AJAX-TEKNIIKKA

LAHDEN AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka
Opinnäytetyö
5.10.2007
Erno Viitanen

Lahden ammattikorkeakoulu
Tietotekniikan koulutusohjelma

VIITANEN, ERNO: Ajax-tekniikka

Ohjelmistotekniikan opinnäytetyö, 48 sivua

Syksy 2007

TIIVISTELMÄ

Tässä opinnäytetyössä perehdytään Ajax-tekniikkaan: mikä on Ajax, mitä sillä tekee ja mihin sen avulla pystytään. Työssä tarkastelun kohteena on myös Ajaxin koostavien tekniikoiden pääperiaatteet. Lisäksi käsittelyssä on suuren suosion saanut Web 2.0 - arkkitehtuuri, jonka yksi tärkeimmistä avainkomponenteista on Ajax.

Opinnäytetyön käytännön osuuden tarkoituksena oli suunnitella ja toteuttaa sovellus, jolla pystyy ohjaamaan selaimen avulla talon sisällä olevia laitteita, kuten valaistusta, lämpöpattereita ja ilmastointia. Työn tuloksena on toimiva selainpohjainen suunnittelusovellus, jonka avulla talon pohjapiirustuksen avulla pystyy suunnittelemaan laitekomponenttien sijainti ja toiminta talon sisällä sekä ohjaamaan talossa olevia laitteita selaimen avulla. Sovellus toteutettiin käyttäen Ajax-tekniikkaa.

Ohjelmointikieliä opinnäytetyön työosuuden tekemisessä oli PHP ja Ajaxin koostavat tekniikat. Sovellus toimii tavallisessa internet – ympäristössä, käyttäjän selaimen täytyy tukea JavaScriptiä ja XMLHttpRequest – objektia.

Asiasanat: Ajax, Web 2.0, Asynkroninen JavaScript ja XML, monipuoliset verkkosovellukset, dynaamiset verkkosovellukset

Lahti University of Applied Sciences
Faculty of Technology

VIITANEN, ERNO: Ajax technology

Bachelor's Thesis in Software Engineering, 48 pages

Autumn 2007

ABSTRACT

This thesis is deals with Ajax technology: what is Ajax and what it is capable of. The main principles of the techniques that Ajax is composed of are also under inspection. Ajax is one of the key components of Web 2.0 architecture, and therefore the thesis also gives general survey of Web 2.0.

The objective of the practical part of the thesis was to design and implement an application that controls standard household equipment like the lighting, radiator and air conditioning by means of a web browser. The result of the work was a functioning designing application. Designing the places of the household equipment and the operation of the equipment can be done with the help of the application. The household equipment can be controlled remotely from the application. Ajax technology was used in the implementation of the application.

The programming languages used in the implementation of the application were PHP and some of the techniques that Ajax is composed of. The application runs in an ordinary Internet environment, and the web browser of the user must have the support for JavaScript and the XMLHttpRequest object.

Keywords: Ajax, Web 2.0, Asynchronous JavaScript and XML, versatile web applications, dynamic web applications

SISÄLLYS

1 JOHDANTO

Lukemattomien aliverkkojen muodostama maailmanlaajuinen tietoliikenneverkko, yleisimmin tunnettu nimellä internet, on kehittynyt vuosien varrella paljon. Se syntyi alun perin 1960-luvulla Yhdysvalloissa tutkimusprojektina nimellä ARPANET (U.S. Advanced Research Projects Agency). Projektin tarkoituksena oli synnyttää ydiniskun kestävä luotettava tietoverkko. Myöhemmin 1980-luvulla ARPANET hajotettiin kahtia, jolloin sotilaallinen toiminta siirtyi MilNet – verkkoon ja ARPANET jäi tiedeyhteisölle. ARPANET koostui muutamasta huippuluokan tutkimuslaitoksesta, jonka tarkoitus oli mahdollistaa tieteellisten tutkimustietojen jako. Tutkijat käyttivät internetiä tutkimuspapereiden vaihtoon, ja yliopistot julkaisivat pitämistään luennoistaan tarkempia tietoja. Liikemaailmalla ei ollut siihen aikaan mitään tietoa internetin mahdollisuuksista.

Internet suunniteltiin aluksi siten, että kaikki verkkosivut olisivat staattisia – käyttäjät pyysivät jotain resurssia, jonka palvelin lähetti käyttäjille. Sivustot olivat pääasiassa tekstidokumenttien sähköisiä kopiota. Ei kestänyt kauaakaan, kun käyttäjät kaipasivat sivustoilta enemmän dynaamisuutta ja interaktiivisuutta. Tästä syystä kehitettiin palvelinsovelmat, kuten ASP ja PHP, joiden tarkoituksena oli mahdollistaa dynaamisen sisällön luonti palvelimella ja palauttaa se staattisena käyttäjälle. Ongelma palvelinsovelmissa oli sivujen uudelleen lataus. Aina kun käyttäjä lähetti palvelimelle resurssipyynnön, koko sivu jouduttiin päivittämään aina uudelleen. Päivitysongelmia ruvettiin kiertämään ns. ”purkkaviritelmien”, kuten HTML – kuvauskielen IFRAME tagien avulla, mikä mahdollisti sivustojen osien päivityksen, tai päivitykset pystyttiin tekemään käyttäjältä näkymättömissä. Se ei kuitenkaan ollut optimaalisin ratkaisu päivitysongelmaan. Webin alkuperäisen käyttötarkoituksen mukaan web-sovellusten toiminta on perustunut palvelimelle lähetettyihin HTTP-pyyntöihin, jonka käyttäjä on laukaissut selaimen käyttöliittymän avulla. Palvelin käsitteli pyynnön ja palautti vastauksen selaimelle. Jokainen verkkosovellusten hitauteen närkästynyt käyttäjä kuitenkin tietää, ettei tämä välttämättä ole paras tapa

rakentaa sovelluksia. Tällä hetkellä merkittävin ja paljon mielenkiintoa saanut vaihtoehto monipuolisten verkkosovellusten rakentamiseen on Ajax. Ajax-tekniikka mahdollistaa sivuston osien päivittämisen asynkronisesti ja sen avulla palvelimen ja selaimen välillä siirrettävän tiedon määrä saadaan pienemmäksi.

2 WEB-SOVELLUSKEHITYS JA TEKNIIKAT

2.1 Web-sovelluskehitys

Web-sovellusten kehitys on tehnyt selväksi internetin aseman yhä merkittävämpänä tietojärjestelmien toteutusympäristönä. Aluksi internet-pohjaiset järjestelmät olivat hypertekstilinkein yhdistettyjen staattisten web-sivujen kokoelmia. Käyttö soveltui vain tapauksiin, joissa tieto ei muutu ajan mittaan paljoa. Tietorakenteena tällaiset järjestelmät olivat ongelmallisia hallita. Pian otettiinkin avuksi relaatiotietokannat rakenteellisen tiedon taltiointiin, ja osa web-sivuista luotiin dynaamisesti esimerkiksi tietyn SQL – tietokantakyselyn avulla. Tällaisia sovelluksia kutsuttiin *Thin-client* – toteutuksiksi, jolloin työasemassa oli ajonaikaisesti vain selain, ja kaikki tiedon prosessointi tapahtui web-palvelimella. Tällä hetkellä tekniikka ja kehitys ovat mahdollistaneet jo monipuolistenkin järjestelmien suunnittelun ja rakentamisen. (Kilpikivi 1997.)

Tietoturva on syystäkin ollut todella tärkeä aihe koko web-historian ajan. Tekniset mahdollisuudet ovat nykypäivänä kyllä jo varsin monipuoliset, mutta silti vieläkään web-palvelimet, -selaimet ja kehitysvälineet eivät sisällä kaikkia mahdollisia tietoturvaominaisuuksia. (Kilpikivi 1997.)

Internet-arkkitehtuuri ja – tekniikka ovat periaatteeltaan erittäin yksinkertaisia, eikä niissä ole mitään ”ihmeellistä”. Kiinnostavaa siinä on se, että internet on integroitavissa myös olemassa oleviin tietojärjestelmiin. Tämä taas mahdollistaa sen, että tarvittava tieto ja järjestelmät on saatavilla mistä päin maailmaa tahansa. (Kilpikivi 1997.)

2.2 XHTML

2.2.1 Yleistä

XHTML – merkintäkieli (Extensible Hypertext Markup Language) perustuu vanhempaan HTML-kieleen, joka on XML-kielen (Extensible Markup Language) syntaksien mukaisesti määritelty. Itse Ajax koostuu HTML- tai XHTML – merkintäkielestä sekä muiden tekniikoiden yhdistelmistä. HTML – tekniikan vanhennuttua ajallisesti tässä opinnäytetyössä käsitellään pääasiallisesti vain XHTML – kieltä. Tärkeimmät erot XHTML:n ja HTML:n välillä ovat XHTML:n tiukemmat muotosäännöt sekä jotkin XHTML:stä poistetut tagit ja attribuutit, joita HTML-kieleen on jätetty. XHTML – merkintäkielessä kaikki tunnisteet on kirjoitettava pienillä kirjaimilla, attribuuttien nimet on merkittävä lainausmerkeillä, ja elementit on aina lopetettava tagilla.

Elementiksi kutsutaan aloitus- ja lopetustagin ja itse sisällön ryhmittymää. Attribuutit taas laajentavat elementin merkitystä. Koska XHTML on osa XML:ää, attribuutit noudattavat XML-sääntöjä. XHTML-attribuutit sijoitetaan aina aloitustagiin, ja sijoitettavan arvon kuuluu olla lainausmerkeissä. (Boumphrey, Greer, Raggett, Schnitzenbaumer, & Wugofski 2001.)

2.2.2 Versiot

XHTML:stä on julkaistu kolme virallista versiota, ja neljäs on tekeillä. XHTML:n versio 1.0 on suoraan johdannainen HTML 4.0 kielestä, lukuunottamatta poikkeusta että XHTML elementtien syntaksi noudattaa XML-kieltä. XHTML:n ensimmäistä versiota voidaan kuvailla siirtymäkauden versioksi, jonka tarkoitus oli helpottaa HTML-ohjelmoijien siirtymistä XML:n vaatimaan syntaksiin. XHTML 1.0 sisältää ulkoasuun liittyviä elementtejä, joiden vuoksi sitä ei voi pitää puhtaana XML-sovelluksena. XHTML 1.0:sta on kolme eri määrittelyä: *Strict*, *Transitional* ja

Frameset. *Strict*-määrittely on nimensä mukaisesti kaikista tiukin noudattamaan XML-kielen syntaksia estämällä mm. muotoiluelementtien käyttämisen. *Transitional*-määrittely on *Strict*-määrittelyn ”vastakohta” eli mahdollisimman yhteensopiva HTML 4.0 kanssa, määrittely sisältää muotoiluelementtejä. *Frameset* määrittelyä käytetään kehyksiä hyödyntävillä sivustoilla. (XHTML-Opas 2007.)

XHTML version 1.0 jälkeen julkaistiin uusi versio nimeltä XHTML Basic. XHTML Basic oli modulaarisen XHTML – kuvauskielen ensimmäinen askel. Modulaarisuudella tässä tarkoitetaan kielen erikoiselementtien jakoa erillisiin moduuleihin, jotka määrittävät elementtejä ja attribuutteja. XHTML Basic version tarkoitus oli muodostaa nk. ydinmoduuli, jota käytettäisiin kaikissa XHTML sovelluksissa ja sovelluksia käyttävissä päätelaitteissa, kuten mobiililaitteissa. XHTML – kuvauskielen kannalta XHTML Basic sisältää ainoastaan olennaiset elementit, tekemällä XHTML Basic versiosta nimensä mukaisesti erittäin pelkistetyn. (XHTML-Opas 2007.)

Viimeisin virallisin versio XHTML – kuvauskielestä on XHTML 1.1, jonka moduulijako on hieman laajentunut ja XML-standardia on noudatettava entistä tiukemmin. XHTML versio 1.1 ei sisällä lainkaan dokumentin ulkoasun määrittelevää elementtiä. Kaikki ulkoasuun liittyvät määrittelyt tehdään tyylimäärittelyiden avulla. (XHTML-Opas 2007.)

2.2.3 Dokumentti ja sen rakenne

XHTML – tiedosto on ASCII muotoinen tekstidokumentti, jonka pääte voi olla ”.htm”, ”.html” tai ”.xhtml”. XHTML:n ydintä kutsutaan sanalla rakennemoduuli, koska se määrittää XHTML-sisällön. Rakennemoduuli koostuu html-, title-, head- ja body - elementeistä, jotka ovat dokumentin tärkeimpiä rakenneosia. Rakennemoduulin elementit ovat käytettävissä kaikissa XHTML versioissa. (XHTML-Opas 2007.)

XHTML – dokumentin rakenne eli XHTML – tiedosto koostuu kolmesta ns. pääosasta, jotka ovat dokumenttityypin määrittely, dokumentin ylätunniste ja dokumentin runko. Dokumenttityypin määrittely kertoo selaimelle, millaista syntaksia dokumentissa käytetään, mutta käytännössä dokumenttityypin määrittelyn tarkoitus on sivun tarkistamista eli validointia varten. Dokumenttityypin määrittely kertoo myös, mitä kieltä ja kielen versiota dokumentissa käytetään. XHTML – kuvauskieltä käytettäessä on pakko käyttää dokumenttityypin määrittelyä. Dokumentin ylätunnisteeseen kuuluu head – elementti ja sen sisällä olevat elementit. Dokumentin rungoksi kutsutaan body – elementtiä. (XHTML-Opas 2007.)

XHTML – dokumentin juuren muodostava html-elementti sijaitsee alimmalla tasolla ja se koostuu kahdesta rakenneosasta, head- ja body - elementeistä. Html – elementin ainoa pakollinen attribuutti on *xmlns*, joka kertoo XHTML – dokumentin nimiavaruuden (KUVIO 1). XHTML – dokumentteja käsiteltäessä nimiavaruus on aina http://www.w3.org/1999/xhtml. (XHTML-Opas 2007.)

```
1  <html xmlns = "http://www.w3.org/1999/xhtml">
2  </html>
```

KUVIO 1. XHTML – dokumentin nimiavaruus

Head – elementti sijoitetaan html – elementin jälkeen ja se sisältää XHTML – dokumentin otsikkotiedot. Otsikkotiedot ovat käyttäjältä näkymättömissä ja ne sisältävät ohjeistuksen selaimelle ja hakukoneille. Otsikkotietoja ovat mm. dokumentin otsikko (title – elementti), tyylimääritykset (style – elementti), hakukonekuvaukset (meta – elementti) ja mahdolliset linkit ulkoisiin tiedostoihin (link – elementti). Title – elementti määrittää dokumentin otsikon, joka näkyy selaimessa ylhäällä tilarivillä. Title – elementti on XHTML:ssä pakollinen, joka sijoitetaan esimerkin mukaisesti head – elementin sisälle (KUVIO 2). (XHTML-Opas 2007.)

```
1  <head>
2  <title>Esimerkki otsikko</title>
3  <meta name="author" value="Erno Viitanen" />
4  <link rel="stylesheet" type="text/css" href="style.css" />
5  </head>
```

KUVIO 2. XHTML – dokumentin title – elementti

Body – elementin paikka on head – elementin jälkeen, jonka avulla määritellään XHTML – dokumentin runko. Body – elementti määrittää varsinaisen dokumentin, joka näkyy käyttäjän selaimessa. Body – elementille voidaan määrittää kaksi eri attribuuttia: onload (KUVIO 3) ja onunload. Onload-komento suoritetaan dokumentin latautuessa, ja onunload-komento suoritetaan, kun dokumentista ollaan poistumassa. (XHTML-Opas 2007.)

```
1  <body onload="doJavaScriptFunction();">
2  </body>
```

KUVIO 3. Body - elementtiä ladattaessa suoritettava JavaScript funktio

XHTML:ssä on pääelementtien lisäksi myös muita elementtejä. Muut elementit sijoitetaan pääelementtien sisään, jotka voivat olla tyypiltään lohko-, taulukko-, luettelo-, lomake- tai rivinsisäisiä elementtejä. Elementeillä on ominaisuuksia, jotka määrittävät sen piirteitä ja sisältävät ohjeita kyseisestä elementistä. Ominaisuuksilla pystytään mm. yksilöimään elementti (id), luokittelemaan elementti (class) tai vaikka otsikoimaan elementti (title). Lisäksi eri elementeillä on ns. yksilöllisiä ominaisuuksia, kuten kuvaelementillä kuvan lähde (src) tai tauluelementillä korkeus (height) ja leveys (width). Elementtien ominaisuudet määritellään ns. attribuuttien avulla. Elementit kirjoitetaan XHTML – merkintäkielessä aina pienellä, ja niillä täytyy aina olla aloitus ja lopetusmerkinnät. Elementeillä on myös ns. arvojärjestys, joka määrää, missä järjestyksessä elementit sijaitsevat dokumentissa. Elementit kuvaavat vain ja ainoastaan niiden sisällä olevaa tietoa ja sen rakennetta, itse dokumentin ulkoasu määritellään tyylimäärittelyllä. (XHTML-Opas 2007.)

XHTML – merkintäkielessä on myös ns. entiteettejä, joiden avulla pystytään ilmaisemaan skandinaavisia kirjaimia tai matemaattisia merkkejä. ASCII-koodauksena esimerkiksi ä – kirjain ilmaistaisiin entiteetillä ”&auml”. Täydellinen entiteettilistaus löytyy internetistä mm. RFC – suosituksista. XHTML:ssä kommenttien avulla pystytään upottamaan dokumentin lähdekoodiin tekijän omia ohjeita ja huomautuksia (KUVIO 4). Kommentit eivät tulostu dokumenttia selaaville käyttäjille, joten niitä käytetään yleensä selventämään dokumentin rakennetta ja itse lähdekoodia. Kommentit kuitenkin pystytään lukemaan selaimella (näytä lähdekoodi), joten kommentteihin ei pidä sijoittaa esim. salasanoja tai käyttäjätunnuksia. (XHTML-Opas 2007.)

```
1  <!- Kommentti kirjoitetaan näin -->
```

KUVIO 4. HTML- ja XHTML-kommentti

2.3 CSS

2.3.1 Tarkoitus ja versiot

CSS (Cascading Style Sheet) on WWW-dokumenteille kehitetty tyyliohjejärjestelmä, jonka avulla määritellään HTML, XHTML ja XML – dokumenttien ulkoasua ja esitystapaa. 1990-luvulla HTML-kieli lakkasi vastaamasta sille asetettuja vaatimuksia. Selainvalmistajat joutuivat selainsodan (Internet Explorer ja Netscape Navigator) syystä lisäämään HTML-kieleen selainkohtaisia piirteitä, joilla määriteltiin mm. tekstin ulkoasua. Tämä johti ongelmaan joka esti sivustojen toimimisen eri selaimilla, minkä syystä alettiin kehittää tyyliohjejärjestelmää. Kehitystyön tavoitteena oli itse asiakirjan sisällön ja sen rakenteen erottaminen toisistaan, ja työn tuloksena syntyi CSS – tyylikieli. (CSS – Opas 2007.)

CSS – tyylikielen avulla saadaan tarvittavan XHTML – kuvauskielen määrää vähennettyä jopa 60 %. CSS – tyylimäärittely voi sijaita ulkoisessa CSS – tyylitiedostossa (KUVIO 5), tai tyylimäärittely voidaan upottaa osaksi XHTML – dokumenttia. Mikäli samaa CSS – tyylimäärittelyä käytetään useammalla kuin yhdellä sivustolla, kannattaa käyttää omaa tyylitiedostoa, jonka pääte on ”.css”, muutoin CSS – määrittely voidaan upottaa osaksi XHTML – koodia. (CSS – Opas 2007.)

```
1  <link href="style.css" rel="stylesheet" type="text/css">
```

KUVIO 5. Ulkoinen CSS – tyylimäärittely

CSS – tyylikielestä on kaksi ns. virallista versiota, CSS 1 ja CSS 2 / 2.1. W3C suositteli CSS 1:n standardiksi vuonna 1996, ja standardi uudistettiin vuonna 1999. CSS 1 määritti CSS:n syntaksin ja ominaisuudet, mm. yksiköt, fontit ja laatikkomallin. CSS 2 laajensi CSS 1:n ominaisuuksia lisäämällä kansainvälisiä piirteitä, ja vuonna 1998 W3C suositteli CSS – tyylikielen 2 version standardiksi. Kuitenkin vuonna 2004 versio 2 avattiin uudelleen korjauksia ja tarkennuksia varten ja se odottaa vieläkin hyväksyntää standardiksi (versio 2.1). Vuonna 1998 aloitettu kehitystyö tulevalle CSS 3 on vielä luonnosasteella. CSS 3 sisältää lisää laajennuksia edellisiin versioihin mm. uusilla valitsinmalleilla, käyttäytymismalleilla ja muilla kehittyneillä piirteillä. CSS – tyylikielestä kehitettiin myös nk. CSS-MP (CSS Mobile Profile), joka on CSS:n alamurre ja se on suunnattu mobiililaitteille. CSS-MP:ssä on otettu huomioon mm. mobiililaitteiden rajoittunut suorituskyky. Nykyselainten tuki toimia eri CSS – versioilla vaihtelee. CSS 1:lle on täydellinen tuki melkein jokaisessa selaimessa. CSS 2 / 2.1:lle vain muutamissa selaimissa on täydellinen tuki. Selaimien tuki CSS 3:lle on varsin vaihteleva: CSS-MP:lle tuki löytyy melkein jokaisesta mobiiliselaimesta, vaikkakin itse CSS-MP on edelleen luonnosasteella.
(CSS – Opas 2007.)

2.3.2 Tyylikielen periaate

CSS – tyylikielen keskeinen toimintaperiaate on erottaa asiakirjan rakenne ja rakenteen esittävä ulkoasu toisistaan. CSS on piirteiltään rikas tapa kuvata rakenteellisen dokumentin ulkoasu, yksinkertaisuutensa ja joustavuutensa ansiosta se eroaa muista tyylikielistä (DSSSL ja XSL). (CSS – Opas 2007.)

CSS – säännöt saadaan linkitettyä itse dokumentin rakenteeseen ns. valitsinten avulla. Valitsimella tarkoitetaan dokumentissa olevaan elementtiin, johon viitataan. Esimerkiksi XHTML:n elementti p:lle (tagi <p>) määriteltäisiin tyyli käyttämällä elementtivalitsinta p. Kaikilla CSS – määriteltäville elementeille on yksi yhtenäinen perussyntaksi (KUVIO 6). Kuviossa 7. määritelty on määritelty otsikkoelementeille väri- ja fonttiominaisuudet. (CSS – Opas 2007.)

```
1  valitsin{
2      ominaisuus: arvo;
3  }
```

KUVIO 6. CSS – elementtien perussyntaksi

Toisin kuin XML – dokumenteissa, XHTML – dokumenteissa mikään elementtivalitsin ei ole ns. ”case-sensitive”, eli kirjainkoolla ei ole väliä. Myös kaikki CSS:n sisäiset ominaisuudet ja yksiköt ovat riippumattomia kirjainkoosta.

```
1  h1{color: red;}
2  h2{color: black;}
3  h3{color: white;}
4  h1, h2, h3{font: 12px arial;}
```

KUVIO 7. Otsikkoelementtien väri- ja fonttiominaisuuksien määrittely

2.4 XML

2.4.1 Yleistä

XML (Extensible Markup Language) on World Wide Web Consortiumin (W3C) projekti ja julkinen formaatti. XML on myös XHTML:n tapaan merkintäkieli ja ns. metakieli (Metakieli on kielen määritelmä), sen avulla laajojen tietomassojen jäsentäminen on selkeämpää. Merkintäkielellä kuvataan dokumentin muotoa, eli sillä kuvataan sitä, miten dokumentin sisältöä tulkitaan. XML – merkintäkieltä ei kuitenkaan ole XHTML:n tapaan tarkoitettu sivun kuvauskieleksi, vaan sillä kuvataan ennemminkin tiedon rakennetta ilman ennalta määrättyjä koodeja. XML sisältää varsinaisen tiedon sekä tiedon tiedosta, esimerkiksi tiedon nimen, sen ominaisuudet ja tietotyypin. Tieto koostuu tekstimuotoisesta informaatiosta, joka on jonkin periaatteen mukaan järjestetty. (XML – opas 2007.)

2.4.2 XML – kielen ulkoasu

XML:n tarkoitus on elementtien ja niiden ominaisuuksien määritteleminen, joilla kuvataan itse tietoa ja sen rakennetta. XML – dokumentin elementti koostuu aloitus- ja lopetusmerkistä. Mikäli elementillä ei sisältöä ole voidaan aloitus- ja lopetuselementti ilmaista lyhennetyssä muodossaan (KUVIO 8). (XML – opas 2007.)

```
1  <element></element>
2  <element/>
```

KUVIO 8. XML – elementtien eri esitystavat

W3C on määrittänyt XML:lle erikseen hyvin muodostetut (well-formed) ja lailliset (valid) dokumentit:

- Dokumentti on hyvin muodostettu, jos se on XML-määrittelyn ja kieliopin mukainen, vaikka siihen ei liittyisi lainkaan DTD:tä.
- Dokumentti on laillinen, jos se sisältää XML:n mukaisen dokumentin tyyppiesittelyn (XML Document Type Declaration, DTD).

Suurin osa internetissä olevista dokumenteista ovat hyvin muodostettuja, mutta ne eivät ole laillisia. (Kuronen 2007.)

Hyvin muodostettu XML – dokumentti koostuu ylimmällä tasolla ainoastaan yhdestä elementistä, jonka sisällä on seuraavia elementtitasoja. Elementtejä voi olla rajattomasti sisäkkäin. XML – dokumentille asetettuja esivaatimuksia, joita pitää noudattaa:

- Dokumentilla on vain yksi ylin elementti ns. juurielementti.
- Elementtien on noudatettava XML – tiedolle asetettuja hierarkiasääntöjä, jossa esim. kaikki elementit on suljettava elementin lopettavalla merkillä.
- Elementtien ominaisuuksien arvot merkitään lainausmerkeillä.
- Elementtien on nimellisesti noudatettava XML – määrittelyä
    - Elementin nimi alkaa aina kirjaimella
    - Elementin nimessä saa olla kaksoispistettä tai XML – alkuliitettä.
- Elementtien tekstimuotoinen sisältö ei saa sisältää joitain erikoismerkkejä
    - (<, >, ”, ’ ja &) - merkit täytyy korvata niitä vastaavilla entiteettimerkinnöillä (< > yms).
- XML:ssä on väliä kirjainkoolla eli kieli on ”case sensitive”.
- XML säilöö tyhjät merkit (white-space), ellei PI – komennossa toisin määrätä).

XML – tiedosto on tekstimuotoinen, parsittavissa oleva tiedosto, jonka pääte on ”.xml”. Itse dokumentti voi sisältää prosessointikäskyjä, kommentteja, elementtejä ja elementteihin liittyviä ominaisuuksia. Elementit koostuvat kolmesta tiedosta: nimi, ominaisuudet (attribuutit) ja itse tietosisältö. (XML – opas 2007.)

2.4.3 DTD ja Skeemat

DTD (Document Type Definition) eli dokumenttityyppimääritys on tapa kuvata rakenteellisten dokumenttien syntaksia. DTD – teknologia on kehitetty ennen XML – teknologiaa SGML – kielen ja EDI – ratkaisujen tarpeisiin. Siitä syystä se soveltuu XML:n tarpeisiin hyvinkin rajallisesti (XML – opas). DTD määrittelee dokumentin sallittujen elementtien nimet, elementtien suhteet ja elementtien attribuutit. DTD - määrittelyn avulla tarkistetaan XML – dokumenttien oikeellisuus. (XML – opas 2007.)

Rakenteellisen dokumentin yksi tärkeimmistä ominaisuuksista on sen mahdollisuus pakottaa looginen rakenne tietyn mallin mukaiseksi. Dokumentin laatiminen on helppoa, kun on mahdollista määrätä elementit ja niiden esittämisjärjestystä. Kun dokumenttiin liitetään DTD, kirjoitetaan dokumentti tietyn mukaisesti ja näin säilytetään itse rakenne. DTD:tä hyödynnetään erityisesti julkaisutoiminnassa, tekstidokumenttien rakenteellisessa määrityksessä. DTD voi olla sisäinen tai ulkoinen. Sisäinen DTD sijaitsee XML-dokumentissa hakasulkujen sisällä ja ulkoinen omassa tiedostossaan XML-dokumentin ulkopuolella (KUVIO 9). (Tuikka & Kanala 2001.)

```
1  <!DOCTYPE dokumenttityyppi […]>
```

KUVIO 9. Sisäisen DTD:n määritys

Koska DTD:n ominaisuudet eivät ole täysin riittävät kaikkia XML:n käyttöalueita varten (esim. tietokanta siirrot, joissa vaaditaan tarkkaa tiedon määrittelyä) on täytynyt DTD:n sijasta kehittää käytettäväksi ns. skeemakieliä. XML-skeemojen avulla pystytään laatimaan tarkempia rakennemäärityksiä, ja ne voivat sisältää myös muutakin hyödyllistä tietoa dokumentista. XML-skeemat ovat myös XML-dokumentteja, niiden tiedostopääte on ”.xsd” (xml schema description). Yksi näistä

skeemoista on W3C:n kehittämä skeemakieli nimeltä XML Schema, joka on kirjoitettu XML:llä. (Tuikka & Kanala 2001.)

2.5 DOM – malli

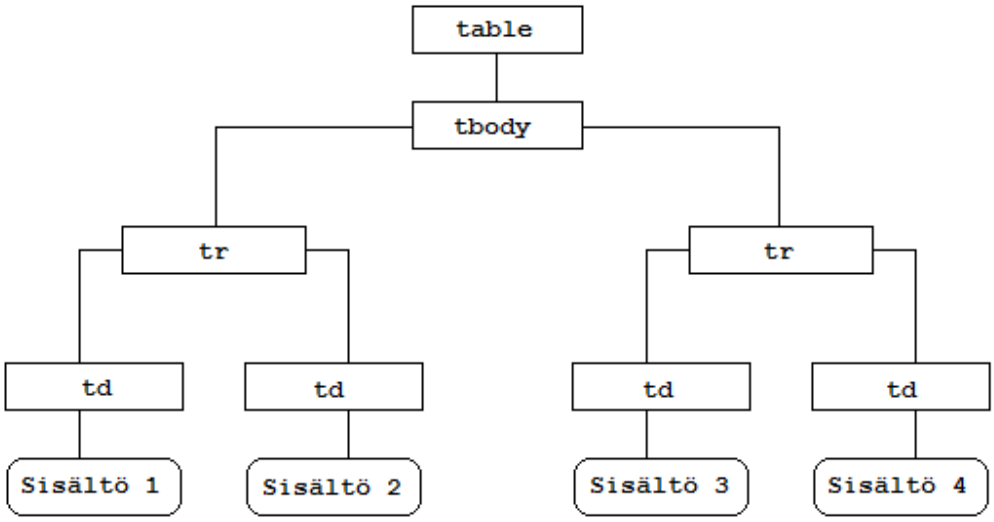
2.5.1 Yleistä

DOM (Document Object Model) – esitysmallin avulla kuvataan rakenteellisen asiakirjan rakennemallia. DOM on nimensä mukaisesti oliomalli. DOM käsittelee dokumentin osia erilaisia ominaisuuksia omaavina olioina, ja se sisältää metodit näiden olioiden muokkaamiseen. DOM myös rajapinta XHTML- ja XML-dokumenttien käsittelyyn. DOM – mallin näkökulmasta katseltuna jokainen XHTML- tai XML – asiakirja on hierarkkinen dokumenttipuu, joka koostuu rajoittamattomasta määrästä solmukohtia (nodes). Jokainen solmukohta kuvaa yhtä elementtiä dokumentin hierarkiassa. Ylimmällä tasolla dokumenttipuussa on dokumentin juurisolmu (root node). Jokaisella solmukohdalla voi olla lapsisolmuja (child node), jotka ovat kyseisen solmukohdan sisällä olevia elementtejä, ominaisuuksia ja tekstiä. Jokaisella lapsisolmulla voi olla omien lapsisolmujen lisäksi hierarkian yläpuolella ns. äitisolmu (parent node), ainoa poikkeus on ylimmällä tasolla oleva solmu, jolla ei ole äitisolmua. Dokumenttipuun rakenne muistuttaa oikean puun rakennetta (KUVIOT 10 ja 11):

- Puulla on juuret ↔ dokumenttipuulla on juurisolmu (root node).
- Puussa on oksia ↔ dokumenttipuussa on lapsisolmuja (child node).
- Puun oksilla on lehtiä ↔ dokumenttipuun lapsisolmuilla on uusia lapsisolmuja.

```
<table>
    <tbody>
        <tr>
            <td> Sisältö 1 </td>
            <td> Sisältö 2 </td>
        </tr>
        <tr>
            <td> Sisältö 3 </td>
            <td> Sisältö 4 </td>
        </tr>
    </tbody>
</table>
```

KUVIO 10. Dokumenttipuun rakenne



KUVIO 11. Yksinkertainen dokumenttipuu XHTML taulukkorakenteesta

Ohjelmistokehitysmielessä DOM – mallinen dokumentti on tehokas. Jokainen solmukohta on oma erillinen olio, jonka sisällä on joukko ominaisuuksia ja menetelmiä solmukohdan käsittelyyn. DOM – malli mahdollistaa vapaan liikkumisen edestakaisin XML – muotoisessa datassa. (XML – opas 2007.)

DOM – malli tarjoaa suuren joukon käyttökelpoisia ominaisuuksia, joiden avulla dokumenttipuuta voidaan hallita. Taulukko 1 listaa DOM – elementtien

käyttökelpoisimmat ominaisuudet ja taulukko 2 listaa DOM – elementtien hyödyllisimmät metodit.

TAULUKKO 1. XML-dokumenttien yleisimpiä DOM – elementtien ominaisuuksia (Asleson & Scutta 2006)

Ominaisuus	**Kuvaus**
childNodes	Palauttaa nykyisen elementin lapsisolmut taulukkona
firstChild	Palauttaa ensimmäisen lapsisolmun
lastChild	Palauttaa viimeisen lapsisolmun
nextSibling	Palauttaa seuraavan solmun (sisaren)
nodeValue	Ilmoittaa solmun luku- ja kirjoitusoikeuksien arvon
parentNode	Palauttaa äitisolmun
previousSibling	Palauttaa edellisen solmun (sisaren)

Taulukko 2. DOM – elementtien metodeja XML-dokumenttien läpikäymiseen (Asleson & Scutta 2006)

Metodi	**Kuvaus**
getElementById(id)	Palauttaa dokumentista löytyvän elementin id:n perusteella
getElementsByTagName(nimi)	Palauttaa taulukkona kaikki lapsisolmut jolla on tietty nimi
hasChildNodes()	Palauttaa totuusarvon lapsisolmujen olemassaoloon
getAttribute(nimi)	Palauttaa attribuutin arvon

Mikäli sovelluksen halutaan käsittelevän XML-tietoa, täytyy sovelluksella olla XML:ää ymmärtävä rajapinta. XML-dokumentti on yhteydessä rajapintaan, jolloin se

näkee vain rajapinnan eikä sovellusta. Samoin sovellus näkee vain pelkästään itse rajapinnan ja näin välittää kaiken tiedon rajapinnan kautta. Fyysisesti rajapinta on luokka, joka määrittää metodeja, joiden avulla sovellusohjelma pääsee käsiksi XML-dokumentin sisältämään tietoon. Rajapintoja hyödyntävät käytännössä ohjelmoijat. Tärkeimpiä XML-rajapintoja ovat DOM ja SAX (Simple Api for XML). (Tuikka & Kanala 2001.)

JavaScriptiä käytetään DOM – mallisen sisällön esittämiseen ja muokkaamiseen, ilman DOM – mallia JavaScriptiä ei käytettäisi juuri lainkaan. Sen tärkein ominaisuus on se, että jokaisen elementin sisältö voidaan muuttaa ja koska jokaiseen elementtiin päästään käsiksi, voidaan elementtejä myös lisätä tai poistaa (Peltomäki 2000). Ajaxin asynkronisen liikenteen mahdollistaa DOM – mallin ja JavaScriptin yhteiskäyttö. Nämä kaksi tekniikkaa mahdollistavat XML – kielen voimakkaan ja joustavan käytön selaimen ja palvelimen välisessä liikenteessä.

2.6 JavaScript

2.6.1 Historia lyhyesti

JavaScript – kielen kehitti Netscape vuonna 1995. Kielen versio 1.0 tuli Netscapen julkaiseman Navigator 2 – selaimen, mukana ja se mahdollisti ensimmäistä kertaa interaktiivisten WWW – dokumenttien tuottamisen. Alun perin kieltä kutsuttiin nimellä LiveScript, mutta markkinointisyistä kielen nimi vaihdettiin JavaScript:ksi. JavaScript kielestä kehittyi nopeasti ohjelmointikieli, jolla voitiin toteuttaa nopeasti latautuvia WWW – dokumenttiin upotettavia ohjelmia. JavaScriptin tulkki upotettiin suoraan selaimeen. (Peltomäki 2000.)

JavaScript on puhtaasti tulkattava kieli, joka sisältää alkeellisia oliosuuntautuneita ominaisuuksia. JavaScriptin oliot kuitenkin ovat enemmänkin *perl:*in assosiatiivisten taulukoiden kaltaisia kuin C:n struktuurien tai C++:n olioita. JavaScript on nimensä

mukaisesti skriptikieli, sen tunnetuin käyttökohde on selaimessa. Selaimessa JavaScriptin avulla dokumentin käyttäjä kokee välittömän vuorovaikutuksen sivuston kanssa. Kieli pystytään sulauttamaan myös skriptikieleksi muihinkin ympäristöihin, kuten web-palvelimiin (Netscape Enterprice Server ja Microsoft IIS). Microsoft kehitti oman versionsa JavaScriptistä nimeltä JScript, jonka se on pyrkinyt pitämään mahdollisimman pitkään yhteensopivana JavaScriptin kanssa. Molempiin kieliin on kuitenkin hiljalleen kehittynyt uusia ominaisuuksia. Lauseopillisesti JavaScript muistuttaa C-, C++ - ja Java-kieliä sisältäen rakenteita kuten *if*-lauseet, *while*-silmukat ja *&&*-operaattorin. JavaScript on ei-tyypitetty kieli (muuttujille ei tarvitse määrittää niiden tyyppiä). (Flanagan 1997;Peltomäki 2000.)

2.6.2 EcmaScript

JavaScriptin standardoitu versio EcmaScript on suunniteltu JavaScript- ja JScript – kielten pohjalle. Se on suunniteltu sovellusriippumattomaksi skriptikieleksi, jonka ansiosta se sopii minkä tahansa sovelluksen ohjelmointiin. EcmaScript - standardi määrittelee ainoastaan kielelle rungon, joten sovelluskohtaiset asiat täytyy hoitaa jollakin sovelluskohtaisella oliomallilla. WWW-dokumenteissa käytettävä DOM – malli on yksi esimerkki EcmaScriptiin sovitetusta oliomallista. EcmaScript on toisin sanoen standardoitu versio JavaScriptistä, jota suurin osa tulkkien toteuttajista noudattaa. (Peltomäki 2000.)

2.6.3 Peruskäyttö

JavaScript – kieltä voi kirjoittaa millä tahansa teksti- tai HTML – editorilla. Ohjelmakoodi kirjoitetaan *SCRIPT* – elementin sisään, joka normaalisti sijaitsee WWW – dokumentin alussa dokumentin otsikkoelementin sisällä. Ohjelmakoodi voidaan kirjoittaa myös runkoelementtiin. (Peltomäki 2000.)

Laajemmissa JavaScriptiä käytettävissä projekteissa JavaScript - ohjelmat kannattaa pitää erillisissä tiedostoissa ja liittää kyseinen JavaScript – koodi dokumenttiin *SCRIPT* – elementin SRC – parametria hyväksikäyttäen (KUVIO 12). (Peltomäki 2000.)

```
1  <SCRIPT LANGUAGE="JavaScript" SRC="URL-osoite">
2  </SCRIPT>
```

KUVIO 12. *SCRIPT* – elementin SRC – parametrin määrittely

Jotta SRC – parametri toimisi oikein, ulkopuolisten tiedostojen pääte on oltava ”.js” – muotoisia.

JavaScript – kielessä on kahdenlaisia kommentteja: monirivisiä ja yksirivisiä (KUVIO 13). Moniriviset kommentit nimensä mukaisesti kommentoivat usean kerrallaan, kun taas yksiriviset vain yhden rivin. Monirivisellä kommentilla on aina aloitus- ja lopetustagit, yksirivisillä on vain aloitustagi ja kommentointi päättyy aina rivin loppuun.

```
1  /*  Monirivinen kommentti
2      Kirjoitetaan näin     */
3
4  var x = 1; // Yksirivinen kommentti
```

KUVIO 13. JavaScript kommenttien kirjoitus

2.6.4 JavaScript olio-ohjelmointikielenä

Olio-ohjelmointi on yksi nykyaikaisen ohjelmistoteollisuuden kulmakivistä, mutta se ei sinällään ole ratkaisu ohjelmointiongelmiin. Olio-ohjelmointi helpottaa

kokonaisuuden suunnittelua, ohjelmistojen ylläpitoa ja ohjelmien jakamista loogisiksi kokonaisuuksiksi. JavaScriptissä voidaan olla käyttämättä tai käyttää olioita, suurissa kokonaisuuksissa olioiden käyttö on järkevää, mutta pienissä skriptin pätkissä niistä on vain työllisesti haittaa. JavaScriptissä olioiden käyttö perustuu hyvin pitkälle itse kielen sisäänrakennettuihin olioihin, mutta jokainen voi kirjoittaa myös omia olioitaan.

JavaScript - kielessä koko WWW – dokumentti käsitellään yhtenä suuren oliona, joka tunnetaan nimellä ”document”. Tämän document – olion sisällä on muita olioita kuten HTML-lomake, document – oliolla on myös ominaisuuksia, kuten taustaväri (BGCOLOR) (KUVIO 14).

```
1  <BODY BGCOLOR="RED">          // HTML
2  document.bgColor = "red";     // JavaScript
```

KUVIO 14. Taustavärin asetus HTML:llä ja JavaScriptillä

”JavaScriptissä taustaväriä voidaan muuttaa sijoittamalla document-olion bgColor ominaisuudelle uuden väriarvon. Jos halutaan viitata olion ominaisuuteen tai funktioon, se kirjoitetaan olion nimen perään pisteellä erotettuna.”

WWW – dokumenttiin tulostaminen on yksi useimmin käytetyistä toimenpiteistä JavaScript – ohjelmassa, joka tapahtuu write()-metodia hyväksikäyttämällä (KUVIO 15).

```
1  document.write("Tämä tulostuu dokumentille");
```

KUVIO 15. HTML – dokumenttiin tulostus

2.6.5 Drag and Drop

Useat Ajax:ia hyödyntävät sivustot käyttävät myös JavaScriptistä tuttua ns. *”Drag and Drop”*-tekniikkaa. Tämä tekniikka on sama kuin työpöytäsovelluksista tuttu raahaus (Drag)-menetelmä. Esimerkiksi, kun tiedostoa kopioidaan toiseen paikkaan, niin ikonista otetaan kiinni ja viedään tallennettavaan kansioon. ”*Drag and Drop”*-tekniikka itsessään on aika mielenkiintoinen. Tekniikka koostuu pääosin muutamasta tapahtumasta (event, tapahtumia tietyssä ajassa). Tapahtumia ovat mm. raahauksen aloitus, raahauksen jatkuminen ja lopuksi raahauksen päättäminen (Drop). Näiden tapahtumien lisäksi on myös toisten elementtien tapahtumien laukaisu elementin raahauksen aikana. Kun elementtiä raahataan jonkun toisen elementin yli, alla olevan elementillä olevan *onmouseover*-tapahtuma laukeaa ja siihen liittyvä ohjelmakoodi suoritetaan (Snook 2006). Tällä tekniikalla mahdollistetaan työpöytäsovelluksista tuttu interaktiivinen kanssakäyminen sovelluksen kanssa ja Web-sovelluksista saadaan enemmän työpöytäsovellusten kaltaisia. Google käyttää JavaScriptin *”Drag and Drop”*-tekniikkaa hyödyntävät mm. Google Maps ja iGoogle. Googlen Web-sovelluksessa iGoogle pystyy jokainen käyttäjä räätälöimään oman Google näkymän, joka aukeaa esim. aloitussivuksi ja joka sisältää mahdollisesti RSS-syötteitä. RSS-syötteitä pystyy liikuttamaan haluamaan kohtaan *”Drag and Drop”*-tekniikkaa hyväksi käyttäen.

3 AJAX-TEKNIIKKA

3.1 Asynkroninen JavaScript ja XML

3.1.1 Yleistä

Ajax on Web 2.0 – arkkitehtuurin yksi avainkomponenteista, ja se on nimi joukolle vanhoja webteknologioita. Ajaxin nimen (Asynchronous JavaScript and XML) keksi ja julkaisi Adaptive Path – nimisen yrityksen johtaja Jesse James Garrett verkkoartikkelissaan nimeltä ”Ajax A New Approach to Web Applications”. Artikkelissaan Garrett kertoo, kuinka web-sovellukset ja tavalliset työpöytäsovellukset ovat lähestymässä toisiaan. Hän siteerasi esimerkkeinä uusia tekniikoita ja useita Googlen projekteja, joissa tavallisia työpöytäsovelluksia käytettiin selainpohjaisesti. Merkittävin osa Ajaxin ympärillä pyörivästä kohusta johtui näin Googlesta, joka loi tekniikalle tunnettavuutta julkistaessaan Google Mapsin ja Google Suggestin. Garrett lausui artikkelissaan kaksi lausetta, jonka jälkeen Ajax – artikkeleita ja koodiesimerkkejä rupesi ilmestymään internetiin ja josta kaiken lisäksi itse Ajaxin nimi syntyi:

“Google Suggest and Google Maps are two examples of a new approach to web applications that we at Adaptive Path have been calling Ajax. The name is shorthand for Asynchronous JavaScript + XML, and it represents a fundamental shift in what’s possible on the Web (Garrett 2005).”

Garrett siteeraa artikkelissaan Googlen sovelluksia kuten Google Suggest ja Google Maps, jotka kuvastavat esimerkkejä uudesta lähestymistavasta web-sovelluksiin, jota he kutsuvat Adaptive Path:ssa nimellä Ajax. Nimi on lyhenne sanoista Asynkroninen JavaScript + XML, ja Garrett kertoo, että se kuvastaa sitä kaikkea, mikä on mahdollista verkossa.

Vaikka Ajaxissa ei varsinaisesti ole mitään uutta, on sen tarjoama lähestymistapa merkittävä kehitysaskel internetin perinteisiin pyyntöjen ja vastausten varaan rakennettuihin toimintamalleihin. (Zakas, McPeak & Fawcett 2006, 2; Asleson & Scutta 2006.)

Garrett määrittelee artikkelissaan Ajaxin kokoelmaksi itsenäisiä teknologioita, jotka yhdistettynä ovat uudistetulla tavalla vielä tehokkaampia kuin yksinään. Garretin mukaan Ajax koostuu 5 osasta:

1. standardien mukainen esitystapa käyttäen XHTML- ja CSS-tekniikoita
2. sivuston dynaaminen päivitys ja interaktiivisuus DOM:n avulla
3. tiedon siirto ja muokkaaminen XML ja XSLT-tekniikoilla
4. asynkroninen tiedon vastaanotto XMLHttpRequest-olion avulla
5. kaikkien sitominen yhteen käyttäen JavaScriptiä (Garrett 2005).

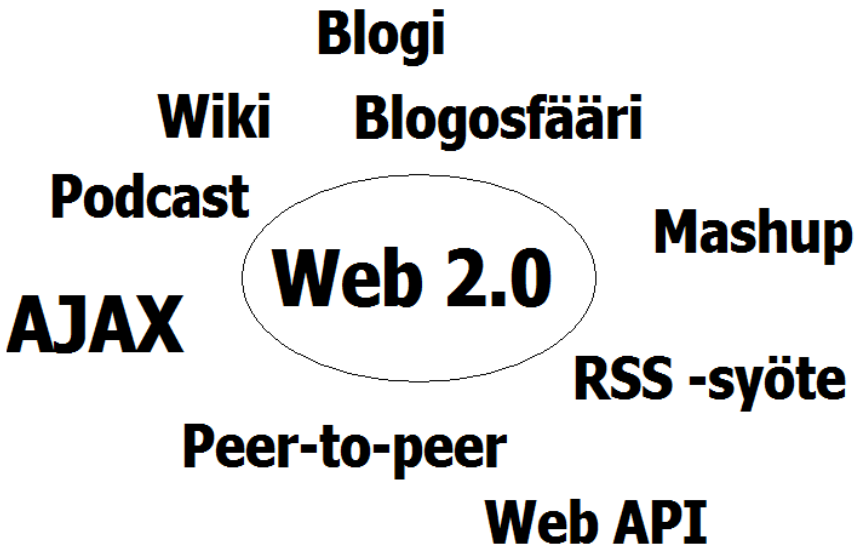
3.1.2 Web 2.0

Termi Web 2.0 viittaa siihen, mitä pidetään Web:n toisena vaiheena. Web 2.0 ei kuitenkaan ole yksiselitteisesti määriteltävä malli tai tekniikka. Kyseessä on lähinnä konsepti joukosta uusia ja hyväksi havaittuja toimintatapoja WWW-ympäristössä. Web 2.0 on ensisijaisesti ns. konsepti tai vanhan Web-ajattelutavan muutos, joka kattaa muun muassa uudet ja tuottavat WWW-liiketoimintamallit, pc-ohjelmien siirtymisen WWW-palveluiksi, käyttäjälähtöisen sisältötuotannon ja palvelukehityksen sekä ulkopuolisen maksuttoman datan jalostamisen WWW-tuotannossa. (Hintikka 2006-2007.)

Web 2.0 on siis kooste uusia ideoita, vanhoja tekniikoita, ilmaantuneita ominaisuuksia ja havaittuja piirteitä, joilla ei kaikilla ole edes mitään tekemistä toistensa kanssa. Osa tekniikoista ja piirteistä keksittiin jo yli kymmenen vuotta sitten, mutta vasta nyt niitä voidaan hyödyntää laajakaistayhteyksien ja riittävän

WWW-tallennustilan myötä. Jos Web 2.0:a halutaan määritettävän, niin internet-ansaintalogiikoiden pioneeri John Hagel on määrittänyt sen sanoilla ”hajautettua ja yhteisöllistä käyttäjien tuottamaa sisältöä verkkokeskeisellä alustalla”. (Hintikka 2006-2007.)

Web 2.0 keskeisimpiin tekniikoihin kuuluvat mm. blogit, podcastit, RSS-syötteet, yhteisöllisyyttä tukevat WWW-sovellukset kuten Wikipedia ja Flickr, WWW-pohjaiset ohjelmointirajapinnat ja tietenkin kaikista keskeisimmäksi käsitteeksi väitetty tekniikka on Ajax.



KUVIO 16. Web 2.0:n keskeisimpiä käsitteitä

Keskustelevia verkkoyhteisöjä on ollut 1970-luvulta lähtien. Web 2.0:n näkökulma on kuitenkin uusi. Yhteisölliset WWW-palvelut (Wikipedia ja Flickr) perustuvat käyttäjien itsensä tuottaman datan sekä sisällön jakamiseen ja järjestelemiseen

vapaamuotoisilla avainsanoilla sekä muita käyttäjiä kiinnostavan datan vaivattomaan löytämiseen.

Blogien toteutuksessa käytetään useita Web 2.0:an ideoita ja tekniikoita mikrotasolla. Blogit ovat periaatteessa kotisivuja päiväkirjamuodossa. Niistä on kuitenkin kehittynyt uudenlainen verkosto internetin sisälle. Ristiinlinkitysten ja rss-syötteiden avulla uutinen tai oletus esimerkiksi Googlen suunnittelemasta käyttöjärjestelmästä leviää tehokkaammin kuin koskaan aikaisemmin. Nykyään puhutaankin jo blogosfääristä eli blogien muodostamasta informaatioavaruudesta. Blogit suodattavat omalta kannaltaan kiinnostavimman datan paremmin kuin yksittäinen ihminen. Toisaalta blogit voivat myös olla tiedottajalle painajainen. Internetiin päässyt harkitsematon kommentti, tuotekuva tai vaikkapa virheellinen ohjelmisto leviää blogien kautta salamannopeasti. (Hintikka 2006.)

Web 2.0-konseptin suurin uutuus liittyy WWW-ohjelmoinnin käytäntöihin. Levytilan tarjonta, nopeutuneet kaistat sekä hybridiohjelmointi mahdollistavat yhä monipuolisempia palveluita. Kotikäyttäjille, pienyrityksille ja julkisyhteisöille maksuttomat WWW-palvelut ja niiden tarjoama ”rajoittamaton” tallennuskapasiteetti voivat lähivuosina olla potentiaalinen vaihtoehto. WWW-selaimetkin ovat jo sen verran stabiileja, että niiden varaan voidaan rakentaa monimutkaisia palveluita. Konsepti pyrkii harmonisoimaan WWW-selainta käyttöliittymäksi kaikille sovelluksille ja WWW:ta alustaksi yksittäisen päätelaitteen sijaan. Suuri osa uusista WWW-palveluista ei kuitenkaan ole läheskään yhtä monipuolisia kuin pc-ohjelmat, mutta niillä hoituvat perusasiat. Google ei tosin ole vieläkään saanut kaikkea toimimaan WWW-selaimessa, mutta se on julkaissut useita desktop-sovelluksia erikseen Windowsille, OS X:lle ja Linuxille. (Hintikka 2006.)

Web 2.0 on ensisijaisesti käyttäjälähtöistä suunnittelua. Näin ollen käyttäjät tulisi ottaa tasavertaisesti mukaan sisältöjen ja palveluiden kehittämiseen. Monestihan kuluttajat tuntevat tuotteet paremmin kuin itse markkinoijat, esimerkkinä kuluttajabrändit. Mikäli organisaatiota ja sen toimintaa edesauttavat muuttuvat

markkinatilanteet, käyttäjälähtöinen sisältö- tai palvelutuotanto, kehityssyklin nopeuttaminen tai evoluutiopohjainen liikeidea, kannattaa Web 2.0:aa seurata tarkasti. (Hintikka 2006.)

Web 2.0:n myötä WWW-alustan ohjelmointiin on vakiintumassa Ajax (Asynchronous JavaScript and XML). Tämä termi yhdistelee JavaScriptia, dynaamista HTML:ää ja CSS:ää XML:ään saaden aikaiseksi ohjelmistoarkkitehtuurin webtop-palveluille desktop-sovellusten tapaan. Monet Googlen WWW-palvelut toimivat ”teknologia-alustalla”, toisin sanoen Ajax-tekniikalla. Sekatekniikan avulla on mahdollista välttää muun muassa WWW-näkymän jatkuva uudelleenlataus. (Hintikka 2006.)

Web 2.0 ja Ajax ovat tällä hetkellä suosittuja, tämän kyllä huomaa siitä, että aiheita on käsitelty lähes jokaisessa ohjelmisto-alaa edes sivuavassa tapahtumassa. Kuten taannoin tapahtui San Frasciscossa pidetyssä Java-One-messuista, jossa verkkoteknologioiden suosio pystyttiin määrittelemään pelkästään luennoilla olevien ihmisten määrästä. Web 2.0 tai Ajax esityksissä jonot kasvoivat kadulle asti, siitäkin huolimatta, että messujen yhtenä pääaiheena oli Sunin mobiili-Javan julkistus. Näyttäisi siltä että Web 2.0 ja Ajax ovat tulleet jäädäkseen. (Poropudas 2007.)

3.1.3 Vahvuudet

Ajax-sovellukset reagoivat heti käyttäjän toimenpiteisiin eivätkä ärsytä käyttäjää turhilla ja häiritsevillä sivunlatauksilla. Tämä parantaa osittain Web-sovellusten käytettävyyttä. Tämän myötä myös palvelimen ja asiakkaan välinen liikennemäärä vähenee verrattaessa perinteisiin Web-sovelluksiin. Perinteinen WWW-malli on kuitenkin suunniteltu pelkästään hypertekstille eikä sovelluksille. Perinteinen malli voidaan kuitenkin muokata hieman enemmän sovellusmaiseksi käyttämällä käyttöliittymän ja palvelimen välissä Ajax-moottoria. (Paulson 2005; Garrett 2005.)

Ajax on myös sovelluskehittäjien suhteen vahvoilla, kun sen sisältämien tekniikoiden osaajia on entuudestaan paljon. Tekniikat, joista Ajax koostuu, ovat vanhoja, tuttuja ja kertaalleen jo opittuja. Lisäksi Ajax on alustariippumaton ja sen toimivuus lähes kaikissa selaimissa on kehittäjien mieleen. (Garrett 2005.)

3.1.4 Heikkoudet

Ajaxin heikkoutena on sen mahdollistama uudenlainen käyttöliittymä, koska se rikkoo perinteistä Web-selauksen mallia ja sivusto-ajattelua. Kun käyttäjä lisää sivuston selaimensa suosikkeihin, nykyinen sivu ei välttämättä mene sellaisenaan suosikkeihin johtuen Ajaxissa käytettävästä sivuston osien lataamisesta. Tämä myös estää sivuston osoitteesta suoraan linkin kopioimisen, koska näkyvä sivusto ei välttämättä olekaan juuri osoitteessa lukeva sivusto. Myös kaikenlainen sortuminen uusiin trendeihin, kuten ”Ajax on uusi ja kova juttu”, heikentää palveluiden käytettävyyttä. Ajaxia kuuluisi käyttää vain, jos tietää, miten sitä käytetään eikä vain sen vuoksi, koska se on nyt uusi ja hieno juttu (Ajaxia käytetään Ajaxin takia). (Nielsen 2005.)

Vaikka sovelluskehittäjien oppimiskynnys on verrattain matalalla, kun tekniikat ovat tuttuja ja kertaalleen opittuja. Siitä huolimatta haasteita syntyy myös sovelluskehittäjien piireissä. Pieniä Ajax-komponentteja on helppo lisätä yksinkertaisiin Web-sovelluksiin, mutta kun pienistä komponenteista siirrytään laajoihin Ajax-kokonaisuuksiin, niiden hallinta ilman tarkkaa suunnittelua onnistuu harvalta sovelluskehittäjältä. Ajaxin käyttöönottoa saattaa osaltaan myös hillitä sovelluskehittäjien pahat muistot DHTML-sovelluksista. Esimerkiksi JavaScriptillä on itsellään huono maine sovelluskehittäjien keskuudessa. JavaScriptiä on haukuttu yhdeksi maailman väärinymmäretyimmäksi ohjelmointikieleksi. JavaScriptin suurimmat ongelmat ovat selainten implementointierot ja muistivuodot. (Paulson 2005; Crockford 2001.)

3.2 XMLHttpRequest – objekti

3.2.1 Yleistä

Ajaxissa kaiken takana on XMLHttpRequest – objekti. Tämä Ajaxin selkärangan muodostava objekti on JavaScript-funktio, jota kutsutaan, kun palvelimelle täytyy lähettää pyyntö. Objekti tekee pyynnön palvelimelle asynkronisesti, jolloin selain ei pysähdy odottamaan vastausta palvelimelta, vaan jatkaa toimintaansa. XMLHttpRequest – objekti ei ole W3C:n hyväksymä standardi, vaan se on luonnosasteella. Alun perin XMLHttpRequest – objektin esitteli Microsoft, joka toteutti objektin Internet Explorer 5 selaimessa ActiveX – komponenttina. (Asleson & Scutta 2006.)

Ennen kuin XMLHttpRequest – objektia voidaan hyödyntää, on se ensin luotava JavaScriptilla, jotta selaimelle voidaan lähettää pyyntöjä ja käsitellä vastauksia. Objektin voi luoda kahdella eri tavalla:

- Internet Explorer toteuttaa XMLHttpRequest – objektin ActiveX – komponenttina.
- Muut selaimet toteuttavat XMLHttpRequest – objektin sisäisenä JavaScript – oliona.

Koska eri selaimilla toteutetaan objekti eri tavalla, molemmille selainryhmille joudutaan toteuttamaan oma logiikkansa. (Asleson & Scutta 2006.)

Selainryhmän tunnistaminen tapahtuu vertaamalla selaimen ActiveX tukea. Mikäli selain tukee ActiveX – komponenttia käytetään Internet Explorerille tarkoitettua toteutusta. Mikäli selain ei tue ActiveX – komponenttia, käytetään muille selaimille tarkoitettua toteutusta. Seuraava listaus havainnollistaa tätä selainryhmän tunnistusta. (Asleson & Scutta 2006.)

```
1  var xmlHttp;
2
3  function createXMLHttpRequest(){
4      if(window.ActiveXObject){
5          xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
6      }
7      else if(window.XMLHttpRequest){
8          xmlHttp = new XMLHttpRequest();
9      }
10 }
```

KUVIO 17. XMLHttpRequest – objektin luominen

Kuten kuviosta 17. huomataan, XMLHttpRequest – objektin luominen on melko yksinkertaista. Ensin luodaan globaali muuttuja xmlHttp ja laitetaan se osoittamaan varsinaiseen olioon. Funktio createXMLHttpRequest() suorittaa varsinaisen selaimentunnistus algoritmin. (Asleson & Scutta 2006.)

Turvallisuuden varmistamiseksi XMLHttpRequest – objektiin on asetettu tiettyjä ehtoja. XMLHttpRequest - objektin pyytämän resurssin on sijaittava saman verkkotunnuksen sisällä, josta koodin suoritus aloitettiin. Tämä estää XMLHttpRequest – objektia vastaamasta alkuperäisen verkkotunnuksen ulkopuolisiin pyyntöihin. (Asleson & Scutta 2006.)

3.2.2 Metodit

void open(string method, string url, bool async, string username, string password): Metodi avaa yhteyden palvelimeen, käytetään ainoastaan ns. ”avaamaan” (alustamaan) yhteys tulevaa palvelupyyntöä varten. Kaksi ensimmäistä parametria ovat pakollisia ja loput kolme ovat valinnaisia. (Asleson & Scutta 2006.)

- ”Method” on valittu lähetysmuoto (GET, POST tai PUT).
- ”URL” on yhteysosoite, johon pyyntö lähetetään.

- ”Async” arvolla määritellään käsitelläänkö kutsua asynkronisena (true, oletusarvo) vai synkronisena (false). Mikäli kutsua käsitellään synkronisena, se mitätöi koko Ajax – tekniikan tarkoituksen, tästä syystä oletusarvoisesti on aina asynkroninen käsittelykutsu.
- ”Username” – parametrilla pystytään käyttämään tiettyä käyttäjänimeä metodin suorituksen yhteydessä.
- ”Password” – parametrilla pystytään käyttämään tiettyä salasanaa metodin suorituksen yhteydessä.

void send(content):
Metodi lähettää pyynnön palvelimelle. Jos pyyntö on asynkroninen, päättyy metodin suoritus välittömästi ja ohjelman muu suoritus jatkuu. Jos pyyntö on synkroninen, ohjelman suoritus keskeytyy, kunnes palvelin palauttaa vastauksen. Metodilla on yksi valinnainen parametri (content), jolla voidaan lähettää palvelimelle esimerkiksi merkkijonotyyppistä tekstiä. (Asleson & Scutta 2006.)

void SetRequestHeader(string header, string value):
Metodi sijoittaa arvon määritellyn otsakkeen (header) HTTP-pyynnön arvolle. Metodia voidaan kutsua onnistuneesti vasta open()-metodin jälkeen.

void abort():
Metodi keskeyttää aiemmin tehdyn pyynnön suorituksen.

string getAllResponseHeaders():
Metodi palauttaa HTTP-pyynnöstä vastauksen otsakkeen (header) merkkijonona.

string getResponseHeader(string otsake):
Metodi palauttaa tietystä otsake-parametrista määritellyn otsakkeen arvon.

3.2.3 Ominaisuudet

XMLHttpRequest – objektilla on ominaisuuksia, joiden avulla se reagoi eri tapahtumiin ja tilanmuutoksiin. Esimerkiksi XMLHttpRequest – objektilla on *readyState*-ominaisuus, joka kuvaa palvelinpyynnön tilaa. Taulukossa 3. on listattu XMLHttpRequest – objektin ominaisuudet ja niiden arvot.

TAULUKKO 3. XMLHttpRequest – objektin ominaisuuksia

Nimi	**Selite**	**Arvo**
Onreadystatechange	Tapahtumankäsittelijä	Reagoi jokaiseen tilamuutokseen, yleensä kutsutaan jotain JavaScript-funktiota.
readyState	Pyynnön tila	Pyynnöllä on viisi mahdollista arvoa: 0 – alustamaton; 1 – lataamassa; 2 – ladattu; 3 – interaktiivinen; 4 – valmis.
responseText	Vastausmuoto	Vastaus palautetaan merkkijonona.
responseXML	Vastausmuoto	Vastaus palautetaan XML-muodossa, jolloin sitä voidaan käsitellä DOM-mallin mukaan.
status	Tilakoodi	Palvelin palauttaa tilakoodin esimerkiksi: 200 – OK tai 404 – Ei käytettävissä.
statusText	Tilakoodi merkkijonona	Palvelin palauttaa tilakoodin esimerkiksi: ”OK” tai ”Ei käytettävissä”.

3.2.4 Yksinkertainen kysely

Palvelimen ja selaimen välinen kommunikaatio hoidetaan Ajaxin XMLHttpRequest-objektin välityksellä seuraavalla tavalla:

1. Luodaan viittaus XMLHttpRequest – objektiin.
2. Määritellään XMLHttpRequest – objektille funktio, joka käsittelee muutoksia objektin tilasta. Asetetaan onreadystatechange-ominaisuus osoittamaan johonkin JavaScript – funktioon.
3. Asetetaan pyynnölle ominaisuudet open()-metodin parametrien avulla. Asetetaan lähetysmuoto (GET tai POST), kohderesurssin osoite (URL) ja async totuusarvo (TRUE tai FALSE).
4. Lähetetään pyyntö palvelimelle XMLHttpRequest – objektin send()-metodia hyväksikäyttäen.

```
1  var xmlHttp;
2  var url = "serverFile.php";
3  ...
4  xmlHttp = createXMLHttpRequest(); // 1.
5  xmlHttp.onreadystatechange = useThisFunction(); // 2.
6  xmlHttp.open("GET", url, true); // 3.
7  xmlHttp.send(null); // 4.
8  ...
```

KUVIO 18. Yksinkertainen kyselyesimerkki

Kuviossa 18. on esitetty pääosin mitä JavaScript ohjelmakoodin tulee sisältää, jotta selaimen ja palvelimen välinen kommunikaatio olisi mahdollista.

3.3 Kommunikointi palvelimen kanssa

3.3.1 Vastausten käsittely

XMLHttpRequest – objektilla oli kaksi ominaisuutta, joiden avulla pääsee käsiksi palvelimelta saapuneeseen vastaukseen. Ensimmäinen on responseText, joka palauttaa vastauksen tekstimuotoisena. Toinen on responseXML, joka palauttaa vastauksen XML – objektina. Palvelimelta saatava tekstimuotoinen vastaus ei ole kovin joustava ja se soveltuu lähinnä yksinkertaisiin käyttötapauksiin. Tekstillä kun ei ole minkäänlaista rakennetta, joten myös dynaamisen sisällön luominen voi olla melko vaikeaa. Tekstimuotoinen vastaus on kyllä kätevä, jos sitä käytetään XHTML – elementtien innerHTML – ominaisuuden kanssa. Yhdistämällä innerHTML ja responseText, voidaan palvelin laittaa tuottamaan XHTML – sisältöä, johon päästään käsiksi selaimelta kätevästi. Monimuotoiset tietorakenteet on kuitenkin parempi lähettää XML – muodossa eli XML – objektina. (Asleson & Scutta 2006.)

3.4 Ajax sovelluksia

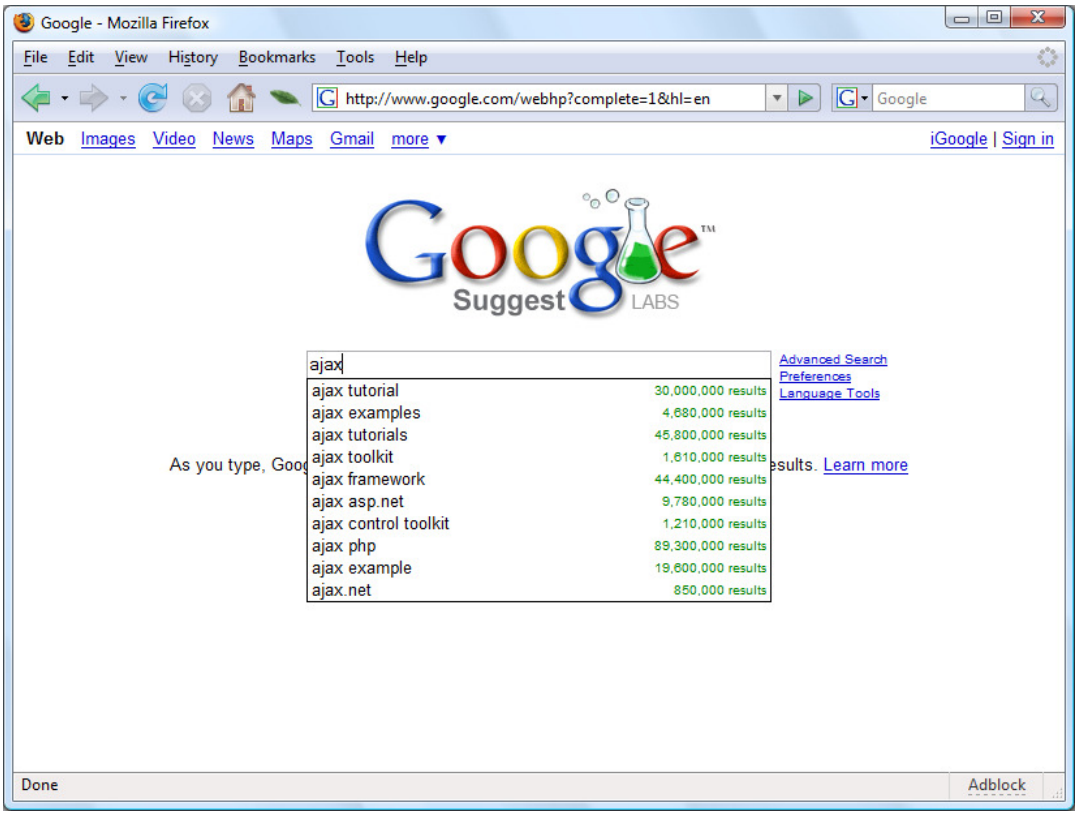
3.4.1 Yleistä

Ajax toimii useimmissa nykyaikaisissa selaimissa ilman minkäänlaisia plug-in-ohjelmia. Ajaxin vahvuus onkin sen syntyminen jo olemassa olevien tekniikoiden varaan, jolloin kehittäjien ei tarvitse opetella vaikeita ja uusia ohjelmointikieliä tai muokata jo valmiita palvelimilla olevia ratkaisuja. Ajax mahdollistaa täysin uudenlaiset websovellukset, jotka vastaavat enemmän nykyaikaisia työpöytäsovelluksia kuin websivuja. Ajax-sovelluksista puhuttaessa ei voidakaan enää välttämättä puhua WWW-sivusta vaan ennemminkin www-sovelluksesta. Ajaxin käyttökohteina ovat esimerkiksi automaattisesti päivittyvät sivut, työkaluvihjeet, automaattisesti täydentyvät lomakkeet, edistykselliset käyttöliittymäkomponentit ja virheiden tarkistukset suoraan lennosta. Google on

Ajaxin merkittävin hyödyntäjä, ja se käyttää Ajaxia mm. Google Mapsissa, Google Suggestissa ja Gmailissa. (Aselsson & Schutta 2006; Paulson 2005.)

3.4.2 Google Suggest

Ensimmäisissä internetiin ilmestyneissä Ajax-artikkeleissa, suunnittelijat siteerasivat lukuisia Ajax-tekniikalla toteutettuja sovelluksia. Yksi näistä sovelluksista oli Google Suggest. Sovellus muistuttaa ulkonäöltään normaalia Googlen käyttöliittymää, jossa sivun keskiosassa olevaan tekstikenttään syötetään hakuparametrit ja hakunappia painamalla saadaan hakusanaa vastaavat tulokset. Google Suggest sovelluksessa erona on sen nimen mukaisesti, että kun hakua kirjoitetaan tekstikenttään, ilmestyy tekstikentän alapuolelle lista, joita Google Suggest ns. ”ehdottaa” hakuparametreiksi. Lista ehdotetuista hakuparametreista tulee palvelimelta, joka haetaan asynkronisesti kun käyttäjä syöttää kenttään alustavan hakuparametrin. Kuviosta 19 näkyy sovelluksen ns. ”ehdotuslista” hakuparametrille ajax. (Zakas ym. 2006.)

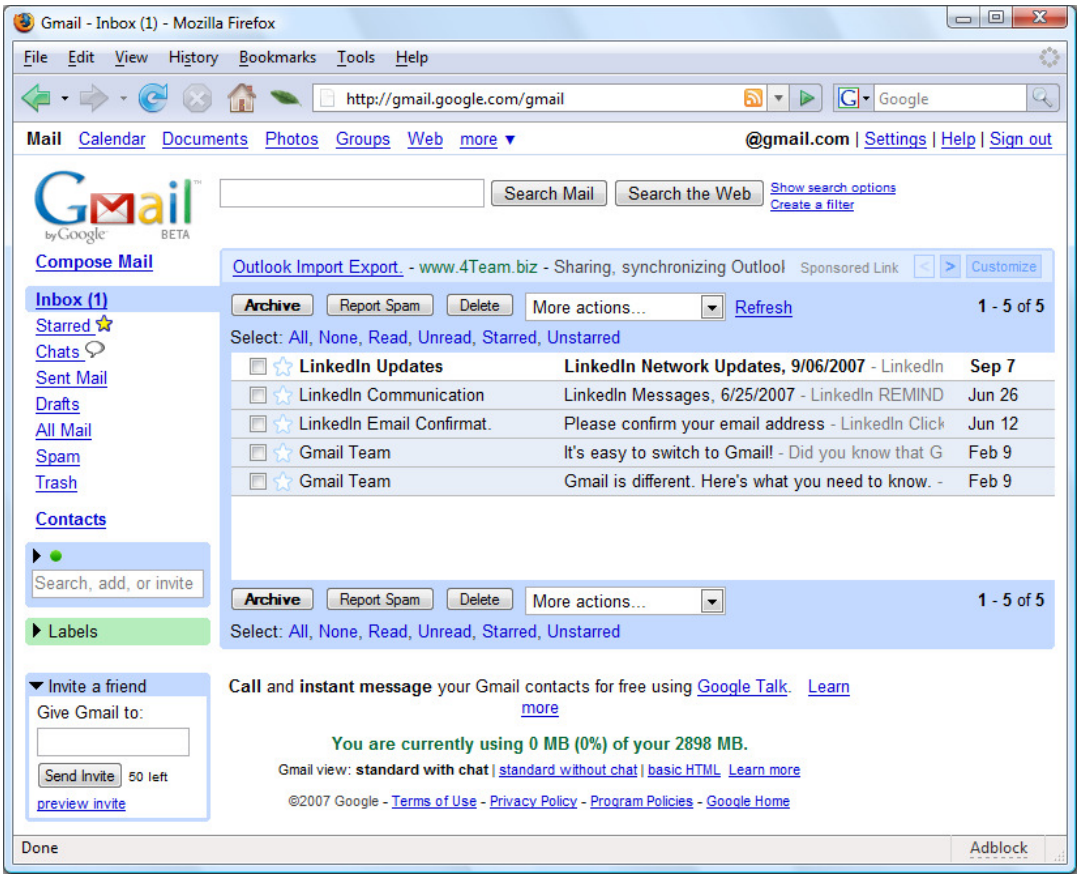


KUVIO 19. Google Suggest

3.4.3 Google Gmail

Gmail on Googlen ilmainen webmail-pohjainen sähköpostipalvelu (KUVIO 20). Palvelu julkaistiin 31. maaliskuuta 2004, mutta monet ihmiset saivat kuitenkin tietää palvelusta vasta 1. huhtikuuta ja luulivat sitä aprillipilaksi. Tämä johtui Gmailin lupaamasta ilmaisesta gigatavun tallennuskapasiteetista, joka oli silloin erittäin suuri verrattuna toisiin webmail-palveluihin. Gmail oli aluksi kokeiluasteella, jolloin uudet käyttäjät pystyivät rekisteröitymään palveluun vain toisten käyttäjien kutsumina. Alkuvaiheessa näitä kutsuja haluttiin jopa ostaa rahalla tai vaihtaa tavaraan.

Gmail on pyritty rakentamaan mahdollisimman samanlaiseksi kuin mikä tahansa työpöytä käytössä ajettava sähköpostisovellus, siinä esimerkiksi tarkastetaan oikeinkirjoitus viestiä kirjoittaessa, ja kirjoitettavat viestit tallentuvat automaattisesti.

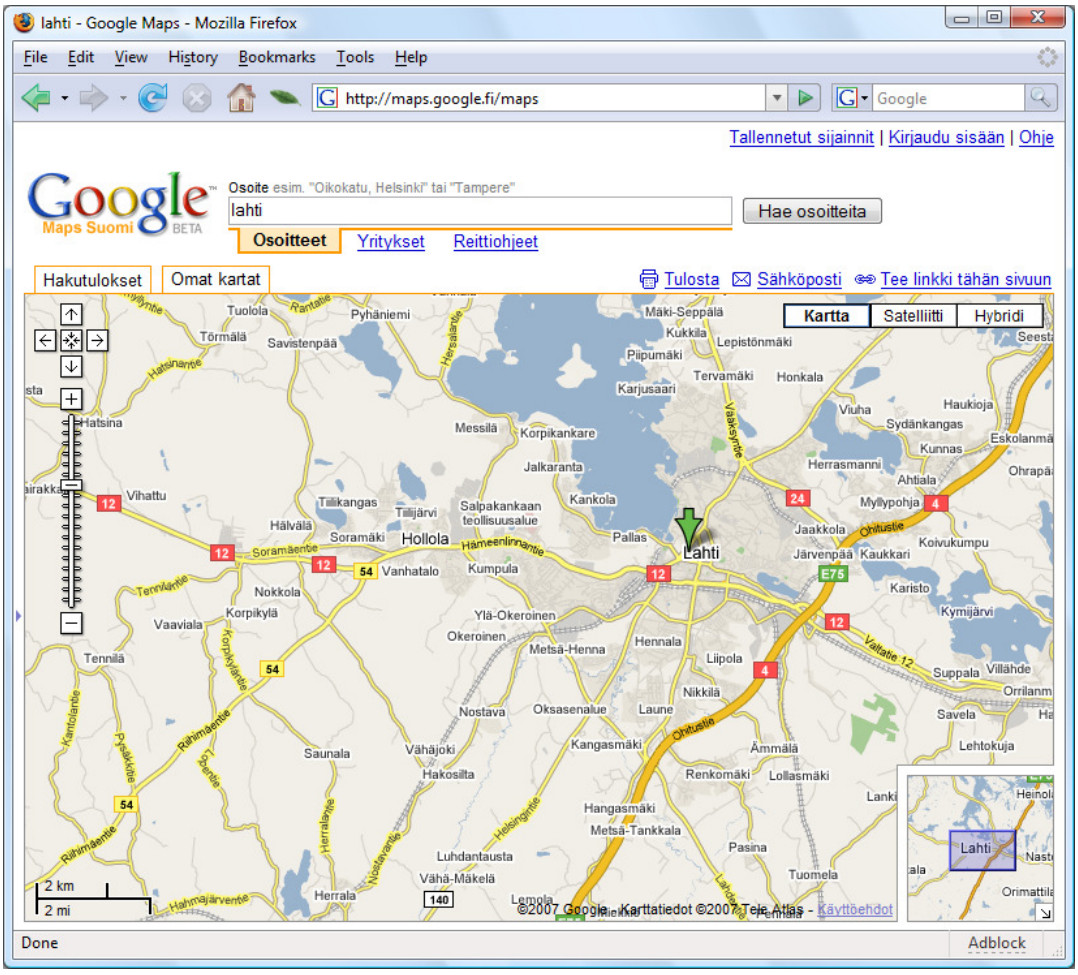


KUVIO 20. Google Gmail

3.4.4 Google Maps

Google Maps on Googlen tuottama karttapalvelu (KUVIO 21), aiemmin tunnettu myös nimellä Google Local. Sovelluksessa käytetään Ajaxia estämään pääsivun päivitys, tietojen siirto tapahtuu asynkronisesti palvelimen kanssa piilotetun *iframe*:n avulla. Google Mapsissa voi karttaa liikutella eri suuntiin suoraan hiiren

painalluksella, hiiren rullaa käyttämällä voidaan zoomata lähemmäs tai kauemmas. Sinällään ns. ”*drag and drop*” – menetelmässä ei ole mitään uutta JavaScript ohjelmoijille, mutta kartassa olevat kuvat ja niiden rinnakkaisuus ovat uutta. Kartan muodostavien kuvien määrä on rajallinen, joten samoja kuvia käytetään aina uudestaan ja uudestaan näyttämään eri kohtia kartasta. (Zakas ym. 2006.)



KUVIO 21. Google Maps

4 SUUNNITTELU-OHJAUS SOVELLUS

4.1 Tarkoitus ja spesifikaatio

Sovellus tehtiin tilaustyönä WTS Networks Oy – nimiselle yritykselle. Se huomasi tarpeensa sovellukselle, kun se sai tilauksen toiselta yritykseltä. Toisen yrityksen vastuulla oli rakennuttaa ja toteuttaa yleisrakennuksiin helposti käytettävä ylläpitojärjestelmä, jolla esimerkiksi talonmies pystyisi säätämään monen kerrostalon laitteita selainpohjaisesti. WTS Networks Oy otti siis vastuulleen käyttöjärjestelmän suunnittelun ja toteuttamisen, joka lopulta siirtyisi toisen yrityksen käytettäväksi. WTS Networks Oy antoi laajasti vapaat kädet sovelluksen suunnitteluun ja toteutukseen. Sovelluksen lopulliset viimeistelyt kuitenkin teki WTS Networks Oy itse.

WTS-Networks tarvitsi helppokäyttöisen sovelluksen, jolla suunnitella ja ohjata talon sisällä olevia komponentteja, kuten lämpöpattereita, valaistusta, ilmastointilaitteita yms. Projektin tarkoituksena oli siis web-pohjainen sivuston suunnittelu ja toteutus, jolla loppukäyttäjä pystyisi säätämään elektronisia laitteita asunnossaan suoraan selaimella.

4.2 Sovelluksen ominaisuuksia

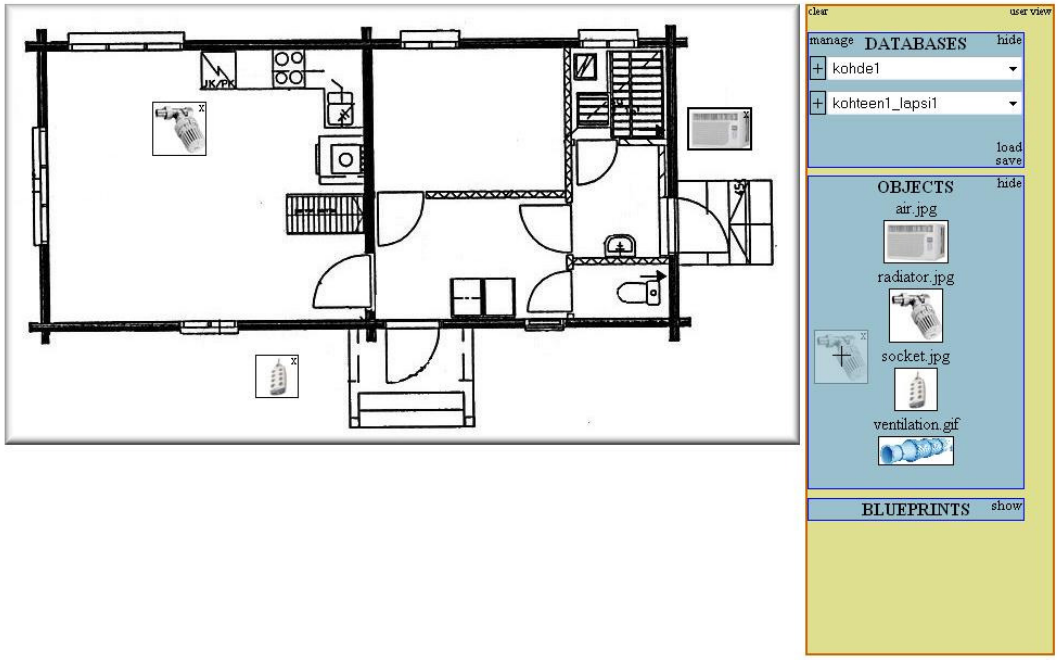
4.2.1 Yleistä

Sivustot suunniteltiin käytettäväksi kahdella eri käyttäjätasolla: normaalit käyttäjät ja järjestelmän ylläpitäjät. Normaaleilla käyttäjillä on oikeus tiettyyn pohjapiirustukseen ja siinä oleviin kontrolloitaviin objekteihin. Ylläpitäjät rakentavat pohjapiirustusten perusteella talon sähköiseen muotoon ja lisäävät pohjaan objekteja, jotka kuvastavat jotain tiettyä laitetta talossa. Laitteita voidaan näin kontrolloida käyttäjänäkymästä. Sovelluksessa on samantapainen ominaisuus kuin työpöytäsovelluksissa, joka

varmistaa käyttäjältä halutun tapahtuman tekemisen, tämän avulla estetään esimerkiksi vahingossa painettujen taulukoiden poistot. Kuviosta 24 nähdään ponnahdusikkuna, joka kysyy käyttäjältä varmistusta tietystä toimenpiteestä.

4.2.2 Ylläpitäjänäkymä

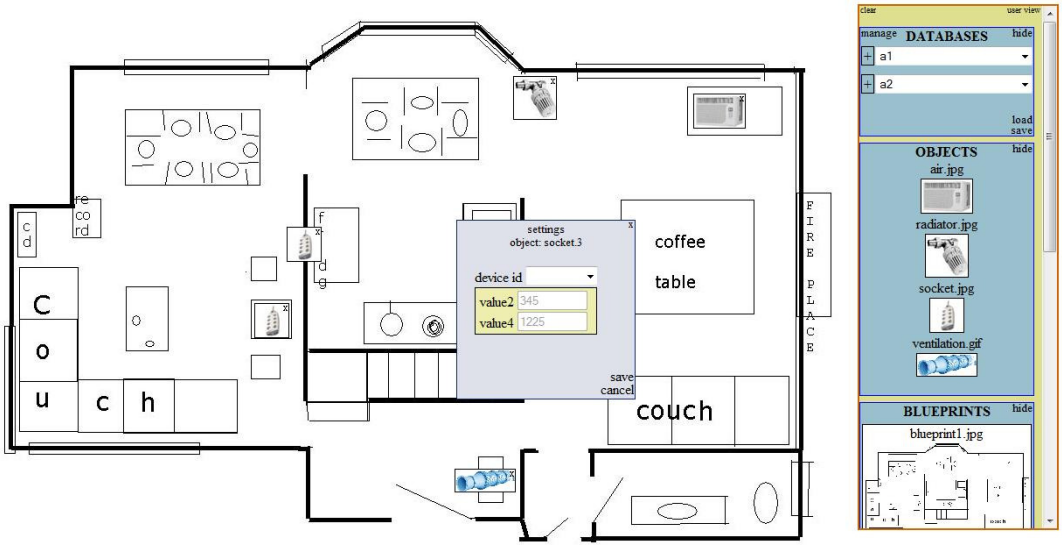
Ensin valitaan oikealla puolella olevasta valikosta *blueprints* (valikkoa avattaessa sovellus hakee tietystä hakemistosta kuvat, jossa pohjapiirustukset ovat ja näyttää ne menussa) ja valitaan tietty pohjapiirustus, johon halutaan liittää objekteja. Seuraavaksi valitaan halutut objektit valikosta ja ns. ”heitetään” pohjapiirustuksen päälle (KUVIO 22).



KUVIO 22. Objektien raahaus ylläpitäjänäkymässä

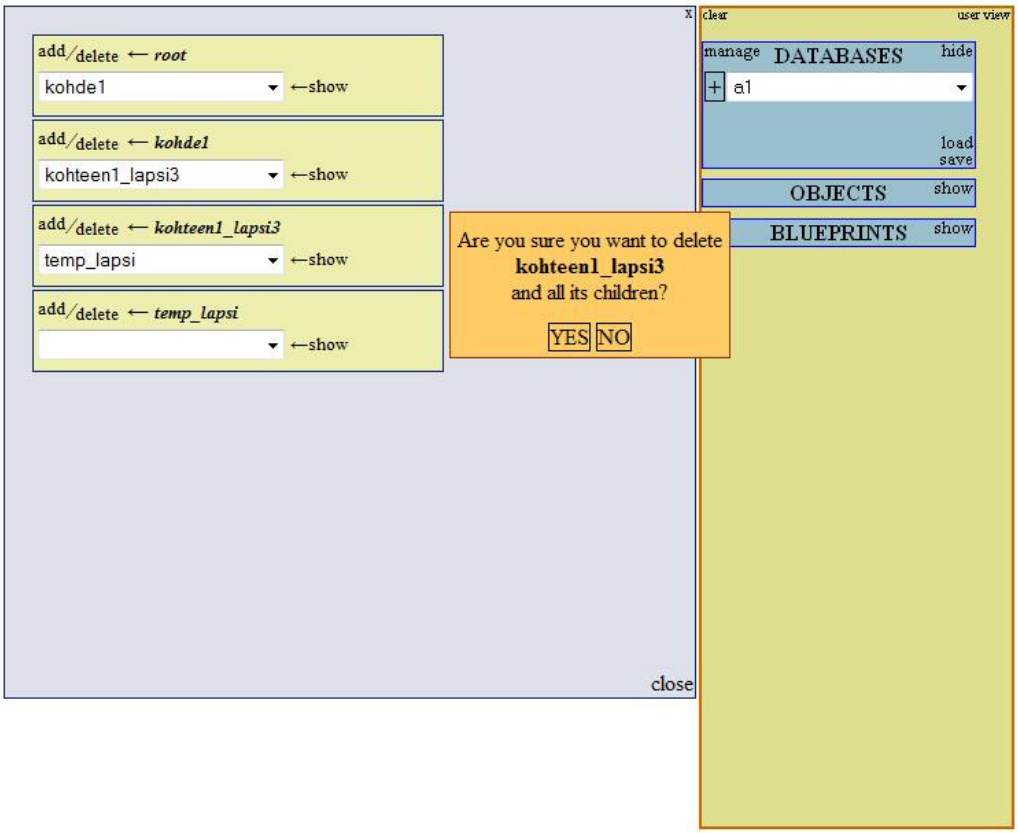
Kun objektit ovat siirretty pohjapiirustuksen päälle, niitä tuplaklikkaamalla avautuu objektin päälle uusi valikko, josta objekti linkitetään MySQL – tietokantaan

(kannassa on linkitykset oikeisiin laitteisiin). Lopuksi kun kaikkiin objekteihin on valittu oikeat linkitykset, valitaan valikosta *databases* – kohdasta kanta, johon halutaan tallentaa kyseinen näkymä (KUVIO 23).



KUVIO 23. Ylläpitäjänäkymä

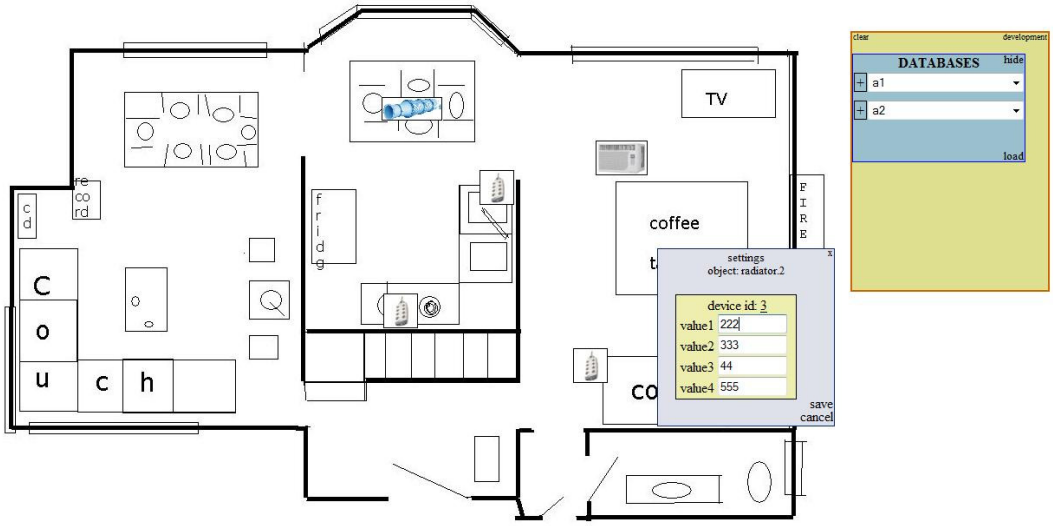
Ylläpitäjänäkymästä pystytään muuttamaan myös tietokannassa olevaa puuhierarkiaa, johon tallennetaan eri talojen ja tilojen pohjapiirustukset ja objektit. Esimerkiksi ensin tehdään kohteen nimeksi ”Messut Helsingissä 2005”. Sen jälkeen valitaan kerros kyseisestä messusta ja lopuksi mutta ei välttämättä viimeiseksi kyseisen kerroksen tietty osasto. Sovellus sallii myös sen, että puuhierarkiaa voidaan jatkaa loputtomiin.



KUVIO 24. Ylläpitäjän tietokannan puuhierarkia

4.2.3 Käyttäjänäkymä

Ylläpitäjänäkymällä tehty rakennus näkyy käyttäjänäkymässä (KUVIO 25) hieman erilaisena. Erona ylläpitäjänäkymään on hieman karsittu menuvalikoima, objektien avaus ei linkitä niitä kantaan, vaan avaa näkymän, jolla voidaan vaihtaa arvoja kyseisessä laitteessa (esimerkiksi patterin lämpötilaa muutetaan). Lisänä käyttäjänäkymässä on, ettei objekteja pysty siirtelemään uusiin paikkoihin, eikä tallennusvaiheessa tallenneta uusia objekteja vaan asetetaan valitut arvot tietokantaan.

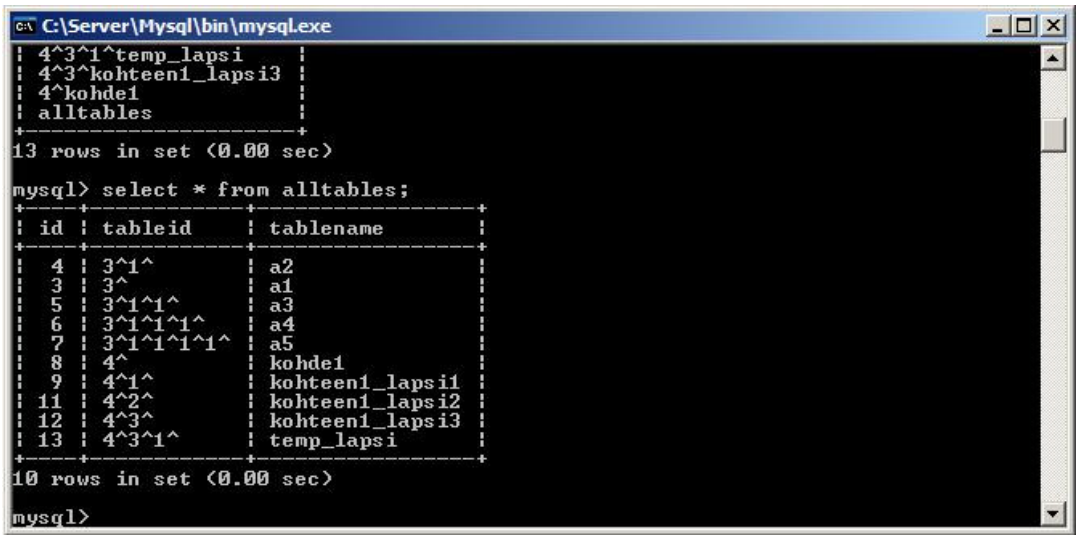


KUVIO 25. Käyttäjänäkymä

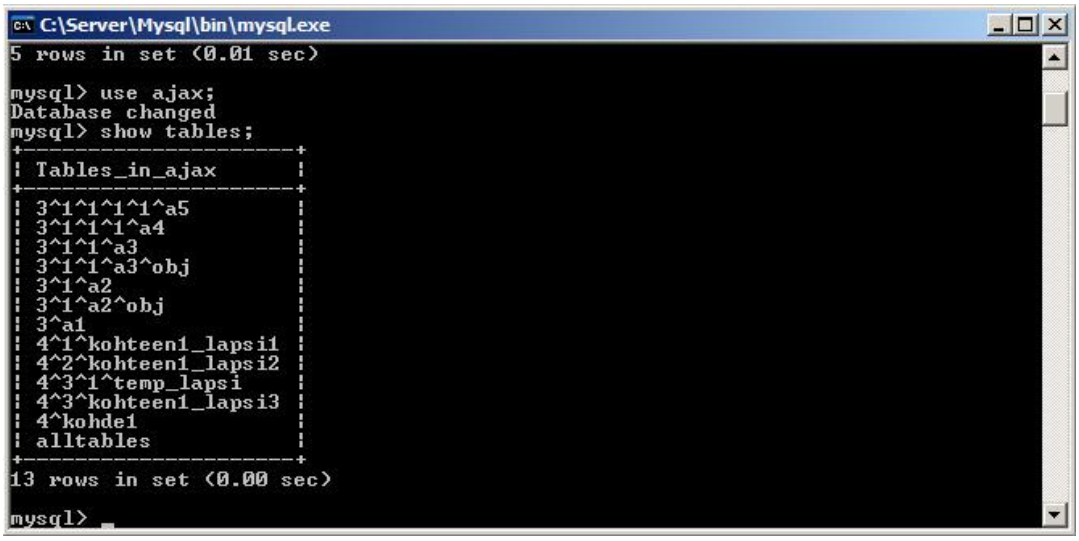
4.2.4 Tietovarastot

Tietovarastona sovelluksessa on käytetty MySQL – tietokantaa. Tietokantaan tallennetaan tiedot objekteista, niiden paikasta ruudulla, niiden linkeistä toiseen tietokantaan (laitetietokanta jossa on oikeiden laitteiden arvot) sekä pohjapiirustuksen nimestä kyseistä rakennuksesta.

MySQL – tietokantaan tallennettu puumainen hierarkia nähdään kuviosta 27. Esimerkiksi taulu ”4^1^kohteen1_lapsi1”, jossa ensimmäinen numero tarkoittaa ylimpänä olevan äidin numerotunnistetta, seuraavana ^ - merkin jälkeen tuleva numero on edellisen lapsi jne. Viimeisenä ^ - merkin jälkeen tulee kohteelle annettu nimi.



KUVIO 26. MySQL – tietokanta & kohteiden hierarkia



KUVIO 27. MySQL – tietokannan taulut

4.3 Rakenteellinen kuvaus

4.3.1 Tiedostolistaus

Sovelluksessa käytettiin useita PHP-tiedostoja, joiden ohjelmakoodia kutsuttiin tarvittaessa. Esimerkiksi, kun kohteita oltiin poistamassa puuhierarkiasta, kutsuttiin JavaScript ohjelmakoodilla ”Deletefromdb.php”-tiedostoa, joka poisti tietokannasta kyseisen taulun ja ilmoitti selaimelle pyynnön onnistumisesta. Taulukossa 4. on listattuna ohjelman kaikki tiedostot ja niiden tarkoitus.

TAULUKKO 4. Sovelluksen tiedostolistaus

Tiedosto	**Selite**
Index.php	Etusivu
Createchild.php	Puuhierarkian tallennus tietokantaan
Deletefromdb.php	Kohteiden poisto tietokannasta
Fetchtomenu.php	Tietojen haku alkuvalikkoon
Gettablephp.php	Tietojen haku tietokannasta objekteille, ponnahdusikkunoille jne.
Open.php	Tietokantayhteyden avaaminen
Savetochild.php	Objektien tietojen tallennus tietokantaan
Style.css	Dokumentin tyylimäärittelyt
Dragndrop.js	Tietojen siirrot ja itse sovelluskoodi

4.3.2 Sovelluksen rakenne

Sovelluksen tekemiseen käytettiin XHTML – kieltä, dokumenttien ulkoasu hoidettiin CSS - tyyliohjejärjestelmällä. Palvelinpuolella käytettiin PHP – kieltä kommunikoimaan tietokannan kanssa, joka oli MySQL – pohjainen. JavaScript

hoitaa kaiken toiminnallisuuden selaimessa, kuten objektien siirtäminen. Ajaxia on käytetty hoitamaan asynkroninen tiedonvälitys palvelimelle.

5 YHTEENVETO JA JOHTOPÄÄTÖKSET

Tässä opinnäytetyössä tutkittiin Ajax-tekniikkaa, sen koostavia tekniikoita käsiteltiin perustasolla. Ajax siis koostuu tekniikoista XHTML, CSS, DOM, XML ja JavaScript. Tekniikoita XHTML ja CSS käytetään esittämään sivustoa ja sen ulkoasua. DOM-mallin avulla päivitetään sivustoa dynaamisesti. XML-tekniikalla muokataan ja siirretään tietoa, joka vastaanotetaan XMLHttpRequest-olion avulla. Nämä kaikki tekniikat sidotaan toisiinsa käyttämällä JavaScript-skriptikieltä. Ajaxin käyttöä tulisi harkita tarkoin, sen käyttö kannattaa, jos sitä oikeasti tarvitsee sovelluksensa toimintaan eikä vain sen vuoksi että se on uusi ja hieno juttu.

Työn tarkastelussa Ajaxin lisäksi oli Web 2.0-arkkitehtuuri, jonka avainkomponentti on Ajax. Web 2.0-arkkitehtuuri on kooste uusista ideoista (Ajax), vanhoista tekniikoista (XHTML, CSS, DOM, XML, JavaScript), ilmaantuneista ominaisuuksista sekä havaituista piirteistä. Web 2.0:a voidaan vasta nyt hyödyntää tulleiden laajakaistayhteyksien ja riittävän WWW-tallennustilan myötä. Tarkastelun jälkeen voidaan todeta, että Web 2.0 ja Ajax ovat tulleet jäädäkseen.

Käytännöntoteutuksen tarkoituksena oli mahdollistaa rakennusten sisäisten laitekomponenttien kontrolloiminen Web-selaimella. Rakennuksessa olevia laitekomponentteja, kuten lämpöpattereita, valaistuksia tai ilmastointilaitteita voitaisiin siis ohjata suoraan Web-selaimesta esimerkiksi talonmiehen toimesta. Sovelluksen toimivuus testattiin toimivaksi Internet Explorer 7.0- ja Mozilla Firefox 2.0-selaimilla. Työtä hankaloitti selkeästi tarve saada sovellus toimimaan useimmilla selaimilla. Sivustoa testattiin herkeämättä samaan aikaan, kun sitä rakennettiin. Toteutus vastasi suurelta osin omia odotuksiani, ja sain kiitosta WTS Networks Oy:ltä hyvin tehdystä työstä. Sain myös myöhemmin kuulla, että sovellukseni on tälläkin hetkellä heillä käytössä. Sivustot yritettiin saada rakennettua niin, että niitä olisi mahdollisimman yksinkertainen ja helppo käyttää sekä ymmärtää, mikä mielestäni onnistui erittäin hyvin.

LÄHTEET

Asleson, R & Scutta, N. 2006. Ajax – Tehokas hallinta. Readme.fi, Helsinki.

Boumphrey, F., Greer, C., Raggett, D., Ragget, J. Schnitzenbaumer, S. & Wugofski T. 2001. XHTML Ohjelmoijan käsikirja. Oy Edita Ab, Helsinki.

Crockford, D. 2001. JavaScript: The World's Most Misunderstood Programming Language [online]. [viitattu 21.9.2007] Saatavissa: http://www.crockford.com/javascript/javascript.html

CSS – opas [online]. 2kmediat.com. [viitattu 9.7.2007]. Saatavissa: http://www.2kmediat.com/css/

Ek, J & Eriksson, U. 2001. XHTML-käsikirja, 1. painos. Schildts Kustannus Oy, Vantaa.

Flanagan, D. 1997. JavaScript – tehokäyttäjän opas. Suomen Atk-kustannus Oy, Espoo.

Garrett, J. 2005. Ajax: A New Approach to Web Applications [online]. [viitattu 18.9.2007] Saatavissa: http://www.adaptivepath.com/publications/essays/archives/000385.php

Hintikka, K. 2006. Web 2.0. Tietokone 6/2006.

Hintikka, K. 2007. Web 2.0 – johdatus internetin uusiin liiketoimintamahdollisuuksiin [online]. Tieke, Helsinki. [viitattu 18.9.2007] Saatavissa: http://www.tieke.fi/mp/db/file_library/x/IMG/20815/file/julkaisu_28.pdf

Holzner, S. 2001. Inside XML. Edita Oyj, Edita.

Kilpikivi, P. 1997. Internet – sovellusten kehittäminen [online]. [viitattu 9.7.2007] Saatavissa: http://www.pcuf.fi/sytyke/lehti/kirj/st19972/972novo.pdf

Kuronen, T. XML – laajennettava merkintäkieli [online]. [viitattu 9.7.2007]. Saatavissa: http://herkules.oulu.fi/isbn951425242X/html/x1033.html

Nielsen, J. 2005. Why Ajax Sucks (Most of the Time). Jacob Nielsen's Alertbox [online]. [viitattu 18.9.2007] Saatavissa: http://www.usabilityviews.com/ajaxsucks.html

Nimimerkki Juice. HTML:stä XHTML:ään [online]. Saatavissa: http://www.ohjelmointiputka.net/opas.php?tunnus=xhtml

Paulson, L. 2005. Building Rich Web Applications with Ajax. Computer. Vol. 38, no. 10, 14-17.

Peltomäki, J. 2000. JavaScriptin peruskirja, 1. painos. Teknolit Oy, Jyväskylä.

Poropudas, T. 2007. Web 2.0 vetää kuin robotti [online]. [viitattu 22.9.2007] Saatavissa: http://www.digitoday.fi/page.php?page_id=15&news_id=200712046

Profium. Ajax [online]. [viitattu 8.9.2007]. Saatavissa: http://www.profium.com/index.php?id=678

Snook, J. 2006. JavaScript: Anatomy of a Drag and Drop [online]. [viitattu 22.9.2007] Saatavissa: http://snook.ca/archives/javascript/anatomy_of_a_drag_and_drop/

Tuikka, T & Kanala, S. 2001. XML Basic. Oy Edita Ab, Helsinki.

XHTML-opas [online]. 2kmediat.com. [viitattu 9.7.2007]. Saatavissa: http://www.2kmediat.com/xhtml/

Zakas, N., McPeak, J., Fawcett, J. 2006. Professional Ajax. Wiley Publishing, Inc., Indianapolis, Indiana.