navigator.geolocation.getCurrentPosition(showPosition,showError);
        }
        else{x.innerHTML="Geolocation is not supported by this brow-
ser.";}
        }

     function options()
     {
        enableHighAccuracy : true;
        timeout : 20;
        maximumAge : 15;
     }

     function showPosition(position)
     {
       lat=position.coords.latitude;
       lon=position.coords.longitude;
       latlon=new google.maps.LatLng(lat, lon)
       mapholder=document.getElementById('mapholder')
       mapholder.style.height='380px';
       mapholder.style.width='100%';

       var myOptions={
              center:latlon,zoom:15,
              mapTypeId:google.maps.MapTypeId.ROADMAP,
              mapTypeControl:false,
              navigationControlOptions:
              {style:google.maps.NavigationControlStyle. SMALL}
       };

       var map=new google.maps.Map(document.getElementById("mapholder"),
myOptions);

       var marker=new google.maps.Marker(
       {position:latlon,map:map,title:"Olet tässä!"});
        }

       function showError(error)
       {
              switch(error.code)
       {
              case error.PERMISSION_DENIED:
              x.innerHTML="User denied the request for Geolocation."
       break;
              case error.POSITION_UNAVAILABLE:
              x.innerHTML="Location information is unavailable."
       break;
```

```
                case error.TIMEOUT:
                x.innerHTML="The request to get user location timed out."
        break;
                case error.UNKNOWN_ERROR:
                x.innerHTML="An unknown error occurred."
        break;
        }
        }
</script>
```