

Janne Aalto

# Interaktiivinen verkkosivusto

---

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Mediatekniikan koulutusohjelma

Insinööriytyö

29.4.2014

Tekijä Otsikko	Janne Aalto Interaktiivinen verkkosivusto
Sivumäärä Aika	33 sivua 29.4.2014
Tutkinto	insinööri (AMK)
Koulutusohjelma	mediatekniikka
Suuntautumisvaihtoehto	digitaalinen media
Ohjaajat	tuottaja, projektipäällikkö, Anton Qvist yliopettaja Kari Salo
<p>Insinööriyönä tehtiin televisiokanavalle uusia tekniikoita hyödyntäen palvelu antamaan lisäarvoa elokuvien katseluun. Tutkimusten mukaan suuri määrä ihmisistä, jotka omistavat älylaitteita, käyttää niitä televisiota katsoessaan. Televisiokanavat haluavat hyödyntää tätä tarjoamalla palvelun, josta asiakas saa hyödyllisen jatkeen elokuvankatselukokemukseen.</p> <p>Televisiokanavalle päätettiin suunnitella ja toteuttaa niin sanottu 2nd Screen -sovellus, jossa on moderoituva reaaliaikainen keskustelu, johon asiakas pystyy helposti ajastamaan viestejä. 2nd Screen -sovellukset ovat esimerkiksi television katselun aikana käytettäviä palveluita, jotka tukevat television ohjelmaa. Ajastetut viestit haluttiin rikastuttamaan elokuvakokemusta kertomalla esimerkiksi kuvauspaikoista tai ohjaajasta lisätietoja synkronoituina lähetyksiaikaan. Sivustolle tehtiin Facebook- ja Twitter-integraatio vähentämään viestien törkyisyyttä. Tämän lisäksi viesteistä suodatettiin yleisimmät kirosanat. Viestien lähettäminen tapahtuu Ajaxin ja WebSocket-yhteyden kautta.</p> <p>Verkkosivusto toteutettiin Wordpress-julkaisujärjestelmällä helpottamaan asiakkaan sisällönsyöttöä, ja samalla saatiin käyttöön esimerkiksi kommentoijien musta lista. Wordpress-lisäosalla Gravity Forms lisättiin lomake, josta kerättiin potentiaalisten asiakkaiden yhteystietoja.</p> <p>Verkkosivun Wordpress-teeman pohjana käytettiin Foundation 4 -ohjelmistokehystä. Sivustolla oli todella tärkeää, että se toimi mobiililaitteissa moitteetta, ja Foundationin avulla se onnistui. Kehitykseen otettiin myös avuksi Node.js- ja Socket.IO-kirjastot, joilla saatiin WebSocket-yhteyksille laaja selain- ja laitteistotuki.</p> <p>Verkkosivusto ei kuitenkaan saavuttanut suosiota, eikä sitä enää ylläpidetä.</p>	
Avainsanat	Node.js, Wordpress, Socket.IO, JavaScript, Leffaputki.fi

Author Title	Janne Aalto Interactive website
Number of Pages Date	33 pages 29 April 2014
Degree	Bachelor of Engineering
Degree Programme	Media Technology
Specialisation option	Digital Media
Instructors	Anton Qvist, Project manager, Producer Kari Salo, Principal Lecturer
<p>The subject of the thesis was to make a web application for one of the leading multimedia broadcasting companies in Finland. According to reports, an increasing amount of people use tablets and smartphones while watching TV. The application was designed to give extra value to watching movies by adding an extra screen with useful information to the consumer.</p> <p>The application would consist of a real-time chat that the broadcasting company can moderate in real-time and time messages to be sent during the movie. The messages would for example give more information about the filming locations or the director, synced to the broadcast time. Facebook and Twitter login was made mandatory to hopefully reduce the amount of unsuitable messages to the site. The messages were also filtered for the most common swearwords in Finnish. The messages were sent with Ajax and through Web Socket connections.</p> <p>The website was made with Wordpress content management system, to give the client an easy platform to change images and texts in the site and it also provided a blacklist functionality to ban users from sending messages. The site also included a form made with Gravity Forms Wordpress plugin, to collect contact information of potential customers.</p> <p>Foundation front-end framework was used as a starting point in the Wordpress theme development. It was really important that mobile devices worked flawlessly and Foundation framework provided the necessary tools to make that possible. Node.js and Socket.IO was used to provide good browser and hardware support for real-time chat.</p> <p>The site never became popular and it is not updated anymore.</p>	
Keywords	Node.js, Wordpress, Socket.IO, JavaScript, Leffaputki.fi

# Sisällys

## Lyhenteet

1	Johdanto	1
2	Interaktiivisen verkkosivuston kehittäminen ja yleiset tekniikat	1
2.1	Merkintäkieli	2
2.2	HTML5-rajapinnat	3
2.3	Tyylitiedostot ja dynaaminen tyylimääritelmäkieli	4
2.4	JavaScript ja JavaScript-kirjastot	5
2.5	LAMP, MAMP ja WAMP	12
2.6	Palvelinpuolen JavaScript	14
2.7	HTML-kirjastot ja responsiiviset verkkosivustot	17
2.8	Julkaisujärjestelmät	19
3	Interaktiivisen verkkosivuston kehittäminen	20
3.1	Wordpress-teeman kehitys	20
3.2	Palvelinpuolen JavaScriptin integroiminen Wordpressiin	23
3.3	Mobiilioptimointi	25
4	Interaktiivisen verkkosivuston suunnittelu	27
4.1	Reaaliaikainen keskustelu	27
4.2	Kilpailu	31
5	Yhteenveto	32
	Lähteet	34

## Lyhenteet

Canvas	HTML5-elementti, jolla piirretään grafiikkaan verkkosivustolle
CSS	Cascading Style Sheets, WWW-dokumenttien tyyliohjeiden laji.
DOM	Document Object Model, mahdollistaa merkintäkielen muokkaamisen ohjelmointikielestä riippumatta.
Hakukoneoptimointi WWW-dokumentin näkyminen hakukoneissa.	
HTML	HyperText Markup Language, merkintäkieli, jota käytetään internetsivujen kuvaamiseen.
IP	Internet Protocol, TCP-IP mallin internetkerroksen protokolla. Kuvataan internetpalvelimien osoite
Iframe	HTML-elementti, jolla upotetaan toinen HTML-dokumentti toiseen HTML-dokumenttiin
JavaScript	Netscape Communications Corporationin kehittämä merkintäkieli, jolla voidaan muokata HTML-dokumenttien sisältöä DOM-rajapinnan avulla.
JSON	JavaScript Object Notation, tiedonsiirtomuoto, joka toimii yleisimmillä ohjelmointikielillä.
SASS	Syntatically Awesome Style Sheet, CSS-preprossessori
URL	Uniform Resource Locator, Internetosoite

## 1 Johdanto

Nelonen Medialle on tarkoitus tehdä uusia tekniikoita hyödyntäen palvelu antamaan lisäarvoa elokuvien katseluun. Nelonen Media on yksi Suomen suurimmista monimedialalosta ja jatkuvasti kehittää television ja radion tueksi internetsisältöä. Tutkimusten mukaan suuri määrä ihmisistä, jotka omistavat älylaitteita, käyttää niitä televisiota katsoessaan. Nelonen haluaa hyödyntää tätä tarjoamalla palvelun, josta asiakas saa hyödyllisen jatkeen elokuvankatselukokemukseen. (1.)

Insinööriyönä suunnitellaan ja toteutetaan niin sanottu 2nd Screen -sovellus, jossa on moderoitava reaaliaikainen keskustelu, johon asiakas pystyy helposti ajastamaan viestejä. 2nd Screen -sovellukset ovat esimerkiksi televisionkatselun aikana käytettäviä palveluita, jotka tukevat television ohjelmaa. Ajastetut viestit halutaan rikastuttamaan elokuvakokemusta kertomalla esimerkiksi kuvauspaikoista tai ohjaajasta lisätietoja synkronoituna lähetyksiaikaan. Nelonen Media haluaa myös tarjota yhteistyökumppaneille mahdollisuuden saada näkyvyyttä sivulla mainoksilla. (1.)

Sisällön tuottaminen ja ulkoasun suunnittelu työhön saadaan Nelonen Medialta. Palvelusta halutaan helppokäyttöinen ja tärkeänä pidetään mobiililaitteiden toimivuutta. Lisäksi on tärkeää, että ei käytetä henkilöresursseja palvelun seuraamiseen ja sisällön syöttämiseen elokuvien aikana, vaan se tulee olla mahdollista tehdä ajastetusti.

Mobiilikäyttöjärjestelmille ei haluta alkaa kehittää omia sovelluksia, vaan palvelu toteutetaan verkkosovelluksena. Tämä on kustannustehokasta, ja lisäksi se mahdollistaa palvelun nopean käyttöönoton. Sivuston toteutukseen on varattu aikaa noin viisi henkilöviikkoa.

## 2 Interaktiivisen verkkosivuston kehittäminen ja yleiset tekniikat

Tim Berners Lee loi HTML-merkitäkielen (Hypertext markup language), http-protokollan (Hypertext Transfer Protocol), URL:t (Uniform resource locator) ja maailman ensimmäisen internetselaimen. Elokuussa 1991 hän julkaisi maailman ensimmäisen internetsivuston. Nopeasti sen jälkeen monet yritykset alkoivat kehittää internetse-laimia, ja elokuussa 1995 kehitettiin ensimmäinen versio JavaScriptistä. CSS (Casc-

ding Style Sheets) tuotiin mukaan lisäämään tyyliä ja tunnelmaa. Vuonna 2007 Applen kehittämä iPhone-älypuhelin mullisti suosiollaan internetin käytön tuomalla sen jokaisen taskuun. (2.)

## 2.1 Merkintäkieli

HTML5 on verkkosivustojen ja -sovellusten merkintäkieli. Sitä ei ole vielä hyväksytty suositeltavaksi standardiksi. Se on hyvin laajalti tuettu asiakaspäätteissä, ja emuloimalla halutut ominaisuudet tukea voidaan kasvattaa riittäväksi, jotta sitä voi huoletta käyttää asiakasprojekteissa. HTML5 tuo monia ominaisuuksia, jotka olivat ennen mahdollisia vain internetselainten lisäosien kautta. Esimerkiksi lisäosalla Adobe Flash on hyvin suuri käyttäjäkunta, mutta sen esittäminen verkkosivulla muun sisällön seassa on ongelmallista ja esimerkiksi Apple on estänyt Flashin suorittamisen mobiililaitteissa kokonaan. (3, s. 5–6.)

Google analysoi miljoonia verkkosivustoja ja huomasi, että suurimmalla osalla sivustoista on käytössä samoja elementin div identifiointeja ja luokka-arvoja, kuten header ja footer, joten oli järkevää lisätä ne elementteinä HTML5:een. Samalla lisättiin muitakin sivun lähdekoodia jaksottavia elementtejä, kuten section, article, aside ja nav. Lähdekoodia haluttiin myös selkeyttää poistamalla kuvaavia elementtejä, kuten font, center ja big, ja jättää sivun tyylit kokonaan CSS:lle. Uusilla semanttisilla elementeillä ei kuitenkaan ole mitään merkitystä sivun ulkoasun kannalta. Ne voitaisiin ihan yhtä hyvin nimitä elementti1 ja elementti2 ja niin edelleen. Elementeillä on kuitenkin vaikutusta haku-koneoptimointiin. (3, s. 9–17.)

Audio- ja videotiedostoissa on erikseen raidat videolle, äänelle ja metadatalle. Videoraita ja ääniraita yhdistetään toistettaessa videota. Metadata sisältää lisädataa videosta, kuten kansikuvan, otsikon, tekstitykset ja niin edelleen. HTML5-video- ja -audioelementit mahdollistavat videoiden ja äänen selaimessa suoraan ilman lisäosien lataamista. HTML5:een ei kuitenkaan ole määritelty yhteistä koodekkia, minkä takia sivuille ladataan videot ja ääni monessa eri tiedostomuodossa. Mediaelementtejä tukemattomille selaimille voidaan ohjelmoida tuki videoihin käyttämällä hyväksi lisäosia, kuten Flash Playeriä. (3, s. 65–67.)

Canvas-elementille voidaan ohjelmoida grafiikkaa, kaavioita, kuvia ja animaatioita. Canvas-elementtejä on suhteellisen helppo ohjelmoida: Lisätään <canvas>-elementti HTML5-sivun lähdekoodiin, haetaan JavaScriptillä DOM:sta viittaus canvas-objektiin ja sen piirtokonteksti `document.getElementById('canvas')[0].getContext('2d')`. Metodilla `beginPath()` aloitetaan reitin piirtäminen. Annetaan lähtö- ja loppukoordinaatit elementissä ja `.stroke()`-metodilla piirretään viiva canvas-elementille. Kuviot ja reitit kannattaa aina tehdä origopisteessä ja sen jälkeen siirtää ne haluttuun paikkaan. Kuvioiden ja reittien skaalaaminen ja kääntö tehdään aina suhteessa origoon. (3, s. 6–7, 25–32, 51.)

## 2.2 HTML5-rajapinnat

HTML5:ssä on monia hyvin hyödyllisiä JavaScript-rajapintoja. Niillä voidaan muun muassa pitää reaaliaikaista yhteyttä palvelimen ja käyttäjän välillä, määrittää käyttäjän lokaatio, tallentaa ääntä, kuvaa ja videota käyttäjän laitteesta ja tallentaa tietoa käyttäjän laitteeseen. JavaScript-rajapintoja kehitetään jatkuvasti lisää, ja osaa niistä ei koskaan oteta varsinaiseen käyttöön. (4.)

HTML5 WebSocket -rajapinta mahdollistaa samanaikaisen kaksisuuntaisen yhteyden internetpalvelimen ja käyttäjän välillä. Se on suunniteltu toimimaan nykyisen web-infrastruktuurin kanssa. Siinä vaihdetaan http-protokollan tilalle WebSocket-protokolla, jos se on tuettu. Se otetaan käyttöön luomalla uusi WebSocket-instanssi tietyssä URL-osoitteessa, ja sen jälkeen viestien lähettäminen on mahdollista. On kuitenkin hyvä muistaa, että WebSocket on suhteellisen uusi protokolla ja vanhemmat selaimet eivät tue sitä. (5.)

HTML5 Media Capture and Streams on JavaScript-rajapinta, joka tuo ohjelmoijan käyttöön käyttäjän laitteen kameran, mikrofonin tai molemmat. Sillä on mahdollista tallentaa videota tiedostoksi palvelimelle, ja sen pohjalta on tehty WebRTC-rajapinta, jolla on mahdollista käydä reaaliaikaisia videokeskusteluja käyttäjien kesken ilman selaimen lisäosia. WebRTC kuten suurin osa HTML5-rajapinnoista tarjoaa yksinkertaisen käyttöliittymän sen käyttöön. Tekniikka ei kuitenkaan ole vielä valmista, ja rajapinta luultavasti vielä muuttuu. Tällä hetkellä rajapintaa ei tue kuin muutama selain. (6; 7.)



### 2.3 Tyylitiedostot ja dynaaminen tyylimääritelmäkieli

CSS tulee sanoista Cascading Style Sheets. CSS:llä määritellään tyyliohje WWW-dokumentille. HTML määrittelee WWW-dokumentin rakenteen, ja CSS:llä se saadaan näyttämään hyvältä.

SASS on lyhenne sanoista Syntactically Awesome Style Sheets. Se koostuu kahdesta syntaksista Sass ja SCSS. Sass on kuitenkin vanhentunut, ja tästä eteenpäin viitataan SASS:n SCSS-syntaksiin SASS:na. SASS on CSS-preprosessori. Se tuo CSS:n kirjoittamiseen muun muassa muuttujat, sisäkkäisasettelun, perinnän, ehto- ja toistolausekkeet ja operaattorit. (8.)

CSS:ssä kirjoitetaan jokainen valitsin ja sen perilliset omalle rivilleen ja niille määritellään omat tyylinsä. SASS:ssa taas voidaan kirjoittaa perilliset päävalitsimen sisään, mikä helpottaa huomattavasti tyylimäärittelyjen lukemista, kun valitsimet eivät kasva suuriksi. SASS kuitenkin prosessoidaan CSS:ksi. Koodiesimerkissä 1 prosessoidaan SASS CSS-lausekkeeksi. (8.)

```
//SCSS
.parent{
  background-color:black;
  .child{
    background-color:blue;
  }
}

//CSS
.parent{
  background-color:black;
}
.parent .child{
  background-color:blue;
}
```

Koodiesimerkki 1. SASS-lauseke ja siitä generoituva CSS.

SASS:n etuna on myös muuttujat. Hyvänä esimerkkinä niistä voidaan pitää värejä. CSS:ssä värit voidaan kirjoittaa neljällä eri tavalla: jo määrättyinä merkkijonona, kuten 'black', joka on musta, tai RGB-arvona, jossa annetaan arvo punaiselle, vihreälle ja siniselle, HSL-arvona, johon määritetään värin sävy, kylläisyys ja kirkkaus tai sitten heksadesimaalina. Nämä arvot on kuitenkin vaikeaa muistaa, ja merkkijonoväreistä ei voi olla täysin varma väristä ja sävystä kuin mustan ja valkoisen kanssa. Värit on kuitenkin helppo tallentaa SASS-muuttujiksi, jolloin värille on helppo antaa kuvaava nimi omalle sävyille. Lisäksi jos sävyä tarvitsee muuttaa, se tehdään muuttamalla vain muuttujaa. (8.)

SASS:n valitsimia on mahdollista myös periä. Se tarkoittaa sitä, että kun on kirjoitettu luokka x, jolla on arvo y, sen voi periä luokka z @extend-komennolla, jolloin sille voidaan määrittää lisää arvoja. Lisäksi on mahdollista yhdistää monia SASS-tiedostoja yhdeksi tiedostoksi CSS @import-komennolla. Se vähentää kutsujen määrää sivustolla ja nopeuttaa sivun lataamista. Tuloksena syntyvä CSS-tiedosto on myös mahdollista ja suositeltavaa minimoida. Minimointi tarkoittaa sitä, että tiedostosta poistetaan turhat välilyönnit ja kaikki rivinvaihdot. (8.)

Kaksi muuta suosittua CSS-preprosessoria ovat LESS ja Stylus. Erot LESS:n, SASS:n ja Styluksen välillä ovat kuitenkin lähinnä syntaksissa. LESS:n tekivät suosituksi Twitter Bootstrap-ohjelmistokehys ja LESShat-CSS-elementtikirjasto. SASS oli ensimmäinen preprosessori, joka toi ehto- ja toistolausekkeet preprosessoreihin ja saavutti sillä paljon suosiota. SASS:lle on suunniteltu muun muassa Foundation-ohjelmistokehys ja Compass-CSS-elementtikirjasto. Stylus oli ensimmäinen, joka voitiin prosessoida Node.js:n, Grunt.js:n ja Bower.js:n avulla. Lisäksi siinä on hieman vapaampi syntaksi. (2; 8; 9; 10.)

## 2.4 JavaScript ja JavaScript-kirjastot

Jokaiseen nykyaikaiseen internetselaimeen on tehty JavaScript-tulkitsija ja JavaScriptillä on todella laaja laitteistotuki. Sitä voidaan käyttää puhelimesta pelikonsoleihin. Se kuuluu web-kehittäjien pyhään kolminaisuuteen: HTML, CSS ja JavaScript. (11, s. 1.)

JavaScript on Netscape Communications Corporationin kehittämä merkintäkieli. Sillä voidaan muokata DOM (Document Object Model) -rajapinnan avulla HTML-

dokumenttien sisältöä. Se kehitettiin 1990-luvun loppupuolella. JavaScript suoritetaan asiakaspääätteessä. Koodi ladataan kokonaisuudessaan selaimen, jolloin jatkuvaa palvelinyhteyttä ei ole tarpeellista ylläpitää. Se tekee koodin suorittamisesta sulavam-  
paa. (3, s. 16.)

JavaScript on tietotyyppitön, ja se soveltuu sekä funktionaaliseen että oliopohjaiseen ohjelmointityyliin. JavaScriptin ydin sisältää ohjelmointirajapinnan perustoiminnoille, kuten kokoelmien, tekstin, päivämäärien ja säännöllisten lausekkeiden kanssa toimimisen, mutta se ei kuitenkaan palauta eikä tulosta mitään. Palauttaminen ja tulostaminen on päätelaitteiden vastuulla, jotka tavallisimmin ovat internetselaimia. JavaScriptiä on mahdollista suorittaa myös ilman internetselainta ja verkkoyhteyttä. (11, s. 1–2.)

HTML-merkintäkielen tyylisäännökset suosittelevat käyttämään vain pieniä kirjaimia elementtien kuvauksessa. Se ei ole kuitenkaan pakollista, toisin kuin JavaScriptissä. Muuttujat ovat merkkikokoriippuvaisia, ja esimerkiksi metropolia ja Metropolia ovat kaksi eri muuttujaa. JavaScript-muuttujat voidaan luokitella kahteen tyyppiin, jotka ovat alkeelliset tyypit ja objektityypit. Alkeellisiin tyypeihin kuuluvat boolean-arvot, numerot ja merkkijonot. Lisäksi alkeellisiin tyypeihin kuuluvat "null" ja "undefined", jotka tarkoittavat määrittelemättömiä arvoja. Kaikki muut ovat objektityyppisiä, joihin kuuluvat säännölliset lausekkeet, objektit ja globaalit objektit. Funktiot ovat myös objekteja JavaScriptissä. Funktio-objektilla on suoritettavaa koodia, ja sitä voidaan kutsua palauttamaan laskennallinen arvo. (11, s. 21, 29–30.)

Funktio voidaan myös kirjoittaa käytettäväksi "new"-operaattorin kanssa, jolloin luodaan luokka objekteja, jotka on kuvattu muodostimessa. JavaScriptin ytimeen on luotu muutamia luokkia, kuten Date-luokka, jolla voidaan luoda objekti, joka sisältää tietoa päivämääristä, tai RegExc-luokka, joka mahdollistaa muun muassa säännöllisten lausekkeiden vertailun. (11, s. 30.)

JavaScript-muuttujat ovat tyyppittömiä, mikä tarkoittaa sitä, että muuttujalle voi antaa minkä tahansa tyyppin arvon ja myöhemmin ylikirjoittaa se toisella tyyppillä. Muuttujat luodaan avainsanalla "var", ja muuttujalla voi olla joko globaali näkyvyys tai se voi olla käytettävissä vain funktion sisällä. Globaalit muuttujat luodaan funktioiden ulkopuolella. JavaScript-tulkitsijoissa on automaattinen roskan keruu, joten käyttäjän ei tarvitse huolehtia objektien tuhoamisesta. Sen kanssa kannattaa kuitenkin olla tarkkana. Jos objektiin jää viittaus, sitä ei myöskään vapauteta muistista. (11, s. 30–31.)

JavaScriptissä on tuki matemaattisiin toimintoihin plus (+), miinus (-), kerto (\*), jako (/) ja jakojäännös (%), mutta myös objekti "Math", jota voidaan käyttää haastavampiin laskutoimituksiin. Sen avulla saadaan käyttöön monia matematiikan perustoimia, kuten pyöristäminen, potenssit, neliöjuuret ja monia muita. On hyvä kuitenkin pitää mielessä, että JavaScript käyttää binäärilukujärjestelmää. Binäärijärjestelmä laskee tarkasti jakolaskuja  $1/2$ ,  $1/8$  ja niin edelleen, mutta valitettavasti reaali maailman yleisimmät jakolaskut kuten  $1/10$  ja  $1/100$  eivät ole tasalukuja kuten 0,1, vaan sitä hyvin lähellä oleva desimaaliluku. Sen vuoksi esimerkiksi laskut  $1,5-1,4$  ja  $1,2-1,1$  eivät vastaa arvollisesti toisiaan. Tämän takia suositellaan, että desimaalilaskutoimitukset tehtäisiin kokonaisluvuina. Jos on tarve laskea  $1,5-1,4$ , lasketaan ensin kokonaisluvut  $1-1$  ja sen jälkeen desimaalit muutettuna kokonaisluvuiksi  $5-4$  ja yhdistetään ne desimaaliluvuksi 0,1. (11, s. 31–35.)

JavaScriptillä on mahdollista tehdä rikkaita internetsovelluksia, ja sen takia sen opettelu suositellaan kaikille web-kehittämisestä kiinnostuneille. Rikkaat internetsovellukset ovat verkkosovelluksia, joilla on työpöytäsovellusten ominaisuuksia, kuten esimerkiksi sivulla olevia laatikoita tai ikkunoita, joita voidaan raahata, venyttää, vääristää tai muulla tapaa muokata. Lisäksi on mahdollista ladata tiedostoja ja sivun osia ilman koko sivun päivittämistä. (12.)

JavaScriptiä käytettäessä voi kohdata kuitenkin suuria ongelmia, koska selaimissa on omat parsimensa koodin tulkitsemiseen. Ongelmien ilmetessä valtavasta määrästä koodia voi olla vaikea löytää kohtaa, joka aiheuttaa eri tulostuksen toiseen selaimiin. Tilanne on kuitenkin parantunut viime vuosina. (13, s. xxi.)

#### 2.4.1 jQuery

jQuery on JavaScript-kirjasto, jonka on kehittänyt John Resig vuonna 2006. Se on julkaistu MIT (Massachusetts Institute of Technology)- ja GNU GPL (General Public License) -lisenssien alaisuudessa. Lisensseillä taataan, että käyttäjä saa vapaasti muokata, kopioida ja käyttää jQuery-kirjastoa, kuitenkin sillä ehdolla, että lisenssin teksti säilyy lähdekoodissa. Näin ollen myös käyttäjien tuotteet ovat vapaan lähdekoodin ohjelmistoja. jQuery-kirjasto on yksi noin 30 kb:n kokoinen JavaScript-tiedosto. Sitä käyttää yli 55 prosenttia 10 000 suosituimmasta internetsivusta, ja se on maailman yleisin JavaScript-kirjasto. Sillä on hyvä tuki yleisempiin selaimiin, jolloin välttyään suurimmilta

JavaScriptin ongelmilta. Uusimmat versiot 1.10.2 ja 2.0.3 julkaistiin heinäkuussa 2013. (14.)

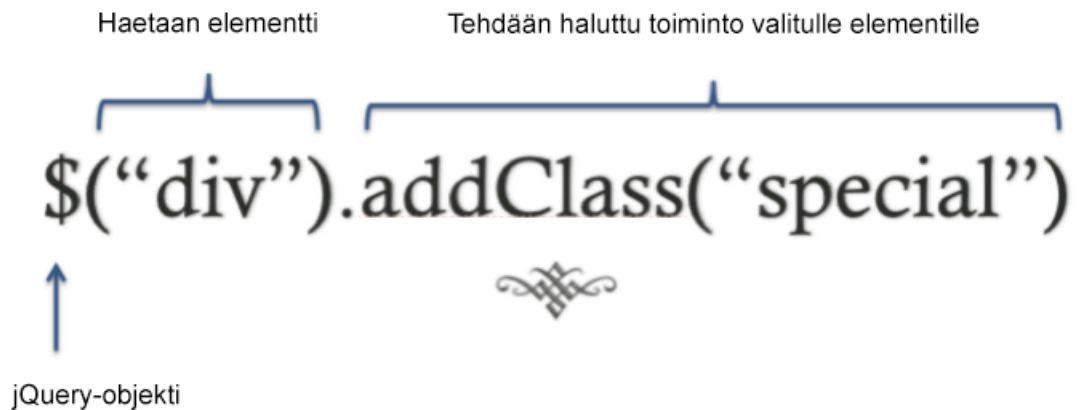
jQuery tekee juuri niin kuin sen motto lupaa "Write less, Do more" (Kirjoita vähemmän, tee enemmän). Se on kehitetty automatisoimaan helppoja JavaScript-tehtäviä ja yksinkertaistamaan vaikeita. jQuery:n rakenne vastaa hyvin paljon HTML:n ja CSS:n rakennetta, joten juuri aloittaneet web-kehittäjät oppivat helposti, miten sitä kirjoitetaan. (14.)

jQuery:n avulla voidaan helposti ja tarkasti kohdistaa valitsimien avulla, mitä HTML-dokumentista halutaan muokata. Sivustolle voidaan lisätä interaktiivisuutta reagoimalla käyttäjän toimiin. Tällöin voidaan tehdä internetsivusta näyttävämpi lataamalla palvelimelta tiedostoja tai tietoja ilman koko sivun päivittämistä. (14.)

#### 2.4.2 Syntaksi

Kuten jo aiemmin mainitsin, jQuery:n syntaksi pohjautuu hyvin pitkälle HTML:n ja CSS:n rakenteeseen. Lausekkeet aloitetaan aina jQuery-objektilla. jQuery-objektina toimii joko `$()` tai `jQuery()`. Objektin merkki on myös mahdollista vaihtaa, jos haluaa käyttää monta eri versiota jQuery:stä tai jos jokin toinen JavaScript-kirjasto käyttää jo mainittuja merkintätapoja. Esimerkeissä käytän `$()`-objektia selvyden vuoksi. (15, s. 17–49.)

jQuery-objektin sisään haetaan elementti tai elementtejä, joita halutaan muokata CSS-valitsimilla. CSS-valitsimia on käytännössä kolme pääryhmää, luokkavalitsimia `$('.luokka')`, elementtivalitsimia `$( 'elementti' )` ja tunnistevalitsimia `$( '#tunniste' )`. Valitsimia on myös mahdollista ketjuttaa, joko valitsemaan suurempi joukko elementtejä, jolloin ne erotellaan pilkulla, tai sitten tarkentamaan valintaa laittamalla valitsinjoukko peräkkäin. (15, s. 17–49.) Kuvassa 1 on esimerkki jQuery-lausekkeesta, jossa haetaan div-elementti ja lisätään sille luokka "special".



Kuva 1: jQuery-lauseke.

Elementtejä voidaan myös suodattaa suotimilla. Tärkeimpiä niistä ovat `:even`, `:odd`, `:has()`, `:first`, `:last` ja `:not()`. `:even`- ja `:odd`-suotimilla voidaan valita joko ensimmäisestä lähtien joka toinen tai sitten toisesta lähtien joka toinen elementti. `:has()`-suotimella voidaan tarkentaa ehtoa, jolla elementit valitaan. `:first` valitsee vain ensimmäisen joukon elementin ja `:last` valitsee viimeisen. `:not()`-suotimella voidaan poistaa elementtejä joukosta. Esimerkiksi haluaisin nostaa suotimista niin sanotun seeprataulukon. (15, s. 17–49.)

Perinteisesti HTML-seeprataulukossa on ollut se ongelma, että kun koodin taulukkoon kirjoittaa ja taulukko on erittäin pitkä, tulee ongelmia sen muokkaamisen kanssa. Taulukko koostuu table-elementistä, jonka sisään laitetaan "tr" eli taulukon rivielementti, jonka sisään laitetaan "td" eli taulukon tietoelementti. Pelkällä HTML:llä ja CSS:llä seeprataulukkoa tehtäessä joka toiselle rivielementille laitetaan luokka-attribuutti ja luokalle määritellään CSS:ssä taustaväri. Ongelmia tulee, jos taulukon keskiväliin pitää sijoittaa uusi rivi, minkä jälkeen pitää muuttaa kaikki rivien luokat uuden rivin jälkeen. (15, s. 17–49.)

jQueryllä kuitenkin pystytään helposti valitsemaan vain joka toinen rivi suotimilla, eikä uudesta rivistä koidu ongelmia, koska rivit valitaan vasta sivun kokonaan latauduttua. (15, s. 17–49.)

### 2.4.3 Metodit

jQueryssä on kymmenittäin erilaisia metodeja, joilla voidaan muuntaa elementtejä. Suurinta osaa niistä käytetään antamaan elementeille jonkin attribuutin arvo, mutta on myös muutamia, joilla voidaan hakea elementin attribuutin arvo. Yleisimmin näistä on käytössä `.addClass()`, jolla voidaan antaa elementille tai elementeille luokka. Muita yleisiä metodeja ovat muun muassa `.css()`, jolla voidaan muokata elementin CSS-ominaisuuksia, `.attr()`-metodi, jolla voidaan antaa elementille mikä tahansa attribuutti ja sen arvo, ja `.append()`-metodi, jolla voidaan tehdä uusi elementti valittujen elementtien sisään. Näillä on myös vastametodit, esimerkiksi `.removeClass()`, jolla poistetaan luokka elementistä. Ne ovat kuitenkin niin yksinkertaisia, että en kerro niistä enempää. (15, s. 17–49.)

Joskus on myös hyödyllistä saada palautettua jonkin elementin attribuutin arvo tai jopa koko elementin ja sen sisäelementtien koodi. Antamalla metodille `.attr()` elementin attribuutin nimen saa palautettua attribuutin arvon. Kuitenkin jos on kyse numeerisesta luvusta, kuten pituus, täytyy käyttää `.val()`-metodia. `.html()`-metodilla saa palautettua koko valitun koodin, jonka voi sen jälkeen kopioida `.append()`-metodilla haluamaansa paikkaan. (15, s. 17–49.)

### 2.4.4 Tapahtumat ja toiminnot

jQuery-kirjastoon on tehty monia todella hyödyllisiä käsittelijöitä, joilla voidaan toteuttaa erilaisia tapahtumia käyttäjän ollessa vuorovaikutuksessa sivun kanssa ja siten tuoda lisää toiminnallisuutta sivustolle. Kun käyttäjä esimerkiksi napsauttaa hiirellä jotain elementtiä, voidaan siihen sijoittaa `.click()`-tapahtuma, jonka sisään kirjoitetaan funktio toiminnosta. Funktiota `.hide()` voidaan käyttää piilottamaan elementti, kun käyttäjä napsauttaa hiirellä elementtiä, johon on sidottu `.click()`-tapahtuma. Yleisimmin käytössä on `.hover()`-tapahtuma, joka laukaisee kaksi funktiota, ensimmäisen, kun hiiri menee valitun elementin päälle, ja toisen, kun hiiri poistuu elementin päältä. Tätä voi käyttää vaikkapa seeprataulukon rivin korostukseen. Kun käyttäjä vie hiiren rivin päälle, sille annetaan joko uusi taustaväri tai sitten luokka, johon on määritelty uusi taustaväri. Kun hiiri menee seuraavalle riville, toinen funktio palauttaa rivin aikaisemman taustaväriin. Monen kolumnin taulukossa se helpottaa huomattavasti lukemista ja näin parantaa käyttäjäkokemusta. (15, s. 17–49.)

### 2.4.5 Lisäosat

jQuerylle on tehty jopa kymmeniätuhansia lisäosia, ja niitä tehdään koko ajan lisää. Osa lisäosista on tehty huolellisesti, ja niissä on otettu huomioon tuki kaikille selaimille. (15, s. 9–16.)

jQuery-kehitysryhmän kehittämä jQuery UI on luultavasti suosituin lisäosa. Se helpottaa elementtien animointia JavaScriptillä huomattavasti. Se mahdollistaa tiedostojen vedä ja pudota -lataamisen sivustoille, sen avulla voidaan lajitella dataa taulukoissa ja paljon muuta. Nykyisin monella hotellilla ja matkanjärjestäjällä on päivämäärävalitsimet käytössä. Ne ovat lähes poikkeuksetta jQuery UI -lisäosalla tehtyjä. (15, s. 9–16.)

Muita hyvin yleisessä käytössä olevia lisäosia ovat lomakkeiden validointilisäosat. Lomakkeen täyttöä voidaan tarkkailla heti täytön yhteydessä ja antaa heti ohjeita, jos jokin kohta on täytetty väärin tai lomakkeessa on puutteellisia tietoja. Niitä käytettäessä on kuitenkin hyvä pitää mielessä, että tämä ei missään nimessä riitä lomakkeen lopulliseksi tarkistukseksi. Kun selaimesta asetetaan JavaScript pois päältä, lomake jää täysin ilman validointia, ja tästä voi aiheutua suuria tietoturvariskejä. (15, s. 9–16.)

Vaikka nykyään internetnopeudet ovat hyvin suuria, täytyy kuitenkin pitää mielessä, että ladattaessa useita lisäosia käyttöön voivat sivuston latausajat kasvaa ärsyttävän pitkiksi. Hieno visuaalinen ulkoasu ja interaktiivisuus eivät kuitenkaan voita käytettävyyttä ja nopeita latausaikoja. Toinen huono puoli on, että Microsoftin vanhemmat versiot Internet Explorerista ovat erittäin hitaita suorittamaan JavaScript-koodia. (15, s. 9–16.)

### 2.4.6 JavaScript-kirjastot

JavaScriptille on rakennettu hyvin monia kirjastoja, ja jQueryn jälkeen niistä toiseksi suosituin on Modernizr. Sitä käytetään tunnistamaan selaimen ominaisuudet. Sitä voidaan hyödyntää tarjoamaan uudemmille selaimille natiivit toiminnallisuudet, ja hieman vanhemmille voidaan tarjota joko vaihtoehtoinen ratkaisu tai poistaa toiminnallisuus käytöstä. (16.)

Socket.io on JavaScript-kirjasto, jonka tavoitteena on tuoda reaaliaikaisuus mahdolliseksi jokaisella selaimella ja mobiililaitteella. Se yhdistää monta eri tekniikkaa, joita



käytetään reaaliaikaisuuden tuomiseksi verkkosovelluksiin. Sen toimintaperiaatteena on käyttää hyväksi parasta mahdollista tapaa reaaliaikaisuuden luomiseksi ja tarjota se mahdollisimman helppokäyttöisenä ohjelmoijalle. (17.)

Zepto on minimaalinen JavaScript-kirjasto nykyaikaisille selaimille, jossa on jQuery-yhteensopiva rajapinta. Sen tarkoitus ei ole korvata jQuerya, vaan tarjota vaihtoehto jQueryyn suhteellisen suurelle koolle. Sen suunnittelussa on keskitytty nopeaan lataukseen ja suoritukseen. (18.)

## 2.5 LAMP, MAMP ja WAMP

LAMP on lyhenne sanoista Linux, Apache, MySQL ja PHP. MAMP on lyhenne sanoista Mac, Apache, MySQL ja PHP, ja WAMP on lyhenne sanoista Windows, Apache, MySQL ja PHP. Apache, MySQL ja PHP on yleisin käytössä oleva web-palvelimen konfiguraatio. Ne ovat kaikki vapaan lähdekoodin ohjelmistoja. (19, s. 13–14.)

Apache on palvelinohjelmisto, joka muun muassa toimittaa kaikki tiedostot selaimelle. Jokainen tiedosto, joka on kuvattu HTML-merkintäkielellä, pyydetään Apache-palvelimelta. Jos se löytyy palvelimelta, Apache tarjoaa sen käyttöön. Tiedostojen ei kuitenkaan ole pakko olla staattisia, vaan ne voidaan generoida esimerkiksi PHP:llä (PHP: Hypertext Preprocessor). Apachelle on myös lukuisia moduuleja, joista yleisimmät ovat URL:n (Uniform Resource Locator) uudelleenkirjoitus- ja välityspalvelinmoduuli. Uudelleenkirjoitusmoduulilla voidaan muuttaa internetsivustojen osoitteet käyttäjystävällisempään muotoon, esimerkiksi muuttamalla <http://leffaputki.fi/index.php> osoitteeseen <http://www.leffaputki.fi/>. Välityspalvelinmoduulilla voidaan tarjota käyttäjille sivuja suoraan välimuistista ja vähentää palvelimen kuormitusta. Lisäksi Apachelle on kirjoitettu huomattava määrä palvelimen turvallisuutta lisäävää moduulia. (19, s. 9–10.)

PHP on palvelinpuolen ohjelmointikieli, joka prosessoidaan palvelimella ja tarjotaan käyttäjälle HTML-tiedostona. PHP:llä lisätään staattisiin HTML-sivuihin dynaamisuutta. PHP-parseria kutsutaan avainsanalla `<?php` ja kutsu lopetetaan avainsanalla `>`. Niiden välissä oleva koodi suoritetaan, ennen kuin tiedosto lähetetään käyttäjälle. PHP:llä voidaan tehdä hyvin monimutkaisia asioita web-palvelimella: pystytään generoimaan tiedostoja, validoimaan esimerkiksi lomakkeita, lisäämään lomakkeen tiedot tietokantaan ja paljon muuta. PHP tarjoaa käyttäjille myös tuhansia funktioita, joilla voidaan

esimerkiksi tulostaa HTML-tiedostoon palvelimen tämänhetkinen päivämäärä. PHP-muuttujille ei ole pakko antaa datatyyppiä, ja muuttujat merkitään dollari-symbolilla. Muuttujat ovat merkkikokoriippuvaisia. PHP tarjoaa muun muassa operaattorit, lausekkeet, ehtolauseet, silmukat, funktiot, objektit ja taulukot ohjelmoijalle. Hyviin tapoihin kuuluu, että PHP-koodi kommentoidaan huolellisesti ja jäsennetään selkeälukuisiksi. Se auttaa muistamaan koodin tarkoituksen, jos ja kun siihen joskus palataan. (19, s. 5–6, 41.)

Usein PHP:llä kirjoitetaan logiikka, jota tarvitaan verkkosivustojen käyttöön. PHP:tä suositellaan kirjoitettavan olio-ohjelmointiin perustuvien sääntöjen mukaisesti. Kirjoitetaan luokkia, joilla on oma logiikka ohjelmistossa. Se tekee ohjelmiston laajentamisesta helpompaa ja selkeää. PHP-laajennosta PDO (PHP Data Objects) voidaan pitää esimerkkinä tästä. Se on luokka, joka on rakennettu keskustelemaan tietokantojen kanssa. Koodiesimerkissä 2 otetaan tietokantayhteys PDO-luokalla MySQL-tietokantaan ja tallennetaan se muuttujaan `"$db_connect"`. Luokalle annetaan parametreinä tietokannan osoite, tietokanta, käyttäjänimi ja salasana. Tämän jälkeen PDO-luokassa olevilla metodeilla voidaan tehdä kyselyitä MySQL-tietokantaan ja saada niihin vastaus. (20, s. 1–4, 44–45.)

```
$db_connect = new PDO('mysql:host=localhost;dbname=tietokanta',
    'kayttaja', 'salasana');
```

Koodiesimerkki 2. Tietokantayhteyden muodostus.

MySQL on relaatiotietokantaohjelmisto, joka keskustelee tietokantojen kanssa. Tietokannat ja tiedon tallennus ovat jokaisen dynaamisen verkkosivuston perusta. MySQL-tietokantaan kuuluu yksi tai useampia tauluja, jotka sisältävät tietoa eli rivejä. Rivit sisältävät yhden tai useamman sarakkeen. Tieto tallennetaan tietokantaan, koska sitä on sen jälkeen helppo hakea, lisätä ja päivittää. Lisäksi eri käyttäjille voidaan näyttää eri sisältöä ja antaa erilaisia oikeuksia verkkosivuston käyttöön. Julkaisujärjestelmissä käytetään hyvin laajasti hyödyksi tietokantoja. Se mahdollistaa sisällön tuottamisen suoraan internetsivustolla, ja sen jälkeen se julkaistaan muiden luettavaksi. (20, s. 161, 39–40.)

Linux on Unixin suosituin alaluokka ja yleisin käyttöjärjestelmä web-palvelimissa, ja yleisesti palvelin koostuu sen lisäksi Apachesta, PHP:stä ja MySQL-tietokannasta.

Käyttäjä kirjoittaa internetselaimessa osoitekenttään www-osoitteen, josta lähtee pyyntö DNS-palvelimelle (Domain Name Service). DNS-palvelin vastaa pyyntöön web-palvelimen IP-osoitteella (Internet Protocol), jonne selain lähettää pyynnön internet-sivusta. Apache ottaa pyynnön vastaan ja selvittää, mitä sisältöä käyttäjälle tarjotaan. Sisältö yleisesti sisältää PHP-koodia ja se parsitaan. PHP ottaa yhteyden MySQL-tietokantaan ja tekee haun sisällöstä. MySQL palauttaa PHP:lle tuloksen hausta, ja PHP palauttaa HTML-tiedoston Apachelle, joka tarjoaa sen selaimelle. Tämän jälkeen selain tulkitsee HTML-tiedoston ja lähettää pyynnön resurssitiedoille, kuten CSS, JavaScript- ja kuvatiedoille. Apache etsii tiedostot ja lähettää ne selaimelle, jossa ne parsitaan internetsivuksi. (19, s. 1–5; 21.)

## 2.6 Palvelinpuolen JavaScript

Suurimmalla osalla verkkosivustoista on sekä asiakaspuoli että palvelinpuoli. Muutamia vuosia sitten ei tullut mieleenkään kirjoittaa palvelinpuolta JavaScriptillä, vaan se kirjoitettiin yleensä staattisimmilla kielillä. Tämä aiheutti sen, että web-kehittäjä joutui yleensä kirjoittamaan koodia kahdella tai useammalla kielellä. (22, s. 14.)

Google Chromen avoimen lähdekoodin V8 JavaScript -moottori paransi JavaScriptin suorituskykyä ja muistinhallintaa huomattavasti ja tarjoaa kattavan rajapinnan sen käyttöön. Ryan Dahl upotti V8:n asynkronisella rajapinnalla palvelimen käyttöjärjestelmän integrointikerrokseen. Tästä syntyi Node.js. Web-kehittäjät pystyivät tämän jälkeen käyttämään JavaScriptiä palvelin- ja asiakaspuolella. (22, s. 14.)

Node.js:n voi asentaa kaikille yleisimmille käyttöjärjestelmille. Windowsille ja OS X:lle Node.js:stä on tehty asennettava tiedosto. Linuxin eri versioille sen voi asentaa APT (Advanced Package Tool) -paketinhallintaohjelmiston kautta. Node.js tulkitsee tiedostoja käskyllä "node polku/tiedostoon.js". Koodiesimerkissä 3 kirjoitetaan yksinkertainen Node.js-http-palvelin ja tarjoillaan dynaamisesti luotu HTML-sivusto.

```
var http = require('http');
var palvelin = http.createServer(function (kutsu, vastaus) {
    vastaus.writeHead(200, { 'Content-type' : 'text/html' });
    vastaus.end('<h1>Hei maailma!</h1>');
```

```
});  
palvelin.listen(3000);
```

Koodiesimerkki 3. Node.js-palvelin.

Node kuuntelee kutsua porttiin 3000. Kun kutsu tulee, Node.js tekee HTTP-palvelimen (Hypertext Transfer Protocol) ja lähettää ylätunnisteena tietotyyppin "text/html" ja tulostaa h1-elementtiin "Hei maailma!". Muutamalla rivillä koodia tehtiin HTTP-palvelin Node.js:n avulla. (22, s. 10.)

Node.js:n tarkoitus on mahdollistaa helppo tapa rakentaa skaalattavia verkkoja. Node.js:n ensimmäinen versio kehitettiin vuonna 2009, ja se on sen jälkeen kasvanut suurimmaksi palvelinpuolen JavaScript-ohjelmistoalustaksi. Node.js:n HTTP-palvelimen protokolla on asynkroninen ja perustuu callback-metodeihin. Suurin osa HTTP-palvelimista käyttää yhtä säiettä asiakkaan ja palvelimen välillä. Asiakas lähettää kutsun ja yleensä sen jälkeen odotetaan, kunnes kutsuun saadaan vastaus. Node.js ei jää odottamaan kutsuun vastausta, vaan se on silmukkapohjainen tapahtuma ja hyödyntää estotonta siirräntää. Tämän avulla on helppo mahdollistaa monia rinnakkaisia yhteyksiä, ja samalla se sallii datan suoratoiston. Tämä mahdollistaa reaaliaikaisten internet-ohjelmien kehityksen. (22, s. 16; 23.)

Node.js:ään kuuluu paketinhallintajärjestelmä npm. Yleinen virhe on kirjoittaa se isoin kirjaimin, mutta itse asiassa se ei ole lyhenne sanoista "node package manager", vaikka niin luullaankin. npm pohjautuu bash-apuohjelmaan "pm", joka on lyhennys "pkmakeinst" bash -funktioista. Komennolla asennetaan ohjelmia eri alustoille. Noden paketinhallintajärjestelmän nimi, npm, valittiin sillä perusteella, että oikeakätisen on helppo kirjoittaa se QWERTY-näppäimistöllä, ja nimen kirjoittamisen jälkeen oikean käden nimetön on merkin "-" kohdalla, jolla asetetaan argumentteja npm:lle. (24.)

npm:llä on helppo hallita moduuleja projekteissa. Se lataa paketit, selvittää riippuvuudet pakettien kesken, testaa ja asentaa komentoriviapuohjelmia. Pakettien lisääminen Node.js-projektiin ei ole pakollista, mutta valmiiden moduulien hyödyntäminen nopeuttaa työskentelyä huomattavasti. (22, s. 10.)

npm on vapaan lähdekoodin ohjelmisto ja samoin ovat useimmat moduulit, jotka löytyvät npm-arkistosta. Sitä pääsee selaamaan osoitteesta <https://npmjs.org/>. Moduuleja

asennetaan komentorivillä komennolla `npm install "moduulin nimi"`. Lisäämällä viittaus -  
g asennetaan moduuli globaalisti, jolloin moduuli on käytettävissä kaikissa projekteissa.  
Node.js-moduulit ja niiden riippuvuudet asentuvat kansioon `node_modules`. (22, s. 11–  
14.)

Omat Node.js-moduulit kuvaillaan `package.json`-tiedostoon. `Package.json`-tiedosto pitää olla validia JSON:a (JavaScript Object Notation). JSON on ohjelmointikieliriippu-  
maton tekstiformaatti, jota ihmisten ja koneiden on helppo lukea ja parsia. `Package.json`-tiedostoon kirjataan moduulin nimi, versio, ja jos moduulilla on riippuvuuksia  
muihin moduuleihin, ne ja niiden versiot kirjataan. `npm install`-komento lukee `package.json`-tiedoston ja asentaa tarvittavat riippuvuudet ja tiedostot. Sen vuoksi ei ole tar-  
peellista lähettää `node_modules`-kansiota muille moduulia käyttäville. Riippuvuuksien  
versionumerointi pitää huolta siitä, että jos moduulia on päivitetty ja sen rajapintaan on  
tullut muutoksia, kirjoitettu moduuli ei hajoa. `npm publish`-komennolla voidaan julkaista  
moduuli `npm`-arkistoon. (22, s. 11–13.)

Noden kanssa pitää olla tarkkana, jos muuttujia muutetaan callback-funktioissa. Toisin  
kuin esimerkiksi PHP:ssä, node ei käy läpi koko palvelinpuolen skriptiä, vaan se odot-  
taa kutsua ja vastaa siihen. Esimerkiksi kun määritellään muuttuja `x` ja annetaan sille  
arvoksi 1: Tehdään funktio, joka palauttaa `x:n` arvon osoitteessa `/testi`, ja samassa  
funktiossa asetetaan `x:n` arvoksi `undefined`. Kirjoitetaan sama funktio PHP:llä ja lada-  
taan ensin node-versio sivusta `/testi` kahdesti, ja sen jälkeen PHP:llä tehdään sama.  
PHP:llä tulostus on molemmilla kerroilla sama, koska koko skripti ladataan uudelleen,  
mutta nodella ensimmäinen kerta palauttaa 1:sen ja toinen kerta `undefined`. (22, s. 28–  
29.)

Node on myös estoton. Koodiesimerkissä 4 on kirjoitettu lähes sama ohjelma No-  
de.js:llä ja PHP:llä.

```
// Node.js
console.log('Hei');
setTimeout(function(){
  console.log('maailma.');
```

```
}, 5000);
console.log(' Näkemiin.');
```

```
// PHP
```

```
print('Hei');
sleep(5);
print('maailma. ');
print('Näkemiin.');
```

Koodiesimerkki 4. Node.js:n ja PHP:n vertailua.

Syntaksissa on eroa, mutta tämä esimerkki kuvastaa myös noden ja PHP:n estotonta ja estollista koodia. PHP `sleep()` -funktio estää koodin suorittamisen, ja mitään ei tapahdu viiteen sekuntiin. Nodessa käytetään tapahtumasilmukkaa, ja se on estoton. PHP:llä tulostus on ”Hei maailma. Näkemiin”, kun taas Node.js-koodista tulostuu ”Hei Näkemiin. maailma.”. Node.js rekisteröi tapahtumat ja ajaa päättymättömänä silmukkana kyselyitä ytimeen. Jos tapahtumat ovat valmiita suoritukseen, ajetaan niiden callback-funktiot. PHP on siis synkroninen ja Node.js asynkroninen. (22, s. 29–31.)

Node.js käyttää yhtä säiettä koodin suorittamiseen. Jos esimerkiksi Node.js:llä suoritetaan erittäin raskas laskutoimitus tai silmukka, samaan aikaan ei voida tehdä mitään muuta. Node.js:ssä ei myöskään ole todellista rinnakkaisuutta, mutta sitä ei oikeastaan tarvita, koska Node.js:ssä käytetään kutsupinoja ja sen parsimiseen käytettävä Googlen JavaScript parsin v8 on todella nopea käymään niitä läpi. Kun lisätään estoton koodin suorittaminen, varmistetaan, että koodin suorittaminen ei tukkeudu. (22, s. 31–33.)

Virheen tapahtuessa Node.js-prosessi pysäytetään kokonaan, jos sille ei ole kirjoitettu virheenkäsittelijää. Tämä tehdään sen takia, koska Node.js ei voi tietää, onko ohjelmaa mahdollista ajaa virheestä huolimatta, ja jos koodin suorittamista jatkettaisiin, se tekisi koodin testaamisen hyvin vaikeaksi.

## 2.7 HTML-kirjastot ja responsiiviset verkkosivustot

Kaksi suosituinta HTML-ohjelmistokehystä ovat Bootstrap ja Foundation. Foundationin on kehittänyt Zurb-säätiö ja Bootstrapin on kehittänyt Twitter. Kummatkin on julkaistu vuonna 2010, ja niistä on kehittynyt sen jälkeen pohja lukuisille internetsivuille. Kumpikin ohjelmistokehys tarjoaa erittäin hyvän dokumentaation, ja ne on molemmat helppo ottaa käyttöön. (25; 26.)

Uusimmat versiot kummastakin ohjelmistokehiksestä hyödyntävät Node.js:lle tehtyä moduulia Bower. Bower on JavaScriptillä kirjoitettu paketinhallintamoduuli, joka mahdollistaa helpon hallittavuuden JavaScript-kirjastoille. Se lataa paketit hyödyntäen Git-versionhallintaa. Uusin versio Foundationista käyttää tämän lisäksi Node.js-moduulia Grunt. Se on JavaScript-tehtävän suorittaja. Sillä voidaan automatisoida yleisiä tehtäviä, kuten JavaScript-tiedostojen kokoaminen yhteen tiedostoon ja tiedoston minimointi. Se myös mahdollistaa todella nopean SASS-prosessoinnin Libsass-kirjastolla. (27; 28.)

Foundation käytti versioon 4 asti Ruby-pohjaista kehitysalustaa. Monien eri Foundation-versioiden ylläpitoon käytettiin Bundleria, joka tallentaa lokaalisti kehittäjän tietokoneelle jokaisen eri Foundationin version SASS-tiedostot. Bootstrapin SASS-versio käyttää edelleen tätä ratkaisua.

Bootstrap ja Foundation eroavat toisistaan melko paljon. Esimerkiksi Bootstrap tarjoaa melkein kaikki elementit, mitä voi kuvitella, valmiiksi suunniteltuna. Zurb Foundation tarjoaa vain kourallisen valmiiksi tyyliteltyjä elementtejä. Lisäksi Foundation käyttää EM-mittayksikköä, kun taas Bootstrap käyttää pikseleitä.





























Bootstrapistä on mahdollista käyttää LESS-, SASS- ja CSS-versioita. Foundationista voidaan käyttää SASS- tai CSS-versioita. Kumpikin ohjelmistokehys vaatii toimiakseen jQuery-kirjaston.

Foundation käyttää joustavaa ruudukkoa, jonka jako voidaan määritellä Foundationin SASS-asetustiedostossa. Foundationin peruseriaate on, että verkkosivuston suunnittelu aloitetaan pienimmästä ruudunkoosta suurimpaan. Ruudukko on jaettu viiteen ennalta määräämättömään pysäytyspisteeseen, joissa vaihdetaan sivuston ulkoasua. Näistä yleisesti kuitenkin käytetään kolmea. Perusasetuksilla ne ovat 0–640 px, 641–1024 px ja 1025 px ja suurempi. (29.)


Bootstrapiin lisättiin tuki joustavalle ruudukolle vasta myöhemmässä vaiheessa, ja aikaisemmin sitä käytettiin rakentamalla sivusto suurille näytöille ja sen jälkeen sovittelemalla sivusto pienemmille. Tämä saattaa aiheuttaa ongelmia sovitettaessa elementtejä mobiililaitteille. Uusimmassa versiossa Bootstrap-ohjelmistokehystä on siirrytty mobiili ensin -ajatusmalliin. (25; 29.)

## 2.8 Julkaisujärjestelmät

Julkaisujärjestelmä on hallintatyökalu, jolla pidetään internetsivulla julkaistava materiaali hallinnassa. Se voi olla oikeastaan mitä tahansa materiaalia, kuten videoita, ääntä, kuvia, tekstiä tai muita dokumentteja. Julkaisujärjestelmät on yleensä suunniteltu niin, että niiden käyttö ei vaadi teknisiä taitoja. Drupal, Joomla ja Wordpress ovat tällä hetkellä kolme suurinta julkaisujärjestelmää. Ne on jokainen rakennettu hyödyntäen PHP:tä ja MySQL:ää. Kuvassa 2 on vertailtu Wordpressin, Drupalin ja Joomlaan eri ominaisuuksia, niiden suosiota ja markkinaosuuksia. Wordpress on ylivoimaisesti suosituin, helpoin ottaa käyttöön ja parhaiten hakukoneille optimoitu julkaisujärjestelmä. (30; 31; 32.)

	 Wordpress	 Drupal	 Joomla
<b>ESTABLISHED</b>	2003	2001	2005
<b>ABOUT</b>	Was initially designed for blogging, but from its conception it grew to be the top CMS.	Is the oldest CMS out of the bunch. Very powerful system that can be used for large & complex websites.	A CMS commonly classed inbetween the two; not as pretty & user friendly as Wordpress nor as powerful as Drupal.
<b>MARKET SHARE</b>	 53.8%	 6.7%	 9.2%
<b>PROGRAMMING LANGUAGE</b>	 php	 php	 php
<b>LAYOUT DESIGN</b>	Themes 	Themes 	Templates 
<b>SETUP EASE OF USE</b>	Easy  Hard	Easy  Hard	Easy  Hard
<b>POPULAR BRANDS THAT USE CMS</b>	 ebay  CNN THE WALL STREET JOURNAL.	 Oxfam  The Economist  McDonalds	 BARNES & NOBLE  PUFFIN
<b>SEO</b>	By default has good SEO options. Great open source plugins available. Gives you more flexibility and more power.	Search friendly out of the box. With the right combination of SEO planning and choice of modules it can yield successful results.	Lacks out of the box SEO features. Requires SEO components for additionally control for SEO settings. Recommend SEO extensions cost money.
<b>FACEBOOK LIKES</b>	 537,160	 46,687	 69,947
<b>Pitfalls</b>	The core system updates frequently, but this can as result break your existing plugins 	Not the most user friendly CMS to use. Needs a Drupal developer for complex sites. 	Not as pretty or easy as Wordpress. Modules are hard to maintain. The admin interface can be quite confusing. 

Sources:  
[cms.about.com](http://cms.about.com)  
[drupal.org](http://drupal.org)  
[wordpress.org](http://wordpress.org)  
[joomla.org](http://joomla.org)

 MEDIA HEROES

Kuva 2. Wordpressin, Drupalin ja Joomlaan vertailua (33).



Drupal on vanhin sisällönhallintajärjestelmä näistä kolmesta ja siihen on myös vaikein oppia. Drupal on suunniteltu yrityskäyttöön, ja se selviää todella suurista määristä sisältöä helposti, mutta sen takia suurin osa moduuleista ja teemoista on maksullisia. Modulaarisuuden takia Drupalilla on kuitenkin mahdollista tehdä oikeastaan mitä tahansa. Drupal vaatii monien tuntien perehtymistä, ja se on kylmiltään hyvin vaikea oppia. (34.)

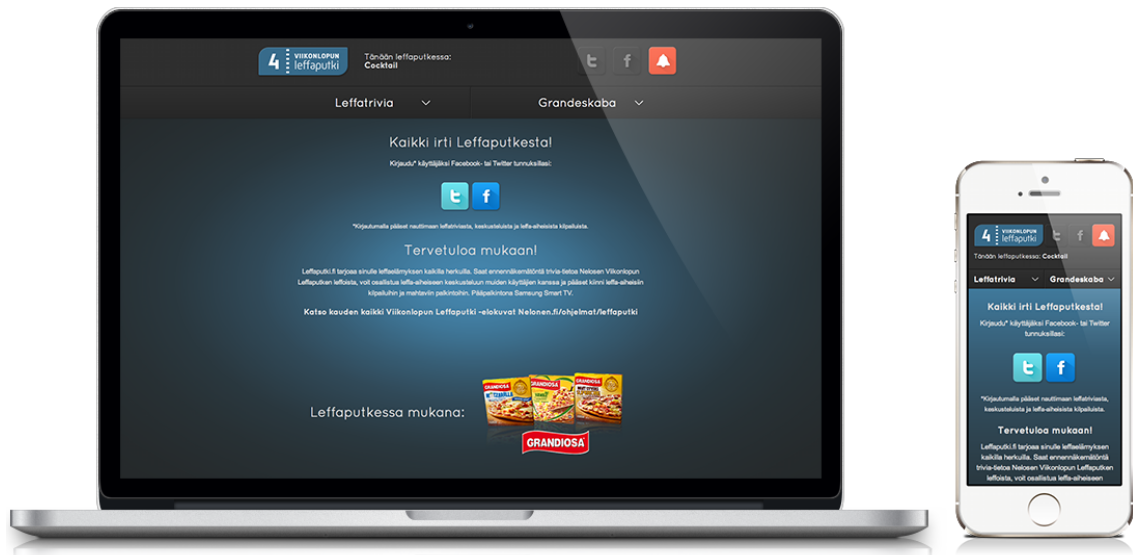
Joomla ei ole niin vaikea oppimisen kannalta kuin Drupal, mutta jos on aloitteleva ohjelmoija, se aiheuttaa vaikeuksia. Joomlailla on laaja kehittäjäyhteisö, ja apua ongelmiin löytyy helposti. Moduuleja ja teemoja on saatavilla ilmaiseksi, ja niitä on moniin tarkoituksiin. Hakukonenäkyvydessä Joomla on heikolla pohjalla. (34.)

Wordpress on näistä sisällönhallintajärjestelmistä helppokäyttöisin niin loppuasiakkaalle kuin kehittäjällekkin. Teeman vaihtaminen on helppoa, ja niitä on tuhansia ilmaiseksi internetissä. Sama pätee lisäosiin, ja niitä on melkein mihin tarkoitukseen tahansa. Wordpress suunniteltiin alun perin blogialustaksi, mutta sivustojen ei ole enää pakko näyttää blogeilta, vaan ne voivat sisältää staattisia sivuja ja sen lisäksi mahdollisuuden blogityyppisiin sivuihin. Wordpressillä on valtava kehittäjäyhteisö ja päivityssykli on hyvin nopea. (32.)

### **3 Interaktiivisen verkkosivuston kehittäminen**

#### **3.1 Wordpress-teeman kehitys**

Tässä työssä käytin Wordpress-julkaisujärjestelmän teeman pohjana Zurbin kehittämää Foundation 4 -ohjelmistokehystä. Foundation 4 on suunniteltu käytettäväksi pienimmästä ruudunkoosta suurempaan. Kuvassa 3 on näkyvillä teeman etusivu pienellä ja suurella näytönkoollla. Todellisessa maailmassa ainakin minun kohdallani tämä tuottaa usein vaikeuksia, koska yleensä kun saan projektin työn alle, sivuston ulkoasu on tehty vain suurimmalle näyttökoolle. Foundation 4 -ohjelmistokehykseen kuuluu mukautuva ruudukko, kolmitasoinen navigaatio, räätälöitävät lomakkeet, modaalit ja paljon muuta. Foundationin viidettä versiota ei ollut vielä julkaistu sivuston kehittämishetkellä. (26.)



Kuva 3. Leffaputki.fi-sivuston etusivu pienellä ja suurella näytönkoolla.

Tärkeintä sivustoissa, joille odotetaan paljon mobiilikäyttäjiä on saada sivun latautumis aika mahdollisimman lyhyeksi, joten valitsin Foundationin SASS-version. Se mahdollistaa suuremman hallittavuuden ohjelmistokehyksestä. Settings.scss-tiedostosta voidaan valita, mitä CSS-elementtejä halutaan ottaa generoituvaan CSS-tiedostoon mukaan. JavaScript-tiedostoista on myös mahdollista valita käyttöön vain tarvittavat.

Pakollisina JavaScript-tiedostoina Foundationissa ladataan Modernizr, jota käytetään tarjoamaan HTML5-elementtien tuki vanhoille selaimille ja tunnistamaan selainten tukea eri ominaisuuksille. Foundation versio 4 tarvitsee toimiakseen joko Zepton tai jQueryn. Päädyin jQueryyn, koska se on minulle tutumpi ja suurin osa Wordpressin lisäosista vaatii sen käytettäväksi. Zepto ja jQuery ei voi käyttää rinnakkain. Sen lisäksi pitää ladata Foundationin ydin-JavaScript-tiedosto. Kaikki muut Foundationin JavaScript-tiedostot ovat lisäosia, eikä niitä ole pakko ladata, jos niitä ei käytetä. Tähän projektiin ei tarvittu muita lisäosia.

Foundation versio 4.3.1 käyttää CSS:n normalisointina normalize.css v2.1.1. Normalisointi-CSS-tiedostoa käytetään normalisoimaan eri selainten elementtien samankaltaisuus. Sen lisäksi pakollisia SASS-komponentteja ovat globaalit Foundationin funktiot ja Foundationin SASS-asetustiedosto. Käyttöön otettiin lisäksi ruudukko helpottamaan sivun asettelua eri näyttökoille, näkyvyys-luokat, typografian tyylit, nappityylit ja lomaketyylit.

Wordpressin teema koostuu useista teematiedostoista, joista jokainen vastaa osaa teemasta. Melkein aina staattisia osia teemasta ovat ylä- ja alatunniste ja sivupalkki. Niitäkin on mahdollista tehdä teeman eri osille erilaisina. Ylätunnisteeseen laitetaan yleensä staattisen HTML:n lisäksi navigaatio. Sivupalkkiin tulevat näkyviin Wordpressin hallintapaneelissa määritellyt pienoisojelmot. Alatunnisteessa suljetaan staattiset HTML-elementit ja ladataan JavaScriptit.

Wordpressin silmukassa haetaan sisältö hallintapaneelissa määritetylle sivulle tai artikkelille, ja se esitetään teeman kehittäjän näkemyksen mukaan. Silmukkaan haettu tieto voi koostua yhdestä tai useammasta artikkelista, tai jokaiselle artikkelille voidaan tehdä oma silmukka, jossa määritellään, miten data näytetään. Tästä huolimatta Wordpress-teemaa kehitettäessä on vain kolme pakollista tiedostoa. Yksi on style.css, jossa määritellään teeman nimi ja muita siihen liittyviä tietoja ja tyylejä. Toinen on functions.php-tiedosto, johon voidaan kirjoittaa teemassa käytettäviä yleisiä funktioita ja ottaa käyttöön tai poistaa Wordpressin ominaisuuksia. Sen lisäksi tarvitaan index.php, johon haetaan Wordpressiin määritelty sisältö. Tämän lisäksi on mahdollista määrittää comments.php, jossa määritetään, miltä kommentointilomake ja kommentit näyttävät.

Projektiin ei ollut tarvetta tehdä kuin comments.php, front-page.php, functions.php, header.php, footer.php ja style.css. Wordpressissä voi valita etusivun teemaksi joko index.php-, home.php- tai jos käytetään staattista sivua, niin front-page.php-tiedoston.

Sivulle valittiin kaksi hyvin suosittua lisäosaa helpottamaan ja nopeuttamaan projektin toteutusta: Gravity Formsilla on mahdollista tehdä lomakkeita hyvin helposti Wordpress sivustolle. Siinä on hyvä graafinen käyttöliittymä, lomakkeiden lähettäminen on kattavasti testattu MySQL-injektioita vastaan ja sitä on kehittäjän kannalta helppo modifioida omaan käyttöön. Kirjoitin Gravity Forms -rajapinnan avulla lomakkeiden tulostuksen vastaamaan Foundationin lomakkeiden tyyliä, jolloin lomakkeiden ulkoasun muokkauksesta tuli huomattavan helppoa. Se tapahtui kirjoittamalla funktio, jota käytetään Gravity Formsin tavallisen HTML-tulostuksen sijaan. Toinen erittäin hyvä lisäosa, joka otettiin käyttöön, on Advanced Custom Fields. Sillä on mahdollista tehdä omia sisältökenttiä ylläpitopuolen artikkeleille ja sivuille. Advanced Custom Fields-lisäosaan asennettiin Repeater Field -lisäosa, joka mahdollistaa aikaisemmin määritettyjen kenttien monistuksen.

Sivulle myös modifioitiin oma versio Ultimate Preloader -nimisestä lisäosasta, jotta latautuvaa sivua ei näytetä. Samalla päätettiin, että ladataan kaikki sivuston sisältö yhdellä kertaa, minkä jälkeen sisältöä ei tarvitse enää kuin piilottaa ja näyttää. Tämä päätös tehtiin myös sen takia, että saadaan pysyvä yhteys sivun ja palvelimen välille, mikä mahdollistaa sen, että annetaan äänimerkki, kun sivulla tulee viesti.

### 3.2 Palvelinpuolen JavaScriptin integroiminen Wordpressiin

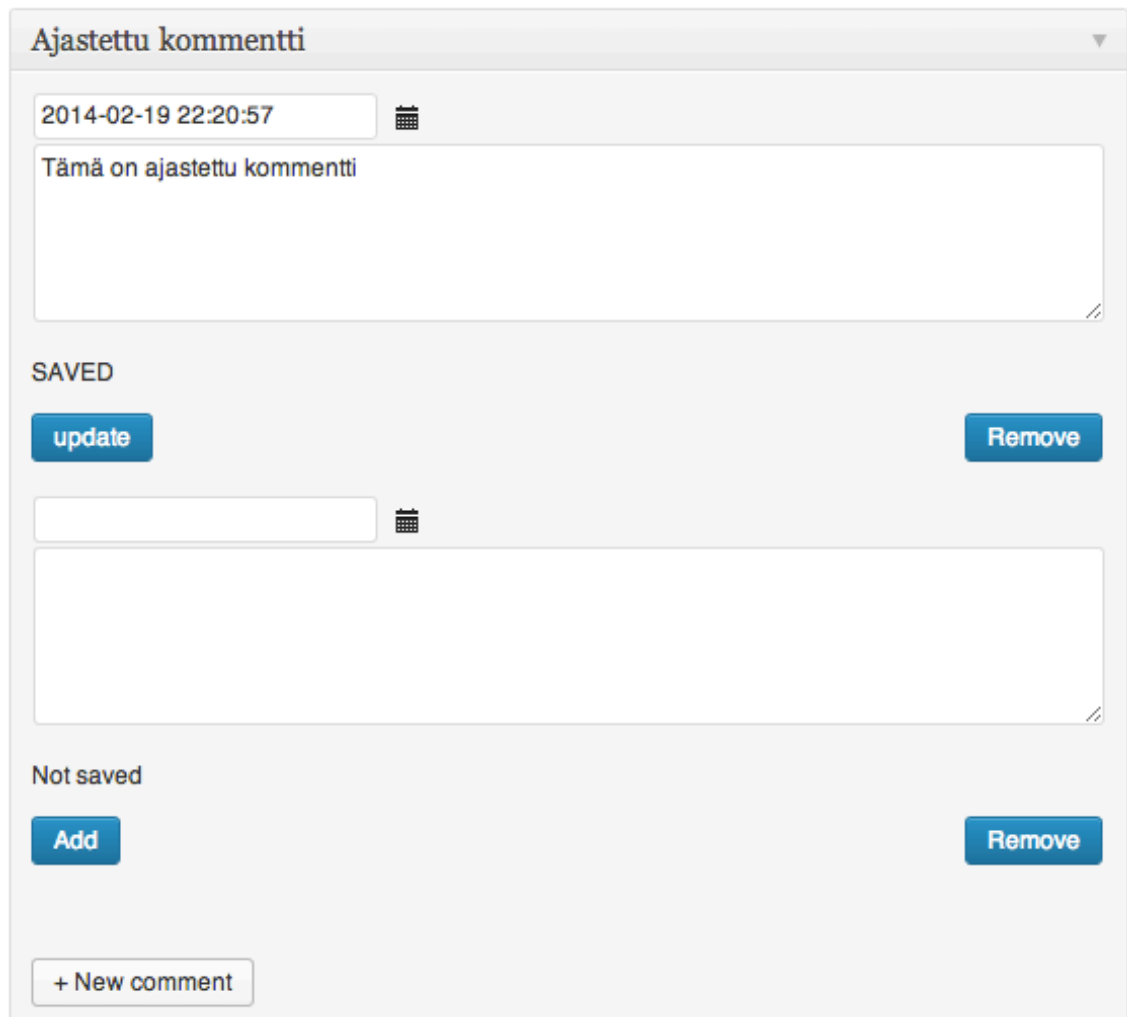
Verkkosivustolla on käytössä ajastetut viestit ylläpitokäyttäjältä. Tämä on toteutettu hyödyntämällä myös Wordpressin hallintapuolella Node.js:lle tehtyä Socket.IO-moduulia, joka mahdollistaa selainriippumattoman reaaliaikaisen yhteyden palvelimeen. Socket.IO käyttää ensisijaisesti HTML5 -rajapinnan WebSocket-yhteyttä, mutta siihen on myös paketoitu Flashin päällä toimivat WebSocket-yhteydet, long polling -menetelmää hyödyntävä jatkuva palvelinyhteys ja moniosainen XMLHttpRequest-yhdistettynä iframe-yhteyksiin. Nämä kaikki yhdistettynä taataan laaja selaintuki. (35.)

Käyttäjä varmennetaan Wordpressin evästeiden ja Node wordpress-auth-moduulin avulla. Sillä varmistetaan, että ajastettu viesti on varmasti ylläpitokäyttäjältä. Samalla viesti tallennetaan Wordpressin tietokantaan, sivun kommentti -tauluun. Wordpressin hallintasivulle luotiin meta-kenttä, johon voidaan syöttää kellonaika ja kommentti. Lisäksi kommentteja on mahdollista päivittää, lisätä tai poistaa.


Meta-kenttää rakennettaessa käytettiin Wordpressin koodityylin mukaisia menetelmiä. `add_meta_boxes`-toimintoon kytketään funktio, jossa kutsutaan `add_meta_box`-funktioita. `add_meta_box`-funktioon annetaan parametreiksi "id", "otsikko", "callback-funktio" ja "sivutyyppe". Callback-funktioon on kirjoitettu kentän näyttöön tarvittava HTML-, CSS- ja JavaScript-koodi.

Kommentin syöttämisessä käytetään tietokantaan tallennettaessa Wordpressin tavallista Ajax-toiminnallisuutta. `add_action`-funktioilla kytketään `wp_ajax`-funktioon callback-funktio. Ajax-funktioita on Wordpressissä kahdentyyppisiä: `wp_ajax_nopriv`-funktioihin pääsevät käsiksi kaikki käyttäjät ja `wp_ajax`-funktioihin vain kirjautuneet käyttäjät, joilla on ylläpitokäyttöoikeudet. On kuitenkin tärkeää generoida Wordpressin `wp_nonce_field`-funktioilla merkkijono, joka tarkistetaan callback-funktiossa. Se lisää Wordpressin Ajax-kutsujen turvallisuutta.

Leffaputki.fi-sivustolla on käytössä eri funktio ajastetun kommentin lisäämiselle, päivittämiselle ja poistamiselle. Kuvassa 4 on ajastetun kommentin ulkoasu. Siitä saa myös hyvän kuvan, miten syöttö tehdään. Viesti lisätään lisäämällä kommentti Wordpressin tarjoamilla funktioilla Wordpressin tietokantaan. Tietokantaan lisäämisen jälkeen Socket.IO:n avulla kommentti lähetetään palvelinpuolen JavaScriptille, missä ensin varmennetaan käyttäjä Wordpressin ylläpitokäyttäjäksi. Sen jälkeen se lisätään objektiin, joka lähettää viestin käyttäjille node-schedule-moduulin avulla valittuun aikaan.




**Ajastettu kommentti**

2014-02-19 22:20:57 

Tämä on ajastettu kommentti

**SAVED**



**Not saved**

Kuva 4. Wordpress-hallintapaneelin kenttä ajastetuille kommenteille.

Kommentit lisätään objektiin Wordpressin tietokannassa olevan kommentin tunnisteen eli ID:n mukaan. Sillä varmistetaan, että viestejä on mahdollista päivittää ja poistaa ajastuksen jälkeen.

Ylläpitokäyttäjien lähettäessä viestejä käyttäjille toistetaan sivustolla HTML5-audio-rajapinnan avulla ilmoitusääni, ja samalla kommenttikenttä välähtää vaaleampana. Viestin yhteyteen ei lisätty sivun tai kommenttikentän vierittämistä yläreunaan, sillä jos käyttäjä haluaa lukea aikaisempia viestejä, sivuston käyttäjäystävällisyys kärsii. Lisäksi ilmoitusäänet on mahdollista ottaa pois käytöstä yläreunassa sijaitsevaa nappia painamalla. Ilmoitusääniä ei kuitenkaan saatu toimimaan iPhone-puhelimilla, koska äänen toistaminen ei ole mahdollista ilman, että käyttäjä on jollain tapahtumalla aloittanut äänen toiston, eli esimerkiksi painamalla toisto-nappia.

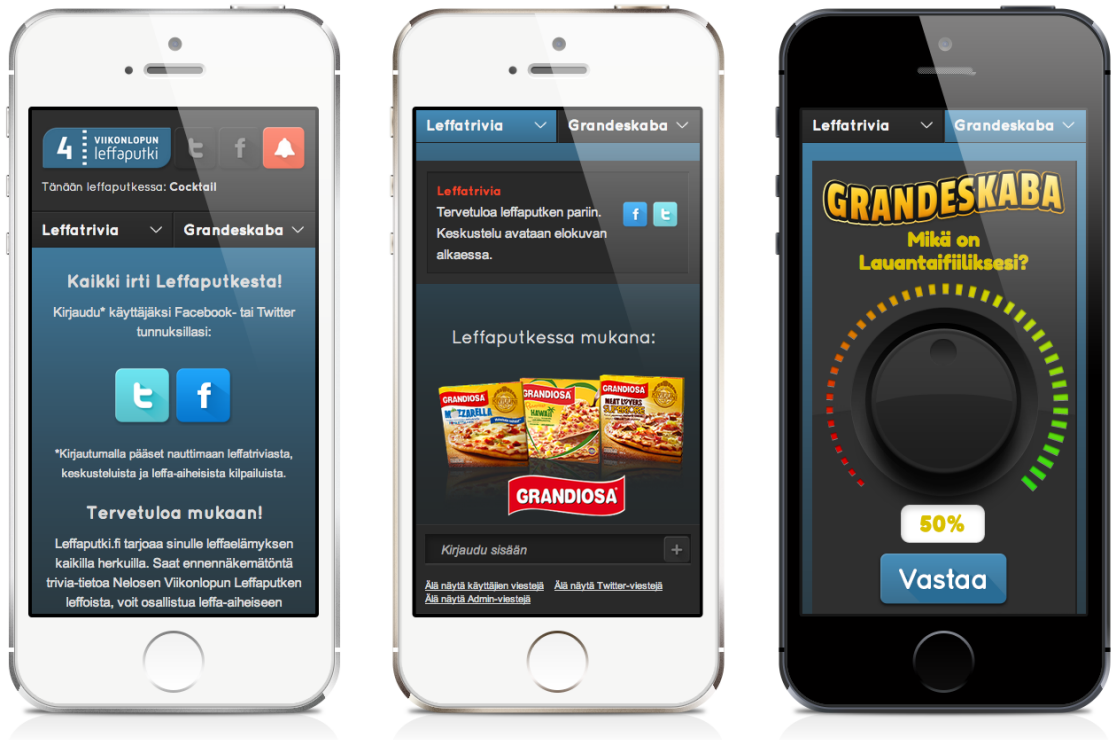
Ylläpitoviesteihin lisättiin myös mahdollisuus jakaa ne Facebookissa tai Twitterissä. Facebook-jako toteutettiin generoimalla viestin yhteydessä hyödyntäen Facebook-rajapinnan tarjoamaa fb.ui-kutsua, jolla voidaan lähettää jako-kutsu Facebookiin. Jako-viestinä lähetettiin Nelosen logo, kommentti ja sivuston osoite. Twitteriin viesti lähetetään Twitterin jako-linkkiä hyväksikäyttäen. Twitterin jako-linkkiin lisätään halutut kentät parametreinä. Twitteriin lähetetään samat tiedot pois lukien jakokuva.

Node.js-palvelun suoritus on varmistettu Forever-moduulilla. Se käynnistää Node.js-prosessin uudelleen heti virheen sattuessa. Näin varmistetaan, että reaaliaikainen keskustelu on aina päällä. Palvelu on tietenkin myös testattu kattavasti, ja virheen sattuessa tallennetaan lokitiedostoihin virheiden syyt, jotta voidaan tarkkailla, onko kaikki virhetilanteet huomioitu.

Node.js-palvelun käynnistyessä haetaan Wordpressin tietokannasta kaikki ylläpitokäyttäjän viestit, jotka on ajastettu tulevaisuuteen. Sen jälkeen ajastusobjekti rakennetaan uudelleen ja varmistetaan, että käyttäjille lähetetään ylläpitäjän viestit.

### 3.3 Mobiilioptimointi

Leffaputki.fi-sivustosta tehtiin mahdollisimman kevyt, jotta se latautuisi mahdollisimman nopeasti mobiilissa käyttäjille. Sivulla käytettiin mahdollisimman paljon CSS:llä muotoiluja elementtejä kuvien sijasta. Napeista ja fonttikoosta tehtiin mahdollisimman suuret, jotta niitä olisi helppo käyttää ja teksti on luettavaa. Kuvassa 5 on esimerkki sivuston eri näkymistä mobiililaitteella. Käytettävyyteen kiinnitettiin paljon huomiota.



Kuva 5. Leffaputki.fi-sivuston mobiilinäkymät.

JavaScript-tiedostot minimoitiin, ja kaikki mahdolliset tiedostot yhdistettiin yhdeksi tiedostoksi, jotta pyyntöjen määrä saadaan mahdollisimman pieneksi. Sama tehtiin tyyli-tiedostoille.

Kaikki kuvat optimoitiin ImageOptim-nimisellä ohjelmalla mahdollisimman pieneksi. Sivustolla hyödynnetään myös selaimen välimuistia kuvien lataamiseen ja kaikkiin muihin resursseihin, mihin se oli mahdollista lisätä.

Viestien listauksesta poistettiin myös sisäinen vierityspalkki käytön helpottamiseksi, ja kommentointikenttä asetettiin ruudun alareunaan aina näkyville. Navigaatio asetettiin sivun yläreunaan, niin että se on käyttäjälle koko ajan näkyvillä.

jQuery-kirjasto on pakollista ladata sivun head-osiossa Gravity Forms -lisäosan takia. Gravity Forms lisää lomakkeen loppuun Ajax-koodin, ja siitä aiheutuu virhe, jos jQueryä ei ole ladattu sitä ennen. Tämä aiheuttaa sivun lataamiseen tukkeutumisen, mutta siihen ei ollut muuta ratkaisua.

W3 Total Cache -Wordpress-lisäosalla otettiin käyttöön selaimen, tietokannan ja objektien välimuistin hyödyntäminen. Tämän avulla sivuston latausaika saatiin hyväksyttävälle tasolle.

## 4 Interaktiivisen verkkosivuston suunnittelu

### 4.1 Reaaliaikainen keskustelu

Reaaliaikaisessa keskustelussa on mahdollista lisätä viestejä, jotka päivittyvät kaikille käyttäjille välittömästi Socket.IO-moduulin avulla. Viestien lähettämiseen vaaditaan kuitenkin tunnistautuminen, joko Twitterin tai Facebookin avulla. Tunnistautuminen on tehty Twitterin ja Facebookin tarjoamien OAuth 2.0 -tunnistautumisen avulla. Ensin kyseisissä palveluissa tehdään sovellus palvelulle, jolla varmennetaan sivun verkkotunnus ja saadaan rajapinta-avaimet palvelun käyttöön.

Twitterin OAuth-tunnistautumiseen käytettiin hyväksi Abraham Williamsin kirjoittamaa PHP-kirjastoa, joka tekee tunnistautumisen ohjelmoinnin kannalta huomattavasti helpommaksi. Se tarjoaa tunnistautumista varten käyttöön tarvittavat perusfunktiot. Facebook-tunnistautuminen tehtiin käyttämällä Facebookin tarjoamaa PHP SDK:ta (Software development kit). Tämän lisäksi sivulle lisättiin Facebookin JavaScript SDK, jolla sivua on mahdollista jakaa Facebookiin. (25.)

Tunnistautumisessa tarkistetaan ensin, onko käyttäjä jo valmiiksi kirjautuneena valitsemaansa palveluun. Jos käyttäjä ei ole kirjautuneena, häntä pyydetään ensin kirjautumaan palveluun, ja sen jälkeen käyttäjältä pyydetään profiilin perusoikeudet, joihin kuuluvat käyttäjän tunniste, nimi ja sähköpostiosoite. Reaaliaikaiseen keskusteluun nämä tunnistetiedot riittävät, ja lisäoikeuksien pyytäminen saattaisi estää valvutuneempia käyttäjiä kirjautumasta palveluun.

OAuth 2.0 -kirjautuminen tehdään kahdessa vaiheessa, ilman kirjautumistietojen lähettämistä kolmansille osapuolille. Sivustolta ohjataan palveluntarjoajan sivustolle, ja mukana lähetetään varmenne ja takaisinohjausosoite. Tämän jälkeen käyttäjä hyväksyy oikeuksien luovuttamisen palvelulle ja käyttäjä ohjataan takaisin sivustolle. Sen jälkeen tarkistetaan, että varmenteet vastaavat toisiaan, ja annetaan käyttöoikeudet palveluun.



Kirjautumisen jälkeen sivustolla annettiin mahdollisuus jakaa sosiaalisessa mediassa viesti, että on alkanut käyttää palvelua ja jakaa tätä tietoa muille valitun palvelun käyttäjille. Tätä ei kuitenkaan tehty pakolliseksi, jotta ei rikota Facebookin ja Twitterin käyttöehtoja.

Facebook ja Twitter valittiin tunnistautumismenetelmiksi sillä perusteella, että ne ovat suosituimmat sosiaalisen median palvelut. Lisäksi niiden ajateltiin vähentävän herjauvien viestien lähettämistä palveluun, koska hyvin usein niitä käytetään henkilöiden oikeilla nimillä. Viestien lähettämisaikajankohta rajoitetaan alkamaan puoli tuntia ennen elokuvan alkua ja päättymään kello 00.00. Viestikenttä tyhjennetään joka yö kello neljä.

Viestien lähettämisessä ensimmäisessä vaiheessa käytettiin ainoastaan Socket.IO:n tarjoamia emit-funktioita. Se huomattiin kuitenkin nopeasti riittämättömäksi, koska tällöin viestejä ei pystytty valvomaan tarpeeksi hyvin. Viestejä pystyi lähettämään sivustolla oleville käyttäjille ilman palveluun kirjautumista, etsimällä tietoonsa tarpeellisten funktioiden toiminnan.

Asiakkaan mainetta haluttiin suojella ja päätettiin suhtautua asiaan vakavasti. Käytännössä tämä tarkoitti sitä, että uhrattiin osa noden nopeudesta lukemalla palvelimen kiintolevyltä MySQL-tietokantaa. Tämän avulla mahdollistettiin viestien parempi valvonta. Käyttäjien viestit lähetetään Ajaxilla PHP:lle, joka käsittelee kutsun. Lähetyksen mukana lähtee käyttäjän nimi, kommentti, sähköpostiosoite ja Wordpressin Ajax-varmenne. Ensin PHP:llä tarkistetaan, että viestin Wordpress-varmenne on oikea. Tämän jälkeen datasta poistetaan ylimääräiset merkit. Sen jälkeen varmennetaan, että nimi ja sähköposti vastaavat Twitterin tai Facebookin kirjautumistietoja. Lopuksi vielä varmistetaan, että käyttäjältä ei ole poistettu kommentointioikeuksia.

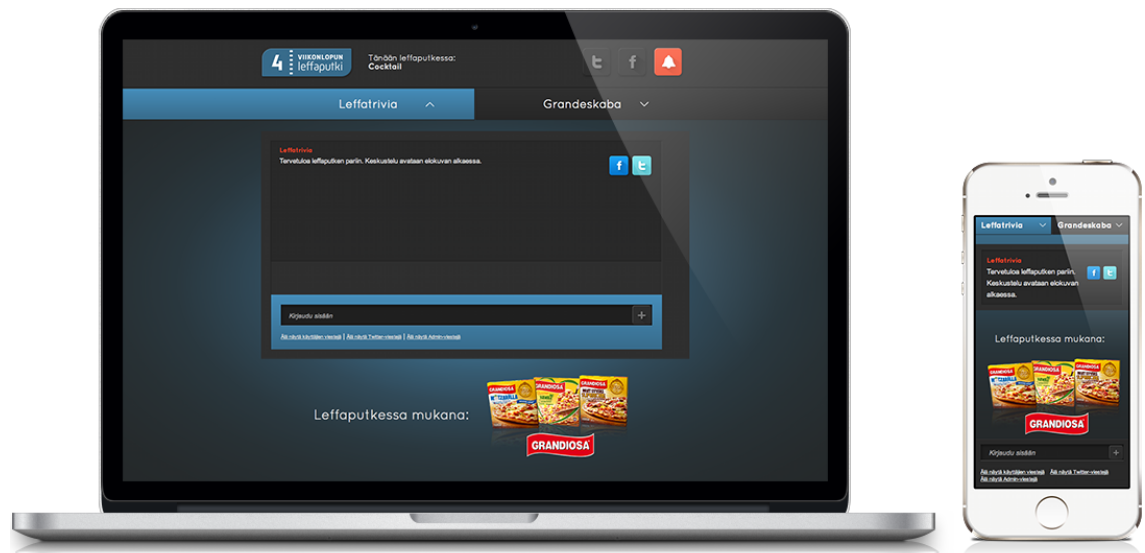
Tämän jälkeen kommentista poistetaan sanat, jotka on lisätty kiellettyjen sanojen listalle, josta löytyvät esimerkiksi yleisimmät kiro sanat. Ne korvataan "sensuroitu"-sanalla. Lisäksi viestistä poistetaan mahdollinen HTML-muotoilu ja tarkistetaan, onko kommentissa verkko-osoitteita. Sen jälkeen haetaan kommentoijan ip-osoite, selain, käyttöjärjestelmätiedot ja kellonaika, ja tarkistetaan, onko viestin lähettämisaikajankohta oikea. Ajankohta lisätään Wordpressin kommentteihin Wordpressin tarjoamalla wp\_insert\_comment-funktiolla. Kommenttiin lisätään vielä metatietona, onko käyttäjä kirjautuneena Twitteriin vai Facebookiin, ja lisäämään turvallisuutta viestistä poimituista

tiedoista koostetaan 10-merkkinen varmenne, joka on mahdollista myöhemmin tarkistaa.

Viesti lisätään Wordpressin kommentteihin kahdesta syystä. Ensinnäkin voidaan hyödyntää Wordpressiin rakennettua kommentoijien hallintaa, joka on asiakkaalle helppokäyttöinen. Lisäksi pystytään näyttämään aiemmin lisätyt kommentit käyttäjille, jotka tulevat sivustolle keskustelun avaamisen jälkeen tai päivittävät sivun keskustelun ollessa käynnissä.

Lisäämisestä palautetaan käyttäjälle kommentin tunniste, varmenne ja viesti, onko tiedon tallentamisessa tapahtunut virheitä. Nämä tiedot lähetetään palvelinpuolen JavaScriptille, jossa haetaan varmenteen ja tunnisteen avulla viesti, joka kuulutetaan jokaiselle käyttäjälle sivustolla.

Käyttäjät vastaanottavat objektin, jossa on kommentoijan nimi, viesti, kellonaika ja tieto, onko käyttäjä kirjautuneena Twitterin vai Facebookin kautta. Tämä parsitaan jQueryllä ja lisätään viesteihin. Kuvassa 6 on keskustelunäkymä pienellä ja suurella näytönkoollla. Huomioitavaa on, että mobiilinäkymässä kommentinsyöttökenttä on koko ajan ruudun alareunassa ja navigaatio yläreunassa. Viestit ilmestyvät näkymän yläreunaan värikoodattuna pääkäyttäjän, Twitterin tai Facebookin, mukaan.



Kuva 6. Leffaputki.fi-keskustelunäkymä suurella ja pienellä näyttökoolla.

Tämän lisäksi sivun ylläpitäjälle annettiin oikeudet poistaa viestejä reaaliajassa. Kirjautuneena olevan ylläpitäjän ei tarvinnut kuin painaa asiaton viesti -painiketta viestissä, minkä seurauksena se vedetään pois kaikilta käyttäjiltä reaaliaikaisesti. Samalla käyttäjä lisättiin mustalle listalle ja henkilön kirjoitusoikeudet evättiin. Normaalikäyttäjillä on mahdollisuus ilmiantaa asiaton viesti, jolloin se lähetettiin ylläpitäjän sähköpostiosoitteeseen. Ilmiannetussa viestissä on mukana ilmiantajan sähköpostiosoite, nimi ja palvelu, johon hän on kirjautunut, viesti, joka ilmiannettiin ja viestin kirjoittajan nimi, sähköpostiosoite ja palvelu, johon käyttäjä oli kirjautunut.

Suuria näytönkokoja varten tehtiin räätälöity vierityspalkki mCustomScrollbar jQuery-kirjastolla. Oletusvierityspalkkia käytettäessä sivuston ulkoasu olisi rikkoutunut täysin. Mobiilikäyttäjille päätettiin tulostaa näkymään kaikki viestit ja käyttää tavallista sivun vieritysmenetelmää, jolla pyrittiin parantamaan käyttömukavuutta.

Viestejä on myös mahdollista lähettää Twitteristä asiakkaan valitsemalla aihe-tunnisteella. Tämä on toteutettu hieman muokatulla versiolla Twitter Mentions As Comments -Wordpress-lisäosaa käyttäen. Twitter Mentions As Comments -lisäosa hakee Twitteristä kaikki twiitit, joissa on linkki johonkin Wordpressin sivuun, lisää ne Wordpressin kommentteihin, tallentaa uusimman twiitin tunnisteeseen ja seuraavassa haussa jatkaa tästä tunnisteesta. Autentikoitu sivusto voi tehdä enintään 450 hakua Twitteriin viidesätoista minuutissa.

Lisäosa muokattiin käyttämään verkko-osoitteiden sijasta hashtagia. Lisäksi käyttäjien kuvahaku otettiin pois käytöstä, jolloin saadaan mahdollisimman paljon hakuja Twitteriin. Tämän jälkeen lisäosa ajastettiin Linuxin cron-ajastuspalvelun avulla tekemään jokaisella minuutilla haku, joka palauttaa kaikki uudet viestit, joita palveluun on lisätty. Sen jälkeen ne käsiteltiin samalla tavalla kuin käyttäjien lähettämät viestit. Tällä varmistettiin, että vaikka Twitteriin tulisi tuhansia viestejä, eivät rajoitukset tule vastaan. Viiveenä on hyväksyttävä yksi minuutti.

Keskustelunäkymä tehtiin myös suodatettavaksi, ja siihen valittiin kolme kriteeriä: näkymästä oli mahdollista poistaa suoraan Twitteristä tulleet viestit, käyttäjien viestit tai Leffatrivia-viestit.

## 4.2 Kilpailu

Leffaputki.fi-sivustolla oleva kilpailu on muutettava moduuli, jonka asiakas voi myydä yhteistyökumppaneille halutuksi ajaksi. Ensimmäisessä kilpailussa haluttiin pyöritettävä nappi, johon asetetaan fiilis ja sen jälkeen täytetään lomake, josta osallistutaan kilpailuun. Lomakkeiden vastaukset lähetetään hyödyntäen Gravity Formsin rajapintaa Nelonen Median Drupal-järjestelmään, mistä ne on mahdollista poimia TV-lähetykseen. Päivän voittaja ilmoitettiin aina lähetyksen lopuksi.

Pyöritettävään nappiin käytin jQuery-lisäosaa nimeltä jQuery Knob. Ensivaikutelma lisäosasta oli, että se on toteutettu hyvin ja se toimii juuri niin kuin pitääkin. jQuery Knob lisää canvas-elementin sivustolle. Elementtiin piirretään ympyrä, ja ympyrän kaarta seurataan laskemalla osoittimen sijainti canvas-elementillä. Sitä jouduttiin kuitenkin muokkaaman hieman, että saatiin haluttu grafiikka nappiin. Toteutin napin muuttamalla valitsimen ja taustan värit läpinäkyviksi ja lisäämällä kuvaelementin seuraamaan ympyrää. Laskutoimituksen kopioin Stackoverflow-keskustelufoorumilta. Vaikka insinöörimatematiikka on hallussa, en keksinyt tähän vastausta. Kuvassa 7 on näkyvässä räätälöity pyöritettävä nappi.



Kuva 7. Leffaputki.fi-sivuston kilpailu suurella ja pienellä näyttökoolla.

Pyöritettävän valitsimen käytettävyys on hyvä kaikilla muilla laitteilla, paitsi Windows Phone 7 -käyttöjärjestelmällä. Windows Phone 7 -käyttöjärjestelmässä olevalla Internet Explorer 9 -selaimella ei ole mahdollista raahata elementtejä, ja tämä aiheuttaa luulta-

vasti hämmennystä. Valinta tehdään joko syöttämällä kenttään prosenttiluku tai painamalla halutusta kohtaa ympyrän kaarta. Jouduin myös rakentamaan tuen jQuery Knob -kirjastoon uudemmille Windows Phone -laitteille. Windows Phone -laitteet käyttävät hieman eri tapahtumaa kosketuksen rekisteröimiseen. Rekisteröinti onnistui kuitenkin helposti lisäämällä tarvittavat tapahtumat koodiin. Rajapinta on muuten samanlainen kuin muissa kosketusnäyttöissä, kutsu on vain eri. Kilpailun alareunassa on mainos yhteistyökumppaneiden tuotteisiin.

Tämän lisäksi sivun latausnäkyvä, etusivu ja keskustelusivu brändättiin yhteistyökumppanin logolla.

## 5 Yhteenveto

Insinööriyönä tehtiin Nelonen Medialle uusia tekniikoita hyödyntäen palvelu antamaan lisäarvoa elokuvien katseluun. Nelonen Medialle päätettiin suunnitella ja toteuttaa niin sanottu 2nd Screen -sovellus, jossa on moderoitava reaaliaikainen keskustelu, johon asiakas pystyy helposti ajastamaan viestejä. 2nd Screen -sovellukset ovat esimerkiksi television katselun aikana käytettäviä palveluita, jotka tukevat television ohjelmaa. Ajastetut viestit haluttiin rikastuttamaan elokuvakokemusta kertomalla esimerkiksi kuvauspaikoista tai ohjaajasta lisätietoja synkronoituna lähetyksiaikaan. Sivustolle tehtiin Facebook- ja Twitter-integraatio vähentämään viestien törkyisyyttä. Tämän lisäksi viesteistä suodatettiin yleisimmät kirosanat. Viestien lähettäminen tapahtuu Ajaxin ja WebSocket-yhteyden kautta.

Verkkosivusto toteutettiin Wordpress-julkaisujärjestelmällä helpottamaan asiakkaan sisällönsyöttöä, ja samalla saatiin käyttöön esimerkiksi kommentoijien musta lista. Wordpress-lisäosalla Gravity Forms lisättiin lomake, josta kerättiin potentiaalisten asiakkaiden yhteystietoja.

Verkkosivun Wordpress-teeman pohjana käytettiin Foundation 4 -ohjelmistokehystä. Sivustolla oli todella tärkeää, että se toimi mobiililaitteissa moitteetta, ja Foundationin avulla se onnistui. Kehitykseen otettiin myös avuksi Node.js- ja Socket.IO-kirjastot, joilla saatiin WebSocket-yhteyksille laaja selain- ja laitteistotuki.

Verkkosivun idea oli hyvä, mutta se ei saavuttanut suosiota. Elokuvia katsoessa halutaan keskittyä elokuvaan, ja elokuvasta keskustelu käydään yleensä elokuvan jälkeen. Toki elokuvasta sai aina hyvää lisätietoa palvelussa, ja vaikka sitä aluksi mainostettiin hyvinkin paljon elokuvien aikana, palvelulle sopivampi formaatti olisivat esimerkiksi urheilu- tai keskusteluohjelmat. Urheiluohjelmissa kannattajat ovat aina yhteisöllisiä ja keskusteluohjelmien aiheista olisi varmasti saatu jatkokeskustelua internetissä.

Nelonen Media ei enää ylläpidä sivustoa, vaikka se edelleen on osoitteessa leffaputki.fi. Keskustelua on kuitenkin mahdollista edelleen käydä, vaikka Leffatrivia-käyttäjä on hiljentynyt.

Verkkosivu sai vähän, mutta positiivista palautetta, ja asiakas oli tyytyväinen toteutukseen. Nelonen Media oli yllättynyt palvelun helppokäyttöisyydestä ja varsinkin kommenttien ajastuksesta ja päivittämisestä. Palvelua oli tarkoitus kehittää niin, että se sisältää tiedot viikonlopun leffaputken elokuvista ja niiden trailerit. Tiedot oli tarkoitus hakea Nelosen tarjoaman rajapinnan kautta.

Toinen mahdollinen kehitysidea olisi ollut lisätä mahdollisuus vastata yksittäisiin viesteihin. Sen toteutukseen olisi luultavasti mennyt muutama henkilötyöpäivä, muuttamalla logiikkaa siten, että viestit saadaan luokitella tunnisteiden mukaan ryhmiksi ja muokkaamalla ulkoasua kuvaamaan näitä ryhmittymiä. Viestien luettavuutta ja keskustelun rakennetta olisi kuitenkin suunniteltava mahdollisesti hyvinkin paljon, koska viestiketjuun saattaisi tulla uusia viestejä mihin kohtaan tahansa. Jos keskustelu olisi kiivasta, viestejä tulisi mahdollisesti ruudun ylä- ja alapuolelle, jolloin luettava kohta ruudulla myös vaihtuisi.

Sivustoa mahdollisesti muokataan käytettäväksi johonkin muuhun formaattiin kuin elokuvaan. Siihen ei vaadittaisi kuin ulkoasun päivittäminen ja mahdollisesti pieniä muutoksia sivun toimintaan.

Projekti valmistui nipin napin aikataulussa. Viimeiset kaksi yötä menivät ohjelmoidessa viimeisiä ominaisuuksia, mutta kaikki oli valmista julkaisuajankohtana. Viimeisimpinä ongelmina oli julkaisupalvelimen kello, joka oli GMT +0 -ajassa, ja tämän vuoksi viestit ilmestyivät kaksi tuntia liian aikaisin. Tämä huomattiin kuitenkin, ennen kuin sivusto oli käytössä. Sivuja julkaistiin 13.9.2013, ja sen käyttö lopetettiin 24.11.2013.

## Lähteet

- 1 Tribbey, Chris. 2014. Nielsen Report Confirms Second Screen Growth. Verkkodokumentti. 2nd Screen Society. <<http://www.2ndscreen.com/blog/2014/02/11/nielsen-report-confirms-second-screen-growth/>>. Päivitetty 11.2.2014. Luettu 1.4.2014.
- 2 Bracey, Kezz. 2014. Why I Choose Stylus (And You Should Too). Verkkodokumentti. Envato Pty Ltd. <<http://webdesign.tutsplus.com/articles/why-i-choose-stylus-and-you-should-too--webdesign-18412>>. Päivitetty 21.2.2014. Luettu 1.4.2014.
- 3 Lubbers, Peter & Albers, Brian & Salim, Frank. 2010. Pro HTML5 Programming: Powerful APIs for Richer Internet Application Development. New York, USA: Ap-ress.
- 4 JavaScript APIs Current Status. 2014. Verkkodokumentti. W3C. <[http://www.w3.org/standards/techs/js#w3c\\_all](http://www.w3.org/standards/techs/js#w3c_all)>. Luettu 1.4.2014.
- 5 About HTML5 WebSockets. 2013. Verkkodokumentti. Kaazin Corporation. <<http://www.websocket.org/aboutwebsocket.html>>. Luettu 1.4.2014.
- 6 Bidelman, Eric. 2012. Capturing Audio & Video in HTML5. Verkkodokumentti. HTML5Rocks. <<http://www.html5rocks.com/en/tutorials/getusermedia/intro/>>. Päivitetty 29.10.2013. Luettu 1.4.2014.
- 7 General Overview. 2012. Verkkodokumentti. Google Inc. <<http://www.webrtc.org/reference/architecture>>. Luettu 1.4.2014.
- 8 Sass Basics. 2014. Verkkodokumentti. Sass. <<http://sass-lang.com/guide>>. Luettu 12.1.2014.
- 9 Stylus. 2014. Verkkodokumentti. Learnboost. <<http://learnboost.github.io/stylus/>>. Luettu 1.4.2014.
- 10 Getting Started. 2014. Verkkodokumentti. Less Core Team. <<http://lesscss.org/>>. Luettu 1.4.2014.
- 11 Flanagan, David. 2011. JavaScript: The Definitive Guide. Sixth Edition. Sebastopol CA, USA: O'Reilly.
- 12 W3Schools JavaScript Introduction. 2014. Verkkodokumentti. Refsnes Data. <[http://www.w3schools.com/js/js\\_intro.asp](http://www.w3schools.com/js/js_intro.asp)>. Luettu 16.1.2014.

- 13 Heilmann, Christian. 2006. Beginning JavaScript with DOM Scripting and Ajax. New York, USA: Apress.
- 14 jQuery. 2012. Verkkodokumentti. jQuery Project . <<http://www.jquery.com/>>. Luettu 4.5.2012.
- 15 Castledine, Earle & Sharkie, Craig. 2010. jQuery Novice to Ninja. Collingwood, Australia: Sitepoint Rtye.
- 16 Why use Modernizr. 2013. Verkkodokumentti. Modernizr. <<http://modernizr.com/>>. Luettu 1.4.2014.
- 17 Faq. 2012. Verkkodokumentti. Learnboost. <<http://socket.io/#faq>>. Luettu 1.4.2014.
- 18 Why use Zepto. 2014. Verkkodokumentti. Freckle Online Time Tracking. <<http://zeptojs.com/>> Luettu 1.4.2014.
- 19 Nixon, Robin. 2012. Learning PHP, MySQL, JavaScript, and CSS. Second Edition. Sebastopol CA, USA: O'Reilly.
- 20 Mitchell, Lorna & Shafik, Davey & Turland, Matthew. 2011. PHP Master: Write Cutting-edge Code. Collingwood, Australia: Sitepoint Rtye.
- 21 Usage of operating systems for websites. 2014. Verkkodokumentti. W3Techs. <[http://w3techs.com/technologies/overview/operating\\_system/all](http://w3techs.com/technologies/overview/operating_system/all)>. Päivitetty 1.4.2014. Luettu 1.4.2014.
- 22 Rauch, Guillermo. 2012. Smashing Node.js Javascript Everywhere. Chichester, UK: Wiley.
- 23 Node's goal is to provide an easy way to build scalable network programs. 2014. Verkkodokumentti. Joyent Inc. <<http://nodejs.org/about/>>. Luettu 4.2.2014.
- 24 npm-faq. 2014. Verkkodokumentti. Joyent Inc. <<https://npmjs.org/doc/faq.html>>. Luettu 4.2.2014.
- 25 About. 2014. Verkkodokumentti. Bootstrap. <<http://getbootstrap.com/about/>>. Luettu 6.2.2014.
- 26 Foundation. 2014. Verkkodokumentti. Zurb Inc. <<http://foundation.zurb.com/>>. Luettu 6.2.2014.
- 27 Grunt The JavaScript Task Runner. 2014. Verkkodokumentti. Grunt. <<http://gruntjs.com/>>. Luettu 6.2.2014.



- 28 Bower A package manager for the web. 2014. Verkkodokumentti. Twitter. <<http://bower.io/>>. Luettu 6.2.2014.
- 29 Nardi, Felipe. 2014. Top 5 Core Differences Between Bootstrap 3 and Foundation 5. Verkkodokumentti. Medium. <<https://medium.com/frontend-and-beyond/8b3812c7007c>>. Päivitetty 25.1.2014. Luettu 6.2.2014.
- 30 What is Joomla. 2014. Verkkodokumentti. Open Source Matters, Inc. <<http://www.joomla.org/about-joomla.html>>. Luettu 10.2.2014.
- 31 Usage of content management systems for websites. 2014. Verkkodokumentti. W3Techs. <[http://w3techs.com/technologies/overview/content\\_management/all](http://w3techs.com/technologies/overview/content_management/all)>. Päivitetty 1.4.2014. Luettu 1.4.2014.
- 32 Mikoluk, Kasia. 2013. Drupal vs Joomla vs Wordpress: CMS Showdown. Verkkodokumentti. udemy. <<https://www.udemy.com/blog/drupal-vs-joomla-vs-wordpress/>>. Päivitetty 18.7.2013. Luettu 10.2.2014.
- 33 Coffey, Thomas. 2013. Top 3 CMS: Wordpress Drupal & Joomla Comparison. Verkkodokumentti. Media Heroes. <<http://www.seohero.com.au/blog/platform-reviews/top-3-cms-content-management-system-comparison.php>>. Päivitetty 14.6.2013. Luettu 10.2.2014.
- 34 Drupal vs Joomla. 2014. Verkkodokumentti. ITX Design. <<http://itxdesign.com/the-cms-war-drupal-vs-joomla/>>. Luettu 10.2.2014.
- 35 Rauch, Guillermo. 2010. Websockets everywhere with Socket.IO. Verkkodokumentti. HowToNode.org. <<http://howtonode.org/websockets-socketio>>. Päivitetty 29.8.2010. Luettu 18.2.2014.
- 36 Williams, Abraham. 2009. TwitterOAuth. Verkkodokumentti. GitHub, Inc. <<https://github.com/abraham/twitteroauth>>. Päivitetty 14.6.2013. Luettu 18.2.2014.

