SAVONIA

Opinnäytetyö - Ammattikorkeakoulututkinto
Tekniikan ja liikenteen ala

SMART KOTISIVUTYÖKALUN PÄIVITYS

TEKIJÄ: Jere Borgelin

SAVONIA-AMMATTIKORKEAKOULU OPINNÄYTETYÖ

Tiivistelmä

Koulutusala			
Tekniikan ja liikenteen ala Tietotekniikan koulutusohjelma			
Työn tekijä(t) Jere Borgelin			
Työn nimi Smart Kotisivutyökalun päivitys			
Päiväys	5.5.2016	Sivumäärä/Liitteet	36
Ohjaaja(t) Lehtori Jukka Kinnunen, lehtori Sami Lahti			
Toimeksiantaja/Yhteistyökumppani(t) Hurja Solutions Oy			

Tiivistelmä

Tämän opinnäytetyön aiheena oli Hurja Solutions Oy:n Smart Kotisivutyökalun päivitys. Työhön kuului uudistuksien määrittely, suunnittelu ja implementointi. Työssä tarkasteltiin sivuston käytettävyyttä ja pyrittiin parantamaan sitä tarpeen tullen. Uudistukset olivat pääsääntöisesti ulkoasun päivittämistä responsiiviseksi, käyttäen suosittua Bootstrap-kehitysalustaa. Päivityksen myötä myös sivuston mobiilikäytettävyys parani.

Järjestelmän suuruuden vuoksi uudistuksien kohteet rajattiin Smartin eniten käytettyihin sivuihin ja lisäosiin. Kohdesivuista laadittiin ehdotelmana kuvat, joita tarvittaessa jalostettiin asiakkaan toiveiden mukaan. Ehdotelmien hyväksynnän jälkeen voitiin aloittaa päivitysten implementointi testiympäristöön.

Lopputuloksena saatiin responsiiviset versiot Smartin yleisemmin käytetyistä sivuista. Sivut toteutettiin käyttämällä HTML5- ja CSS3-kieliä yhdessä Bootstrap-alustan kanssa. Muutoksien myötä Smartin ulkoasu muuttui käytettävämmäksi myös pienemmillä näytöillä. Muutoksien ansiosta Smart on nyt uusien web-standardien mukainen ja sen ulkoasu on uuden tuntuinen, mutta säilyttäen kuitenkin asiakkaille tutuksi tulleen ilmeen. Koska opinnäytetyöhön rajatut sivut ja lisäosat ovat vain osa Smartia, tullaan ulkoasun ja muiden ominaisuuksien päivitystä jatkamaan vielä opinnäytetyön jälkeenkin.

Avainsanat

julkaisujärjestelmä, bootstrap, css, html, javascript

Field of Study Technology, Communication and Transport			
Degree Programme Degree Programme in Information Technology			
Author(s) Jere Borgelin			
Title of Thesis Updating Smart Content Management System			
Date	5 May 2016	Pages/Appendices	36
Supervisor(s) Mr. Jukka Kinnunen, Lecturer, Mr. Sami Lahti, Lecturer			
Client Organisation /Partners Hurja Solutions Ltd			

Abstract

In this thesis, the goal was to update Hurja Solutions' content management system called Smart. The job included defining, planning and implementing the updates. Updates were mainly for improving the responsiveness of Smart's layout using the popular Bootstrap 3 framework, which in turn improved the application's performance on mobile devices.

Because of large size of Smart, the project started by defining smaller sections of the system that were to be updated during the thesis. After the definition phase, image prototypes were drawn up from each page, which were improved according to the customer's wishes. After the prototypes had been finished and accepted by the customer, the planned updates were implemented via code to the development environment.

As a result of this thesis, the target pages were updated to use the latest Web standards along with the Bootstrap 3 framework making them more usable on smaller screens. The site's overall look is now cleaner, while simultaneously retaining the look and feel that customers are used to.

Keywords
control management systems, bootstrap, css, html, javascript

SISÄLTÖ

TERMIT JA LYHENTEET

Back-end
Sovellus tai ohjelma, joka palvelee ja tukee front-end-kehitystä. Mukaan luetaan palvelinpuolen kieliä kuten PHP.

Bootstrap
Avoimen lähdekoodin kehitysrajapinta.

CERN
Euroopan hiukkasfysiikan tutkimuskeskus.

CSS
Cascadian Style Sheet. Kieli jolla kerrotaan, miten HTML-dokumentin sisältämät elementit tulee näyttää.

Front-end
Sovelluksen osa, jonka kanssa käyttäjät ovat suorassa vuorovaikutuksessa. Sisältää mm. kielien HTML:n, CSS:n ja JavaScriptin käytön.

HTML
Hyper Text Markup Language. Kieli jolla kuvataan verkkosivut.

Hyper-G
Tietoverkkojärjestelmä, joka oli webin aikaisempia kilpailijoita.

JavaScript
Staattisille verkkosivuille interaktiivisuutta lisäävä kieli.

jQuery
Avoimen lähdekoodin JavaScript-kirjasto.

PHP
PHP: Hypertext Preprocessor (aiemmin Personal Home Page). Palvelinpuolen ohjelmointikieli, jota käytetään muun muassa verkkosivujen logiikan toteuttamiseen ja tietokannan kanssa kommunikoimiseen.

PSD
Photoshopin käyttämä tiedostojen tallennus formaatti.

Tag
HTML merkintätyyli (<p>, , <img>, jne.)

1 JOHDANTO

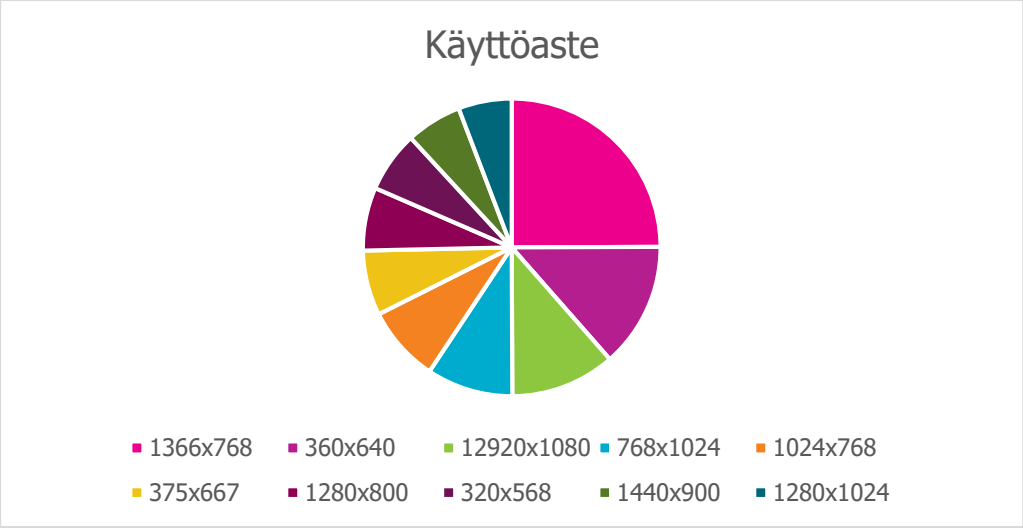
Tämän opinnäytetyön aiheena on Hurja Solutions Oy:n Smart Kotisivutyökalun päivitys. Smart on PHP:lla tehty järjestelmä, jolla on ketterä luoda uusia ja hallinnoida jo olemassa olevia web-sivustoja. Smart on ollut jo pitkään Hurjan palveluvalikoimassa ja sitä kehitetään jatkuvasti, joko asiakkaan tarpeiden mukaan tai muuten nousseen kehitysidean myötä. Kuitenkin muun kehityksen ohella järjestelmän ulkoasu on jäänyt vähäiselle päivittämiselle sitten vuoden 2003. Ennen vuotta 2007 pienempiä näyttöjä ei oikeastaan huomioitu verkkosivuja rakennettaessa. Tästä syystä työssä tehdyt uudistukset ovat pääsääntöisesti visuaalisen puolen kunnostamista käyttäen suosittua Bootstrap-kirjastoa.

Tässä raportissa käydään läpi lyhyesti responsiivisuuden tarkoitus ja sen historia. Sitten esitellään julkaisujärjestelmien tarkoitus sekä työssä käytetyt menetelmät ja työkalut. Työkaluihin kuului HTML5-, CSS3-, JavaScript-kielet sekä Bootstrap- ja jQuery-kirjastot. Lopuksi esitellään työstä kertynyt dokumentaatio.

2 WWW-SIVUJEN RESPONSIIVISUUS

2.1 Tarkoitus

Sivun responsiivisuudella tarkoitetaan sivua, joka mukautuu käytössä olevan laitteen leveyteen muuttamalla sivun sisältämien elementtien kokoa tai joissakin tapauksissa jopa piilottamalla joitakin elementtejä. Tarkoituksena on joka tapauksessa saada sivuston käyttökokemuksesta mahdollisimman hyvä, laitteesta riippumatta. Sivuston tulee olla käytettävissä tietokoneen näytöllä, joka on yleensä 1366x768 pikseliä ja mobiilinäytöllä, joka on pienimmillään 320x568 pikseliä. Responsiivista sivua rakennettaessa mahdollisimman moni resoluutio tulee ottaa huomioon. Kuvassa 1 on luokiteltu yleisimmät resoluutiot niiden käyttöasteiden mukaan.



KUVA 1 Resoluutiot käyttöaseteiden mukaan (W3Counter, 2016).

Responsiivisuus on myös osa mobiilioptimointia, jossa otetaan huomioon mobiililaitteissa käytettävä mahdollinen hitaampi internetyhteys. Jotta sivut latautuvat nopeasti hitaammillakin yhteyksillä, on kuvien ja muiden erikseen ladattavien tiedostojen määrä oltava pieni.

2.2 Historia

2.2.1 Kiinteäleveyssuunnittelumalli

Vielä muutama vuosi sitten sivustoja suunniteltiin erityisesti tietokoneiden näytöille, koska ylivoimaisesti suurin osa käyttäjistä selasi webiä tietokoneilla. Useimmat näytöt olivat 800 tai 1024 pikseliä leveitä ja sivut suunniteltiin sopivaksi näihin näyttöihin. Ennen pitkää muun kokoisia näyttöjä alkoi ilmestyä markkinoille ja web-suunnittelijat halusivat sivujensa näyttävän samalta joka laitteella. Responsiivinen suunnittelu on ratkaisu kiinteiden leveyksien aiheuttamiin ongelmiin. Yllätykseksi kiinteän leveyden suunnittelumallia käytetään vielä nykyisinkin. (McWaters, 2016)

2.2.2 Mobiilimurto

Matkapuhelimilla pystyttiin selailemaan webiä jo 1990-luvulla, mutta ne pystyivät näyttämään vain tekstiä. Vasta 2000-luvun ensikymmenen puolivälissä älypuhelimet saavuttivat suosionsa ja niiden selaimet pystyivät näyttämään oikeita web-sivuja. Älypuhelimet pystyivät näyttämään Web-sivun kuten tietokoneet, mutta sivustoilla navigointi oli vielä vaikeaa. Tehdäkseen sivulla mitään, käyttäjä joutui zoomaamaan sivua ja latausajat olivat todella pitkiä. (McWaters, 2016)

Mobiililaitteiden käyttö internetissä alkoi yleistyä hiljalleen vasta vuoden 2007 jälkeen, kun iPhone tuli markkinoille. Sitä ennen sivustot rakennettiin vain tietokoneiden näytöille. Kun pienemmät ruudut alkoivat yleistyä, sivustoista tehtiin erilliset versiot pienemmille ja isommille näytöille (Peterson, 2014).

Statcounter-sivuston antaman raportin mukaan 2016 maaliskuussa internetin selauksista 40,6 % oli mobiililaitteilla ja 5,2 % tableteilla maailmanlaajuisesti (StatCounter, 2016 A). Tarkastellessa aikaväliä tammikuu 2010 – huhtikuu 2016, mobiililaitteiden sivulautaukset ovat huomattavasti lisääntyneet, kun taas tavallisten tietokoneiden sivulataukset ovat vähentyneet (StatCounter, 2016 B).

3 JULKAISUJÄRJESTELMÄT

3.1 Esittely

Julkaisujärjestelmä on ohjelmistopaketti, joka automatisoi Web-julkaisun joitakin tehtäviä. Tämä tehostaa sivujen ja niiden sisällön hallintaa. Järjestelmät koostuvat useammasta osasta. Hallintapaneelit, arkistointi, jne. ovat järjestelmän eri osia. Kuitenkin kaikki nämä osat on pidetty yhtenä kokonaisuutena, jota kutsutaan lyhenteellä CMS (Content Management System). (Barker, 2016 A)

3.2 Käyttötarkoitus

CMS:llä saadaan tuotettu sisältö kuriin. Se tietää missä sisältö sijaitsee, missä kunnossa se on, kuka voi käyttää sitä ja miten se liittyy muuhun sisältöön (Barker, 2016 B). Sisällöntuottajan näkökulmasta julkaisujärjestelmien yleisimpiin toimintoihin kuuluu vähintään sivujen, artikkeleiden ja muun median hallinta. Niillä on selkeä hallintapaneeli, jolla kaikki toiminnot suoritetaan. Lisäksi julkaisujärjestelmästä voi löytyä valmiiksi rakennettuja lisäosia. Yleisimpiin lisäosiin kuuluu bannerit, liukukuvat, tapahtumakalenterit, pikakyselyt ja lomakkeet. Näiden lisäksi voidaan tarvita tiettyyn tarkoitukseen räätälöityjä lisäosia. Tällöin kehittäjien pitää pystyä lisäämään tai muokkaamaan julkaisujärjestelmien ominaisuuksia.

3.3 Oma CMS

Oman julkaisujärjestelmän luonnissa on sekä hyviä että huonoja puolia. Jos kehittäjällä on tietty idea miten järjestelmän tulisi toimia sen juuritasolla, on oman järjestelmän kehittäminen ehkä järkevä ratkaisu. Kun järjestelmää voidaan muuttaa ja laajentaa mieleiseksi, voi sen päivittäinen käyttökin olla paljon sulavampaa.

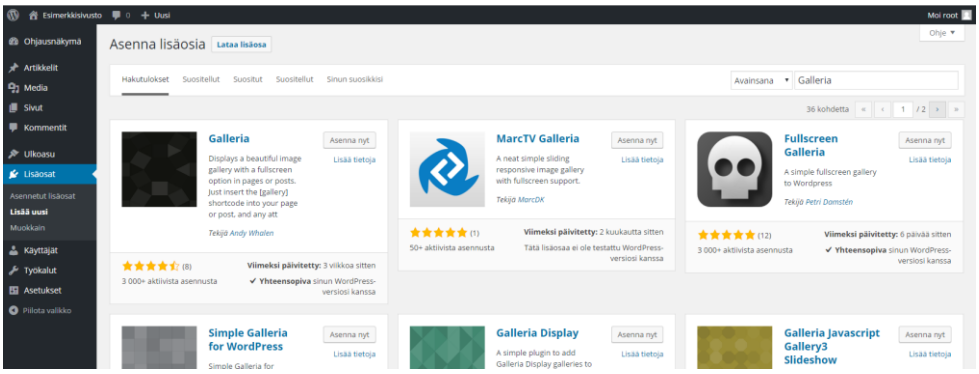
Kehittäjille on tarjolla ilmaisia järjestelmiä, joilla on pitkä historia takanaan. Jo valmiin julkaisujärjestelmän valitseminen voi olla hyvinkin kannattavaa. Ne tarjoavat paljon toimintoja, jotka oman järjestelmään tulisi kehittää itse. Jo valmiilla ja etenkin ilmaisilla järjestelmillä on lukuisia käyttäjiä, joilta saa apua ongelmatilanteissa. Tekniikoiden kehittyessä, myös järjestelmiin tulee päivityksiä. Näin voidaan keskittyä asiakkaisiin ja heidän tarvitsemiin toimintoihin.

Jotkin järjestelmistä sallivat kehittäjien tehdä omia lisäosia, jotka laajentavat järjestelmän toimintaa. Kehitetty lisäosa voidaan jakaa internetin välityksellä muille käyttäjille. Tämä on WordPress-nimisessä julkaisujärjestelmässä yksi sen vahvimmista ominaisuuksista. Tarvitut ominaisuudet voivat olla jo olemassa ja vapaasti käytettävissä, ladattavan lisäosan avulla. Huono puoli tässä on se, ettei kehittäjä voi aina tarkoin tietää miten muiden kehittämä lisäosa toimii. Lisäksi kaikki lisäosat eivät ole yhteensopivia toistensa kanssa.

3.4 WordPress

WordPress on henkilökohtainen, ilmainen julkaisualusta. Sen painopisteinä ovat esteettisyys, Web-standardit ja käytettävyys (WordPress, 2016). WordPress voidaan asentaa omalle koneelle tai sitä voidaan käyttää suoraan selaimen kautta. Järjestelmää voidaan laajentaa valitsemalla halutut laajennukset omasta hallintapaneelista (KUVA 2). Sillä voidaan lisätä sivuston ylläpitoon käyttäjiä, jotka pääsevät muuttamaan sivuston sisältöä. Käyttäjille voidaan asettaa rooleja, joilla rajataan heidän oikeuksiaan sivustoon. WordPressillä on mediakirjasto, jonne ladataan sivuston käyttämät tiedostot (Kuva 3), minkä jälkeen niitä voidaan käyttää ympäri sivustoa. Toki materiaali voi tulla järjestelmän ulkopuoleltakin.

WordPressin avulla saa tehtyä monenlaisia sivuja, eikä sivujen tekoon tarvita välttämättä ohjelmoijaa. Toki jotkin sivustoista vaativat enemmän työtä kuin toiset. Esimerkiksi verkkokaupan luominen ostoskoreineen ja tilausprosesseineen vaatii käyttäjän muuttamaan WordPressin toimintaa turvautumalla johonkin lisäosaan. (MacDonald, 2014)



KUVA 2 WordPressin lisäosahaku. Haussa avainsanalla "Galleria" olevat lisäosat.

KUVA 3 Wordpressin mediakirjasto. Kuvat ovat Negativespace-sivustolta.

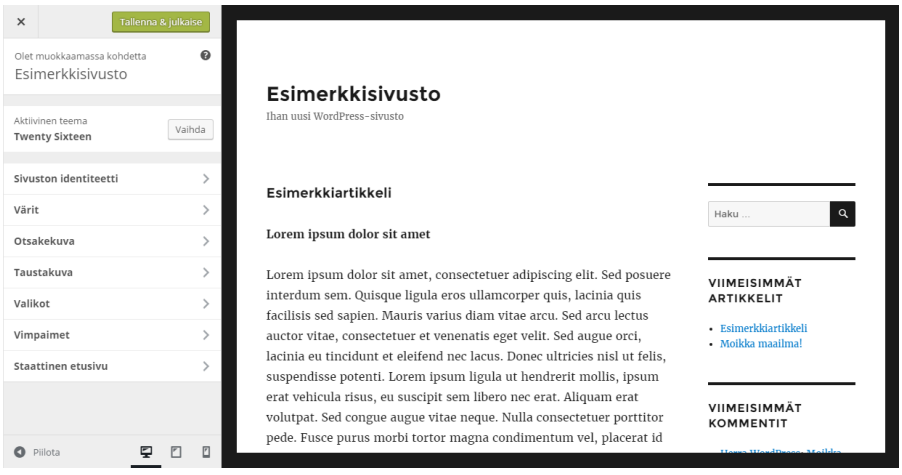
3.5 Vertailu

Smart ja WordPress poikkeavat ominaisuuksiltaan paljon. Suurin ero tulee kuitenkin siitä, että WordPress on maailmanlaajuisesti käytetty. WordPressillä on monta kehittäjää, jotka tekevät eri lisäosia ja ulkoasuja. Tähän kun lisätään järjestelmän laajennettavuus vapaasti valittavien lisäosien avulla, kasvaa WordPressin ominaisuudet huomattavasti.

WordPress ei kuitenkaan sovellu useamman sivuston rakentamiseen, vaan se on asiakaskohtainen. Kun WordPressin päälle tehty sivusto on valmis, on toisen sivuston rakennus aloitettava alusta. Lisäksi jos useammalla sivustoilla käytettyihin lisäosiin tulee päivityksiä, esimerkiksi haavoittuvuuden korjaus, pitää kaikki sivustot päivittää erikseen. Tähän on olemassa työkaluja, jotka antavat kehittäjien hallinnoida useampaa sivustoa samanaikaisesti. Ylläpidettävien sivustojen määrän kasvaessa, voi päivitysten mukana pysyminen olla hankalaa. Smartissa tämä ei ole ongelma. Kaikki sivustot hyödyntävät yhtä pääohjelmaa, jonka päivittyessä (esimerkiksi ulkoasun osalta) muutokset vaikuttavat jokaiseen sivustoon. Samoin useiden käyttäjätunnusten hallinta yhden järjestelmän kautta nopeuttaa työnteossa. WordPressissä perustoimintoimintoihin ja lisäosiin kuten kuvagalleriaan ja lomake-editoriin on ladattava erillinen lisäosa. Smartissa nämä ovat heti käytettävissä.

3.5.1 Asetukset

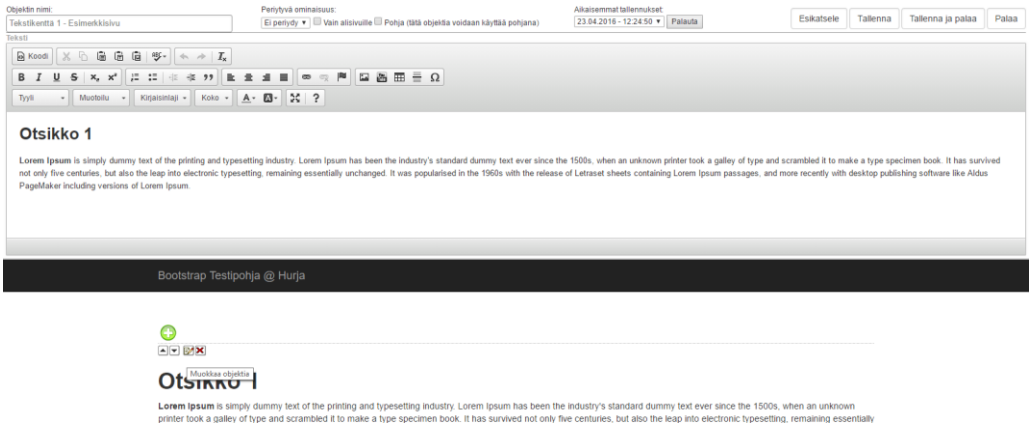
Käyttäjän ja sivuston asetuksissa WordPress on Smartia edellä. WordPress tarjoaa perusasetusten (kuten sivuston nimen tai kuvauksen) lisäksi myös aikavyöhykkeelle, päivämäärän esitysmuodolle ja jopa kalenterin ensimmäiselle viikolle erilliset valinnat. Käyttäjän omista asetuksista voi valita niin oman nimen kuin hallintapaneelin taustavärin. Vaikka jotkin asetukset eivät vaikuta itse sivustoon, tuo ne lisäarvoa järjestelmälle. Käyttäjä pääsee myös muokkaamaan julkisenpuolen ulkoasua (Kuva 4).



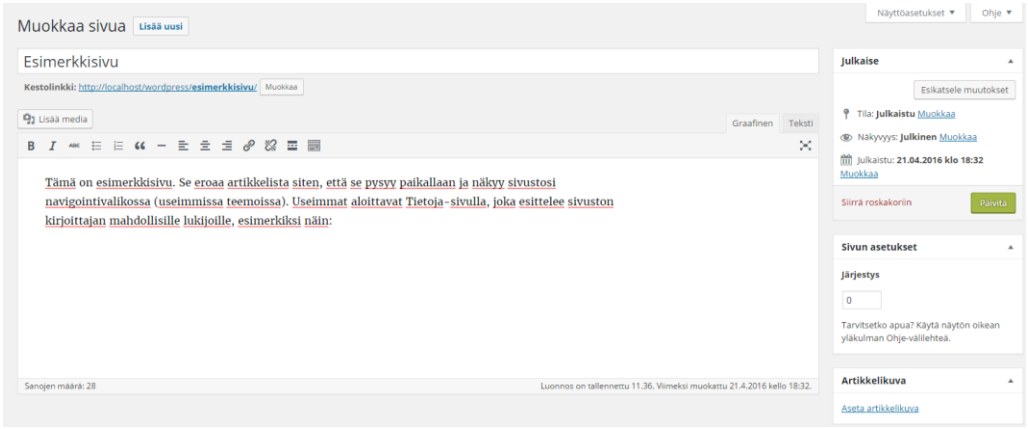
KUVA 4 Sivuston ulkoasun muokkaus

3.5.2 Sisällönsyöttö

Sisällönsyötössä kummallakin järjestelmällä on puolensa. Molemmissa muokattava sivu valitaan sivuhallinnasta ja tämän jälkeen käyttäjälle esitetään tarvittavat kentät muutoksia varten (Kuvat 5 ja 6). Muutosten jälkeen sivu voidaan julkaista tai jättää luonnokseksi.



KUVA 5 Smartin sisällönsyöttönäkymä, jossa yksi tekstieditori alla olevan tekstin muuttamiseksi.



KUVA 6 Wordpressin sisällönsyöttönäkymä yhden sivun muokkaamiseen

Smartin yksi vahvuuksista on sivuihin upotettavien objektien luonnin helppous. Objekti voi koostua yhdestä tai useammasta tekstieditorista tai kuvasta. Jos sivulla on vierekkäin kolme nostoa, on järkevää käyttää objektia, jossa on kolme tekstieditoria. Jos nostoissa on vielä erikseen kuva ja linkki, voidaan nekin lisätä objektiin. Tällöin objektissa on kentät tekstille, kuvalle ja linkille. Objekti ei vaikuta siihen, miten tiedot näytetään. Kehittäjä on sen alun perin määritellyt, joten käyttäjä voi huoletta syöttää tarvittavat tiedot ja tallentaa objektin. Yhdelle sivulle voidaan kasata useampia objekteja, jolloin käyttäjällä on mahdollisuus vaikuttaa koko sivun ulkoasuun ilman ohjelmoijaa. Objektien tekoon ei tarvitse luoda uutta lisäosaa, vaan ne määritetään sivustokohtaisesti kehittäjien toimesta.

WordPressistä löytyy valmiina hallintapaneelit artikkelien ja sivujen (kuva 6) muokkaukseen. Kummassakin on kentät otsikolle ja sisällölle. Käyttäjä voi hakea ja liittää mediaa sivuun tai artikkeliinsa, aikaisemmin mainitusta mediapankista. Julkaisu voidaan ajastaa tai piilottaa. Sivujen järjestyksen muuttamiseksi hallinnassa on erillinen numerokenttä ja artikkelit voidaan kategorisoida.

Kategoriat auttavat sisällön organisoinnissa ja haussa. Ilman erillisiä lisäosia nämä hallintapaneelit eivät paljon muuta tarjoa.

Mikäli jokin lisäosa on asennettu ja se on tarkoitus upottaa sivulle, voi käyttäjä kutsua sitä kehittäjän määrittämällä koodilla. Koodi kirjoitetaan sisällölle tarkoitettuun kenttään. Lisäosalla voi olla myös oma hallintasivu.

3.5.3 Yhteenveto

Kummankin julkaisujärjestelmän heikkoudet ja vahvuudet ovat niiden lisäominaisuuksissa ja siinä, mitä järjestelmät tarjoavat valmiiksi. Sivuston päivittäminen järjestelmän kautta tulee olla vaivatonta. Molemmat järjestelmät antavat käyttäjän muokata sivuja sisällöltään mieleisekseen, eikä toiminnon suorittamiseksi tarvita useampaa osanottajaa. WordPress soveltuu pienien sivustojen, kuten blogien tekoon ja ylläpitämiseen. Heti järjestelmän asennettua sivuston käyttämät värit ja fontit voidaan muuttaa ja yhteystiedot syöttää. Isompien sivustojen kohdalla muutokset eivät yleensä ole näin yksinkertaisia, joten järjestelmään joudutaan kaivautumaan hallintapaneelia syvemmälle.

Smartissa useamman sivuston samanaikainen teko ja ylläpito on helpompaa. Kunkin sivuston hallintapaneeli on sama, mutta sisällönsyöttöön tarvittavia työvälineitä pystytään räätälöimään nopeasti asiakkaiden tarpeiden mukaan. Yleisimmät lisäosat löytyvät Smartista jo valmiina, mutta lisäosia ei ole läheskään yhtä paljon kuin WordPressissä. Asiakkaille kehitettävät lisäosat ovat aina erilaisia, eikä näihin välttämättä löydy juuri oikeaa jo valmista lisäosaa. Tällöin molempien järjestelmiin se tullisi kehittää itse.

4 OHJELMOINTIKIELET

Bootstrapin lisäyksen myötä myös dokumenttien sisältämät tagit ja niihin liitetyt muut tiedostot voivat muuttua. Bootstrapin yhteydessä käytetään mm. HTML-, CSS-, ja JavaScript-kieliä sekä jQuery-nimistä JavaScript-kirjastoa. Nämä kielet ovat kehittyneet, sitten Smartin viime ulkoasupäivityksen. Seuraavaksi esitellään työssä käytetyt kielet ja tehdään hyvin pieni ja yksinkertainen sivu, johon lisätään aina ko. kielen ominaisuuksia.

4.1 HTML

HTML (HyperText Markup Language) on kieli, jolla määritellään dokumentin rakenne. Määrittelyyn käytetään tag-merkintöjä, jotka näkyvät kuvassa 7. Kuvan koodi määrittää sivun, jolla on valikko(`nav`), lista(`ul` ja `li`), ensimmäisen tason otsikko(`h1`) sekä sen perässä olevat sivukommentti(`small`) ja kappale(`p`). Dokumentin merkistö kerrotaan selaimelle `meta`-tagilla. Tagissa on attribuutti `charset`, jonka arvo on UTF-8. Attribuutteja on myös listan sisältämissä `a`-tageissa. Tagit ovat linkkejä ja attribuutti `href` kertoo linkin osoitteen. Selain ei näytä attribuutteja muun sisällön ohessa. HTML on staattista ja se ei sisällä mitään logiikkaa tai tyylejä. Päätökset siitä, mitä käyttäjälle loppujen lopuksi näytetään, tehdään muilla kielillä.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Esimerkkisivu</title>
    </head>
    <body>
        <nav>
            <ul>
                <li><a href="/">Etusivu</a></li>
                <li><a href="/products.html">Tuotteet</a></li>
                <li><a href="/contact.html">Yhteystiedot</a></li>
            </ul>
        </nav>
        <div>
            <h1>Esimerkkisivu</h1>
            <small>Tämän sivun tarkoituksena on demonstroida esitellyt kielet.</small>
            <p>Hei maailma!</p>
        </div>
    </body>
</html>
```

KUVA 7 Esimerkki HTML:stä

KUVA 8 Kuvan 7 tuottama näkymä selaimessa

4.1.1 Historia

HTML:n kehitti Tim Berners-Lee työskennellessään CERNissä. Hän turhautui joutuessaan aina kirjautumaan eri tietokoneille löytääkseen haluamansa tiedon. Tiedon löytämiseen oli oltava parempi tapa. Tapa, jolla voisi hyppiä tietolähteestä toiseen. Tästä hypertekstijärjestelmäkonseptista (yhdistäen tietoverkkotekniikan ja protokollan, jolla voidaan siirtää dataa tietokoneesta toiseen) muodostui pohja World Wide Webin peruskielelle (HTML).

HTML:n ensimmäinen versio (HTML 1.0) eroaa hyvin paljon nykypäivän HTML:stä. Se oli paljon nykyistä rajoitteisempi. Internet ei ollut vielä yleistynyt ja vain muutama ihminen oli mukana web-kehityksessä. HTML:llä ei saatu aikaiseksi muuta kuin tekstiä. HTML 1.0 sisälsi vain 22 tagia. Vuonna 1995 Web alkoi saavuttaa suosiota ja HTML 2.0 julkaistiin. Kielen toinen versio ei paljon eronnut aiemmasta versiostaan.

Kolmas versio HTML:stä antoi kehittäjille enemmän työkaluja, joilla web-sivu luotiin. Sen aikaiset selaimet olivat hitaita implementoimaan näitä ominaisuuksia, joten HTML 3.0 jäi vain luonnokseksi. Vuonna 1994 luotiin organisaatio nimeltä W3C (World Wide Web Consortium), jonka tarkoituksena oli standardoida HTML. Tästä syntyi HTML 3.2, josta tuli virallinen standardi vuonna 1997.

Saman vuoden joulukuussa julkaistiin HTML4, joka oli iso askel eteenpäin kielen kehityksessä. Neljäs versio kielestä standardoitiin huhtikuussa 1998. Versio esitteli tageja tyylimäärittelyille (CSS), skripteille(JavaScript), kehyksille, upotetuille objekteille sekä monimutkaisemmille tauluille ja lomakkeille.

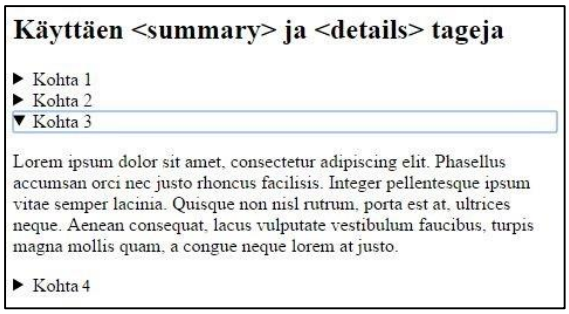
(Land Of Code, 2014)

4.1.2 HTML5

Ennen HTML antoi merkinnät vain sisällöille kuten listat, kappaleet, ylätunnisteet ja taulukot. Nykyään HTML sisältää tageja jo videolle ja audiollekin. Se myös määrittelee monimutkaisempiakin toimintoja, kuten elementtien ”raahaa ja pudota”-toiminnon, palvelin tapahtumat ja jopa asynkroniset toiminnot, kuten web workers (Reid, 2015 A). HTML5 esitteli myös semanttiset elementit, jotka kuvaavat selaimelle elementin sisällön tarkoituksen (w3schools, 2016).

HMTL5 sisältää uusia interaktiivisia elementtejä, jotka on tarkoitettu antamaan valmiiksi rakennettuja käyttöliittymä elementtejä. Niitä voidaan käyttää web-sivuissa ja sovelluksissa. Suurin osa moderneista selaimista ei vielä tue näitä elementtejä. Ehkä kiinnostavimmat näistä elementeistä ovat dialogit (Kuva 9) ja haitarit (Kuva 10). Dialogielementit avautuvat keskelle ruutua ja niiden sisällön voi itse määrittää. Haitarit ovat elementtejä, jotka saadaan klikkaamalla avautumaan ja sulkeutumaan (Reid, 2015 B). Molemmat näistä elementeistä toteutetaan tällä hetkellä jollakin kirjastolla, esimerkiksi Bootstrapilla.

KUVA 9 Esimerkki Dialog-elementistä



KUVA 10 Esimerkki Summary-elementistä

4.2 CSS

CSS (Cascading Style Sheets) on HTML-elementtejä muotoileva kieli. Jos elementit halutaan mitenkään poikkeavan tavallisesta näkymästä, pitää muutokset tehdä CSS:llä. CSS on yksi avoimen webin keskeisistä kielistä ja se on standardoitu W3C-määritys. (Mozilla Developer Network, 2016)

```
1  body  {
2      padding: 15px;
3      font-family: Arial;
4      text-align: center;
5      background-color: #DEDEDE;
6  }
7  #main-navigation {
8      background-color: #000000;
9  }
10 .navbar {
11     padding: 0;
12     list-style: none;
13 }
14 .navbar li {
15     display: inline-block;
16 }
17 .navbar li a {
18     color: #FFFFFF;
19     display: block;
20     font-size: 18px;
21     padding: 5px 10px;
22 }
23 .container {
24     padding: 20px;
25     background-color: #FFFFFF;
26 }
```

KUVA 11 CSS-muotoilut kuvassa 7 oleviin elementteihin

Kuvassa 11 muotoillaan HTML-osiossa luotua sivua. CSS:llä määritetyt säännöt täytyy aina kohdistaa johonkin elementtiin. Tämän voi tehdä monellakin tapaa, mutta yksi yleisimmistä tavoista on suora kohdistus tietyntyyppisen elementtiin (kuvan ensimmäinen rivi). Rivillä 7 on kohdistus `id`-attribuutin ja rivillä 10 `class`-attribuutin kanssa. Kohdistus voidaan aloittaa hierarkkisen HTML-dokumentin

ylemmästä elementistä ja jatkaa siitä itse kohteeseen (rivi 14 ja 17). Kohdistuksen jälkeen kerrotaan säännöt. Sääntö koostuu elementin ominaisuudesta ja sen arvosta. Esimerkiksi rivillä 18 laitetaan linkin tekstin väri valkoiseksi. Määritysten ei tarvitse olla missään järjestyksessä. Jos sääntö kerrotaan useammin, on kaikista alin määritys kuitenkin se mitä noudatetaan. Lopuksi CSS-tiedosto täytyy linkittää HTML-tiedostoon, jotta selain huomio CSS:n sivua esittäessä. Lisäksi HTML-tiedostoon tulee lisätä puuttuvat `id`- ja `class`-attribuutit, jotka luotiin CSS:n puolella.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Esimerkkisivu</title>
        <link rel="stylesheet" type="text/css" href="styles.css">
    </head>
    <body>
        <nav id="main-navigation">
            <ul class="navbar">
                <li><a href="/">Etusivu</a></li>
                <li><a href="/products.html">Tuotteet</a></li>
                <li><a href="/contact.html">Yhteystiedot</a></li>
            </ul>
        </nav>
        <div class="container">
            <h1>Esimerkkisivu</h1>
            <small>Tämän sivun tarkoituksena on demonstroida esitellyt kielet.</small>
            <p>Hei maailma!</p>
        </div>
    </body>
</html>
```

KUVA 12 Muutettu HTML-tiedosto

KUVA 13 CSS:n lisäyksen jälkeiset muutokset, kuvan 8 sivuun

4.2.1 Historia

CSS:stä on nyt kolme versiota. Sen saga alkoi vuonna 1994, jolloin webiä aloitettiin käyttämään sähköisen sisällön julkaisualustana. Yksi alustan tärkeimmistä osista puuttui, sillä dokumentteja ei voitu muotoilla mitenkään. Työskennellessään MIT:n media laboratoriossa työstäen yksilöityjä sanomalehtimalleja Lie Håkon Wium näki tarpeelliseksi sivujen muotoiluun tarkoitetun kielen.

Selaimen tyylitiedostot eivät olleet täysin uusi idea. Dokumentin rakenteen ja sen muotoilun erottaminen toisistaan oli HTML:n tavoitteena alusta alkaen. Tim Berners-Lee kirjoitti hänen NeXT selain/editorin siten, että hän voisi määrittää tyylit yksinkertaisella tyylitiedostolla. Tyylitiedostojen syntaksia ei vielä julkaistu, jättäen kunkin selaimen vastuulle sen, miten sivu näytetään käyttäjille. Pei Wein Viola- ja Harmony-selain Hyper-G-järjestelmälle omasivat jo vertailukelpoisia muotoilukieliä. Kehittyneimpien tyylitiedostojen sijaan, tulevat selaimet tarjosivat käyttäjilleen entistä

vähemmän vaihtoehtoja tyylien määrittämiseen. Vuonna 1993 NCSA (National Center for Supercomputing Applications) kehitti selaimen nimeltä Mosaic, joka yleisti webin käytön. Sivujen tyylittelyjen kannalta se oli kumminkin askel taaksepäin, sillä Mosaic antoi käyttäjien ainoastaan vaihtaa joitakin sivun värejä ja fontteja. (Lie & Bos, 1999)

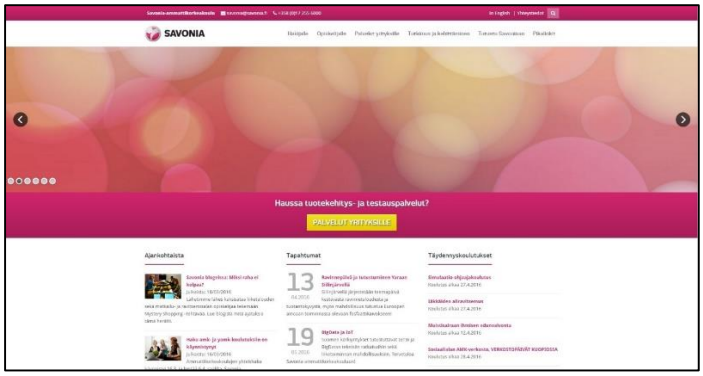
Siirtyen joulukuuhun 1996, CSS1 julkaistiin W3C:n suosituksena. Sen tärkeimpinä ominaisuuksina ja etuina olivat sivujen värien ja fonttien muutokset sekä käyttäjän että julkaisijan toimesta. Halutessaan käyttäjä pystyi muuttamaan julkaisijan tarjoaman muotoilun haluamakseen. Syy, miksi käyttäjä saattoi muuttaa julkaisijan tarjoamaa tyyliä voi olla esimerkiksi käyttäjän heikentynyt näkö.

CSS erotti tyylimäärittelyt erilliseen tiedostoon, joten sivujen ylläpidettävyys nousi. Sivujen koko pieneni, sillä jopa fonttien muotoilu oli aikaisemmin toteutettu kuvin. Nyt tekstin pystyi kirjottamaan dokumenttiin ja käytetty fontti voitiin muotoilla CSS:llä. (World Wide Web Consortium, 2016)

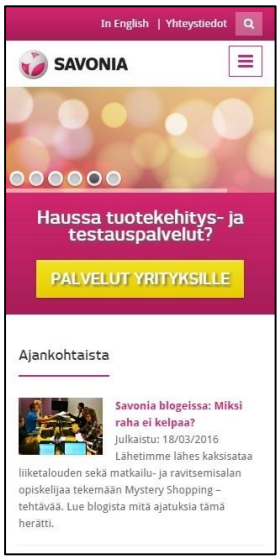
CSS2:n kehitti W3C ja se julkaistiin suosituksena toukokuussa 1998. Se sisälsi uusia ominaisuuksia, kuten elementtien absoluuttinen, relatiivinen ja kiinteä sijoitus. Lisäksi mukana oli käsite mediatyypeistä ja uusia fonttiominaisuuksia, kuten varjostus. (Hissom-Daugherty, 2016)

4.2.2 CSS3

CSS3 toi uusia tyylisääntöjä elementeille esimerkiksi varjostukset, pyöreät kulmat, läpinäkyvyyden ja RGBA-väriarvot. Ennen jotkin näistä jouduttiin tehdä kuvin. Tyylisääntöjen lisäksi CSS3 esitteli animaatiot, jotka tätä ennen tehtiin JavaScriptillä tai Flashillä. Myös pseudoluokat olivat CSS3:n esittelemiä. Pseudoluokalla määritellään elementin erikoistila. Yksi erikoistiloista tulee voimaan, kun hiiri viedään elementin päälle. Viimeisempänä ja ehkä tärkeimpänä ominaisuutena mobiililaitteiden kannalta on media queryt. Media queryillä voidaan kohdistaa säännöt tiettyyn näyttökokoon tai mediaan. Tämä helpottaa kehittäjiä suunnattomasti, kun sivua rakennetaan useammalle näyttökoolle. Ennen media queryjä, sivustoista saatettiin tehdä erilliset versiot pienempiä ja suurempia näyttökokoja varten. Tämä tarkoitti sitä, että aina jos sivustoa haluttiin päivittää, jouduttiin päivitykset tekemään useampaan kertaan. Kuvissa 14 ja 15 nähdään sama sivu suuremmalla ja pienemmällä näytöllä. Kuvan 15 muutokset ovat saatu aikaan media queryillä. Mobiiliversiossa osa elementeistä on piilotettu, fontteja on pienennetty ja navigaatio on klikattavan valikkolinkin takana.



KUVA 14 Savonian etusivu leveydellä 1980px



KUVA 15 Savonian etusivu leveydellä 320px

4.3 JavaScript

JavaScript on ohjelmointikieli, jolla staattisia sivuja saadaan elävöitettyä lisäämällä sivuun interaktiivisuutta. Sivut saadaan reagoimaan käyttäjän toimintoihin, ilman erillisiä sivulatauksia. Tämä voidaan toteuttaa lisäämällä, poistamalla tai muuten muuttamalla jo ladatulla sivulla olevia elementtejä. JavaScripit:n syntaksi muistuttaa jo hieman enemmän ohjelmointikieliä, kuten Java ja C#. Kieli ei ole kuitenkaan niin tyypitetty kuin edellä mainitut.

```
var id = 'main-container';
document.addEventListener("click", function() {
    if(!document.getElementById('how-are-you')) {
        document.getElementById(id).innerHTML += "<p id='how-are-you'>Mitä kuuluu?</p>";
    }
});
```

KUVA 16 JavaScript esimerkki

Kuvassa 16 oleva koodinpätkä ei tee mitään järkevää. Kun käyttäjä klikkaa ruutua, lisätään teksti ”Mitä kuuluu?” elementtiin, jolla on `id`-attribuutti arvolla `main-container`. Tekstin ympärillä on `p`-tag, jolla on `id` arvolla `how-are-you`. Ennen lisäystä tarkastetaan, ettei tällaista elementtiä ole.

KUVA 17 Kuvan 16 koodin tulos, kun ruutua on klikattu

JavaScript voidaan CSS:n tavoin sijoittaa omaan tiedostoonsa, joka täytyy sittemmin upottaa sivulle. Upotus tapahtuu yleensä sivun alalaitaan, juuri ennen `body`-tag:n sulkeutumista (Kuvan 18 rivi 21). Tämä nopeuttaa sivun latautumista, sillä skriptien kokoaminen voi hidastaa sivun esittämistä.

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <meta charset="utf-8">
5          <title>Esimerkkisivu</title>
6          <link rel="stylesheet" type="text/css" href="styles.css">
7      </head>
8      <body>
9          <nav id="main-navigation">
10             <ul class="navbar">
11                 <li><a href="/">Etusivu</a></li>
12                 <li><a href="/products.html">Tuotteet</a></li>
13                 <li><a href="/contact.html">Yhteystiedot</a></li>
14             </ul>
15         </nav>
16         <div id="main-container" class="container">
17             <h1>Esimerkkisivu</h1>
18             <small>Tämän sivun tarkoituksena on demonstroida esitellyt kielet.</small>
19             <p>Hei maailma!</p>
20         </div>
21         <script src="layout.js"></script>
22     </body>
23 </html>
```

KUVA 18 Muutettu HTML-tiedosto

4.3.1 Historia

Vuonna 1995, kymmenessä päivässä Netscape:ssa (nykyään Mozilla:lla) työskennellyt Brendan Eichn keksimä JavaScript on melkein yhtä vanha kuin web itse. Kieli on tunnettu myös nimillä Mocha ja LiveScript. Vasta saman vuoden jouluna nimi muutettiin JavaScriptiksi.

Vuosien 1996 ja 1997 välisenä aikana JavaScript vietiin ECMAn (European Computer Manufacturers Association) käsittelyyn, jossa siitä rajattiin standardit, joita selaintoimittajat pystyivät noudattamaan. Työn tuloksena oli standardi nimeltä ECMAScript. JavaScript noudatti tätä standardia suosituimpana kielenä, mutta mukana oli myös kieli nimeltä ActionScript 3. (W3C, 2012)

Vaikka nykyään JavaScript on suosittu, sillä on jokseenkin karkea menneisyys. JavaScriptiä pidettiin ennen harrastelijoiden ohjelmointikielenä, jolla lisättiin sivuille efektejä, kuten selaimen alareunaa pitkin liukuvia viestejä tai hiirtä seuraavia, animoituja perhosia. Viimeisten vuosien aikana JavaScript on kokenut uudelleen syntymisen, isompien sivustojen kuten Googlen, Yahoo !:n ja Flickr:n käyttäessä sitä laajasti luodakseen vuorovaikutteisia web-sovelluksia. (McFarland, 2014 A)

4.3.2 Tilanne nyt

Nykyään JavaScriptiä ei käytetä pelkästään web-sivuissa. Sillä voidaan luoda Yahoo ! lisäosia, Google sovelluksia ja iPhone ohjelmia. Adobe ohjelmat kuten Acrobat, Photoshop, Illustrator ja Dreamweaver sisältävät JavaScriptillä ohjelmoitavia osia. Jopa MAC OS X:n Yosemite versiossa, Apple antaa käyttäjien automatisoida laitteensa käyttämällä JavaScriptiä. Kieltä käytetään myös monissa kehitystyökaluissa kuten Gulp.js ja Bower. JavaScriptistä on myös tulossa suosittu palvelinpuolen kehityksessä. Yksi näistä on Node.js, jota suosivat yritykset kuten Walmart, PayPal ja eBay. (McFarland, 2014 A)

4.4 jQuery

jQuery on JavaScriptin päälle rakennettu kirjasto, joka helpottaa JavaScript-ohjelmointia. Aiemmin mainitut kielet ovat heti käytettäviä. Selaimet ymmärtävät niitä ilman lisätoimenpiteitä. jQuery pitää erikseen ladata ja liittää sivulle, jotta selain ymmärtää sen syntaksia. Kirjasto on mahdollista ladata omalle tietokoneelle tai sen voi suoraan liittää sivulle käyttämällä palveluita kuten MaxCDN.

jQuery on web-suunnittelijan salainen ase, taistelussa JavaScript ohjelmointia vastaan. jQueryllä voidaan suorittaa yhdellä rivillä koodia toiminto, joka vaatisi pelkältä JavaScriptiltä satoja rivejä koodia ja tunteja testailua eri selaimilla. (McFarland, 2014 B)

Kuvissa 19 ja 20 asetetaan elementille `class`-attribuutti ensiksi JavaScriptillä ja vastaava jQueryllä. Molemmissa tapauksissa tulos on kuvan 21 mukainen. Ero ei ole suuri, mutta koodimäärän kasvaessa näinkin pieni asia tekee koodin tuottamisesta mielekkäämpää ja nopeampaa. Kuvassa 20 nähdään, kuinka jQuery hyödyntää jo olemassa olevia valitsimia. Kaikkia valitsimia, jotka löytyvät jo CSS:n puolelta voidaan käyttää myös jQueryssä.

```
var testDiv = document.getElementById("test");
testDiv.setAttribute("class", "error");
testDiv.innerHtml = " Tapahtui virhe.";
```

KUVA 19 `class`-attributtin asetus JavaScriptillä

```
$("#test").addClass("error").html("Tapahtui virhe."));
```

KUVA 20 `class`-attribuutin asetus jQueryllä.

```
<div id="test" class="error">Tapahtui virhe.</div>
```

KUVA 21 Lohkoelementti, jota on muutettu käyttämällä kuvien 18 tai 19 koodeja

4.5 Historia

John Resig kirjoitti web-ohjelmia ja kuten muutkin kehittäjät hän turhautui pitkän JavaScriptin kirjoittamiseen. Lisäksi koodi joka saattoi toimia toisessa selaimessa, ei välttämättä toiminut toisessa. Ongelman ratkaisuksi piti kirjoittaa ylimääräistä koodia. Koodilla toteutettu toiminto oli kuitenkin sama, eikä siitä maksettu ylimääräistä.

Jatkaessaan web-kehitystä, Resign yhdisteli tarvitsemiaan koodikirjastoja. Tämä yhdistely tuotti loppujen lopuksi uuden kokoonpanon nimeltä jQuery. Resig työsti kokoonpanoaan, jonka hän julkaisi kehitysalustana 26.8.2006. Se mullisti front-end-kehityksen. Tämän jälkeenkin Resig on jatkanut alustansa kehittämistä. (Chaudhary & Kumar, 2015 A)

4.5.1 Laajennettavuus

jQueryn yksi hyvistä puolista on sen laajennettavuus. Siihen on olemassa useita eri laajennuksia, jotka ovat useimmiten ilmaisia ja todella helppokäyttöisiä. Esimerkkinä voidaan mainita jQuery UI, joka antaa kehittäjälle valmiit työkalut monen eri toiminnon lisäämiseksi sivustollensa. Lisäksi kuka tahansa web-suunnittelija voi kantaa kortensa kekoon ja luoda oman laajennuksensa sekä halutessaan jakaa sen muiden käyttöön. Nämä laajennukset saattavat säästää lukuisia työtunteja. jQuery UI antaa käyttäjällensä elementtejä kuten välilehdet, pudotusvalikot, popup-ikkunat ja kalenterit, jotka ovat todella helppokäyttöisiä. Yleensä pelkkä komponentin aktivointi riittää, eikä ylimääräisiä koodirivejä tarvita.

4.6 Bootstrap

Nykyään www-sivujen rakentaminen tyhjästä voi viedä paljon aikaa ja vaivaa. Bootstrap antaa kehittäjille todella laajan joukon käytettäviä HTML-elementtejä. Bootstrap tarjoaa myös valmiin ruudukkomallin, joka auttaa suunnattomasti responsiivisuuden saavuttamisessa. Bootstrap, kuten aiemmin mainittu jQuery, vähentää monia tunteja, jopa päiviä sivuston kehitysajasta. (Rahman, 2014)

Bootstrap on Twitter kehittäjien Mark Ottonin ja Jacob Thorntonin vuonna 2011 tekemä kehitysalusta. Heidän päätavoitteenaan oli tuoda koodiinsa yhteneväisyyttä ja ylläpidettävyyttä. Tuolloin Bootstrap sisälsi vain HTML- ja CSS-komponentit. Bootstrap sisältää myös JavaScript-komponentteja, jotka tulivat vasta versiossa 1.3.0. Vuonna 2013 päästiin nykyiseen Bootstrapin versioon, jossa kaikki Bootstrapin komponentit ovat responsiivisia. (Rahman, 2014)

Twitterin alkuvaiheilla sen kehittäjät käyttivät melkein mitä vain kirjastoa, jotka olivat heille tuttuja ja auttoivat heitä saavuttamaan front-end-vaatimukset. Epäsäännöllisyydet yksittäisten sovellusten välillä hankaloitti niiden laajennusta ja ylläpitämistä. Bootstrapin ensimmäinen tehtävä oli vastata näihin ongelmiin. Runsaan palautteen takia Bootstrapin sisältö on kasvanut huomattavasti. (Twitter, 2011)

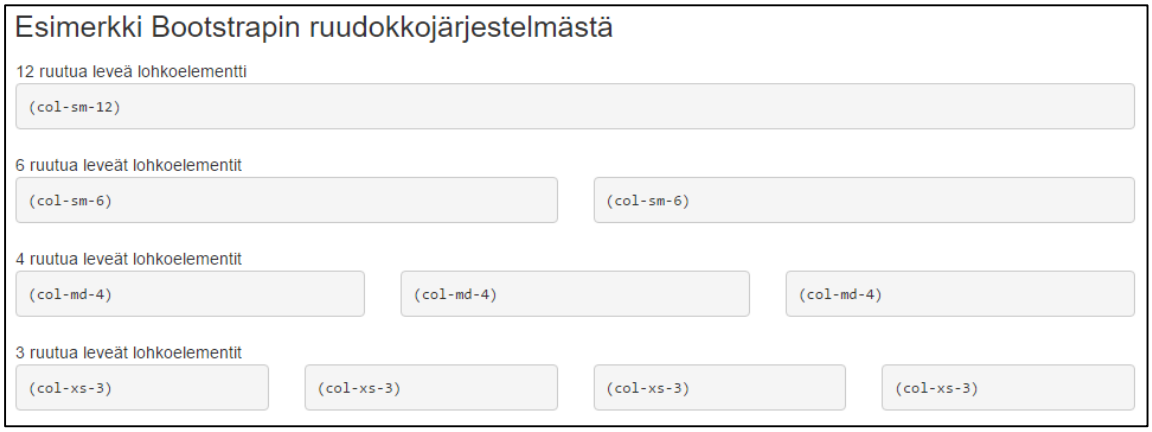
4.6.1 Bootstrap nykyisin

Bootstrap on suosituin HTML-, CSS- ja JavaScript-kehitysalusta responsiivisille sivuille. Vuoden 2013 jälkeen siitä on tullut yksi suosituimmista projekteista myös GitHub-koodinjakamissivustolla. Sillä on hyvä yhteisöllinen tuki ja kattava valikoima valmiita kokonaisia sivupohjia ja laajennuksia. Nämä ovat täysin ilmaisia, kuten kehitysalusta itse.

4.6.2 Ruudukkojärjestelmä

Bootstrap tarjoaa kehittäjälle 12 sarakkeisen ruudukkojärjestelmän, joka jakaa näytön pienempiin osiin ja mukautuu näyttökokojen vaihdellessa. Ruudukko sopii hyvin eri sivujen rakentamiseen.

Esimerkki ruudukosta nähdään kuvassa 22, jossa ruudukon jokainen rivi sisältää lohkoelementin, joilla on erilaisia luokka-attribuutteja.

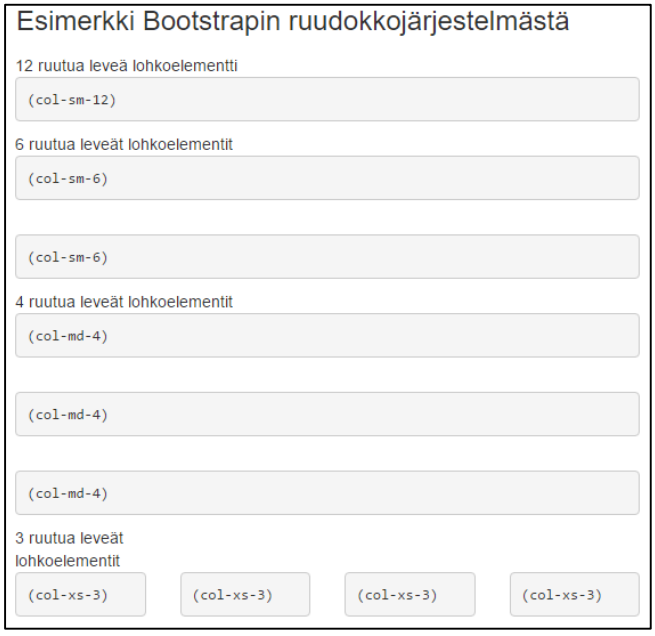


KUVA 22 Esimerkki Bootstrapin ruudukkojärjestelmästä

Yksi ruutu määritetään aina aloittamalla luokan nimi `col-`, jota seuraa kirjaimet `xs-`, `sm-`, `md-` tai `lg-` ja lopuksi numero väliltä 1-12. Merkkijonossa keskimmäiset kirjaimet kertovat, millä ruutukoolla elementti kasvattaa kokoaan. Taulukossa 1 ja kuvassa 23 nähdään, miten ruutukoon vaihtuessa kukin ruutu mukautuu ympäristöönsä.

TAULUKKO 1 Näyttöjen leveyksiin mukautuvat luokat

Näyttöjen leveyksiin vastaavat luokat				
Näytön leveys	< 786px	>= 768px	>= 992px	>= 1200px
Merkkijonossa	`.col-xs-`	`.col-sm-`	`.col-md-`	`.col-lg-`

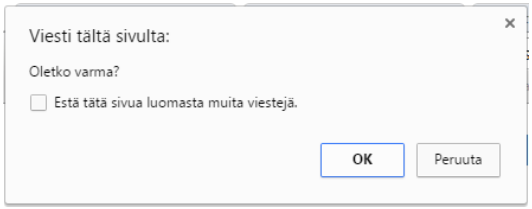


KUVA 23 Ruudukkojärjestelmä pienemmällä näytöllä

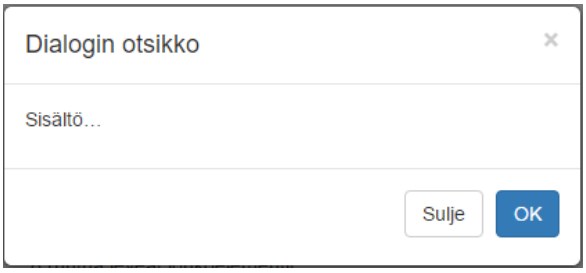
4.6.3 Dialogit

HTML:n esittelyssä mainittu `dialog`-elementti ei ole vielä laajalti tuettu. Sen takia se tehdään esimerkiksi Bootstrapilla. Dialogit ovat hyvä tapa säästyä siirtymiltä näkymien väleillä, huomauttaa käyttäjää tai kuitata hänen toimintonsa. JavaScriptistä löytyy tähän myös `alert`- ja `confirm`-toiminnot (Kuvat 24 ja 25). Näissä dialogeissa saa muotoiltua vain tekstiä. Bootstrapin dialogi (Kuva 26) antaa kehittäjän määrittää ylä- ja alatunnisteen sekä itse sisällön HTML:llä. Dialogi voidaan myös muotoilla CSS:n avulla.

KUVA 24 JavaScript alert



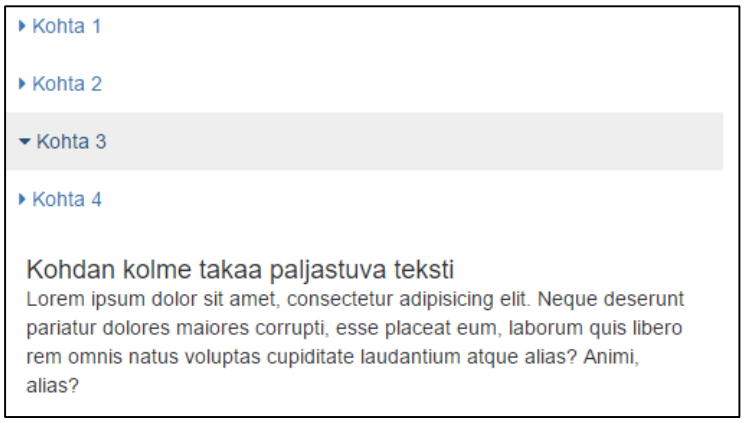
KUVA 25 JavaScript confirm



KUVA 26 Bootstrapin dialogi

4.6.4 Romahtavat elementit

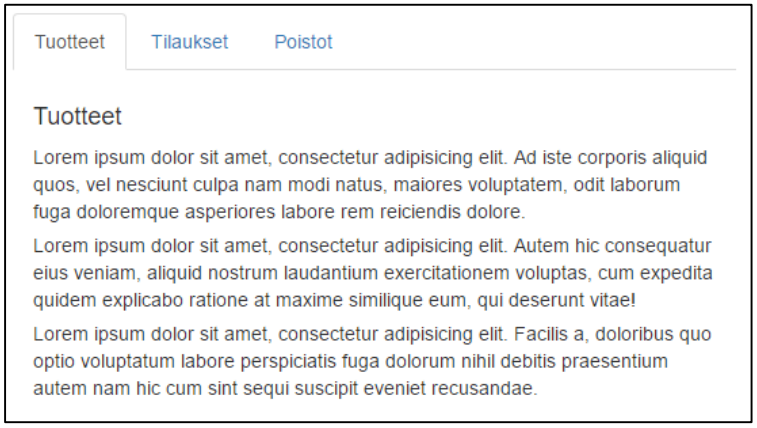
Kuvassa 27 näkyvät linkit paljastavat ja piilottavat niihin yhdistetyt paneelit. Käyttäjää ei tarvitse kuormittaa tekstillä, joka ei häntä välttämättä kiinnosta. Bootstrapin `collapse`-apuluokka antaa suoraan tämän toiminnon, ilman erillistä konfiguraatiota JavaScriptillä. Toiminnon toteuttaminen onnistuisi myös HTML:n `summary`-elementillä, mutta suurin osa selaimista ei vielä tue sitä. Linkit ja niiden paljastamat sisällöt ovat täysin muokattavissa. Bootstrapilla on myös `accordion`-apuluokka, jolla linkitetään nämä kaikki yhteen. Se toimii kuten `collapse`, mutta aina vain yksi kohdista voi olla näkyvissä.



KUVA 27 Bootstrapin `collapse`-tekniikka

4.6.5 Välilehdet

Bootstrap tarjoaa valmiin välilehtijärjestelmän, jolla on helppo jakaa sivuston sisältöä. Lehdille voidaan sijoittaa sitä, mitä kehittäjä saa HTML:n ja muiden kielien avulla aikaiseksi. Välilehtiin kuuluu yleensä kuvan 28 mukainen navigaatio. Lehdet voidaan myös aktivoida suoraan koodista, jonkin muun käyttäjän suorittaman toiminnon seurauksena.



KUVA 28 Bootstrapin välilehtijärjestelmä

4.6.6 Navigointipalkki

Bootstrapin valmiit navigointipalkit vastaavat hyvin responsiivisen sivun kehitykseen. Heti kehittäjän lisätessä navigaation sivullensa, pystyy se mukautumaan mobiilinäytölle (Kuva 30). Yleisimmin navigaatiopalkit koostuvat logosta ja pääsivujen linkeistä (Kuva 29). Palkissa voi myös olla hakukenttä tai muita linkkejä esimerkiksi kielen valintaan tai järjestelmästä uloskirjautumiseen. Pienimmällä näytöillä linkit ja hakukenttä piiloutuvat menukuvakkeen taakse, jota painamalla paljastetaan sama navigaatio horisontaalisena.

KUVA 29 Navigointipalkki leveällä näytöllä

KUVA 30 Navigointipalkki kapeammalla näytöllä

4.6.7 Käyttö

Bootstrap tulee kolmen eri kokoonpanon kanssa, joita voidaan käyttää omien tarpeiden mukaan. Bootstrap, kuten jQuery täytyy upottaa sivustolle, jotta selain ymmärtää sitä. Kirjaston voi suoraan lisätä sivulle MaxCDN-nimisen palvelun kautta tai sen voi ladata omalle koneelle. Bootstrapin lataus onnistuu sen omilta sivuilta, jossa voidaan myös rakentaa omanlainen kokoonpano. Kaikkia komponentteja ei ole pakko ottaa lataukseen mukaan, jos niitä ei tarvitse. Lisäksi komponenttien käyttäytymistä voidaan muokata. Esimerkiksi navigaation pudotusta menukuvakkeen taakse (Kuva 30) voidaan aikaistaa. Tämä tapahtuu oletuksena näytön leveydellä 768px, mutta halutessaan sen voi nostaa leveyteen 992px. Muokkaukset voivat myös olla jotain pientä, kuten päävärien ja fonttikokojen vaihtoa. Bootstrapin määrittämät säännöt voidaan ylikirjoittaa omilla tyyleillä, joten kehittäjät eivät suinkaan ole yhden käyttötuntuman varassa, kun he käyttävät Bootstrappia. Kirjaston hakeminen sivulle MaxCDN:n kautta voi nopeuttaa sivun latautumista, sillä kun kirjasto on kerran ladattu(omien tai muiden sivustojen yhteydessä), se tallentuu välimuistiin, jolloin sitä ei tarvitse ladata enää uudestaan. Toisaalta jos kirjasto linkitetään sivustolle palvelun kautta, eikä palveluun saada yhteyttä, on sivuston käyttöliittymä pilalla.

5 TYÖKALUT

Jotta nettisivut saadaan pelkästä ajatuksesta selaimiin, tarvitsee siihen suunnittelua, asiakaspuolen kehitystä (front-end) palvelinpuolen kehitystä (back-end) ja paljon testausta. Suunnittelu voi lähteä asiakkaan tarpeiden kartoittamisesta ja jatkaa siitä graafikon käsiin, jossa hän rakentaa sivustolle ulkoasun. Tämän jälkeen kuvat ulkoasusta saapuu web-kehittäjälle. Kehittäjä koodaa tekstieditorin avulla sivuston, noudattaen toimitettuja kuvia. Tämä on hyvin karkeasti, hyvin pienen sivuston aikaisemmat kehitysvaiheet. Joka tapauksessa, prosessissa (tässäkin työssä) käytetään aina joitakin mainitsemisen arvoisia työkaluja.

5.1 Kuvanmuokkausohjelma

Sivun ulkoasusta on aina nopeampaa tehdä kuvallinen esitys, ennen kuin aloitetaan sivupohjien kääntö koodille. On nopeampaa muuttaa kuvia kuin koodia. Jos työ on asiakkaan tilaama, vasta kuvan esittämän ulkoasun hyväksyttyään sivupohjat kannattaa siirtää koodattavaksi. Tämä tosin riippuu sivun tarkoituksesta. Jos sivulla on kaksi nappia ja vähän tekstiä, siitä tuskin kannattaa tehdä erillistä kuvaa.

Photoshop on eniten käytetty ohjelma, kun puhutaan sivustojen ulkoasun suunnittelusta ja toteutuksesta. Ohjelmalla on nopea tehdä erilaisia muotoja ja asettaa niihin esimerkiksi varjostus tai reunat. Useat eri työtasot antavat kehittäjän purkaa ja kasata rakentamansa ulkoasun, joka helpottaa useamman sivun tai version tekoa. Tämä ei läheskään kata kaikkia ohjelman ominaisuuksia, sillä ohjelmaa käytetään muuhunkin kuin ulkoasujen rakentamiseen. Photoshop on maksullinen, eikä siitä ole versiota Linux käyttöjärjestelmille.

Yksi vaihtoehto Photoshopille on GIMP, joka on ilmainen ja se toimii Windowsilla, Linuxilla sekä Mac OS X:llä. Kirjoittajan kokemuksen mukaan GIMP ei ole läheskään yhtä monipuolinen ja vaivaton käyttää kuin Photoshop. Lisäksi se vääristää kuvia, jotka ovat toimitettu PSD-formaatissa.

5.2 Tekstieditori

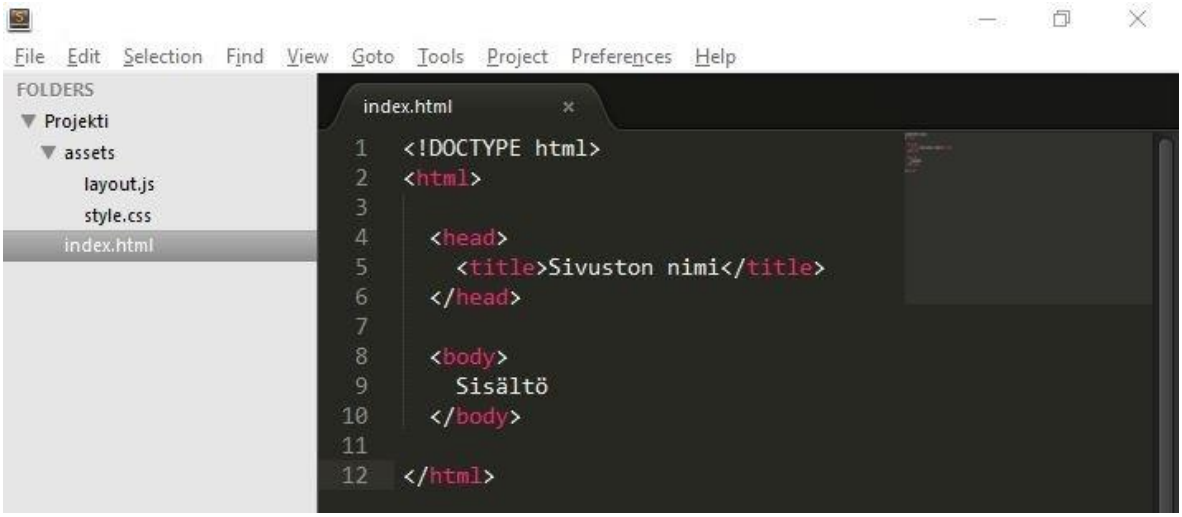
Koodaajan käyttämäksi editoriksi kelpaa jopa Windowsin muistio, mutta internet tarjoaa kattavan valikoiman tekstieditoreita, jotka antavat käyttäjänsä tehdä muutakin kuin kirjoittaa tekstiä. Sivustot yleensä koostuvat useammasta tiedostoista, joita täytyy päästä muokkaamaan. Tällöin on helpompaa, jos tiedostot ja mielellään koko projekti on näkyvillä jossakin. Syntaksin väritys ja avainsanojen automaattinen täyttö ovat myös hyvän tekstieditorin ominaisuuksia. Useimmat ohjelmointiin tarkoitetut tekstieditorit tekevät nämä asiat ja paljon muuta.

Editorin valinnassa ei oikeastaan ole yhtä oikeaa vaihtoehtoa, vaan se määräytyy omien tarpeiden, käytetyn ohjelmointikielen ja oman tottumuksen mukaan. Lisäksi jos työpaikalla suositaan yhtä editoria ylitse muiden, on luultavampaa että käyttöön otetaan juuri tämä editori.

5.2.1 Sublime Text 2

Tämänkin työn aikana käytetty Sublime on ilmainen Windowsilla, Linuxilla ja OS X:llä toimiva tekstieditori. Se on kevyt, laajennettava ja muutoinkin muokattavissa oleva tekstieditori. Jos käyttäjä huomaa kirjoittavansa saman koodipätkän useampaan otteeseen, se voidaan tallentaa omaksi koodipätkäkseen, joka taas saadaan tuotua editorille, itse määrittelemällä komennolla.

Kuvassa 31 näkyy aiemmin mainittu projektinäkymä (harmaalla taustalla) ja syntaksin väritys (mustalla pohjalla). Näiden lisäksi kuvassa oleva teksti syntyi kirjoittamalla editoriin doctype ja painamalla sarkainnäppäintä. Sublimelle voidaan hakea määritystä, tiedostoja ja jopa yhtä sanaa koko projektista. Editorin toiminnot eivät suinkaan lopu tähän. Tarkemmat kuvaukset löytyvät editorin sivuilta (www.sublimetext.com).

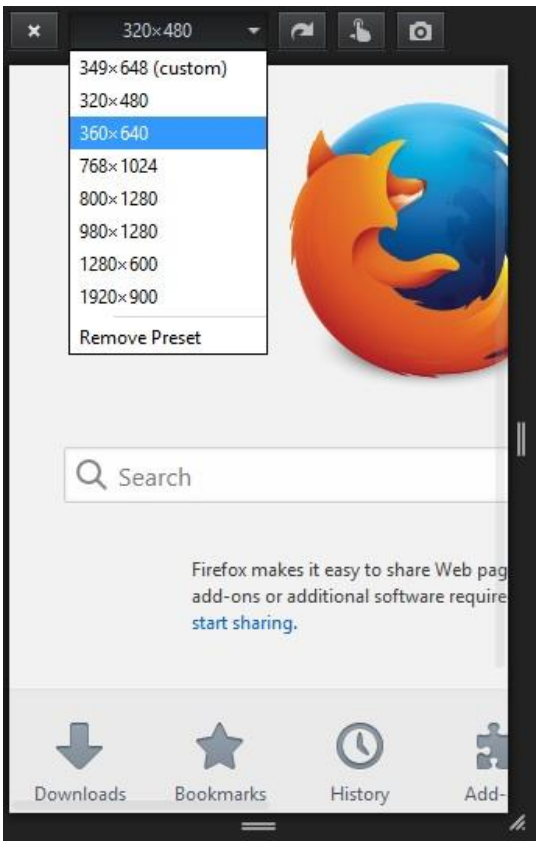


KUVA 31 Sublime, jossa pieni projekti

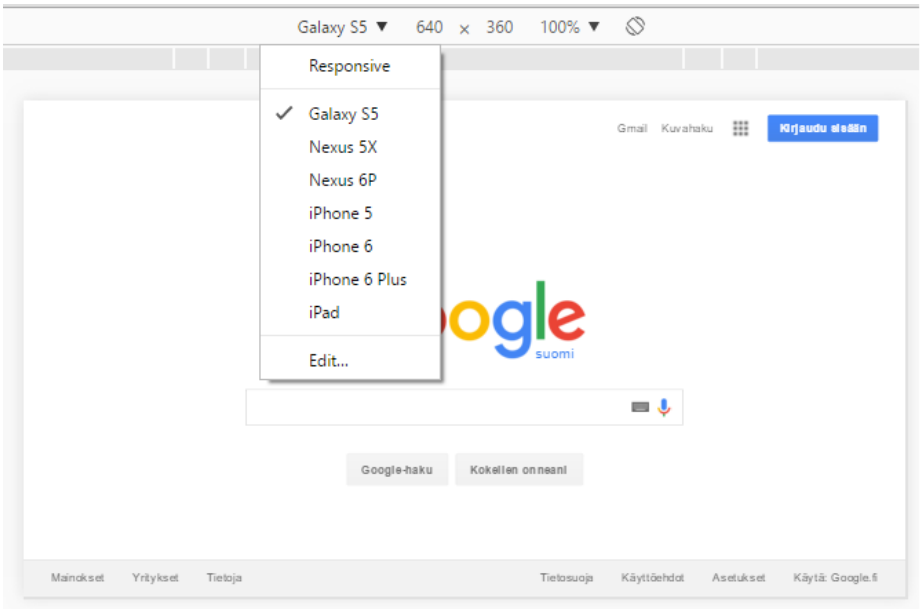
5.2.2 Testausvälineet

Web-sivuja testataan eri selaimilla. Eniten käytettyjen selainten joukosta löytyy Chromen ja Firefoxin lisäksi 4 muuta. Testauksessa on mukana myös eri laitteet (puhelimet ja tabletit) ja niiden omat selaimet sekä näyttökoot. Tästä kertyy iso kasa testattavaa. Kehittäjien onneksi, useimmilla selaimilla on hyvät työkalut, jolla voidaan emuloida useamman laitteen näyttöjä. Testauksessa olisi kuitenkin hyvä olla jokin kosketusnäytön omaava laite.

Firefoxin kehitystyökaluihin kuuluu responsiivinen näkymä (Kuva 32), jossa näytön leveyden ja korkeuden voi säätä haluamakseen ja säädöt voidaan tallentaa, jotta seuraavassa testauksessa niitä ei tarvitse luoda uudestaan. Samasta työkalusta löytyy kuvankaappaustoiminto, jolla voidaan nopeasti tallentaa testauksessa ilmenneet virheet, ilman suurempaa taukoa raportoinnin ja testauksen välillä. Selaimissa on myös tarkistuksiin tarkoitettu konsoli ja jopa hidasta internetyhteyttä voidaan simuloida.



KUVA 32 Firefoxin responsiivinennäkymä



KUVA 33 Chromen responsiivinennäkymä

6 PROJEKTI

6.1 Määrittely

Smartin laajuuden vuoksi opinnäytteen aikana päivitettävät sivut rajattiin 12 sivuun. Sivut päivitetään nykypäivän standardeihin sekä käyttämään Bootstrap 3 kehitysalustaa. Projektin aikana sivujen ulkoasu tulee muuttumaan, mutta tavoite on myös säilyttää asiakkaille tutuksi tullut käyttöliittymä. Projektin vetäjän kanssa sovittiin, että sivuista tehdään aluksi ehdotelmat kuvin, jotka käydään sittemmin läpi.

Järjestelmän ulkoasua suunniteltaessa ainoa sääntö oli, ettei muutos saanut olla liian radikaali. Smartilla on satoja niin pitkäaikaista kuin uusiakin käyttäjiä, joille on koulutettu tietynlainen käyttöliittymä. Liian iso muutos voi aiheuttaa turhanpäiväisiä sekaannuksia. Niinpä järjestelmän värimaailma, kuvakkeet ja elementtien sijoitus pyrittiin pitämään ennallaan.

6.2 Prototyyppien teko

Riittävien määrittelyjen puitteissa voitiin aloittaa sivujen suunnittelu. Työkaluina käytettiin aikaisemmin mainittua Photoshop-kuvanmuokkausohjelmaa. Vaikka ohjelma on maksullinen, on siitä käytettävissä 30 päivän ilmaisversio. Jos uudistukset vaativat valmiiden kirjastojen liittämistä sivulle, kartoitettiin nämä jo ennen implementointivaihetta. Uusia kirjastoja tai lisäosia etsittäessä on tärkeää ottaa huomioon sen tuoreus. Vanhempia lisäosia joita ei enää päivitetä, on turha lisätä oman tuotteen kehitykseen. Vanhat lisäosat saattavat sisältää vanhentunutta syntaksia tai olla riippuvaisia, jostakin muusta kirjastosta, joka on jo vanhentunut. Vanhemmissa jQueryn ja Bootstrapin päälle rakennetuissa lisäosissa esiintyy yleensä edellä mainittuja ongelmia.

6.3 Prototyyppien tarkastelu

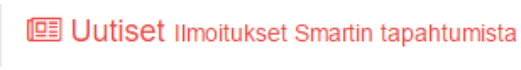
Aikaisemmin määritellyt päivitykset käytiin läpi ja niitä verrattiin prototyyppeihin. Tehdyistä prototyypeistä hyväksytyt päätettiin siirtää mahdollisimman pian testiympäristöön. Näin saatiin heti kartoitettua mahdollisia ongelmia. Osa prototyypeistä siirrettiin jatkokehitykseen saatujen palautteiden kera. Seuraavista kuvista nähdään, miten pieniäkin muutokset voivat olla.

Kuva 34 Vanha päävalikko

Kuva 35 Uusi päävalikko

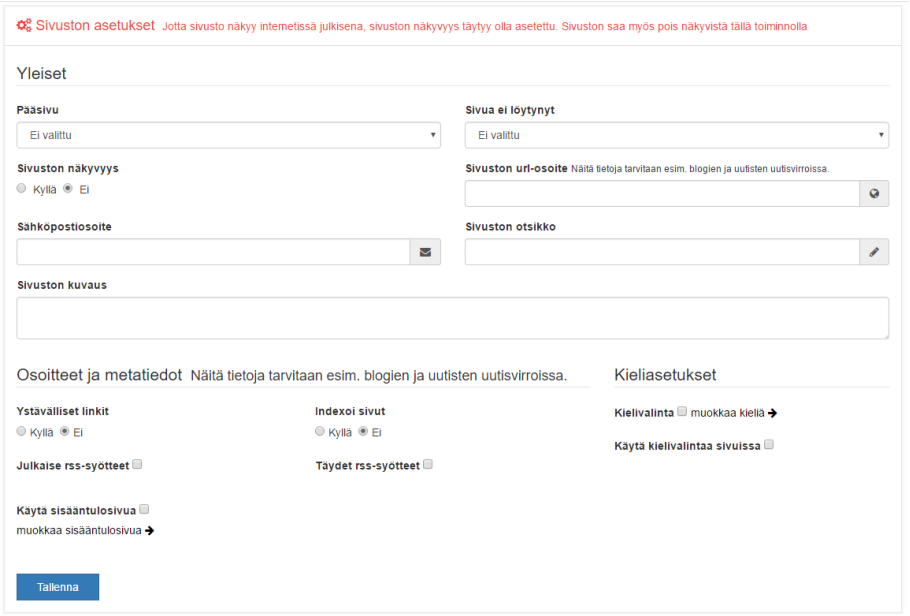
6.4 Käyttäjien ohjeistus

Käyttäjän ohjeistuksen lisäämiseksi, linkkien, nappien ja sopivissa tapauksissa myös otsikkotekstien yhteyteen lisättiin tarkoitusta kuvaavia kuvakkeita. Käytetyt kuvakkeet ovat peräisin Font Awesome-kirjastosta. Kuvakkeet käyttäytyvät kuten muu teksti, joten niiden väriä ja kokoa on helppo muuttaa. Lisäksi otsikoiden perään saatettiin liittää pieni ohjeteksti.



Kuva 36 Otsikko, kuvake ja lyhyt ohjeteksti

Järjestelmän isompia lomakkeita pilkottiin väliotsikoin ja joihinkin kenttiin lisättiin HTML5:n esittelemä placeholder-attribuutti, jolla voidaan esittää haluttu teksti tyhjän tekstikentän sisällä.



Kuva 37 Sivuston asetukset-lomake

6.5 Implementointi testiympäristöön

Sivujen päivitys tapahtui iltaisin tai vapaapäivinä, kun testiympäristö ei ollut muuten käytössä. Päivityksen kohteita piti aina säännöstellä, jotta työt saataisiin valmiiksi ennen seuraavaa työpäivää, niin ettei muu kehitystyö häiriinny.

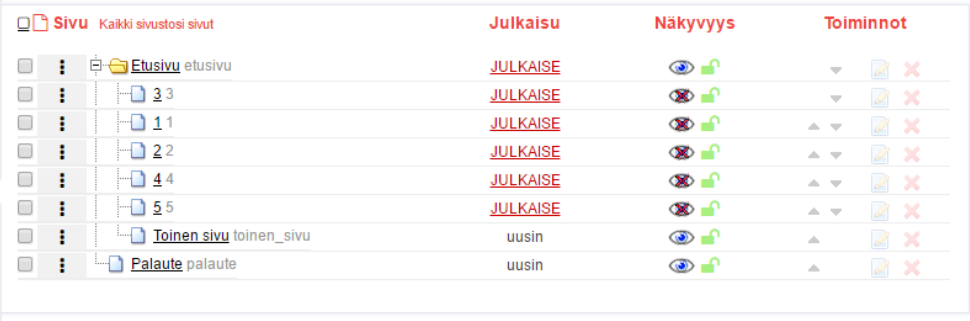
Työ oli hyvin itsenäistä, joten työn alussa aikaa kului oikeiden tiedostojen etsimiseen. Lisäksi uudistuksien vaatimat kirjastot, kuvat ja muut tiedostot pitivät sijoittaa niille tarkoitettuihin kansioihin ja upottaa itse sivustolle. Tässä kohtaa oli vaarana, että uusien ja vanhojen kirjastojen kanssa tulee päällekkäisyyksiä. Koska uudistukset koskevat vain osaa sivuista, on vanhojen kirjastojen oltava mukana projektissa. Myös versiot muuttamattomista tiedostoista tuli säilyttää, joten työ siirrettiin GitLab-nimiseen versiohallintaan, jolla voidaan tarvittaessa palauttaa Smart aikaisempaan versioon.

6.6 Sivujen hallinta

Sivujen hallinta on tärkeä osa CMS:n perustoimintoja. Smartissa sivujenhallinta on erillinen sivu, jossa listataan kaikki sivustolla olevat sivut. Listauksen kautta sivuja voidaan luoda, poistaa, julkaista tai piilottaa. Myös sivukohtaisiin asetuksiin, esim. osoitteen tai nimen muuttamiseen päästään tämän sivun kautta.

Kuva 38 Vanha Sivut-näkymä.

Kuvassa 38 olevassa käyttöliittymässä nähdään sivujen siirtoon tarkoitetut Toiminto-otsikon alla olevat nuolet, joita klikkaamalla sivu siirtyy osoitettuun suuntaan. Jos sivuja on yli 30 ja siirto tapahtuu ensimmäisestä viimeiseksi, voi näinkin yksinkertainen tehtävä käydä työlääksi.
Tämä ongelma on ratkaistu lisäämällä sivujen valintaruudun viereen kuvake, josta hiirellä vetämällä sivu irtoaa listasta ja sen voi siirtää haluamaansa kohtaan. Siirron jälkeen toiminto vahvistetaan käyttäjältä. Yksittäisen sivun lukitseminen on siirretty asetusnäkymästä Näkyvyys-otsikon alle ja sivun asetukset aukeavat nyt modaalisena, näkymän vaihtamisen sijaan.



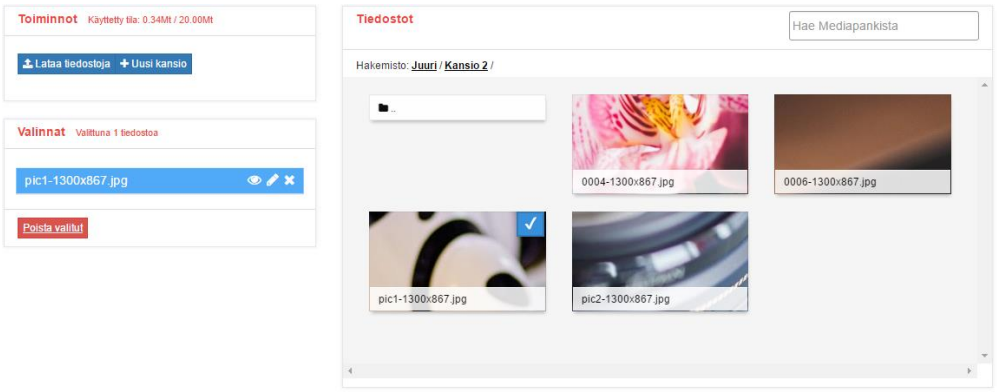
Kuva 39 Uusi Sivut-näkymä

6.7 Mediapankki

Mediapankkiin käyttäjä lataa kaikki sivuillaan käytettävät tiedostot. Toki tiedostot voivat tulla järjestelmän ulkopuoleltakin. Tiedostot voidaan ryhmitellä kansioin, nimetä uudelleen, poistaa, jne.

Aikaisemmassa mediapankin versiossa kansiot sekä tiedostot listattiin allekkain ja niitä sai muokata, joko sisällöltään tai nimeltään. Uudessa versiossa mediapankin toiminnot ovat samat, mutta ulkoasu on päivitetty sekä muutamia toimintoja on lisätty (Kuva 40). Toimintonapit on sijoitettu sivun vasempaan laitaan ja valitut tiedostot avautuvat niiden alle. Valituilla tiedostoilla on kullakin omat toimintonsa. Esimerkiksi suuremmista kuvatiedostoista voidaan halutessa rajata tietty osa ja tallentaa se erilliseksi kuvaksi. Myös kuvan skaalaaminen onnistuu.

Tiedostojen lataaminen aloitetaan ”Lataa tiedostoja”-napista, jolloin käyttäjälle esitetään ikkuna, josta ladattavat tiedostot valitaan. Tämän lisäksi mediapankkiin voidaan raahata tiedostoja selaimen ulkopuolelta. Lataus käynnistyy kun käyttäjä tiputtaa tiedostot työalueelle. Jos toiminto tehdään työalueella olevan kansion yläpuolella, kysytään käyttäjältä haluaako hän ladata ne osoitettuun vaiko aktiivisena olevaan kansioon.



Kuva 40 Uusi mediapankki

7 YHTEENVETO

Opinnäytetyön tavoitteena oli suunnitella ja toteuttaa Smart Kotisivutyökalun sivujen päivitys, käyttämällä suosittua Bootstrap-kehitysalustaa. Tavoitteena oli myös parantaa omaa ymmärrystä Smartin toiminnasta sekä kasvattaa osaamista ohjelmistokehittäjänä ja käyttöliittymä suunnittelijana. Sivujen suunnittelussa tuli ottaa huomioon se, että työkalu on jo käytössä, joten muutoksien piti olla hillittyjä.

Web-ohjelmointi ja käytettyjen tekniikoiden hyödyntäminen ei ollut kirjoittajalle vierasta. Smart Kotisivutyökalusta käyttäjänä omattiin noin vuoden kokemus, mutta sen kehittäjänä oleminen oli täysin uutta. Smartin toiminnan tarkastelusta ja kehityksestä koettiin olleen suurtakin hyötyä, joka toi lisävarmuutta järjestelmän muussa päivittäisessä käytössä.

Työn koodaamisosa oli helppo aloittaa, sillä pohja ja ideat olivat jo valmiina. Vaikkakin työ oli ensiksi hidasta, oikeiden tiedostojen ja paikallaan olevien menetelmien tarkastelua, tuli siitä nopeasti rutiinin omaista ja mielekästä. Jo valmiina oleviin pohjiin lisättiin Bootstrapin tarjoamat elementit ja joitakin vanhoja elementtejä korvattiin uusilla. Suurin osa työstä oli elementtien muuttamista Bootstrapille sopiviksi, joka on käytännössä pelkkä HTML-syntaksimuutos. Ainoa ongelma oli vanhojen ja uusien kirjastojen päällekkäisyys, mutta tämä oli helposti ratkaistavissa. Vanhat ja uudet tiedostot säilytettiin erillisinä versiona joiden välillä pystyttiin tarvittaessa liikkumaan, mikäli jokin toiminto ei toiminut vielä uusien kirjastojen lisäämisen jälkeen.

Jos työ aloitettaisiin uudelleen, voitaisiin myös Hurjan asiakkailta kysyä mahdollisia kehityskohteita. Ideat työssä oleviin päivityksiin tulivat vain yrityksen sisältä. Käytettyihin tekniikoihin voitaisiin lisätä WebPack, jolla yhdistellään ja minimoidaan projektin eri resurssitiedostoja, joka taas nopeutetaan sivulatauksia. Lisäksi LESS:n tai SASS:n käyttäminen alusta alkaen olisi vähentänyt kirjotettavan CSS:n määrää.

Opinnäytetyön aloitus viivästyi, eikä kaikkia sivuja saatu päivitettyä opinnäytetyön aikana. Tämä ei kuitenkaan estä päivityksien jatkamista. Opinnäytetyönä tehtyä osiota voidaan pitää Smartin päivitysprojektin kickstarttina. Jatkossa loput Smartin sivuista tullaan päivittämään käyttäen samoja tekniikoita. Yleisimmät asiakkaalle tarjottavat lisäosat kuten artikkelit, liukukuvat, lomake-editorit, postituslistat ja kuvagalleriat ovat päivityslistassa ensimmäisiä. Kuvagalleriaan, liukukuviin ja lomake-editoriin voidaan käyttää samaa ”vedä ja pudota”-toimintoa kuin sivujenhallinnassa. Tämä olisi hyödyllinen toiminto, kun kuvan tai lomakekentän paikkaa halutaan vaihtaa. Loppujen päivityksien valmistuttua ja huolellisten testauksien jälkeen, projekti voidaan siirtää tuotantoon.

LÄHTEET

Barker, D. (2016) A. What a CMS Does, *Web Content Management.* O'Reilly Media, Inc.

Barker, D. (2016) B. What Is a Content Management System?, *Web Content Management.* O'Reilly Media, Inc.

Chaudhary, M.;& Kumar, A. (2015) A. Who Was the Revolutionary?, *Practical jQuery.*

Chaudhary, M.;& Kumar, A. (2015) B. Why jQuery. *Practical jQuery.* Apress.

Hissom-Daugherty, A. E. (24. 4 2016). *Introduction to HTLML5 and CSS3*. Haettu osoitteesta Amy E. Hissom-Daugherty: http://amyhissom.com/HTML5-CSS3/history.html#4

jQuery. (24. 4 2016). *jQuery*. Haettu osoitteesta jQuery: https://code.jquery.com/

Land Of Code. (23. 4 2014). *HTML History*. Haettu osoitteesta Land of Code: http://www.landofcode.com/html-tutorials/html-history.php

Lie, H. W.;& Bos, B. (1999). The CSS saga, *Cascading Style Sheets.* Addison Wesley. Haettu osoitteesta Chapter 20 - The CSS saga.

MacDonald, M. (2014). What You Can Build with WordPress. *WordPress: The Missing Manual, 2nd Edition.* O'Reilly Media, Inc.

McFarland, D. S. (2014) A. What Is JavaScript?, *JavaScript & jQuery: The Missing Manual, 3rd Edition.* O'Reilly Media, Inc.

McFarland, D. S. (2014) B. What Is jQuery?, *JavaScript & jQuery: The Missing Manual, 3rd Edition.* O'Reilly Media, Inc.

McWaters, M. D. (19. 1 2016). *A Short History of Responsive Web Design*. Haettu 25. 4 2016 osoitteesta LinkedIn: https://www.linkedin.com/pulse/short-history-responsive-web-design-matthew-d-mcwaters

Mozilla Developer Network. (23. 4 2016). *CSS*. Haettu osoitteesta Mozilla Developer Network: https://developer.mozilla.org/en-US/docs/Web/CSS

Peterson, C. (2014). Preface., *Learning Responsive Web Design.* O'Reilly Media, Inc. .

Rahman, S. F. (2014). Jump Start Bootstrap., *Jump Start Bootstrap.* SitePoint.

Reid, J. (2015) A. What Is HTML5? Teoksessa J. Reid, *HTML5 Programmer's Reference.* Appress.

Reid, J. (2015) B. Interactive Elements., *HTML5 Programmer's Reference.* Appress.

StatCounter. (2016) A. *StatCounter*. Haettu 5. 5 2016 osoitteesta StatCounter: http://gs.statcounter.com/#desktop+mobile+tablet-comparison-ww-monthly-201603-201603-bar

StatCounter. (2016) B. *StatCounter - Global Stats*. Haettu 24. 4 2016 osoitteesta StatCounter - Global Stats: http://gs.statcounter.com/#desktop+mobile+tablet-comparison-ww-monthly-201001-201604

Twitter. (19. 8 2011). *Bootstrap from Twitter*. Haettu 24. 4 2016 osoitteesta The Twitter Developer Blog: https://blog.twitter.com/2011/bootstrap-from-twitter

W3C. (27. 6 2012). *A Short History of JavaScript*. Haettu 24. 4 2016 osoitteesta W3C - Community: https://www.w3.org/community/webed/wiki/Main_Page

W3Counter. (24. 4 2016). *Browser & Platform Market Share - March 2016*. Haettu osoitteesta W3Counter: https://www.w3counter.com/globalstats.php?year=2016&month=3

W3Schools. (ei pvm). *HTML5 Semantic Elements*. Haettu 3. 4 2016 osoitteesta W3Schools: http://www.w3schools.com/html/html5_semantic_elements.asp

WordPress. (24. 4 2016). *WordPress*. Haettu 21. 4 2016 osoitteesta Suomi - WordPress: https://fi.wordpress.org/

World Wide Web Consortium. (24. 4 2016). *CSS1 Fact Sheet*. Haettu osoitteesta World Wide Web Consortium: https://www.w3.org/Press/CSS1-fact.html