KARELIA-AMMATTIKORKEAKOULU
Viestinnän koulutusohjelma

Hanna Kinnunen

# RESPONSIIVINEN VERKKOSUUNNITTELU JA SEN TOTEUTTAMINEN OFFLINE-MOBIILISOVELLUKSEN RAKENTAMISESSA

Opinnäytetyö
Helmikuu 2014

Karelia AMMATTIKORKEAKOULU

**OPINNÄYTETYÖ**
**Helmikuu 2014**
**Viestinnän koulutusohjelma**

Länsikatu 15
80110 JOENSUU
p. (013) 260 6991

Tekijä
Hanna Kinnunen

Nimeke
Responsiivinen verkkosuunnittelu ja sen toteuttaminen offline-mobiilisovelluksen rakentamisessa

Toimeksiantaja
Tourist Guide for Northern Periphery -hanke

Tiivistelmä

Mobiiliteknologian kehittyminen ja mobiililaitteiden yleistyminen ovat asettaneet verkon kehittäjille haasteen saada verkkopalvelut toimimaan kaikilla alustoilla ja kaikenlaisilla laitteilla. Tuotantotehokkuuden haasteeksi nousee monelle mobiilialustalle rakentaminen ja ratkaisua on haettu mukautuvista verkkoteknologioista ja HTML5:stä.

Opinnäytetyön tarkoituksena on tutkia verkkopalvelun suunnittelua, jonka lähtökohtana ovat mobiililaitteet. Mobiililaitteilla on erilainen käyttökonteksti kuin tietokoneilla, joten verkkopalvelun suunnittelun lähtökohdankin tulee olla erilainen. Opinnäytetyössä perehdytään responsiiviseen verkkosuunnitteluun teknisestä näkökulmasta: miten responsiivisuus toteutetaan ja kuinka sitä voidaan hyödyntää mobiilisovellusten rakentamisessa.

Opinnäytetyön toiminnallisessa osuudessa iPadille rakennettu tablet-sovellus muutetaan eri näyttökokoihin mukautuvaksi Android-sovellukseksi. Raportti kuvaa, kuinka projekti etenee suunnittelusta toteutukseen ja millaisiin ratkaisuihin on päädytty responsiivisuuden toteutuksessa. Opinnäytetyö tuo esiin ja tutkii ongelmia, joita sovelluksen muutoksen aikana ilmenee.

Lopuksi pohditaan Internetin tulevaisuutta ja miten verkon kehittäjät pystyvät vastaamaan verkkosuunnittelun nopeaan muutokseen.

Kieli
suomi

Sivuja 46
Liitteet 0
Liitesivumäärä 0

Asiasanat
Responsiivinen verkkosuunnittelu, mobiilisovellus, internet, verkon kehittäminen

Karelia
UNIVERSITY OF APPLIED SCIENCES

**THESIS**
**February 2014**
**Degree Programme in Communication**
Länsikatu 15
FI 80110 JOENSUU
FINLAND
Tel. 358-13-260 6991

Author
Hanna Kinnunen

Title
The Implementation of Responsive Web Design in an Offline Mobile Application

Commissioned by
Tourist Guide for Northern Periphery -project

Abstract

The development of mobile technology and the spread of mobile devices have set a challenge for web developers to get their web services to work across multiple platforms, operating systems and all different devices. When developing for several mobile platforms the production efficiency poses large challenges. Developers have looked into HTML5 and web technologies to solve this challenge.

This thesis focuses on studying web design from the mobile point of view. Mobile devices have a different use context than desktop devices so the premise of developing a web service should also be different. The thesis concentrates on responsive web design from a technical perspective: how the responsiveness can be implemented and how it can be used when constructing mobile applications.

In the functional part of the thesis an application designed for iPad is modified to create a responsive and adaptive Android application. The report presents how the project progresses from planning to implementation and what kind of solutions are used when making the application responsive. The thesis discusses the problems that are faced during the building of this application.

Lastly the thesis discusses the future of the Internet and how web developers might be able to meet the ever-changing challenges of web design.

| Language | Pages 46 |
| --- | --- |
| Finnish | Appendices 0 |
| | Pages of Appendices 0 |

Keywords
Responsive Web Design, mobile applications, internet, web development

## Sisältö

# 1 Johdanto

Kymmenen vuotta sitten tulevaisuuden näkymiin kuului, että Internet siirtyy jokaisen ihmisen taskuun pieninäyttöisille mobiililaitteille ja laajentuu kaikkialla olevaksi. Langattoman verkon ajateltiin tuovan meille uuden keinon, jolla voimme tehdä töitä, pelata ja kommunikoida. (Hiltunen, Laukka & Luomala 2002, 173.) Jo 10 vuotta sitten osattiin päätellä myös, että hyvän käytettävyyden keinot, kuten saavutettavuus, toimivuus ja luettavuus tulisivat yhä tärkeämmiksi mobiiliverkossa. Arvailtiin, että silloisten verkkopalvelujen siirtämiseen mobiililaitteisiin oli kaksi eri tapaa: suunnitella palvelut uudestaan myös mobiililaitteille sopiviksi tai muokata nykyiset palvelut mukautumaan mobiilimaailman rajoituksiin. (Hiltunen ym. 2002, 173.) Jo tuolloin kehotettiin tekemään sivustoista yksinkertaisia ja käytettäviä niin ulkoasuiltaan kuin sisällöiltään, jotta sivustot eivät vanhentuisi niin nopeasti ja niitä voisi käyttää pienilläkin näytöillä. Tekstit näkyvät millä laitteella tahansa ja ne ovat verkkopalvelujen tärkeintä sisältöä, joten niiden asetteluun ja muotoiluun kannattaa panostaa.

Word Wide Web on nyt suuremmassa muutoksen tilassa, kuin se on koskaan ollut. Olemme siirtymässä aikaan, jossa erilaisia verkkoselaimia on paljon ja laitteista tulee pienempiä ja suurempia yhtä aikaa. Jo tällä hetkellä pieninäyttöiset laitteet ovat suosituin tapa, joista verkkoa käytetään ja samalla modernit pelikonsolit ovat tuoneet televisiokokoisten laitteiden kautta käytettävän verkon helposti saataville. Valtavan suosituksi tulleiden taulutietokoneiden kautta meille avautuu verkon muoto, joka ei ole täysin ”mobiili” eikä ”tietokoneympäristö”, vaan jotain siltä väliltä. (Marcotte 2011, 6.)

Internetistä on viimeisen vuosikymmenen aikana tullut kaikkialla oleva (Johnson 2012). Enää verkon käyttäminen ei katso aikaa eikä paikkaa, ja koska käyttäjien tarpeet ovat kasvaneet, on myös verkon sopeuduttava tähän muutokseen. Verkon kehittäjien haasteeksi tulee saada verkon sisällöt toimimaan kaikilla laitteilla ja kaikissa tilanteissa, joissa verkkoa käytetään, sillä verkkopalvelujen on aina

pyrittävä mahdollisimman hyvään käytettävyyteen. Verkon luonteeseen kuuluu yleisesti nopeus ja tiedonsaannin helppous. Jos käyttäjän etsimä tieto ei löydy nopeasti ja helposti, käyttäjä siirtyy toiselle sivustolle.

Verkon kehittäjien haasteena on luoda jotain, mikä toimii monilla alustoilla ja tämä koskee sekä verkkosivuja että mobiilisovelluksia. On tuotantotehokkaampaa luoda monelle alustalle yhtä aikaa, kun esimerkiksi samoja, joskus paljonkin työtä vaativia toimintoja ei tarvitse rakentaa moneen kertaan. Jotta mobiilisovellusten rakentaminen olisi tehokasta, ovat kehittäjät siirtyneet rakentamaan itsenäisiä mobiilisovelluksia (standalone) HTML5:llä laitteiden omien ohjelmakielien sijaan. (Bricklin 2013.)

Mobiililla verkolla tarkoitetaan world wide webin käyttöä kannettavalla mobiililaitteella, kuten älypuhelimella, tavallisella matkapuhelimella tai tablettaulutietokoneella, matkapuhelinverkon tai muun langattoman verkon kautta (Wikipedia 2013b). Vuonna 2010 ITU:n (International Telecommunication Union) raportti kertoi, että silloisen kasvutahdin mukaan verkkoa tultaisiin luultavasti käyttämään liikkeellä ollessa – kannettavien tietokoneiden ja älypuhelinten kautta – enemmän, kuin pöytäkoneiden kautta seuraavan viiden vuoden kuluessa (ITU 2010).

Useat verkon alkuaikojen verkkopalvelut ovat kestäneet hyvin aikaa, sillä niiden sisältö oli pääasiassa tekstiä ja teksti näkyy jokaisella laitteella laitteen koosta riippumatta. Verkkostandardien kehityttyä pystyttiin kiinnittämään enemmän huomiota ulkoasuun. Tähän asti suuri osa verkkopalveluista on suunniteltu jäsentämällä ulkoasut ja sisällöt Photoshopissa tai muussa kuvankäsittely- tai grafiikkaohjelmassa tiettyyn kiinteään pikselileveyteen. Kuvamallin perusteella verkkopalvelu siirrettiin verkkoon HTML:ksi ja CSS:ksi pikselin tarkkuudella.

Kun mobiililaitteet alkoivat yleistyä, alettiin verkkopalveluja muokata paremmin pienille näytöille ja hitaisiin datayhteyksiin sopiviksi. Hyvin usein poistettiin suurikin osa muotoilusta sekä grafiikoista ja näkyviin jätettiin vain teksti- ja kuvasisällöt. Koska mobiililaitteiden internet-yhteydet olivat hitaita, joskus myös kuvat jätettiin kokonaan pois.

Mobiililaitteita varten tehtiin myös erillinen versio verkkosivustosta yleensä aladomainien (m.domain.com, mobile.domain.com) tai oman verkkopäätteen (www.domain.mobi) alle. Eri domainin alle sisältöjen siirtäminen aiheutti ongelmia erityisesti verkon yhtenäisyyden kanssa. Sisällöt mobiilisivustolla ja tavallisen verkkosivun välillä vaihtelivat ja ylläpitäjien täytyi pitää yllä kahta sivustoa. Kaksi versiota verkkopalvelusta vie myös tietenkin enemmän palvelintilaa. Erityisesti käyttäjille hankalaksi koitui se, että sisällöt ja niiden sijainnit sivuston rakenteessa vaihtelivat versioiden välillä ja url-linkkien jakaminen ja tallentaminen hankaloitui.

Responsiivinen verkkosuunnittelu (Responsive Web Design) on kehittynyt vastaamaan juuri mobiililaitteiden tuomiin ongelmiin. Idea skaalautuvasta verkkosisällöstä juontaa juurensa koko selainikkunan tai näytön kokoisiin Flash-animaatioihin mutta terminä responsiivinen verkkosuunnittelu nousi pinnalle vasta vuonna 2010 kun Ethan Marcotte lähti esittelemään jokaiseen päätelaitteeseen sopeutuvan verkkopalvelun periaatteita. Tämän opinnäytetyön tarkoituksena on tutustua responsiivisen verkkosuunnittelun lähtökohtiin ja tutkia, kuinka mobiilisovellukset voivat hyötyä tästä verkkosuunnittelun uudesta suunnasta.

# 2 Verkkopalvelun suunnittelu

## 2.1 Palvelun suunnitteluvaiheet ja tehtävät

Palvelun suunnitteluvaihe koostuu useista vaiheista ja virstapylväistä, jotka tulee täyttää saavuttaakseen päämääränsä. Tehtävät, joita suunnittelun aikana tehdään, täydentävät toisiaan ja helpottavat aina seuraavan vaiheen toteuttamista. Kaikki lähtee liikkeelle konseptin luomisesta, tavoitteiden tunnistamisesta ja suunnittelusta. Esimerkiksi yksinkertaisimmillaan konsepti voi olla yrityksen verkkosivusto, palvelun tavoitteena on kertoa yrityksestä, sen palveluista ja hankkia asiakkaita. Suunnitteluun sen sijaan kuuluu enemmän eri vaiheita.

Suunnittelun tarkoituksena on toteuttaa verkkopalvelun konseptin tavoitteita ja jos pohjatyö on tehty hyvin (esimerkiksi kohderyhmän määrittely, tavoitteiden konkretisointi) on suunnittelun aloittaminen ja projektin läpivieminen helpompaa. (Hiltunen ym. 2002, 88.)

Verkkopalvelun konseptia joudutaan usein tarkentamaan suunnitteluvaiheessa. Suunnittelijan tulee pyrkiä vastaamaan kysymyksiin, kuten:

- Mitä sivuja verkkopalveluun tarvitaan?
- Kuinka jaan ja ryhmittelen sisällöt?
- Mitkä ovat suurimmat ja selkeimmät kokonaisuudet?
- Millainen ulkoasu tai värimaailma palvelee parhaiten palvelun päämäärien toteutumista?
- Missä palvelua tullaan käyttämään eniten ja millä päätelaitteella?

Varsinainen suunnittelu siis tapahtuu konseptin luomisen ja käyttöliittymän luomisen yhteydessä ja parhaiten se onnistuu aloittamalla kokonaisuuksista ja siirtymällä vähitellen työstämään yksityiskohtaisempia osia. Ulkoasun suunnittelussa suuresta kokonaisuudesta pienempään siirtyminen tarkoittaa sitä, että ensin valitaan suurpiirteisesti esimerkiksi navigaation paikka ja eri sisältöalueiden sijainnit. Suurpiirteisen suunnitelman pohjalta voidaan alkaa määrittää tarkempia mittoja, fontteja ja värejä. (Hiltunen ym. 2002, 90.)

## 2.2 Mitkä asiat vaikuttavat erityisesti mobiililaitteille suunnitteluun?

Kun uutta verkkopalvelua aletaan suunnitella, on jo olemassa jonkinlainen käsitys verkkopalvelun tarkoituksesta: ketkä luultavasti tulevat käyttämään palvelua, mihin tarpeeseen palvelu vastaa ja miksi palvelu ylipäänsä rakennetaan? Vaikeampi on kuitenkin määrittää, kuinka palvelua tullaan käyttämään ja millaisissa tilanteissa. Käyttökontekstilla tarkoitetaan sitä ympäristöä, jossa palvelua tullaan käyttämään ja sen määrittäminen esimerkiksi listaamalla erilaisia käyttöympäristölle tyypillisiä piirteitä voi auttaa palvelun suunnittelussa (Hiltunen ym. 2002, 21).

Hiltunen ym. (2002, 21) ovat jakaneet käyttökontekstin viiteen käytännönläheiseen näkökulmaan, jotka ovat käyttäjä, tehtävä sekä fyysiset, sosiaaliset ja teknologiset käyttöympäristön näkökohdat. Nämä viisi elementtiä vaikuttavat käyttäjän kokemukseen verkkopalvelusta, esimerkiksi kokemus voi olla hauska tai ärsyttävä riippuen onko käyttäjä kiireessä vai ei. Näiden elementtien tarkastelu mobiilipalvelua suunnitellessa on vielä tärkeämpää, kuin työpöytälaitteelle suunnitellessa. Verrattuna turvalliseen ja liikkumattomaan työpöytäympäristöön, mobiililaitteita ja -palveluja käytetään monissa eri konteksteissa eri käyttäjien toimesta, riippumatta ajasta ja paikasta, ja useilla eri laitetyypeillä. (Hiltunen ym. 2002, 21.)

Käyttäjät tulevat usein mobiiliverkkosivulle eri syystä, kuin miksi he käyttävät tavallista verkkosivua, eli mobiiliverkkopalvelussa halutaan suorittaa eri tehtävä. Esimerkkinä lentoyhtiön verkkosivut; jos käyttäjä on matkalla lentokentälle ja hän haluaa tarkistaa onko lento vielä aikataulussa, mikä on lähtöportin numero, hän ei halua selata koko sivuston läpi tarkastaakseen tarvitsemansa tiedon (Fling 2009, 66). Tiedon on löydyttävä nopeasti ja helposti.

Fyysinen käyttöympäristö asettaa ehkä suurimmat haasteet mobiililaitteille suunnitteluun. Etenkin pienten yksityiskohtien tarkastelu vaikeutuu, kun näyttö on pieni eikä pysy paikoillaan. Liian pieni fontti tekee tekstistä mahdottoman lukea ja liian pienet kuvat eivät aja asiaansa viestin välittäjinä. Kosketuskäyttöisissä laitteissa, kuten useissa älypuhelimissa ongelmaksi muodostuvat pienet linkit ja niihin osuminen, kun käytössä ei olekaan tarkka hiiren osoitin vaan suuri ja kömpelö sormi. Linkkien ja nappien ympärille tarvitaan tarpeeksi tilaa, jotta niihin on helppo osua. On siis suunniteltava isommin pienelle näytölle. Myös tekstirivien väliin tarvitaan lisää tilaa luettavuuden parantamiseksi, mutta toisaalta pienillä näytöillä ei ole ”ylimääräistä tilaa”.

Muita fyysisiä ominaisuuksia ovat esimerkiksi valon määrä, taustaäänet ja niiden voimakkuus, värinä, toiset ihmiset, lika, muu laitteisto ympärillä jne. Enimmäkseen nämä tekijät vaikuttavat laitteiden ja käyttöliittymän käyttöön. (Hiltunen ym. 2002, 29.)

Teknologiset näkökohdat käyttökontekstiin liittyvät siihen, millä teknologialla ja miten verkkosivut on rakennettu ja millainen käyttölaite on. Mobiililaitteiden verkkoyhteydet eivät usein ole yhtä nopeita tai vakaita, kuin pöytäkoneiden, ja niiden suorituskyky on usein muutenkin huonompi, mikä johtaa latausaikojen pitenemiseen. Eniten tämä vaikuttaa esimerkiksi kuvien ja muun sisällön lataamiseen mutta hitaampien latausaikojen tuomat haasteet tulee ottaa huomioon myös verkkosivun toimintoja, kuten animaatioita rakentaessa.

Kun tiedetään, millainen käyttökonteksti verkkopalveluilla ja etenkin mobiiliverkkopalveluilla on, on se otettava huomioon koko palvelun suunnittelu- ja toteutusprosessin ajan. Käytännössä tämä tarkoittaa sitä, että esimerkiksi palvelun käytettävyyttä tulisi testata oikeissa tilanteissa. Tällöin asiat ja mahdolliset ongelmat palvelun käytössä, jotka eivät olleet suunnittelijoiden tiedossa, voidaan ottaa huomioon ja parannuksia voidaan tehdä. (Hiltunen ym. 2002, 47.)

Mobiililaitteet tarjoavat myös ominaisuuksia, joita muista alustoista puuttuu. Tärkeimpiä ovat esimerkiksi ajan ja paikan vapaus ja liikkuvuus, reaaliaikainen informaatio, paikkatieto, kaikkialla oleva yhteys ja yhteydenpito ja näiden kaikkien yhdessä tuoma uusi käyttökonteksti. Kun nämä kaikki hyödynnetään verkkopalveluissa, ei mikään toinen alusta voi kilpailla palvelun hyödyllisyyden kanssa (Hiltunen ym. 2002, 47-48).

Laitteita on paljon, eikä käyttökokemus vaihtele pelkästään näytön koon, vaan myös laitteiden erilaisuuden takia. Mobiililaitteissa on paljon enemmän eri selaimia, kuin tietokoneilla, sillä mobiililaitteiden valmistajat usein kehittävät omat selaimensa tai muokkaavat toisen valmistajan selainta omaan käyttöönsä sopivaksi, oli se tarpeellista tai ei (Koch 2012, 11). Ei voi myöskään olettaa, että käyttäjillä olisi käytössään selaimen tai laitteen tuorein versio, vaan myös vanhemmat versiot on otettava huomioon suunnittelussa. Jotta käyttökokemus olisi kaikilla samanlainen, on suosituksi verkkosuunnittelun keinoksi noussut responsiivinen verkkosuunnittelu.

# 3 Responsiivinen verkkosuunnittelu

## 3.1 Responsiivisuuden perusajastus

Englannin kielen termi ”responsive” kääntyy suomessa reagoivaksi, vastaavaksi, sopeutuvaksi, mukautuvaksi. ”Responsive web design” – usein myös RWD – tarkoittaa reagoivan ja sopeutuvan verkon suunnittelua. Toisaalta responsiivinen verkkosuunnittelu on hyvin yksinkertainen käsite, sillä sen pystyy selittämään yhdellä lauseella ja samalla hieman hahmottamaan mihin tarkoitukseen se on kehitetty: Responsiivisen verkkosuunnittelun tavoitteena on suunnitella verkkopalvelun käyttäjälle paras mahdollinen käyttökokemus verkon selaamiseen käytetystä laitteesta riippumatta (Marcotte 2010).

Marcotten (2010, 9) määrityksen mukaan Responsive Designin kolme raaka-ainetta ovat joustava, ruudukkoon perustuva ulkoasu (responsive grid), joustavat kuvat (responsive images) sekä mediatyypin tunnistaminen (media queryt). Ennen kun voimme ymmärtää edellä mainittuja osa-alueita, on aloitettava perusasioista, sillä responsiivisuus lähtee liikkeelle joustavasta perustasta ja CSS:n perusyksiköiden, kuten pikselien uudelleen ajattelusta. Responsiivista verkkosuunnittelua voi lähestyä monelta kannalta, mutta aluksi tärkeintä on ymmärtää kolme responsiivisen verkkosuunnittelun peruspilaria ja niiden käyttöajatus pääpiirteittäin.

## 3.2 Joustava teksti

Pikseleillä mitattujen leveyksien sijaan responsiivisessa verkkosuunnittelussa käytetään mittayksikköä joustavaa ja suhteellista ”em” -yksikköä. 1 em vastaa aina nykyistä fonttikokoa, esimerkiksi 16 pikseliä, jolloin 2 em on kaksi kertaa fonttikoko eli 32 pikseliä. Em-yksikkö mahdollistaa sen, että voimme tulkita pikseleihin perustuvat arvot suhteessa fonttikokoon ja samalla suhteessa toisiinsa ja rakentaa CSS-tyylitiedostot näiden mittojen pohjalta.

Em-yksiköt perustuvat yksinkertaiseen laskukaavaan:

```
kohde ÷ konteksti = tulos
```

Otamme halutun *kohteen* fonttikoon ja jaamme sen ympäröivän elementin, *kontekstin*, fonttikoolla. *Tulos* on haluamamme fonttikoko ilmaistuna em-yksikköinä. Jos esimerkiksi olemme toteuttamassa ulkoasua, jossa otsikon fonttikoko on 24 pikseliä ja sitä ympäröivän body-elementin fonttikoko on 16 pikseliä, voimme lisätä molemmat luvut suoraan kaavaan.

```
24 ÷ 16 = 1,5
```

24 pikseliä on siis 1,5 kertaa suurempi kuin 16 pikseliä, joten fonttikooksi otsikolle tulee 1.5 emiä.

On kuitenkin muistettava, että konteksti voi muuttua ja että em:llä viitataan aina ympäröivän elementin fonttikokoon. Jos yllä mainitun otsikko-elementin sisällä on tekstiä, jonka halutaan olevan pienempi, esimerkiksi 11 pikseliä, on sen koko laskettava uudestaan:

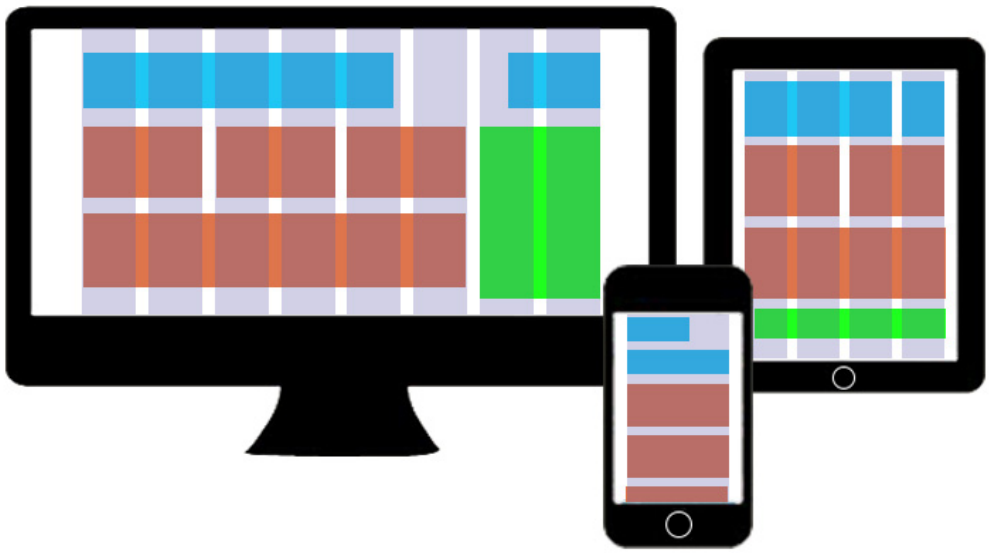
```
11 ÷ 24 = 0,45833333333
```

Vaikka luku ei näytäkään tyylikkäältä ja se olisi helppo pyöristää 0.46 emiin, on muistettava, että 0,45833333333 vastaa täydellisesti haluttua fonttikokoa suhteellisesti ilmaistuna eikä luvun ymmärtäminen tuota selaimelle vaikeutta. Mitä tarkemman luvun annamme selaimelle, sitä paremman lopputuloksen saamme. (Marcotte 2010, 22.)

### 3.3 Joustava ruudukko

Verkkosivustot rakennetaan usein ruudukon (engl. grid) päälle, sillä ruudukot auttavat verkkosivun sisältöjen jäsennyksessä. Ruudukko jakaa verkkosivun sarakkeisiin ja riveihin, joihin kuvat ja tekstisisällöt pystytään asettelemaan helposti. Ruudukkoja käytettiin ja käytetään yhä printtimedioiden, kuten lehtien jä-

sentämisessä mutta tapa on omaksuttu myös verkkosuunnittelun piiriin. Verkkosivustojen ruudukot toteutetaan jakamalla ulkoasu jonkin pikselimäärän levyisiksi sarakkeiksi, esimerkiksi 100 pikseliä. Sarakkeita voi olla kuinka monta tahansa, esimerkiksi 10, jolloin koko verkkosivun leveydeksi tulee 1 000 pikseliä ja niistä voidaan jakaa sisältöalueita, toinen esimerkiksi leveydeltään 300 pikseliä ja toinen 700 pikseliä. Kuva 1 on esitetty yksi tapa jakaa verkkosivu sarakkeisiin.



Kuva 1. Joustava ruudukko (responsive grid).

Responsiivinen ruudukko sen sijaan jakaa verkkosivun pikselintarkkojen leveyksien sijaan suhteessa selainikkunan leveyteen. Mittayksikkönä käytetään prosentteja ja kuten suhteellisesti määritellyissä fonteissa, myös elementtien leveyksien määrittämisessä leveydet ovat aina suhteessa ympäröivän elementin leveyteen. Kun body-elementin leveys on 100 % eli koko selainikkunan leveys, voidaan verkkosivun sisältöjen leveydeksi asentaa esimerkiksi 80 %, jolloin sisältöjen pikselein mitattu leveys muuttuu aina, kun selainikkunan leveys muuttuu mutta leveys pysyy aina samana suhteessa selainikkunan leveyteen. Jos sisältöä ympäröivä elementti keskitetään, kuten usein tehdään, jää sisältöjen molemmille puolille 10 %:n levyiset marginaalit.

Jos sisällöt halutaan jakaa esimerkiksi pääalueeseen ja oikeaan sivupalkkiin, asetetaan alueiden leveydet jälleen suhteessa niitä ympäröivään elementtiin: pääalueen leveys on esimerkiksi 60 %, sen oikealle puolelle tulee 5 %:n levyinen marginaali ja oikean sivupalkin leveydeksi asetetaan 35 %, jolloin alueiden yhteenlaskettu summa on 100 % eli koko niitä ympäröivän alueen leveys.

Prosenttien laskemisessa käytetään samaa kaavaa kuin fonttikokojen laskemisessa:

```
kohde ÷ konteksti = tulos
```

Kaavaa voi hyödyntää etenkin tapauksissa, joissa ulkoasusuunnitelma on tehty kuvankäsittely- tai muussa grafiikkaohjelmassa ja se halutaan toteuttaa responsiivisesti. Kaavan avulla lasketaan sisältöalueiden ja niitä ympäröivien marginaalien (engl. margins) ja täytteiden (engl. padding) prosentuaaliset leveydet mutta näitä laskettaessa on huomioitava kaksi asiaa:

1. Kun asetetaan joustavia *marginaaleja (margin)* elementille, on kontekstina elementtiä ympäröivän elementin leveys.
2. Kun asetetaan joustavaa *täytettä (padding)*, on konteksti itse elementin leveys. (Marcotte 2010, 35.)

Laskukaavaa käytetään samalla tavalla myös tapauksissa, joissa marginaali on negatiivinen (Marcotte 2010, 39).

### 3.4 Joustavat kuvat

Kun 600 pikselin levyinen kuva asetetaan 300 pikselin levyisen elementin sisään ilman mitään CSS-sääntöjä, kuva tulee ympäröivän elementin reunojen yli. Jos kuva muokataan kuvankäsittelyohjelmassa 300 pikselin levyiseksi, se asettuu täydellisesti sitä ympäröivän elementin sisään, kunnes elementin leveys muuttuu. Siinä missä HTML:n elementtien on oltava joustavia, on myös kuvien mukauduttava ulkoasun mukana.

Kuvien asettaminen joustaviksi tapahtuu yhdellä CSS-säännöllä:

```
img {
     max-width: 100%;
}
```

Kyseinen sääntö määrää, ettei kuva saa ylittää sitä ympäröivän elementin leveyttä. Mikäli kuvaa ympäröivä elementti on leveämpi kuin kuvan oma leveys, kuva asettuu elementin sisään sellaisenaan, mutta jos ympäröivä elementti on kapeampi kuin kuva, yllä mainittu sääntö pakottaa kuvan asettumaan ympäröivän elementin levyiseksi. Selaimet osaavat myös skaalata kuvan oikeassa kuvasuhteessa ja sääntöä voi käyttää myös muihin elementteihin, kuten videoon, ”embed” ja ”object” -elementteihin. (Marcotte 2010, 45.)

On huomionarvoista, että Internet Explorer 6 ja sitä vanhemmat versiot eivät tue max-width -ominaisuutta. Ongelman voi korjata käyttämällä esimerkiksi JavaScript-pohjaista ratkaisua ympäröivän elementin koon tunnistamiseen ja kuvan asettamiseen siihen sopivaksi, mikäli kuvan leveys ylittää elementin leveyden. Voidaan myös ladata kyseiselle selaimelle oman CSS-tyylitiedostonsa, jossa kuvan leveydeksi asetetaan 100 prosenttia.

Verkko koostuu suuresta määrästä kuvia ja usein niiden lataaminen vie suurimman osan sivun latausajasta. Kuviin tuleekin kiinnittää erityisesti huomiota responsiivisessa verkkosuunnittelussa, sillä kuvien toimintaan liittyy monta tekijää, kuten näyttöjen kasvavat resoluutiot ja vaihtelevat verkkonopeudet. (Frost 2012.) Käyttämällä responsiivisen verkkopalvelun kuvissa sääntöä *max-width* voidaan useimmiten olettaa, että kuva skaalautuu hyvin riippumatta selainnäkymän koosta. 1 000 pikselin levyisen kuvan lataaminen 480 pikselin levyiselle laitteelle vie kuitenkin turhaan verkkokaistaa ja pidentää verkkosivun latausaikaa. Vaikka modernit selaimet pystyvätkin skaalaamaan kuvan riittävän hyvin sen laatua huonontamatta, rasittaa ylimääräinen latausaika verkkosivun toimintaa, varsinkin jos isoja kuvia on sivulla useita.

Verkkosivun kuvat pystytään lataamaan myös erilaisten ehtojen avulla. Näyttöjen koot ja pikselitarkkuudet voidaan tunnistaa esimerkiksi mediatyypin kyselyn (media queries) tai JavaScriptin avulla ja ladata sopivan kokoinen kuva kullekin näytölle. Jos eri kuvaversiot määritetään media queryjen sisällä elementtien taustakuvina, selaimet pystyvät usein hyvin varmistamaan, ettei ylimääräisiä kuvia ladata (Grigsby 2010, Kadlec 2012). Mobiililähtöistä toteuttamismenetelmää voidaan soveltaa myös taustakuvien lataamiseen ja oletuksena ladata pienempi taustakuva, mutta optimoida taustakuvat suuremmille resoluutioille lataamalla suuremmat kuvat. (Frost 2012.)

### 3.5 Media Queries

CSS:n käsitteistössä sana ”media” viittaa yleensä laitteen tai muun yleisen esitysalustan ominaisuuksiin ja mediatyypin kyselyn (media query) avulla voidaan tietyt CSS-säännöt kohdistaa tietyille mediatyypeille. Taustalla on se, että erilaisilla laitteilla saatetaan tarvita hyvinkin erilaisia asetteluja ja muuta muotoilua, esimerkiksi tulostettavan version asettelu verrattuna näytöllä näkyvään asetteluun. (Korpela 2008, 353.) HTML4:n ja CSS2:n tukemat mediasta riippuvat tyylitiedostot mahdollistavat sen, että esimerkiksi HTML-tiedosto voi käyttää päätteetöntä -fonttia näytöillä ja päätteellistä fonttia tulostusversiossa. Mediatyyppejä ovat

- aural (ääni)
- braille (pistekirjoitus)
- handheld (kannettava)
- print (tulostus)
- projection (projektio)
- screen (näyttö)
- tty (faksi)
- tv

Pelkän mediatyyppin tarkastamisen lisäksi voidaan media querylle asettaa muitakin ehtoja, joiden perusteella tietyt CSS-säännöt voidaan ladata. Voidaan tar-

kastella, mitä fyysisiä ominaisuuksia on laitteella ja selaimella, johon verkkosivun sisältö ladataan (Marcotte 2010, 74). Tarkempia ominaisuuksia mediatyypeille ovat esimerkiksi korkeus, leveys, väri ja orientaatio.

Media queryjen tarkoitus responsiivisen verkkosuunnittelun osa-tekijänä on korjata ulkoasu silloin, kun joustavaan ruudukkoon rakennettu ulkoasu rikkoutuu. Esimerkiksi tekstisarake voi kaventua liian pieneksi tai navigaation linkit eivät asetu toimivalla tavalla, kun näyttö on kapea.

```
@media screen and (min-width: 768px) {
     body {
         font-size: 100%;
     }
}
```

Media queryt koostuvat kahdesta osasta. Ensin nimetään mediatyyppi (screen), sen jälkeen kerrotaan itse *query,* kysely/tiedustelu, jolle määritellään ominaisuus (min-width) ja arvo (768 px) (Marcotte 2010, 74). Jos laite tai selain, jolla verkkosivua tarkastellaan, vastaa näihin kahteen ehtoon (laite on "näyttö" ja selainikkuna on vähintään 768 pikselin levyinen), HTML-tiedostossa esitetään kyselyn sisällä olevat CSS-säännöt. Jos ei, selain jättää kyseiset säännöt huomiotta.

Media queryjä voi käyttää HTML-tiedostossa kolmella tavalla. Yllä olevan esimerkin media querysta voi kirjoittaa suoraan CSS-tiedostoon ja samaan tiedostoon voi lisätä muitakin media queryja erilaisilla ehdoilla. Kyseinen tyylitiedosto linkitetään HTML-tiedostoon *<link />* -tagin avulla. Toinen tapa on asettaa media query tyylitiedoston linkityksen yhteyteen:

```
<link rel="stylesheet" href="tyyli-yli-768.css" me-
     dia="screen and (min-width: 768px)" />
```

Yllä oleva esimerkki asettaa tyylitiedoston lataamiselle ehdon ja kyseinen tiedosto *tyyli-yli-768.css* ladataan vain, jos media-attribuutin hakukriteerit täyttyvät. Kolmas tapa on käyttää CSS-tiedoston sisällä @import -toimintoa.

```
@import url("tyyli-yli-768.css") screen and (min-width:
     768px);
```

Jokainen yllä mainituista keinoista esittää CSS-säännöt vain, jos media queryn kriteerit täyttyvät. Ehkä yleisin käytössä oleva keino on ensimmäisenä mainittu *@media*, sillä sen avulla kaikki tyylimääritykset voidaan pitää yhdessä tiedostossa ja selaimen ei tarvitse hakea kuin yksi tiedosto palvelimelta (Marcotte 2010, 75).

Media queryjen käyttötarkoitus on siis korjata ulkoasua silloin, kun selainikkunan koon muuttaminen rikkoo ulkoasua mutta media queryt mahdollistavat myös sisältöjen optimoinnin eri näyttöversioille sopivaksi. Voidaan räätälöidä erilaiset versiot ulkoasusta eri resoluutioisille näytöille (Marcotte 2010, 79). Usein media queryjen välipisteet (breakpoint) määritetään kolmen yleisimmän laiteleveyden mukaan; tietokoneet, tablet-laitteet ja älypuhelimet. Vaikka laitteitten leveydet vaihtelevat ja erilaisia näyttökokoja on useita, voidaan esittää jonkinlainen arvio näytön yleisimmästä koosta sen perusteella, mitä laitteita on myyty eniten.

Yhtenä hyvänä suuntaviivana eniten käytetyistä laiteleveyksistä ovat Applen laitteet iPhone ja iPad. Andy Clarken "Hardboiled CSS3 Media Queries" (2010) on eräs malli kuinka media queryjen välipisteet usein asetetaan:

```
/* Smartphones (portrait and landscape) ----------- */
@media only screen and (min-device-width: 320px) and (max-
     device-width: 480px) { … }

/* Smartphones (landscape) ----------- */
@media only screen and (min-width: 321px) { … }

/* Smartphones (portrait) ----------- */
@media only screen and (max-width: 320px) { … }

/* iPads (portrait and landscape) ----------- */
@media only screen and (min-device-width: 768px) and (max-
     device-width: 1024px) { … }
```

```
/* iPads (landscape) ----------- */
@media only screen and (min-device-width: 768px) and (max-
     device-width: 1024px) and (orientation: landscape)
     { … }

/* iPads (portrait) ----------- */
@media only screen and (min-device-width: 768px) and (max-
     device-width: 1024px) and (orientation: portrait) { … }

/* Desktops and laptops ----------- */
@media only screen and (min-width: 1224px) { … }

/* Large screens ----------- */
@media only screen and (min-width: 1824px) { … }

/* iPhone 4 and high pixel ratio devices ----------- */
@media
only screen and (-webkit-min-device-pixel-ratio : 1.5),
only screen and (min-device-pixel-ratio : 1.5) { … }
```

Media queryjen välipisteet voidaan määrittää laitekokojen ehdoilla tai ulkoasun ehdoilla. Missä kohdassa ulkoasu rikkoutuu riippuu kokonaan siitä, millaisesta ulkoasusta on kyse, ja jokaisella ulkoasulla on omat ongelmakohtansa.

# 4 Responsiivisuuden rakentaminen

## 4.1 Suunnittelutapojen uudelleen ajattelu

Responsiivinen verkkosuunnittelu ei ole vain elementtien uudelleen asettelua tai automaattisesti skaalautuvia kuvia, vaan pikemminkin kokonaan uusi tapa ajatella verkkosuunnittelua (Knight 2011). Se tarkoittaa, että verkkosuunnittelijat joutuvat muuttamaan totuttuja suunnittelutapojaan, sillä samat säännöt eivät päde leveydeltään kiinteisiin verkkosivuihin ja joustaviin, responsiivisiin sivuihin: suunnittelun tavoitteet ovat muuttuneet. Sen sijaan, että suunniteltaisiin yksi tietyn kokoinen ulkoasu, joudutaan suunnittelemaan kolme tai neljä erilaista versiota eri näyttökokojen vuoksi. Joudutaan myös miettimään sisältöelementtien tärkeysjärjestystä, navigaatiolinkkien käyttäytymistä pienemmässä tilassa, luettavuutta ja toimivuutta (Clements 2012). Suunnittelijat joutuvat tekemään

paljon enemmän päätöksiä liittyen sivuston toimintaan, koska jos jokin tapa toimii tietokoneen ruudulla, se ei välttämättä toimi niin hyvin älypuhelimen näytöllä.

Verkkosuunnittelijat ovat kehittäneet useita erilaisia työtapoja responsiivisen verkkosivuston suunnitteluun ja rakentamiseen. Lähtökohtana suunnittelutavan valinnalle voi olla esimerkiksi, mihin käyttötarkoitukseen verkkosivusto kehitetään, mitkä ovat sen luultavasti yleisimmät käyttötilanteet tai mikä on verkkopalvelun viesti. Tärkeimpänä lähtökohtana pidetään yleisesti sisältöä. ”Content is King” on lausahdus, jonka taustalla on ajatus siitä, ettei verkkosivu ole menestynyt, jollei sen sisältö ole onnistunutta tai sitä ei ole onnistuttu välittämään tehokkaasti ja mielenkiintoisesti (Callan 2009).

Valintaan vaikuttavat myös verkkosuunnittelijan tottumukset ja taidot. Jos verkkosuunnittelija on tottunut aloittamaan suunnitteluprosessin Photoshopissa, voi sitä edelleen käyttää lähtökohtana, mutta graafista suunnittelua toteutettaessa on myös muistettava suunniteltavan verkkosivun joustava luonne ja skaalautuvuus.

## 4.2 Sisällön priorisointi

Käyttäjät tulevat verkkosivulle sisällön takia. Siksi ensimmäiseksi tavoitteeksi verkkosivuston suunnittelemiselle tulisikin asettaa sisällön sopiva ja luettava esittäminen. (Callahan 2011.) Sisällöllä voidaan tarkoittaa niin tekstisisältöä, kuvia kuin videoitakin. Sisällön priorisointi korostuu etenkin mobiililaitteille suunnitellessa, sillä mobiililaitteita käytetään usein liikkeessä, kiireessä ja niiden latausajat ovat hitaammat, jolloin on entistä tärkeämpää tarjota käyttäjälle hänen etsimänsä sisältö helposti.

Callahanin (2011) tapa lähestyä sisältölähtöistä suunnittelua on rakentaa valmiina olevien sisältöjen perusteella prototyyppi, jossa sisällöt on aseteltu sopiville paikoille ilman värejä, kuvia tai ikoneita. Ainoa asia, johon kiinnitetään huomiota, on fonttityyppi, fontin koko, palstan leveydet ja niiden asettelu. Suunnittelu aloitetaan grafiikkaohjelmassa, kuten Photoshopissa ja tämän visuaalisen

suunnitelman pohjalta koodataan prototyyppi käyttäen pohjana joustavaa ruudukkoa. (Callahan 2010.)

Clemens (2012) lisää sisältöprototyyppiin mobiililähtöisen näkökulman. Koska mobiilinäkymä on kapea ja tekstit joudutaan usein asettelemaan lineaarisesti ja yhteen palstaan, on tärkeää esittää tärkein sisältö ensimmäisenä. Tällöin sisältöjen hierarkia nousee tärkeäksi osaksi verkkosivusuunnitelmaa. Kun sisältöjen tärkeysjärjestys tiedetään, pystytään paremmin luomaan visuaalinen ilme niin tietokoneen näytöille, kuin mobiililaitteillekin (Clemens 2012).

Kummassakin tapauksessa pelkkää sisältöä esittävän prototyypin avulla on helppo etsiä media queryjen välipisteet suurentamalla ja pienentämällä selainikkunaa. Tätä menetelmää käyttämällä sisältö kertoo, missä kohtaa joustava ruudukko tarvitsee korjausta. Muutokset sisältöjen asettelussa tulee kuitenkin muistaa tehdä sisältöjen luettavuuteen keskittyen. (Callahan 2011.)

### 4.3 Sivuston muuttaminen responsiiviseksi

Responsiivisuus on helppo tuoda myös vanhemmille sivustoille tai uudemmille sivustoille, joita ei alun perin ole suunniteltu mobiililaitteille. Muutos kannattaa toteuttaa käytettävyyden takia, sillä jos suureen leveyteen tarkoitettu sivusto kutistetaan puhelimen näytölle, voi tekstin lukeminen käydä mahdottomaksi ja linkkeihin osuminen vaikeaksi. Helpoin tapa lähestyä muutosta on nostaa tärkein sisältö keskiöön. Kun sivuston tärkeimmät elementit tiedetään, voidaan alkaa jäsentää sivuston asettelua.

Responsiivisen verkkopalvelun luonteeseen kuuluu, ettei se ole zoomattavissa tai pienennettävissä kosketuseleillä (zoom, pinch) vaan fonttien ja elementtien koko on muutettu valmiiksi. Kun verkkosivun head-osioon lisätään seuraavat rivit:

```
<meta name="viewport" content="width=device-width,initial-
    scale=1,maximum-scale=1,user-scalable=no">
```

asetetaan sivun kuvasuhde 1x1:n ja poistetaan oletuksena oleva toiminto, jolla sivustoa voi zoomata lähemmäksi tai loitontaa nipistämällä (Jung 2014).

Lisäämällä media queryja sivuston asettelua pystytään muuttamaan eri kokoihin sopivaksi. Yleensä mobiililaitteilla sivuston elementtien leveydet asetetaan 100 %:iin, jolloin sisällöt pääsevät täyttämään kaiken vapaana olevan tilan. Jos esimerkiksi sivusto koostuu pääalueesta ja toissijaisesta alueesta, voidaan molemmat laittaa täyteen leveyteen siten, että toissijainen alue siirtyy pääalueen alle. Kohdat, joihin media queryjen pisteet tulevat riippuvat tietenkin sivustosta, mutta yleisimpiä kohtia asettelun muutokseen ovat tablet-laitteen ja älypuhelinten koot.

Keskeistä responsiiviselle ulkoasulle ovat kuvat ja videot sekä niiden asettuminen. Responsiivisten kuvien oleellisin asia on se, että ne skaalautuvat niitä ympäröivän elementin kokoon sitä ylittämättä, eli jos kuvien koko laitetaan enimmillään sataan prosenttiin, kuten kohdassa 3.4 on esitetty, kuvat sopeutuvat sivuston asetteluun (Jung 2014).

# 5 Responsiivisuus mobiilisovelluksissa

## 5.1 Mobiilisovellusten rakentaminen verkkoteknologioilla

Yritysten ja verkkokehittäjien haasteena on tuottaa palveluita ja tuoda ne mahdollisimman monen saataville. Mobiilisovellusten rakentaminen onnistuu nykyään helposti verkkoteknologioita käyttämällä. Varsinkin sovellukset, joissa on paljon luettavaa ja katseltavaa sisältöä, pystyvät hyödyntämään verkkoteknologioita.

Yksi suosituimmista ympäristöistä rakentaa HTML5-pohjaisia sovelluksia on PhoneGap, joka mahdollistaa mobiilisovelluksen rakentamisen Androidille, iOS:lle, Windows Phonelle, Blackberrylle, Symbianille, Badalle sekä WebOS:lle. PhoneGap kääntää HTML:llä rakennetun sovelluksen näille alustoille ja mahdol-

listaa laitteiden omien ominaisuuksien, kuten kameran ja GPS-sijainnin hyödyntämisen. (Phonegap 2013.) Toinen suosittu ympäristö on Sencha Touch, joka tarjoaa työkalut sovellusten käyttöliittymien rakentamiseen sekä niiden pakkaamiseen (Sencha 2014).

Laitteiden kirjo on nykypäivänä valtava ja mobiilisovelluksen rakentamisen näille kaikille laitteille luulisi tietävän valtavasti lisätöitä. Laitekirjon ongelmaan ratkaisu löytyy kuitenkin responsiivisesta verkkosuunnittelusta. PhoneGap ja Sencha Touch rakentavat sovelluksen verkkoselainnäkymään; toisin sanoen sovellus hyödyntää verkkoselaimen moottoria rakentaessaan laitteen näytön levyisen ja korkuisen näkymän, johon sovellus rakentuu (Phonegap 2013). Tällöin verkkoteknologioilla rakennettu sovellus näkyy ja käyttäytyy kuin verkkosivu, ollen kuitenkin itsenäinen sovellus.

Responsiivisuus mobiilisovelluksissa mahdollistaa sovelluksen skaalautuvuuden eri laitteisiin. Koska sovelluksen leveyden voi asettaa sataan prosenttiin, sovellus skaalautuu aina laitteen levyiseksi. Toisaalta mobiilisovellusten ominaisuuksiin usein kuuluu, ettei niitä tarvitse selata pystysuunnassa – riippuen tietysti sovelluksesta – jolloin sovellus on joskus saatava skaalautumaan myös pystysuunnassa menettämättä esimerkiksi kuvien kuvasuhdetta ja elementtien asettelun menemättä sekaisin. Media queryjen avulla on mahdollista tunnistaa myös, onko laite pysty- vai vaakasuunnassa.

Jos mobiilisovellus rakennetaan Internet-yhteyttä käyttämättömäksi offline-sovellukseksi, on kaikki sovelluksen sisällöt pakattava sovelluksen sisälle. Tällöin esimerkiksi responsiivisten kuvien käyttäminen rajoittuu kuvien skaalautumiseen, sillä erikokoisten kuvaversioiden lataaminen sovelluksen sisälle ei ole kannattavaa sovelluskoon kasvaessa turhaan. Vaikka retina-näyttöiset laitteet vaativat resoluutioltaan isot kuvat, niiden skaalaus pienemmäksi onnistuu laitteilta melko vaivattomasti.

Yksi suurimmista eduista mobiilisovellusten rakentamisessa HTML5:llä on, että niitä on mahdollista kehittää tietokoneella selaimessa. Selaimessa rakentaminen ja testaaminen sekä selaimilla että erilaisilla laitteiden ominaisuuksia mallin-

tavilla emulaattoreilla mahdollistaa nopean työtahdin mutta oikeassa kosketusnäyttöisessä laitteessa testaamista selaintestaus ei voi tietenkään korvata.

## 5.2 Laitteen tunnistaminen

Mobiililaitteet voi nopeasti jakaa kahteen pääryhmään: älypuhelimiin ja taulutietokoneisiin. Applen tuotteilla jako on yksinkertainen iPhonen, iPadin ja uudehkon iPad minin välillä, mutta Android-laitteiden koot vaihtelevat enemmän ja jako kahteen ryhmään hankaloituu. Esimerkiksi osa Samsungin Galaxy Noteista on määritelty ”älypuhelin-tabletiksi”, joka on hieman normaalia älypuhelinta isompi älypuhelin (Wikipedia 2013c).

Androidin kehittäjädokumentoinnissa laitteet jaetaan neljään eri ryhmään: *xlarge*, *large*, *normal* ja *small*. Laitekoot, joita sovellus tukee, voidaan määrätä AndroidManifest-ohjetiedostossa, jolloin vain tuetut laitteet voivat ladata sovelluksen. (Google Android Developer Documentation 2013.) Samalla tavalla iOS:lle pystytään määrittämään, tukeeko sovellus iPadeja vai iPhonea määrittelemällä laitetuki Info.plist-tiedostossa.

Jos verkkoteknologioilla rakennetun sovelluksen ei haluta tukevan pelkästään älypuhelinta tai tablettia, onnistuu leveyden tunnistaminen ja siihen mukautuminen media queryjen avulla samalla tavalla kuin verkkosivuja rakennettaessa. Osa responsiivisuudesta pystytään toteuttamaan JavaScriptiä hyödyntämällä, esimerkiksi laitteen korkeuden hakeminen JavaScriptin avulla helpottaa sovelluksen skaalaamista ja kuvasuhteen säilyttämisessä. Yksinkertaisin JavaScript koodi laitteen *näkymän* (viewport) leveyden ja korkeuden tunnistamiseen on seuraava.:

```
var viewportWidth  = document.documentElement.clientWidth;
var viewportHeight = document.documentElement.clientHeight;
// jQuery
jQuery(window).width();
```

Jos halutaan tunnistaa *laitteen* leveys ja korkeus, voidaan käyttää seuraavaa koodia:

```
var deviceWidth  = window.screen.width;
var deviceHeight  = window.screen.height;

var availWidth = window.screen.availWidth;
var availHeight = window.screen.availHeight;
```

Kaksi alinta hakevat laitteen koon, josta vähennetään käyttöjärjestelmän tehtäväpalkki. (Van Etten 2012.)

Laitteen tunnistamiseen voi media queryjen ja JavaScript-koodin lisäksi käyttää valmiita, edellä mainittuja keinoja hyödyntäviä koodikirjastoja, kuten *Device.js* (Smus 2014) tai *Modernizr*. Ne ovat hyväksi todettuja ja varmempia tunnistamaan laitteen tyypin ja ominaisuudet (Bhanot 2012).

### 5.3 Kehitystyökaluja

Suurin etu verkkoteknologioilla rakennettavissa sovelluksissa on, että niitä pystytään kehittämään selaimessa. Sovellukset voidaan rakentaa tavallisen verkkosivun tapaan asettamalla selaimen mobiililaitteelle ominaiseen leveyteen ja korkeuteen. Sovellusten rakentamiseen voi myös käyttää mitä tahansa verkkoeditoria. Suosituimpia työkaluja ovat kenties Adoben Dreamweaver, Applen kehitystyökalu Xcode tai Androidin suosittelema Eclipse. Kaksi viimeistä ovat hyödyllisiä, sillä niihin saa yhdistettyä emulaattorit.

Google Chrome selain on erittäin kehittäjäystävällinen ja siihen on saatavilla hyödyllisiä laajennuksia, joiden avulla mobiilikehittäminen helpottuu. Chromen omat kehitystyökalut mahdollistavat koodin virheiden etsimisen, tyylien muokkaamisen ja verkkosivun ominaisuuksien, kuten latausaikojen tarkastelun (Google Developers 2013). Chromen laajennukset tekevät testaamisesta helpompaa. Selainikkunan asettaminen mobiililaitteen, esimerkiksi iPadin leveyteen ja korkeuteen onnistuu esimerkiksi laajennuksella Window Resizer (Google Chrome Web Store 2013a).

PhoneGap -sovelluksien rakentamiseen hyvä apuvälinen on Chromen laajennus Ripple (Google Chrome Web Store 2013b), jonka avulla pystytään pyörittämään PhoneGap-sovelluksia selaimessa ja jäljittelemään erilaisia mobiililaitteiden ominaisuuksia, kuten kameraa tai GPS:ää.

Selaimessa kehittäminen on hyvä lähtökohta sovellusten rakentamiselle, mutta paremman kuvan sovelluksen toiminnasta saa, kun sitä testaa emulaattoreissa. Emulaattorit mallintavat laitteiden toimintaa ja käyttäytymistä paremmin ja kertovat, kuinka sovellus tulee suurin piirtein toimimaan itse laitteessa. Yleensä käyttöjärjestelmien valmistajat (Android, Apple, Windows) tarjoavat kehitystyökalujensa mukana omat emulaattorit, joilla sovelluksia voi testata, mutta löytyy myös erilaisia ja kevyempiä laitteen toimintaa mallintavia simulaattoreita. Kattava lista emulaattoreista ja simulaattoreista löytyy esimerkiksi Breaking the Mobile Web -blogista (Firtman 2014).

# 6 Oma työ

## 6.1 Vyöhyketerapia-sovellus

Testatakseni responsiivisuutta mobiilisovellusten rakentamisessa, käänsin aikaisemmin rakentamani iPad-sovelluksen skaalautuvaksi Android-sovellukseksi. Sovelluksen nimi on Vyöhyketerapia. Vyöhyketerapia on uskomushoito, jonka takana on ajatus siitä, että koko keho heijastuu vyöhyketerapiapisteinä eli heijasteina ihmisen jalkaterään. Heijasteiden käsittelyn tavoitteena on käynnistää kehon oma parantumisprosessi. Esimerkiksi painelemalla aineenvaihduntaan vaikuttavia heijastepisteitä jalkapöydässä, voidaan yrittää korjata ihmisen aineenvaihdunnan ongelmia.

Sovellus sisältää aloitus- ja lisätieto-sivujen lisäksi 11 opastekuvaa, joiden avulla on tarkoitus paikallistaa vyöhyketerapiapisteitä jalkaterästä. Jokaisen pisteen kohdalla on kerrottu tarkemmin, kuinka pistettä tulee käsitellä ja mihin hoito vai-

kuttaa. Sovelluksen aloitussivu toimii sovelluksen navigaationa (Kuva 2). Kuvia klikkaamalla pääsee siirtymään haluttuun kuvaan, joka liukuu esiin sivun oikeasta laidasta. Kuvien välillä navigointi tapahtuu pyyhkäisemällä kuvan päällä oikealle tai vasemmalle, jolloin pääsee siirtymään edelliseen tai seuraavaan kuvaan; tai palaamalla aloitussivulle.

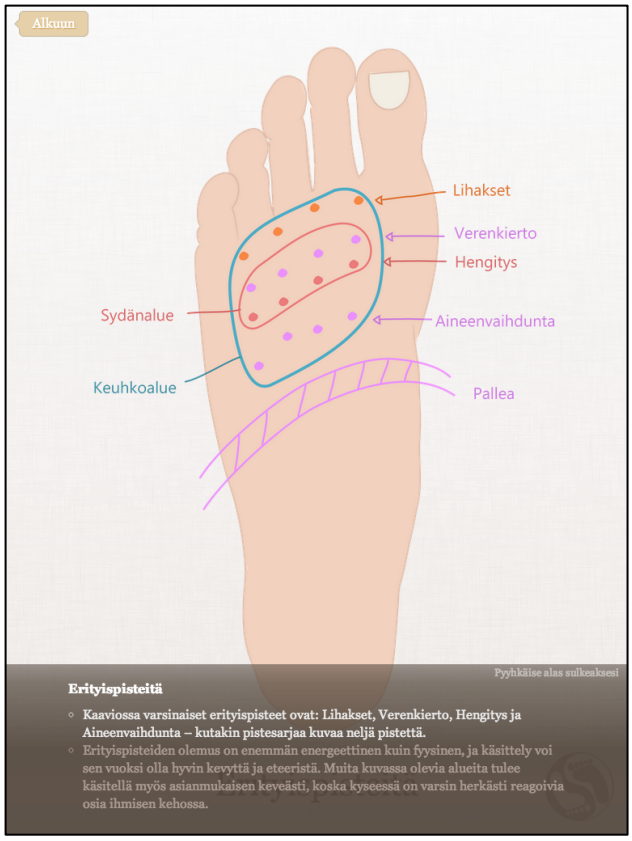

Kuva 2. Sovelluksen aloitussivu.

Sovelluksen ensimmäinen versio on rakennettu iPadille kokoon 1 024x768 pikseliä ja siinä olevat kuvat on toteutettu tukemaan Retina-näyttöjä siten, että koko näytön täyttävät kuvat ovat kokoa 2 048x1 536 px ja ne on skaalattu kokoon 1 024x768. Jokaisen kuvan päällä on niin sanottuja hotspot-linkkejä, joiden sijainti on määritetty top- ja left -arvojen avulla. Jokaisella hotspot-linkillä on uniikki id, jonka perusteella kyseisen heijastepisteen ohjetekstit osataan asettaa ohjeille varattuun osioon:

```
#jalka1_lannenikamat {
    top: 520px;
    left: 364px;
    width: 131px;
}
```

Jokaisella pisteellä oli oma yksilöllinen id:nsä siksi, että niiden tunnistaminen olisi helpompaa; pisteet eivät asetu loogiseen järjestykseen kuvassa, eikä niitä siksi voi lukea tarkasti ylhäältä alas tai vasemmalta oikealle. Oikea teksti tulee kuitenkin saada oikean linkin taakse, joten virheiden minimoimiseksi käytetään yksilöllisiä tunnisteita.

Jokaisen kuvan alla on alue, johon heijastepisteen kuvausteksti tulee. Alue on sijoitettu y-akselilla kohtaan 768 pikseliä, eli aivan näkymän alalaidan alle. Kun jotakin heijastepistettä klikataan, sovellus asettaa tekstin alueen sisään, laskee alueen korkeuden ja siirtyy vastaavan verran ylöspäin sivun alalaidasta (Kuva 3).




Kuva 3. Opastekuva Erityispisteistä.

Sovelluksen pakkaamiseen on käytetty Adoben PhoneGap Buildia. Se on pilvipalvelu, joka pakkaa sovellukset vaivattomasti ja mahdollistaa saman koodin kääntämisen useille mobiilialustoille. Kun palveluun lataa sovellukseen kuuluvat HTML5, CSS ja JavaScript-tiedostot sekä kuvat ja muut tiedostot, ne pakataan sovellukseksi automaattisesti ja sovellus on valmis asennettavaksi laitteelle. (Adobe Systems Incorporated 2013.)

## 6.2 Projektin eteneminen

Sovelluksen kääntämisen prosessi eteni loogisesti suunnittelusta toteutukseen. Sovellusversioinnin alkukartoituksessa selvisi, että haasteita sovelluksen kääntämisessä aiheuttaisivat Android-laitteiden suuri määrä ja sen mukana eri näyttökokoihin skaalautuminen. Koska sovellus oli rakennettu PhoneGap Buildilla, joidenkin Androidin ominaisuuksien käyttöönotto ei ole ollut mahdollista. PhoneGap Buildilla ei ole esimerkiksi mahdollista Android tablettien ja Android älypuhelinten erottamista toisistaan, joten sovellus oli rakennettava skaalautumaan kaiken kokoisiin näyttöihin.

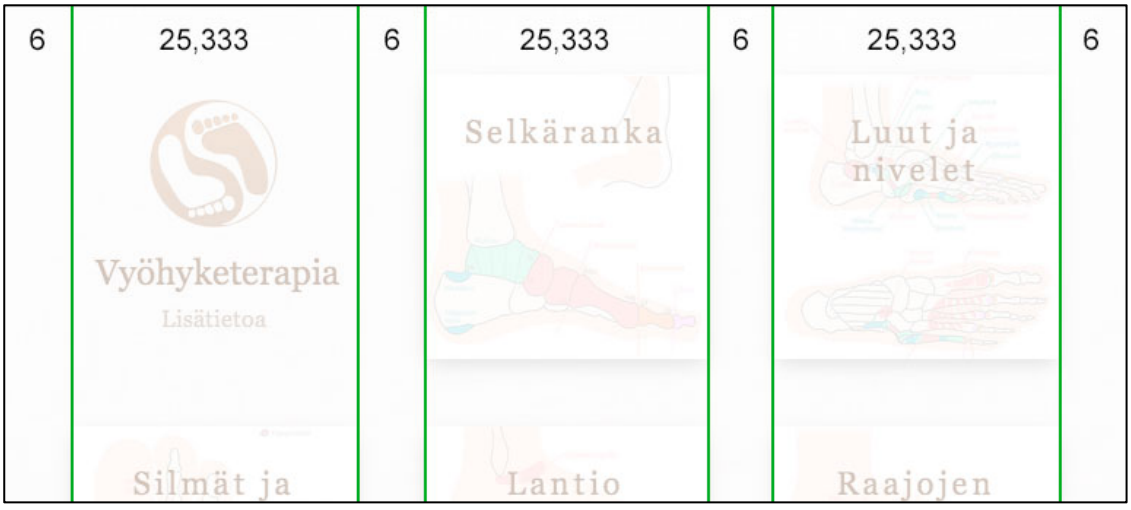
Eniten ongelmia ja muutettavaa sovelluksessa oli pikselimittojen kanssa. Koska sovellus oli rakennettu vain iPadille, oli kaikki mitat ja elementtien sijainnit asetettu pikselintarkkuudella. Ensimmäiseen vaiheeseen projektin etenemisessä kuului siis pikselintarkkojen arvojen kääntäminen suhteellisiksi arvoiksi ja prosenteiksi.

Sovelluksen muuttamisen jälkeen alkoi testausvaihe. Sovellusta muokattiin ja testattiin koko ajan selaimessa mutta tietyt ominaisuudet, kuten linkkien käyttäytyminen, swipe-toiminnot ja ulkoasun asettuminen tuli testata laitteessa. Muutaman käyttäjän voimin suoritetuissa käyttötesteissä tulleita puutteita ja huomioita korjattiin ja sovelluksen valmistuttua siirryttiin sovelluksen julkaisuvaiheeseen.

### 6.3 Responsiivisuus suhteellisilla mitoilla

Sovelluksen muuttaminen responsiivikseksi alkoi muokkaamalla koko sovelluksen perusta prosentuaaliseksi. Html- ja body-elementit oli määritetty kokoon 1 024x767 ja ne oli asetettava korkeudeltaan ja leveydeltään 100 prosenttiin. Jo alkuperäisessä versiossa bodyn ”overflow” oli määritelty ”hidden”, jolloin kaikki 1 024x768-kokoisen näkymän yli menevät osat eivät näkyisi eikä sovellusta pystynyt vierittämään ylös tai alas. Lähinnä se tuo tietyn sovellusmaisen tunnun sovellukseen, kun sovellus ei käyttäydy kuin verkkosivu.

Sovelluksen aloitussivu oli jo alusta lähtien rakennettu osittain sopeutumaan laitteen leveyteen. Aloitussivu oli jaettu neljään riviin, jotka oli jaettu kolmeen sarakkeeseen (Kuva 4). Esikatselukuvien väliin jätettiin väliä 6 prosenttia, joten yhden kuvan leveydeksi jäi 25,333 prosenttia.




Kuva 4. Sovelluksen aloitussivun marginaalit.

Ainoa muokattava asia tällä sivulla on fonttikoko. Lähdin etsimään fontin rikkoutumispistettä skaalaamalla selainikkunaa pienemmäksi. Mitä pienemmäksi näyttökoko meni, sitä vähemmän tilaa otsikkoteksteille jäi ja kun tekstit viimein näyttivät rikkoutuneilta, lisäsin media queryn pienentämään fonttikokoa.

Opastekuvien skaalautuminen aiheutti myös ongelman. Jouduin hyödyntämään JavaScriptiä kuvien asettamiseksi oikean kokoisiksi ja hotspottien laittamiseksi oikeille paikoilleen. Ensin sovellus tunnistaa laitteen mitat JavaScriptin avulla.

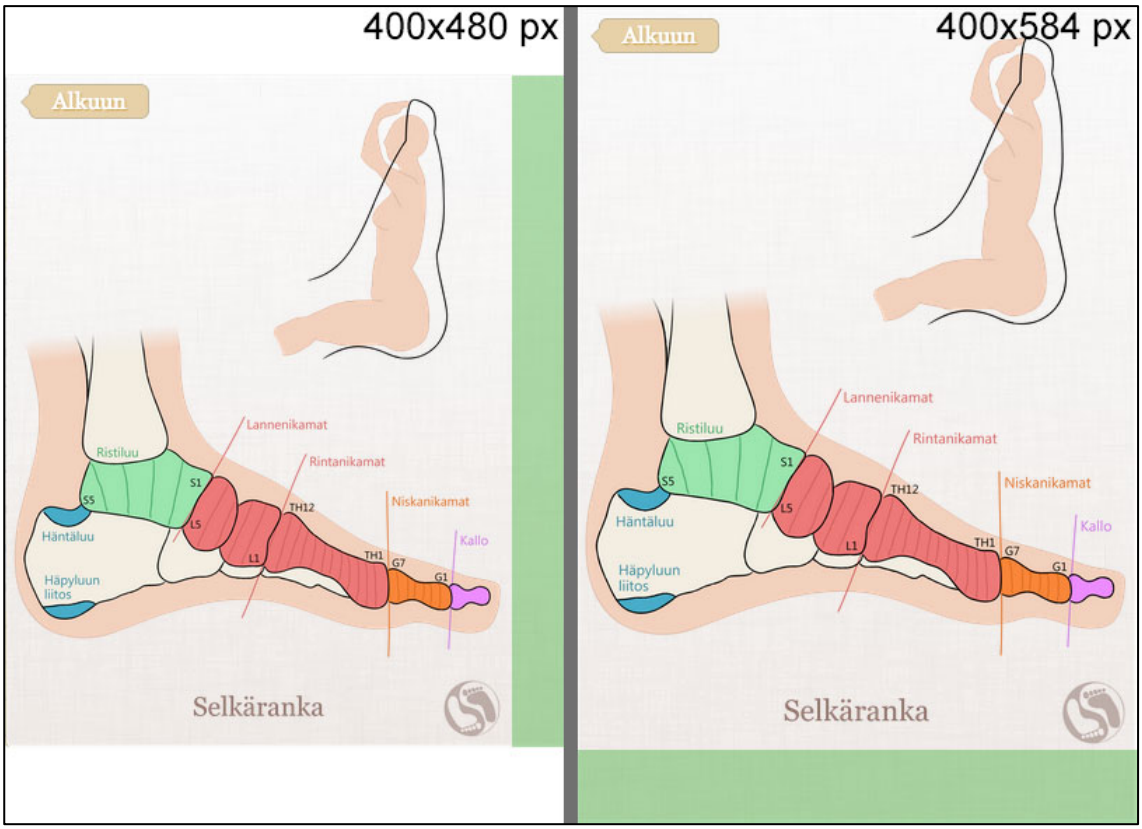
Sen jälkeen sovellus laskee alkuperäisen sovelluksen kuvasuhteen (768:1 024) sekä nykyisen laitteen kuvasuhteen. Tällöin voidaan verrata, tulisiko kuva skaalautumaan pystysuunnassa vai vaakasuunnassa. Jos laitteen kuvasuhde olisi isompi kuin iPadin kuvasuhde eli näyttö olisi suhteessa leveämpi, olisi kuvan korkeus koko näytön korkuinen. Tällöin kuvan leveys joudutaan laskemaan erikseen ja asettamaan muuttujan avulla. Jos taas näyttö olisi suhteessa kapeampi kuin iPad, olisi kuva koko näytön levyinen, mutta kuvan alle jäisi tyhjä kaistale ja enemmän tilaa opastetekstille:

```
var viewportWidth  = document.documentElement.clientWidth;
var viewportHeight = document.documentElement.clientHeight;

var viewportRatio = viewportWidth / viewportHeight;
var compareRatio = 768/1024;

if (viewportRatio > compareRatio) {
     $(".kuva").each(function() {
         var imagewidth = viewportHeight * compareRatio;
         $(this).css('width', imagewidth);
     });
}
```

Tämä keino toteuttaa kuvien skaalautumista oli nopein ja käytännöllisin. Toinen vaihtoehto olisi ollut kuvien rajaaminen mutta kuvan sivulle tai alle jäävä tyhjä tila ei haitannut sovelluksen toimintaa tai näyttänyt huonolta, joten päädyin tähän ratkaisuun. Tyhjä tila näkyy Kuva 5 vihreänä.




Kuva 5. Kuvien skaalautuminen erikokoisilla näytöillä.

Kuvien päälle asetetut hotspotit oli asetettu kohdilleen pikselimittojen tarkkuudella, joten jos sen alla oleva kuva skaalattaisiin yli tai alle 768-levyiseen näyttöön, eivät hotspotit asettuisi kohdilleen. Koska kuvat tulisivat olemaan aina koko näytön levyisiä (toisin sanoen niiden leveys olisi aina 100 prosenttia) tai koko näytön korkuisia, pystyin asettamaan hotspottien sijainnin myös prosenttiyksikköjen avulla. Koska hotspotteja oli lähes 150 kappaletta, tuntui jokaisen top- ja left-arvon yksitellen muuttaminen prosenteiksi turhalta tehtävältä. Automatisoin prosenttien laskemisen käyttämällä JavaScriptiä ja jQuerya top- ja left-pisteiden eli y- ja x-koordinaattien laskemiseen:

```
$(".maplink").each(function() {
      var top = $(this).css('top').replace(/[^-\d\.]/g, '');
      var left = $(this).css('left').replace(/[^-\d\.]/g,
      '');
      var width = $(this).css('width').replace(/[^-\d\.]/g,
      '');

      var newtop = (top/1024) * 100;
      var newleft = (left/768) * 100;
      var newwidth = (width/768) * 100;
```

```
        $(this).css({'top': newtop + "%", 'left': newleft+"%",
        'width': newwidth+"%"});
});
```

Kaava hakee jokaisen hotspotin (hotspot linkit käyttävät CSS-luokkaa .maplink), ottaa hotspotin sijaintikoordinaatit ja tekee niistä muuttujia. Sitten kaava kääntää koordinaatit prosenttiarvoksi ja asettaa hotspotin uudelle paikalleen.

Myös hotspotien leveydet oli muutettava pikseleistä prosenteiksi. Korkeuden olisi voinut laittaa myös mukautumaan sovelluksen korkeuteen, mutta koska hotspottien korkeuden ei tarvinnut muuttua niin paljon, riitti kun löytyi yksi breakpoint, jossa korkeutta täytyi pienentää.

```
if (viewportWidth < 590) {
     $(".maplink").each(function() {
         var currenth = $(this).height();
         var newh = currenth*0.7;
         $(this).css('height', newh);
     });
}
```

Viimeinen vaihe responsiivisuuden tekemisessä oli info-sivun fonttikoon muuttaminen. Skaalasin selainikkunaa etsien tekstin rikkoutusmispistettä ja sen löydyttyä lisäsin media queryn, jossa muutin fonttia pienemmäksi.

### 6.4 Testaaminen

Sovellusta testattiin kolmella eri laitteella: Android-tabletilla (Asus Transformer Pad TF300, käyttöjärjestelmänä Android 4.0), Android-älypuhelimella (Samsung Galaxy S4, Android 4.2.2) sekä iPadilla 2:lla (iOS 7) (Taulukko 1). Sovellusta testattiin iPadilla siksi, että sovelluksen muutamia tekstejä oli jouduttu päivittämään ja samalla kun julkaistaisiin Android versio, pystyttäisiin päivittämään myös iPad-versio, joka käyttäisi samaa responsiivista koodia. Testauslaitteet ovat kaikki melko uusia ja tehokkaita. Niiden käyttöjärjestelmät ovat ajan tasalla, joten ne tukevat ominaisuuksia, joita vanhemmat käyttöjärjestelmät eivät tue. Toisaalta Vyöhyketerapia-sovellus on rakennettu yksinkertaisilla CSS-

säännöillä, eikä siinä käytetä esimerkiksi uusia HTML5-elementtejä, joten vanhemmillakaan käyttöjärjestelmillä ei pitäisi olla vaikeutta piirtää sovellusta.

Taulukko 1. Testauslaitteet.

| | **Malli** | **Käyttöjärjestelmä** | **Julkaisuvuosi** |
|---|---|---|---|
| **iPad** | iPad 2 | iOS 7.0.1 | 2011 |
| **Asus** | Transformer Pad TF300 | Android 4.0 | 2012 |
| **Samsung** | Galaxy S4 | Android 4.2.2 | 2013 |

IPadilla testatessa sovellus toimi kuten sen oli suunniteltu toimivan: Sovellus piirtyi näytölle saman näköisenä, kuin selaimessa ja aloitussivun kuvalinkkejä painamalla sivusta liukui esiin valittu kuva. Hotspot-linkkejä painamalla alareunaan ilmestyi ohjetekstille varattu ruskea alue oikean korkuisena ja oikean ohjetekstin kanssa. Sivujen väliset siirtymät toimivat pienellä viiveellä mutta liukumisanimaatio oli sujuva ja terävä, kestoltaan noin koodissa asetetun 600 millisekunnin mittainen.

Android tabletin testitulokset olivat hieman huonommat. Sovellus piirtyi laitteelle oikean näköisenä ja kaikki navigointi kuvien välillä toimi, mutta kaikkien linkkien klikkaamisen jälkeen oli pieni viive, ennen kuin mitään tapahtui. Kuvien liukumisanimaatiot olivat hitaampia ja takkuisia: animaatio lähti liikkeelle viiveellä, kuva liukui hieman eteenpäin ja hyppäsi sitten animaation loppukohtaan. Sama ongelma toistui Galaxy S4:llä animaatioissa. Kaikki sovelluksen muut toiminnot toimivat eli kuvat ja hotspotit skaalautuivat halutulla tavalla. Jottei sovellus vaikuttaisi Android-laitteilla niin takkuiselta, päätin vaihtaa animaatioiden tyyliä kokonaan. Liukumisen sijaan laitoin seuraavan kuvan ilmestymään vanhan päälle häivyttämällä (fade) ja testasin uudelleen animaatioiden toiminnan ja nopeudet.

### 6.5 Animaatioiden testaaminen

Koska animaatiot aiheuttivat suurimmat ongelmat sovelluksen toiminnassa, testasin niiden toimintaa tarkemmin. Taulukko 2 on häivytys-animaatioiden testien tulokset. Ajat on otettu sekuntikellolla ja käsivaralla, joten ne voivat heittää muutaman sadasosan. Virhemarginaalin pienentämiseksi mittasin kunkin ajan 8 kertaa. Kirjasin ylös vain mielestäni tarkimmat mittaukset ja jätin pois omasta virheestä johtuvat virheelliset ajat. Taulukossa esitetyt arvot ovat testiaikojen keskiarvot.

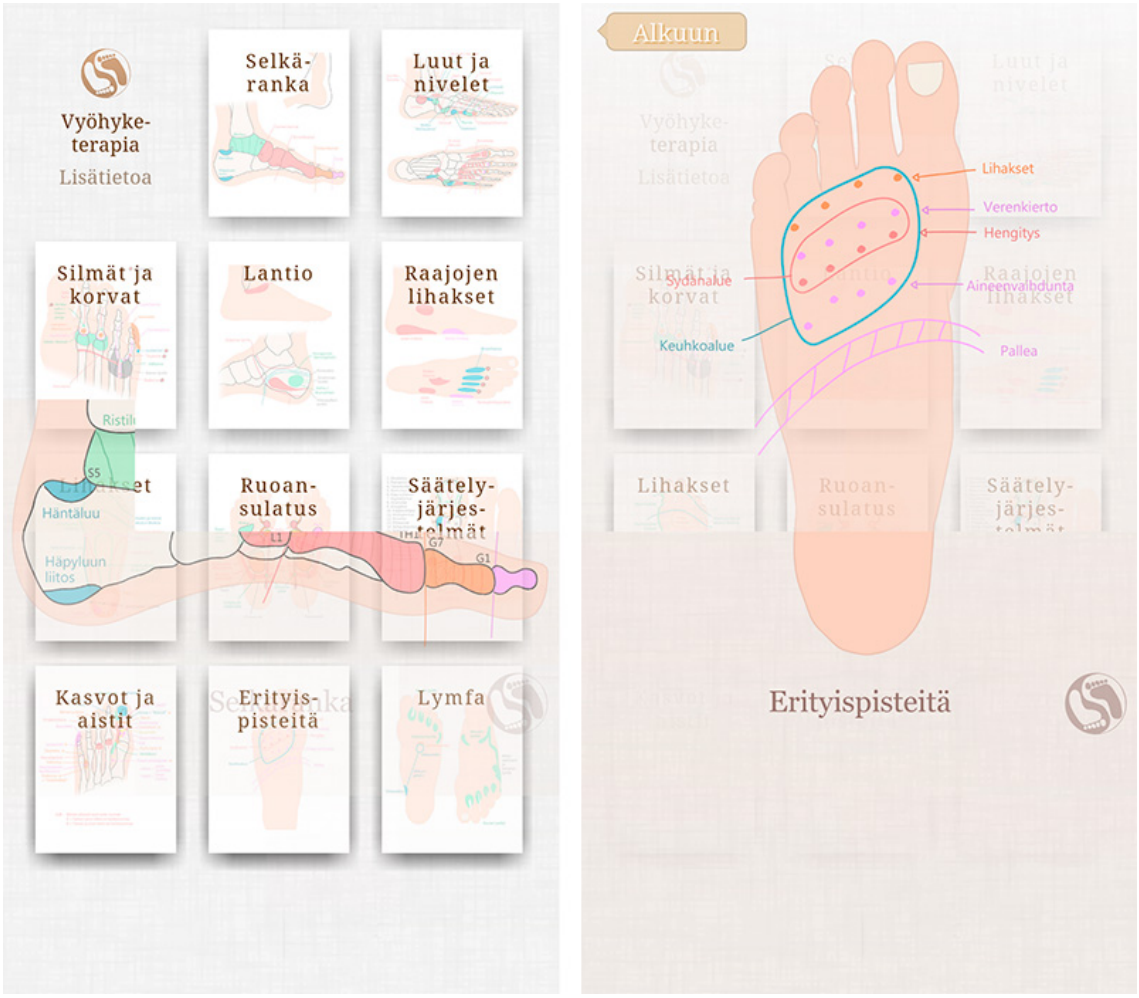
Taulukko 2. Animaatiotestit. Ajat esitetty millisekunteina (1000 ms = 1 s).

| | **iPad** | **Asus** | **Samsung** |
|---|---|---|---|
| **Animaation aloitusviive (oletus 0 ms)** | 1700 ms | 1867 ms | 544 ms |
| **Animaation kesto (600 ms)** | 600 ms | 750 ms | 650 ms |
| **Animaation sujuvuus** | Aloitusviiveen jälkeen sujuva. Ei nykimistä | Aloitusviiveen jälkeen hyvin pientä nykimistä. Ei esiinny joka kerralla. | Aloitusviiveen jälkeen hyvin nykivä ja epätasainen animaatio. |

Mittasin ajat siirtymissä, jotka lähtivät liikkeelle aloitussivun linkkejä painamalla, mutta samat ajat toistuivat kuvien välillä vasemmalle tai oikealle pyyhkäisemällä käynnistyvissä animaatioissa. Animaatioiden kestoja oli vaikea mitata mutta aloitin mittauksen kuvan ensimmäisestä nykäyksestä ja lopetin, kun kuva oli täysin näkyvissä.

Asuksen kohdalla nykiminen oli hyvin pientä. Kuva ei aina ilmestynyt edellisen näkymän päälle sujuvasti vaan animaatio saattoi alkaa sujuvasti ja hypähtää yhtäkkiä animaation loppuun, jolloin kuva näkyi kokonaan. Samsungin kohdalla nykivyys oli selkeämpää ja kuva piirtyi edellisen kuvan päälle ikään kuin loh-

koissa, alhaalta ylöspäin ja aivan kuin kuva olisi piirtynyt kaksi kertaa, ensin himmeäksi, sitten kokonaan näkyväksi. Onnistuin tallentamaan Samsung-laitteen animaation nykimisen kuvakaappaukseen (Kuva 6).




Kuva 6. Animaation nykimistä Samsung Galaxy S4:llä.

Aloin etsiä syytä animaatioiden hitauteen tutustumalla tarkemmin kuvien välisiin siirtymiin käyttämääni JavaScript-liitännäistä, Flexslideria (WooThemes 2013). Liitännäinen käyttää animoimiseen JavaScriptiä, jonka avulla se asettaa mm. kuvien läpikuultavuuden (opacity) nollaan tai yhteen ja vaihtaa kuvan syvyystason (z-index).

Androidilla on tunnettuja ongelmia JavaScript-animaatioiden kanssa. JavaScript-animaatiot vaativat paljon suorittimen (CPU) kapasiteettia, sillä ne vaativat paljon laskentaa. Sen sijaan CSS3-siirtymät (CSS-transition) vaativat vähemmän laskentaa ja pystyvät käyttämään enemmän GPU:n, grafiikkaprosessorin

tehoja animointien toteuttamiseen. Uusimmat versiot suosituimmista selaimista, kuten Firefox, Chrome, IE9, Safari ja Opera, pystyvät käyttämään laitteistokiihtyvyyttä mutta GPU-kiihdytys käynnistyy vain, jos selain havaitsee viitteitä siitä, että raskasta laskentaa mahdollisesti tapahtuu. Raskasta laskentaa vaativat mm. CSS3 siirtymien lisäksi CSS3:n 3D-muutokset (CSS3 3D transform) ja canvas piirrokset. Kun selaimen huijaa uskomaan, että CSS-animaatioissa aiotaan käyttää 3D-muutoksia, selain aktivoi GPU-tehonsa sivuston tai HTML5-sovelluksen käyttöön. (Matyus 2013.) GPU:n voi pakottaa käyttöönsä alla olevalla CSS-säännöllä:

```
transform: translate3d(0,0,0);
```

Kun GPU:n tehoja on valjastettu sovelluksen käyttöön, toimivat myös JavaScript-animaatiot nopeammin (Coyier 2013). Kaikki vanhemmat selaimet eivät kuitenkaan tue CSS3 3D-animointia (Can I Use 2014). Tästä menetelmästä on myös tulossa koko ajan vähemmän relevantti, sillä esimerkiksi uusimmat Chromen versiot käyttävät automaattisesti GPU:ta läpikuultavuuden ja 2D-muutosten toteuttamiseen. iOS 6 Safari ei myöskään aina käynnistä laitteistokiihdytystä ja on jouduttu etsimään toinen keino toteuttaa nopeampia animaatioita. (Maccaw 2013.)

Sovelluksessa käytettävä Flexslider-liitännäinen ei käyttänyt 3D-animaatioita. Testatakseni, nopeutuisivatko animaatiot jos GPU:n tehot olisivat käytössä, lisäsin yllä esitetyn CSS-rivin sovellukseen ja paketoin sovelluksen laitteeseen asetettavaksi. IPadilla animaatio ei käynnistynyt yhtään nopeammin ja Samsungilla animaatio oli yhtä tökkivä kuin aikaisemminkin. Samat testitulokset toistuivat, kun kokeilen toista 3D-ominaisuutta tarvitsevaa muutosta:

```
transform: translateZ(0);
```

Kokeilin lisätä CSS-tyylin ensin body-elementtiin, sitten kuvia ympäröivään div-elementtiin ja lopuksi elementtiin, jota Flexslider varsinaisesti animoi. Näkyvää vaikutusta animaatioiden nopeudessa ei ollut ja sama nykivä animaatio säilyi Samsungilla testattaessa.

Koska testasin sovellusta vain muutamalla laitteella, on mahdollista, että CSS-sääntö voi nopeuttaa animointia jollakin toisella laitteella. Toinen vaihtoehto, joita animaatioiden nopeuttamiseksi voisi kokeilla, olisi vaihtaa Flexslider-liitännäinen johonkin kevyempään, esimerkiksi CSS3-animointia käyttävään liitännäiseen. CSS3-animoinnit JavaScript-animaatioiden sijaan ovat eräiden testien mukaan nopeampia (Bradshaw 2012), joten sovelluksen seuraavaan versioon lähden luultavasti etsimään liitännäistä, joka animoi CSS3:n avulla.

### 6.6 Pohdinta

Jos alkaa rakentamaan mobiilisovellusta, on siitä tehtävä mahdollisimman nopea ja toimiva. Kaikki ylimääräinen koodi on poistettava viimeistään julkaisuvaiheessa ja sekin tulee hioa siistiksi ja yksinkertaiseksi, jotta sen tulkitsemisessa ei mene pitkään ja sovellus toimii viiveettä.

Responsiivinen HTML5 sisältää väkisin hieman enemmän koodia kuin yhdelle laitteelle tai yhteen kokoon rakennettu sivusto, verkkopalvelu tai sovellus. Vaikka selain osaakin tulkita media queryja nopeasti, on tiedostokoko aina hieman isompi, jos yksi sovellus tukee montaa alustaa. Osa koodeista toimii vain tietyillä alustoilla, myös osa CSS-säännöistä on selainkohtaisia.

Toisaalta responsiivisuuden lisääminen sovelluksiin takaa, että sovellus toimii varmasti useilla laitteissa. Testausta ei välttämättä vaadita niin paljon, sillä jos sovellus toimii yhdellä laitteella, sen voi olettaa toimivan myös muilla vastaavilla laitteilla. Täydellistä optimointia ja testausta jokaisella laitteella on muutenkin mahdotonta toteuttaa, joten responsiivisuus on melko hyvä oikotie.

Vyöhyketerapia-sovelluksen optimointia olisi voinut toteuttaa ainakin siten, että tableteille ja älypuhelimille olisi julkaistu eri versiot. Ulkoasu toisaalta skaalautuu hyvin molemmille ilman suuria CSS-määriä, mutta kuvien koko on joillekin älypuhelimille kaksi kertaa liian iso niiden näyttökokoon ja resoluutioon verrattuna. Toisaalta huippuresoluutioisille älypuhelimille tarvitseekin suuret kuvat, jotta ne pysyvät terävinä ja jokaisen sovelluksen kohdalla on harkittava, kuinka oleellisia

tarkat kuvat ovat ja ainakin tässä sovelluksessa koko sovelluksen käytettävyys liittyy osittain kuvien ja niiden tekstien luettavuuteen. Koska sovellus rakennettiin alun perinkin Internet-yhteyttä käyttämättömäksi offline-sovellukseksi, ei eri kuvaversioita eri resoluutioisille laitteille pystynyt lataamaan palvelimeltakaan mutta se lienee harkittavissa oleva vaihtoehto, kun sovelluksen kehitystä jatketaan.

Toinen kehitettävä asia optimoinnin lisäksi tässä sovelluksessa on sen animaatioiden nopeus. Siirtymien sujuvuus jäi hiomatta tässä vaiheessa, sillä sovellus oli tärkeää saada julkaistua. Sovellusta tullaan vielä kehittämään kattavammaksi tulevaisuudessa ja animaatioiden hiomista on helppo jatkaa näiden alustavien tutkimusten ja testien pohjalta.

On vaikeaa arvioida, kuinka paljon työaikaa säästyi, kun tämä sovellus rakennettiin mukautuvaksi kahden eri version – tablet ja älypuhelin – rakentamisen sijaan. Kahden eri version sisältöjen ylläpito on vaikeampaa, eri kokoisten kuvien koostamiseen menee aikaa. Voi kuitenkin olla, että jotta sovelluksesta saadaan optimoidumpi ja toimivampi versio, on jotakin kuvien optimointia pakko toteuttaa. Tämän responsiivisen version toteuttaminen on tuonut esiin hyviä ja huonoja puolia mukautuvien sovellusten rakentamisesta. Luulen, että valinta laitekohtaisen ja mukautuvan sovelluksen välillä on tehtävä jokaisen sovelluksen kohdalla erikseen. Voi myös olla, että mobiilisovellus ei edes ole järkevin ratkaisu, vaan olisi parempi toteuttaa suunniteltu verkkopalvelu mobiililaitteille optimoituna, edelleen responsiivisena verkkopalveluna.

# 7 Tulevaisuuden näkymiä

Vaikka pystymme määrittämään, mitkä teknologiat saavuttavat sellaisen kypsyyden, että niitä voidaan tuoda kuluttajien käyttöön, on vaikeampaa ennustaa mitkä kasvavat nopeimmin, mitä käyttäytymistapoja ihmisille kehittyy ja mitä uusia ja odottamattomia tuotteita syntyy. Voi myös olla, että jumiudumme johonkin vaiheeseen esimerkiksi patenttitaistojen takia. (Rieger 2012, 56.)

Paras keino tarkastella tulevaisuutta on tutkia suuntauksia, jotka jo nyt muokkaavat elämäämme ja tulevat tekemään sitä jatkossakin. Nämä suuntaukset liittyvät usein liittyvyyteen ja yhteyden pitoon. Älypuhelimia ja internetiä käyttäviä laitteita myydään koko ajan enemmän ja pian me kaikki olemme yhteydessä verkkoon ja sen kautta toisiimme. Tämä yhteys vaikuttaa luultavasti jokaiseen elämän alueeseen; mobiililaitteiden ohjelmistot tekevät eksymisestä lähes mahdotonta; yhteistyön tekeminen ympäri maailmaa olevien ihmisten kanssa tulee yhä helpommaksi ja halvemmaksi; lähes kaikki tieto maailmasta on ulottuvillamme kaiken aikaa ja kaikkialla. (Rieger 2012, 57.)

Sen lisäksi, että kaikki ihmiset ovat pian yhteydessä toisiinsa verkon välityksellä, myös kaikki asiat, laitteet tulevat olemaan yhteydessä toisiinsa ja yhteydessä toisiinsa olevat laitteet pystyvät keskustelemaan keskenään. Täydellinen päivä olisi, esimerkiksi, että heräät aamulla herätyskellon soittoon, joka on katsonut kalenteristasi, milloin aamun ensimmäinen kokous alkaa ja mille autolautalle sinun tulee ehtiä ollaksesi ajoissa paikalla. Pesuhuoneen lämmitin olisi ollut päällä puoli tuntia aikaisemmin, jotta aamukylpyyn mennessäsi pesuhuone olisi sopivan lämpöinen. Jos yön lämpötilat olisivat olleet alhaiset, sinun tulisi lähteä viisi minuuttia aikaisemmin raaputtaaksesi jään pois auton ikkunoista. Lautalle ajaessasi saisit tietoja reaaliajassa, että lautta on viisi minuuttia myöhässä, joten kiirettä ei ole. (IBMSocialMedia 2010.)

Jo nyt käytössä olevat erilaiset pilvipalvelut ovat hyvä esimerkki siitä tulevaisuuden suuntauksesta, jossa käyttäjä ei ole enää sidottu siihen laitteeseen, jota käyttää, vaan voi jatkaa työskentelyä samasta kohdasta laitteelta toiselle siirtyessään. Pilvipalvelut toimivat siten, että ne tallentavat tietoa ja tiedostoja verkon yli palvelimelle ja käyttäjä pääsee käsiksi tallennuksiin mistä tahansa laitteelta, jos sillä on internet-yhteys. Kuluttajien suosiossa pilvipalveluja ovat esimerkiksi tiedostojen jakoon ja synkronointiin tarkoitetut Dropbox ja Googlen Drive. Adobe käyttää pilvipalveluja hyödykseen tarjoamalla käyttäjille Creative Cloud -palvelua, jossa käyttäjä voi kuukausilisenssillä ladata ohjelmiston koneelleen, tallentaa tiedostoja verkon yli pilveen ja käyttää synkronoida samat ohjelmat

mihin tahansa käyttämäänsä tietokoneeseen. Jos sovellukseen tulee uusi versio, sen voi päivittää heti ilman mitään lisämaksua.

Maxthon cloud browser, kuten myös Google Chrome, on verkkoselain, joka mahdollistaa sen, ettei käyttökokemus ole sidottu yhteenkään laitteeseen. Selain tallentaa pilveen tilillesi tiedot välilehdistä, jotka ovat avoinna kun suljet selaimen ja avaa ne uudestaan kun käynnistät selaimen uudestaan millä laitteella tahansa. Aamulla selaamasi uutiset tabletilla tallentuvat ja voit jatkaa artikkelin lukemista kun avaat puhelimesi työmatkalla junassa. Tämä on yksi tulevaisuuden suunnista, sillä jo nyt suuri osa esimerkiksi verkko-ostoksista käynnistyy käyttäjän selatessa verkkokauppaa ensin mobiililaitteellaan ja myöhemmin jatkaessaan ostoksia tietokoneen ääressä.

Yksi tulevaisuuden mielenkiintoisimmista muutoksista näyttöjen teknologian osalta on se, kun joustavat ja fyysisesti taipuisat näytöt julkaistaan kaupallisesti. Kaksi teknologiaa, jotka tulevat hallitsemaan näitä markkinoita, ovat luultavasti e-paperi sekä AMOLED (Active-matrix organic light-emitting diode). (Rieger 2012, 70.) E-paperi on teknologia, joka on luotu jäljittelemään tavallisen mustepaperin ulkoasua. Toisin kuin perinteiset taustavalaistut näytöt, jotka säteilevät valoa, sähköinen paperi heijastaa valoa kuten tavallinen paperi ja se tekee siitä mukavamman lukea ja antaa näytölle laajemman katselukulman verrattuna tavallisiin näyttöihin. (Wikipedia 2013a.)

E-paperi on kasvattanut suosiotaan sen hinnan halpenemisen ja menestyneiden e-kirjojen lukijoiden kuten Amazon Kindlen avulla. Suuret valmistajat, kuten LG, ovat vihjanneet, että ensimmäiset e-paperit tullaan julkaisemaan vuoden 2013 aikana mutta mihin nämä näytöt ilmestyvät ensimmäisenä on arvailujen varassa. Ainakin aikakausilehtiala on ilmaissut kiinnostuksensa liittää interaktiivisia alueita olemassa olevaan printtimediaan. (Rieger 2012, 71.)

Joustavien AMOLE -näyttöjen luvataan tarjoavan samaa virrankulutusta, kirkkautta, värikylläisyyttä ja kontrasti kuin nykyiset AMOLED -näyttö, samalla myös ollen joustavia ja täysin läpinäkyviä, jolloin näytön läpi voi nähdä. Näytön tark-

kuus voi saavuttaa 300 pikseliä tuumalla ja kuitenkin näyttö on niin kevyt ja taipuisa, että sen voi rullata kasaan kuten sanomalehden. (Rieger 2012, 71.)

Heijastusnäytöt (HUD eli Heads-up display) ovat läpinäkyviä näyttöjä jotka esittävät dataa ilman, että käyttäjän tarvitsee katsoa pois hänen tavallisesta katseen suunnastaan. Esimerkkinä on Google Glass.

Miten tämä kaikki teknologioiden ja suuntauksien kehittyminen vaikuttaa verkkopalvelujen suunnitteluun? Verkon kehittäjä ei voi koskaan tietää, millä laitteella ja missä tilanteessa verkkopalvelua käytetään, joten mitään ei voi olettaa. Näyttöruutujen ja kosketusnäyttöjen hinnat tulevat koko ajan alaspäin ja niitä liitetään yhä useampiin laitteisiin, jolloin yhä useammilla laitteilla halutaan yhteys internetiin. Esimerkiksi jääkaapit, joiden oveen on upotettu näyttö uutisten tai reseptien selaamista varten, ovat jo tätä päivää. Entä miltä verkkopalvelu tulee näyttämään ranteen tai käsivarren ympärille kierretyllä näytöllä tai päässä pidettävän laitteen avulla eteesi heijastettuna?

Koska ei voida tietää, missä verkkopalveluamme tullaan käyttämään ja miten se joutuu kommunikoimaan tulevien laitteiden kanssa, on palveluista tehtävä joustavia. Joustavuus antaa käyttäjälle mahdollisuuden päättää itse missä ja miten hän tulee avaamaan tuotteen. Toisaalta, koska tulevaisuuden ennustaminen on mahdotonta ja voimme vain arvailla tulevaisuuden suuntauksia tämän päivän trendien perusteella, ei tulevaisuutta kannata edes ajatella liikaa. Kehitä verkkopalvelusi käyttäjien nykyisten tilanteiden, päämäärien ja rajoitteiden mukaan ja valmistaudu siihen, että voit joutua muuttamaan palveluasi tulevaisuudessa (Rieger 2012, 87).

# 8 Lopuksi

Responsiivinen verkkosuunnittelu vaatii ajattelutavan muuttamista. Verkkosuunnittelijat ovat tottuneet suunnittelemaan verkkosivustot tietokoneen näytölle mutta ihmisten tapa käyttää verkkoa on muuttunut. Pystyäksemme vastaamaan

käyttäjien tarpeisiin, joudumme muuttamaan suunnittelun lähtökohtia mobiilimpaan suuntaan, samoin kuin käyttäjien tavat ovat muuttuneet.

Lähdin rakentamaan Vyöhyketerapia-sovellusta HTML5-teknologialla juuri sen mukautuvuuden vuoksi. Vaikka sovellus aluksi rakennettiin iPadille, oli pitkän ajan suunnitelmissa koko ajan viedä sovellus myös Androidille ja kenties laajentaa sovelluksen sisältöä enemmän dynaamisemmaksi. Responsiivisuuden tuominen alun perin demomaisesti yhdelle laitteelle rakennettuun sovellukseen oli seuraava askel sovelluksen ympärille kehitetyssä isommassa suunnitelmassa.

Mutta miksei sovellusta alun pitäen rakennettu skaalautuvaksi? Skaalautuvuuden rakentaminen ei lisää kovinkaan paljon työmäärää, pikselien sijaan on helppo kirjoittaa prosenttimerkkiä ja ulkoasun ja sisältöjen rikkoutumispisteet löytyvät pelkkää selainta katsomalla. Yritykset ovat huomanneet mobiilin trendin ja haluavat kaikille laitteille sopivia sovelluksia mahdollisimman pienellä kustannuksella. Kaikille alustoille HTML5-sovellusten kehittäminen on noussut suosituksi jo pelkästään sen helppouden takia: verkkosivuja varten rakennettuja JavaScript-kirjastoja pystytään hyödyntämään myös sovelluksia rakennettaessa ja mitä enemmän verkkoteknologia kehittyy, sitä näyttävämpiä ja toimivampia sovelluksia saadaan rakennettua. Myös yhä useamman verkkokehittäjän on nyt helpompi lähteä rakentamaan mobiilisovelluksia, koska tekniikka on sama.

## Lähteet


Adobe Systems Incorporated. 2013. Adobe PhoneGap Build. https://build.phonegap.com/. 23.1.2014.

Bhanot, S. 2012. Developing Cross-Device HTML5 Apps Using Visualforce. Salesforce.com, Inc. http://blogs.developerforce.com/developer-relations/2012/05/cross-device-html5-apps-using-visualforce.html. 23.1.2014.

Bradshaw, R. 2012. Why are CSS3 transitions worth using? http://css3.bradshawenterprises.com/blog/jquery-vs-css3-transitions/. 4.2.2013.

Bricklin, Dan. 2013. Responsive Web App Design. http://www.bricklin.com/responsiveapps.htm. 23.1.2014.

Callahan, B. 2011. Content Prototyping In Responsice Web Design. Smashing Media. http://mobile.smashingmagazine.com/2011/09/26/content-prototyping-in-responsive-web-design/. 25.3.2013.

Callan, D. 2009. Content is king. AKA Marketing.com http://www.akamarketing.com/content-is-king.html. 25.3.2013.

Can I Use. CSS3 3d Transforms. Alexis Deveria. http://caniuse.com/transforms3d. 3.2.2014.

Clarke, A. 2010. Hardboiled CSS3 Media Queries. https://gist.github.com/jensgro/1362209. 23.3.2013.

Clemens, D. 2012. Design Process In The Responsive Age. Smashing Media. http://uxdesign.smashingmagazine.com/2012/05/30/design-process-responsive-age/. 25.3.2013.

Coyier, C. Myth Busting: CSS Animations vs. JavaScript. CSS-Tricks. http://css-tricks.com/myth-busting-css-animations-vs-javascript/. 3.2.2014.

Firtman, M. 2014. Breaking the Mobile Web. http://www.mobilexweb.com/emulators. 23.1.2014.

Fling, B. 2009. Mobile Design and Developement. Sebastopol: O'Reilly Media.

Frost, B. 2012 The Mobile Book. Germany: Smashing Media.

Gallagher, N. 2011. Responsive images using CSS3. http://nicolasgallagher.com/responsive-images-using-css3/. 11.4.2013.

Grigsby, J. 2010. CSS Media Query for Mobile is Fool's Gold. Cloud Four. http://blog.cloudfour.com/css-media-query-for-mobile-is-fools-gold/. 22.4.2013.

Google Android Developer Documentation. 2013. Supporting Multiple Screens. http://developer.android.com/guide/practices/screens_support.html. 12.11.2013.

Google Developers. 2013. Chrome Developer Tools. https://developers.google.com/chrome-developer-tools/. 23.1.2014.

Google Chrome Web Store. 2013a. Window Resizer. https://chrome.google.com/webstore/detail/window-resizer/kkelicaakdanhinjdeammmilcgefonfh/details. 12.11.2013.

Google Chrome Web Store. 2013b. Ripple Emulator (Beta). https://chrome.google.com/webstore/detail/ripple-emulator-beta/geelfhphabnejjhdalkjhgipohgpdnoc. 12.11.2013.

Hiltunen, M., Laukka, M., Luomala, J. 2002. Mobile user experience. Helsinki: Edita Publishing Inc.

IBMSocialMedia. 2010. The Internet of Things. IBM. http://www.youtube.com/watch?v=sfEbMV295Kk. 16.2.2013.
International Telecommunication Union. 2010. Press Release: ITU sees 5 billion mobile subscriptions globally in 2010. ITU. http://www.itu.int/newsroom/press_releases/2010/06.html. 19.2.2013.
Jehl, S. 2012. Picturefill. https://github.com/scottjehl/picturefill
Johnson, J. 2012. Responsive Design: Why You're Doing It Wrong. Design Shack. http://designshack.net/articles/css/responsive-design-why-youre-doing-it-wrong/. 4.2.2013.
Jung, J-B. 2012. Cats Who Code. http://www.catswhocode.com/blog/making-a-website-responsive-in-3-easy-steps. 20.1.2014.
Kadlec, T. 2012. Media Query & Asset Downloading Results. http://timkadlec.com/2012/04/media-query-asset-downloading-results/. 22.4.2013.
Keith, J. 2011. Conditional Loading for Responsive Design. 24 Ways. http://24ways.org/2011/conditional-loading-for-responsive-designs/. 2.4.2013.
Knight, K. 2011. Responsive Web Design: What It Is and How to Use It. Smashing Media. http://coding.smashingmagazine.com/2011/01/12/guidelines-for-responsive-web-design/. 25.3.2013.
Koch, P-P. 2012. The Mobile Book. Germany: Smashing Media.
Korpela, J. K. 2008. CSS verkkosivujen muotoilussa. Helsinki: WSOY.
Maccaw, A. 2013. All you need to know about CSS Transitions. http://blog.alexmaccaw.com/css-transitions. 3.2.2014.
Marcotte, E. 2010. Responsive Web Design. A list apart. http://alistapart.com/article/responsive-web-design. 3.2.2013.
Marcotte, E. 2011. Responsive Web Design. New York: A Book Apart.
Matyus, L. 2013 Improving HTML5 app performance with GPU accelerated CSS transitions. Urban Insight, Inc. http://www.urbaninsight.com/2013/01/04/improving-html5-app-performance-gpu-accelerated-css-transitions. 3.2.2014.
Modernizr. 2013. http://modernizr.com/. 23.1.2014.
Phonegap. 2013. Adobe Systems Inc. http://phonegap.com/. 11.1.2014.
Rieger, S. 2012. The Mobile Book. Germany: Smashing Media.
Roberts, H. 2011. Responsive images right now. http://csswizardry.com/2011/07/responsive-images-right-now/. 10.10.2013.
Russell, C. 2011. Responsive Images and Context Aware Image Sizing. http://craig-russell.co.uk/2011/01/22/responsive-images-and-context-aware-image-sizing.html#.UWbypJO-18E. 11.4.2013.
Sencha. 2014. Sencha Inc. http://www.sencha.com/. 23.1.2014.
Stephen, Thomas. 2013. Simple Responsive Images With CSS Background Images. Smashing Media. http://mobile.smashingmagazine.com/2013/07/22/simple-responsive-images-with-css-backgrounds/. 12.11.2013.
Sums, B. 2014 Device.js. https://github.com/borismus/device.js. 23.1.2014.
Van Etten, R. 2012. Device and Viewport size in JavaScript. http://ryanve.com/lab/dimensions/. 12.11.2013.
Wikipedia. 2013a. Electronic paper. http://en.wikipedia.org/wiki/Electronic_paper. 19.2.2013.

Wikipedia. 2013b. Mobile Web. http://en.wikipedia.org/wiki/Mobile_Web. 19.2.2013.
Wikipedia. 2013c. Samsung Galaxy Note series. http://en.wikipedia.org/wiki/Samsung_Galaxy_Note_series. 12.11.2013.
WooThemes. 2013 FlexSlider 2.2.2 https://github.com/woothemes/flexslider. 3.2.2014.