LAPPEENRANNAN TEKNILLINEN YLIOPISTO
TEKNILLISTALOUDELLINEN TIEDEKUNTA
TIETOTEKNIIKAN KOULUTUSOHJELMA

Kandidaatintyö

Ajax: WWW-sovellusten uudet mahdollisuudet

Kandidaatintyön aihe on hyväksytty 3.10.2008.

Tarkastaja: prof. Kari Smolander

Lappeenrannassa 28. marraskuuta 2008

Ville Kangas

TIIVISTELMÄ

Lappeenrannan teknillinen yliopisto
Teknillistaloudellinen tiedekunta
Tietotekniikan koulutusohjelma

Ville Kangas

Ajax: Web-sovellusten uudet mahdollisuudet

Kandidaatintyö

2008

32 sivua, 5 kuvaa

Tarkastaja: Professori Kari Smolander

Hakusanat: Ajax, WWW, Web, JavaScript, web-sovellus, web-palvelu, web 2.0, tietoturva

Tässä työssä selvitettiin Ajax-tekniikan tilannetta web-sovellusten kehityksessä. Sitä varten kehitettiin demosovellus, jonka avulla tekniikoiden käyttökelpoisuutta voitiin arvioida. Samalla työssä on esitelty eri tekniikoita, jotka liittyvät kiinteästi Ajax-sovellusten toteuttamiseen. Demosovellus tehtiin vapaalla LAMP (Linux, Apache, MySQL and PHP) -alustalla.

Työssä on arvioitu Ajax-tekniikan käyttökelpoisuutta ja ongelmia nykyisen webin, web-kehittäjien, käytössä olevien selainten ja käyttäjien kannalta. Lopussa on myös pohdittu hieman webin tulevaisuutta ja Ajaxin osaa siinä.

ABSTRACT

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

Ville Kangas

Ajax: New Potential of Web Applications

Bachelor's Thesis

2008

32 pages, 5 figures

Examiner: Professor Kari Smolander

Keywords: Ajax, WWW, Web, JavaScript, web application, web service, web 2.0, security

In this thesis the state of Ajax technology in development of web applications was studied. To achieve the goal, a demonstration application was developed to be capable of evaluating the techniques used. Other techniques that constantly relate to implementing Ajax applications are also explained. The demonstration application was implemented on the free LAMP (Linux, Apache, MySQL and PHP) platform.

In the thesis feasibility of Ajax technology and its problems are evaluated from the viewpoint of the current web, the current browsers, the web developers and the end users. In the end, the future of the web and the part of Ajax in it is also slightly discussed.

SISÄLLYSLUETTELO

LYHENTEET

Ajax	Asynchronous JavaScript and XML
CSS	Cascading Style Sheets
DOM	Document Object Model
DTD	Document Type Definition
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
OWL	The Web Ontology Language
RDF	Resource Description Framework
RFC	Request For Comments
SGML	Standard Generalized Markup Language
SVG	Scalable Vector Graphics
SQL	Structured Query Language
URI	Uniform Resource Identifier
W3C	World Wide Web Consortium
WWW	World Wide Web
XHTML	Extensible HyperText Markup Language
XML	Extensible Markup Language
XSS	Cross-Site Scripting

1 JOHDANTO

1.1 Tausta

Webin (World Wide Web) alkuaikoina sisältö koostui pääasiassa tavallisista staattisista HTML-sivuista, joita linkitettiin toisiinsa. Sivut sisälsivät lähinnä tekstiä, taulukoita ja kuvia. Selainmarkkinoita hallitsi Netscape Navigator (alunperin Mosaic). [2]

Tämän päivän web on hyvin paljon muutakin. Verkossa jaetaan liikkuvaa kuvaa ja ääntä, käytetään työpöytäsovellusten kaltaisia ohjelmistoja selaimella ja pelataan java-pelejä. Näiden eri asioiden toteuttamiseen käytettiin vielä 2000- luvun alkupuolella pääosin samaa tekniikkaa joka on ollut olemassa jo webin alkuajoista lähtien. [19]

Suuri muutos tapahtui vuoden 2005 aikana Jesse Jamesin julkaistessa artikkelin, jossa hän määritteli uuden käsitteen: Ajax, Asynchronous Javascript and XML. [5] Perinteisessä web-sovelluksessa koko sivu ladataan jokaisen käyttäjän toimen jälkeen kokonaan uudelleen. Nyt, Ajaxin myötä, voidaan vain muuttuva osa sivusta päivittää tekemällä HTTP-pyyntö selaimessa ajettavassa Javascript-koodissa. Tällöin sovellus tuntuu usein reagoivan käyttäjänsä toimiin nopeammin kuin perinteinen web-sovellus, varsinkin jos sivun rakenne on monimutkainen.

Uuden tekniikan avulla web-sovellusten käyttöön saadaan lisää samankaltaisia piirteitä, joita työpöytäsovelluksissa käytetään; sivun osia voidaan muuttaa reaaliajassa käyttäjän toimien seurauksena.

1.2 Tavoitteet ja rajaukset

Työni tavoitteena on selvittää, kuinka Ajax-sovellus käytännössä toteutetaan. Työssä toteutan web-palvelun, jolla voidaan hallita useiden ihmisten tehtäviä, seurata niiden toteutumista ja selata vanhoja tehtäviä. Palvelu tulee käyttämään Ajax-tekniikkaa.

Itse web-palvelun tarkoituksena on lähinnä esitellä Ajax-tekniikan mahdollisuuksia, ja siten sen toiminnallisuus jäänee rajalliseksi. Ominaisuudet, joita sovellus tarjoaa, toteutan kuitenkin täysin toimiviksi.

Tämän lisäksi esittelen työssä lyhyesti kaikki tekniikat, jotka liittyvät suoraan Ajaxiin tai työssä käytettäviin muihin tekniikoihin.

1.3 Työn rakenne

Tämän johdantokappaleen jälkeen, luvussa 2, käsitellään Ajaxia ja siihen liittyviä tekniikoita sekä historiaa. Tarkoituksena on selventää työhön liittyviä peruskäsitteitä ja niiden yhteyttä Ajaxiin ja uudentyyppisiin web-sovelluksiin.

Demosovelluksen toteutus ja valitut käytännöt on kuvattu luvussa 3. Siinä selvitetään, kuinka demosovelluksen toteutus tapahtui.

Työn tuloksia pohditaan tarkemmin luvussa 4. Käsiteltäviä asioita ovat toteutetun ohjelmiston käyttökelpoisuus, Ajaxin käyttökelpoisuus nykyisessä webissä ja siihen liittyvät haasteet.

2 AJAX JA WEB-SOVELLUKSET

2.1 Johdatus Ajaxiin

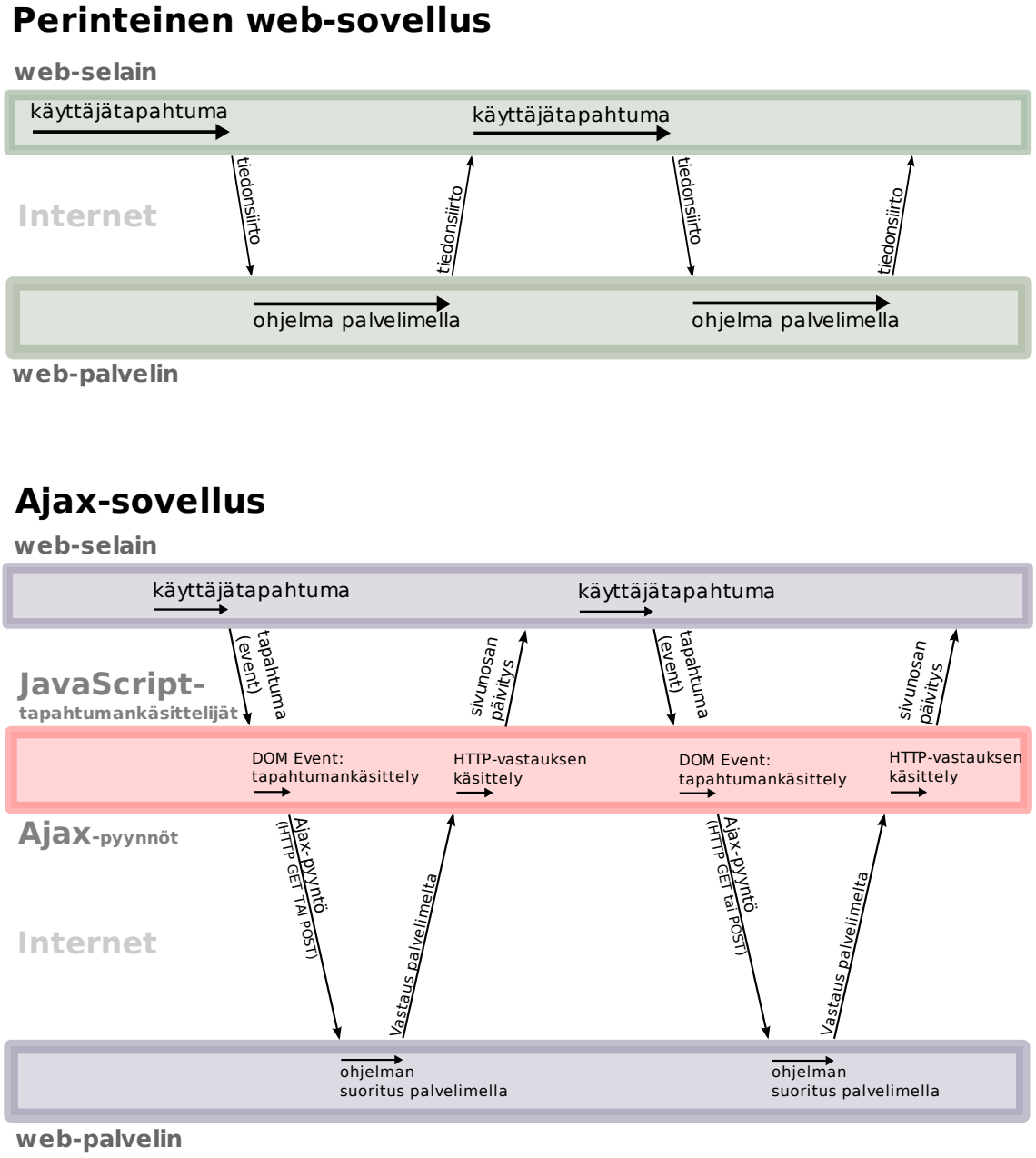
Ajaxin laajaan käyttöönottoa edeltävät tapahtumat alkoivat vuonna 1996, jolloin Microsoft julkaisi Internet Explorerin version 5. Se sisälsi XMLHTTP-toteutuksen, jolla esimerkiksi VBScript-koodin avulla voitiin tehdä HTTP-pyyntöjä. Nykyinen, XMLHttpRequest -luokka, josta W3C on aloittanut standardointiprosessin[1], sai alkunsa vuonna 2002, kun Mozillan versio 1.0 julkaistiin. Myöhemmin tuki sille on lisätty kaikkiin tärkeimpiin selaimiin, myös Internet Exploreriin version 7 myötä. [8]

Ensimmäisiä varsinaisia Ajax-sovelluksia oli Google maps, jossa kartalla voidaan siirtyä eri suuntiin, ilman että koko sivu tarvitsee ladata välillä uudestaan, kuten aiemmissa karttasovelluksissa jouduttiin tekemään. Ensimmäinen versio palvelusta julkaistiin 8. helmikuuta 2005 [7]. Vain vajaat pari viikkoa myöhemmin Jesse James Garrett julkaisi tunnetun artikkelinsa “Ajax: A New Approach to Web Applications”, jossa hän esitteli termin Ajax. Garrettin määritelmän mukaan Ajax on uusi lähestymistapa web-sovelluskehitykseen. Hänen mukaansa nimen taustalla on lähinnä tarve helpommalle nimelle kuin “JavaScript+CSS+DOM+XMLHttpRequest”. [5]

Kuvassa 1 näemme ensin perinteisen web-sovelluksen ja jäljempänä Ajax-sovelluksen tiedonsiirtomallin. Perinteisessä mallissa jokainen käyttäjän toimiin reagoiminen vaati koko sivun lataamisen uudelleen muutoksineen. Ajax-mallissa käyttäjän toimiin reagoi JavaScript-tapahtumankäsittelijä, joka saa tiedon, että esimerkiksi jotakin painiketta sivulla on painettu. Tapahtumankäsittelijä tekee palvelimelle HTTP-pyynnön ja pyytää sopivaa dataa ja kertoo, mikä funktio käsittelee saapuvan vastauksen. Kun vastaus saapuu, ennalta määritelty funktio käsittelee sen, eli usein onnistuneen siirron seurauksena, liittää sivun dokumenttipuuhun uuden, päivitetyn HTML-pätkän ja selain päivittää näkymän. Käyttäjä huomaa vain, että jokin osa sivusta muuttuu pyynnön seurauksena.

Myöhemmin, vuoteen 2008 mennessä, Ajax on otettu hyvin laajalti käyttöön web-sovelluksissa. Lähes kaikissa suuremmissa verkkopalveluissa on ainakin joitakin täydentäviä osia toteutettu Ajaxin avulla. Seuraavissa kappaleissa tutkimme Ajaxia eri näkökulmista.

[1] XMLHttpRequest-luokasta on julkaistu viimeisin “Working Draft” 15. huhtikuuta 2008 (Tilanne 21.9.2008).



Kuva 1: Perinteisen web-sovelluksen ja Ajax-sovelluksen tiedonsiirtomallit.

2.1.1 HTTP ja Ajax

HTTP-protokollan olemusta kuvaa hyvin lyhyt lainaus HTTP 1.1:n kuvaavasta RFC numero 2616:sta: “It is a generic, stateless, protocol which can be used for many tasks beyond its use for hypertext, such as name servers and distributed object management systems, through extension of its request methods, error codes and headers.” [9] Sen välityksellä voidaan välittää monenlaista informaatiota. HTTP-protokollassa ei ole tukea eri tiloille, vaan sen näkökulmasta web-selaimet vain lähettävät erilaisia pyyntöjä, kuten GET, POST ja PUT. Kuten kuvasta 1 näkyi, HTTP-pyynnöt palvelimelle ovat samanlaisia

kuin ennenkin.

Monet nykyiset web-sovellukset kuitenkin tarvitsevat tilatietoa toimiakseen, niin myös tässä työssä kehitettävä web-sovellus. HTTP-protokollan yksinkertaisuuden vuoksi eri palvelimella käytettävät kehitysympäristöt, kuten tässäkin työssä käytetty PHP, tarjoavat ratkaisuja, joilla voidaan avata istuntoja (session) käyttäjille. Tällöin palvelimella ajettavalla ohjelmalla on mahdollisuus tallentaa istuntoon liittyviä tietoja helposti, ja siten, että ne ovat useista eri HTTP-pyynnöistä huolimatta aina uuden pyynnön saapuessa sovelluksen käytettävissä.

PHP:n istunnot perustuvat yleensä evästeiden (cookie) käyttöön, mutta myös HTTP-pyynnön mukana URI:n kysely (query)-osassa määriteltävää istunnon tunnistetta voidaan käyttää [13, 9].

Myös Ajax-sovellukset tulevat hyvin toimeen nykyisen HTTP-protokollan kanssa. Niissä erona perinteisiin web-sovelluksiin on vain se, että JavaScript-koodilla voidaan tehdä HTTP-pyyntöjä ja käsitellä palvelimelta tulevaa tietoa, kun aikaisemmin käyttäjän toimien seurauksena web-selain on ladannut aina koko sivun uudelleen välittäen samalla sovelluksen tarvitsemat parametrit.

2.1.2 Yksinkertainen Ajax-sovellus

Ajax-sovellus tarvitsee yksinkertaisimmillaan yhteensäkin vain muutamia kymmeniä rivejä HTML ja JavaScript-koodia. Listauksessa 1 näemme hyvin yksinkertaisen Ajaxia käyttävän “sovelluksen”.

Sovellus koostuu yhdestä linkistä, jota käyttäjä voi painaa. Käyttäjän painaessa linkkiä kutsutaan tapahtumankäsittelijää, $greet()$. Tapahtumankäsittelijä tekee palvelimelle Ajax-pyynnön ja sen käsittelijä, $messageReceived()$, hakee tunnisteen ’messageContainer’ avulla elementin, jonka alle se liittää palvelimelta saadun HTML-koodin. Lopputuloksena käyttäjä huomaa, että sivulle ilmestyi teksti “Hello, World!”.

Listauksessa haettava tiedosto message.html sisältää tekstin “Hello, World!”; muuta ei tarvita.

2.1.3 DOM vai innerHTML?

Ajaxia hyödyntävällä web-sivulla on usein tarve vaihtaa jonkin elementin sisällä oleva HTML-sisältö hakemalla palvelimelta tietoa käyttäjän toimien mukaan. Siirron valmistut-

Listaus 1 .

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <title>Ajax example</title>
  <script language="javascript" type="text/javascript">
    function getXMLHttpRequest() {
      if (window.XMLHttpRequest) {
        return new XMLHttpRequest();
      }
      else if (window.ActiveXObject) {
        return new ActiveXObject("Microsoft.XMLHTTP");
      }
      else {
        alert("Your browser doesn't implement XMLHttpRequest!");
        return null;
      }
    }
    var reqObj = getXMLHttpRequest();

    function greet() {
      if (reqObj.readyState == 4 || reqObj.readyState == 0) {
        reqObj.open("GET", "message.html", true);
        reqObj.onreadystatechange = messageReceived;
        reqObj.send(null);
      }
    }
    function messageReceived() {
      if (reqObj.readyState == 4) {
        var e = document.getElementById('messageContainer');
        e.innerHTML = reqObj.responseText;
      }
    }
  </script>
</head>
<body>
  <p><a href="javascript:greet()">Greet them all!</a></p>
  <p id="messageContainer"></p>
</body>
</html>
```

tua aiemmin määritelty funktio tekee tarvittavat toimet, joilla osa sivun sisällöstä muuttuu.

Käsitteillä DOM ja innerHTML tarkoitetaan tässä kahta eria tapaa muokata selaimen muistissa olevaa dokumenttipuuta. DOM:lla tarkoitetaan joukkoa JavaScript-luokkien metodeita, joiden avulla voidaan luoda uusia HTML-elementtejä, poistaa elementtejä ja muokata olemassa olevien attribuutteja. Toinen vaihtoehto (innerHTML) saman asian tekemiseen on hakea dokumenttipuusta elementti, jonka alle halutaan liittää valmis pala HTML-koodia. Siinä vain sijoitetaan JavaScript-olion innerHTML-attribuutin arvoksi merkkijono, joka on saatu Ajax-pyynnön avulla palvelimelta.

Sivun sisältöä muutettaessa tehdään usein valinta helppouden ja standardin menetelmän väliltä. Alunperin Internet Explorer -selaimen tuoma lisä, HTML-elementtien

innerHTML-ominaisuus, on paljon käytetty ja hyvin muissakin selaimissa tuettu tapa muuttaa HTML-elementin sisältö. Menetelmä on yksinkertainen ja helppo tapa liittää Ajax-pyynnöllä haettu HTML-sisältö dokumentin sisältämän elementin alle.

Listauksessa 1 esitetystä saadaan innerHTML-vapaa versio muuttamalla hieman messageReceived-funktiota, kuten näemme listauksessa 2.

Listaus 2 . Hello, World! toteutettuna ilman innerHTML-ominaisuutta.

```
function messageReceived() {
    if (reqObj.readyState == 4) {
        var e = document.getElementByID('messageContainer');
        e.appendChild(document.createTextNode(reqObj.responseText));
    }
}
```

innerHTML-ominaisuuden käyttöä vastaan on esitetty esimerkiksi seuraavat ongelmat: tietoturvaongelmat mikäli HTML-pätkään on mahdollisuus sisällyttää haitallista koodia[2] tai muistivuodot selaimessa mikäli operaatio aiheuttaa joidenkin tapahtumakäsittelijöihin liitettyjen elementtien poistamisen. Näiden lisäksi pienempiä ongelmia ovat esimerkiksi se, että uusiin elementteihin ei saada suoraa viitettä, vaan tarvittaessa sellainen on haettava normaalia tietä DOM-metodeilla, ja että innerHTML ei ole tuettu kaikilla elementeillä kaikissa selaimissa. [12]

Tietoturvariski on tietenkin olemassa siinä tapauksessa, että HTML-koodi sisältää jotain käyttäjältä peräisin olevaa syötettä. Tämän ehkäiseminen jo palvelimella on kuitenkin triviaalia. Muistivuodot selaimen tukemien operaatioiden seurauksena on tietenkin huono asia, mutta ei web-sovelluksen, vaan selaimen kehittäjien asia. Uusiin HTML-elementteihin viitteiden hakeminen ei usein edes ole tarpeen, ja silloinkin kun se on, ei se ole sen kummempaa kuin muissakaan dokumentin osissa. Vaikka selaimien tuki innerHTML-ominaisuudelle ei olekaan täydellinen, on se kuitenkin paljon kattavampi kuin monien DOM-standardissa mainittujen ominaisuuksien.

Molempien käyttöön on siis hyviä perusteita, ja tässä työssä käytetäänkin molempia. [10, s. 139-140]

[2] Kyseessä on niin sanottu Cross-Site Scripting (XSS), jota on käsitelty tarkemmin kappaleessa 2.2.2

2.2 Tietoturva

Ajax-sovelluksissa on omat haasteensa tietoturvan takaamiseksi. Tärkeää on, ettei selaimessa ajettava JavaScript-koodi sisällä mitään sellaista, jota vihamielinen taho ei saisi nähdä tai tietää, koska se on aina käyttäjän luettavissa ja kontrolloitavissa. Kaikki päätökset käyttäjän oikeuksiin liittyen tulee tehdä palvelimella ja käyttää Ajax-pyyntöjä vain lomakkeiden tietojen välittämiseen. [22]

Seuraavaksi käsiteltävät tietoturvaongelmat koskettavat lähinnä Ajax-sovelluksia, mutta yleisellä tasolla on paljon muitakin asioita, joita sekä Ajax- että muunkin sovelluksen tekijän tulee ottaa huomioon. Seuraava lainaus Kevin Mitnickin kirjasta "The Art of Intrusion" kuvaa hyvin ongelmaa sovellusten tietoturvariskien takana.

> *"I'm not a cryptanalyst, not a mathematician. I just know how people make mistakes in applications and they make the same mistakes over and over again."*
>
> – Former hacker turned security consultant [15]

Todellisuudessa lähes kaikki ongelmat, joita edellisessä lainauksessa tarkoitetaan, ovat eliminoitavissa. Hyvin suuri osa turvallisuusongelmista johtuu käyttäjän antamien syötteiden huolimattomasta käsittelystä. Esimerkiksi nimellä "SQL-injection" tunnettu tekniikka, jossa syötetään tarkoituksella virheellistä dataa, jonka sovellus liittää suoraan osaksi SQL-kyselyä, voidaan estää helposti triviaalisti, kunhan se muistetaan tehdä aina. [21]

Tietoturva-asiat eivät missään tapauksessa ole triviaaleja, mutta tärkeintä on muistaa ajatella myös potentiaalisia väärinkäyttötapoja ja suhtautua niihin huolellisuudella.

2.2.1 Haavoittuvainen sisäänkirjautuminen JavaScriptillä

Listauksessa 3 näemme, miksi JavaScriptillä toteutetun sisäänkirjautumisen toteutuksessa tulee olla hyvin huolellinen. Ongelmaa ei välttämättä heti huomaa; käyttäjätunnus ja salasana tarkistetaan $validateUser()$ -funktion toimesta ja vasta sen jälkeen kutsutaan $logUserIn()$-funktiota.

Vihamielinen käyttäjä kuitenkin huomaa lähdekoodia lukiessaan, että todellisuudessa salasanaa ei tarvitse tietää, vaan hän kirjoittaa selaimensa osoitekenttään: $javascript : logUserIn('tunnus')$ ja näin hän pääsee kirjautumaan ilman salasanaa.

Virhe on helppo korjata hoitamalla kirjautuminen pelkästään palvelimella. JavaScriptillä voidaan lähettää lomakkeen tiedot, mutta palvelimella olevan ohjelman tulee päättää itse uuden session luomisesta ja tietojensa pohjalta kontrolloida myöhemminkin sivupyyntöjä ja selvittää, onko käyttäjällä oikeus saada pyytämänsä tiedot.

Listaus 3 . Esimerkki haavoittuvasta sisäänkirjautumisesta.

```
function loginRequestCallback ()
{
    var userOk = validateUser ($F('user'), $F('pass'));

    if (userOk) {
        logUserIn ($('user'));
    }
    return false;
}


function logUserIn (user)
{
    // Tee Ajax-pyyntö kirjataksesi käyttäjä sisään
    new Ajax.Request ('check.php?action=login&user='+user, {
        method: 'post',
        onSuccess: function(loginResponse) {
            // Sisäänkirjautuminen onnistui
        },
        onFailure: function(loginResponse) {
            // Käsittele epäonnistunut yritys
        }
    });
}
```

2.2.2 Cross-Site Scripting

Cross-Site Scripting, josta käytetään lyhennettä XSS, on käytetyimpiä ja helpoimpia tekniikoita web-sivujen murtamisessa, joten jokaisen web-sovelluksia kehittävän tulisi tuntea nämä vaarat ja osata varautua niihin.

Cross-Site scripting -tekniikalla tarkoitetaan HTML-koodin ja siihen liitetyn JavaScript-koodin sisällyttämistä web-sivulle. Yleensä se on mahdollista siksi, että dynaamisen web-sivun mahdollistamaa käyttäjältä tulevaa syötettä ei ole tarkistettu asianmukaisesti. Sen uhka koskettaa yhtenäisesti kaikkia web-sovelluksia, eikä siten ole vain Ajaxia hyödyntävien sovellusten haaste. [22]

HTML-koodin sisällyttäminen web-sivulle ei kuulosta kovin vaaralliselta. Se kuitenkin mahdollistaa myös esimerkiksi script-elementtien sisällyttämisen sivulle, ja kun muistetaan että sitä kautta on mahdollista muodostaa yhteyksiä Ajax-pyyntöjen kautta tai ladata

kuvia muualta verkosta, niin ongelma on todellinen. Myös lomakkeen syöttäminen toiselle sivulle voi saada käyttäjän kirjoittamaan tunnuksensa ja salasanansa lomakkeeseen, joka lähettää tiedot kolmannen osapuolen palvelimelle.

Usein tietoturva-aukon muodostaa dynaamisesti luotu sivu, jolle annetaan parametrina jokin arvo, joka myöhemmin liitetään suoraan osaksi sivua tarkistamatta sisältöä. Tällöin HTML-koodin syöttäminen on helppoa. Esimerkiksi URL, joka on muotoa

$$http://localhost/unsecure.php?q = search + string \ ,$$

on helposti hyödynnettävissä ilman varotoimenpiteitä. Esimerkiksi:

$$.../unsecure.php?q =< img\ src = http://foo/bar.jpg\ width = 0\ height = 0 >$$

Kun vielä yhdistetään script-elementti ja tietoja evästeistä (cookie), on hyökkääjällä hyvät mahdollisuudet vaikkapa ottaa käyttäjän sessio haltuunsa lähettämällä JavaScript-koodin ja vaikka kuvan avulla käyttäjän kaikki evästeet omalle palvelimelleen.

Aiemmin tällaiseen hyökkäykseen vaadittiin, että käyttäjä itse aktivoi jonkin linkin sivulla. Nyt, Ajaxin avulla hyökkääjän voi joissain tapauksissa olla mahdollista syöttää esimerkiksi tietokantaan koodia, jonka toisten käyttäjien selaimet ajavat käyttäjien tietämättä asiasta mitään. Tämä koodi voi itse tehdä Ajax-pyyntöjä, ilman että käyttäjä saa siitä mitään visuaalista informaatiota.

Vaikka Ajax-pyynnöt on rajattu samalle palvelimelle, josta alkuperäinen sivu on ladattu, turvattoman web-sovelluksen aukkojen hyödyntämistä se ei estä. Haittasovellus voi tehdä pyynnön samalle palvelimelle ja huijata selaimen ottamaan muualle yhteyttä vaikkapa kuvan kautta.

Listauksessa 4 on esitetty kaksi funktiota, joista ensimmäinen on haavoittuva ja toinen saman asian tekevä XSS:n kannalta turvallinen versio. Kuten edellä todettiin, ongelma ensimmäisessä funktiossa on se, ettei käyttäjän antamaa syötettä ole mitenkään tarkistettu. Jälkimmäisessä HTML-koodin syöttäminen on tehty tehottomaksi käyttämällä $htmlspecialchars()$-funktiota, joka muuttaa HTML:n tietyt erikoismerkit entiteeteiksi, jolloin selain käsittelee koko sisällön kuten tavallisen tekstin, ja HTML-koodi näkyy sivulla sellaisenaan.

Tässäkään esimerkissä ei ole tarkistettu esimerkiksi syötteen pituutta, mutta vaikka syöt-

teen pituuden tarkistaminen olisikin hyödyllistä, ei se tässä muodosta ongelmaa. Web-palvelimen ja PHP:n asetuksista riippuen jompi kumpi ilmoittaa jossain vaiheessa että URI on liian pitkä. Mikäli HTML-koodia ei haluta sallia edes näytettävän, voidaan PHP:ssä käyttää esimerkiksi funktiota $strip\_tags()$.

Listaus 4 . GET-parametrin sisällyttäminen web-sivulle PHP-kielellä.

```
// Tämä koodi on altis hyökkäyksille
function showSearchParam () {
    printf ("<b>%s</b>", $_GET['q']);
}

// Tämä on XSS:n kannalta turvallinen
function showSearchParam () {
    printf ("<b>%s</b>", htmlspecialchars ($_GET['q']));
}
```

2.3 Muita käytettyjä tekniikoita

Listauksessa 2 esitetty minimalistinen Ajax-demo ei anna kovin realistista kuvaa Ajax-sovellusten rakenteesta. Siinä toteutettu toiminnallisuus onnistuu HTML- ja JavaScript-tekniikoiden yhdistelmänä eikä vaadi esimerkiksi web-palvelimelta muuta kuin staattisten HTML-sivujen tarjoamista.

Tässä työssä esiteltyjen asioiden lisäksi on olemassa vielä hyvin paljon erilaisia palvelimia, ohjelmakirjastoja, apuvälineitä ja tekniikoita, jotka liittyvät nykyaikaisiin web-sovelluksiin. Jo pelkästään XML-pohjaisia web-sovelluksiin liittyviä tekniikoita, joille on oma kirjainlyhenteensä, on olemassa kymmeniä. Tässä käsitellään hieman muutamia perustekijöitä, joita käytetään käytännössä aina web-sovellusten osana.

2.3.1 XHTML vs. HTML4

Perinteisesti webissä dokumentit on kirjoitettu HTML-kielellä, josta on eri versioita. Näistä nykyinenkin W3C:n suositus, HTML 4.01 pohjautuu vanhaan SGML-kieleen, ja sen SGML-määrittely on kohtuullisen löyhä. Tästä johtuen se sallii merkkauksessa epätarkkuuksia, kuten elementtien avaamisen niitä sulkematta, isoja ja pieniä kirjaimia elementtien nimissä ja attribuuteissa, attribuuttien arvojen esittäminen lainausmerkeissä tai ilman sekä paljon muuta. Listauksessa 5 on esitetty samanlainen esimerkkisivu perinteisellä HTML-merkkauksella (versio 4.01) ja toisessa XHTML-merkkauksella (versio 1.0). Molemmat merkkaukset ovat oikein muodostettuja (valid), eli ne noudattavat DTD:n määrityksia.

Näitä HTML 4:n ominaisuuksia käyttämällä ja muitakin merkkauksen epätarkkuuksia harjoittamalla on helppo horjuttaa selainten kykyä tulkita web-sivun lähdekoodia. [25]

Listaus 5 . Validi HTML4-dokumentti ja validi XHTML 1.0 -dokumentti.

```
<!-- Validi HTML 4 Transitional dokumentti -->
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
<HTML>
  <HEAD>
    <title>foo</title>
  </HEAD>
  <BODY>
    <p style=font-size:x-large>foo<br>
  </BODY>
</html>



<!-- Validi XHTML 1.0 Transitional dokumentti -->
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <title>foo</title>
  </head>
  <body>
    <p style="font-size: x-large">foo<br/>bar</p>
  </body>
</html>
```

XHTML-kieli on uusi, HTML:n korvaajaksi suunniteltu merkkauskieli, joka pohjautuu XML-metakieleen. Vuodesta 1996 alkaen kehitetty XML on SGML:n “kevennetty” versio, ja se on täysin yhteensopiva SGML:n kanssa. Yhteensopivuuden lisäksi esimerkiksi dokumenttien käsittelyn helppous ja suoraviivainen käyttö Internetissä olivat painavia suunnitteluperiaatteita XML-metakieltä laadittaessa. Tässä työssä onkin käytetty XHTML 1.0:n mukaista merkkausta, joka on W3C:n suositus. [24, 4]

2.3.2 Ulkoasun määritys: CSS

Tässä työssä web-sovelluksen ulkoasu on määritelty CSS-tyylien avulla. Niiden käyttö on hyvän tavan mukaista ja myös W3C:n suositus. Tyylimäärittelyjen käyttö pitää dokumentin rakenteen ja ulkoasun erillään; se on tärkeää, jotta dokumentin rakenne pysyy selkeänä ja tyylien muuttaminen onnistuu helposti. CSS on myös W3C:n suositus. CSS-tyylimääritysten ensimmäinen versio CSS1, julkaistiin jo joulukuussa 1996, ja tällä hetkellä CSS:n versio 2.1 on yleisessä käytössä, ja melko hyvin tuettu tärkeimmissä selaimissa, vaikka se onkin vasta “W3C Candidate Recommendation”. [23]

CSS-tyylien kohdistaminen tiettyihin elementteihin onnistuu useilla eri tavoilla. CSS-määrityksiä voidaan liittää tiettyyn elementtiin käyttämällä elementin style-attribuuttia HTML-koodissa. Tyyleille on myös oma elementtinsä, <style>, mutta usein käytetään erillisiä CSS-tiedostoja, joihin viitataan dokumentin head-osassa. Tällöin HTML-elementille, jonka yhteydessä tyyliä halutaan käyttää, määritellään luokka (class) tai tunniste (id), jolloin CSS-tiedostossa käytetään tiettyyn luokkaan viitattaessa luokkavalitsinta (class selector) tai tunnistevalitsinta (id selector). Myös kaikkien tietyn tyyppisten elementtien tyyliin on mahdollista vaikuttaa universaalin valitsimen (universal selector) avulla. Universaalit valitsimet antavat mahdollisuuksia myös erilaisten dokumenttirakenteiden hyödyntämisen tyylien määrittelyssä, jolloin voidaan esimeriksi määritellä että: "mikäli elementin x vanhempi on tyyppiä y, käytä näitä tyylimäärityksiä". CSS:n versio 2.1 tarjoaa paljon muitakin mahdollisuuksia ja osaa puutteista, ainakin dokumentin rakenteen suhteen, voidaan paikata palvelinpäässä, mikäli HTML-koodi tuotetaan dynaamisesti.

Myös CSS-tyylejä voidaan kontrolloida JavaScriptin avulla. Monien elementtien CSS-ominaisuuksia voidaan muuttaa käyttäjän toimien seurauksena. Esimerkiksi näppäinten painalluksia ja hiiren liikkeitä sekä lomakkeisiin (form) liittyviä toimintoja voidaan seurata. Kappaleessa 3.4.1 on käsitelty tarkemmin JavaScriptin DOM-tapahtumia (DOM event).

3 DEMOSOVELLUKSEN TOTEUTUS

Työn käytännön osuudessa luotiin web-sovellus Ajaxia käyttäen. Koko työ toteutettiin käyttämällä vapaita ohjelmistoja. Sovellus koostuu pääosin palvelimella ajettavasta PHP-ohjelmasta sekä JavaScript-pätkistä, jotka ajetaan selaimessa. Työssä käytettyjä ohjelmistoja olivat myös HTTP-palvelinohjelmisto Apache, relaatiotietokanta MySQL ja monet käytännön työvälineet kuten tekstieditori vim.

Kaikki työssä käytetyt ohjelmistot ovat vapaaohjelmia, eli ne ovat vapaasti käytettävissä lisensseillä, jotka Free Software Foundation luokittelee vapaiksi. [4]

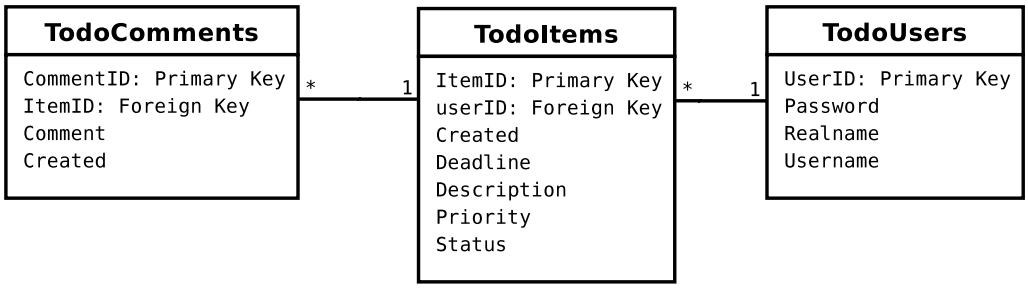
3.1 Web-palvelin

Työssä palvelinpuolella käytettiin Apachea, jota ajettiin Linux-alustalla. Palvelimen kannalta työllä ei ollut erityisiä vaatimuksia; PHP- ja MySQL-tuki sekä PHP:n normaaliasennukseen kuuluvat osat riittivät sovelluksen tarpeisiin mainiosti.

3.2 Tietokanta

Sovelluksessa käytetään tietokantanta MySQL-relaatiotietokantaa, johon saadaan helposti tallennettua kaikki tarvittava tieto. Samalla saadaan myös tallennettua käyttäjätunnukset ja salasanat.

Kuvassa 2 näemme työssä käytetyn, hyvin yksinkertaisen tietokantarakenteen, joka sisältää sovelluksen tarvitsemat taulut.



Kuva 2: Tietokantarakenne.

3.3 Palvelimella ajettava PHP-sovellus

Ajaxin käytön vaikutukset ulottuvat myös palvelimella ajettavaan ohjelmaan, vaikka JavaScriptin avulla tehty HTTP-pyyntö onkin aina samankaltainen. Sivun osien päivitystä varten tehdyt HTTP-pyynnöt vaativat erillisiä käsittelijöitään palvelimella ajettavaan ohjelmaan.

3.3.1 Ohjelman rakenteen suunnittelu

Ohjelman rakenteen suunnitteluun käytettiin aluksi kynää ja paperia. Perusluokkarakenne oli helpohko hahmottaa sovelluksen hyvin rajallisen koon takia.

3.3.2 Olio-ohjelmointi PHP:llä

PHP:n viidennessä versiossa, jota tässä työssä käytettiin, on jo varsin hyvät edellytykset olio-ohjelmointiin. Muutamia monissa muissa kielissä olevia helpottavia ominaisuuksia puuttuu vielä, mutta perusasiat onnistuvat hyvin. [3]

Periytyminen (inheritance) on tuettu Javan tapaan, eli moniperintä (multiple inheritance) ei ole suoraan tuettu. Useita rajapintoja voidaan kuitenkin toteuttaa samassa luokassa. Muodostin (constructor) on PHP:n uusissa versioissa aina nimeltään $\_\_construct()$. Kantaluokan muodostinta ei kutsuta implisiittisesti, vaan sen kutsuminen on jätetty ohjelmoijan harkinnan alaiseksi. PHP:n viidennen version mukana on myös tuki hajottimille (destructor), mutta niille on harvoin tarvetta PHP-sovelluksissa kielen luonteen ja sovellusten luonteen vuoksi.

Uudessa versiossa myös tiedon kapselointi (encapsulation) on tuettu käyttäen avainsanoja $private$, $protected$ ja $public$. Myös polymorfismi, reflektio sekä abstraktit luokat ja metodit kuuluvat PHP:n viidennen version ominaisuuksiin. [20, 3]

3.3.3 Apukirjasto HTML-koodin tuottamiseen

Verkkosovellusten haasteena on palvelimella suoritettavan sovelluksen lopputulos, sivun lähdekoodi. Siten ohjelmalogiikan sekaan on jollain tavalla sotkettava tietyllä tavalla muotoillun HTML-koodin tuottaminen. Usein koodin luettavuus kärsii siitä, että suuri osa ohjelmakoodista voi olla pelkästään HTML-koodin tuottamiseen, ja itse sovelluslogiikan kannalta yhdentekevää.

[3] Jotkut pitävät olio-ohjelmointia turhana PHP:n tapauksessa. Verkkosovellukset eivät kuitenkaan eroa pyrkimyksiltään muista sovelluksista, ja monissa tutkimuksissa on todettu olio-ohjelmoinnin olevan hyödyllistä projektin kannalta. [16]

Tässä työssä haluttiin erottaa suuri osa HTML-koodin tuottamisesta muista toiminnoista, ja kirjoitettiin muutamille tärkeille HTML-elementeille omat luokat, joiden kautta niiden tarvitseman HTML-koodin tuottaminen tehdään. Tällöin saadaan yksi abstraktiotaso lisää, eikä esimerkiksi sovelluksen osan, joka tuottaa HTML-taulukon, tarvitse tietää millaista HTML-merkkausta tarvitaan, vaan se voi käyttää valmista HTMLTable-luokkaa, jonka metodeilla $getHTML()$ tai $writeHTML()$ voidaan merkkaus tallentaa muuttujaan tai tulostaa suoraan.

Apukirjastoon toteutettiin luokat HTMLTable, HTMLOptionMenu, HTMLDiv, HTMLAnchor ja niiden kantaluokka HTMLElement. Kantaluokka toteuttaa iHTML-rajapinnan, jossa on määritelty kaikille HTML-elementeille yhteiset metodit $getHTML()$ ja $writeHTML()$.

3.3.4 Luokkarakenne

Suurin osa luokista oli helppo muodostaa. Useimpien luokkien tehtävä on vastata jonkin sivun osan HTML-koodin tuottamisesta, joko suoraan tai käyttäen muita luokkia apuna. Näiden luokkien lisäksi on esimerkiksi HTMLManager ja ContentBlock, joiden tehtävä on päättää syötteen perusteella minkä luokan olio osaa toimittaa halutut tiedot.

SQLInsert ja SQLFetch hoitavat tarvittavien tietojen hakemisen ja tallentamisen tietokantaan, joten muiden luokkien ei tarvitse tehdä suoria SQL-kyselyitä.

Lisäksi config-, util- ja constants -luokat tarjoavat staattisia metodeita erilaisten pikkuasioiden tekemiseen.

3.3.5 HTTP-pyyntöjen käsittely

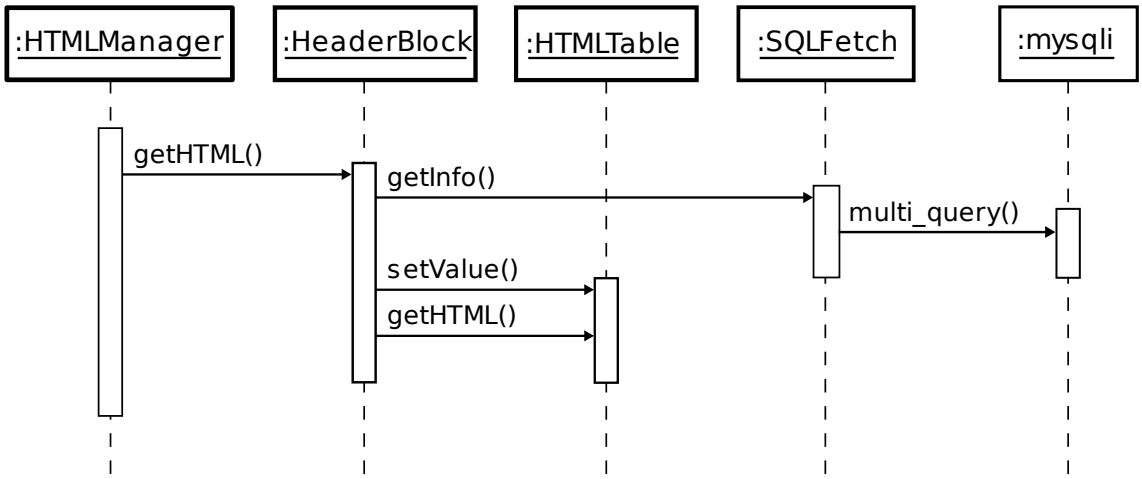
Kaikkien HTTP-pyyntöjen käsittely alkaa index.php-tiedostossa olevan ehtorakenteen kautta. Ensin käsittely jaetaan kahteen osioon sen mukaan, onko käyttäjä kirjautunut sisään vai ei. Osaan pyyntöjä on oikeus kirjautumatta ja osaan ei.

Sen jälkeen tutkitaan palvelimelle pyynnön mukana välittyneen do-parametrin sisältö. Sen arvoja voivat olla esimerkiksi “login”, “logout”, “postTodoItem”, “updateData” tai “gethtml”. Kirjautumistilan perusteella kahteen jaettu rakenne sisältää molemmissa tapauksissa switch-rakenteen, jossa on lista valideista arvoista ja lopussa oletusarvo, mikäli parametrin arvo ei vastaa mitään tunnettua arvoa.

Seuraavassa käsitellään tapahtumien kulku siinä tapauksessa, että do-parametrin arvo on “gethtml” ja block-parametrin arvo on “content”. Käytännössä tämä pyyntö tulee javascript-funktiolta. Vastaavat toimet suoritetaan, kun oletussivun tapauksessa generoidaan ensimmäinen etusivu ja tietysti siinä tapauksessa että selain ei tue JavaScriptiä, jolloin sovellus toimii ilman Ajax-pyyntöjä.

Eri sivun osien pyyntöjen hallitsemista varten toteutettiin HTMLManager-luokka, josta luodun olion tehtävä on päätellä saamastaan merkkijonosta, onko sitä vastaava sivun osa olemassa. Mikäli kyseinen osa löytyy, luo luokka osaa vastaavasta luokasta ilmentymän, jonka tulosteena syntynyt sivun osa lähetetään selaimelle. Selaimessa ajettava käsittelijä toteuttaa, vastauksen saatuaan, selaimessa näkyvillä olevan sivun päivityksen.

Alla olevassa tapahtumasekvenssikaaviossa nähdään tapahtumien kulku, kun sovellukseen liittyvän ylätunnisteen näyttämiseen tarvittavaa HTML-koodia pyydetään web-palvelimelta. Selaimelta tulee GET-pyyntö, jonka mukana tulee parametri “do”, jonka perusteella PHP-sovellus tietää, millainen toiminto halutaan suorittaa. Esimerkissä toiminto on “gethtml”, jolloin tehtävä annetaan HTMLManager-luokan oliolle. HTMLManager, tehtävänsä mukaan päättää parametrin “block” arvon “header” perusteella lähettää pyynnön getHTML() eteenpäin luokan HeaderBlock oliolle. HeaderBlock-luokan olio käyttää SQLFetch-luokan oliota tarvittavien tietojen hakemiseen ja lopulta palauttaa pyydetyn HTML-koodin HTMLManager-oliolle, joka palauttaa sen selaimelle.



Kuva 3: Sivun ylätunnisteen hakemista kuvaava tapahtumasekvenssikaavio.

Luokkarakenne on käytännössä samanlainen kuin sivun rakenne. Eri sivun osakokonaisuuksia vastaavat omat olionsa, joiden avulla PHP-koodi on helppo rakentaa järkevästi ja tietoturvasta on helppo huolehtia.

3.4 Selaimessa suoritettava JavaScript

Olennainen osa Ajax-sovelluksia on selaimessa suoritettava JavaScript-koodi, jota käytetään tavallaan palvelimen ja selaimen välissä. JavaScript-koodi tulkitsee käyttäjän toimenpiteitä ja tekee tarvittavia HTTP-pyyntöjä palvelimelle, ja muuttaa sivun rakennetta niiden avulla.

3.4.1 JavaScript ja DOM

DOM on malli, jonka avulla JavaScript-ohjelmoija lähestyy web-sivua. Siinä web-sivua tarkastellaan puumallina, jonka muodostavat HTML-elementit. DOM määrittelee tarvittavat toiminnot puun tehokkaaseen käsittelyyn ja muokkaamiseen.

JavaScriptiä tukevien selainten osalta käyttäjän toimiin reagointi alkaa selaimessa DOM-tapahtumien (DOM Event)[4] käsittelyllä. Käsittelyfunktiossa tehdään usein jokin HTTP-pyyntö, johon vastauksena saadaan dataa tai suoraan HTML-koodia, joka sitten siirretään osaksi koko dokumentin puurakennetta (document object tree), ja selain päivittää näkymänsä.

Vaikka DOM tarjoaakin menetelmän puurakenteen muuttamiseen, usein halutaan liittää palvelimelta saatu valmis HTML-koodin pätkä suoraan sivun jonkin elementin sisällöksi, mikä ei onnistu DOM-metodeilla. Standardoimaton innerHTML-ominaisuus auttaa tässä kohtaa; sitä käytetään tässä työssä ja myös prototypen useat toiminnot pohjautuvat sen käyttöön. innerHTML-ominaisuutta on käsitelty tarkemmin luvussa 2.1.3.

3.4.2 Prototype

Prototype on JavaScript-sovelluskehys (Framework), joka tarjoaa paljon hyödyllisiä toimintoja web-sovelluksen laatijalle. Se tarjoaa valmiiksi testattua koodia moniin toimintoihin, joita web-sovellukset usein tarvitsevat. Esimerkiksi XMLHttpRequest-objektin hieman erilaiset toteutukset valtaselaimissa on otettu huomioon Prototypen Ajax.Request-luokassa, eikä tavallisen sovelluskehittäjän tarvitse niitä pohtia.

[4]DOM on ohjelmointirajapinta, jonka avulla web-sivun muodostamaa puurakennetta voidaan muokata JavaScriptiä käyttäen. W3C:n määrittelemän DOM 2.0-version, joka on kohtuullisen hyvin tuettu valtaselaimissa, tapahtumat löytyvät dokumentista “Document Object Model (DOM) Level 2 Events Specification”, joka on W3C:n suositus.

Tässä työssä on käytetty prototypeä kaikkiin Ajax-pyyntöihin, koska se on eräs suosituimmista sovelluskehyksistä Ajax-käytössä ja tavallisesti ei ole syytä olla käyttämättä jotakin kehystä apuna. Selaimissa on yksityiskohdissa pieniä eroja, ja yksi abstraktiotaso välissä tarjoaa helpon tavan ottaa huomioon tulevat muutokset päivittämällä sovelluskehys uudempaan versioon.

4 TYÖN TULOKSET

Työssä perehdyttiin Ajaxin käyttöön web-sovellusten apuna. Tähän liittyen toteutettiin demo-tyyppinen web-sovellus, tjs. Siinä on pyritty tekemään asiat “oikein”; siten että Ajaxin käytöstä olisi hyötyä, mutta mahdollisimman vähän haittaa.

Demo-sovelluksen tekemisen yhteydessä ennakko-oletusten realisoituminen todellisiksi ongelmiksi toi työhön haasteita. Loppujen lopuksi ongelmat saatiin pitkälti ratkaistua ja ennalta suunnitellut asiat toteutettua.

4.1 tjs - web-sovellus tehtävien hallintaan

Tämän työn tuloksena syntyi tjs, web-sovellus tehtävien hallintaan. Toiminnoiltaan sovellus on hyvin suppea, mutta siihen on saatu toteutettua tarvittavat ominaisuudet, jotta sen avulla voidaan esitellä Ajax-tekniikan mahdollisuuksia.

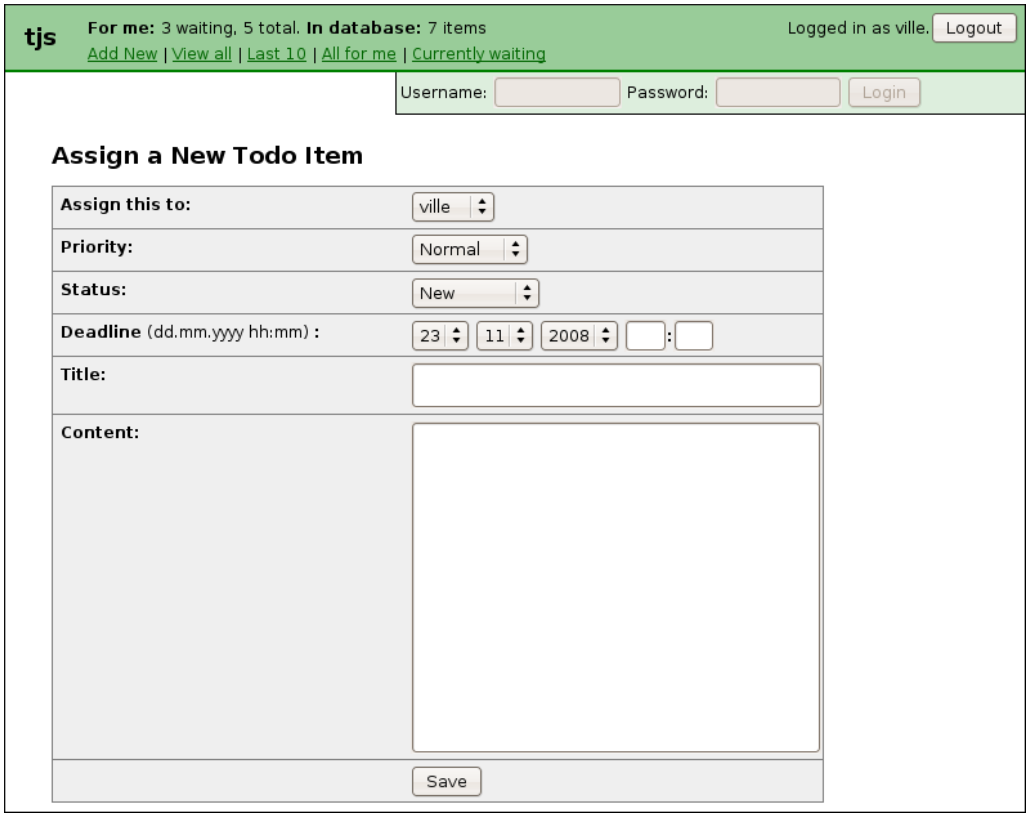
Itse sovellus tarjoaa mahdollisuuden pienryhmän kesken jaettavien tehtävien hallitsemiseen. Tällä hetkellä siinä on toteutettu uusien tehtävien lisääminen ja yksinkertaisia toimintoja tehtävien selaamiseen. Kuvassa 4 on esitetty kuvakaappaus, jossa näkyy tjs:n lomake uuden tehtävän syöttämiseen.

PHP-sovellus sisältää tyhjät rivit mukaan laskettuna noin 2000 riviä ohjelmakoodia, ja suuri osa siitä on “ylimääräistä” olio-ohjelmointiin liittyvää, kuten luokkamäärityksiä, joista tulee hieman enemmän koodia kuin hyvin tiiviisti kirjoitetusta versiosta. Vastaavasti kuitenkin ohjelman rakenne on hyvin helppo ymmärtää ja laajentaminen on tässä tapauksessa hyvin yksinkertaista.

4.1.1 Jatkokehitys

Varsinaista suunnitelmaa jatkaa ohjelmiston kehitystä tulevaisuudessa ei ole. Se on kuitenkin mahdollista, koska hyvä pohja on olemassa ja omassa sovelluksessa on aina se etu, että sitä on helppo laajentaa tarpeiden mukaan.

Työssä käytettyä HTML:n tuottamiseen tehtyä apukirjastoa kehitetään varmasti jatkossa, jotta siitä olisi myöhemmin vieläkin enemmän hyötyä. Se on hyvä esimerkki uudelleenkäytettävästä komponentista, jota voidaan käyttää myöhemmin ilman muutoksia muissakin projekteissa.



Kuva 4: tjs ja näkymä jossa voidaan lisätä tietokantaan uusi tehtävä.

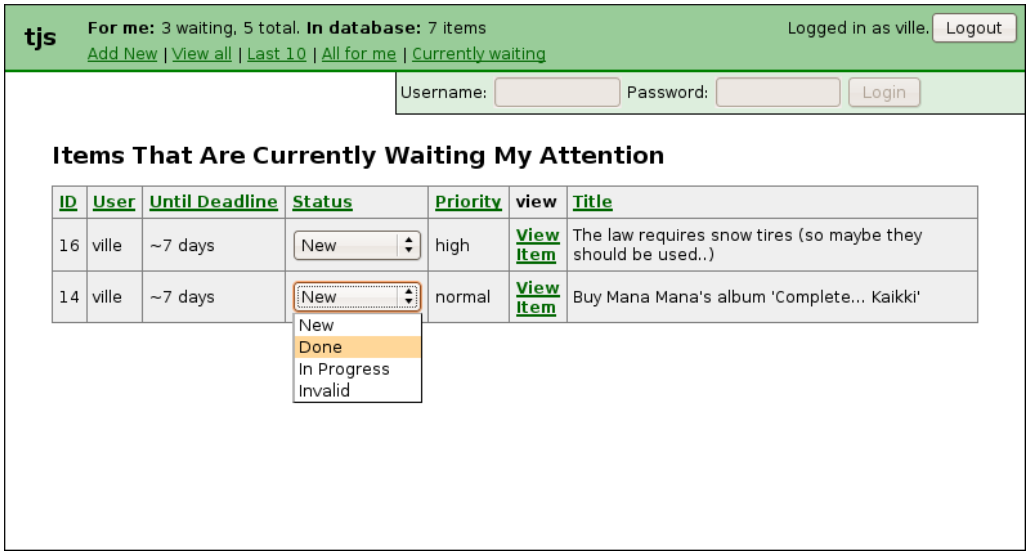
4.2 Ajax sovelluksen apuna

Mahdollisuudet Ajaxin kanssa ovat lähes rajattomat. Tällä hetkellä haasteet liittyvät selaimien tukeen eri toiminnoille. Nykyiset selaimet kuitenkin mahdollistavat jo pääsemisen työpöytäsovellusten kaltaiseen käyttäjäkokemukseen. Yhteensopivuusongelmien takia on kuitenkin syytä pohtia hyötyjä tarkkaan ennen Ajaxin käyttöönottoa.

Mikäli webiin toteutettava kokonaisuus on luonteeltaan selkeästi “sovellus”, voidaan Ajaxin käyttöä pitää perusteltuna, koska tällöin käyttäjältä voidaan helpommin edellyttää uudehkon Javascriptiä tukevan selaimen käyttöä. Muissa tapauksissa on syytä olla hyvin tarkkana mahdollisten Ajaxia käyttävien “pienten helpotusten” aiheuttamista ongelmista vanhoille selaimille, kämmenlaitteille, puhesyntetisaattoreille ja muille valtavirrasta poikkeaville selainratkaisuille.

Kuvassa 5 näemme kuvakaappauksen tjs-sovelluksen tilasta, jossa näkyvät tällä hetkellä suoritustaan odottavat tehtävät. Kuvassa on valittuna pudotusvalikko, josta voidaan vaihtaa tehtävän tilaa. Mikäli valitaan uudeksi tilaksi tehty (done), tehtävä häviää tältä näytöltä. Ajaxin takia koko sivua ei kuitenkaan tarvitse päivittää, vaan JavaScript-

tapahtumankäsittelijä lähettää tiedon muuttuneesta tilasta palvelimelle ja pyytää taulukon HTML-koodin uudestaan, ja saatuaan sen palvelimelta, asettaa sen osaksi dokumenttipuuta. Sen jälkeen selain huolehtii näkymän päivittämisestä.



Kuva 5: tjs: Tilan vaihtaminen pudotusvalikon avulla.

4.2.1 Toimivuus ilman Javascriptiä

Vaikka JavaScriptiä tukevien selaimia käyttävien ja JavaScriptin päälle kytkettynä pitävien osuus on kasvanut, edelleen osa käyttäjistä koettaa eri syistä tulla toimeen ilman sitä. Syitä ovat esimerkiksi pelko heikentyneestä tietoturvasta tai erikoinen käyttöympäristö, joka pakottaa yksinkertaisemman selaimen käyttöön. Web-kehittäjän kannalta olennaista on huomata, että palvelun tulee olla käytettävissä myös ilman JavaScript-koodia ajavaa selainta.

Listauksessa 6 näemme JavaScript-linkin toteutettuna siten, ettei se aiheuta ongelmia palvelua ilman JavaScript-tukea käyttäville. Mikäli JavaScript on käytössä, suoritetaan onclick-attribuutissa kerrottu JavaScript-funktio, ja palautetaan false, jolloin selain tietää, että sen ei tarvitse enää välittää onclick-tapahtumaa muille käsittelijöille. Eli tässä tapauksessa selain ei siirry linkin osoittamaan osoitteeseen, mikäli JavaScript on tuettu.

Listaus 6 Esimerkki JavaScript-linkin hyvästä toteutuksesta.

```
<a href="index.php?page=addnew" onlick="changeContent('addnew'); return false;">Add New</a>
```

4.3 Ajaxin ongelmat

Ajaxin käyttöön liittyy muutamia ongelmia, joihin kehittäjän täytyy kiinnittää huomiota. Ongelmat liittyvät pääosin perinteisestä eroavaan tiedonsiirtomalliin, jossa käyttäjän näkökulmasta eri osaan sivustoa siirryttäessä selaimen näkökulmasta pysytään samalla sivulla.

4.3.1 Kirjanmerkit (bookmarks)

Koska selaimen osoiterivin sisältö ei välttämättä kerro yksiselitteisesti sovelluksen tilaa, voi kirjanmerkkien kanssa tulla ongelmia. Usein voi käydä niin, että tehty kirjanmerkki osoittaa palvelun aloitussivulle, vaikka kirjanmerkkiä luotaessa sovelluksen tila olisikin ollut jotain muuta.

Tätä varten on luotu erilaisia JavaScriptillä selaimessa toimivia "korjauksia", jotka muuttavat osoiterivin sisältöä aina sovelluksen tilan muuttuessa, jolloin selaimen osoiterivillä näkyvä merkkijono yksilöi sovelluksen tilan, ja siihen on mahdollista palata uudelleen. [17]

Vaikka kirjanmerkit on mahdollista saada toimimaan siten, että sovelluksen eri tiloihin voidaan viitata osoitteella, on se monissa tapauksissa turhaa. Kun Ajaxia käytetään sovelluksen toiminnallisuuden kehittämiseen, usein riittää, että palvelun sisäänkirjautumissivu saadaan talletettua. Sovellusta käytettäessä on halutun toiminnon valitseminen yleensä suoraviivaista, ja mikäli ei ole, sen toteuttaminen kirjanmerkeillä toimivaksi olisi hyvin haastavaa.

Asia muuttuu jonkin verran mikäli Ajaxia sekoitetaan "tavallisen" verkkosivun sisältöön, jolloin olisi mielekästä tallentaa se kirjanmerkiksi. Usein näissä tapauksissa kuitenkin on järkevää joko jättää Ajax-toiminnallisuus pois tai toteuttaa osoiterivin muutokset hyvin yksinkertaisesti.

4.3.2 Navigointi selaimen eteen- ja taakse-toiminnoilla

Edellisessä kappaleessa mainitut ongelmat koskettavat myös selaimen eteen- ja taakse sivuhistoriassa vieviä toimintoja. Käyttäjät ovat tottuneet siirtymään linkkejä seuraten eri sivuille, ja tarvittaessa palaamaan takaisin edelliselle sivulle. Kun Javascriptin avulla linkkien seuraaminen johtaakin saman sivun muokkaamiseen, jolloin selaimen näkökulmasta pysytään samalla sivulla, sivuhistoria ei muutu, vaikka käyttäjä kokeekin, että siirrytään "toiselle sivulle".

Korjaaminen onnistuu samalla tavoin, kuin kirjanmerkkien tapauksessa edellä. [17]

4.3.3 Hakukoneet

Myös hakukoneita koskettavat edellisissä kappaleissa käsitellyt ongelmat. Hakukoneet löytävät tiensä uusille sivuille muilla sivuilla olevien linkkien kautta. Kun hakukone saa käsiteltäväkseen sivuston, jonka navigointi on toteutettu pelkästään JavaScriptin avulla, ei hakukone enää saa käsiinsä sitä sisältöä, jota sivustolla on haluttu julkaista. [10, s. 921-922]

Mikäli sivulla on sisältöä, joka halutaan hakukoneen saataville, täytyy linkkien toimia ilman JavaScript-tukea, eli kuten kappaleessa 4.2.1 on kuvattu. Kunhan tiettyyn sivun osioon on olemassa yksikäsitteinen URL, jonka avulla hakurobotti osaa ladata sivun, toimii indeksointi normaalisti mikäli hakurobotit yleensä päätyvät kyseiselle sivustolle.[5]

4.4 Webin tulevaisuus ja Ajax

Webin tulevaisuus, sivujen teknisen toteutuksen kannalta, on tietyiltä osin nähtävissä, koska tulevaisuudessa käyttöönotettavista standardeista monet ovat jo valmiina. Toiset kuitenkin otetaan ensin käyttöön ja sen jälkeen standardoidaan ei välttämättä edes standardoida. Ajaxin yhden rakennuspalikan, XMLHttpRequest-objektin osalta standardista on vasta luonnos olemassa, ja tekniikkaa on käytetty webissä yleisesti jo useampia vuosia.

Uusista tekniikoista, joiden parempaa tukea selaimissa odotetaan, ovat ainakin XForms, SVG, XHTML, DOM ja JavaScript ovat tekniikoita, jotka kehittyvät jatkuvasti ja erilaisten ominaisuuksien käyttö lisääntyy kun tuki selaimissa paranee. Näiden kehittyminen monipuolistaa Ajax-sovelluksia ja vähentää yhteensopivuusongelmia, kun selaimien tuki standardeille paranee.

Isompien muutosten ennustaminen on hieman hankalampaa. World Wide Webin alunperin kehittänyt Tim Berners-Lee pitää tiedon yhdistämistä (data integration) semanttisen webin (semantic web) kehittymistä tärkeänä tulevaisuuden tavoitteena. Siihen liittyy paljon erilaisia XML-metakieliä, kuten OWL ja RDF, joiden avullla tiedon rakennetta pystytään paremmin kuvaamaan.

[5] Ainakin hakukoneet Google ja Yahoo! seuraavat linkkejä ja usein päätyvät indeksoimaan sivun tietokantoihinsa ilman että verkkopalvelun ylläpitäjän täytyy tehdä mitään asian eteen. [6, 27]

Toinen Berners-Leen esiintuoma tulevaisuuden suunta on laitteiden monimuotoisuuden korostuminen. Hän tuo myös esille käsitteen kaikkialla läsnä olevasta webistä (ubiquitous web applications), jolloin erilaisia webiä hyödyntäviä näyttölaitteita näkyy joka paikassa. [1]

Molemmista edellä mainituista tekniikoista on puhuttu paljon muutenkin tietotekniikan kehittymisen yhteydessä. Muutoinkin web kehittyy laitteiden kehityksen myötä. Viime vuosina on saatu nähdä esimerkiksi videoiden tulo webiin hyvin paljon laajemmin kuin aiemmin.

Webin ilmiöiden ennustaminen on kuitenkin etukäteen hyvin vaikeaa tai mahdotonta; moni ei olisi uskonut 1990-luvulla että tänä päivänä vapaaehtoisvoimin ylläpidettävässä wikipediassa on tällä hetkellä jo yli 2,5 miljoonaa englanninkielistä artikkelia. Muita esimerkkejä webin ilmiöistä on lukuisia. [11]

5 YHTEENVETO

Ajaxille on webissä paikkansa ja sillä on omat rajoitteensa. Nykywebissä monenlaiset sovellukset vaativat enemmän kuin ennen; Ajax on yksi uusista tekniikoista, jotka tarjoavat lisää mahdollisuuksia web-kehittäjille. Ajaxia ei kuitenkaan pidä sotkea joka paikkaan vain Ajaxin käyttämisen takia.

Työssä toteutettiin Ajax-sovellus, jonka rajallinen toiminnallisuus saatiin toteutettua hyvin eri selaimilla toimivaksi. Se toimii esimerkkinä sen kaltaisesta web-palvelusta, jossa Ajaxin käyttöä voidaan perustella. Suppeutensa vuoksi se ei varsinaisesti nykytilassaan ole mikään malliesimerkki Ajaxin eduista, mutta demotarkoituksessaan se toimii mainiosti. Sovelluksen ohjelmointiin käytettiin palvelinpäässä PHP:tä ja sen mahdollistamaa olio-ohjelmointia, ja työn kylkiäisenä syntyi uudelleenkäytettävissä oleva apukirjasto HTML-koodin tuottamiseen PHP-sovelluksissa.

Ajaxin mahdollisuudet nimenomaan sovellusten kannalta ovat siinä, että oikein käytettynä Ajaxin avulla on mahdollista päästä lähelle työpöytäsovelluksen käyttäjäkokemusta. Enää ei tarvitse ladata koko käyttöliittymää uudestaan, kun jotain pientä sivulla muuttuu käyttäjän toiminnan seurauksena. Etenkin hitaan yhteyden ja raskaan käyttöliittymän kanssa toimittaessa edut voivat olla suuria.

Sovelluksissakin tulee ottaa huomioon käyttäjien tavat käyttää web-selaimia. Etenkin navigointitoimintojen ja kirjanmerkkien toiminta ei Ajaxia käytettäessä välttämättä vastaa tavallista web-sivustoa ilman erityisiä toimenpiteitä. Vaikka JavaScriptillä on mahdollista korjata puutteita, joissain tapauksissa voi olla järkevämpää toteuttaa yksinkertainen web-sovellus perinteisin keinoin, jolloin varmistutaan toimivuudesta ja saadaan samat asiat toteutettua yksinkertaisemmin.

Webin tulevaisuutta on hyvin vaikea ennustaa, mutta ainakin lähitulevaisuudessa Ajax tulee olemaan hyvin käytetty tekniikka uusien mahdollisuuksiensa takia ja siihen liittyvien toimintojen tuki sekä yhtenäisyys eri selaimissa paranevat koko ajan.

LÄHTEET

[1] Berners-Lee, Tim. Hearing on the "Digital Future of the United States: Part I – The Future of the World Wide Web"
http://dig.csail.mit.edu/2007/03/01-ushouse-future-of-the-web.html (2.11.2008)

[2] Connolly, Dan. A Little History of the World Wide Web.
http://www.w3.org/History.html (19.9.2008)

[3] Converse, Tim & Park, Joyce. PHP5 and MySQL Bible. Wiley, 2004. s. 370. ISBN: 0764557467

[4] Free Software Foundation. Licenses.
http://www.fsf.org/licensing/licenses/ (28.10.2008)

[5] Garrett, Jesse James. Ajax: A New Approach to Web Applications.
http://www.adaptivepath.com/ideas/essays/archives/000385.php (19.9.2008)

[6] Google. Usein kysytyt kysymykset.
http://www.google.fi/intl/fi/faq.html (2.11.2008)

[7] Google. Google Timeline.
http://www.google.com/corporate/timeline/cse/ (28.11.2008)

[8] IEBlog. Native XMLHTTPRequest object.
http://blogs.msdn.com/ie/archive/2006/01/23/516393.aspx (6.11.2008)

[9] The Internet Society. RFC 2616, Hypertext Transfer Protocol, 1999.
http://www.ietf.org/rfc/rfc2616.txt (21.9.2008)

[10] Holdener, Anthony T III. Ajax: The Definitive Guide. O'Reilly, 2008. ISBN: 978-0-596-52838-6

[11] Lai, Linda S. L. & Turban, Efraim. Groups Formation and Operations in the Web 2.0 Environment and Social Networks, 2008
Group Decision and Negotiation, 2008. vol. 17, issue 5.

[12] Lecomte, Julien. The Problem With innerHTML.
http://www.julienlecomte.net/blog/2007/12/38/ (26.10.2008)

[13] Lerdorf, R. & Tatroe, K. Programming PHP, First edition, 2002. ISBN: 1-56592-610-2

[14] Message, Robin ja Mycroft, Alan. Controlling Control Flow in Web Applications. ENTCS 200(3) pp. 119-131, 2008.

[15] Mitnick, Kevin. The Art of Intrusion: The Real Stories Behind the Exploits of Hackers, Intruders & Deceivers. Wiley, 2005. ISBN: 0764569597

[16] Haikala, Ilkka & Märijärvi, Jukka. Ohjelmistotuotanto. 11. painos, Talentum, 2006. ISBN: 952-14-0850-2

[17] Neuberg, Brad. AJAX: How to Handle Bookmarks and Back Buttons. http://www.onjava.com/pub/a/onjava/2005/10/26/ajax-handling-bookmarks-and-back-button.html (2.11.2008)

[18] O'Neill, Mark. Mapping Ajax's weaknesses, 2007. Infosecurity: (volume 4, issue 6) p. 38

[19] O'Reilly, Tim. What is Web 2.0. http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html

[20] PHP Documentation Group. Classes and Objects (PHP 5). http://fi.php.net/manual/en/language.oop5.php (6.11.2008)

[21] PHP Documentation Group. Security. http://fi.php.net/manual/en/security.php (6.11.2008)

[22] Ritchie, Paul. The security risks of AJAX/web 2.0 applications, 2007. Network Security (vol. 2007, issue 3) s. 4

[23] World Wide Web Consortium (W3C). Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification http://www.w3.org/TR/CSS21/ (26.9.2008)

[24] World Wide Web Consortium (W3C). Extensible Markup Language (XML) 1.0 (Fourth Edition) http://www.w3.org/TR/xml/

[25] World Wide Web Consortium (W3C). Comparison of SGML and XML. (W3C Note 15. joulukuuta 1997) http://www.w3.org/TR/NOTE-sgml-xml-971215 (19.9.2008)

[26] World Wide Web Consortium (W3C). XHTML 1.0 The Extensible HyperText Markup Language (Second Edition) http://www.w3.org/TR/xhtml1/ (19.9.2008)

[27] Yahoo! Search Help.
http://help.yahoo.com/l/us/yahoo/search/ (2.11.2008)