Iikka Manninen

HTML5:n hyödyntäminen verkkopohjaisen kuvagallerian käyttöliittymässä

HTML5:n hyödyntäminen verkkopohjaisen kuvagallerian käyttöliittymässä

Iikka Manninen
Opinnäytetyö
Kevät 2012
Tietojenkäsittelyn koulutusohjelma
Oulun seudun ammattikorkeakoulu

TIIVISTELMÄ

Oulun seudun ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma

Tekijä: Iikka Manninen
Opinnäytetyön nimi: HTML5:n hyödyntäminen verkkopohjaisen kuvagallerian käyttöliittymässä
Työn ohjaaja: Jouni Juntunen
Työn valmistumislukukausi ja -vuosi: Kevät 2012 Sivumäärä: 40

HTML5 on verkkosivujen rakenteen kuvailemiseen käytetyn HTML-kielen uusin versio. Opinnäytetyö tehtiin oululaiselle mainostoimisto Rapid Riverille, jonka verkossa toimiva kuvagalleria oli opinnäytetyön kehitystehtävien kohteena. Työn tarkoituksena oli tutustua HTML5-kieleen ja tutkia, kuinka sen tuomia uusia ominaisuuksia voisi hyödyntää toimeksiantajan kuvagalleriassa. HTML5-standardista rajattiin kehitystehtäviksi alkuvaiheessa neljä eri kokonaisuutta, joita olivat semanttiset elementit, raahaus ja pudotus -rajapinta, lomakkeet ja videoiden näyttämiseen tarkoitettu <video>-elementti. Työn tavoitteena oli luoda kuvagalleriaan HTML5-kieltä hyödyntävä käyttöliittymä niin, että mahdollisuus aikaisempien HTML-versioiden käyttöön kuitenkin säilytettäisiin.

Viitekehyksessä käsiteltiin lyhyesti HTML5:n historiaa ja sen suunnitteluperiaatteita sekä kerrottiin yleisesti kehitystehtävien kohteena olevien toimintojen ja rakenteiden käytöstä verkossa. Käytännön työn raportointi tehtiin pääosin HTML:n näkökulmasta, vaikka työhön kuului myös PHP- ja JavaScript-ohjelmointia. Lähteinä käytettiin pääasiassa verkkolähteitä, joista HTML-standardeja hallinnoivan W3C:n virallinen spesifikaatio oli selkeästi hyödyllisin.

Kehitystehtävien tulokset olivat vaihtelevia. Semanttisten elementtien havaittiin olevan suurimmaksi osaksi merkitykseltään sellaisia, etteivät ne soveltuneet kuvagalleriatyyppisen sisällön semanttiseen määrittelyyn. Osittain tähän oli syynä myös se, että W3C:n viralliset määritykset elementtien käyttötarkoituksille olivat vaikeasti tulkittavia tai sovellettavia. Raahaus ja pudotus-rajapintaan liittyvää tehtävää voitiin taas pitää onnistuneena, koska rajapinnan avulla saatiin toteutettua sama toiminnallisuus kuin JavaScriptillä ja jQuery-kirjastolla luodussa versiossa.

Lomakkeiden kohdalla uusien attribuuttien käytöstä oli nähtävissä selkeää hyötyä, sillä varsinkin syötteiden tarkastaminen oli HTML5:n avulla huomattavasti helpompi toteuttaa. Selaintuen vähäisyydestä johtuen suurin osa uusista syöte- eli input-kentistä jäi opinnäytetyössä käsittelemättä. Käyttöliittymässä käytettyjen input-kenttien osalta suurempia puutteita ei ilmennyt, vaikka niistä saavutettu hyöty jäikin vain syötteentarkistukseen liittyviin asioihin. Videoelementin suurimmat ongelmat olivat eri selainten vaihtelevassa videoformaattien tuessa, minkä johdosta kyseisen elementin käyttäminen kuvagalleriassa ei ole tällä hetkellä järkevää.

Kuvagallerian jatkokehittämistä ajatellen opinnäytetyön teon aikana nousi esiin erityisesti raahaustoiminnon käyttäminen tiedostojen lataamiseen käyttäjän tietokoneelta. Myös videoelementtiin liittyviin ongelmiin voi olla mahdollista löytää vaihtoehtoisia ratkaisuja esimerkiksi muuttamalla tiedostojen lataamisprosessia.

Asiasanat: HTML5, verkko-ohjelmointi, JavaScript

ABSTRACT

Oulu University of Applied Sciences
Tietojenkäsittelyn koulutusohjelma

Author: Iikka Manninen
Title of thesis: Using HTML5 in a web-based image gallery
Supervisor(s): Jouni Juntunen
Term and year when the thesis was submitted: Spring 2012 **Number of pages: 40**

HTML5 is the newest version of the HTML markup language that is used in creating the structure and the content of a web page. The commisioner of this thesis was Rapid River, which is an Oulu-based company that specializes in advertising. The aim of the thesis was to investigate the advantages of using HTML5 in the client's own web-based photo gallery. Four separate development tasks were selected from the HTML5 specification: semantic elements, drag and drop API, web forms and <video> element. The objective was then to create a HTML5 user interface with the possibility of falling back to earlier versions of HTML if preferred.

The context consisted of a section on the history of HTML5 and its design principles. The sections of each development task discussed the current usage of each structure or action on the web. Finally, the implementation process was then described in its own chapter with each development task allocated its own sub-section. In addition to HTML, PHP and JavaScript were also used during the implementation, but the thesis as whole concentrated on HTML. The source material that was used consisted mainly of W3C's official specification.

The results of the development tasks were varied. Semantic elements were discovered to be not suitable to a photo gallery type of application. Part of this was due to official W3C specification, where the usage guidelines of each element were too vague. Drag and drop API in turn did not differ from the earlier jQuery version; therefore, that task can be considered successful.

With forms a definite advantage could be observed from using them. Input sanitization in particular was considerably easier to implement with the aid of new attributes from HTML5. As a result of inadequate browser support most of the new input elements were not part of the final user interface. However, the elements that were implemented did not have any problems associated with them although the advantages of using them were limited to input sanitization. At the moment, the video element was not usable in its current state as most of the current browsers do not support the same codecs thus making the use of the tag impractical.

For future development purposes the drag and drop API could be used for uploading files from user's computer. Also the difficulties in using the video element could be possibly solved by modifying the photo gallery's current upload process.

Keywords: HTML5, web programming, JavaScript

SISÄLLYS

1 JOHDANTO

Vaikka IT-alan kehittyminen on useimmilla osa-alueilla erittäin nopeaa, ovat jotkin asiat kuitenkin syystä tai toisesta jääneet kehityksessä paikoilleen. Verkkosivujen luomisessa käytetty HTML-kuvauskieli (HyperText Markup Language) vietti pitkään hiljaiseloa vuonna 1999 julkaistun 4.01-version jälkeen, mutta 2000-luvun puolivälissä alkanut kehitystyö on johtanut siihen, että HTML-kielen uusin versio HTML5 on vähitellen tulossa laajempaan käyttöön (Keith 2010, 2-7).

HTML-kieltä käytetään sivun rakenteen määrittelyssä, mutta sen tietynlaisesta yksinkertaisuudesta johtuen monia asioita on jouduttu toteuttamaan enemmän tai vähemmän epäkäytännöllisillä tavoilla. Myöskään toiminnallisen puolen, eli useimmiten JavaScriptillä tapahtuvien asioiden, toteuttamiseen on usein käytetty puhtaan JavaScriptin sijasta erillisiä JavaScript-kirjastoja.

HTML5-kielen tuomat muutokset koskettavat lähes kaikkia verkkosivujen luomisen eri osa-alueita. HTML on saanut uusia elementtejä, joita voi käyttää esimerkiksi semantiikan määrittelyyn tai median esittämiseen. Sekä uudet että vanhat elementit ovat saaneet myös uusia attribuutteja, joiden avulla muun muassa tietojen oikeellisuuden tarkastaminen on aiempaa helpompaa. Vanhoja elementtejä ja attribuutteja on lisäksi joissakin tapauksissa muutettu, ja tarvittaessa myös poistettu kokonaan standardista. On tärkeää kuitenkin muistaa, että HTML5 on yleisellä tasolla suunniteltu taaksepäin yhteensopivaksi, joten uusimpienkin selainversioiden tulisi tukea käytöstä poistettuja ominaisuuksia, vaikkei sivujen tekijöille suositeltaisikaan niiden käyttöä. (Van Kesteren & Pieters 2012a, hakupäivä 12.2.2012.)

Muita HTML5:n uusia asioita ovat sisältömallin muutokset ja rajapintoihin liittyvät asiat. Sisältömalli määrittelee sen, kuinka elementtejä voi asettaa sisäkkäin toistensa kanssa. Elementtien ja attribuuttien tavoin HTML5:ssä on muokattu ja poistettu jo olemassa olevia rajapintoja. Muutokset ja poistot ovat keskittyneet pääasiassa DOM-malliin (Document Object Model). Lisäksi standardiin on tullut myös täysin uusia rajapintoja, joiden avulla voi muun muassa luoda helpommin erilaisia web-sovelluksia. (Van Kesteren & Pieters 2012a, hakupäivä 12.2.2012.)

2 TAUSTA

Opinnäytetyön toimeksiantajana on oululainen mainosalan yritys Rapid River. Työn tarkoituksena oli tutkia sitä, kuinka HTML5-kieltä voisi hyödyntää toimeksiantajan omassa kuvagalleriassa ja sen käyttöliittymässä. Aiheen valintaan vaikutti vahvasti se, että olen pyrkinyt suuntautumaan opinnoissani web-ohjelmointiin niiltä osin kuin se on ollut mahdollista. HTML5 tarjosi aiheena mielenkiintoisen yhdistelmän uutta ja vanhaa, sillä monia sen uusista toiminnoista ja ominaisuuksista on ollut mahdollista käyttää jo aiemminkin esimerkiksi lisäosien tai vaihtoehtoisten toteutustapojen kautta.

Koska jokainen verkossa oleva sivu on pohjimmiltaan tehty HTML:llä, on luonnollista olettaa, että HTML5 tulee olemaan tärkeä osa verkon tulevaisuutta. Viimeisimpien tutkimusten mukaan HTML5:n käyttöönotolle ennustetaan nopeaa kasvua vuoden 2012 aikana (Curtis 2012, hakupäivä 12.2.2012). Selvästi tärkeimmäksi syyksi oli mainittu HTML5:n hyvä yhteensopivuus eri laitetyyppien välillä. Jo aiemmin Adobe oli ilmoittanut lopettavansa mobiili-Flashin, joten erityisesti mobiililaitteissa HTML5:n asema on jo nyt erittäin vahva (Winokur 2011, hakupäivä 12.2.2012). Pöytälaiteympäristössä Flash on toisaalta vielä niin laajalti käytössä, etteivät esimerkiksi grafiikkaan ja videontoistoon liittyvät HTML5-kielen ominaisuudet tule tuskin läheskään yhtä nopeasti laajempaan käyttöön kuin mobiilipuolella.

Theseusta seuraamalla on voinut havaita, että HTML5-kieltä jollakin tavalla käsittelevien opinnäytetöiden määrä on ollut hiljalleen kasvussa, mutta verrattuna esimerkiksi muihin verkkosivujen luomiseen liittyviin tekniikoihin, kuten PHP:hen tai aikaisempiin HTML:n versioihin, on määrä edelleen minimaalinen (Theseus 2012, hakupäivä 18.3.2012).

Opinnäytetyön käytännön toteutuksen aikana kehitysympäristönä toimi avoimeen lähdekoodiin perustuva NetBeans. Vaikka toteutuksen aikana HTML5-tuelle ei kehitysympäristön osalta ollut suurempaa tarvetta, on NetBeansin uusimmissa versioissa olemassa tuki HTML5-kielelle (Oracle 2011, hakupäivä 12.2.2012).

2.1 Työn sisältö ja rajaus

Aihevalinnan jälkeen opinnäytetyötä varten valittiin HTML5-standardista neljä eri kehitystehtävää: semanttiset elementit, raahaus ja pudotus -rajapinta, lomakkeet ja videoiden näyttämiseen tarkoitettu

<video>-elementti. Pieniä muutoksia lukuun ottamatta valitut tehtävät pysyivät samoina koko opinnäytetyön teon ajan. Opinnäytetyön käytännön osuudessa ohjelmoitiin Rapid Riverin kuvagalleriaan HTML5-kielen ominaisuuksia sisältävä käyttöliittymä. Ohjelmoinnista ja sen vaiheista kerrotaan tarkemmin toteutukseen liittyvissä luvuissa.

Opinnäytetyön tietoperustassa käsitellään yleisesti HTML5-kieltä ja sen historiaa. Lisäksi tietoperustassa tarkastellaan kehitystehtävien kohteena olevia asioita HTML5:n näkökulmasta. Raportissa on myös muilta osin keskitytty pääosin HTML5:een, vaikka toteutuksen aikana käytettiin myös hyvin paljon PHP- ja JavaScript-ohjelmointikieliä. Tietoperustan ja käytännön työn lähteinä käytettiin ensisijaisesti HTML5-standardia kehittävien WHATWG:n ja HTML5:n virallisia määritelmiä (spesifikaatioita).

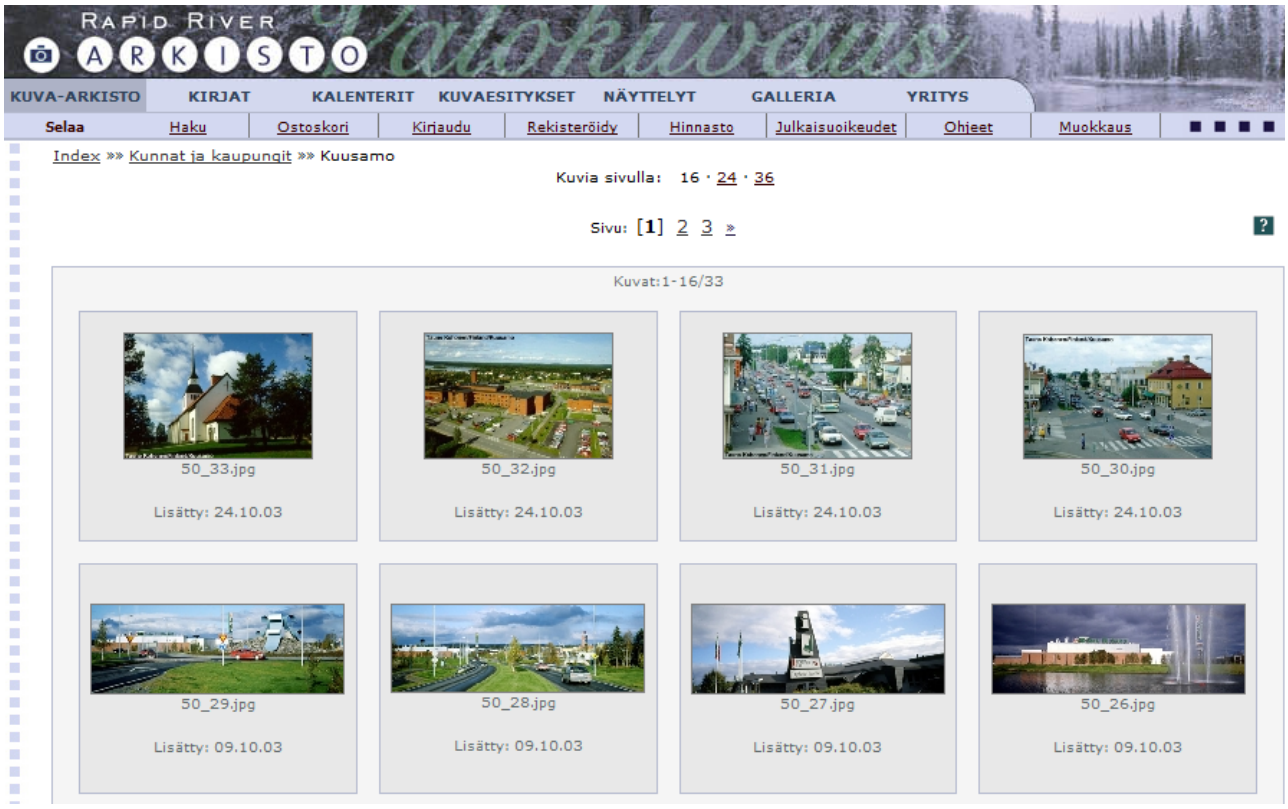
Tällä hetkellä jokaisen suurimman selainvalmistajan uusimmassa selainversiossa on ainakin jonkin tasoinen tuki HTML5:lle (Deveria 2012a, hakupäivä 12.2.2012). Joitakin ominaisuuksia on mahdollista saada toimimaan erinäisten JavaScript-ratkaisujen avulla, mutta tämän opinnäytetyön puitteissa ei kuitenkaan kiinnitetty liiemmin huomiota eri selainten mahdollisten erojen tai taaksepäin yhteensopivuuteen liittyvien seikkojen läpikäymiseen.

Yhteensopivuuksien lisäksi opinnäytetyön ulkopuolelle rajattiin myös lukuisia muita HTML5:een liittyviä aihealueita. Näistä tärkeimmiksi tai mielenkiintoisimmiksi voitaneen laskea piirtämiseen liittyviin asioihin tarkoitettu Canvas-elementti, äänitiedostojen soittamiseen tarkoitettu <audio>-elementti, paikallisesti toimivien verkkosovellusten mahdollistamiseksi luodut rajapinnat ja toiminnallisuudet ja HTML-elementtien semanttiseen määrittelyyn tarkoitettu Microdata, jolla HTML-elementteihin voidaan lisätä metatietoa esimerkiksi hakukoneita varten. (Hickson 2012c, hakupäivä 18.3. 2012.)

Näiden ja lukuisten muiden HTML5-standardin mukana olevien uusien ominaisuuksien lisäksi HTML5:een on sen kehityksen aikana liitetty monia muitakin tekniikoita ja rajapintoja, jotka syystä tai toisesta on kuitenkin siirretty erilleen HTML5-standardista (Van Kesteren & Pieters 2012a, hakupäivä 12.2.2012). Näistä esimerkkejä ovat muun muassa Web Storage, jota voidaan pitää huomattavasti kehittyneempänä versiona cookieista eli kekseistä ja joka on jo lähellä varsinaista relaatiotietokantaa, sekä käyttäjien paikallistamiseen tarkoitettu Geolocation-sovellusrajapinta.

2.2 Rapid Riverin kuvagalleria

Kehityskohteena oleva kuva-arkisto on alun perin kahden opiskelijan opinnäytetyönään tekemä verkossa toimiva kuva-arkisto. Lähes kymmenen vuoden iästään huolimatta kuva-arkisto on edelleen käytössä yrityksen omilla kotisivuilla. Kuva-arkisto pitää sisällään yleisimmät kuvagallerioiden perustoiminnot. Ylläpitäjä voi lisätä kuvia ja videoita yksitellen sekä muokata niiden lisätietoja. Tarvittaessa kansioita tai kohteita voi myös siirrellä kansioista toiseen. Peruskäyttäjä taas voi selailla kuvia sekä rekisteröityä kuvagallerian käyttäjäksi. Kuviossa 1 on esitetty Rapid Riverin sivuilla tällä hetkellä käytössä oleva kuvagalleria.



KUVIO 1. Rapid Riverin omassa käytössä oleva kuvagalleria.

Kuva-arkiston luomiseen on käytetty HTML- ja CSS-kuvauskielten (Cascading Style Sheets) lisäksi PHP-ohjelmointikieltä (PHP: Hypertext Preprocessor). Näiden lisäksi joitakin ominaisuuksia on toteutettu JavaScriptin avulla. Arkiston tietokantana toimii MySQL.

Omakohtaisesti olen ollut tekemisissä kuva-arkiston kanssa työharjoittelujaksoni aikana, jonka suoritin Rapid Riverillä vuoden 2010 syksyn aikana. Tehtäviini kuuluivat tämän opinnäytetyön tavoin erilaiset kehitystehtävät, joita olivat mm. vanhojen toimintojen uudistaminen ja arkiston päivittäminen

nykyaikaisemmaksi. Kuvagalleriaa on ollut kehittämässä myös muita tekijöitä, jotka ovat keskittyneet muun muassa käyttöliittymän kehittämiseen.

Työharjoittelun jälkeen kuvagalleriaa on kehitetty eteenpäin erinäisten projektien toimesta. Opinnäytetyön kannalta merkityksellisimpänä voidaan pitää siirtymistä MVC-malliin (Model-view-controller) pohjautuvaan rakenteeseen, jota noudattaen käytännön toteutuksen aikana syntynyt ohjelmointikoodi luotiin.

3 HTML5

HTML sai alkunsa jo 1990-luvun alussa, vaikka sen ensimmäinen virallinen versio julkaistiinkin vasta vuonna 1995 (Palmer 2004, hakupäivä 18.3.2012). Toisin kuin myöhempien HTML-versioiden kohdalla, version 2.0 julkaisijana toimi IETF, eli The Internet Engineering Task Force. 2.0-version jälkeen HTML-standardin kehittäminen siirtyi W3C:n (World Wide Web Consortium) hallinnoitavaksi, joka on kehittänyt standardia eteenpäin aina vuonna 1999 julkaistuun HTML 4.01:een asti. Toisin kuin IETF, joka kehittää Internetiin liittyviä standardeja, W3C on keskittynyt pelkästään World Wide Webin eli WWW:n standardien ylläpitämiseen ja kehittämiseen. (Keith 2010, 2.)

HTML 4.01:n jälkeen W3C lopetti HTML:n kehittämisen ja siirtyi kehittämään XHTML:ää (Lawson & Sharp 2010, xi). XHTML:n (eXtensible HyperText Markup Language) hyödyiksi katsottiin sen parempi laajennettavuus (uudet elementit ja attribuutit) ja yhteensopivuus erilaisten käyttäjäagenttien, kuten esimerkiksi mobiililaitteiden kanssa (W3C HTML Working Group 2002, hakupäivä 22.2.2012). XHTML 1.0 ei eronnut HTML 4.01:stä muuten kuin syntaksinsa osalta: XHTML oli nimensä mukaisesti XML-versio HTML:stä, minkä seurauksena XHTML noudatti XML:n (Extensible Markup Language) tiukempia syntaksisääntöjä, kuten esimerkiksi lainausmerkkien käyttöä attribuuttien yhteydessä ja tagien sekä attribuuttien kirjoittamista pienillä kirjaimilla. XHTML 1.1:n myötä standardissa siirryttiin puhtaaseen XML:n käyttöön, mikä tarkoitti muun muassa sitä, ettei sen hetken suosituin web-selain Internet Explorer kyennyt näyttämään ollenkaan XHTML 1.1:n mukaisia sivuja. (Keith 2010, 2-3.)

XHTML 1.1:n jälkeen W3C:n ryhtyi kehittämään XHTML 2:sta. Toisin kuin XHTML:n aikaisemmissa versioissa, XHTML 2:n ei suunniteltu olevan ollenkaan taaksepäin yhteensopiva. Kyseinen kehityssuunta ei kuitenkaan ollut kaikkien mieleen, ja sen johdosta vuonna 2004 W3C:n rinnalle syntyi WHATWG (Web Hypertext Application Technology Working Group), joka perustettiin Operalla, Applella ja Mozillalla työskennelleiden henkilöiden toimesta. (Lawson & Sharp 2010, xi-xii.)

Aluksi WHATWG ryhtyi kehittämään kahta uutta standardia: Web Forms 2.0:aa ja Web Apps 1.0:aa, joiden kummankin tarkoituksena oli laajentaa HTML 4.01:tä. Myöhemmin kummatkin standardeista yhdistettiin yhdeksi kokonaisuudeksi, jonka nimeksi tuli HTML5. (Keith 2010, 5.)

Sillä aikaa kun WHATWG kehitti HTML5:ttä, jatkoi W3C edelleen työtään XHTML 2:n parissa. Kehitystyö oli kuitenkin hidasta, eikä puhtaaseen XML:ään siirtyminen näyttänyt enää vuonna 2006

realistiselta tavoitteelta (Keith 2010, 5). Samana vuonna W3C ilmoitti kiinnostuksensa osallistua HTML5:n kehittämiseen ja vuodesta 2007 lähtien HTML5:ttä on kehitetty W3C:n ja WHATWG:n välisenä yhteistyönä (Hickson 2012a, hakupäivä 22.2.2012). XHTML 2.0:n kehitystyö jatkui kuitenkin aina vuoteen 2009 saakka, jolloin W3C päätti lopullisesti lopettaa sen kehittämisen ja siirtyä kokonaan HTML5:n taakse (Keith 2010, 6).

HTML5:n suunnitteluperiaatteista tärkeimmiksi on mainittu yhteensopivuus, käyttökelpoisuus, yksinkertaisuus ja saavutettavuus. Toisin kuin XHTML 2.0:n kohdalla, HTML5:ssä taaksepäin yhteensopivuus on keskeisessä roolissa; vanhoja elementtejä, rajapintoja tai attribuutteja ei ole syytä hylätä ja HTML5-sisällön tulisi näkyä myös vanhemmissa selaimissa siedettävästi. Lisäksi HTML5:ssä on suurten muutosten sijasta keskitytty enemmän siihen, että jo olemassa olevista käytännöistä on pyritty luomaan osa standardia sen sijasta, että oltaisiin luotu jotakin täysin uutta. (Van Kesteren & Stachowiak 2007a, hakupäivä 18.3.2012.)

Käyttökelpoisuudessa painotetaan muun muassa sisällön ja sen esittämisen erottamista HTML-koodissa, tietoturvan ottamista huomioon suunnittelussa, DOM:n säännönmukaisuutta eri parserien välillä, ja keskittymistä käytännön ongelmiin teorian sijasta. Standardin tulisi myös olla ensisijaisesti käyttäjää ajatellen edeten siitä tekijöiden, toteuttajien ja määrittelijöiden kautta lopulta teorian ns. ”puhtauteen” sillä periaatteella, että jokaisen ryhmän tarpeet asetetaan seuraavana listassa tulevaa tärkeämmäksi. (Van Kesteren & Stachowiak 2007b, hakupäivä 18.3.2012.)

Yksinkertaisuudessa on keskitytty toimintojen säännönmukaisuuteen, monimutkaisuuden välttämiseen ja virheiden käsittelyyn. Toimintojen käyttäytymisen tulisi olla selkeästi määriteltyä ja monimutkaisten ratkaisujen sijasta pitäisi keskittyä yksinkertaisuuteen aina kun se vain on mahdollista. (Van Kesteren & Stachowiak 2007c, hakupäivä 18.3.2012.)

Saavutettavuudessa otetaan huomioon eri laitealustat, kielet ja erityisryhmät. Ominaisuuksien tulisi toimia eri laitteissa tai medioissa, sisällön julkaisun tulisi olla mahdollista eri kielillä ja erityisryhmille tulisi tarvittaessa tarjota vaihtoehtoisia tapoja sisällön näyttämiseen ja hyödyntämiseen. (Van Kesteren & Stachowiak 2007d, hakupäivä 18.3.2012.)

Seuraavissa alaluvuissa ja pääluvuissa käsitellyt elementit, attribuutit ja rajapinnat eivät sisällä läheskään kaikkia HTML5:n tuomista muutoksista. Hyvän ja ytimekkään yleiskuvauksen muutoksista saa esimerkiksi W3C:n sivuilta osoitteesta http://www.w3.org/TR/html5-diff/. On kuitenkin

huomioitava, että määritelmä ei ole saavuttanut vielä lopullista muotoaan, joten muutoksia voi tulla ja on tullutkin esimerkiksi tämän opinnäytetyön teon aikana. Lisää epämääräisyyttä saattavat tuoda myös WHATWG:n ja W3C:n viralliset määritelmät, jotka eroavat joiltakin osin toisistaan (Hickson 2012l, hakupäivä 19.3.2012).

Vaikka HTML5:n ei odoteta tulevan lopullisesti valmiiksi vielä vähään aikaan, voi useita sen ominaisuuksia käyttää jo nyt uusimmissa selainversioissa (WHATWG 2012, hakupäivä 18.3.2012). Tämän hetken tietojen mukaan HTML5 on etenemässä suositusvaiheeseen vuoden 2014 alkupuolella (W3C 2011b, hakupäivä 18.3.2012).

3.1 Semanttiset elementit

Ennen HTML5:ttä verkkosivujen sisäisen rakenteen määrittely tapahtui erityisesti sisällön osalta erittäin pienellä määrällä eri elementtejä (W3C HTML Working Group 1999a, hakupäivä 12.2.2012). Oli sitten kyseessä taulukoilla eli <table>-elementillä tai div-elementeillä luotu ulkoasu, ei sivun sisäinen rakenne avautunut kovinkaan helposti ulkoiselle katselijalle, kuten ihmiselle, hakukoneelle tai ruudunlukuohjelmalle. Ulkoasun lisäksi myös tietojen jäsentelyyn tarkoitettujen elementtien määrä oli aikaisemmissa HTML:n versioissa erittäin pieni (W3C HTML Working Group 1999c, hakupäivä 18.3.2012).

Sivustojen kehittäjien näkökulmasta apua toivat muun muassa elementteihin liitetyt luokka- tai tunnistemääritteet, joiden avulla tiedettiin, mikä osa sivusta oli esimerkiksi sen ylätunniste, leipäteksti tai uusi luku. Kokonaisuuden kannalta tämäntyyppinen toiminta on kuitenkin ennemminkin ongelmien kiertämistä kuin niiden ratkaisua, koska kyseisiä elementtien ominaisuuksia ei varsinaisesti ole tarkoitettu semantiikan esittämiseen (W3C HTML Working Group 1999b hakupäivä 14.2.2012). Esimerkki HTML4-tyyppisestä rakenteen määrittelystä:

```
<div id="ylätunniste">Yläpalkki</div>
        <div class="sisältö">
            <div class="artikkeliOtsikko">
                <h1>Artikkelin otsikko</h1>
            </div>
            <p>Artikkelin teksti</p>
```

```
            <div>Julkaisuaika: ...</div>
        </div>
<div id="alatunniste">Alapalkki</div>
```

HTML5 tuo mukanaan noin kaksikymmentä uutta elementtiä, joiden avulla sivuston rakenteen määrittelemisestä pitäisi ainakin periaatteessa tulla selkeämpää. Esimerkiksi ylä- ja alatunnisteille on nyt omat <header>- ja <footer>-elementtinsä, ja dokumentin eri osiot voidaan kääriä tapauksesta riippuen joko <section>- tai <article>-elementin sisään. (Van Kesteren & Pieters 2012a, hakupäivä 12.2.2012.) Esimerkki HTML5-tyyppisestä rakenteen määrittelystä:

```
<header>Yläpalkki</header>
        <article>
            <header>
                <h1>Artikkelin otsikko</h1>
            </header>
            <p>Artikkelin teksti</p>
            <footer>Julkaisuaika: </footer>
        </article>
<footer>Alapalkki</footer>
```

Useimpia uusista elementeistä voi käyttää luontevasti niiden yksiselitteisyytensä vuoksi: <time>-tagi kuvaa nimensä mukaisesti aikaa, kun taas <progress>-tagin avulla voidaan seurata toiminnon edistymistä (Van Kesteren & Pieters 2012a, hakupäivä 12.2.2012). Joidenkin elementtien kohdalla niiden käyttäminen ei kuitenkaan ole niin yksinkertaista kuin mitä nimestä voisi päätellä. Muun muassa <header>- tai <footer>-elementtiä voi käyttää sivupohjan rakenteen määrittelyn lisäksi esimerkiksi <article>- ja <section>-tagien sisällä kuvaamaan artikkelin ja osion sisältöä (Osborne 2009, hakupäivä 19.3.2012). Edellä mainittujen <article>- ja <section>-tagien välinen ero on toinen hyvä esimerkki siitä, että eri elementtien käyttötarkoitukset ja merkitykset eivät ole vielä kovinkaan selkeitä päätellen lukuisista Google-haun kautta paljastuvista aihetta käsittelevistä artikkeleista (Lawson 2010a, hakupäivä 19.3.2012). On kuitenkin luultavaa, että sitä mukaa kun HTML5:n käyttö yleistyy, elementtien käyttöön liittyvät käytänteet selkiytyvät ja mahdolliset epäselvyydet häviävät (Lawson 2010a, hakupäivä 19.3.2012).

3.2 Raahaus ja pudotus

Raahaus- ja pudotustoiminto löytyy nykyisellään jo useista websovelluksista. Googlen toimisto-ohjelmat tai cPanelin-tapaiset hallintapaneelit eivät ehkä vielä vastaa käytettävyydeltään sitä, mitä ne olisivat työpöytäsovelluksina, mutta siitä huolimatta taulukkolaskennan solujen liikuttelu tai tiedostojen siirto eri hakemiston välillä onnistuu suoraan selaimen kautta. Ryhtymättä sen tarkemmin tutkimaan mainittujen sivujen tai muiden verkossa toimivien sovellusten raahaus- ja pudotus-toimintojen toteutustapaa, on todennäköistä, että suurin osa niistä on toteutettu JavaScriptin avulla.

Pelkän JavaScriptin avulla raahaus- ja pudotustoiminnon luominen on erittäin työlästä, joten kehitystyön helpottamiseksi ja selainyhteensopivuuden takaamiseksi on järkevämpää käyttää apuna jotakin erillistä JavaScript-kirjastoa. Kuvagallerian osalta kirjastona on käytetty jQuerya, joka on Prototypen, MooToolsin ja Yahoon YUI:n ohella yksi tunnetuimmista JavaScript-kirjastoista (Cheung 2011, hakupäivä 19.3.2012). Kuvagalleriassa raahaus- ja pudotustoimintojen avulla ylläpitäjä voi siirtää kansioita ja mediaelementtejä toisiin kansioihin (Kuvio 2).

KUVIO 2. Kuvagallerian kuvien siirtotoiminto.

3.3 Lomakkeet

Lomakkeiden käyttötarkoitus verkkosivuilla on yleensä käyttäjän syöttämän tiedon kerääminen ja lähettäminen palvelimelle käsiteltäväksi. Lomakkeet koostuvat <form>-tagista, jolla määritellään lomakkeen aloitus- ja lopetuskohdat, ja sen sisällä olevista lomake-elementeistä. Lomake-elementtejä ovat esimerkiksi teksti- ja salasanakentät, valintaruudut, pudotuslistat ja valintanapit. (W3C HTML Working Group 1999d, hakupäivä 19.3.2012.) Kuvagalleriassa lomakkeita käytetään muun muassa käyttäjän rekisteröinnin aikana, etsi-toiminnossa ja mediaelementtejä lisättäessä ja muokattaessa.

Semanttisten elementtien tavoin myös lomake-elementtien tai elementtien tyyppien määrä on ollut niukahko aiemmissa HTML:n versioissa. Esimerkiksi tekstimuotoisen tiedon esittämiseen on käytetty useimmiten input-elementtiä, jonka tyypiksi on määritelty ”text”. (W3C HTML Working Group 1999d, hakupäivä 19.3.2012.) HTML5 tuo mukanaan huomattavasti laajemman valikoiman erilaisia input-elementin tyyppejä. Muun muassa puhelinnumerolle, sähköpostille, hakukentille, erilaisille ajoille (kuukausi, viikko, kellonaika) ja url-osoitteille on nyt omat tyyppimäärityksensä (Van Kesteren & Pieters 2012a, hakupäivä 12.2.2012). Joitakin poikkeuksia lukuun ottamatta näiden ja aikaisemman text-määrityksen välillä ei ole kuitenkaan mitään näkyvää eroa, vaan kentät näyttäytyvät käyttäjälle edelleen tekstikenttinä. Ihannetilanteessa esimerkiksi kalentereihin liittyvistä kentistä avautuisi erillinen kalenteri-näkymä, joka on täytynyt toteuttaa ennen JavaScriptin avulla, mutta tällä hetkellä vain Opera-selain osaa näyttää oikeanlaisen kalenterin muiden selainten tyytyessä pelkkään tekstikenttään (Deveria 2012b, hakupäivä 19.3.2012).

Tämän lisäksi HTML5 sisältää muutaman kokonaan uuden elementin (datalist, keygen, output) ja lisäksi lukuisia uusia attribuutteja olemassa oleville elementeille (Van Kesteren & Pieters 2012b, hakupäivä 19.3.2012). Uusia elementtejä ei hyödynnetty kuvagalleriassa, koska niille ei löydetty sopivaa käyttötarkoitusta tai elementtien toiminnallisuus ei ollut vielä riittävän tuettua tai luotettavaa. Uusien attribuuttien lukumäärä on noin kymmenen kappaletta, mutta tässä luvussa niistä on otettu käsittelyyn vain ne, joita käytettiin toteutuksessa tai joita mahdollisesti tullaan käyttämään myöhemmin kuvagallerian kehityksen aikana.

Autofocus-attribuutin tarkoituksena on määrittää se kenttä, johon kursori asetetaan sivua ladattaessa (Van Kesteren & Pieters 2012b, hakupäivä 19.3.2012). Kuten useimman muunkin tässä luvussa käsitellyn attribuutin kohdalla, olisi tämän toiminnon luomiseen vaadittu aiemmin JavaScriptiä, jonka avulla kursori olisi tässä tapauksessa pitänyt erikseen asettaa oikean elementin kohdalle. HTML5:ssä

sama tapahtuisi yhden elementin sisällä, mikä parantaa huomattavasti ylläpidettävyyttä, koska samaan elementtiin liittyviä toimintoja ei ole useammassa kuin yhdessä paikassa. Seuraavassa esimerkissä on esitetty yksinkertaistetussa muodossa HTML5:n ja aikaisempien HTML-versioiden ero:

```
// HTML4-versio
<script>
function autofocus() {
 var keskitettäväKenttä = document.getElementById(” tekstikenttä” );
 keskitettäväKenttä.focus();
}
</script>
<body onload=” autofocus();” >
 <input type=’text’ id=’tekstikenttä’/>
</body>

// HTML5
<input type=’text’ id=’tekstikenttä’ autofocus/>
```

Form-attribuutin ajatuksena on se, että tiettyyn lomakkeeseen voidaan sisällyttää sellaisia elementtejä, jotka eivät välttämättä ole <form> … </form>-tagien sisällä, vaan niiden ulkopuolella. Attribuutin käyttäminen tapahtuu niin, että <form>-elementille annetaan id-määritys, jonka jälkeen haluttuun <input>-elementtiin lisätään form-attribuutti, jossa kerrotaan <form>-elementin id-tieto. (Van Kesteren & Pieters 2012b, hakupäivä 19.3.2012.) Esimerkki käytöstä:

```
<form id=’rekisteröinti’ method=’post’>
...
</form>
<input type=’text’ form=’rekisteröinti’ />
```

Pattern-attribuutilla voidaan validoida <input>-elementin text-, search-, url-, tel-, email- ja password-tyyppisiä kenttiä (Hickson 2012e, hakupäivä 19.3.2012). Validointi tapahtuu säännöllisten lausekkeiden avulla, joiden syntaksi on sama kuin mitä se on JavaScriptin säännöllisissä lausekkeissa (Hickson

2012d, hakupäivä 19.3.2012). Toisin sanoen, pattern-attribuutilla voidaan tarkastaa onko käyttäjän antama syöte oikeassa muodossa.

Placeholder-attribuutti asettaa <input>-kenttään niin sanotun vihjetekstin (Van Kesteren & Pieters 2012b, hakupäivä 19.3.2012). Esimerkiksi hakukentissä voidaan käyttää sanaa ”haku” sen sijasta että se olisi erillisenä tekstinä hakukentän vasemmalla puolella.

Required-attribuutin avulla tietyt <input>-kentät lomakkeesta voidaan määritellä pakollisiksi, eli sellaisiksi, joihin käyttäjän täytyy syöttää tietoa (Van Kesteren & Pieters 2012b, hakupäivä 19.3.2012). Required toimii myös yhdessä pattern-attribuutin kanssa, joten vaikka käyttäjä olisikin syöttänyt jotakin tekstiä tekstikenttään, ei häntä päästetä eteenpäin, ellei tieto ole oikeassa muodossa.

3.4 Videoelementti

HTML5:n videoelementin tarkoituksena on sen nimensä mukaisesti toistaa videoita HTML-sivuilla. Aikaisemmissa HTML-versioissa videoiden laittaminen sivuille tapahtui yleensä joko <embed>- tai <object>-tagia käyttäen. Joitakin poikkeuksia lukuun ottamatta videoiden katseluun ei kuitenkaan riittänyt se, että videotiedoston nimi liitettiin edellä mainittuihin tageihin, vaan tilanteesta riippuen videoiden toistamiseen vaadittiin aina jokin lisäosa tai erillinen soitin. (Lawson 2010b hakupäivä 19.3.2012.) Esimerkiksi suositun JW Player -soittimen kotisivuilla yksittäisen videotiedoston soittamista varten on esitetty seuraavanlainen koodiesimerkki:

```
<script type='text/javascript' src='jwplayer.js'></script>
<div id='mediaspace'>This text will be replaced</div>
<script type='text/javascript'>
  jwplayer('mediaspace').setup({
    'flashplayer': 'player.swf',
   'file': 'http://content.longtailvideo.com/videos/flvplayer.flv',
    'controlbar': 'bottom',
    'width': '470',
    'height': '320'
  });
```

```
</script>
```

Verkkosivuston ylläpidon kannalta varsinaisen videotiedoston sijasta sivuille upotettiin yleensä jokin videon tiedostomuotoa tukeva soitin, jota käskettiin erikseen toistamaan haluttu videotiedosto. HTML5:n <video>-tagin avulla tästä ja muista vaiheista olisi tarkoitus päästä eroon niin, että videon näyttämiseen tarvittaisiin pelkästään tiedoston nimi, eikä mitään muuta. Esimerkki käytöstä:

```
<video>
 <source src="video.mp4" type="video/mp4" />
</video>
```

Todellisuus ei kuitenkaan tällä hetkellä näytä yhtä helppokäyttöiseltä. Toistaiseksi HTML5-standardi ei sisällä virallista määritelmää siitä, mitä videoformaatteja selaimien tulisi tukea (W3C 2011a, hakupäivä 12.2.2012). Tästä seurauksena on ollut se, ettei yksikään eniten käytetyistä videoformaateista toimi sellaisenaan kaikissa tunnetuimmissa selaimissa (Mozilla Developer Network 2012a, hakupäivä 19.3.2012). Esimerkiksi suosituinta H.264-formaattia eivät tällä hetkellä tue Firefox, Chrome ja Opera, vaan tuki on mahdollista saada vain erillisiä lisäosia asentamalla, mikä taas sotii sitä ajatusta vastaan, että <video>-elementin käytön tulisi olla mahdollisimman helppoa. Sama tilanne on myös kahden muun eniten esillä olevan formaatin, eli Ogg Theoran ja WebM:n kohdalla. (Pilgrim 2011a, hakupäivä 19.3.2012.)

HTML5:n <video>-elementti pitää sisällään noin kymmenkunta attribuuttia. Näistä aiemmin mainittu src määrittelee videon sijainnin, kun taas posterin avulla videolle on mahdollista asettaa ns. kansikuva. Preloadilla voidaan määritellä se, missä vaiheessa video ladataan, kuten esimerkiksi jo silloin, kun käyttäjä tulee sivulle. Autoplaylla videon voi asettaa käynnistymään heti sivun latautumisen jälkeen. Loopin avulla videon voi asettaa toistamaan itsensä uudelleen. Videon korkeudelle ja leveydelle sekä äänen mykistämiselle on myös omat attribuuttinsa. Controlsin avulla videoelementtiin saa näkymään soittimen toimintopainikkeet. (Mozilla Developer Network 2012b, hakupäivä 19.3.2012.)

Edellä mainitun lisäksi videoelementtiä on mahdollista käyttää JavaScriptin ja <canvas>-elementin kanssa. Javascriptin avulla voidaan esimerkiksi lukea videon metatietoja, muuttaa soittimen kokoa, tarkastaa puskuroinnin tilaa, muuttaa videon nopeutta tai vaikkapa vaihtaa toistettava tiedosto kokonaan uuteen (Mozilla Developer Network 2012c, hakupäivä 19.3.2012). Canvasin ja <video>-

elementin yhteiskäytöstä on olemassa verkossa useita demoja. Videoihin on esimerkiksi mahdollista lisätä omia tehosteita tai vaihtoehtoisesti koko sivuston ulkoasua voi muuttaa videon sisältöä lukemalla (Alvares 2010, hakupäivä 19.3.2012). Viimeisenä voidaan mainita tekstityksiä varten luotu <track>-elementti, jota voidaan käyttää yhdessä videoelementin kanssa (Mozilla Developer Network 2012d, hakupäivä 19.3.2012).

4 TOTEUTUS

Opinnäytetyön toteutusvaihetta varten valittiin neljä erillistä kehitystehtävää. Semanttisten elementtien avulla oli tarkoitus selkeyttää sivuston rakenteen määrittelyä käyttämällä HTML5-kielen uusia elementtejä. Raahaus- ja pudotustoimintoon liittyvässä kehitystehtävässä pyrittiin toteuttamaan HTML5-kielen raahaus- ja pudotusrajapintaa käyttämällä sama toiminnallisuus kuin aiemmin käytössä olleessa jQueryllä tehdyssä versiossa. Lomakkeisiin liittyvän tehtävän perusperiaate oli sama kuin semanttisten elementtien kohdalla. Tehtävän tarkoituksena oli hyödyntää HTML5-kielen uusia tapoja määritellä tietoa, joka tässä tapauksessa tapahtui syöttökenttien uusien tietotyyppien kautta. Lomakkeissa oli tarkoitus myös käyttää HTML5-kielen uusia attribuutteja semantiikkaan liittyvien määrittelyjen lisäksi. Neljännessä tehtävässä keskityttiin <video>-elementtiin, jota käytetään videotiedostojen esittämiseen.

4.1 Semanttiset elementit

Semanttisten elementtien toteuttamista varten luotiin erillinen GalleriaHelper-luokka. GalleriaHelper-luokasta tehtiin lisäksi omat luokat sekä HTML4- että HTML5-versiota varten, jotka perivät funktionsa PHP:n Extends-määritelmän avulla kantaluokastaan. GalleriaHelper-kantaluokan konstruktorissa (alustajassa) määriteltiin HTML_VERSIO-vakiota käyttäen se, luotiinko GalleriaHelperistä HTML4:ään vai HTML5:een pohjautuva versio.

```
public static function GalleriaHelperInit() {
   if(strcmp(HTML_VERSIO, "HTML5") == 0) {
     self::$htmlgalleriahelper = new GalleriaHelperHTML5();
   }
   else if(strcmp(HTML_VERSIO, "HTML4") == 0) {
     self::$htmlgalleriahelper = new GalleriaHelperHTML4();
    }
}
```

HTML5-versiossa kantaluokan funktiot ylikirjoitettiin (override) niiltä osin kuin sille oli tarvetta. Toistaiseksi ylikirjoittamista on käytetty div- ja close_tag-funktioiden kohdalla. HTML4-versiossa div-

funktio loi määriteltyjen arvojen perusteella div-elementin. Close_tag-funktion tarkoituksena taas oli sulkea aiemmin avattu HTML-elementti.

Sekä div- että close_tag-funktio sisälsivät lisäparametrina $html5tag-parametrin, johon oli asetettu luotavaa tai suljettavaa elementtiä vastaava HTML5-elementti. Jos kuvagalleria oli määritelty käyttämään HTML5:ttä, elementin luomisessa tai sulkemisessa käytettiin HTML5-elementin nimeä, muutoin nimenä käytettiin diviä tai close_tagin ollessa kyseessä sitä elementin nimeä, mikä oli sivupohjassa määritelty. Seuraavassa on esitelty GalleriaHelperHTML5-luokan div-funktio, jonka alussa tarkastetaan, löytyykö haetulle HTML5-tagille vastinetta GalleriaHelperHTML5-luokan $htmltags-muuttujasta, joka sisältää luettelon HTML5:n semanttisista tageista.

```
public static function div($id=null, $class=null, $html5tag=null) {
 if(in_array($html5tag, self::$html5tags)) {
      print "<$html5tag";
    }
    else {
      print "<div";
    }
...
```

Div-funktiota kutsuttiin pääasiassa kuvagallerian käyttäjälle näkyvien sivujen sisältöä kuvaavien PHP-tiedostojen kautta. Funktiokutsuun lisättiin mahdollisen HTML5-tagin lisäksi elementin id- ja class-attribuutti. Tagin sulkeminen tapahtui close_tagin avulla, jolle annettiin parametreinä HTML4-tagin lisäksi mahdollinen HTML5-tagi. Tämän järjestelyn ansiosta sivuston versiota voitiin vaihdella HTML4- ja HTML5-versioiden välillä sivuston asetustiedostoista löytyvän muutettavan asetuksen avulla. Seuraavassa esimerkissä on esitetty nav-tyyppisen HTML5-tagin avaaminen ja sulkeminen:

```
<?php
print GalleriaHelper::div("navigointipolku", null, "nav");
print GalleriaHelper::close_tag("div", "nav");
?>
```

Semanttisten elementtien kohdalla toteutuksessa tulleet ongelmat liittyivät pääosin niiden käyttöön. Koska teksti ei ole kuvagalleriassa pääosassa, ei suurta osaa HTML5:n uusista semanttisista elementeistä voitu kunnolla hyödyntää joko niiden epäsopivuuden tai epäselvien määritelmien vuoksi. Esimerkiksi <article>-tagia voitaisiin periaatteessa käyttää kuvaamaan yksittäistä kuvaa, jos sen ajatellaan olevan itsenäinen osa sisältöä, ts. se voitaisiin esimerkiksi liittää jollekin muulle sivulle. <details>- ja <summary>-tagien avulla käyttäjälle voitaisiin taas näyttää lisätietoa jostakin asiasta, jonka käyttäjä voisi halutessaan piilottaa tai asettaa uudelleen esiin (Hickson 2012f, hakupäivä 19.3.2012). Kuvagalleriassa kuvien lisätiedot ovat kuitenkin tällä hetkellä aina näkyvissä, joten mitään järkevää syytä näiden tagien käytölle ei ollut.

Selkeitä tapauksia elementtien käytöstä olivat <header>, <footer>, <nav> ja <time>. Koska edellisessä kappaleessa mainittua <article>-tagia ei otettu käyttöön, jäivät <header>- ja <footer>-tagit sivupohjaa varten, jossa ne merkitsivät sivun ylä- ja alatunnisteita. Navigointia kuvaamaan tarkoitettua <nav>-tagia käytettiin kansiopuun lisäksi murupoluissa ja hakukentässä, joihin kyseinen elementti soveltuu myös käytettäväksi (Leadbetter 2009, hakupäivä 19.3.2012). <time> on hyvä esimerkki HTML5-standardin jatkuvista muutoksista: kyseinen elementti poistettiin vuoden 2011 loppupuolella vähäksi aikaa standardista kokonaan, mutta palautettiin hetken päästä takaisin. <time>:n tarkoituksena on esittää aika (vuosi, vuosi ja kuukausi, päivämäärä, kellonaika jne.) tietokoneen ymmärtämässä muodossa. Kuvagalleriassa <time>-elementtiä käytettiin kuvien ja videoiden lisäämispäivämäärän yhteydessä. (Lawson 2012, hakupäivä 19.3.2012.) Kuviossa 3 on esitetty kuvallinen yhteenveto kuvagalleriassa käytetyistä semanttisista elementeistä.

KUVIO 3. Semanttisten elementtien käyttö kuvagalleriassa.

Kuvitusta, kaavioita, kuvia ja muita vastaavia varten luotujen <figure> ja <figcaption>-tagien osalta kohdattiin samoja ongelmia kuin <article>:n ja <section>:n kohdalla, sillä niiden tavoin <figure>:n ja <figcaption>:n käyttötarkoitus oli enemmänkin tekstimuotoisen sisällön semanttisessa määrittelyssä (Hickson 2012g, hakupäivä 19.3.2012). <aside>-tagi taas on tarkoitettu pääosin lisätiedon esittämiseen, mutta virallisen määrityksen mukaan sitä voisi käyttää myös navigoinnin yhteydessä (Hickson 2012h, hakupäivä 19.3.2012). Kuvagalleriassa sivupalkin navigointipuun asema sivun navigoinnissa on kuitenkin erittäin keskeisessä osassa, joten lisätiedon esittämiseen tarkoitettu tagi ei ainakaan maalaisjärjellä tulkittuna ole sopiva kyseiseen käyttöön.

4.2 Raahaus ja pudotus

Raahaa ja pudota -toiminnon luominen HTML5:n avulla tapahtui samoilla perusperiaatteilla kuin jQueryllä tehtäessä. Luonnollinen aloituspiste toiminnon tekemiseen oli määritellä ne sivuston elementit, joita käyttäjä pystyi raahaamaan. Elementtien alustaminen tapahtui käytännössä niin, että jokaiseen raahattavaan kohteeseen liitettiin joko HTML:n tai PHP-koodin puolella attribuutiksi ”draggable”, jonka arvoksi annettiin ”true”.

Alustuksen jälkeen raahattaviin elementteihin lisättiin tapahtumankuuntelijat, joiden avulla voitiin määritellä eri tapahtumien aikana tapahtuvat toiminnot. Lisääminen tapahtui helpoiten hakemalla halutut elementit aluksi JavaScriptin getElementsByClassName-metodilla, jonka jälkeen tapahtumankuuntelijat voitiin lisätä silmukan avulla elementteihin.

```
var kuvadivit =
document.getElementsByClassName("yksittainenkuvadiv");
for(var i = 0; i < kuvadivit.length; i++) {
    kuvadivit[i].addEventListener
    ("dragstart", dragStartForDiv, false);
    kuvadivit[i].addEventListener
    ("dragend", dragEndForDiv, false);
}
```

HTML5:n rajapinta tukee seitsemää erilaista raahaustapahtumaa, joita ovat dragstart, drag, dragenter, dragleave, dragover, drop ja dragend (Hickson 2012i, hakupäivä 19.3.2012). Dragstart-tapahtuma määrittelee sen, mitä tapahtuu silloin kun elementtiä ryhdytään raahaamaan. Kuten muidenkin tapahtumien kohdalla, suurin osa dragstart-tapahtuman sisällä olevasta JavaScript-koodista liittyi käyttäjälle näkyvien visuaalisten tehosteiden tekemiseen, eikä niinkään itse kuva-arkiston toimivuuteen. Esimerkiksi dragstartin kohdalla tämä ilmeni niin, että raahattavaa kohdetta himmennettiin CSS-tyylimääreiden avulla lähes läpinäkymättömäksi, jotta käyttäjän on helpompi nähdä mihin kohde pudotetaan. Vastaavasti dragover- ja dragleave-tapahtumissa voitiin määritellä kohdealue muuttamaan ulkoasuaan sen mukaan milloin käyttäjä liikutti raahattavaa kohdetta kohdealueen päällä ja sen ulkopuolella.

Raahaustapahtumien aikana voidaan käyttää DataTransfer-oliota, jonka metodien avulla raahattavalle kohteelle olisi voitu esimerkiksi asettaa raahattavana olevan kuvan, videon tai kansion tunnistenumero. Koska kuvagallerian käytettävyyden kannalta oli parempi, että käyttäjä pystyi valitsemaan useampia kohteita siirrettäväksi yhdellä kerralla, haettiin kohteiden tunnistenumerot erikseen käyttämättä DataTransferin setData- ja getData-metodeja, joilla tietoa voitiin asettaa ja lukea raahattavasta kohteesta (Hickson 2012j, hakupäivä 19.3.2012). Seuraavassa koodissa esitellään raahattavan divin dragstart-tapahtuma, jossa DataTransferin setData-metodia on käytetty siihen, että tiedetään raahattavan kohteen olevan div-elementti.

```
function dragStartForDiv(event) {
    $(this).addClass("valitutKohteet");
    $(this).css("opacity", "0.1");
    event.dataTransfer.setData('text/plain', "div");
}
```

Metodien lisäksi DataTransfer-sisältää muun muassa dropEffect-, effectAllowed- ja files-nimiset attribuutit. DropEffect voi saada arvokseen jonkin seuraavista: copy, move, link, none. EffectAllowed voi saada arvokseen jonkin edellä mainituista, tai yhdistelmän kyseisistä arvoista, kuten esimerkiksi copyLinkin tai linkMoven. (Hickson 2012j, hakupäivä 19.3.2012.) Yksinkertaisimmillaan käyttäjälle voidaan visuaalisten tehosteiden avulla näyttää se, mitkä toiminnot ovat mahdollisia elementtien välillä. Esimerkiksi asettamalla dropEffectin arvoksi ”move” hiiren kursori muuttuu vastaavanlaiseksi kuin mitä se on Windowsin resurssienhallinnassa, jos ollaan siirtämässä tiedostoa toiseen kansioon. EffectAllowedin avulla taas voidaan säädellä sitä, millaisia toimintoja kohde tukee. Jos raahattavalle kohteelle on esimerkiksi asetettu dropEffectiksi ”move” ja kohdealue hyväksyy vain ”copyn”, näytetään käyttäjälle silloin Windowsista tuttu kieltomerkki, eikä toimintoa myöskään silloin suoriteta. DataTransferin Files-attribuutin ja HTML5-standardiin liittyvän, mutta ei siihen kuuluvan, File API:n avulla käyttäjän olisi periaatteessa mahdollista siirtää tiedostoja raahaamalla ne työpöydältä selainikkunaan ja siellä olevalle kohdealueelle, jossa tiedostot voitaisiin esimerkiksi ladata kuvagalleriaan (Ranganathan & Sicking 2011, hakupäivä 19.3.2012). Seuraavassa on esitelty navigointipuun linkkien dragover-tapahtuma:

```
function dragoverForLink(event) {
        $(this).css("opacity", "0.1");
        event.dataTransfer.effectAllowed = 'move';
        if (event.preventDefault) event.preventDefault();
        event.dataTransfer.dropEffect = 'move';
        return false;
}
```

Raahaus- ja pudotustoiminnon varsinainen toiminnallisuus sijaitsee drop-tapahtumassa. Kuvagallerian tapauksessa kyseisessä tapahtumassa kerätään aluksi kaikki siirrettäväksi valitut

kohteet näkyvältä sivulta. Useamman kohteen siirtäminen tapahtuu napsauttamalla kuvan, kansion tai videon sisältävää elementtiä, jonka jälkeen siirtäminen tapahtuu raahaamalla jokin valituista kohteista kohdekansioon.

Siirrettävistä kohteista kerättiin niiden tunnistenumerot taulukkoon. Taulukko muutettiin lopuksi JSON-muotoon, joka soveltuu yksinkertaisuutensa ja PHP:n kanssa helposti käytettävyytensä vuoksi hyvin tämäntyyppiseen tiedonsiirtoon.

Tunnistenumeroiden yhteydessä käytettiin hyväksi HTML5:n data-attribuutteja, joita ei alun perin oltu kirjattu opinnäytetyön sisältöön. Data-attribuuttien avulla mihin tahansa HTML-elementtiin voidaan lisätä sellaista tietoa, joka ei sovi ennalta määriteltyihin attribuutteihin, joita ovat esimerkiksi elementtien luokkanimet (class) tai tunnisteet (id). Data-attribuutit alkavat aina ”data-”-muodossa, jonka jälkeen elementille annetaan nimi ja arvo samaan tapaan kuin miten tehdään tavallistenkin attribuuttien kohdalla (Hickson 2012b, hakupäivä 12.2.2012).

JQueryllä ja HTML4:llä tehdyssä versiossa tunnistenumeroiden lukeminen tapahtui erillisestä input-elementistä, joka oli piilotettu jokaisen kansio-, kuva- tai video-elementin sisälle. Data-attribuuttien avulla tunnistenumero voitiin lisätä suoraan kohteen div-elementin määrittelyihin, mikä mahdollisti sen, ettei tunnistenumeroa tarvinnut etsiä erikseen kohde-elementin sisältä. Seuraavassa esimerkissä on esitetty HTML4- ja HTML5-versioiden ero tunnistenumeroiden lukemisessa:

```
// HTML5:
<div class='yksittainenkuvadiv' draggable = "true"
data-kohde-id="299">

// JavaScript:
$(".yksittainenkuvadiv").data("kohde-id"));

// HTML4:
<div class='yksittainenkuvadiv ui-draggable'>
<input type='hidden'  class='kohde_id'  value='299' />

//  JavaScript:
```

```
$(".yksittainenkuvadiv").find("input.kohde_id").val()
```

Tunnistenumeroiden keräämiseen jälkeen kohteiden siirtäminen tapahtui samalla periaatteella kuin alkuperäisessäkin versiossa. Siirrettävien kohteiden ja kohdekansion tunnistenumerot lähetetään siirtoa käsittelevälle PHP-tiedostolle, joka suorittaa siirron ja päivittää sivun, jotta käyttäjä näkisi tehdyt muutokset. Seuraavassa esimerkissä on kuvattu erittäin yksinkertaistettuna drop-tapahtumassa tapahtuva siirto ($.ajax-kutsun kohde_id:llä tarkoitetaan kohdekansiota):

```
var siirrettavatKohteet = [];
$("div.valitutKohteet").each(function() {
  siirrettavatKohteet.push($(this).data("kohde-id"));
});
var siirrettavatKohteetJSON = JSON.stringify(siirrettavatKohteet);

$.ajax({
  type: "POST",
  url: "kohde_siirra",
  dataType: "html",
    data:    "siirrettavatKohteet    =    "    +    siirrettavatKohteetJSON    +
"&kohde_id= " + $(this).data("kohde-id")
            ...
});
```

4.3 Lomakkeet

Lomakkeiden luomisessa käytettiin samaa periaatetta kuin semanttisten elementtien kohdalla. Aluksi tehtiin FormHelper-niminen kantaluokka, jonka lisäksi luotiin FormHelperHTML4- ja FormHelperHTML5-nimiset lapsiluokat. FormHelper-luokan konstruktori (alustaja):

```
public static function FormHelperInit() {
  if(strcmp(HTML_VERSIO, "HTML5") == 0) {
```

```
        self::$formhelper = new FormHelperHTML5();
    }
    else if(strcmp(HTML_VERSIO, "HTML4") == 0) {
        self::$formhelper = new FormHelperHTML4();
    }
}
```

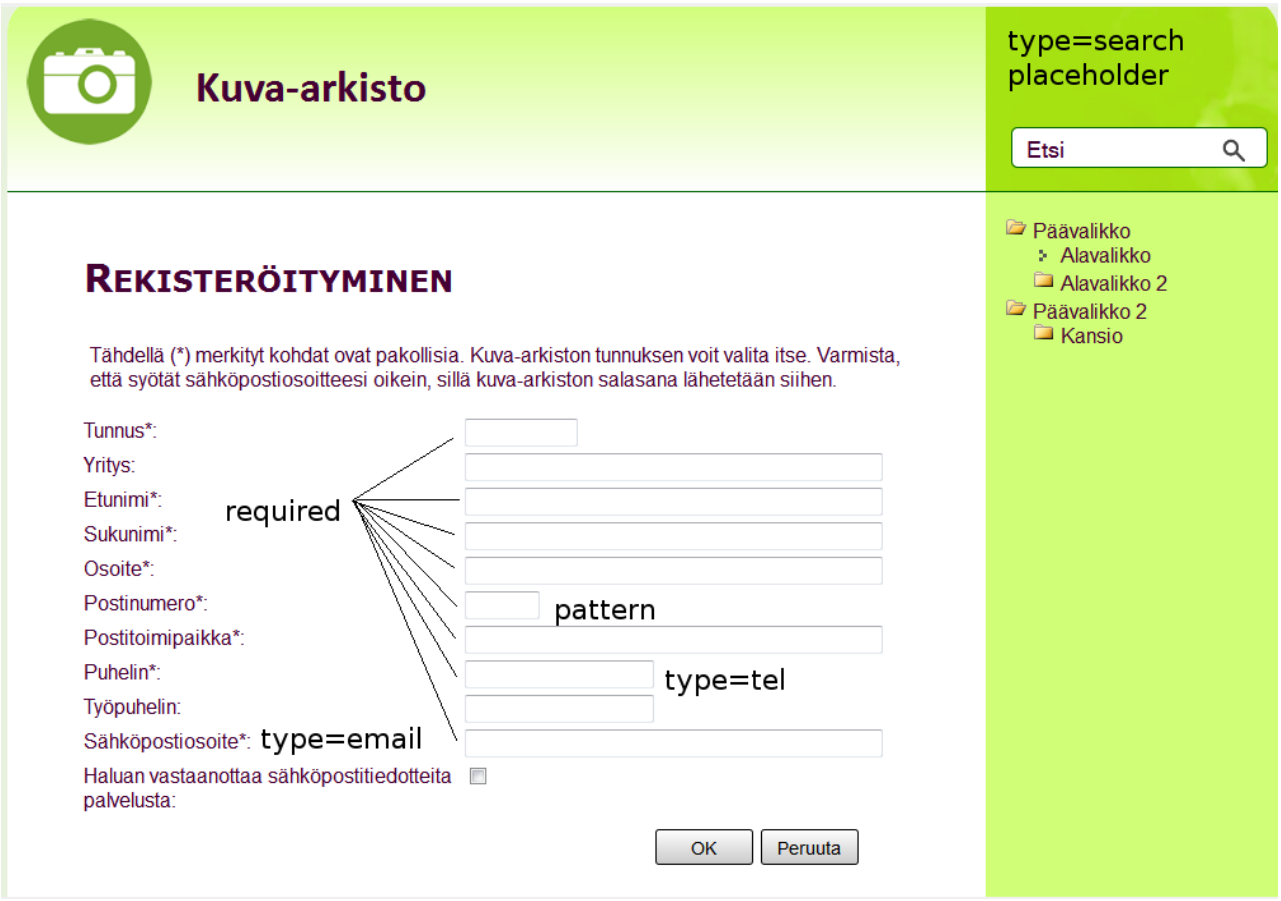
FormHelper-luokat sisälsivät funktioita lomakkeissa käytettävien elementtien, kuten esimerkiksi tekstikenttien, painikkeiden ja valintaruutujen luomiseen. FormHelper-luokkien metodeja kutsuttiin samalla tavalla kuin semanttisten elementtien tapauksessa. Jokaiselle lomake-elementille määriteltiin erikseen elementin attribuutit ja niiden arvot, jotka lähetettiin parametrina FormHelper-luokan metodille, joka lopuksi tulosti halutun elementin. Seuraavassa esimerkissä on esitetty rekisteröitymislomakkeen postinumero- ja puhelinkenttien luonti:

```
<?php
$lomakekentta = array("name" => "postinumero", "value" => $this->getValueVar('postinumero'), "required" => "required", "type" => "text", "size" => "5", "pattern" => "[0-9]{5}");
print FormHelper::form_input($lomakekentta);

$lomakekentta = array("name" => "puhelin", "value" => $this->getValueVar('puhelin'), "required" => "required", "type" => "tel" , "size" => "20");
print FormHelper::form_input($lomakekentta);
?>
```

HTML4- ja HTML5-versioiden välillä ei FormHelperin kohdalla lopulta ollut mainittavaa eroa. Koska vanhatkin selaimet osaavat muuntaa input-elementtien uudet tyypit (tel, email jne.) käyttäytymään teksti- eli text-tyypin tavalla, ei mitään erityisjärjestelyjä jouduttu tekemään (Lawson & Sharp 2010, 68-69). Ruvettaessa toteuttamaan monimutkaisempia asioita, kuten esimerkiksi lomakkeiden tai lomakekenttien validointia, jouduttaisiin sama toiminnallisuus luomaan erikseen JavaScriptin avulla.

HTML5:n uusista attribuuteista käytettiin rekisteröitymisen sekä käyttäjän tietojen muuttamisen yhteydessä requiredia ja patternia. Requiredilla määriteltiin tietyt tiedot pakollisiksi ja patternin avulla tarkastettiin käyttäjän syöttämän postinumeron oikeamuotoisuus. Attribuuttien lisäksi mainituissa lomakkeissa käytettiin uusista syötekenttätyypeistä teliä (puhelinnumero) ja emailia (sähköposti). Kuvagallerian hakukentässä käytettiin syötekentän tyyppinä searchia. Lisäksi hakukentälle asetettiin placeholder, eli vihjeteksti, jossa kerrottiin kentän olevan etsimistä varten. Kuviossa 4 on esitetty yhteenveto lomakkeisiin liittyvistä muutoksista.



KUVIO 4. Lomakkeissa käytetyt HTML5-uudistukset.

4.4 Videoelementti

Videoelementin voi yksinkertaisimmillaan luoda yhdellä tagilla. <video>-elementtiin lisätään vain src-attribuutti, joka kertoo toistettavan videotiedoston sijainnin. Selvyyden vuoksi videon sijainti oli kuitenkin hyvä sijoittaa erilliseen <source>-tagiin, joka laitettiin <video> … </video>-rakenteen sisälle. Sijainnin lisäksi <video>-tagiin lisättiin controls- ja autoplay-attribuutit, joista ensimmäinen määritteli

näkymään videon toimintopainikkeet (toisto, pysäytys, äänen voimakkuus) ja jälkimmäinen taas asetti videon toistumaan heti kun sivu oli ladattu. (Hickson 2012k, hakupäivä 19.3.2012.)

Toimeksiantajan kannalta <video>-tagista ei tällä hetkellä ollut kovinkaan paljon hyötyä, koska heidän käyttämäänsä Flash Video -formaattia ei tukenut yksikään testauksessa käytetyistä selaimesta. Jos formaattina taas olisi ollut esimerkiksi Youtubessa käytetty H.264 (.mp4:n ollessa säiliömuotona), ei kyseistä formaattia ole tällä hetkellä myöskään mahdollista saada toimimaan millään tavalla kaikissa selaimissa (Mozilla Developer Network 2012a, hakupäivä 19.3.2012).

Yksittäiselle <video>-tagille on mahdollistaa asettaa useampia lähteitä eli source-tageja. Tämän avulla on mahdollista toteuttaa sellainen toiminnallisuus, jossa selaimen ollessa kykenemätön toistamaan jotakin tiettyä formaattia, voidaan siirtyä seuraavaan source-tagiin niin kauan, että tuettu formaatti löytyy. Tarvittaessa <video>-tagin sisälle voidaan myös asettaa <embed>-tagi, jolla voidaan toistaa esimerkiksi H.264-videoita niissä selaimissa, joihin on asennettu Flash-selainlaajennus. Esimerkki käytöstä:

```
<video controls="controls" autoplay>
 <source src="video.mp4" type='video/mp4' />
 <source src="video.ogg" type='video/ogg' />
 <embed src="" ... ></embed>
</video>
```

Kehittäjän kannalta tämä tarkoittaa kuitenkin sitä, että jokaisesta videotiedostosta täytyisi olla monta erilaista versiota fyysisesti. Kuvagallerian tapauksessa tämä tapahtuisi luultavasti niin, että videon latausvaiheessa videosta luotaisiin apuohjelmien avulla useampi eri versio. Kyseisenlaisen toiminnon toteuttaminen vaatii kuitenkin oman aikansa, mikä ei taas sovi kovinkaan hyvin ideaan siitä, että <video>-tagin tulisi olla mahdollisimman helppokäyttöinen.

Toinen pieni huomion arvoinen asia <video>-elementissä on se, että jokaisen selaimen näkemys soittimen ulkoasusta on erilainen, joten käytännössä soittimen ulkoasu ja sen tyyli joudutaan määrittelemään erikseen (Pilgrim 2011b, hakupäivä 19.3.2012). Toisaalta tämän voi nähdä hyvänäkin asiana, koska ulkoasusta voi tehdä juuri sellaisen kuin itse haluaa. Kuviossa 5 on nähtävissä Chrome-selaimen oletusulkoasu HTML5:n videosoittimelle.

KUVIO 5. Chrome ja <video>-elementin oletusulkoasu.

5 POHDINTA

Opinnäytetyön tavoitteena oli tutkia ja selvittää kuinka HTML5:n uusia ominaisuuksia voisi hyödyntää toimeksiantajan kuvagalleriassa. Teoriaosuuden lisäksi opinnäytetyöhön kuului käytännön osuus, jossa valitut toiminnot toteutettiin kuvagalleriaan.

Jo alussa kävi selväksi, että työssä käytetyt tulisivat koostumaan pääosin verkkolähteistä. Tähän syynä oli se, että alustavaksi lähdemateriaaleiksi kaavailtujen paperilähteiden sisältämä tieto oli saatavilla helposti myös verkosta. Lisäksi HTML5-standardi ei ollut vielä saavuttanut lopullista muotoaan opinnäytetyön aikana, joten verkkolähteiden käyttäminen oli järkevää myös ajan tasalla pysymisen kannalta. Lähdeluettelossa mainittujen lähteiden lisäksi erinäiset ohjelmistoaiheiset keskustelupalstat ja yhteisöt olivat tärkeässä asemassa erityisesti käytännön työn aikana, koska W3C:n ja WHATWG:n viralliset standardit olivat ajoittain luonteeltaan liiankin teknisiä ja kaukana käytännöstä.

Opinnäytetyön sisälle rajattiin neljä suurempaa kokonaisuutta, joita olivat uudet semanttiset elementit, HTML5:n natiivi raahaus ja pudotus -rajapinta, lomakkeiden uudet ominaisuudet ja <video>-elementti. Näiden lisäksi pienenä lisäyksenä otettiin myöhemmin mukaan -data-attribuutit raahaus- ja pudotustoiminnon yhteydessä.

Semanttisten elementtien varsinaisessa ohjelmointityössä ei kohdattu suuremmin ongelmia, vaan toimintojen ja ominaisuudet toteutustavat löytyivät melko vaivattomasti. Myös lopuissakaan kolmessa kehitystehtävässä ei esiintynyt ohjelmoinnin osalta mainitsemisen arvoisia ongelmatilanteita. Vaikka opinnäytetyössä ei tarkasteltu kovinkaan tarkasti mahdollisia yhteensopivuusongelmia, ei kuvagallerian HTM4- ja HTML5-versioiden välillä ollut sen suurempia toiminnallisia tai ulkoisia eroja.

Semanttisissa elementeissä haasteita aiheuttivat eniten elementtien käyttötarkoitukset. Toteutuksen aikana huomattiin, että useimpien uusien elementtien käytölle ei löytynyt luotettavaa perustetta tämänhetkisillä tiedoilla tai käyttötottumuksilla, vaan monet elementeistä olivat tarkoitettu enemmänkin tekstiä sisältäville sivustoille kuin kuvagalleriatyyppisille sivuille. Opinnäytetyön teon aikana kuvagalleriaa vastaavien sivustojen löytäminen oli erittäin hankalaa, joten muiden toteutustavoista mallin tai suuntaviivojen ottaminen ei vielä onnistunut.

Lopulta uusia semanttisia elementtejä käytettiinkin vain muutamia, ja niitäkin pääasiassa vain sivupohjan määrittelyssä (<nav>, <header>, <footer>). Semanttiset elementit tulivat kuitenkin työn aikana kohtuullisen tutuiksi, joten niiden käyttäminen muissa yhteyksissä on luultavasti helpompaa ja selkeämpää kuin mitä se oli ennen opinnäytetyön aloittamista.

Raahaus- ja pudotus-rajapinnan kohdalla hyvää vertailupohjaa antoi aikaisemmin jQueryn avulla toteuttamani vastaava toiminto. Joitakin ulkoasuun liittyviä seikkoja lukuun ottamatta HTML5:llä ja jQueryllä luoduissa raahaus-ja pudotustoiminnoissa ei toimivuuden kannalta ollut mitään eroa, joten ainakaan siltä osin HTML5:stä ei löytynyt moitittavaa. Ulkoasua suuremmat erot löytyivät ohjelmistokoodista, mutta en henkilökohtaisesti kokenut, että kumpikaan näistä kahdesta toteutustavasta olisi ollut selkeästi toista helpompi tai selkeämpi. Erityisesti raahaus- ja pudotus-toiminnon toteutuksen aikana Firefox-selaimen Firebug-lisäosasta oli runsaasti hyötyä.

Lomakkeiden osalta selvästi hyödyllisimmiltä vaikuttivat uudet attribuutit, joiden avulla monia JavaScriptillä aiemmin toteutettuja asioita voi tehdä nyt suoraan ohjelmakoodista käsin. Pöytälaitteissa uusista <input>-tagin tyypeistä (type) ei ainakaan kuvagallerian kohdalla ollut havaittavaa eroa HTML4-versioon verrattuna, mutta esimerkiksi required-attribuutti vaikutti erittäin hyödylliseltä ja aikaa säästävältä verrattuna siihen, mitä sen toteuttamiseen olisi aikaisemmin vaadittu.

<video>-elementti osoittautui opinnäytetyön alun mielikuvien mukaiseksi. Elementti on ideana erittäin hyvä ja helppokäyttöinen, mutta eri selainten vaihtelevasta videoformaattien tuesta johtuen sen toimivuus ei tällä hetkellä ole läheskään riittävä.

Kuvagallerian jatkokehitystä ajatellen opinnäytetyön ulkopuolelle jääneistä HTML5:n uusista ominaisuuksista löytyisi hyvin todennäköisesti paljon lisääkin tutkittavaa. Myös opinnäytetyön teon aikana nousi esiin muutamia kiinnostavia kehityskohteita ja -ideoita. Raahaus- ja pudotustoimintoon olisi periaatteessa mahdollista lisätä työpöydältä useamman kohteen siirtämisen mahdollistava tiedonsiirto-ominaisuus HTML5:een kuuluvan File API:n avulla. <video>-elementin kohdalla taas voisi tutkia sitä, olisiko mahdollistaa muuttaa videotiedoston lataaminen sellaiseksi, että videoista luotaisiin latausvaiheessa useampi erilainen versio.

Yhteenvetona HTML5:n osalta voi todeta, että selainyhteensopivuudet kummittelivat edelleen vahvasti taustalla, vaikka yhteensopivuuksien painoarvo pyrittiinkin pitämään matalana.

Yhteensopivuusongelmien vastapainona HTML5:ssä oli lukuisia hyviäkin asioita. Erityisesti tähän opinnäytetyöhön valittujen kehitystehtävien kohdalla oma henkilökohtainen tuntemukseni oli se, että sisäänrakennettu tuki eri toiminnoille tai ominaisuuksille teki ohjelmoinnista yleisesti selkeämpää kuin aiempi lisäosien kanssa toimiminen.

Opinnäytetyö itsessään olisi ehkä kaivannut selkeämpää suunnitelmallisuutta, mutta aikataulun venymistä lukuun ottamatta työn aikana ei ilmennyt mitään suurempia ongelmia. Tulevaisuutta ajatellen on hyvää, että opinnäytetyöhön kuuluu myös raportointi, vaikkei raporttien luominen aina olekaan mitenkään erityisen innostavaa. Vaikka ohjelmoinnista voikin muodostua helposti kuva pelkkänä koodauksena, ovat erilaiset dokumentaatiot ja niiden luonti kuitenkin melko keskeinen osa ohjelmistokehitystä.

LÄHTEET

Alvares, M. 2010. Ambilight for video tag—prepare to cry. Hakupäivä 19.3.2012 http://beautifulpixels.com/web/ambilight-html5/.

Cheung, R. 2011. Javascript Frameworks and jQuery (Infographic). Hakupäivä 19.3.2012 http://www.webappers.com/2011/06/14/javascript-frameworks-and-jquery-infographic/.

Curtis, S. 2012. Mobile fragmentation drives HTML5 adoption: Kony. Hakupäivä 12.2.2012 http://news.techworld.com/mobile-wireless/3328005/mobile-fragmentation-drives-html5-adoption-kony/.

Deveria, A. 2012a. Compatibility tables for support of HTML5, CSS3, SVG and more in desktop and mobile browsers. Hakupäivä 12.2.2012 http://caniuse.com/#cats=HTML5.

Deveria, A. 2012b. Date/time input types. Hakupäivä 19.3.2012 http://caniuse.com/#feat=input-datetime.

Hickson, I. 2012a. History. Hakupäivä 22.2.2012 http://dev.w3.org/html5/spec/Overview.html#history-1.

Hickson, I. 2012b. Embedding custom non-visible data with the data-* attributes. Hakupäivä 12.2.2012 http://dev.w3.org/html5/spec/Overview.html#embedding-custom-non-visible-data-with-the-data-attributes.

Hickson, I. 2012c. HTML standard. Hakupäivä 18.2.2012 http://www.whatwg.org/specs/web-apps/current-work/.

Hickson, I. 2012d. The pattern attribute. Hakupäivä 19.3.2012 http://dev.w3.org/html5/spec/single-page.html#the-pattern-attribute.

Hickson, I. 2012e. The input element. Hakupäivä 19.3.2012 http://www.whatwg.org/specs/web-apps/current-work/#the-input-element.

Hickson, I. 2012f. Interactive elements. Hakupäivä 19.3.2012 http://www.whatwg.org/specs/web-apps/current-work/#interactive-elements.

Hickson, I. 2012g. The figure element. Hakupäivä 19.3.2012 http://www.whatwg.org/specs/web-apps/current-work/#the-figure-element.

Hickson, I. 2012h. The aside element. Hakupäivä 19.3.2012 http://www.whatwg.org/specs/web-apps/current-work/#the-aside-element.

Hickson, I. 2012i. Events summary. Hakupäivä 19.3.2012 http://www.whatwg.org/specs/web-apps/current-work/#dndevents.

Hickson, I. 2012j. The DataTransfer interface. Hakupäivä 19.3.2012 http://www.whatwg.org/specs/web-apps/current-work/#the-datatransfer-interface.

Hickson, I. 2012k. The video element. Hakupäivä 19.3.2012 http://dev.w3.org/html5/spec/single-page.html#the-video-element.

Hickson, I. 2012l. Is this HTML5?. Hakupäivä 19.3.2012 http://www.whatwg.org/specs/web-apps/current-work/#is-this-html5?.

Keith, J. 2010. HTML5 for Web Designers. New York: A Book Apart.

Lawson, B. & Sharp, R. 2011. Introducing HTML5. Berkeley: New Riders.

Lawson, B. 2012. The best of <time>s. Hakupäivä 19.3.2012 http://www.brucelawson.co.uk/2012/best-of-time/.

Lawson, B. 2010a. HTML5 articles and sections: what's the difference?. Hakupäivä 19.3.2012 http://www.brucelawson.co.uk/2010/html5-articles-and-sections-whats-the-difference/.

Lawson, B. 2010b. Introduction to HTML5 video. Hakupäivä 19.3.2012 http://dev.opera.com/articles/view/introduction-html5-video/.

Mozilla Developer Network. 2012a. Browser compatibility. Hakupäivä 19.3.2012 https://developer.mozilla.org/En/Media_formats_supported_by_the_audio_and_video_elements#Browser_compatibility.

Mozilla Developer Network. 2012b. Video. Hakupäivä 19.3.2012 https://developer.mozilla.org/en/HTML/Element/video.

Mozilla Developer Network. 2012c. HTML media element. Hakupäivä 19.3.2012 https://developer.mozilla.org/en/DOM/HTMLMediaELement.

Mozilla Developer Network. 2012d. Track. Hakupäivä 19.3.2012 https://developer.mozilla.org/en/HTML/Element/track.

Oracle. 2011. NetBeans IDE 7.0.1 Release Information. Hakupäivä 12.2.2012 http://netbeans.org/community/releases/70/index.html.

Osborne, J. 2009. The Footer Element Update. Hakupäivä 19.3.2012 http://html5doctor.com/the-footer-element-update/.

Palmer, S. 2004. Early History of HTML. Hakupäivä 18.3.2012 http://infomesh.net/html/history/early.

Pilgrim, M. 2011. Video on The Web. Hakupäivä 19.3.2012 http://diveintohtml5.info/video.html#what-works.

Pilgrim, M. 2011. At Last, The Markup. Hakupäivä 19.3.2012 http://diveintohtml5.info/video.html#markup.

Ranganathan, A. & Sicking, J. 2011. File API. Hakupäivä 19.3.2012 http://dev.w3.org/2006/webapi/FileAPI/.

Theseus. 2012. Haku. Hakupäivä 18.3.2012 https://publications.theseus.fi/search?query=html5&submit=Hae.

Van Kesteren, A. & Pieters, S. 2012a. HTML5 differences from HTML4. Hakupäivä 12.2.2012 http://www.w3.org/TR/html5-diff/.

Van Kesteren, A. & Pieters, S. 2012b. New attributes. Hakupäivä 19.3.2012 http://www.w3.org/TR/html5-diff/#new-attributes.

Van Kesteren, A. & Stachowiak, M. 2007a. HTML Design Principles. Hakupäivä 18.3.2012 http://www.w3.org/TR/html-design-principles/.

Van Kesteren, A. & Stachowiak, M. 2007b. Utility. Hakupäivä 18.3.2012 http://www.w3.org/TR/html-design-principles/#utility.

Van Kesteren, A. & Stachowiak, M. 2007c. Interoperability. Hakupäivä 18.3.2012 http://www.w3.org/TR/html-design-principles/#interoperability.

Van Kesteren, A. & Stachowiak, M. 2007d. Universal Access. Hakupäivä 18.3.2012 http://www.w3.org/TR/html-design-principles/#universal-access.

W3C HTML Working Group. 1999a. The global structure of an HTML document. Hakupäivä 12.2.2012 http://www.w3.org/TR/html4/struct/global.html#h-7.5.

W3C HTML Working Group. 1999b. Grouping elements: the DIV and SPAN elements. Hakupäivä 14.2.2012 http://www.w3.org/TR/html401/struct/global.html#h-7.5.4.

W3C HTML Working Group. 1999c. Text. Hakupäivä 18.3.2012 http://www.w3.org/TR/html401/struct/text.html.

W3C HTML Working Group. 1999d. Forms. Hakupäivä 19.3.2012 http://www.w3.org/TR/html401/interact/forms.html.

W3C HTML Working Group. 2002. Why the need for XHTML?. Hakupäivä 22.2.2012 http://www.w3.org/TR/xhtml1/#why.

W3C. 2011a. FAQs – HTML Wiki. Hakupäivä 12.2.2012 http://www.w3.org/html/wiki/FAQs#Does_HTML5_provide_for_Royalty-Free_video_and_audio_codecs.3F.

W3C. 2011b. HTML Working Group Charter. Hakupäivä 18.3.2012 http://www.w3.org/2007/03/HTML-WG-charter.html.

WHATWG. 2012. When will HTML5 be finished?. Hakupäivä 18.3.2012 http://wiki.whatwg.org/wiki/FAQ#When_will_HTML5_be_finished.3F.

Winokur, D. 2011. Flash to Focus on PC Browsing and Mobile Apps; Adobe to More Aggressively Contribute to HTML5. Hakupäivä 12.2.2012 http://blogs.adobe.com/conversations/2011/11/flash-focus.html.