Opinnäytetyö (AMK)

Tietotekniikka

Mediatekniikka

2014

Kalle Nisula

RESPONSIIVINEN TEEMAPOHJA JULKAISUJÄRJESTELMILLE

– suunnittelu ja sovitus testijärjestelmään

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tieotekniikan ko | Mediatekniikka

2014 | Sivumäärä 42

Mika Luimula

Kalle Nisula

RESPONSIIVINEN TEEMAPOHJA JULKAISUJÄRJESTELMILLE

– suunnittelu ja sovitus testijärjestelmään

Työn toimeksiantajan Mobiilimarkkinointi Routa Oy:n käyttöön tarvittiin kaikille päätelaitteille skaalautuva verkkosivupohja, joka olisi helposti kustomoitavissa ulkoasultaan eri asiakkaiden tarpeisiin.

Opinnäytetyön tarkoituksena oli esitellä responsiivista verkkosuunnittelua ja hyviä käytäntöjä responsiivisten verkkosivujen suunnitteluun sekä suunnitella ja toteuttaa responsiivinen teemapohja, joka on sovitettavissa eri julkaisujärjestelmille teemaksi.

Responsiivisessa verkkosuunnittelussa sivusto suunnitellaan mukautumaan kävijänsä käyttöympäristöön. Perusperiaatteet responsiiviselle verkkosivulle ovat joustava rakenne, joustava sisältö, sekä uusimpien verkkotekniikoiden mukana tuomat mediakutsut, jotka mukauttavat sivuston rakennetta selainnäkymän mukaan. Hyvän responsiivisen sivuston suunnittelu vaatii kuitenkin muutakin, on otettava huomioon käytettävyyteen liittyviä seikkoja, kuten kosketusohjauksen tuomat haasteet sekä datan käyttö hitailla mobiiliyhteyksillä.

Teemapohja toteutettiin HTML5 ja CSS3 -verkkotekniikoita hyödyntäen, kuitenkin pyrkien säilyttämään yhteensopivuus kaikkien kohtalaisesti käytössä olevien selainten ja selainversioiden kanssa.

Työn lopputuotoksena toteutettiin testijärjestelmäksi valitun Concrete5-julkaisujärjestelmän pohjalle testisivusto, johon suunniteltu teemapohja on sovitettu valmiiksi teemaksi. Sovitettaessa pohjaa testijärjestelmään huomattiin sen vaativan vielä viimeistelyä ollakseen valmis tuotantokäyttöön. Viimeistelyä ei nähty järkeväksi toteuttaa enää tämän opinnäytetyön puitteissa.

ASIASANAT:

responsiivinen verkkosuunnittelu, julkaisujärjestelmä, HTML5, CSS3

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information Technology | Media Technology

2014 | Total number of pages 42

Mika Luimula

Kalle Nisula

RESPONSIVE WEBSITE TEMPLATE FOR CONTENT MANAGEMENT SYSTEMS

- design and implementation to test system

The commissioner of the project, Mobiilimarkkinointi Routa Oy, needed a website template that is scalable to different devices and easily customizable to their clients.

The purpose of this thesis was to provide an overview to responsive web design and demonstrate good practices for designing responsive web sites as well as to design and implement responsive website template that is adaptable to a theme for different content management systems and publishing platforms.

In responsive web design the site is designed to accommodate the visitor's environment. The basic principles for responsive web design include flexible structure, flexible content, and the use of CCS media queries to adjust the structure based on browser view. Good responsive design, however, additionally requires consideration of usability issues, such as touch interface, or data use in slow mobile connections.

The template was carried out with web technologies such as HTML5 and CSS3, still aiming it to be compatible with all fairly used browsers.

The output of the thesis was a demo web site, based on the Concrete5 content management system. The demo site used the theme adapted from the designed responsive template. During the implementation was discovered that some finishing to the template was needed. The finishing part was proved to be a project of its own and therefore was considered to be outside the scope of this thesis.

KEYWORDS:

Responsive Web Design, Content Management System, HTML5, CSS3

SISÄLTÖ

KUVAT

SANASTO

Ajax	Joukko verkkotekniikoita, joiden avulla verkkosivu tai sovellus vaihtaa dataa palvelimen kanssa taustalla, niin ettei verkkosivua tarvitse ladata aina kokonaan uudelleen. (Asynchronous JavaScript And XML)
CDN	Useasta palvelimesta koostuva järjestelmä, jonka tarkoitus on jakaa sisältöä käyttäjille verkon yli tehokkaasti ja toimintavarmasti. (Content Delivery System)
CSS	Tyyliohjekieli, jota käytetään mm. HTML-elementtien ulkoasun määrittelyyn. (Cascading Style Sheets)
FTP	Tiedostonsiirtomenetelmä kahden tietokoneen, yleensä asiakaan ja palvelimen välille. (File Transfer Protocol)
HTML	Verkkosivujen esittämiseen käytetty kuvauskieli. (Hypertext Markup Language)
MVC	Ohjelmistoarkkitehtuurityyli, jota käytetään mm. graafisten käyttöliittymien suunnittelussa. (Model-View-Controller)
PHP	Ohjelmointikieli, jota käytetään laajasti verkkosovellusten ohjelmointiin. (PHP: Hypertext Preprocessor)
W3C	Kansainvälinen yhteisö, joka ylläpitää ja kehittää World Wide Webin standardeja. (World Wide Web Consortium)
MySQL	Avoimen lähdekoodin relaatiotietokantaohjelmisto

1 JOHDANTO

Erilaisten mobiilipäätelaitteiden valtaisa yleistyminen ja määrän räjähdysmäinen kasvu on muuttanut ihmisten tapaa selata internetiä. Yhä suurempi osa käyttää päivittäiseen internetselailuun puhelinta, tablettia tai muuta pienikokoisella näytöllä varustettua päätelaitetta.

Laitteiden suuri kirjo ja erilaiset näyttökoot asettavat haasteita verkkosivujen suunnitteluun, kun kävijälle halutaan taata miellyttävä käyttökokemus niin suurella työpöytänäytöllä kuin käteen sopivan puhelimen kosketusnäytöllä. Responsiivisella suunnittelulla verkkosivun sisältö saadaan skaalautumaan erikokoisille näytöille sopivaksi.

Internetin merkitys mediana on kasvanut ja sivustoilla tarjotaan suuria määriä sisältöä kävijöille. Sisältöä myös päivitetään usein. Sivustojen hallintaa helpottamaan on luotu erilaisia julkaisu- ja sisällönhallintajärjestelmiä. Useimmissa järjestelmissä sivuston rakenne ja ulkoasu on erotettu sisällöstä ulkoasuteemaan.

Työn toimeksiantajan Mobiilimarkkinointi Routa Oy:n perusajatuksena on mobilisoida asiakkaidensa verkkonäkyvyys. Monissa tapauksissa samalla kertaa uudistetaan myös asiakkaan verkkosivujen työpöytäversio. Tätä varten tarvittiin verkkosivupohja, johon voidaan tehokkaasti toteuttaa asiakkaan verkkosivusto sekä mobiili- että työpöytäkäyttöön.

Joissain tapauksissa asiakkaalla on käytössään jo jokin julkaisujärjestelmä, ja hän haluaa uuden verkkosivustonsa samalle järjestelmälle, tällöin verkkosivupohjan pitäisi olla pienellä vaivalla sovitettavissa tähän järjestelmään.

Opinnäytetyössä on tarkoituksena esitellä responsiivista verkkosuunnittelua sekä eritellä hyviä käytäntöjä responsiivisen verkkosivuston toteutukseen. Lisäksi tarkoituksena on suunnitella ja toteuttaa responsiivinen, eri julkaisujärjestelmille sovitettavissa oleva teemapohja sekä sovittaa se teemaksi Concrete5-julkaisujärjestelmälle. Teemapohjaa ei ole tarkoitus käyttää tulevaisuudessa

asiakkaiden sivustoilla sellaisenaan, vaan sen ulkoasu kustomoidaan tapauskohtaisesti esimerkiksi yrityksen graafisen ohjeiston mukaiseksi.

2 RESPONSIIVINEN VERKKOSUUNNITTELU

Responsiivisella verkkosuunnittelulla tarkoitetaan lähestymistapaa, jossa sivustot suunnitellaan mukautumaan kävijänsä käyttöympäristöön. Termiä *Responsive Web Design* käytti ensimmäisenä Ethan Marcotte *A List Apart* -julkaisulle kirjoittamassaan artikkelissa. Tämä tapahtui vuonna 2010, ja viimeisten vuosien aikana responsiivisuus onkin ollut yksi alalla vallinneista trendeistä. Käytännössä responsiivinen verkkosuunnittelu tarkoittaa sitä, että sivuston rakenne ja muotoilu määrittyvät päätelaitteen ja sen näyttökoon mukaan. Jos kävijä vaihtaa tietokoneen esimerkiksi tablet-laitteeseen tai älypuhelimeen, sama sivusto mukautuu automaattisesti asettelultaan ja toiminnoiltaan päätelaitteelle sopivaksi. [1] Käyttökokemuksesta tulee miellyttävä päätelaitteesta riippumatta ja vältytään tarpeelta suunnitella ja ylläpitää erillisiä versioita sivustosta eri laitetyypeille.

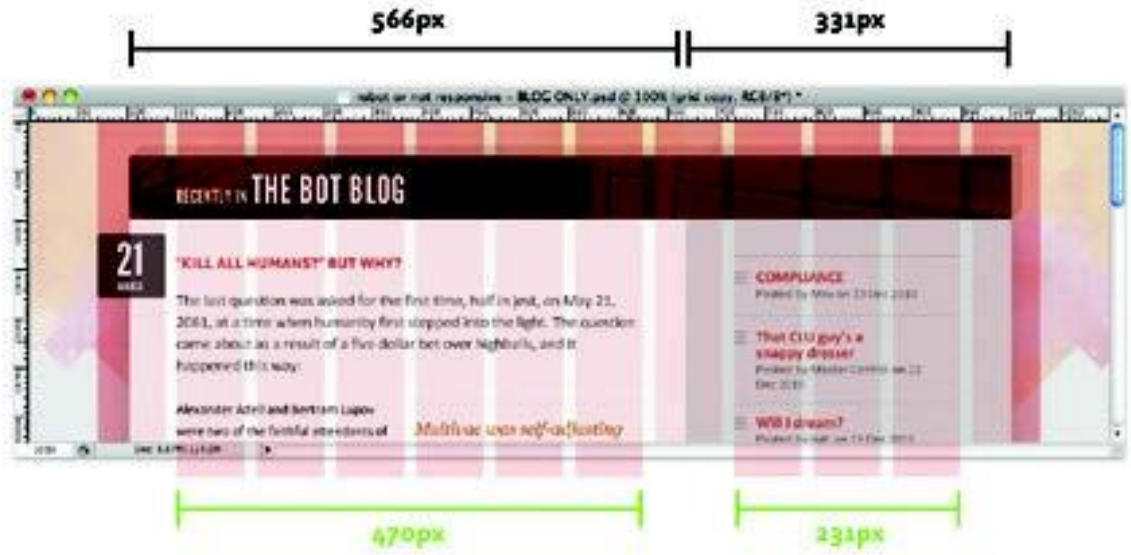
2.1 Responsiivisuuden toteutus

Responsiivinen verkkosivu ei vaadi mitään erillistä tekniikkaa, vaan se on toteutettavissa olemassa olevin menetelmin ja tekniikoin. Tarvitaan vain hieman toisenlaista lähestymistapaa ja tekniikoiden luovaa hyödyntämistä.

Responsiivisen sivun toteutuksessa on kolme ydinosaa [2]:

- joustava rakenne
- joustava sisältö
- CSS3 mediakutsut.

Joustava rakenne pyrkii siihen, ettei sivua tarvitse missään tilanteessa pienimmälläkään selainikkunalla rullata vaakasuunnassa. Tämä onnistuu siten, ettei sivun rakenteellisia elementtejä tehdä leveydeltään kiinteiksi, vaan leveys määritellään riippuvaiseksi ympäröivän elementin leveydestä. Elementille määritellään leveys prosenttiyksiköinä eikä tavallisesti elementtien koon määrittämiseen käytettäviä pikseleinä. Rakenteellinen joustavuus on helposti toteutettavissa käyttämällä ruudukkomaista *grid-layout*-asettelua (Kuva 1) [2].



Kuva 1. Havainnekuva *grid-layout*-asettelusta [2].

Toinen responsiiviseen suunnittelun kolmesta ydinosasta on joustava sisältö. Tekstielementit rivittyvät kutakuinkin automaattisesti elementin leveyden mukaan, mutta ongelmallisia ovat kuvat ja muut sivuille upotettavat sisällöt, joiden leveys on ennalta määrätty. Näiden leveys täytyy määrittää CSS-tyyleillä prosentuaaliseksi samaan tapaan kuin rakenteelliset elementit. [2]

Pelkästään joustavuus ei tee sivusta responsiivista ja jokaiselle näyttökoolle sopivaa. Sama asettelu ei toimi kapealla älypuhelimen näytöllä ja suurella työpöytänäytöllä. Kolmas ydinosa on CSS-tyyliohjeiden uusimman, kolmannen version mukanaan tuomat kehittyneet *media query* -mediakutsut. Mediakutsujen avulla elementeille voidaan määritellä erilliset tyylit riippuen selainikkunan mitoista tai päätelaitteen ominaisuuksista. Tärkeä pala mediakutsuissa on niin sanottujen *breakpointien*, eli murtumiskohtien määrittely. *Breakpoint* tarkoittaa sitä selainikkunan tai näytön leveyttä, jossa *media query* astuu voimaan. Leveys voidaan määrittää minimileveytenä, jolloin kutsu astuu voimaan kun näytön tai ikkunan leveys on suurempi kuin annettu arvo:

```
@media only screen and (min-width: 480px) {
    css rules...
}
```

Toinen vaihtoehto on määrittää *breakpoint*-leveys maksimileveytenä, jolloin kutsu on voimassa näytön tai ikkunan leveyden ollessa arvoa pienempi.

```
@media only screen and (max-width: 480px) {
    css rules...
}
```

Näistä nimenomaan `min-width`-arvo on käytännöllinen, jos sivuja suunnitellaan *mobile first* -ajattelumallia hyödyntäen. [2]

2.2 Hyviä käytäntöjä responsiiviseen suunnitteluun

Macrotte loi artikkelissaan ja myöhemmin julkaistussa kirjassaan pohjan responsiiviselle verkkosuunnittelulle. Viimeisten vuosien aikana on kehitetty lukuisia tekniikoita ja käytäntöjä parantamaan responsiivisen sivun käyttökokemusta.

Kuten todettua, responsiivisen verkkosivun toteutus vaatii yksinkertaisuudessaan vain joustavan sivurakenteen, joustavan sisällön sekä mediakutsut, sisällön uudelleenjärjestelyyn. Käytettävyydeltään hyvä responsiivinen sivusto vaatii muutakin, kuin sarakkeiden uudelleenjärjestelyä allekkain näyttökoon kutistuessa. On otettava huomioon mm. laitteiden syöttötavat sekä niiden erilaiset käyttötavat ja tarpeet.

2.2.1 Syöttötavat

Koska responsiivinen sivu on tarkoitettu skaalautumaan erilaisille laitetyypeille, on suunnittelussa otettava huomioon eri laitteiden erilaiset syöttötavat. Pöytätietokoneella tai kannettavalla selatessa on useimmiten käytössä hiiri ja näppäimistö, kun taas kosketusnäytöllisissä laitteissa osoittimena toimii useimmiten sormi. Hiiri ja näytöllä näkyvä kursori ovatkin huomattavasti tarkemmin ohjattavissa, kuin tylppä sormenpää.

Hiirtä käytettäessä esimerkiksi navigaation linkit voivat olla tiiviisti rivissä, mutta kosketusohjauksessa ne on sijoitettava huomattavasti väljemmin, jotta voidaan välttyä virhepainalluksilta.

Esimerkiksi Apple ohjeistaa tekemään painikkeista näytöllä vähintään 40 × 40 pikselin kokoisia. Kuitenkin keskimääräinen peukalonpää vie näytöltä halkaisijaltaan jopa 72 pikselin kokoisen tilan, joten sitä pienempien painikkeiden painamisessa voi tulla käytettävyysongelmia. Hyvä ohjenuora olisikin käyttää peukalomittaa tarvittavan tilan arvioimiseen. Pienellä mobiilinäytöllä, jossa tilaa on vähän, sekään ei aina ole kovin käytännöllistä. [3]

Toinen huomioitava seikka syöttötavoissa on se, että kosketusohjauksessa ei voida käyttää hiiriohjauksesta tuttua *hover*-efektiä, eli sitä, että asioita tapahtuu vietäessä kursori elementin päälle.

2.2.2 Datan käyttö

Monessakin suhteessa suurin erottava ja rajoittava tekijä tietokoneella ja mobiililaitteella selaamisen välillä ei olekaan näytön koko, vaan internetyhteyden nopeus sekä data-rajoitukset. Mobiiliyhteyksien nopeudet ovat toki kasvaneet viime aikoina ja vetävät jo vertoja kiinteille yhteyksille, joten yhteysnopeuden ja sitä kautta sivujen latausaikojen merkitys on vähentynyt. Sitä suuremmaksi tekijäksi nouseekin sivun lataamiseen tarvittavan datan määrä.

Suomessa ollaan siinä mielessä onnekkaita, että täällä on totuttu niin kiinteissä kuin langattomissa yhteyksissä kiinteän kuukausimaksun hinnoittelumalliin riippumatta liikkuvan datan määrästä. Monissa muissa maissa, esimerkiksi Yhdysvalloissa, mobiiliyhteyksien laskutus perustuu pitkälti juuri datan määrään. Eli mitä enemmän ja tiedostokooltaan suurempia sivuja, kuvia, videoita tai ääntä ladataan internetyhteyden yli, sitä enemmän se maksaa.

Tästä syystä on kiinnitettävä huomiota sivuston kokoon ja erityisesti sivuston mediasisällön, kuten kuvien ja videoiden käyttöön.

2.2.3 Responsiiviset kuvat

Responsiivisen verkkosuunnittelun isoimmaksi ja puhutuimmaksi haasteeksi onkin muodostunut juuri kuvien käyttö sivustolla. Suurille ja tarkoille näytöille halutaan tarjota isoja ja tarkkoja kuvia, kun taas pienellä näytöllä sekä hitaammalla ja rajoitetulla datayhteydellä varustetulle älypuhelimelle olisi tarpeen näyttää huomattavasti mitoiltaan ja tiedostokooltaan pienempiä kuvia. [4] W3C yhdessä muiden tahojen kanssa suunnittelee lisäyksiä HTML-kieleen sekä standardia responsiivisille kuville [5], mutta sitä ennen kuvien kanssa on tehtävä kompromisseja.

Ratkaisuja kuvien esittämiseen responsiivisella sivustolla on kehitetty viime vuosien aikana useita. Eri ratkaisut lähestyvät asiaa hieman eri kanteilta ja sen vuoksi sekä niiden hyödyt että myös haitat on punnittava tarkoin. Täydellistä ratkaisua ei ole, joten on tarkasteltava, mikä ratkaisuista sopii parhaiten omaan projektiin. [4]

Picturefill on komentosarja, joka mimikoi HTML-kieleen lisättäväksi suunniteltua `<picture>`-elementtiä. Uutta elementtiä käytetään `<img>`-elementin sijasta ja se mahdollistaa useamman kuvalähteen, jolloin kuvasta voidaan näyttää erikokoista versiota erikokokoisille näytöille. Koska *Picturefill* ei käytä standardia merkkaustapaa kuvien esittämiseen, sen käyttö esim. julkaisujärjestelmässä voi olla melko haastavaa. [4]

Adaptive Images on palvelimelle asennettava ohjelma, joka skaalaa kuvan lennossa sopivan kokoiseksi ennen kuin se ladataan näytettäväksi selaimessa. Hyvä puoli siinä on, että sivuston lähdekoodiin ei tarvitse tehdä muutoksia. Koska kyseessä on palvelinpuolen ratkaisu, täytyy sivuston pyöriä sellaisella alustalla, joka tukee ratkaisun tarvitsemia tekniikoita. [4]

ReSRC.it on kolmannen osapuolen pilvipalvelu, joka toimii ikään kuin kuvien välityspalvelimena. Kuvan lähdeosoitteen alkuun lisätään palvelun osoite, jolloin kuvaa ei ladatakaan suoraan näytettäväksi selaimessa, vaan palvelun palveli-

men kautta, jossa se käsitellään etukäteen määritellyin asetuksin sopimaan eri laitteille. Palvelu tunnistaa näytön koon lisäksi myös mm. kaistanleveyden, joka on erityisen hyvä ominaisuus mobiiliselaamista ajatellen. *ReSRC.it* on maksullinen palvelu, mutta siitä on saatavilla määräaikainen kokeiluversio. [4]

2.2.4 Navigointi ja valikot

Kuten edellä syöttötavoista kerrottaessa mainittiin, responsiivisessa sivustossa navigaatio vaatii usein jonkinlaisia muutoksia päätelaitteen näyttökoosta riippuen.

Perinteisessä verkkosivun rakenteessa navigaatio on toteutettu sivun yläosaan asettamalla navigointilinkit vaakariviin tai vaihtoehtoisesti pystysuuntaisesti sivun jompaankumpaan laitaan. Nämä ovat toimivia ratkaisuja kun tietokoneen näytöllä on runsaasti tilaa.

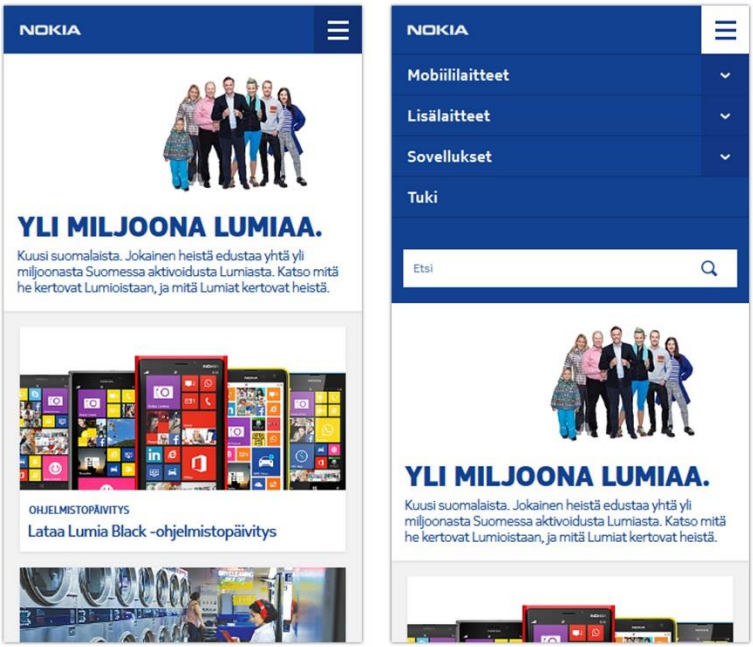
Kun näytön koko pienenee, eivät navigointilinkit enää mahdukaan yhdelle siistille riville. Tässä suhteessa pystysuuntainen navigointi toimii paremmin, koska se sopii kapealle mobiilinäytölle paremmin. Vaakasuuntainen navigaatio kannattaakin järjestää pystysuuntaiseksi, kun näytön koko pienenee.

Pelkkä pystysuuntainen navigaatiovalikko ei tee vielä sivusta mobiilikäytettävyydeltään kovin hyvää. Mobiililaitteille optimoidulla sivulla on tilan käyttöön kiinnitettävä erityistä huomiota. Pystysuuntainen navigaatio vie luonnollisesti paljon tilaa ja varsinkin paljon linkkejä sisältävä navigaatio muun otsakkeen sisällön kera täyttää äkkiä koko näytön sivulle tultaessa jättäen sivun varsinaisen sisällön piiloon.

Tilan maksimoimiseksi varsinaiselle sisällölle navigaatiovalikko voidaankin piilottaa. Mobiilisovellusten tapaan esimerkiksi sivun otsakkeeseen tehdään painike, jota painamalla valikon saa näkyviin.

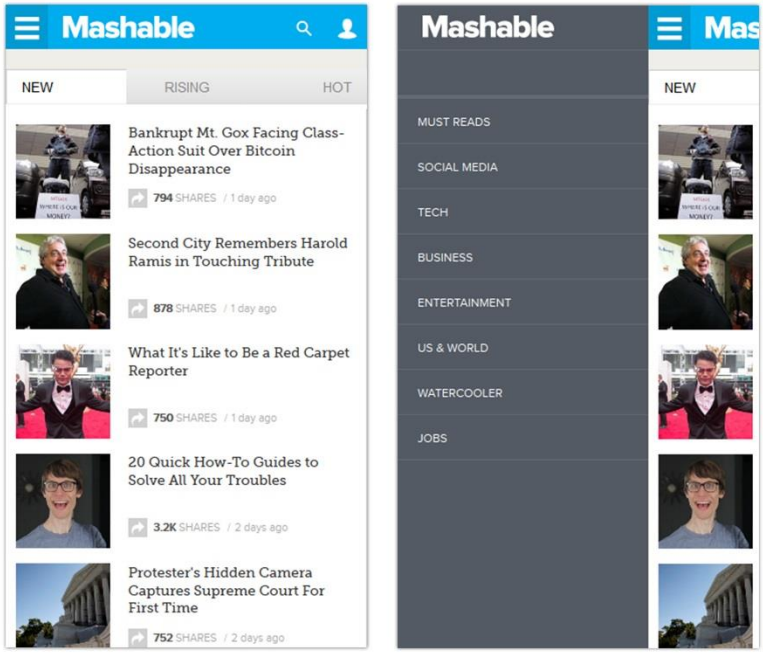
Piilotetun valikon toteuttamiseenkin on useampia tapoja. Tavallisimmin otsakkeessa sijaitsevaa valikkopainiketta painaessa pystysuuntainen valikko liukuu

esiin otsakkeen alle työntäen samalla sisältöä alaspäin (Kuva 2). Vaihtoehtoisesti valikko voi painiketta painamalla aueta muun sisällön päälle.



Kuva 2. Valikon toteutus mobiilinäkymässä [6].

Tällä hetkellä suosiossa on mm. Facebookin mobiilisovelluksessaan tutuksi tekemä *off canvas* -navigaatio. Tällainen sivulta esiin liukuva valikko toimii hyvin varsinkin isoissa ja monitasoisissa navigaatioissa. (Kuva 3) Se on myös kätevä paljon sisältöä sisältävissä sivuissa, kuten blogi-sivustoilla. Kun valikko on aina liu'utettavissa esiin, ei kävijän tarvitse rullata sivun ylälaitaan päästäkseen näkemään valikon.



Kuva 3. Mashable uutisportaalin off canvas -navigaatio [7].

2.2.5 *Mobile First* -ajattelu

Aikaisemmin verkkosivuja luettiin pääosin pöytä- tai kannettavilla tietokoneilla ja sivut myös suunniteltiin sen mukaan. Kun ensimmäisiä erillisiä mobiililaitteille suunniteltuja versioita alettiin tekemään, ne olivat yleensä sivun työpöytänäkymän karsittuja versioita. Responsiivisessa suunnittelussa aloitetaan myös usein täyden koon sivusta ja työstetään sen pohjalta pienempiin näyttöihin soveltuvat näkymät.

Mobile first -ajattelu kääntää suunnittelun lähtökohdat toisinpäin. *Mobile first* -ajattelun peruslähtökohta on, että halutaan tarjota kaikille sama ydinsisältö päätelaitteesta riippumatta. Suunniteltaessa ensin pienelle mobiilinäytölle ja mm. datan käytön rajoitukset huomioon ottaen, tulee pakostikin karsittua kaikki epäoleellinen pois. [8]

Kun mobiilinäkymä ydinsisältöineen on suunniteltu, voidaan jatkaa suurempiin näyttökokoihin. *Media query* -kutsuilla voidaan tehdä muutoksia rakenteeseen tai näyttää lisäsisältöä, kun näytön leveys ylittää kutsuun asetetun minimiarvon.

3 TEKNIIKAT

3.1 HTML5 ja CSS3

HTML5:llä tarkoitetaan yksinkertaisimmillaan HTML-kielen uutta versiota. Termiä käytetään kuitenkin joskus puhuttaessa monista muista uusista verkkosivujen ja -sovellusten tekniikoista, joilla ei varsinaisesti ole mitään tekemistä HTML-kielen kanssa. HTML5 voidaankin pitää yleisnimityksenä työlle HTML:n laajentamiseksi ja HTML-dokumentteihin liittyvien toimintojen määrittelemiseksi. [9]

HTML5 pitää sisällään paljon uudistuksia ja muutoksia merkintäkieleen. Yksi näkyvimmistä on tapa jäsennellä dokumenttia uusilla semanttisilla rakenne-elementeillä. Aikaisemmin sivun jäsentelyyn on käytetty yleisesti `div`-elementtiä, joka nimetään tai luokitellaan käyttötarkoitusta varten, esim. `<div id="header">…</div>` tai `<div class="nav">…</div>`. HTML5:ssä näille löytyy omat merkitystään kuvaavat elementit, kuten ylätunniste: `<header>…</header>` ja navigointiosa `<nav>…</nav>`. Muita semanttisia rakenne-elementtejä ovat mm. artikkelityyppinen sivun osa `article`, alatunniste `footer` ja sivun rakenteellista osaakokonaisuutta merkitsevä `section`. [9]

HTML-dokumentissa merkitään vain verkkosivun sisältö ja rakenne. Verkkosivun ulkoasullinen muotoilu toteutetaan CSS-tyyliohjeilla. CSS3:ksi kutsutaan CSS-tyyliohjekielen uusinta kolmatta versiota, joka tuo mukanaan tukun uusia ominaisuuksia. Näitä ovat esimerkiksi elementtien animointi sekä muuttaminen 2D- ja 3D-tilassa. Uusilla ominaisuuksilla verkkosivuista voidaan tehdä entistä näyttävämpiä ja dynaamisempia. CSS3:n määrittely on vielä kesken, mutta monet uusista ominaisuuksista ovat jo tuettuja moderneissa selaimissa. [9]

Vanhempia, mutta vielä laajasti käytettyjä selaimia varten on monissa CSS3:n mahdollistavissa toiminnoissa käytettävä esimerkiksi *JavaScriptia*.

3.2 JavaScript

JavaScript on komentosarjakieli, joka on yleisimmin käytössä ohjelmoitaessa internetselaimia suorittamaan haluttuja toimintoja. [9]

JavaScript komennot lisätään upotettuna HTML-dokumenttiin `script`-elementin sisään tai vaihtoehtoisesti omaan JavaScript-tiedostoon, johon viitataan HTML-dokumentissa.

Upotettu:

```
<script>
    omaa koodia..
</script>
```

viittaus omaan tiedostoon:

```
<script src=”omakoodi.js"></script>
```

jQuery [10]

jQuery on kompakti *JavaScript*-kirjasto, joka on kehitetty yksinkertaistamaan ja nopeuttamaan selainpuolen koodausta HTML-sivuilla. Sen avulla tapahtumakäsittely, animaatiot sekä *Ajax* voidaan tehdä yksinkertaisemmin, mikä nopeuttaa ja helpottaa kehitystyötä. *jQuery* on avoimen lähdekoodin projekti ja sen käyttäminen on kaikille ilmaista. *jQuery*-tuki löytyy käytännössä kaikista yleisesti käytössä olevissa selaimista.

jQuery otetaan käyttöön viittaamalla *JavaScript*-kirjastoon HTML-dokumentissa. Tämä tehdään `script`-elementillä seuraavasti:

```
<script src="http://code.jquery.com/jquery-latest.js"></script>
```

Viittaus voidaan tehdä joko paikallisessa hakemistossa olevaan *JavaScript*-tiedostoon tai CDN-ylläpidettyyn tiedostoon. Mm. Google ja Microsoft ylläpitävät ja tarjoavat omaa CDN-kopiotaan *jQuerysta*.

3.3 Selain- ja laitetuki

Uusimmat verkkotekniikat, kuten HTML5 ja CSS3 ovat jo melko hyvin tuettuja eri selainten uusimmissa versioissa. Varsinkin mobiililaitteiden selaimissa tuki on varsin kattava. Vielä merkittävä osa internetin käytöstä kuitenkin tapahtuu selaimilla tai selainversioilla, joista ei tukea kaikille uusimmille tekniikoille löydy tai tuki on toteutettu vain osittain. Esimerkiksi verkkoliikennettä tilastoivan StatCounterin mukaan yli 4 vuotta vanhan Internet Explorer 8:n suosio on toki pitkään laskenut, mutta yltää yhä Suomessa muutaman prosentin markkinaosuuteen [11]. Maailmanlaajuisesti IE8 saavuttaa StatCounterin mukaan yli 6 % markkinaosuuden [12] ja toisen tilastoja tarjoavan tahon Net Applicationsin mukaan jopa yli 21 % maailmanlaajuisen markkinaosuuden [13]. Pitkään jatkunutta suosiota selittää mm. se, että monet käyttävät selaamiseen tietokoneensa käyttöjärjestelmään valmiiksi asennettua selainta ja esimerkiksi yrityksissä hyvin suosittuun Windows XP -käyttöjärjestelmään ja sitä vanhempiin Windows-versioihin ei ole saatavilla Internet Explorerista kahdeksatta versiota uudempaa. IE8:n tuki uusimmille verkkotekniikoille on lähes olematon, joten uusimpia tekniikoita käytettäessä on varauduttava tekemään erilaisia varmistustoimienpiteitä, jotta sivut toimivat myös näillä vanhemmilla selaimilla.

Selaimen tuki erilaisille ominaisuuksille voidaan testata esimerkiksi tähän tarkoitukseen luoduilla *JavaScript*-kirjastoilla. Yksi suosituimmista ja kattavimmista tällaisista kirjastoista on *Modernizr*, jota käyttävät sivustoillaan mm. Googlen, Microsoftin ja Twitterin kaltaiset suuret internettoimijat [14]. *Modernizr* ladataan halutuilla tarkistuskomponenteilla samalle palvelimelle sivujen kanssa ja siihen viitataan sivun `head`-elementissä. Tämän jälkeen voidaan esimerkiksi *JavaScript*-koodissa testata tukea halutuille ominaisuuksille ja tarvittaessa tehdä toimenpiteitä, esimerkiksi luoda varasisältöä, jos tukea ei löydy.

Varasisällön lisäksi monia uusia ominaisuuksia voidaan toteuttaa vaihtoehtoisilla tavoilla, mm. hyödyntämällä *JavaScript*ia. Yksinkertaisimmillaan esim. HTML5:n uudet semanttiset rakenne-elementit voidaan opettaa vanhoille selaimille *JavaScript*illa. Tällä tavoin ei voida lisätä toiminnallisuutta esim. `video`-

elementille, mutta se saa vanhat selaimet tunnistamaan uudet elementit CSS-muotoilussa ja *JavaScript*-koodissa. [9]

Jokaisen tarvittavan elementin voi luoda *JavaScript*illa erikseen seuraavasti:

```
<script>
   document.createElement('header');
   document.createElement('footer');
</script>
```

Tämä on kuitenkin työlästä, jos käytössä on useampia eri elementtejä. Lisäksi uusien elementtien oletusmuotoilut on määriteltävä itse CSS-tyyleissä. Saatavilla onkin valmiita *JavaScript*-kirjastoja, kuten *Html5shiv*, joka luo uudet HTML5-elementit ja lisää niille oletusmuotoilut. Tällöin elementit käyttäytyvät samoin kuin moderneilla selaimilla [15]. *Html5shiv* on saatavilla myös *Modernizr*-kirjaston mukana.

Lähes kaikille muillekin uusille HTML5:n ja CSS3:n ominaisuuksille on tehty tällaisia *polyfill*eiksi kutsuttuja komentosarjoja, joilla ”ei tuettuja” ominaisuuksia pystytään käyttämään myös vanhoilla selaimilla. [16]

3.4 PHP

PHP on ohjelmointikieli, jota käytetään laajasti dynaamisten verkkosivujen luontiin. PHP perustuu avoimeen lähdekoodiin ja kielen syntaksi on saanut vaikutteita muun muassa C, Java ja Perl -kielistä. Se sopii erityisen hyvin verkkosisällön kehitykseen, koska sitä voidaan kirjoittaa upotettuna suoraan HTML-sivun sisälle. PHP eroaa esimerkiksi *JavaScript*ista siinä, että koodi suoritetaan selaimen sijasta palvelimella ja tulos lähetetään HTML-dokumenttina selaimelle. [17]

PHP on laajasti käytössä myös julkaisu- ja sisällönhallintajärjestelmissä. Suosituimmista avoimen lähdekoodin julkaisujärjestelmistä monet, kuten esimerkiksi WordPress, Drupal ja Joomla! käyttävät PHP-ympäristöä [18]. Myös testijärjestelmänä käytetty Concrete5 on rakennettu PHP:llä [19].

Tarvittaessa PHP:llä voitaisiin tehdä paljonkin erilaisia toiminnallisuuksia sivustoille, mutta teemapohjan sovittamiseen julkaisujärjestelmän teemaksi riittää tiettyjen järjestelmäkohtaisten PHP-komentojen omaksuminen ja käyttäminen teemassa.

4 TEEMAPOHJAN SUUNNITTELU

Verkkosivua tai -sivustoa suunniteltaessa lähtökohtia ovat usein sisältö, eli mitä sivulla esitetään, sekä graafinen ulkoasu, eli miltä sivu näyttää. Toisinaan on tiedossa mitä sivuilla halutaan esittää ja sen ympärille aletaan rakentaa sivua ulkoasuineen, joskus ulkoasu suunnitellaan ennen, kuin sisältö tarkentuu. Teemapohja taas ei ole valmis verkkosivu, vaan perusta tai runko, jonka päälle sivut rakennetaan. Lähtökohtina ovat siis toimiva rakenne ja vaivaton muunneltavuus.

4.1 Sivun perusrakenne

Tyypillisesti verkkosivut noudattelevat rakenteeltaan seuraavanlaista kaavaa (Kuva 4): Yläreunassa on *header* eli otsake, josta löytyvät logo sekä navigointielementit. Otsakkeen alla on suurehko otsakekuva tai ns. *feature*-alue, joka on alue, jossa voidaan nostaa esille tärkeimmät aiheet sivustolla, kuten uutissivuston pääuutiset tai yrityksen päätuotteet. Näiden alla on varsinainen sisältöalue, joka on jaettu rinnakkain oleviin pääsisältöalueeseen sekä sivupalkkiin. Alimpana sivulla on *footer* eli alatunniste, joka sisältää yleensä tietoja sivun julkaisijasta.

Kuva 4. Tyypillinen verkkosivun rakenne.

Verkkosuunnittelussa ei ole olemassa varsinaisesti sääntöjä, joten mikään ei estä tekemästä verkkosivusta täysin edellä kuvatusta poikkeavaa. Käyttäjät ovat kuitenkin oppineet tietynlaiseen sivurakenteeseen, joten usein on järkevää lähteä liikkeelle niin sanotusti tutusta ja turvallisesta. Varsinkin tässä tapauksessa, kun ei tehdä valmista sivua, vaan useiden sivujen tuotannossa käytettävää sivupohjaa.

Kun sivun rakennetta aletaan suunnittelemaan, on hyvä aloittaa rautalankamallista. Rautalankamalli on joko käsin tai tietokoneella tehty yksinkertainen visuaalinen esitys, josta käy ilmi miten elementit asettuvat sivulle [20]. Kuten teemapohjan rautalankamallista (Kuva 5) näkee, sen perusasettelu noudattelee tyypillistä verkkosivujen asettelua otsakkeineen, sisältöalueineen ja alatunnisteineen. Ulkoasullisesti valmis teemapohja on lähes yhtä yksinkertainen kuin rautalankamalli, mutta pitää sisällään monenlaista.

Kuva 5. Teemapohjan rautalankamalli.

Teemapohjassa käytetään HTML5 määrittelyn mukaista merkintää. Tähän päädyttiin siksi, että HTML5 on kieli, johon verkkokehitys tulee keskittymään tulevaisuudessa ja melko kattavan selaintuen myötä se on mielestäni tarpeeksi käypä tulevaisuuden lisäksi myös tänä päivänä. Toki vanhojen, HTML5:tä tukemattomien selainten käyttö on vielä paikoin yleistä, mutta suurimmaksi osaksi tätä voidaan paikata *polyfill*-paikkauksilla. Jatkoa ajatellen on järkevämpää tehdä heti tulevaisuuden tekniikoilla ja aikanaan poistaa turhiksi jääneitä paikkauksia, kuin käyttää nyt kaikille selaimille soveltuvaa merkkausta ja myöhemmin päivittää koko teemapohja uusimpiin tekniikoihin.

Sivun merkkaus noudattelee tavanomaista HTML-dokumenttia. Se aloitetaan dokumenttityypin ilmoituksella, joka on HTML5-dokumentissa yksinkertainen:

```
<!DOCTYPE html>
```

Sivun `head`-elementti on tavanomainen ja siitä löytyvät mm. viittaukset tyyli- ja *JavaScript*-tiedostoihin. `Head`-elementin pakolliset osat, kuten `title` ja merkistö-

koodauksen määrittelevä `meta`-sisältö generoidaan tyypillisesti julkaisujärjestelmän kutsulla.

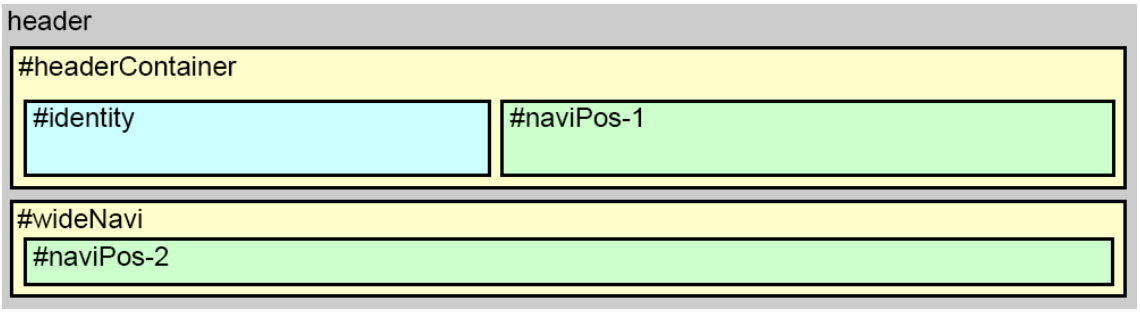
HTML-dokumentin varsinainen sisältö ja rakenne löytyvät `body`-elementin sisältä. Teemapohjan sivurakenne on yksinkertaistetusti seuraavanlainen:

```
<header role="banner" class="wrapper">
    <div id="headerContainer" class="container">
        Header content...
    </div>
</header>
<section id="first" class="wrapper">
    <div id="firstContainer" class="container">
        Section content...
    </div>
</section>
<footer class="wrapper">
    <div id="footerContainer" class="container">
        Footer content...
    </div>
</footer>
```

Sivurakenne jakautuu ylä- ja alatunnisteisiin sekä `section`-rakenneosiin. Näille osille on annettu luokka *wrapper*, joka on määritelty koko sivun levyiseksi. *wrapper*-osien sisällä on `div`-elementit, joiden sisään kunkin osan sisältö tulee. Elementtien luokka on *container* ja luokalle määritelty leveys on koko sivun sisällöllisten osien kokonaisleveys. Seuraavassa tarkastellaan hieman lähemmin eri rakenneosien sisältöä.

4.1.1 Otsake

Sivun otsake pysyy yleensä samanlaisena läpi sivuston. Otsakkeen merkintään käytetään `header`-elementtiä ja sen sisällä on paikat mm. sivuston tunnistetiedoille, sekä navigoinnille (Kuva 6). Ylätunnisteessa voisi olla muutakin toiminnallista sisältöä kuten kielivalinta ja hakupalkki.



Kuva 6. Ylätunnisteen rakenne.

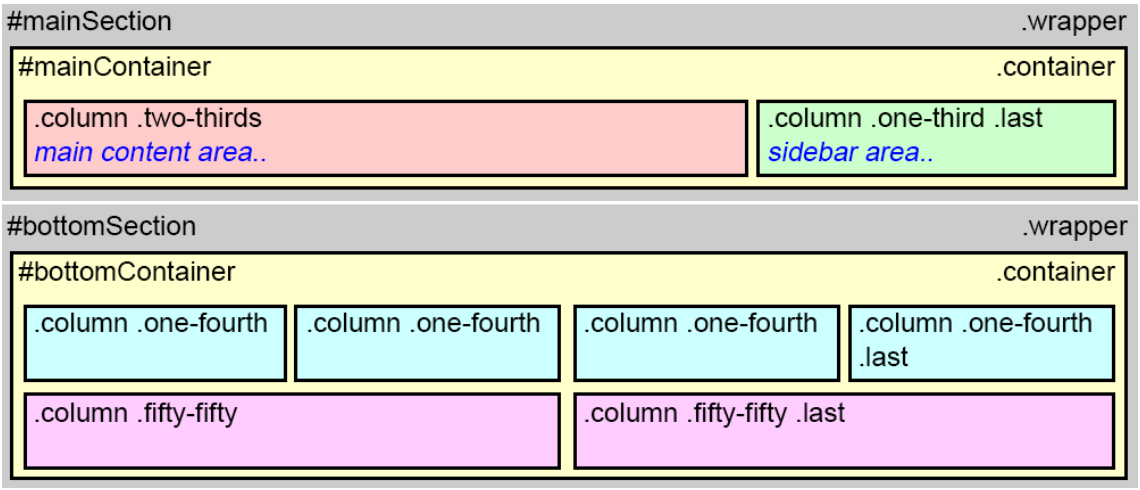
Tunnistetiedot on sijoitettu otsakkeen vasempaan reunaan `div`-elementtiin, jolle on määritelty id-attribuutti *identity*. Elementti voi sisältää esimerkiksi yrityksen nimen, logon tai koko sivustoa kuvaavan otsikon.

Navigaatiolle on varattu kaksi vaihtoehtoista paikkaa. Jos sivustolla on vähän sivuja, voidaan päänavigaatiopalkki sijoittaa *identity*-elementin rinnalle ylätunnisteen oikeaan laitaan. Mikäli sivuja on paljon, navigaatiopalkki voidaan sijoittaa *identity*-elementin alapuolelle koko sisältöalueen levyisenä. Paikkoja voi myös käyttää yhdistetysti esimerkiksi niin, että päänavigaatio on tunnistetietojen vierellä ja alasivujen navigaatiopalkki tunnistetietojen alapuolella.

Navigaatiolle varattuja paikkoja voi käyttää melko vapaasti myös muunlaisen sisällön näyttämiseen. Niihin voi sijoittaa esimerkiksi jo mainitut hakutoiminnon ja kieliversioiden valinnan tai jotain helposti huomattavaksi tarkoitettua sisältöä, kuten puhelinnumeron tai muita yhteystietoja.

4.1.2 Sisältöalue

Sivun sisältöalue koostuu tarpeen mukaan yhdestä tai useammasta `section`-elementistä, joiden sisältöä ja rakennetta voidaan muuttaa käyttötarkoituksen mukaan. Perusrakenteeltaan ne ovat kuitenkin samankaltaisia (Kuva 7) ja noudattelevat löyhästi *grid-layout*-asettelua.



Kuva 7. Viitteellinen havainnekuva sisältöalueiden rakenteesta.

`section`-elementin sisällä uloimpana on *container*-alue, joka pitää sisällään varsinaiset sisältöelementit. Kaikille sisältöelementeille on määritelty *column-luokka* eli sarake ja lisäksi sarakkeen leveyden määrittelevä luokka. Leveysluokkia on kuusi: *fifty-fifty*, *one-third*, *two-thirds*, *one-fourth*, *three-fourths* sekä *full*. Kuten luokkien nimistä huomaa, ne kertovat suoraan, kuinka monta osaa sarakkeen leveys on *container*-elementin leveydestä. Sarakkeita käytetään riveittäin siten, että rivistä tulee aina täysi. Esimerkiksi Kuvassa 7 näkyvällä tavalla rivissä voi olla kaksi *fifty-fifty*-saraketta, neljä *one-fourth*-saraketta tai kaksi eri levyistä saraketta, *one-third* ja *two-thirds*. Lisäksi kunkin rivin viimeiselle sarakkeelle määritellään *last*- luokka, jonka avulla sarakkeet saadaan tasattua *container*-elementin oikeaan laitaan. Sarakkeiden ominaisuuksista on kerrottu enemmän luvussa 4.2.

4.1.3 Alatunniste

Alatunniste merkitään `footer`-elementillä. Se noudattelee rakenteeltaan sisältöalueita. Myös footer sisältää *container*-alueen, johon sisältöä voidaan jäsennellä *column*-sisältöelementeillä.

4.2 Skaalautuvuus eri päätelaitteille

Työn lähtökohdista responsiivisuus ja sen onnistunut toteuttaminen nousevat luultavasti tärkeimmiksi osa-alueiksi.

Jotta responsiivisuus toimii kuten halutaan, on sivun lähdekoodiin lisättävä *viewport*-`meta`-elementti. Monissa mobiiliselaimissa on ominaisuus, jonka johdosta verkkosivut tulostetaan moninkertaisessa koossa ja skaalataan näytölle sopivaksi, jolloin selaimen piirtämät pikselit ovat pienempiä kuin näytön oikeat fyysiset pikselit. Tämä sopii kiinteän asettelun sivuille, jolloin sivu näkyy kokonaisuudessaan ja sitä voidaan tarpeen vaatiessa zoomata esimerkiksi tekstiä luettaessa. Responsiivinen verkkosivu on kuitenkin tarkoitettu näkymään niin, että selaimen piirtämät pikselit ovat täsmälleen näytön fyysisten pikselien kokoisia. Tämä onnistuu lisäämällä sivun `head`-elementtiin *viewport*-`meta`-elementti ja asettamalla sille leveyden määrittävä arvo

```
<meta name="viewport" content="width=device-width,
initial-scale=1.0, maximum-scale=1.0, user-scalable=no"/>
```

`scale`-arvot määrittelevät sivun zoomauksen oletus- ja raja-arvot sekä sen, onko käyttäjän ylipäätään mahdollista zoomata sivua. Responsiivisessa sivussa ei zoomaukselle pitäisi olla tarvetta, mutta sen voi myös sallia esimerkiksi huononäköisiä käyttäjiä ajatellen.

Kuten edellisistä luvuista on käynyt ilmi, sivu perusrakenteeltaan noudattelee löyhästi *grid-layout*-asettelua. Jotta siitä saadaan joustava, elementtien leveydet määritellään CSS-tyyleillä prosentuaalisiksi.

Sivun rakenteellisten, `header`, `footer` ja `section`-osien, joille on määritelty luokaksi *wrapper,* halutaan olevan koko dokumentin levyisiä. Niinpä *wrapper*-luokalle määritellään leveys CSS:llä seuraavasti:

```
.wrapper {
    width: 100%;
}
```

Myös *wrapper*-elementeissä sisältöä ympäröivä *container*-luokan `div`-elementti määritellään koko dokumentin levyiseksi. Sille määritellään lisäksi maksimileveys, koska isolla työpöytänäytöllä esimerkiksi koko näytölle leviävä sivu ei välttämättä näytä hyvältä ja tekstikappaleiden luettavuus voi kärsiä, jos rivit ovat liian leveitä. Maksimileveydellä sisältöalueesta saadaan sopivan levyinen myös isoilla näytöillä. Vasemman ja oikean marginaalin asettaminen automaattiseksi jättää container-alueen molemmille puolille yhtä paljon tilaa, jolloin se asettuu ikkunan keskelle:

```
.container {
    width: 100%;
    max-width: 1180px;
    margin-left: auto;
    margin-right: auto;
}
```

Varsinaisille sisältöalueille on siis määritelty *column*-luokka sekä leveyden määrittelevä luokka. Leveysluokille on määritelty luokkaa vastaava prosentuaalinen leveys, jossa on huomioitu kaikille *column*-sarakkeille määritelty oikeanpuoleinen marginaali. Alla olevasta esimerkistä voi huomata, että leveyksien ja marginaalien yhteenlaskettu summa on 102 % eikä 100 %, tämän takia rivin viimeiselle sarakkeelle annetaan luokaksi *last*, joka poistaa siltä oikeanpuoleisen marginaalin.

```
.column {
    display: block;
    float: left;
    margin-right: 2%;
}
.one-third {
    width: 32%;
}
.two-thirds {
    width: 66%
}
.last {
```

```
    margin-left: 0;
}
```

Lähes kaikki rakenteelliset elementit noudattavat vastaavanlaista kaavaa, joka tekee sivun rakenteesta joustavan ja asettelu pysyy samanlaisena selainikkunan tai näytön koosta riippumatta.

4.2.1 *Media query* -kutsut ja *breakpoint*-kohdat

Joustava perusrakenne on edellytys responsiiviselle sivustolle, mutta käytettävyyden kannalta eri näyttökoot vaativat kuitenkin asettelun mukautuvuutta. Tämä onnistuu *media query* -kutsuilla.

Mediakutsujen täysi hyödyntäminen vaatii, että *breakpointien* paikat ovat järkevästi suunniteltuja. Useimmiten käytetään laajalti käytössä olevien laitteiden näytön resoluutioita.

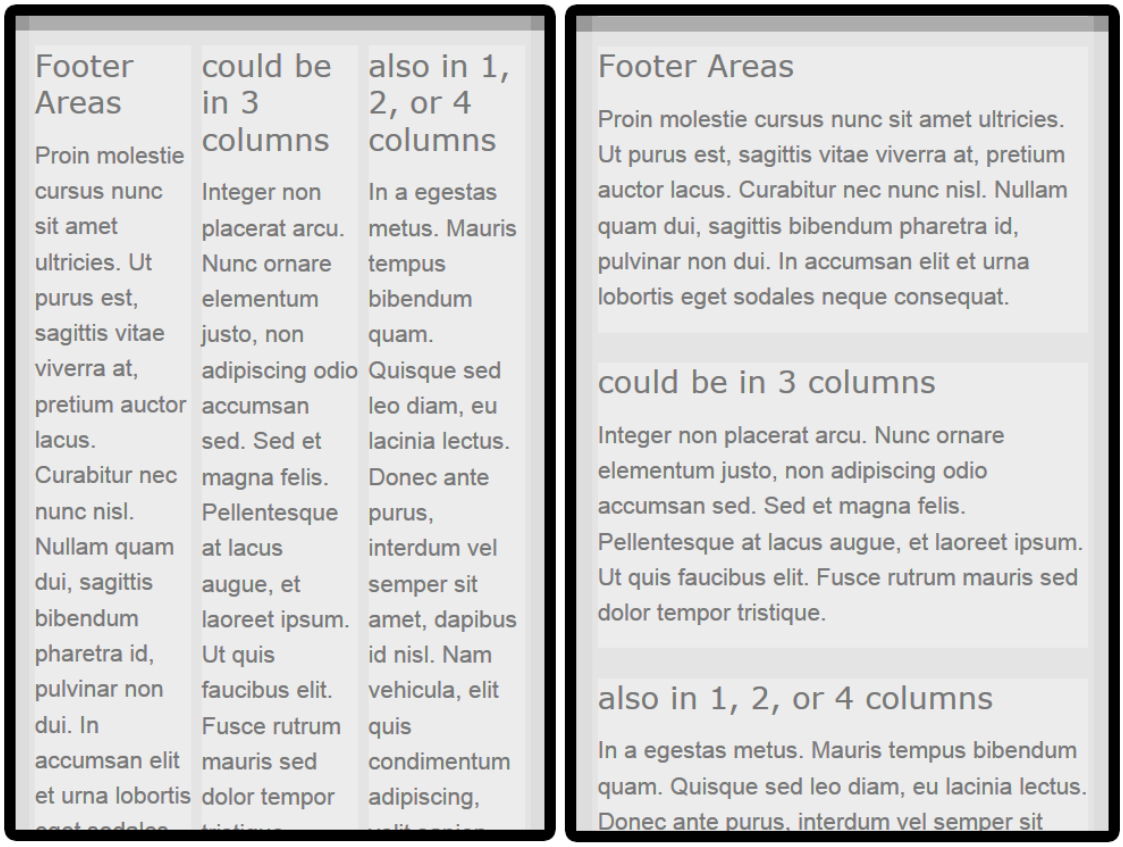
Lähdettäessä leveimmästä liikkeelle teemapohjassa ensimmäinen mediakutsu astuu voimaan, kun näytön tai selainikkunan leveys on pienempi kuin 1 220 pikseliä, joka on hieman suurempi kuin *container*-alueen leveys. Tällöin astuvat voimaan muun muassa seuraavat css-tyylit:

```
@media screen and (max-width: 1220px) {
    .container {
        width: 95%;
        padding: 0 .5%;
        margin: 0 2%;
    }
}
```

Juuri ennen kuin *container* olisi koko ikkunan levyinen, mediakutsu astuu voimaan ja asettaa *container*-elementin leveyden marginaalien kanssa 100 %:iin. Käytännössä nämä asetukset ovat voimassa esimerkiksi monilla miniläppäreillä, joillakin vanhemmilla tietokoneilla sekä tablet-koneilla, joiden näytön resoluutio on 1 024 × 768 pikseliä.

Ehkä tärkein *breakpoint* on se leveys, jota pienemmille näytöille sisältö esitetään pienelle näytölle optimoidusti. Teemapohjaan täksi leveydeksi valikoitui 768 pikseliä hyvin pitkälle markkinoilla olevien mobiililaitteiden näyttökokojen perusteella. Yleisesti ottaen tablet-koneiden resoluutiot ovat pienimmillään sellaisia, että lyhemmän sivun pikselileveys on juuri 768 pikseliä, joten tablet-koneet näyttävät työpöytänäkymää. Nykyään älypuhelintenkin näytön resoluutio on jo samaa luokkaa tablet- ja pc-koneiden kanssa. Tämä voisi aiheuttaa sen, että mobiililaitteella näkyykin suurelle näytölle optimoitu näkymä. Koska suuri pikselitiheys aiheuttaisi sen, että normaali leipäteksti ja ikonit jäisivät hyvin pieniksi, mobiiliselaimissa on ominaisuus, joka saa ne käyttäytymään esimerkiksi juuri mediakutsuja kohtaan kuin pienemmän resoluution laitteet. Tästä syystä voidaan hyvällä syyllä luottaa siihen, että niiden laitteiden resoluutio jää alle tuon 768 pikselin, joiden halutaan näyttävän pienille näytöille optimoitua näkymää.

Sivun asettelu isoilla näytöillä perustuu vierekkäisiin sarakkeisiin. Näkymän mennessä tarpeeksi kapeaksi, esimerkiksi kolmen tai neljän vierekkäisen sarakkeen tekstit rivittyisivät maksimissaan parin sanan riveiksi, jolloin luettavuus kärsii ja asettelu näyttää rikkinäiseltä. Tästä syystä selainikkunan leveyden ollessa alle 768 pikseliä, sarakkeet asettuvat allekkain koko näkymän levyisiksi.



Kuva 8. Mediakutsun vaikutus pienen näytön näkymään.

Kuvasta 8 voidaan todeta, että kun mediakutsu ei ole voimassa, vasemmanpuoleisessa näkymässä sarakkeet käyvät ahtaiksi ja tekstin luettavuus on huono. Oikeanpuoleisessa näkymässä mediakutsu on asettanut sarakkeet allekkain ja luettavuus on hyvä.

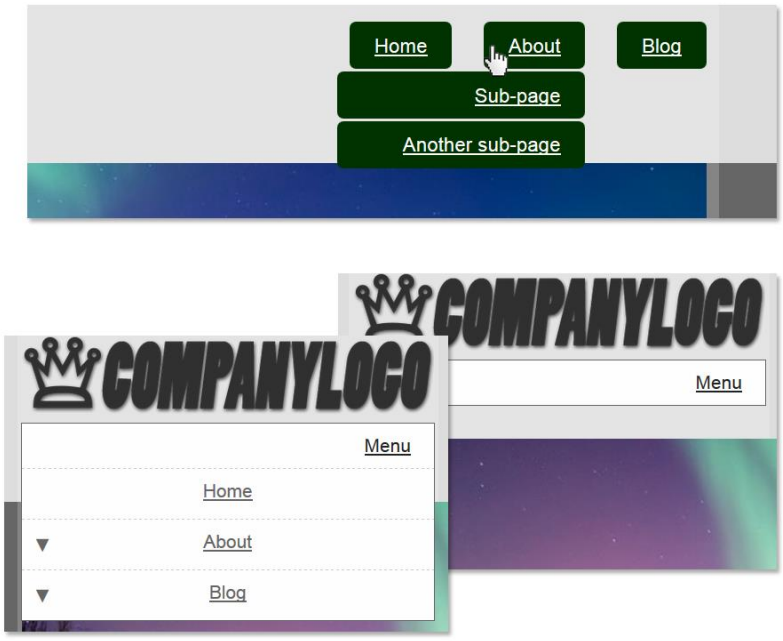
4.2.2 Valikot

Sisältöalueiden uudelleen järjestelyn lisäksi myös navigointivalikoiden asettelu ja ulkonäkö muuttuu riippuen näkymän leveydestä. Ison näytön näkymässä suositaan vaakasuuntaista linkkiriviä, josta alasivut tulevat näkyviin viemällä osoittimen ko. pääsivun päälle. Pienellä näytöllä ei ole tarpeeksi tilaa samankaltaiseen toteutukseen. Lisäksi pienen näytön näkymän ollessa kyseessä selaaminen tapahtuu oletettavasti kosketusnäytöllä, jolloin muun muassa linkki-

painikkeiden koossa täytyy huomioida sormen kosketusalueen koko verrattuna tarkkaan hiiren osoittimeen.

Pienen näytön näkymässä navigointivalikolle on varattava huomattavan paljon tilaa, tällöin hyväksi havaittu tapa on piilottaa valikko niin, että se tulee näkyviin painiketta painamalla.

Yhdessä teemapohjan valikkototeutuksessa (Kuva 9) ison näytön näkymässä navigaatio on toteutettu niin, että ylätunnisteessa on painikkeet pääsivuille. Kun osoittimen vie pääsivun painikkeen päälle, tulevat kyseisen sivun alasivut allekkain näkyviin.



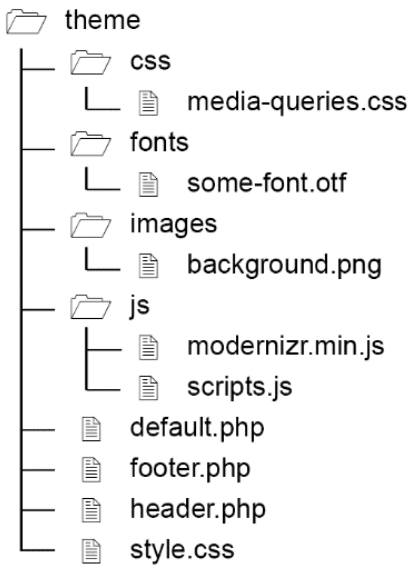
Kuva 9. Navigaatio ison ja pienen näytön näkymissä.

Pienen näytön näkymässä valikko on piilotettu ja se liukuu näkyviin *Menu*-painiketta painamalla. Kunkin sivun alasivut tulevat näkyviin valikon vasemmassa laidassa olevia nuolia painamalla.

4.3 Tiedosto- ja hakemistorakenteiden suunnittelu

Kaikki verkkosivuston sivut eivät välttämättä noudata samaa asettelua, näin ollen julkaisujärjestelmän teemaan voi usein tehdä useampia *templateja*, eli sivupohjia. Yleensä julkaisujärjestelmiä käytettäessä myöskään yksittäinen verkkosivu ei ole yhtä kuin yksi html-dokumentti vaan sivun eri osat on pilkottu omiin tiedostoihinsa. Yleisimmin erillisinä tiedostoina ovat ainakin otsake ja alatunniste, jotka toistuvat samanlaisina sivulta toiselle. Lisäksi julkaisujärjestelmästä riippuen esimerkiksi sivupalkki saattaa olla omana tiedostonaan. Näitä sivun osia kutsutaan sivupohjassa. Tällaisen modulaarisen rakenteen etuja on muun muassa se, että samaa sivun osaa ei tarvitse kirjoittaa jokaiseen sivupohjaan uudelleen ja niiden muokkaaminen on mahdollista toisistaan riippumatta.

Kuvassa 10 esitetty teemapohjan tiedosto- ja kansiorakenne on pyritty tekemään sellaiseksi, että se olisi mahdollisimman helppo siirtää ja muokata erilaisiin julkaisujärjestelmiin. Hakemiston juuresta löytyvät sivupohjat, niistä erotellut sivun osat sekä ensisijainen CSS-tyylitiedosto *style.css*. Alahakemistoihin on järjestelty kaikki muut sivulla tarvittavat tiedostot kuten fontit, kuvat sekä JavaScript-tiedostot. Muut CSS-tiedostot, esimerkiksi responsiivisuuteen käytettävät *media query* -kutsut ovat myös selvyyden vuoksi omassa hakemistossaan.



Kuva 10. Teemapohjan tiedosto- ja hakemistorakenne.

Sivun osista on erotettu omiin tiedostoihinsa ylätunniste *header.php* ja alatunniste *footer.php*. Sivupohjia on lähtökohtaisesti yksi *default.php*, joka pitää sisällään kaikki mahdolliset sisältöalueet. Ideana on, että siitä voidaan julkaisujärjestelmään sovittamisen yhteydessä muokata erilaisia sivupohjia eri käyttötarkoituksiin.

4.4 Kustomoitavuus

Koska teemapohjaa tullaan käyttämään monien valmiiden sivustojen pohjana, on siitä pystyttävä helposti muokkaamaan erilaisia variaatioita.

Luultavasti yksinkertaisin tapa muuttaa sivun ulkoasua on värimaailman ja taustojen muuttaminen. Tästä syystä sivun perusrakenne jaettiin `section`-osiin. Näillä osilla on CCS-luokka *wrapper*, joka on koko näkymän levyinen. *Wrapper*-elementin sisällä on *container*-alue, joka ikään kuin kehystää sisällön ja keskittää sen näkymän keskelle. Jokaisen *wrapper*-osan ja *container*-alueen taustakuvaa tai -väriä voidaan muuttaa erikseen, joka antaa lukemattoman määrä erilaisia vaihtoehtoja sisältöalueiden ja sivun osien erottamiseen toisistaan. Perinteinen toteutus olisi ollut se, että sivulla on yksi kehyselementti, joka rajaa sivun sisältöalueen leveyden. Tällöin tietyn sivun osan taustaa, ei pystyisi niin helposti muuttamaan koko sivun leveydeltä.

Myös julkaisujärjestelmille tyypillinen modulaarisuus antaa mahdollisuuden siihen, että sivun osista, esimerkiksi navigaatiovalikosta, voidaan tehdä useampia vaihtoehtoisia toteutustapoja, joista sitten valitaan kunkin sivuston rakenteeseen ja ulkoasuun sopiva vaihtoehto.

5 SOVITUS CONCRETE5-JÄRJESTELMÄÄN

Teemapohjan sovitusta ja testaamista varten tehtiin sisäiseen käyttöön tarkoitettu testisivusto. Sivusto tehtiin Concrete5-julkaisujärjestelmälle. Julkaisujärjestelmäksi valittiin Concrete5, koska se todettiin potentiaaliseksi vaihtoehdoksi yleisimmin käytössä oleville avoimen lähdekoodin järjestelmille, kuten WordPress, Drupal tai Joomla.

Testisivusto käsittää etusivun lisäksi muutaman alasivun, joihin luotiin yleisesti asiakkaiden sivuilla esiintyvää sisältöä, esimerkiksi tekstiä, kuvia ja yhteydenottolomakkeita.

5.1 Concrete5-julkaisujärjestelmä

Concrete5 on avoimen lähdekoodin sisällönhallinta- ja julkaisujärjestelmä. Sen perustana on vuodesta 2003 kehitetty Concrete CMS, jonka 5. versio julkaistiin vapaana ohjelmistona MIT-lisenssillä vuoden 2008 tienoilla. [21]

Concrete5 on PHP ja MySQL-pohjainen sovellus, joka pohjautuu Zend Framework -sovellusalustaan. Käyttöliittymässä on hyödynnetty *jQuery* ja *jQuery UI* -kirjastoja. [22]

Concrete5 eroaa monista muista julkaisujärjestelmistä muun muassa siinä, että kaiken tyyppinen sisältö koostuu palikoista, joita voi asettaa haluamallaan tavalla ennalta määrättyihin sisältöalueisiin.

Järjestelmä on saatavilla useilla eri kielillä, myös suomeksi. Kielet asennetaan erillisinä kielipaketteina ja eri käyttäjille voidaan asettaa erikieliset käyttöliittymät.

5.2 Järjestelmän rakenne

Concrete5 pohjautuu MVC-arkkitehtuuriin, joka tekee järjestelmän muokkaamisesta ja laajentamisesta yksinkertaista ja tehokasta.

MVC on ohjelmistoarkkitehtuurityyli, jota käytetään esim. graafisten käyttöliittymien suunnittelussa. Ohjelman osat on eritelty kolmeen osaan: *Model* eli malli kuvaa järjestelmän tietoa ja sen käsittelyä, *View* eli näkymä määrittää ulkoasun ja miten tieto esitetään, *Controller* eli ohjain käsittelee näkymästä tulleet komennot ja muuttaa mallia ja näkymää niiden perusteella. [23]

Kun järjestelmän osat on eroteltu niin, että ulkoasu, logiikka ja tiedonkäsittely ovat muokattavissa lähestulkoon toisistaan riippumatta, voidaan muutoksia tai laajennuksia tehdä täsmällisesti määrättyyn osaan järjestelmässä.

Hakemistorakenne on monikerroksinen niin, että kaikki järjestelmän osat voidaan ohittaa muokatuilla tiedostoilla. Järjestelmän ydinosat ja sisäänrakennetut osat sijaitsevat *concrete*-hakemistossa järjesteltyinä omiin hakemistoihinsa. Tämä hakemisto on ikään kuin puhdas versio, johon ei tehdä muutoksia. Sen sijaan juurihakemistosta löytyy sama hakemistorakenne kuin *concrete*-hakemistosta, jonne voidaan tallentaa muokattu versio mistä tahansa *concrete*-hakemiston sisältämästä tiedostosta. [24]

Concrete5:een voidaan asentaa muokkauksia ja laajennuksia paketteina. Paketti sisältää tarvittavat tiedostot samanlaisissa hakemistopoluissa, kuin ne olisivat järjestelmän juuressa tai *concrete*-hakemistossa. Lisäksi paketti sisältää *controller.php*-tiedoston, johon on määritelty mitä järjestelmän osia paketin mukana asennetaan. [25]

5.3 Teemapohjan muokkaaminen Concrete5-teemaksi

Koska teemapohja on jo valmiiksi pilkottu teemaan sopivaan muotoon, ei sen muuttaminen *Concrete5*-teemaksi vaadi paljon. Hakemistorakennetta muutetaan hieman vastaamaan järjestelmän vaatimaa. Concrete5:ssa *header.php* ja

footer.php -tiedostot eivät ole teeman juuressa, vaan ne sijoitetaan omaan *elements*-hakemistoon. Lisäksi teeman juureen lisätään teemaa kuvaava pienoiskuva, *thumbnail.png* sekä tekstitiedosto *description.txt*, johon tulee kaksi riviä: ensimmäiselle riville kirjoitetaan teeman nimi ja toiselle riville teemaa kuvaava lause. Muuten hakemistorakenne toimii sellaisenaan.

Itse tiedostoihin tehdään myös järjestelmäkohtaiset lisäykset. Kaikki Concrete5:n PHP-tiedostot aloitetaan aina rivillä

```
<?php  defined('C5_EXECUTE') or die("Access Denied."); ?>
```

joka estää tiedoston käyttämisen järjestelmän ulkopuolella [23].

5.4 Teeman asennus ja käyttöönotto

Ennen asennusta teemasta tehdään paketti. Se helpottaa ja nopeuttaa teeman ja siihen liittyvien muiden osien, esimerkiksi tiettyihin lohkoihin tehtyjen muutosten asentamista. Teema ja siihen liittyvät muut osat voitaisiin myös sijoittaa omiin polkuihinsa järjestelmän hakemistopuussa ja asentaa erikseen manuaalisesti. Paketointi nopeuttaa asennusta ja *controller*-tiedoston asennuskomennot huolehtivat, että kaikki tarvittavat osat tulevat asennettua. Paketointi helpottaa myös valmiin teemapohjan jakelua edelleen käyttöön asiakasprojekteissa sekä mahdollisten korjausten ja lisäysten päivittämistä projekteihin jälkikäteen.

Paketti voi siis sisältää mitä vain järjestelmän osia. Tässä tapauksessa paketista löytyy itse teema ja muutoksia muutamiin lohkoihin, muun muassa valikkoihin käytettävään *Autonav*-lohkoon sekä *Page List* -lohkoon, jolla voidaan tehdä esim. blogi-kirjoitusten listauksia sivustolle.

Paketti siirretään FTP-yhteydellä palvelimelle, järjestelmän *packages*-hakemistoon. Kun paketti on palvelimella, se tulee näkyviin järjestelmän selainkäyttöliittymässä, josta se voidaan asentaa. Paketin asentaminen asentaa teeman ja muut sen mukana tulevat osat järjestelmään.

Kun paketti on asennettu, teema otetaan käyttöön järjestelmän selainkäyttöliittymän Teemat-osiosta.

6 YHTEENVETO

Työn tavoitteena oli tutustua responsiiviseen verkkosuunnitteluun sekä suunnitella ja toteuttaa responsiivinen teemapohja, jonka voisi pienellä vaivalla sovittaa yleisimmin käytössä oleviin julkaisujärjestelmiin. Lisäksi teemapohja oli määrä sovittaa teemaksi testialustaksi valitulle Concrete5-julkaisujärjestelmälle.

Projektia aloitettaessa responsiivinen verkkosuunnittelu terminä ja perusteet siihen olivat jo tuttuja. Toimivan teemapohjan suunnittelu ja toteutus vaativat kuitenkin selkeästi syvempää paneutumista aiheeseen. Projektin aikana tulikin vastaan paljon hyödyllistä tietoa responsiivisesta suunnittelusta ja verkkosuunnittelusta yleensä. Esimerkiksi hyviä käytäntöjä tutkittaessa mieleen jäi monia toimintatapoja ja työkaluja hyvään ja tehokkaaseen verkkosuunnitteluun. Ei pelkästään puhtaan teknisiä asioita, vaan myös esimerkiksi käytettävyyteen liittyviä asioita, joita voidaan varmasti hyödyntää myös tulevissa projekteissa.

Teemapohjan responsiivisesta rakenteesta itsessään tuli melko toimiva. Rakenteen eri osat käyttäytyvät erikokoisilla päätelaitteilla odotetusti ja pohjan muokattavuus kunkin asiakasprojektin tarpeisiin otettiin hyvin huomioon.

Testisivustolle pohjaa sovitettaessa kuitenkin huomattiin, että teemapohjassa on vielä melko paljon viimeisteltävää. Jotta sitä voitaisiin käyttää tuotantokäytössä, olisi viimeistelyyn ja testaamiseen käytettävä vielä runsaasti työtunteja. Samalla todettiin, ettei resurssien käyttäminen pohjan viimeistelyyn ole tällä tuotantotahdilla järkevää.

Lähes järjestelmälle kuin järjestelmälle on olemassa täysin niille kehitettyjä teemoja, joiden muokkaaminen eri projektien tarpeiden mukaisiksi on tässä tilanteessa kustannustehokkaampaa kuin oman teemapohjan viimeistely tuotantokuntoon.

Ala menee eteenpäin hurjaa vauhtia ja jopa projektin aikana osa menetelmistä tai tekniikoista sai uusia, entistä tuoreempia ja parempia vaihtoehtoja. Responsiivinen suunnittelu on tällä hetkellä valtavan paljon enemmän, kuin se oli muu-

tama vuosi sitten Macrotten esitellessä erikokoisille näytöille soveltuvan ratkaisunsa verkkosivun rakenteesta. Responsiivinen verkkosuunnittelu on tällä hetkellä vallalla oleva trendi, mutta kuka tietää onko se sitä vielä muutaman vuoden päästä. Verkkosuunnittelun ydin on ennen kaikkea sitä, että on kyettävä oppimaan uutta ja mukautumaan ihmisten tapaan käyttää internetiä sekä pyrkiä tekemään internetin käyttökokemuksesta ihmisille hyvä.

LÄHTEET

[1] Knight K., Smashing Magazine, *Responsive Web Design: What It Is and How To Use It*. [www-dokumentti]. Saatavilla: http://coding.smashingmagazine.com/2011/01/12/guidelines-for-responsive-web-design/ (Luettu: 13.4.2013).

[2] Macrotte E., *Responsive Web Design*, New York: A Book Apart, 2011, 150 s.

[3] Anthony T, Smashing Magazine, *Finger-Friendly Design: Ideal Mobile Touchscreen Target Sizes*. [www-dokumentti]. Saatavilla: http://uxdesign.smashingmagazine.com/2012/02/21/finger-friendly-design-ideal-mobile-touchscreen-target-sizes/ (Luettu 15.3.2014).

[4] Alexander S., Smashing Magazine, *Choosing a Responsive Image Solution.* [www-dokumentti]. Saatavilla: http://mobile.smashingmagazine.com/2013/07/08/choosing-a-responsive-image-solution/ (Luettu: 15.3.2014).

[5] Nokia. [www-dokumentti]. Saatavilla: http://www.nokia.com (Luettu 2.3.2014).

[6] Mashable. [www-dokumentti]. Saatavilla: http://www.mashable.com, (Luettu 2.3.2014)

[7] W3C, The srcset attribute. [www-dokumentti]. Saatavilla: http://www.w3.org/TR/html-srcset/ (Luettu 23.3.2014)

[8] Wroblewski L., *Mobile First*, New York: A Book Apart, 2011, 130 s.

[9] Korpela, J.K., *HTML5 - Uudet ominaisuudet*, Jyväskylä: WSOYpro Oy, 2011, 328 s.

[10] The jQuery Foundation. [www.dokumentti]. Saatavilla: http://jquery.com/ (Luettu 23.3.2014).

[11] StatCounter Global Stats. [www-dokumentti]. Saatavilla: http://gs.statcounter.com/#browser_version_partially_combined-FI-monthly-201303-201402 (Luettu: 23.3.2014).

[12] StatCounter Global Stats. [www-dokumentti]. Saatavilla: http://gs.statcounter.com/#browser_version_partially_combined-ww-monthly-201303-201402 (Luettu: 23.3.2014).

[13] Netmarketshare. [www-dokumentti]. Saatavilla: http://www.netmarketshare.com/ (Luettu 23.3.2014).

[14] Modernizr. [www-dokumentti]. Saatavilla: http://modernizr.com/ (Luettu: 3.5.2013).

[15] HTML5shiv. [www-dokumentti]. Saatavilla: http://code.google.com/p/html5shiv/ (Luettu: 3.5.2013)

[16] Modernizr, *HTML5 Cross Browser Polyfills.* [www-dokumentti]. Saatavilla: https://github.com/Modernizr/Modernizr/wiki/HTML5-Cross-Browser-Polyfills (Luettu: 3.5.2013).

[17] The PHP Group, *PHP Manual.* [www-dokumentti]. Saatavilla: http://www.php.net/manual/en/ (Luettu: 15.5.2013).

[18] Built With, *CMS Usage Statistics*. [www-dokumentti]. Saatavilla: http://trends.builtwith.com/cms (Luettu: 25.3.2014).

[19] Wikipedia, *List of content management systems*. [www-dokumentti]. Saatavilla: http://en.wikipedia.org/wiki/List_of_content_management_systems (Luettu: 19.4.2013).

[20] Vanhala-Nurmi, V., *Rautalankamalli.* [www-dokumentti]. Saatavilla: http://myy.haaga-helia.fi/~vanvu/www/ suunnittelu/rautalankamalli.html (Luettu 18.4.2014).

[21] Concrete5 Suomi, *Mikä on Concrete5*. [www-dokumentti]. Saatavilla: http://www.concrete5.fi/concrete5/mikae-on-concrete5/ (Luettu: 14.4.2013).

[22] Tolvanen, P. Vierityspalkki.fi, *Esittelyssä: Concrete5-julkaisujärjestelmä*. [www-dokumentti]. Saatavilla: http://vierityspalkki.fi/2012/12/10/concrete5-julkaisujarjestelma/ (Luettu: 19.4.2013).

[23] Phillips, O., *MVC in Concrete5* [www.dokumentti]. Saatavilla: http://www.eantics.co.uk/whats-fresh/mvc-in-concrete5/ (Luettu: 22.3.2014).

[24] Concrete5 Suomi, *Järjestelmätiedostojen ylikirjoittaminen*. [www-dokumentti]. Saatavilla: http://www.concrete5.fi/ohjeet/kehittaminen/yleinen-kehitys/ylikirjoittaminen/ (Luettu: 18.4.2014).

[25] Concrete5 Suomi, *Lisäosien kehitys*. [www-dokumentti]. Saatavilla: http://www.concrete5.fi/ohjeet/kehittaminen/lisaeosien-kehitys/ (Luettu: 18.4.204).