Tuukka Hakala

SPEKSIYÖ-TAPAHTUMAN WWW-SIVUT

Viestinnän koulutusohjelma

2014

samk
Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

SPEKSIYÖ-TAPAHTUMAN WWW-SIVUT

Hakala, Tuukka
Satakunnan ammattikorkeakoulu
Viestinnän koulutusohjelma
Helmikuu 2014
Ohjaaja: Kuusinen, Jere
Sivumäärä: 68
Liitteitä: 0

Asiasanat: www-sivu, verkkosivusto, Drupal, responsiivinen

Tämän opinnäytetyön aiheena oli verkkosivuston toteuttaminen Porin ylioppilasteatteri ry:n järjestämää SpeksiYö-tapahtumaa varten. Opinnäytetyön toiminnallinen osa oli graafisen suunnittelijan toteuttaman ulkoasun muuntaminen toimivaksi verkkosivustokokonaisuudeksi hyödyntäen erilaisia ohjelmointi- ja merkintäkieliä. Sivusto suunniteltiin ja toteutettiin responsiiviseksi.

Yhteyshenkilönä toimi alusta asti Porin ylioppilasteatterin jäsen Julia Hannula, joka myös suunnitteli sivuston ulkoasun ja ideoi erilaiset toiminnot suunnittelutasolla. Hän myös toimitti kaiken sivustolle tarkoitetun materiaalin minulle. Tekstimateriaali tuotettiin kokonaisuudessaan asiakkaan toimesta.

Sivustosta tuli moderni kokonaisuus, jossa ei ole ollenkaan alasivuja. Päädyimme kyseiseen ratkaisuun jo projektin alkuvaiheessa, sillä sivustolle ei ollut missään vaiheessa tarkoitus tuottaa kovinkaan paljon tekstiä. One page -ratkaisu mahdollistaa sivuston nopean selaamisen ja tärkeimpien tietojen löytymisen jo ensimmäisellä vilkaisulla, eikä pakota käyttäjää etsimään haluamaansa sisältöä monelta alasivulta.

WEBSITE FOR SPEKSIYÖ EVENT

Hakala, Tuukka
Satakunnan ammattikorkeakoulu, Satakunta University of Applied Sciences
Degree Programme in Media and Communications
February 2014
Supervisor: Kuusinen, Jere
Number of pages: 68
Appendices: 0

Keywords: website, Drupal, responsive

The subject of this thesis was *creating a website for SpeksiYö*, an event organized by Porin ylioppilasteatteri. The functional part of the thesis was creating a website based on the layout designed by a graphic designer by using different programming and markup languages. The site was also designed and created to work responsively.

The contact person was Julia Hannula, a member of Porin ylioppilasteatteri, who also designed the layout and some of the functionalities on the site. She also provided me with all the material that was intended for the site. All the text material was created by the client.

The site turned out as a modern entity that had no subpages at all. We ended up with this solution already in the early stages of the project, since the site had never been intended to contain a large amount of text. One Page -solution enables the user to quickly browse through the site and provides the most important information easily on the first glance, instead of forcing the user to search for it from many subpages.

SISÄLLYS

1 JOHDANTO

Viestinnän koulutusohjelmassa opinnäytetyöhön liittyy usein jokin käytännön työ eli toiminnallinen osa, jonka tekemisen prosessia kuvataan opinnäytetyöraportissa. Oman opinnäytetyöni toiminnallinen osa oli verkkosivuston tekninen toteuttaminen Porin ylioppilasteatteri ry:n järjestämää SpeksiYö-tapahtumaa varten. Sivusto julkaistiin 2.1.2014 ja tapahtuma järjestettiin 7.–8.2.2014.

Sivuston ulkoasun suunnitteli Satakunnan ammattikorkeakoulusta valmistunut graafinen suunnittelija ja Porin ylioppilasteatteri ry:n jäsen Julia Hannula, joka toimi myös yhteyshenkilönä koko projektin ajan. En osallistunut visuaaliseen suunnitteluprosessiin ollenkaan, mutta jaoimme Hannulan kanssa ideoita keskenämme projektin eri vaiheissa. Sivuston sisältö tuotettiin myös kokonaisuudessaan asiakkaan toimesta, joten sain itse keskittyä täysin omaan tehtävääni, eli Hannulan toimittaman ulkoasun ja hänen ideoimiensa toiminnallisuuksien muuntamiseen toimivaksi verkkosivustokokonaisuudeksi hyödyntäen erilaisia ohjelmointi- ja merkintäkieliä.

Päätin rakentaa sivuston Drupal-nimisen sisällönhallintajärjestelmän päälle. Sisällönhallintajärjestelmälle ei olisi välttämättä ollut tarvetta, sillä tapahtumaa varten tehdyn sivuston elinikä on melko lyhyt ja sivuston päivitys on vähäistä. Päädyin kuitenkin varmuuden vuoksi käyttämään sisällönhallintajärjestelmää. Sivusto haluttiin toteuttaa responsiivisena, eli sen toimivuus ja käytettävyys haluttiin varmistaa kaikilla mahdollisilla resoluutioilla ja laitteilla.

Sivuston rakenne noudattaa one page -mallia, eli erillisiä alasivuja ei ole, vaan kaikki sisältö on jaettu yhdelle sivulle. Tämä mahdollistaa nopean silmäilyyn perustuvan selailun ja auttaa käyttäjää löytämään hakemansa tiedon nopeasti.

Tässä raportissa kerron projektin eri vaiheista ja tarkennan muun muassa responsiivisen suunnittelun ja one page -mallin tarkoituksia ja vahvuuksia. Perustelen käyttämäni tekniikat ja ratkaisut johdonmukaisesti. Lopuksi arvioin prosessia kokonaisuutena ja sen merkitystä oman ammattitaitoni kehityksen kannalta.

2 OPINNÄYTETYÖN ALOITTAMINEN

2.1 Aiheen hankinta

Lähtökohtanani opinnäytetyön aiheen hankkimiselle oli halu toteuttaa jonkinlainen verkkosivusto oikealle asiakkaalle. Tämä lähtökohta oli luontainen ja helppo valinta, sillä olen työskennellyt web-suunnittelijana/-kehittäjänä ja opiskellut erilaisia ohjelmointi- ja merkintäkieliä itsenäisesti vuosien ajan. Graafisen alan opinnoistani on myös ehdoton hyöty verkkosivustoprojekteissa, sillä se vähentää graafisen suunnittelijan työmäärää teknisessä toteutusvaiheessa huomattavasti ja antaa itselleni muun muassa mahdollisuuden kerätä Photoshop-tiedostosta eri elementit haluamallani tavalla. Pystyn myös tekemään ulkoasuun mahdollisia korjauksia jälkeenpäin.

Työskentelin opinnäytetyöni aloittamisen aikaan porilaisessa Viestintä- & mainostoimisto Kumppaniassa ja kerroin muutamille työkavereilleni alustavia suunnitelmiani opinnäytetyöni osalta. Kumppaniassa työskentelevä graafinen suunnittelija Julia Hannula ehdotti, että tekisin verkkosivuston Porin ylioppilasteatterin järjestämää SpeksiYö-tapahtumaa varten. Hän on itse kyseisen yhdistyksen jäsen ja kertoi suunnittelevansa verkkosivuston ulkoasun.

Halusin tarttua Hannulan ehdotukseen useista syistä. Ensinnäkin valmistumiseni oli jo kertaalleen viivästynyt, joten halusin saada opinnäytetyötäni varten aiheen, joka varmasti toteutuisi. Projektin aikataulu ei olisi voinut venyä, koska tapahtumaan oli kolme kuukautta aikaa ja sivuston piti olla julkaistuna jo huomattavasti ennen sitä. Näiden seikkojen lisäksi projektin laajuus oli mielestäni juuri sopiva; ei liikaa tehtävää eikä liian laajalta aihealueelta, mutta kuitenkin riittävästi toiminnallista tekemistä opinnäytetyön vaatimuksia ajatellen. Tiesin myös, että tämä tulee olemaan projekti, josta voisin myöhemmin olla ylpeä.

2.2 Projektin rajaus

Ulkoasun lisäksi varsinainen sisältö SpeksiYö-tapahtuman verkkosivustolle tuotettiin asiakkaan toimesta, joten en itse osallistunut käytettävien kuvien valintaan tai tekstisisällön tuottamiseen. Minun osuuteni oli muuntaa visuaalinen suunnitelma toimivaksi verkkosivustoksi, joka toimii yhdessä sisällönhallintajärjestelmän kanssa, sekä ennakkoon toimitettujen materiaalien syöttäminen sivustolle. Tämän lisäksi testasin sivuston toimivuuden eri laitteilla ja selaimilla ja hoidin teknisen toteutuksen vaatimat juoksevat asiat, kuten sivuston domainin ja palvelintilan varaukset. Toteutustapojen suhteen sain täysin vapaat kädet.

3 PROJEKTIN LÄHTÖKOHDAT

3.1 Porin ylioppilasteatteri

Porin ylioppilasteatteri perustettiin vuonna 2009 Porin yliopistokeskuksen opiskelijoiden toimesta. Perustajissa oli mukana jo ennestään kokeneita teatterintekijöitä, mutta myös teatteritoiminnasta kiinnostuneita aloittelijoita. Yhdistyksen tavoitteena oli yhdistää nämä ihmiset ja tarjota Poriin opiskelemaan tulleille nuorille aikuisille mahdollisuus lähteä mukaan teatteritoimintaan uudessa opiskelukaupungissa.

Teatteriryhmän näytelmät ovat yhdistelmä provosoivia aiheita ja monipuolisia ilmaisumuotoja. Porin ylioppilasteatteri tekee myös yhteistyötä monien satakuntalaisten kulttuuriyhdistysten kanssa. (Porin ylioppilasteatteri 2014.)

3.2 SpeksiYö-tapahtuma

Speksi on interaktiivinen koko illan musiikkinäytelmä, jossa parodioidaan ajankohtaisia ilmiöitä. Speksi käyttää tehokeinona improvisaatiota ja sille on ominaista vuorovaikutteisuus yleisön kanssa (Viikkispeksi 2014). Porin

ylioppilasteatterin järjestämän SpeksiYö-tapahtuman ajankohta oli 7.–8. helmikuuta vuonna 2014. Tapahtuman tavoitteena oli tuoda Poriin speksikulttuuria ja tarjota kokeneemmille speksin ystäville täysin uudenlainen elämys. Tapahtuman pääasiallinen ohjelma sijoittui Kulttuuritehdas Kehräämölle, jossa eri korkeakouluista kerätyt joukkueet ottivat mittaa toisistaan minispeksikilpailussa.

Speksikilpailun lisäksi viikonloppu täyttyi monipuolisesta oheisohjelmasta, joka levittäytyi ympäri kaupunkia. Kulttuuripainotteinen viikonloppu ei ollut suunnattu pelkästään opiskelijoille, vaan tapahtuma oli avoin kaikille halukkaille. Esityksiin oli mahdollista ostaa sekä yksittäisiä lippuja että tapahtumapasseja. (Porin ylioppilasteatteri 2014.)

3.3 Sivuston rooli tapahtuman kannalta

Tapahtumaa varten rakennettava verkkosivusto näytteli suurta osaa tapahtuman yhteydessä järjestetyn minispeksikilpailun onnistumisessa, sillä joukkueet ilmoittautuivat mukaan kilpailuun sivustolla olevan lomakkeen kautta. Tapahtuman luonteen vuoksi ilmoittautuneet joukkueet olivat tietysti tärkeä osa tapahtuman onnistumista, joten sivustosta on luotava houkutteleva ja sen oli tarjottava tarpeeksi informaatiota mahdollisille osallistujille.

Markkinoinnin suhteen verkkosivusto oli Facebook-sivun ohella ehdottomasti suurin kanava ja panostus tapahtuman näkyvyyden kasvattamiselle. Asianmukaisesti ja uskottavasti toteutettuna verkkosivusto antaisi SpeksiYö-tapahtumasta yleisölle jo etukäteen tapahtuman ansaitseman ammattimaisen ja osaavan kuvan. Sivuston pääasialliset tavoitteet olivat siis ihmisten houkutteleminen mukaan tapahtumaan ja minispeksikilpailuun osallistuvien joukkueiden ilmoitusten lähettäminen ja vastaanottaminen.

3.4 Budjetti ja palveluntarjoajien valinta

Projektille ei sovittu erikseen budjettia, sillä jo alussa puhuimme Hannulan kanssa, että varsinaista korvausta ei teknisen toteutuksen osalta suoriteta. Verkkosivuston

ylläpitäminen vaatii kuitenkin jonkin verran rahallista panostusta, sillä palvelintilat ja domainit eivät ole ilmaisia. Näistä kustannuksista sovimme suullisesti ja hyväksytin kaikki väistämättömät kustannukset Hannulan kautta ennen maksusuorituksia. Nämä kustannukset pitivät sisällään lähinnä domainin ja palvelintilan varausmaksun. Molemmat varaukset tehtiin Webhotelli.fi-palvelun kautta. Kyseiseen palveluntarjoajaan päädyttiin hyvän hinta-laatusuhteen ja omien positiivisten kokemuksieni kautta.

4 SIVUSTON SUUNNITTELU

4.1 Alkuvaiheet

Jokaisen käytännöllisen tuotteen sydämessä – oli tuote sitten verkkosivusto, teollisuuspakkaus, rakennus, tuotantojärjestelmä, huonekalu, työkalu tai mikä tahansa vastaava – on jokin käytännön tarkoitus; tehtävä, joka tuotteen odotetaan suorittavan. Ei riitä, että tuote on kaunis ja että pystyt käyttämään sitä itse, vaan sinun on mietittävä ajaako tuote tarkoituksensa muiden käyttäjien kohdalla, suorittaako se sille annetun tehtävän? Tiivistettynä sinun täytyy miettiä, onko tuote käytännöllinen. (Wax 2014.)

Verkkosivuston kohdalla on mietittävä ainakin seuraavat asiat etukäteen:

- Mikä on sivuston lopullinen tavoite?
- Kuka tai millainen käyttäjäryhmä tulee käyttämään sivustoa?
- Mitä käyttäjät haluavat sivustolla tehdä tai mitä he haluavat löytää sieltä?
- Onko sivuston käyttötapa selkeä?
- Mistä käyttäjät tietävät sen toimivan?
- Ilmoittaako se virheistä?

Ylempänä esitetyistä kysymyksistä kaksi viimeisintä eivät välttämättä ole jokaiselle selkeitä asioita, sillä varsinkaan käyttäjän näkökulmasta niitä ei tule helposti miettineeksi, vaan niitä pidetään itsestäänselvyyksinä. On kuitenkin äärimmäisen

tärkeää, että jo suunnitteluprosessin alussa tiedetään, milloin sivuston täytyy ilmoittaa mahdollisista virheistä, kuten esimerkiksi siitä, että lomaketta ei täytetty oikein ja näin ollen sitä ei lähetetty eteenpäin. Vastaavasti on mietittävä, miten lomakkeen onnistunut lähetys ilmaistaan käyttäjälle, jotta hänelle ei jäisi epävarma olo, vaan hän tietäisi sivuston ja lomakkeen toimineen oikein.

Ennen kuin Hannula alkoi suunnittelemaan sivuston ulkoasua, kävimme suullisesti läpi perusideoita sivuston sisällön ja rakenteen suhteen. Halusimme yhdistää nämä kaksi asiaa mahdollisimman tehokkaasti niin, että ne tukisivat toinen toistaan ja parantaisivat siten lopullista käyttäjäkokemusta. Koska varsinaista sisältöä ei tulisi missään vaiheessa olemaan paljon, päädyimme Hannulan ehdottamaan one page -ratkaisumalliin, joka helpottaisi vähäisen sisällön jäsentämistä kätevästi ja yksinkertaisen näyttävästi yhdelle sivulle.

Tämä selkeä ja helppolukuinen ulkoasun rakenne tulisi auttamaan sivuston lopullisen tavoitteen saavuttamisessa, sillä käyttäjiä ei haluttu eksyttää useille alasivuille, vaan informaatio haluttiin esittää jämäkästi, selkeästi ja nopeasti. Moderni ulkoasu valittiin herättämään mielenkiintoa opiskelijoissa, jotka olivat sivuston pääasiallinen kohderyhmä. Tälle kohderyhmälle tietokoneiden ja erilaisten verkkosivustojen käyttö on varmasti tuttua. Jos kohderyhmä olisi ollut esimerkiksi vanhukset, sivuston ulkoasu ja rakenne olisivat olleet aivan toisenlaiset. (Arch 2014.)

4.2 Aikataulu

Projekti oli määrä aloittaa teknisen toteutuksen osalta joulukuussa 2013 ja sivuston julkaisu ajoitettiin saman kuun loppuun. Aikataulutin oman osuuteni niin, että saatuani sekä valmiin ulkoasun että sivustolle laitettavan sisällön, tarvitsisin noin viikon työajan sivuston saattamiseksi julkaisukuntoon. Hannula toimitti minulle ulkoasun katsottavaksi 5.12.2013, mutta tämä ei vielä sisältänyt kaikkea sivustolle tulevaa sisältöä (Hannula sähköposti 5.12.2013). Tämän ulkoasun avulla pääsin kuitenkin hyvin alkuun ja sivupohjan rakentaminen lähti käyntiin.

Sisällöt minulle toimitti Porin ylioppilasteatterin tuottaja Reetta Turtiainen. Ensimmäiset tekstisisällöt sain 18.12.2013 sähköpostilla, joka sisälsi kaiken muun paitsi ohjelmaosuuden joka oli vielä kesken (Turtiainen sähköposti 18.12.2013). Sivusto päätettiin lopulta julkaista ilman ohjelmaosuutta 2.1.2014 joukkueiden ilmoittautumisen mahdollistamiseksi, ja ohjelma lisättiin sivustolle jälkeenpäin heti sen valmistuttua.

Aina kun kyseessä on tapahtumasivusto, aikataulusta joustaminen on erittäin vaarallista. Varsinkin tässä tapauksessa, kun tapahtuman ajankohta oli vain noin kuukauden verran sivuston julkaisun jälkeen ja sivusto näytteli suhteellisen suurta roolia tapahtuman markkinoinnin suhteen, ei aikataulussa ollut paljonkaan pelivaraa.

4.3 Valitut toteutusmallit

Web Design Talks -sivuston artikkelin mukaan (Rathinam 2014) vuoden 2013 käytetyimpiä trendejä www-suunnittelussa olivat muun muassa minimalismi, responsiivinen suunnittelu, infinite scrolling, parallax-efektit ja erittäin suuret kuvaelementit. Myös SpeksiYö-tapahtuman sivustolla on nähtävissä joidenkin näiden trendien vaikutus. Suurimmaksi tekijäksi täytyy kuitenkin nostaa responsiivinen suunnittelu, joka ei oikeastaan ole edes trendi, vaan kokonaan uusi tapa suunnitella ja toteuttaa verkkosivustoja. Se ei ole tyylisuuntaus tai nopeasti ohimenevä erikoisefekti, joka lisätään jokaiselle sivustolle, vaan paikkansa vakiinnuttanut suunnittelu- ja toteutusmalli.

Hannula halusi hyödyntää sivustolla aiemmin mainitsemaani one page -mallia. Tämä lähestymistapa mahdollistaa sivuston kaiken sisällön järjestämisen selkeästi ja yksinkertaisesti yhdelle sivulle, mikä voi nykyisen tietomäärän ja ihmisten lyhentyneen keskittymiskyvyn vuoksi olla todella tehokas vaihtoehto saavuttaa ihmisten huomio (Gosha 2014). Samainen malli myös tukee suurien kuvien käyttöä ja koko näytön hyödyntämistä sivustolla olevan informaation esittämisessä.

Koska sekä responsiivinen suunnittelu että one page -malli ovat SpeksiYö-tapahtuman sivuston kohdalla avainasemassa ja vahvasti esillä tämänhetkisessä www-suunnittelussa, avaan molempia aiheita vielä hieman yksityiskohtaisemmin.

4.3.1 Responsiivinen suunnittelu ja toteutus

Internet-selailun ympäristö muuttuu vauhdilla, ja viimeisten vuosien aikana se on saattanut muuttua jopa nopeammin kuin web-suunnittelijat ja -kehittäjät olisivat lopulta halunneet. Mobiililaitteiden käyttö verkkosivustojen selaamisessa on kasvanut jatkuvasti ja sen odotetaan ohittavan pöytäkoneilla tapahtuvan surffailun määrän lähivuosien aikana. Myös pelikonsoleissa on nykyään www-selaimet käytettävissä. Tällä hetkellä suunnittelijat siis suunnittelevat sivustoja, joita käytetään hiirellä, näppäimistöllä, peliohjaimella ja kosketusnäytöllä. Näiden kaikkien ohjainlaitteiden käyttötapojen välillä on selkeät eroavaisuudet, ja kyseiset eroavaisuudet onkin otettava huomioon sivuston käytettävyyttä suunniteltaessa. Tämän lisäksi näyttökokoja ja -resoluutioita on satoja erilaisia ja niiden erot ovat huomattavan suuria; esimerkiksi iPhonen näyttö verrattuna 27-tuumaiseen pöytäkoneen näyttöön asettaa selvästi erilaiset puitteet sivuston rakennetta varten. (Marcotte 2014.)

Responsive web design -käsitteellä (ei tarkkaa suomennosta) tarkoitetaankin kokonaisvaltaista suunnitteluprosessia, jonka lopullisena tavoitteena on verkkosivuston mukautuminen erilaisille laitteille, joilla sitä todennäköisesti tullaan selaamaan. Tämä sisältää tietysti graafisen ilmeen mukautumisen laitteen kokoon nähden, mutta myös laitteiden erityisominaisuuksien ja käyttäjälähtöisten tarpeiden huomioimista, kuten kosketusnäytön ja sillä tapahtuvien eleisiin perustuvien vuorovaikutustapojen tai värisokeiden tarpeiden huomioimista (Gustafson 2011, 13).

Tarpeet tällaiselle kokonaisvaltaiselle käytännölle ovat syntyneet viimeisten vuosien aikana, kun erilaisten ja erikokoisten mobiililaitteiden määrä on kasvanut huomattavasti. Laitteiden suuren määrän vuoksi yksittäisiä laitteita tai tietyn kokoista näyttöä ei kuitenkaan voida huomioida erikseen verkkosivuston suunnitteluvaiheessa, vaan sivusto on suunniteltava niin, että se toimii

mahdollisimman hyvin kaikilla laitteilla ja kaikenkokoisilla näytöillä. Tämä prosessi voi tietenkin vaatia kompromisseja ja se ehdottomasti vaatii lisää työtä sekä suunnittelijalta että teknisen osuuden toteuttajalta, mutta hyvän käyttökokemuksen antamiseksi se on välttämätöntä.

Ehkä paras todellinen suomennos termille responsive web design olisi *mukautuva suunnittelu*, viitaten juurikin kyseisen suunnitteluprosessin lopulliseen tavoitteeseen: verkkosivuston ulkoasun ja toimintojen mukauttaminen erilaisten ja erikokoisten laitteiden tarpeita vastaaviksi. Käytän kuitenkin tässä opinnäytetyöni raporttiosuudessa suomalaisten käyttöön vakiintunutta termiä *responsiivinen suunnittelu*.

Kuva 1. Honkajoen kunnan verkkosivusto suunniteltuna responsiivisesti eli niin, että sivusto näyttää hyvältä ja toimii erilaisilla laitteilla ja erikokoisilla näytöillä ilman ulkoasun hajoamista.

SpeksiYö-tapahtuman sivujen kohdalla responsiivinen toteutus on aivan yhtä tärkeä ominaisuus kuin muillakin verkkosivustoilla. Enää ei voidakaan sanoa ”tämän sivuston ei tarvitse olla responsiivinen” tai vastaavasti, että ”juuri tämän sivuston

pitää olla responsiivinen”. Tosiasiassa kaikkien sivustojen tulee olla responsiivisia, sillä käyttäjät ohittavat huonosti toimivat sivustot ja palvelut kokonaan lähes poikkeuksetta (eLab Communications 2014). Vaikka SpeksiYön sivuston kohderyhmänä oli opiskelijat, joista lähes jokaisella on jonkinlainen älypuhelin ja jota he myös käyttävät verkossa liikkumiseen, olisi sivusto toteutettu responsiivisena vaikka oletettuun käyttäjäryhmään kuuluisikin päinvastaisia henkilöitä.

4.3.2 One page -malli

Verkkopalvelu on tehtävä niin houkuttelevaksi ja helpoksi käyttää kuin suinkin mahdollista, varsinkin jos mobiililaitteiden käyttäjät halutaan saada asiakkaiksi. Hyöty on parhaimmillaan molemminpuolinen, kun sivuston käyttäjät löytävät sivustolta vastauksen kysymykseensä ennen kuin edes ehtivät muotoilla tämän kysymyksen (Sinkkonen, Nuutila & Törmä 2009, 17). Tätä ajatusta tukee tällä hetkellä vallitseva eräänlainen minimalistisuutta ihannoiva vaihe; miksi tehdä asioista liian vaikeasti hahmotettavia, jos se ei ole välttämätöntä?

Lukija haluaa myös hahmottaa verkkosivustolla olevan tekstin kohtuullisen nopeasti. Pidemmät tekstit sopivat toki perusteellisemman tiedon etsijöille tai esimerkiksi blogin sisällöksi, mutta sivuston luonne ja kohderyhmä on aina mietittävä tapauskohtaisesti (Kortesuo 2009, 80). Kun teksti pidetään tiiviinä ja asiapitoisena, lukija ei eksy informaation sekaan, vaan hänelle ikään kuin syötetään väkisin se tieto, mikä hänelle halutaankin syöttää; se mikä on tärkeää ja olennaista.

Samoin liian vaikea tai monimutkainen alasivujen käyttö voi ohjata lukijan väärille poluille ja hankaloittaa halutun viestin vastaanottamista (Kortesuo 2009, 80). Jos sivustolla on vähän sisältöä, sitä ei kannata turhaan piilottaa klikkauksien taakse, vaan se on hyvä laittaa esille niin, että se löytyy nopeasti ja tehokkaasti. Joitakin vuosia takaperin oli myös uskomus, että ihmiset eivät halua käyttää vierityspalkkeja ja että kaikki tärkeä sisältö täytyi sijoittaa sivuston ylälaitaan. Tilanne on kuitenkin muuttunut, sillä älypuhelimet ja muut kosketusnäytöllä toimivat laitteet ovat tehneet ylös ja alas rullaamisesta luonnollista, jopa klikkailua helpompaa ja nopeampaa. (UXMyths 2013.)

Tämä on synnyttänyt myös uuden trendin www-suunnittelun kohdalla, one page -mallin. Kaikessa yksinkertaisuudessaan se tarkoittaa sitä, että kaikki sivustolla oleva sisältö on jaettu järjestelmällisesti ja selkeästi yhdelle ainoalle sivulle. Käyttäjä selaa sivua pystysuunnassa – tai joissakin tapauksissa vaakasuunnassakin – ja pelkästään silmäilemällä sisältöä hän on nähnyt kaiken sivulle tuotetun sisällön.

One page -mallin etu onkin sen helppoudessa niin suunnittelijan kuin loppukäyttäjänkin osalta; käyttäjän ei tarvitse klikata välttämättä yhtään linkkiä (vaikka linkkejä voidaankin käyttää helpottamaan tietyn asiakohdan löytämistä), mutta malli helpottaa myös tiedon esittämistä ytimekkäästi ja näyttävästi, jolloin se on huomiota herättävää ja helppolukuista. Kun sisältöä ei jaeta monelle erilliselle alasivulle, käyttäjän huomio saattaa kiinnittyä asiaan jota hän ei välttämättä tullut etsimään, mutta josta hän kiinnostui sen nähtyään.

SpeksiYö-tapahtuman kohdalla one page -malli oli erittäin toimiva ratkaisu. Sisältöä ei ollut paljon, mutta se oli sitäkin arvokkaampaa sekä sivustolla vieraileville henkilöille että tapahtuman järjestäjille. Sisällön jaoin lohkoihin allekkain, ja yksi lohko käsitteli yhden aihealueen sisällön. Kun käyttäjä saapuu sivulle, hän näkee heti ensimmäisestä lohkosta, mistä tapahtumasta oikein on kyse ja milloin se on; kiinnostus on herätetty. Liikkuessaan sivustolla alaspäin hän näkee kehotuksen osallistua mukaan tapahtumaan ja tämän jälkeen varsinaisen speksikilpailun ilmoittautumislomakkeen, jolla voi helposti ilmoittaa joukkueensa mukaan. Alempaa löytyvät vielä ohjelma ja viimeisenä yhteystiedot.

5 VALMISTELUT SIVUSTON RAKENTAMISTA VARTEN

Verkkosivustojen toteuttamista ja sen vaiheita varten ei ole olemassa mitään valmista kaavaa, jota olisi pakko tai välttämättä edes suositeltavaa noudattaa. Jokaisella suunnittelijalla ja kehittäjällä tuntuu olevan omat tavat hoitaa prosessin eri vaiheet, ja lopulta noilla tavoilla ei ole mitään merkitystä, kunhan valmis tuote on tarkasti ja

tehokkaasti toteutettu, se kunnioittaa alkuperäistä ulkoasua ja aikataulu on pitänyt tuotteen julkaisun suhteen paikkansa.

Näitä eri vaiheita, jotka voitaisiin pilkkoa vielä useampiin pienempiinkin vaiheisiin, on suunnittelu- ja toteutusprosessissa huomattava määrä, mutta läheskään kaikki näistä vaiheista eivät näy ulospäin tai ne eivät ole missään nimessä itsestäänselvyyksiä muille kuin niistä vastuussa olevalle henkilölle. Käyn seuraavaksi läpi SpeksiYö-tapahtuman verkkosivuston toteutusprosessin eri vaiheita, keskittyen nimenomaan tekniseen toteutusprosessiin alkaen siitä, kun sain työstettäväkseni valmiin ulkoasun Photoshop-tiedostona. Tarjoan myös teoreettisempaa yleistietoa näihin vaiheisiin liittyvistä asioista.

5.1 Graafisten elementtien kerääminen ulkoasutiedostosta

Monelle on itsestäänselvyys, että mitä enemmän verkkosivusto sisältää kuvatiedostoja, sitä hitaammin se latautuu. Nykyajan selaimet yhdistettynä tarjolla oleviin verkkoyhteyksiin mahdollistavat kuitenkin yllättävänkin suurien kuvatiedostojen käytön ilman häiritsevää muutosta latausajoissa. Toisaalta graafisia elementtejä pystytään nykyään luomaan entistä tehokkaammin pelkästään ohjelmointi- ja merkintäkieliä käyttäen. Jos mennään joitakin vuosia taaksepäin, niin esimerkiksi pyöreiden kulmien lisääminen neliön malliselle elementille ei ollut tehokkaasti mahdollista ilman kuvina toteutettuja pyöristettyjä kulmia. Nykyään se kuitenkin onnistuu helposti ja yksinkertaisesti tavalla, joka nopeuttaa sekä toteutusprosessia että loppukäyttäjälle kohdistuvaa latausaikaa, eikä siihen tarvita ollenkaan kuvatiedostoja.

Kaikkea ei kuitenkaan voi tai ei ole järkevää toteuttaa hyödyntämättä kuvatiedostoja. Itse aloitan aina verkkosivuston rakentamisen keräämällä minulle toimitetusta ulkoasutiedostosta (Photoshop-tiedosto joka sisältää sivustolle suunnitellun ulkoasun) kaikki ne elementit, joita ei ole mahdollista tai järkevää toteuttaa käyttämällä ohjelmointi- tai merkintäkieltä.

Nämä elementit oppii erottamaan lopulta automaattisesti pelkästään tutkailemalla ulkoasua; kehittäjä tietää jo etukäteen, millaiset mahdollisuudet hänellä on toteuttaa mitäkin, ja mikä niistä hänelle tutuista toteutustavoista on paras, tehokkain ja viisain kussakin tilanteessa. Joku toinen kehittäjä voi olla täysin eri mieltä siitä mikä olisi lopulta ollut se viisain tapa, ja joissakin tilanteissa erilaiset toteutustavat voivat tarjota lähes identtisen lopputuloksen, mutta tärkeintä on, että suunnittelija ottaa itse selvää erilaisista mahdollisuuksista ja punnitsee niiden vahvuuksia ja heikkouksia ja tekee oman päätöksensä tietojensa pohjalta.

SpeksiYö-tapahtuman ulkoasusta keräsin seuraavat elementit erillisiksi graafisiksi elementeikseen ja tallensin omiksi tiedostoikseen:

- Logo
- Harmaa ylöspäin osoittava nuoli
- Lila alaspäin osoittava nuoli
- Valkoinen alaspäin osoittava nuoli
- Vaaleanlila alaspäin osoittava nuoli
- Twitter- ja Facebook-nappulat (yhdistettynä yhdeksi tiedostoksi, eli *spriteksi*)
- Ensimmäisen lohkon taustakuva

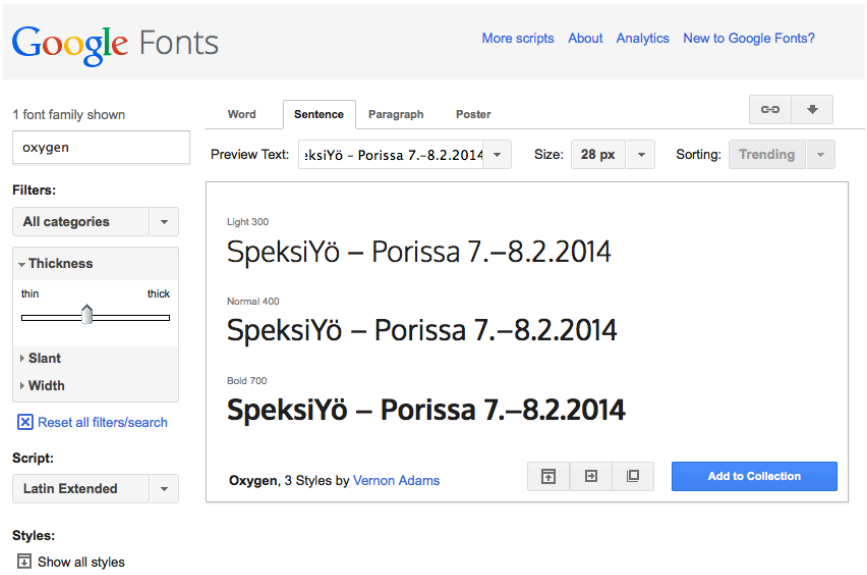
Vaihtoehtoisesti olisin voinut toteuttaa nuolet joko yhdistettynä yhteen tiedostoon spriteksi tai käyttäen HTML5:n mukanaan tuomaa <canvas>-elementtiä, joka mahdollistaa visuaalisten muotojen, kuvioiden, kaavioiden ja muiden vastaavien piirtämisen suoraan HTML-pohjaan esimerkiksi JavaScriptiä hyödyntäen (Dive Into HTML5 2011). Kiireisen aikataulun takia totesin kuitenkin erilliset kuvat parhaaksi vaihtoehdoksi.

Näiden elementtien kerääminen, muokkaaminen ja tallentaminen heti prosessin alkuvaiheessa nopeuttaa omasta mielestäni varsinaisen sivuston rakennusvaihetta. On hyvä tehdä kaikki mahdolliset etukäteisvalmistelut, jotta ohjelmointi- ja rakennusvaiheessa kaikki elementit ovat valmiina käytettäviksi, eikä niitä tarvitse lähteä keräämään ulkoasutiedostosta enää myöhemmässä vaiheessa. Näin kehittäjä pystyy keskittymään yhteen asiaan kerrallaan.

5.2 Typografia ja Google Fonts

Koska en itse suunnitellut ulkoasua, en voinut juurikaan vaikuttaa sivuston typografisiin ratkaisuihin, kuten fontin valintaan. Ulkoasun suunnittelut Hannula hyödynsi Googlen tarjoamaa Fonts-palvelua, joka tarjoaa vapaaseen käyttöön laajan valikoiman verkkosivustoille tarkoitettuja fontteja (Google Fonts 2014). Aikaisemmin, kun vastaavia palveluita ei ollut olemassa, suunnittelijat olivat sidottuja käyttämään sellaisia fontteja, jotka löytyivät oletuksena ainakin käytetyimmistä käyttöjärjestelmistä, sillä fontit eivät latautuneet käyttäjälle selaimen kautta, vaan sivustolla käytetyn fontin täytyi löytyä käyttäjän omalta koneelta valmiina. Tähän on vuosien aikana kehitelty erilaisia kiertoteitä uusien mahdollisuuksien avaamiseksi, mutta Google Fontsin kaltaiset palvelut ovat olleet ensimmäiset todella toimivat ratkaisut.

Google Fontsin kautta suunnittelija voi selata fontteja ja niiden eri leikkauksia ja hakea toimivia yhdistelmiä. Hakutoiminnon avulla voi etsiä fontteja tiettyjen kriteerien mukaan, esimerkiksi kirjaimien leveyden tai tiheyden perusteella. Kaikki fontit ovat tietysti myös ladattavissa palvelun kautta, mikä mahdollistaa niiden käyttämisen omalla tietokoneella suunnitteluvaiheessa. SpeksiYö-tapahtuman kohdalla Hannula oli päätynyt Oxygen-nimiseen fonttiin (kuva 2).



Kuva 2. Oxygen-fontti Google Fontsissa.

Google Fonts antaa todella suorat ja helpot ohjeet siitä, kuinka nämä fontit siirretään käytettäväksi myös lopulliselle verkkosivustolle. Toiminta perustuu siihen, että itse fontit sijaitsevat Googlen omalla palvelimella, josta ne liitetään osaksi verkkosivustoa kutsumalla niitä lyhyellä HTML-merkinnällä, aivan kuten toisella palvelimella sijaitseva kuvakin voitaisiin liittää sivustolle viittaamalla sen sijaintiin tuolla kyseisellä palvelimella (Feldmann 2011). HTML-tiedoston <header>-tagien väliin lisätään SpeksiYön tapauksessa tämä rivi:

<link type="text/css" rel="stylesheet" href="http://fonts.googleapis.com/css?family =Oxygen:400,300,700"></link>

Käytännössä tämä rivi tarkoittaa, että sivustolle liitetään osoitteessa http://fonts.googleapis.com/ sijaitseva lyhyt CSS-tyyliohje, jonka sisältö rajataan osoitteen jälkeen tulevilla lisätiedoilla, jotka viittaavat fontin nimeen ja sen leikkauksiin. Kyseinen Googlen tarjoama CSS-tyyliohje hyödyntää @font-face-ominaisuutta, jonka avulla voidaan määritellä palvelimelle ladatun fontin tiedot. Kehittäjä voi käyttää tätä ominaisuutta vapaasti myös omissa CSS-tyyliohjeissaan. @font-face–ominaisuuden tiedoista täytyy löytyä ainakin halutun fontin nimi sekä tietysti varsinaisen fonttitiedoston sijainti ja nimi. Yksinkertainen @font-face-määritelmä voi näyttää esimerkiksi tällaiselta:

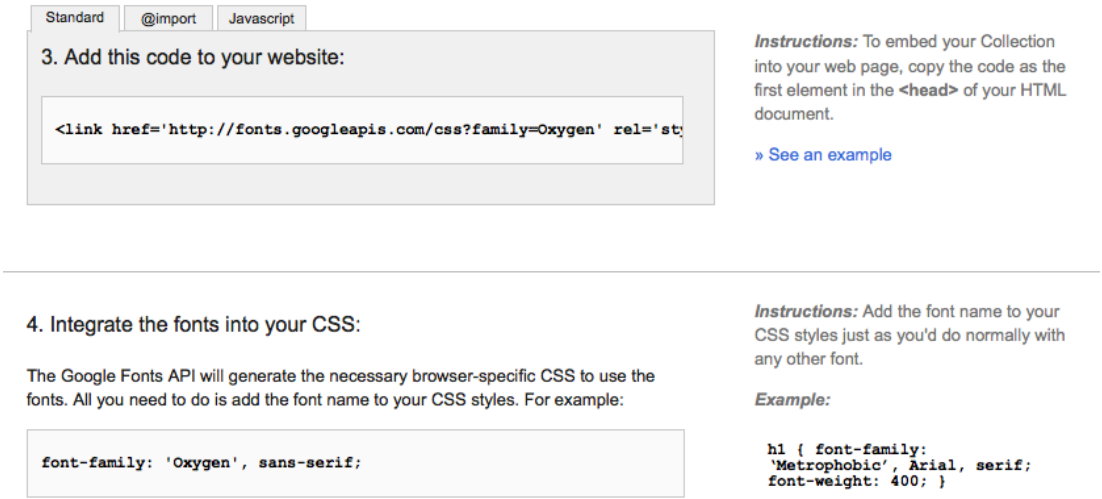
```
@font-face  {
                font-family:"OmaFontti";
                src: url(fonts/omafontti.ttf);
}
```

Ylhäällä olevassa esimerkissä *font-family* määrittää nimen kyseiselle fontille ja *src* ilmoittaa fontin sijainnin palvelimella. Tässä onkin huomion arvoista se, että kehittäjän on myös ladattava fontit palvelimelle itse ja pidettävä huolta, että hänellä on käyttöoikeudet/lisenssit fonttien käyttämiseksi kyseistä tarkoitusta varten. Kun @font-face-ominaisuus on määritelty ja fontti ladattu palvelimelle, kyseistä fonttia voidaan hyödyntää sivuston muissa CSS-määritelmissä helposti viittaamalla sille annettuun nimeen (*font-family*). Jos fonttia halutaan käyttää esimerkiksi kaikissa

sivuston tekstikappaleissa, voidaan kaikille tekstikappale-elementeille antaa seuraava määritelmä:

```
p {
        font-family: ”OmaFontti”;
}
```

Google Fonts on kuitenkin tehnyt tästä koko prosessista nopeamman ja helpomman kehittäjille, sillä heidän ei tarvitse huolehtia @font-face-ominaisuuden käytöstä, fonttien lataamisesta palvelimelle tai lisenssien hoitamisesta. Riittää, että he lisäävät <head>-tagien väliin ylempänäkin kuvatun Googlen palvelimelle ja haluttuun fonttiin viittaavan <link>-tagin ja määrittelevät CSS:n font-family-ominaisuuden avulla kyseisen fontin kaikkiin haluttuihin elementteihin. Google Fonts tarjoaa nämä lisättävät tekstinpätkät automaattisesti (kuva 3), joten ne voi vain kopioida oikeisiin kohtiin; minkäänlaista muokkaamista tai personointia ei tarvitse suorittaa.



Kuva 3. Google Fonts –palvelu tarjoaa HTML- ja CSS-tiedostoihin lisättävät tagit ja määritelmät automaattisesti käyttäjän fonttivalinnan mukaan. Sivulta löytyvät myös selkeät ohjeet näiden käyttämistä varten.

5.3 Kuvat ja värit

SpeksiYö-tapahtuman verkkosivustolla on vain yksi kuva, jonka tarkoitus ei ole funktionaalinen, eli sillä ei ole varsinaista toimintoa kuten ohjailevilla nuolilla tai Facebook-nappulalla on. Jo jonkin aikaa www-suunnittelussa on ollut trendinä käyttää suuria, mahdollisesti jopa koko selaimen ikkunan kokoisia taustakuvia (Rocheleau 2013), ja samaa trendiä noudatetaan myös SpeksiYön sivustolla. Ensimmäisen lohkon taustakuvan ainoa tehtävä on luoda sivustolle tietynlainen tunnelma ja herättää kiinnostusta kävijöissä.

Sivuston visuaalinen ilme välittääkin kahta viestiä. Ensimmäinen on sivuston sisällön esittäminen: informaatio ja käyttäjän mahdollisuudet toimia – ulkoasun täytyy auttaa huomaamaan, jäsentämään ja ymmärtämään asiat, jotka täytyy huomata ja ymmärtää – tätä kutsutaan *visuaaliseksi käytettävyydeksi*. Toinen visuaalisen ilmeen tehtävä on välittää käyttäjälle palveluun, tuotteeseen, yritykseen tai ihmisiin liittyvä brändi, kokonaisilme, tunnelma ja persoonallisuus. Kumpi näistä kahdesta tehtävästä on tärkeämpi, on täysin tapauskohtaista. (Sinkkonen ym. 2009, 242.)

Koska Porin ylioppilasteatterilla on tietty brändi ja persoonallisuus, joka varmasti halutaan siirtää myös sen järjestämiin tapahtumiin, on ainoastaan luonnollista, että tuota persoonallisuutta tuodaan esille myös SpeksiYö-tapahtuman verkkosivustolla. Ison abstraktin kuvan käyttäminen heti alkunäkymässä luo rennon, luonnollisen ja nuorekkaan vaikutelman tapahtuman tunnelmasta.

Koska kyseinen kuva ei sisällä minkäänlaista informaatiota tai minkään osan siitä ei tarvitse olla koko ajan nähtävillä, se myös helpottaa sivuston skaalautuvuutta eri kokoisille näytöille; ei ole väliä pienennetäänkö ja rajataanko taustakuva sopivaksi älypuhelimen pienelle näytölle vai katsotaanko sitä suuremmalta näytöltä, tunnelma säilyy aina samana koosta ja rajauksesta huolimatta.

Värien osalta SpeksiYö-tapahtuman verkkosivustolla käytetään suhteellisen rauhallisia sävyjä tummasta lilasta haaleaan vaaleanpunaiseen. Tehosteväri on kirkas keltainen ja myös valkoinen ja musta ovat käytössä. Näitä kaikkia värejä käytetään tehokkaasti sekaisin ilman mitään varsinaista järjestelmällisyyttä; teksti

saattaa olla yhdessä kohtaa valkoista ja toisessa tummanlilaa. Värien kanssa leikkimällä sivusto on kuitenkin saanut sille ominaisen ilmeen ja tämä ilme tukee sitä rentoa ja nuorekasta tunnelmaa, joka käyttäjälle välittyy jo ylempänä kuvaillun kuvaratkaisun kautta.

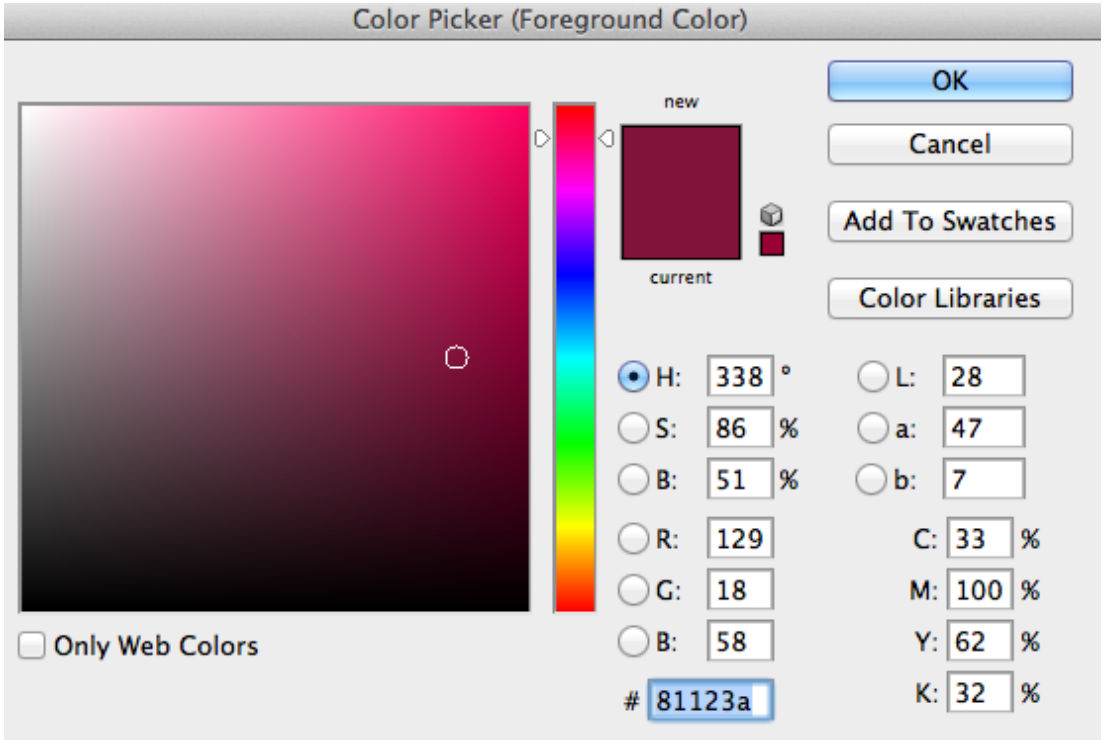
Koska sivuston sisältö on jaettu allekkain oleviin lohkoihin, on nämä lohkot myös eroteltava toisistaan selkeästi, jotta käyttäjä tietää, milloin mikäkin aihe loppuu ja seuraava alkaa. Sivustolla käytössä olevia värejä onkin hyödynnetty tehokkaasti tämän jaottelun selkeyttämiseksi, ja kuvassa 4 näet vasemmalla värit siinä järjestyksessä, kuin ne ovat käytössä lohkojen taustaväreinä.

Kuva 4. SpeksiYö-tapahtuman verkkosivustolla käytetyt päävärit (vasemmalla) ja tehosteväri (oikealla).

Kuvassa 4 näkyvien värilaatikoiden oikealla puolella on kunkin värin HEX-arvo, joka yhdistää punaisen, vihreän ja sinisen eri arvot. Jokaisen värin arvot määritellään kahdella merkillä, jotka voivat olla kirjaimia tai numeroita, joten yhdessä HEX-arvossa on aina yhteensä kuusi merkkiä (Salakapakka 2012). Tässä raportissa en kuitenkaan selvitä sen enempää, miten HEX-arvo määräytyy, mutta kyseinen arvo on

erittäin ratkaisevassa osassa CSS-tyyliohjeita tehtäessä, sillä helpoin tapa määritellä eri elementeille väriarvoja on käyttää nimenomaan HEX-arvoja. Tätä arvoa voidaan hyödyntää elementin taustavärissä, tekstin värissä, reunuksien väreissä ja oikeastaan missä tahansa muussakin tilanteessa, missä halutaan antaa jollekin elementille tai elementin osalle tietty väri.

Nämä värit ja niiden HEX-arvot on helppo kerätä suoraan ulkoasun Photoshop-tiedostosta. Pipettityökalua ja Color Picker -toimintoa hyödyntäen HEX-arvot voi kerätä talteen ja lisätä esimerkiksi CSS-tiedoston alkuun, josta ne on tarvittaessa helppo kopioida ja liittää haluttuun kohtaan (kuva 6).



Kuva 5. HEX-arvon kopioiminen Photoshopin Color Picker –työkalun avulla.

```
55
56  /* ## VÄREJÄ ## */
57
58  /* #81123A; /* TUMMEMPI LILA */
59  /* #91224E; /* VAALEAMPI LILA */
60  /* #EA7979; /* VAALEANPUNAINEN */
61  /* #F7FF10; /* KELTAINEN */
62  /* #C4B8B8; /* HARMAA */
63
```

Kuva 6. HEX-arvot tallennettuna muistiin CSS-tiedostoon. Huomaa, miten vinoviivaa ja asteriskia käyttäen jokainen rivi on niin sanotusti *kommentoitu pois*, eli nämä rivit eivät vaikuta tyylitiedoston toimintaan millään tavalla, vaan ne ohitetaan.

Itselläni on myös tapana jättää jokaisen CSS:n värimääritelmän perään merkintä, mikä väri on kyseessä. Näin ollen näen heti CSS-tiedostoa selaillessani tai muokatessani, mitä väriä missäkin kohtaa on käytetty.

```
168
169     nav#main ul li a:hover,
170     nav#main ul li a:active {
171         color: #EA7979; /* VAALEANPUNAINEN */
172     }
173
```

Kuva 7. Kun värimääritelmän perään jättää merkinnän siitä, mikä väri on kyseessä, on käytettyjen värien hahmottaminen myöhemmässä vaiheessa helpompaa. Huomaa jälleen vinoviivan ja asteriskin avulla luotu kommenttimerkintä.

5.4 Teknisen toteutuksen valmistelut

Kuten aikaisemmin kerroin, verkkosivuston rakentamiselle ei ole olemassa mitään tiettyä kaavaa, mitä olisi pakko noudattaa. Itselläni on tapana – käytinpä sisällönhallintajärjestelmää tai en – rakentaa sivupohja ensin mahdollisimman pitkälle paikalliselle palvelimelle. Kun pohja on siinä pisteessä, että sitä ei voi enää viedä pidemmälle tai sen edistäminen paikallisella palvelimella ei ole enää järkevää, siirrän sivuston lopulliselle palvelimelle, niin sanotulle testipalvelimelle tai integroin rakentamani pohjan jonkin sisällönhallintajärjestelmän kanssa.

Tämän toimintamallin edut ovat mielestäni sen lopullisessa ajan säästämisessä; vältetään turha liikenne www-palvelimen ja oman koneen/palvelimen välillä kehitysvaiheessa ja lopullisen sivuston julkaiseminen on selkeämpää, kun kaikki on viilattu täysin loppuun asti ennen tiedostojen vientiä www-palvelimelle. Näin sivuston kokonaiskuvasta ja -tilanteesta säilyy eheämpi kuva koko prosessin ajan. Mutta kuten sanottua: oikeaa tai väärää järjestystä sen liiemmin kuin toteutustapaakaan ei ole olemassa, kunhan lopputulos on hyvä, tehokas ja toimiva.

5.4.1 Sisällönhallintajärjestelmän valinta

Projektin alkuvaiheessa en suunnitellut käyttäväni minkäänlaista julkaisu-/sisällönhallintajärjestelmää ollenkaan. Tähän ajatukseen vaikuttivat muun muassa sivuston lyhyt käyttöikä sekä sen odotettavissa olevien päivitysten vähäinen määrä; olisi helpompaa ja lopulta nopeampaa luoda sivusto täysin käsin ja julkaista se sellaisenaan, ja koska olin itse vastuussa tulevien päivitysten hoitamisesta, olisi todennäköisesti ihan yhtä nopeaa tehdä muutokset suoraan sivupohjaan ja päivittää pohja palvelimelle kuin päivitysten tekeminen sisällönhallintajärjestelmän kautta.

Päädyin kuitenkin käyttämään sisällönhallintajärjestelmää, koska halusin antaa myös asiakkaalle mahdollisuuden päivittää sivustoa tarpeen niin vaatiessa. Tein valintani kolmen eri sisällönhallintajärjestelmän välillä: WordPress, Joomla ja Drupal. Syy siihen, miksi valinta muodostui juuri näiden kolmen välille on siinä, että edellä mainitut järjestelmät ovat yksinkertaisesti minulle itselleni tutuimmat heti kaupallisen Crasmanager-sisällönhallintajärjestelmän jälkeen.

Tiputin joukosta Joomlan melko pian jo pelkästään henkilökohtaisten kokemusteni takia. En koe kyseistä järjestelmää niin hyväksi työkaluksi, että saisin sitä hyödyntäen aikaiseksi hienoja ratkaisuja tehokkaasti. WordPress ei taas tullut kysymykseen sen blogimaisuuden takia; one page -mallia hyödyntävän sivuston toteuttaminen WordPressiä hyödyntäen ei ole tehokasta tai joissakin tapauksissa edes mahdollista. WordPress olisi kyllä ollut helppokäyttöinen vaihtoehto asiakkaan kannalta ja minulle ennestään tutuin näistä kolmesta.

Lopulta päädyin Drupaliin, joka kahden edellisenkin tavoin on avoimeen lähdekoodiin perustuva sisällönhallintajärjestelmä (Drupal.fi 2014). Kyseinen järjestelmä ei ollut minulle ennestään kovinkaan tuttu, vaikka olinkin selvillä sen perusteista. Drupalin ehdottomat vahvuudet ovat kuitenkin sen lähes rajattomassa muokattavuudessa ja joustavuudessa, mitkä mahdollistavat oikeastaan minkälaisen sivuston toteuttamisen tahansa (Mitchell 2013), joten ei ole ihme, että monet verkkosivustojen kehittäjät valitsevat Drupalin omaksi alustakseen.

5.4.2 Käytetyt ohjelmat

Verkkosivuston rakentaminen ei välttämättä vaadi suuria investointeja ohjelmistojen osalta. Adoben ohjelmistot ovat toki vallanneet suuren osan graafisen suunnittelun markkinoista ja Adoben julkaisema Creative Cloud -palvelu antaa www-suunnitteluunkin uusia työkaluja (Adobe 2014), mutta merkintä- ja ohjelmointikielien kirjoittamiseksi riittää periaatteessa mikä tahansa kirjoitusohjelma. Lopulta ohjelmien valinnassa kyse on vain mielipiteistä ja tottumuksista, sekä tietysti lopullisesta käyttötarkoituksesta.

Itse olen tottunut käyttämään Adoben Dreamweaveria, joka tarjoaa paljon erilaisia työkaluja ja mahdollisuuksia kokonaisvaltaiseen sivuston kehittämiseen. Tässäkin tapauksessa kyse on silti vain tottumuksesta, sillä reilusti yli puolet ohjelman ominaisuuksista ovat sellaisia, joita en ole ikinä tarvinnut tai edes kokeillut. Olen nimittäin tottunut kirjoittamaan kaiken käsin, enkä kaipaa turhaa automaatiota tai avustusta merkintä- tai ohjelmointikielien kirjoittamiseen. Ainoa ehdoton vaatimukseni on toimiva sisennystoiminto sekä koodin värikorostus.

```
20      <article id="second" style="background-color: #81123A;">
21          <a name="second"></a>
22          <div class="content">
23              <?php print render($page['artikkeli2']); ?>
24          </div>
25      </article>
```

Kuva 8. HTML-merkintäkieltä ja pätkä PHP-ohjelmointikieltä väritettynä. Värit helpottavat hahmottamaan, mitä rivit pitävät sisällään ja huomaamaan virhemerkinnät.

```
<article id="second" style="background-color: #81123A;">
        <a name="second"></a>
        <div class="content">
            <?php print render($page['artikkeli2']); ?>
        </div>
</article>
```

Kuva 9. Sama pätkä kuin ylempänä, mutta ohjelmassa joka ei väritä koodia.

Graafisten elementtien keräämiseen ja muokkaamiseen ja ulkoasutiedoston tutkailuun käytän niin ikään Adoben valmistamaa Photoshop-ohjelmaa, joka on vakiinnuttanut asemansa käytetyimpänä kuvankäsittelyohjelmana ja digitaalisen grafiikan tuottamiseen tarkoitettuna ohjelmana graafisten suunnittelijoiden keskuudessa. Ohjelma tarjoaa todella laajan valikoiman erilaisia työkaluja ja ominaisuuksia moneen eri tarpeeseen. Sen suuren suosion takia se on myös lähes varma valinta yhteistyötilanteissa, kun esimerkiksi ulkoasutiedosto siirtyy suunnittelijalta tekniselle toteuttajalle.

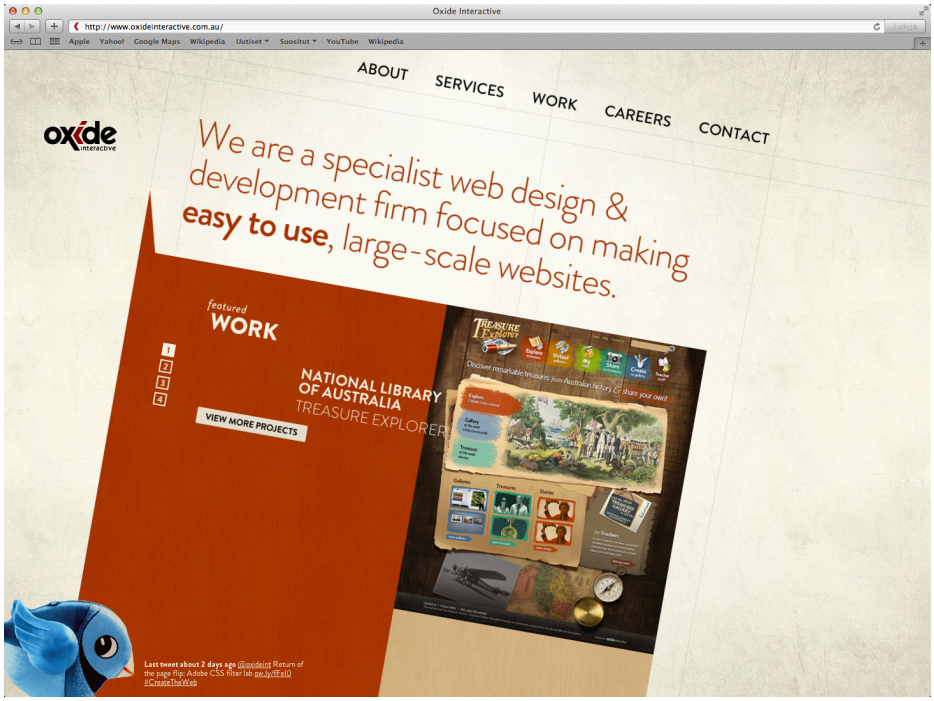
Kun verkkosivusto julkaistaan ulkoisella palvelimella, vaaditaan tiedostojen siirtämiseen FTP-ohjelma. Vaihtoehtoja on monia, ja Dreamweaverissa on sisäänrakennettunakin FTP-ominaisuus. Itse olen käyttänyt jo useamman vuoden ajan ilmaista FileZilla-ohjelmaa, joka perustuu avoimeen lähdekoodiin ja on saatavana useammalle käyttöjärjestelmälle (FileZilla Wiki 2009).

Verkkosivusto on myös testattava useammalla selaimella ja laitteella (Wroe 2013). Tällä hetkellä käytetyimmät selaimet ovat Firefox, Internet Explorer, Safari, Chrome ja Opera (w3schools 2014). Tarkkoja statistiikkoja käyttömääristä on turha ruveta tutkimaan, sillä kaikkia edellä mainittuja selaimia käytetään joka tapauksessa niin paljon, että verkkosivusto on testattava jokaisella erikseen. Tavoitteena on saada sivusto näyttämään samalta ja toimimaan samalla tavalla jokaisessa selaimessa, vaikka varsinkin selainten vanhempien versioiden kohdalla voikin joutua tekemään kompromisseja ulkoasun tai toiminnallisuuksien suhteen. Syynä on se, että vanhat versiot eivät tue kaikkia samoja tekniikoita ja standardeja kuin uudemmat versiot.

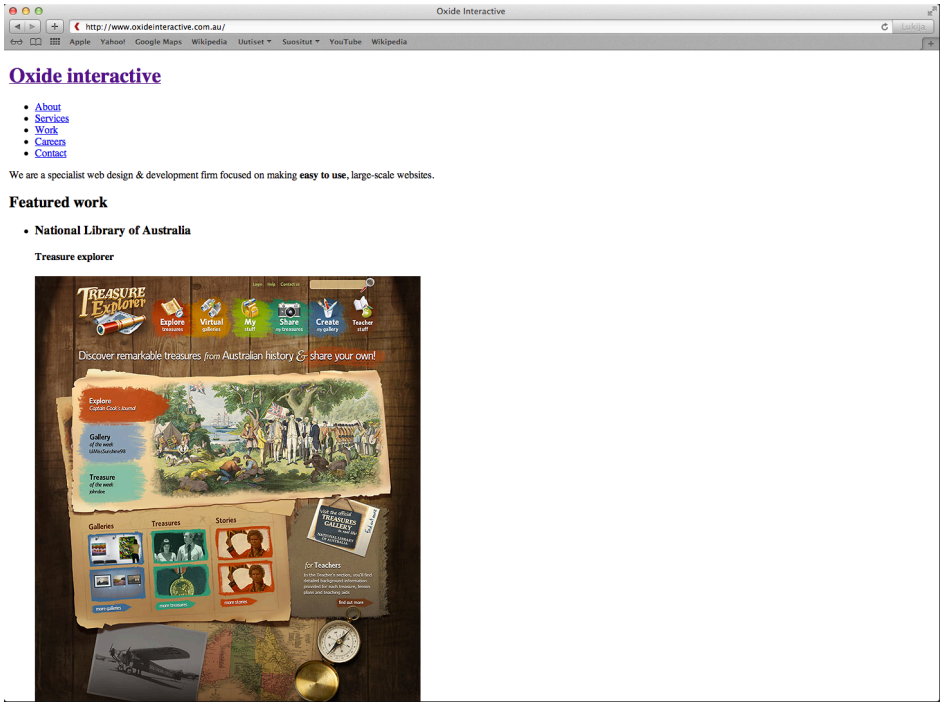
6 SIVUPOHJAN RAKENTAMINEN

Sivupohjalla tarkoitetaan sivuston perusrakennetta ja ulkoasua, johon sisältö voidaan sijoitella jälkeenpäin. Toisin sanoen sivupohjan ei ole välttämätöntä sisältää mitään sivustolle tulevaa sisältöä, kuten tekstiä, vaan pohja suunnitellaan niin, että sisältö asettuu siihen automaattisesti. Sivupohjaa luodessa voi käyttää täytetekstiä mallina. Samaa sivupohjaa voidaan myös käyttää useamman kerran samalla sivustolla, mikä onkin monen sisällönhallintajärjestelmän yksi perusideoista; päivittäminen helpottuu huomattavasti, kun esimerkiksi useat alasivut käyttävät samaa pohjaa; yhteen pohjaan tehtävät muutokset tulevat voimaan jokaisella alasivulla automaattisesti.

Sivupohja – ja verkkosivusto yleensäkin – koostuu lähes poikkeuksetta vähintään kahdesta osasta: HTML-tiedostosta ja CSS-tiedostosta. Ensimmäinen näistä sisältää sivuston varsinaisen rakenteen ja eri elementit kuten tekstit, kuvat, linkit, videot ja navigaatiot. Kaikki konkreettinen on siis HTML-tiedostossa. CSS-tiedosto taas määrittelee sivustolle sen varsinaisen ulkoasun, eli miten nämä HTML-tiedostossa olevat elementit esitetään; esimerkiksi miten ne sijoittuvat suhteessa toisiinsa, minkä kokoisia ne ovat, minkä väristä teksti on, millainen fontti otsikossa on ja kuinka paljon kuvan alapuolella on tyhjää tilaa ennen kuvatekstiä. Toisin sanoen CSS määrittelee hyvin pitkälti sivuston koko ulkoasun (kuvat 10 ja 11).



Kuva 10. Kuva verkkosivustosta, kun CSS-määritelmät ovat käytössä.



Kuva 11. Sama sivusto ja sisältö kuin kuvassa 10, mutta nyt ilman CSS-määritelmiä.

6.1 HTML ja HTML5

HTML on merkintäkieli, jota käytetään pääasiassa verkkosivustojen sisällön luomiseen ja sivuston sisällön esittämiseen. Sisältöä voidaan jäsentää käyttämällä erilaisia *tageja* merkitsemään, minkälaisesta sisällöstä missäkin kohdin on kyse; kappaleet, otsikot, kuvat, linkit, listat ja muut vastaavat voidaan erottaa toisistaan käyttämällä niille suunnattuja tageja (w3 2013). Esimerkiksi pääotsikko merkitään *<h1>*-tagilla ja tekstikappale * *-tagilla:

<h1>Tähän tulee otsikko</h1>
* Ja tästä alkaa otsikon jälkeinen tekstikappale *

Sisällön jälkeen tuleva tagi, jossa on vinoviiva, merkitsee kyseisen sisältötyypin päättymistä. Käyttämällä erilaisia tageja koko sivuston sisältö voidaan jäsentää ja esittää selkeällä ja järkevällä tavalla. HTML on lopulta hyvin yksinkertaista ymmärtää ja kirjoittaa, mutta haasteena on tagien oikeaoppinen ja semanttinen käyttö sekä sivuston rakenteen hahmottaminen ennen varsinaista kirjoittamista; kehittäjän on ymmärrettävä heti prosessin alkuvaiheessa, miten hän lähtee rakentamaan kyseistä sivupohjaa HTML:n avulla, eli miten hän hyödyntää hänelle annettuja HTML:n työkaluja. Kaikki tagit eivät kuitenkaan ole näin yksiselitteisiä tai vain yhteen käyttöön tarkoitettuja.

6.1.1 HTML5

HTML on vuosien aikana kehittynyt huomattavasti. Ensimmäiset viitteet HTML:stä ja sen käyttämisestä www-sivujen luomiseksi saatiin jo vuonna 1991 (w3 Archives 1991), ja siitä lähtien HTML on toiminut verkkosivustojen perustana. Se on kuitenkin muuttunut vuosien varrella. Pääasiassa muutokset ovat koskeneet tageja, kun osa poistetaan käytöstä ja joitakin taas lisätään standardeihin.

Viimeisin villitys ja toisinaan väärissäkin yhteyksissä käytetty termi onkin HTML5, joka on itse asiassa terminä hieman epämääräinen ja kiistanalainenkin. Osittain kyse on uusien web-tekniikoiden keskittämisestä yhden nimikkeen alle, mutta HTML5

voidaan myös käsittää HTML:n uutena versiona tai kehitysaskeleena. Sen suosion salaisuus on siinä, että se tarjoaa jokaiselle jotakin. Kaikki jotka ovat seuranneet verkon ja verkkosivustojen kehitystä, löytävät HTML5:stä jotakin sellaista, mitä voidaan pitää selvänä kehitysaskeleena (Korpela 2011, 13).

Kehityksenä voidaan pitää ainakin sitä, että HTML5 toi mukanaan täysin uusia ja hyödyllisiä tageja, kuten <header>, <article> <canvas>, <video> ja <nav>, joita käyttämällä HTML-tiedoston rakenne selkeytyy entisestään. Vastaavasti joitakin tageja on merkitty vanhentuneiksi, kuten font-elementti. Tämä ei tarkoita, että sivustot joilla on käytetty vanhentuneita elementtejä, eivät enää toimisi oikein, sillä HTML5-luonnosten mukaan selaimia vaaditaan tukemaan myös vanhentuneita piirteitä (Korpela 2011, 16). Useimmiten vanhentuneeksi merkkaamisen taustalla on se, että haluttu vaikutus voidaan toteuttaa tehokkaammin jollakin toisella tekniikalla, kuten CSS:llä tai JavaScriptillä. Tästä voidaan mainita esimerkkinä HTML5:n mukainen suositus, jonka mukaan sivuston ulkoasu tulee luoda käyttämällä pelkästään CSS-määritelmiä (Korpela 2011, 15). Ei siis ole tarpeen määritellä esimerkiksi fontteja tai värejä HTML:n avulla.

HTML5 on kuitenkin käytännössä paljon enemmän kuin pelkästään uusi versio HTML-merkintäkielestä. Kuten aiemmin mainitsin, se voidaan osittain käsittää myös useamman nykyaikaisen web-tekniikan keskittämisenä yhden nimikkeen alle. Tämä voi tarkoittaa käytännössä esimerkiksi HTML:n, CSS:n ja JavaScriptin yhteiskäyttöä jonkinlaisen web-sovelluksen toteuttamiseksi. Tämä voi olla hankala asia hahmottaa, sillä näitä kolmea tekniikkaa on käytetty jo useita vuosia yhdessä. Erona voidaan pitää kuitenkin sitä, että HTML5:tä on kehitetty nimenomaan tämä yhteiskäyttö mielessä, ja näin ollen HTML5 tarjoaa lisää mahdollisuuksia sovellusten toteuttamiseksi tätä yhteiskäyttöä hyödyntäen. (Korpela 2011, 14.)

On kuitenkin vaikea määritellä, mikä on ”HTML5-sivu”. Jos sivustolla on yksikin HTML5:n mukanaan tuoma elementti, sen tekijä voi sanoa tehneensä HTML5-sivun. Toisaalta HTML5-sivuna voidaan pitää sellaistakin sivua, joka sisältää ainoastaan HTML5:n luonnoksissa mainittuja elementtejä ja määritteitä. Tosin, siinä mielessä myös jokainen HTML 4:n mukainen sivu on myös HTML5-sivusto. Jos ajatellaan suppeammin, niin ”HTML5-sivu” tarkoittaisi sivustoa, joka on toteutettu niiden

vaatimusten mukaan, jotka HTML5:n luonnoksissa asetetaan sivuille, eli se ei esimerkiksi sisällä vanhentuneita elementtejä (Korpela 2011, 27.)

Kehittäjältä ei vaadita erityistoimenpiteitä siirtymiseksi HTML5:een, vaan siirtymisen voi aloittaa esimerkiksi vaiheittain lisäämällä joitakin tarpeelliseksi katsottuja HTML5:n pohjautuvia elementtejä. Siirtymisen helppous perustuukin siihen, että HTML5:tä ei ole rakennettu vanhan tilalle, vaan sen päälle. Vanhaa ei siis tarvitse purkaa, vaan siihen voidaan tehdä lisäyksiä tai se voidaan pikkuhiljaa muuntaa muotoon, joka on HTML5:n suositusten mukainen. (Korpela 2011, 14-16).

HTML-tiedostoa on helppo lukea ylhäältä alaspäin. Alla on lista yksinkertaisen HTML-tiedoston rakenteesta:

1. Head-, eli otsaketiedot
    a. Sivuston meta-tiedot (eivät pakollisia, mutta suositeltavia)
        i. Sisältää tietoja sivustosta ja sen sisällöstä, esimerkiksi hakukoneet hyödyntävät näitä tietoja
    b. Sisällytetyt ulkopuoliset tiedostot, esim. CSS ja JavaScript
    c. Title, eli sivuston otsikko
2. Body, eli sivuston konkreettinen ja näkyvä rakenne
    a. Sivuston varsinainen rakenne ja sisältö

Yksinkertainen HTML-tiedosto voisi siis näyttää esimerkiksi tältä:

```
<html>
<head>
<title>Sivuston otsikko tai nimi</title>
</head>

<body>
Sivuston sisältö.
</body>
</html>
```

6.1.2 SpeksiYö-tapahtuman verkkosivuston HTML-pohja

Kun lähden rakentamaan sivustolle pohjaa, aloitan lähes poikkeuksetta HTML-rakenteesta. Mielestäni on loogista, että CSS tuodaan mukaan kuvioon vasta myöhemmin. On helpompi luoda ensin elementit ja vasta sen jälkeen määritellä niiden ulkoasu. Aivan ensimmäiseksi tutkin suunniteltua ulkoasua ja mietin, millainen rakenne olisi paras vaihtoehto. Yleensä pyrin pitämään asiat mahdollisimman yksinkertaisena, jolloin sivuston toimiminen yhtenäisesti kaikissa selaimissa on varmempaa ja HTML-rakenne oikeaoppista. On hyödytöntä tehdä lisätyötä tai asioita liian vaikeasti, jos yksinkertaisemmallakin ratkaisulla pääsee samaan – tai usein parempaankin – lopputulokseen.

Luon HTML-pohjan ensin omalle koneelleni huomioimatta esimerkiksi sisällönhallintajärjestelmän vaatimuksia sen enempää, sillä oikein toteutettuna mikä tahansa pohja on integroitavissa mihin tahansa sisällönhallintajärjestelmään. Tavoitteenani on vain luoda täydellinen malli siitä, miltä sivu tulee näyttämään ja miten se toimii. Kun pohja on viilattu loppuun asti, se on helppo sisällyttää sisällönhallintajärjestelmään, jonka jälkeen voin aloittaa muun sisällön sijoittamisen paikoilleen. Mikäli sivustolla on useita alasivuja joilla on samanlainen pohja, en tietenkään tee jokaista alasivua valmiiksi, vaan luon yhden mallin, jonka vien sisällönhallintajärjestelmään ja luon alasivut sen pohjalta. SpeksiYön sivuston kohdalla tätä ei tietenkään tarvinnut ottaa huomioon, koska sivusto koostuu vain yhdestä sivusta.

Kun mallipohja luodaan ensin omalle palvelimelle/koneelle, eikä sitä viedä heti alusta lähtien sisällönhallintajärjestelmään, on prosessi mielestäni nopeampi ja selkeämpi. Muutosten tekeminen rakennusvaiheessa sisäistä palvelinta hyödyntäen on nopeampaa, kuin muutosten tekeminen ulkoiselle palvelimelle. Ei ole myöskään harvinaista, että vielä rakennusprosessin aikanakin tietyt elementit ja sisältöön liittyvät osaset – kuten vaikkapa kuvat – hakevat lopullista muotoaan ja muuttuvat useaan otteeseen.

Aloitin SpeksiYö-tapahtuman sivupohjan luomisen käymällä ulkoasua läpi ylhäältä alaspäin ja kirjoittamalla samalla HTML-merkintöjä pääpiirteittäin, eli merkitsemällä

tärkeimmät ensimmäisen tason elementit niille tarkoitetuin tagein. Tämän jälkeen mietin jokaisen ensimmäisen tason elementin omaa sisältöä ja sen sisältämiä elementtejä tarkemmin, eli miten ne luodaan ja mikä olisi järkevin tapa esittää nämä elementit. Esimerkiksi navigaation sisälle tarvitsee luoda sen sisältämät linkit, toisen lohkoalueen sisälle tulee tekstiä ja pari linkkiä, viidennen lohkoalueen sisälle täytyy luoda kolmeen palstaan eri sisältöä; vasempaan palstaan tuottajan yhteystiedot, keskipalstaan Porin ylioppilasteatterin ja SpeksiYö-tapahtuman linkit ja oikeanpuoleiseen palstaan Facebook-linkki ja -kuvake.

Mielestäni on helpointa käydä jokainen ensimmäisen tason elementti ja sen sisältö yksitellen läpi. Keskittyminen yhteen asiaan kerrallaan on helpompaa ja rakentamisesta tulee johdonmukaisempaa.

```
<nav id="main">
    <ul>
        <li><a href="#second">Mikä speksiyö?</a></li>
        <li><a href="#third">Mukaan kisaan?</a></li>
        <li><a href="#fourth">Ohjelma</a></li>
        <li><a href="#fifth">Ota yhteyttä</a></li>
    </ul>
</nav>
```

Kuva 12. Navigaatio-elementti HTML-muodossa.

Sivuston yläreunassa näkyvä navigaatiopalkki on merkattu HTML-tiedostossa kuvan 12 osoittamalla tavalla. Elementti aloitetaan <nav>-tagilla ja vastaavasti lopetetaan viimeisenä olevalla </nav>-tagilla. Näiden välissä on -elementti, joka merkitsee listaelementtiä. Tämän elementin sisällä taas on -tageja, joista jokainen merkitsee kyseisen listan yhtä riviä. Jokaisen rivin sisältämä <a>-tagi merkitsee linkkiä, jonka arvo on määritelty tagiin itseensä merkityllä *href*-arvolla. <a>-tagin sisällä oleva teksti taas on se teksti, mikä meille näkyy selaimessa linkkinä.

- Mikä speksiyö?
- Mukaan kisaan?
- Ohjelma
- Ota yhteyttä

Kuva 13. Sama navigaatio-elementti selaimessa (ilman CSS-määritelmiä).

Tässä kohdin ei tarvitse välittää siitä, miltä sivusto näyttää selaimessa, sillä ilman CSS-määritelmiähän sivustolla ei ole varsinaista ulkoasua. Havainnollistavana esimerkkinä alla on kuitenkin kuva (kuva 14) tuosta samasta navigaatioelementistä niin, että se on CSS:ä käyttäen muotoiltu suunniteltua ulkoasua vastaavaksi.

MIKÄ SPEKSIYÖ? / MUKAAN KISAAN? / OHJELMA / OTA YHTEYTTÄ

Kuva 14. Lopullinen, tyylitelty, versio samasta elementistä kuin kuvassa 13.

Tällä samalla idealla kävin sivuston läpi alusta loppuun, eli HTML:n työkaluja hyödyntäen merkitsin jokaisen elementin ja sen sisällä olevan elementin niille sopivilla tageilla, ja tuloksena minulla oli valmis HTML-pohja. Mutta kuten todettua, pelkkä HTML ei näytä juuri miltään, joten HTML-pohjan valmistumisen jälkeen siirryn luomaan tälle pohjalle sen lopullisen ilmeen ja ulkoasun CSS:n avulla. Sivupohjan rakentamisen vaiheet menevät siis oman työnkulkuni mukaan näin:

1. Ulkoasun läpikäyminen ja suunnitelman tekeminen
2. Graafisten elementtien ja tietojen kerääminen ulkoasutiedostosta
3. HTML-pohjan rakentaminen pääpiirteittäin
4. Kaikkien lopullisten elementtien lisääminen HTML-pohjaan
5. CSS-tyylien määritteleminen

6.2 CSS ja CSS3

CSS, eli Cascading Style Sheets, on tarkoitettu sekä HTML-, että XML-tiedostojen muotoiluun (Korpela 2013). XML:n ominaisuuksien selvittäminen ei kuitenkaan ole tässä raportissa olennaista, joten keskitymme CSS:n yhteiskäyttöön HTML:n kanssa ja erityisesti siihen, miten se liittyy SpeksiYö-tapahtuman verkkosivuston toteuttamiseen.

Vaikka CSS on erittäin olennainen osa verkkosivuston muodostumista ja modernien web-sovellusten toteuttamista, sitä ei pidä luulla HTML5:een sisältyväksi. CSS on täysin erillinen osansa ja HTML5 on oma osansa, mutta näiden tekniikoiden välillä on hyvin vahva yhteiskäyttömahdollisuus. (Korpela 2013.)

CSS:n avulla mikä tahansa HTML-elementti voidaan muotoilla halutulla tavalla. Elementtiin viitataan CSS-tiedostossa käyttämällä elementin tagia, luokkaa tai yksilöllistä ID-merkintää. Tätä kutsutaan selektoriksi. Selektori siis määrittelee, mitä elementtiä tai elementtejä kyseinen CSS-sääntö koskee. Yksinkertainen CSS-sääntö voisi olla esimerkiksi tällainen:

```
h1 {
        font-size: 1em;
        color: #FFFFFF;
}
```

Tässä esimerkissä käytetään selektoria *h1*, joka siis kertoo, että kaikki sivustolla olevat <h1>-tagilla merkityt elementit noudattavat tätä sääntöä. Tämän jälkeen varsinainen sääntö aloitetaan {-merkillä, ja vastaavasti lopetetaan määritelmien jälkeen }-merkillä. Tässä esimerkissä määritellään <h1>-elementtien fonttikooksi 1em ja tekstin väriksi valkoinen, joka on merkitty HEX-arvona (#FFFFFF).

Jos sivustolla on useita <h1>-tagein merkattuja elementtejä, joista osa halutaan muotoilla eri tavoin, voidaan näille elementeille lisätä luokka. Käytännössä HTML-tagi muutetaan esimerkiksi muotoon <h1 class=”keltainen”>, joka antaa tälle

elementille luokan ”keltainen”. Nyt CSS-tiedostoon voidaan lisätä uusi sääntö, jonka selektoriin lisätään tagitunnisteen jälkeen piste ja sen perään haluttu luokka:

```
h1.keltainen {
        color:  #F7FF10;
}
```

Tämä sääntö koskee ainoastaan niitä <h1>-elementtejä, joille on annettu luokaksi ”keltainen”. Sen sisältämät määritelmät, tässä tapauksessa tekstin keltainen väri, korvaavat aiemmat määritelmät, jotka annettiin yleisesti <h1>-elementeille.

CSS-sääntö voidaan myös kohdistaa elementtiin, jolla on ID-merkintä. Samaa ID:tä ei tule käyttää kuin yhdessä elementissä samalla sivulla; toisin sanoen ID:llä merkitty elementti on ainoa laatuaan. Sama luokka voi toistua sivustolla useitakin kertoja, mutta ID:n ideana on merkitä täysin uniikki elementti. SpeksiYö-tapahtuman sivustolla tällainen elementti on esimerkiksi navigaatioelementti (<nav>), jonka ID on ”main”. CSS-tiedostossa tälle elementille on määrätty seuraavanlainen sääntö:

```
nav#main {
        width: 100%;
        height: 90px;
        position: fixed;
        top: 0;
        background-color: #FFFFFF;
}
```

Kun jokin sääntö halutaan kohdistaa elementille jolle on määritelty oma ID, lisätään selektoriin #-merkki ja heti perään elementille annettu ID. Yllä olevassa esimerkissä kohdistamme säännön <nav>-elementtiin aloittamalla selektorin sanalla ”nav”, jonka jälkeen lisäämme #-merkin viitaten siihen, että haluamme kohdistaa säännön elementille jolla on tietty ID, joka lisätään heti #-merkin jälkeen (”main”).

Miksi sitten jollekin elementille annetaan oma ID, kun yhtä hyvin voisi käyttää luokkaa? Tai jos sivulla on esimerkiksi vain yksi <nav>-elementti, miksi se pitää

yksilöidä vielä omalla ID:llä? Syitä voi olla useampia. Yksi on se, että sivustolla saattaa olla useampia samankaltaisia elementtejä, joilla on jopa sama luokka, mutta yksi näistä halutaan nostaa vielä muiden yläpuolelle vaikkapa jonkinlaisen visuaalisen korostuksen luomiseksi. Toisaalta, tämä voitaisiin tehdä lisäämällä elementille toinen luokkakin, jolloin elementtien tagit voisivat näyttää tältä:

*<li class=”mallilista”>Malli 1
*
*<li class=”mallilista korostus”>Malli 2
*
*<li class=”mallilista”>Malli 3
*

Tässä esimerkissä keskimmäinen elementti erotetaan kahdesta muusta antamalla sille toinenkin luokka, ”korostus”. Onkin kiistelty, onko ID:n käyttäminen järkevää tai pakollista, jos sama tavoite voidaan saavuttaa käyttämällä luokkaa. ID voi aiheuttaa ulkoasuun jopa sekaannuksia, koska ID:n avulla annetut CSS-ohjeet korvaavat aina yleisesti elementille annetut tai luokan avulla annetut ohjeet. (Roberts 2014).

Toisaalta tämä ID:n mukana tuleva ”voima” voi olla nimenomaan syy käyttää sitä, sillä joissakin tilanteissa tärkeät ja täysin yksilölliset elementit voi olla fiksua erottaa toisistaan uniikeilla ID:llä. Samaten ID voi helpottaa elementtiin viittaamista esimerkiksi JavaScriptin kautta tai luodessa linkkejä, joiden avulla hypätään automaattisesti tiettyyn osaan sivustolla.

Lopulta on yhdentekevää käyttääkö kehittäjä ID:tä vai ei, kunhan lopputulos on toimiva. ID ei ole missään nimessä paha asia, eikä sen käyttäminen kerro sivuston tekijän taidottomuudesta, mutta jokaisen verkkosivustojen toteuttamisen parissa työskentelevän on hyvä olla täysin perillä siitä, missä tilanteessa ID:tä on järkevää käyttää ja milloin kannattaa turvautua muihin ratkaisuihin. (Roberts 2014).

6.2.1 SpeksiYö-tapahtuman verkkosivuston CSS-ohjeet

Kuten HTML-pohjankin rakentamisenkin, aloitan myös CSS-tiedoston rakentamisen käyden sivustoa läpi ylhäältä alaspäin. Käsittelen yhden elementin ja sen elementin sisältämät muut elementit yksi kerrallaan, jolloin ulkoasu rakentuu järjestelmällisesti.

Tämä on mielestäni selkeä toteutustapa, joka antaa aikaa keskittyä yksityiskohtiin kunnolla; sen sijaan että yrittäisin hahmottaa koko ajan pelkkää kokonaiskuvaa ja rakentaa sivustoa kokonaisvaltaisesti, pilkon ulkoasun moneen pienempään osaan ja panostan jokaiseen niistä erikseen.

CSS:n kanssa ongelmaksi on vuosien aikana muodostunut eri selainten erilaiset tavat tulkita joitakin sen ominaisuuksia, jolloin sivuston ulkoasu on näyttänyt eri selaimissa erilaiselta. Jokaisella selaimella on myös niin sanottu ”perustyyli”, jota se käyttää tulkitsemaan sivustoja ja elementtejä, joille ei ole määritelty omia tyyliohjeita. Esimerkiksi linkit tyylitellään usein oletuksena siniseksi ja lähes jokaisen elementin ympärille lisätään jonkin verran tyhjää tilaa. Näissä tyyleissä on kuitenkin eroja eri selainten välillä, mikä aiheuttaa sivuston rakenteen erilaisen tulkinnan, eli ulkoasun näyttämisen eri tavoin. Esimerkiksi yksi selain saattaa lisätä oletuksena 10 pikseliä tyhjää tilaa jokaisen otsikon yläpuolelle, kun toinen selain lisää samaan kohtaan 15 pikseliä. (CSS Reset 2014).

Tämän ongelman kiertämiseksi on luotu ”CSS reset”-ohje. Se on käytännössä lyhyt pätkä normaaleja CSS-tyyliohjeita, jotka voidaan lisätä sivuston CSS-tiedostoon. Nämä tyyliohjeet kumoavat kaikki selainten oletuksena käyttämät ohjeet, mikä asettaa kaikki selaimet samalle viivalle ja antaa kehittäjälle ”tyhjän kankaan”, jonka päälle ulkoasua on selkeämpi rakentaa. (CSS Reset 2014). Valmiita reset-ohjeita on tarjolla useampia erilaisia, mutta kaikki antavat pääasiassa vastaavan lopputuloksen.

Lisättyäni reset-ohjeen heti tiedoston alkuun, lähdin luomaan tyyliohjeita sivuston elementeille. SpeksiYö-tapahtuman sivuston CSS-tyylien osalta haasteena on muun muassa se, että sivusto ei noudata täysin yhtenäistä linjaa esimerkiksi fonttikokojen suhteen; muun muassa jokaisen lohkoalueen otsikot ovat hieman eri kokoisia ja värisiä. En siis voinut luoda yhtä yleistä tyyliä kaikille otsikoille, vaan kävin lopulta jokaisen läpi yksitellen. Samanlaista lisätyötä oli leipätekstissä ja linkeissä, sillä taustavärien suurien vaihtelujen takia (tummanlilasta valkoiseen) myös tekstin väriä oli vaihdettava. Linkkien värit eivät taas noudata sivulla minkäänlaista logiikkaa, vaan niiden värit vaihtelevat samanvärisenkin taustan kohdalla (kuva 14).

Kuva 14. Kaksi erilailla väritettyä linkkiä vierekkäin.

Joku voisi tässä kohtaa mainita siitä, miten tällainen toteutus on huonoa käytettävyyttä ja miten heikosti käyttäjä hahmottaa linkit, mutta kuten aikaisemminkin tässä raportissa mainittiin, sivuston ulkoasun tarkoitus on tuoda tapahtuman ja sen järjestäjän imagoa esille ja luoda tietynlainen tunnelma sivustolle. Linkit voidaan myös nostaa esille muillakin keinoilla; kuvassa 14 näkyvät linkit ovat erillisellä rivillä ja ne käyttävät leipätekstiä suurempaa fonttikokoa, joten käyttäjän huomio kiinnittyy niihin varmasti. Myös linkeissä käytetyt tekstit kehottavat käyttäjää klikkaamaan niitä.

Tärkeintä on pysyä uskollisena graafisen suunnittelijan tekemälle ulkoasulle, kunnioittaa hänen tekemiään ratkaisujaan ja pyrkiä toteuttamaan sivusto niin, että sen lopullinen ulkoasu näyttää samalta kuin alkuperäinen suunnitelma. CSS tarjoaa tähän nykyään hyvät ja tehokkaat työkalut. Vaikka SpeksiYö-tapahtuman verkkosivuston ulkoasu onkin suhteellisen räväkkä ja erilainen, sen CSS-ohjeet ovat kuitenkin hyvin yksinkertaisia. Erikoisemmilta tuntuvat tai näyttävät ratkaisut eivät välttämättä tarvitse kovin erikoisia CSS-määritelmiä.

CSS-tyyliohjeiden kirjoittamista helpotti se, että SpeksiYö-tapahtuman verkkosivuston ulkoasu on kuitenkin perusrakenteeltaan hyvin järjestelmällinen; yksi tyyliohje yhdelle lohkoalueelle, yksi ohje sen lohkoalueen otsikolle, yksi ohje sen otsikon alapuolella olevalle leipätekstille ja niin edelleen. Myöhemmässä vaiheessa yhdistin osan näistä CSS-ohjeista yhdeksi ohjeeksi; esimerkiksi kaikki lohkoalueet noudattavat osaltaan samoja määritelmiä, joten on järkevämpää kohdistaa nämä yhteiset määritelmät yhden ohjeen sisälle, jota kaikki halutut elementit noudattavat. Yksilölliset ohjeet näille elementeille voidaan kuitenkin osoittaa käyttämällä luokkaa tai ID-merkintää. Alla on esimerkki tästä.

```
189  article {
190      width: 92%;
191      min-height: 100%;
192      padding: 0 4%;
193      background-size: cover;
194      background-position: center;
195      position: relative;
196  }
197
198      article#second {
199          background-image: url(../gfx/pyt_nuoli-valk.png);
200      }
201
202      article#third {
203          background-image: url(../gfx/pyt_nuoli-lila.png);
204      }
```

Kuva 15. Kolme CSS-tyyliohjetta SpeksiYön verkkosivuston CSS-tiedostosta. Kaikki ohjeet viittaavat <article>-elementteihin, mutta eri tavoin.

Yllä olevassa kuvassa (kuva 15) on kolme CSS-tyyliohjetta. Ylimpänä oleva koskee kaikkia sivuston <article>-elementtejä. Tämän ohjeen alla on kuitenkin kaksi muuta ohjetta, joilla viitataan myös <article>-elementteihin. Huomaa kuitenkin, että keskimmäinen ohje viittaa <article>-elementtiin, jolle on annettu ID ”second” ja alin taas <article>-elementtiin, jolle on annettu ID ”third”. Sekä ”second” että ”third” –elementit noudattavat siis ylintä, kaikille <article>-elementeille tarkoitettua ohjetta, mutta sen lisäksi molemmat noudattavat myös niille yksilöllisesti tarkoitettua ohjetta, jossa elementeille määritellään oma taustakuva.
Tällaisessa tilanteessa sääntöjen niputtaminen yhteen on viisasta useammastakin käytännöllisestä syystä:

1. CSS-tiedostoon tulee vähemmän rivejä
2. CSS-tiedosto on rakenteeltaan selkeämpi
3. Muutosten tekeminen on helpompaa, kun halutun muutoksen voi tehdä yhteen ohjeeseen ja se päivittyy automaattisesti kaikkiin haluttuihin elementteihin

6.2.2 CSS ja responsiivinen suunnittelu

CSS on myös oleellinen osa sivuston responsiivista toteutusta. Kuten raportissa on jo todettu, CSS on tehokkain työkalu sivuston ulkoasun toteuttamiseen, ja sitä se on tietysti myös ulkoasun muokkaamiseen. Responsiivisessa toteutuksessahan on lopulta hyvin vahvasti kyse nimenomaan ulkoasun muuntautumisesta eri tilanteisiin sopivaksi. Perusidea on se, että kun näytön koko muuttuu, myös sivuston rakenne/ulkoasu muuttuu niin, että sivusto näyttää edelleen hyvältä; ulkoasu optimoidaan tuohon näyttökokoon sopivaksi.

CSS:n avulla voidaan luoda yksinkertainen ohje, josta yksi esimerkki voitaisiin muuttaa lauseenomaiseen muotoon näin: *jos näyttö on kapeampi kuin XXX pikseliä, nämä kaksi elementtiä ovat allekkain. Jos taas näyttö on leveämpi kuin XXX pikseliä, nämä samat elementit ovat vierekkäin*. Käytännössä CSS:n avulla voidaan tehdä millaisia muutoksia tahansa, mihin elementtiin tahansa, ihan milloin tahansa. Tärkeintä on tehdä nämä muutokset niin, että sivuston ulkoasu ei hajoa – eli muutu epäselväksi tai käyttökelvottomaksi – missään tilanteessa tai millään näytöllä. Miten nämä muutokset sitten käytännössä tehdään?

Kun CSS3 julkaistiin, sen mukana tuli *media queries* -ominaisuus. Tämä ominaisuus antaa mahdollisuuden tarkastella juuri sen laitteen tietoja ja fyysisiä ominaisuuksia, jolla sivustoa sillä hetkellä katsotaan (Marcotte 2010). Alla (kuva 16) on yksinkertainen esimerkki siitä, miten itse hyödynsin media queries -ominaisuutta muuttaakseni SpeksiYön verkkosivuston navigaation muotoa näytön kaventuessa.

```
121  nav#main {
122      width: 100%;
123      height: 90px;
124      position: fixed;
125      top: 0;
126      background-color: #FFFFFF;
127      z-index: 599;
128  }
129
130  @media all and (max-width: 940px) {
131
132  nav#main {
133      width: 92%;
134      padding: 0 4%;
135  }
136
137  }
```

Kuva 16. Ylempänä on tavallinen CSS-ohje ja sen alapuolella samalle elementille suunnattu media queries –ominaisuutta hyödyntävä ohje.

Tässä esimerkissä (kuva 16) kohdistetaan kaksi ohjetta samalle elementille, mutta vain toinen niistä on kerrallaan käytössä. Ylempänä oleva ohje on niin sanottu ”yleinen ohje”, jonka voimaan tulemiselle ei olla asetettu minkäänlaista ehtoja. Toisin sanoen se on käytössä, mikäli sen korvaavaa ohjetta ei ole olemassa. Media queries -ominaisuutta voi kuitenkin käyttää nimenomaan muiden ohjeiden korvaamiseen, minkä avulla ulkoasua voi muuttaa lennosta.

Tämän esimerkin media queries -ohje alkaa riviltä 130: *@media all and (max-width: 940px)*. Tämä rivi tarkoittaa yksinkertaisesti sitä, että jos media (eli sen laitteen näyttö jolla sivustoa katsotaan) on enintään 940 pikseliä leveä, tulee tämän ohjeen aaltosulkeiden väliin kirjoitetut ohjeet voimaan. Tässä tapauksessa <nav>-elementin leveys muuttuu 100 prosentista 92 prosenttiin ja elementin molemmille laidoille lisätään 4 prosenttia täytettä (englanniksi *padding*). Ne määritelmät, jotka on annettu jo aikaisemmin yleisessä ohjeessa, jäävät voimaan automaattisesti, mikäli niille ei anneta uusia arvoja. Tässä tapauksessa esimerkiksi korkeus pysyy edelleen 90 pikselinä, koska sitä ei media queries -ohjeessa määritellä uudestaan. Media queries -ohjeita voidaan lisätä tarpeiden mukaan mille tahansa näytön leveydelle.

```
616  @media all and (max-width: 590px) {
617
618  nav#main {
619      height: 120px;
620      width: 84%;
621      padding: 0 8%;
622  }
```

Kuva 17. Jälleen uusi media queries –ohje samalle elementille.

Kuvassa 17 oleva media queries -ohje korvaa edellisessä esimerkissä (kuva 16) olleen ohjeen, mikäli sivustoa katsotaan näytöltä joka on leveydeltään enintään 590 pikseliä (huomaa ohjeessa oleva ”max-width: 590px”). Tässä tapauksessa elementin korkeutta, leveyttä ja täytettä muutetaan. Tämä ei kuitenkaan ole ainoa käyttötapa media queries -ominaisuudelle. Esimerkiksi *max-width* voidaan kääntää muotoon *min-width*, jolloin viitataankin laitteen vähimmäisleveyteen tai sana *all* voidaan korvata sanalla *screen*, jolloin viitataan näyttöpäätelaitteeseen, mutta ei esimerkiksi tulostimeen (*print*). Myös laitteen korkeutta voidaan hyödyntää ja tarvittaessa voidaan hyödyntää sekä leveyttä että korkeutta. (Marcotte 2010).

Responsiivista ulkoasua ei kuitenkaan saavuteta pelkästään media queries –ominaisuutta hyödyntämällä. Myös hyvä käytettävyys ja laitteiden ominaispiirteiden – kuten kosketusnäyttöjen – huomioiminen on tärkeää. Suuri osa responsiivisen sivuston toiminnasta kuitenkin tulee jollain tavalla CSS:n kautta, vaikka kaikkeen ei media queries -ominaisuutta käytettäisikään. Yksi esimerkki SpeksiYön kohdalla on ensimmäisen lohkon iso teksti, jossa ilmoitetaan tapahtuman ajankohta (kuva 18). Tämä tekstinpätkä on toteutettu kuvana sen rivityksen pitämiseksi varmasti samanlaisena jokaisessa selaimessa ja kaikilla näytöillä.

Kuva 18. Sivuston ensimmäisessä lohkoalueessa näkyvä kuva.

Kyseinen kuva (kuva 18) on kuitenkin 802 pikseliä leveä, joten jos sivustoa katsottaisiin laitteella jonka näyttö on kapeampi kuin 802 pikseliä, ulkoasu hajoaisi ja/tai kuva rajautuisi osittain piiloon. Tämän korjaamiseksi ei tarvita kovinkaan monimutkaista ratkaisua, sillä kuvalle tarvitsee vain antaa CSS-ohjeessa määritelmä, jonka mukaan kuvan maksimileveys saa olla korkeintaan 100 prosenttia sen isäntäelementin leveydestä:

max-width: 100%;

Näin ollen, kun sivusto kaventuu näytön kaventuessa, myös kuva kaventuu, sillä tämän säännön takia kuva ei voi missään tilanteessa olla isäntäelementtiään leveämpi. Kuvan leveyden muuttuessa kuvan korkeus pysyy automaattisesti suhteessa leveyteen, joten korkeutta ei tarvitse erikseen määritellä. Vaikka tämä on hyvin yksinkertainen lisäys CSS-ohjeisiin, se on silti olennainen osa sivuston kokonaisvaltaista responsiivisuutta. Nyrkkisääntönä voidaankin pitää, että jokainen sivustolla oleva kuva saa olla korkeintaan 100 prosenttia isäntäelementtinsä leveydestä, jolloin varmistutaan siitä, että kaikki kuvat näkyvät kokonaisuudessaan hajottamatta sivuston ulkoasua. Asiat eivät tietenkään ole aina näin yksinkertaisia, mutta tämä sääntö on hyvä muistaa lähtökohtana ja periaatteena.

6.3 jQueryn hyödyntäminen SpeksiYö-tapahtuman verkkosivustolla

jQuery on vuonna 2006 julkaistu avoimen lähdekoodin JavaScript-kirjasto, joka soveltuu esimerkiksi erilaisten toimintojen käsittelyyn, animaatioiden tekemiseen ja Ajax-sovellusten toteuttamiseen. Sen ideana on yksinkertaistaa ja helpottaa JavaScriptin kirjoitusprosessia (jQuery 2014). jQuery on tällä hetkellä maailman suosituin JavaScript-kirjasto (w3techs 2014). Myös SpeksiYö-tapahtuman verkkosivustolla hyödynnetään jQuerya.

Tästä ehkä näkyvin ja oleellisin esimerkki on yläreunan navigaation toiminta. Kun navigaatiosta painaa jotakin linkkiä, sivu rullautuu automaattisesti ja animoidusti tuon linkin osoittamaan lohkoalueeseen; jos esimerkiksi painat *ohjelma*-linkkiä, sivu

rullautuu nätisti alaspäin kunnes *ohjelma*-lohkoalue on näkyvissä. Tämä toiminnallisuus on toteutettu HTML:n ja jQueryn yhteiskäytöllä.

Ensinnäkin jokainen navigaatiovalikossa oleva linkki on osoitettu omaan lohkoalueeseensa hyödyntämällä <a>-tagin *href*-attribuuttia. Esimerkiksi *ohjelma*-linkki on toteutettu näin:

Ohjelma

Tässä esimerkissä linkki siis osoitetaan kohteeseen *#fourth*. Sivusto on rakennettu niin, että jokainen lohkoalue on saanut oman ID:n, ja nämä ID:t on nimetty järjestyksessä *first*, *second*, *third*, *fourth* ja *fifth*. HTML-pohjassa esimerkiksi *säännöt*-lohkoalue (joka on järjestyksessään neljäs) alkaa tällaisella tagilla:

<article id="fourth"...

Näin syntyy selvä yhteys linkistä lohkoalueeseen johon linkin halutaan vievän. Tämä yksinään ei kuitenkaan riitä, vaan varsinainen toiminnallisuus on toteutettava jQueryn avulla. En kuitenkaan näe tarpeellisena lähteä selventämään sanatarkasti miten jQuery toimii, sillä se ei ole tässä raportissa olennaista, mutta pyrin selventämään tämän esimerkkitapauksen toteutuksen mahdollisimman selkeästi.

Tämä automaattinen rullausefekti pohjautuu siihen, että jQuerya hyödyntäen käyttäjän valitsemasta linkistä poimitaan linkin *href*-attribuutin arvo (eli tässä esimerkissä *#fourth*). Tämän jälkeen etsitään <article>-elementti (eli lohkoalue), jonka ID vastaa juuri äsken poimittua tietoa. Näin olemme myös jQueryn avulla saaneet yhteyden näiden kahden elementin välille. Kun tämä yhteys on löydetty, sivuston koko sisältö animoidaan rullautumaan niin, että kyseisen lohkoalueen yläreuna ja navigaatiopalkin alareuna kohtaavat; toisin sanoen linkin osoittama lohkoalue tuodaan käyttäjälle näkyviin animaation avulla. Kaikki tämä tapahtuu tietysti niin nopeasti, ettei käyttäjä ehdi huomaamaan muuta kuin lopullisen toiminnallisuuden, eli animoidun liikkeen.

Tämä on SpeksiYö-tapahtuman verkkosivustolla oikeastaan ainoa käyttäjälle oleellinen ja todella näkyvä jQuerylla toteutettu toiminnallisuus. jQuerya on kuitenkin hyödynnetty myös joihinkin ulkoasullisiin seikkoihin, kuten lohkoalueiden korkeuden laskemiseen ja säätämiseen näyttöön sopivaksi sekä kilpailusääntöjen näyttämiseen/piilottamiseen animoidusti.

7 SIVUSTON SIIRTÄMINEN DRUPALIIN JA JULKAISU

Kun sivupohja oli viimeistelty loppuun asti, eli HTML-pohja sisälsi kaikki tarpeelliset elementit, CSS-tyyliohjeet oli määritelty riittävän kattaviksi ja kaikki toiminnallisuudet oli toteutettu kokonaisuudessaan, aloitin sivuston siirtämisen sisällönhallintajärjestelmään. Sivuston lopullinen kansiorakenne näytti tältä:

- Pääkansio
    - index.html (tiedosto, varsinainen sivupohja)
    - js (kansio)
        - JavaScript-tiedostot
    - css (kansio)
        - CSS-tyyliohjetiedosto
    - gfx (kansio)
        - Kaikki kuvatiedostot

7.1 Drupalin valmistelu ja teeman luominen

Kun olin valmistellut oman materiaalini julkaisukuntoon, latasin ja asensin Drupalin omalle koneelleni ja siirsin sen sitä kautta palvelimelle, jossa SpeksiYö-tapahtuman verkkosivusto tultaisiin julkaisemaan. Asensin Drupalin palvelimelle toistaiseksi salasanan taakse estääkseni ei-toivotun liikenteen ennen sivuston lopullista julkaisua.

Tekemäni sivupohja ei toimisi Drupalissa täysin sellaisenaan, vaan se täytyi muokata Drupalin vaatimaan muotoon, jotta se toimisi oikein. Ensinnäkin minun täytyi

irrottaa varsinainen sisältö itse teemasta, eli ulkoasusta, sillä sisältöhän tullaan myöhemmässä vaiheessa syöttämään Drupalin hallintapaneelin kautta uudelleen. Sisältö ei siis saa olla kiinni sivupohjassa. Jäljelle jäisi ainoastaan sivuston peruselementit ilman tekstejä, kuvia ja muuta sellaista sisältöä, mitä halutaan mahdollisesti muokata myöhemmin. Toisekseen jäljelle jäänyt sivupohja täytyi jakaa ja muotoilla niin, että Drupal ymmärsi sen olevan yksi kokonainen teema. Kolmannekseen tuo sivupohja täytyi vielä valmistella niin, että se pystyi ottamaan vastaan hallintapaneelin kautta lisätyt sisällöt. Tämä kaikki on osa *teeman luomista*.

Drupal antaa kehittäjälle mahdollisuuden lisätä sivustolle *lohkoja*, joka on erittäin hyödyllinen ominaisuus, kun luodaan one page -mallilla toteutettu sivusto. Lohkot antavat kaikessa yksinkertaisuudessaan mahdollisuuden merkitä sivupohjaan kohdan, johon voidaan lisätä sisältöä hallintapaneelin kautta, oli sisältö sitten mitä tahansa. Toisin sanoen minun täytyi lisätä lohkoa varten tarkoitettu *tunniste* jokaiseen sellaiseen kohtaan, mihin halusin tuottaa sisältöä hallintapaneelin kautta. Järkevin vaihtoehto oli mielestäni lisätä yksi tällainen tunniste jokaiseen lohkoalueeseen; *ohjelma*-kohtaan yksi tunniste, *ota yhteyttä* -kohtaan yksi ja niin edelleen. Tunniste merkitään sivupohjaan näin:

<?php print render($page['lohkon_nimi']); ?>

Tunniste perustuu PHP-ohjelmointikieleen. Lohkon nimeksi voi antaa periaatteessa mitä vain, mutta paras käytäntö on käyttää kuvailevaa nimeä, jotta lohkon tunnistaa myös jatkossa niin sivupohjassa kuin hallintapaneelissakin, josta lohkot löytyvät teeman asennuksen ja käyttöönoton jälkeen automaattisesti. Ennen kuin korvasin jokaisen sisältökohdan tällaisella tunnisteella, tallensin tietenkin varsinaisen sisällön väliaikaisesti toiseen tiedostoon, josta sain myöhemmässä vaiheessa poimittua sen kätevästi lisätessäni sisältöä hallintapaneelin kautta.

Pilkoin sivupohjan myös kahteen osaan, sillä Drupal käyttää teemoissaan useampia .php-tiedostoja, joiden yhdistelmästä lopullinen sivupohja koostuu. Tämän jaon jälkeen toisessa tiedostossa oli sivuston niin sanonut näkymättömät osat, kuten <head>-tagit ja niiden välissä oleva sisältö, ja toisessa tiedostossa oli varsinainen sivuston rakenne, eli <body>-tagit ja niiden väliin jäävä sisältö elementteineen. Tämä

helpottaa sivuston rakentamista esimerkiksi siinä tilanteessa, jossa sivuja on useita erilaisia (etusivu, alasivu, blogisivu), mutta näkymättömät osat halutaan pitää samana kaikilla sivuilla. Jos näkymättömiin osiin tulee myöhemmin joitakin muutoksia, kuten esimerkiksi hakukoneoptimointia, riittää kun muutokset tehdään tuohon yhteen tiedostoon, koska kaikki sivuston sivut käyttävät tuota samaa tiedostoa pohjassaan.

Lopulta tallensin nämä sivupohjan osat Drupalin oletuksena käyttämillä tiedostonimillä:

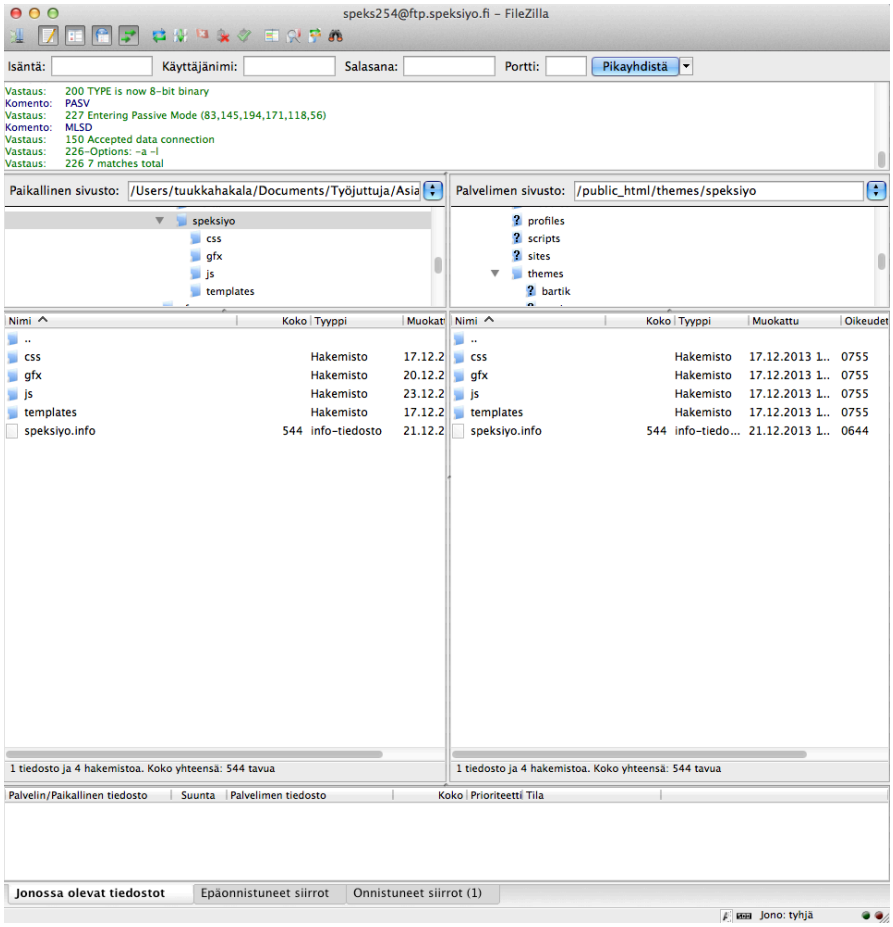
- html.tpl.php
- page--front.tpl.php

Ensimmäinen tiedosto sisältää ”näkymättömät” osat ja toinen taas sisältää sivuston varsinaisen rakenteen. Tiedostonimissä näkyvä tpl-lyhenne viittaa sanaan *template*. Tiedostojen tallentamista varten minun täytyi myös luoda uudelle teemalle oma kansio. Loin Drupalin juurihakemistosta löytyvän *themes*-kansion sisälle uuden *speksiyo*-nimisen kansion, jonka sisälle loin vielä seuraavat kansiot:

- js
- css
- gfx
- templates

Nämä kaksi tiedostoa siirsin *templates*-kansioon. Kolmeen muuhun kansioon siirsin tiedostot suoraan niistä kansioista, joissa alkuperäisen sivupohjanikin materiaalit sijaitsivat. Nyt minulla oli siis kasassa Drupalia varten muokattu sivupohja, kaikki oheistiedostot ja oikeanlainen kansiorakenne. Tämän lisäksi Drupal vaatii vielä jokaiselta teemalta info-tiedoston, joka sisältää perustiedot teemasta (kuten teeman nimen), mutta myös kaikkien käytettävissä olevien lohkojen nimet. Info-tiedosto on hyvin yksinkertainen kirjoittaa ja siihen löytyy valmiita pohjia ja kattavat ohjeet Drupalin omilta verkkosivuilta.

Kun info-tiedostokin oli tallennettu *speksiyo*-teemakansioon, edessä oli teeman asentaminen ja testaaminen itse Drupalissa. Teema täytyy tietysti viedä ensin palvelimelle, jossa Drupal sijaitsee, joten FTP-ohjelmaa hyödyntäen siirsin luomani teemakansion palvelimelle *themes*-kansioon (kuva 19).

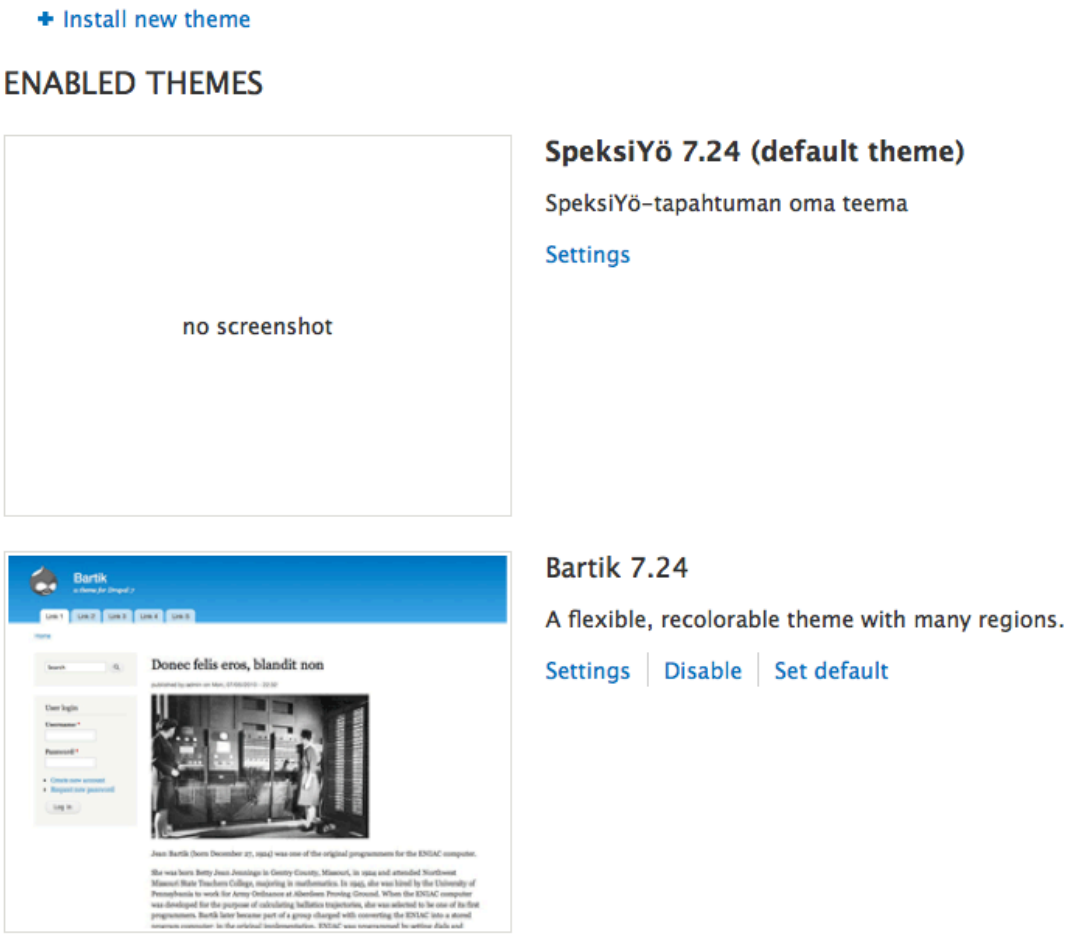


Kuva 19. FileZilla-ohjelman käyttöliittymää; vasemmalla näkyy omalla koneellani sijaitsevat tiedostot ja oikealla näkyy ulkoisella palvelimella olevat tiedostot.

7.2 Teeman käyttöönotto

Kun teemaan liittyvät tiedostot oli siirretty palvelimelle oikeaan kansioon, teema täytyi vielä aktivoida Drupalin hallintapaneelin kautta. Hallintapaneelin *appearance*-välilehdellä (kuva 20) näkyvät kaikki valittavissa olevat teemat, ja halutun teeman käyttöönotto tapahtuu yksinkertaisesti valitsemalla *set default*, eli asettamalla teema

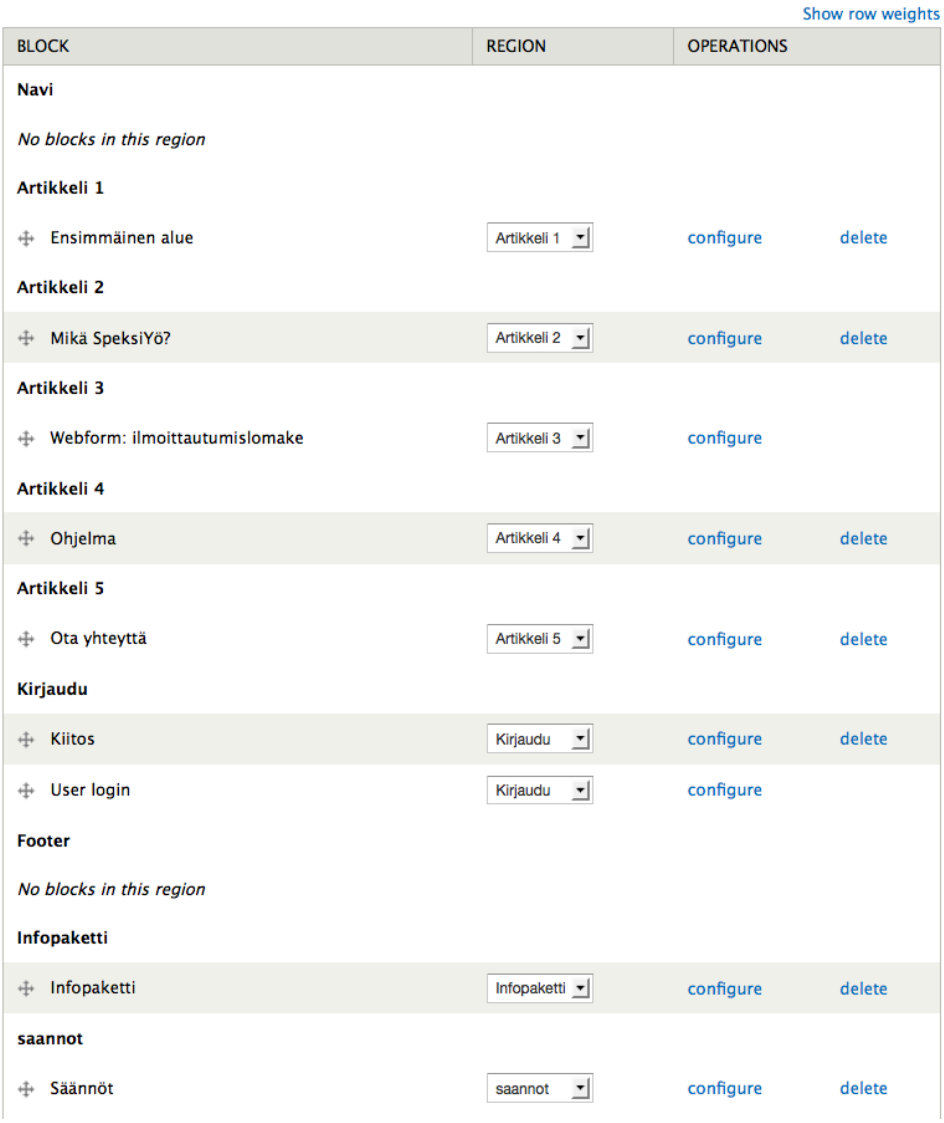
oletusteemaksi. Kun tämä oli tehty, tarkistin teeman ja sivuston toimivuuden vielä siltä varalta, että se olisi kaivannut joitakin viimehetken viilauksia.



Kuva 20. Teeman valinta Drupalin hallintapaneelissa. ”SpeksiYö” on asetettu oletusteemaksi. ”Bartik” on yksi Drupalin mukana tulevista valmiista teemoista.

7.3 Sisällön syöttäminen sivustolle

Kun tekemäni teema oli otettu käyttöön ja siihen kuuluvaan sivupohjaan oli merkitty sisältö- eli lohkokohdat, oli aika siirtää varsinainen sisältö sivustolle. Koska sisältökohdat (eli lohkot) oli merkitty sivupohjaan ja ne oli myös nimetty teemaa varten laadittuun info-tiedostoon, nämä kohdat näkyivät myös hallintapaneelissa (kuva 21).

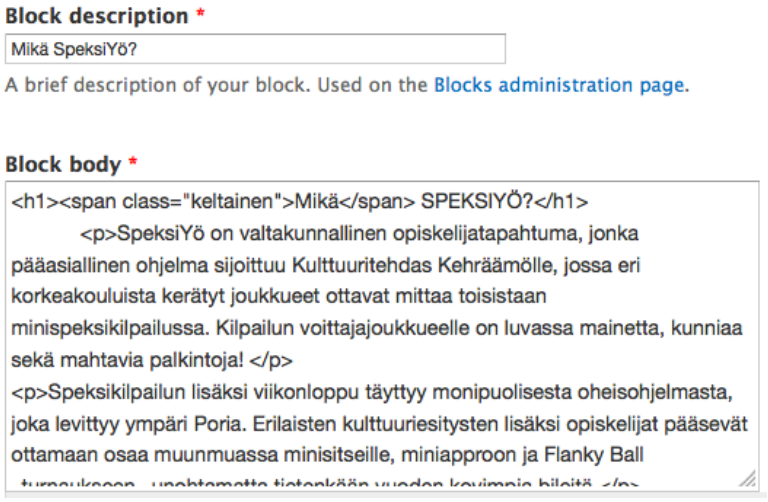


Show row weights

BLOCK	REGION	OPERATIONS	
Navi			
No blocks in this region			
Artikkeli 1			
Ensimmäinen alue	Artikkeli 1	configure	delete
Artikkeli 2			
Mikä SpeksiYö?	Artikkeli 2	configure	delete
Artikkeli 3			
Webform: ilmoittautumislomake	Artikkeli 3	configure	
Artikkeli 4			
Ohjelma	Artikkeli 4	configure	delete
Artikkeli 5			
Ota yhteyttä	Artikkeli 5	configure	delete
Kirjaudu			
Kiitos	Kirjaudu	configure	delete
User login	Kirjaudu	configure	
Footer			
No blocks in this region			
Infopaketti			
Infopaketti	Infopaketti	configure	delete
saannot			
Säännöt	saannot	configure	delete

Kuva 21. Lohkot hallintapaneelissa.

Jokaisen lohkon sisältöä pääsee muokkaamaan valitsemalla *configure*. Se sisältö – oli se mitä tahansa – mikä tuohon lohkoon tallennetaan, näkyy sivustolla siinä kohdassa, mihin kyseisen lohkon tunniste on lisätty (kuvat 22 ja 23).



Kuva 22. Toisen lohkoalueen sisältö hallintapaneelissa. Sisältö on tallennettu HTML-muodossa, mutta se voitaisiin tallentaa tarvittaessa myös muotoiltuna tekstinä, mikä olisi hyvä vaihtoehto jos sivustolla olisi useampia päivittäjiä.

Kuva 23. Sama sisältö kuin kuvassa 22 tulostettuna sivupohjaan, eli tältä kyseinen sisältö näyttää selaimessa.

Tallensin jokaiseen lohkoon, paitsi ilmoittautumislomakkeen sisältävään lohkoon, niihin kuuluvan sisällön ja viimeiseksi tarkistin sivuston puolelta linkitykset, värit, fontit, kappalejaot ja ulkoasun rakenteen yleisesti, jotta sivusto varmasti toimii niin

kuin sen on alun perin suunniteltukin toimivan. Tässä kohtaa jätin ainoastaan ilmoittautumislomakkeen asentamatta, sillä halusin toteuttaa sen hyödyntäen Drupalille suunniteltua *Webform*-moduulia.

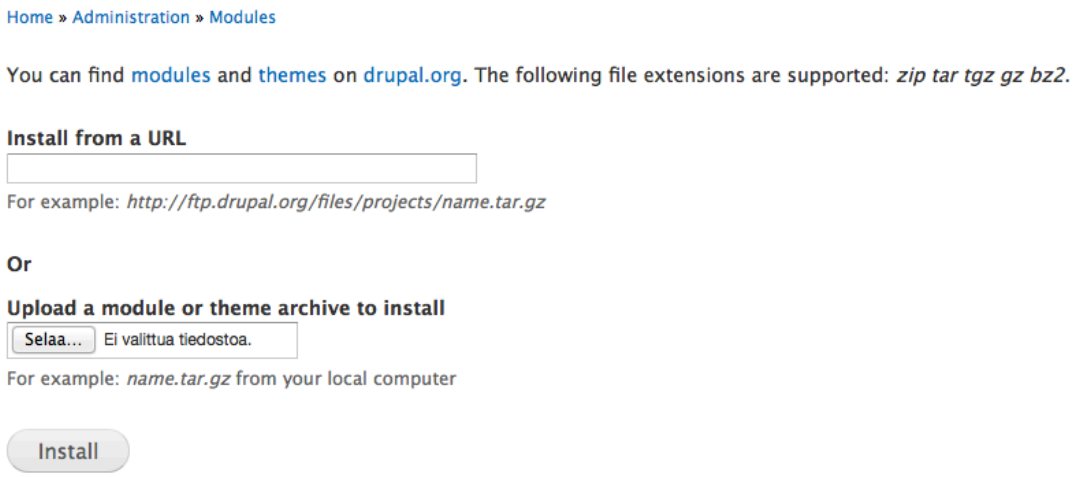
7.3.1 Ilmoittautumislomakkeen asentaminen

Drupal sisältää paljon erilaisia toimintoja, jotka on rakenteen selkeyttämiseksi jaettu *moduuleihin*. Moduuleja voivat olla esimerkiksi kommentit, foorumit ja gallupit. Sivuston ylläpitäjä voi kytkeä tarpeettomat moduulit pois päältä hallintapaneelin kautta, mutta hän voi vastaavasti asentaa uusia moduuleja, joita ovat kehittäneet esimerkiksi muut Drupalin käyttäjät. Perustarpeet kattavat moduulit tulevat kuitenkin oletuksena Drupalin asennuksen mukana. (Drupal.fi 2008.)

Lomake on yksi toiminto, joka on mahdollista ja järkevääkin toteuttaa moduulina. Toinen vaihtoehto olisi ohjelmoida vastaava lomake itse hyödyntäen PHP:ia, jQuerya ja HTML:iä, mutta moduulin kautta tämä on useammassa tapauksessa vaivattomampaa ja nopeampaa.

Päädyin lopulta käyttämään *Webform*-nimistä moduulia, jonka löysin käyttämällä Drupalin sivustolta löytyvää hakukonetta. Kyseinen moduuli täytti kaikki SpeksiYö-tapahtuman ilmoittautumislomakkeelta vaaditut ominaisuudet; yksinkertainen tapa luoda lomake, täytetyn lomakkeen tietojen tarkistus, virheellisesti täytetyistä kentistä ilmoittaminen ja tietojen lähettäminen eteenpäin (Drupal.org 2013).

Moduulin asentaminen oli tässä tapauksessa hyvin yksinkertaista. Moduulin asennustiedoston sai ladattua suoraan Drupalin sivuston kautta, ja tämän tiedoston pystyi avaamaan SpeksiYö-tapahtuman sivuston hallintapaneelin kautta (kuva 24), jolloin Drupal asensi moduulin automaattisesti. Tämän jälkeen moduuli löytyi hallintapaneelin *modules*-välilehdeltä, jonka kautta sen pystyi aktivoimaan ja sen asetuksia muuttamaan.

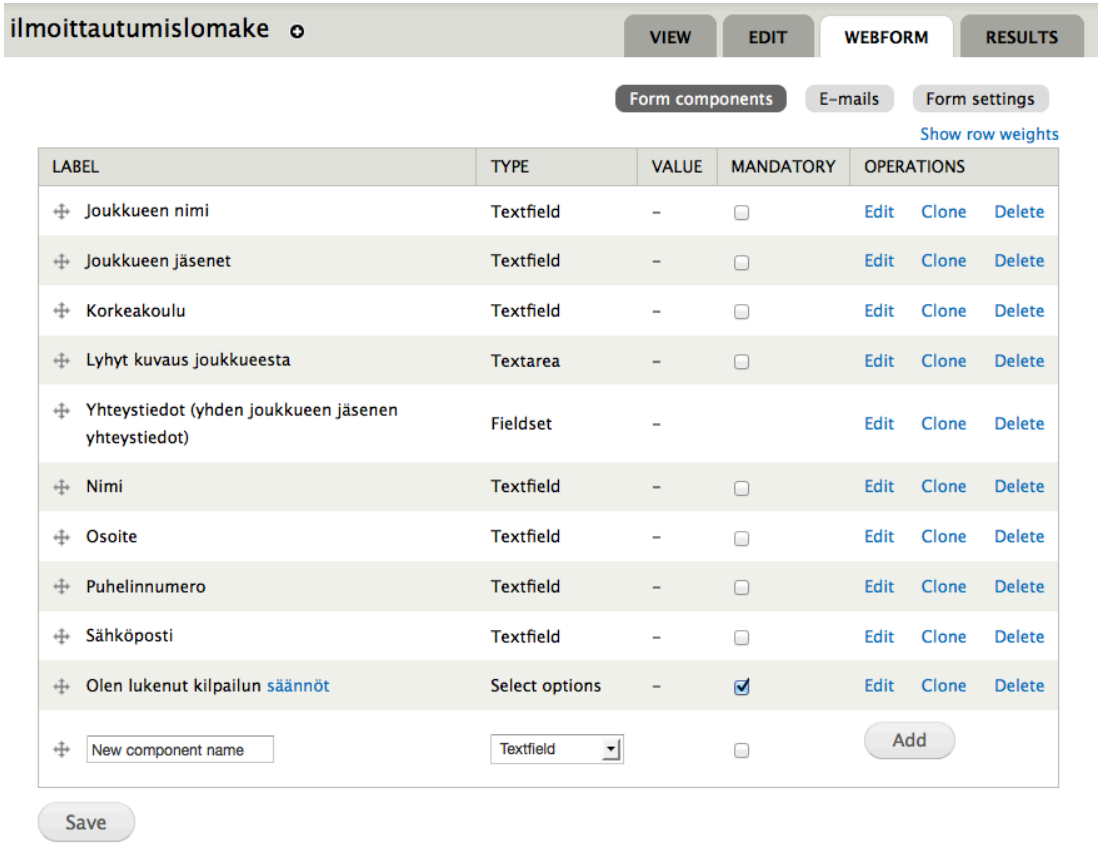


Kuva 24. Moduulin asentaminen hallintapaneelin kautta.

Kun moduuli oli asennettu ja otettu käyttöön, se ilmestyi myös yhdeksi vaihtoehtoiseksi sisältömuodoksi. Pystyin siis luomaan sisältöä, jonka muoto oli *lomake*. Käytännössä tämä tarkoitti sitä, että loin hallintapaneelia hyödyntäen haluamani lomakkeen, tallensin sen ja osoitin tuon sisällön (eli lomakkeen) haluamaani lohkoalueeseen, joka oli merkitty sivupohjaan. Koska tämä kaikki tapahtui nimenomaan Drupalille suunnitellun moduulin kautta, koko lomakkeen luominen onnistui selkeän käyttöliittymän avulla erilaisia valintoja tehden. Varsinaista ohjelmointia ei siis tarvittu, vaan kaikki lomakkeen sisältämät kentät ja lomakkeen edelleenlähetyksen asetukset ja muut vastaavat toiminnot onnistuivat vaivattomasti (kuva 25).

Haastetta toi ainoastaan lomakkeen muotoilu, sillä Webform-moduulilla luodussa lomakkeessa on oletuksena tietynlainen rakenne, joka oli täysin erilainen kuin mallisivustolle suunnittelemani lomakkeen rakenne, mikä aiheutti pientä hienosäätöä CSS-tyyliohjeiden kanssa. Webform-moduulilla luotu lomake sisältää esimerkiksi paljon -elementtejä, joita minun suunnittelemassani lomakkeessa ei ollut yhtään. Todella kireän aikataulun takia minulla ei kuitenkaan ollut aikaa lähteä tutkimaan, olisiko tämän rakenteen saanut jollakin tavalla ”nollattua”, joten päätin muokata lomakkeen haluamani tyyliseksi tehden muutoksia ainoastaan CSS-tyyliohjeisiin. Lopulta nämä muutokset olivat hyvin yksinkertaisia ja sisälsivät

lähinnä ohjeiden uudelleennimeämistä, ja uskon voittaneeni aikaa verrattuna siihen, että olisin lähtenyt hakemaan mahdollisuutta nollata Webform-moduulin luoman lomakkeen rakenne.



Kuva 25. Webform-moduulin asetuksia hallintapaneelissa. Tässä kuvassa näkyvät kaikki tekstikentät ja muut lomakkeen osat, joita ilmoittautumislomakkeelle oli luotu. Jokaista kenttää pääsee muokkaamaan valitsemalla *edit*. Lomakkeen muita asetuksia pääsee muokkaamaan yläreunassa näkyvien välilehtien kautta.

7.4 Varsinainen julkaisu

Kun lomake oli paikoillaan, kaikki sivustolle suunniteltu sisältö ohjelmaa lukuun ottamatta oli näkyvillä. Ohjelma ei ollut vielä tässä kohtaa valmis, mutta sen tilalle oli laitettu täytesisältöä. Tässä vaiheessa tein vielä viimeiset testaukset eri selaimilla

ja laitteilla ja varmistin ilmoittautumislomakkeen toimivuuden, mutta koska kaikki toimi kuten pitikin, laitoin sivuston heti asiakkaalle nähtäväksi.

Asiakas vastasi hyvin pian ja kertoi olevansa hyvin tyytyväinen sivustoon, mutta ohjelma ei ollut vieläkään valmis. Päätimme lopulta, että sivusto julkaistaan ensin ilman ohjelmaa, joka lisättäisiin sivustolle sen valmistuttua (Turtiainen sähköposti 30.12.2013). Sivusto julkaistiin lopulta 2.1.2014. Tämä tarkoitti käytännössä sitä, että kaikki sivustoon liittyvät tiedostot siirrettiin palvelimella salasanan takaa palvelimen juurihakemistoon, eli siihen hakemistoon, jonka sisällön sivustolla vierailevat oletuksena näkevät. Sivuston varsinainen julkaisu oli lopulta se pienin ja helpoin osuus koko projektissa, mutta julkaisun valmistelu ja sivuston integroiminen sisällönhallintajärjestelmään veivät yllättävän paljon aikaa.

8 PROJEKTIN ARVIOINTI

8.1 Aikataulussa pysyminen

Sivuston julkaisu oli alun perin ajoitettu joulukuun loppuun. Tämä oli suhteellisen kova aikataulu, ottaen huomioon, että sivustoa lähdettiin rakentamaan vasta joulukuun aikana. Vaikka olisi helppo kuvitella, että näin pienimuotoisen sivuston saa julkaistuksi nopeallakin aikataululla, totuus voi olla kuitenkin toisenlainen. Sivuston laajuus ei välttämättä kerro koko totuutta sivuston vaatimuksista tai toteutuksen monipuolisuudesta. Toinen tekijä oli sekä minun että ulkoasun suunnitelleen Hannulan omat aikataulut, sillä työskentelimme molemmat täysipäiväisesti tämän projektin ulkopuolellakin. Myös joulukuun loppupuolelle ajoittuvat pyhät oli otettava huomioon.

Sivusto valmistui aivan joulukuun lopussa, mutta se julkaistiin kuitenkin vasta tammikuun puolella 2.1.2014, joten todellista ”viivästymistä” tuli vain pari päivää. Osittain tämä viivästyminen johtui omasta epävarmuudestamme julkaista sivusto ilman ohjelmaa, joka ei ollut vielä valmis. Toisaalta, jos sivusto olisi tehty ilman sisällönhallintajärjestelmää, aikaa olisi voitettu useampia päiviä johtuen siitä, että

sivuston siirtäminen Drupaliin ja kaikkien toimintojen valmiiksi saattaminen sen kanssa vei yllättävän paljon aikaa. Jälkeenpäin ajatellen olisikin ollut viisaampaa julkaista sivusto ilman sisällönhallintajärjestelmää jo ennen joulun pyhiä, eikä lähteä siirtämään sivustoa Drupalin yhteyteen pyhien aikana, kun aikataulu oli muutenkin tiukka.

Kokonaisuudessaan olen kuitenkin tyytyväinen siihen, millä aikataululla saimme sivuston toteutettua, vaikka olisin varmasti pystynyt tekemään joitakin parempiakin päätöksiä ajan säästämiseksi tietyissä projektin vaiheissa.

8.2 Ongelmat, haasteet ja niiden ratkaiseminen

Kohtasin tämän projektin aikana suhteellisen vähän varsinaisia ongelmia. Haasteita kylläkin oli, sillä osa sivustolle suunnitelluista toiminnoista olivat sellaisia, joita en ollut aikaisemmin toteuttanut muiden projektien yhteydessä, ainakaan juuri siinä muodossa kuin ne tällä sivustolla tulisivat olemaan. Otin kuitenkin haasteet vastaan ja löysin keinot toteuttaa nämä toiminnallisuudet juuri sellaisina kuin ne oli suunniteltukin.

Oikeastaan ainoat varsinaiset ongelmat tulivat siinä vaiheessa, kun aloitin sivuston siirtämisen Drupaliin. Nämä ongelmat johtivat juurensa silti omasta kokemattomuudestani kyseisen sisällönhallintajärjestelmän kanssa, sillä olin käyttänyt Drupalia aikaisemmin vain vähän ja siitäkin oli jo useampi vuosi aikaa. Vaikka Drupalin perusidea toimintoineen olikin hallussa ja hallintapaneeli tuttu, oli esimerkiksi kokonaan uuden teeman ja sivupohjan rakentaminen ja vieminen juuri Drupaliin minulle suhteellisen vieras prosessi.

Vaikka tämäkään ei muodostunut varsinaiseksi ongelmaksi, on uuden asian opetteleminen kesken projektin aina aikaa vievää ja aikataulun ollessa tiukka se saattaa aiheuttaa turhaa hätäilyäkin, mikä voi kostautua huonosti toteutettuina kokonaisuuden osina. Olen kuitenkin oppinut kokemuksen kautta, että tällaisessa tilanteessa kaikista tehokkain ja nopein tapa saada tilanne ratkaistuksi on opetella uusi asia rauhassa ja kunnolla, eikä lähteä hakemaan ratkaisua nopeasti ja

huolimattomasti. Jälkimmäinen tapa nimittäin aiheuttaa mitä todennäköisimmin vain lisää haasteita ja suoranaisia ongelmia, joiden ratkaiseminen vie taas oman aikansa.

Tässä tapauksessa hain ohjeita Drupalin oman yhteisön kautta; kyseinen yhteisö tarjoaa runsaasti ohjeita lähes kaikkiin mahdollisiin tilanteisiin, joihin Drupalin kanssa voi joutua. Kysehän oli myös vain siitä, että minun piti oppia käyttämään uutta työkalua saavuttaakseni sen, minkä olen jo saavuttanut monien muidenkin työkalujen – eli sisällönhallintajärjestelmien – kanssa; sivustolle suunnitellun ja rakennetun teeman ja sivupohjan siirtäminen kyseiseen sisällönhallintajärjestelmään.

8.2.1 Haasteiden ja ongelmanratkaisun kautta tapahtuva oppiminen

Hyvien ohjeiden ja omien kokeilujen kautta löysin ratkaisun yllä mainittuunkin haasteeseen. Toisaalta, kuten aikaisemminkin tässä raportissa mainitsin, olisin voittanut huomattavasti aikaa, jos en olisi lähtenyt viemään sivustoa ollenkaan Drupaliin. Mutta, jos tekisin saman projektin – tai minkä tahansa muun projektin – nyt uudelleen, tuokin vaihe projektissa sujuisi kokemuksen takia huomattavasti nopeammin. Tätä voidaan siis pitää tietynlaisena kehityksenä omassa osaamisessani ja ammattitaidossani. Pidänkin haasteiden kautta oppimista yhtenä tehokkaimmista tavoista oppia uusia asioita ja kehittää omaa ammattitaitoa.

Omasta mielestäni on myös hienoa päästä toteuttamaan verkkosivustoja jonkun muun suunnitteleman ulkoasun pohjalta. Tämä asetelma tuo eteen aina uusia haasteita, sillä jos tekninen toteuttaja myös suunnittelee ulkoasun itse, hän saattaa tietyissä tilanteissa mennä sieltä mistä aita on matalin, eli suunnitella ulkoasun pohjautuen omaan senhetkiseen ammattitaitoonsa verkkosivuston kehittäjänä. Hän ei siis haasta itseään niin helposti, koska tietää ettei välttämättä pysty vastaamaan niihin haasteisiin opettelematta uusia tekniikoita tai työkaluja. Voi olla paljon houkuttelevampaa suunnitella jotain sellaista, minkä tietää pystyvänsä toteuttamaan helposti.

Kun ulkoasun suunnittelee henkilö, joka ei tiedä teknisen toteuttajan rajoituksia – tai ei yksinkertaisesti piittaa niistä, kuten ei pidäkään – lopputuloksena on tilanne, jossa

tekninen toteuttaja joutuu asettamaan oman ammattitaitonsa kyseenalaiseksi ja lopulta kehittämään sitä. Mielestäni tämä on erittäin tehokas tapa kehittyä ja saada uusia haasteita, joita ei itse osaisi välttämättä asettaa itselleen.

8.3 Palaute asiakkaalta

Saamani asiakaspalaute oli lyhytsanaista, mutta positiivista (Turtiainen sähköposti 13.1.2014). Sivustoa, sen toiminnallisuuksia tai rakennetta ei muutettu enää sen jälkeen kun olin lähettänyt sen asiakkaalle katsottavaksi, eikä toiminnallisuuksissa ilmennyt minkäänlaisia puutteita tai vikoja. Vaikka palautetta on ainakin tässä vaiheessa tullut suhteellisen vähän, on hyvä huomioida, että kaikki palaute on ollut pelkästään positiivista. Jos sivustossa olisi jotakin vikaa tai joku sen toiminnallisuuksista ei toimisi oikealla tavalla, siitä annettaisiin varmasti aiheellista palautetta. Myös sisältöön tulleiden päivitysten osalta olen saanut positiivista palautetta toimintani nopeudesta, mikä onkin mielestäni tärkeä osa ylläpitäjän tehtävää.

8.4 Itsearviointi

Onnistuin omasta mielestäni projektissa kokonaisuudessaan erittäin hyvin. Tärkein tavoitteeni oli toimittaa asiakkaalle aikataulun puitteissa valmis sivusto, jolle ei tarvitse tehdä korjauksia enää julkaisun jälkeen tai jonka toiminnallisuuksissa ei paljastu toimintavirheitä. Äärimmäisen tärkeänä pidin varsinkin ilmoittautumislomakkeen toimivuutta.

8.4.1 Mitä olisin voinut tehdä toisin

Pelkästään tätä projektia ajatellen olisin voinut jättää sisällönhallintajärjestelmän kokonaan pois suunnitelmista. Aluksi meinasinkin, ettei sisällönhallintajärjestelmää kannata tuoda tällä aikataululla mukaan ollenkaan, sillä sivuston integroimiseen tällaiseen järjestelmään menisi joka tapauksessa jonkin verran aikaa. Vastaavasti sivuston käyttöikä, luonne ja rakenne eivät välttämättä vaatisi

sisällönhallintajärjestelmää ollenkaan, varsinkin kun otetaan huomioon, että sivuston sisältöön liittyvistä päivityksistä vastasin kokonaisuudessaan minä. Todennäköisesti olisin tehnyt jokaisen päivityksen aivan yhtä nopeasti suoraan staattiseen sivupohjaan kuin Drupalin hallintapaneelinkin kautta. Sisällönhallintajärjestelmän pois jättämistä olisi tukenut myös se, että tapahtuman jälkeen sivustolle ei enää ole käyttöä, joten minun ei tarvitse ajatella esimerkiksi päivittämisen helppoutta jatkossa.

Usein on myös pieniä asioita joita projektin valmistuttua miettii uudelleen siltä pohjalta, olisiko jotakin voinut tehdä paremmin tai olisiko jonkin toiminnallisuuden voinut toteuttaa tehokkaammin. Tämä koskee usein hyvin pieniä osia esimerkiksi CSS-tyyliohjeissa tai JavaScriptillä toteutetuissa toiminnallisuuksissa. On hyvä olla itsekriittinen ja ottaa siitä opiksi. Vaikka sivusto toimisi kuten on tarkoitettukin ja niin HTML, CSS, JavaScript kuin kaikki muutkin sivustolla käytetyt merkintä- ja ohjelmointikielet olisi kirjoitettu oikein, useissa tapauksissa voi silti toimia vieläkin tehokkaammin. Tämä tarkoittaa esimerkiksi oman työnkulun ja työtapojen tarkastelemista: missä järjestyksessä asiat voisi olla helpompi tehdä, pitäisikö CSS-ohjeideni olla tiiviimpiä, voinko hyödyntää HTML5:n tuomia mahdollisuuksia enemmän ja niin edelleen.

Tämän projektin osalta näitä asioita jäi mieleeni pääasiassa CSS:in liittyvistä seikoista. Olisin varmasti voinut tiivistää tyyliohjeitani entisestään ja lyhentää niiden yhteistä pituutta jonkin verran. Tärkeintähän on, että sivusto toimii oikein, mutta itse arvostan tiivistä, yksinkertaista ja tehokasta CSS:n ja merkintä- ja ohjelmointikielien käyttöä. Toisaalta olisin voinut myös käyttää CSS:n media queries –ominaisuutta järjestelmällisemmin ja näyttävämmin, mutta olen silti lopputulokseen tyytyväinen. Osittain tämän sivuston CSS-ohjeiden rakentamista vaikeuttivat myös sivustolla käytettävät monet eri fonttikoot, värit ja muut vastaavat yhtenäisyyttä selkeästi rikkovat tekijät.

8.4.2 Missä onnistuin

Pysyin aikataulussa suhteellisen hyvin. Sivusto oli valmis sille sovitussa ajassa, vaikka itse julkaisu vietiinkin vasta heti tammikuun alkuun. Olen myös erittäin

tyytyväinen siihen, että julkaisun jälkeen ei ole ilmennyt ongelmia toiminnallisuuksissa tai muitakaan yleisiä ongelmia sivustolla. Tärkein tavoitteeni olikin saada kehitettyä sivusto siihen pisteeseen, että se on sovitussa aikataulussa julkaisuvalmis, ja tämän jälkeen sille ei tarvitse sisältöpäivityksiä lukuun ottamatta tehdä mitään muutoksia. Saavutin tämän tavoitteen omasta mielestäni täydellisesti, vaikka olisinkin halunnut toimittaa sivuston jo hieman aikaisemmin asiakkaalle katsottavaksi.

En joutunut turvautumaan minkäänlaisiin hätäratkaisuihin sivustoa rakentaessani, vaan kaikki on tehty perustellusti ja viimeistellysti. Tämä myös näkyy sivuston toiminnallisuudessa, sillä kaikki toimii luonnollisesti ja jouhevasti. Pidän erityisesti siitä, miten joustavasti sivusto skaalautuu eri näyttökokoihin ja miten navigaation automaattinen rullaustoiminto on rakennettu.

Opin myös projektin aikana paljon uutta, pääasiassa tietysti Drupaliin liittyviä asioita. Jokaisen projektin jälkeen pitäisikin olla tunne, että pystyy lähtemään seuraavaan projektiin entistä luottavaisempana, ja tässä tapauksessa kävi ehdottomasti niin.

9 LOPUKSI

Olen toiminut web-suunnittelijana/-kehittäjänä jo jonkin aikaa ennen tämän projektin aloittamista. Siinä mielessä en voi sanoa lähteneeni tähän projektiin ainoastaan opiskelijana, vaan pikemminkin jonkinlaisena opiskelijan ja ammattilaisen risteytyksenä. Tiedän omat heikkouteni ja ne kohdat, joissa minulla on vielä kehitettävää, ja tämä projekti olikin hyvä mahdollisuus kehittää näitä kohtia.

Ensinnäkin jouduin työskentelemään paljon itsenäisemmin kuin tehdessäni töitä yrityksessä; kukaan ei hoida asiakasyhteyksiä puolestani, anna tiukkoja välitavoitteita tai kysele projektin etenemisestä. On myös tavallaan haastavampaa työskennellä vapaasti omalla ajalla kuin toimistossa tietyn aikataulun mukaan. Tämä

projekti nostikin oman vastuun tärkeyttä entistä selkeämmin esille, kun jouduin ottamaan vastuuta myös sivuston kehittämisen ulkopuolelta.

Pidin myös lähteiden etsimistä hyödyllisenä osana opinnäytetyöprojektia. Tämä laittoi minut kertaamaan asioita joita pidin itsestäänselvyyksinä ja hakemaan perusteita omille olettamuksilleni. Tämä taas tuo tietynlaista itsevarmuutta omaan tekemiseen, kun tietää tekevänsä asioita oikein ja olleensa koko ajan oikealla tiellä. Vastaavasti voi löytää aivan uusia puolia asioista, joissa on luullut olevansa oikeassa tai joiden on luullut olevan ainoa mahdollisuus. Raportin kirjoittaminen kokonaisuudessaan laittoikin minut miettimään sitä, mitä ihan oikeasti teen ja miksi teen juuri niin. Tällainen perusteleminen selkeyttää omia toimintatapoja ja antaa niille ikään kuin perustan, jolle niitä on hyvä rakentaa jatkossa.

Tämä projekti myös vahvisti haluani toimia web-kehittäjänä entisestään. Innostuin verkkosivustoista ja niiden rakentamisesta jo ala-asteikäisenä, eikä tuo innostus ole missään vaiheessa kadonnut mihinkään. Tämä ala vaatii jatkuvaa itsensä kehittämistä, mutta se vain kannustaa minua jatkamaan eteenpäin. Tämä on sitä, mitä haluan jatkossakin tehdä.

LÄHTEET

Adobe www-sivut. 2014. Ominaisuudet : Työkalut ja palvelut. Viitattu 27.1.2014. http://www.adobe.com/fi/products/creativecloud/tools-and-services.html

Arch, A. 2008. Web Accessibility for Older Users: A Literature Review. Viitattu 20.1.2014. http://www.w3.org/TR/wai-age-literature/

CSS Reset www-sivut. 2014. What Is A CSS Reset?. Viitattu 30.1.2014. http://www.cssreset.com/what-is-a-css-reset/

Dive Into HTML5 www-sivut. 2014. Canvas. Viitattu 24.1.2014. http://diveintohtml5.info/canvas.html

Drupal.fi www-sivut. 2008. Moduulit: valinnaisia toimintoja ja lisäosia. Viitattu 2.2.2014. http://drupal.fi/fi/kayttajan-opas/moduulit-valinnaisia-toimintoja-ja-lisaosia

Drupal.fi www-sivut. 2014. Mikä Drupal? Viitattu 26.1.2013. http://drupal.fi/fi/drupal-suomi

Drupal.org www-sivut. 2013. Webform. https://drupal.org/project/webform

eLab Communications www-sivut. 2014. The Mobile Dilemma: Mobile Website or Mobile App? Viitattu 23.1.2014. http://www.elabcommunications.com/blog/category/responsive-web-design/

Feldmann, C. 'Introducing Google Web Fonts'. Far Reach. 25.1.2011. Viitattu 24.1.2014. http://blog.farreachinc.com/2011/02/25/introducing-google-web-fonts/

Google Fonts www-sivut. 2014. FAQ. Viitattu 24.1.2014. https://developers.google.com/fonts/faq

Gosha, G. 2013. 15 Creative Single Page Web Designs. Viitattu 20.1.2014. http://www.sitepoint.com/15-creative-single-page-web-designs/

Gustafson, A. 2011. Adaptive Web Design: Crafting Rich Experiences with Progressive Enhancement. Easy Readers.

jQuery www-sivut. 2014. Viitattu 31.1.2014. http://jquery.com/

Korpela J. 2013. CSS3 – uudet mahdollisuudet. Jyväskylä: Docendo.

Korpela, J. 2011. HTML5 – uudet ominaisuudet. Jyväskylä: Docendo.

Kortesuo, K. 2009. Tekstiä ruudulla – Kirjoitamme verkkoon. Keuruu: Infor Oy

Marcotte, E. 2010. Responsive Web Design. Viitattu 20.1.2014. http://alistapart.com/article/responsive-web-design/

Mitchell R. 2013. Choosing an open-source CMS, part 1: Why we use Drupal. Viitattu 26.1.2014. http://www.computerworld.com/s/article/9236648/Choosing_an_open_source_CMS_part_1_Why_we_use_Drupal

Porin ylioppilasteatterin www-sivut. 2014. Viitattu 20.1.2014. http://www.porinylioppilasteatteri.fi/

Rathinam, G. 2013. Latest Web Design Trends – 2013. Viitattu 20.1.2014. http://www.webdesigntalks.com/latest-web-design-trends-2013/

Roberts, H. 2014. When using IDs can be a pain in the class.... Viitattu 30.1.2014. http://csswizardry.com/2011/09/when-using-ids-can-be-a-pain-in-the-class/

Rocheleau, J. 2013. Large Background Images in Web Design: Tips and Examples. Viitattu 24.1.2014. http://www.hongkiat.com/blog/oversized-background-image-design/

Salakapakka www-sivut. 2012. HTML- ja CSS-opas. Viitattu 24.1.2014. http://salakapakka.net/oppaat/html-ja-css-opas/html_opas_varit.php

Sinkkonen, I., Nuutila, E. & Törmä, S. 2009. Helppokäyttöisen verkkopalvelun suunnittelu. Hämeenlinna: Tietosanoma

UXMyths www-sivut. 2013. Myth #3: People don't scroll. Viitattu 20.1.2014. http://uxmyths.com/post/654047943/myth-people-dont-scroll

Viikkispeksin www-sivut. 2014. Viitattu 20.1.2014. http://mmylspeksi.com/mika-speksi/

w3 Archives www-sivut. 1991. Re: status. Re: X11 BROWSER for WWW. Viitattu 29.1.2014. http://lists.w3.org/Archives/Public/www-talk/1991SepOct/0003.html

w3 www-sivut. 2013. Viitattu 29.1.2014. http://www.w3.org/MarkUp/

w3schools www-sivut. 2014. Browser Statistics. Viitattu 29.1.2014. http://www.w3schools.com/browsers/browsers_stats.asp

w3techs www-sivut. 2014. Usage of JavaScript libraries for websites. Viitattu 31.1.2014. http://w3techs.com/technologies/overview/javascript_library/all

Wax, D. 2008. 7 Essential Guidelines For Functional Design. Viitattu 20.1.2014. http://www.smashingmagazine.com/2008/08/05/7-essential-guidelines-for-functional-design/

Wroe, N. 2013. The Importance of Cross Browser Testing. Viitattu 29.1.2014. http://www.platform81.com/index.php/news-a-offers/59-the-importance-of-cross-browser-testing

ZileZilla Wiki www-sivut. 2009. Viitattu 29.1.2014. https://wiki.filezilla-project.org/Main_Page