JAVASCRIPT-KEHITYSKIRJASTOJEN VERTAILU

HAMK
HÄMEEN AMMATTIKORKEAKOULU

Ammattikorkeakoulututkinnon opinnäytetyö

Tietojenkäsittely

Hämeenlinna, 25.3.2010

Mikko Mäkelä

HAMK
HÄMEEN AMMATTIKORKEAKOULU

OPINNÄYTETYÖ

Tietojenkäsittely
Hämeenlinna

Työn nimi	JavaScript-kehityskirjastojen vertailu
Tekijä	Mikko Mäkelä
Ohjaava opettaja	Lasse Seppänen
Hyväksytty	_____._____.20_____
Hyväksyjä	

HAMK
HÄMEEN AMMATTIKORKEAKOULU

TIIVISTELMÄ

HÄMEENLINNA
Tietojenkäsittely
Systeemityö

Tekijä Mikko Mäkelä **Vuosi** 2010

Työn nimi JavaScript-kehityskirjastojen vertailu

TIIVISTELMÄ

Opinnäytetyön tarkoituksena oli selvittää, mikä JavaScript-kehityskirjasto sopii työn toimeksiantajan, Logia Software Oy:n, ohjelmistokehitykseen parhaiten.

Tutkimusmenetelmänä oli vertaileva tutkimus, jossa käytettiin ohjaavaa metodia. Työn käytäntö sisältää jokaisella kolmelle kehityskirjastolla tehdyt kolme JavaScript-komponenttia, jollaisia Logia Softwaren järjestelmässä tullaan käyttämään. Vertailu perustuu niihin havaintoihin, joita tuli esiin toteutettaessa kyseisiä komponentteja.

Työn teoria käsittelee kehityskirjastoja yleisesti ja esittelee JavaScript-kielen sekä AJAX-tekniikan. JavaScript-kehityskirjastot perustuvat JavaScript-kieleen ja yksi kirjastojen tärkeimpiä ominaisuuksia on helpottaa AJAX-kehitystä. Aineistona käytettiin suurimmaksi osaksi kirjalähteitä, mutta myös joitakin verkkodokumentteja.

Opinnäytetyön vertailun tuloksena voidaan esittää, että vertailluista kehityskirjastoista jQuery sopii parhaiten Logia Softwaren www-kehitykseen. Se oli vertailukriteerien perusteella selvästi parempi kuin kaksi muuta kehityskirjastoa. JQuery on tällä hetkellä suosituin kirjasto, jolla on suuri käyttäjäkunta, toimiva kehitysyhteisö ja se nopeuttaa www-kehitystä muista kehityskirjastoista selvästi eroavalla syntaksillaan.

Avainsanat javascript, kehityskirjasto, ajax-ohjelmointi.

Sivut 27 s.

HAMK
UNIVERSITY OF APPLIED SCIENCES

HÄMEENLINNA
Business Information Technology
Software Engineering

Author Mikko Mäkelä **Year** 2010

Subject of Bachelor's thesis Comparing of JavaScript libraries

ABSTRACT

The purpose of this thesis was to find out which JavaScript library is the most suitable for software development at Logia Software Oy, which commissioned this work.

The research method is a comparative study using a directional method. The practical part of the work includes three JavaScript components which were made using each of the three JavaScript libraries. The aim of the research was to clarify how these JavaScript components work with the systems of Logia Software. Comparison was based on the observations that came out when implementing these components.

The theoretical background deals with frameworks in general and presents the JavaScript language and the AJAX technique. The JavaScript libraries are based on the JavaScript language, and one of the most important features of the libraries is to facilitate AJAX development. The main sources of information were relevant literature but also some www documents.

The result of the research proves that of the compared JavaScript libraries, jQuery suits best for the development of Logia Software. On the basis of the comparison criteria, jQuery was clearly better than the other two JavaScript libraries. At the moment jQuery is the most popular library with a large number of users and an efficient community, and it makes www development distinctly different from other libraries due to its coding syntax.

Keywords JavaScript, programming, AJAX, library.

Pages 27 p.

SANASTO

HTML	HyperText Markup Language on hypertekstidokumenttien merkkauskieli. Käytännössä kaikki www-sivut on tehty HTML-kielellä. Se on tarkoitettu käytettäväksi muiden web-tekniikoiden ja ohjelmien kanssa.
CSS	Cascading Style Sheets, tyylitiedostoilla määritellään dokumentin ulkoasuun vaikuttavat asiat. Tyylitiedostoja käyttämällä voidaan erottaa rakenne ja ulkoasu toisistaan.
DHTML	DHTML tai Dynaaminen HTML on yhdistävä termi neljälle tekniikalle: HTML, CSS, JavaScript ja DOM. Näin saadaan luotua staattisen HTML-sivun sijaan dynaamista toimintaa käyttäjän toimintojen mukaisesti.
JavaScript	JavaScript on suunniteltu yleiskäyttöiseksi ja kevyeksi skriptikieleksi erilaisiin tarpeisiin. Pääasiassa kieltä käytetään www-selaimessa.
Avoin lähdekoodi	Avoin lähdekoodi merkitsee sitä, että käyttäjä näkee ohjelman koodin ja voi lisenssistä riippuen muokata ja julkaista koodia uudelleen tarpeidensa mukaisesti, kunhan vain alkuperäiset tekijänoikeusmerkinnät säilyvät koodissa.
DOM	Document Object Model, dokumenttioliomalli on W3C:n standardi, joka määrittelee ohjelmointikieliriippumattoman ohjelmointirajapinnan HTML ja XML-dokumenteille.
AJAX	Asynchronous JavaScript and XML on tekniikka, joka yhdistää vanhempia tekniikoita lähettämään ja palauttamaan data www-selaimen ja www-palvelimen välillä.
ActiveX	Microsoft IE-selaimessa käytettävät ActiveX-komponentit toimivat, kuten käyttöliittymässä suoritettavat ohjelmat.
W3C	Word Wide Web Consortium koostuu liikeyrityksistä, tutkimuslaitoksista ja jäsenorganisaatioista, joiden tavoitteena on kehittää ja standardoida web-tekniikoita.
XML	Extensible Markup Language on tekniikka tekstimuotoisten dokumenttien merkkaamiseen ja käsittelyyn. Se on yhteinen tietorakenne, josta tieto on luettavissa helpolla ja yksinkertaisella tavalla.

SISÄLLYS

1 JOHDANTO

Opinnäytetyön tarkoituksena on selvittää, mikä JavaScript-kehityskirjasto sopii parhaiten Logia Software Oy:n ohjelmistokehitykseen. Logia Software Oy:n varastohallintajärjestelmä toimii www-selaimella ja on käytännössä normaali www-sivu. Varastonhallintajärjestelmän käyttäjät ovat tottuneet käyttämään www-sivuja, joilla on parannettu käytettävyyttä JavaScriptin ja AJAXin avulla. Tämän vuoksi he odottavat varastonhallintajärjestelmästäkin löytyvän näitä nykyaikaisia toiminnallisuuksia. Tästä syntyi tarve selvittää JavaScript-kehityskirjastojen mahdollisuuksia ja niiden sopivuutta yrityksen käyttöön. Tavoitteena on tehdä varastonhallintajärjestelmän toiminnasta nykyaikaista ja helpottaa sen käyttöä.

Tutkimusongelmana on selvittää, mikä JavaScript-kehityskirjasto sopii valittujen kriteerien perusteella parhaiten Logia Softwaren ohjelmistokehitykseen. Pääkriteereitä kehityskirjastojen vertailussa ovat kehittäjäyhteisö, käyttö www-sivulla ja yleiskäyttöisyys.

JavaScript-kehityskirjastot ovat JavaScript-ohjelmoinnissa käytettäviä valmiita funktiokirjastoja, joiden tarkoituksena on nopeuttaa ja selkeyttää kehitystä. Sen sijaan, että jokainen kehittäjä kirjoittaisi omat versionsa funktioista, voidaan käyttää yhteistä kehityskirjastoa, jolloin kaikilla on käytössä samat komponentit. Näin koodien versiot ovat yhtenäiset ja säilytyksessä samassa paikassa.

JavaScript-kehityskirjastojen kehityksestä vastaavat yhteisöt, jotka parantavat entisiä ja luovat uusia komponentteja kirjastoihin. Kun yksittäiset ohjelmoijat kehittävät uusia komponentteja tai muokkaavat vanhoja, yhteisö arvioi, olisiko komponenteista hyötyä muillekin. Jos on, kyseiset komponentit lisätään kirjastoon. Kehityskirjastosta julkaistaan tällöin uusi versio, jonka muut kehittäjät ympäri maailmaa voivat ottaa käyttöön.

JavaScript-kehityskirjastoja käytetään www-selaimella toimivien ohjelmistojen ja www-sivujen kehityksessä. Tämä on niin sanottua asiakaspuolen ohjelmointia, koska toiminnallisuus tapahtuu käyttäjän selaimessa eikä palvelimella.

2 JAVASCRIPT-KEHITYSKIRJASTOT

JavaScript-kehityskirjastot ovat ohjelmistokehityksessä käytettäviä funktiokirjastoja. Ne sisältävät valmiita funktioita, joita voidaan tarpeen mukaan ottaa käyttöön ohjelmissa. Kehityskirjastot sisältävät myös erilaisia työkaluja ja oikopolkuja kehityksen nopeuttamiseksi. Lisäksi monet niistä tarjoavat kehittyneet mahdollisuudet www-käyttöliittymien ulkoasun luomiseen.

2.1 Kehityskirjastoja on moneen käyttötarkoitukseen

JavaScript-kehityskirjastoja on kymmeniä, ehkä jopa satoja erilaisia, riippuen siitä, mitkä kaikki lasketaan mukaan. Aluksi kirjastot olivat pieniä funktiokokoelmia yksittäisen ohjelmoijan työn nopeuttamiseksi. Kun nämä kirjastot alkoivat levitä toisille ohjelmoijille, niihin lisättiin uusia ominaisuuksia ja niiden koko kasvoi. Lopulta yhtä kirjastoa kehitti yhteisö, jolla oli johtohahmot ja yhteinen tavoite.

Yhteisön tarkoitus ei ole tuottaa voittoa vaan lähes kaikki kehityskirjastot ovat ilmaisia. Niitä kehitetään avoimen lähdekoodin mukaisin opein, joten kirjaston käyttöön ottava yhteisön ulkopuolinen kehittäjä saa vapaasti tehdä kirjastolla myös kaupallisia sovelluksia ilman korvausvelvoitetta.

Monet kehityskirjastoista ovat erikoistuneet tiettyyn tehtävään ja pysyneet melko pieninä. Jotkin kirjastot ovat laajentuneet yleiskäyttöisiksi, jolloin yhdellä kirjastolla voi tehdä kaiken, mitä www-käyttöliittymässä tarvitaan. Tässä työssä vertailuun on valittu kolme suurta ja yleiskäyttöistä kehityskirjastoa.

2.2 Kehityskirjastojen käytöstä saatavat edut

Yksi suurimpia syitä JavaScript-kehityskirjastojen käyttöön on tarve saada jokaisessa koodissa toistuvat ratkaisut yhteen pakettiin, joka olisi helppo ottaa käyttöön. Tällä tavalla kehittäjä voi keskittyä sovelluksen kannalta olennaisiin toimintoihin eikä hänen aikaansa kulu perustoimintojen toteuttamiseen.

Eri selaimet, kuten Microsoft IE-selain ja Mozilla Firefox, käsittelevät HTTP-pyyntöä eri tavalla. Tätä varten kehittäjän täytyy kirjoittaa jokaiseen projektiin erikseen tunnistusfunktio, joka tarkistaa käytössä olevan selaimen ja lähettää sen mukaisesti HTTP-pyynnön. JavaScript-kehityskirjastoa käytettäessä tunnistusfunktio on valmiiksi rakennettuna kirjastossa, eikä kehittäjän tarvitse puuttua sen toimintaan. Se on siis automaattisesti käytössä vaikka kehittäjä ei sitä varten kirjoita erillistä koodia. Kehityskirjaston käyttö sekä nopeuttaa kehitystä että lyhentää koodi-

rivien määrää, koska monet päivittäin käytettävät toiminnot on tehty valmiiksi funktioiksi kirjastoon. (Heilmann 2006, 419.)

Toinen esimerkki kehityskirjastojen eduista on JavaScript-koodin suoritus dokumentin latauduttua. Normaalisti JavaScript-koodi suoritetaan heti, kun sivu latautuu, jolloin voi käydä niin, ettei HTML-rakenne ole vielä valmis. Tämä aiheuttaa virheen jos suoritettavan JavaScript-koodin tarkoitus on lisätä taulukkoon yksi rivi, mutta taulukkoa, johon rivi tulisi lisätä, ei ole vielä luotu.

Tähän tarkoitukseen on mahdollista käyttää body-elementin onload-tapahtumankäsittelijää, joka on esitetty kuvassa 1. Sen käytössä on kuitenkin kaksi huonoa puolta. Sivun latautumista odotetaan kunnes kaikki sivun elementit ovat valmiina. Jos sivulla on isoja kuvia, niiden täytyy ensin latautua ja vasta sen jälkeen body-elementissä oleva onload-tapahtumankäsittelijä aktivoituu. Toinen ongelma liittyy siihen, että body-elementissä oleva tapahtumankäsittelijä aktivoituu vain kerran, kun sivu latautuu. Onload-tapahtumankäsittelijälle voi antaa kyllä useampia funktioita suoritettavaksi, kuten on tehty kuvassa 1, mutta kyseiset funktiot suoritetaan vain kerran sivun latauduttua.

```
1  <body onload="funktio_kutsu(); toinen_funktio_kutsu();">
```

KUVA 1 *Body-elementin onload-tapahtumankäsittelijä*

Tähän ongelmaan löytyy ratkaisu monista JavaScript-kehityskirjastoista. Kuvassa 2 on esimerkki jQueryn tavasta ratkaista ongelma. Kaikki suoritettava koodi kirjoitetaan document.ready() -tapahtuman sisään. Tällöin sivun latautuessa odotetaan, että sivun DOM-rakenne on valmis, jonka jälkeen suoritetaan funktiot, jotka ovat tämän rakenteen sisässä. Tämä tapa ei kuitenkaan odota esimerkiksi kuvien latautumista, joten sivun toiminta nopeutuu. Lisäksi näitä $(document).ready() -tapahtumia voi olla yhdellä sivulla useampia ja ne kirjoitetaan script-elementtien sisään.

```
1  $(document).ready(function() {
2      // jQuery-koodia
3  });
```

KUVA 2 *jQuery document.ready()-tapahtuma*

Käytännössä jQuery-tapahtuman taustalla on kymmeniä rivejä koodia, jonka tietysti voisi liittää jokaiseen html-dokumenttiin erikseen ilman kehityskirjaston käyttöä. Mutta oleellista onkin juuri se, miten kymmenien rivien mittainen koodi saadaan aktivoitua kahdella rivillä koodia. Sama lyhentäminen toistuu kaikessa kehityskirjaston toiminnassa. Käytännössä JavaScript on taustalla, mutta sitä käskytetään ikään kuin omalla skriptikielellä. Kehittäjä kirjoittaa vähemmän koodia, mutta saa aikaan samat toiminnot kuin kirjottaisi kaiken suoraan JavaScriptillä.

2.3 Kirjastojen käytöstä aiheutuvat ongelmat

JavaScript-kehityskirjastojen käyttöön liittyy joitakin ongelmia tai vähintään asioita, joita tulee miettiä ennen niiden käyttöönottoa. Kehityskirjastoa ei kannata ottaa käyttöön vain jonkin yksittäisen toiminnon toteuttamiseksi, vaan on syytä suunnitella etukäteen, miten paljon kirjastolle lopulta tulee käyttöä. Vaikka kehityskirjaston käyttö tuokin monia oikoteitä kehitykseen, on jonkin yksittäisen komponentin tekeminen järkevämpää suoraan JavaScriptillä ilman kirjastoa.

Ensimmäinen ongelma on kehityskirjaston lisäämä välitaso kehitykseen ja kirjaston oma syntaksi. Kun perinteisessä JavaScript-kehityksessä on vain HTML-koodi ja JavaScript-koodi, tuo kirjaston käyttö HTML:n ja JavaScript:n väliin oman tasonsa. Tästä tulee ongelma siinä vaiheessa, kun projektiin tulee mukaan uusi ohjelmoija, joka ei ole aikaisemmin käyttänyt koodikirjastoa. Uusi kehittäjä hallitsee sujuvasti JavaScriptin syntaksin, mutta projektissa käytetäänkin kirjaston omaa syntaksia, jolloin kehittäjä joutuu opettelemaan kokonaan uuden tavan kirjoittaa koodia. Tämä hidastaa kehitystä ja vaikeuttaa uusien henkilöiden mukaan ottamista projektiin. (Heilmann 2006, 426.)

Toinen ongelma on kehityskirjastojen dokumentointi, jossa esitellään kirjastojen komponenttien toimintaa, mutta niissä ei välttämättä huomioida, mitä tapahtuu jos käyttäjän selaimessa ei ole JavaScript päällä. Kehityskirjastojen tarkoitus on nopeuttaa ja helpottaa kehitystä, mutta ei tehdä kehittäjistä ja sivustoista riippuvaisia kirjastosta ja JavaScriptistä. (Heilmann 2006, 427.)

3 JAVASCRIPT

JavaScript on olio-pohjainen ohjelmointikieli, mikä on tehty helpottamaan vuorovaikutteisuuden kehittämistä www-sivuille. Kielellä voi luoda erilaisia toiminnallisuuksia ja tarkistuksia HTML-elementteihin, kuten tekstikenttiin ja nappeihin. JavaScript sisältää erilaisia tapahtumankäsittelyyn luotuja valmiita funktioita, joihin www-käyttöliittymän toiminta perustuu.

3.1 Skriptikieli

JavaScript luokitellaan skriptikieliin. Skriptikielellä tarkoitetaan ohjelmointikieltä, jota käytetään muokkaamaan, käsittelemään ja automatisoimaan jonkin toimivan järjestelmän palveluita. Skriptikielen ja sovellusohjelmointikielen erona on tulkattavuus. Sovellusohjelmointikielellä kirjoitettu lähdekoodi joudutaan ensin kääntämään binäärimuotoiseksi ohjelmaksi, ennen kuin sitä voidaan käyttää. Vastaavasti JavaScript-sovellus toimii, kun sen lähdekoodi tulkataan esimerkiksi selaimessa tai jossakin muussa JavaScript-tulkin sisältävässä ohjelmassa. JavaScriptillä tehty ohjelma ei siis ole itsenäinen sovellus, eikä sillä voida luoda uutta järjestelmää, kuten sovellusohjelmointikielillä. (Peltomäki & Nykänen 2006, 96-97.)

JavaScript-koodi voidaan kirjoittaa suoraan HTML-dokumenttiin tai erilliseen tiedostoon, joka liitetään HTML-dokumenttiin. Www-selaimen sisäänrakennettu tulkki tulkitsee koodin, kun sivu avataan. Eri selaimissa on kuitenkin erilaiset JavaScript-tulkit, eikä yhtenäistä standardia ole. Tämä on ongelma erityisesti vanhojen selaimien kanssa toimittaessa, uusimmissa selaimissa ongelmia ei juuri ilmene. (Peltomäki & Nykänen 2006, 101.)

Skriptikielet ovat periaatteessa alustariippumattomia, koska ne tarvitsevat toimiakseen vain tulkin. Vastaavasti binäärimuotoiset ohjelmat ovat tarkkoja alustasta, jolloin eri käyttöliittymille täytyy olla omat versiot ohjelmasta. Skriptikielet ovat kevytrakenteisia ja tästä syystä ominaisuuksiltaan rajoittuneita, jos niitä verrataan sovellusohjelmointikieliin. Tämä ei kuitenkaan ole ongelma nykyään, koska koneiden laskuteho riittää tulkkaukseen vaikka kyseessä olisi laajempikin sovellus. Lisäksi kevytrakenteisuus tekee niistä yksinkertaisia ja näin ollen helpompia oppia. (Pohjois-Karjalan Aikuisopisto Lieksan yksikkö 2009.)

3.2 Olio-ohjelmointi

JavaScript on hybridikieli, jolla tarkoitetaan mahdollisuutta käyttää perinteistä lausekielistä ohjelmointitapaa tai modernia olio-ohjelmointitapaa. JavaScriptissä merkkijonot, numerot ja funktiot ovat olioita. Selainohjelmointi JavaScriptillä perustuu suurimmaksi osaksi valmiisiin olioihin, omien olioiden kirjoittaminen on myös mahdollista. Ohjelmoija voi vaihtoehtoisesti kirjoittaa myös funktioita. (Peltomäki & Nykänen 2006, 107.)

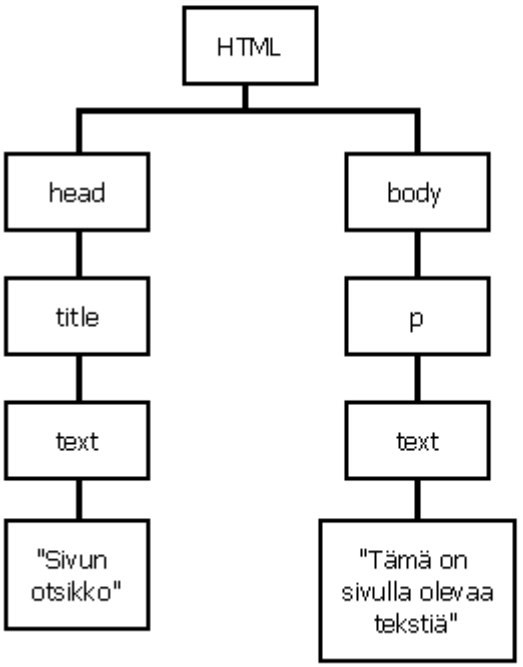
3.3 Dokumenttioliomalli

Dokumenttioliomalli, lyhennettynä DOM, tulee sanoista Document Object Model. Se on W3C:n määrittelemä ohjelmointikieliriippumaton ohjelmointirajapinta HTML- ja XML-dokumenteille. Malli jakaa dokumentin puurakenteeseen, jonka sisältöä ja elementtien järjestystä voidaan muuttaa. DOM-mallilla on ollut suuri merkitys JavaScript-kielen yleistymisessä. (Peltomäki & Nykänen 2006, 97.)

3.3.1 Puurakenne

Puurakenteen jokainen elementti on olio ja oliolla on erilaisia ominaisuuksia. Oliota voidaan kutsua sijainnista huolimatta missä tahansa kohdassa dokumenttia. Vaikka dokumentin elementtien järjestystä muutettaisiin, viittaus toimii edelleen. Kuvassa 3 on esitetty, miten HTML-dokumentti muodostaa DOM-mallin mukaisen puurakenteen. (Peltomäki 2001, 244-245.)

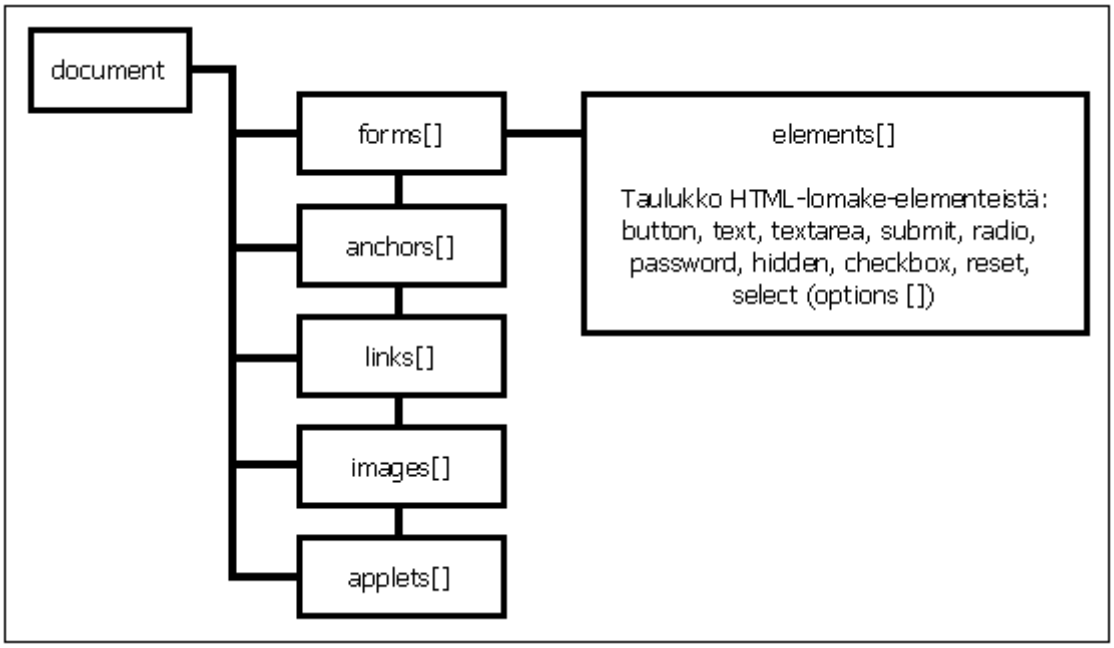
```
1  <html>
2  <head>
3      <title>Sivun otsikko</title>
4  </head>
5  <body>
6      <p>Tämä on sivulla olevaa tekstiä</p>
7  </body>
8  </html>
```



KUVA 3 *HTML-dokumentin muodostama DOM-puurakenne*

3.3.2 DOM-versiot

Dokumenttioliomallia kehitetään jatkuvasti lisää ja valmisteilla on DOM 3 -standardi. Tällä hetkellä kehittäjien on noudatettava kuitenkin DOM 2 –standardia, joka on käytössä uusimmissa selaimissa. Ensimmäinen DOM-standardi oli DOM 1, mutta ennen sitä käytettiin DOM 0 -dokumenttioliomallia. DOM 0 oli pohjana seuraaville DOM-määrityksille, joten sitä pidetään teollisuusstandardina. Kuvassa 4 on kuvattu DOM 0 document-olion luokkahierarkia. (Peltomäki & Nykänen 2006, 183.)



KUVA 4 *HTML-dokumentin muodostama DOM-puurakenne*

DOM 1 -standardi julkaistiin vuonna 1998 ja se määrittelee, miten elementit välittävät tietoa toisilleen HTML- tai XML-dokumenteissa. Parannus DOM 0 -standardiin verrattuna oli mahdollisuus viitata dokumentin jokaiseen elementtiin käyttäen DOM-rajapinnan metodeja. (Peltomäki & Nykänen 2006, 175.)

DOM 2 -standardi on modulaarinen, jolloin toteutukseen voidaan valita vain tarvittavat moduulit. Ainoa pakollinen moduli on Core, joka sisältää dokumentin peruspuurakenteen. DOM 2 myös yhtenäisti erityisesti selainten tapahtumankäsittelyä. Edelleen laajasti käytetty selain Microsoft IE6 ei kuitenkaan tue Event-modulia, jossa tapahtumankäsittely määritellään. (Peltomäki 2001, 25.)

3.3.3 IDL-rajapinta

Interface Definition Language on kieli, jonka avulla sovellukset ja niiden sisältämät oliot voivat kommunikoida toistensa kanssa huolimatta siitä, millä kielellä ne on kirjoitettu. IDL-rajapinnat eivät siis ole riippuvaisia ohjelmointikielestä. Näin ollen on mahdollista kirjoittaa DOM-rakenteen muodostava ohjelma Java-kielellä ja IDL-rajapinnan ansiosta dokumenttia voidaan muokata JavaScriptillä tai muulla IDL-rajapintaa käyttävällä kielellä. (Peltomäki 2001, 249.)

3.4 Tapahtumankäsittely

Graafiset käyttöliittymät perustuvat tapahtumiin ja niihin reagoimiseen. Oli käyttöliittymä sitten työpöytäohjelmassa tai www-sivulla, sen tapahtumaketju on seuraavanlainen:

- Näytetään käyttöliittymä
- Odotetaan, että jotakin tapahtuu
- Reagoidaan tapahtumaan määritellyllä tavalla
- Toistetaan

Ensimmäinen vaihe määrittelee käyttöliittymän ulkoasun ja kolme seuraavaa vaihetta määrittelevät toiminnot. Www-käyttöliittymissä www-selain hoitaa ensimmäisen vaiheen ja sivulle liitetyt skriptit tekevät sivulle toiminnan. (Bibeault & Katz 2008, 83.)

Tapahtumat voivat olla ilman käyttäjän toimia käynnistyviä kuten erilaiset laskurit. Suurin osa tapahtumista on kuitenkin käyttäjän toimintaan perustuvia. Kun käyttäjä esimerkiksi painaa hiiren nappia jonkin elementin päällä, siitä muodostuu tapahtuma, johon ohjelma reagoi määritellyllä tavalla. Näitä tapahtumia varten JavaScriptiin on rakennettu tapahtumankäsittelijä. Tapahtumankäsittelyfunktioksi kutsutaan sitä funktiota, jonka tapahtumankäsittelijä käynnistää.

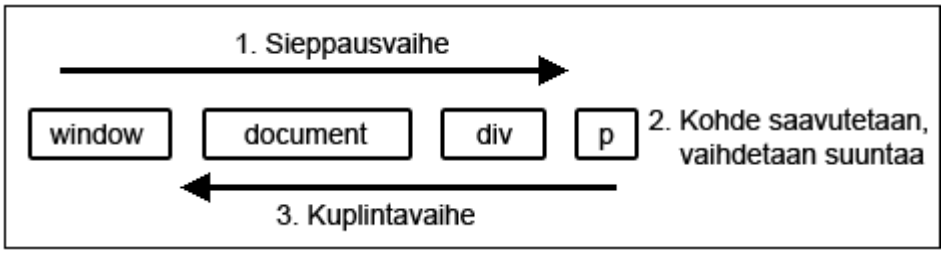
3.4.1 Tapahtumankäsittelyn kehittyminen www-selaimissa

Tapahtumankäsittely toimi perustasolla jo DOM 0-mallissa, mutta DOM 2 -malli standardoi sen. DOM 0 -mallin tapahtumankäsittelyssä tapahtumat sijoitettiin suoraan elementtiin ja elementtien määrä oli rajoitettu. Esimerkiksi napin painaminen käynnisti tapahtuman, mutta tekstikentän painaminen ei. (Peltomäki & Nykänen 2006, 216-217.)

Varsinainen tapahtumankäsittelyn standardi tuli vasta DOM 2 -mallin mukana. Kun DOM 0 ja DOM 1 -malleissa elementillä saattoi olla vain yksi tapahtuma, DOM 2 -malli lainaa Javassa käytettyä tapahtumankuuntelijaa. Tämä erikoisolio määrittelee, mitä ohjelma tekee, kun tapahtuma aiheutuu. Tapahtumankuuntelija liitetään elementtiin, jolloin se kuuntelee sille kuunneltaviksi määriteltyjä tapahtumia. Elementille saapuva tapahtuma käynnistää jonkin toiminnon, joka on yleensä funktio. (Peltomäki & Nykänen 2006, 230.)

3.4.2 DOM 2 -mallin tapahtumankäsittely

DOM 2 -mallissa tapahtumankäsittely on rakennettu Event-moduuliin, jossa yhdistyy tapahtuman sieppaaminen ja kupliminen. Sieppaamisella tarkoitetaan toimintoa, joka lähtee liikkeelle window-oliosta ja jatkaa matkaa kohde-elementtiin. Kupliminen on vaihe, jossa tapahtuma saavuttaa kohteen ja kääntyy takaisin, palaa samaa reittiä kuin tulikin ja päätyy lopulta selainikkunaan. Kuvassa 5 on mallinnettu tapahtumankäsittely, kun kohde on div-elementin lapsielementtinä oleva p-elementti.



KUVA 5 *DOM 2-mallin mukainen tapahtumankäsittely*

Suurin ongelma muuten hyvin toimivassa tapahtumankäsittelyssä on Microsoft IE 6-selain, joka ei tue Event-moduulin käyttöä. Selain on edelleen melko laajasti käytetty, vaikka onkin tekniikaltaan jo vanhentunut.

4 AJAX

AJAX muodostuu sanoista Asynchronous JavaScript and XML. Se ei ole varsinaisesti oma tekniikka vaan tapa yhdistää JavaScript- ja XML-tekniikat toimimaan yhdessä. Lisäksi mukaan lasketaan myös CSS-tyylitiedostot ja DOM-malli. AJAXia käytetään datan lähettämiseen ja vastaanottamiseen www-palvelimen ja www-selaimen välillä.

AJAX eroaa tavallisesta www-sovelluksesta siinä, että tekniikkaa käyttävä sovellus voi lähettää ja vastaanottaa dataa osissa, jolloin koko sivua ei ladata uudelleen. Perinteisessä mallissa sovellus lähettää palvelimelle dataa ja palvelin palauttaa takaisin kokonaan uuden sivun. Tämä malli ei toimi dynaamisien www-sovelluksien kanssa. (Crane & Pascarello 2006, 33.)

4.1 Tekniikan keskeisin olio XMLHttpRequest

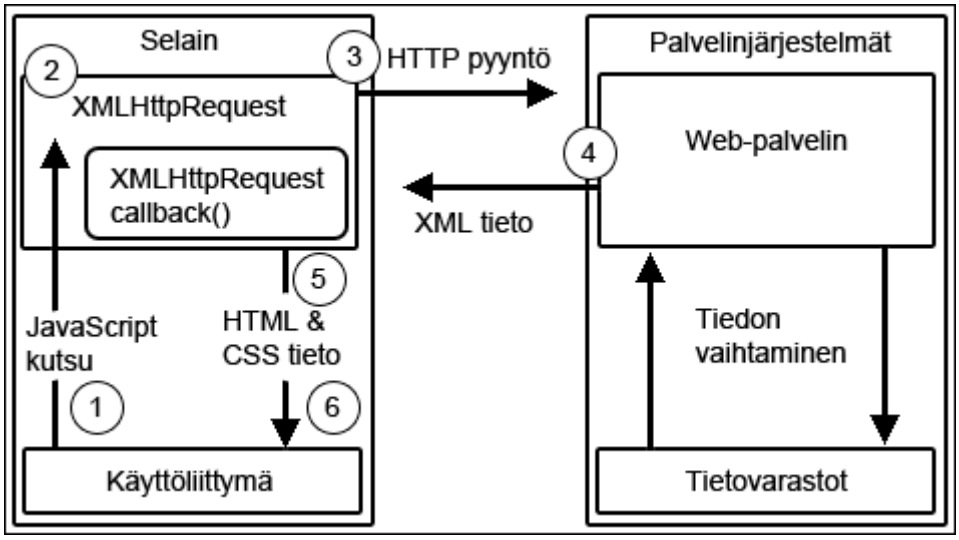
XMLHttpRequest-oliota voidaan pitää AJAX-tekniikan sydämenä ja täten se on myös yksi tärkeimpiä keksintöjä selainkäyttöliittymien kehityksessä. Kuvassa 6 on yksinkertaisin esimerkki XMLHttpRequest-oliosta. Oliolle ei ole standardia. Se luodaan eri tavalla ActiveX-komponenttia käyttävässä Microsoft IE-selaimessa kuin muissa selaimissa, kuten esimerkiksi Mozilla Firefoxissa, jossa XMLHttpRequest on natiivi JavaScript-olio. Luomisen jälkeen olion käyttö on samanlaista eri selaimilla, jolloin sovelluskehittäjällä on käytössään samat olion metodit ja ominaisuudet huolimatta käytettävästä selaimesta. (Peltomäki & Nykänen 2006, 299-300.)

```
1  function luodaanXMLHttpRequest() {
2      if (window.ActiveXObject) {
3          pyynto = new ActiveXObject("Microsoft.XMLHTTP");
4      }
5      else if (window.XMLHttpRequest) {
6          pyynto = new XMLHttpRequest();
7      }
8  }
```

KUVA 6 *XMLHttpRequest-olio*

4.2 AJAX-sovelluksen toiminta

AJAX-sovelluksen toiminta web-selaimessa on esitetty kuvassa 7.
1. Käyttäjän toiminta saa asiakasohjelmassa aikaan tapahtuman, joka aiheuttaa JavaScript kutsun.
2. JavaScript funktio luo ja asettaa XMLHttpRequest-objektin sekä määrittää JavaScript callback-funktion.
3. XMLHttpRequest-objekti tekee web-palvelimelle asynkronisen http-pyynnön.
4. Web-palvelin käsittelee pyynnön ja palauttaa XML-dokumentin, joka sisältää tulokset.
5. XMLHttpRequest-objekti kutsuu callback-funktiota ja saatuaan vastauksen web-palvelimelta se voi käsitellä pyynnön.
6. Selain päivittää HTML DOM-puurakennetta esittäen www-sivulla uuden tiedon.
(Ort E. & Basler M. 2006.)



KUVA 7 *AJAX-sovelluksen toiminta web-selaimessa*

4.3 Ongelmat

AJAX on huomattava parannus www-sivujen käytettävyyteen, mutta johtuen sen poikkeavuudesta verrattuna perinteisiin staattisiin www-sivuihin, sisältyy sen käyttöön myös ongelmia.

Käyttäjät ovat tottuneet siihen, että selaimen Takaisin-painikkeella pääsee edelliselle sivulle. AJAX-sovellusten kanssa tästä tulee ongelma, koska käyttäjä on jatkuvasti samalla sivulla ja vain sivun sisältö muuttuu käyttäjän toiminnan mukaisesti. Jos tällaisella sivulla painetaan selaimen Takaisin-painiketta, päädytään sivulle, jossa oltiin ennen nykyistä AJAX-sovelluksen sisältämää sivua. Käyttäjä olisi halunnut vain sivun alkuun, käytännössä siihen pisteeseen, josta hän aloitti sivun selailun. Tähän ongelmaan on kuitenkin ratkaisu monissa JavaScript-kirjastoissa, joiden avulla Takaisin-painike saadaan toimimaan käyttäjän odottamalla tavalla.

Toinen ongelma liittyy sivun osoitteeseen ja sivun lisäämiseen selaimen kirjanmerkkeihin. Sivun sisällön vaihtuessa AJAX-sovelluksessa sen osoite pysyy edelleen samana. Kaikki toiminta tapahtuu yhdellä sivulla, joten sivun osoitteen perusteella pääsee vain takaisin siihen tilanteeseen, jolloin käyttäjä ensimmäistä kertaa tuli sivulle. Tämä ongelma tulee vastaan myös kirjanmerkkejä käytettäessä. Tämän vuoksi joudutaan kiinnittämään erityistä huomiota siihen, että jokaiseen sivun osaan pystytään linkittämään. Tällöin sivulla tapahtuvat muutokset päivittävät myös selaimen osoitekenttää, jolloin käyttäjä voi poimia osoitteen talteen ja sen avulla viitata juuri tiettyyn kohtaan AJAX-sovellusta ja sovelluksen sisältämää sivua. Myös tähän ongelmaan on useimmissa JavaScript-kirjastoissa valmiita ratkaisuja. (Thau, D. 2007, 274.)

5 VERTAILTAVAT KIRJASTOT

Vertailussa haluttiin tehdä jokaisella kehityskirjastolla samanlaiset komponentit ja toteuttamisesta saatujen havaintojen avulla vertailla kirjastojen sopivuutta Logia Softwaren www-kehitykseen. Aluksi tarkistettiin, että jokaisesta kehityskirjastosta löytyvät nämä kyseiset valmiit komponentit. Komponenttien toteutuksissa oli totta kai eroavaisuuksia. Oleellista oli löytää suurin piirtein samalla tavalla toimivat komponentit, joita voi muokata paremmin tarpeeseen sopiviksi.

5.1 Valintakriteerit

Valittaessa kehityskirjastoja vertailtavaksi valintakriteereinä olivat riittävän laajat ominaisuudet, aktiivinen yhteisö sekä tunnettavuus. Riittävän laajat ominaisuudet tarkoittavat kirjaston antamia mahdollisuuksia erilaisiin www-kehityksen tarpeisiin. Kehityskirjastolla on hyvä olla valmiiksi rakennettuja komponentteja, joita kehittäjä voi ottaa käyttöön pienellä vaivalla eikä hänen tarvitse koodata kaikkea alusta asti itse. Toisaalta kirjaston pitää tarjota mahdollisuus pienien toiminnallisuuksien tekemiseen www-sivulle, kuten tekstikentän sisällön tarkistukseen ja vastaaviin perustoimintoihin. Jotkin kehityskirjastot on luotu vain jotakin tiettyä toiminnallisuutta varten. Koska tarkoituksena on käyttää kehityskirjastoa hyvin monipuolisesti, valinta kohdistui kirjastoihin, jotka tarjoavat mahdollisimman monipuolista kehitystä.

Aktiivinen yhteisö on tärkeä asia kehityskirjaston kehityksen kannalta. Kirjastojen kehitys perustuu suurimmaksi osaksi vapaaehtoisuuteen, jolloin kehittäjät antavat tuotoksensa ilmaiseksi toisille. Ilman riittävän suurta ja toimivaa yhteisöä kehityskirjaston kehitys ei jatku pitkään. Ei riitä, että yhteisö kehittää vain kirjaston koodia ja julkaisee uusia versioita. Yhtä tärkeää on dokumentointi ja yhteisön vapaan keskustelun ylläpito. Kehittäjille pitää tarjota ohjeita kehityskirjaston käyttöön, kuten myös koodiesimerkkejä ja keskustelualueita. Toimiva yhteisö voi olla kehityskirjaston toimivin mainos. Sellainen houkuttelee mukaansa, koska tällöin voidaan vakuuttua siitä, että ongelmia kohdatessaan saa apua toisilta kehittäjiltä.

Kolmas valintakriteeri, tunnettavuus, ei sinänsä kerro kehityskirjaston ominaisuuksista paljoakaan. Siitä voi kuitenkin päätellä kirjaston saavuttaneen kehittäjien hyväksynnän. Jos muut ovat sen hyväksyneet, todennäköisesti kehityskirjasto on hyödyllinen ja käyttökelpoinen. Tunnettavuuden etuna on mahdollisuus löytää erilaisia ohjeita ja neuvoja kehittäjien sivustoilta ja blogeista. Yleisesti ottaen voidaan sanoa, että mitä tunnetumpi kehityskirjasto, sitä enemmän siitä kirjoitetaan. Tämä auttaa kirjaston kehityksen seuraamista ja uusien ominaisuuksien oppimista.

5.2 Kehityskirjastojen yhteiset ominaisuudet

Kehityskirjastoille yhteistä on tavoite nopeuttaa kehittäjien työtä karsimalla pois toistuvia työvaiheita. Lisäksi kirjastot tuovat helpotusta eri selaimista johtuvien virheiden kiertämiseen, joten myös tässä kehittäjä pääsee helpommalla verrattuna perinteiseen JavaScript-koodiin, jossa kehittäjän täytyisi itse huomioida selainten eroavaisuudet.

Yhteistä on myös samanlainen lisensointi. YUI ja Dojo käyttävät BSD-lisenssiä, joka on useimpien avoimen lähdekoodin -projektien lisenssi. Lyhyesti sanottuna lisenssi antaa mahdollisuuden vapaasti käyttää, muokata ja poistaa koodia omien tarpeiden mukaan. Ei ole myöskään merkitystä, käyttääkö koodia yksityisissä vai kaupallisissa projekteissa. Ehtona on vain se, että alkuperäinen tekijänoikeusmerkintä säilytetään koodissa. JQuery käyttää MIT lisenssiä, joka on ominaisuuksiltaan lähes identtinen BSD-lisenssin kanssa. Kehityskirjastojen käyttö on siis täysin vapaata myös kaupallisiin tarkoituksiin. Lähes kaikki kehityskirjastot ovat tällä tavalla avoimia ja kaikkien käytettävissä ilman, että niistä tarvitsee maksaa mitään.

5.3 Vertailuun valitut kehityskirjastot

Vertailuun valittiin kehityskirjastot Dojo, jQuery ja YUI. Ne vastaavat hyvin valintakriteereitä eli ovat riittävän monipuolisia, yhteisöiltään aktiivisia sekä suhteellisen tunnettuja.

5.3.1 Dojo

Dojo sai alkunsa vuonna 2004, kun Alex Russell, David Schontzler ja Dylan Schiemann työskentelivät Informatica –nimisessä yrityksessä samassa DHTML-projektissa. He alkoivat hahmotella uutta JavaScript-kehityskirjastoa ja mukaan tuli myös muita kehittäjiä. Lopulta perustettiin Dojo Foundation -organisaatio. Ensimmäinen versio Dojo -kehityskirjastosta julkaistiin alkuvuodesta 2005. (The Dojo Foundation 2009.)

Dojon taustalla on The Dojo Foundation, joka on voittoa tavoittelematon avoin yhteisö. Dojon lisäksi yhteisöllä on myös muita avoimen lähdekoodin -projekteja ja sitä tukee monet suuret yritykset, kuten IBM, AOL ja Sun. Varsinaisen yhteisön kuitenkin muodostavat sadat kehittäjät. Yhtei-

sön tavoitteena on pitää kehityskirjaston kehitys avoimena kaikille, jolloin kaikki saavat osallistua sen kehittämiseen. (Russell, M. 2008, 25.)

5.3.2 jQuery

Ensimmäinen versio jQuery-kirjastosta julkaistiin vuonna 2006. Sen oli pääosin kirjoittanut John Resig, joka työskentelee Mozilla Software Foundationin palveluksessa. Kyseinen yritys on tunnettu ennen kaikkea Firefox-selaimesta. Julkistuksen jälkeen kirjaston suosio on kasvanut lähes räjähdysmäisesti ja se onkin tällä hetkellä käytetyin kehityskirjasto. (Wellman 2009.)

JQuery on nopeasti saavuttanut suuren suosion www-kehittäjien keskuudessa ja sitä käytetään monissa arvostetuissa avoimen lähdekoodin -projekteissa sekä monilla suurilla www-sivustoilla. Kehityskirjasto haluaa tarjota www-kehittäjille kokonaan uudenlaisen tavan kirjoittaa JavaScript-koodia. (Bibeault & Katz 2008, 2.)

5.3.3 YUI

YUI on lyhenne sanoista Yahoo! User Interface, joka on kokoelma JavaScript- ja CSS-työkaluja, joiden avulla voidaan kehittää nykyaikaisia www-sovelluksia ja –käyttöliittymiä. Kehityskirjaston ensimmäinen versio julkaistiin vuonna 2006. Kirjastolla on siis jo melko pitkä historia, jos ajatellaan, miten nopeasti asiat muuttuvat ja kehittyvät internetissä. (Yahoo! Inc. 2010.)

YUI erottuu kahdesta muusta kirjastosta ehkä juuri kaupallisen taustan vuoksi. Siihen on sijoitettu rahaa ja projektilla vaikuttaisi olevan selvä suunnitelma, jonka mukaan se etenee. Kehittämisestä vastaavat suurimmaksi osaksi projektissa mukana olevat Yahoon työntekijät. Kehityskirjaston kotisivut ovat kattavat ja niiden sisältöön on sijoitettu aikaa. Lisäksi ohjeistusta ja esimerkkejä on huomattavasti enemmän verrattuna toisiin kirjastoihin.

Kehittäjälle tärkeää on tieto siitä, että YUI on todellisessa käytössä Yahoon ja monien muiden www-sivustojen taustalla. Tämä vakuuttaa ainakin testauksen olleen kattavaa, koska suuren sivuston yhteydessä pienetkin virheet tulevat nopeasti esiin. Koska kyse on suuren ja tunnetun internetyhtiön projektista, se tuo tietynlaista varmuutta kehityskirjaston kehityksen jatkumisesta. Yhtiöllä on pitkä kokemus www-teknologioista ja se on ollut internetin alkuajoista mukana kehittämässä erilaisia www-palveluita. Tämä on kirjastolle merkittävä etu, kun yrityksissä valitaan käyttöön otettavaa kehityskirjastoa, jolla on tarkoitus tehdä töitä seuraavat viisi vuotta.

6 VERTAILU

Vertailu perustuu havaintoihin, joita tehtiin kehitettäessä jokaisella kehityskirjastolla samat kolme komponenttia. Yksi komponenteista oli niin sanottu autocompele-tekstikenttä, joka täydentää käyttäjän kirjoittamaa tekstiä sitä mukaa, kun käyttäjä lisää kirjaimia. Sellainen tehtiin ensin Dojolla, sen jälkeen jQuerylla ja lopuksi YUI:a käyttäen. Kaksi muuta komponenttia olivat tietotaulukko ja raahaus-toiminto. Tietotaulukko hakee tiedot tietokannasta, jonka jälkeen niitä voi järjestellä eri tavoin ja muuttaa kerralla näytettävien tietojen määrää. Raahaus-toiminnon avulla voidaan järjestellä esimerkiksi listassa olevia tietoja eri järjestykseen. Käyttäjä painaa hiiren nappia siirrettävän listaelementin kohdalla ja vetää sen oikealle kohdalle. Tämän jälkeen uusi järjestys tallennetaan tietokantaan.

Vertailua tehdessä tuli hyvin esille kehityskirjastojen yleiskäyttöisyys, ohjeistuksen laajuus ja laatu sekä yhteisön toiminta. Samoin tuli esiin, kuinka nopeaa ja helppoa kirjaston syntaksi on omaksua. Tämä on tärkeä piirre yrityskäytössä, koska se kertoo, miten paljon koulutusta tarvitaan ennen kehityskirjaston käyttöönottoa. Eli käytännössä, kuinka nopeasti kirjaston käytöstä saadaan tuottavaa.

Vertailussa kehityskirjastoista käytettiin kahta versiota. Osa komponenteista tehtiin aikaisemmin ja puoli vuotta myöhemmin ne tehtiin uudelleen kirjastojen uusimmilla versioilla. Tämä oli sinänsä ylimääräistä työtä, mutta se toi esiin kehityskirjastojen eri versioiden yhteensopivuuden. Kirjastot päivittyvät melko usein ja työn kirjoittamisen aikana kehityskirjastoista tuli uusia versioita. Näitä uusien versioiden mukanaan tuomia muutoksia ei kuitenkaan huomioitu vaan vertailu perustuu havaintoihin, joita tehtiin aikaisemmilla versioilla.

6.1 Kehitysyhteisö

Kehitysyhteisö vertailukriteerinä sisältää keskustelualueet, ohjeistuksen määrä ja laatu sekä uudet versiot. Keskustelualueet ovat useimmiten paras tapa selvittää ongelmia ja vaikeuksia eri tekniikoiden käytössä. Kehityskirjastojen tapauksessa keskustelualueiden merkitys korostuu laajan käyttäjäkunnan vuoksi. Ohjeistuksen määrä ja laatu on tärkeä kriteeri erityisesti uusien kehittäjien kannalta, joiden pitäisi päästä nopeasti perille perusasioista. Uudet versiot ovat kaksijakoinen asia, koska usein tulevat versiot kertovat aktiivisuudesta kehityskirjaston kehityksessä, mutta toisaalta uusien versioiden käyttöönotto tuo ylimääräistä työtä kehittäjille.

6.1.1 Keskustelualueet

Dojon keskustelualueet on jaoteltu moneen osaan, ehkä jopa liian moneen. Kysymykselleen on vaikea löytää oikeaa paikkaa. Keskusteluissa viestejä on määrällisesti paljon, mutta uusimmat on kirjoitettu kahdeksan viikkoa sitten. Vaikuttaa siltä, ettei yhteisö käytä keskusteluita ongelmanratkaisussa, vaan nojautuu suurimmaksi osaksi ohjeistuksen varaan. Kehityskirjas-

ton esittelyssä kuitenkin toimivaa keskustelua korostetaan, joten tässä asiassa lupauksia ei täysin lunasteta.

JQueryn yhteisö on selvästi aktiivinen keskusteluissa. Viestejä tulee kymmeniä päivittäin ja kaikki viestit tulevat samaan paikkaan. Niitä ei siis jaotella kategorioihin, mikä vaikuttaa ensin oudolta, mutta hakutoiminto hoitaa tehtävänsä niin hyvin, ettei jaottelua jää kaipaamaan.

YUI on jQueryn kanssa samalla tasolla. Keskustelu on aktiivista ja vastauksia kysymyksiin tulee nopeasti. Keskustelut on jaoteltu selkeisiin kategorioihin. Erityisesti jaottelu versionumeron mukaan on käytännöllinen, koska uusimpaan versioon ei varsinkaan yritysmaailmassa voida siirtyä heti, vaan tukea tarvitaan edelliselle versiolle. Keskusteluissa ovat aktiivisia kirjaston kehityksestä päävastuussa olevat Yahoon kehittäjät, joten vastaukset ovat kattavia ja niissä ei esiinny perinteisten keskustelualueiden tyypillistä arvailua. Lisäksi tällä tavalla kehittäjät saavat helposti keskusteltua kirjaston kehitystarpeista ja puutteista suoraan asiasta vastaavien henkilöiden kanssa.

6.1.2 Ohjeiden määrä ja laatu

Dojo on selvästi panostanut ohjeistukseen. Yhteisöllä on erillinen Dojo-Campus -sivusto, johon on koottu erilaisia ohjeita sekä aloittelijoille että pidemmälle ehtineille kehittäjille. Lisäksi on versiokohtaiset API-referenssit. Vaikka ohjeille on oma sivustonsa, se on hankala käyttää ja on lisäksi hyvin keskeneräinen. Monien ohjeiden kohdalla kerrotaankin ohjeen olevan vasta tulossa. Päällisin puolin näyttää, että Dojo on hyvin dokumentoitu, mutta todellisuudessa ohjeistus on pahasti kesken ja tämä vaikeuttaa kehityskirjaston opettelua.

JQueryn ohjeistus on vertailun paras. Sen omilta sivuilta löytyy paljon erilaisia ohjeita aloittelijoille ja pidemmälle ehtineille, mutta vielä enemmän ohjeita löytyy erilaisista blogeista. JQuery on suosituin kehityskirjasto, mikä tulee ilmi ohjeiden määrässä. Blogien ja jQueryn omien keskustelualueiden lisäksi kirjastosta löytyy keskustelua monilta www-kehittäjien sivuilta ja keskustelualueilta. Ongelmiin on helppo löytää vastauksia ja tämä auttaa huomattavasti pääsemään alkuun kehityskirjaston käytössä.

YUI tarjoaa ohjeistusta sivuillaan, mutta sen lisäksi ohjeet tulevat aina mukana, kun massiivisen YUI-paketin lataa koneelleen. Ohjeita ja esimerkkejä on paljon, ja niiden yhteyteen on aina selitetty ominaisuuksien vaikutus jos kehittäjä haluaa muokata valmista koodia. Silti hieman parempaa ohjeistusta jää kaipaamaan, koska pelkkä ominaisuuksien ja muutosmahdollisuuksien selittäminen ei riitä, jos valmista koodia halutaan muokata selvästi erilaiseksi. Ohjeissa oletetaan, että valmis komponentti tarjoaa kaiken tarpeellisen, eikä sitä ole syytä sen enempää muokata.

6.1.3 Uudet versiot

Dojon versioita tuli vuoden aikana seitsemän kappaletta ja ne ovat suurimmaksi osaksi taaksepäin yhteensopivia. On hyvä, että parannuksia ja uudistuksia tulee usein, jolloin kehittäjät eivät joudu odottamaan tärkeitä päivityksiä pitkään. Kuitenkin tällainen versiomäärä tuo ylimääräistä työtä, koska aina pitää selvittää vähintään se, sisältääkö päivitys kriittisiä tietoturvapäivityksiä.

Myös JQueryn versioita tuli vuodessa seitsemän kappaletta. Osa päivityksistä tosin oli vain edellisen version virheenkorjauksia. Kuitenkin tässä pätee sama asia kuin Dojon versioiden määrässä. On toisaalta hyvä, että korjauksia tulee nopeasti ja kehittäjät saavat uusia toiminnallisuuksia käyttöönsä, mutta vastaavasti uusiin versioihin liittyy aina päivityksen selvitystyö ja itse päivittämisen vaiva.

YUI versioita tuli vuoden aikana kolme, joten se on kahteen muuhun kehityskirjastoon verrattuna huomattavasti harvemmin päivittyvä. Versiot olivat suurimmaksi osaksi taaksepäin yhteensopivia. Päivitysten ollessa kerralla suurempia kokonaisuuksia niiden uudistuksien tutkiminen on hieman työläämpää. Uusia toiminnallisuuksia joutuu odottamaan kauemmin, mutta toisaalta harvemmin tulevat päivitykset vähentävät jatkuvaa tarkkailua uusista päivityksistä ja niiden uudistuksista.

6.1.4 Vertailutaulukko

TAULUKKO 1 *Kehitysyhteisö*

	Dojo	jQuery	YUI
Keskustelualueet	1	3	3
Ohjeiden määrä ja laatu	2	3	2
Kehityskirjaston uudet versiot	2	2	3

Kehitysyhteisön vertailussa jQuery ja YUI olivat tasavertaiset, molemmat saivat kahdeksan pistettä, kun Dojo sai viisi pistettä. Dojo korostaa toimivaa ja avointa yhteisöä, mutta tehtyjen havaintojen perusteella se ei täysin pidä paikkaansa.

6.2 Kehityskirjaston käyttö www-sivulla

Kehityskirjaston käyttö www-sivulla sisältää kehityskirjaston liittämisen www-sivulle, syntaksin sekä paketin osioinnin. Kehityskirjaston liittäminen www-sivuille vertailee kehityskirjaston tapaa liittää koodia sivulle sekä koodin määrää ja liittämisen helppoutta. Syntaksi on yksi suurimmista eroavaisuuksista kirjastojen välillä ja se on myös tärkeimpiä vertailukriteereitä, koska sen varassa on suurimmaksi osaksi kehitysnopeus ja oppimiskynnys. Paketin osiointi tarkoittaa kehityskirjastojen tapaa paketoida kirjaston koodit yhteen pakettiin, jota jaetaan kehittäjille. Käytännössä tässä vertaillaan paketin kokoa ja muokattavuutta tarpeiden mukaan.

6.2.1 Kehityskirjaston liittäminen www-sivulle

Dojo on hyvin selvä ottaa käyttöön. Moniin toimintoihin riittää yhden script-elementin lisäys ja koodin määrä pysyy maltillisena. Dojo käyttää Javan tyylistä syntaksia, mikä näkyy myös kehityskirjaston käyttöönotossa. Jos on käyttänyt Javaa, on helppo ottaa käyttöön kirjaston komponentteja. Käyttöönottoa helpottaa myös kirjaston jakautuminen kolmeen pääosaan. Opeteltuaan näiden osien eron, on helppoa tietää, missä osassa haluttu komponentti sijaitsee.

JQuery on lähellä Dojoa ja yhtä helposti käyttöön otettava. Kehityskirjasto vaatii yleensä vain varsinaisen kirjaston koodin eli yhden tiedoston liittäminen riittää. Jos käyttöön halutaan jQueryn lisäosia, ne liitetään sivulle erikseen. Tämä on selvä ja toimiva tapa, jossa kehittäjä tietää tarkkaan, mitä hänen tarvitsee liittää www-sivulle, jotta koodi toimisi. Käyttöönotossa koodin määrä on useimmissa tapauksissa huomattavasti vähäisempi kuin kahdella muulla kehityskirjastolla.

YUI on selvästi monimutkaisin käyttöönotossa. Perustoimintojakin varten tarvitaan useita script-elementtejä koodin alussa ja jokainen käytettävä komponentti vaatii oman script-elementin. Kehittäessä monesti unohtui lisätä jonkin komponentin käyttöönotossa tarvittava script-elementti, eikä koodi sen vuoksi toiminut. Tämä oli selvästi vain YUI:n ongelma, koska muissa kehityskirjastoissa unohtelua ei tapahtunut. Varsinainen koodin määrä on komponenteissa monin paikoin samaa tasoa kuin muissakin kirjastoissa, mutta esimerkiksi raahaus-toiminnallisuus vaati nelinkertaisen määrän koodia verrattuna kahteen muuhun kehityskirjastoon.

6.2.2 Syntaksi

Dojo on syntaksiltaan helposti verrattavissa Javaan, sillä esimerkiksi luokkien esittely ja hierarkia on hyvin samankaltaista. Tämä on kehityskirjastolle suuri etu, koska Java on yksi suosituimpia ohjelmointikieliä ja siihen tottuneet koodaajat pääsevät helpolla sinuiksi Dojon käytössä. Syntaksi ei silti ole täysin samanlaista Javan kanssa ja opetteluun kuluu aikaa.

JQuery erottuu kehityskirjastoista selvästi syntaksin osalta. Kirjastolla on aivan oma tyylinsä kirjoittaa koodia ja yhdistellä funktioita. Eri toimintoja voidaan yhdistää yhteen ohjelmalausekkeeseen pisteillä erottelemalla. Tämä on jQueryn ehdoton vahvuus, mutta myös heikkous. Tekniikan tavoitteena on vähentää koodin määrää ja sitä kautta nopeuttaa koodaamista. Jos syntaksin oppii, koodaaminen nopeutuu selvästi ja tästä syystä kehityskirjasto onkin saanut suosiota. Kuitenkin varjopuolena on oppimisen vaikeus. Tämä on ongelma varsinkin tilanteissa, joissa projektiin tulee uusi ihminen, joka ei ole aikaisemmin käyttänyt kirjastoa. Koska syntaksi on selvästi erilaista kuin tavallinen JavaScript-syntaksi, menee uudella henkilöllä kauemmin päästä mukaan tuottavaan työhön. Syntaksin opetteluun menikin eniten aikaa verrattuna kahteen muuhun kehityskirjastoon.

YUI on lähimpänä perinteistä JavaScript-koodia syntaksin osalta. Kahteen muuhun kehityskirjastoon verrattuna YUI on vähiten tehokas koodin määrän vähentämisessä. Uusia komponentteja käyttöön otettaessa on syytä tarkastella liitettävää koodia, jotta ei turhaan otettaisi ylimääräisiä rivejä mukaan, vaan ainoastaan juuri ne toiminnot, joiden halutaankin tulevan käyttöön. Koodin tarkastelu on tärkeää myös muiden kehityskirjastojen kanssa, mutta kolmesta vertailussa olevasta kirjastosta YUI on komponenttiesimerkkien osalta selvästi eniten optimoitu kaikille sopivaksi. Siitä johtuen koodin määrä on suuri. Syntaksin opettelu on kuitenkin helpompaa kahteen muuhun kirjastoon verrattuna, johtuen lähinnä lähellä perinteistä JavaScriptiä olevasta koodaustavasta.

6.2.3 Paketin osiointi

Dojo on jaettu neljään osaan: Base, Core, Dijit ja Dojox. Osiointi on selvä ja toimii hyvin. Kaiken perustana oleva Base täytyy liittää aina sivulle, jossa halutaan Dojoa käyttää ja sillä onnistuu monet www-sivulla tarvittavat perus JavaScript-toiminnot. Kolme muuta osaa tuovat lisää ominaisuuksia ja toiminnallisuuksia. Dojo onkin hyvin helppo ottaa käyttöön, eikä kehityskirjasto pakota ottamaan ylimääräisiä toiminnallisuuksia käyttöön. Helppo käyttöönotto on tärkeä asia varsinkin alussa, kun kirjaston käyttöä vasta opettelee. Joka tapauksessa kehittäjän tulee ottaa selvää eri osien tarkoituksesta, joten käytännössä aikaa ei välttämättä säästy ollenkaan. Lisäksi neljään osaan jaettu kehityskirjasto ei ole aivan niin hyvin yksilöitävissä kuin YUI, jossa otetaan komponentteja käyttöön lähes yksi kerrallaan.

JQuery kulkee myös paketin osioinnissa omaa tietään, eikä lainaa käytäntöjä muilta kehityskirjastoilta. Koko kirjaston saa käyttöönsä liittämällä sivulle yhden tiedoston, jolla pystyy toteuttamaan kaikki kirjastoon liitetyt ominaisuudet. Tiedoston voisi olettaa olevan hyvin suuri, mutta näin ei ole. JQuery onkin saatu pakattua yllättävän pieneksi tiedostoksi, joka on kuitenkin toiminnoiltaan täysin samalla tasolla muiden kehityskirjastojen kanssa. Yhden tiedoston käytöstä herää väkisin kysymys, pakotetaanko kehittäjille turhaan kaikki toiminnot käyttöön vaikka niistä tarvittaisiin vain yhtä tai kahta. Tämä ei kuitenkaan ole ongelma, koska tiedoston koko on niin huomattavan pieni. Lisäksi kaikkien ominaisuuksien mukana olo ei tee kehityskirjastosta millään tavalla sekavaa tai vaikeaa, kuten voisi äkkiseltään olettaa.

YUI jakaantuu neljään osaan: Core, Utilities, Controls, CSS Tools. Core on koko kirjaston perusta, tarjoten muun muassa työkalut DOM-rakenteen käsittelyyn ja tapahtumien hallintaan. Suurin osa muista kehityskirjaston elementeistä vaatii toimiakseen Coren toiminnallisuuksia. Utilities tarjoaa erilaisia toiminnallisuuksia vuorovaikutteisten käyttöliittymien tekemiseen. Controls sisältää valmiita komponentteja, jotka voi ottaa www-sivulla käyttöön helposti ja nopeasti ilman suurempaa muokkausta. Esimerkki komponenteista on autocomplete-tekstikenttä, joka ehdottaa käyttäjälle mahdollisia hakusanoja sen mukaan, mitä kenttään on kirjoitettu. CSS Tools on tehty määrittelemään www-sivulla käytettävien elementtien

ulkoasu. Huolimatta tästä osioinnista paketti, joka sisältää koko kehityskirjaston, on kooltaan todella suuri verrattuna kahteen muuhun kirjastoon. Mukana tulee koko kehityskirjasto komponentteineen ja sen lisäksi esimerkkejä sekä muuta ylimääräistä sisältöä. Näistä kehittäjän täytyy karsia pois tarpeettomat komponentit ja tiedostot, mikä on todella turhauttavaa toimintaa. Lisäksi YUI:n uuden version käyttöönotossa täytyy sama karsinta tehdä uudelleen.

6.2.4 Vertailutaulukko

TAULUKKO 2 *Kehityskirjaston käyttö www-sivulla*

	Dojo	jQuery	YUI
Kehityskirjaston liittäminen www-sivulle	3	3	1
Syntaksi	3	2	2
Paketin osiointi	2	3	1

Vertailtaessa kehityskirjaston käyttöä www-sivulla, olivat Dojo ja jQuery tasaväkiset molempien saadessa kahdeksan pistettä. YUI sai viisi pistettä ja hävisi selvästi kahdelle muulle kirjastolle. Silti YUI ei ole missään tapauksessa vaikea, mutta verrattuna kahteen muuhun kirjastoon, se ei ole käyttöönotossa yhtä joustava.

6.3 Kehityskirjaston yleiskäyttöisyys

Kehityskirjaston yleiskäyttöisyys sisältää valmiit komponentit, kevyen JavaScript-kehityksen ja toimialueen. Kirjastojen suosiossa on suuressa roolissa valmiit komponentit, koska monilla niistä saa nopeasti aikaan suuria parannuksia www-sivuille lähinnä vain käyttöönoton vaivalla. Kevyt JavaScript-kehitys on käytännössä valmiiden komponenttien vastakohta, jossa verrataan kehityskirjastojen mahdollisuutta tehdä www-sivuille pieniä tarkistuksia ja muita toiminnallisuuksia. Toimialue tarkoittaa kehityskirjaston pääasiallista käyttötarkoitusta, sitä mihin tarkoitukseen kirjasto on suunniteltu ja miten se vastaa kyseiseen tarkoitukseen.

6.3.1 Valmiit komponentit

Dojo on kehityskirjastoista heikoin valmiiden komponenttien vertailussa, koska niille ei ole minkäänlaista listaa, jossa olisi lueteltu eri komponentit ja niiden käyttö. Ongelma on sama kuin ohjeiden määrän ja laadun vertailussa. Vaikuttaisi siltä, että Dojoa kehitetään jatkuvasti ja melko nopeastikin, mutta dokumentointi ei ole pysynyt vauhdissa mukana. On todella hidasta ja vaikeaa etsiä komponentteja Googlen avulla. Muiden kehityskirjastojen sivuilla on melko ajankohtaiset listat erilaisista valmiista toiminnallisuuksista, joita kehittäjä voi vain ottaa käyttöön. Kyllä ohjeita ja esimerkkejä lopulta löytyy, mutta silloin täytyy etukäteen tietää, mitä etsitään, eikä tämä ole kovin tehokas tapa toimia. Lisäksi valmiiden komponenttien määrä on vähäisempi kuin kahdella muulla vertailussa mukana olevalla kehityskirjastolla.

JQuery on suosittu kirjasto ja sille on paljon erilaisia valmiita komponentteja, mutta ongelma on niiden etsintä ja vertailu. Komponenteilla on usein omia sivuja tai niiden käyttöä esitellään yksittäisten käyttäjien blogeissa. Varsinaista koottua listaa ei ole. Yksi lista on jQueryn kotisivulla, mutta se on sekava, eikä hakutoimintokaan ole erityisen toimiva. Komponentteja pitääkin etsiä selailemalla blogeja ja keskustelualueita, mikä on selvä miinus-tekijä. Tietyn tyyppisistä komponenteista on usein kerätty listoja käyttäjien blogeissa ja niiden yhteydessä on pientä vertailua. Komponentit kuitenkin kehittyvät jatkuvasti, joten tieto vanhenee nopeasti. Lisäksi komponenttien käyttöönottoa ei ole ohjeistettu aina riittävän tarkasti, joten kehittäjä joutuu suurimmaksi osaksi kokeilemalla selvittämään komponentin toimintaa ja ominaisuuksia.

YUI on selvästi edellä kahta muuta kirjastoa valmiiden komponenttien osalta, sillä niitä on paljon ja esimerkkejä sekä eri variaatioita on tarjolla useita. Valmiiden elementtien käyttöönotto on helppoa ja esimerkkien avulla löytää helposti omaan tarkoitukseen sopivan tavan käyttää komponenttia. Esimerkeissä on myös hyvät selitykset komponenttien muokkaamista varten. Suuri merkitys on myös sillä, että komponentteja on käytössä Yahoon www-sivulla, joka on yksi käytetyimpiä www-sivuja maailmassa. Tällaisessa todellisessa käytössä komponenteista tulee nopeasti esiin virheet ja hidastelut, joten niitä käyttöönottaessa voi olla vakuuttunut, että niiden testaus on suoritettu riittävän laajasti.

6.3.2 Kevyt JavaScript-kehitys

Dojo on kahden muun kehityskirjaston välissä kevyessä JavaScript-kehityksessä, sillä se ei ole aivan niin sujuva kuin jQuery, mutta ei myöskään yhtä vaikea kuin YUI. Pieniä toiminnallisuuksia pystyy tekemään melko helposti ja koodin määrä pysyy vähäisenä. Www-sivulle ei myöskään tarvitse liittää useita Dojo:n komponentteja YUI:n tapaan, jos haluaa tehdä vain joitain pienimuotoisia tarkistuksia.

JQuery ei pärjää valmiiden komponenttien vertailussa, mutta vastaavasti se on kehityskirjastoista selvästi paras kevyessä JavaScript-kehityksessä, sillä sen koko toimintaperiaate tukee tällaista käyttöä. Www-sivulle tarvitsee liittää vain yksi script-elementti, jolla otetaan jQuery käyttöön ja tämän jälkeen esimerkiksi pienien tarkistuksien tekeminen sivun elementeille on erittäin helppoa. Kirjaston syntaksi on erittäin nopea tähän tarkoitukseen, kunhan sen toimintaperiaatteet sisäistää.

YUI on täydellinen vastakohta verrattuna jQueryyn. YUI on vahva valmiiden komponenttien vertailussa, mutta vastaavasti kevyessä JavaScript-kehityksessä se on kehityskirjastoista huonoin. YUI:n kotisivuilla esimerkit käsittelevät vain kokonaisia komponentteja, mutta pienien JavaScript tarkistusten tekemisestä ei mainita mitään. Kestää aikansa ennen kuin löytää esimerkkejä tällaisien toiminnallisuuksien tekemiseen YUI:n avulla. Kehityskirjasto onkin selvästi painottunut valmiiden komponenttien käyttöön ja jos niistä ei löydy riittävästi ominaisuuksia, aloitteleva kirjaston

käyttäjä on pulassa. Kehittäessä on vaikea ymmärtää, miksi muutaman pienen JavaScript-toiminnon tekemiseen tulee www-sivulle liittää useita YUI:n komponentteja ennen kuin pääsee kirjoittamaan itse tarkistukset. Käytännössä useasti onkin helpompaa kirjoittaa pienet tarkistukset suoraan perinteisellä JavaScriptillä ja jättää YUI pois käytöstä.

6.3.3 Toimialue

Dojo haluaa tarjota kehittäjille kaiken tarpeellisen, mitä www-kehityksessä tarvitaan. Tämä tarkoittaa varsinaisen koodin lisäksi myös työkaluja testaukseen ja virheen etsintään. Näiden avulla kehittäjän työ tehostuu, koska käytössä ei ole erillisiä työkaluja vaan kaikki tulee samassa paketissa. Tämä on toimiva idea, mutta kehityskirjaston syntaksin opettelun lisäksi testaus- ja virheen etsintä -työkalujen opettelu vaatii pidempää perehtymistä, jotta niistä saa hyödyn irti. Perehtymiseen tarvittava dokumentointi on Dojon kohdalla valitettavan heikkoa, kuten aikaisemmin on jo tullut ilmi. Sinänsä kehityskirjasto lunastaa tavoitteensa kaiken kattavasta www-kehityspaketista, mutta käytön ohjeistus on niin heikkoa, ettei kirjastosta saa täyttä tehoa irti kovin nopeasti.

JQueryn kotisivulla oleva slogan kertoo kehityskirjaston tavoitteen ”Write less”, kirjoita vähemmän. Kirjaston tavoite ei ole tarjota kaiken kattavaa valmista ratkaisua www-kehittämiseen vaan nopeuttaa kehittämistä. JQuery antaa mahdollisuuden käyttää kehityskirjaston toiminnallisuuksia kaikkeen mahdolliseen www-sivulla tai vastaavasti tehdä vain pieniä JavaScript-toiminnallisuuksia muutamiin elementteihin sivulla. Kirjasto ei tarjoa samanlaista kokonaisvaltaista otetta kuin YUI, mutta on vastaavasti täysin muokattavissa ja laajennettavissa tarpeiden mukaan. Komponenttien käyttöönotossa kehittäjän täytyy tehdä töitä hieman enemmän itse verrattuna YUI:n tapaan, mutta näin saavutetaan myös täydellinen muokattavuus. Kahteen muuhun kehityskirjastoon verrattuna jQuery on selvästi yleiskäyttöisin, joskin kehittäjän täytyy paneutua alussa hieman enemmän syntaksiin ja toimintaperiaatteisiin. Kun ne sisäistetään, kirjaston käytöllä ei ole rajoja vaan se sopii kaikkeen, mitä www-kehityksessä tarvitaan.

YUI:n tavoite on tarjota ns. päästä-päähän -ratkaisu www-kehittämiseen, jolla voi tehdä kaiken mahdollisen www-sivulle. Tavoite täyttyy ja YUI tarjoaa kokonaisen paketin kaikkeen, mitä www-sovellusten käyttöliittymissä ja käyttöliittymien kehityksessä tarvitaan. Kehityskirjaston mukana tulee varsinaisien koodien lisäksi testausta ja virheen etsintää helpottavia työkaluja. Kirjasto huolehtii myös ulkoasusta CSS Tools -toiminnallisuuksilla. YUI on hyvin dokumentoitu, eri komponenttien esimerkit ovat kattavia ja käyttöönotto melko helppoa. Tavoitteena on selvästi antaa kehittäjille valmiita komponentteja, jotka tarvitsee vain ottaa käyttöön www-sivulla ja tällä tavoin nopeuttaa käyttöliittymien kehittämistä. Heikkoutena vastaavasti on ongelmat kevyessä JavaScript-kehityksessä, johon käyttöön kehityskirjasto ei sovellu. YUI on todella hyvä, jos komponenteista saa suoraan hyödyn irti ilman suurempaa muokkausta ja pienet JavaScript-tarkistukset ja vastaavat toiminnot toteutetaan perinteisellä JavaScriptillä.

6.3.4 Vertailutaulukko

TAULUKKO 3 *Kehityskirjaston yleiskäyttöisyys*

	Dojo	jQuery	YUI
Valmiit komponentit	1	2	3
Kevyt JavaScript-kehitys	2	3	1
Toimialue	1	3	2

Kehityskirjaston yleiskäyttöisyydessä jQuery oli selvästi paras saaden kahdeksan pistettä. YUI hävisi paljolti joustamattomuutensa vuoksi ja sai kuusi pistettä. Dojon selvä heikkous on keskeneräisyys erityisesti dokumentoinnissa, jolloin yleiskäyttöisyydestä on vaikea saada hyötyä irti ja kirjasto saikin vain neljä pistettä.

7 VERTAILUTULOKSET

Vertailussa eniten pisteitä keräsi jQuery, 24 pistettä. Seuraavaksi tuli YUI, 18 pistettä. Heikoiten menestyi Dojo, 17 pistettä. Piste-ero jQuery:n ja YUI:n välillä oli selvä, mutta Dojo ja YUI vastaavasti olivat melko tasaväkisiä.

Vertailun tuloksena paras kehityskirjasto oli jQuery, joka on selvästi suosituin myös internetin kehittäjäsivustoilla ja blogeissa. Suosiota ei tarvitse ihmetellä, sillä kirjasto tarjoaa omanlaisensa lähestymistavan www-kehitykseen. Toiminta-ajatus ja syntaksi erottuvat selvästi muista kehityskirjastoista.

Kehitettäessä jokaisella kehityskirjastolla samat komponentit, jQueryn käyttö oli ehdottomasti mukavinta ja alusta asti sujuvaa. Vaikka tekijä oli jokaisen kehityskirjaston suhteen täysin aloittelija, tällainen havainto yhdestä kirjastosta muita mukavampana kertoo jotain sen onnistuneesta suunnittelusta.

Vertailun tulokseen vaikutti tietynlaisen kehitystarpeen mukana tuoma painotus. Se, että jQuery vaikuttaa kehityskirjastoista parhaalta Logia Softwaren tarpeisiin, ei tietenkään tarkoita muiden kirjastojen olevan yhtään huonompia. JQuery vain vastaa asetettuihin kriteereihin paremmin ja tällä tavoin vakuuttaa siitä, että sillä tehtävä kehitys on tehokkainta tässä kyseisessä tapauksessa.

7.1 Kehityskirjaston käyttäminen osana www-kehitystä

Jotkin kehityskirjastot ovat hyvin erikoistuneita, mutta on myös hyvin paljon saman kaltaisia kirjastoja, joiden ominaisuudet ovat hyvin samanlaisia ja ne tarjoavat vastinetta saman kaltaisiin kehitystarpeisiin. Onkin selvää, että kehityskirjastojen kehityksessä lainataan toisista kirjastoista asioita, jotka sitten muunnetaan omaan kirjastoon sopivaksi ja vielä paremmiksi. Tästä johtuen voidaan todeta, ettei valinnan kanssa ole varsinaista epäon-

nistumisen mahdollisuutta, jos valittavana on kehityskirjastoja, jotka tarjoavat mahdollisuuden kokonaisvaltaiseen www-kehitykseen. Käytännössä kehityskirjaston valinta onkin hyvin paljon kiinni kehittäjien mieltymyksistä ja aikaisemmasta kokemuksesta.

Ehkä tärkein kysymys mietittäessä jonkin kehityskirjaston käyttöönottoa on: Onko siitä todellista hyötyä? Kirjastojen ongelmana on siitä aiheutuva välitaso perinteisen JavaScriptin ja HTML-koodin väliin. Tämä lisää koulutuksen tarvetta ja uusien toteuttajien palkkaaminen projekteihin on hankalampaa. Kirjaston käyttöönotossa tuleekin huomioida projektin laajuus ja kesto. Nopeasti tehtävissä ja pienissä projekteissa, joiden elinkaari tulee olemaan lyhyt, voi kehityskirjaston käyttö olla hyvä ratkaisu. Tällöin kehittäjät ovat luultavasti alusta asti samat henkilöt, jotka jo osaavat käyttää kirjastoa.

Vastaavasti pidempi kestoisissa projekteissa täytyy kehityskirjaston käyttöä harkita tarkemmin. Kehittäjät saattavat vaihtua projektin aikana ja projektin laajentuessa heitä ehkä tarvitaan lisää. Kirjastoa valittaessa täytyy selvittää, löytyykö sille osaajia Suomesta nyt ja tulevaisuudessa. Ja jos ei löydy, miten paljon koulutusta vaaditaan, jotta uudet henkilöt saadaan projektiin mukaan. Samoin on syytä harkita käyttöönoton laajuutta, sillä kehityskirjastoa ei ole pakko käyttää joka paikassa vaikka sellainen onkin projektin käyttöön valittu. Osassa projektin www-sivuista voidaan käyttää perinteistä JavaScriptiä ja toteuttaa kehityskirjaston avulla toiminnallisuuksia vain tiettyihin sivuihin. Tällaisesta osittaisesta käytöstä seuraa kuitenkin omat ongelmansa projektin hallinnan suhteen.

Kehityskirjaston käyttöönotto on kuin mikä tahansa muu tekninen osa-alue projektissa, joten sen käyttöä on syytä suunnitella etukäteen. Vaikka kirjaston käytöstä ei varsinaisesti haittaa olisikaan, ei sellaista ole järkevää ottaa käyttöön vain varmuuden vuoksi. Suurin hyöty saadaan, kun kehityskirjastoa valittaessa katsotaan riittävän kauaksi tulevaisuuteen ja käytetään sitä vain kohteissa, joissa siitä on selvästi hyötyä.

7.2 Kehityskirjaston käytöstä saatava hyöty Logia Softwaren www-kehityksessä

Vertailuun valituista kehityskirjastoista Dojo ja YUI olivat monella tapaa samanlaisia ja niiden opiskelu oli suurin piirtein yhtä haastavaa. Kehityskirjaston alkuperäinen tarkoitus, www-kehityksen helpottaminen ja nopeuttaminen, toteutui näissä kahdessa paljolti samalla tavalla. Kyseiset kirjastot eivät kuitenkaan ole aivan yhtä tunnettuja kuin jQuery, vaikka molemmilla onkin suuria yrityksiä tukijoina ja kirjastot ovat käytössä suurillakin sivustoilla.

JQuery on monella tavalla erilainen ja sitä käytettäessä asioita tehdään hyvin eri tavalla verrattuna kahteen muuhun kehityskirjastoon. JQuery on selvästi suosituin kehityskirjastoja tällä hetkellä, ainakin erilaisten blogikirjoitusten, ohjesivustojen ja vastaavien www-kehittäjien sivustojen perusteella. Kehityskirjaston lupaus uudenlaisesta tavasta kirjoittaa JavaSc-

ript-koodia tuli hyvin esiin kehitettäessä komponentteja ja ehkä juuri se on syynä kirjaston saamaan suosioon.

Kehityskirjaston käyttöönotossa tärkeintä on saada siitä todellista hyötyä kehitykselle. Havaintojen perusteella jQueryn käyttöönotosta olisi Logia Softwaren www-kehityksessä selvästi hyötyä. Kyseinen kirjasto on samankaltainen kuin YUI, jolla on valmiita komponentteja ja niitä voi ottaa helposti käyttöön. Mutta jQueryn avulla onnistuu myös pienimuotoinen JavaScript-kehitys, kuten monilla pienemmillä kehityskirjastoilla. JQuery onkin löytänyt sopivan keskitien, jolloin se tarjoaa välineet erityyppisiin kehitystarpeisiin.

Opinnäytetyön tarkoituksena oli selvittää, mikä kirjasto olisi paras, kun tavoitteena on kehittää www-sovelluksen käyttöliittymää. Tähän tarkoitukseen jQuery tarjoaa loistavat mahdollisuudet. Nykyaikaisen www-käyttöliittymän luominen on sillä helppoa ja lisäksi käyttöliittymä voidaan rakentaa juuri sellaiseksi kuin halutaan, ilman valmiiden komponenttien asettamia rajoituksia tai vaikeita muutoksia.

Nykyaikaisen käyttöliittymän lisäksi jQuery tarjoaa hyvät mahdollisuudet muihinkin JavaScript-toiminnallisuuksiin, joita tähän asti on tehty perinteisellä JavaScriptillä. Vanhoja skriptejä tuskin kannattaa uudestaan rakentaa jQueryn avulla, mutta uudet komponentit voisi tehdä kehityskirjaston avulla. Olisi mahdollista luoda yrityksen oma jQuery komponenttikirjasto, jossa olisi toiminnallisuuksia eri tarpeisiin ja joita voisi ottaa käyttöön vain liittämällä koodi www-sivulle. Kaikki koodit olisivat yhdessä paikassa ja yksi tai kaksi henkilöä pitäisi niistä listaa, jossa voisi olla myös selitykset komponenttien toiminnasta ja ohjeet käyttöönottoon. Koodit eivät olisi enää siellä täällä hajallaan vaan niillä olisi oma versionhallinta ja vastaavat henkilöt, jotka tietäisivät versioiden eroista.

Ainoana haittapuolena on jQueryn käytön tuoma lisätaso HTML-koodin ja JavaScriptin välille. Tämä tuottaa hieman enemmän vaivaa uusien henkilöiden koulutuksessa. Tämä ei kuitenkaan ole kovin suuri haitta. Jos henkilö osaa valmiiksi JavaScriptiä, hänen on melko helppo oppia jQueryn syntaksi ja toteutustapa, jolloin tuottavaan työhön päästään nopeasti. Vastaavasti jos henkilö ei osaa JavaScriptiä tai osaa sitä välttävästi, jQuery voi olla jopa helpompaa oppia, koska henkilö ei sovella vanhoja tapoja JavaScript-kehityksestä kehityskirjaston uudenlaiseen ohjelmointitapaan.

Huolimatta siitä, että JavaScript on tällä hetkellä vain pieni lisä Logia Softwaren varastonhallintajärjestelmän kehityksessä, voisi jQuery selkeyttää sitä huomattavasti. Lisäksi parannukset käyttöliittymään saattavat lisätä JavaScriptin osuutta käyttöliittymissä, kun käyttäjät haluavat vastaavia parannuksia kaikkiin järjestelmän toimintoihin. Tällöin olisi hyvä jos käytössä on kehityskirjasto, joka yhtenäistää www-kehityksen ja nopeuttaa sitä.

7.3 Kehityskirjaston käyttöönotto Logia Softwaren www-kehityksessä

Kun kehityskirjasto otetaan käyttöön Logia Softwaren www-kehityksessä, tarvitaan pienimuotoinen peruskoulutus muutamalle henkilölle, jotka pääasiassa vastaavat JavaScript-kehityksestä. Kuitenkin suurin osa opettelusta tapahtuu kehittäjien itse tehdessä töitä jQueryllä. Kuten muissakin ohjelmointikielissä, paras tapa oppia kieltä on käyttää sitä ja ratkaista ongelmia sitä mukaa, kun niitä ilmenee.

Kaikkien www-kehityksessä mukana olevien ei ole tarpeellista opetella jQueryn syntaksia, koska sillä tehdään lähinnä parannuksia käyttöliittymään ja niistä vastaa vain tietyt henkilöt. Näin ollen kyseiset käyttöliittymäparannukset voidaan ohjata tehtäväksi henkilöille, jotka ovat oppineet kehityskirjaston syntaksin. Lisäksi monet jQueryllä tehtävät toiminnot yhdelle www-sivun elementille on helppo ottaa käyttöön myös toisen elementin kanssa kopioimalla ja vaihtamalla vain elementin nimi. On siis mahdollista, että kehityskirjaston käytöstä tietämättömätkin pystyvät liittämään jQueryn toimintoja esimerkiksi www-sivulle tulevaan uuteen tekstikenttään pienellä vaivalla.

Olisi silti hyvä, että yksi tai kaksi henkilöä kantaisi kehityskirjastosta suurempaa vastuuta, koska kirjasto kehittyy nopeasti ja tätä kehitystä on syytä seurata. Tästä olisi etua uusien versioiden käyttöönotosta ja muutoksista tiedottamisessa, jolloin satunnaisesti kirjastoa käyttävät eivät turhaan joutuisi itse selvittämään asioita.

8 YHTEENVETO

Vertailuja tehdessä nousi jatkuvasti esiin kysymys, olinko varmasti ottanut asioista riittävän hyvin selvää, ennen kuin tein päätelmiä. Oliko oma tapani komponentin käyttöön varmasti se oikea, jonka perusteella voisin kehityskirjastoa arvostella. Miten aikaisempi ohjelmointikokemukseni vaikutti kehityskirjastojen syntaksin omaksumiseen? Mitä jos negatiiviset havaintoni johtuivat vaan osaamattomuudestani?

Työn dokumentoinnin aikana syntyi kuitenkin ajatus, että tämä oli minun näkemykseni opiskelun vaikeudesta ja se antaa ainakin jossain määrin vaikutelman kehityskirjastojen käyttöönotossa tarvittavasta koulutuksesta ja mahdollisista vaikeuksista. Onhan täysin eri asia ottaa käyttöön yksi kehityskirjasto ja käyttää sitä päivittäin kuin tehdä samanaikaisesti samoja komponentteja kolmella eri kehityskirjastolla. Kehittämiseen ei voi syventyä, mutta siitä saa silti käsityksen, jonka perusteella pystyy valitsemaan sopivan kirjaston juuri Logia Softwaren tarpeisiin.

Tehdessäni huomasin monesti, miten helppoa jokin asia on tehdä kehityskirjaston avulla. Vaikka kirjaston toimintaperiaate ja syntaksi täytyy opetella, uskon sen olevan ehdottomasti sen arvoista, sillä opettelun jälkeen moni asia muuttuu huomattavasti yksinkertaisemmaksi. Työn ollessa kesken huomasin usein perinteistä JavaScript-koodia kirjoittaessani toivovani, että käytössäni olisi jokin kehityskirjasto. Olin sisäistänyt niiden tarjoamat

edut nopeasti ja perinteisen JavaScriptin kirjoittaminen tuntui jo hieman typerältä.

Uskon, että Logia Softwaren kehittäjät kokevat samanlaisen havainnon JavaScript-kehityksen helpottumisesta jos heillä vain on aikaa paneutua jQueryn uudenlaiseen tapaan kirjoittaa koodia. JQuery on suosittu kehityskirjasto, joka on jo muuttanut monien kehittäjien työtä selvästi helpommaksi ja nopeammaksi. Kirjaston suosio tuskin tulee hiipumaan tulevaisuudessa. Näin ollen kyseinen kirjasto on turvallinen valinta ja se täyttää ne kriteerit, joita kehityskirjastolle asetettiin työtä aloitettaessa.

LÄHTEET

JULKAISUT

Bibeault, B. & Katz, Y. 2008. jQuery in Action. USA. Manning Publications Co.

Crane, D & Pascarello, E. 2006. Ajax In Action. USA. Manning Publications Co.

Heilmann, C. 2006. Beginning JavaScript With DOM Scripting And Ajax. USA. Apress.

Ort E. & Basler M. 2006. Ajax Design Strategies [verkkodokumentti]. [viitattu 25.11.2009]. http://java.sun.com/developer/technicalArticles/J2EE/AJAX/DesignStrategies/

Peltomäki, J. 2001. JavaScript. Jyväskylä. Docendo.

Peltomäki, J. & Nykänen, O. 2006. Web-selainohjelmointi. Jyväskylä. Docendo.

Pohjois-Karjalan Aikuisopisto Lieksan yksikkö. 2009. JavaScript-ohjelmointi [verkkodokumentti]. [viitattu 20.1.2010]. http://aikoledu.pkky.fi/virtuaalikoulu/materiaali/javascript.pdf

Russell, M. 2008. Dojo: The Definitive Guide. USA. O'Reilly Media, Inc.

Thau, D. 2007. The Book Of JavaScript, 2nd Edition. USA. No Starch Press, Inc.

The Dojo Foundation. 2009. A Brief History of Dojo [verkkodokumentti]. [viitattu 15.1.2010]. http://docs.dojocampus.org/quickstart/introduction/history

Wellman D. 2009. jQuery Overview [verkkodokumentti]. [viitattu 14.1.2010]. http://www.devarticles.com/c/a/JavaScript/jQuery-Overview/

Yahoo! Inc. 2010. YUI FAQ [verkkodokumentti]. [viitattu 14.1.2010]. http://developer.yahoo.com/yui/articles/faq/