Lappeenrannan teknillinen yliopisto
Tuotantotalouden tiedekunta
Tietotekniikan koulutusohjelma

Kandidaatintyö

Kalle Koponen

USEAMMAN MONITORIN WEB-SOVELLUKET: TEKNISET HAASTEET JA MAHDOLLISUUDET

Työn tarkastaja: Tutkijatohtori Ari Happonen

Työn ohjaaja: Tutkijatohtori Ari Happonen

TIIVISTELMÄ

Lappeenrannan teknillinen yliopisto
Tuotantotalouden tiedekunta
Tietotekniikan koulutusohjelma

Kalle Koponen

Useamman monitorin web-sovellukset: tekniset haasteet ja mahdollisuudet

Kandidaatintyö

2014

35 sivua, 6 kuvaa, 4 taulukkoa

Työn tarkastaja: Tutkijatohtori Ari Happonen

Hakusanat: usean monitorin sovellus, web-sovellus, web-tekniikat
Keywords: multi-monitor application, web-application, web-technologies

Työssä tutkikaan, miten web-teknologiat soveltuvat usean monitorin sovelluksiin web-ympäristössä, ja todeta tähän liittyvät ongelmat sekä niiden mahdolliset ratkaisut teknisellä tasolla. Tutkituista teknologioista valitaan parhaiten soveltuva ratkaisu ja sen avulla luodaan usean monitorin sovellus, jonka avulla pureudutaan teknillisiin mahdollisuuksiin ja rajoitteisiin. Työn tuloksena nähdään, että usean monitorin web-sovellukset sisältävät monia ohjelmistoteknillisiä ongelmia ja ne sopivat lähinnä järjestelmiin, joiden ympäristö, esimerkiksi käyttöjärjestelmä tai selain, tiedetään. Tällöin voidaan turvautua käyttökokemusta parantaviin selaimen laajennuksiin tai ulkopuolisiin liitännäisiin, jotka toimivat kyseisessä ympäristössä.

ABSTRACT

Lappeenranta University of Technology
Faculty of Technology Management
Degree Program in Information Technology

Kalle Koponen

Multi-monitor web-applications: technical limitations and possibilities

Bachelor's Thesis

35 pages, 6 figures, 3 tables

Examiner: D.Sc. (Tech.) Ari Happonen

Keywords: multi-monitor application, web-application, web-technologies

This thesis studies which web-technologies are suitable for multi-monitor applications in web-environment and state any technical problems and their solutions. A multi-monitor web-application is developed using most applicable technology. The solution evaluates current technical possibilities and limitations. The outcome shows that multi-monitor web-applications suffers from many technical limitations and are most suitable for systems which environment is known. This way it's possible to use browser extensions or third-party plugins which work in this environment to enhance user experience.

SISÄLLYSLUETTELO

SYMBOLI- JA LYHENNELUETTELO

API	Application Programming Interface
DOM	Document Object Model
HTML	Hypertext Markup Language
URL	Uniform Resource Identifier
W3C	World Wide Web Consortium

1 JOHDANTO

Kun vielä joitain vuosia sitten 24-tuumainen monitori saattoi kustantaa lähemmäs 1000 euroa, saa samanlaisen näytön nykyään kymmenesosa hinnalla. Web-sovellukset alkavat muistuttaa yhä enemmän työpöytäsovelluksia. Nykyään ollaan siinä tilanteessa, että useammat näytöt ovat arkipäivää käyttäjien muokatessa dokumentteja web-sovelluksilla samalla sähköpostin ollessa jatkuvasti valvovan silmän alla. Useampaa monitoria ei kuitenkaan käytetä hyväksi web-sovelluksissa, vaikka niiden kehittyminen on lisännyt kaikki edellytykset suurempien näyttöpinta-alojen hyödyntämiseen. Web-kehityksessä voitaisiin hyödyntää työpöytäympäristön tarjoama potentiaali nykyistä huomattavasti paremmin, jolloin saataisiin entistä monipuolisempaa sisältöä selaimessa ajettavaksi.

1.1 Tausta

Web-teknologiat ovat kehittyneet viime vuosina todella paljon. Nykyään web-kehityksessä suositaan niin sanottua responsiivista suunnittelua eli sivun ulkoasu suunnitellaan mukautuvaksi eri käyttölaitteiden mukaan. Samaan aikaan kynnys useamman monitorien hankkimiseen on laskenut hintojen tullessa alas ja edullisimpien näytönohjainten tukiessa jopa neljää näyttöä. Tämä suuntaus ei kuitenkaan näy millään tavalla web-kehityksessä vaan sivut suunnitellaan yhden näytön ja mobiililaitteiden ehdoilla, sillä niiden osuus on kasvanut huomattavasti viime vuosien aikana. Tässä työssä lähdetään uimaan vastavirtaan ja tutkitaan web-sovelluksien soveltuvuutta useammalle monitorille.

1.2 Tavoitteet ja rajaukset

Työn tavoitteena on tarkastella web-teknologioiden soveltuvuus monen monitorin hyödyntämiseen, todeta tähän mahdollisesti liittyvät ongelmat ja niiden ratkaisut teknisellä tasolla. Työssä ei tarjota ratkaisuja ei-teknisiin haasteisiin. Työssä toteutetaan useampaa monitoria ja web-teknologioita hyödyntävä sovellus, jonka avulla pureudutaan teknologioiden tarjoamiin mahdollisuuksiin. Sovelluksen päätavoitteita ovat tehokas useamman monitorijärjestelmän hyödyntäminen ja skaalautuvuus.

1.3 Työn rakenne

Toisessa luvussa käydään läpi aiemmin suoritettuja tutkimuksia useampiin monitoreihin liittyen. Kolmannessa luvussa määritetään tutkimusongelma sekä käydään läpi toteutus metodologia. Neljännessä luvussa käydään läpi suosituimmat asiakaspääntoteutukset (eng. client-side), jotka soveltuvat web-kehitykseen useammalle monitorille. Luku viisi sisältää toteutustavan valinnan ja kuvauksen sovelluksen toteutuksesta, sekä tähän liittyvät tulokset. Luku kuusi sisältää omia pohdintoja työn aiheesta ja viimeinen luku sisältää työn johtopäätökset.

2 KIRJALLISUUSKATSAUS USEAMMAN MONITORIN KÄYTÖSTÄ

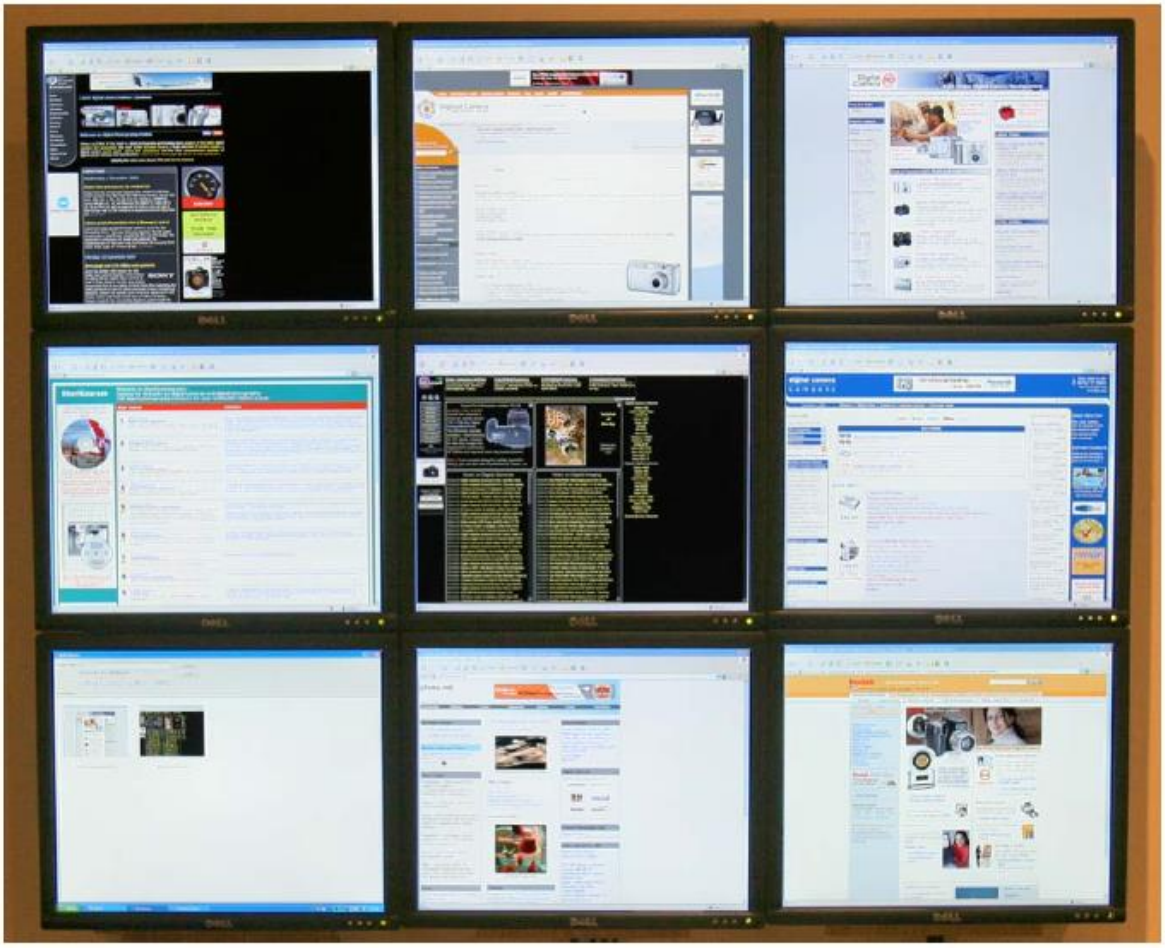
Useamman monitorin järjestelmistä on julkaistu useita tutkimuksia, jotka ovat keskittyneet usein järjestelmän tuomiin etuihin ja käyttötapauksiin. Anderson et al. (2004) totesivat useamman monitorin parantavan suorituskykyä ja tuotettavuutta. Tuotettavuus parantui 10-prosenttia, tehtävät suoritettiin 7-prosenttia nopeammin ja virheiden määrä väheni 33-prosentilla. Robert Ball et al. (2005) päätyivät samankaltaisiin tuloksiin, kun tutkimuksessa keskityttiin visuaalisen datan käsittelyyn. Stegman et al. (2011) eivät huomanneet mitään eroa yhden ja kahden monitorin välillä tehtävän suoritusajan suhteen. Syyksi epäiltiin esimerkiksi kohonneesta tarpeesta suurentaa ja siirtää ikkunoita koituvaa haittaa.

Robert Ball et al. (2005) tutkivat myös monitorijärjestelmiin liittyviä käyttäjätottumuksia. Tutkimuksessa sanottiin, että vaikka käyttäjät eivät yleisesti suosi monitorien reunuksista (eng. bezel) aiheutuvia katkelmia, todettiin reunusten kuitenkin auttavan erottelemaan erilaista sisältöä eri monitoreihin. Reunusten koettiin olevan ongelma, jos sisältö, esimerkiksi kuva, oli levitetty koko järjestelmän kokoiseksi. Tämä käyttäjätottumus tarjoaa yhden lähestymistavan web-sovelluksien useamman monitorin hyödyntämiseen. Sovellus on suunniteltu siten, että erilainen sisältö on helposti eroteltavissa eri monitoreihin.

Web-ympäristöön liittyvät tutkimukset ovat usein käsitelleet web-sisällön siirtoa eri laitteiden välillä (Ghiani, et al., 2012; Huber & Ding, 2012). Brad Johanson et al. esittelivät (2001) multibrowsing-viitekehyksen. Tähän työhön liittyen viitekehyksessä on esitelty tapa avata web-linkit toiseen monitoriin. Toteutus nojaa vahvasti järjestelmätasolla toimiviin sovelluksiin ja palveluihin, joten se ei hyödynnä web-teknologioita muuten kuin web-linkin esitystavassa.

George Robertson et al. (2005) esittelivät useamman monitorin mahdolliseksi käyttökohteeksi web-sisällön jakamisen useammalle näytölle. Esimerkkinä annettiin

käyttäjän tekemä haku, joka jakautui yhdeksälle monitorille. Kahdeksan ensimmäistä monitoria näyttivät kahdeksan ensimmäistä hakutulosta ja yhdeksänteen monitoriin käyttäjällä oli mahdollista poimia mielenkiintoisia tuloksia. Toteutus tehtiin Internet Explorer-selaimen laajennuksena. Toteutustavasta ei kuitenkaan kerrota tämän tarkemmin ja jääkin epäselväksi käytettiinkö laajennuksen teossa apuna mitään web-teknologioita.



Kuva 1. Useamman monitorin hakupalvelu (Robertson, et al., 2005)

Tutkimuksia on haettu seuraavista tietokannoista: ACM Digital Library, EBSCO - Academic Search Elite, EBSCO - Business Source Complete, Elsevier (Science Direct), Emerald Journals (Emerald), IEEE, Springer Link eBooks, Springer Link eJournals, Talentum lehtiarkisto ja Google Scholar.

Tutkimuksia on haettu seuraavilla hakusanoilla ja näiden yhdistelmillä: Multiple monitor,

Multi-monitor, Multi-display, Multiple display, Web, Application, Human Computer Interaction, Tiled display, Multi monitor environment, Multibrowsing, WWW ja World Wide Web.

3 TUKIMUSONGELMA

Työssä tutkitaan miten nykyiset web-teknologiat soveltuvat useamman monitorin web-sovelluksiin teknisiltä ominaisuuksiltaan. Lopputuloksena on todeta nykytilanne sekä minkälaisia parannuksia valittua teknologia kaipaa, jotta sen avulla olisi mahdollista luoda varteen otettavia useamman monitorin web-sovelluksia. Tuloksissa on myös taulukoituna yhteenveto puuttuvista ja jo löytyvistä ominaisuuksista.

Tilannetta lähdetään selvittämään tekemällä web-pohjainen sovellus, joka hyödyntää tehokkaasti useampaa monitoria. Toteutuksen avulla pureudutaan mahdollisuuksin ja ongelmiin.

Tutkimusongelman käsittely etenee ensin valitsemalla toteutettava sovellus. Tämän jälkeen valitaan parhaiten soveltuva toteuttava teknologia. Lopuksi toteutetaan itse sovellus ja otetaan kantaa toteutuksen aikana ilmenneisiin huomioihin työn rajauksen puitteissa.

Työn tutkimuskysymykset ovat:

1. Miten asiakaspään web-teknologiat soveltuvat työn ratkaisun toteuttamiseen?
2. Mitä teknisiä haasteita ja mahdollisuuksia työn ratkaisu tarjoaa?

3.1 Toteutus metodologia

Robertson et al (2005) mainitsivat tutkimuksessaan useamman monitorin käyttökohteeksi kuvassa 1 esitetyn web-pohjaisen tietohaun. Sen luonnehdittiin olevan jokapäiväinen esimerkki sovelluksesta, joka on suunniteltu useammalle monitorille. Toteutuksen mainittiin myös olevan konfiguroitavissa useammalle näyttökoolle ja määrälle, joten sen skaalautuvuutta pidettiin tärkeänä ominaisuutena.

Esimerkiksi pelkästään Google tarjoaa kymmenkunta erilaista hakumahdollisuutta. Tämän päälle rakennettu käyttäjän konfiguroitavissa oleva sovellus tarjoaa hyvät mahdollisuudet

useamman monitorin hyödyntämiseen ja tähän käytettyjen teknologioiden kartoitukseen. Sovellus on helposti skaalattavissa aina kahdesta monitorista suuriinkin järjestelmiin näyttämällä hakutulostyypin per monitori käyttöjärjestelmän rajat huomioon ottaen. Kirjallisuuskatsaus-osiossa todettiin, että näyttöjen reunukset auttavat jakamaan erilaista sisältöä näyttöjen kesken, joten hakutulostyypin jako per monitori myös varmistaa järjestelmän tehokkaan hyödyntämisen.

Ensimmäiseen tutkimuskysymykseen saadaan vastaus tutkittaessa suosituimpia asiakaspään web-teknologioita ja valittaessa näistä toteutukseen soveliain vaihtoehto. Toteutus- ja tulokset-osio tuovat vastaukset toiseen tutkimuskysymykseen.

4 ASIAKASPÄÄN WEB-TEKNOLOGIAT

Asiaskaspään web-teknologioilla tarkoitetaan selaimessa ajettavaa koodia käyttäjäpäässä. Mahdollisia asiakaspään toteutusvaihtoehtoja löytyy selaimille useita ja osa niistä vaatii erillisen liitännäisen asentamisen. Työssä esitellään useimmiten käytetyt asiakaspään toteutukset eli JavaScript, Flash, Silverlight ja Java, sekä näiden lisäksi Chrome-selaimen laajennukset.

W3Techs:in mukaan (W3Techs, 2014) web-sivuista 87,9% käyttää JavaScriptiä asiakaspään toteutukseen. Flashiä käytetään 14,9% sivustoilla, Silverlightin osuus on 0,2% ja Javan osuus 0,1%. Tuloksissa on hyvä huomioida, että sivusto voi käyttää useampaa toteutusta samanaikaisesti eli esimerkiksi sivusto voi käyttää osan sisällöstä tuottamiseen Flashiä tai Silverlightia mutta osa sivuston toiminnallisuudesta on tehty käyttäen JavaScriptiä. Tuloksia voidaan pitää suuntaa antavina asiakaspään toteutusten suosiosta.

Vertailuun on myös otettu mukaan Chrome-selaimen laajennukset, jotka tehdään käyttäen HTML5:sta ja JavaScriptiä. Lisäksi laajennuksilla on myös pääsy Chromen tarjoamaan rajapintaan (Google, 2014).

4.1 JavaScript

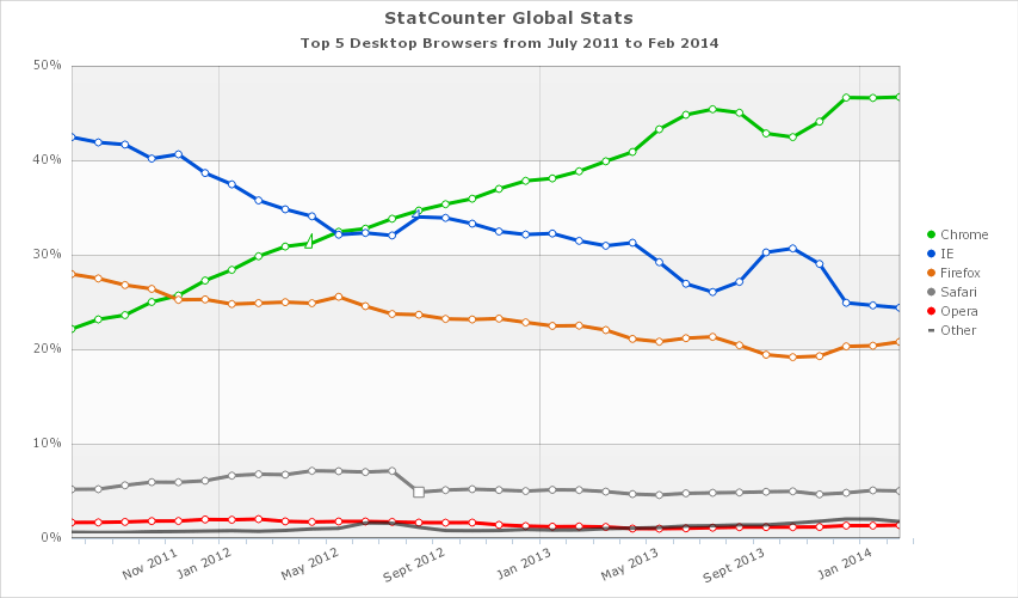
JavaScript on web-kehitykseen soveltuva ohjelmointikieli, joka mahdollistaa dynaamisten web-sivujen luonnin (Flanagan, 2011). JavaScript pohjautuu ECMAScript-standardiin (ECMA-262, 2011), joten kaikki modernit web-selaimet tukevat sitä, myös mobiililaitteissa. Esimerkiksi DOM (Document Object Model) käsittely ei ole kuitenkaan määritelty ECMAScript-standardissa, joten W3C (World Wide Web Consortium) on pyrkinyt DOM-käsittely standardointiin (W3C, 2013). Tästä johtuen saattaa selaimien DOM-käsittelystä löytyä poikkeavuuksia.

JavaScriptin suorittaminen tapahtuu suoraan selaimessa ilman kolmannen osapuolen

liitännäisiä. Hyvänä esimerkkinä JavaScriptin käytöstä toimii useiden sivujen hakutoiminnot. Käyttäjän kirjoittaessa hakusanaa alkaa samalla hakutoimenpiteiden suoritus, esimerkiksi hakuehdotusten näyttäminen, ilman erillistä sivulatausta.

JavaScript tarjoaa monitoreiden käsittelyyn todella rajoittuneet mahdollisuudet. Tarjotun rajapinnan avulla on mahdollista saada selville vain käyttäjän päämonitorin resoluutio, eikä useamman monitorin hallintaa ole otettu huomioon millään tavalla huomioon (W3Schools, 2014). JavaScript kuitenkin tarjoaa mahdollisuuden uusien ikkunoiden siirtämiseen toiseen monitoriin. Tämä kuitenkin edellyttää käyttäjän monitorikonfiguraation tietämistä etukäteen, jotta tiedetään monta pikseliä esimerkiksi ikkunaa pitää siirtää vasemmalle pääikkunan suhteen. Johtuen JavaScriptin erilaisista toteutuksista on selainten suhtautumisessa ikkunoiden siirtämiseen havaittavissa poikkeamia. Esimerkiksi Chrome-selaimen uusin versio (33) käyttää V8-moottoria (Google, 2013), joka ei anna siirtää uutta ikkunaa pois päämonitorista ilman pääsyä Chrome-selaimen omaan rajapintaan, joka vaati lisäosien käytön. Toisaalta Firefox-selaimen uusin versio (26.0) käyttää SpiderMonkey-moottoria (Mozilla, 2014), joka mahdollistaa ikkunoiden siirtämisen pois päämonitorista.

4.2 Chrome-selaimen laajennukset



Kuva 2. Chrome-selaimen suosioon nousu (StatCounter, 2014)

Chrome-selain on noussut muutamassa vuodessa suosituimmaksi selaimeksi (StatCounter, 2014). Laajennuksilla on mahdollista lisätä selaimeen lisätoimintoja, kuten esimerkiksi injektoida JavaScript-koodia sivustolle (Google, 2014), jolloin voidaan piilottaa vaikkapa tietyt elementit. Laajennuksilla on myös pääsy Chromen tarjoamaan rajapintaan, joka tarjoaa esimerkiksi ikkunoiden käsittelyyn JavaScriptin-toimintojen ohella uusia ominaisuuksia, kuten ikkunatyypin valinnan.

4.3 Java

Vaikka Javan ja JavaScriptin voisi nimen perusteella luulla liittyvän toisiinsa, ei näillä ole kuitenkaan mitään tekemistä keskenään. Javan kehityksestä huolehtii itsenäinen Oracle-yhtiö, eikä Javaa ole samalla tavalla standardoitu yleiseksi web-kehityskieleksi kuten JavaScriptiä. Javan käyttäminen web-kehityksessä vaatiikin alustariippuvaisen liitännäisen asentamisen käyttäjän koneelle.

Java tarjoaa hyvät mahdollisuudet järjestelmätietoihin käsiksi pääsyyn ja sen avulla on mahdollista saada tarkat tiedot käyttäjän monitorikonfiguraatiosta, kuten monitoreiden lukumäärän ja niiden resoluutiot (Oracle, 2013).

4.4 Adobe Flash

Flash on Adoben kehittämä ympäristö, jonka avulla voidaan luoda niin web-sivujen sisältöä kuin esimerkiksi mobiilisovelluksia. Flash vaati alustariippuvaisen liitännäisen asentamisen käyttäjän koneelle.

Flash ei tarjoa järjestelmätietoihin pääsyä. Sitä käytetäänkin lähinnä interaktiivisten elementtien tuottamiseen kuten videot tai pelit. Sen avulla on mahdollista kuitenkin kutsua web-sivun sisällä JavaScriptin funktioita mutta tällöin luonnollisesti kärsitään JavaScriptiin liittyvistä rajoittuvuuksista.

4.5 Silverlight

Silverlight on Microsoftin luoma kehitystyökalu interaktiivisten web- ja mobiilisovellusten tekemiseen. Silverlightia ei ole Javan tai Flashin tavoin standardoitu yleiseksi web-kehityskieleksi, joten sekin vaatii alustariippuvaisen liitännäisen asentamisen.

Silverlightin ei tarjoa monitoreiden käsittelyyn mitään rajapintaa. Flashin tavoin Silverlightia käytetään lähinnä interaktiivisen sisällön tuottamiseen ja sen avulla on myös mahdollista ajaa JavaScript-koodia.

4.6 Yhteenveto

Asiakaspään teknologiat voidaan karkeasti jaotella kahteen eri luokkaan: alustariippumattomiin ja erillisen liitännäisen vaativiin ratkaisuihin. Työssä käydyistä teknologioista vain JavaScript kuuluu ensin mainittuun luokkaan. Muut läpikäydyt teknologiat vaativat liitännäisen.

Taulukko 1. Yhteenveto asiakaspään tekniikoista

	Hyödyt	Haitat
JavaScript	Alustariippumattomuus	Ei pääsyä järjestelmätietoihin
Chromen laajennukset	Pääsy Chromen rajapintaan	Vain Chrome-selaimelle
Java	Pääsy järjestelmätietoihin	Vaatii liitännäisen
Flash		Vaatii liitännäisen, ei pääsyä järjestelmätietoihin
Silverlight		Vaatii liitännäisen, ei pääsyä järjestelmätietoihin

Taulukko 1 sisältää yhteenvedon läpikäydyistä asiakaspään teknologioista. Järjestelmätietoihin pääsyllä viitataan erityisesti mahdollisuutta hakea käyttäjän

monitorikonfiguraatio automaattisesti, jolloin vältytään tarpeelta käyttäjän manuaalisesti syöttää tiedot sovellukseen. Flash ja Silverlight eivät tarjoa tämän työn ratkaisulle mitään hyödyllisiä ominaisuuksia, joilla niiden liitännäisvaatimuksen voisi perustella.

5 TOTEUTUS

Toteutusta lähdetään työstämään ensin valitsemalla sovelluksen idea. Toteutus metodologia-osiossa sovelluksen tärkeiksi ominaisuuksiksi todettiin skaalautuvuus eri näyttökonfiguraatioille ja tehokas useamman monitorin hyödyntäminen, mikä tarjoaa hyvän mahdollisuuden arvioida valittua teknologiaa paljastamalla sen puutteet ja mahdollisuudet. Kohteeksi valittiin nämä ominaisuudet tehokkaasti hyödyntävä hakupalvelu.

Kun sovelluksen idea on selvillä, valitaan sen toteuttava teknologia aikaisemmin työssä käydyistä vaihtoehdoista. Toteutustavan valintaan vaikuttaa teknologian tarjoamien mahdollisuuksien lisäksi web-kehityksen nykysuuntaus ja tätä myötä teknologian tulevaisuuden kestävyys.

5.1 Johdatus toteutustavan valintaan

Mitkään neljännessä luvussa esitellyistä tekniikoista eivät tee autuaaksi vaan ne tulevat asettamaan kompromisseja toteutuksen suhteen. Flash ja Silverlight eivät tarjoa mitään, millä voisi perustella liitännäisen vaatimisen. Javasta löytyy mahdollisuus onkia järjestelmätietoja. JavaScript puolestaan toimii ilman liitännäisiä. Vaa'an tasapainoon saattamisesta taistelevat valmis tuki mutta rajoittuneisuus vastaan alustariippuvuus paremman potentiaalin kera.

Taulukko 2. Sovelluksen pääominaisuudet

Tarve	JavaScript	Java
Monitorikonfiguraation automaattinen selvitys/asetus		
Ikkunoiden avaus, paikan määritys, sisällön päivitys		
Asetusten tallennus muistiin (keksit jne.)		
Tulevaisuuden kestävyys		

5.1.1 Nykysuuntaus, tulevaisuuden kestävyys ja tietoturva

Viime aikoina web-kehityksessä on pyritty alustariippumattomuuteen, joka näkyy hyvin HTML5:n kehityssuunnasta. Yksi uusista ominaisuuksista on tuki videoille (W3C, 2014). Ennen videoiden toistossa täytyi luottaa kolmannen osapuolen liitännäisiin kuten Flashiin. HTML5 tarjoaa myös tuen canvas-elementille, jonka avulla on mahdollisuus renderöidä esimerkiksi grafiikkaa. Ennen tämäkin ominaisuus täytyi hoitaa kolmannen osapuolen liitännäisillä. Kehityksessä on siis pyritty riippuvuuksien vähentämiseen ja tämän avulla pyritty rakentamaan tulevaisuuden kestävä ratkaisu ottamalla jokainen alusta huomioon.

JavaScript perustuu avoimeen ECMAScript-standardiin ja on ainut täysin alustariippumaton vaihtoehto eli sen suorittaminen ei vaadi kolmannen osapuolen liitännäistä. Esimerkiksi W3C määrittelee web-arkkitehti standardeissaan JavaScriptin keskeisenä kielenä HTML5:n ohella (W3C, 2013). Tuki löytyy moderneista selaimista, myös mobiilipuolella. Javan tuki mobiilipuolella on hyvin rajoittunutta, eikä esimerkiksi Android tai iOS-pohjaisista laitteista löydy tukea (Oracle, 2014). Nämä kattavat noin 93 % älypuhelinkäyttäjistä (Gartner, 2013).

Tämän lisäksi Javan mainetta ovat syöneet lukuisat tietoturvaongelmat ja nykyään suosituksena onkin Javan kytkeminen pois päältä (Kaspersky Lab, 2013). Esimerkiksi Danske Bank on luopunut Javan käytöstä kokonaan web-sivuillaan huonon käyttäjäpalautteen takia (Danske Bank, 2013). Haittaohjelmia on myös levitetty sivuilla olevien mainosten avulla käyttäen hyväksi Javan tietoturva-aukkoja (Threatpost, 2014). Tietoturvaongelmien laajuus on rantautunut myös suoraan selaimiin. Esimerkiksi Firefox-selain ei automaattisesti toista Java- tai Silverlight-sisältöä, vaan tämä on täysin käyttäjän kontrolloitavana (Mozilla Security Blog, 2013).

JavaScriptinkään maine ei selviä täysin puhtain paperein liittyen tietoturvaongelmiin. Aiemmista selain versioista on löydetty tietoturva-aukkoja, joissa on hyödynnetty selaimen virheellistä JavaScript tulkkausta, mikä on mahdollistanut muistiylivuotojen hyödyntämisen haitallisen koodin ajamiseen. (Daniel, et al., 2008; Mozilla, 2013).

JavaScriptiä on myös käytetty hyväksi suorittamaan haitallista koodia web-sivun kontekstissa, jos sivusto on tämän mahdollistanut huonoista ohjelmointitavoista johtuen. Nykyään selaimet käyttävät automaattisesti varotoimenpiteitä estääkseen kyseiset ongelmat esimerkiksi estämällä kutsujen tekemisen vieraaseen domainiin.

Edellä mainitut kohdat huomioon ottaen ei asiakaspääntoteutuksista puhuttaessa Javaa voida pitää tulevaisuuden kestävänä vaihtoehtona verrattuna JavaScriptiin. Java soveltuu tapauksiin, joissa voidaan olla aivan varmoja ajettavan sovelluksen ympäristöstä eli esimerkiksi Javan vaatima liitännäinen löytyy. Tätä ei voida Javan nykytilanteessa taata.

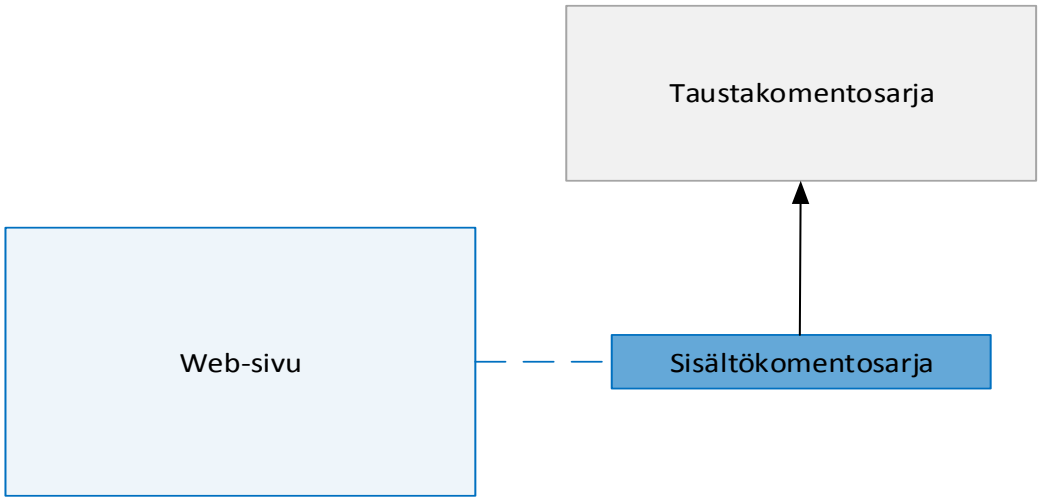
5.2 Toteutustavan valinta

Vaikkei JavaScript ominaisuuksiltaan olekaan paras ratkaisu, on se mainituista teknologioista kestävin tulevaisuuden suhteen ja linjassa kehityskulun kanssa. Tässä työssä tullaan selvittämään nykyiset ongelmat sekä minkälaisilla ratkaisuilla JavaScript-toteutus pystyisi haastamaan kolmannen osapuolen toteutukset. Tähän liittyen Chrome-selaimen laajennus tarjoaa työlle uuden näkökulman, sillä se pohjautuu HTML5:n ja JavaScriptin käyttöön: voidaan toteuttaa se mitä pystytään edellä mainituilla mutta tarvittaessa voidaan turvautua tarjottuun rajapintaan. Tämän avulla saadaan osviittaa siitä mikä nyt on mahdollista ja mikä kaipaa parannusta. Samalla mahdollisesti nähdään, miten JavaScriptin puuttuva toiminto on toteutettu Chromen rajapinnassa.

Laajennuksissa on myös se hyvä puoli, että niiden avulla esimerkiksi JavaScript koodin injektointi sivuille on helppoa. Normaalisti tämä vaatisi esimerkiksi välityspalvelimen (eng. Proxy) käytön, mikä ei puolestaan liity tämän työn aiheeseen. Koodin injektoimisella vältetään pyörän uudelleen keksiminen ja voidaan käyttää jo valmiita palveluita. Tällöin saadaan myös osviittaa siitä, kuinka suuren työmäärän pelkästään useamman monitorin tukeminen aiheuttaa.

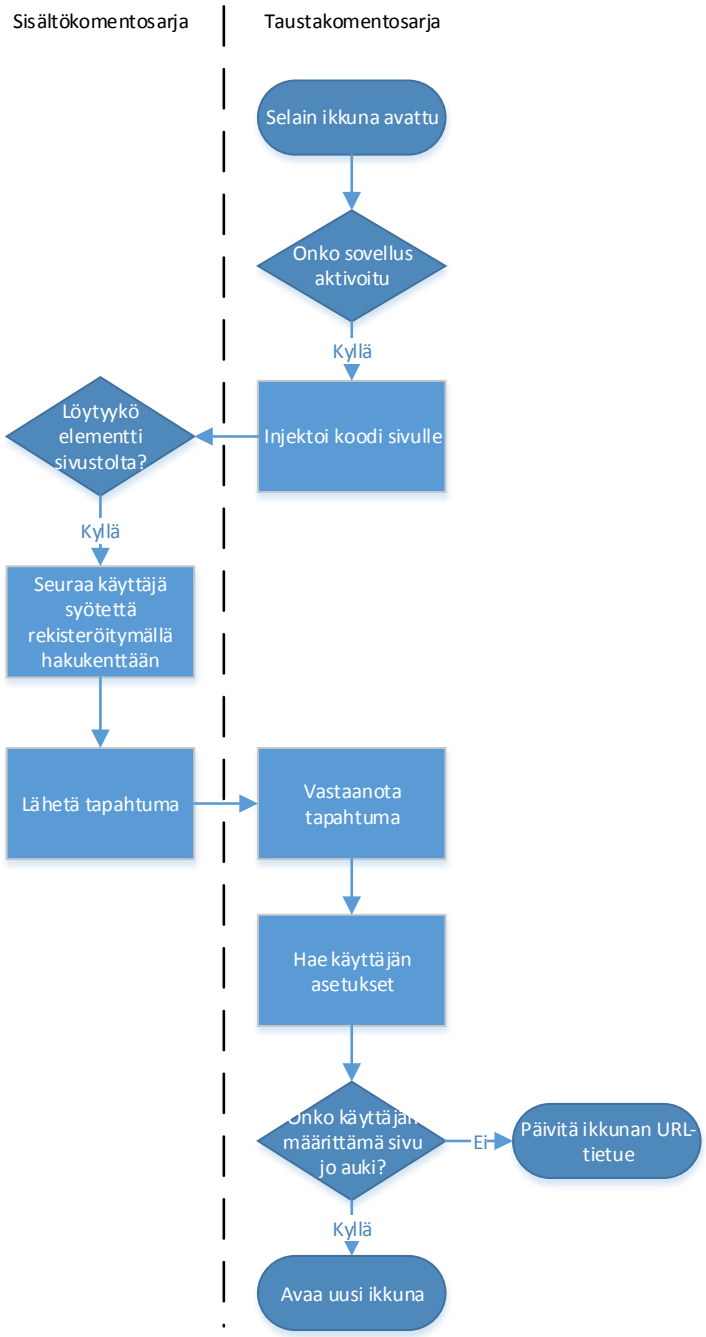
5.3 Toteutuksen kuvaus

Työssä tullaan toteuttamaan hakupalvelu, joka hakee eri lähteistä hakutuloksia eri monitoreihin. Hakulähteet ovat käyttäjän muokattavissa. Hakupalvelun pohjana toimii Googlen hakusivu, jonne injektoidaan JavaScript–koodia Chromelle tehdyn laajennuksen avulla. Yksinkertaisuudessaan käyttäjän syöttämä merkkijono poimitaan, asetetaan käyttäjän antamaan URL (Uniform Resource Identifier)-tietueeseen käyttäjän haluamaan kohtaan ja avataan ikkuna käyttäjän haluamaan monitoriin. Jos ikkuna on jo auki, päivitetään ikkunan sisältö. Sovellus skaalautuu aina kahdesta näytöstä äärettömään määrään monitoreita käyttöjärjestelmän ja selaimet asettamat rajoitteet kuitenkin huomioon ottaen.



Kuva 3. Sovelluksen arkkitehtuuri

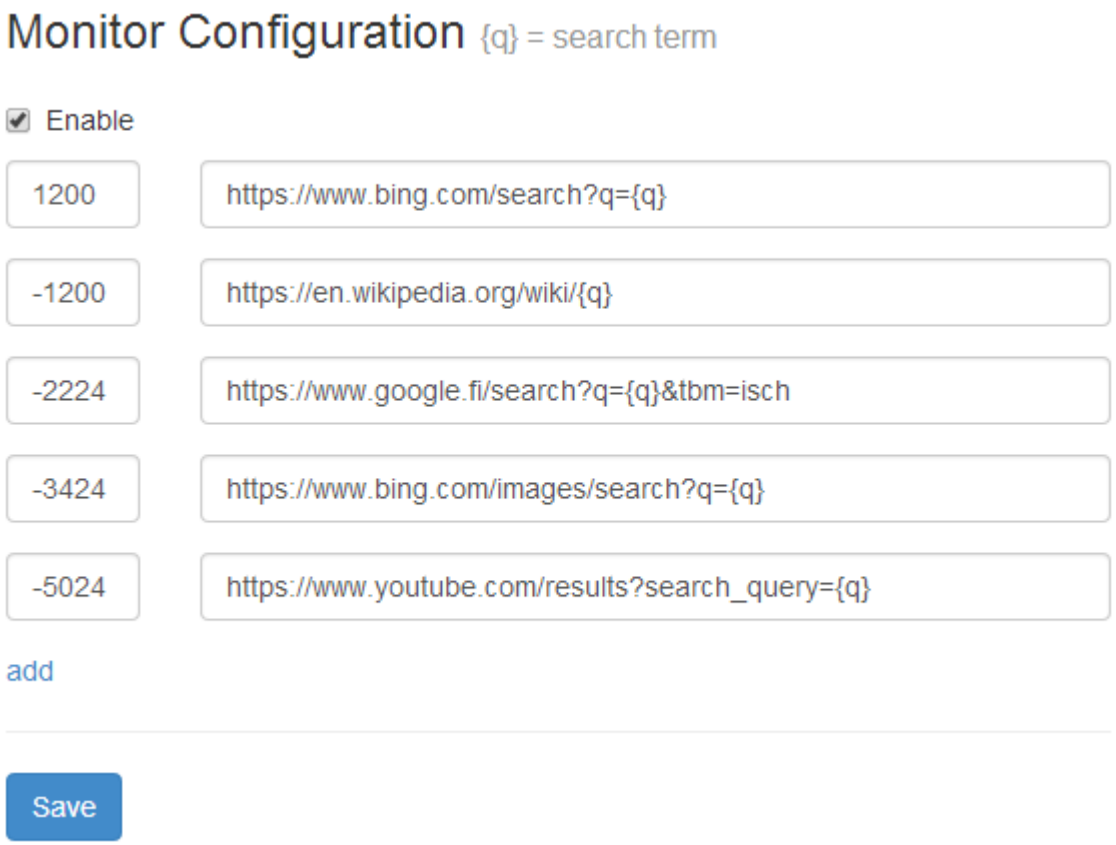
Kuvassa 3 on kuvattu sovelluksen arkkitehtuuri. Sisältökomentosarja on web-sivulle injektoitu koodi, joka toimii sivun kontekstissa. Sisältökomentosarjalla päästään esimerkiksi käsiksi sivun DOM-rajapintaan, jonka avulla voidaan muun muassa muokata sivuston rakennetta tai lukea sivuston sisältöä. Sisältökomentosarjassa hyödynnetään erityisesti DOM:in tapahtuma kuuntelijoita (eng. event listener), joiden avulla on mahdollista seurata esimerkiksi input-elementtien muutosta. Taustakomentosarja hallitsee sisältökomentosarjoja eli niputtaa kokonaisuuden yhteen ja sillä on pääsy Chromen-rajapintaan. Eri osat voivat kommunikoida keskenään lähettämällä tapahtumia (eng. event).



Kuva 4. Sovelluksen perustoiminnot sisältä vuokaavio.

Kuvassa 4 on esitetty sovelluksen vuokaavio, joka kuvaa komentosarjojen tehtävät. Taustakomentosarja hoitaa sisältökomentosarjan injektoimisen sivulle. Sisältökomentosarjan tehtävänä on seurata käyttäjäsyötettä rekisteröitymällä hakukenttään. Jos käyttäjäsyöte muuttuu, lähettää sisältökomentosarja taustakomentosarjalle tapahtuman,

joka sisältää käyttäjän antaman hakusanan. Taustakomentosarjalla on pääsy Chromen rajapintaan, jota hyödynnetään uusien ikkunoiden luontiin ja päivittämiseen. Taustakomentosarja hakee käyttäjän antamat asetukset ja avaa uusia ikkunoita tai päivittää jo avattuja ikkunoita. Tämän lisäksi taustakomentosarja myös valvoo ikkunoiden tilaa ja reagoi, jos käyttäjä sulkee ikkunan.

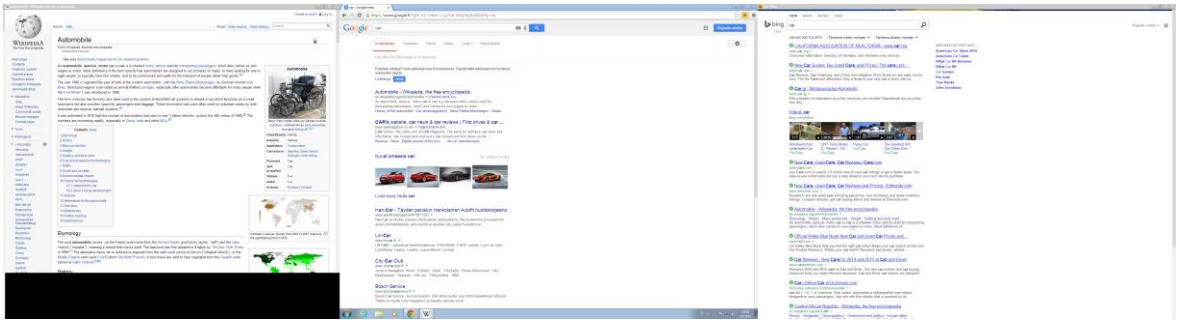


Kuva 5. Asetussivulle viisi näyttöä lisättynä

Sovellus tarjoaa myös asetukset sivun, jossa on mahdollista määrittää monitoriasetukset. Monitoriasetuksiin kuuluu URL-kenttä sekä ikkunan sijainti, joka määritetään päämonitorin suhteen vasemman puoleisten pikseleiden etäisyydellä. Tällä hetkellä sovellus tukee vain X-akselin suuntaan asetettuja monitoreita. Tuen lisääminen pystysuuntaan ei ole kuitenkaan hankalaa: sijainnin määrittämisessä otetaan huomioon myös Y-akseli.

Käyttöliittymän toteutuksessa hyödynnetään kirjallisuuskatsaus–osiossa mainittua

monitoreiden reunusten tarjoamia mahdollisuuksia, mikä auttaa käyttäjää hahmottamaan ja organisoimaan erilaista sisältöä eri monitoreihin. Eri hakulähteiden jako, esimerkiksi tietosanakirjahaku, per monitori tarjoaa ideaalisen lähestymistavan reunusten hyödyntämiseen.



Kuva 6. Car-hakusanalla tehty haku ja kolmen monitorin tulokset. Vasemmalla Wikipedia, keskellä Google ja oikealla Bing.

Chromen rajapinnasta käytettiin kahta ikkunanhallintaan liittyvää metodia: ikkunoiden luonti ja ikkunoiden päivitys. Tähän päädyttiin siksi, että näiden avulla on mahdollista avata uusi ikkuna toiseen monitoriin. Pelkästään JavaScriptiä käyttämällä Chrome antaa avata uuden ikkunan vain siihen monitoriin, jossa selain on aktiivisena. Rajapinnan avulla on myös mahdollista suurentaa ikkuna koko ruudun kokoiseksi. JavaScriptistä ei löydy tälle toimenpiteelle mitään funktiota, vaan ikkunan koko täytyy siirtää tiettyyn kohtaan ja syöttää sen koko manuaalisesti. Tämän lisäksi Chromen rajapinta antaa mahdollisuuden vaihtaa luotavan ikkunan tyyppiä, esimerkiksi normaalin popup-ikkunan lisäksi on mahdollista avata kokonaan uusi selainikkuna.

Rajapinnasta on myös hyödynnetty mahdollisuutta injektoida JavaSript-koodia sivustolle. Ilman rajapintaa täytyisi turvautua esimerkiksi välitypalvelimeen, joka hoitaisi koodin injektoinnin. Sovellusten asetusten tallentamiseen käytetään Chromen tarjoamaa tietokantarajapintaa, joka on hyvin samantyyppinen kuin HTML5:n tarjoama localStorage-rajapinta.

5.4 Tulokset

Työssä esitetyn mukaisen useamman monitorin hakupalvelun tuottaminen nykyisillä web-teknologioilla on mahdollista mutta siihen liittyy useita ongelmia. Suurin ongelma on se, ettei käyttäjän monitoritietoihin ole pääsyä ilman kolmannen osapuolen liitännäisiä. Lisäksi työssä tehty sovellus tukee vain staattisia ikkunointeja eli esimerkiksi, jos käyttäjä siirtää pääselainikkunan pois alkuperäisestä monitorista, ei tätä voida havaita ja asetukset täytyisi syöttää uudelleen.

Myöskin ikkunoiden hallinta jättää toivomisen varaan JavaScriptiä käytettäessä. Nykyisellään, jos ikkuna halutaan avata tietyssä monitorissa, täytyy tietää monitoreiden resoluutiot. Tämän avulla voidaan avata uusi ikkuna ja siirtää se oikeaan kohtaan pysty- ja leveysresoluutioita hyväksi käyttäen, sillä muuta vaihtoehtoa ei ole.

Ongelmia aiheuttaa myös jo nyt selainten erilainen JavaScript-tulkkaus. Esimerkiksi Chrome-selain ei anna web-sivun avata uutta ikkunaa aktiivisena olevan monitorin ulkopuolelle. Firefox-selaimella tämä puolestaan onnistuu. Nykyään myös selaimet estävät automaattisesti web-sivun luoda uusia ikkunoita, ellei käyttäjä tätä erityisesti halua.

Ikkunoiden suurentamisen koko ruudun kokoiseksi ja ikkunatyypin valinnan lisäksi ei Chromen rajapinta tarjonnut mitään sellaista mikä ei olisi myös JavaScriptillä toteutettavissa. Nämä ovat kuitenkin hyviä ominaisuuksia ja näiden toivoisi myös löytyvän tulevaisuuden toteutuksista. Ikkunatyypin valinnalla voidaan esimerkiksi hallita sovelluksen toiminnollisuutta ja käyttökohteita. Popup-tyyliset ikkunat tarjoavat mahdollisuuden avata niiden sisältämät linkit pääselainikkunaan, jolloin niiden rooli on tarjota avustavia palveluita pääselainikkunan tueksi. Esimerkiksi hakupalvelusta voidaan mielenkiintoiset tulokset poimia pääselainikkunaan uusiksi välilehdiksi. Normaalit ikkunatyypit taasen toimivat samalla tavalla kuin pääselainikkuna ja niiden linkit avautuvat aina kyseessä olevaan selainikkunaan. Chromen rajapinnan tarjoamat ratkaisut olisivat jo sellaisenaan tervetulleita ja auttavat toteuttamaan paremmin useammalle monitorille soveltuvia web-sovelluksia mutta lopullista ratkaisua ne eivät kuitenkaan tarjoa.

Taulukko 3. Yhteenveto löytyvistä ja puuttuvista ominaisuuksista.

	JavaScript	Chrome API
Ikkunoiden luonti, siirto, koon muuttaminen, sisällön päivitys		
Ikkunatyypin valinta		
Ikkunan automaattinen suurentaminen koko ruudun kokoiseksi		
Käyttäjän monitorikonfiguraation selvittäminen		
Ikkunan siirto suoraan tiettyyn monitoriin		
Ikkunajärjestyksen muutoksen huomiointi		

Taulukossa 3 on yhteenveto tärkeimmistä löytyvistä sekä puuttuvista ominaisuuksista tämän työn perusteella vastaten samalle toiseen tutkimuskysymykseen, mitä teknisiä mahdollisuuksia ja haasteita työn ratkaisu tarjoaa. Mukaan on otettu myös Chromen rajapinnan tuomat lisäominaisuudet, jotka koettiin olevan hyödyllisiä toteutusta tehtäessä. Ikkunoiden siirrossa on huomioitavaa se, että vaikka JavaScriptistä löytyy tähän rajapinta, ei esimerkiksi Chrome-selaimella ole mahdollista siirtää ikkunaa pois päämonitorista ilman pääsyä Chromen omaan rajapintaan.

Tuen lisääminen useammalle monitorille ei tämän työn perusteella vaadin kovinkaan suurta työpanosta. Asetussivu mukaan lukien esimerkkitoteutus koostuu noin 240 koodirivistä. Toteutus sisältää perustoiminnallisuuden ja pienillä muutoksilla, kuten lisäämällä tuen Y-akselin monitoreille saisi sovelluksesta jo huomattavasti käyttäjäystävällisemmän.

6 POHDINTA JA TULEVAISUUS

Useamman monitorin web-sovelluksia tuskin tulevat yleistymään lähitulevaisuudessa räjähdysmäisesti. Tekniset mahdollisuudet eivät ole nykyisellään riittäviä. Useammat web-sivut eivät edes kunnolla hyödynnä yhden näytön pinta-alaa mutta tämä on tosin lähihistoriassa ottanut huomattavan askeleen eteenpäin. Jotta näyttöpinta-alan hyödyntäminen useamman monitorin avulla tulisi ajankohtaiseksi yleisesti, pitäisi web-sivujen osata aluksi hyödyntää jo yhden näytön haasteet. Tämän lisäksi 4k-näytöt ja suuremmat näyttökoot tekevät tuloaan, jolloin tehokasta näyttöpinta-alaa on vieläkin enemmän käytössä. Toisaalta tutkimusten mukaan näyttöjen reunukset auttavat kategorisoimaan ei tyyppistä sisältöä, joten tätä voitaisiin jo hyödyntää nykyisissä toteutuksissa.

Käyttökohteeksi soveltuvatkin hyvin räätälöitävissä olevat palvelut, joissa käyttäjällä on jokin tarve ja motivaatio useamman monitorin hyödyntämiseen. Tällöin käyttäjä luultavasti jaksaa nähdä hieman vaivaa järjestelmän konfiguraatioon, eivätkä pienet käyttäjäystävällisyyskukkaset haittaa kokonaisuutta. Tätä lähestymistapaa tukee myös Robert Ball et al. (2005) tekemä tutkimus käyttäjätottumuksista useammalla monitorilla. Tutkimuksessa todetaan, että tehokäyttäjät pyhittävät tietyt monitorit tietyille ohjelmille, esimerkiksi sähköposti oikeanpuoleisimpaan monitoriin. Tehokäyttäjät siis haluavat kustomoida järjestelmän omien tarpeidensa mukaan. Tällaisia sovelluksia voisi olla juurikin työssä toteutettu hakupalvelu, joka tarjoaisi laajat kustomointimahdollisuudet. Sovelluksen käyttö vaikka kirjallisten lähteiden etsimiseen voisi tarjota riittävän motivaatiotarpeen.

Edellisessä kappaleessa mainittu tutkimus myös nostaa esiin ongelman, joka koskee enemmänkin valmiita useamman monitorin palveluita ilman konfigurointivaatimusta. Jos sivusto automaattisesti avaa uuden ikkunan, millä perusteella valitaan oikea monitori. Tutkimuksen mukaan osa käyttäjistä haluaa sähköpostin olevan jatkuvasti auki, jolloin sen päälle ilmestyvä web-sisältö voi saada aikaan kirosanojen sulosoinnun. Samalla täytyisi myös tarkemmin miettiä ikkunoiden luontiin liittyvää käytettävyyttä, ettei koeta samaa

kohtalo kuin popupien kanssa. Popupithan tulivat tunnetuiksi mainosten spämmilähteenä ja nykyään niiden käyttö on muussa käytössä kuihtunut lähes kokonaan selainten estäessä niiden luonnin automaattisesti. Täytyisi löytää jonkinlainen tasapaino automaation ja käyttäjäasetusten välillä. Useamman monitorin käyttö ei saisi olla liian vaivalloista mutta toisaalta sen ei pitäisi häiritä normaalikäyttöä. Tämä aiheuttaa jo omat haasteensa selaimille, jotka ovat lopullisessa vastuussa hyvän käyttökokemuksen luonnissa.

Työssä toteutettu hakupalvelu on vain yksi mahdollisen web-sisältöä hyödyntävä useamman monitorin käyttökohde. Taulukossa 4 on listattu suosituimmat web-sivustot. Viidestätoista suosituimmasta sivustosta yksitoista tarjoavat joko hakupalveluita tai sosiaalisen median palveluita. Näistä viisi tarjoavat toisena mainittua.

Taulukko 4. Suosituimmat sivustot (Alexa, 2014) ja niiden palvelumallit.

Sijoitus	Sivusto	Sivuston tyyppi
1	google.com	Hakupalvelu
2	facebook.com	Sosiaalinen media
3	youtube.com	Sosiaalinen media / hakupalvelu
4	yahoo.com	Hakupalvelu
5	baidu.com	Hakupalvelu
6	wikipedia.org	Hakupalvelu
7	qq.com	Erilaisia palveluita tarjoava kiinalainen sivusto.
8	linkedin.com	Sosiaalinen media
9	twitter.com	Sosiaalinen media
10	taobao.com	Kauppasivusto
11	live.com	Hakupalvelu
12	amazon.com	Kauppasivusto
13	sina.com.cn	Uutissivusto
14	google.co.in	Hakupalvelu
15	blogspot.com	Sosiaalinen media

Yksi mahdollinen web-sisältöä hyödyntämä useamman monitorin kohde voisi olla

sosiaaliseen mediaan liittyvä palvelu, sillä niiden suosio on kasvanut huomattavasti. Sovelluksen tarkoituksena voisi olla hakea eri sosiaalisen median sisältöä eri monitoreille, esimerkiksi facebookista ja twitteristä.

Käyttöpotentiaalin kannalta suurimmaksi haasteeksi sosiaalisten palveluiden kohdalla muodostuu niiden käyttäjätunnusvaatimuksen. Tämä vaikeuttaa ohjelman skaalautuvuutta useammalle monitorille. Esimerkiksi Facebook ja Twitter käytännössä vaativat käyttäjätunnukset, jotta niiden sisällöstä päästään nauttimaan. Toisaalta tämä tarjoaa myös mahdollisuuden tallentaa monitoriasetukset suoraan käyttäjätilien yhteyteen mutta, jos palvelua käytetään useammalta erilaiselta laitteelta, voidaan pärjätä myös asetusten tallentamisella selaimen muistiin. Tällöin asetukset ovat laitekohtaisia. Huonona puolena asetukset menetetään, jos selaimen välimuisti tyhjennetään.

Viihdekäytön ulkopuolella yksi kohde voisi olla Chrome-laajennusta hyväksi käyttävä projektinhallintatyökalu avaimet käteen periaatteella. Työkalu voisi esimerkiksi sumputtaa yhteen sprintin tilanteen, avoimet tehtävät ja virhetietokannan eri monitoreihin kaikki rakennettuna web-sovelluksen päälle. Sovellus olisi tällöin jokaisella alustalla kuitenkin saatavilla, mutta lisäpalikoilla saisi lisättyä uusia ominaisuuksia, kuten useamman monitorin tuen ja niin edelleen. Tämä olisi lähestymistapa, jota voitaisiin jo hyödyntää nykyisten teknologioiden puitteissa. Lisäpalikat ja laajennukset toisivat sovellukseen lisäominaisuuksia mahdollisesti parantaen käyttökokemusta mutta niiden puuttuminen ei kuitenkaan tuhoaisi koko sovelluksen päätarkoitusta.

Jotta useamman monitorin web-sovelluksista tulisi varteen otettava vaihtoehto, pitäisi alustariippumattomista ratkaisuista löytyä vielä tulos-osion taulukossa 3 esille nostetut ominaisuudet. Käyttäjän monitorikonfiguraation automaattinen selvittäminen on varsinkin käyttökokemuksen kannalta yksi tärkeimmistä puuttuvista ominaisuuksista. Nykyisellään myöskin ikkunan siirto haluttuun monitoriin tuottaa päänvaivaa, sillä tähän toimenpiteeseen tarvitaan monitoreiden resoluutiot. Parhaimmassa tapauksessa voisi halutun monitorin valita rajapinnan avulla suoraan ja siirtää haluttu sisältö siihen. Järjestelmän pitäisi myös jotenkin pystyä reagoimaan, jos käyttäjä muuttaa

monitoriasetuksiaan. Tästä pitäisi esimerkiksi saada jokin event eli tapahtuma, johon olisi mahdollista reagoida. Jotta nämä ominaisuudet saataisiin parhaiten hyödynnettävään muotoon, pitäisi ne määritellä HTML:n tai ECMAScriptin spesifikaatiossa. Tämä mahdollistaisi monipuolisten alustariippumattomien web-sovellusten luonnin kuitenkaan haittaamatta esimerkiksi mobiilikäyttöä.

Loppujen lopuksi työssä toteutettu hakupalvelu osoittautui ennakoitua paljon helpommaksi. Alun perin oli tarkoitus toteuttaa vain todella yksinkertainen valmiilla arvoilla toteutettu sovellus, jossa monitorikonfiguraatiot ja hakutulokset olisivat olleet valmiiksi määritelty. Helppouteen vaikutti se, että toteutus pohjautui lähes täysin valmiisiin palveluihin, joiden päälle lisättiin toiminnallisuutta. Muutaman ominaisuuden lisääminen kohentaisi käyttäjäkokemusta huomattavasti, jolloin voisi miettiä palikan laittamista julkiseen jakoon. Esimerkiksi käyttäjällä olisi mahdollisuus valita sivulta input-elementti, jonka yhteyteen olisi mahdollista konfiguroida monitoriasetuksen ja niiden näyttämät haut. Sovellus voisi myös jo valmiina sisältää yleisemmin käytetyt hakuoptiot. Näitä ominaisuuksia ei kuitenkaan lisätty tämän työn aikana, sillä niiden ei nähty vaikuttavan työn tuloksiin ja johtopäätöksiin.

Työn rajaus web-tekniikoiden suhteen olisi voinut olla paljon tiukempi. Ensimmäiset prototyyppikokeilut tuli tehtyä jo aivan työprosessin alkumetreillä ja siitä lähtien suuntauksena oli nimenomaan avoimuus sekä alustariippumattomuus eli lähinnä JavaScriptin hyödyntäminen. Tässä kohdassa työn tekniikat olisi voinut rajata suoraan koskemaan vain avoimia tekniikoita ja tutkia tämän lähestymistavan tuomia mahdollisuuksia paljon tarkemmin kuin nykyisessä työssä. Tällöin työssä olisi luultavasti päästy täysin samaan lopputulokseen mutta aihetta olisi voinut käsitellä paljon laajemmin.

7 JOHTOPÄÄTÖKSET

Työssä tehtiin hakupalvelu, joka jakaa eri monitoreihin hakulähteitä käyttäjän määritysten mukaan. Palvelu tehtiin käyttäen hyväksi jo olemassa olevia palveluita, eikä koko alustaa ja sen kaikkia komponentteja lähdetty tekemään alusta asti uusiksi. Toteuttavaksi teknologiaksi valittiin JavaScript, koska sen nähtiin olevan ainut nykykehityksen suuntainen teknologia ja tätä myötä tulevaisuuden kestävä. Palvelun pohjana toimii Googlen hakusivu, jonka toiminnallisuuteen päästiin käsiksi injektoimalla sivustolle JavaSript-koodia Chrome-selaimen laajennuksen avulla.

Hakupalvelun luonti onnistui mutta sen yhteydessä havaittiin useampi ongelma. Käyttäjän monitorikonfiguraation saaminen selville ilman kolmannen osapuolen liitännäisiä ei ole mahdollista käyttäen työssä esitettyjä menetelmiä. Jos halutaan toteuttaa alustariippumaton vaihtoehto, johtaa tämä siihen, että käyttäjän täytyy itse syöttää monitorintiedot, mikä ei ole kovinkaan käyttäjäystävällistä.

Myöskin selainten JavaScript-tulkkaus aiheuttaa ongelmia. Firefox selain antaa web-sivun avata ikkunan päämonitorin ulkopuolella mutta tämä ei onnistu Chrome-selaimella ilman pääsyä Chromen-selaimen tarjoamaan laajennusten käytössä olevaan rajapintaan. Jos työssä esitetyn hakupalvelun haluaisi toteuttaa täysin alustariippumattomasti, täytyisi Chrome-selaimen käyttäjien siirtää avautuneet ikkunat manuaalisesti toisiin monitoreihin. Tämän tyyppiset selainten poikkeavuudet rajoittavat huomattavasti alustariippumattoman vaihtoehdon mahdollisuuksia ja kyseenalaistavat sen järkevyyden esimerkiksi yllä mainitussa tapauksessa. Selaimet myös järjestään estävät popup-ikkunoiden automaattisen luonnin ellei käyttäjä tätä itse hyväksy.

Nykyisellään useamman monitorin web-sovellukset soveltuvat lähinnä valmiiksi räätälöityihin toteutuksiin, joissa tiedetään sovelluksen ympäristö. Tällöin voidaan turvautua kolmannen osapuolen liitännäisiin tai selainten tarjoamiin laajennuksiin, koska tiedetään, että nämä varmasti löytyvät kohdekoneista. Näin käyttäjäkokemuksesta saadaan edes jollain tavalla järkevä. Usean monitorin hyödyntämien soveltuukin parhaiten

lisäominaisuuksien lisäämisen sovellukseen, jolloin web-ympäristön hyvät puolet säilyvät, kuten saatavuus jokaisella alustalla, mutta web-sovelluksen pääidea ei kuitenkaan tuhoudu lisäosien puutteesta.

Jotta nämä ongelmat saataisiin ratkaistua alustariippumattomasti, pitäisi esimerkiksi seuraavassa HTML-versiossa olla määriteltynä rajapinta useamman monitorin hallintaan. Näin web-sivut voisivat kysellä selaimelta esimerkiksi käyttäjän monitorikonfiguraation valmiiksi ja siirtää uuden ikkunat tiettyyn monitoriin. Koska rajapinta olisi virallisesti määritelty, tukisivat myös selaimet tätä ominaisuutta, jolloin alustariippumattomien vaihtoehtojen luonti olisi mahdollista.

LÄHTEET

Alexa, 2014. *Alexa - Top Sites.* Saatavilla: http://www.alexa.com/topsites/global [Haettu 4 3 2014].

Anderson, J. A., Colvin, J. & Tobler, N., 2004. *Productivity and Multi-Screen Displays.*

Ball, R. & North, C., 2005. *Analysis of User Behavior on High-Resolution Tiled Displays.*

Ball, R. & North, C., 2005. *Effects of Tiled High-Resolution Display on Basic Visualization and Navigation Tasks.*

Daniel, M., Honoroff, J. & Miller, C., 2008. *Engineering Heap Overflow Exploits with JavaScript.* Saatavilla: http://www.usenix.org/legacy/event/woot08/tech/full_papers/daniel/daniel.pdf [Haettu 16 4 2014].

Danske Bank, 2013. *Danske Bankin verkkopankki uudistuu – Javasta luovutaan. S*aatavilla: http://www.danskebank.fi/fi-fi/tietoa-danske-bankista/media/Tiedotteet/Pages/20130129_VerkkopankkiJavastaluovutaan.aspx [Haettu 3 3 2014].

ECMA-262, 2011. *ECMAScript Language Specification.* Saatavilla: http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf [Haettu 14 4 2014].

Flanagan, D., 2011. *JavaScript: The Definitive Guide: Activate Your Web Pages.*

Gartner, 2013. *Gartner Says Smartphone Sales Grew 46.5 Percent in Second Quarter of 2013 and Exceeded Feature Phone Sales for First Time.* Saatavilla: http://www.gartner.com/newsroom/id/2573415 [Haettu 14 04 2014].

Ghiani, G., Paternò, F. & Santoro, C., 2012. *Push and Pull of Web User Interfaces in Multi-Device Environments.*

Google, 2013. *V8.* Saatavilla: https://code.google.com/p/v8/ [Haettu 3.3.2014 3 2014].

Google, 2014. *Extensions.* Saatavilla: https://developer.chrome.com/extensions/overview [Haettu 7 3 2014].

Huber, J. & Ding, Y., 2012. *Adapting web pages using graph partitioning algorithms for user-centric multi-device web browsing.*

Johanson, B., Ponnekanti, S., Sengupta, C. & Fox, A., 2001. *Multibrowsing: Moving Web Content across Multiple Displays.*

Kaspersky Lab, 2013. *Kaspersky Lab Report: Java under attack – the evolution of exploits in 2012-2013.* Saatavilla: http://media.kaspersky.com/pdf/Report_Java_under_attack_2012-2013.pdf [Haettu 3 3 2014].

Mozilla Developer Network, 2014. *Mozilla Developer Network.* Saatavilla: https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Media_queries [Haettu 19 3 2014].

Mozilla Security Blog, 2013. *Putting Users in Control of Plugins.* Saatavilla: https://blog.mozilla.org/security/2013/01/29/putting-users-in-control-of-plugins/ [Haettu 3 3 2014].

Mozilla, 2013. *Mozilla Security Center.* Saatavilla: https://www.mozilla.org/security/announce/2013/mfsa2013-12.html [Haettu 1 4 2014].

Mozilla, 2014. *SpiderMonkey.* Saatavilla: https://developer.mozilla.org/en-US/docs/Mozilla/Projects/SpiderMonkey [Haettu 3.3.2014 3 2014].

Oracle, 2013. *GraphicsDevice (Java Platform SE 7).* Saatavilla: http://docs.oracle.com/javase/7/docs/api/java/awt/GraphicsDevice.html [Haettu 4 3 2014].

Oracle, 2014. *Java.* Saatavilla: http://www.java.com/en/download/faq/java_mobile.xml [Haettu 14 04 2014].

Robertson, G. et al., 2005. *Large Display User Experience.*

StatCounter, 2014. *StatCounter*. Saatavilla: http://gs.statcounter.com/#desktop-browser-ww-monthly-201107-201402 [Haettu 5 4 2014].

Stegman, A., Ling, C. & Shehab, R., 2011. *A Comparison between Single and Dual Monitor Productivity and the Effects of Window Management Styles on Performance.*

Threatpost, 2014. *Malicious Ads on DailyMotion Redirect to Fake AV Attack.* Saatavilla: http://threatpost.com/malicious-ads-on-dailymotion-redirect-to-fake-av-attack/103494 [Haettu 3 3 2014].

W3C, 2013. *W3C JavaScript Web APIs.* Saatavilla: http://www.w3.org/standards/webdesign/script [Haettu 4 4 2014].

W3C, 2014. *HTML5.* Saatavilla: http://www.w3.org/TR/html5/ [Haettu 3 3 2014].

W3Schools, 2014. *The Window Object.* Saatavilla: http://www.w3schools.com/jsref/obj_window.asp [Haettu 4 3 2014].

W3Techs, 2014. *Usage of client-side programming languages for websites.* Saatavilla: http://w3techs.com/technologies/overview/client_side_language/all [Haettu 7 3 2014]