POHJOIS-KARJALAN AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma

Jaani Uniluoma

HTML5 JA UUDET WEB-TEKNOLOGIAT YRITYSOHJELMISTO-KEHITYKSESSÄ

Opinnäytetyö
Toukokuu 2012

POHJOIS-KARJALAN
AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Toukokuu 2012
Tietotekniikan koulutusohjelma

Karjalankatu 3
80200 JOENSUU
p. (013) 260 6800

Tekijä
Jaani Uniluoma

Nimeke
HTML5 ja uudet web-teknologiat yritysohjelmistokehityksessä

Toimeksiantaja
Lemonsoft Oy

Tiivistelmä

Opinnäytetyön tarkoituksena oli tutkia HTML5:en mahdollisuuksia yritysohjelmien kehittämiseen sekä suunnitella ja toteuttaa tuloksia havainnollistava esimerkkisovellus. Sovelluksen tuli sisältää yksinkertaisen asiakasrekisterin toiminnallisuudet. Toimeksiantajana toimi Lemonsoft Oy.

Työssä tutkittiin HTML5:en lisäksi myös muiden uusien web-teknologioiden tarjoamia mahdollisuuksia yritysohjelmistojen kehittämisen näkökulmasta sekä käsiteltiin web-sovelluskehitykseen liittyviä ongelmia ja niiden erilaisia ratkaisuvaihtoehtoja. Esimerkkisovellus toteutettiin käyttäen Microsoft Visual Web Developer 2010 -ohjelmistoa ja palvelinpuolen rajapinta toteutettiin käyttäen ASP.NET MVC 4 :ää.

Työn tuloksena syntyi tutkielma HTML5:stä ja uusista web-teknologioista sekä niiden soveltuvuudesta yritysohjelmiston toteutukseen. Opinnäytetyön osana tehtiin tuloksia havainnollistava esimerkkisovellus. Opinnäytetyö on tiivis johdatus web-sovellusten suunnitteluun ja ohjelmointiin yritysohjelmistojen näkökulmasta.

Kieli
suomi

Sivuja 46
Liitteet 0
Liitesivumäärä 0

Asiasanat
HTML5, web-sovelluskehitys, yritysohjelmisto

NORTH KARELIA
UNIVERSITY OF APPLIED SCIENCES

THESIS
May 2012
Degree Programme in Information Technology

FIN 80200 JOENSUU
FINLAND
Tel. 358-13-260 600

Author
Jaani Uniluoma

Title
HTML5 and New Web-Technologies in Line-of-Business Application Development

Commissioned by Lemonsoft Oy

Abstract

The purpose of this study was to investigate the capabilities of HTML5 for developing line-of-business application and to design and implement application to demo the results. The aim of the demo was to be simple customer data register application. The commission was made by Lemonsoft Oy.

The subjects of investigation were HTML5 and new web-technologies in view of developing line-of-business application. The study discussed the difficulties of web-application development and some solutions to them. The demo application was implemented using Microsoft Visual Web Developer 2010 and the server side interface using .NET MVC 4.

The results consist of the study of HTML5 and new web-technologies, their adaptabilities to line-of-business application developing and also customer data register demo application. The study is a concise introduction to web-application development in view of line-of-business application.

Language	Pages 46
Finnish	Appendices 0
	Pages of Appendices 0

Keywords
HTML5, web-application development, line-of-business application

Sisältö

Termit ja lyhenteet

RIA	Rich Internet Application eli rikas internet sovellus. Web-sovellus jonka ominaisuudet ja toiminnallisuus muistuttavat perinteisiä työpöytäsovelluksia.
HTML5	HTML:n uusin versio.
HTML	Hyper Text Markup Language eli hypertekstin merkintä/kuvaus –kieli.
XHTML	eXtensible HyperText Markup Language eli XML:n muotovaatimukset täyttävä, laajennettava hypertekstin merkintä/kuvaus –kieli.
CSS3	Cascading Style Sheets eli porrastetut tyyliarkit on internet-sivujen tyylin ja asetteluun tarkoitettu kieli.
line-of-business application	yritysohjelmisto (tekijän suomennos)
ERP	Enterprise Resource Planning eli toiminnanohjausjärjestelmä.
CRM	Customer Relationhtip Management eli asiakkuudenhallintajärjestelmä.
SCM	Supply Chain Management eli toimitusketjun hallintajärjestelmä.
REST	REpresentational State Transfer on arkkitehtuurimalli hajautetuille järjestelmille.
RPC	Remote Procedure Call eli proseduurien etäkutsu.
HTTP	HyperText Transfer Protocol eli hypertekstin siirtoprotokolla.
SOAP	Simple Object Access Protocol (SOAP) on hajautetussa järjestelmässä rakenteellisen ja tyypitetyn tiedon välittämiseen eri osapuolten kesken suunniteltu siirtoprotokolla.
WebGL	Web Graphics Library on JavaScript API interaktiivisen 3D grafiikan renderöintiin sitä tukevissa internet-selaimissa.

XML	Extensible Markup Language eli laajennettava kuvaus/merkintä –kieli.
W3C	World Wide Web Consortium yritysten ja yhteisöjen muodostama konsortio, joka kehittää ja ylläpitää WWW:n standardeja.
IETF	the Internet Engineering Task Force vastaa internet-protokollien standardoinnista.
WHATWG	Web Hypertext Application Technology Working Group on selain-valmistajien ja muiden uusien web-teknologioiden kehittämisestä kiinnostuneiden osapuolten yhteistyöryhmä.
origin	on palvelimen osoitteen, protokollan ja porttinumeron sisältävä käsite.
offline	tarkoittaa verkkoyhteydetöntä tilaa.
whitelist	on lista/rekisteri kohteista tai asioista, joille on annettu tietyt oikeudet.

1 Johdanto

Internet-selain suunniteltiin alun perin pääosin tekstisisältöisten dokumenttien selailuun ja katseluun. Nykypäivänä internet-selain toimii alkuperäisen tehtävänsä lisäksi ohjelmistoalustana web-sovelluksille joiden toteuttamiseen käytetään useasti selainliitännäisiä. Selainliitännäisiä tarvitaan sillä nykyiset web-sovelluskehitykseen liittyvät standardit ja määrittelyt eivät mahdollista monipuolisten RIA-sovellusten kehittämistä suoraan internet-selaimelle.

HTML5 on HTML-standardin uusin versio joka tulee laajentamaan mahdollisuuksia RIA-sovellusten tekemiseen suoraan internet-selaimelle. Tämän lisäksi kehitteillä on satoja erillisiä määrittelyitä joista useat ovat vasta luonnoksia (Internet Draft) [1].

Opinnäytetyö käsittelee HTML5:ttä ja uusia web-teknologioita yritysohjelmiston (line-of-business application) kehittämisen näkökulmasta. Englanninkielinen termi ”line-of-business application” on suomennettu yritysohjelmistoksi. Tässä työssä käsitellään myös muita määrittelyjä ja luonnoksia niiltä osin kuin ne ovat merkityksellisiä yritysohjelmiston kehittämisen näkökulmasta. Osana opinnäytetyöprosessia tehtiin esimerkkisovellus, joka sisältää yksinkertaisen asiakasrekisterin toiminnallisuudet.

2 Yritysohjelmisto

Yritysohjelmistolla tarkoitetaan tässä opinnäytetyössä line-of-business sovellusta. Tämä ohjelmisto on kriittisen tärkeä yrityksen toiminnan kannalta. Tällaisia ovat muun muassa ERP-, SCM- ja CRM-ohjelmat. Yhteistä näille ohjelmistoille ovat tietokannat ja moninaiset liitokset toisiinsa. [2.]

Yritysohjelmiston ansiosta yrityksen prosesseja saadaan tehostettua. Suurempi kerätyn tiedon määrä ja sen analysointi auttavat kohdentamaan resursseja oikein ja seuraamaan yrityksen kehitystä. Ongelmat voidaan tunnistaa ajoissa ja strategiset päätökset pystytään tekemään merkityksellisten tietojen pohjalta. Työntekijät, kuten asiakaspalvelijat tai markkinointihenkilöstö saavat tallennettua ja jaettua asiakkaiden tiedot ja toiveet reaaliajassa.

Suurille yrityksille tällaiset ohjelmat ovat elintärkeitä, sillä isojen tietomäärien hallittavuus asettaa suuria haasteita. Ilman monipuolisia ja luotettavia yritysohjelmia suuryritysten kasvu nykyiseen kokoonsa ei olisi todennäköisesti ollut mahdollista.

3 Internet ohjelmistokehitysympäristönä

Internet on kasvanut lyhyessä ajassa maailman tiedonvälityksen keskeisimmäksi välineeksi. Monet reaalimaailman asiat ovat saatavilla palveluina myös internetistä. Aluksi internet-sivut eivät olleet kovin monipuolisia ja sisälsivät hyvin vähän jos ollenkaan multimediaa kuten kuvia ja videoita. Nykyään kuvat ja videot internet-sivuilla ovat tavanomaisia sekä sivujen toiminnallisuudet ovat laajentuneet paljon internetin alkuajoista. Kehityssuunta on selkeästi kohti graafisempia ja monipuolisempia internet-sivuja. Esimerkiksi Open GL ES 2.0:een perustuva WebGL tuo mahdollisuuden ohjelmoida 3D-grafiikkaa internet-selaimille [3].

Web-ohjelmoijien kehittäessä yhä uusia tapoja käyttää olemassa olevaa arkkitehtuuria monipuolisemmin ja tehokkaammin, standardointi on kulkenut jäljessä pyrkien löytämään yhteistä säveltä eri selainvalmistajien kesken. Selainvalmistajien erilaiset näkemykset sekä eturistiriidat ovat aiheuttaneet sen, että web-standardien toteutukset ovat usein erilaisia eri selaimilla. Tämä on puolestaan vaikeuttanut kaikille selaimille suunnattujen sivustojen ja sovellusten kehittämistä sekä niiden ylläpitoa. Tätä yhteensopimattomuutta ovat pyrkineet paikkaamaan erilaiset JavaScript-koodikirjastot ja selainliitännäiset. Selainliitännäiset ovat tuoneet myös lisäominaisuuksia selaimelle ja mahdollistaneet RIA-sovellusten kehityksen selainten omia rajapintoja paremmin.

3.1 REST

Vuonna 2000 Roy Fielding esitteli tohtorin väitöskirjassaan “Architectural Styles and the Design of Network-based Software Architectures” arkkitehtuurityylin REST. Väitöskirjansa alussa hän käsittelee erilaisia arkkitehtuurityylejä ja niiden vahvuuksia sekä heikkouksia. Näiden pohjalta Fielding kehittää väitöskirjassaan arkkitehtuurityylin ”Representational state transfer” (REST).

REST koostuu rajoitteista jotka se asettaa ohjelmistoarkkitehtuurille. Arkkitehtuurin joka noudattaa näitä rajoitteita voidaan sanoa olevan REST-arkkitehtuurityylin mukainen. REST:in ydin koostuu yhtenevästä rajapinnasta, asiakas-palvelin arkkitehtuurityylistä, tilattomuudesta, mahdollisuudesta välimuistin käyttöön ja resurssinkäsitteestä, joka on keskeinen REST:issä. [4.]

Suurin ero muihin arkkitehtuurityyleihin on kaikkien komponenttien välisten rajapintojen yhteneväisyys. Väitöskirjassaan Fielding kirjoittaa tämän parantavan komponenttien välisten toimintojen näkyvyyttä ja yksinkertaistavan arkkitehtuuria kokonaisuudessaan. [4.]

Asiakas-palvelin-arkkitehtuurityyli mahdollistaa molempien ohjelmien (asiakas ja palvelinohjelman) ja niiden komponenttien kehittämiseen toisistaan erillään. Tämä parantaa käyttöliittymien siirrettävyyttä eri alustojen välillä. [4.]

Tilattomuudella Fielding tarkoittaa palvelimen tilattomuutta. Palvelimen tulisi ymmärtää asiakas-ohjelman lähettämä pyyntö ainoastaan sen sisällön perusteella. Näin ollen palvelimen skaalautuvuus paranee, sillä sen ei tarvitse pitää muistissaan tietoa aikaisemmista pyynnöistä käsitelläkseen uusia pyyntöjä. Myös näkyvyys on parempi sillä pyynnön sisällön perusteella voidaan päätellä asiakasohjelman tila tarvitsematta tulkita palvelinohjelman tilaa tai tilatietoja, joita asiakasohjelman aikaisemmat pyynnöt ovat palvelimelle luoneet. [4.]

Välimuistin käytöllä Fielding tarkoittaa palvelinohjelman mahdollisuutta asettaa sen asiakasohjelmalle lähettämä vastaus ”välimuistiin tallennettavaksi” (cacheable) tai ”ei välimuistiin tallennettavaksi” (non-cacheable). Tällä tehostetaan verkon käyttöä. Toisaalta tämä vähentää luotettavuutta tilanteissa joissa välimuistiin tallennetun datan eroavaisuudet suoraan palvelimelta haettuun dataan aiheuttaa ongelmia. [4.]

Resurssilla REST:issä tarkoitetaan abstraktiota joka voi olla mitä tahansa informaatiota jonka voi nimetä. Resurssin esitys voi vaihdella, esimerkiksi ”viimeisin lukemani kirja” on resurssi jonka esitys vaihtuu luettuani uuden kirjan. Fieldingin mukaan tällä saavutetaan verkko-arkkitehtuurin keskeisimmät piirteet joita ovat:

1. Yleisyys
   Kun tiedon lähteitä ei erotella niiden tyypin tai toteutuksen mukaan.
2. Myöhäinen viittauksen sidonta sen esitykseen
   Neuvottelu pyynnön kohteen sisällöstä on mahdollista käydä pyynnön perusteella.
3. Viittaus käsitteeseen, ei sen esitykseen
   Esityksen muuttuessa ei tarvitse uusia viittausta. [4.]

Viime vuosina REST-arkkitehtuurityyli on kasvattanut suosiotaan. Tämä johtuu osaksi siitä, että REST-tyylin mukainen arkkitehtuuri on monissa tapauksissa helpompi toteuttaa ja toisaalta se on hyvin skaalautuva. Esimerkiksi Microsoft on tuonut mahdollisuuden toteuttaa WCF-service rajapinnat SOAP-protokollan lisäksi REST-tyylisesti hyödyntäen HTTP-protokollan metodeja. Yritysohjelmiston kannalta skaalautuvuus on useasti tärkeä ominaisuus erityisesti jos käyttäjien määrä voi kasvaa tai vaihdella nopeasti. [4.]

3.2 Yhden sivun sovellus

Käyttäjän näkökulmasta yksi suurimmista eroista perinteisen työpöytäsovelluksen ja internet-sivuston käyttöliittymissä on navigoinnissa. Työpöytäsovelluksen käyttöliittymä on usein ikkuna, joka sisältää useita ikkunoita sovelluksen eri toiminnallisuuksiin. Käyttäjä voi avata, sulkea, siirtää ja muuttaa ikkunoiden kokoa. Ikkunoita avataan työkalupalkista tai suoraan pikanäppäimillä.

Internet-sivuston navigointilogiikka on erilainen. Perinteisesti jokaisella sivulla on oma osoitteensa (URL). Lähettämällä pyynnön osoitteeseen sivu saadaan vastauksena. HTTP-protokolla sisältää metodit myös tallentamiseen, päivittämiseen ja poistamiseen ja se on tilaton eli protokolla ei itsessään sisällä määrittelyjä tilan ylläpitämisestä. Internet-sivustoilla navigointi tapahtuukin perinteisesti sivulta sivulle linkkien avulla jokaisen siirtymän aiheuttaessa uuden pyynnön jonka vastauksena uusi sivu saadaan. Jokaisen sivun kohdalla koko sivu ladataan uudelleen asiakasohjelmassa (User Agent). Tämä näkyy käyttäjällä useimmiten sivun välähdyksenä ja on häiritsevää erityisesti silloin, kun vain pieni osa sivun sisällöstä muuttuu.

Dynaamisemmat internet-sivustot teki mahdolliseksi XmlHttpRequest-olio ja sen jälkeen kehittynyt, sitä hyödyntävä ohjelmointitekniikka Ajax (Asynchronous JavaScript And Xml). Aluksi Microsoft toteutti Internet Explorer -selaimeensa XmlHttpRequest-olion, jonka jälkeen muut selainvalmistajat kiinnostuivat siitä ja alkoivat kehittää samankaltaista toiminnallisuutta selaimiinsa. Vuonna 2006 W3C aloitti XmlHttpRequest-olion standardisointiprosessin [5]. XmlHttpRequest-olio mahdollistaa HTTP-pyynnön lähettämisen ja vastauksen käsittelyn JavaScriptillä. Uusi data voidaan vastauksen käsittelyssä liittää osaksi DOM:ia (Document Object Model) ilman kokonaisen sivun latausta. Tätä tekniikkaa kutsutaan nimellä Ajax.

Internet-sivustojen dynaamisuus kasvoi ja useilla internet-sivustoilla pyrittiin välttämään koko sivun latausta Ajax-tekniikan avulla. Työpöytäsovelluksista tuttu käyttöliittymä, jossa navigoidaan yhden ikkunan sisällä avaten uusia ikkunoita sovelluksen eri osiin, oli nyt helpompi toteuttaa. Internet-sivustoilla ja web-sovelluksilla ei kuitenkaan nähty suurta innostusta suoraan kopioida perinteisien työpöytäsovelluksien käyttöliittymää mutta yhden sivun internet-sivustot ja web-sovellukset alkoivat saavuttaa suosiota. Niissä suurin osa toiminnoista tapahtuu yhdellä sivulla, eikä kokosivun uudelleen latausta tarvita. Tämä parantaa käytettävyyttä ja selkeyttää navigointia sovelluksessa. [6.]

3.3 Käyttöliittymä

Web-sovelluksen käyttöliittymä voi olla internet-sivujen tyylinen tai se on mahdollista koota työkalupalkeista ja ikkunoista perinteisten työpöytäsovellusten tapaan. Visuaalisen ilmeen helppo muokattavuus tekee käyttöliittymän suunnittelusta web-sovelluksen vahvuuden perinteisiin työpöytäsovelluksiin nähden. HTML-sivun tyylin saa vaihdettua ja elementtien sijaintia muutettua ilman suuria ponnisteluja. Sivun sisällön kasvaessa tämä tietysti vaikeutuu ja vaatii enemmän suunnittelua ja aikaa toteuttaa.

Yritysohjelman käyttöliittymä on perinteisesti painottanut informatiivisuutta ja selkeyttä. Usein hallitsevana värinä on harmaa ja käyttöliittymä on koostunut pääosin tekstikentistä ja painikkeista. Samat elementit löytyvät myös html-elementtien joukosta ja erilaisten lomaketietojen syöttämiseen HTML5 tuokin

useita uudistuksia. Tietojen syöttäminen ja käsittely, usein määrämuotoisena on keskeisessä roolissa yritysohjelmissa. Internet-sivut ovat muuttuneet visuaaliselta ilmeeltään monipuolisemmiksi ja näyttävämmiksi verraten perinteisiin tekstikenttiin ja painikkeisiin koostuvaan käyttöliittymään. Käyttäjät ovat tottuneet monipuolisempaan visuaaliseen ilmeeseen ja osaavat arvostaa sitä myös yritysohjelmistossa. Työssään yritysohjelmistoa käyttävä henkilö viettää käyttöliittymän parissa suuren osan työajastaan, jolloin käyttöliittymän suunnittelulla on merkitystä. Esimerkiksi SalesForcen käyttöliittymät ovat muokattavissa [7]. Käyttöliittymän muokattavuus tuo lisäarvoa yritysohjelmalle ja web-sovelluksessa sen toteuttaminen on monia muita ympäristöjä helpompaa.

Web-sovelluksen käyttöliittymän muokattavuus helpottaa sen sovittamista erikokoisille näytöille. HTML5 on kasvattanut suosiotaan erilaisilla kannettavilla laitteilla ja esimerkiksi ABI Research:in arvion mukaan kannettavia laitteita joissa on HTML5:ttä tukeva selain tulee olemaan vuonna 2016 2.1 biljoonaa kappaletta. [8.]

3.4 Kehitystyökalut

HTML-koodin tarkistamiseen on olemassa HTML-validaattorit, jotka tarkastavat ovatko HTML-sivut sääntöjen mukaisia. W3C:n validaattoria voi käyttää osoitteessa: http://validator.w3.org ja se hyväksyy syötteeksi paikallisen tiedoston koneelta, URL:in tai HTML-koodin suoraan leikkaamalla ja liimaamalla. [9, s. 67.]

Nykyisistä selaimista löytyy myös kehittäjille suunnattuja työkaluja (kuva 1 ja 2). Selaimiin voi myös asentaa kehittämiseen tarkoitettuja työkaluja lisäosina (add-on). Esimerkiksi Firefox-selaimelle löytyy Firebug-lisäosa joka auttaa web-sovelluksen tai sivuston virheenetsinnässä (debugging). [9, s. 36–37.]

Koodin kirjoittamiseen ja muokkaamiseen käy periaatteessa mikä tahansa tekstieditori mutta käytännössä on hyvä valita editori joka tukee hyvin HTML-syntaksia. Editoreihin on tulossa myös HTML5-tuki ja esimerkiksi opinnäytetyön yhteydessä kehitettävän esimerkkisovelluksen tekoon käytetty Microsoft Visual Web Developer 2010 Express sisältää tuen HTML5-syntaksille.

Selaimet toteuttavat uusia toiminnallisuuksia ja ominaisuuksia eri tahtiin. Tämän vuoksi web-sovelluskehitykseen kuuluu olennaisena osana selaintuen tarkistaminen käytetyille ominaisuuksille ja toiminnallisuuksille. Selainten valmiuksien tutkimiseen on kokeilun lisäksi myös sivustoja joille on taulukoihin koottu selainten tukemat ominaisuudet (esimerkiksi http://caniuse.com).

3.5 Valmiit komponentit

Siitä huolimatta, että HTML5-standardi ei ole vielä valmis on HTML5:en ominaisuuksia hyödyntäviä valmiita komponentteja jo markkinoilla. Komponentit ovat suurimmaksi osaksi käyttöliittymäkomponentteja ja muista web-sovelluksen toiminnallisuuksia toteuttavista osista puhutaankin yleensä koodikirjastoina.

Valmiita komponentteja ja komponenttivalmistajia:

- Wijmo (www.wijmo.com), Component One
- Kendo UI (http://www.kendoui.com/), Telerik
- Sencha (http://www.sencha.com/), Sencha Inc

3.6 Vanhempien selainten tukeminen

Tässä opinnäytetyössä käsitellään HTML5:en ja uusien web-teknologioitten tarjoamia mahdollisuuksia yritysohjelmiston toteuttamiseksi web-sovelluksena. Kaikkia ominaisuuksia ei välttämättä tueta vielä uusimmissa selainversioissa puhumattakaan vanhemmista selaimista. Web-sovelluksen suunnittelussa on tärkeää linjata mitä selaimia ja mitä selainversioita tuetaan. Mitä vanhempia selaimia halutaan tukea, sitä enemmän tarvitaan työtä ja koodia korvaamaan uudemmat toiminnallisuudet vanhoissa selaimissa. Tämä opinnäytetyö ei käsittele vanhempien selainversioiden tuen toteuttamista kuin lyhyesti tässä luvussa.

Vanhojen selainten tukemiseen on monia eri toimintatapoja joista tärkeimmät ja suositelluimmat ovat:

- Testaa selaimen tukea käyttämillesi uusille ominaisuuksille (feature detection)

- Käytä työtä helpottavia koodikirjastoja (esim. Modernizer)
- Suunnittele ja määrittele selkeästi mitä selaimia ja selainversioita tuetaan (web-sovellus tai sivusto kannattaa testata jokaisella tuetulla selaimella ja selainversiolla)
- Suunnittele minkälaista varasisältöä tarjotaan (mahdollisimman samankaltainen toiminnallisuus vai karsitumpi versio)
- Älä tue Internet Explorer 6 –selainta (useissa lähteissä suositellaan jätettävän IE6 pois tuen piiristä, että siitä päästäisiin jo eroon)
  [10, s. 275–284; 9, s. 76–78]

4 HTML5

HTML (HyperText Markup Language) on hyperlinkkejä sisältävän dokumentin rakenteen merkitsemiseen kehitetty kuvauskieli joka mahdollistaa monisisältöisten dokumenttien julkaisun internetissä. Dokumentti rakennetaan tageista, jotka muodostetaan käyttäen alkutageja <*”tagin nimi”*> ja lopputageja </*”tagin nimi”*>. Keskelle kirjoitetaan tag:in sisältö jos siihen on tarkoitus liittää sisältöä. Lisäksi alkutagin hakasulkeiden sisälle taginnimen perään <*”tagin nimi” ”attribuutin nimi” = ”attribuutin arvo”*> on mahdollisuus määritellä elementille attribuutteja ja niille arvoja. [11.]

Internet-sovelluksen osana HTML toimii ohjelman sisäisen rakenteen kielenä. Se muodostaa sisällön perusrungon jota voi muokata CSS-tyylitiedostoilla ja tuoda toiminnallisuutta JavaScript-koodilla. HTML5:en myötä ulkoasuun liittyvät määrittelyt suositellaan tehtäväksi CSS:llä eikä HTML-elementeillä. HTML-elementit saavat HTML5:n myötä suuremman merkityksen dokumentin sisäisen semanttisen rakenteen kuvaajina ja merkitsijöinä sekä aiemmin selkeästi tyylillisten elementtien merkitykset ovat uudelleen määriteltyjä [9, s.127]. Tällaista selkeän erottelun kehityssuuntaa edustaa myös suosiota saanut unobtrusivejavascript eli ”huomaamattomaksi” JavaScriptiksi kutsuttu ratkaisu, johon kuuluu pyrkimys erottaa JavaScript JavaScript-tiedostoihin pois HTML-tiedostoista. Tämä selkeyttää lähdekoodia ja helpottaa ylläpidettävyyttä [12]. Tämä ei ole osa HTML5:ttä mutta on verrattavissa HTML5:en ratkaisuun erottaa tyyli ja ul-

koasu CSS:en tehtäväksi ja usein CSS-koodikin pyritään kirjoittamaan erilliseen CSS-tiedostoon.

4.1 Historia

HTML suunniteltiin ensisijaisesti hyperlinkkejä sisältävien tieteellisten dokumenttien kuvauskieleksi. Ensimmäinen versio HTML:sta julkaistiin 1990-luvun alussa. Se kehittyi ensimmäiset viisi vuotta CERN:in ja myöhemmin IETF:n johdolla. W3C:n perustamisen myötä HTML:ää ryhdyttiin jälleen päivittämään. Versioiden HTML 3.0 ja HTML 3.2 jälkeen seurasi nykyinen HTML 4.0, joka valmistui vuonna 1998. W3C:n jäsenten päätöksellä HTML:n kehittäminen pysäytettiin ja ryhdyttiin luomaan XML-kieleen pohjautuvaa XHTML:ää. [11.]

HTML:n kehityksen pysähtyessä selainvalmistajat jatkoivat kuitenkin selainten ohjelmointirajapintojen kehittämistä ja ne julkaistiin vuosien 1998–2004 välillä. Kiinnostus HTML:ää kohtaan heräsi uudelleen vuonna 2003 XForms-teknologian myötä, jolla oli tarkoitus luoda parempia ”seuraavan sukupolven” Web-lomakkeita ollen kuitenkin yhteensopiva HTML4:n kanssa. [11.]

W3C:ssa päätettiin testata HTML:n kehityksen avaamista vuonna 2004 jolloin aloitteentekijöinä toimivat Mozilla ja Opera. HTML:en jatkokehittämisen katsottiin kuitenkin olevan ristiriidassa aikaisemman suunnittelulinjauksen kanssa ja W3C päätti jatkaa XHTML:n kehittämistä. Tämän seurauksena Apple, Mozilla ja Opera perustivat WHATWG yhteistyöryhmän, jonka keskeisimpänä päämääränä oli kehittää uutta säilyttäen kuitenkin yhteensopivuuden vanhan kanssa ja tehdä määrittelyistä niin selkeitä, että saavutettaisiin täydellinen yhteensopivuus selainten kesken. Vuonna 2006 W3C ilmaisi kuitenkin aikaisemmasta poiketen olevansa kiinnostunut HTML5:en kehittämisestä ja perusti työryhmän vuonna 2007 joka alkoi toimia yhteistyössä WHATWG:n kanssa. [11.]

HTML5:stä julkaistaan WHATWG:n sekä W3C:n määrittely. WHATWG:n määrittely on elävämpi ja kaikkia sen sisältämiä HTML5:n ominaisuuksia ei tule W3C:n määrittelyyn. WHATWG kehittää HTML:ää inkrementaalisesti ja määrittelyn nimikin on muuttunut HTML5:stä ”HTML Living Standardiksi”. [13.]

4.2 HTML5-sivu ja dokumenttioliomalli

HTML-sivun perusrakenne koostuu <*html*>-, <*head*>- ja <*body*> -tageista. Näin on myös HTML5:ssä. Sivun on aikaisemmin aloittanut suhteellisen monimutkainen dokumenttityypin määrittely, joka on lyhennetty muotoon <*!DOCTYPE html*>. Tämän lisäksi merkistön valintaan käytetty <*meta*>-tag on nyt <*meta charset=utf-8*> [11]. Määrittelyn mukaan HTML5:en perusrunko on siis seuraavanlainen:

```
<!DOCTYPE html>
<html>
<head>
<title>Sample page</title>
</head>
<body>
<h1>Sample page</h1>
<p>This is a <a href="demo.html">simple</a> sample.</p>
<!-- this is a comment -->
</body>
</html>
```
[11]

Näiden muutosten takana on pyrkimys yksinkertaistaa ja samalla parantaa koodin luettavuutta. Uuden standardin myötä HTML-sivun lähdekoodi siistiytyy ja selkeytyy. Uudet rakenteelliset elementit vähentävät liiallista <*div*>-elementin käyttöä sekä auttavat selainta ymmärtämään paremmin sivun sisältöä. [11.]

Internet-selain muodostaa HTML-sivusta sisäisen dokumenttioliomallin (DOM), joka on HTML-sivua laajempi kokonaisuus. Tageista rakentuva HTML-sivu toimii ikään kuin käsikirjoituksena internet-selaimelle, jonka pohjalta se rakentaa varsinaisen dokumentin oliomallin. DOM mahdollistaa sivun dynaamisen muokkaamisen ja pääsyn dokumentin sisältöön skripteille tarjoamansa rajapinnan kautta. Nämä DOMin liittymät JavaScriptiin ja Javaan kuvataan Web IDL – liittymän määrittelykielellä jota käytetään kuvaamaan elementteihin liittyviä ominaisuuksia ja funktioita HTML5:en määrittelyssä. HTML5:essä DOM:in määrittely on tärkeässä roolissa sillä pyrkimyksenä on, että samanlainen HTML-sivu tuottaisi samanlaisen DOM:in eri selaimilla. DOMin perusmäärittely on Web DOM Core johon HTML5:en määrittelyssäkin viitataan. [14;9, s. 60–61.]

DOM on tietorakenteena puu ja se esitetään visuaalisesti usein puurakennekuvana. Useimmissa nykyisissä internet-selaimissa on kehittäjille tarkoitettu työkalu jonka avulla voi tarkastella internet-sivun DOM-rakennetta. Useasti puun solmut (DOM node) on merkitty vastaavilla HTML-tageilla.

Kuva 1. Kuvassa on W3C:n DOM info -sivuston DOMin rakenne Google Chromen kehittäjätyökalulla tarkasteltuna.

HTML-sivun juurielementti on *<html>*-elementti, jonka lapsielementeistä sivu rakentuu. DOM sisältää HTML-sivun tageilla merkityt elementit, niiden attribuutit ja attribuuttien arvot sekä alku- ja lopputagien väliin kirjoitetut sisällöt. [9, s.44.]

4.3 Dokumentin rakenne

Ennen HTML5:ttä dokumentin rakenteen jaot määriteltiin pääosin käyttäen * * -elementtiä. Asettamalla * * -elementin id-attribuutille kuvaava teksti on jaoille saatu nimet ja merkitys, kuten * * jolla on merkitty otsikkoalue.

HTML5-spesifikaation dokumentaation ylläpitäjä Ian Hickson kävi vuonna 2004 Google Indexin avulla läpi noin biljoona internet-sivua. Hän halusi selvittää minkälaisia rakenteita internet-sivut sisältävät ja osana selvitystä hän julkaisi lista-

uksen suositummista luokan nimistä (class-attribuuteille annetuista arvoista). Vuonna 2009 Operan MAMA-hakurobotti kävi läpi noin 2 miljoonaa satunnaisesti valittua URL:ia tarkistaen mitä class-attribuutteja sivujen elementeille oli annettu sekä noin 1,8 miljoonaa URL:ia tallentaen sivustojen id-attribuutit. Näiden tietojen pohjalta saatiin tietoa rakenteista joita internet-sivustoilla esiintyy. HTML5:n uudet rakenteelliset elementit ovat saaneet innoituksensa näistä tiedoista. [10, s.6.]

Standardin pyrkimyksenä ei ole ollut luoda täysin uusia dokumentin rakenteita pienen asiantuntijaryhmän sanelemana, vaan ennemmin mahdollistaa nykyiset käytössä olevat rakenteet suoraan omilla elementeillään. Näin ollen aiemmin otsikkoaluetta merkitsemään käytetty * * -elementti voidaan jatkossa korvata HTML5:n <*header*> -elementillä jne.

Web-sovelluksessa voi useasti olla myös elementtejä jotka on asetettu sitä varten, että niihin voidaan myöhemmin skriptissä asettaa jotakin. Usein tällaisille tyhjille elementeille on vaikea löytää vastinetta HTML-elementtien joukosta jolloin <*div*> -elementti voi olla paras ratkaisu. Kuitenkin olemassa olevia elementtejä kannattaa käyttää aina, kun on mahdollista jolloin HTML-koodista tulee luettavampaa. HTML5:en uusien elementtien määrittelyissä on otettu huomioon myös web-sovellukset, esimerkiksi <*article*> -elementillä voisi merkitä yksittäistä sähköpostia tai vaikka uutistenlukijan yksittäistä uutista [10, s. 38].

4.3.1 Header

Sivun otsikkoalueen rajaamiseen käytetään <*header*> -elementtiä. HTML5-spesifikaation mukaan sen tulisi sisältää johdannollista tai sivulla navigoimiseen liittyviä asioita. Elementti voi rajata sisälleen esimerkiksi otsikkoryhmän <*hgroup*> ja navigaatioelementin <*nav*>. Eräs huomionarvoinen seikka on, että <*header*> ei ole sama kuin <*head*>, joka on sivun perusrakenteeseen kuuluva elementti. Yleensä <*header*> vastaa internet-sivuilta löytyvää ”yläbanneria”. <*header*> voi esiintyä html-sivulla monta kertaa. Esimerkiksi <*article*>-elementille voi asettaa otsikon <*header*> -elementillä. [10, s.13.; 9, s.94.]

Yritysohjelmassa voi <*header*> -elementtiä käyttää esimerkiksi yrityksen sisäisen tiedotteen otsikon merkitsemiseen tai ohjelman sisäistä rakennetta merkitsemään.

4.3.2 Footer

Sivun alaviite-elementti joka spesifikaation mukaan sisältää tyypillisesti tietoja kirjoittajasta, linkkejä osioon liittyville sivuille tai tekijänoikeustietoja. Tämä elementti viittaa lähimpään jaottelevaan esi-isä-elementtiin joka käytännössä tarkoittaa sitä, että <*footer*> voi esiintyä sivulla useampaan kertaan sivunosien alaviitteinä sekä koko sivun alaviitteenä. [9.]

Elementtiä käytetään yleensä koko sivun alaviitteenä jolloin se auttaa hahmottamaan millä sivustolla ollaan [9]. Sitä voi käyttää myös esimerkiksi <*article*>-elementin sisällä jolloin siitä tulee artikkelin alaviite. Esimerkiksi sähköposti, joka on merkitty <*article*>-elementillä, voisi sisältää <*footer*> -elementin, jonka sisältönä olisi sähköpostin allekirjoitus.

4.3.3 Article

<*article*> -elementtiä on verrattu useissa lähteissä lehtiartikkeliin sen merkitystä kuvailtaessa. Itsenäiselle kokonaisuudelle jonka sisältö ei ole riippuvainen ympäristöstään vaan on ymmärrettävissä ilman sitä, käytetään <*article*> -elementtiä. Uutiskirjoitus, blogikirjoitus tai esimerkiksi sähköpostit web-sovelluksessa ovat esimerkkejä elementin käytöstä. [10, s.19.]

Yritysohjelmassa elementillä voidaan merkitä esimerkiksi yksittäisen asiakkaan tai tuotteen tietoja listauksessa jossa näkyy monta asiakasta tai tuotetta.

4.3.4 Section

<*section*> on aiheenmukaisesti jaotteleva elementti jonka sisältö on kiinteästi osa ympäröivää kokonaisuutta toisin kuin <*article*>. Elementtiä voidaan käyttää sivun, <*article*>-elementin tai esimerkiksi itsensä sisällä. Huomionarvoinen seikka on, että jokaisen <*section*> -elementin sisällä otsikkotasojen numerointi al-

kaa alusta eli jokaisella <*section*> -elementillä on oma ”otsikkoavaruutensa”. [9, s.98.]

Elementtiä voidaan käyttää yritysohjelmassa sen eri osa-alueiden jaotteluun. Esimerkiksi asiakastietojen käsittelyosio ja tuotetietojen hallintaan liittyvät toiminnallisuudet omiksi <*section*> elementeikseen. Aikaisemmin <*div*> -elementillä merkityt sivun rakenneosat voidaan usein merkitä HTML5:ssä <*section*> -elementillä.

4.3.5 Nav

<*nav*> -elementillä merkitään sivun sisällä liikkumiseen tarkoitettujen linkkien kokoelma jota voi verrata sisällysluetteloon. Elementtiä voidaan käyttää useampaan kertaan sivun eri osioiden sisällä. Tällöin on kyseessä vain sen osion ”sisällysluettelo”, ei koko sivun. Liiallista <*nav*> -elementin käyttöä tulee välttää, sillä ei ole tarkoituksenmukaista merkitä kaikkia linkkejä vaan ne, jotka liittyvät sivun tai web-sovelluksen sisällä navigointiin. [10, s.15–16.]

Elementin ideana on tehdä ero sivun navigointiosioiden ja muun sisällön välille. Tämä auttaa selaimia sivun esittämisessä. Esimerkiksi kannettavien laitteiden näytöillä joissa on yleensä vähän tilaa, selain voi esimerkiksi piilottaa <*nav*> -elementtien sisällön silloin kun käyttäjä ei halua navigoida sivulla. [9, s. 98.]

4.3.6 Aside

<*aside*> -elementtiä käytetään merkitsemään reunahuomautusta. Tällaisia ovat esimerkiksi uutisartikkeliin liittyvä lisätieto, mainos tai jotakin muuta pääsisältöä sivuava esitys. Usein on tarkoituksenmukaista asettaa pääsisällöstä erottuva tyyli <*aside*>-elementin sisällölle. Myös asettelulla johon elementin nimi viittaa saadaan <*aside*> -elementin sisältö erottumaan pääsisällöstä. [9, s.90.]

Internet-sivuilla näkee useasti sivunreunassa pystysuunnassa kulkevan elementin, joka sisältää linkkejä muille sivustoille, mainoksia tai muuta sivuston sisällöstä erottuvaa materiaalia. Tällaista elementtiä merkitään <*aside*> -elementillä. [10.]

Yritysohjelmassa <*aside*> -elementtiä voidaan käyttää lisätiedon esittämiseen.

4.3.7 Figure ja figcaption

<*figure*> -elementillä merkitään kuva ja siihen liittyvä kuvateksti. Myös kuvitus, kaavio tai jokin muu vastaavanlainen esitys voidaan merkitä tällä elementillä. HTML5 spesifikaatio antaa myös esimerkin <*figure*> -elementin käytöstä koodilistauksen ympärillä. [11.]

<*figure*> -elementti helpottaa tyylin antamista yhdelle kuvakokonaisuudelle, elementti voi sisältää myös useita kuvia. Kyse on kokonaisuudesta jonka voidaan ajatella kuuluvan yhteen ja jolle mahdollisesti halutaan yhtenevä tyyli tai koristelu (esimerkiksi reunaviiva tai taustaväri). Kuvatekstittömiä kuvia, pieniä kuvakkeita tai merkitykseltään koristuksellisia elementtejä ei ole tarkoituksenmukaista merkitä <*figure*> -elementillä. [9, s. 92.]

Kuvateksti <*figure*> -elementin sisällä merkitään <*figcaption*> -elementillä. Elementti sijoitetaan <*figure*> -elementin sisällön alkuun tai loppuun. [9, s. 90–91]

Web-sovelluksessa elementtiä voidaan käyttää merkitsemään esimerkiksi asiakaslomakkeessa asiakkaan kuva ja siihen liittyvä kuvateksti. Videot joiden yhteydessä on niitä kuvaava tai selventävä teksti ovat myös elementtejä, jotka voidaan sijoittaa <*figure*> -elementin sisälle.

4.4 Lomakkeet

Tietojenkerääminen käyttäjiltä tapahtuu internet-sivuilla yleensä lomakkeilla. HTML5 tuo uudistuksia lomakkeisiin liittyviin elementteihin. Tavoitteena on tarjota yksinkertainen ja yhtenäinen tapa luoda lomakkeita sekä niiden tarvitsemia toimintoja. Tietojen tarkistus ja erilaiset tiedon syöttämiseen tarkoitetut kentät ovat suurimmat HTML5:en tuomat muutokset lomakkeille. [11.]

Tämän luvun alaluvut ovat Range-alalukuun asti <*input*> -elementin type-attribuutiksi annettavia arvoja. Niillä voidaan tarkemmin tyypittää tekstikenttään tulevaa tietoa, jolloin selain voi tarjota käyttäjälle tiedonsyöttämistä helpottavia ratkaisuja. Tällaisia ovat muun muassa kalenteri päivämäärän syöttämistä varten tai esimerkiksi numeronäppäimistö kosketusnäyttöpuhelimen näytölle, kun on kysymyksessä puhelinnumeron syöttäminen. Kirjoittamishetkellä vain osa selaimista tukee näitä uudistuksia ja selainten type-attribuuttien perusteella tar-

joamien elementtien visuaalisen ilmeen muokkaaminen ei ole mahdollista. [10, s. 83, 86.]

Lomakkeiden erityyppisille <*input*> -elementeille yhtenäisenä uutena ominaisuutena on *placeholder*-attribuutti. Sen arvoksi voi asettaa tekstin, joka väistyy elementin aktivoituessa. Tämän tarkoituksena on tuoda helppo tapa asettaa esimerkkiteksti joka häviää, kun käyttäjä alkaa täyttää esimerkiksi tekstikenttää. Selainten toteutukset ovat vaihtelevia ja vielä joudutaan asettamaan jokin vaihtoehtoinen vihje niille joiden selaimet eivät tunne vielä *placeholder*-attribuuttia. [9, s. 156.]

4.4.1 Date, time, datetime, month ja week

Uudet, ajanilmauksiin liittyvät syötekenttien attribuutit muuttavat kentän sellaiseksi, että *date*, *time* tai *datetime*n tapauksessa määrittelyn mukaan selaimen täytyy muuttaa kenttään annettu syöte ISO 8601 – standardin mukaiseen muotoon (VVVV-KK-PP) tai kellonajan kanssa (VVVV-KK-PPT00:00). Jos input-elementin tyypin attribuutiksi on asetettu ”*month*” selain käyttää sisäisesti arvoja 1-12 kun taas *week* -arvolla käytetään muotoa (VVVV-W00), jossa 00 on viikon numero. Näin ollen selain pystyy tarjoamaan käyttäjälle esimerkiksi kalenterin päivämäärän syöttämistä varten. Toteutus on kirjoittamishetkellä todella vaihtelevaa. Esimerkiksi Opera tarjoaa visuaalisen kalenterin käyttäjälle monien selainten käyttäessä vain tavallista tekstikenttää. [9, s. 151.]

Kuva 2. Opera-selaimen <*input type=date* /> toteutus

Syötearvoja voi myös rajata *min*-ja *max*-attribuuteilla. Lisäksi *step*-attribuuttilla voi rajata käyttäjän antaman arvon tarkkuutta. Aikavyöhyke on mukana ainoastaan *datetime*a käytettäessä. Aika-arvo on silloin UTC-formaatissa ja se näkyy ajan sisäisessä esityksessä sen perässä olevana z-kirjaimena. Tämä tekee selaimille mahdolliseksi esittää aika käyttäjän paikallisajan mukaan. [9, s. 151; 10, s. 82–83.]

Aikaisemmin vuodelle, kuukaudelle ja päivälle on asetettu omat kentät html-sivulle. Tämä on tehnyt arvojen käsittelyn ja tarkistuksen helpommaksi, sillä yhden kentän arvon parsiminen ja tarkistaminen skriptillä on monimutkaisempaa sekä virhealttiimpaa. Nyt on mahdollista asettaa vain yksi päivämääräkenttä, jonka arvon käsittely ja validointi on helpottunut huomattavasti. Harmillista on, etteivät selainvalmistajat ole vielä toteuttaneet näitä HTML5:en määrittelyjä kuin osittain. [9, s. 151; 4, s. 82–83.]

Vanhemmissa selaimissa, jotka eivät tue HTML5:ttä kentät näkyvät tekstikenttinä. Tällainen vanhojen toteutusten rikkomattomuus onkin yksi HTML5:en suunnittelu periaatteista.[9, s. 151; 4, s. 82–83.]

4.4.2 Number

number-attribuutilla varustettu <*input*> -elementti on tarkoitettu luvuille ja sitä voi rajoittaa *min*-ja *max*-attribuuteilla sekä asettaa oletusarvon *value*-attribuutilla. Lisäksi voidaan asettaa ”askellus” *step*-attribuutilla. Esimerkiksi elementti <*input type=number step=5*> vaati käyttäjän antamaan arvoja viiden välein 5, 15, 20 … Kokonaislukujen kanssa elementti toimii hyvin mutta desimaalilukujen kanssa voi tulla ongelmia jos halutaan, että käyttäjä voi käyttää desimaalierottimena pilkkua. Sisäisessä esitysmuodossa käytetään pistettä ja selaimet voivat toteuttaa luvun näyttämisen eri tavoin. Esimerkiksi Chrome osaa suomenkielisessä versiossaan esittää desimaalierottimen pilkkuna mutta sitä ei voi selaimilta olettaa. [9, s. 146–147.]

Puhelinnumeroita ei ole tarkoitus syöttää *input*-elementtiin, johon on lisätty *number*-attribuutti sillä puhelinnumerot voivat sisältää myös ei-numeerisia

merkkejä kuten ”+”. Valitettavasti *number*-attribuutin asettaminen ei aiheuta kentän tekstin validointia numeerisena, vaan selaimet hyväksyvät kaiken tekstin. Selain ei missään vaiheessa muuta tekstisyötettä numeroksi vaan käsittelee sitä tekstinä. [4, s. 84–85]

4.4.3 Color

Kun *color*-määrettä käytetään input-elementin type-attribuuttina se tarkoittaa, että syötetietona annetaan värin RGB-koodi. Tämän tavoitteena on, että selaimet voisivat tarjota käyttäjälle visuaalisen värinvalintaan tarkoitetun elementin tai jonkin muun tiedon syöttämistä helpottavan elementin. Selainten toteutukset ovat kirjoitushetkellä vaihtelevia mutta ainakin Opera tarjoaa visuaalisen elementin värin valintaan. [9, s. 149–150]

Yritysohjelmassa värivalintaa voisi käyttää esimerkiksi halutunlaisen käyttöliittymän teeman värejä valitessa.

4.4.4 Url

Kun halutaan tekstikenttä johon on tarkoitus kirjoittaa internet-osoitteita, käytetään input-elementillä ”*url*” type-attribuutin arvona. Selainten tulee HTML5:en määrittelyn mukaan tarkistaa onko merkkijono URL. Kirjoittamishetkellä selainten toteutukset ovat erilaisia. Osa selaimista tarkistaa osoitteen tarkemmin kuin toiset. [9, s. 146.]

4.4.5 Email

Sähköpostiosoitteen syöttämiseen voi käyttää input-elementin type-attribuutin arvona ”*email*”. HTML5-määrittelyn mukaan selainten tulee tarkistaa voisiko annettu merkkijono olla sähköpostiosoite. Jos halutaan antaa monta sähköpostiosoitetta pilkuin erotettuna, voidaan input-elementille antaa attribuutti ”*multiple*”. Tarkoituksena on, että selaimet voisivat esimerkiksi avata käyttäjän sähköpostiosoitteiston josta käyttäjä saisi valita sähköpostiosoitteen.

Kirjoitushetkellä selaimilta ei vielä löydy tämänkaltaisia toteutuksia mutta kokeellinen Firefox Contacts add-on kerää yhteistietoja eri lähteistä, joita se sitten

tarjoaa käyttäjälle tämän syöttäessä tietoa input-kenttään jolle on asetettu type-attribuutiksi ”*email*”. [10, s. 82]

4.4.6 Tel

Puhelinnumerojen syöttämiseen tarkoitettu “*tel”* ei selaimissa validoi tekstikentän sisältöä. Tämä siksi, että puhelinnumeroille on olemassa monia erilaisia muotoja, jotka sisältävät myös muita kuin numeerisia merkkejä. Tarkoituksena on, että selaimet voivat tarjota puhelinnumeron syöttämiseen tarkoitetun elementin käyttäjälle. [10, s. 86]

4.4.7 Search

Hakuja varten tarkoitettu kenttä saadaan aikaiseksi kun input-kentän type-attribuutin arvoksi asetetaan *”search”*. Toistaiseksi HTML5 ei anna tarkempia suuntaviivoja miten selainten tulisi hakukenttää käsitellä. [9, s. 150]

Selainten toteutukset ovat vaihtelevia. Suurimmaksi osaksi tämä vaikuttaa vain kentän visuaaliseen ilmeeseen. Esimerkiksi Safari lisää ruksin hakukentän oikeaan reunaan josta kentän saa tyhjäksi halutessaan. [9, s. 151]

4.4.8 Range

Yleensä liukusäätimet, joiden tarkoituksena on valita jokin arvo tietyltä väliltä, on tehty käyttäen JavaScriptiä ja sivulle tuoduilla pienillä kuvakkeilla. Tämä sen vuoksi, ettei liukusäädintä ole aikaisemmissa HTML:n määrittelyissä ole ollut mukana. [10, s. 85]

HTML5:en myötä yksinkertaisen liukusäätimen saa yksinkertaisesti merkitsemällä input-elementin type-attribuutiksi ”*range*”. Lisäämällä *min*, *max* tai *step*-attribuutin voi säädellä miltä väliltä ja millaisin askelluksin arvoja voi antaa. Arvoa ei voi antaa liukuvana vaan se on aina kokonaisluku. [10, s.85;9, s.148.]

4.4.9 Pattern

Input-kentän validointiin tarkoitettu attribuutti *pattern*, on mahdollista asettaa input-kentille, johon on *type*-attribuutiksi asetettu *text, search, url, tel, email* tai *password.* Attribuutin arvoksi asetetaan säännöllinen lauseke (regular expression), jonka avulla selain validoi syötteen. Säännöllisen lausekkeen määrittely on samanlainen kuin JavaScript-kielessä. [10, s. 169.]

Ohjeena vaaditusta muodosta toimii Input-kentälle asetettava *title*-attribuutti, jota selain käyttää työkaluvihjeenä (tooltip) ja virheilmoituksen yhteydessä, kun vääränlainen syöte on annettu. Kaikki selaimet eivät vielä tue *pattern*-attribuuttia. [9, s. 169.]

Yritysohjelmissa käsitellään paljon määrämuotoista tietoa. Syötetietojen tarkistaminen on siksi monessa kohtaa mahdollista tehdä käyttäen säännöllisiä lausekkeita ja *pattern*-attribuuttia. Selaimessa tapahtuva validointi vähentää palvelimelle lähetettävän virheellisen tiedon määrää. Tämä nopeuttaa palvelimen suorittamaa tarkistusta ja käyttäjä saa tiedon virheellisistä syötteistä nopeammin.

HTML5 antaa mahdollisuuden säätää minkälaisia tarkistuksia selain eri kentille suorittaa. Näitä tarkistuksia voi säädellä asettamalla kentille attribuutteja tai käyttäen JavaScriptillä Constraint Validation API:a. [9, s. 170.]

4.4.10 Multiple

Tiedostojen lähetykseen palvelimelle käytetään <*input*>-elementin *type*-attribuuttia *file*. Määritysten vastaisesti selaimet ovat kuitenkin rajoittaneet tiedostojen määrän yhteen. HTML5:en myötä monien tiedostojen valitseminen on mahdollista lisäämällä *multiple*-attribuutti <*input*>-elementille. Attribuutti on sama kuin aiemmin <*email*>-elementin yhteydessä. [9, s. 173.]

Useiden tiedostojen lähettäminen palvelimelle on ennen HTML5:ttä toteutettu käyttämällä esimerkiksi selainliitännäisiä [4, s. 90] tai luomalla tarpeen mukaan uusia tiedoston lähetykseen tarkoitettuja kenttiä JavaScriptillä [9, s. 173].

Yritysohjelmistojen sisällä tallennetaan ja käsitellään paljon tiedostoja. Monen tiedoston valinta lähetystä varten on keskeinen käyttömukavuutta parantava ominaisuus ja HTML5 yksinkertaistaa tähän tarvittavaa koodia huomattavasti aikaisempaan nähden.

4.4.11 List

Yhdistelmäkenttä (combo box) luodaan HTML:ssä <*select*>-elementillä. Useasti halutaan valmiiden vaihtoehtojen lisäksi myös mahdollisuus syöttää oma teksti. Tällainen toiminnallisuus on aiemmin saatu aikaiseksi käyttäen JavaScript-koodia ja <*input*>- ja <*select*>-elementtiä. HTML5 tuo uudenlaisen ratkaisun. Asettamalla <*datalist*>-elementtille *id*-attribuutille arvo ja <*input*>-elementin *list*-attribuutille sama arvo, saadaan aikaiseksi yhdistelmäkenttä, johon on mahdollista syöttää myös oma teksti. [9, s. 88.]

Määrittely on herättänyt kysymyksen siitä miksi ei ole luotu täysin uutta elementtiä tätä toiminnallisuutta varten. Syynä tähän on se, että näin saavutetaan yhteensopivuus vanhemman määrittelyn kanssa. Jos käyttäjän selain ei tue HTML5:ttä niin hän näkee vain <*input*>-elementin ja voi syöttää edes jotain. Taaksepäin yhteensopivuus on osa HTML5:en suunnittelulinjauksia, poiketen aikaisemmin kehitteillä olleesta XHTML:stä (s. 16).

4.4.12 Meter ja Progress

Suureen arvon graafiseen kuvaamiseen voidaan käyttää <*meter*> -elementtiä. Asettamalla elementille attribuutit *min, max ja value* selain esittää arvon graafisena elementtinä joka useimmissa toteutuksissa on palkki, josta arvoa kuvaava osa on erotettu esimerkiksi eri värillä. Asetettavat arvot ovat liukulukuarvoja ja vain *value*-attribuutti, jolloin selain käyttää *min*-attribuutin arvona nollaa ja *max*-attribuutin arvona yhtä. [9, s. 108; 10, s. 94.]

Prosessin etenemistä voidaan kuvata <*progress*> -elementillä. Sille asetetaan vain *max* ja *value* attribuutit sillä ajatellaan, että prosessi lähtee aina liikkeelle nollasta. Selainten graafinen toteutus on hyvin samankaltainen <*progress*> ja <*meter*> elementtien kohdalla. Ero on selkeästi enemmän semanttisuudessa kuin toiminnallisuudessa. [10, s. 94.]

Työntekijä käyttää yritysohjelmistoa usein suuren osan työpäivästään, jolloin käytettävyyteen tulee kiinnittää huomiota. Työpöytäsovelluksista tutut prosessin etenemistä kuvaavat latauspalkit (progress bar) parantavat ohjelman käytettävyyttä. Käyttäjä pystyy paremmin arvioimaan prosessiin etenemistä ja käyttämään esimerkiksi tiedoston lataamisen kuluvan ajan johonkin hyödylliseen odottamisen sijasta.

5 HTML5 määrittelyt, luonnokset ja oheismäärittelyt

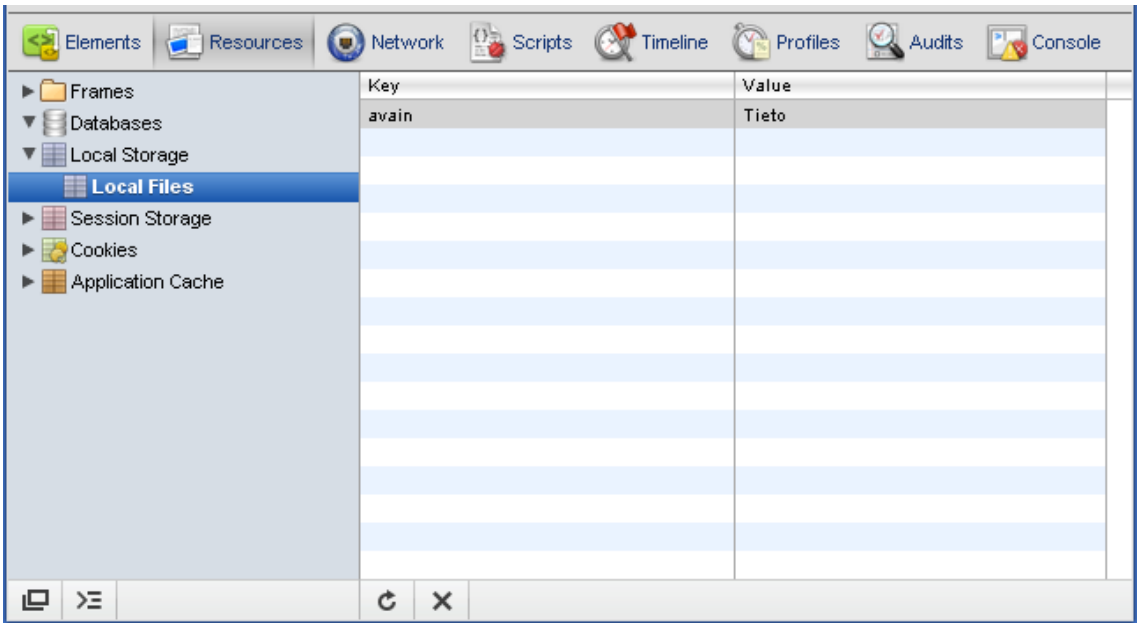
HTML5:stä on olemassa kaksi määrittelyä (specification), WHATWG:n ja W3C:n. Työnjako on karkeasti sellainen, että WHATWG tekee kehitystyötä ja W3C standardisoi ajallaan osan työn tuloksista W3C:n suositukseksi. Määrittelyiden luonnokset (internet draft) ovat teknisiä ja sisältävät paljon yksityiskohtaista tietoa sekä ovat rakenteeltaan osittain erilaisia. [9, s. 58.]

Monet uudet web-teknologiat yhdistetään HTML5:een vaikka ne ovatkin erillisiä luonnoksia tai määrittelyitä. Esimerkiksi SVG (Scalable Vector Graphics) on joissain yhteyksissä mainittu osana HTML5:ttä, vaikka se on täysin erillinen W3C:n määrittely ja jo 10 vuotta vanha. [10.]

Tässä luvussa esitellyt määrittelyt ja luonnokset eivät kaikki ole osa HTML5:ttä. Ne ovat uusia web-teknologioita, joista osa kuuluu HTML5:en määrittelyyn muiden ollessa omia erillisiä luonnoksia tai määrittelyitä.

5.1 Muisti ja tietokannat

Tietojen tallentamiseen selaimille on olemassa kolme eri määrittelyä: Web Storage, joka sisältää *sessionStoragen* ja *localStoragen*, Web SQL Database ja Indexed DB. Aikaisemmin selaimeen tallentaminen oli mahdollista käyttäen evästeitä (cookie), lomakkeiden piilokenttiä tai verkkosivun osoitteen kyselyosaa. [10, s. 169–171; 9, s. 230.]



Kuva 3. Tietokantoja ja muisteja voi tarkastella selainten mukana tulevilla työkaluilla. Kuvassa on tarkasteltu *localStoragea* Google Chromen kehittäjän työkaluilla.

Web Storage on kirjoittamishetkellä parhaiten tuettu ja sen rajapinta on muita yksinkertaisempi, mitä tulee kirjoitetun koodin määrään. Sen tarjoaman muistin koko on rajoitettu tavallisesti 5 megatavuun mutta useat selaimet mahdollistavat suurempienkin tietomäärien tallennuksen kysyttyään ensin käyttäjältä luvan tietokannan koon kasvattamiseen. [10, s. 171.]

Web Storageen tallennetaan tietoa avain/luku – pareina. API on todella yksinkertainen ja kuvaava:

```
localStorage.avain = ”Tieto”;          // kolme eri tapaa tallentaa
localStorage[”avain”] = ”Tieto”;
localStorage.setItem(”avain”, ”Tieto”);

localStorage.avain;                    // kolme eri tapaa hakea
localStorage[”avain”];
localStorage.getItem(”avain”);

delete localStorage.avain;             // kolme eri tapaa poistaa
delete localStorage[”avain”];
localStorage.removeItem(”avain”);
```

```
localStorage.clear();                    // koko muistin tyhjennys
```

Web Storage mahdollistaa kahden erilaisen muistin käytön. Muistit ovat: *localStorage,* joka on tarkoitettu pitkäaikaisemman tiedon säilytykseen ja *sessionStorage,* joka säilyttää tiedot selaimen muistissa selainikkunan sulkemiseen asti ja näin ollen jokaisella selainikkunalla on oma *sessionStorage* muistinsa. Yllä olevassa koodi esimerkissä *localStoragen* tilalle voidaan kirjoittaa *sessionStorage*, jolloin käytetään *sessionStoragea*. Nämä vaihtoehdot muistuttavat paljon evästeiden voimassaoloajan määrittelyä: jos evästeen voimassaoloaikaa ei aseteta, siitä tulee *session cookie*, jolloin sitä säilytetään selaimen muistissa vain selaimen sulkemiseen asti [19].

Taulukko 1. Web Storage – tiedon säilyvyys selaimen muistissa

	selainikkuna tai selain suljetaan	käyttäjä tai web-sovellus poistaa tiedon
localStorage	tiedot säilyvät	tiedot häviävät
sessionStorage	tiedot häviävät	tiedot häviävät

Web Storage sisältää myös tapahtumajärjestelmän joka ilmoittaa selainikkunoille niitä koskevan muistin muutoksista. Tapahtumat eivät koskaan muodostu siinä selainikkunassa, jossa *sessionStoragea* tai *localStoragea* muokkaamalla tapahtuma on aiheutettu. Tapahtumia voidaan käyttää muun muassa selainikkunoiden sisältämien tietojen synkronointiin. [10, s. 180.]

Tapahtuma sisältää kaikki muutokseen liittyvät tiedot: Avaimen nimi (key), vanha arvo (oldValue), uusi arvo (newValue), URL (url) ja muistin nimi (storageArea). Asettamalla skriptissä tapahtumalle kuuntelijan (eventListener), päästään tapahtuman muodostuessa esimerkiksi päivittämään selainikkunan sisältöä tapahtuman sisällön pohjalta. [10, s. 180.]

Tapahtumat toimivat *sessionStoragea* käytettäessä siten, että selainikkunan sisältämät *iframe*-elementit ja sivulta luodut pop-up ikkunat muodostavat tapah-

tuman, jos *sessionStoragea* muokataan. Käytettäessä *localStoragea* tapahtuma muodostuu kaikkiin avoimiin selainikkunoihin, jotka ovat ”samaa alkuperää” (origin) *localStoragea* muokanneen selainikkunan kanssa. [10, s.180.]

Web SQL Database mahdollistaa nimensä mukaisesti SQL-tietokannan internet-selaimeen. Määrittely ei ole kovin tarkka Web SQL -tietokannan koosta. Selainten toteutukset vaihtelevat ja tämä on otettava huomioon koodissa. Toinen huomioonotettava asia on tietokannan versio. Web SQL-tietokantaa luotaessa määritellään tietokannan versio. Samaa versiota täytyy käyttää myös tietokantaa avatessa. Tämä tarkoittaa käytännössä sitä, että web-sovelluksen täytyy tietokanta versiota päivittäessä päivittää myös käyttäjän selaimeen muodostamansa tietokanta uudempaan versioon tai ottaa huomioon eri tietokantaversiot tietokantoja käyttäessään. Tämän tekee vaikeammaksi vielä se, ettei web-sovelluksella ole suoraa mahdollisuutta tutkia versionumeroa, vaan se on otettava selville kokeilemalla. Esimerkiksi kirjoittamalla avaamiskutsut tietokannan eri versioille *try/catch* –lohkoihin löydetään se versio, joka ei aiheuta poikkeusta. [10, s. 184–186.]

Web SQL Database sisältää myös transaktiot ja jokainen SQL-komento annetaankin transaktion sisällä. Lisäksi, poiketen perinteisestä SQL-tietokannasta, yhteyttä Web SQL-tietokantaa ei koskaan suljeta ja yhteyden voi avata monta kertaa tämän aiheuttamatta poikkeusta. [10, s. 184.]

IndexedDB on objektitietokanta, josta on toistaiseksi käytössä vain selainvalmistajien omia toteutuksia, joita merkitään valmistajakohtaisilla alkuliitteillä (vendor-prefix). Tietokantaan tallennetaan objekteja avain/objekti – pareina ja tallennus tapahtuu transaktiona. [10, s. 199–200.]

Esimerkiksi web-sovellukset voivat käyttää IndexedDB:tä käyttömuistina ja siirtää sinne dataa pois DOMista. Toistaiseksi IndexedDB:stä ei ole vielä vakaata versiota.

Tietokannat ovat keskeinen osa yritysohjelmistoja ja ne ovat usein kooltaan suuria. Web Storage, Web SQL Database ja Indexed DB ovat aloituskooltaan pieniä. Tietokantaa suurennettaessa selain joutuu kysymään käyttäjältä lupaa lisätilan käyttöön ja tietokannan maksimikoossa on selainten toteutuksissa ero-

ja. Suurten tietomäärien tallennukseen täytyykin käyttää edelleen perinteisiä SQL-tietokantoja, jotka ovat paljon monipuolisempia ja turvallisempia kuin selaimiin integroidut tietokannat. Selainten tietokantojen tarkoitus ei olekaan korvata vaan mahdollistaa tietokannat pienemmin ominaisuuksin pienemmässä koossa suoraan internet-sivujen tai web-sovellusten käyttöön. Tämä selkeyttää web-sovelluksen arkkitehtuuria mahdollistaen kerroksellisen arkkitehtuurin konkreettisesti asiakas-ohjelmassa (client).

5.2 Yhteydettömyys

HTML5:en myötä web-sovelluksen on mahdollisuus toimia ilman internet-yhteyttä. Yhteydettömässä tilassa (offline) toimiva web-sovellus voi esimerkiksi tarjota osan toiminnallisuudestaan yhteyskatkosten aikana tai olla kokonaan suunniteltu käytettäväksi ilman internet-yhteyttä. Web-sovelluksen käynnistyminen voi usein viedä paljon aikaa. Käynnistymistä voidaan nopeuttaa määrittelemällä sovellus toimimaan osittain yhteydettömässä tilassa eli toisin sanoen säilyttämään osan tiedostoista selaimen muistissa. [10, s. 268; 9, s. 268.]

Yhteydettömässä tilassa toimimiseen web-sovellus tai internet-sivusto tarvitsee *manifest*-tiedoston, joka sisältää selaimelle tarkoitettuja tietoja. *Manifest*-tiedoston sisällön perusteella selain osaa tallentaa oikeat HTML, CSS, JavaScript ja muut tiedostot yhteydetöntä käyttöä varten sovellusvälimuistiin (application cache) sekä ohjata sivuston tai web-sovelluksen toimintaa. Sovellusvälimuistia voi tarkastella esimerkiksi selainten mukana tulevilla työkaluilla. Luvussa 5.1 esitetyssä kuvassa 3 näkyy sovellusvälimuisti (Application Cache). [10, s. 208.]

Käytettäessä *manifest*-tiedostoa siihen on viitattava HTML-tiedoston <*html*>-elementissä seuraavasti: <*html manifest=”/tiedostonnimi.appcache”*>. Yksinkertainen perusrakenne *manifest*-tiedostolle on seuraavanlainen:

CACHE MANIFEST	otsikko
CACHE:	mitä ladataan talteen
index.html	
FALLBACK:	vararesurssin määritely
skripti.js vara-skripti.js	
NETWORK:	näihin osoitteisiin lähtevät
*	pyynnöt menevät ensin internettiin
#version 3	kommentti

Tiedostossa määriteltävä ”CACHE:” ei ole välttämätön, sillä sovellusvälimuistiin ladattavien tiedostojen nimet tulevat oletuksena aina ensimmäisenä. Vararesurssin määrittäminen toimii siten, että selain lataa jälkimmäisen tiedoston (vara-skripti.js) sovellusvälimuistiin ja ottaa sen käyttöön jos ei saa ensimmäistä tiedostoa (skripti.js) ladattua palvelimelta. Verkkomäärittely on ”valkoinen lista” (whitelist), johon kirjoitetaan osoitteet joihin lähtevät pyynnöt menevät ensisijaisesti internettiin eikä sovellusvälimuistiin. Kommentti alkaa risuaidalla ja on tarkoitettu vain ihmisten luettavaksi, eikä selain käsittele kommentteja erikseen. Sen avulla voi kuitenkin pakottaa selaimen päivittämään sovellusvälimuistin sisältöä silloin, kun *manifest*-tiedoston sisältö ei muuten muutu. Selain vertaa palvelimelta löytyvää tiedostoa sovellusmuistissa olevaan ja jos ne eroavat toisistaan vaikkakin vain kommentin osalta, selain lataa tiedot uudelleen muistiinsa. [10, s.209–211; 9, s. 278–279.]

Selaimen ja palvelimen välinen kommunikointi etenee seuraavanlaisesti:

1. **Selain** pyytää palvelimelta index.html:n
2. **Palvelin** palauttaa index.html:n
3. **Selain** parsii ja pyytää kaikki tarvittavat lisäresurssit (kuvat, CSS, JS) ja *manifest*-tiedoston, johon oli viitattu index.html-tiedostossa
4. **Palvelin** palauttaa pyydetyt resurssit
5. **Selain** prosessoi *manifest*-tiedoston ja pyytää sen mukaan lisäresursseja (selain suorittaa pyynnöt vaikka olisi jo hakenut kohdassa 3 samoja resursseja)

6. **Palvelin** palauttaa pyydetyt resurssit
7. **Selain** sovellusvälimuisti on nyt päivitetty, ilmoittaa päivityksestä tapahtumalla
   [10, s. 215]

Sovellusvälimuistin käyttö etenee seuraavasti silloin, kun *manifest*-tiedosto palvelimella ja selaimessa ovat sisällöltään samat:

1. **Selain** pyytää palvelimelta index.html:n
2. **Selain** huomaa, että sillä on tiedosto välimuistissa
3. **Selain** parsii index.html-tiedoston ja palvelee kaikki sovellusvälimuistiin tallennetut tiedot paikallisesti
4. **Selain** pyytää *manifest*-tiedoston palvelimelta
5. **Palvelin** palauttaa koodin 304, *manifest*-tiedosto ei ole muuttunut
   [10, s. 216]

Sovellusvälimuistin ja *manifest*-tiedoston käyttö on hyödyllistä isoille web-sovelluksille. Sovelluksen käynnistymisaika ja käyttö nopeutuu, kun sovelluksen usein tarvitsemat tiedostot säilyvät selaimessa, eikä niitä tarvitse hakea verkon yli palvelimelta. [9, s. 274.]

Yritysohjelmistot ovat tyypillisesti laajoja ja web-sovelluksena ne hyötyvät HTML5:en tuomasta sovellusvälimuistista. Edellä mainittujen hyötyjen lisäksi sovellus voi esimerkiksi tallentaa käyttäjän tekemiä muutoksia väliaikaisesti selaimen tietokantaan (Web Storage, Web SQL Database tai IndexedDB) ja myöhemmin ottaa yhteyden ja päivittää tiedot yhdellä kertaa. Onhan mahdollista, että internet-yhteyden muodostaminen ei onnistu ja töitä pitäisi silti pystyä jatkamaan.

5.3 Yhteydet ja suorituskyky

Web Socket API:n tarkoituksena on mahdollistaa nopea ja jatkuva kaksisuuntainen viestintäyhteys web-sovellusten ja sivustojen käyttöön. Se ei kuulu HTML5-määrittelyyn mutta on merkittävä uusi webteknologia, joka tekee mahdolliseksi toteuttaa reaaliaikaisia sovelluksia. [10, s. 266.]

Server-Sent Events tarjoaa mahdollisuuden ”työntää” palvelimelta tietoa selaimelle. Se on yksisuuntainen ja soveltuu asiakasohjelman tietojen päivittämiseen ilman käyttäjän toimia, lukuun ottamatta ensimmäistä yhteyden avausta jonka täytyy tulla selaimelta. [9, s. 290.]

Web Workers mahdollistaa JavaScript-koodin ajamisen taustalla. Tämän tarkoituksena on parantaa käyttöliittymän reaktiivisuutta, koska paljon tehoja vievän koodinsuorituksen voi viedä taustalle. [9, s. 293.]

Tausta-ajo toimii siten, että luodaan *worker*-olio, jolle annetaan suoritettava JavaScript-koodi. *Worker*-oliolle voidaan lähettää dataa *postMessage*-funktiolla ja siltä voi vastaanottaa dataa asettamalle oliolle *onMessage*-funktion. [9, s. 294–295.]

Messaging API on W3C:n luonnos, jota vastaa WHATWG:n ”Cross document messaging” rajapintaa. Sen tarkoituksena on mahdollistaa HTML-sivujen välinen viestintä silloinkin, kun ne eivät ole samassa sivustossa, joka ei muuten olisi mahdollista johtuen ”saman alkuperän käytännöstä” (same origin policy). [9, s. 289.]

5.4 Multimedia ja grafiikka

HTML5 tuo mukanaan uusia median toistamiseen ja grafiikan tuottamiseen liittyviä elementtejä. Nämä elementit ovat <*canvas*>, <*video*> ja <*audio*>. [11]

Uusi elementti <*canvas*> on graafisten esitysten luomista varten ja se on todella monipuolinen. Siihen voi piirtää viivoja, tekstiä, kuvia (tiedostoista), erilaisia täytevärejä ja paljon muuta. Sen sisällön voi myös tallentaa tiedostoon. [10, s.162.]

Yritysohjelmistoissa tietoja esitetään useasti erilaisina kaavioina. Tällaiseen tehtävään <*canvas*>-elementti sopii hyvin.

Videoiden esittämiseen on aiemmin käytetty selainliitännäisiä. HTML5:en myötä videota ja ääntä on mahdollista upottaa suoraan HTML-sivulle. Kuitenkin erilaisten video- ja äänikodekkien käyttö selaimissa on johtanut siihen, että palvelimella, joudutaan säilyttämään sama video- tai äänitiedosto kahdessa eri muodossa: H.264 (Internet Explorer ja Safari) ja WEBM (Firefox, Chrome ja Opera).

WEBM on vapaalla lisenssillä toisin kuin H.264, jonka käytöstä joutuu maksamaan rojalteja. Ei lienekään yllätys mitkä kaksi selainta ovat jäseninä organisaatiossa, joka lisenssimaksuja H.264 käytöstä kerää. [10, s. 118]

Yritysohjelmistoissa ei pääasiallisesti käsitellä videoita tai ääntä. Kuitenkin erilaisten video- ja äänitallenteiden säilytykseen sekä jakeluun on toisinaan tarvetta. Esimerkkeinä konferenssiesitysten ja tuote-esittelyjen videotallenteet tai asiakaspalvelun tarve nauhoittaa puhelut asiakaspalvelun kehittämiseksi tai todistusaineistoksi käydystä keskustelusta.

5.5 Turvallisuus

Web-sovelluksen suunnittelu turvalliseksi on tärkeää. Erityisesti, kun kyseessä on yritysohjelmistot ja liiketoimintoihin liittyvää tietoa, joka päästessään vääriin käsiin voi aiheuttaa vahinkoa yritykselle. Tietomurrot ovat viime aikoina olleet jatkuvasti uutisotsikoissa ja huomionarvoista on se, että vuodetun, tuhotun tai muutellun datan lisäksi kärsii myös yrityksen maine.

OWASP (Open Web Application Security Project) on avoin yhteisö, joka pyrkii edistämään web-sovellusten turvallisuutta. Sen sivustolta (http://www.owasp.org/) löytyy myös tietoa HTML5:en ja uusien webteknologioiden turvallisuus ongelmista. Sivusto on hyvä lähtöpiste HTML5:ttä ja uusia webteknologioita käyttöön ottavalle sovelluskehittäjälle. [15.]

5.6 Valmistuminen

HTML5:en valmistumisajankohtaa W3C:n suositukseksi on arvailtu ja tiedusteltu työryhmän jäseniltä. Virallisesti HTML5:stä voi muodostua W3C:n suositus, kun on olemassa kaksi täysin valmista ja yhteensopivaa toteutusta [16].

Ongelmia muodostuu erityisesti niille jotka tahtoisivat viitata johonkin standardin kohtaan määrittelyissään jo nyt vielä, kun standardisointiprosessi on kesken. Yksi mahdollisuus on viitata standardin siihen versioon jota käyttää, ja tallentaa määrittelyyn viittaava internet-linkki osaksi määrittelydokumentaatiota. [9, s. 64.]

Jatkuvan kehityksen alla oleva WHATWG:n määrittelyt ja muut sadat eri luonnosvaiheessa olevat määrittelyt kuvastavat hyvin uusien webteknologioiden

kehityskulkua. Varmaa ja tarkasti sovittua päämäärää ei ole, vaan kyse on luovasta prosessista johon voivat liittyä kaikki uusien webteknologioiden kehityksestä kiinnostuneet yksilöt, yhteisöt ja yritykset. Ajatuksena on, että jokainen voi poimia riittävän ”kypsiksi” katsomansa teknologiat ja tekniikat suuresta valikoimasta silloin, kun parhaaksi katsoo vastakohtana sille, että kaikki odottaisivat koko ”sadon kypsymistä”.

6 Microsoft Silverlight vai HTML5

Silverlight on tarkoitettu monipuolisten sovellusten tekoon selainympäristössä, mutta Windows Phonen myötä se toimii myös sovelluskehyksenä puhelimelle kehitettäville sovelluksille (app). [17.]

HTML5:en myötä Microsoftin painopiste koskien monille eri alustoille suunnattuja sovelluksia on muuttunut. Silverlight tulee olemaan edelleen kehitysalusta Windows Phone sovelluksille ja se tarjoaa myös mahdollisuuksia media ja line-of-business sovelluksille, mutta monille eri alustoille suunnatuille sovelluksille, HTML tulee olemaan ensisijainen vaihtoehto. [18.]

HTML5 ei ole vielä valmis standardi ja selaimet eivät ole vielä lähelläkään HTML5:en tavoitetta yhteensopivuudesta eri selainten kesken. Selaimet ovat toteuttaneet uusia ominaisuuksia eri tahtiin ja osana kehitystyötä on huolehdittava siitä, että HTML5:ttä käyttävät web-sovellukset todella toimivat niillä selaimilla ja selainversioilla joille ne on suunnattu. [17.]

Silverlight on selkeämpi kokonaisuus, joka toimii jokaisessa selaimessa johon se on mahdollista liittää niin kuin on tarkoitettukin. Tämä helpottaa kehittämistä sillä paikkauksia ja selainversioiden tarkistuksia eri tarvita kuten HTML5:en kanssa. Lisäksi Silverlightin media ominaisuudet ovat HTML5 paremmat vaikkakin HTML5 sisältää uusia media elementtejä. [19.]

HTML5 ja Silverlight eivät sulje toisiaan pois. Molemmissa on hyviä ja huonoja ominaisuuksia sekä ominaisuuksia joita ei löydy toisesta. HTML5:en moniympäristöisyys on sen vahvin puoli ja samalla heikkous, sillä se ei ainakaan vielä toimi eri selaimilla ilman paikkauksia (polyfill). Silverlight on rajattu kokonaisuus, joka toimii hyvin ja ilman ylimääräistä testausta niissä ympäristöissä joissa sitä

on mahdollista käyttää. Rajoitetumpi erilaisten ympäristöjen ja alustojen määrä on Silverlightin keskeisin ongelma. Tämän lisäksi Silverlight ei ole vapaa teknologia, vaan Microsoftin omistama ja hallinnoima joka vähentää kilpailijoiden ja joidenkin laitevalmistajien innokkuutta tukea sitä ympäristöissään ja laitteillaan. [18, 19, 20.]

7 Esimerkkisovellus

Toimeksiannon mukaan esimerkkisovelluksen tarkoituksena oli esitellä HTML5:en ominaisuuksia. Ohjeistuksena oli tehdä sovellus jossa pystyy luomaan, päivittämään ja poistamaan (CRUD) asiakastietoja. Palvelinpään tuli olla toteutettu Microsoftin tekniikoilla.

7.1 Suunnittelu

Erilaisia vaihtoehtoja sovelluksen toteutukseen oli useita. Keskeisimmät vaihtoehdot olivat kuitenkin Microsoftin ASP.NET WebForms ja ASP.NET MVC. Toteutukseen valittiin ASP.NET MVC ja valinnan ratkaisi ASP.NET MVC 4 Beta versiosta löytyvä malli Single Page App, joka on AJAX-painotteinen ja sisältää monta JavaScript-kirjastoa, jotka helpottavat datankäsittelyä asiakasohjelmassa sekä datansiirtoa asiakasohjelman ja palvelimen välillä.

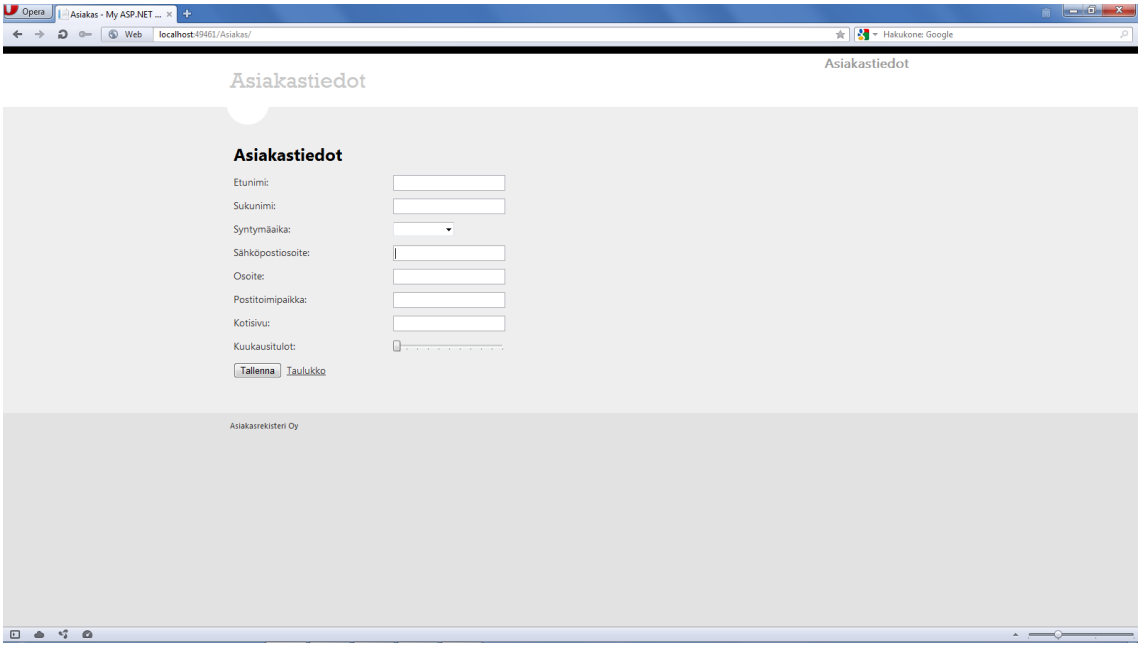
Mallin pohjalta toteutettiin yksinkertainen asiakastiedoille tarkoitettu CRUD-sovellus. Asiakastietojen käsittelyyn tarkoitettu lomake toteutettiin käyttäen HTML5:en uusia kenttiä ja kaavion piirtämiseen käytettiin <*canvas*>-elementtiä.

7.2 Toteutus

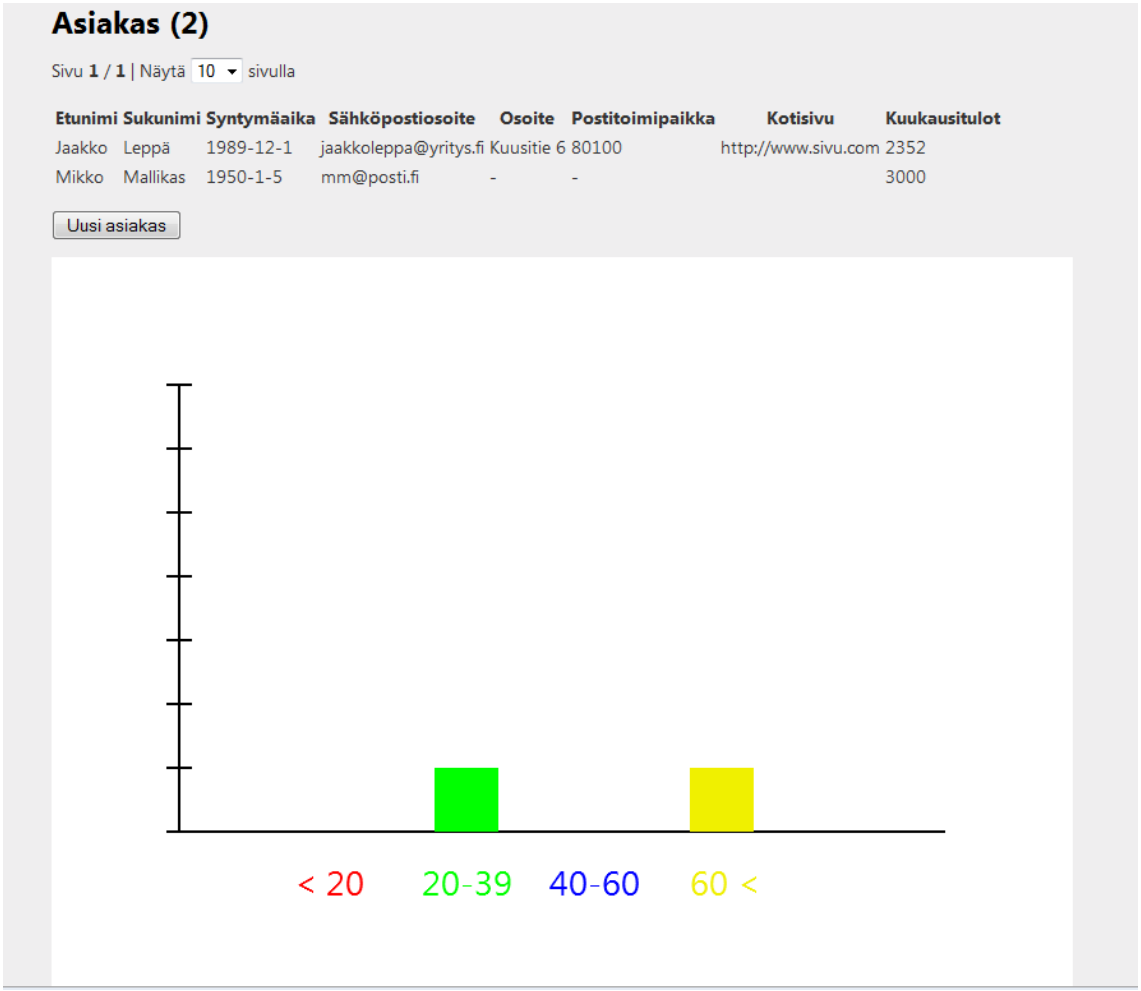
Sovelluksen toteuttaminen alkoi Microsoft Visual Web Developer Express 2010-ohjelman ja ASP.NET MVC 4 Beta:n asentamisella. Asennuksen jälkeen käynnistettiin Microsoft Visual Web Developer ja aloitettiin uusi MVC4 – Single Page App – projekti. Projektiin lisättiin uusi ”malliluokka” Asiakas, jota käyttämällä Entity Framework pystyi luomaan tietokannan web-sovelluksen käyttöön. Luokassa tuli yksilöivänä propertynä käyttää attribuuttia <key()> tai propertyn nimen tuli loppua ”Id”.

Tämän jälkeen luotiin uusi kontrolleri, jolle annettiin nimeksi ”AsiakasController”. Sen vastuualueeseen kuuluu hoitaa osoitteeseen: ”palvelimen osoite” + ”/asiakas” tulevat HTTP-pyynnöt.

Asiakas editori näkymästä, joka generoitiin automaattisesti kontrollerin luonnin yhteydessä, muokattiin kenttiä HTML5-lomakekentiksi. Lisäksi lisättiin *<canvas>*-elementti, jonka toiminnallisuudeksi tuli esittää asiakkaiden ikäjakaumaa pylväsdiagrammina.

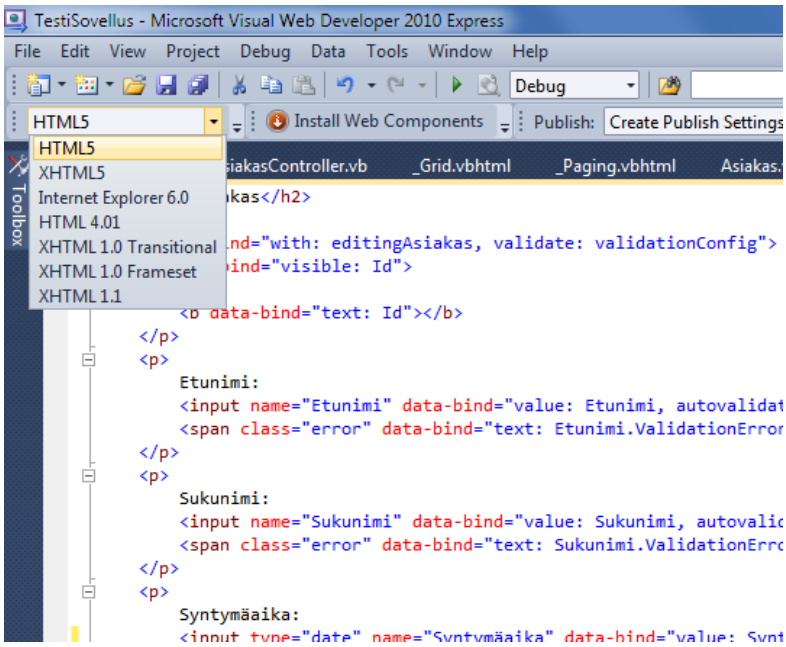


Kuva 4. Esimerkkisovelluksen asiakastietolomake

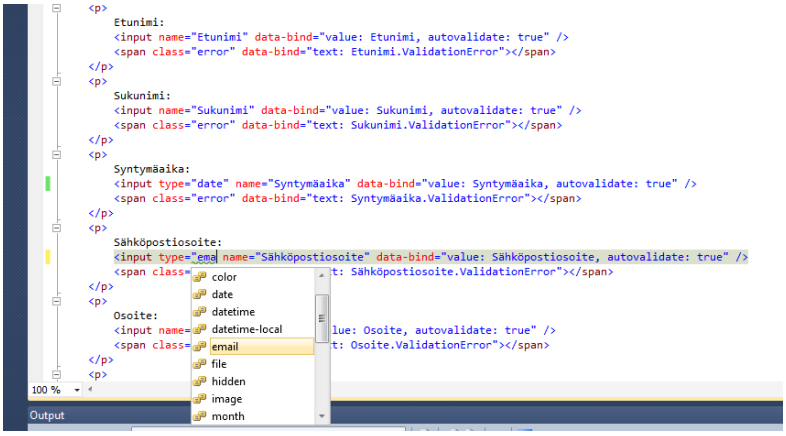


Kuva 5. Asiakkaiden ikäjakama esittävä pylväsdiagrammi joka on toteutettu <*canvas*>-elementillä.

Ohjelmasta löytyy tuki HTML5-syntaksille, mikä helpottaa ja nopeuttaa huomattavasti ohjelmointia.



Kuva 6. Tuki HTML5 syntaksille



Kuva 7. HTML5-syntaksi tuki käytössä

8 Tulokset

8.1 HTML5 ja uudet web-teknologiat

Tämän opinnäytetyön tarkoituksena oli tutkia HTML5:en ja uusien web-teknologioiden mahdollisuuksia yritysohjelmistokehityksessä. Opinnäytetyö täyttää sille asetetut tavoitteet painottaen käsittelyssään HTML5:ttä ja sovelluskehittäjän näkökulmaa.

Opinnäytetyöprosessissa saatiin selville, että HTML5 on moniympäristöisyytensä ja uusien ominaisuuksien johdosta mielenkiintoinen ja lupaava. Kuitenkin standardisointiprosessi on vielä kesken ja monet selaimet eivät vielä ole toteuttaneet kuin osan HTML5:en ominaisuuksista.

Poiketen muista ohjelmistokehitysympäristöistä HTML5 ja uudet web-teknologiat vaativat tarkempaa suunnittelua niin kohdeympäristönsä (eri selaimet ja selainversiot) kuin myös käytettävien ominaisuuksien suhteen. Lisäksi web-sovelluksissa käytetään usein monia eri koodikirjastoja kehitystyötä helpottamaan. Muun muassa datan siirto palvelimen ja asiakasohjelman välillä, erilaiset käyttöliittymäelementit sekä asiakasohjelman arkkitehtuuri ovat osa-alueita, joille on olemassa valmiita koodikirjastoja kehitystyötä helpottamaan ja nopeuttamaan. Monesti nämä ovat täysin erillisten osapuolien kehittämiä ja niiden yhteensovittaminen sekä testaaminen ovat keskeinen osa sovelluskehitystä.

Ilmaisuus ja sitoutumattomuus yhteen valmistajaan ovat myös etuja. HTML5:ttä kehittämään on lähtenyt kaikki suuret selainvalmistajat ja sitä pidetään yleisesti yhteisenä alustana tulevaisuuden web-sovelluksille.

8.2 Esimerkkisovellus

Opinnäytetyön osana kehitettiin yksinkertainen asiakastietojen tallennukseen tarkoitettu esimerkkisovellus joka esittelee HTML5:en uusia lomakekenttiä sekä <*canvas*>-elementin mahdollisuuksia. Esimerkkisovellus koostuu asiakastietojen tallennukseen tarkoitetusta lomakkeesta, asiakastietoja esittävästä taulukosta ja <*canvas*>-elementillä toteutetusta pylväsdiagrammista joka esittää asiakkaiden ikäjakauman.

Palvelinpuolen tekniikkana toimi tavoitteiden mukaisesti Microsoftin tekniikka. ASP.NET MVC 4 Beta on suositun internet-sivustojen ja web-sovellusten kehittämiseen tarkoitetun tekniikan uusin versio. Web-sovellus toteutettiin käyttäen Microsoft Visual Web Developer 2010 Express –ohjelmistoa.

Esimerkkisovellus täyttää sille asetetut tavoitteet ja uusinta Microsoftin tekniikkaa käyttäen se esittelee, miten Microsoft on alkanut tukea HTML5:ttä omissa ohjelmistokehitystuotteissaan (Visual Studio – tuoteperhe).

ASP.NET MVC 4 Beta :sta löytyvä Single Page App –malli, jota hyödyntäen esimerkkisovellus toteutettiin, sisälsi monia JavaScript-koodikirjastoja, jotka yhdessä muodostivat asiakasohjelman sisäisen arkkitehtuurin. Useiden koodikirjastojen käyttö web-sovelluksissa on yleistä ja myös HTML5:ttä sekä uusia web-teknologioita hyödyntäviä koodikirjastoja tullaan näkemään. Ne tulevat helpottamaan uusien ominaisuuksien käyttöönottoa.

Selainten erivaiheessa olevat HTML5:en toteutukset konkretisoituvat hyvin kun esimerkkisovellusta käyttää eri selaimilla ja selainversioilla. Eroja on myös HTML5:en toteutuksissa, kuten esimerkiksi ilmoituksissa, jotka selaimet esittävät kun kenttiin ei syötetä oikeanmuotoista tietoa. Tämä ei ole virhe, sillä HTML5 ei määrittele ilmoitusten ulkoasua.

Esimerkkisovellus näytti, että HTML5:ttä on mahdollista käyttää jo nyt. HTML5:en käyttöön ottaminen sovelluskehityksessä ja sen kannattavuus riippuvatkin paljolti lähtökohdista ja tavoitteista. Ensisijaisesti erilaisille alustoille ja ympäristöihin tarkoitettujen ohjelmistojen ja sovellusten toteuttamista kannattaa harkita toteutettavan web-sovelluksena ilman selainliitännäisiä.

9 Pohdinta

Opinnäytetyön tavoitteena oli tutkia HTML5:en ja uusien web-teknologioiden mahdollisuuksia yritysohjelmiston kehittämiseen web-sovelluksena sekä toteuttaa pieni esimerkkisovellus.

HTML5:en sisällön keskeisimmät asiat ja suunnittelun tavoitteet tulevat vahvasti esiin opinnäytetyössä. Historian läpikäynti luo pohjaa ymmärtää HTML5:en sisältöä ja sen suunnittelussa tehtyjä linjauksia. Yritysohjelmisto käsitteen määrittely opinnäytetyön alussa ja eri osa-alueiden käsittely sen näkökulmasta opinnäytetyössä avaavat aihetta tavoitteen mukaisesti. Yritysohjelmiston kehittämisen näkökulma olisi voinut olla mukana vielä vahvemmin avartaen aiheita käytännönläheisesti esimerkeillä olemassa olevista web-pohjaisista yritysohjelmistoista.

Web-sovelluskehitykseen liittyvien haasteiden ja kehitysympäristöön liittyvien asioiden esittely hahmottivat kokonaisuutta luoden pohjaa HTML5:en tarkem-

malle läpikäynnille. Näiden aiheiden käsittely jäi aihekohtaisesti katsoen lyhyeksi, mutta kokonaisuuden kannalta keskeisimmät asiat ovat esillä opinnäytetyössä. Käytännönläheisten esimerkkien määrä olisi voinut olla suurempi. Esimerkiksi olemassa olevien koodikirjastojen ja komponenttien tarkempi käsittely yritysohjelmiston kehittämisen näkökulmasta olisi auttanut hahmottamaan sovelluskehitystä käytännöntasolla.

Esimerkkisovelluksen toteutus onnistui teknologian valinnan osalta sillä ASP.NET MVC 4 beta edustaa Microsoftin uusimpia ratkaisuja web-sovelluskehityksen sekä internet-sivustojen kehityksen osalta. Tavoitteen mukaiset toiminnallisuudet löytyvät esimerkkisovelluksesta ja se toimii hyvin esimerkkinä HTML5:stä. Esimerkkisovellus olisi voinut olla laajempi ja sisältää enemmän opinnäytetyössä esiin tulleita HTML5:en ominaisuuksia ja uusia web-teknologioita kuten esimerkiksi muistit ja tietokannat.

Opinnäytetyö käsittelee aiheita kriittisesti ja tukeutuu tiedoissaan eri lähteisiin. Vaikka opinnäytetyön osana tehty esimerkkisovellus toteutettiin käyttäen Microsoftin ohjelmistoa, ei opinnäytetyö käytä pääasiallisena tiedonlähteenään Microsoftin sivustoja tai kirjallisuutta. Opinnäytetyön keskeisinä tiedonlähteinä toimivat W3C:n ja WHATWG:n määrittelyt sekä HTML5:ttä käsittelevä kirjallisuus, joka on opinnäytetyön luotettavuuden kannalta hyvä asia.

Prosessina opinnäytetyön tekeminen oli haasteellinen ennen kaikkea sen aiherajausten suhteen. Laajan aihealueen läpikäynti ja olennaisimpien asioiden löytäminen olivat suurin osa opinnäytetyöprosessista. Ammatillisesta näkökulmasta asioiden selvittäminen ja tutkiminen toivat uusia näkökulmia selaimesta sovelluskehitysalustana.

Opinnäytetyö sisältää useita aihealueita josta voisi aloittaa jatkotutkimuksen. Erityisesti web-sovellusten ja internet-sivustojen tietoturvallisuus liittyen HTML5:een ja uusiin web-teknologioihin olisi tärkeä sekä ajankohtainen aihe jatkotutkimukselle.

Lähteet

1. W3C, HTML5 for Business Apps. [Viitattu 3.5.2012]. Saatavissa: http://www.w3.org/2011/Talks/1015-wakanday/
2. SearchCIO, What is LOB (line-of-business). [Viitattu 3.5.2012]. Saatavissa: http://searchcio.techtarget.com/definition/LOB
3. Khronos, WebGL. [Viitattu 3.5.2012]. Saatavissa: www.khronos.org/webgl
4. Fielding Roy, Architectural Styles and the Design of Network-based Software Architectures [Viitattu 3.5.2012]. Saatavissa: http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm
5. W3C, XmlHttpRequest. [Viitattu 3.5.2012]. Saatavissa: http://www.w3.org/TR/XMLHttpRequest/XmlHttpRequest
6. Mahemoff Michael, Single Page Apps and the Future of History. [Viitattu 3.5.2012]. Saatavissa: http://www.infoq.com/presentations/Single-Page-Apps
7. SalesForce, Programmable User Interface. [Viitattu 3.5.2012]. Saatavissa: http://www.salesforce.com/platform/cloud-platform/programmable-ui.jsp
8. ABIResearch, 2.1. Billion HTML5 Browsers on Mobile Devices by 2016 says ABI Research [Viitattu 3.5.2012].Saatavissa: http://www.abiresearch.com/press/3730-2.1+Billion+HTML5+Browsers+on+Mobile+Devices+by+2016+say+ABI+Research
9. Korpela, Jukka. HTML5 – Uudet ominaisuudet. Porvoo 2011. 328 s. ISBN 978-951-0-38137-3
10. Lawson, Bruce & Sharp, Remy. Introducing HTML5 second edition. Berkeley 2012. 295 s. ISBN 978-0-321-78442-1
11. W3C. HTML5 Specification. [Viitattu 3.5.2012]. Saatavissa: http://www.w3.org/TR/html5/
12. Opera. Unobtrusive JavaScript [Viitattu 3.5.2012] Saatavissa: http://dev.opera.com/articles/view/unobtrusive-javascript/
13. WHATWG. Living Standard [Viitattu 3.5.2012]. Saatavissa: http://www.whatwg.org/specs/web-apps/current-work/multipage/
14. WHATWG. Web DOM Core. [Viitattu 3.5.2012]. Saatavissa: http://www.w3.org/DOM/
15. OWASP. Open Web Application Security Project [Viitattu 3.5.2012]. Saatavissa: https://www.owasp.org/index.php/Main_Page
16. WHATWG. FAQ. [Viitattu 3.5.2012]. Saatavissa: http://wiki.whatwg.org/wiki/FAQ#What.27s_this_I_hear_about_2022.3F
17. Microsoft. Silverlight. [Viitattu 3.5.2012]. Saatavissa: http://www.microsoft.com/silverlight/
18. Foley Mary-Jo. Microsoft – Our strategy with Silverlight has shifted.[Viitattu 3.5.2012]. Saatavissa: http://www.zdnet.com/blog/microsoft/microsoft-our-strategy-with-silverlight-has-shifted/7834
19. Jebaraj Daniel. Html or Silverlight [Viitattu 3.5.2012]. Saatavissa:

http://www.infoq.com/articles/Html5-or-Silverlight
20. Brion Davy. Why were going with html5 instead of silverlight. [Viitattu 3.5.2012]. Saatavissa: http://davybrion.com/blog/2011/03/why-were-going-with-html5-instead-of-silverlight/
21. Hanselman Scott. ShouldIUseHTML5OrSilverlighOneMansOpinion. [Viitattu 3.5.2012]. Saatavissa: http://www.hanselman.com/blog/ShouldIUseHTML5OrSilverlightOneMansOpinion.aspx