RESPONSIIVINEN WEB-KEHITYS WORDPRESS-JULKAISUJÄRJESTELMÄSSÄ

Tiina Jutila

Opinnäytetyö
Huhtikuu 2013

Mediatekniikan koulutusohjelma
Tekniikan ja liikenteen ala

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖN
KUVAILULEHTI

Tekijä(t) JUTILA, Tiina	Julkaisun laji Opinnäytetyö	Päivämäärä 30.4.2013
	Sivumäärä 66 + 9	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty (X)

Työn nimi
RESPONSIIVINEN WEB-KEHITYS WORDPRESS-JULKAISUJÄRJESTELMÄSSÄ

Koulutusohjelma
Mediatekniikka

Työn ohjaaja(t)
MANNINEN, Pasi

Toimeksiantaja(t)
SINISALO, Natanael - MEOM Oy

Tiivistelmä

Opinnäytetyössä tutkittiin responsiivisen web-kehityksen toteuttamista WordPress-julkaisujärjestelmässä. Tavoitteena oli toteuttaa yrityksen käyttöön WordPress-teema, joka sisälsi tarvittavat ominaisuudet mukautuvien verkkosivujen toteuttamiseen. Teeman tarkoituksena oli toimia pohjana, jonka päälle verkkosivun rakentaminen olisi mahdollisimman vaivatonta.

Työssä käsiteltiin mobiililaitteiden asettamia haasteita web-kehityksessä, sekä tekniikoita niiden ratkaisemiseen. Päätelaitteiden ominaisuudet sekä niiden näyttökoot muuttuvat alinomaan. Laitevalmistajat julkaisevat koko ajan suorituskykyisempiä laitteita, mutta suurella osalla käyttäjistä on yhä käytössä laitteen vanhempi versio. Verkkosivun tulisi palvella kaikkia käyttäjiä riippumatta siitä, onko hänellä tietokone vai älypuhelin, kuinka nopea hänen verkkoyhteytensä on ja mitä selainta ja sen versiota hän käyttää.

Keskeisimpinä asioina ovat sisällön skaalaaminen päätelaitteen leveyteen CSS:n avulla, mukautuvat kuvat sekä suorituskyky. Opinnäytetyössä tarkasteltiin myös pintapuolisesti modernien web-tekniikoiden, kuten HTML5 ja CSS3, hyödyntämistä nykyaikaisessa web-kehityksessä. Etenkin kuvien näyttämiselle pyrittiin löytämään mahdollisimman suorituskykyinen ratkaisu, joka toimii sekä normaaleilla että korkearesoluutioisilla näytöillä.

Lopputuotteena tehty teema tehtiin vastaamaan WordPressin asettamia teeman standardeja. Teemaan sisällytettiin ratkaisu dynaamisten sekä staattisten kuvien mukautumiseen, mobiilinavigaatio, mahdollisuus vaihtoehtoisen sisällön näyttämiseen ja kustomoitava CSS-grid. Teemassa käytettiin CSS:n esikäsittelijää Stylusta, jonka tarkoituksena oli nopeuttaa ja helpottaa tyylitiedoston hallintaa. Teemalla toteutettiin esimerkkisivusto, jonka tarkoituksena oli esitellä teeman ja sen ominaisuuksien toimintaa. Teemasta myös kirjoitettiin toimeksiantajana toimineelle yritykselle dokumentti, joka sisälsi dokumentaation teemassa käytetyistä ominaisuuksista sekä ohjeistuksen.

Avainsanat (asiasanat)
Responsiivisuus, web-kehitys, WordPress

Muut tiedot

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

DESCRIPTION

Author(s)	Type of publication	Date
JUTILA, Tiina	Bachelor´s Thesis	30.4.2013
	Pages 66 + 9	Language Finnish
		Permission for web publication (X)

Title
RESPONSIVE WEB DESIGN IN WORDPRESS CMS

Degree Programme
Media Engineering

Tutor(s)
MANNINEN, Pasi

Assigned by
SINISALO, Natanael – MEOM Oy

Abstract

This Bachelor's Thesis studies the implementation of responsive web design in WordPress CMS. The aim was to make for the assigner company's use a WordPress theme, which would include all necessary features for generating responsive websites. The purpose of the theme was to work as a blank template on which the construction of a web page would be as easy as possible.

The work dealt with the challenges in web-development posed by mobile devices and also techniques for solving those problems. The features and screen sizes of devices are changing all the time. Manufactures are constantly publishing higher performer devices, but the majority of users is still using an older version of the device. Web pages should serve all users, regardless of whether they have a computer or a smartphone, how fast their network connection is and what browser and version they are using.

Key issues are scaling the contents to width of the device by using CSS, adaptive images and performance. The thesis also discusses slightly modern web technologies such as HTML5 and CSS3 and how those techniques are used in modern web-development. One of the major objectives was to find the most efficient solution for displaying images for both the standard and high-resolution screen.

The end-product of the theme was made to respond WordPress theme standards. The theme included the solution for displaying adaptive dynamic and static images, mobile navigation, the possibility to display alternative content and a customizable CSS-grid. The theme used CSS preprocessor Stylus, which was intended to accelerate and facilitate the style file management. The theme was used to build an example website the aim of which was to introduce the theme and its features. Theme was also documented for the company. The document contained descriptions of the properties of the theme, as well as a guidance on HTML and CSS structure.

Keywords
Responsive web design, web-development, WordPress

Miscellaneous

Sisältö

Kuviot

Taulukot

1 Lähtökohdat

1.1 Toimeksiantaja

Opinnäytetyön toimeksiantaja oli MEOM Oy, jyväskyläläinen digipuoleen erikoistunut mainostoimisto. MEOM työllistää neljän yrittäjän lisäksi yhden työntekijän sekä useamman freelancerin. Yrityksen toimialaan kuuluvat verkkosivut, sähköpostimarkkinointi, painotuotteet, yritysilmeet sekä sisällöntuotanto.

MEOM toteuttaa verkkosivut WordPress-julkaisujärjestelmän päälle.

1.2 Tavoitteet

Opinnäytetyön tavoitteena oli tutkia mobiililaitteille responsiivisuuden avulla mukautuvien verkkosivujen kehittämistä WordPress-julkaisujärjestelmässä. Samalla tutkittiin nykyaikaisia tekniikoita, jotka ovat olennaisia verkkosivujen kehittämisessä.

Tuloksena luotiin WordPress-julkaisujärjestelmää varten teema, joka sisälsi kaikki tarvittavat ominaisuudet, joiden avulla saadaan luotua responsiivinen verkkosivusto mahdollisimman yksinkertaisesti. Vaikka tarkoituksena oli ratkaista ensisijaisesti mobiililaitteiden aiheuttamia ongelmia, otettiin toteutuksessa huomioon myös desktop-selaimet ja niiden vanhemmat versiot.

Teeman tärkeimpinä ominaisuuksina olivat CSS-grid, joka vastaa yrityksen tarpeita, sisällytetty mobiilinavigaatio sekä valmiudet näyttää eri kuva eri resoluutiolla. Sen tarkoituksena oli toimia ohjenuorana siihen, kuinka verkkosivuista saadaan responsiiviset mahdollisimman pienillä resursseilla.

2 Mobiililaitteet

2.1 Yleistä

Aikaisemmin on totuttu siihen, että verkkosivuja katsotaan suhteellisen isolta näytöltä suhteellisen pienellä määrällä selaimia. Vuosien saatossa näyttökoot vain kasvoivat ja verkkoyhteyksistä tuli nopeampia. Oli turvallista suunnitella entistä

leveämpiä verkkosivuja tarvitsematta olla huolissaan siitä, että sivusto ei mahtuisi käyttäjän näytölle. Hitaista yhteyksistä ei tarvinnut välittää, joten suuria kuvia pystyi käyttämään huolettomasti.

Vuonna 2007 Apple toi markkinoille ensimmäisen iPhonen ja kilpailijat seurasivat perässä. Enää ei ollut itsestäänselvyys, että käyttäjän resoluutio olisi suurempi kuin 800 x 600 pikseliä. Päinvastoin, päätelaitteen resoluutiot olivat pienempiä kuin aikoihin. Lisäksi käyttäjä ei oletusarvoisesti ollut kiinni nopeassa verkkoyhteydessä, joten sivukoon merkitys kasvoi jälleen.

Kun puhutaan mobiililaitteista, niillä tarkoitetaan kahta asiaa: puhelinta sekä taulutietokonetta eli tablettia. Näiden kahden laitteen ero on siinä, että tabletti on puhelinta suurempi, eikä sitä ole ensisijaisesti tarkoitettu puheluihin. Joissakin tabletti-malleissa kuitenkin on puhelumahdollisuus. Joidenkin mielestä tabletteja ei edes lasketa mobiililaitteiksi, koska niiden resoluutio vastaa useimmiten kannettavaa tietokonetta. Tablettia voidaan kuitenkin käyttää myös pystyasennossa, jolloin näytön leveys on kapeampi. Lisäksi niissä on kosketusnäyttö, joka tekee käyttökokemuksesta täysin erilaisen kuin kannettavalla tietokoneella.

Mobiililaitetta voidaan käyttää missä tahansa: junassa, puistossa, töissä, kotona. Ympäristö, jossa laitetta käytetään, vaihtelee. Siksi käyttökokemuksesta pitäisi tehdä mahdollisimman yksinkertainen: saat haluamasi sisällön huolimatta siitä, joudutko käyttämään puhelinta yhdellä kädellä, kun teet toisella kädelläsi jotain muuta.

Graafinen suunnittelija tietää, mille paperille hänen tekemänsä mainos painetaan, missä se painetaan, kuinka värit käyttäytyvät ja mitä sisältöä mainoksessa on. Verkkosivuja suunnitellessa ei useimmiten edes tiedetä sivuston lopullista sisältöä ja vaikka tiedettäisiinkin, voi se muuttua julkaisun jälkeen. Puhumattakaan siitä, että voitaisiin sanoa varmaksi, että sivua käytetään vain tietyillä laitteilla, saatikka selaimella. Mobiililaitteissa valikoima on laaja, selainvalikoima vielä laajempi. Siksi niitä kaikkia ei edes kannata yrittää hallita.

2.2 Ero desktop-selaimiin

2.2.1 Resoluutio

Mobiililaitteen resoluutio voi olla mitä tahansa. Eri laitteita on niin suuri valikoima, että on mahdotonta tietää, mikä loppukäyttäjän resoluutio tulee olemaan. Käyttäjä voi käyttää mobiililaitettaan pysty- tai vaakasuunnassa, jolloin selaimen leveys on eri. Myös dekstop-käyttäjäkin voi muuttaa selainikkunansa kokoa, mutta harvemmin ikkuna on yhtä kapea kun pienimmissä puhelimissa.

2.2.2 Suorituskyky ja verkon nopeus

Nykyään edullisessakin tietokoneessa on suhteellinen suorituskyky, mutta mobiililaitteissa tehoa ei ole yhtä paljon. Tämä ongelma tulee vastaan etenkin silloin, kun käytetään paljon CSS3-transitioita ja JavaScriptiä. Pahimmassa tapauksessa sivu voi kaatua, vaikka tietokoneella se latautuisi ongelmitta. (Avola & Raasc 2013.)

Mobiililaitetta voidaan käyttää missä tahansa, se voi olla internetissä 3G:n, 4G:n tai WLAN:in kautta. Kun desktop-käyttäjät ovat tottuneet nopeaan internetiin, voi mobiilikäyttäjän 3G olla huonolla kantavuusalueella tuskallisen hidas. Tässä vaiheessa on erityisen tärkeää se, ettei päätelaitteella näytetä 1200 pikseliä leveää kuvaa, mikäli näytön leveys on vain 300 pikseliä.

2.2.3 Hiiri ja näppäimistö

Tietokoneella käyttäjällä on hiiri sekä näppäimistö. Mobiililaitteessa käyttäjä käyttää hiirenään omaa sormeaan, eikä näppäimistökään toimi samalla periaatteella kuin tietokoneessa. Desktop-selaimessa käyttäjä on tottunut siihen, että vietäessä hiiri linkin päälle, linkin ulkoasu muuttuu ja hän saa varmistuksen siitä, että linkki tosiaan on linkki. Mobiilikäyttäjällä tätä varmistusta ei tule, linkistä saa varmistuksen vasta silloin, kun sitä on painanut.

Linkin painamisessa on mobiililaitteissa 300 – 500 millisekunnin viive, ennen kuin linkin toiminta suoritetaan. Tämä johtuu siitä, että mobiiliselain odottaa, aikooko

käyttäjä vain klikata linkkiä vai tehdä jonkin muun toiminnon, kuten scrollauksen. (Avola & Raasch 2013.)

On otettava myös huomioon, että desktop-käyttäjällä hiiren kursori on pieni ja sillä on helppo osua pieniinkin kohteisiin. Käyttäjän sormella ei kuitenkaan ole vakiokokoa, joten klikattavien elementtien tulisi olla tarpeeksi suuria, jotta käyttäjä saa koskettua haluamaansa elementtiin.

Desktop-käyttäjällä on erillinen näppäimistö, jota voi käyttää missä vaiheessa tahansa verkkosivuja selatessa. Mobiilikäyttäjän näppäimistö puolestaan ilmestyy käytettäväksi vain silloin kun klikataan kenttää, johon pystyy kirjoittamaan. Tämä poistaa mobiilikäyttäjältä mahdollisuuden käyttää kaikkia ominaisuuksia, jotka liittyvät näppäimistön käyttämiseen, kuten kuvien selaamiseen nuolinäppäimillä.

2.3 Näyttökokojen kartoitus

Taulukossa 1 on esitelty muutamien mobiililaitteiden resoluutiot, pikselisuhteet ja PPI:t. PPI on luku, joka kuvaa sitä, kuinka monta pikseliä on tuumalla, mitä suurempi arvo sitä tarkempi näyttö on. Mobiililaitteissa PPI:tä pyritään kasvattamaan, koska niitä katsotaan lähempää kuin tietokoneen näyttöä. Liian tarkasta PPI:stä ei kuitenkaan ole hyötyä, sillä ihmissilmä pystyy erottamaan pikseleitä vain tiettyyn rajaan asti.

Taulukko 1. Eri mobiililaitteiden koot (Talja 2012), (Edwards n.d.)

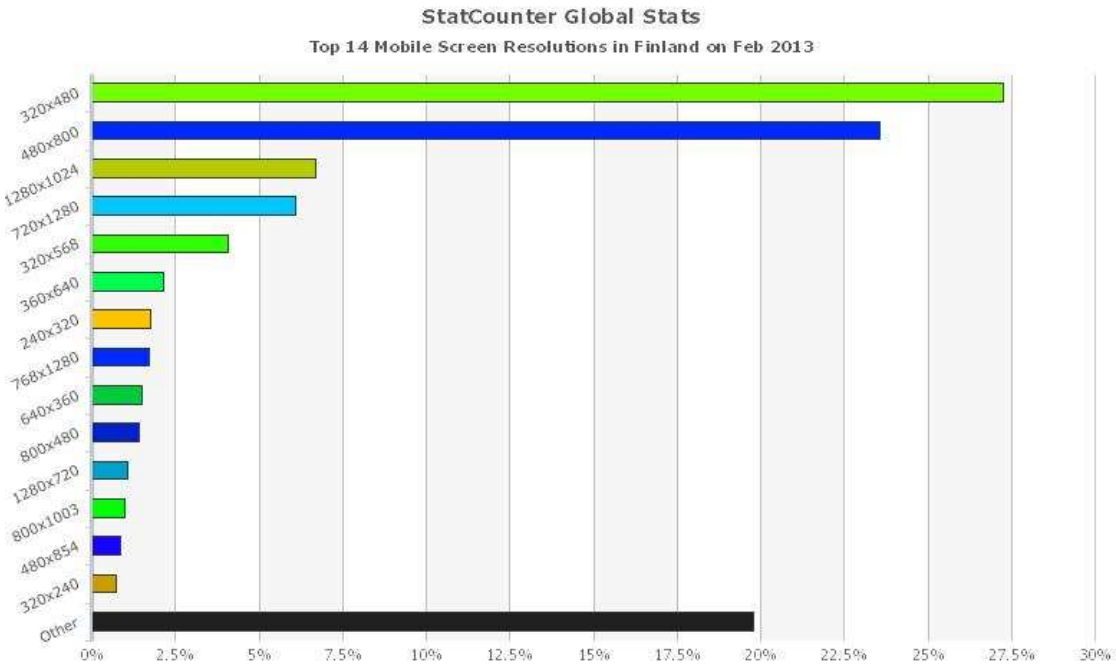
Malli	**Tyyppi**	**Resoluutio**	**Pikselisuhde**	**PPI**
Apple iPad 1 & 2	Tabletti	1024 x 768	1	132
Apple iPad 3 & 4	Tabletti	2048 x 1536	2	264
Apple iPad Mini	Tabletti	1024 x 768	1	163
Apple iPhone	Puhelin	320 x 480	1	163
Apple iPhone 4	Puhelin	640 x 960	2	326
Apple iPhone 5	Puhelin	640 x 1136	2	326
Nokia Lumia 710	Puhelin	480 x 800	1.5	252
Samsung Galaxy Y	Puhelin	240 x 320	0.75	133
Samsung Galaxy Gio	Puhelin	320 x 480	1	164
Samsung Galaxy S II	Puhelin	480 x 800	1.5	219
Samsung Galaxy S III	Puhelin	720 x 1280	2	306
Samsung Galaxy S4	Puhelin	1080 x 1920	3	441
Samsung Galaxy Note	Puhelin	800 x 1280	2	285
Samsung Galaxy Tab	Tabletti	1024 x 600	1	171
Samsung Galaxy Tab 10.1	Tabletti	1280 x 800	1	149

Vanhemmissa laitteissa yleinen resoluutio on 320 x 480 tai 480 x 800 pikseliä. Uudemmissa puhelimissa resoluutio alkaa lähennellä tabletteja: Samsung Galazy S III ja Samsung Galaxy Note ovat resoluutioltaan vastaavia kuin Samsung Galaxy Tab 10.1 tabletti. Puhelimien fyysinen koko on kuitenkin pienempi kuin tabletin, tämä johtuu suuremmasta PPI arvosta. Näissä puhelimissa verkkosivut eivät kuitenkaan näytä samalta kuin resoluutioltaan samassa tabletissa, koska puhelin kertoo selaimelle todellista pienemmän resoluution.

Ilmoitettavaan resoluutioon vaikuttaa laitteen pikselisuhde, eli kuinka monta normaalia pikseliä laitteen yksi pikseli vastaa. Pikselisuhteeseen vaikuttaa laitteen PPI-arvo. Esimerkiksi iPhone 5:n PPI on 326, ensimmäisen iPhonen PPI on 163.

Uudemman iPhonen PPI-arvo on kaksinkertainen vanhaan, joten pikselisuhde on 2. Kyseisen iPhonen resoluutio on siis puolet ilmoitetusta: 320 x 568.

Vaikka uudempien laitteiden resoluutiot suurenevat, ei se tarkoita että kuluttajat siirtyisivät heti puhelimensa uudempaan malliin. Lisäksi suurempi resoluutio tarkoittaa yleensä suurempaa pikselisuhdetta, jolloin selaimelle ilmoitettava resoluutio kuitenkin on pienempi kuin laitteen todellinen resoluutio. Kuviosta 1 voidaan huomata, että helmikuussa 2013 Suomessa käytettiin eniten mobiililaitteita, joiden resoluutio oli 320 x 480 tai 480 x 800.



Kuvio 1. Yleisimmät mobiililaitteiden resoluutiot Suomessa helmikuussa 2013 (StatCounter 2013)

2.4 Käyttöjärjestelmät

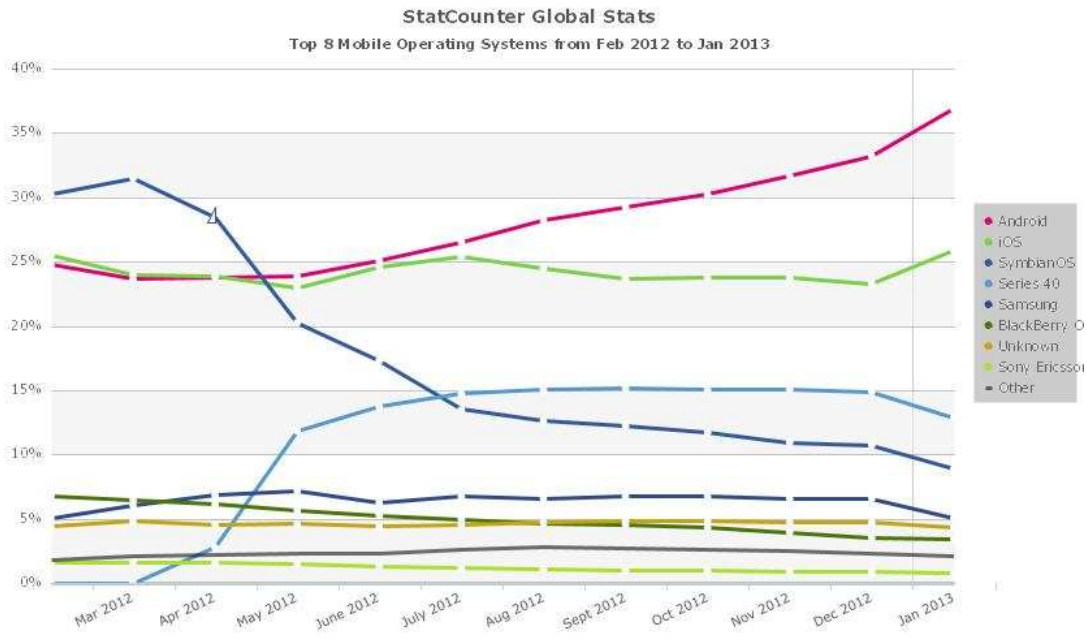
2.4.1 Yleistä

Suomen markkinoilla kolme yleisintä käyttöjärjestelmää ovat Android, iOS sekä Windows Phone. Windows Phone on selkeä altavastaaja, mutta luultavasti Nokian ansiosta se on saavuttanut Suomessa kohtalaisen aseman. Muita mainittavia käyttöjärjestelmiä Suomen näkökulmasta ovat Nokian kehittämä Symbian OS, joka oli

vielä vuonna 2010 suositumpi kuin Android. (STT-REUTERS 2011.) Vaikka Symbian on jo haudattu, ei se tarkoita, etteikö sitä edelleen käytettäisi.

Nokia ja Intel kehittivät MeeGoa, johon perustuvia laitteita ehdittiin julkaista muutama. Vuoden 2011 alussa Nokia julkaisi alkavansa valmistaa matkapuhelimiaan Windows Phone 7 -alustalle, joten MeeGon kehitys lakkautettiin. MeeGosta ovat lähtöisin suomalaisen Jolla Oy:n Sailfish-alusta sekä Intelin kehittämä Tizen, mutta kumpaakaan ei ole vielä virallisesti julkaistu.

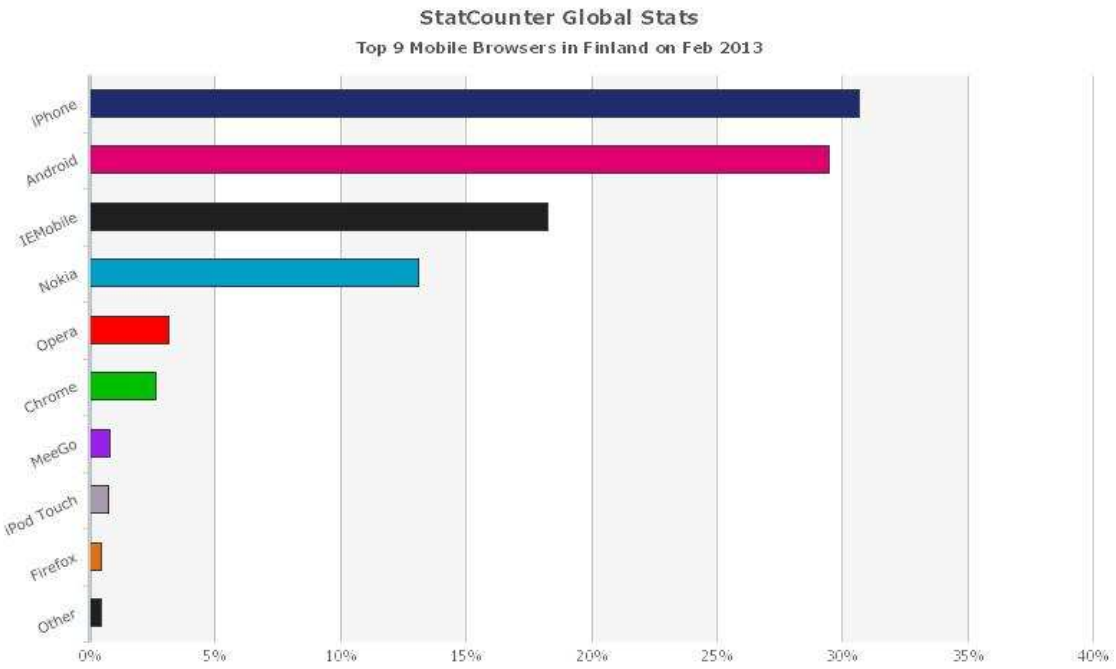
Muita mainittavia käyttöjärjestelmiä ovat BlackBerry Limitedin (entinen RIM) BlackBerry sekä Samsungin Bada. BlackBerry ei koskaan ole saapunut Suomen markkinoille suuressa mittakaavassa, eikä Samsung ole jatkanut Badan kehittämistä suurella tehokkuudella. Kuviossa 2 on kuvattu kahdeksan suosituimman käyttöjärjestelmän suosiota maailmanlaajuisesti. Symbianin hurja suosio ja sen dramaattinen lasku keväällä 2012 johtuu siitä, että StatCounter laski kaikki Nokian laitteet käyttöjärjestelmästä huolimatta Symbian OS:in alle.



Kuvio 2. Mobiilikäyttöjärjestelmien suosio maailmanlaajuisesti viimeisen vuoden aikana (StatCounter 2013)

Web-kehityksessä käyttöjärjestelmien eroavaisuudet tulevat esiin lähinnä järjestelmässä olevan selaimen myötä. Kehittäjät ovat useimmiten tottuneet siihen,

että desktopissa selainten määrä rajoittuu viiteen: Firefox, Google Chrome, Internet Explorer, Opera sekä Safari. Mobiililaitteisiin on kuitenkin tarjolla paljon enemmän vaihtoehtoja, pahimmassa tapauksessa ladattavissa on 35 eri selainta. (Smashing Magazine 2012.) Koska saatavilla olevien selaimien määrä on rajaton, ei niiden kaikkien tukemisesta ja testaamisesta kannata edes haaveilla. Kuten kuvio 3 osoittaa, että käytetyimmät selaimet ovat käyttöjärjestelmien oletusselaimia.



Kuvio 3. Mobiiliselainten käyttöprosentit Suomessa helmikuussa 2013 (StatCounter 2013)

Selaimia on kahta tyyppiä: proxy-selaimet, joista tunnetuin on Opera Mini, sekä täydet selaimet. Kun normaali selain renderöi sivun ja näyttää sen, proxy-selain käyttäytyy hieman eri tavalla. Selain lähettää pyynnön proxy-serverille, joka renderöi sivun ja tekee siitä kuvan, joka näytetään selaimessa. Käyttäjä pystyy normaalisti klikkaamaan sivulla olevia linkkejä ja valitsemaan tekstiä. Proxy-selaimet ovat tavallisia selaimia nopeampia; tiivistetty kuva on paljon kevyempi kuin täysi verkkosivu. Ne eivät kuitenkaan takaa yhtä laadukasta käyttökokemusta kuin normaalit selaimet, mutta ovat mainio vaihtoehto niille, joilla ei ole varaa älypuhelimeen tai nopeaan yhteyteen. (Smashing Magazine 2012.) Proxy-selaimen käyttäytyminen verrattuna tavalliseen selaimeen voidaan nähdä kuviosta 4, jossa

vasemmalla on proxy-selain ja oikealla normaali selain Samsung Galaxy Note 10.1 tabletissa.

Kuvio 4. Proxy-selaimen ja normaalin selaimen ero

2.4.2 Android

Android on Googlen kehittämä avoimen lähdekoodin käyttöjärjestelmä, joka julkaistiin vuonna 2008. Android kiinnostaa laitevalmistajia, koska se on ilmainen ja siihen on mahdollista tehdä muutoksia. Tästä johtuen eri valmistajan puhelimessa oleva Android-versio voi näyttää erilaiselta.

Android kulkee jo versiossa 4.2, joka julkaistiin marraskuussa 2012. Silti helmikuussa 2013 tehdyn tutkimuksen mukaan 45,4 % Android käyttäjistä käyttää järjestelmäversiota 2.3.3–2.3.7. Kyseiseen Gingerbread-versioon ei ole tullut merkittäviä parannuksia sitten vuoden 2011 heinäkuun. (Platform Versions 2013.)

Androidin puhelinmalleille on yleistä, että puhelimen käyttöön tarvittavat näppäimet on sijoitettu laitteen yhteyteen sen alareunaan. Tällöin oletusselainta käytettäessä

(ks. kuvio 5) näytössä ainoana kiinteänä osana on puhelimen yläreunassa oleva infopalkki sekä selaimen osoiterivi, joka menee piiloon.

Kuvio 5. Samsung Galaxy Gion (Android 2.2.1) oletusselain

2.4.3 iOS

iOS on Applen kehittämä käyttöjärjestelmä iPhoneen, iPadiin sekä iPodiin. iOsin ensimmäinen versio julkaistiin 2007.

iOSin käytetyin versio on huomattavasti tuoreempi kuin Androidin. Helmikuussa 2013 käyttöjärjestelmän version 6.X osuus oli 82,7 %. (Smith 2013.) Järjestelmäversio iOS 6 julkaistiin syyskuussa 2012. Koska Apple on ainoa iOS:in jakelija, pystyy se tarjoamaan uuden järjestelmäpäivityksen kaikille laitteille, lukuun ottamatta aivan ensimmäisiä versioita laitteista.

iPhonen oletusselaimessa, Safarissa, on näytön alalaitaan kiinnitetty selaimen toimintaan liittyviä painikkeita, jotka ovat koko ajan näkyvissä (kts. kuvio 6). Osoiterivi menee piiloon, kun sivua selataan alaspäin.

Kuvio 6. iPhone 4 oletusselain Safari

2.4.4 Windows Phone

Windows Phone on Microsoftin vastine mobiilikäyttöjärjestelmälle. Käyttöjärjestelmä julkaistiin 2010 ja kuten Androidilla, silläkin on useampia valmistajia. Windows Phonen uusin versio 8 julkaistiin marraskuussa 2012.

Windows Phone on laitevalmistajalle maksullinen, eikä siihen saa tehdä muutoksia, toisin kuin Androidiin. Tästä syystä Windows Phone ei kiinnosta valmistajia yhtä paljon kuin Android.

Nokian Lumia 800 puhelimessa käyttöjärjestelmänä on Windows Phone 7 ja oletusselaimena Internet Explorer. Osoiterivi on sivun alalaidassa ja se pysyy koko ajan näkyvissä (kts. kuvio 7).

Kuvio 7. Nokia Lumia 800 oletusselain

3 Responsiivinen web-kehitys

3.1 Mitä on responsiivisuus?

Mobiiliystävällisten sivujen tekemiseen on kaksi vaihtoehtoa: joko tehdään mobiililaitteille täysin oma sivusto tai toteutetaan sivu responsiiviseksi, eli sisältö mukautuu päätelaitteen näytön leveyden mukaan. Responsiivisuuden sijasta voidaan käyttää termejä kuten ”mukautuva” tai ”selainriippumaton”. Tässä opinnäytetyössä käytetään kuitenkin termiä ”responsiivinen”, koska se on suorin käännös sen alkuperäisestä termistä: RWD (Responsive Web Desing).

Se, tehdäänkö sivusta responsiivinen vai erillinen mobiilisivusto, riippuu täysin verkkosivusta, sen kohderyhmästä sekä käyttötarkoituksesta. Aina mobiililaitteiden tukeminen ei ole edes järkevää. Responsiivisuus on usein hyvä vaihtoehto pienille tai

keskisuurille sivustoille, jotka haluavat palvella myös mobiilikäyttäjiä. Erillisen mobiilisivun hyviä puolia on se, että siitä saa helpommin applikaatiomaisen, kosketusnäytön ominaisuuksia hyödyntävän sivuston. Usein erillinen mobiilisivu on sellainen, joka tarjoaa käyttäjälle vain osan sisällöstä, kuten toimipaikat ja aukioloajat. Tämä ei kuitenkaan ole käyttäjää palveleva ratkaisu, etenkin jos pääsyä kokonaiselle sivulle ei ole mahdollistettu.

3.2 Responsiivisen suunnittelun periaatteet

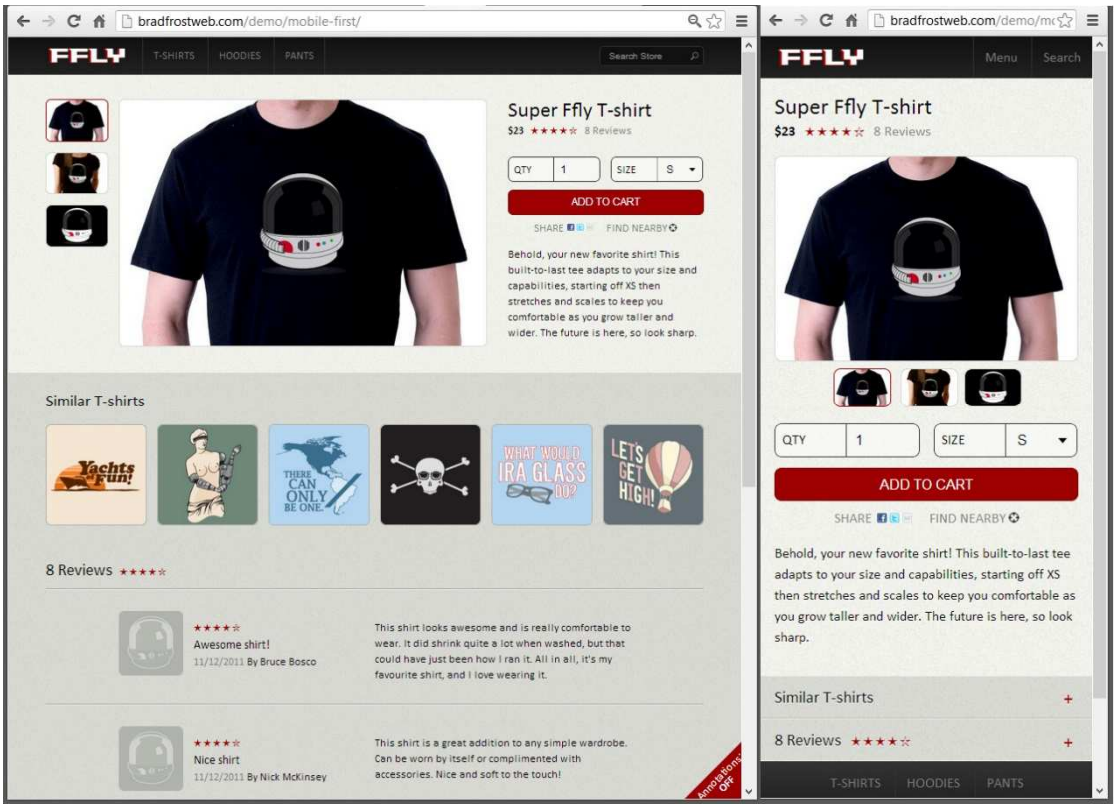
Responsiivisen suunnittelun virhe on useimmiten siinä, että ajatellaan liikaa fyysisiä laitteita ja sitä, miltä verkkosivu tulisi näyttämään juuri kyseisessä laitteessa. Suunnittelun pitäisi lähteä liikkeelle sisällön mukautumisesta selainikkunassa, oli ikkunan leveys mikä tahansa.

Responsiivisen suunnittelun yhteydessä puhutaan usein suunnittelutyylistä mobile first. Tämä tarkoittaa sitä, että suunnitellaan ensin kaikkein pienin näkymä, johon sijoitetaan sisältö sijoitetaan HTML/CSS:n avulla. Tämän jälkeen lähdetään venyttämään selainikkunaa ja tarkastellaan, milloin sisältö ei enää mukaudu halutulla tavalla. Tähän selaimen leveyteen lisätään breakpoint eli muutospiste ja sisältö järjestetään uudelleen. Selaimen venyttämistä ja breakpointtien lisäämistä jatketaa, kunnes sisältö on saavuttanut maksimileveytensä. Breakpointteja ei kannata lisätä liian ankaralla kädellä, sillä useampi breakpoint tarkoittaa useampaa riviä CSS:ää, joka puolestaan kasvattaa tiedoston kokoa. Lisäksi useiden breakpointtien hallitseminen on hankalaa, etenkin jatkokehitystä ajatellen.

Suunnittelussa ongelmana on usein sisällön priorisointi. Kapeimmissa näyttökooissa sisältö pystyy harvemmin olemaan vierekkäin, joten täytyy tehdä päätös siitä, mikä sisältö näytetään ensin. Lähtökohtaisesti mobiilikäyttäjältä ei tulisi piilottaa mitään sellaista sisältöä, joka on saatavilla desktop-versiossa. Kuten Josh Clark sanoo: ”Saying mobile design should have less is like saying paperbacks have smaller pages, so we should remove chapters” (Myers 2011).

Sisällön priorisointi ei ole ainoa ongelma, täytyy myös ottaa huomioon kuinka sisältö näytetään. Mikäli sisältöä on paljon, ei paras tapa ole latoa kaikkea sisältöä toistensa

päälle, sillä silloin sivun pituus saattaa kasvaa liian pitkäksi ja käyttäjällä on hankaluuksia löytää haluamansa sisältö. Tällöin voi olla järkevää piilottaa osa sisällöstä, mutta antaa käyttäjälle mahdollisuus saada se esiin esimerkiksi erillisen painikkeen kautta (kuvio 8).



Kuvio 8. Sisällön vahtoehtoinen näyttäminen (Frost, n.d.)

Suunnittelussa on otettava huomioon, että kosketusnäytöllä linkkejä on hankalampi painaa, koska sormi on suurempi kuin hiiren kursori. Kuviossa 9 on esitetty eri yhtiöiden ja tutkimusten näkemystä siitä, kuinka suuri linkin painettavan alueen tulisi olla. Linkin on myös selkeästi näytettävä linkiltä, koska mobiilikäyttäjät eivät voi hyödyntää hover-efektiä samalla tavalla kuin desktop-käyttäjät.

Kuvio 9. Suosituksia linkin painettavan alueen koosta (Anthony T 2012)

3.3 Tekniikat

3.3.1 Yleistä

Responsiivisuuden kulmakiviä ovat HTML:n viewport meta tagi, mukautuvat kuvat sekä CSS:n mahdollistamat media queryt. Näiden ominaisuuksien avulla verkkosivu saadaan skaalautumaan päätelaitteen näytölle sopivaksi, eikä käyttäjän tarvitse zoomata nähdäkseen sivun sisällön kokonaan. Lisäksi HTML5 sekä CSS3 ovat mahdollistaneet yksinkertaisemman ja paremman lähestymistavan, joka tekee web-kehityksestä entistä tehokkaampaa ja mutkattomampaa.

Kuviossa 10 vasemmalla on verkkosivu, joka ei ole responsiivinen. Oikealla on responsiivinen versio kyseisestä sivusta. Sivun ulkonäkö päätelaitteen leveydessä on määritelty media queryjen avulla ja viewport meta tagin avulla responsiivinen sivu on määritetty näkymään ilman zoomausta, päätelaitteen leveyden mukaisesti.

Kuvio 10. Ei-responsiivisen ja responsiivisen sivun ero

3.3.2 Viewport

Ilman viewport meta tagia mobiililaite skaalaa verkkosivun mahtumaan laitteen näytölle ja mahdollistaa sisällön zoomaamisen. Normaalilla selaimella näin ei käy: mikäli sivusto on leveämpi kuin selainikkuna, ikkunan alareunaan tulee vierityspalkki, jolla näytölle mahtumattoman sisällön voi vierittää näkyviin. Viewport-tagilla pystytään hallitsemaan sitä, miten mobiililaite sovittaa sivun näytölle ja kuinka käyttäjän annetaan zoomata sitä.

Viewport meta tagi määritetään head-elementin sisälle. Seuraava viewport määrittää sivun oletus- ja minimiskaalaukseksi yhden, mahdollisuuden skaalata sivua kaksinkertaiseksi sekä mahdollistaa sivun skaalaamisen:

```
<meta name="viewport" content="width=device-width,
initial-scale=1.0, minimum-scale=1.0, maximum-scale=2.0,
user-scalable=yes">
```

Viewportin ominaisuudet on kuvattu taulukossa 2.

Taulukko 2. Viewportin ominaisuudet

Ominaisuus	**Selitys**
Width	Leveys, johon sivusto skaalataan. Yleensä leveydeksi määritetään *device-width*, jolloin sivusto skaalautuu päätelaitteen mukaan. Leveydeksi voidaan antaa myös pikseliarvo, mutta se ei ole kannattavaa, ellei sivustoa ole suunniteltu toimivaksi vain tietyn levyisillä laitteilla.
Height	Korkeus, johon sivusto skaalataan. Mikäli korkeutta ei määritetä, se muodostuu automaattisesti. Korkeutta kannattaa käyttää vain silloin, mikäli halutaan estää sivuston scrollaaminen vertikaalisti.
Initial-scale	Mihin kokoon sivu skaalataan. Oletus on 1, jolloin sivu näyttää normaalilta.
Maximum-scale	Kuinka paljon käyttäjän annetaan zoomata sivustoa lähemmäksi. Mitä korkeampi luku, sitä enemmän zoomausta sallitaan. Mikäli luvuksi määritetään 1, jotkut laitteet estävät zoomaamisen kokonaan.
Minimum-scale	Kuinka paljon käyttäjän annetaan zoomata sivustoa kauemmaksi.
User-scalable	Sallitaanko käyttäjälle mahdollisuus zoomata sivua. Oletuksena zoomaus ”yes” eli sallittu, mutta se voidaan estää määrittämällä ”no”.

Tulee tarkkaan harkita, sallitaanko käyttäjälle sivuston zoomaaminen vai ei. Zoomauksen estäminen voi olla haitallista käyttäjille, joilla on näköongelmia:

oletuskoossa fontti ei välttämättä ole sen kokoista, jota kaikki näkevät lukea. Zoomaus estetään yleensä siksi, ettei käyttäjä vahingossa suurenna sivua ja koska koetaan, ettei zoomaaminen ole tarpeellista. Mikäli zoomaus kuitenkin halutaan estää, kannattaa se tehdä määrittämällä minimi- ja maksimiskaalaukset yhteen. Mikäli skaalaus estetään user-scalable-attribuutin avulla, selaimet jotka oletusarvoisesti eivät tule CSS:n position fixed-ominaisuutta, ottavat kyseisen ominaisuuden käyttöön (Avola & Raasch 2013). Tämä ei ole suotavaa, sillä mikäli ominaisuutta on käytetty, se ei toimi kyseisissä selaimissa oikein.

3.3.3 CSS3

CSS3 on Cascading Style Sheetsin uusin versio. CSS:llä annetaan HTML:lle tyyliohjeita, joiden mukaan se tyylittelee itsensä. CSS3 sisältää paljon uusia ominaisuuksia, jotka nopeuttavat kehittäjän työtä ja parantavat sivun suorituskykyä. Valitettavasti vanhemmat selaimet, kuten Internet Explorer 8, eivät näitä ominaisuuksia tue. Siksi verkkosivu tulisi suunnitella sillä periaatteella, että CSS3:n uudet ominaisuudet tuovat siihen vain lisäarvoa, eivätkä ole suunnittelun perusta. Vanhempia selaimia ei kuitenkaan pidä hyysätä liikaa, sillä niiden käyttäjäkunta on koko ajan laskemaan päin. Lisäksi esimerkiksi Twitter on päättänyt tiputtaa tweet- ja seurauspainikkeiden tuen Internet Explorerin versioille 6 ja 7 (Ward, 2013).

CSS3:sen hyödyllisimmät ominaisuudet ovat liukuväri, varjot, pyöristetyt kulmat, taustakuvat, parantuneet valitsimet, transitiot sekä transform. Enää pyöristettyjä kulmia, varjoa tai liukuväriä ei tarvitse tehdä kuvana. Taustakuvia pystyy kiinnittämään samaan elementtiin useita ja taustan hallitsemiseen on enemmän mahdollisuuksia. Transformin avulla pystytään elementtejä hallitsemaan monipuolisemmin, esimerkiksi kääntämään. Transition puolestaan antaa mahdollisuuden tehdä elementille yksinkertaisia animaatioita ilman JavaScriptiä. Parantuneiden valitsimien avulla voidaan määrittää eri tyyli esimerkiksi joka toiselle elementille.

Etenkin valitsimien käytössä kannattaa olla tarkkana, ettei tiputa pois tukea vanhemmilta selaimilta. Esimerkiksi MEOMin sivuilla on määritetty nth-child()-valitsimen avulla, että kolme ensimmäistä Instagram-kuvaa ovat muita suurempia

(kuvio 11), mutta Internet Explorer 8 ei tue tätä selainta. Tässä tapauksessa välitöntä harmia ei tapahdu, vaikka kolme ensimmäistä kuvaa eivät suurene halutulla tavalla.

Kuvio 11. Esimerkki sivusta, jossa CSS:n valitsimet eivät ole täysin tuettuja (MEOM 2012)

Mikäli jonkin CSS:n ominaisuuden tuki mietityttää, voi sen tarkistaa Can I use… (2013) verkkosivulta. Etenkin fixed-positionia, eli elementin kiinnittämistä pysyvästi tiettyyn kohtaan, kannattaa välttää. Se ei aina käyttäydy halutulla tavalla vanhemmissa mobiiliselaimissa, vaan pahimmassa tapauksessa lähtee nykien mukaan käyttäjän skrollatessa sivua.

3.3.4 Mittayksiköt

CSS:ssä on useita vaihtoehtoja sille, mitä mittayksikköä käytetään. Pikselin käyttäminen yksikkönä on yleistä, koska se on näyttöpäätteissä kuvaavin yksikkö. Selain asettaa tekstille oletuskoon, yleensä 16 pikseliä. Koko voi kuitenkin vaihdella, esimerkiksi Kindle Touchin tekstin oletuskoko on 26 pikseliä. (Kadlec 2013.)

Tekstin koon ja rivivälistyksen määrittämisessä käytetään usein mittayksikköä em. Se on relatiivinen yksikkö, mikä tarkoittaa sitä, että se perii arvon ylemmältä elementiltä ja käyttäytyy suhteessa siihen. Esimerkiksi bodylle annetaan tekstin kooksi 100 % eli selaimen oletuskoko, joka yleensä on 16 pikseliä. Elementti p eli parahgraph on bodyn alempi elementti. Mikäli sen tekstin kooksi määritetään 0.9 em, se on 90 % tekstin oletuskoosta eli 14 pikseliä.

Em:n hyöty tulee esiin etenkin silloin, jos halutaan muuttaa kaikkien tekstien kokoa suhteessa toisiinsa. Tällöin muutetaan vain bodyn lähtökohtaista tekstin kokoa, esimerkiksi 150 prosenttiin.

3.3.5 Media query

Media query on CSS:n ominaisuus, jonka avulla voidaan määrittää eri CSS-tyylejä eri mediatyypeille tai ominaisuuksille. Media queryn perusmuoto on (Kadlec 2013):

```
@media [not|only] type [and] (expression) { /* Tyylit */ }
```

Kaikki vanhemmat selaimet eivät täysin tue media queryjä ja yrittävät suorittaa ne, vaikka ehto ei vastaisikaan todellisuutta. Tämä voidaan estää laittamalla media queryyn avainsana ’only’. Tälloin vanhat selaimet ohittavat media queryn, mutta niitä täysin tukeva selain suorittaa media queryn normaalisti.

Mediatyypit

Mahdollisia mediatyyppejä ovat mm. print eli tulostin tai tulostuksen esikatselu, handheld eli kädessä pidettävä laite, tv eli televisio sekä screen eli tietokoneen näyttö. Yleisimmin käytetyt mediatyypit ovat all, screen ja print. Kyseisiä mediatyyppejä käytetään, vaikka kohdelaitteelle olisikin oma mediatyyppi. Tästä syystä useat laitteet ovat alkaneet tukea tyyppiä screen, vaikka ne eivät sitä virallisesti olisikaan. (Kadlec 2013.) Mediatyypille print suunnattu media query näyttää tältä:

```
@media only print { /* Tyylit */ }
```

Voisi kuvitella, että mediatyyppi handheld kohdistaisi tyylit mobiililaitteille. Näin ei kuitenkaan ole, osa mobiiliselaimista tukee vain mediatyyppiä handheld, osa vain screenia, ja osa tukee molempia. Dominique Hazaël-Massieux (2009) listasi

artikkelissaan, mitkä mobiiliselaimet tukevat mitäkin mediatyyppiä. Vuonna 2009 pelkän handheldin tuki nykyaikaisissa mobiiliselaimissa oli melko olematon, joten kaiken kattavana mediatyyppinä voidaan pitää screeniä.

Ominaisuudet

Media queryillä voidaan määrittää tyylejä myös laitteen ominaisuuksien mukaan. Etenkin responsiivisuudessa yleisin käytetty ominaisuus on leveys eli width, joka tarkoittaa selaimen leveyttä. Muita käytettäviä ominaisuuksia ovat esimerkiksi selainikkunan korkeus (height), ikkunan mittasuhde (aspect-ratio), ikkunan orientaatio (orientation) sekä laitteen resoluutio (resolution) ja pikseleiden suhde (device-pixel-ratio). Kyseisiin ominaisuuksiin, orientaatiota lukuun ottamatta, voidaan määrittää minimi- ja maksimi-arvoja.

Device-pixel-ratiossa on otettava huomioon, että se ei ole varsinainen standardi. Jotkut selaimet kuitenkin tukevat sitä, jolloin mahdollistetaan korkearesoluutioisten näyttöjen tunnistaminen. Device-pixel-ratio tarvitsee toimiakseen etuliitteen, kuten -webkit tai -moz, eikä Internet Explorer -selaimessa ole sille tukea. (Jankord, 2012)

Ominaisuuksista width, height ja aspect-ratio on olemassa myös device-ominaisuudet, kuten device-width. Ero tavalliseen leveyteen on se, että device tarkoittaa koko laitteen leveyttä, ei selainikkunan leveyttä. Esimerkiksi: näyttö on 1024 pikseliä leveä ja on määritetty seuraava media query:

```
@media only screen and (max-width: 900px) {
       body {background: #ccc;}
}
```

Tyyli esiintyy leveydessä 900 pikseliä ja tulee käyttöön, jos selainikkunaa kavennetaan alle 900 pikselin leveyden. Mikäli sama tyyli määriteltäisiin max-device-width-ominaisuudella seuraavasti:

```
@media only screen and (max-device-width: 900px) {
       body {background: #ccc;}
}
```

Tyyli ei tule 1024 pikseliä leveässä näytössä käyttöön, vaikka selainikkunaa pienennettäisiin. Useimmiten kannattaa käyttää ominaisuutta width, koska se

tarkkailee selainikkunan leveyttä, joka on juuri se alue joilla sivut sijaitsevat. Vaikka useimmiten selainikkuna on laitteen näytön levyinen, pystyy joissain laitteissa selainikkunaa pienentämään. Lisäksi width laukaisee breakpointit myös desktop-selaimessa, joten kaikkea testausta ei tarvitse tehdä kohdelaitteella.

Breakpointit

Breakpointeilla tarkoitetaan niitä pisteitä, joissa selain ottaa vaihtoehtoiset tyylit käyttöön. Breakpointit määritetään selainikkunan leveyden mukaan aina silloin kun koetaan, että sisältö ei mukaudu enää sulavasti. Niin sanotuiksi standardeiksi mobiililaitteiden suhteen ovat muodostuneet breakpointit 320 px eli puhelimen pystykoko, 480 px eli puhelimen vaakakoko sekä 768 px eli tabletin pystykoko. (Kadlec 2013.) Vaikka nämä breakpointit kuvastavatkin keskimääräisesti laitteiden näyttökokoja eri asennoissa, ei niiden varaan kannata tuudittautua, sillä kohdelaite voi olla minkä kokoinen tahansa. Niitä ei siis kannata pitää vakioina, vaan ohjenuorana.

Breakpointteja määritettäessä tulee tarkkaan miettiä, mitä yksikköä niiden määrittämiseen käytetään. On kahdenlaisia pikseleitä: laitteen pikselit sekä CSS:n pikselit. Laitteen pikselit kuvastavat selainikkunan kokoa ja pysyvät vakiona, vaikka sivua zoomattaisiin. CSS:n pikselit puolestaan kasvavat sitä mukaan, kun sivua zoomataan lähemmäksi. (Kadlec 2013.) CSS:n pikselit ovat niitä, joita media queryt kuuntelevat. Käyttäjä voi saapuu sivulle tilassa, jossa hän on määrittänyt zoomauksen lähemmäs, tällöin CSS:n pikselit ovat kasvaneet. Tämä voi tapahtua sekä mobiili- että desktop-selaimessa.

Kuvio 12 havainnoillistaa verkkosivua, jossa on määritetty breakpoint selaimen maksimileveydelle 656 pikseliä. Ennen kyseistä breakpointtia, headerille on annettu seuraavat määreet CSS:n avulla:

```
header {
    background: url(images/header.jpg) no-repeat center #000;
    height: 367px;
}
```

Kun breakpoint astuu voimaan selaimen leveyden ollessa 656 pikseliä, seuraava media query määrittää headerille uuden korkeuden:

```
@media only screen and (max-width: 656px) {
    header { height: 111px; }
}
```

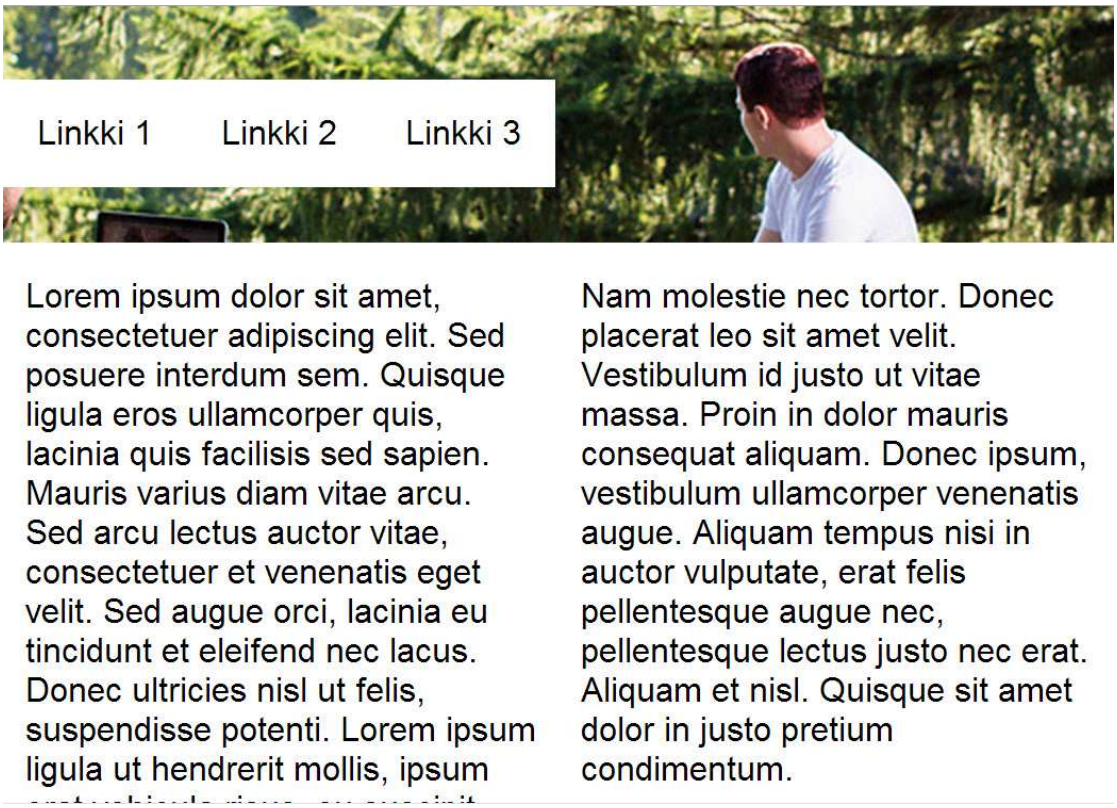
Kuvio 12. Pikseleillä määritelty breakpoin ilman zoomaamista

Breakpoin toimii hienosti, kun sivua ei ole zoomattu. Oletetaan, että käyttäjä saapuu sivulle selaimen leveyden ollessa yli 656 pikseliä, mutta zoomauksen ollessa 200 %. Häntä odottaa kuvion 13 mukainen näkymä.

Kuvio 13. Verkkosivu 200 % zoomauksella, pikseleillä määritetty breakpoint

Lopputulos ei ole toivottu, sillä headerin kuva vie sivulta liikaa tilaa pystysuunnassa. Mikäli breakpoint määritellään em:nä, päästään kuvion 14 osoittamaan lopputulokseen, joka on parempi. Pikselit on helppo muutaan em:iksi PX to EM muuntimella (Cray n.d.). Kyseisen muuntimen avulla laskettiin, että 656 pikseliä on 41 em:ää, jolloin media query määritetään uudelleen kyseisellä arvolla:

```
@media only screen and (max-width: 41em) {
    header {height: 111px;}
}
```



Kuvio 14. Verkkosivu 200 % zoomauksella, em:llä määritetty breakpoint

Haluttu breakpoint tuli tässä tapauksessa käyttöön. Aina ei kuitenkaan voida olla varmoja, milloin sivu käyttäytyy zoomatessa halutulla tavalla. Breakpointtien määrittäminen em:ien avulla on kuitenkin askel parempaan. Täytyy myös ottaa huomioon, että breakpointit eivät vaikuta, mikäli sivua zoomataan latauksen jälkeen. Ne vaikuttavat vain silloin, kun zoomaus on päällä sivulle saavuttaessa.

3.3.6 HTML5

HTML5 on viides versio HTML:stä, mutta sitä ei ole vielä standardisoitu. Useimmat selaimet kuitenkin tukevat HTML5:n ominaisuuksia, joten kehittäjät ovat alkaneet käyttää niitä. Termiä HTML5 käytetään usein väärin viitatessa sivuihin, jotka hyödyntävät uusien tekniikoiden, kuten CSS3, geolocation ja client storage, tuomia mahdollisuuksia (Way 2011). Tämä on virheellistä, sillä HTML5 ei itsessään tee ihmeitä. Se tarvitsee tuekseen muita tekniikoita, kuten JavaScriptiä.

HTML5 esitteli uusia median hallintaan liittyviä elementtejä, kuten <video>, <audio> ja <canvas>. Semantiikkaa parannettiin elementtien, kuten <article>, <header> ja <nav> myötä. Nämä elementit ovat vastineita div-elementille ja niiden avulla nähdään suoraan, mikä osa verkkosivusta on mikäkin. Tällöin merkkauksen struktuuria on helpompi ymmärtää. (HTML Reference - (HTML5 Compliant), n.d.) Litteessä 1 on esitelty HTML5:n uudet semantiikkaan liittyvät elementit, jotka ovat tuettuja yleisimpien selaimien uusimmissa versioissa.

Kaikki selaimet eivät tue HTML5:n uusia elementtejä. Tämä ilmenee siinä, etteivät ne ota vastaan kyseisille elementeille asetettuja CSS-tyylejä. HTML5 Shiv (html5shiv 2013) mahdollistaa JavaScriptin avulla HTML5-elementtien käyttämisen selaimissa, jotka eivät niitä tue. Näitä selaimia ovat Internet Explorer 6-9, Safari 4.x (iPhone 3.x) sekä Firefox 3.x.

Vaikka HTML5 tarjoaa sematiikkaan parannuksia, ei vanhan tutun div-elementin käyttäminen ole paheksuttavaa. Parannetun sematiikan hyödyt ovat vielä varsin avonaisia, eihän HTML5 ole vielä edes varsinainen standardi. On kuitenkin järkevää tutustua tulevaan standardiin jo nyt, ettei siihen siirtyminen ole myöhemmin kynnyskysymys.

3.3.7 Fontit

CSS3:n @font-face on mahdollistanut muidenkin kuin järjestelmäfonttien käyttämisen verkkosivuilla. Sen avulla CSS-tiedostoon liitetään fonttitiedosto, jota verkkosivu pystyy käyttämään. Tuki font-facea kohtaan on melko laaja, lukuun ottamatta Opera Miniä sekä useiden selaimien erittäin vanhoja versioita (Can I use...

2013). Selaimille täytyy kuitenkin tarjota fontista oikea muoto (EOT, WOFF, SVG tai TTF/OTF), jotta se osaisi käsitellä sitä. Vaikka Windows Phone 7:ssa olevassa Internet Explorer 9 -selaimessa on tuki font-face-ominaisuudelle, se ei kuitenkaan käytä fontteja oikein, vaan jättää ne lataamatta (Thebeebs 2011).

On olemassa useita eri palveluita, ilmaisia sekä maksullisia, jotka tarjoavat web fontteja. Tunnetuin ilmaisista web fonttien tarjoajista on Google web fonts (n.d.), joka tarjoaa tällä hetkellä 625 eri fonttia.

Fontit eivät kuitenkaan näytä jokaisella laitteella samalta, sillä eri käyttöjärjestelmät ja selaimet voivat renderöidä ne eri tavalla. OS X pyrkii näyttämään fontin sellaisena millaiseksi se on suunniteltu, vaikka tuloksena olisi fontin sumentuminen. Windows puolestaan pyrkii näyttämään fontin mahdollisimman terävänä, jotta sen luettavuus olisi parempi. Tällöin fontti saattaa muuttua epätasaiseksi ja näyttää huonolta (kts. kuvio 15). (Atwood 2007.) Windowsin käyttäytyminen riippuu myös selaimesta, sillä Interner Explorerin uusimmat versiot renderöivät fontit paremmin kuin esimerkiksi Google Chrome tai Mozilla Firefox.

Kuvio 15. Google web fontsin tarjoama Montserrat-fontti Plazan etusivulla eri selaimissa

Web fontteja käyttäessä tulee olla erityisen varovainen silloin, kun niitä käytetään leipätekstissä. Mikäli fontti renderöityy huonosti ja se vaikeuttaa lukemista, käyttäjä

tuskin viitsii lukea tekstiä loppuun. Tekstin ulkonäköä voi yrittää muuttaa CSS:n avulla, mutta ensin kannattaa yrittää muuttaa sen kokoa ja leikkauksia. Etenkin liian pieni teksti voi näyttää rosoiselta, mutta muuttuu paremman näköiseksi mikäli sen kokoa suurennetaan. Samoin leikkauksissa on eroa, lihavoitu teksti voi näyttää paremmalta kuin puolilihavoitu teksti.

CSS:n text-shadow-ominaisuuden avulla saadaan joissain tapauksissa teksti näyttämään pehmeämmältä. WebKit-selaimissa yhtenä ratkaisuna on määrittää tekstille text-stroke eli reunus, jolloin teksti näyttää huolitellummalta. Tämä jättää ongelman kuitenkin niille selaimille, jotka eivät ole WebKit-selaimia. (Panique 2013.)

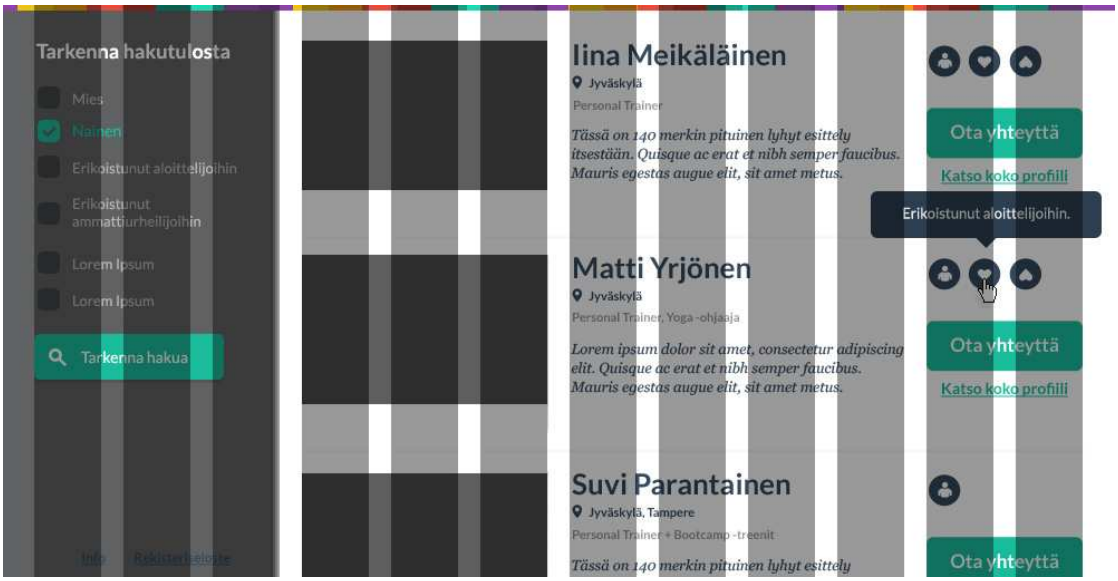
Web fontit kasvattavat typografian mahdollisuuksia verkkosivuilla, mutta niitä tulisi silti käyttää maltillisesti. Ne kasvattavat sivun kokoa ja hidastavat latautumista, etenkin kun käytetään kolmannen osapuolen palveluita (Mobile Best Practices n.d.). Mikäli fontit lisää sivulle itse, täytyy olla varma että fonttien käyttäminen on sallittua kyseisessä tarkoituksessa. Etenkin itse lisättyjen fonttien toimivuus ja käyttäytyminen kannattaa tarkistaa useammassa laitteessa.

3.3.8 CSS-gridit

CSS-gridit ovat valmiita CSS-pohjia, joissa sisältö sijoitellaan palstoihin. Yleisimmät palstojen määrät ovat 12 ja 16. Kun ikkunaa pienennetään leveyssuunnassa, myös palstojen koko muuttuu. Gridien maksimileveys on usein määritelty 940 tai 960 pikseliin. Jotkut gridit, kuten Twitter Bootstrap (n.d.), kuitenkin ottavat huomioon myös leveämmät näytöt ja määrittelevät gridin maksimileveyden 1170 pikseliin. Näin leveässä gridissä on kuitenkin otettava huomioon, että leipäteksti ei voi mennä koko gridin levyiseksi, koska silloin sen luettavuus huononee.

CSS-gridit nopeuttavat sekä suunnittelijan että kehittäjän työtä. Suunnittelija suunnittelee layoutin niin sanotun gridipohjan päälle, johon palstat on merkitty. Kehittäjä toteuttaa HTML:n suunnitelman pohjalta määrittäen palstat HTML:ään luokiksi. Tällöin hänen ei tarvitse itse laskea palstojen leveyksiä ja testata, että ne toimivat varmasti kaikissa ikkunan leveyksissä oikein. Kuviossa 16 on esimerkki gridin päälle suunnitellusta layoutista. Tummat osat tarkoittavat palstoja ja välit niiden

välissä pastojen välistä marginaalia. Henkilön kuvasta voidaan laskea, että se vie kolme palstaa.



Kuvio 16. Esimerkki gridin päälle suunnitellusta layoutista

3.3.9 Navigaatio

Navigaatio on sivun tärkein osa, sillä se mahdollistaa käyttäjän liikkumisen sivulla. Sen responsiivinen käyttäytyminen tulee miettiä tarkkaan, sillä samanlainen navigaatio kuin desktop-koossa harvoin palvelee mobiilikäyttäjää. Yleinen tapa mobiilinavigaation näyttämiseen on niin sanottu toggle-navigaatio, jossa käyttäjä saa navigaation esiin klikkaamalla sen avaamiseen tarkoitettua painiketta.

Navigaation toimintaa voidaan parantaa, mikäli touch-tapahtumaa tukevissa laitteissa sen avaamiseen käytetään click-tapahtuman sijaan touch-tapahtumaan. Tällöin päästään eroon normaalin click-tapahtuman aiheuttamasta viiveestä, jolloin navigaatio avautuu nopeammin.

3.4 Vaihtoehtoisen sisällön näyttäminen

3.4.1 Yleistä

Toisinaan on tilanteita, jolloin mobiiliselaimessa halutaan näyttää eri html-rakenne tai sisältö kuin desktop-selaimessa, tai jopa piilottaa sisältöä. Elementtien

piilottamiseen yksinkertaisin vaihtoehto on määrittää elementti piilotetuksi CSS:n avulla, mutta se ei ole millään tavalla järkevä ratkaisu. Kun elementti määritetään CSS:n avulla piilotetuksi, se ladataan silti. Tällöin kulutetaan turhaan suorituskykyä asian vuoksi, jota ei edes näytetä. Yksi vaihtoehto on muokata sisältöä JavaScriptin avulla, mutta sekin on suorituskyvyllisesti huono ratkaisu, eikä toimi selaimissa jotka eivät tue JavaScriptiä.

Ajoittain halutaan myös tietää, mitä ominaisuuksia käyttäjän selain tukee. Esimerkiksi selaimessa joka ei tule jotain tiettyä ominaisuutta, kuten SVG:tä, halutaan tehdä kojaavia toimenpiteitä käyttäjäkokemuksen parantamiseksi. Tässä tapauksessa näyttää SVG:n tilalla normaali kuva.

3.4.2 Palvelinpuolen tarkastelu - User Agent Detection

Palvelinohjelmoinnilla on mahdollista testata, mitä selainta eli user agentia käyttäjä käyttää. User agent detection ei kuitenkaan ole aina luotettava, sillä selain saattaa palauttaa merkkijonon, joka ei vastaa sen todellisia ominaisuuksia. User agent detectionin avulla pystytään muun muassa selvittämään, onko laite mobiililaite vai ei. (Kadlec 2013.) Se on suorituskyvyllisesti hyvä ratkaisu, koska kaikki työ tehdään palvelinpäässä.

Samsung Galaxy Note 10.1 palauttaa seuraavan user agent merkkijonon:

```
Mozilla/5.0 (Linux; U; Android 4.1.2; fi-fi; GT-N8000
Buid/JZO54K) AppleWebKit/534.30 (KHTML, like Gecko) Version/4.0
Safari/534.30
```

Kyseinen tabletti ei erikseen ilmoita olevansa mobiililaite, mutta se kertoo käyttöjärjestelmäkseen Androidin. Tästä voidaan päätellä, että se lukeutuu mobiililaitteisiin.

Mobile Detect (2013) on yksi user agent detectionia hyödyntävä avoimen lähdekoodin PHP-luokka, jolla pystytään havaitsemaan mobiililaitteita. Se vertaa laitteen palauttamaa user agent -merkkijonoa sekä http-headereita sen tallentamiin tietoihin eri laitteista. Luokka tarjoaa vaihtoehtoja mobiililaitteiden, tablettilaitteiden, käyttöjärjestelmien sekä eri selainten havaitsemiseen. Tästä on hyötyä esimerkiksi silloin, kun halutaan muuttaa click-tapahtumat touch-

tapahtumiksi. Mobile Detect ei kuitenkaan ole täysin luotettava, sillä markkinoille tulee koko ajan uusia laitteita, joita se ei pysty tunnistamaan. Sitä kuitenkin ylläpidetään aktiivisesti, jotta sen luotettavuus pysyisi ajan tasalla.

3.4.3 Selainpuolen tarkastelu - Feature detection

Feature detection eli ominaisuuksien havaitseminen on yleensä selaimessa JavaScriptillä tapahtuvaa tarkastelua siitä, mitä ominaisuuksia selain tukee. Selain saattaa kuitenkin sanoa tukevansa jotain ominaisuutta, mutta ei todellisuudessa tue sitä kunnolla. Koska kyseessä on selaimessa tapahtuva tarkastelu, se ei ole yhtä tehokas kuin palvelinohjelmoinnilla tehty tarkastelu. (Kadlec 2013.)

Yksi suosituimmista apuvälineistä ominaisuuksien havaitsemiseen on JavaScript-skripti Modernizr (2013). Se testaa yli 40 eri ominaisuutta ja kertoo, tukeeko selain niitä. Lopputuloksena Morednizr lisää html-elementtiin luokkina tiedon siitä, mitä ominaisuuksia selain tukee ja mitä ei. Testausta kannattaa kuitenkin käyttää valikoiden ja testata vain niitä ominaisuuksia, jotka ovat verkkosivun kannalta tarpeellisia, sillä kaikkien ominaisuuksien testaaminen on raskasta. (Kadlec 2013.)

Modernizrista on olemassa myös palvelinpuolen vaihtoehto modernizr-server, jolla tuettavat ominaisuudet saadaan tietoon ennen kuin sivu generoidaan ja lähetetään selaimeen (Pearce 2010). Huonoa kyseisessä ratkaisussa on se, että saavuttaessa sivulle ensimmäisen kerran, modernizr-server tarkistaa tuettavat ominaisuudet, lisää ne evästeeseen ja lataa sivun uudelleen. Parhaimmillaan käyttäjä ei huomaa sivun uudelleenlatausta, mutta ei voida kuitenkaan poissulkea sitä mahdollisuutta, että uudelleenlataus hidastaa sivulle pääsyä huomattavasti. (Kadlec 2013.)

3.4.4 RESS

RESS (Responsive Web Design + Server Side Components) pyrkii yhdistämään palvelin- ja selainpuolen tekniikat, joilla kohdelaitteen ominaisuuksia voidaan havaita jo palvelimella. Palvelut kuten Detector ja WURFL pitävät kirjaa laitteiden palauttamista user agent -merkkijonoista ja kyseistä merkkijonoa vastaavan laitteiden ominaisuuksista. Kun käyttäjä menee sivulle, selaimelle palautetaan lista ominaisuuksista käyttäjän user agent -merkkijonon perusteella.

Detector

Detector on avoimen lähdekoodin PHP-kirjasto, joka hyödyntää Modernizria ominaisuuksien havaitsemisessa. Toisin kuin modernizr-server, Detector pitää listaa laitteista sekä niiden ominaisuuksista. Kun käyttäjä saapuu sivulle, Detector testaa löytyykö tämän user agent merkkijonon mukaista laitetta rekisteristä. Mikäli ei, se testaa tuettavat ominaisuudet ja lataa sivun uudelleen. Käyttäjä ei siis joudu näkemään sivun uudellenlatausta mikäli hän saapuu laitteella, joka löytyy jo rekisteristä. (Olsen, 2012)

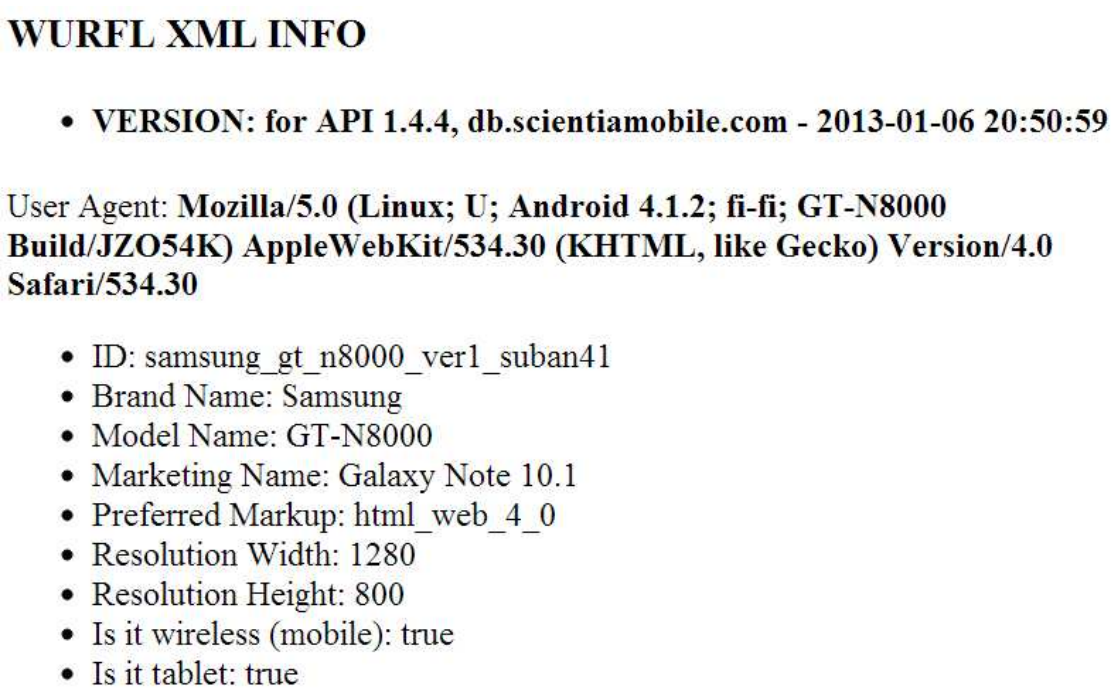
Detector palauttaa yhtenä ominaisuutena ikkunan leveyden ja korkeuden. Palautettavat arvot eivät kuitenkaan olleet testattaessa niin luotettavia, että niitä kannattaisi käyttää. Arvot olivat eri selaimilla aivan erilaisia, kuin laitteen todellinen leveys ja korkeus, jotka olivat 1280 x 800 pikseliä (kts. kuvio 17).

Detector Per Request Test Features	Your Browser	Detector Profile
screenattributes->colorDepth:	32	32
screenattributes->windowHeight:	488	638
screenattributes->windowWidth:	980	1281
Detector Per Request Test Features	**Your Browser**	**Detector Profile**
screenattributes->colorDepth:	32	32
screenattributes->windowHeight:	1370	488
screenattributes->windowWidth:	980	980
Detector Per Request Test Features	**Your Browser**	**Detector Profile**
screenattributes->colorDepth:	32	32
screenattributes->windowHeight:	509	1145
screenattributes->windowWidth:	980	800

Kuvio 17. Detectorin antamia arvoja (Olsen 2012).

WURFL

WURFL (the Wireless Universal Resource FiLe) on laaja varasto laitteista ja niiden ominaisuuksista. WURFL tarjoaa ohjelmointirajapintoja useille ohjelmointikielille, kuten PHP:lle. Se on lisenssoitu Affero GPL lisenssillä, joten sen käyttämisessä on tiettyjä rajoituksia. WURFL kykenee kertomaan laitteesta tarkkojakin tietoja, kuten laitteen merkin tai sen näytön fyysisen koon, mutta selaimen ominaisuuksissa se ei ole yhtä kykenevä kuin Modernizr. Se kykeni testattaessa palauttamaan kohdelaitteen oikean resoluution (kts. kuvio 18), mutta pöytäkoneella testattaessa resoluutioksi ilmoitetaan 800 x 600. (ScientiaMobile, Inc 2012.)



Kuvio 18. WURFL:in kertomia tietoja laitteesta Samsung Galaxy Note 10.1

WURFL tarvitsee toimiakseen lukuisia tiedostoja ja sen asentaminen on raskas prosessi. Tämä massiivisuus onkin syy siihen, miksi sitä tuskin kannattaa ottaa käyttöön pienillä sivustoilla, joissa ratkaisuksi kelpaisi kevyempikin vaihtoehto.

3.5 Responsiiviset kuvat

3.5.1 Yleistä kuvista

Responsiivisuuden suurimpia ongelmia on kuvien käyttäytyminen eri näyttökoissa. 1200 pikseliä leveää kuvaa ei ole järkeä ladata sellaisenaan laitteessa, jonka näytön leveys on 300 pikseliä. Lisäksi ongelmia aiheuttavat laitteet, joiden pikselisuhde on suurempi kuin yksi. Apple on lanseerannut näitä näyttöjä kuvaavan termin retina, jolla tarkoitetaan korkearesoluutioisia näyttöjä. (Digitoday, 2012) Näissä laitteissa kuvat näyttävät suttuisilta, mikäli niitä ei ole määritelty oikein.

Responsiivisten kuvien käyttämiselle ei ole mitään standardia, saatikka yhtä oikeaa tapaa. Ongelmaa varten on perustettu Reponsive Images Community Group, joka pyrkii kehittämään ongelmaan ratkaisua. (Responsive Images Community Group 2013.)

Yksinkertaisin tapa tehdä kuvista responsiivisia, näytön kokoon mukautuvia, on määrittää kuville seuraava tyyli CSS:llä:

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

Kyseinen tyyli pakottaa kuvan skaalautumaan pienemmäksi, mikäli elementti, jonka sisällä se sijaitsee, on kapeampi kuin kuvan fyysinen koko.

3.5.2 Korkearesoluutioiset näytöt

Puhuttaessa korkearesoluutioisista näytöistä, käytetään usein Applen termiä retina, vaikka muutkin laitteet jakavat tämän ongelman. Kyseisissä laitteissa pikselisuhde ei ole normaali: esimerkiksi Applen uusimmissa laitteissa yksi laitteen pikseli vastaa kahta normaalia pikseliä. Tämä ei aiheuta ongelmia silloin, kun käsitellään vektorigrafiikkaa kuten fontteja, mutta bittigrafiikka voi näkyä huonolaatuisena.

Kun laitteen pikseli vastaa kahta normaalia pikseliä, kuvat skaalataan kaksinkertaisiksi. Kun kuvan fyysinen leveys on 400 piskeliä, sen teoreettinen leveys laitteessa on 800 pikseliä. Tämä luonnollisesti aiheuttaa sen, että kuva näkyy sumeana. Siksi korkearesoluutioisille näytöille pitäisi tarjota kuva, jonka koko on kerrottu pikselisuhteella: Applen laitteissa siis kaksinkertainen. Kuvan dpi-arvolla ei ole väliä, ainoa mikä merkitsee, on sen fyysinen koko pikseleinä. Tämä ei kuitenkaan riitä, sillä kuva on selaimessa pienennettävä vastaamaan sitä kokoa, jossa sen haluttaisiin normaaleissa pikseleissä näkyvän.

3.5.3 Compressive Images – yksi kuva tavallisille ja korkearesoluutioisille näytöille

Filagment Group (2012) esitteli yksinkertaisen ratkaisun siitä, kuinka ratkaistaan kuvien näyttäminen korkearesoluutioisille näytöille. Lähtökohtaisesti kaikki kuvat voitaisiin tarjota kaksinkertaisessa koossa, jolloin ne näyttäisivät korkearesoluutioisilla näytöillä hyviltä. Vastaan tulee kuitenkin ongelma kuvien tiedostokoossa, sillä ison kuvan tiedostokoko on tietenkin isompi. Compressive Imagesin ideana on se, että kuva tallennetaan yli kaksi kertaa näytettävää kuvaa suurempana, mutta niin huonolla pakkauslaadulla kuin mahdollista. Koska kuva

täytyy kuitenkin HTML/CSS:n avulla pienentää siihen kokoon, missä sen halutaan näkyvän, ei kuvan huonoa laatua huomaa. Parhaimmassa tapauksessa yli kaksi kertaa isompi huonolaatuinen kuva on tiedostokooltaan pienempi, kuin fyysisesti halutun kuvan kokoinen siedettävällä laadulla tallennettu.

3.5.4 SVG – vektorigrafiikka kuvana

SVG (Scalable Vector Graphics) on verktorikuva, joka määritetään XML:n avulla. Kuten muukin vektorigrafiikka, SVG:tä voi skaalata ilman, että laatu huononee tai tiedostokoko kasvaa. SVG-tiedostot ovat kuitenkin yleensä kuvia suurempia ja selain tarvitsee enemmän aikaa rasteroida ja näyttää ne (Friedman 2012). SVG:tä voisi käyttää juurikin vektorimaisiin kuviin, mutta tarkkaa valokuvaa on mahdoton muuntaa vektoriksi. Internet Explorer versiosta 8 alaspäin sekä Android 2.3 -selain ja selaimet siitä alaspäin eivät tue SVG:tä. Kyseisille selaimille on siis tarjottava vaihtoehtoinen tapa kuvan näyttämiseen. (Kadlec 2013.)

3.5.5 Taustakuvat

Mikäli verkkosivuilla on elementtejä, joihin kuva voidaan laittaa taustakuvana, voidaan eri selaimen leveyksille tarjota eri kuva media queryjen avulla. Eri kuvia määriteltäessä täytyy ottaa huomioon myös korkearesoluutioiset näytöt, joille voi määrittää eri kuvan media queryn device-pixel-ratio sekä resoluutio-ominaisuuden avulla. Tällöin täytyy muistaa, että taustakuva täytyy pienentää haluttuun kokoon background-size-ominaisuuden avulla. Eri taustakuva määritettäisiin seuraavalla media queryllä (Jankord 2012):

```
@media
(-webkit-min-device-pixel-ratio: 2), (min-resolution: 192dpi) {
     /* Tyyli */
}
```

Taustakuvia määritettäessä tulee ottaa huomioon, kuinka sen määrittävä media query muodostetaan. Tim Kadlec testasi, kuinka mobiiliselaimet käyttäytyvät, kun niille määritetään uusi taustakuva laitteen leveydestä riippuen. Testin tavoitteena oli

selvittää, milloin selain lähettää pyynnön kuvasta, jota todellisuudessa ei haluta ladata. (Kadlec 2012) Testin tulokset on tiivistetty taulukossa 3.

Taulukko 3. Kuvan latautuminen, kun sitä yritetään poistaa CSS:llä (Kadlec 2012)

Toimenpide	**Lopputulos (tulos toteutuu / testilaitteiden määrä)**
Elementti jonka sisällä on img-elementti, piilotetaan CSS:n avulla	Vanha kuva latautuu (12/14)
Elementti johon on määritelty taustakuva, piilotetaan CSS:n avulla	Vanha kuva latautuu (12/15)
Elementin sisällä olevaan elementtiin määritetään taustakuva, ylempi elementti piilotetaan CSS:n avulla	Vanha kuva ei lataudu (12/13)
Elementille määritetään uusi taustakuva	Vanha kuva ei lataudu (11/16)
Kaikissa taustakuvan määrittämisissä on käytetty media querya	Vanha kuva ei lataudu (12/13)

Kaikkein tehokkaiten vanhan kuvan latautuminen voidaan siis välttää sillä, että elementille annetaan määreitä vain media queryjen avulla.

3.5.6 Ikonit – vaihtoehtoja yksittäiselle kuvalle

Ikonit ovat pieniä kuvakkeita, jotka eivät yleensä ole valokuvamaisia. Pieniä ikoneja ei kannata laittaa yksittäisinä kuvina, sillä niiden näyttämiseen on muitakin tekniikoita: ikonifontit ja spritet. (Smashing Magazine 2012.)

Ikonifonttien ideana on se, että tietty merkki vastaa tiettyä ikonia. Merkille pitää vain antaa CSS:llä oikea font-family, jolloin se muuttuu ikoniksi. Fonttien käytön hyvä puoli on se, että koska ne ovat vektorigrafiikkaa eivätkä bittigrafiikkaa, ne skaalautuvat helposti ja pysyvät hyvälaatuisina. Lisäksi niiden väriä pystyy muuttamaan pelkän CSS:n avulla, toisin kuin kuvien. Kuten muidenkin fonttien kohdalla, myös ikonifontit saattavat renderöityä huonosti eri selaimissa. Esimerkiksi Pictos (2012) ja Shifticons (2013) ovat palveluita, jotka tarjoavat ikonifontteja.

Spritejä käytettäessä yhteen kuvaan yhdistetään monta kuvaa kuvion 19 mukaisesti. Halutun ikonin ottaminen käyttöön ei toimi yhtä yksinkertaisesti kuin fonttia

käytettäessä. Ikoni täytyy ottaa käyttöön jossain elementissä, kuten i tai span, jolle annetaan ikonia kuvaava luokka. Luokalle annetaan taustakuvaan liittyviä CSS-määreitä, jolloin kokonaisesta kuvasta saadaan näytettyä vain haluttu osa.

Kuvio 19. Esimerkki sprite-kuvasta, jossa monta kuvaa on yhdistetty yhteen kuvaan

3.5.7 Picturefill – staattiset kuvat responsiivisiksi

Picturefill yhdistää media queryt ja JavaScriptin kuvien näyttämiseen eri näyttökoissa (Jehl 2013). Picturefillin käyttäminen on kätevää layouttiin liittyvien suurien kuvien näyttämisessä, joita ei haluta näyttää taustakuvina. Picturefillin perusmerkkaus on seuraava (Jehl, 2013):

```
<div data-picture data-alt="A giant stone face at The Bayon
temple in Angkor Thom, Cambodia">
    <div data-src="small.jpg"></div>
    <div data-src="medium.jpg" data-media="(min-width: 400px)"></div>
    <div data-src="large.jpg" data-media="(min-width: 800px)"></div>
    <div data-src="extralarge.jpg" data-media="(min-width:
    1000px)"></div>
    <!-- Fallback content for non-JS browsers.
    Same img src as the initial, unqualified source element. -->
    <noscript>
        <img src="external/imgs/small.jpg" alt="A giant stone
        face at The Bayon temple in Angkor Thom, Cambodia">
    </noscript>
</div>
```

Data-src-attribuuttiin määritetään kuvapolku kuvaan, jota halutaan käyttää ja data-media-attribuutissa kerrotaan media query, jolla kuva otetaan käyttöön. Noscriptin avulla voidaan tarjota kuva selaimille, joissa ei ole JavaScript-tukea. Selaimet, jotka eivät tule media queryitä, ottavat käyttöönsä kuvan, jossa ei ole data-media-attribuuttia.

Picturefill vastaa syntaksiltaan Responsive Images Community Groupin suunnittelemaa picture-elementtiä. Sen ehdottomana hyötynä on se, että eri näyttökoille voidaan tarjota juuri sellainen kuva, kun halutaan. Picturefill ei kuitenkaan sovellu dynaamisten kuvien näyttämiseen, koska se vaatii tietyn merkkauksen toimiakseen.

3.5.8 Adaptive images – dynaamiset kuvat responsiivisiksi

Adaptive Images on Matt Wilcoxin (n.d.) PHP:n avulla kehittämä ratkaisu responsiivisiin kuviin. Adaptive Imagesin ideana on korvata img-elementin sisältö kuvalla, joka riippuu laitteen leveydestä. Kuten media queryihin, myös Adaptive Imagesiin määritetään breakpointteja, joiden perusteella tarjotaan oikea kuva.

Adaptive Imagesissa on useita hyviä puolia: se pienentää kuvat automaattisesti oikeaan kokoon silloin, kun niitä tarvitaan. Kuvat tallennetaan palvelimelle väliaikaiseen kansioon, josta ne poistuvat oletusarvoisesti seitsemän päivän päästä, mikäli niitä ei käytetä. Tällöin sivutila käyttö ei kasva holtittomaksi, kun jokaisesta kuvasta ei tarvitse tallentaa useita eri versioita eri näyttökokoja varten, vaikka niitä ei ikinä käytettäisi. Lisäksi se toimii mainiosti dynaamisten kuvien pienennyksessä, sillä se ei tarvitse poikkeavaa merkkausta. Kuvan rajaukseen tai sommitteluun eri näyttökoissa ei kuitenkaan pysty vaikuttamaan.

3.6 Suorituskyky

3.6.1 Yleistä suorituskyvystä

Compuwaren (2011) vuonna 2011 teettämän tutkimuksen mukaan 71 % mobiilikäyttäjistä oletti, että verkkosivu latautuu yhtä nopeasti mobiililaitteessa kuin tietokoneella. Saman tutkimuksen mukaan 74 % odotti vain viisi sekuntia sivun latautumista.

Keskimääräinen sivun koko on 1 mb, joka on paljon (Breaking Development 2012). Parasta olisi siis pyrkiä siihen, että sivun koko on enintään 1 mb ja latautumisaika maksimissaan viisi sekuntia. Näitä tavoitteita on kuitenkin toisinaan mahdotonta saavuttaa. Yksi työkalu suorituskyvyn mittaamiseen mobiililaitteissa on Mobitest,

joka kertoo hyödyllistä tietoa sivun latautumisesta ja siihen vaikuttavista tekijöistä (Mobitest 2012). Googlen PageSpeed Insights puolestaan käy sivun läpi ja katsoo, mitä suorituskyvyn kannalta hyödyllisiä asioita sivuilla on otettu huomioon ja tekee ehdotuksia, kuinka suorituskykyä voisi parantaa (Google Developers n.d.). Tarkistettavat asiat liittyvät lähinnä kuvien optimointiin, tiedostojen pakkaukseen sekä tyylien ja koodin järjestykseen.

Suorituskykyyn vaikuttavat ladattavien tiedostojen määrä sekä niiden koko. Mitä enemmän ladattavia tiedostoja on, sitä useamman palvelinpyynnön verkkosivu joutuu tekemään. Mitä enemmän sivulla on palvelinpyyntöjä, sitä hitaammin sivu latautuu. Mobiiliverkossa palvelinpyynnön tekeminen voi kestää 4-5 kertaa kauemmin, kuin kiinteässä verkossa. (Kadlec 2013)

3.6.2 Kuvat

Kuvat ovat verkkosivun raskain osa. Niiden fyysinen koko on suuri, ja vaikka kuva olisi kuinka pieni tahansa, aiheuttaa se silti palvelinpyynnön. Kuvien osalta onkin tärkeää, että ne optimoidaan oikein. Valittu tallennusformaatti vaikuttaa paljon. Mikäli kyseessä on valokuva, on parasta käyttää formaattia JPG. Muille kuville paras vaihtoehto on PNG. Vaikka kuvan tallentaisikin PhotoShopissa kaikkien ohjekirjojen mukaisesti optimaalisesti verkkosivua varten, voi se silti sisältää ylimääräistä dataa. Siksi kuvat olisi hyvä käyttää jonkin optimointi-ohjelman kautta, joka poistaa niistä ylimääräisen datan. (Smashing Magazine 2012.)

Optimointiin tarkoitettuja ohjelmia on useita. Kraken Image Optimizeria (Kraken.io 2012) kokeiltiin projektiin, jonka ulkoasuun tarvittavia kuvia oli yhteensä 20. Kuvien koko ennen Krakeniin viemistä oli yhteensä 380 kb, häviöttömän optimoinnin jälkeen 337,27 kb. Optimointi pienensi kuvien yhteiskokoa 11,25 prosentilla ilman, että kuvien laatu heikkeni.

Kuvien aiheuttamien palvelinpyyntöjen määrää voidaan vähentää määrittämällä kuva data URI:n avulla. Tämä tarkoittaa sitä, että img-elementin src-attribuuttiin tai CSS:n taustakuvan osoitteeseen ei laiteta kuvan osoitetta, vaan kuva liitetään ns. inline-datana seuraavasti (Coyier 2010):

```
<img width="16" height="16" alt="star" src="data:image/gif;base64
,R0lGODlhEAAQAMQAAORHHOVSKudfOulrSOp3WOyDZu6QdvCchPGolfO0o/XBs/fN
wfjZ0frl3/zy7///wAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAACH5BAkAABAALAAAAAAQABAAAAVVICSOZGlCQAosJ6mu7fiyZe
KqNKToQGDsM8hBADgUXoGAiqhSvp5QAnQKGIgUhwFUYLCVDFCrKUE1lBavAViFIDl
TImbKC5Gm2hB0SlBCBMQiB0UjIQA7">
```

Src-attribuutissa base 64:n jälkeen on näytettävä kuva koodattuna eli muutettuna binääridataksi. Mitä suurempi kuva on kyseessä, sitä pidemmän merkkijonon se aiheuttaa. Data URI:n tuki selaimissa on hyvä, mutta Internet Explorerissa tuki alkaa vasta versiosta 8, eikä sekään tule kuin alle 32 kb:n kokoisia data URI:a (Can I use... 2013).

On myös syytä kyseenalaistaa, onko kuvien käyttäminen aina tarpeellista ja kuinka niitä tulisi käyttää. Hyvä käytäntö on tehdä CSS3:lla kaikki sellainen, minkä sillä voi tehdä, jotta ylimääräisiltä kuvilta vältyttäisiin. Aina kannattaa miettiä, voiko kuvan laittaa jollain muulla tavalla kuin perinteisenä kuvana: ikonien kohdalla spritet ja ikonifontit ovat hyvä vaihtoehto.

3.6.3 CSS ja JavaScript

CSS:n ja JavaScriptin kannalta oleellista on, että tiedostokoot pyritään pitämään mahdollisimman pieninä. Käytännössä tämä tarkoittaa sitä, että tiedostosta poistetaan ylimääräiset rivinvaihdot ja välit. Tämä toimenpide kuitenkin hankaloittaa tiedoston ylläpitoa ja muokkausta.

Selain lataa ja jäsentää yhden JavaScriptin kerrallaan, joten luonnollisesti useamman eri tiedoston lataaminen on hitaampaa (Stocks, ym. 2012). Tästä johtuen alle 4 kb:n kokoisia tyylejä tai skriptejä ei kannata ladata, vaan ne on järkevämpi sisällyttää HTML:ään (Smashing Magazine 2012). Tämä voi tietenkin aiheuttaa ylläpidollisia ongelmia, mikäli sama tyyli tai skripti esiintyy useassa eri paikassa.

CSS:n suhteen on oleellista muistaa, että vaikka media queryillä määritellään eri tyylejä eri ehdoilla, selain joka tapauksessa lataa kyseiset tyylit. Tällöin se on valmiudessa näyttämään kyseiset tyylit, mikäli niille tulee tarvetta. CSS-tyylien osalta on hyvä miettiä, ovatko media queryillä määritellyt tyylit samassa tiedostossa, vai jokainen omassa tiedostossaan. Tyylitiedostoa liittäessä sille voidaan määrittää

media-attribuutti, joka kertoo media queryn jolla tiedosto otetaan käyttöön. Tällöin kaikki eri media queryn tyylit voidaan määrittää omaan tiedostoonsa ja kertoa jo tiedostoa liitettäessä, milloin se tulisi ottaa käyttöön. Tästä ei kuitenkaan ole suorituskyvyn kannalta muuta kuin haittaa, sillä sivua ladattaessa tyylit ladataan kuitenkin, vaikka sen hetkiset olosuhteet eivät vastaisikaan media querya. Tietenkin eri tiedostossa olevia tyylejä on helpompi hallita, mutta ne aiheuttavat silti uuden palvelinpyynnön. (Kadlec 2013.)

Animaatioiden toteuttaminen CSS:n avulla vie vähemmän suorituskykyä kuin JavaScriptillä toteutetut animaatiot (Avola & Raasch 2013). Animaatioiden käyttämistä tulisi kuitenkin harkita tarkkaan, sillä huonomman suorituskyvyn laitteet eivät toista niitä sulavasti, eikä tämä paranna käyttäjäkokemusta.

JavaScript-kirjastoa valitessa tulisi olla kriittinen, sillä suuri kirjasto hidastaa sivun latautumista ja pahimmassa tapauksessa pysäyttää muun sisällön latautumisen. Tämä voidaan ehkäistä liittämällä kirjasto headiin viimeisenä, ennen body-elementin alkua. Esimerkiksi jQuery, joka on yksi suosituimmista JavaScript-kirjastoista, on tiedostokooltaan suhteellisen suuri. Se sisältää koodia, joka lisää tukea Internet Explorer 6 -selaimelle, eikä kyseinen tuki ole nykyään kovin hyödyllinen. (Avola & Raasch 2013.) WordPressissä useat lisäosat kuitenkin käyttävät jQuerya, joten sen sisällyttäminen sivuun on usein välttämättömyys.

4 WordPress-frameworkin toteutus

4.1 Yleistä WordPressistä

MEOM käyttää WordPressiä julkaisujärjestelmänään, koska se on dokumentoitu hyvin, sen avulla on nopeaa kehittää verkkosivuja ja lisäksi se on asiakkaalle helppokäyttöinen. Vaikka WordPress ei tehokkuudeltaan pärjää esimerkiksi Drupalille, on se sopiva ratkaisu pieniin ja keskisuuriin verkkosivuihin.

WordPressissä kehittäjällä on kaksi ominaisuutta, joiden avulla rakentaa ja muokata sivua: teemat ja lisäosat. Molempia on kehitetty WordPressiä varten useita, eikä etenkään lisäosien kanssa kannata lähteä keksimään pyörää uudelleen. Teemojen

kanssa asia on hieman toinen; useimmiten on tehokkaampaa rakentaa teema itse, kun lähteä muokkaamaan valmisteemaa tarkoitukseen sopivaksi. Kaikkein yksinkertaisinta on lähteä kehittämään teemaa pohjasta, jossa on kaikki perusominaisuudet valmiina.

4.2 Teeman toteuttaminen

4.2.1 Lähtökohta

Teeman toteuttamiseen oli tiettyjä vaatimuksia. Sen tuli vastata WordPressin standardeja teeman kehittämisestä, HTML-rakenteen tuli olla mahdollisimman yksinkertainen ja HTML5:n semantiikkaa hyödyntävä. CSS:n rakentamisessa tuli käyttää tyylikieltä Stylus, jonka avulla CSS:n kirjoittaminen on tehokkaampaa.

Aikaisempi kokemus oli osoittanut, että on tehokkaampaa lisätä ominaisuuksia ja HTML-rakennetta teematiedostoihin, kuin poistaa niitä. Tästä syystä teematiedostoista tehtiin mahdollisimman yksinkertaisia, jotta aikaa turhien ominaisuuksien ja elementtien poistamiseen ei kuluisi.

Teemaan sisällytettiin tuki sekä staattisille että dynaamisille mukautuville kuville, mobiilinavigaatiolle sekä vaihtoehtoisen sisällön näyttämiselle. Tyylitiedostoon tehtiin Styluksen avulla ominaisuus, joka mahdollisi CSS-gridin laskemisen annetuilla muuttujilla, jolloin minkä tahansa palstoituksen toteuttaminen on mahdollista ilman manuaalisesti tehtävää leveyksien laskemista.

Ennen toteutusta tutustuttiin WordPressin vaatimuksiin ja ohjeistuksiin teemaa kohtaan. WordPress (n.d.) suosittelee, että teeman kehittämisessä käytettäisiin hyvin rakenneltua ja virheetöntä PHP:tä, validia HTML:ää ja CSS:ää. Näillä pyritään siihen, että lähdekoodi olisi selkeää ja helppolukuista. Etenkin PHP:ta koskevat suositukset kirjattiin teeman käyttöä ohjeistavaan dokumenttiin. WordPress Theme Review Team puolestaan on koonnut tiettyjä käytäntöjä, joita teeman olisi hyvä noudattaa (WordPress 2013). Näiden käytäntöjen rikkominen ei vaikuta millään tavalla teeman toimivuuteen, mutta niitä olisi silti hyvä noudattaa.

Teeman valmistuttua siitä kirjoitettiin yritykselle ohjeistus, joka esitteli teeman toimintoja ja sen periaatteita HTML- ja CSS-rakenteen suhteen. Teeman nimeksi annettiin Kalakala ja sille tehtiin yksisivuinen verkkosivu, joka esittelee teeman toimintaa.

4.2.2 Teematiedostot

WordPressissä on tietty rakenne, jota se käyttää teeman tiedostoja eri sisältötyyppien näyttämiseen. Mikäli sisältötyyppiä vastaava teematiedosto on olemassa, näytetään sisältö sen mukaan. Muutoin sisältö näytetään sisältötyypin ylemmän teematiedoston mukaan ja mikäli sitäkään ei löydy, käytetään sisällön näyttämiseen tiedostoa index.php.

Kalakalan teematiedostojen rakenteen ja kuvauksen niiden sisällöstä voi lukea liitteestä 2.

4.2.3 Vaihtoehtoisen sisällön näyttäminen

Ennen kuin teemaan toteutettiin mahdollisuus vaihtoehtoisen sisällön näyttämiselle, tutkittiin kuinka palvelinpuolen tarkastelu selaimen ominaisuuksien suhteen toimi WordPressissä. Aikaisemmin käsitellyistä WURFL:ista ja Detectorista löytyi lisäosat, joiden avulla niiden käyttäminen WordPressissä pitäisi olla yksinkertaista. WURFL:in lisäosan asentaminen oli niin raskas prosessi, ettei se onnistunut ensimmäisellä yrityksellä. Lisäksi sen massiivisuuden vuoksi sivuston siirtäminen palvelimelta toiselle osoittautui hitaaksi. Detectorin lisäosa puolestaan ei toiminut ollenkaan. Sitä yritettiin myös asentaa manuaalisesti, mutta sivusto palautti edelleen samoja virheilmoituksia.

Koska selaimen ominaisuuksien tarkastelu palvelimella ei onnistunut, täytyi ominaisuuksien tarkastelu rajoittaa selaimeen Modernizrilla. Sen avulla oli tarkoitus tarkastella tukeeko selain SVG:tä. Tämän ominaisuuden avulla pystyttiin CSS:ssä näyttämään eri kuvalähde riippuen siitä, mitä lähdettä selain tukee.

Palvelimella puolestaan tarkasteltiin vain laitteen tyyppi, eli onko se mobiililaite vai ei. Tässä tapauksessa mobiililaitteiksi laskettiin vain puhelimet, sillä eri sisältöä

haluttiin näyttää eri lailla vain pienemmissä näytöissä. Palvelinpuolen tarkastusta varten valittiin Mobile Detectistä tehty WordPress-lisäosa sen keveyden ja yksinkertaisuuden takia. Kyseinen lisäosa mahdollistaa myös lyhytkoodien käyttämisen WordPressin sisältökentässä, jolloin myös dynaamisessa sisällössä on mahdollista näyttää eri sisältöä eri laitetyypille.

Vaihtoehtoisen sisällön näyttämiseen käytettiin Mobile Detectin ja Ajax-Include JavaScriptin yhdistelmää, jonka avulla haluttu sisältö lisätään sivulle vasta silloin, kun käyttäjä klikkaa sisällön näyttämiseen tarkoitettua painiketta. Tällöin Ajax-Include korvaa painikkeen AJAX:in avulla näytettävällä sisällöllä (Jehl, An Ajax-Include Pattern for Modular Content 2012). Koska JavaScriptin tarkastelua ei voitu suorittaa palvelimella, ei kyseinen ominaisuus toimi käyttäjillä, jotka eivät ole sallineet JavaScriptiä. Heille annetaan noscript-elementin kuitenkin ilmoitus, jossa JavaScriptin sallimista suositellaan koko sisällön näkemiseksi.

Kalakalan esimerkkisivustossa sosiaalisen median jakopainikkeet (Twitter, Facebook, Pinterest, Google+), sisältöosioiden pääsisältö sekä sivun kommentit jätetään puhelimilla lataamatta. Sen sijaan käyttäjälle näytetään kuvion 20 mukainen näkymä, jossa hän näkee kuvauksen sisällöstä sekä painikkeen, jota painamalla hän saa koko sisällön näkyviin.

Kuvio 20. Vaihtoehtoisen sisällön näyttäminen

4.2.4 Mukautuvat kuvat

Adaptive Images

WordPressissä sisältöön lisätyt kuvat pienennetään automaattisesti kohdelaitteen leveyteen Adaptive Imagesin avulla. Koska kyseisestä ominaisuudesta ei ole toimivaa WordPress-lisäosaa, täytyy sitä varten asettaa muutamia asioita käsin. Sivuston juuressa olevaan .htaccess tiedostoon lisätään seuraava koodi:

```
<IfModule mod_rewrite.c>
    Options +FollowSymlinks
    RewriteEngine On
    # Adaptive-Images ----------------------------------------
    RewriteCond %{REQUEST_URI} !wp-content/themes/kalakala_land/images
    RewriteCond %{REQUEST_URI} !wp-content/uploads/ai-cache
    RewriteRule \.(?:jpe?g|gif|png)$ wp-content/themes/kalakala/includes/adaptive
        images.php
    # END Adaptive-Images ------------------------------------
</IfModule>
```

Koodissa estetään Adaptive Imagesin pääsy kansioihin, joissa olevia kuvia ei haluta pienentää. Näitä kansioita ovat teeman images-kansio, sekä ai-cache jonne Adaptive Images tallentaa pienennetyt kuvat. Molempien kansioiden polut tulee uuden WordPress-asennuksen ja -siirron yhteydessä tarkistaa ja muuttaa oikeiksi. Koodin lopussa määritellään kansio, jossa Adaptive Images -luokka sijaitsee, myös se täytyy muuttaa uuden asennuksen ja sivuston siirron yhteydessä.

Itse Adaptive Images -luokassa on sen toimintaan vaikuttavia asetuksia:

```
// the resolution break-points to use (screen widths, in pixels)
$resolutions   = array(1382, 992, 768, 480);
// where to store the generated re-sized images. Specify from your
   document root!
$cache_path    = "kalakala/wp-content/uploads/ai-cache";
// the quality of any generated JPGs on a scale of 0 to 100
$jpg_quality   = 75;
// Shrinking images can blur details, perform a sharpen on re-scaled
   images?
$sharpen       = TRUE;
// check that the adapted image isn't stale (ensures updated source
   images are
recached)
$watch_cache   = TRUE;
// How long the BROWSER cache should last (seconds, minutes, hours, days.
   7days by default)
$browser_cache = 60*60*24*7;
```

Olennaisin kohta on `$cache_path`, joka tulee muuttaa oikeaksi, mikäli WordPress-asennus ei sijaitse domainin juuressa. Kyseiseen polkuun tallennetaan pienennetyt kuvat. Muut muuttujat vaikuttavat siihen, missä koossa ja millä laadulla kuvat näytetään, ja kuinka kauan selaimen cache kestää.

Picturefill

Picturefill on sisällytetty teemaan automaattisesti teemassa olevien staattisten, ensisijaisesti headerissa olevien koko näytön levyisten kuvien näyttämistä varten. Sitä kutsutaan teeman funktiolla, joka generoi Picturefilliä varten sen tarvitseman HTML-rakenteen. Oletusarvoisesti kuvat haetaan teeman images-kansiosta jpg-

muodossa ja ne tulee olla nimettyinä ja muutettuina oikeisiin kokoihin taulukon 4 mukaisesti.

Taulukko 4. Picturefillissä käytettävien kuvien nimeäminen ja koot

Kuvan nimi	**Näyttö jossa käytetään**	**Leveys**
nimi-small.jpg	Normaali resoluutio, leveys enintään 480 px	480 px
nimi-small@2.jpg	Korkea resoluutio, leveys enintään 480 px	960 px
nimi-medium.jpg	Normaali resoluutio, leveys enintään 768 px	768 px
nimi-medium@2.jpg	Korkea resoluutio, leveys enintään 798 px	1536 px
nimi-large.jpg	Normaali resoluutio, leveys enintään 992 px	992 px
nimi-large@2.jpg	Korkea resoluutio, leveys enintään 992 px	1984 px
nimi-xlarge.jpg	Normaali resoluutio, leveys vähintään 992 px	1980 px tai vastaava haluttu maksimileveys
nimi-xlarge@2.jpg	Korkea resoluutiom leveys vähintään 992 px	3960 px tai vastaava haluttu maksimileveys

Yksinkertainen grafiikka

Ikonit tehdään ensisijaisesti ikonifonttien avulla tai spriteinä, joista tehdään sekä SVG että PNG-vaihtoehdot. Kun ikonit määritellään taustakuviksi CSS:llä, Modernizrin tarjoaman SVG-tarkastelun avulla CSS-tiedostoon pystytään määrittämään seuraavalla tavalla eri kuvalähde riippuen siitä, tukeeko selain SVG:tä:

```
.icon { background-image: url(images/icon.png); }
.svg .icon { background-image: url(svg/icon.svg); }
```

Yksinkertaiset kuvat voidaan myös määrittää käyttäen data URI:a. Kaikkia kuvia ei sen avulla kannata määrittää, koska monimutkaisten kuvien näyttäminen data URI:n avulla voi olla hitaampaa, kuin normaalin tiedoston lataaminen. Data URI:a käytettäessä on kuitenkin tarjottava Internet Explorer 8 ja 7 -selaimille kuvasta

normaali kuvapolku niille erikseen tarjottavassa tyylitiedostossa, jotta kuvien toimiminen mahdollistetaan myös kyseisissä selaimissa.

4.2.5 Mobiilinavigaatio

Navigaation ei haluttu olevan riippuvainen kohdelaitteesta, vaan selainikkunan leveydestä. Sen toteutukseen valittiin JavaScript lisäosa Responsive Nav (Salminen n.d). Se sisältää tuen touch-tapahtumille ja avaa navigaation sulavasti CSS3-animaation avulla. Lisäksi selaimissa, jotka eivät tue JavaScriptiä, navigaatio on automaattisesti auki. Navigaatio sisällytettiin teemaan niin, että teeman päänavigaatio muuttuu mobiilinavigaatioksi selaimen ollessa 640 pikseliä leveä. Kyseistä muutospistettä voi helposti vaihtaa, mikäli koetaan että navigaation on tarve muuttua mobiilinavigaatioiksi aikaisemmin tai myöhemmin. Navigaation toiminta on havainnoillistettu kuviossa 21.

Kuvio 21. Teeman normaali navigaatio, mobiilinavigaatio kiinni sekä mobiilinavigaatio auki

4.2.6 CSS-grid

Teema varten ei valittu valmista CSS-gridiä niiden joustamattomuuden vuoksi, vaan gridi päätettiin toteutaa Styluksen avulla automaattisesti muodostuvaksi annettujen

arvojen perusteella. Tällöin grid ei rajoita suunnittelu- ja kehitystyötä, koska sitä voidaan muuttaa jopa kesken kehityksen.

Styluksessa oleva laskemiseen tarkoitettu funktio käyttää seuraavia muuttujia:

```
cols = 12
gutter = 60
widt = 960
```

`Cols` tarkoittaa sarakkeiden määrää, `gutter` kahden palstan väliä ja `widt` sivun leveyttä. Muuttujien avulla lasketaan ensin yhden palstan leveys:

$$onecol = \frac{\big(widt - gutter * (cols - 1)\big)}{cols}$$

Yksi palsta on sivun leveys, josta vähennetään marginaalien viemä tila, jaettuna palstojen kokonaismäärällä. Koska ensimmäisessä palstassa ei koskaan ole marginaalia, vähennetään palstojen määrästä yksi.

Tämän jälkeen voidaan laskea halutun palstan leveys prosentteina arvolla n, joka tarkoittaa haluttujen palstojen määrää:

$$\frac{(onecol * n + (n - 1) * gutter)}{widt} * 100\%$$

Haluttu palsta on yhden palstan leveys kerrottuna palstojen määrällä. Tähän lukuun lisätään marginaalien viemä tila. Marginaalien määrä on palstojen määrä kerrottuna yhdellä marginaalilla. Palstojen määrästä vähennetään 1, joka tarkoittaa palstan omaa marginaalia. Lopullinen luku jaetaan koko leveydellä ja kerrotaan sadalla, jotta leveys saadaan prosentteina.

Yhden marginaalin suuruus prosentteina on marginaalin leveys jaettuna koko leveydellä, joka kerrotaan sadalla prosenttiluvun saamiseksi:

$$onemarg = \frac{gutter}{widt} * 100\%$$

Lisäksi Internet Explorerin vanhempia versioita varten on laskettava ns. korjausarvo, koska kyseiset selaimet eivät pyöristä lukuja kuten muut selaimet. Kyseiset selaimet pyöristävät kaikki luvut ylöspäin, jolloin normaalisti laskettu tulos voisi olla liian leveä, eivätkä palstat mahtuisi olemaan vierekkäin. Tyler Taten (2012) artikkelin

perusteella ongelma voidaan ratkaista, kun IE:lle tarjottavasta palstan leveydestä miinustetaan seuraavalla kaavalla laskettava korjausluku:

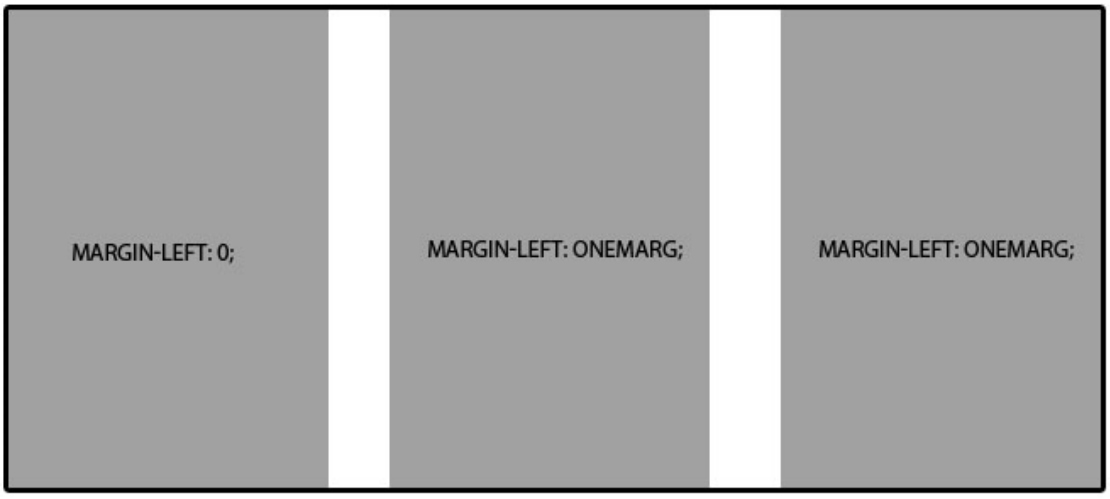
$$correction = \frac{0.5}{widt} * 100 * 1\%$$

Joissain tapauksissa elementille halutaan antaa tyhjää tilaa, joka vastaa palstojen määrää (kts. kuvio 22). Tätä kutsutaan offsetiksi. Offsetin laskeminen poikkeaa palstojen laskemisesta vain siinä, että siihen täytyy yksi marginaali ennen luvun muutamista prosentiksi seuraavasti:

$$\frac{(onecol * n + (n-1) * gutter) + gutter}{width} * 100\%$$

Kuvio 22. Rivin ensimmäistä elementtiä on siirretty sivun reunasta offsetin avulla

Näiden laskutoimitusten jälkeen haluttua gridiä voidaan alkaa muodostamaan. Gridi toimii kuvion 23 mukaisella periaatteella, jossa joukkion ensimmäisen elementin margin-left-arvo on 0, muiden elementtien margin-left-arvo on `onemarg`. Tällöin kaikkien elementtien leveyksien ja marginien leveys riittävät täyttämään koko niitä ympäröivän alueen.



Kuvio 23. Gridin toimintaperiaate marginien suhteen

Ensimmäisen elementin leveys voidaan nollata joko antamalla sille luokka first, tai paketoimalla elementit divin sisään, jonka ensimmäisen lapsen margin-left menee automaattisesti nollaan :first-child-selektorin avulla. Kyseinen selektori toimii myös Internet Explorerin aiemmissa versioissa, joten sen käyttö on turvallista.

Jotta gridi toimisi oikein, tulee sen avainelementit määritellä oikein. Gridiä käytetään container-luokan sisällä, jossa määritellään koko sivun leveys. Containerilla on seuraavat määreet:

```
.container
        max-width 960px
        margin 0 auto
        *zoom 1
        &:before, &:after
                  display table
                  content ""
                  line-height 0
        &:after
                  clear both
```

Palstat on hyvä laittaa divin sisään, jolloin niiden jälkeen tulevat elementit asettuvat oikein. Teemassa tätä tarkoitusta vastaavalla divillä on luokka tank-row, jolla on seuraavat ominaisuudet:

```
.tank-row
        width 100%
        &:before, &:after
                display table
                line-height 0
                content ""
        &:after
                clear both
        [class*="kala"]:first-child
                margin-left 0
```

Yksittäistä palstaa kutsutaan kalaksi, joka tarvitsee toimiakseen seuraavat perusmääreet:

```
[class*="kala"]
        display block
        float left
        margin 0 0 0 (onemarg)
        *margin 0 0 0 (onemarg - correction)
```

Kyseinen selektori vaikuttaa kaikkiin luokkiin, joiden nimessä on kala. Siksi nimeksi on valittu erikoisempi nimi, koska esimerkiksi luokka `col` saattaisi sekoittaa joitain ulkopuolisia CSS-tyylejä. Marginaalin määrittämisessä käytetään `onemarg`-muuttujaa, jolla palstan vasempaan reunaan määritetään marginaali. Viimeisenä oleva margin-arvo jonka edessä on tähti, on tarkoitettu vanhemmille Internet Explorerin versioille.

Haluttuja palstoja voidaan lisätä määrittämällä luokan nimeksi kala`n`, jossa `n` tarkoittaa pastojen lukumäärää jota luokka edustaa. Luokalle oikeat leveydet seuraavalla tavalla:

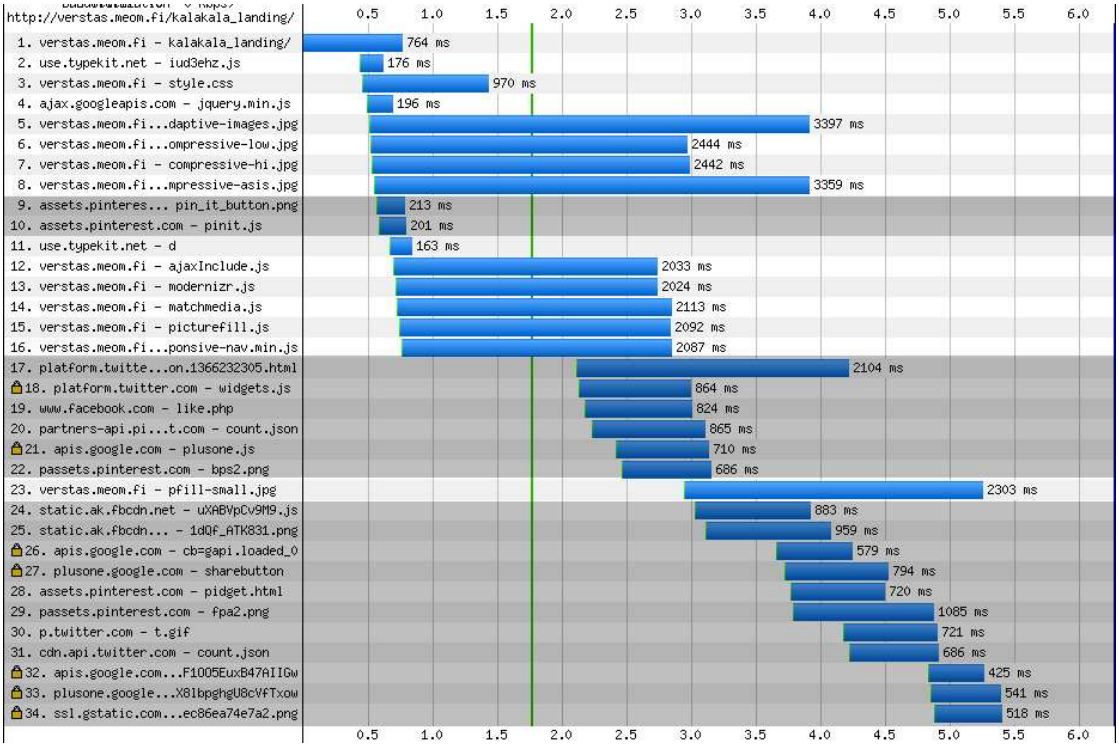
```
.kala7
        width kalacol(7)
        *width (kalacol(7) - correction)
```

Leveyden määrittämiseen kutsutaan kalacol-funktiota, jolle annetaan parametriksi haluttujen palstojen määrä.

4.3 Suorituskyvyn testaus

Kalakalan suorituskykyä mobiililaitteissa testattiin Mobitestin avulla. Palvelimella jossa Kalakala sijaitsi, oli gzip käytössä. Kaikki skriptit jQueryn kirjastoa lukuun ottamatta sijoitettiin sivun loppuun. Suorituskykyä testattaessa ajettiin aina kolme testiä, joista ilmoitettiin keskimääräinen tulos. Tarkasteltavat kaaviot on otettu testistä, joka oli lähimpänä keskiarvoa.

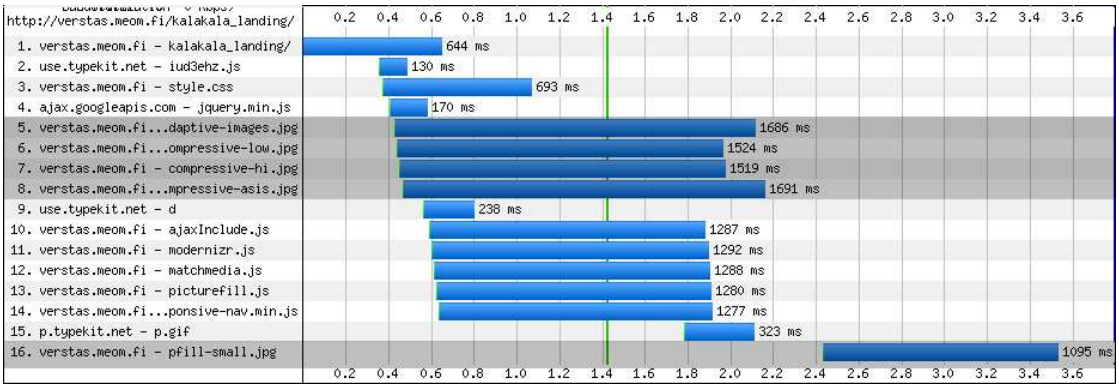
Ensimmäisessä testausvaiheessa layoutin kuvat näytettiin yksittäisinä taustakuvina Data URI:n kautta. Layoutissa oli kolme kuvaa: iso logo sekä kolme pienempää symbolia, jotka ilmaisivat teeman ominaisuuksia. Sisällön vaihtoehtoista näyttämistä ei ollut vielä suoritettu, joten kaikki sisältö latautui sivulle. Lisäksi tyyli- ja JavaScript-tiedostoja ei ollut minimoitu. Kolmen testin tuloksena sivun keskimääräinen latausnopeus oli 6,85 sekuntia ja keskimääräinen sivun koko 823,74 kb. Kun tarkasteltiin sivun elementtien latausta, huomattiin että sosiaalisen median painikkeet hidastivat sivua huomattavasti (kts. kuvio 24).



Kuvio 24. Sosiaalisen median painikkeiden aiheuttamat palvelinpyynnöt

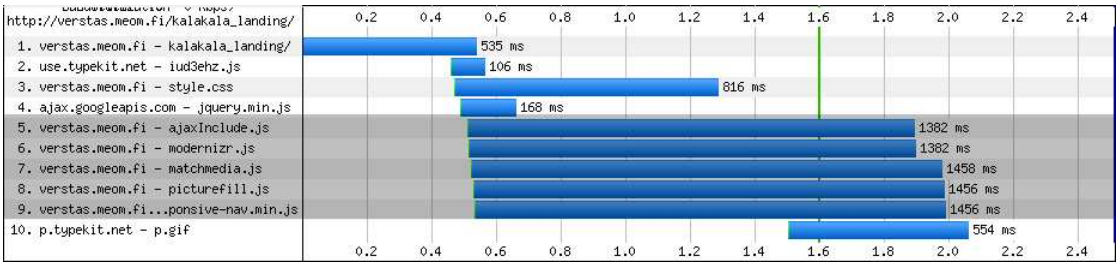
Suorituskykyä lähdettiin parantamaan kokeilemalla, kuinka paljon nopeammaksi sivu tulee, kun sosiaalisen median painikkeita ei oletuksena ladata, vaan ne tarjotaan

linkin takaa. Tämän jälkeen sivun keskimääräinen latausnopeus oli 4,31 sekuntia ja sivun keskimääräinen koko 522,28 kb. Kun alun perin palvelinpyyntöjä oli 35, väheni niiden määrä tämän toimenpiteen jälkeen 17:ään. Latautumista hidastivat sisällössä olevat kuvat (kts. kuvio 25), joten sivun osioiden pidempää sisältöä sekä kommentteja ei ladattu oletuksena, vaan ne tarjottiin linkin takaa.



Kuvio 25. Kuvien aiheuttamat palvelinpyynnöt

Kun kuvat jätettiin lataamatta, oli latausnopeus 2,73 sekuntia ja sivun koko 151,57 kb. Koska layoutissa käytettävät kuvat näytettiin data URI:n avulla, palvelinpyyntöjä aiheuttivat enää JavaScript-tiedostot (kts. kuvio 26), tyylitiedosto sekä fontit tarjoava Typekit. JavaScript-tiedostot jQuerya lukuun ottamatta pienennettiin ja yhdistettiin yhdeksi tiedostoksi. Tämä toimenpide ei antanut enää suurta hyötyä: sivu latautui 2,71 sekunnissa ja sen koko oli 133,22 kb.



Kuvio 26. JavaScript-tiedostojen aiheuttamat palvelinpyynnöt

Jäljellä oli enää tyylitiedosto, jonka koko oli suhteellisen suuri tyylitiedostoksi: 86 kb. Tämä johtui siitä, että kaikki kuvat oli laitettu data URI:na. Kuvat päätettiin määrittää CSS-tiedostoon normaaleina kuvapolkuina, jota nähdään kuinka paljon data URI:sta on hyötyä. Tässä täytyy huomata, että jokaisesta kuvasta tarjotaan kaksi eri

vaihtoehtoa: sekä SVG ja PNG. Tällöin selain lataa myös sen kuvan, jota ei kuitenkaan kyseisessä selaimessa käytetä.

CSS-tiedoston koko pieneni huomattavasti, se oli data URI:en poiston jälkeen vain 17 kb. Sivun suorituskyky meni palvelinpyyntöjen lisääntyessä odotetusti vain huonompaan suuntaan, sillä sivun latausaika oli 3,08 sekuntia ja koko 190,38 kb.

Tämän jälkeen kokeiltiin kuvien yhdistämistä yhdeksi kuvaksi, jotta palvelinpyyntöjen määrä vähenisi. Kaikista muista kuvista sivun päälogoa lukuun ottamatta tehtiin yksi sprite, sekä SVG-tiedostoista että png-tiedostoista. Koska kuvat olivat suhteellisen suuria, ei parannusta edelliseen juurikaan tullut: sivun latautumiseen meni 3,02 sekuntia ja sen koko oli 171,72 kb.

Loppujen lopuksi kuvien näyttäminen data URI:na oli kaikkein suorituskykyisin vaihtoehto, vaikka CSS-tiedoston koko kasvoikin sen johdosta dramaattisesti. Layoutissa olevat kuvat määritettiin uudelleen data URI:na ennen seuraavaan vaiheeseen siirtymistä.

Sivustolla käytettiin kolmea web-fonttia, jotka haetiin fonttipalvelu Typekitin kautta. Fontit otettiin kokonaan pois käytöstä, eli sivulla käytettiin vain järjestelmäfontteja. Sivuston suorituskyky parani entisestään, kun latausaika oli 2,38 s ja sivun koko 85,30 kb.

Taulukossa 5 on tiivistetty latausajan ja sivun koon kehittyminen testauksen aikana. Yhteenvetona voidaan todeta, että kuvien näyttäminen data URI:n avulla on kaikista suorituskykyisin ratkaisu. Ne kuitenkin kasvattavat CSS-tiedoston kokoa huomattavasti, joten niiden käyttämisessä tulee olla maltillinen. Lisäksi jokaisesta data URI:lla näytettävästä kuvasta on muistettava tarjota vanhemmille Internet Explorer -selaimille normaali kuvapolku. Vaikka JavaScript-tiedostojen yhdistäminen yhteen tiedostoon nopeutti sivun latautumista, kannattaa tätä vaihtoehtoa kuitenkin harkita. Ladattavasta tiedostosta voi helposti tulla niin suuri, että useamman pienen tiedoston lataaminen olisi nopeampaa.

Taulukko 5. Latausajan ja sivun koon kehittyminen testauksen aikana

Toimenpide	**Latausaika**	**Sivun koko**
Lähtötilanne (layoutin kuvat data URI:na)	6,85 s	823,74 kb
Sosiaalisen median painikkeet jätetään lataamatta mobiilissa	4,31 s	522,28 kb
Pääsisällöt jätetään lataamatta mobiilissa	2,73 s	151,57 kb
JavaScript-tiedostot pienennetään ja yhdistetään	2,71 s	133,22 kb
Layoutin kuvien osoite muutetaan CSS:ssä data URI:sta kuvapoluksi	3,08 s	190,38 kb
Layoutin kuvat logoa lukuun ottamatta muutetaan spriteksi	3,02 s	171,72 kb
Layoutin kuvien vaihtaminen takaisin data URI:in, web-fontit poistetaan käytöstä	2,38 s	85,30 kb

Sivu testattiin lopuksi myös WebPagetestin (n.d.) avulla, jotta nähtiin millainen sivun suorituskyky on desktop-käyttäjällä. Testauksessa valittiin käytettäväksi selaimeksi Chrome ja sen tulokset ovat taulukossa 6.

Taulukko 6. Testaustulokset desktop-selaimella

	Document Complete	**Täysin ladattu**
Aika	3,820 s	5,906 s
Pyyntöjä	33	33
Koko	994 kb	1 066 kb

Näiden testaustulosten puitteissa mobiililaitteella ladattava sivu on kooltaan yli 90 % pienempi ja sen latautumisaika on vain 40 % desktop-sivun latautumisajasta. Nämä testitulokset eivät tietenkään ole täysin luotettavia, mutta ne antavat jonkinlaista osviittaa kovalla kädellä tehdyn suorituskyvyn optimoinnin tuomista hyödystä mobiililaitteissa.

5 Yhteenveto

Opinnäytetyön tavoitteena oli tutkia responsiivisten verkkosivujen kehittämistä WordPress-julkaisujärjestelmässä. Lopputuotteena tehtiin yrityksen tarpeita vastaava responsiivinen WordPress-teema, jonka tarkoituksena oli toimia pohjateemana verkkosivuja kehitettäessä. Opinnäytetyön kirjallisessa osuudessa keskityttiin responsiivisuuden tekniseen puoleen, jolloin WordPressin teeman tekemiseen ei paneuduttu syvällisellä tasolla.

Aihetta rajattiin omien kokemuksen perusteella siten, että työssä keskityttiin lähinnä niihin asioihin, jotka aikaisemmin oli koettu ongelmallisiksi. Aiheen ulkopuolelle jätettiin modernien web-teknologioiden tuomat ominaisuudet kuten paikannus, offline-tietokanta sekä sivun toiminen offline-tilassa.

Työssä käsitellään pintapuolisesti moderneista web-teknologioista HTML5:sta sekä CSS3:sta, jotka ovat mahdollistaneet yksinkertaisemman tavan toteuttaa verkkosivuja uusien ominaisuuksiensa avulla. Kyseisten tekniikoiden tärkeimpiä ominaisuuksia responsiivisuuden suhteen ovat viewport meta tagi, sekä CSS:n media queryt, jotka usein luokitellaan kahdeksi responsiivisuuden kulmakiveksi. Kolmanneksi kulmakiveksi luokitellaan mukautuvat kuvat, jotka ovat myös responsiivisuuden suurin haaste.

Kuvien käyttämiseen responsiivisuudessa ei ole olemassa valmista standardia, joten opinnäytetyössä pyrittiin löytämään ratkaisuja kuvien käsittelemiseen. Kuvien suhteen haasteita asettavat korkearesoluutioiset näytöt, joille tarjottava kuva täytyy olla kaksi kertaa alkuperäistä suurempi, sekä kuvien aiheuttamat haasteet suorituskyvyn suhteen. Jokaiselle näyttökoolle on haastavaa tarjota optimaalisen kokoinen kuva, mutta ratkaisut kuten Adaptive Images sekä Picturefill auttavat tässä tehtävässä.

Responsiivisuudessa on tärkeää pyrkiä tarjoamaan jokaiselle näyttökoolle sama sisältö, mutta tarvittaessa eri muodossa. Tässä ratkaisuna on vaihtoehtoinen sisällön näyttäminen, jolloin käyttäjälle pystytään tarjoamaan laitteen ominaisuuksien

perusteella sisältö mahdollisesti eri muodossa. Tämä vaatii selaimen- tai laitteen ominaisuuksien tarkastelua joko palvelimella tai selaimessa.

Nykyaikaisilla verkkosivuilla yksi suurimmista haasteista on suorituskyky. Vaikka lähtökohtaisesti mobiililaitteessa näytettävän sivun tulisi olla kooltaan desktop-versiota pienempi, näin kuitenkin harvoin tapahtuu käytännössä. Suorituskyvyn suurimpia ongelmia ovat kuvat, joten niiden optimointiin sekä näyttämiseen eri näyttökoissa tulisi kiinnittää huomiota. Lisäksi useat palvelinpyynnöt hidastavat sivun toimintaa, joten niiden määrä täytyisi pitää mahdollisimman pienenä. Esimerkiksi kuvien näyttäminen data URI:n avulla, sekä tyylien ja skriptien yhdistäminen useasta tiedostosta yhteen, vähentävät palvelinpyyntöjen määrää. Opinnäytetyössä tehdyssä suorituskyvyn testauksessa saatiin rajulla optimoinnilla puhelimessa näytettävä sivu yli 90 % pienemmäksi kuin desktop-selaimessa näytettävä sivu. Tämän testauksen perusteella voitiin todeta, että oikein tehty ja optimoitu responsiivinen verkkosivu voi hyvinkin tarjota suorituskyvyllään vastuksen erilliselle mobiilisivulle.

Itse teeman tekemisessä haasteeksi osoittautui laitteen ja selaimen ominaisuuksien testaus, jossa ei pystytty käyttämään alun perin siihen suunniteltua työkalua: Detectoria. Muut työssä käsitellyt lisäosat ja kirjastot pystyttiin kuitenkin sisällyttämään WordPress-teemaan. Seuraavaksi teemaa aletaan käyttää tuotannossa, jolloin sen toimintaa voidaan tarkastella ja parantaa. Toteutettu teema on kuitenkin suuri parannus yrityksen aikaisempaan pohjateemaan. Uusi teema noudattaa paremmin WordPressin standardeja, se on yksinkertaisempi ja siihen on lisätty perustyylejä, jotka puuttuivat aikaisemmasta teemasta. Teemaa varten toteutettu ominaisuus CSS-gridin laskemiseen mahdollistaa joustavamman tavan toteuttaa verkkosivuja, kun suunnittelu ja toteutus eivät ole sidottu valmiiseen gridiin. Teemaan sisällytettiin mobiilinavigaatio, jonka lisääminen oli aikaisemmin erillinen prosessi.

Vaikka olin ennen opinnäytetyön aloittamista toteuttanut yli kymmenen responsiivista verkkosivua, toi työn aikana tehty tutkimus ja opiskelu paljon uutta tietoa. Etenkin kuvien näyttämisen, suorituskyvyn ja vaihtoehtoisen sisällön

näyttämisen suhteen löytyi uusia näkökulmia, joiden avulla responsiivisuuden hyödyntäminen on entistä helpompaa ja tehokkaampaa.

Lähteet

Atwood, J. 2007. *Font Rendering: Respecting The Pixel Grid*. Viitattu 29.3.2013: http://www.codinghorror.com/blog/2007/06/font-rendering-respecting-the-pixel-grid.html

Avola, G. & Raasch, J. 2013. *Mobile Web Development.* United States: Wiley.

Bootstrap. n.d. *Bootstrap*. Viitattu 31.3.2013: http://twitter.github.com/bootstrap/

Breaking Development. 2012. Brad Frost - Responsive Design Vs Separate Mobile Sites: Presidential Smackdown Edition - BDConf, Sept 2012. Viitattu 27.3.2013: http://vimeo.com/50545236

Can I use... 2013. *Can I use...* Viitattu 29.3.2013: http://caniuse.com/

Compuware. 2011. *Infographic – Summary of Key Mobile Survey Findings*. Viitattu 10.3.2013: http://www.gomez.com/thank-you/what-users-want-from-mobile-infographic

Coyier, C. 2010. *Data URIs*. Viitattu 29.3.2013: http://css-tricks.com/data-uris/

Cray, B. n.d. *PX to EM conversion made simple.* Viitattu 10.3.2013: http://pxtoem.com/

Digitoday. 2012. *Retina-sanasta tuli Applen tuotemerkki*. Viitattu 17.3.2013: http://www.itviikko.fi/uutiset/2012/12/05/retina-sanasta-tuli-applen-tuotemerkki/201243331/7

Edwards, M. n.d. *Device pixel density tests*. Viitattu 10.4.2013: http://bjango.com/articles/min-device-pixel-ratio/

Filament Group. 2012. *Compressive Images*. Viitattu 17.3.2013: http://filamentgroup.com/lab/rwd_img_compression/

Friedman, V. 2012. *Responsive Web Design: Clever Tips and Techniques*. Viitattu 29.3.2013: http://www.slideshare.net/vitalyfriedman/responsive-web-design-clever-tips-and-techniques

Frost, B. n.d. *Mobile-First Responsive Web Design (demo)*. Viitattu 31.3.2013: http://bradfrostweb.com/demo/mobile-first/

Google Developers. n.d. *PageSpeed Insights*. Viitattu 27.3.2013: https://developers.google.com/speed/pagespeed/insights#

Google web fonts. n.d. *Google web fonts*. Viitattu 29.3.2013: http://www.google.com/fonts/

Hazaël-Massieux, D. 2009. *Return of the Mobile Stylesheet*. Viitattu 6.3.2013: http://alistapart.com/article/return-of-the-mobile-stylesheet

HTML Reference - (HTML5 Compliant). n.d. Viitattu 1.3.2013: http://www.w3schools.com/tags/default.asp

html5shiv. 2013. Viitattu 1.3.2013: https://github.com/aFarkas/html5shiv

Jankord, B. 2012. *Cross Browser Retina/High Resolution Media Queries*. Viitattu 17.3.2013: http://www.brettjankord.com/2012/11/28/cross-browser-retinahigh-resolution-media-queries/

Jehl, S. 2012. *An Ajax-Include Pattern for Modular Content*. Viitattu 14.4.2013: http://filamentgroup.com/lab/ajax_includes_modular_content/

Jehl, S. 2013. *Picturefill*. Viitattu 23.3.2013: https://github.com/scottjehl/picturefill

Kadlec, T. 2012. *Media Query & Asset Downloading Results*. Viitattu 24.3.2013: http://timkadlec.com/2012/04/media-query-asset-downloading-results/

Kadlec, T. 2013. *Implementing Responsive Design.* United States of America: New Riders.

Kraken.io. 2012. *Kraken Image Optimizer*. Viitattu 10.3.2013: http://kraken.io/

MEOM. (2012). *Tekijät*. Viitattu 12.3.2013: http://www.meom.fi/tekijat/

Mobile Best Practices. n.d. *Be Mindful of Font Choice*. Viitattu 29.3.2013: http://mobilewebbestpractices.com/visual-design/be-mindful-of-font-choice/

Mobile Detect. (2013). *Mobile Detect*. Viitattu 29.3.2013: http://mobiledetect.net/

Mobitest. 2012. *Free Mobile Web Performance Measurement Tool*. Viitattu 27.3.2013: http://mobitest.akamai.com/m/index.cgi

Modernizr. 2013. *Morernizr*. Viitattu 27.3.2013: http://modernizr.com/

Myers, C. B. 2011. *Josh Clark debunks the 7 Myths of Mobile Web Design*. Viitattu 26.3.2013: http://thenextweb.com/dd/2011/11/07/josh-clark-debunks-the-7-myths-of-mobile-web-design/

Olsen, D. 2012. *Detector*. Viitattu 29.3.2013: http://detector.dmolsen.com/

Panique. 2013. *Stack overflow: font smoothing?* Viitattu 29.3.2013: http://stackoverflow.com/a/11493510

Pearce, J. 2010. *Modernizr on the server-side*. Viitattu 27.3.2013: http://tripleodeon.com/2010/10/modernizr-on-the-server-side/

Pictos. 2012. *Pictos*. Viitattu 12.3.2013: http://pictos.cc/

Platform Versions. 2013. Viitattu 25.2.2013: http://developer.android.com/about/dashboards/index.html#Platform

Responsive Images Community Group. 2013. *Responsive Images Community Group*. Viitattu 17.3.2013: http://responsiveimages.org/

Salminen, V. n.d. *Responsive Nav*. Viitattu 14.4.2013: http://responsive-nav.com/

ScientiaMobile, Inc. 2012. *WURFL*. Viitattu 30.3.2013: http://wurfl.sourceforge.net/

Shifticons. (2013). Viitattu 12. 3 2013 osoitteesta https://www.shifticons.com/

Smashing Magazine. 2012. *The Mobile Book.* Freiburg: GmbH.

Smith, D. 2013. *iOS Version Stats*. Viitattu 25.2.2013: http://david-smith.org/iosversionstats/

StatCounter. 2013. *StatCounter - Global Stats*. Viitsttu 25.2.2013: http://gs.statcounter.com/

Stocks, E. J.;Boag, P.;Andrew, R.;Schwarz, B.;Verou, L.;Storey, D.;. . . Clarke, A. 2012. *The Smashing Book #3 - Redesign The Web.* Freiburg, Germany: Smashing Media GmbH.

STT-REUTERS. 2011. *Tutkimuslaitos: Android jyräsi jo Symbianin ohi*. Viitattu 25.2.2013: http://www.iltalehti.fi/digi/2011013113104238_du.shtml

T, A. 2012. *Finger-Friendly Design: Ideal Mobile Touchscreen Target Sizes*. Viitattu 31.3.2013: http://uxdesign.smashingmagazine.com/2012/02/21/finger-friendly-design-ideal-mobile-touchscreen-target-sizes/

Talja, T. 2012. *Puhelinten ja taululaitteiden näyttö- ja pikselikoot*. Viitattu 10.4.2013: http://tanssivakarhu.posterous.com/puhelinten-ja-taululaitteiden-naytto-ja-pikse

Tate, T. 2012. *Coping with Sub-pixel Rounding in IE*. Viitattu 14.4.2013: http://tylertate.com/blog/2012/01/05/subpixel-rounding.html

Thebeebs. 2011. *The differences between IE9 on the desktop and IE9 on WP7*. Viitattu 22.4.2013: http://www.ubelly.com/2011/11/the-differences-between-ie9-on-the-desktop-and-ie9-on-wp7/

Ward, B. 2013. *Deprecating IE6 and IE7 support in Twitter for Websites (TFW)*. Viitattu 21.42013: https://dev.twitter.com/blog/tfw-ie6-ie7-support

Way, J. 2011. *28 HTML5 Features, Tips, and Techniques you Must Know*. Viitattu 8.3.2013: http://net.tutsplus.com/tutorials/html-css-techniques/25-html5-features-tips-and-techniques-you-must-know/

WebPagetest. n.d. *WebPagetest*. Viitattu 22.4.2013: http://www.webpagetest.org/

Wilcox, M. n.d. *Adaptive Images*. Viitattu 24.3.2013: http://adaptive-images.com/

WordPress. 2013. *Theme Review*. Viitattu 1.4.2013: http://codex.wordpress.org/Theme_Review

WordPress. n.d. *Theme Development*. Viitattu 1.4.2013: http://codex.wordpress.org/Theme_Development

Liitteet

Liite 1: HTML5:n uudet elementit

HTML Reference - (HTML5 Compliant) n.d.

Elementti	**Selitys**
<article>	Määrittää artikkelin, eli sisällön, jonka voi erottaa itsenäiseksi sisällökseen. Articlen käyttötarkoituksia ovat mm. blogipostaukset, kommentit ja uutiset.
<aside>	Määrittää sisältöä, joka on pääsisällön sivussa. Asiden sisällön tulisi liittyä sitä ympäröivään sisältöön, mutta sitä voi käyttää myös sivupalkkina.
<figure>	Määrittää omavaraisen sisällön, kuten kuvituksen, kaavion tai kuvan
<figcaption>	Määrittää kuvauksen figure-elementille
<footer>	Määrittää sivun footerin, joka sisältää yleensä tekijänoikeustiedot, yhteystiedot ym.
<header>	Ylätunniste dokumentille tai osiolle. Headeria suositellaan käyttämään kehyksenä johdantomaiselle sisällölle, sekä navigaatiolinkeille.
<hgroup>	Yhdistää otsikkotason elementit (<h1>-<h6>), mikäli niitä on monta peräkkäin.
<mark>	Määrittää merkatun tekstin, jota halutaan korostaa
<nav>	Määrittää navigaation linkit, suositeltavaa käyttää vain olennaisimmille linkeillä
<progress>	Esittää jonkin toiminnon edistymisen, käytetään yhdessä JavaScriptin kanssa
<ruby>, <rp>, <rt>	Määrityksiä Itä-Aasian typografisien merkkien kääntämiseen
<section>	Määrittää sektioita dokumentissa, kuten kappaleita, headereita, footereita tai vastaavia
<time>	Määrittää ajan

Liite 2. Ohjeistus Kalakala-teeman käyttämiseen

PHP:n käyttäminen

Lainausmerkit	Käytä puolilainausmerkkejä. Käytä tarvittaessa attribuuteihin menevissä teksteissä `esc_attr()` -functiota
Sisennys	Käytä sisennyksessä sarkaimia, älä välilyöntejä
Välien käyttäminen	Laita välilyönti aina pilkun jälkeen, sekä vertailun, merkkijonon tai sijoitusoperaattorin molemmille puolille. Laita välilyönti funktioiden sekä lohkojen if, else, elseif, foreach, for, switch sulkumerkkien molemmille puolille.
Nimeäminen	Käytä muuttujien, funktioiden sekä toimintojen nimissä pieniä kirjaimia ja erota sanat alaviivalla Laita luokkien nimiin sanan ensimmäinen kirjain isolla ja erota sanat väliviivalla Nimeä tiedostot pienillä kirjaimilla ja erota sanat väliviivalla
Ternary Operator	`(if statement is true) ? (do this) : (else, do this);` Käytä Ternary operatoria vain lausunnoissa, jotka ovat joko tosia tai epätosia

Muuttujan sijoittelu vertailussa (Yoda Conditions)	Sijoita loogisissa vertailuissa muuttuja aina oikealle puolelle
	`if( true == $the_force ) {`
	`  $victorious = you_will( $be );`
	`}`

Teeman rakenne ja tiedostojen sisältö

Yleiset teematiedostot

404.php Käytetään näytettäessä sivu, jota ei löydy. Staattinen sisältö: Sivun otsikko: Ei löytynyt Kuvaus virhestä: Valitettavasti etsimääsi sivua tai artikkelia ei löytynyt	*archive.php* Arkistosivu artikkeleille. Sisältää tuen artikkeleiden näyttämiselle kategorian, päivän, kuukauden, vuoden ja kategorian mukaan. Avainsanat näytetään kategorioina.

	Näytettävä sisältö: Nimi, jonka perusteella arkisto näytetään * sama kuin index.php **Staattinen sisältö:** Merkinnät kategoriasta Merkinnät päivältä Merkinnät kuukaudelta Merkinnät vuodelta * sama kuin index.php
attachment.php Liitteen sivu. Liitetystä kuvasta näytetään medium-kokoinen kuva. **Näytettävä sisältö:** Liitteen nimi Päivämäärä Liitetty kuva	*author.php* Artikkelit kirjoittajalta. **Näytettävä sisältö:** Kirjoittajan nimi * sama kuin index.php **Staattinen sisältö:** Merkinnät kirjoittajalta * sama kuin index.php
comments.php Kommentit ja lomake kommentointia varten. Otetaan käyttöön single-tiedostossa comments_template()-funktiolla. **Näytettävä sisältö:** Artikkelin nimi Kommentoijan nimi ja kommentti Kommentointiaika Kommenttien sivutus Kommenttilomake **Staattinen sisältö:** Yksi kommentti artikkeliin % kommenttia artikkeliin Vastaa Vanhemmat kommentit	*footer.php* Sivun alaosa, otetaan käyttöön teematiedostoissa get_footer()-funktiolla. Tiedoston lopussa on Responsive navin vaatima JavaScript. Mobiilinavigaation avauspainike määritetään label-kohdassa. **Staattinen sisältö:** Rakkaudella MEOM

Uudemmat kommentit Kommentointi on suljettu	
front-page.php Etusivu, mikäli hallintapaneelissa etusivuksi on määritetty jokin sivu. **Näytettävä sisältö:** * sama kuin page.php **Staattinen sisältö:** * sama kuin page.php	*header.php* Sivun yläosa, otetaan käyttöön teematiedostoissa get_header()-funktiolla. Head-tagin sisällä on pätkä Adaptive Imagesin vaatimaa JavaScriptiä. **Näytettävä sisältö:** Navigaatio. Mikäli kustomoitu navigaatio on tehty, näytetään se. Oletuksena asetettu näyttämään kustomoitu navigaatio nimeltä Main Navi.
single.php Yksittäisen artikkelin sisältö, metatiedoissa ei näytetä kommentteja. **Näytettävä sisältö:** Kommenttien haku Linkki seuraavaan ja edelliseen artikkeliin * sama kuin index.php (ei sivutusta) Staattinen sisältö: Muokkaa artikkelia	*page.php* Oletuspohja sivulle. **Näytettävä sisältö:** Otsikko Sisältö **Staattinen sisältö:** Muokkaa sivua
search.php Näyttää hakutuloksen sisällön, ei sivutusta. **Näytettävä sisältö:** Hakusana Hakulomake, mikäli tuloksia ei löydy * sama kuin index.php **Staattinen sisältö:** Hakutulokset sanalle:	*sidebar.php* Sivupalkki, haetaan artikkeleihin liittyvissä sivuissa get_sidebar()-funktiolla **Näytettävä sisältö:** Oletuksena oleva sivupalkki hallintapaneelin vimpaimista

Ei tuloksia Haullasi ei löytynyt tuloksia, yritä uudelleen * sama kuin index.php	

index.php

Oletustiedosto, mikäli sisältötyypille ei löydy omaa tiedostoa. Näyttää mm. kaikki artikkelit.

Pääotsikkona näytetään se sivu, jolla artikkelit sijaitsevat.

Näytettävä sisältö:

Artikkelin otsikko ja linkki siihen

Artikkelin julkaisupäivä

Artikkelin kategoria

Kommenttien määrä

Kirjoittajan nimi

Sisältö

Sivutus

Sivupalkki

Staattinen sisältö:

Ei kommentteja, 1 kommentti, % kommenttia, Kommentointi suljettu

Muokkaa artikkelia

Vanhemmat kirjoitukset

Uudemmat kirjoitukset

Yhtään artikkelia ei ole vielä julkaistu

Functions-tiedosto

Tiedostossa on kolme yleistä muuttujaa, joita voidaan käyttää teematiedostoissa

- HOME_URI – sivuston osoite
- THEME_URI – teeman osoite
- THEME_IMAGES – teematiedostossa images-kansiossa olevat kuvat

Funktio **kk_setup** ajetaan aina sivun latautuessa. Se tekee seuraavat toiminnot:

- Lisää automaattiset feed-linkit
- Lisää tuen artikkelikuville
- Lisää tekstieditorin tyylitiedoston
- Rekisteröi paikan navigaatiolle Main Menu
- Poistaa RSD-linkit headista
- Poistaa Windows Live Writerin headista
- Poistaa WordPressin versionumeron headista

Poistaa artikkeleiden yhteyksiin liittyvä linkit
Poistaa WordPressin galleriaan liittyvän oletustyylin

Funktio **kk_widgets_init** suorittaa vimpaimiin liittyvät toiminnot:
Rekisteröi oletussivupalkin
Poistaa WordPressin lisäämään uusiin kommentteihin liittyvän tyylin

Funktio **kk_admin_init** suorittaa kirjautuneen käyttäjän kohdalla seuraavat toiminnot:
Mikäli kyseessä ei ole pääkäyttäjä, häneltä poistetaan mahdollisuus WordPressin päivittämiseen hallintapaneelista

Funktio **kk_style_script** suorittaa seuraavat toiminot tyylien ja skriptien suhteen:
Vaihtaa jQueryn WordPressin paikallisesta googlen tarjoamaan
Lisää teeman oletustyylin
Lisää (tarvittaessa) AjaxInclude JavaScriptin
Lisää Modernizr JavaScriptin
Lisää Responsive Nav JavaScriptin
Lisää Picturefillin JavaScriptin

Funktio **kk_limit_content** on teemasta kutsuttava funktio, jolla sivun tai artikkelin sisällöstä näytetään vain tietty määrä merkkejä. Funktio ottaa ainoana parametrina numeron, johon sisällön pituus halutaan rajoittaa. Käyttö:

```
kk_limit_content( 150 );
```

Tulostaa 150 merkkiä sisällöstä

Funktio **kk_limit_title** toimii samoin kuin kk_limit_content, mutta se on tarkoitettu otsikolle. Käyttö:

```
kk_limit_title( 10 );
```

Tulostaa 10 merkkiä otsikosta

Funktio **kk_picturefill** palauttaa kuvan Picturefillillä näytettävän kuvan näyttämiseen tarvittavan rakentee. Funktio ottaa kolme parametria: kuvan nimen, kuvan alt-tekstin sekä kuvan kuvapolun. Oletuksena kuvapolkuna käytetään teeman images-kansiota. Käyttö:

```
echo kk_picturefill( 'aamu', 'ihana aamu' );
```

Tulostaa Picturefillin rakenteen kuvasta aamu.jpg alt-tekstillä ”ihana aamu”. Kuva haetaan teeman images-kansiosta.

Funktio **kk_datauri** palauttaa kuvan Data URI:n base64-muodossa. Se ottaa parametreina kuvan osoitteen sekä sen mime-tyypin. Käyttö:

```
echo kk_datauri( 'http://osoite.fi/kuva.jpg', 'image/jpg');
```

Tulostaa kuvan Data URI:n. Yleisimmät mime-tyypit ovat: image/jpg, image/png ja image/svg+xml.

Kansiorakenne

CSS

Tyylitiedostot teeman päätyyliä lukuunottamatta. Oletuksena tekstieditoriin vaikuttava editor-style.css

images

Teemassa käytettävät kuvat

includes

Ulkopuoliset PHP-tiedostot, joka eivät kuitenkaan ole lisäosia. Oletuksena adaptive-images.php ja ajax-includes.php

js

JavaScript-tiedostot. Oletuksena picturefill, ajaxInclude.js, html5shiv.js, modernizr.js ja responsice-nav.min.js

svg

SVG-tiedostot

HTML:n ja CSS:n periaatteet

HTML-rakenne ja valitsimet

Header

Pääheaderille annetaan ID:ksi primary-head, käytä header-elementtiä

Navigaatio

Päänavigaation ID:ksi annetaan primary-nav, käytä nav-elementtiä. Navigaatio voi olla header-elementin sisällä.

Sisältö

Lähtökohtaisesti sivun koko sisältö headeria ja footeria lukuun ottamata on divin sisällä, jonka luokka on content. Itse sivun sisällön rakenne on seuraava:

```
section id=”main”
    article
        h1 class=”page-title”
        div class=”page-content
```

Artikkeleiden suhteen rakenne on seruaava:

```
section id=”main”
    h1 class=”page-title
    article
        h2 class=”entry-title
```

```
        div class=”entry-meta
        div class=”entry-content
    nav class=”paginate-nav
```

Tarkoituksena on pitää sivut ja artikkelit jaoittelemalla ne luokilla page ja entry. Pääotsikko tulee merkitä h1-tason otsikoksi ja artikkelin otsikko artikkelilistauksissa h2-tason otsikoksi. Artikkelin omalla sivulla sen otsikko voi olla h1-tason otsikko.

Sivupalkki
Sivupalkki tulee content-divin sisään. Se määritetään aside-elementin avulla, jolle annetaan luokka sidebar. Artikkeleihin liittyvällä sivulla sivupalkin ID on blog-sidebar

Footer
Footer määritellään footer-elementin avulla ja sille annetaan ID primary-footer. Footerin sisälle ilmaistaan MEOM sille sopivalla tavalla.

CSS:n rakenne

CSS noudattaa seuraavaa rakennetta:

Gridin laskemiseen tarvittavat funktiot

Muuttujat

Muut funktiot

HTML5 elementteihin liittyvät normalisoinnit

Perustyylit (html + body)

Gridin muodostaminen

Typografia

Sisällytetty sisältö

Figurit

Lomakkeet

Taulukot

WordPress galleria

WordPress kommentit

Yleiset luokat

Pääheader ja -navigaatio

Sivuun liittyvät tyylit

Blogiiin liittyvät tyylit

Sivupalkki

Footer

Media queryt

Muokkaa perustyylejä tarpeen mukaan, mutta ala lisäämään omia tyylejä vasta pääheaderin jälkeen tyyleille sopivaan kohtaan, jonka teet tarvittaessa. Älä käytä liian syviä valitsimia, eli ei `nav#primary-nav ul li a`, vaan `#primary-nav a`.

Suhteessa skaalautuvat taustakuvat CSS:n avulla

Mikäli haluat käyttää taustakuvaa niin, että kuva pienenee leveyssuunnassa ja sen korkeus muuttuu suhteessa leveyteen, määritä seuraava:

```
<div class="img-holder">
       <div class="dummy-holder">
              <div class="img-element kuvan-oma-luokka"></div>
       </div>
</div>
```

Anna rakenteelle seuraava tyyli:

```
.img-holder {
  display: inline-block;
  position: relative;
  width: 100%;
  max-width: 300px;
}
.dummy-holder {
  margin-top: 100%;
}
.img-element {
  position: absolute;
  top: 0;
  bottom: 0;
  left: 0;
  right: 0;
  background-repeat: no-repeat;
  background-size: 100%;
}
.img-element.kuvan-oma-luokka {
  background-image: url(images/kuva.jpg);
}
```

Dummy-holderin margin-top arvo määrittää kuvan mittasuhteen, arvon ollessa 100%, kuva näkyy neliönä. IE7:lle täytyy määrittää muutamia eriäviä ominaisuuksia, koska se ei tule inline-blockia:

```
.img-element { position: relative; }

.dummy-holder { margin: 0; }

.img-element.kuvan-oma-luokka { width: 300px; height: 300px; }
```