jamk.fi

VERKKOSIVUSTOJEN KEHITTÄMISEN NOPEUTTAMINEN

Janne Leppänen

Opinnäytetyö
Marraskuu 2013

Mediatekniikan koulutusohjelma
Tekniikan ja liikenteen ala

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

KUVAILULEHTI

Tekijä(t) LEPPÄNEN, Janne	Julkaisun laji Opinnäytetyö	Päivämäärä 12.11.2013
	Sivumäärä 54	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty (X)

Työn nimi
Verkkosivustojen kehittämisen nopeuttaminen

Koulutusohjelma
Mediatekniikan koulutusohjelma

Työn ohjaaja(t)
MANNINEN, Pasi

Toimeksiantaja(t)
Webtakomo

Tiivistelmä

Opinnäytetyö tehtiin toimeksiantona Webtakomolle, joka suunnittelee ja toteuttaa verkkosivustoja ja verkkopalveluita. Yrityksellä ei aiemmin ollut käytössä mallipohjaa, vaan jokainen projekti oli aloitettu täysin alusta. Opinnäytetyön tavoitteena oli koostaa mallipohja, jonka avulla uusien projektien aloittaminen olisi joustavaa. Mallipohjaan pyrittiin sisällyttämään projekteissa toistuvia tekniikoita ja verkkokehityksen uusimpia käytänteitä. Samalla perehdyttiin ohjelmiin ja tekniikoihin, jotka eivät tuoneet mallipohjaan lisää toiminnallisuuksia, mutta jotka paransivat verkkokehitysprojektien kulkua.

Opinnäytetyössä perehdyttiin verkkokehityksen yleisimpien haasteiden ratkaisemiseen. Pyrittiin ymmärtämään erilaisten päätelaitteiden ominaisuuksia ja niiden huomioimista kehityksessä. Verkkokehitystä tarkasteltiin myös käytettävyyden ja saavutettavuuden kannalta.

Opinnäytetyön tuloksena syntyi WordPress-teeman ympärille rakennettu ja itsenäinen HTML-pohjainen mallipohja, joissa hyödynnettiin HTML5- ja CSS3-tekniikoita. Pohja pyrkii ottamaan vaikutteita ja yhdistämään verkkokehityksen parhaita käytänteitä. Pohjaan liittettiin mm. HTML5-Boilerplate, jota laajennettiin lisäämällä sivuston sisällölle pohjarakenne. Mallipohjan ansiosta uusien projektien aloittaminen helpottui, sillä keskeisimmät kehityksessä tarvittavat tekniikat olivat mallipohjassa valmiina.

Tuloksena syntyi mallipohja, joka on valmis käyttöön. Sitä kehitetään myös jatkossa, sillä se pyrkii heijastamaan verkkokehityksen uusimpia tekniikoita ja parhaita käytäntöjä.

Avainsanat (asiasanat)
Responsiivisuus, verkkokehitys, mallipohja

Muut tiedot

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

DESCRIPTION

Author(s)	Type of publication	Date
LEPPÄNEN, Janne	Bachelor´s Thesis	12.11.2013
	Pages 54	Language Finnish
		Permission for web publication (X)

Title
Hastening page development

Degree Programme
Media Engineering

Tutor(s)
MANNINEN, Pasi

Assigned by
Webtakomo

Abstract

This Bachelor's thesis was conducted as an assigment for Webtakamo, which designs and develops websites and web services. The company did not have a web site template earlier, but each project was started from scratch. The objective of this bachelor's thesis was to collect techniques, which seem to repeat from project to project, into a template. The template would make it easier and more flexible to start new projects. Another objective in the Bachelor's thesis was to research techniques, which did not add new features to template but would help to improve the workflow in projects.

In the Bachelor's thesis the most common challenges in web development are discussed. The aim was to understand and take account into the characteristics of various types of web environments. Usability and accessibility in web development were examined as well.

The results of the Bachelor's thesis were WordPress starter theme and standalone HTML template, which take advantage of HTML5 and CSS3. HTML5 Boilerplate was included in the template and it was expanded by adding a structure for a content. With the template new projects were much easier to start, because commonly used features were already included.

The result was a template that is ready to use. It will be developed further in future, because it will be reflecting latest techniques and best practises of web development.

Keywords
Responsive web design, web development, template

Miscellaneous

SISÄLTÖ

Kuviot

Taulukot

1 TYÖN LÄHTÖKOHDAT

1.1 TAUSTA JA TOIMEKSIANTAJA

Verkkokehityksen alkuvaiheilla kehittäjien haasteena oli selainten standardien puute. Nykyään eri selaimet noudattavat melko tarkasti standardeja, mutta selainten kirjo ja uudet päätelaitteet ovat astuneet kuvaan. Verkkokehittäjien on tärkeää ymmärtää, miten tekniikat toimivat eri selaimilla ja millaisen käyttökokemuksen ihmiset saavat uusilla päätelaitteilla, kuten kosketusnäytöillä tai pelikonsoleilla. Uusia kehittäjän arkea helpottavia tekniikoita tulee koko ajan lisää, mutta samalla peruskäyttäjät jäävät jälkeen selainten päivittämisessä. Kehittäjän on siis pystyttävä tarjoamaan samalla uusia tekniikoita ja huolehtimaan, että ne eivät haittaa käyttökokemusta selaimilla, jotka eivät tue niitä.

Toimeksiantaja oli Jyväskyläläinen yksityisyrittäjä, joka toimii yritysnimellä Webtakomo. Yritys on aloittanut toimintansa keväällä 2013. Webtakomo tekee nettisivuja pienille yrityksille ja yhteisöille sekä tarjoaa sivustojen teknistä toteutusta mainostoimistoille. Webtakomolla ei ole tarkasti mietittyä mallipohjaa, vaan projektit on aloitettu useimmiten täysin alusta. Kunnollinen mallipohja nopeuttaisi projektien aloitusta huomattavasti. Mallipohjan (template) ansiosta sivustonkehityksessä ei tarvitsisi miettiä selainten välisiä eroja joka kerta uudestaan, vaan suurimmat ongelmatilanteet olisi jo valmiiksi ratkaistu.

1.2 TAVOITTEET

Opinnäytetyössä oli tavoitteena perehtyä verkkokehityksen nykypäivän haasteisiin ja koostaa Webtakomolle mallipohja, joka helpottaisi ja nopeuttaisi uusien projektien aloittamista. Mallipohjan tulisi tasoittaa selainten välisiä eroja ja nopeuttaa responsiivisten sivustojen rakentamista. Tämä pyrittiin saavuttamaan valitsemalla mallipohjaan tekniikoita, joita useimmissa projekteissa tarvitaan. Mallipohjaa raken-

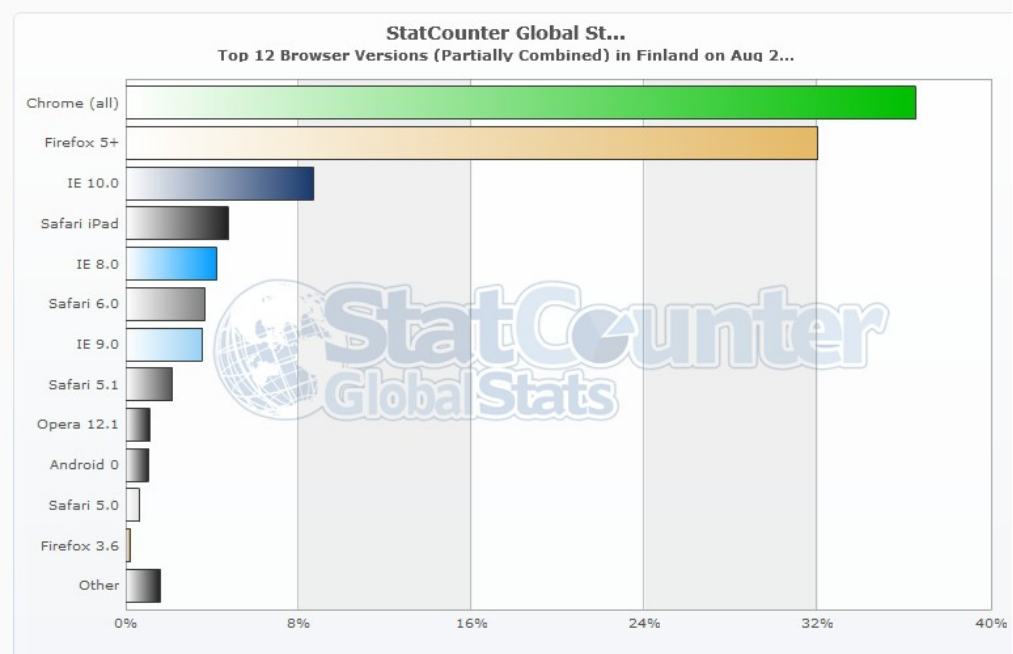
taessa oli tarkoitus vertailla verkkotekniikoita keskenään ja valita niistä parhaat. Lisäksi opinnäytetyössä pyrittiin tutustumaan tekniikoihin, jotka eivät tuo sivustoihin lisää ominaisuuksia, mutta nopeuttavat sivustojen kehittämistä.

Opinnäytetyössä ei haluttu vertailla sisällönhallintajärjestelmiä keskenään, sillä Webtakomo on tottunut käyttämään WordPressiä tuotannossaan. Mallipohja pyrittiin rakentamaan itsenäiseksi pohjaksi ja WordPress-teemaksi.

2 MODERNIN NETTISIVUN HAASTEET

2.1 SELAIMET JA NIIDEN VERSIOT

Vanhojen selainten ja uusien tekniikoiden yhdistäminen on usein haasteellista ja aikaa vievää. Eniten selaimista päänvaivaa kehittäjille ovat aiheuttaneet Internet Explorerin vanhemmat versiot, jotka ovat olleet suhteellisen käytettyjä. Internet Explorerin vanhemmat versiot tukevat melko heikosti uusia tekniikoita sekä W3C:n asettamia standardeja. Vuosien saatossa näiden selainten käyttö on kuitenkin vähentynyt selvästi, ja kuviosta 1 käy ilmi, kuinka StatCounterin (2013) mukaan Internet Explorerin versiota 7 käyttää Suomessa alle 0,5 %. Tästä huolimatta versio 8 on kuitenkin pitänyt pintansa ja sitä käyttää yhä noin 4 %, joten nettisivustot olisi hyvä pitää yhteensopivina version 8 kanssa.



Kuvio 1. Selainten käyttö Suomessa elokuussa 2013 (StatCounter 2013)

W3C julkaisee uusista tyylimäärityksistä luonnokset ennen kuin se standardoi ne. Selainten valmistajat pääsevät näin lisäämään tyylimääritykset hyvissä ajoin selaimiin.

Selaimet asettavat aluksi uusiin tyylimäärityksiin etuliitteet, jolloin sivustojen kehittäjät voivat halutessaan käyttää tiettyjä ominaisuuksia vain muutamilla selaimilla. Esimerkiksi opinnäytetyön kirjoitushetkellä alla oleva transform-määritys toimii ainoastaan uusimmassa Firefox-selaimessa.

```
transform : scale(2,2);
```

Skaalauksen saa kuitenkin toimimaan myös muissa selaimissa lisäämällä tarvittavat etuliitteet seuraavasti.

```
-ms-transform: scale(2,2);  /*IE 9*/
-webkit-transform: rotate(2,2); /*Safari ja Chrome*/
```

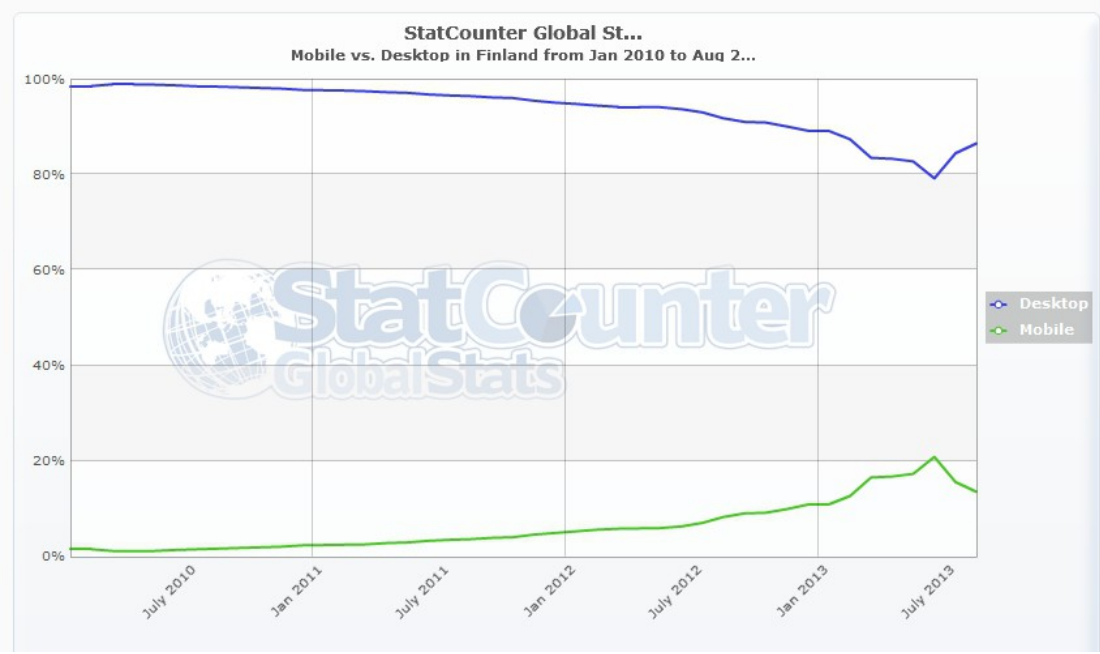
Uusia tyylimäärityksiä lisättäessä on hyvä tarkistaa selaimien tuki niille. Hyvä sivusto tekniikoiden tuen tarkistamiseen on Can I use -sivusto (www.caniuse.com), jossa on esitelty uusimpien tekniikoiden tuki ja mahdolliset puutteet. Nettisivustoille on myös saatavilla Modernizr-kirjasto, joka pystyy tunnistamaan tukeeko selain haluttuja tekniikoita.

Sivuston toimivuuden testaaminen on tärkeää. Paras mahdollinen tapa on testata sivusto alkuperäisillä laitteilla ja selaimilla. Koska alkuperäisillä ympäristöillä on kuitenkin käytännössä vaikea testata, on olemassa palveluita, joilla voidaan testata sivusto usealla eri selaimella ja laitteella. Browserling (https://browserling.com/) on selaimessa toimiva palvelu, jolla voidaan testata miltä sivuston ulkoasu näyttää useilla selaimilla (Krumins 2013). Sivustoja kannattaa myös testata mobiiliemulaattoreilla kuten Operan Mobile Emulator -ohjelmalla.

2.2 PÄÄTELAITTEET JA RESOLUUTIOT

Internetiä hyödyntävien mobiililaitteiden yleistyttyä nettisivujen kehittäjät halusivat erottua edukseen tekemällä sivustoja, jotka oli suunniteltu pelkästään mobiililaitteille. Pian kuitenkin huomattiin, että mobiililaitteiden ja työpöytäkoneiden välinen ero kapeni jatkuvasti. Lisäksi erillisten mobiilisivustojen rakentaminen oli työlästä. Kuviosta 2 huomataan, kuinka mobiililaitteiden käyttö on kasvanut tasaisesti viimeisten kol-

men vuoden aikana. Suomessa kesäkuun alussa Internetin käyttö mobiililaitteilla oli noin 21 %. Kesäkuun jälkeen Suomessa on tapahtunut notkahdus, mutta koko maailman kattavassa vertailussa tätä notkahdusta ei kuitenkaan ole tapahtunut.



Kuvio 2. Mobiililaitteiden ja työpöytäkoneiden käytön kehitys vuodesta 2010 (StatCounter 2013)

W3 Consortium on standardoinut CSS2-versiossa mediatyypit, joilla voidaan kirjoittaa eri päätelaitteille omat tyylimääritykset. Handheld-mediatyypillä pystytään kohdistamaan tyylimääritykset päätelaitteille, joissa on pieni näyttö.

```
@media handheld {
        .container{
                width: 92%;
                margin: 0% 4%;
        }
}
```

Suurin osa verkkokehittäjistä eivät nähneet puhelimien selaimissa potentiaalia niiden alkeellisuuden takia. Myöhemmin puhelimien selainten parantuessa puhelimien kehittäjät eivät taas nähneet syytä miksi puhelimien tulisi käyttää handheld-mediatyyppiä, sillä harva sivusto hyödynsi sitä.

Koska päätelaitteiden kehittäjät eivät noudattaneet näitä standardeja, niistä ei ollut juuri hyötyä sivustojen kehittäjille. Vuonna 2009 W3C alkoi CSS3-version myötä kehittää paranneltuja mediatyyppejä, joilla pystyisi kohdistamaan tyylimäärittelyt laitteille niiden ominaisuuksien mukaan.

Toukokuussa 2010 Ethan Marcotte esitti kasan ratkaisuja, joilla voitaisiin vastata mobiililaitteiden tuomaan haasteeseen. Pääasiassa mukautuvien sivustojen kehitys perustuisi suhteellisiin mittoihin ja CSS3:n mediakyselyihin. Tällaiselle toimintamallille syntyi nimeksi Responsive web design. Se pyrki nimensä mukaisesti tarjoamaan verkkosivustoja, jotka mukautuvat käyttäjän päätelaitteille sopiviksi. (Marcotte 2011.)

3 MALLIPOHJAN KOOSTAMINEN

3.1 SELAIMEN OMINAISUUKSIEN TUNNISTAMINEN

Opinnäytetyön lähtökohtana oli tehdä joustava mallipohja, joka toimisi mahdollisimman monella päätelaitteella. Tästä johtuen oli tärkeää pystyä tunnistamaan selainten rajoitukset suunnittelu- ja toteusvaiheessa. Nykyisessä tekniikkaviidakossa tulisi samaan aikaan pystyä tarjoamaan moderneille selaimille parhaat mahdolliset tekniikat, mutta silti sivustojen pitäisi toimia myös rajoittuneimmilla selaimilla. On hyvin tärkeää osata varautua esimerkiksi siihen, että käyttäjillä ei ole asetettu selaimeen JavaScriptille tukea tai loppukäyttäjien päätelaitteet toimivat kosketusnäytöllä.

Modernizr on JavaScript-kirjasto, joka pystyy tunnistamaan selaimen tukemat tekniikat. Modernizr on tiivistetyltä kooltaan alla 20 kilotavua ja sisältää mahdollisuuden räätälöidä vain halutut osat mukaan pakettiin. Selainten tukemien tekniikoiden tunnistamisen lisäksi Modernizr tekee HTML5-elementit käytettäviksi vanhemmilla selaimilla, jolloin sivustoille ei tarvitse lisätä erikseen esimerkiksi html5shim-kirjastoa. Lisäämällä HTML-elementtiin no-js-luokan, voidaan Modernizrilla myös varautua siihen, että sivuston käyttäjillä ei ole välttämättä tukea JavaScriptille. Modernizr-kirjasto on hyvä sijoittaa muista JavaScript-tiedostoista poiketen head-elementin sisälle, jotta HTML5-elementit voitaisiin luoda ennen kuin ne esitellään dokumentissa. Loput JavaScript-tiedostot tulee sijoittaa body-elementin loppuun, jotta ne eivät hakukoneiden kriteerien mukaan hidastaisi dokumentin lataamista. Vaikka Modernizrin avulla on todella helposti saatavilla listaus selaimen ominaisuuksista, pitää muistaa että selaimet ja päätelaitteet kehittyvät koko ajan ja poikkeavat toisistaan, joten Modernizr ei välttämättä toimi täydellisesti jokaisessa tapauksessa. (Modernizr dokumentaatio 2013.)

Modernizr lisää HTML-elementtiin listan tekniikoista, joita selain tukee tai ei tue. Näin voidaan tyylimäärittelyillä varautua esimerkiksi siihen miten kosketusnäytöillä hover-valikot näytetään.

```
.submenu {
        display: none;
}
.submenu:hover {
        display: block;
}

/* Jos käytetään kosketusnäyttöä */
.touch .submenu {
        display: block;
}
```

Tyylimäärittelyiden lisäksi Modernizria voidaan käyttää JavaScript-tiedostoissa. Modernizrillä voidaan esimerkiksi tutkia onko selaimessa tuki kosketusnäytöille.

```
if( Modernizr.touch ) {
        // Jos käytetään kosketusnäyttöä
} else {
        // Jos kosketusnäyttö ei ole saatavilla
}
```

Modernizr tarjoaa erittäin hyvät ominaisuudet joustavalle ja tehokkaalle verkkokehitykselle eikä sillä ole varteenotettavia kilpailijoita, joten se oli erittäin hyvä kirjasto mallipohjaan.

3.2 TYYLIEN YHDENMUKAISTAMINEN

Eric Meyerin on kirjoittanut reset.css-kirjaston, jonka tarkoituksena on tasata selainten väliset oletuserot tyylien osalta. Resetissä kaikki kirjasintyypit muutetaan samankokoisiksi ja niille annetaan sama riviväli. (Meyer 2011.)

Normalize on reset.css:ää laajempi CSS-kirjasto, joka tasaa selainten tyylien välisiä eroja, mutta myös korjaa selainten bugeja, parantaa käytettävyyttä ja tyylittelee elementtejä. Normalize esimerkiksi pitää eri tason otsikot erikokoisina ja jättää listaelementeihin listamerkit. Normalize.css:n lähdekoodissa on kommentoitu jokainen kohta, joten koodin muokkaaminen on tehty helpoksi. Normalize tarjoaa kaksi eri versiota, joista vanhempi tukee vanhoja selaimia ja uudempi versio tukee uudempia(IE8+). Uudempi versio on kooltaan hieman pienempi, koska uudet selaimet ovat standardien mukaisesti lähempänä toisiaan. (Gallagher 2013.)

Mallipohjaan valittiin varmuuden vuoksi vanhempi versio normalize.css-kirjastosta, sillä kokoero uudempaan versioon ei ole suuri. Normalize nopeuttaa kehitystä Resetiin verrattuna, koska se ei poista kaikilta elementeiltä tyylejä pois.

3.3 HTML-POHJA

Mallipohjan rakentamisessa ensimmäinen askel on index-tiedoston luominen. Mallipohjan rakentamisessa lähtökohtana oli käyttää HTML5-merkkausta, joten sen semantiikkaa hyödyntäen ja tavanomaisen sivun rakenteen huomioiden, sivupohjassa tulisi olla sivun yläosalle, pääsisällölle sekä alatunnisteelle omat osionsa. Nämä osiot kuitenkin olisi kiedottu yhden elementin sisälle, joka helpottaa sivuston leveyden tyylittelemistä. Linkit tyylitiedostoihin tulisi sijoittaa sivuston head-elementin sisään. Sivuston JavaScript-tiedostot sen sijaan kannattaa sijoittaa body-elementin loppuun, jotta sivu latautuisi hakukoneiden mielestä nopeammin (Yahoo. 2013). Poikkeuksena tosin Modernizr-apukirjasto, joka sijoitetaan head-elementin sisään, jotta se voi alustaa HTML5-elementtejä käytettäväksi vanhoille selaimille (Modernizr dukumentaatio 2013).

Index-tiedoston (ks. Liite 1) rakentamisessa on otettu mallia suositun Boilerplate-mallipohjan index-tiedostosta. Sitä on kuitenkin muutettu soveltumaan paremmin toimeksiantajan tottumuksiin. Google Analytics-skripti on poistettu mallipohjasta, koska suurin osa projekteista ei käytä kyseistä palvelua. Boilerplaten index-tiedosto ei sisällä ollenkaan sivuston varsinaiseen sisältöön liittyviä elementtejä, mutta mallipohjaan lisättiin oletuksena ylätunniste, navigaatio, sisältöalue, sivupalkkialue ja alatunniste.

3.4 KIRJASINTYYPIT

Verkkosuunnittelussa kirjasintyypin (font) koon valinta jakaa mielipiteitä. Responsiivisen kehityksen myötä sivustoilla käytettävän leipätekstin keskimääräinen koko on kasvanut hieman. Constantin (2013) tutki toukokuussa 2013 suurimpien blo-

gi- ja uutissivustojen kirjainasuja. Suurin osa näistä sivustoista käytti kirjasinkokona 14px tai 16px. Luettavuuden kannalta olisi suositeltavaa käyttää kokona vähintään 16px, koska se on luettava pidemmältäkin etäisyydeltä sekä pienemmiltä näytöiltä. Lisäksi se ottaa paremmin huomioon heikkonäköisemmät käyttäjät.

Käyttäjille tulee kuitenkin tarjota mahdollisuus vaihtaa kirjasinkokoa manuaalisesti selaimen asetuksista, jolloin CSS-määrityksissä tulisi pyrkiä välttämään koon määrittämistä px-arvoilla, koska ne estävät kirjasintyypin asettamisen manuaalisesti. Vaihtoehtoisia arvoja koon määrittämiseen ovat em ja rem. Näistä arvoista em on kuitenkin tyylittelyn kannalta erittäin hankala, koska em on suhteellinen mitta ylemmän elementin kirjasintyypin kokoon. Näin ollen rem on tässä suhteessa helpompi, koska se on suhteellinen body-elementin kirjasinkokoon. Opera Minillä eikä IE8:ssa rem-arvo ei kuitenkaan toimi lainkaan. Paras kompromissi lienee olevan rem-arvon käyttö pääarvona sekä px-arvon käyttö niille selaimille, joissa rem ei toimi. Alla olevassa esimerkissä asetetaan kirjasinkooksi 2rem, joka vastaa kokoa 32px body-elementin tekstin koon ollessa 16px. Koska kaikki selaimet eivät tunnista rem-arvoa, on ensin asetettu koko näille selaimille px-arvona.

```
h1 { font-size: 32px; font-size: 2rem; }
```

Tekstin luettavuutta lisää rivivälin ja kappaleiden välin määrittely. Smashing Magazinen (2011) tekemän tutkimuksen mukaan suurimpien blogi- ja uutissivustojen riviväli oli keskimäärin 1,5 kertainen leipätekstiin verrattuna. CSS-määrittelyissä rivivälin määrittäminen em-arvoilla johtaa siihen, että riviväli ei periydy suhteellisena lapsielementeille. Riviväli periytyy lapsielementeille suhteellisena, kun sen määritellään pelkkänä desimaalilukuna (Mayer 2006). Alla on nähtävissä esimerkki rivivälin periytymisen erosta, jossa käytössä on em-arvo ja täysin suhteellinen arvo.

```
ul { font-size: 16px; line-height: 1em; }
li { font-size: 10px; } // line-height = 16px


ul { font-size: 16px; line-height: 1.0; }
li { font-size: 10px;  } // line-height = 10px
```

3.5 SAAVUTETTAVUUS

Verkkopalveluita kehittäessä on tärkeää muistaa, että verkko on suunniteltu kaikkien ulottuville laitteistosta, sijainnista tai fyysisestä rajoitteista riippumatta. Verkkopalveluita käyttävät erilaiset käyttäjäryhmät, jotka ovat joka tapauksessa palveluiden potentiaalisia käyttäjiä tai asiakkaita. Sivustojen kehitysvaiheessa kannattaa kiinnittää saavutettavuuteen erityistä huomiota, koska sen lisääminen projektin aikana tai lopussa on paljon hitaampaa ja vaikeampaa. Saavutettavuuden lisääminen tuo tuotteelle markkina-arvoa, koska kaikki sivustot eivät kiinnitä siihen erityistä huomiota. (W3C 2013.)

W3 Consortium kertoo sivustollaan muutamista hyvistä käytänteistä saavutettavuuden kannalta. Kuville tulisi asettaa alt-teksti, joka kertoo mitä kuvassa on. Tämä helpottaa sivun tulkitsemista lukulaitteilla verkkoa käyttäville ja käyttäjille, jotka ovat ottaneet kuvat pois käytöstä verkkokaistan säästämiseksi. On myös hyvä käytäntö välttää dynaamisia sisältöjä. ”Klikkaa tästä”-tyylisiä linkkejä tulisi välttää, ja linkin tekstin tulisi kuvailla selkeästi mitä linkin kohteessa on.

HTML5:ssä on käytössä paranneltu role-attribuutti. Sen tehtävä on kertoa elementin tarkoitus sivustolla, niin että sitä voitaisiin hyödyntää esimerkiksi hakukoneissa ja lukulaitteissa. Role-attribuutilla on määrätyt arvot, joten niitä ei tule itse keksiä. Hyödyllisimmät arvot ovat:

- **Main** kertoo elementin tiedon olevan sivukohtaista pääsisältöä.
- **Banner** on sivuston yläosa, jossa on yleensä logo ja sivuston nimi sijaitsevat.
- **Navigation** sisältää sivuston päänavigaation
- **Complementary** on tietoa joka liittyy pääsisältöön. Tyypillisesti tässä on kyse sivupalkista.
- **Contentinfo** on tiivistettyä tietoa koko sivustosta. Yleensä tämä tarkoittaa sivuston alapalkkia.
- **Search** kertoo elementin sisältävän sivuston hakutoiminnon.

Opinnäytetyössä otettiin role-attribuutti käyttöön. Sitä sovellettiin mallipohjaan alla esitetyllä tavalla.

```
<body>
	<header role="banner">
		<nav role="navigation"></nav>
	</header>

	<div>
		<article role="main"></article>
		<aside role="complementary"></aside>
	</div>

	<footer role="contentinfo"></footer>
</body>
```

3.6 RESPONSIIVISUUS

Sivustojen kehityksessä on hyvä suunnitella skaalautuvuus eri laitteille jo projektin alussa. Elementtien leveydet tulisi määritellä suhteellisina toisiinsa nähden. Tämä onnistuu kätevästi kaavalla:

Kohde ÷ Konteksti = Leveys

Mikäli sivun leveys on esimerkiksi 1024px ja sivupalkin leveys on 300px, voidaan sivupalkille laskea suhteellinen leveys 300px ÷ 1024px = 0.29296875, eli 29,296875%. Vaikka laskutoimitus antaa monesti pitkiä desimaalilukuja, on tärkeää olla pyöristämättä lukuja, jotta sivuston elementtien suhteellisuus pysyy eheänä. Suhteellisia mittoja voi myös lähestyä päinvastaisesta suunnasta. Mikä sivustolla on vierekkäisiä elementtejä tulee niille antaa mitat suhteellisina siten, että niiden leveys on yhteensä 100%. Esimerkiksi sivupalkin leveys olisi 30% ja sisältöalueen leveys 70%.

Responsiivisissa sivustoissa kuvat saattavat rikkoa ulkoasua. Kuvien leveys kannattaa määrittää niin, että se on aina korkeintaan vanhempielementin leveys ja kuvan korkeus on suhteellinen leveyteen nähden.

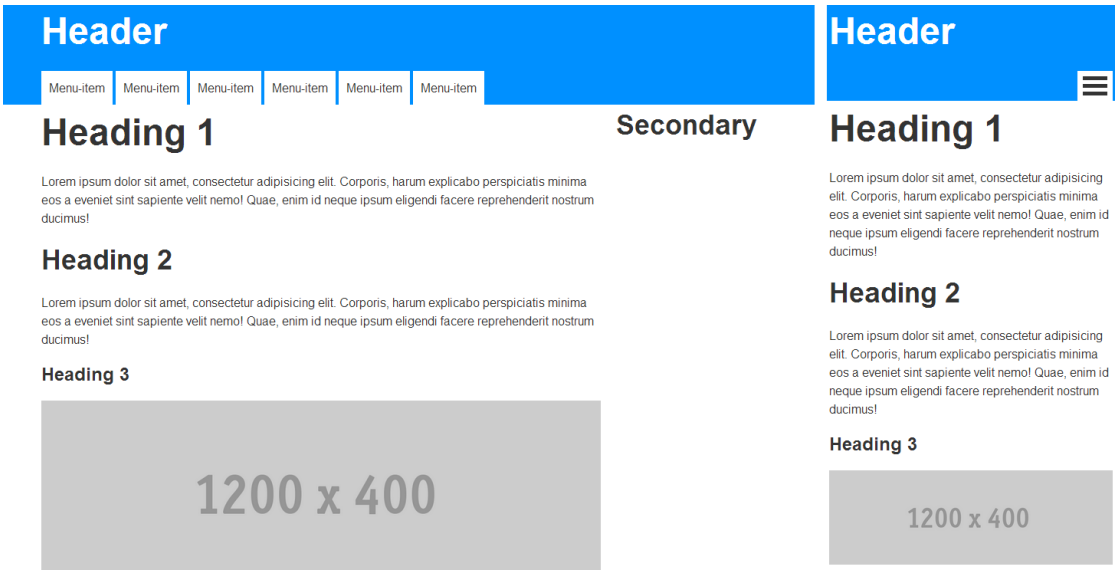
```
img {
	max-width: 100%;
	height: auto;
}
```

Kun sivua kavennetaan tarpeeksi, sivu hajoaa vaikka elementtien leveydet olisi määritelty suhteellisina. Sivun elementtien sisällöt muuttuvat tällöin epäselviksi, koska elementit leveys kapenee todella pieneksi. Tällöin vierekkäiset elementit kannattaa si-

joittaa allekkain mediakyselyiden avulla. Esimerkiksi seuraavilla tyylimäärittelyillä voidaan sivupalkki ja pääsisältöalueet sijoittaa allekkain, kun sivuston leveys on pienempi kuin 540px.

```
@media screen and (max-width: 540px) {
        .primary { float: none; width: 100%; }
        .sidebar { float: none; width: 100%; }
}
```

Mediakyselyiden maksimileveyksiä ei kannata yleensä määritellä valmiiksi, vaan ne tulee olla sivukohtaisia ja lisätä silloin kun havaitsee sivun hajoavan leveyttä skaalattaessa. Mallipohjaan lisättiin työpöytänäkymän lisäksi mobiilinäkymä, joka näytetään oletuksena selaimen leveyden ollessa pienempi kuin 62,5em. Alla olevassa kuviosta nähdään kuinka sivusto muuttuu työpöytä- ja mobiilinäkymän välillä. Kuviossa 3 olevaan sivustoon on lisätty sisältöalueeseen 1200px leveä kuva, joka skaalautuu sisältöalueeseen sopivaksi.



Kuvio 3. Mallipohjan näkymä eri selaimen leveyksillä.

3.7 RUUDUKOT

Responsiiviseen kehitykseen kuuluu tärkeänä osana ruudukot (grids), joilla voidaan määritellä nettisivun elementtien leveydet sopimaan toisiinsa. Parhaimmillaan ruudu-

kot nopeuttavat sivuston rakentamista ja helpottavat kapeampien näkymien tyylittelyä. Hyvässä ruudukkojärjestelmässä (grid system) tulisi elementtien leveydet olla määritetty prosentteina, jolloin sivun leveyden muuttaminen olisi helpompaa osien skaalautuessa suhteellisesti toisiinsa nähden. Lisäksi ruudukkojärjestelmän pitäisi antaa kehittäjälle mahdollisuus valita hajoamiskohdat.

Semantic Grid System on ruudukkojärjestämä, jonka avulla HTML-merkkaus voidaan pitää puhtaana, koska erillisiä ulkoasuun liittyviä luokkia ei tarvitse lisätä. Tyylitiedostossa voidaan määritellä sivustolla käytettävän kehikon koko ja yhden kolumnin leveys. Alla esimerkki 12-kolumnisen sivuston ruudukkomäärittelyistä, jossa sivuston ylätunnisteen leveys on yhtä leveä kuin koko sivuston leveys, sisältö alueen leveys on 9/12 ja sivupalkin leveys on 3/12 sivuston leveydestä.

```
@column-width: 60;
@gutter-width: 20;
@columns: 12;

header { .column(12); }
article { .column(9); }
aside { .column(3); }
```

Semantic Grid System ei kuitenkaan toimi CSS-kielessä suoraan, vaan vaati LESS-, SCSS-, tai Stylus-kielen toimiakseen. Semantic Grid System vaikutti erittäin hyvältä tavalta lähestyä ruudukoiden rakentamista sivustolle. Koska mallipohjassa käytettiin LESS-kieltä, oli tämän ruudukkojärjestelmän käyttäminen tyylimäärittelyissä (ks. Liite 2) erinomainen valinta.

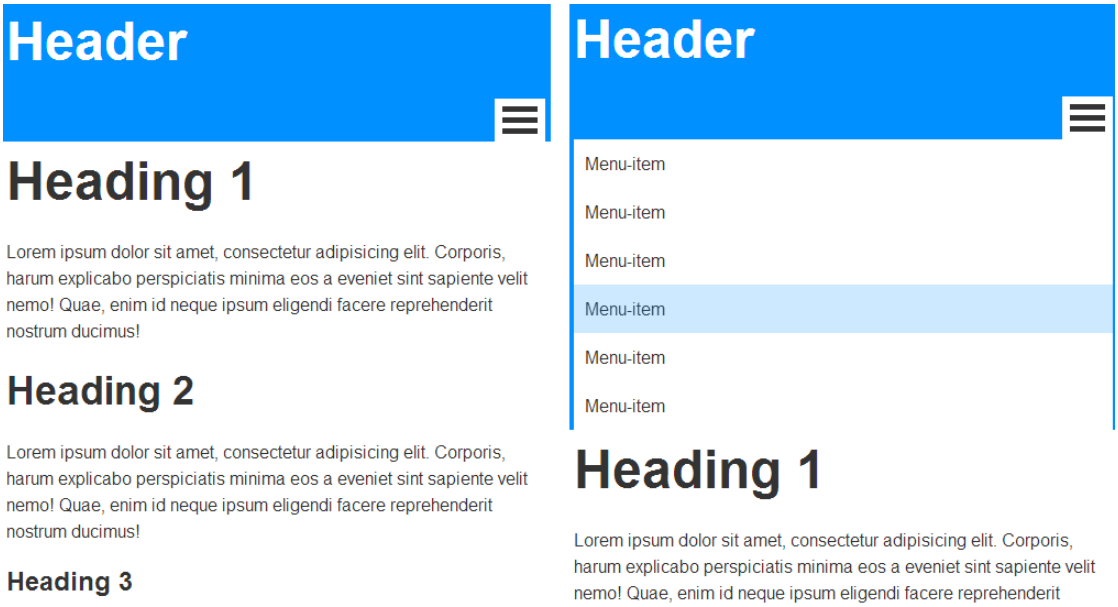
3.8 RESPONSIIVINEN NAVIGAATIO

Isoilla näytöillä navigaatio on lähes poikkeuksetta sivun ylälaidassa tai vasemmassa laidassa. Mobiililaitteille sivustoa suunniteltaessa navigaatioon täytyy panostaa, jotta käyttäjä saa selkeän kuvan sivustosta ja sen rakenteesta. Ahtaassa näkymässä käyttäjä ei saa sivustosta kokonaiskuvaa yhdellä vilkaisulla, vaan käyttäjän pitää selata sivua hetki ennen kuin kuva sivuston rakenteesta muodostuu.

Brad Frost (2012) on blogissaan vertaillut erilaisia ratkaisuja responsiiviselle navigaatiolle. Useimmat vaihtoehdot tarvitsevat JavaScriptin toimiakseen, mutta se tuskin on suuri ongelma mobiililaitteilla, varsinkin kun muistaa tehdä ilman JavaScriptiä toimivalle sivulle tuen esimerkiksi Modernizrin avulla. Useimmissa tapauksissa, kun sivuston rakenne ei ole kovin syvä, voidaan valikko näyttää normaalisti sivun ylälaidassa listauksena tai piilottaa valikko painikkeen taakse, jolloin säästyy paljon tilaa. Navigaation rakentaminen muuttuu kuitenkin nopeasti hankalaksi sivuston rakenteen syventyessä. On erittäin haasteellista rakentaa navigaatio sivustolle, kun sen sivurakenne on kolmetasoinen tai syvempi.

Responsiivisten navigaatioiden lisäosien valikoima on melko suppea. Viljami Salminen on kuitenkin kehittänyt lisäosan nimeltä Responsive Nav. Lisäosa lisää sivustolle valikkopainikkeen mobiilinäkymiin. Responsive Nav on todella hyvin kehitetty lisäosa, sillä se on erittäin pieni kooltaan eikä vaadi jQuerya toimiakseen. Lisäksi se lisää elementteihin luokkia, niin että omien animaatioiden tekeminen CSS:n avulla onnistuu. Responsive Nav poistaa kosketusnäytön painalluksen ja JavaScriptin klikkaustapahtuman välisen viiveen. Responsive Nav toimii mobile first -periaatteella, mutta koska CSS-tiedosto on todella pieni, on lisäosa helppo räätälöidä omaan projektiin sopivaksi. Opinnäytetyössä lisäosan tyylit siirrettiin täysin sivuston yleiseen tyylitiedostoon. Responsive Nav lisättiin mallipohjaan, koska se toimisi todennäköisesti useimmissa projekteissa loistavasti. Isompien sivustojen kohdalla tulee kuitenkin tarkastella ongelmaa laajemmin ja valita kyseiseen projektiin sopiva ratkaisu.

Oletuksena navigaatio näytetään mallipohjan ylälaidassa. Mobiilinäkymässä valikko kuitenkin piilotetaan ja sivuston ylälaitaan lisätään painike, josta navigaatio voidaan avata ja sulkea. Alla olevassa kuvassa 4 on havainnollistettu mallipohjan mobiilinavigaatiota.



Kuvio 4. Responsive Navin käyttö mallipohjassa.

3.9 KUVAGALLERIAT

Verkkosivustojen tärkein osa on sen sisältö, joten on hyvä miettiä projektin alusta lähtien kuinka sen esittämistä voidaan tukea. Keskeisenä osana sivustoja, tekstien ohella, ovat kuvat. Monesti kuvat kuitenkin esitetään sivustoilla melko pienenä, joten kannattaa varmistaa, että niille pystytään tarvittaessa luomaan koko näytön levyinen katselukehys. Mallipohjaa varten tarkasteltiin kolmea yleisintä kirjastoa, jotka mahdollistavat kuvien katselemisen isompana: *Lightbox 2, ColorBox ja Magnific Popup*. Ennen testaamista lisäosille mietittiin taulukon 1 mukaisesti haluttuja ominaisuuksia ja niiden tarpeellisuutta.

Taulukko 1. Gallerialisäosien vertailukriteerien painoarvot

Osa-alue	Painoarvo
Lisäosan koko	20 %
Responsiivisuus	30 %
Tuki kosketusnäytöille	10 %
Toimivuus ilman JavaScriptiä	10 %
Muokattavuus	30 %

Lightbox 2 -galleria oli kooltaan vertailuryhmän kevyin 11 kt koollaan. Se ei kuitenkaan toiminut pienemmillä näytön leveyksillä ja lisäosa rikkoutui oletuksena mallipohjan tyyleistä. Lightbox 2 toimi oletetulla tavalla ilman JavaScriptiä. Lisäosan muokkaaminen oli hankalaa, sillä sille ei löytynyt lainkaan dokumentaatiota.

Colorbox oli kooltaan hieman Lightbox 2:ta isompi. Colorbox oli hyvin dokumentoitu ja toimi oletetulla tavalla ilman JavaScriptiä. Lisäosa ei ollut kuitenkaan oletuksena responsiivinen vaan vaati kehittäjältä lisäosan laajentamista.

Magnific Popup oli kooltaan n. 28 kt, joka oli lähes kaksin kertainen muihin vertailuryhmän lisäosiin. Muista lisäosista poiketen Magnific Popup toimi oletuksena pienemmillä näytön leveyksillä erinomaisesti. Magnific Popupin dokumentaatio oli myös erittäin kattava ja sen avulla pystyi näyttämään myös muuta sisältöä kuin kuvia popup-ikkunassa.

Taulukko 2. Gallerialisäosien vertailukriteerien pisteytys

	Lightbox 2	**Colorbox**	**Magnific Popup**
Lisäosan koko	5	4	3
Responsiivisuus	2	2	5
Tuki kosketusnäytöille	5	5	5
Toimivuus ilman JavaScriptiä	5	5	5
Muokattavuus	1	5	5
Yhteensä	2,6	3,9	4,6

Mallipohjaan valittiin Magnific Popup, sillä se oli vertailuryhmän ainoa lisäosa, joka oli responsiivinen. Erittäin kattavan dokumentaation ansiosta se oli myös muokattavissa niin toiminnallisuudeltaan kuin tyyleiltäänkin. Vaikka Magnific Popup oli muihin nähden kooltaan melko iso, se ei kuitenkaan noussut vertailussa kynnyskysymykseksi.

3.10 KUVAKARUSELLIT

Useiden käytettävyystutkimuksien mukaan kuvakarusellit ovat erittäin epätehokkaita esittelemään sivuston sisältöä. Jakob Nielsenin tutkimusryhmä (Nielsen 2013) puhuu termistä bannerisokeus, joka tarkoittaa netin käyttäjien tiedostamatonta sisällön suo-

datusta. Käyttäjät ovat alkaneet suodattamaan alitajuisesti sivustoilta mainoksia, eivätkä reagoi niihin vaikka ne olisivat mielenkiintoisen näköisiä ja sijoitettu tärkeimmille paikoille. Bannerisokeuden myötä myös kuvakaruselleihin suhtaudutaan samalla lailla kuten mainoksiin. Käytettävyystutkimuksien mukaan käyttäjät eivät esimerkiksi osanneet sanoa, onko sivustolla pesukoneille mitään alennusta, vaikka kyseinen tieto oli sijoitettu etusivun yläreunassa olevaan kuvakaruselliin ensimmäiselle kuvalle. Huonosta käytettävyydestään huolimatta sivustoille halutaan varsin usein kuvakaruselli tuomaan eloa ja modernia ulkoasua, joten on hyvä varautua toteuttamaan tämä ominaisuus.

Opinnäytetyössä vertailtiin neljää kuvakarusellilisäosaa, jotka olivat kaikki rakennettu jQuery-kirjaston varaan. Kaikissa lisäosissa oli laaja valikoima asetuksia sekä selkeä dokumentaatio. Lisäksi kaikissa lisäosissa oli ainakin fade- ja slide-efektit, jotka tuntuivat toimivan erilaisista efekteistä parhaiten mobiililaitteilla.

Taulukko 3. Kuvakarusellien vertailukriteerien painoarvot

Osa-alue	**Painoarvo**
Lisäosan koko	25 %
Responsiivisuus	25 %
Tuki kosketusnäytöille	10 %
Toimivuus ilman JavaScriptiä	10 %
Muokattavuus	30 %

SlidesJS oli vertailuryhmän kevyin lisäosa, sillä se ei sisältänyt kuvia eikä oletustyylejä. Pienestä koostaan huolimatta SlidesJS osoittautui yllättävän joustavaksi, koska se toimi responsiivisissa sivustoissa ja siinä oli tuki kosketusnäyttöjen pyyhkäisyyn. SlideJS-karuselli pyöritti ainoastaan kuvia, joten tekstin lisääminen ei onnistunut. Lisäksi JavaScriptin ollessa pois päältä lisäosa ei näyttänyt edes ensimmäistä kuvaa. Nämä heikkoudet saattavat olla kuitenkin korjattavissa. SlideJS sopii sivustoihin joihin halutaan kuvakaruselli, mutta joiden halutaan myös olevan erittäin kevyitä.

BxSlider oli vastaavasti kooltaan kolminkertainen verrattuna SlideJS-lisäosaan. Siinä oli kuitenkin mahdollisuus lisätä kuvien lisäksi tekstiä karuselliin. BxSliderissa oli val-

miit oletustyylit, tuki responsiivisille sivustoille sekä kosketusnäytön pyyhkäisylle. Ilman JavaScriptiä BxSlider näytti kaikki kuvat allekkain oletuksena.

Nivo Slider erottui muista lisäosista erilaisilla siirtymäefekteillään, jotka olivat kuitenkin raskaita eivätkä toimineet mobiililaitteilla halutulla tavalla. Lisäosa toimi responsiivisissa sivustoissa, mutta siinä ei ollut tukea kosketusnäyttöjen pyyhkäisylle. Efekteistä huolimatta lisäosa oli kooltaan keskiluokkaa ja siinä oli valmiina oletusteema. Nivo Slider tarjoaa efekteillään vaihtelua fade- ja slide-efekteille mikäli sivuston ei tarvitse toimia mobiililaitteilla parhaalla mahdollisella tavalla.

FlexSlider oli vertailuryhmästä ainoa, josta ei löytynyt juurikaan huonoja puolia. Se toimi responsiivisissa sivustoissa, siinä oli tuki kosketusnäyttöjen pyyhkäisylle. FlexSliderissa oli oletusteema, jota oli kuitenkin helppoa muokata sivustoon sopivammaksi. Myös muista vertailuryhmän lisäosista poiketen FlexSliderin karusellin näkymiä pystyi vaihtamaan nuolinäppäimillä.

Taulukko 4. Kuvakarusellien vertailukriteerien pisteytys

		SlidesJS	**BxSlider**	**Nivo Slider**	**FlexSlider**
Lisäosan koko	25 %	5	3	3	3
Responsiivisuus	25 %	5	5	2	5
Tuki kosketusnäytöille	10 %	5	5	0	5
Toimivuus ilman JavaScriptiä	10 %	1	1	2	4
Muokattavuus	30 %	2	4	4	4
Yhteensä		**3,7**	**3,8**	**2,65**	**4,1**

Mallipohjaan valittiin FlexSlider, koska se toimi vertailuryhmästä parhaiten mobiililaitteissa. FlexSliderista on kaksi versiota. Ilmaisen kehittäjäversion lisäksi lisäosasta on olemassa WordPress-lisäosa, jonka hinta on 49-149 dollaria riippuen siitä kuinka monelle sivustolle lisenssi halutaan. Ilmaisesta kehittäjäversiosta voi kuitenkin tehdä oman lisäosan WordPressiin GPLv2-linsenssin turvin.

3.11 LÖYDETTÄVYYS HAKUKONEISSA

Hakukoneilla on kaksi keskeistä tehtävää — etsiä sivuja ja indeksoida niitä. Hakukoneiden robotit käyvät sivustoja läpi. Käyttäjän hakiessa tietoa hakukoneella hakukone listaa sivuja kahden perusajatuksen mukaan; sivuston sisällön yhtäläisyyden haettuun hakuehtoon nähden sekä sivuston hyödyllisyyden hakijalle. Hakukoneet tulkitsevat hyödyllisyyden muiden käyttäjien perusteella. Mikäli sivustolla on paljon käyttäjiä ja käyttäjät viettävät niillä paljon aikaa, tulkitaan sivuston sisältö hyödylliseksi. (Moz. 2013.)

Google kehottaa rakentamaan sivustot käyttäjiä varten, eikä hakukoneiden robotteja varten. Hakukoneet saattavat tarkoituksella heikentää sivuston löydettävyyttä mikäli sivusto yrittää tarjota erilaista sisältöä hakukoneille kuin käyttäjille. Sivuston ei tulisi myöskään hokea yhtä haluttua hakutermiä sisällössään, vaan mieluummin tarjota rikkaampaa sisältöä ja käyttää erilaisia aihealueeseen liittyviä termejä. Sivustojen tulisi tarjota hyvä käytettävyys ja mielenkiintoista sisältöä käyttäjille. Mitä mielenkiintoisempaa ja rikkaampaa sisältö on sitä paremmin hakukoneet osaavat eritellä sivun sisältöä ja sitä paremmin käyttäjät viihtyvät sivustolla. (Google 2011.)

Vaikka hyvä sisältö on tärkein kriteeri sivuston löydettävyydessä, voidaan kehitysvaiheessa vaikuttaa löydettävyyteen tekemällä sivusto selkeämmäksi hakukoneille. Sivuston sisällön tulisi olla HTML-muotoisena. Mikäli sivustolla on sisältöä, joka vaatii JavaScriptiä toimiakseen, eivät hakukoneiden robotit välttämättä löydä sitä. Hakukoneet eivät myöskään ymmärrä kuvissa olevaa sisältöä, ellei sitä ole lisätty alt-attribuuttiin. Hakukoneet eivät pääse Flash- ja Java-sovellusten tai videoiden ja äänitiedostojen sisältöön käsiksi.

Hakukoneiden täytyy pystyä rakentamaan sivuston eri sivuista kokonaiskuva. Hakukoneet etsivät etusivulta kaikki sivun sisäiset linkit ja edelleen näiltä sivuilta sisäiset linkit. Kaikkiin sivuston sivuihin täytyy siis päästä käsiksi selkeästi HTML:ssä olevien linkkien kautta.

Sivuston title-elementti on keskeisessä roolissa käytettävyyden ja löydettävyyden kannalta. Sen tulisi esittää sivun sisältö hyvin tarkasti ja sisältää haluttuja avainsanoja.

Mitä lähempänä avainsanat ovat otsikon alkua, sitä tärkeämpinä hakukoneet ne kokevat. On myös hyvä muistaa, että otsikkoon mahtuu ainoastaan 65-75 merkkiä. Alasivun title-elementti olisi hyvä olla esimerkiksi seuraavanlainen.

```
<title>Alasivu | Domain</title>
```

Description meta-elementti kuvaa sivun sisältöä. Hakukoneet käyttävät tätä hakukoneen listauksessa, mutta eivät indeksoinnissa. Google näyttää listauksessaan kuvauksesta 160 merkkiä.

```
<meta description="Kuvaus sivuston sisällöstä">
```

Sivuston osoitteet tulisi pitää mahdollisimman ymmärrettävinä ihmisille, niin että jo sivuston url-osoitteesta näkee mitä sisältöä sivulla on. Selkokieliset URL-osoitteet parantavat sivuston käytettävyyttä ja lisäävät luotettavuutta. (Moz 2013.)

Rich snippets -attribuuttien avulla voidaan kuvailla sivuston sisältöä paremmin hakukoneille. Esimerkiksi tuotteen voi esittää seuraavasti:

```
<div itemscope itemtype="http://schema.org/Product">
        <h1 itemprop="name">Tuotteen Nimi</h1>
        <p itemprop="description">Tuotteen kuvaus</p>
        <span itemprop="weight">3 kg</span>
</div>
```

Itemscope- ja itemtype-attribuutit kertovat, että kyseessä on tuote. Itemprop-attribuutilla voidaan täsmentää mitä tuotteen tietoa elementti kuvastaa. Hakukoneet ymmärtävät tämän perusteella paremmin että kyseinen sisältö liittyy tuotteeseen ja sivusto näyttäytyy hakukoneissa tehokkaammin. (Schema 2011.)

4 TEKNIIKOITA PROJEKTIN TUEKSI

4.1 DYNAAMISET TYYLIKIELET

Dynaamiset tyylikielet ovat skriptikieliä, jotka voidaan kääntää CSS-tyylimäärittelyiksi. Näiden kielien avulla voidaan tyylimäärittelyiden koostamisesta tehdä helpompaa ja joustavampaa ohjelmoijan näkökulmasta, sillä ne sisältävät muuttujia, operaatioita sekä funktioita. Dynaamiset tyylikielet ovat kuitenkin käännettävä CSS-määritteiksi. Opinnäytetyön kirjoitushetkellä on olemassa kaksi suurta yhteisön suosimaa dynaamista tyylikieltä LESS ja Sass. Dynaamisilla tyylikielillä on myös omat heikkoutensa. Julkaisun jälkeen sivustolle on hieman työläämpi tehdä pieniä muutoksia puhtaaseen CSS-kieleen verrattuna. LESS-kielen käyttö vaatii myös kaikkien sivuston kehitykseen osallistuvien omaksuvan tekniikan.

Sass on vuonna 2007 kehitetty skriptikieli, joka voidaan kääntää Rubyn avulla CSS-tyylimäärittelyiksi. Sass-skriptikielessä ei ole alun perin aaltosulkuja, mutta myöhemmin on julkaistu SCSS (Sassy CSS), joka saanut vaikutteensa LESS -kielestä ja muistuttaa syntaksiltaan enemmän CSS-tyylimäärittelyä. (Sass 2013.)

LESS on vuonna 2009 kehitetty skriptikieli, joka on saanut vaikutteensa Sass-kielestä. LESS kuitenkin pyrki olemaan lähempänä CSS:n alkuperäistä syntaksia. LESS on alun perin kehitetty Rubylla, mutta kehityskieleksi on myöhemmin vaihdettu JavaScript. LESS-tiedostot voidaan kääntää joko komentorivillä, kääntämiseen tarkoitetulla ohjelmalla tai sivustolle ladattavalla JavaScript-kirjastolla. LESS-kielen kääntäminen ei ole suositeltavaa selaimella kuin kehitysvaiheessa, sillä se hidastaa turhaan sivun lataamista. (Sellier 2013.)

Tietokoneelle asennettavan kääntäjän avulla LESS-tiedostojen kääntäminen CSS-tiedostoiksi onnistuu kahdella tavalla. Joko asennetaan erillinen ohjelma kuten SimpLESS, joka pitää huolta valittujen LESS-tiedostojen kääntämisestä ohjelman ollessa käynnissä (Kiss 2011) tai vaihtoehtoisesti asennetaan LESS-kääntäjä koneelle node-pakettihallinnan kautta ja tämän jälkeen asennetaan editoriin LESS-kääntäjää hyödyn-

tävä lisäosa, joka kääntää tallennuksen yhteydessä LESS-tiedostot automaattisesti samannimisiksi CSS-tiedostoiksi.

Opinnäytetyön mallipohjaan valittiin LESS. Koska se on syntaksiltaan hyvin lähellä CSS-kieltä, on se helppo omaksua. LESS-kielen kääntämiseen käytettiin SublimeText-editoriin asennettavaa lisäosaa, joka kääntää LESS-tiedostot automaattisesti tallennuksen yhteydessä.

Mallipohjaan tehtiin LESS-kirjasto (ks. Liite 3), joka lisää kasan funktioita, joiden ansiosta useiden uusien tyylimäärittelyjen lisääminen on selkeämpää. LESS-kirjasto sisältää Chris Coyerin (2011) julkaisemassa artikkelissa olevat hyödyllisimmät funktiot. Kirjastoa on laajennettu lisäksi muutamalla omalla funktiolla.

CSS-kielessä tulee yleensä paljon toistoa sen syntaksin luonteen takia. Esimerkiksi mikäli elementille halutaan asettaa tyylimuutoksiin transitio, kirjoitetaan puhtaalla CSS-kielellä:

```
.selector {
	-webkit-transition: all 0.2s;
	-moz-transition: all 0.2s;
	-ms-transition: all 0.2s;
	-o-transition: all 0.2s;
}
```

Funktioiden ansiosta koodista voidaan tehdä huomattavasti modulaarisempaa luomalla funktio, jota voidaan käyttää usealla elementillä vaikka niiden arvot olisivat toisistaan poikkeavat. Alla esitetty transition asettaminen funktion avulla.

```
.selector {
	.transition( all 0.2s );
}

.transition (@transition) {
	-webkit-transition: @transition;
	-moz-transition:    @transition;
	-ms-transition:     @transition;
	-o-transition:      @transition;
}
```

4.2 COFFEESCRIPT

CoffeeScript on kieli, joka voidaan kääntää JavaScriptiksi. CoffeeScriptiä voidaan käyttää minkä tahansa JavaScript-kirjaston kanssa, koska CoffeeScript kääntyy puhtaaksi JavaScriptiksi. CoffeeScript pyrkii tekemään JavaScriptin kirjoittamisesta selkeämpää ja nopeampaa. Syntaksiltaan se muistuttaakin huomattavasti Rubya ja Pythonia. CoffeeScript ei sisällä rivin lopetukseen puolipisteitä eikä rajaa funktioiden ja silmukoiden aluetta aaltosuluilla. CoffeeScriptiin on todennäköisesti vaikea päästä sisälle, jos kehittäjä on tottunut työskentelemään pääasiassa syntaksiltaan JavaScriptin kaltaisilla kielillä. Toisaalta Pythoniin tai Rubyyn tottuneille voi CoffeeScript tuntua erittäin luontevalta vaihtoehdolta. (Ashkenas 2013.)

Mikäli projektin kehityksessä ei tarvitse ohjelmoida paljon JavaScriptillä, on CoffeeScriptin käyttö todennäköisesti hidaste. CoffeeScript-tiedostot joudutaan kääntämään joka kerta kun niitä päivitetään, joka voidaan toki automatisoida sopivilla ympäristöllä, mutta pienten muutosten tekeminen voi tuntua hyvin hitaalta. CoffeeScriptin käyttö myös vaatii kaikilta projektin kehittäjiltä sekä jatkokehittäjiltä sen omaksumista. CoffeeScriptin merkittävä heikkous on sen virheenjäljitys, koska CoffeeScript käännetään JavaScriptiksi, ei voida olla varmoja mihin CoffeeScript-tiedoston riviin JavaScriptin virhe viittaa.

Opinnäytetyön mallipohjaan CoffeeScript vaikutti enemmän projektia hidastavalta kuin nopeuttavalta tekniikalta, joten sitä ei otettu käyttöön. Mallipohja pyrkii nopeuttamaan nettisivujen kehitystä, joissa on tyypillisesti vain vähän JavaScriptiä. Lisäksi toimeksiantaja on tottunut ohjelmoimaan JavaScriptin kaltaisilla kielillä, joten CoffeeScriptin syntaksi vaatisi paljon opettelua. CoffeeScript on kuitenkin erittäin mielenkiintoinen kieli, jota kannatta harkita suurempien JavaScript-projektien kehityksessä.

4.3 LIVERELOAD

Sivujen taittaminen on usein melko suoraviivainen prosessi – ensin tehdään muutoksia PHP/HTML/CSS-tiedostoihin ja tarkistetaan selaimella, että kaikki näyttää kuten pi-

tääkin. Editorin ja selaimen näkymien vuorottelu ja selaimen päivittäminen vievät kuitenkin huomaamatta paljon aikaa. LiveReload on tietokoneelle asennettava sovellus, jonka ansiosta sivun näkymää ei tarvitse päivittää kesken kehityksen. Aina kun jokin projektin tiedostoista päivittyy, selain päivittää sivun näkymän automaattisesti. Vaikka sivun näkymän päivitys ei manuaalisestikaan vie paljon aikaa, LiveReload kuitenkin nopeuttaa työskentelyä huomattavasti. Mikäli kehittäjällä on käytössä kaksi näyttöä, joista toiseen asetetaan kehitysympäristö ja toiseen selain on LiveReloadin käyttö erityisen mukavaa, sillä sivuston muutokset näkee lähes reaaliajassa viereiseltä näytöltä, jossa selain on auki.

4.4 VERSIONHALLINTA

Verkkokehityksessä on hyvä olla versionhallintajärjestelmä käytössä. Versionhallintajärjestelmän avulla voidaan pitää yllä tietoa projektin tiedostojen muutoksesta. Suurin hyöty saadaan, kun projektissa on useita kehittäjiä, koska he voivat muokata samoja tiedostoja yhtäaikaisesti ilman konflikteja. Versionhallinnasta on myös hyötyä, vaikka projektissa olisi vain yksi kehittäjä. Toisinaan projekteissa tarvitsee tietää miltä projekti näytti aikaisemmassa vaiheessa tai halutaan kokeilla ominaisuutta, josta ei kuitenkaan olla varma sisällytetäänkö se lopulliseen versioon. Näissä tilanteissa versionhallintajärjestelmä osoittautuu korvaamattomaksi. Hyviä vaihtoehtoja version hallintaan ovat Git ja SVN. Mikäli projektin lähdekoodia ei tarvitse pitää salaisena, voidaan projektin versiohallintaa ylläpitää Github-palvelussa.

4.5 WAMPSERVER

WampServer on Windows-ympäristöön asennettava avoimen lähdekoodin ohjelmapaketti, jotka yhdessä muodostavat WWW-palvelimen, johon kuuluu Apache-palvelin, PHP-ympäristö sekä MySQL-rajapinta. WampServerin tarkoitus on muuttaa kone lokaaliksi WWW-palvelimeksi, jolle voidaan kehittää verkkopalveluita ilman tiedostoja tarvitsee ladata erilliselle palvelimelle. Sivustoa voidaan kehittää hiekkalaatikossa, jonne ei voida ottaa yhteyttä muilta koneilta. WampServerin mukana tulee

myös phpMyAdmin-työkalu, joka mahdollistaa MySQL-tietokantojen hallinnan graafisen käyttöliittymän avulla.

4.6 EMMET

Emmet on editoriin asennettava lisäosa, joka mahdollistaa HTML:n ja CSS:n nopean kirjoittamisen. Emmet on aikaisemmin tunnettu nimellä Zen Coding. Emmetin kehittäjät ja yhteisö ovat kehittäneet lisäosan usealle eri tekstieditorille (Emmet Download 2013). Emmet on saatavana mm. seuraaville tekstieditoreille:

- Sublime Text
- Eclipse/Aptana
- TextMate
- Coda
- Notepad++
- Selaimen Textarea
- Adobe Dreamweaver
- TinyMCE

Emmet tarjoaa nopean tavan kirjoittaa esimerkiksi HTML-elementtejä kirjoittamalla elementin nimen ja valittua näppäintä painamalla editori generoi kirjoitetun elementin kokonaisuudessaan ja jättää valitsimen elementin sisälle. Elementtejä voidaan myös yhdistellä, jolloin voidaan luoda melko vaikeitakin rakenteita suhteellisen lyhyillä komennoilla. Alla olevassa taulukossa 5 on muutamia yleisimpiä Emmetin HTML-lyhenteitä.

Taulukko 5. Esimerkkejä Emmetin käytöstä HTML-tiedostossa (Emmet Cheat Sheet)

Lyhenne	**Tulostus**
nav.menu>ul>li	<nav class="menu"> </nav>
ul>li+li	
ul>li*3	
h1#site-name{Sivun Nimi}	<h1 id="site-name">Sivun Nimi</h1>

Lyhenteiden avulla voidaan myös kirjoittaa CSS-tyylejä. Etenkin uusien CSS-määritteiden kirjoittaminen on helppoa, sillä Emmet generoi kaikki tarvittavat selainten etuliitteet. Alla olevassa taulukossa 6 on esitelty yleisimpiä Emmetin CSS-lyhenteitä.

Taulukko 6. Esimerkkejä Emmetin käytöstä CSS-tiedostossa (Emmet Cheat Sheet)

Lyhenne	**Tulostus**
bg	Background: ;
d	display: block;
fr	float: right;
w100	width: 100px;
bdrs5	-webkit-border-radius: 5px; -moz-border-radius: 5px; border-radius: 5px;
bxsh	-webkit-box-shadow: inset hoff voff blur color; -moz-box-shadow: inset hoff voff blur color; box-shadow: inset hoff voff blur color;
trf:r	-webkit-transform: rotate(angle); -moz-transform: rotate(angle); -ms-transform: rotate(angle); -o-transform: rotate(angle); transform: rotate(angle);

4.7 KUVIEN OPTIMOINTI

PNG-formaatti on erittäin hyvä verkkokehityksessä, koska se on häviötön ja mahdollistaa läpinäkyvyyden. Lisäksi PNG-kuvat eivät ole kuitenkaan kooltaan kovin isoja. PNG sopii parhaiten kuviin, jotka ovat yksinkertaisia tai joissa on iso kontrasti värien välillä. Isommissa valokuvamaisissa kuvissa on kuitenkin useimmiten parempi käyttää JPG-formaattia.

Kuvankäsittelyohjelmilla suunniteltu grafiikka ei ole usein hyvin pakattu PNG-muotoon talletettuna. Syytä ei tunnu löytyvän siihen miksi maksulliset kuvankäsittelyohjelmat eivät pakkaa PNG-kuvia pienempään muotoon. Kyse on kuitenkin häviöttömästä kuvaformaatista, jossa tiedon määrä ja kuvanlaatu eivät heikkene pakkauksen myötä. On kuitenkin olemassa useita sovelluksia, joilla voidaan verkkoprojektissa optimoida sivun latausta pakkaamalla PNG-kuvat. Kuvia pakkaamalla niiden tiedostokoko voi pienentyä lähes 90%. (Nilsson 2013.)

PngOptimizer on koneelle ladattava ohjelma, johon voidaan raahata halutut kuvat optimointia varten. Ohjelma optimoi kuvat automaattisesti ja tallentaa niistä myös alkuperäiset versiot vahinkojen varalta. (PngOptimizer 2013.)

TinyPNG on verkkosivulla toimiva palvelu, johon voi raahata halutut kuvat työpöydältä. Palvelu optimoi kuvat ja ne voi ladata sivustolta omaan käyttöön. TinyPNG on ilmainen ja toimii kaikilla uusimmilla selaimilla. (Voormedia 2013.)

Mallipohjan tueksi otettiin käyttöön PngOptimizer ja sitä tullaan käyttämään jokaisessa projektissa testaus- ja optimointivaiheen aikana pienentämään sivuston kokoa.

5 MALLIPOHJAN TESTAAMINEN

5.1 YLEISTÄ

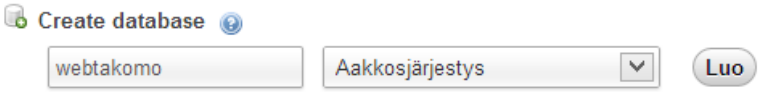
Opinnäytetyön aikana mallipohjaa ja projektia tukevia tekniikoita testattiin ja kehitettiin kuuden verkkosivun kehityksen aikana. Opinnäytetyössä esiteltiin projektin kulkua tekemällä mallipohjan WordPress-versiolla testiverkkosivusto, joka pyrkii testaamaan onko käytössä olevista tekniikoista selvää hyötyä kehityksen kannalta. Opinnäytetyössä käydään myös kehitystä tukevien tekniikoiden asentaminen läpi.

5.2 WAMPIN ASENNUS

Projekti käynnistettiin WampServerin asennuksella, jotta projektin kehitysvaiheessa ei tarvitsisi siirtää tiedostoja verkkoon. WampServer-ohjelmiston asennus oli erittäin yksinkertainen ja nopea. Asennuksen mukana tuli phpMyAdmin-työkalu, jolla voitiin luoda palvelimelle tietokantoja.

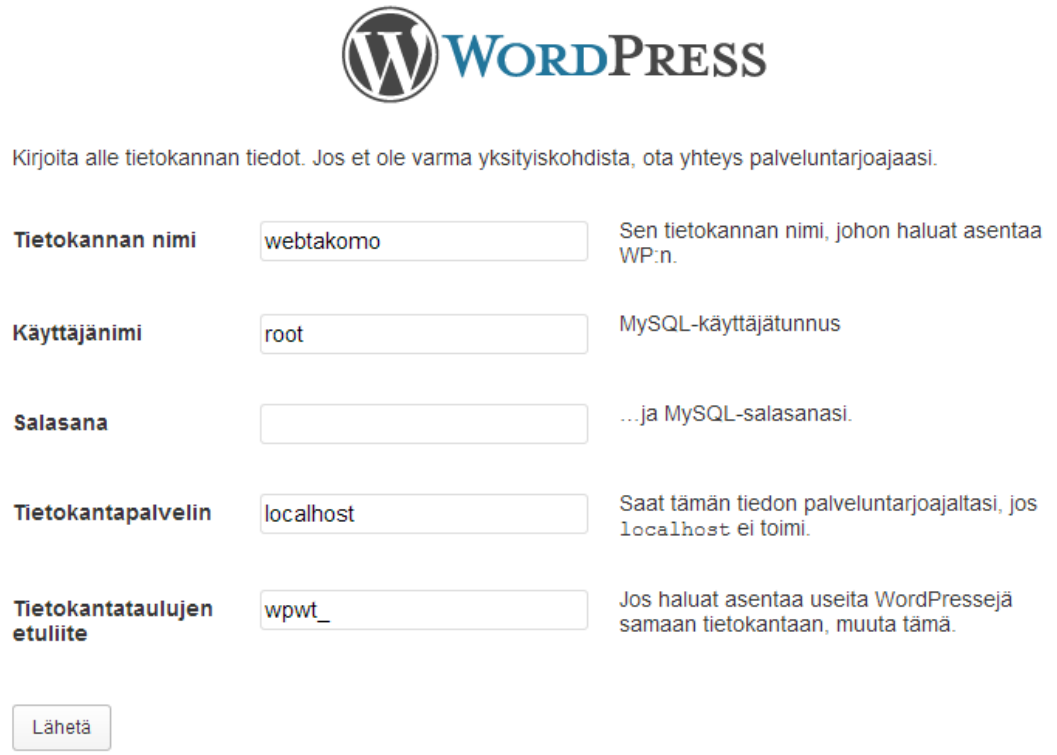
5.3 WORDPRESSIN ASENNUS

Ennen WordPressin asennusta luotiin sivustolle tietokanta WampServerin phpMyAdmin-työkalulla kuvion 5 mukaisesti. Uuden tietokannan luominen oli erittäin yksinkertaista työkalun avulla, sillä sille tarvitsee ainoastaan antaa nimi. Oletuksena Wamp tietokannan tunnus on *root* ja salasana on tyhjä kenttä. Näitä ei tarvitse vaihtaa turvallisuussyistä mikäli WampServeriä käytetään ainoastaan lokaalisti.



Kuvio 5. Tietokannan luominen phpMyAdmin-työkalulla

Wordpressin uusin suomenkielinen asennuspaketti ladattiin osoitteesta http://fi.wordpress.org/. Latauspaketti purettiin Wampin www-kansioon ja asennus suoritettiin selaimessa localhost-osoitteessa http://localhost/webtakomo/. Kuviossa 6 on esitetty WordPressin asennus palvelimelle, joka onnistui hyvin suoraviivaisesti syöttämällä tietokannan tiedot. Asennus ehdotti oletuksena tietokannan etuliitteeksi *wp_,* mutta se muutettiin turvallisuussyistä muuksi, koska tietokanta tullaan mahdollisesti siirtämään myöhemmässä vaiheessa verkkoon.



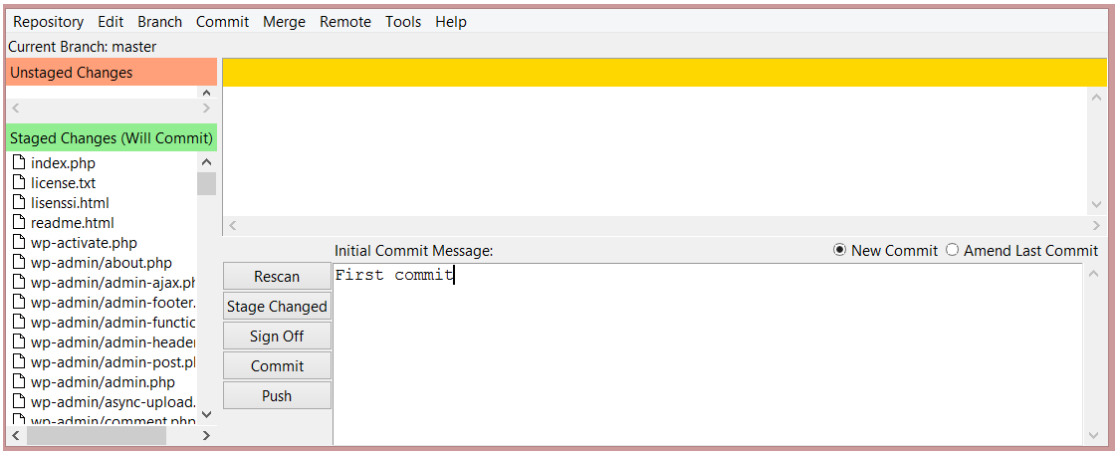
Kuvio 6. WordPressin asennus palvelimelle

Tietokannan asetuksien jälkeen asennuksessa pyydettiin sivuston nimeä ja admin-tunnuksia. Oletuksena admin-tunnuksen käyttäjätunnus on *admin*, joka myös vaihdettiin muuhun turvallisuuden parantamiseksi.

5.4 VERSIONHALLINNAN LISÄÄMINEN

Versionhallinnassa käytettiin Git-työkalun graafista käyttöliittymää, joka ladattiin osoitteesta http://git-scm.com/. Työkalulla tehtiin koko WordPressin asennuskansios-

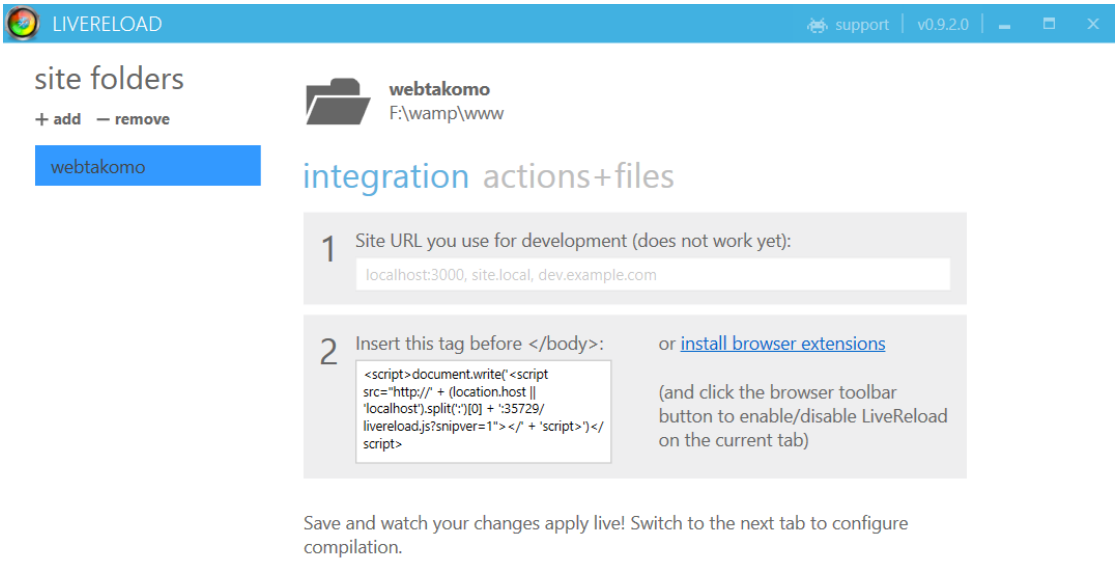
ta uusi repositorio. Kuviosta 7 on havainnollistettu projektin ensimmäisen version luomista versionhallintaan.



Kuvio 7. Versionhallinnan lisääminen projektiin

5.5 LIVERELOAD

LiveReload ladattiin osoitteesta http://livereload.com/ ja asennettiin koneelle. LiveReloadissa valittiin projektin kansio ja asennettiin Chromeen LiveReload-lisäosa, joka lisää sivulle LiveReload-skriptin automaattisesti. LiveReloadiin lisättiin kuvion 8 mukaisesti testiprojektin kansio seurantaan. Aina kun kyseisen kansion jokin tiedosto päivittyy, päivitetään samalla käytössä olevien selainten näkymä.



Kuvio 8. Projektin asettaminen LiveReload-ohjelmaan

5.6 LESS-KÄÄNTÄJÄN ASENNUS

Ensin asennettiin Node.js, joka ladattiin osoitteesta http://nodejs.org/download/. Nodejs asennuksen jälkeen asennettiin LESS-kääntäjä Windowsin konsolin kautta alla olevalla komennolla hyödyntäen Nodejs:n pakettien hallintatyökalua.

```
npm install --global less
```

LESS-kääntäjän asennuksen jälkeen asennettiin less2css -lisäosa SublimeText -editoriin, lataamalla lisäosa sen kotisivulta https://github.com/timdouglas/sublime-less2css ja sijoittamalla tiedostot SublimeTextin Packages-kansioon.

5.7 ULKOASUN TAITTAMINEN

Verkkosivuston kehitys käynnistettiin hahmottelemalla ulkoasua PhotoShopissa, mutta nopeasti siirryttiin jatkamaan kehitystä selaintekniikoilla mallipohjan päälle. Ensin lisättiin sivuston ylälaitaan kuvakaruselli, joka on samalla tausta logolle ja navigaatiolle. Koska mallipohjaan kuului FlexSlider valmiina, tarvitsi sivustolle ainoastaan lisätä siihen kuuluva HTML-koodi:

```
<div class="flexslider">
	<ul class="slides">
		<li>
			<img src="_/images/banner-1.png" title="Title">
		</li>
		<li>
			<img src="_/images/banner-2.png" title="Title">
		</li>
	</ul><!--.slides-->
</div><!--.flexslider-->
```

Koska kuvakarusellin haluttiin olevan ylälaidan taustana, täytyi se asemoida absoluuttiseksi ja koska ylälaidan muun sisällön haluttiin olevan karusellin päällä, annettiin niille relatiivinen asemointi sekä kuvagalleriaa suurempi z-indeksi.

```
.flexslider { position: absolute; }
.page-header .container { position: relative; z-index: 10; }
```

Etusivulle haluttiin kuusi sisältölaatikkoa vierekkäin kahdelle riville. Lisättiin riville oma elementti sekä jokaiselle sisältölaatikolle oma elementti.

```
<div class="clearfix">
	<div class="frontpage-column"></div>
	<div class="frontpage-column"></div>
	<div class="frontpage-column"></div>
</div>
<div class="clearfix">
	<div class="frontpage-column"></div>
	<div class="frontpage-column"></div>
	<div class="frontpage-column"></div>
</div>
```

CSS-tiedostossa merkittiin sisältölaatikot kolumneiksi.

```
.frontpage-column { .column(4); }
```

Kuviosta 9 nähdään miltä sivusto näytti CSS-tyylimäärittelyiden, kuvakarusellin ja kolumnien jälkeen.

Kuvio 9. Testisivuston ulkoasu taiton jälkeen.

5.8 RESPONSIIVISUUDEN LISÄÄMINEN

Selaimen leveyttä kaventaessa huomattiin sivun hajoavan noin 940px kohdalla, joten hajoamispiste määrättiin hieman ennen tätä 960px kohdalle. Kuviosta 10 huomataan kuinka sivuston navigaation ja sisältöalueen ulkoasun hajoaminen.

Kuvio 10. Layoutin hajoaminen mobiilinäkymässä

Peruskirjasinkoon ollessa 16px saadaan hajoamispisteen leveydeksi 60em. Suurimmat syyt ulkoasun hajoamiselle syntyivät navigaatiosta ja kolmen vierekkäisen palstan ti-

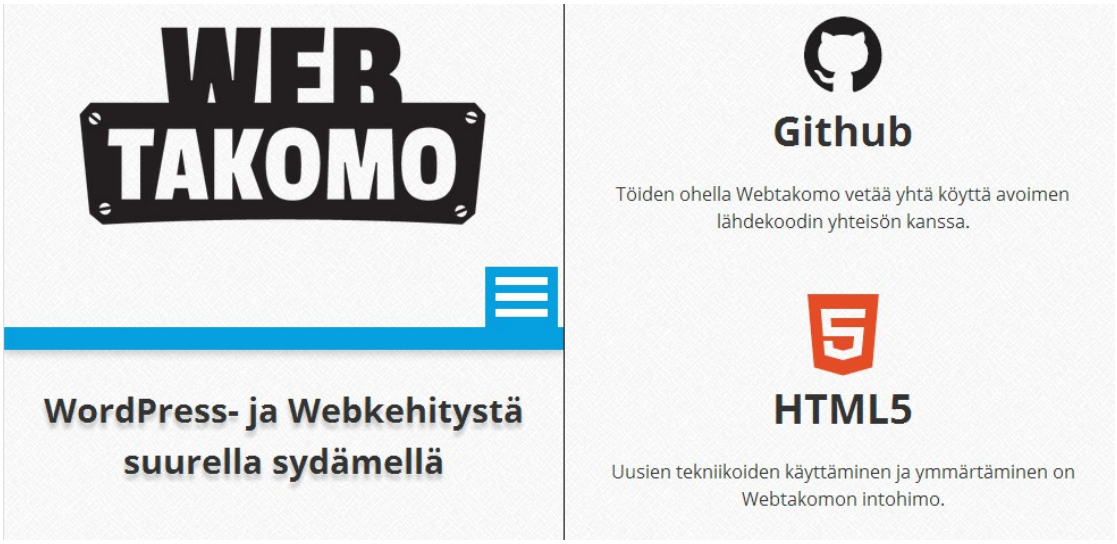
lan kapenemisesta. Navigaation tilalle vaihdettiin painikkeesta aukeava mobiilinavigaatio ja kolmen palstan sisältöjen leveydeksi vaihdettiin koko sivun leveys.

```
@media screen and (max-width:60em) {
        // Layout
        .frontpage-column { .column(12); margin-bottom: 30px; }

        // Navigation
        .js #mobile-menu-btn {
                background: @basecolor; width: 60px;
                padding: 2px; display: block; float: right;
                span {
                        height: 7px; margin: 7px;
                        display: block; background: #fff;


                }
        }
        .js #menu {
                max-height: 0px; clip: rect(0 0 0 0); zoom: 1;
        }
        #menu {
                overflow: hidden; width: 100%;
                background: #333; margin: 0px;
                &.opened { max-height: 9999px; }
                a { width: 100%; text-shadow: none; color: #eee; }
                a:after { display: none; }
        }
}
```

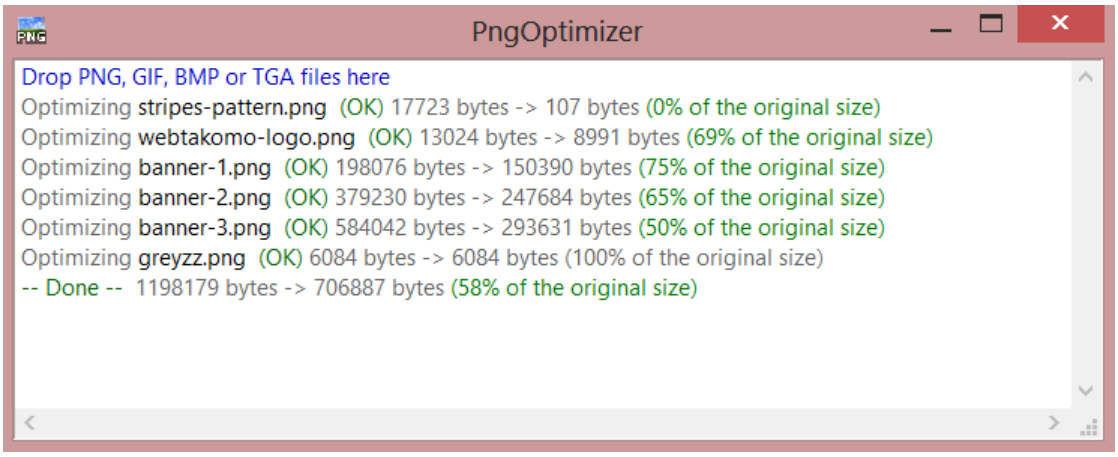
Sivuston mobiilinäkymä korjaantui huomattavasti edellä mainituilla muutoksilla, kuten kuviosta 11 nähdään.



Kuvio 11. Mobiililaitteille mukautettu sivuston ulkoasu

5.9 OPTIMOINTI

Sivuston DOM-elementtien latausaika ennen optimointia oli n. 1,5 sekuntia, kokonaislatausaika oli n. 2,9 sekuntia, palvelinpyyntöjä suoritettiin 24 kappaletta ja sivusto latasi tiedostoja 1,4 MB. Suurimman osan sivuston tiedostokokoa kasvatti kuvat. Etenkin kuvakarusellin suuret PNG-kuvat kasvattivat sivuston koko ja latausaikaa. Kuviosta 12 nähdää, että sivuston kuvien koko pieneni jopa 58 % yhteensä, kun kuvia pakattiin pngOptimizerilla.



Kuvio 12. PNG-kuvien pakkaus PngOptimizerillä

Sivuston tiedostokokoa pienennettiin lisäämällä sivuston .htaccess-tiedostoon alla esitetyt komennot, jotka mahdollistivat pakattujen sivujen tarjoamisen käyttäjälle.

```
# compress text, html, javascript, css, xml:
AddOutputFilterByType DEFLATE text/plain
AddOutputFilterByType DEFLATE text/html
AddOutputFilterByType DEFLATE text/xml
AddOutputFilterByType DEFLATE text/css
AddOutputFilterByType DEFLATE application/xml
AddOutputFilterByType DEFLATE application/xhtml+xml
AddOutputFilterByType DEFLATE application/rss+xml
AddOutputFilterByType DEFLATE application/javascript
AddOutputFilterByType DEFLATE application/x-javascript
AddType x-font/otf .otf
AddType x-font/ttf .ttf
AddType x-font/eot .eot
AddType x-font/woff .woff
AddType image/x-icon .ico
AddType image/png .png
```

Lopuksi WordPressiin asennettiin WP Super Cache -lisäosa joka tallentaa palvelimelle sivuista staattiset HTML-muotoiset tiedostot. Tämä rajoittaa sivuston tietokantakutsu-

ja sekä vähentää palvelimen kuormitusta. Samalla DOM-elementtien lataus nopeutuu.

Optimointien jälkeen DOM-elementtien lataus kesti n. 700 millisekuntia, sivun kokonaislatausaika kesti n. 1,6 sekuntia, palvelin pyyntöjä suoritettiin 23 kappaletta ja sivuston koko oli n. 780 KB. Sivustosta saatiin siis lähes puolitettua latausaika, joka varmasti parantaa käytettävyyttä ja asemaa hakukoneiden silmissä. Optimointi operaatiot olivat helppo ja nopea suorittaa, mutta WP Super Cache -lisäosa saattaa aiheuttaa ongelmia asiakaskäytössä, sillä sisällön muutokset näkyvät sivustolla vasta kun lisäosa on tallentanut ne staattisina tiedostoina palvelimelle.

6 JOHTOPÄÄTÖKSET

Opinnäytetyön tuloksena syntyi verkkoprojekteille mallipohja, joka korjaa selaimien välisiä eroavaisuuksia, lisää projektiin yleisiä toiminallisuuksia ja vähentää toistoa. Tuloksena syntyi HTML-pohjainen ja WordPress-teemaksi muotoiltu mallipohja. Mallipohja pyrittiin tekemään niin yksinkertaiseksi, että se olisi helppo liittää toimimaan erilaisissa ympäristöissä, kuten eri julkaisujärjestelmissä.

Opinnäytetyön lähtökohtana oli vähentää projektien aikana syntyvää toistoa, nopeuttaa projektien etenemistä ja tehdä kehittämisestä mahdollisimman joustavaa. Mallipohja ja projektia tukevat tekniikat vastasivat erittäin hyvin näihin haasteisiin. Mallipohjaa käytettiin opinnäytetyön aikana useassa projektissa ja jokaisen projektin aikana mallipohja kehittyi entisestään.

Eniten hyötyä kehitykseen toivat tekstieditoriin asennettava Emmet-lisäosa ja CSS-kieleksi kääntyvän LESS-kielen käyttöönotto. Emmetin käyttö nopeutti HTML-elementtien ja CSS-kielen kirjoittamista valtavasti. Lyhyen kokeilun jälkeen se tuntui niin luonnolliselta, ettei siitä halunnut luopua. LESS puolestaan teki CSS-kielestä modulaarisempaa ja mahdollisti aivan uudenlaisen ajattelun tyylien määrittelyssä. Vei hetken tottua uuteen ajattelumalliin, mutta pian huomattiin että LESSin ansiosta toisto väheni merkittävästi.

Tarkoituksena oli myös pyrkiä vastaamaan projekteissa usein esiintyviin toiminallisuuksiin kuten gallerioihin ja karuselleihin. Näihin toiminnallisuuksiin poimittiin suosituimpia lisäosia ja vertailtiin niiden toimivuutta erilaisissa tilanteissa ja ympäristöissä. Mikäli jatkossa projekteihin halutaan näitä ominaisuuksia, on niiden toiminallisuus liitetty oletuksena mallipohjaan. Mutta mikäli niitä ei tarvita, ne on erittäin helppo poistaa toiminnasta. Jatkossa mallipohjaan lisätään varmasti lisää toiminnallisuuksia. Esimerkiksi lomakkeiden luomiseen ja tarkistukseen voisi miettiä lisäosien vertailua tai oman lisäosan rakentamista.

Mallipohjaa kehitettiin opinnäytetyön aikana arviolta puoli vuotta. Tekniikat eivät tässä ajassa muuttuneet merkittävästi. Responsiivinen kehityksen vakiintuminen ja verkkokehityksen kypsyminen perinteisemmän ohjelmoinnin suuntaan olivat selvästi opinnäytetyön aikana vallinneita käytäntöjä. Uudet sivustot tuntuivat olevan suurimmaksi osaksi responsiivisia, eikä sitä ajateltu enää lisäominaisuutena vaan perusosana sivuston kehitystä. Lisäksi erilaisten rajapintojen ja modulaarisempaan kehitykseen tähtäävien tekniikoiden käytön lisääntyminen olivat selkeästi suosiossa.

Mallipohjan luonteesta johtuen se ei ole valmis. Sitä käytetään projektien pohjana, mutta uusien tekniikoiden ja käytäntöjen myötä sitä muokataan vastaamaan entistäkin paremmin projektien kulkuun. Verkkotekniikoiden kehittyessä valtavaa vauhtia, on tärkeää pysyä kehityksen mukana ja poimia mallipohjaan sellaiset tekniikat jotka hyödyttävät projekteja parhaiten sekä toimivat eri ympäristöissä.

LÄHTEET

Ashkenas, J. 2013. CoffeeScript. Viitattu 9.4.2013. http://coffeescript.org/

Ates, F., Irish, P., Sexton, A., Seddon, R. & Farkas, A. 2013. Modernizr dokumentaatio. Viitattu 21.3.2013. http://modernizr.com/docs/

Catlin H., Weizenbaum N. & Eppstein C. 2013. Sass. Viitattu 23.3.2013. http://sass-lang.com/

Constantin, J. 2013. Artikkeli Smashing Magazinen sivustolla (2013 Edition). Viitattu 2.10.2013. http://www.smashingmagazine.com/2013/05/17/typographic-design-patterns-practices-case-study-2013/, Typographic Design Patterns And Current Practices (2013 Edition).

Coyer, C. 20011. Useful CSS3 LESS Mixins. Viitattu 9.10.2013. http://css-tricks.com/snippets/css/useful-css3-less-mixins/

Emmet Cheat Sheet. N.d. Emmet Documentation. Viitattu 29.9.2013. http://docs.emmet.io/cheat-sheet/

Emmet Download. N.d. Emmet. Viitattu 29.9.2013. http://emmet.io/download/

Fancybox2. 2012. Lisenssi. Viitattu 8.4.2013. http://fancyapps.com/fancybox/#license

Frost, B. 2012. Viitattu 15.9.2013. http://bradfrostweb.com/blog/web/responsive-nav-patterns/

Gallagher, N. 2013. Normalize.css. Viitattu 17.3.2013. http://necolas.github.com/normalize.css/

Google. 2011. Hakukoneoptimoinnin aloitusopas. Viitattu 13.9.2013. http://www.google.fi/intl/fi/webmasters/docs/search-engine-optimization-starter-guide-fi.pdf

Kiss. 2011. SimpLESS. Viitattu 22.9.2013. http://wearekiss.com/simpless

Krumins P. 2013. Browserling. https://browserling.com/

Marcotte, E. 2011. Responsive Web Design. New York: A Book Apart

Meyer, E. 2011. CSS-tools: Reset CSS . Viitattu 17.3.2013. http://meyerweb.com/eric/tools/css/reset/

Mayer, E. 2006. Unitless line height. Viitattu 2.9.2013. http://meyerweb.com/eric/thoughts/2006/02/08/unitless-line-heights/

Moz. 2013. How search engines operate. Viitattu 13.9.2013. http://moz.com/beginners-guide-to-seo/how-search-engines-operate

Moz. 2013. The Basics of Search Engine Friendly Design & Development. Viitattu 13-.9.2013. http://moz.com/beginners-guide-to-seo/basics-of-search-engine-friendly-design-and-development

Nielsen, J. 2013. Auto-Forwarding Carousels and Accordions Annoy Users and Reduce Visibility. Viitattu. 4.9.2013. http://www.nngroup.com/articles/auto-forwarding/

Nilsson, H. 2013. PngOptimizer. Viitattu 3.10.2013. http://psydk.org/PngOptimizer.php

Schema. 2011. Getting started with schema.org. Viitattu. 13.9.2013. http://schema.org/docs/gs.html

Sellier A. 2013. LESS. Viitattu 23.3.2013. http://lesscss.org/

Browser version partially combined. 2013. StatsCounter. Viitattu 7.3.2013. http://gs.statcounter.com/#browser_version_partially_combined-FI-monthly-201303-201303-bar

Voormedia. 2013. Viitattu 3.10.2013. http://tinypng.org/

World Wide Web Consortium. Viitattu 7.3.2013. http://www.w3.org/TR/css3-mediaqueries/

Yahoo. 2013. Performance. Viitattu 8.4.2013. http://developer.yahoo.com/performance/rules.html#postload

LIITTEET

LIITE 1. INDEX.HTML

```
<!DOCTYPE html>
<!--[if lt IE 7]><html lang="en" class="no-js lt-ie9 lt-ie8 lt-ie7"> <![endif]-->
<!--[if IE 7]><html lang="en" class="no-js lt-ie9 lt-ie8"> <![endif]-->
<!--[if IE 8]><html lang="en" class="no-js lt-ie9"> <![endif]-->
<!--[if gt IE 8]><!--> <html lang="en" class="no-js"> <!--<![endif]-->
<head>
	<meta charset="UTF-8">
	<meta http-equiv="X-UA-Compatible" content="IE=edge">
	<title>Title</title>
	<meta name="description" content="">
	<meta name="viewport" content="width=device-width, initial-scale=1">
	<link rel="stylesheet" href="_/libs/flexslider/flexslider.css">
	<link rel="stylesheet" href="style.css">
	<script src="_/libs/modernizr-2.6.2.min.js"></script>
</head>
<body>
	<div id="wrapper">
		<header class="page-header" role="banner">
			<div class="header-content">
				<a href="#" class="site-name"><h1>Header</h1></a>
				<div id="mobile-menu-btn">
					<span></span>
					<span></span>
					<span></span>
				</div>
				<nav id="menu" class="clearfix" role="navigation">
			<ul>
					<li><a href="#">Menu-item</a></li>
					<li><a href="#">Menu-item</a></li>
					<li><a href="#">Menu-item</a></li>
					<li><a href="#">Menu-item</a></li>
```

```
                                        <li><a href="#">Menu-item</a></li>
                                    </ul>
                                </nav>
                            </div><!--.header-content-->
                        </header><!--.page-header-->

                        <div class="page-content container">
                            <section class="primary" role="main">
                                <h1>Primary</h1>
                            </section><!--.primary-->

                            <aside class="secondary" role="complementary">
                                <h2>Secondary</h2>
                            </aside><!--.secondary-->
                        </div><!--.page-content-->

                        <footer class="page-footer" role="contentinfo">
                            <div class="container">
                                <div class="footer-content">
                                    <h2>Footer</h2>
                                </div>
                            </div>
                        </footer><!--.page-footer-->
                        <!-- scripts →
                        <script
src="http://ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js"></script>
                        <script>window.jQuery || document.write('<script src="_/libs/jquery-
1.9.1.min.js"><\/script>')</script>
                        <script src="_/libs/responsive-nav/responsive-nav.min.js"></script>
                        <script src="_/libs/magnific-popup/magnific-popup.js"></script>
                        <script src="_/libs/flexslider/jquery.flexslider-min.js"></script>
                        <script src="_/js/functions.js"></script>
                    </div><!--#wrapper-->
</body>
</html>
```

LIITE 2. STYLE.LESS

```
/* Imports
======================================== */
@import (less) "_/libs/normalize.css";
@import "_/libs/grid.less";
@import "_/css/mixins.less";
@import (less) "_/libs/responsive-nav/responsive-nav.css";
@import (less) "_/libs/magnific-popup/magnific-popup.css";

/* Variables
======================================== */
@column-width: 60;
@gutter-width: 20;
@columns: 12;
@total-width: 100%;
@blue: #009000;

/* General
======================================== */
.clearfix:after, .container:after { display: table; content: ''; clear: both; }

/* Typography
======================================== */
html, body { color: #333; font: 400 100%/1.5 Helvetic, Arial, sans-sefif; word-wrap: break-word; }
h1, h2, h3, h4, h5, h6, p, ul, ol { margin: 0px 0px 1rem 0px; }
h1 { .font-size(48); }
h2 { .font-size(36); }
h3 { .font-size(24); }
h4 { .font-size(18); }
h5 { .font-size(16); }
h6 { .font-size(16); }
p { .font-size(16); }
img { max-width: 100%; height: auto; }
ul, ol {}
blockquote {
        font-style: italic; font-weight: 300; padding-left: 75px; position: relative;
```

```
        &:before {
                color: #eee; content: '\201C'; font-size: 140px; font-weight: 400; line-height: .8;
padding-right: 25px;
                position: absolute; left: -15px; top: -3px;
        }
}

/* Layout
========================================== */
.header-content { .column(12); }
.primary { .column(9); }
.secondary { .column(3); }
.footer-content { .column(12); }

/* Header
========================================== */
.site-name { text-decoration: none; }
.page-header {
        background: #0090FF;
        h1 { color: #fff; }
}

/* Navigation
========================================== */
#mobile-nav-btn { display: none; }
#menu {
        &.closed { position: static !important; }
        ul { list-style: none; padding: 0px; margin: 0px; }
        li { display: inline; }
        a { background: #fff; color: #333; .inline-block(); padding: 10px; text-decoration: none; }
        a:hover{ opacity: 0.8; };
        .current_page_item a,
        .current-menu-item a {  }
        .current_page_ancestor a,
        .current-page-ancestor a {  }
}

/* Slider
========================================== */
```

```
.flexslider {
        .border-radius( 0px ); .box-shadow( 0px 0px 0px #000 ); border: none;
        .flex-control-nav {
                position: absolute; bottom: 10px; z-index: 10;
                li a { background: #fff; .box-shadow( 0px 0px 0px #000 ); }
                li a.flex-active { background: #000; .box-shadow( 0px 0px 0px #000 ); }
        }
}

/* Footer
========================================== */
.page-footer {
        background: #333;
        h2 { color: #fff; }
}

/* Media queries
========================================== */
@media screen and (max-width:62.5em) { // max-width: 1000px
        // Layout
        .primary { .column(12); }
        .secondary{ .column(12); }

        // Navigation
        .js #mobile-menu-btn {
                background: #fff; width: 42px; padding: 2px; display: block; float: right;
                span { height: 5px; margin: 5px; display: block; background: #333; }
        }
        .js #menu { max-height: 0px; clip: rect(0 0 0 0); zoom: 1; }
        #menu {
                overflow: hidden; width: 100%;
                &.opened { max-height: 9999px; }
                a { width: 100%; }
        }
}
```

LIITE 3. MIXINS.LESS

```
.font-size(@sizeValue) {
	@remValue: (@sizeValue / 16);
	@pxValue: @sizeValue;
	font-size: ~"@{pxValue}px";
	font-size: ~"@{remValue}rem";
}
.inline-block { display: inline-block; /* ie7 fix */ *display: inline; *zoom: 1; }
.text-shadow (@string: 0 1px 3px rgba(0, 0, 0, 0.25)) {
	text-shadow: @string;
}
.box-shadow (@string) {
	-webkit-box-shadow: @string;
	-moz-box-shadow:    @string;
	box-shadow:         @string;
}
.drop-shadow (@x: 0, @y: 1px, @blur: 2px, @spread: 0, @alpha: 0.25) {
	-webkit-box-shadow:      @x @y @blur @spread rgba(0, 0, 0, @alpha);
	-moz-box-shadow:         @x @y @blur @spread rgba(0, 0, 0, @alpha);
	box-shadow:              @x @y @blur @spread rgba(0, 0, 0, @alpha);
}
.inner-shadow (@x: 0, @y: 1px, @blur: 2px, @spread: 0, @alpha: 0.25) {
	-webkit-box-shadow: inset @x @y @blur @spread rgba(0, 0, 0, @alpha);
	-moz-box-shadow:    inset @x @y @blur @spread rgba(0, 0, 0, @alpha);
	box-shadow:         inset @x @y @blur @spread rgba(0, 0, 0, @alpha);
}
.box-sizing (@type: border-box) {
	-webkit-box-sizing: @type;
	-moz-box-sizing:    @type;
	box-sizing:         @type;
}
.border-radius (@radius: 5px) {
	-webkit-border-radius: @radius;
	-moz-border-radius:    @radius;
	border-radius:         @radius;
```

```
        -moz-background-clip:    padding;
        -webkit-background-clip: padding-box;
        background-clip:         padding-box;
}
.border-radiuses (@topright: 0, @bottomright: 0, @bottomleft: 0, @topleft: 0) {
        -webkit-border-top-right-radius:    @topright;
        -webkit-border-bottom-right-radius: @bottomright;
        -webkit-border-bottom-left-radius:  @bottomleft;
        -webkit-border-top-left-radius:     @topleft;

        -moz-border-radius-topright:       @topright;
        -moz-border-radius-bottomright:    @bottomright;
        -moz-border-radius-bottomleft:     @bottomleft;
        -moz-border-radius-topleft:        @topleft;

        border-top-right-radius:           @topright;
        border-bottom-right-radius:        @bottomright;
        border-bottom-left-radius:         @bottomleft;
        border-top-left-radius:            @topleft;

        -moz-background-clip:    padding;
        -webkit-background-clip: padding-box;
        background-clip:         padding-box;
}
.opacity (@opacity: 0.5) {
        -webkit-opacity:  @opacity;
        -moz-opacity:                @opacity;
        opacity:              @opacity;
}
.gradient (@startColor: #eee, @endColor: white) {
        background-color: @startColor;
        background: -webkit-gradient(linear, left top, left bottom, from(@startColor),
to(@endColor));
        background: -webkit-linear-gradient(top, @startColor, @endColor);
        background: -moz-linear-gradient(top, @startColor, @endColor);
        background: -ms-linear-gradient(top, @startColor, @endColor);
        background: -o-linear-gradient(top, @startColor, @endColor);
}
.horizontal-gradient (@startColor: #eee, @endColor: white) {
```

```
        background-color: @startColor;
        background-image: -webkit-gradient(linear, left top, right top, from(@startColor),
to(@endColor));
        background-image: -webkit-linear-gradient(left, @startColor, @endColor);
        background-image: -moz-linear-gradient(left, @startColor, @endColor);
        background-image: -ms-linear-gradient(left, @startColor, @endColor);
        background-image: -o-linear-gradient(left, @startColor, @endColor);
}
.animation (@name, @duration: 300ms, @delay: 0, @ease: ease) {
        -webkit-animation: @name @duration @delay @ease;
        -moz-animation:    @name @duration @delay @ease;
        -ms-animation:     @name @duration @delay @ease;
}
.transition (@transition) {
        -webkit-transition: @transition;
        -moz-transition:    @transition;
        -ms-transition:     @transition;
        -o-transition:      @transition;
}
.transform(@string){
        -webkit-transform: @string;
        -moz-transform:   @string;
        -ms-transform:          @string;
        -o-transform:           @string;
}
.scale (@factor) {
        -webkit-transform: scale(@factor);
        -moz-transform:   scale(@factor);
        -ms-transform:          scale(@factor);
        -o-transform:           scale(@factor);
}
.rotate (@deg) {
        -webkit-transform: rotate(@deg);
        -moz-transform:   rotate(@deg);
        -ms-transform:          rotate(@deg);
        -o-transform:           rotate(@deg);
}
.skew (@deg, @deg2) {
        -webkit-transform:      skew(@deg, @deg2);
```

```
        -moz-transform:   skew(@deg, @deg2);
        -ms-transform:          skew(@deg, @deg2);
        -o-transform:           skew(@deg, @deg2);
}
.translate (@x, @y:0) {
        -webkit-transform:      translate(@x, @y);
        -moz-transform:   translate(@x, @y);
        -ms-transform:          translate(@x, @y);
        -o-transform:           translate(@x, @y);
}
.translate3d (@x, @y: 0, @z: 0) {
        -webkit-transform:      translate3d(@x, @y, @z);
        -moz-transform:   translate3d(@x, @y, @z);
        -ms-transform:          translate3d(@x, @y, @z);
        -o-transform:           translate3d(@x, @y, @z);
}
.perspective (@value: 1000) {
        -webkit-perspective:    @value;
        -moz-perspective:       @value;
        -ms-perspective: @value;
        perspective:            @value;
}
.transform-origin (@x:center, @y:center) {
        -webkit-transform-origin: @x @y;
        -moz-transform-origin:    @x @y;
        -ms-transform-origin:     @x @y;
        -o-transform-origin:      @x @y;
}
```