Teijo Salminen

Uusi Pääkaupunkiseudun opaskartta

OpenLayers-ratkaisu Helsingin kaupungin seudullisessa web-karttasovelluksessa

Metropolia Ammattikorkeakoulu
Insinööri (AMK)
Maanmittaustekniikan koulutusohjelma
Insinöörityö
15.4.2012

Metropolia

Tiivistelmä

Tekijä Otsikko Sivumäärä Aika	Teijo Salminen Uusi Pääkaupunkiseudun opaskartta. OpenLayers-ratkaisu Helsingin kaupungin seudullisessa web-karttasovelluksessa 42 sivua + 3 liitettä 15.4.2012
Tutkinto	insinööri AMK
Koulutusohjelma	maanmittaustekniikka
Ohjaajat	toimistopäällikkö Matti Arponen lehtori Jussi Laari

2000-luvun ensimmäinen vuosikymmen on ollut web-karttapalveluiden voimakkaan kehityksen aikaa. Käyttöliittymän suunnittelun kannalta tekniikka niissä painottui aluksi palvelimessa ajettavan koodin teknologioihin, mutta asiakaspään laskentatehon kasvun ja selaintekniikan kehittymisen ansiosta on painopiste siirtynyt selaimessa ajettavan JavaScript-koodin tehokkaampaan hyödyntämiseen. Vuosikymmenen lopulla vapaan lähdekoodin sovellukset ovat vallanneet markkinoita kaupallisilta sovelluksilta. Tunnetuin vapaan lähdekoodin web-karttakäyttöliittymä on OpenLayers, noin 120 ohjelmoijan yhteisökehityksen tulos.

Tämän insinöörityön tavoitteena on dokumentoida Helsingin kaupungin kaupunkimittausosaston ylläpitämää seudullista Pääkaupunkiseudun opaskartta-sovellusta, siinä esitettyjä aineistoja sekä nykyistä toteutusta OpenLayers-sovellusalustalla. Koska omassa kehitystyössä ei toistaiseksi ole syntynyt systemaattista ja koottua dokumentaatiota ympäristön kehityksestä ja nykytilasta, se toteutetaan tässä työssä. Syntynyttä dokumenttia voidaan pitää käsikirjana paitsi olemassa olevan OpenLayers-ympäristön ylläpidossa, myös apuna mahdollisten uusien sovellusten luomisessa.

Tässä työssä käsiteltävänä olevat ohjelmointi- ja merkintäkielet ovat HTML, XHTML, CSS ja JavaScript. Pääkaupunkiseudun opaskarttasovelluksen ylläpidossa myös PHP-kielen taitoa tarvitaan, mutta koodiesimerkkejä ei ole sisällytetty tähän työhön.

Avainsanat	karttapalvelu, avoin lähdekoodi, opaskartta, Helsingin kaupunki, pääkaupunkiseutu

Metropolia

Abstract

Author Title Number of Pages Date	Teijo Salminen Guide Map of Helsinki Metropolitan Area. OpenLayers solution in Helsinki City Metropolitan web mapping service 42 pages + 3 appendices 15 April 2012
Degree	Bachelor of Engineering
Degree Programme	Land Surveying Technology
Instructors	Matti Arponen, Head of the Bureau Jussi Laari, Senior Lecturer

The first decade of 2000's has been a time for fast development of web mapping applications. In the beginning, most of the web applications were developed primarily for server side scripting techniques. Later, when client side processing power increased, techniques like JavaScript became more popular. Currently the most well known open source JavaScript-based web mapping software is OpenLayers, the result of 120 volunteering programmers.

Helsinki City Survey has been developing "Pääkaupungin opaskartta" (Guide map for Helsinki Metropolitan Area) guide map web application for over a decade. The latest version is based on OpenLayers libraries.

The aim of this Bachelor's thesis is to produce a review of past and present technologies and materials of Guide map, but also a comprehensive documentation and administration manual of the present OpenLayers based solution. The basics for implementing OpenLayers has been included, as well as a short review of other OpenLayers implementations in Finland and worldwide.

The programming and markup languages used in this document are HTML, XHTML, CSS and JavaScript. The knowledge of PHP is required to administer the Guide Map environment in City Survey, but code examples are excluded from this document.

Keywords	Web map application, open source, guide map, Helsinki City, OpenLayers, Helsinki Metropolitan area

Sisällys

Lyhenteet, käsitteet ja määritelmät

Ajax	*Asynchronous JavaScript and XML*. Sekä asiakas- että palvelinpäässä toimivien ohjelmointitekniikoiden yhdistelmä, jolla sovellukset saadaan joustavaksi, vuorovaikutteiseksi kokonaisuudeksi. Asiakaspäässä, eli www-selaimessa tämä tarkoittaa käytännössä JavaScript-koodia, joka kutsuu palvelimessa ajettavaa koodia. Viimeksi mainittu voi olla käytännössä millä hyvänsä ohjelmointikielellä toteutettu.
API	*Application programming interface*. Ohjelmointirajapinta.
ASP/VBScript	*Active Server Pages, Visual Basic Scripting*. Microsoftin omiin käyttöjärjestelmiinsä luoma ohjelmointiympäristö web-ohjelmointikehitystä varten. IIS tukee ASP/VBScript-kielellä kirjoitettuja ohjelmia suoraan.
BSD-lisenssi	*Berkeley Software distribution licensing*. Avoimen lähdekoodin lisensointimenettely. BSD-lisenssi on hyvin vapaa, sallien myös käytön kaupallisten tai suljettujen sovellusten pohjana. Ehtona on vain lisenssitavan sekä alkuperän selvä osoittaminen.
CSS	*Cascaded Style Sheet*. HTML:n ja XHTML:n ja XML:n kanssa käytettäväksi tarkoitettu muotoilukieli, jolla voidaan vapaammin sijoittaa ja määrittää sivunkuvauskielellä luotujen elementtien värejä, kokoa, muotoa, sijaintia ja käyttäytymistä.
HTML5	*Hypertext markup language version 5*. W3C-konsortion HTML-sivunkuvauskielen versio 5. Kielen virallista lopullista standardia ei ole vielä tätä kirjoitettaessa julkistettu.
IIS	*Internet Information Server*. Microsoft Windows Server palvelinkäyttöjärjestelmän varusohjelmistoon kuuluva web-palvelinohjelmisto

JavaScript	Www-sivujen dynaamisen sisällön tuottamiseen kehitetty ohjelmointikieli. Kaikki nykyaikaiset www-selaimet pyrkivät tukemaan ja suorittamaan JavaScript-koodia mahdollisimman nopeasti.
W3C	*World Wide Web Consortium*. World Wide Webin standardeja (mm. HTML, XHTML, XML, CSS) ja kehitystä valvova kansainvälinen yhteistyöelin.
WMS	*Web Map Service*. Rasterimuotoisen paikkatiedon kyselyrajapinta.
WFS	*Web Feature Service*. Vektorimuotoisen paikkatiedon kyselyrajapinta.
ODBC	*Open Database Connectivity.* Tietokantayhteyksiin Windows-palvelinympäristössä käytetty kyselyrajapinta
PHP	*Personal Home Page; PHP: Hypertext preprocessor*. Vapaan lähdekoodin ohjelmointikieli ja sen ajoympäristö. Suunnattu erityisesti web-ohjelmistokehitykseen. Asennettavissa useimpiin olemassa oleviin palvelinympäristöihin. [10]
POI	*Point of interest*. Kiinnostavat kohteet kartalla esitettynä, ”turistikohteet”, kokoontumispaikat, sairaalat jne.
UTF-8	*Unicode Transformation Format*. 8-bittinen häviötön unicode-merkistö. Käytetään mm. XML-kielen oletusmerkistönä. Myös yleinen (X)HTML-sivujen merkistönä laajan merkistötukensa ansiosta. [9]
XHTML	*Extended hypertext markup language*. W3C-konsortion standardoima XML–kuvauskielen hybridiversio, joka on kehitetty HTML 4 -version korvaajaksi nykyaikaisten www-sivustojen, CSS:n ja skriptikielitekniikoiden parempaan dynaamiseen hallintaan. Tällä hetkellä valtavirtaa maailman www-sivujen kehitystyössä.

1 Johdanto

Pääkaupunkiseudun opaskartta -verkkosovellus on jo yli kymmenen vuoden ajan ollut Helsingin kaupunkimittausosaston eniten käytetty verkkopalvelu. Sen vaiheita ei ole tähän saakka juurikaan dokumentoitu. Viimeisen suuren muutoksen palvelu on kokenut, kun sen käyttöliittymätoteutuksessa sovellettiin ajanmukaista dynaamista Ajax-tekniikkaan perustuvaa avoimen lähdekoodin ratkaisua. Uusi käyttöliittymä perustuu OpenLayers-kirjastoon, josta on muodostunut eräänlainen vapaiden paikkatietosovellusten markkinajohtaja. Uusi sovellusversio on kokonaisuudessaan tämän insinöörityön tekijän rakentama, joten ensi käden tiedon tallentamiseen on selkeä tarve. Sovelluksen käyttöönotosta ei ennen tätä ollut olemassa riittävän kattavaa ja yleistajuista dokumenttia. Myöskään kattavaa kirjallista aineistoa OpenLayers-tekniikan soveltamisesta kuntasektorin paikkatietojen esittämiseen ei liene olemassa.

Tämän insinöörityön tarkoituksena on paitsi dokumentoida Pääkaupunkiseudun opaskartan nykyisen version käyttöönotto ja sovellusratkaisut, myös valottaa sovelluksen historiaa ja tutustuttaa lukija taustalla oleviin paikkatietoaineistoihin. Perustietous OpenLayers–sovelluksen käyttöönotosta selvitetään ja kerrotaan tulevaisuuden suuntaviivoista. Esimerkkejä sekä Helsingin kaupungin sovelluksista, että maailmalla tehdyistä toteutuksista esitellään työn loppupuolella.

Palvelimella ajettavaa osuutta (PHP ja ASP VBScript) ohjelmakoodin osalta ei ole sisällytetty tähän insinöörityöhön. Se on dokumentoitu vain Kaupunkimittausosaston sisäiseen käyttöön. Sen sijaan selaimessa ajettavat JavaScript-, CSS-, XHTML-koodit on suurimmaksi osaksi esitelty. Näiden hyödyntämiseen on hyvä hallita perusteet edellä mainituista merkintä- ja ohjelmointikielistä. Myös olio-ohjelmoinnin perusteiden tuntemus auttaa dokumentaation teknisen osan ymmärtämisessä.

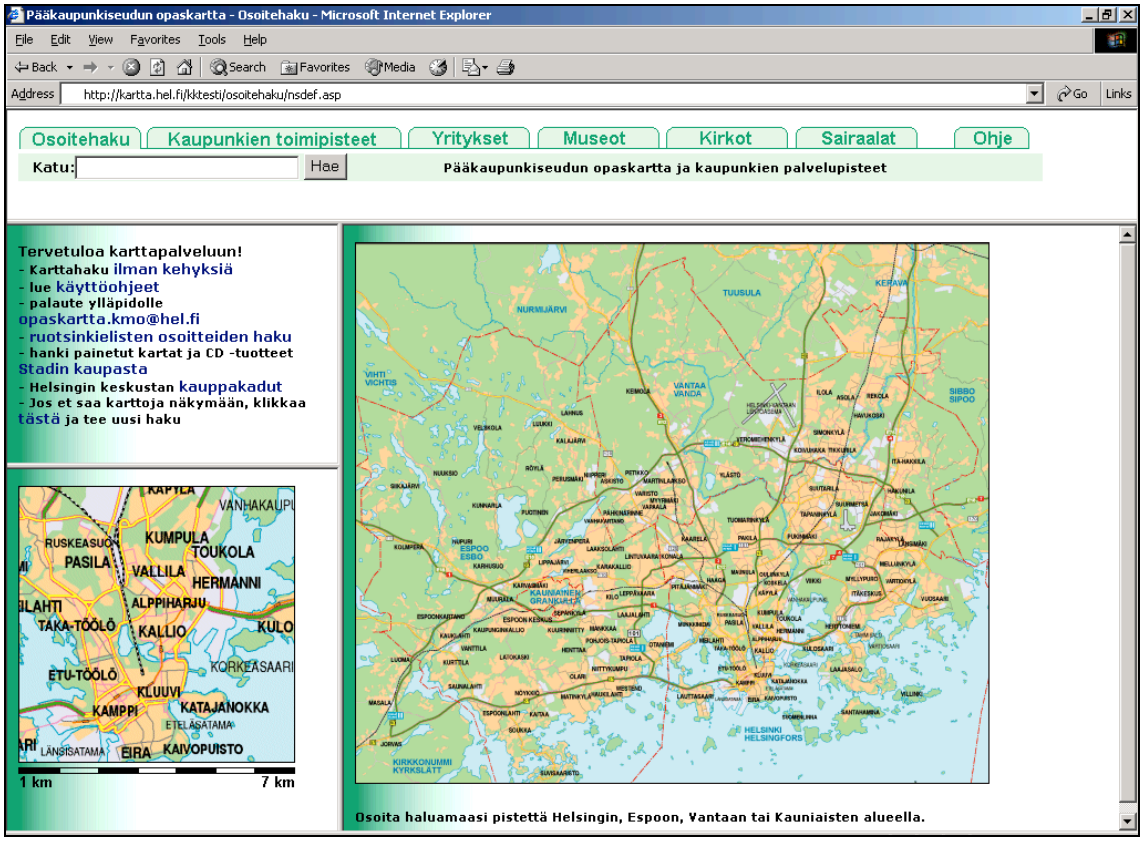
2 Pääkaupunkiseudun opaskarttapalvelun historiaa

2.1 Ensimmäisen sukupolven toteutukset

Ensimmäiset kokeilut World Wide Webissä toimivan opaskarttasovelluksen käyttöönottoon Helsingin kaupungilla toteutuivat kesäkuussa 1998, kun Helsingin kaupungin www-sivuilla julkistettiin Opti Interconsult Oy:n tuottama Net RMV -karttasovellus [1]. Kansalaisten internetin käyttö oli tuolloin vielä melko vähäistä, ja verkkoyhteydet olivat vielä yleisesti hitaita puhelinmodeemeihin perustuvia yhteyksiä, joten palvelun käyttö jäi toistaiseksi suhteellisen vähäiseksi. Pioneerityö oli jo kuitenkin tehty, joten aivan vuosikymmenen lopussa virisi ajatus uuden palvelun tilaamiseen, tällä kertaa koko pääkaupunkiseudun laajuisena.

Seuraava Pääkaupunkiseudun opaskartan www-versio kehitettiin vuoden 1999 aikana, tarkoituksena palvella niin kansalaisia kuin turistejakin. Projektin käynnistäjänä toimi Helsingin kaupungin kiinteistöviraston Kaupunkimittausosasto yhteistyössä kulttuuripääkaupunkivuoden Helsinki 2000-säätiön sekä Kaupunginkanslian kanssa. Näistä ensin mainittu hankki palvelimen ja palvelintilan, jälkimmäisen hoitaessa osan rahoituksesta vuosittaisen maksun muodossa. Kaupunkimittausosasto puolestaan hoiti ohjelmistojen hankinnan, asentamisen ja aineistojen toimittamisen. [2]

Sovelluksen ensimmäinen toteuttaja oli Bentley Finland Oy/Tmi Thomas Nynäs, ja se oli toteutettu Windows NT 4.0 Server -käyttöjärjestelmälle ja IIS/Netscape web-palvelimille ASP VBScript-kielellä toteutettuna HTML-sovelluksena. Osoitehakutoiminnossa oli käytetty myös jonkin verran yksinkertaisia JavaScript-funktioita. Myös CSS-tyylejä hyödynnettiin sisältömuotoilujen tekemiseen. Sivurakenne oli rakennettu kehyksistä (HTML frameset), joissa kartat ja toiminnot esitettiin (kuva 1).



Kuva 1. Pääkaupunkiseudun opaskartta vuonna 2000

Vuoden 2004 aikana karttasovellus koodattiin omana työnä kokonaan uudestaan. Sen jälkeen Pääkaupunkiseudun opaskarttaan ei ole tarvittu ulkopuolista konsultointia, vaan ylläpito on ollut täysin Kaupunkimittausosaston hoidossa. Uudistuksen yhteydessä osoitehakuun tehtiin parannuksia ja joitakin toimintoja tuotiin lisää. Uudessa sovelluksessa luovuttiin kokonaan kehysten käytöstä. Rakenne korvattiin taulukkoelementeillä, ja sivu rakentui yhdestä ainoasta palvelimessa dynaamisesti rakentuvasta HTML-sivusta. Muun muassa kohdesymbolit ja niiden näyttö kartalla tulivat uutena piirteenä, samoin hieman aiempaa suurempi kartta-ala.

2.2 Uusi ratkaisu

Vuosikymmenen loppua kohti staattinen, dynamiikaton karttasivu alkoi olla jo vanhentuvaa tekniikkaa. Vanhassa karttasovelluksessa oli vain kaksi zoom-tasoa, eikä karttaa voinut vierittää hiiren avulla raahaamalla. Suurten näyttöjen vallatessa markkinat ongelmaksi muodostui myös se, että kartta-alue oli aina saman kokoinen, riippumatta asiakkaan näytön koosta. Yleinen kehitys aiheutti paineita uuden,

ajanmukaisemmalla tekniikalla rakennetun karttapalvelun luomiseen. Tässä vaiheessa ohjelmistomarkkinoille oli jo tullut mielenkiintoisia, avoimeen lähdekoodiin perustuvia ratkaisuja, kärjessään mukaillulla BSD-lisenssillä jaettava OpenLayers. Se on puhtaasti JavaScript-kieleen ja nykyselaimissa hyvin tuettuun Ajax-tekniikkaan perustuva ohjelmakirjasto, joka tarjoaa täysin omaan käyttöön ja tarpeeseen mukautuvan ratkaisumallin. Pääkaupunkiseudun opaskartan uusi versio julkistettiin viimein vuonna 2008 OpenLayers-toteutuksena. Julkistus tapahtui osana Helsinginseutu.fi-portaalipalvelua. Toteutus ei kuitenkaan sijaitse fyysisesti Helsinginseutu.fi-portaalin palvelimella, vaan yhdellä Kaupunkimittausosaston palvelimista, joten sekä koodin että palvelinympäristön ylläpito on kokonaan Kaupunkimittausosaston vastuulla. Ulkoasu muotoiltiin yhtenäiseksi Helsinginseutu.fi-sivuston ulkoasun kanssa. Osoitteekseen palvelu sai muodon kartta.helsinginseutu.fi, joka mukailee Suomessa yleisesti sovellettua kuntien karttapalveluiden osoitekäytäntöä.

3 Opaskarttapalvelun paikkatietoaineistot

3.1 Opaskartta

Opaskartta on kartta-aineisto, jota suurin osa Suomen kunnista valmistaa joko omana tai konsulttityönä. Se ei ole kaavoituksen pohjakartan tapaan aineisto, jonka tuottamiseen olisi lakiin perustuvaa pakkoa, vaan tarve on lähinnä käytäntölähtöinen. Opaskartta on käytännöllisin tiedon lähde niin turisteille, osoitteita hakeville kansalaisille kuin virkamiehillekin tapauksissa, joissa tarvitaan helposti omaksuttavaa yleissilmäyskarttaa. Opaskartasta kaupunkirakenteen, liikenneväylät ja seudullisen nimistön voi havainnoida nopeasti ja vaivattomasti. Opaskartta on myös teemakarttatuotannossa hyvä pohja-aineisto. Koska opaskartta on kuvausteknisesti lähinnä suuntaa antava, sitä ei voida käyttää kaikkiin tarkoituksiin. Se on paikoin hyvinkin vahvasti yleistetty esitys todellisuudesta. Esimerkiksi kaavatulkintojen teko opaskarttaa pohjatietona käyttäen olisi jopa vaarallista. Opaskartta ei ole yksityiskohdiltaan mittatarkka, koska esimerkiksi tiet ja kadut ovat liioitellun levyisiä näkyvyyden ja nimistötekstien luettavuuden edellyttämän tilan vuoksi. Joissakin paikoissa, kuten Helsingissä Kulosaaren pohjoisosassa, on rantaviivaa jouduttu

siirtämään merelle päin kymmeniä maastometrejä, jotta kaikki Itäväylän pohjoispuolella olevat kadut nimistöineen on saatu kuvatuksi mukaan.

3.1.1 Opaskartan kuvaustekniikka

Opaskartta kuvaa lähinnä tiestöä, julkisia rakennuksia sekä rakennetun alueen tyyppejä. Tyypit on erotettavissa värikoodein, pääteiden tyypit joko värikoodin tai leveyden perusteella. Nimistöllä on opaskartalla erityisen tärkeä rooli. Koska opaskartan tarkastelumittakaava on yleensä pieni, on kaiken tämän karsitun ja yleistetyn informaation saattaminen karttaan piirtoteknisesti haastavaa. Nimistöä on toisinaan lyhennetty, eikä esimerkiksi ruotsinkielisiä nimikäännöksiä ole läheskään kaikissa kohteissa pystytty lisäämään.

3.1.2 Opaskartta pääkaupunkiseudun kattavana tuotteena

Koko pääkaupungin kattava opaskartta on varsin vanha tuote. Kuntien yhteisten suositusten laatiminen osoite- ja opaskarttatuotteiden valmistamiseksi aloitettiin jo 1970-luvun alussa [3, s. 170]. Helsingin kaupunkimittausosasto tuotti ensimmäisen Helsingin, Vantaan, Espoon ja Kauniaisten yhdistetyn osoitekarttatuotteen 1979 [3, s. 171]. Vuonna 1976 Helsingin Puhelinosuuskunta päätti laajentaa Helsingin seudun puhelinluettelon karttaosan koko pääkaupunkiseudun kattavaksi. Puhelinluetteloa varten yhdistetty opaskartta osoittautui yleishyödylliseksi kartaksi. Se olikin pitkään tunnetuin Helsingin kaupunkimittausosaston karttatuotteista [3, s. 177]. Painettuja Helsingin sekä Pääkaupunkiseudun opaskarttoja valmistetaan edelleen ja niitä toimitetaan tilauksesta pohjustettuina seinäkarttoina lukuisille kaupunkien omille hallintokunnille sekä julkisille ja yksityisille toimistoille. Pääkaupunkiseudun kuntien yhteistyö kaikilla kunnallisen toiminnan sektoreilla tekee yhtenäisen, vakioidulla kuvaustekniikalla varustetun kartta-aineiston tarpeelliseksi. On odotettavissa, että opaskartalla on edelleen vakiintunut tilaus niin kaupungin kuin yksityisenkin puolen organisaatioissa.

3.1.3 Opaskartan digitalisoiminen

Kaikkien paikkatietoaineistojen tapaan myös opaskartta koki ylläpidollisen muutoksen 1990-luvun alussa, kun karttatuotannossa siirryttiin digitaaliseen tuottamiseen ja ylläpitoon. Erityinen haaste on ollut eri kuntien tuotantojärjestelmien tuottamien opaskartta-aineistojen yhdistäminen. Helsingin kaupunkimittausosasto on suorittanut yhdistämisen sekä vanhan analogisen kartantuotannon että digitaalisen tuotannon aikana. Helpotusta digitaalisten aineistojen yhdistämiseen on tuonut lähinnä se, että Vantaalla ja Helsingissä tuotantojärjestelmät perustuvat samaan ohjelmistoympäristöön. Espoon järjestelmän tuottama rasteritasoaineisto on kuitenkin saatu istumaan Helsingin ja Vantaan aineistoon sekä kuvausteknisesti että sijainnillisesti yhteneväiseksi, joskin perinteisesti Espoon aineiston yhdistäminen on ollut aina hieman hankalampi ja aikaavievämpi työ erilaisen tuotantosovellusohjelmiston vuoksi. Kauniaisten alue on sisältynyt Espoon aineistoon, joten siltä osin aineiston yhdistämisestä ja sovittamisesta ei ole tarvinnut erikseen huolehtia.

3.1.4 Yhteinen koordinaatisto

Karttojen yhdistäminen vaatii yhden kaikille sopivan koordinaattijärjestelmän käyttämistä. Koska sekä Espoo että Vantaa käyttävät toistaiseksi kartoitustyönsä perustana ns. vanhaa valtion järjestelmää (tästä eteenpäin lyhennetty VVJ), katsottiin aikoinaan parhaimmaksi tehdä yhdistämistyö tätä järjestelmää käyttäen. Helsingin koordinaatiston ja VVJ:n välillä oli olemassa riittävän tarkat muunnostekijät, joiden avulla Helsingin opaskartta-aineisto (jota siis ylläpidetään toistaiseksi Helsingin järjestelmässä) saadaan aina muunnetuksi VVJ:hin. Vaikka tarkkaan ottaen Vantaan ja Espoon VVJ-järjestemät eivät ole aivan samat, on kuitenkin ero niin pieni, että tarkkuus riittää opaskartan saattamiseksi yhteen. Vuoden 2012 ja 2014 välisenä aikana kaikkien neljän kunnan tuotantojärjestelmien koordinaattijärjestelmät on tarkoitus muuttaa valtakunnallisiin EUREF-FIN-koordinaattijärjestelmiin. Lopullinen yhteinen koordinaattijärjestelmä myös opaskartan osalta tulee näin olemaan ETRS-GK25.

3.1.5 Opaskartta-aineiston tuottaminen

Opaskarttaa ylläpidetään Helsingin kaupunkimittausosastolla Bentley MicroStation-vektorimuodossa. Koska kohteet ovat yhtenäisiä väripintoja, kartta koostuu sulkeutuvista alueista, jotka täytetään. Varsinaista vektoriesitystä käytetään vain harvoin sellaisenaan, vaan varsinaiseen kartan loppukäyttöön käytetään rasteriopaskarttaa, joka on tuloste vektorimuodossa ylläpidetystä aineistosta. Tulostamiseen käytetään Bentleyn Iplot-ohjelmistoa, jonka kynätauluasetuksilla saadaan tuloste halutunlaiseksi värien, viivanpaksuuksien ja täyttöjen määrityksillä. Koska rasteroidusta tulosteesta tulee sellaisenaan yleensä visuaalisesti hieman karkea, tuloste tehdään ensin 1 metrin pikseliresoluutioon 8 bitin väriavaruudella. Tällä tavalla tulostettu kuva voidaan pienentää ja samalla väriavaruus laventaa 24-bittiseksi (RGB). Menettelyn etuna on se, että kun pienennyksessä käytetään niin kutsuttua uudelleennäytteistysmenetelmää (resampling) ja värimaailma lavennetaan, saadaan viivoille ja väripintojen rajoille reunapehmennys ”sivutuotteena”. Lopputuloksesta saadaan näin silmää miellyttävämpi ja teksteistä helppolukuisempia. Lopullisena nimellistarkkuutena opaskartalle on näin ollen määritetty 2 metriä kuvapisteelle. Opaskartasta tehdään myös epätarkempaa, harvennetulla nimistöllä varustettua versiota. Tällöin nimistöä on vain suuremmilla kohteilla, kuten kaupunginosilla, pääkaduilla ja vesialueilla. Harvennetun opaskartan lopullinen nimellismittakaava on 4 metriä/kuvapiste, jolloin sitä voidaan hyödyntää mm. web-karttapalveluissa ”uloszoomauskarttana”.

3.2 Pääkaupunkiseudun ilmakuva

Uutena taustakarttamateriaalina on uuteen Pääkaupunkiseudun opaskarttaan lisätty seudullinen ilmakuva. Tämän mahdollisti ensimmäinen koko pääkaupunkiseudun kattavan ilmakuvauksen suorittaminen vuonna 2009. Pääkaupunkiseudun kuntien lisäksi hanketta oli rahoittamassa myös Helsingin seudun ympäristöpalvelut HSY. Uusi kuvaus suoritettiin kesällä 2011, ja näillä näkymin vastaavanlaisia kuvauksia tullaan jatkamaan säännöllisesti, joten pääkaupunkiseudun laajuinen ilmakuva tulee mitä ilmeisimmin jatkamaan vaihtoehtoisena karttatasona opaskartan ohella.

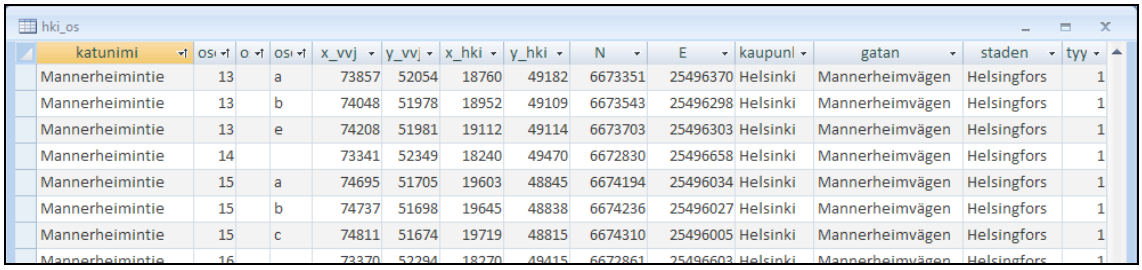
Ilmakuvakartan mittakaava riippuu kuvaustarkkuudesta. Esimerkiksi vuoden 2011 pääkaupunkiseudun ortokuvasarjat toimitettiin 20 cm:n resoluutiolla. Pääkaupunkiseudun opaskarttapalvelussa maksimiresoluutio on toistaiseksi asetettu puoleen metriin kuvapistettä kohden. Koska ilmakuvaukset toimitetaan yleensä tätä tarkempana, ilmakuva-aineistojen laatu riittää hyvin opaskarttapalvelun tarpeisiin. Ilmakuvakarttataso näin jatkaa tarkempana karttatasona siitä mihin opaskartan tarkkuus jää. Jatkossa on mahdollista, että vielä yhtä askelta tarkempaa (25 cm /kuvapiste) aineistoa tarjotaan.

3.3 Osoitekanta

Kaupunkimittausosasto käyttää osoitetietojen palvelukantana Microsoft Access-tietokantatiedostoa, joka sisältää paitsi Helsingin omat osoitteet, myös muiden pääkaupunkiseudun kuntien (Espoo, Vantaa, Kauniainen) osoitteet. Muiden kuntien osoitteistot kerätään noin kolmen kuukauden välein päivitystä varten. Helsingin omat osoitteet ovat lähes reaaliaikaisesti käytettävissä suoraan ylläpidosta. Osoitteisto käsittää sekä aitoja asemakaavan mukaisia osoitteita että nimistönimiä, jotka on Helsingin ylläpidossa koodattu omilla tunnistekoodeillaan. Muut kunnat lisäävät nimistökohteita harkintansa mukaan, mutta näitä ei erotella osoiteaineistossa.

Osoitekannan (kuva 2) tietueen muodostavat katunimi, osoitenumero, toinen osoitenumero (tapauksissa joissa osoitenumerointi osoittaa numeroväliä, esimerkiksi 2–5), osoitekirjain (alakirjain esim. Mannerheimintie 15 a), pohjois- ja itäkoordinaatit omissa sarakkeissaan Helsingin-, VVJ- ja ETRS-GK25-järjestelmissä, katunimi ruotsiksi sekä kaupungin nimi suomeksi että ruotsiksi. Kullakin kunnalla on kannassa oma taulunsa, mutta nämä taulut myös yhdistetään yhdeksi koko pääkaupunkiseudun kattavaksi yhdistelmätauluksi. Osoitetietojen palvelukanta sisältää myös omat taulunsa niin kutsuttujen POI-kohteiden (Point of Interest) keräämistä varten. Näitä ovat mm liikennepaikat (junien ja metrojen asemat), sairaalat, museot ja muut kulttuurikohteet. Näiden kohteiden ylläpito tapahtuu Helsingin kaupunkimittausosastolla. Ylläpito pyritään pitämään ”kevyenä”, jolloin ainoastaan huomattavimmat kohteet kirjataan kantaan, eikä esimerkiksi yksittäisiä terveyskeskuksia, liikkuvia näyttelyitä tai tilapäisiä tapahtumia pyritä sisällyttämään ylläpidon piiriin. Jatkossa tutkitaan myös mahdollisuutta poimia kohteita Palvelukartan rajapintojen kautta. Nimistökohteiden

keräämisen osalta hyvänä vähimmäiskriteerinä voidaan pitää sitä, että opaskartalla olevat nimet löytyisivät myös osoitekannasta.



hki_os

katunimi	osi	o	osi	x_vvj	y_vvj	x_hki	y_hki	N	E	kaupunl	gatan	staden	tyy
Mannerheimintie	13		a	73857	52054	18760	49182	6673351	25496370	Helsinki	Mannerheimvägen	Helsingfors	1
Mannerheimintie	13		b	74048	51978	18952	49109	6673543	25496298	Helsinki	Mannerheimvägen	Helsingfors	1
Mannerheimintie	13		e	74208	51981	19112	49114	6673703	25496303	Helsinki	Mannerheimvägen	Helsingfors	1
Mannerheimintie	14			73341	52349	18240	49470	6672830	25496658	Helsinki	Mannerheimvägen	Helsingfors	1
Mannerheimintie	15		a	74695	51705	19603	48845	6674194	25496034	Helsinki	Mannerheimvägen	Helsingfors	1
Mannerheimintie	15		b	74737	51698	19645	48838	6674236	25496027	Helsinki	Mannerheimvägen	Helsingfors	1
Mannerheimintie	15		c	74811	51674	19719	48815	6674310	25496005	Helsinki	Mannerheimvägen	Helsingfors	1
Mannerheimintie	16			73370	52294	18270	49415	6672861	25496603	Helsinki	Mannerheimvägen	Helsingfors	1

Kuva 2. Ote osoitekannasta. Kuvakaappaus Microsoft Access -ohjelmasta.

4 OpenLayers

4.1 Yleistä

OpenLayers on avoimen lähdekoodin JavaScript-kirjasto, joka mahdollistaa web-ohjelmoijalle helpon tavan luoda www-selaimella käytettäviä web-karttasovelluksia omiin, yhteisön tai yrityksen tarpeisiin. OpenLayers on julkistettu muunnetulla BSD-lisenssillä. Lisenssi sallii koodin muokkaamisen ja uudelleenkäytön ehdolla, että lisenssin tiedot sekä käyttöehdot säilytetään näkyvillä lähdekoodissa. Käytännössä BSD-lisensointi takaa, että koodia saa kuka tahansa käyttää, muokata ja soveltaa omiin tarpeisiinsa rajattomasti.

OpenLayers-projektin perusti yhdysvaltalainen MetaCarta Company vuonna 2005. Ensimmäinen versio 1.0 julkaistiin avoimen lähdekoodin julkaisuna kesäkuussa 2006 MetaCarta Labsin toimesta. Marraskuusta 2007 lähtien OpenLayers siirtyi Open Source Geospatial Foundationin (OSGeo) projektiksi.[4] Tätä kirjoittaessa OpenLayersin vakaa jakeluversio on versionumeroltaan 2.11, ja se on julkaistu syyskuussa 2011. Versioiden hallinta kehitysyhteisössä on aiemmin hoidettu Subversion-sovelluksen avulla, uudemmissa versioissa ollaan siirtymässä joustavampaan ja monipuolisempaan Git:iin.

OpenLayersia on kuluneiden vuosien aikana kehitetty noin kahden tai kolmen versioaskelen vuosivauhtia. Version 1.0 julkistuksen jälkeen ei julkaistu alanumeroita,

vaan samana vuonna (2006) julkaistiin jo versio 2.0. Tämän jälkeen on edetty alanumero kerrallaan (taulukko 1). Version 3 määrittelytyö on aloitettu GitHubissa [5]. Viimeisen tiedon mukaan vanhaa 2-sarjaa julkaistaan vielä ainakin versionumeroon 2.13 [6].

Taulukko 1. OpenLayersin versiot, julkaisuajankohdat ja kirjaston mitat

Versio	Julkaisupäivä	kirjastotiedostoja	Koko kt/Mt
OpenLayers-1.0	26.6.2006	37	230 kt
OpenLayers-2.0	25.8.2006	47	340 kt
OpenLayers-2.1	2.10.2006	50	384 kt
OpenLayers-2.2	15.11.2006	53	387 kt
OpenLayers-2.3	21.2.2007	54	407 kt
OpenLayers-2.4	29.5.2007	98	658 kt
OpenLayers-2.5	9.10.2007	113	945 kt
OpenLayers-2.6	15.4.2008	146	1,31 Mt
OpenLayers-2.7	29.9.2008	180	1,62 Mt
OpenLayers-2.8	22.6.2009	215	2,03 Mt
OpenLayers-2.9	22.4.2010	263	2,34 Mt
OpenLayers-2.9.1	6.5.2010	263	2,34 Mt
OpenLayers-2.10	9.9.2010	282	2,53 Mt
OpenLayers-2.11	11.9.2011	310	2,85 Mt

OpenLayers on suunniteltu siten, että se voi käyttää aineistoina lähes mitä tahansa www:stä löytyvää paikkatietosisältöä. Koska ohjelmointirajapinta on täysin vapaa muutoksille, periaatteessa mihin tahansa suljetunkin paikkatiedon julkaisuun tarkoitetun ohjelmiston rajapintaan voidaan omalla työllä toteuttaa yhteensopivuus.

4.2 OpenLayers API

OpenLayersin toteutuksessa on käytetty hyväksi JavaScript-kielen olio-ohjelmointiominaisuutta. Toiminnot rakennetaan oliomallin mukaisesti määrittämällä arvoja karttaolion alaisilla erilaisilla metodifunktioilla. Kuten tyypillisesti olio-ohjelmoinnissa, oliot rakennetaan omista luokkatiedostoista luettavilla luokilla. JavaScript-kielen oliomallissa tosin ei ole aitoja luokkia, vaan luokat ovat prototyyppejä, joiden pohjalta voidaan uudet oliot ottaa käyttöön kopioimalla. Myös OpenLayersin tapauksessa luokat (Class) ovat kutsumanimi prototyypeille. Ohjelmointitapa eroaa jonkun verran esimerkiksi C++:an, Javan, PHP:n tai muiden aidosti luokallisten olio-ohjelmoitavien kielten tavasta. JavaScriptissä on mahdollista sijoittaa olio muuttujaksi.

Tätä käytäntöä harjoitetaan OpenLayersin ohjelmoimisessa jatkuvasti. Perintä hoidetaan ketjutuksena luokkaprototyypeistä. Ketjutus hoidetaan JavaScriptin yleisen syntaksin mukaan pisteoperaattorilla. Esimerkki karttataso-olion luomisesta:

```
uusiTaso = new OpenLayers.Layer.WMS(<…tähän parametrit…>);
```

Esimerkissä ketjutetaan sekä luokkaprototyypit Layer että WMS.

Periaate selviää kenties parhaiten tutkimalla OpenLayersin kirjaston tiedoston Class.js koodia. Se on dokumentoitu varsin hyvin. Version 2.11 kommenttimerkinnöistä selviää myös erittäin tärkeä tieto tulevaisuutta ajatellen: kommentin mukaan OpenLayersin versiossa 3.0 luokkien luomisesta vanhalla menetelmällä ja luokkien ketjuttamisesta luovutaan. Tämä merkitsee sitä, että uuden version käyttöönotossa oman sovelluksen alustusskripti pitää todennäköisesti laatia kokonaan uudestaan.

Jos OpenLayersin toimintaperiaatetta ja omaa sovelluskehitystä haluaa todella syvällisesti opiskella, on paitsi olioparadigman periaatteet oleellista tuntea, myös JavaScript-kielen erityispiirteet olio-ohjelmointitavan suhteen.

Jokainen OpenLayersin toiminnallisuus sisältää oman prototyyppinsä, joista luoduilla olioilla sovellus rakentuu, ja joiden attribuuteilla toimintaa ohjataan. Attribuuttien arvot ovat literaalimuuttujia eli tekstejä, lukuja, boolean-operaattoreita tai NULL-arvoja. Versiossa 2.11 OpenLayersin kirjasto koostuu 310 erillisestä tiedostosta. Jakelupaketti on koostettu siten, että sovellusta voi käyttää kahdella tavalla. Sen voi joko ottaa käyttöön lataamalla päätiedosto (X)HTML:n script-merkinnän src-attribuutilla. Päätiedosto linkittyy kirjastohakemistoon, joka sisältää jokaista luokkaprototyyppiä vastaavan tiedoston. Kyseinen tapa hajasijoittaa luokat omiin tiedostoihinsa on yleinen muissakin olio-ohjelmoitavissa skriptikielissä. Toinen tapa on ladata yksi suuri tiedosto, joka sisältää koko OpenLayers–koodin tiivistetyssä muodossa. Kumpaa tapaa käyttää, on sovelluksen toteuttaman toiminnan kannalta yhdentekevää, mutta sillä saattaa olla merkitystä suorituskyvyn kannalta. Yksi suuri tiedosto on tiivistetty, eikä sisällä näin ollen kommentteja, rivinvaihtoja tai sisennyksiä, jolloin se on ainakin teoriassa kevyempi ja nopeampi kuin koko kommentoidun ja tiivistämättömän kirjaston lataaminen. Toisaalta erillistiedostoja käyttämällä voidaan pääskriptitiedostoa muokkaamalla määrittää, mitkä kirjaston luokista otetaan käyttöön kommentoimalla tai

poistamalla niitä luokkia periyttävät rivit, joita ei katsota tarvittavan. Tämän jälkeen voidaan suorittaa OpenLayers-jakelupaketin mukana toimitettavista työkaluista löytyvällä python-kielisellä skriptillä toimiva tiivistystoimenpide. Toimenpiteellä saadaan aikaan juuri halutut toiminnot sisältävä räätälöity JavaScript-kooditiedosto.

Kooditiedosto on mahdollista tietysti ladata suoraan sovellukseen myös openlayers.org:in sivulta, osoitteesta

```
http://openlayers.org/api/OpenLayers.js
```

Tämäkin tapa voi olla toimiva, mutta on myös hyvä muistaa menettelyn riskit. Koska OpenLayers-yhteisösivuston ylläpitäjä päivittää aina uusimman vakaan version suoraan latauspaikkaan, menetelmän käyttäjät voivat olla aina varmoja, että käytössä todella on uusin vakaa versio. Toisaalta, ongelmia varmasti ilmenee, jos karttasovelluksen ohjelmoija on tehnyt pitkälle vietyjä räätälöintejä, joissa käytetään esimerkiksi uuden version myötä vanhentuneita metodikutsuja. Ladattava tiedosto sisältää myös kaikki kirjaston luokat, myös ne, joille ei omassa sovelluksessa ole tarvetta.

5 Soveltaminen

5.1 Peruskokoonpano

Kuten yleensä suurin osa tällä hetkellä www:ssä olevista sovelluksista, lähtökohtana on normaali www-sivu, johon on upotettu suuri määrä selaimella ajettavaa JavaScript-koodia. Riippuen ohjelmoijan suosimasta käytännöstä, koodi sijaitsee joko erillisissä tiedostoissa tai suoraan (X)HTML-koodin seassa. Koska koodia tulee sovelluksen monimutkaisuusasteen kasvaessa huomattavan paljon, suositeltava tapa on sijoittaa koodi erillisiin tiedostoihin.

Ensimmäinen toimenpide lähdettäessä laatimaan sovellusta, on koodata pohjaksi standardinmukainen HTML- tai XHTML-sivu. Tällä hetkellä suositeltavin kuvauskieliversio on XHTML 1.0, vaikkakin vanhempi HTML 4 käy. Uutta HTML:n 5-version standardia ei ole vielä julkistettu, joten sen selaintuki voi osittain olla vielä

puutteellista. Riippumatta versiosta, kirjoitettu koodi kannattaa luonnollisesti tehdä W3C-konsortion määrittelemien standardien mukaiseksi. Viimeistään sivun valmistuttua se siis kannattaa tarkistaa sekä W3C:n HTML- että CSS-validaattorilla, jotta mahdolliset koodausvirheet ja puutteellisuudet löytyisivät. Näin saadaan ainakin alustava varmistus sille, että sivu toimisi oikein ja optimoidusti ainakin useimmilla selainohjelmilla. Tosin tämäkin riittää vain alkutoimiksi, ennen julkistusta sovellus tulisi aina testata useilla eri selainohjelmilla eri käyttöjärjestelmissä.

HTML- tai XHTML-pohjaista sovellusta laadittaessa on hyvä muistaa, että merkintäkieltä tulisi käyttää vain sivun perusrakenteen luomiseen. Ulkoasuun vaikuttavat asetukset sen sijaan tehdään CSS-tyylein. Dynaamiset muutokset ja toiminnallisuus saadaan aikaan JavaScriptin ja Ajax-tekniikan avulla.

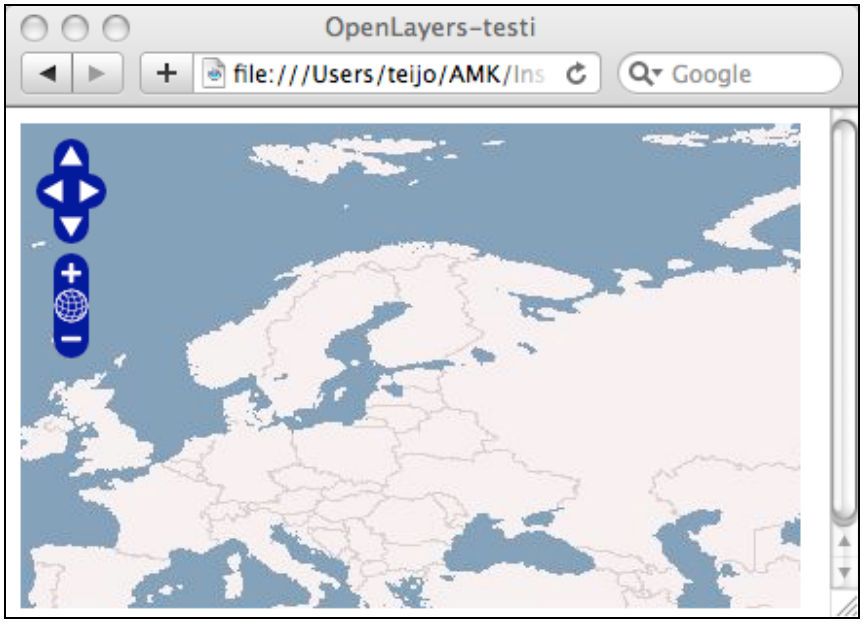
OpenLayersin rakenteen vuoksi pelkkä HTML tai XHTML ei riitä, eli on otettava käyttöön CSS-tyylimääritykset. Koska OpenLayers-kirjaston tuottama kartta eli näkymäportti (viewport) sijoitetaan sivulle -merkinnällä, tämän koko on ilmaistava CSS-tyylimäärityksen height- ja width-parametreilla. Yksinkertaisin mahdollinen koodiesimerkki voi olla seuraavanlainen:

```
<html>
<head>
<title>OpenLayers-testi</title>
<script src="http://openlayers.org/api/OpenLayers.js">
</script>
<script>
var map, layer;
function init(){
        map = new OpenLayers.Map("map");
        layer = new OpenLayers.Layer.WMS("kartta",
        "http://vmap0.tiles.osgeo.org/wms/vmap0",
        {layers:"basic"});
        map.addLayer(layer);
        map.setCenter(new OpenLayers.LonLat(25,60),3);
}
</script>
</head>
<body onload ="init()">
<div id="map" style="height: 250px; width: 400px"></div>
</body>
</html>
```

Tämä vain noin 20 riviin mahtuva kaavamainen koodiesimerkki on toki vaillinainen eikä W3C:n standardien mukainen, mutta sopii periaatteen esittämiseen. Se myös toimii selaimessa, joka sallii huonon ja puutteellisen koodin.

Head-osassa otetaan käyttöön OpenLayersin sovelluskirjasto. Ensimmäisessä script–merkinnässä määritetään OpenLayersin kirjaston sijainti. Tässä tapauksessa se ladataan suoraan OpenLayers-yhteisön sivulta. Toisen script-merkinnän sisään on kirjoitettu minimaalinen alustusfunktio (init). Alustuksessa tehdään välttämättömät toimet, muiden muassa luodaan karttaolio. Body-osassa on yksi ainoa div-merkinnällä oleva elementti. Sen parametrina ovat vaadittavat tunniste (id), sekä CSS-tyylikielellä määritetty korkeus ja leveys.

Kuvassa 3 nähdään kuvakaappaus selainikkunasta, jossa koodiesimerkin koodilla varustettu HTML-sivu on ladattuna.



Kuva 3. Minimaalinen OpenLayers-sovellus Safari-selaimessa. Pohjoismaiden kuvautuminen voimakkaasti litistyneenä johtuu kartta-aineistossa käytetystä projektiosta.

Openlayers.org-verkkosivustolta löytyy lukuisia esimerkkikarttasivuja, joista kukin esittelee eri toiminnallisuuksia, usein joko yhtä tai muutamaa toiminnallisuutta kerrallaan. Näiden lähdekoodia tutkimalla käyttäjän tulisi päästä selville siitä, kuinka eri toimintoja otetaan käyttöön. Seikkaperäisempää dokumentaatiota on olemassa melko vähän. Luonnollisesti jokaisen verkossa nähtävissä olevan OpenLayers-karttasovelluksen koodi on täysin käytettävissä, koska selainohjelmassa ajettavana koodina JavaScript on aina myös sellaisenaan ladattavissa tarkastelua varten. Ohjelmoijan tekemistä ratkaisuista riippuen koodi voi olla paitsi hajautettuna useisiin

erillisiin tiedostoihin, myös tiivistetyssä muodossa. Kumpikin ratkaisu saattaa tehdä koodin tulkinnasta erittäin haastavaa. Tiivistettynä pitkän koodin tulkinta on lähes mahdotonta ilman ohjelmallista uudelleenjäsennystä. Vaikka valmiita ratkaisuja saattaa maailmalta löytää, usein kokonaan oma, huolellisesti dokumentoitu ja rakenteellisesti selkeä koodaustyö osoittautuu kannattavimmaksi ratkaisuksi.

Omaa karttasovellusta suunnittelevalle ohjelmoijalle konfigurointi on helppoa etenkin, jos haluaa käyttää OpenLayersin oman WMS-palvelimen tarjoamaa yleispiirteistä maailmankarttaa, OpenStreetMap-yhteisön avointa karttaa tai vaikkapa Google Mapsin tarjoamaa karttaa. Koordinaatisto kaikissa näissä tapauksissa on asteyksiköissä esitetty WGS-84, jota OpenLayers käyttää oletuksena. Suurempimittakaavaisten, omissa paikallisissa tai kansallisissa koordinaatistoissa olevien karttojen käyttöönottoon nämä koordinaatistot ja yksiköt eivät sovellu, joten sekä koordinaatistomääritys että käytettävät mittayksiköt on määritettävä uudestaan. Samalla kun ne määritetään, pitää myös käytettyjen aineistojen ulottuvuudet valituissa mittayksiköissä määrittää, jotta käyttöliittymä mukautuu oikein käytettävän koordinaatiston suhteessa.

Web-sivulla näytettävä karttaelementti koostuu yhdestä, yleisesti nimellä ”map” esiintyvästä oliosta. Tämän visuaalinen koko ja paikka käyttöliittymässä määritellään CSS-muotoilujen avulla. Karttaelementti pitää käytännössä luoda div-elementin sisään johon CSS-tyyli kohdistetaan id-määreellä. Tyylimääritys on pakollinen koska karttaelementin vaaka- ja pystyulottuvuudet sivulla on määritettävä. Toteutus voi ilmetä vaikkapa koko selainikkunan kokoisena karttana, jonka päälle CSS:n ja JavaScriptin avulla asemoidaan halutut ohjauselementit ”kellumaan”.

Karttaolio sisältää koko OpenLayers–karttaelementin toiminnot. Se muodostetaan OpenLayers.map-prototyypillä. Kaikki OpenLayersin toiminnot muodostetaan omina olioinaan ja ne liitetään map-olion sisään sopivilla metodeilla. Yksinkertainen koodiesimerkki map-olion muodostamisesta:

```
map = new OpenLayers.Map('map').
```

Esimerkin karttaoliolle on annettu nimi ”map”. Nimi voi luonnollisesti olla täysin vapaasti valittava merkkijono, mutta OpenLayers-ohjelmoijat suosivat ”map”-nimeä. Yhtenä syynä voi olla esimerkiksi se, että melkein kaikki OpenLayersia koskevat web-

sivustot käyttävät esimerkeissään tätä nimeä. Mahdollisissa ongelmatilanteissa koodiesimerkkien hahmottaminen on helppoa, kun käytetään yhtenäistä nimeämiskäytäntöä. Esimerkissä OpenLayers.Map -prototyypin parametrina annettava ”map” on tunniste, jonka avulla OpenLayers-karttaelementti voidaan ottaa html-sivulla käyttöön sijoittamalla se div-elementin tunnisteeksi.

Koska pelkkä karttanäyttö ei yksin riitä, sille pitää määritellä sisältö sekä toiminnot, joilla karttanäkymää ja sisältöä voisi käsitellä ja muuttaa. Näitä varten on olemassa prototyypit OpenLayers.Layer ja OpenLayers.Control.

Käyttöliittymän hallintatyökalut otetaan käyttöön luomalla olio OpenLayers.Control-prototyypistä, joka edelleen liitetään map-olioon. Näitä ovat mm. karttanäytön vuorovaikutteista hallintaa säätelevät kontrollit, kuten hiiren ja näppäimistön kontrollit sekä karttanäytön elementit, kuten panorointi ja zoom, indeksikartta ja mittakaavajana.

Kuten esimerkissä minimaalisesta OpenLayers-sovelluksesta huomataan, kontrolleja ei ole erikseen määritetty. Jos kontrollimäärityksiä ei ole tehty, oletuksena silti ladataan sekä yksinkertainen zoomaus- ja panorointikontrolli.

Karttatasot (Layer) luodaan OpenLayers.Layer-prototyypin tukemilla alaluokilla. Karttatason prototyyppejä on kolmea päätyyppiä. Image hoitaa rasterikarttojen esittämisen. Imagen kanssa toimivat kaikki rasterimuotoisten tietolähteiden prototyypit, kuten WMS, Google Maps, MapServer ja rasterikuvat. Imagen ohessa käytetään karttapalveluiden rasterien lataamisessa myös Grid-prototyyppiä, joka hoitaa karttakutsujen palastelun sopiviksi tiiliksi. Vector hoitaa puolestaan vektorimuotoisen datan piirtämisen. Vectorin kanssa käytettävät prototyypit, kuten GML sekä WFS, hoitavat vastaavien tietolähteiden formaattien tulkinnan. Kolmas päätyyppi on Marker, joka esittää kartalla pistemäistä dataa. Marker-olioon voidaan liittää html-elementti, yleensä pieni kuvasymboli.

Esimerkki WMS-karttatason käyttöönotosta:

```
layer = new OpenLayers.Layer.WMS(
”kartta”,
”http://vmap0.tiles.osgeo.org/wms/vmap0”,
{layers:”basic”}
);
```

Koodi muodostaa karttatason ”layer”-nimiseksi olioksi. Parametreiksi annetaan nimiö (esimerkissä ”kartta”), WMS-osoite ja haluttu tai halutut karttatasot (esimerkissä ”’basic”).

Käyttöönotto tapahtuu OpenLayersin lisäysmetodeilla. Äskeisessä esimerkissä luodun layer-olion lisääminen tehdään OpenLayers.AddLayer-metodilla

```
map.AddLayer(layer);
```

Seuraavissa kahdessa esimerkissä taso-olion luomisen jälkeen viimeisenä toimenpiteenä lisätään luotu taso karttaolioon AddLayer-metodin avulla.

Esimerkki WFS-vektoritason lisäämisesta:

```
var wfs = new OpenLayers.Layer.WFS(
        "States",
        "http://demo.opengeo.org/geoserver/wfs",
        {maxfeatures: "300"}
        );
map.addLayer(wfs);
```

Tämä lataa käyttöön Opengeon WFS-palvelusta yhdysvaltain osavaltioiden rajat (states). Kohteita ladataan enintään 300 kpl.

Esimerkki pistedata-tason lisäämisestä:

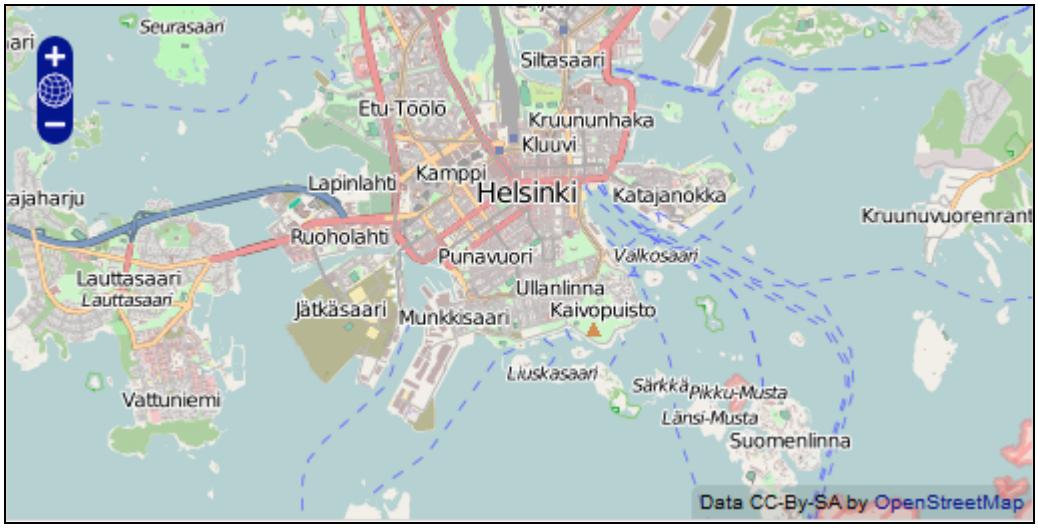
```
var kuvake = new OpenLayers.Icon("./images/icon.png",
        new OpenLayers.Size(16,16));
var grss = new OpenLayers.Layer.GeoRSS('GeoRSS',
        'http://openlayers.org/dev/examples/georss.xml',
        {'icon':kuvake});
map.addLayer(grss);
```

Tässä ladataan tietoa pistemäisiä paikkatietokohteita tarjoavasta palvelusta. Palvelu tarjoaa tietoa GeoRSS-syötteenä. GeoRSS on XML-muotoinen RSS-syöte, joka sopii erityisesti pistegeometriakohteiden esittämiseen.

Lopuksi esimerkki OpenStreetMap-karttatason (OSM) lisäämisestä. Toimenpide on hyvin yksinkertainen, koska OSM:n kartta on vakioitu ja haetaan aina samalta palvelimelta. Osoitetta ei tarvitse siis erikseen määrittää. Koodirivit

```
map = new OpenLayers.Map( 'map');
layer = new OpenLayers.Layer.OSM( "Simple OSM Map");
map.addLayer(layer);
```

riittävät pienen OpenStreetMap-sovelluksen luomiseksi (kuva 4).



Kuva 4. OpenLayersin sovellus, jossa karttatasona OpenStreetMap.

5.2 Pääkaupunkiseudun opaskartan kokoonpano

Pääkaupunkiseudun yhtenäistä kartta- ja osoiteaineistoa ylläpidetään vielä tätä kirjoittaessa niin kutsutussa Vanhassa valtion koordinaattijärjestelmässä (VVJ). Siirtymä uuteen koordinaattijärjestelmään aineistotuotannon osalta tapahtuu vuoden 2012 aikana. Koska käytössä siis on paikallinen koordinaatisto, on järjestelmän oletusasetukset näiden osalta sivuutettava. Tämän mahdollistaa OpenLayersin rakenne, jossa olioiden oletusattribuutit voidaan ylikirjoittaa uudella määrityksellä. Koska OpenLayers-sovellukseen on aina kirjoitettava jonkinlainen alustuskoodi funktion muotoon, niin tehdään myös Pääkaupunkiseudun opaskartan tapauksessa. Funktion nimi on yleisemminkin käytössä oleva ”init”. Funktiolle annetaan vain yksi parametri, käytettävä kieliversio. Kieliversion määritystä käytetään sovelluksen toimintojen nimiötekstien kielen asettamiseen.

5.2.1 Sovelluksen www-sivujen merkintäkieli

Sovelluksen pääsivu on XHTML 1.0-transitional -koodattu sivu. Merkistökoodauksena on UTF-8. Kyseinen merkistökoodaus on myös käytössä kaikissa sovelluksen skriptitiedostoissa, myös OpenLayersin kirjastotiedostoissa ja CSS-tyylitiedostoissa.

5.2.2 Karttaolio

Kuten yleisesti tapana on, karttaoliolle on annettu nimi ”map”. Sen ulottuvuudet määritetään kattamaan riittävä alue VVJ-koordinaattijärjestelmässä. Projektioksi määritetään ”Helsinki”. Tällä ei todellisuudessa ole merkitystä, koska mitään todellista projektiota ei tarvita, eikä sovellus toistaiseksi käytä aitoa WMS-palvelua. Yksiköt määritetään metreiksi ja tarkastelutasoja määritetään neljä kappaletta.

```
map = new OpenLayers.Map( 'map',
{controls:[],
maxExtent: new OpenLayers.Bounds(27000,65000,69000,100000),
projection: "Helsinki",
units: "m",
numZoomLevels: 4
});
```

5.2.3 Karttatasot

Karttatason esimerkiksi on tässä valittu vain yksi taso, muut tasot toimivat täsmälleen samalla periaatteella. Karttataso muodostetaan edelleen omaksi oliokseen. Koska käytämme kartan tietolähteenä skriptiä (map.php) aidon WMS-palvelun sijaan, luomme WMS-prototyyppiä käyttäen olion, jonka lähdeparametrina on WMS-osoite map.php. Puskuriattribuutiksi annetaan arvo 0. OpenLayers voi suuremmalla puskuriasetuksella ladata näkyvillä olevaa kartta-alaa huomattavasti suuremman alueen, mutta käytännössä puskurointi saattaa hidastaa kartan rakentumista näytölle. Maksimi- ja minimiresoluution ja zoom-porrastuksen asetukset on hyvä asettaa järkevästi ja alkuperäisen materiaalin resoluutio huomioiden, koska rasterikartta on visuaalisesti paras tarkasteltava sen nimellisresoluutiossa sekä tuon nimellisresoluution kahden kertoimilla. Opaskartta on nimellisresoluutioltaan 2 metriä/kuvapiste, joten maksimiresoluutiolle annetaan arvo 2. Muut arvot määritetään sen mukaan asettamalla

minimiresoluutio arvoon 16 ja zoom-tasojen määräksi 4. Näin muodostuu resoluutioiksi 2, 4, 8 ja 16 metriä/kuvapiste.

```
var op_map = new OpenLayers.Layer.WMS( maplabel1,
    "map.php",
    { layers: "Opaskartta2" },
    { buffer: 0,
      attribution: (maplabel1+copyright),
      maxResolution: 2,
      minResolution: 16,
      numZoomLevels: 4
    }
);
```

5.2.4 Käyttöliittymän hallintakontrollit

Kontrollien kohdalla on tässä sovelluksessa tehty eniten poikkeavia ratkaisuja. Karttatason vaihtokontrollia varten toteutus poikkeaa normaalista sikäli, että sitä varten on otettu käyttöön kokonaan oma, ylimääräinen kirjastotiedosto KmoLayerSwitcher.js. Tätä ei ole koodattu alusta saakka itse, vaan pohjaksi on otettu luokkatiedosto LayerSwithcer, johon on tehty tarpeelliset lisäykset ja poistettu osa turhaksi katsottua koodia. Ratkaisu on tehty helpottamaan hallintakontrollin sijoittamista karttaikkunan ulkopuolelle.

Kontrollit lisätään karttaolioon käyttämällä addControl-metodia. Kaikkiaan Pääkaupunkiseudun opaskartta -sovellukseen ladataan kontrolleina

- indeksikartta (overViewMap)
- zoom-kontrollin pitkä malli (PanZoomBar)
- mittakaavajana (ScaleLine)
- karttatason metatiedot (Attribution)
- näppäimistökontrollit (KeyboardDefaults)
- hiiren laajennetut kontrollit (MouseDefaults)
- pysyvä linkityslinkki (Permalink)
- räätälöity tasonvaihtokontrolli (KmoLayerSwitcher)
- räätälöity koordinaattinäyttö (coord)

Kukin kontrolli lisätään omana olionaan. Esimerkiksi mittakaavajanan lisäys tehdään seuraavasti:

```
map.addControl( new OpenLayers.Control.ScaleLine({maxWidth:150}) );
```

Esimerkissä nähdään, että mittakaavajanalle on annettu myös maksimileveyttä osoittava lisämääritys. Tämä ei vielä riitä halutun muotoilun saavuttamiseksi, koska mittakaavajana oletuksena on kaksipuolinen: janan alapuoli on hieman eri mittainen, koska se näyttää tasayksiköt anglosaksisissa mitoissa (jalat, mailit). Koska vain metriyksikköinen mittakaavajana halutaan näyttää, ongelma ratkaistaan peittämällä alaosa CSS-muotoilulla. Janan asemointi ja näyttö on toteutettu CSS-osina.

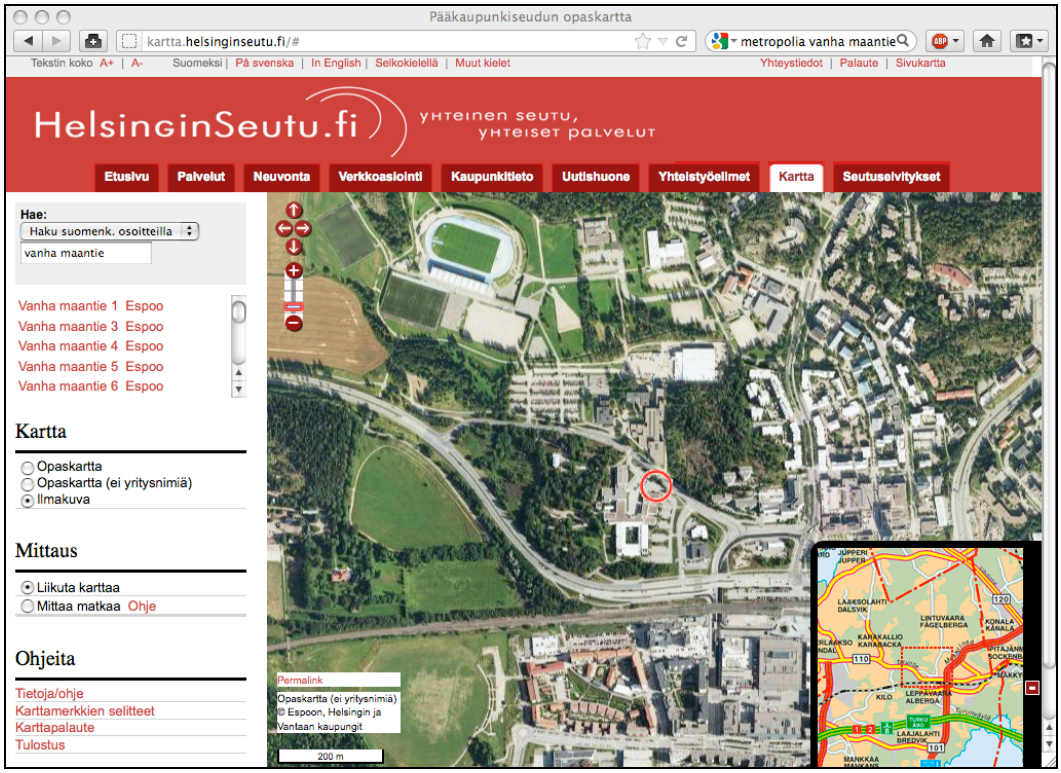
```
.olControlScaleLineBottom {
        display: none
}
```

peittää anglosaksisen osan mittakaavajanasta asettamalla sen näkymättömäksi. Merkintä

```
.olControlScaleLineTop {
        background-color: white
}
```

asettaa mittakaavajanan taustan lukemista helpottavasti valkoiseksi.

Oheiset esimerkit toivottavasti auttavat hahmottamaan perustietouden siitä, kuinka OpenLayers-sovelluksen kontrolleja käytetään ja kuinka olemassa olevaa koodia pitää tulkita. On oleellista, että ymmärtää sekä JavaScript-koodin, oliomallin sekä CSS:n käytön yhteispelin. Pääkaupunkiseudun opaskartan init-funktio on kokonaisuudessaan yli 200 koodirivin pituinen. Lisäksi tähän pitää liittää vielä apufunktiot, joilla sovelluksen toiminnallisuutta täydennetään. Alustuskripti, XHTML-koodi sekä CSS-tyylitiedosto ovat tämän insinöörityön liitteissä toimintaa selventävillä kommenteilla varustettuna.



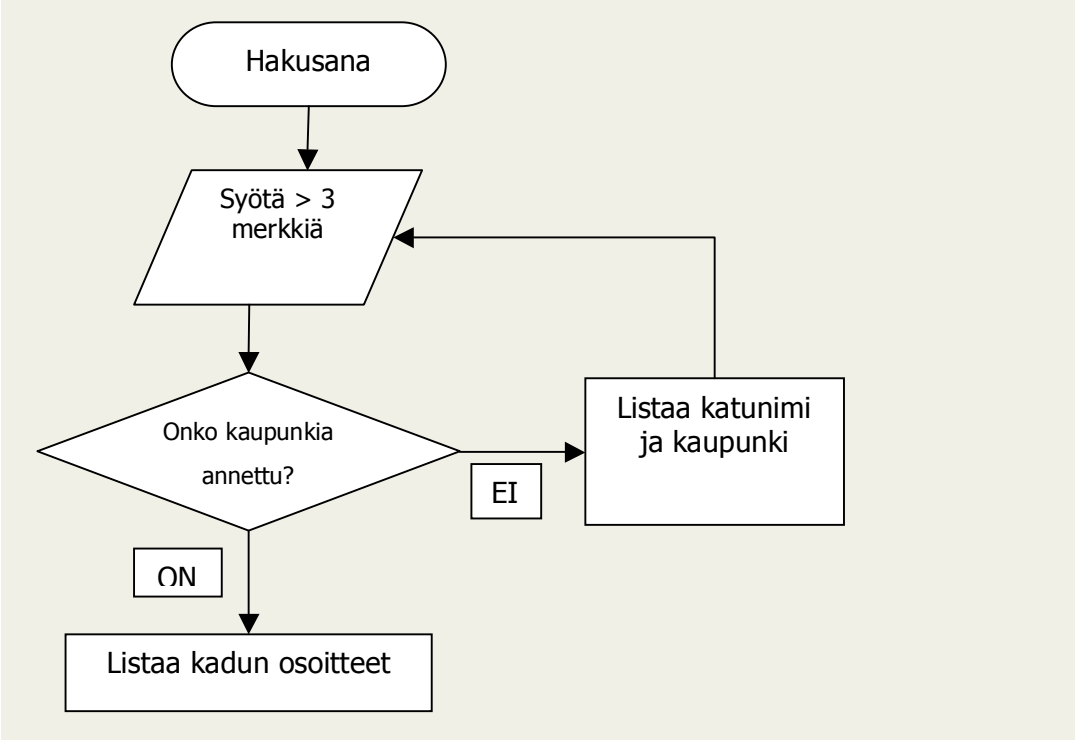
Kuva 5. Pääkaupunkiseudun opaskartta, ilmakuva-karttataso päällä.

6 Osoitehaku

Osoitekannasta tehty vedos kopioidaan ylläpidosta joka arkivuorokausi. Tämä takaa ainakin Helsingin osoitteiden kohdalta ajantasaisuuden päivän tarkkuudella. Web-sovellukset suorittavat tietokantahakuja yleensä joko ODBC-rajapinnan yli tai sitten ohjelmointikielen omilla ajureilla. Pääkaupunkiseudun opaskarttasovellus käyttää tällä hetkellä osoitehaussa PHP:n sisäänrakennettua funktiota odbc_connect. Kesän 2012 aikana tietokantahaku muuttuu käyttämään PDO-tietokantaoliotekniikkaa. Etuna on nopeus, tekniikan mukanaan tuoma tietoturva sekä syötemerkistön esikäsittelyn tarpeen väheneminen.

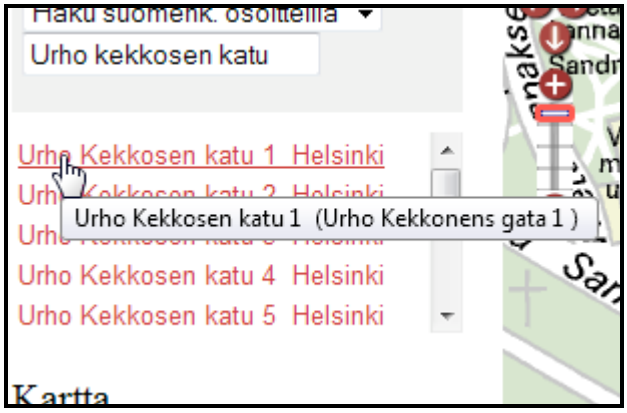
Kuva 6. Osoitehaku. Esimerkki pääkaupunkiseudun kuntien katunimissä esiintyvistä päällekkäisyyksistä.

Kuntakohtainen osoitehaku on yleensä melko helppo toteuttaa yksinkertaisella SQL-kyselyllä ohjelmallisesti. Tilanne mutkistuu huomattavasti kun kyseessä on useamman kunnan yhdistetty aineisto (kuva 6). Periaatteessa jokaisessa kunnassa voi olla saman nimisiä katuja, numerointikäytännöt voivat erota toisistaan ja etenkin kaksikielisillä paikkakunnilla jokaisella kadulla on nimi kummallakin kielellä. Tästä syystä yksinkertaiset SQL-funktiot eivät yleensä riitä halutun hakutuloksen saavuttamiseksi, vaan kyselyihin on pakko tehdä erilaisia erotteluja, esikäsittelyjä ja rekursioita. Pääkaupunkiseudun opaskartassa osoitehaku on toteutettu siten, että haku kaksivaiheinen (kaavio 1). Ensin osoitetta on haettava kadun nimellä, valittava halutun kaupungin katu, ja sen kartta keskitetään osoitenumeron perusteella. Vaihtoehtoisesti haun voisi toteuttaa siten, että jos haetaan koko osoitteella, eri kaupunkien kaikki hakua vastaavat osoitteet näytetään. Pääkaupunkiseudun opaskartassa on käytetty ensin mainittua tapaa lähes koko olemassaolonsa ajan. Ajax-tekniikan käyttö yhdessä prosessoritehon kasvun myötä on mahdollistanut entistä paremman automaattisesti listaavan hakutoiminnan kehittämisen. Nyt haku toteutetaan JavaScriptin onkeyup-tapahtumakäsittelijällä, eli jokaisella näppäinsyötteellä käynnistetään hakuprosessi. Näin tehdyssä toteutuksessa kannattaa rajoittaa hakua siten, että vasta määrätyn merkkimäärän syöttämisen jälkeen aletaan suorittaa tietokantahakua.



Kaavio 0. Osoitehaku vuokaaviona esitettynä.

Hakulomakkeen text-elementti ohjaa Ajax-funktiolla jokaisen käyttäjän tekemän merkkisyötteen PHP-skriptiin. PHP-skriptissä vasta neljännen merkin kohdalta aletaan suorittaa tietokantahakua, muuten tulostetaan vain piste. Mikäli neljännellä merkin syötteellä löytyy koko seudun osoitteistosta vain yksi kadunnimi, tulostetaan saman tien kaikki osoitteet kyseisen kadun osalta. Jos seudulta löytyy useampia nimiä, listataan vain niiden nimet. Osoite- ja kohdehaut on listautuvat sovelluksessa määrämittaiseen div-elementtiin. Listattavat tekstielementit ovat samalla linkkejä, joita osoittamalla suoritetaan jatkotoimet. Linkki on viitattu href-parametrissä tyhjäksi käyttäen ”#”-merkkiä. Linkin toiminnallisuus hoidetaankin linkityksen sijaan JavaScriptin sisäänrakennetulla onclick-tapatumakäsittelijällä. Onclick käynnistää omat JavaScript-funktionsa, jotka hoitavat sekä kartan keskityksen osoitelistalta saatujen koordinaattiarvojen perusteella, että keskusmerkin piirron koordinaattien osoittamaan paikkaan. Ensin mainittu käyttää OpenLayersin karttaolion PanTo-metodia, jälkimmäinen addMarker-metodia. Keskusmerkki on png-kuva, jonka tausta on asetettu läpinäkyväksi ja sisältää punaisen renkaan, jonka keskusta osoittaa haettua paikkaa.



Kuva 7. Kadun osoitteet listattuna

Oheinen koodileike esittää kuvan 7 esittämän linkkirivin sisällön.

```
<tr><td>
<a title="Urho Kekkosen katu 1  (Urho Kekkonens gata 1 )"
onclick="Locate(52001,73408); showPoint(52001,73408,'Urho Kekkosen
katu 1  (Urho Kekkonens gata 1 ), Helsinki')" href="#">Urho Kekkosen
katu 1  Helsinki</a>
</td></tr>
```

7 Karttajulkaisin

Toimivan karttasovelluksen taustalle tarvitaan myös tietolähde, joka tuottaa karttakuvan. Kaupunkimittausosasto käyttää rasterikarttatuotteittensa julkaisemiseen kahta julkaisuohjelmistoa, Intergraphin GeoMedia WebMap-ohjelmistoa ja Bentleyn Geo Web Publisheria. Pääkaupunkiseudun opaskarttapalvelu on alusta saakka käyttänyt rasteriaineiston julkaisussa Bentleyn ohjelmistoa, ja ratkaisu on edelleen tuotantokäytössä myös uudessa opaskarttaratkaisussa.

Näiden paikkatietojulkaisuohjelmistojen rajapintojen toimintaperiaate on samankaltainen kuin WMS-rajapintapalveluissa: HTTP GET -protokollan välityksellä parametriarvoparein kerrotaan palvelun rajapinnalle, mitä karttatuotetta halutaan julkaista, millä tarkkuudella ja miltä alueelta. Karttajulkaisuohjelmien omien rajapintojen selvin ero standardiin WMS-palveluun on, että karttajulkaisuohjelmilla on omat epästandardit rajapinnan komentosyntaksinsa, joka pitää etukäteen tuntea. Jaettavat aineistot on myös etukäteen tunnettava ja niiden nimet ja polut on tiedettävä. Rajapinnat eivät WMS/WFS-palveluiden tapaan ilmaise ominaisuuksistaan

millään erillisellä tätä varten luodulla komennolla. Myöskään koordinaatistot eivät ole vapaasti valittavissa jostain etukäteen määritetystä listasta, vaan pääsääntöisesti toimitaan siinä järjestelmässä kun alkuperäinen aineisto on tallennettu. Bentleyn ja Intergraphin karttapalvelintuotteet on julkaistu jo vuosia ennen WMS- ja WFS-tekniikoiden tuloa markkinoille, joten nämä ominaisuudet on jouduttu rakentamaan niihin jälkikäteen. Vasta viime aikoina ovat sekä Bentley että Intergraph varustaneet ohjelmistojensa uusimmat versiot tuotteeseen valmiiksi integroiduilla WMS- ja WFS-rajapinnoilla.

Vaikka uusimmissa Bentleyn Geo Web Publisherin versioissa on jo WMS-toiminnallisuus mukana, tällä hetkellä tuotantokäytössä olevassa versiossa ei vielä WMS-rajapintaa ole sisäänrakennettuna. OpenLayersin kanssa on kätevintä käyttää WMS:ää karttakutsuissa. Tämä seikka on kuitenkin ollut melko helppo ratkaista itse rakennetulla välittäjäskriptillä (proxy). Se kääntää OpenLayersin WMS-taso-olion luomat standardimuotoiset kutsut suoraan Geo Web Publisherin rajapintamuotoon. Käytännössähän kaikkea ei tarvitse tulkata, lähinnä vain WMS-kutsun BBOX-parametrien arvot sekä karttatason (layer) nimi. BBOX-parametri sisältää julkaistavan karttatiilen vastakkaisten nurkkien koordinaatit. Skripti on kirjoitettu PHP-kielellä.

Bentley Publisher muodostaa julkaistavan karttakuvan siten, että kokonaisen kartan, eli rasterikuvan, mittasuhteet ilmaistaan leveyssuunnassa suhdeluvulla 1 ja korkeussuunnassa kuvan leveys jaettuna kuvan korkeus. Koska kokonaiskuvasta julkaistaan yleensä vain pieni osa, pitää julkaistavan kuvasegmentin sijainti (vasen ylänurkka) ilmaista osamääränä vaakasuunnassa että pystysuunnassa. Ohjelmallisesti laskutoimitus on nopea suorittaa, kunhan ohjelmoija ymmärtää, miten suhdeluvut (Bentleyn terminologiassa REGEN-arvot) lasketaan. Muut arvot jotka vaaditaan, ovat julkaisuformaatti, kuvan pysty- ja vaakamitat kuvapisteinä sekä läpinäkyvyysasetus. Käytännöllisesti katsoen WMS–rajapinta sisältää nämä samat toiminnot, joten tulkkaustoimenpide on yksinkertainen suorittaa. Ratkaisu saattaa olla suorituskyvyltään hieman nopeampi myös uudemman, WMS-kykyisen Geo Web Publisher -version kanssa, koska WMS:n koordinaatistomuunnosprosessointia tai -tarkistusta ei tarvita.

Oheisessa esimerkissä nähdään OpenLayersin muotoilema WMS:n mukainen rajapintakutsu proxyskriptille:

```
map.php?LAYERS=Opaskartta&SERVICE=WMS&VERSION=1.1.1&REQUEST=GetMap&STY
LES=&EXCEPTIONS=application%2Fvnd.ogc.se_inimage&FORMAT=image%2Fjpeg&S
RS=Helsinki&BBOX=51576,73192,52088,73704&WIDTH=256&HEIGHT=256
```

Esimerkki on muotoiltu rivi suoraan web-palvelimen lokitiedostosta. Kuten huomataan, SRS poikkeaa myös standardiksi muodostuneesta EPSG-koodista. Käytännössä SRS-parametriä ei millään tavalla käytetä tämän julkaisutoiminnan yhteydessä, joten parametrin arvo voi olla täysin mielivaltaisesti nimetty. Samoin WMS-kutsussa annetulla karttatiilen HEIGHT- ja WIDTH-parametrilla tai kuvaformaatilla (esimerkissä jpeg) ei ole tässä yhteydessä merkitystä, koska ne on koodattu valmiiksi muunnosskriptiin.

Seuraavaksi edellä kuvattu WMS-kutsu muunnetaan Geo Web Publisherin julkaisukomennoksi. Tällaisena kutsu lähtee WMS–GWP-proxyskriptistä Geo Web Publisher -palvelimeen:

```
?fif=kartat/pks/vvj/opaskartta/oppks_2m.pss&obj=hip,1.3&wid=256&hei=25
6&Fit=5&interpol=average&RGN=0.70217142857143,0.75131428571429,0.01462
8571428571,0.014628571428571&cvt=png
```

Kuva 8. Geo Web Publisherin tuottama karttatiili, 256 x 256 kuvapistettä

Geo Web Publisherinkin julkaisukomento annetaan WMS:n tapaan http-GET -kutsuna, joten palvelin voi olla siis missä hyvänsä verkkoyhteyden saavutettavissa.

OpenLayers lataa oletusarvoisesti karttanäytön 256 x 256 kuvapisteen kokoisista palasista, eli tiilistä (kuva 8). Kokonaisen karttakuvan muodostaminen pienistä paloista

on ilmeisen hyvä ratkaisu, sillä karttapalvelinten puolesta julkaisunopeus monelle pienelle karttapalalle on parempi kuin yhdellä suurella. Toinen etu on parempi käyttäjäkokemus. Käyttäjän näkökulmasta lienee miellyttävämpää, kun kartta viipymättä alkaa koostua näyttöön ja latautuneet tiilet ovat heti luettavissa, kuin että joutuisi odottamaan koko kartan ilmestymistä kerralla.

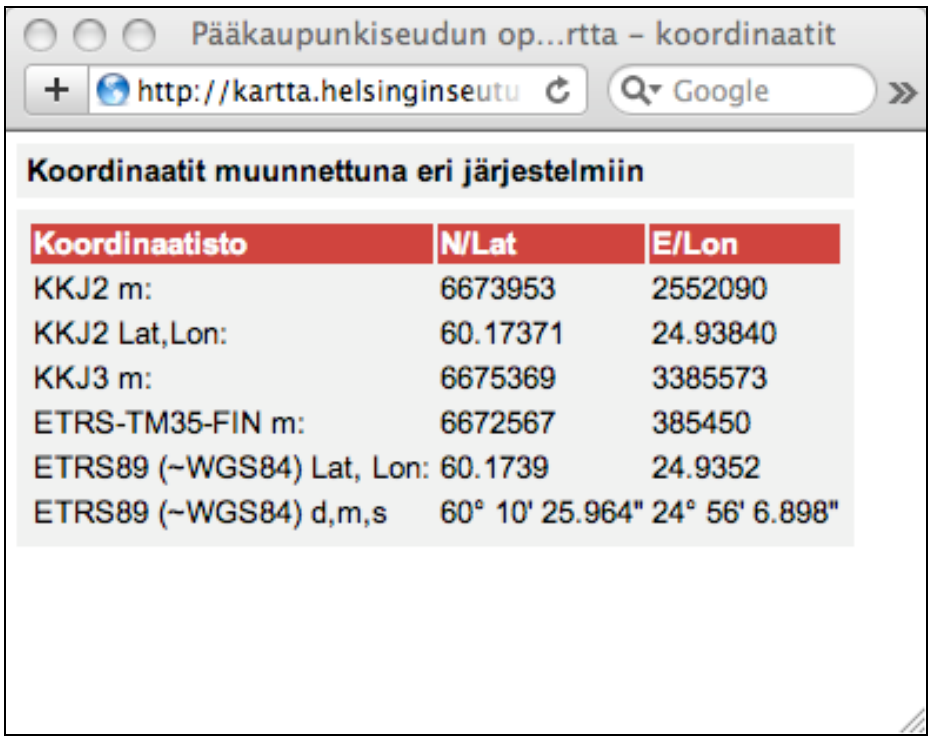
8 Muut toiminnot

8.1 Mittaustoiminnot

OpenLayers on viimeisissä versioissaan sisällyttänyt toimintovalikoimaansa matkanmittauksen. Matkanmittaus toimii useimpien vastaavien kartta- ja paikkatietosovellusten tapaan hiirellä osoittamalla siten, että muodostuu murtoviiva. Viivan kulmapisteiden välisten janojen yhteenlaskettu pituus näytetään mittausnäytössä. Pääkaupunkiseudun opaskartan tapauksessa mittausnäyttö on sijoitettu mittaustyökalun käynnistävän radiopainikevalinnan alle div-elementtinä. Mittaustyökalun käyttö muuttaa samalla hiiren käyttöön varattua tilaa. Normaalissa tilassahan hiirellä vieritetään karttaa, mutta mittaustilassa hiirtä käytetään mittauksen suorittamiseen. Mittaustilassa karttaa voidaan vierittää kartan siirto/zoomauskontrollista, tietokoneen näppäimistön nuolinäppäimillä taikka indeksikarttaa hyväksikäyttäen. Mittaus lopetetaan hiiren kaksoispainalluksella tai lopettamalla mittaustila.

Mittaustoimintoihin kuuluu myös koordinaattimittaus. Tämä toiminto ei ole toistaiseksi kovin näkyvästi esillä, sillä sen olemassaoloa ei huomaa käyttöliittymästä. Toiminto on siis löydettävissä vain käyttöohjetta lukemalla. Mittaustyökalun vieressä on kyllä linkki käyttöohjeen mittaustoimintoja käsittelevään kohtaan, joten kohtuuttoman hankalaa sen löytäminen ei liene. Koordinaattimittaus ei vaadi erityistä mittaustilaa, vaan se on aina otettavissa käyttöön painamalla näppäimistöltä samanaikaisesti control-näppäintä ja osoittamalla hiirellä haluttua kohtaa kartalla. Koordinaattimittaus avaa oman erillisen ponnahdusikkunansa (kuva 9), johon listautuu koordinaatteja eri järjestelmissä. Toistaiseksi koordinaattijärjestelmiä esitetään valtiollisissa järjestelmissä, eli vanhoissa kartastokoordinaattijärjestelmissä sekä EUREF-FIN-

yhtenäisjärjestelmässä (ETRS-TM35FIN), sekä metrisessä suorakulmaisessa koordinaatistossa että geodeettisesti asteina. Koska EUREF-FIN:in osalta jälkimmäiset arvot ovat myös likimääräisiä WGS84–järjestelmän koordinaatteja, näin saadut arvot sopivat myös esimerkiksi navigaattorikäyttöön. Ponnahdusikkunan lisäksi viimeksi mitatut ETRS-TM35 – ja likimääräiset WGS84-koordinaattiarvot jäävät osoite- ja kohdehakukentän alapuolelle näkyviin siihen saakka kunnes osoite- ja kohdehaulla tehdään uusia hakuja. Koordinaattinäyttö on tehty muunnosfunktioilla, jotka on tehty erillisessä projektissaan jo vuonna 2007. Muunnos suoritetaan järjestelmän VVJ-koordinaateista siten, että ensin suoritetaan helmert-muunnos VVJ-KKJ2. Tämän jälkeen voidaan suorittaa sekä kaistamuunnos (KKJ2-KKJ3) että koordinaattimuunnos eri datumien välillä (KKJ2 - ETRS-TM35FIN) JHS 152 liitteen 5 mukaisesti. Muunnosfunktiot on Kaupunkimittausosastolla toteutettu eri ohjelmointikielillä. Tässä sovelluksessa työ on tehty PHP-kielellä. Tarkistusten perusteella funktioilla päästään parhaimmillaan desimetritarkkuuksiin ja heikoimmillaankin noin vajaan metrin tarkkuuteen. Tarkkuus riippuu muunnosten suunnasta ja muunnettavista koordinaatistoista. Pääkaupunkiseudun opaskarttasovelluksessa käytännön erotustarkkuuden toleranssiksi riittää metritarkkuus, joten näiden muunnosten tarkkuuden on katsottu riittävän tähän tarkoitukseen erittäin hyvin.



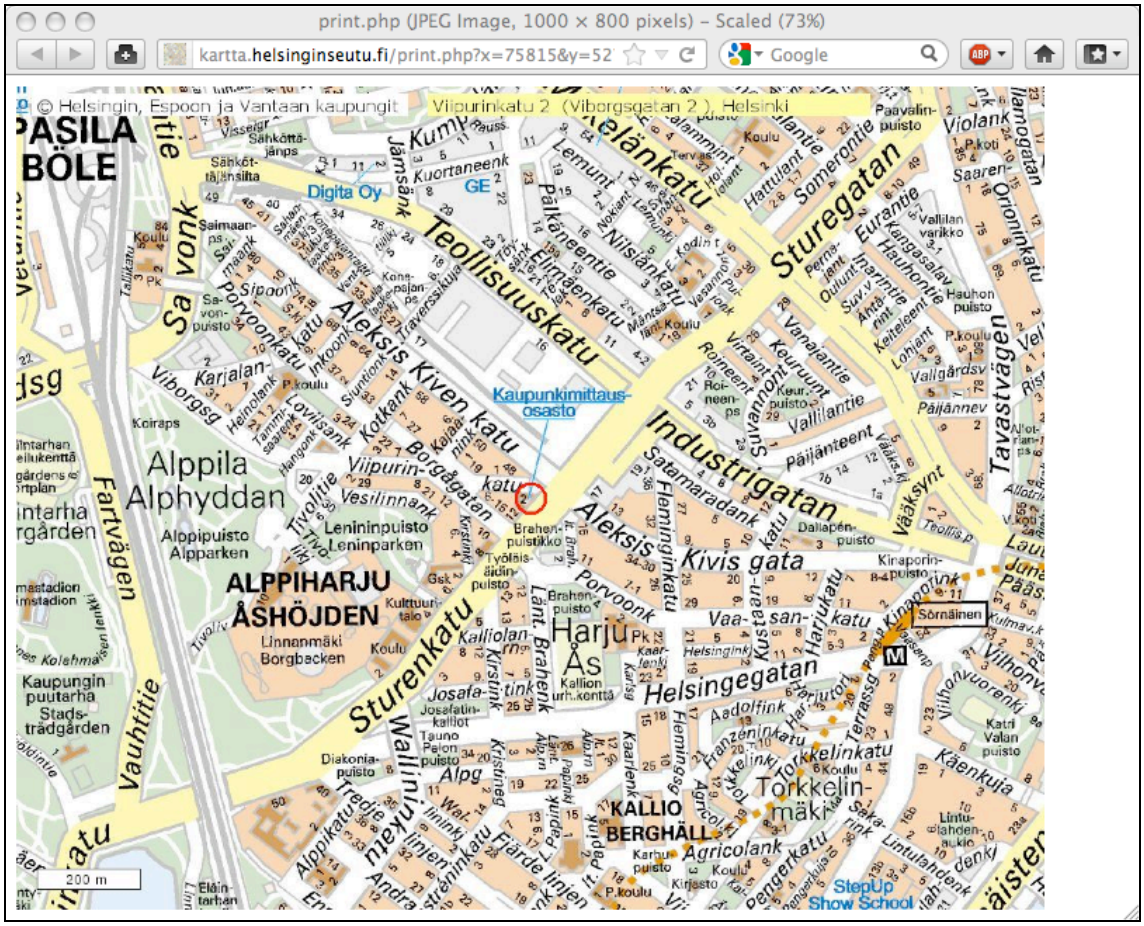
Kuva 9. Koordinaattinäytön ponnahdusikkuna

8.2 Tulostus

Paperille tulostaminen vapaasti näytön mukaan skaalattavalla kartalla on haasteellista, koska paperiarkki on yleensä määrämuotoinen, mutta karttasovelluksen karttaikkuna saattaa olla muodoltaan lähes mitä tahansa. Lisähaasteen tuottaa se seikka, että eri selaimet saattavat tulkita esimerkiksi CSS-asemointia hieman eri tavalla. Tämän vuoksi esimerkiksi keskusmerkin asettelu voi olla viisainta suorittaa ”polttamalla” se suoraan tulostekuvaan. Kahden tai useamman rasterikuvan yhdistäminen onnistuu PHP:n GD2-kuvankäsittelykirjaston avulla.

Tulostustoiminto on tässä sovelluksessa toteutettu siten, että muotoillun tulostussivun sijaan rakennetaankin OpenLayers-näkymän antamien koordinaattiarvojen ja karttamoottoripalvelimesta suoraan saadun rasterikartan avulla yksi kokonainen yhdistelty rasterikuva, joka avataan omaan ponnahdusikkunaansa (kuva 10). Toteutuksessa käytetään hyväksi edellä mainittua GD2-kirjastoa. Ratkaisussa saavutetaan se etu, että selaimesta ja ympäristöstä riippumatta asiakas saa aina samanlaisen tulosteen. Lopputulos riippuu näin ainoastaan tulostuslaitteistosta.

Sen sijaan, että tulostuskuva olisi alaltaan koko näytössä olevan kartan levyinen, se skaalataan keskipisteen mukaan noin A4-kokoon siten, että karttanäytön keskipiste on tulosteen keskellä. Kuvan yläosaan liitetään aineistojen tekijänoikeusmerkinnät. Jos on haettu osoitetta, myös löytynyt osoite kirjoitetaan kuvaan. Koska tulosteessa halutaan käyttää tiettyä tekstityyppiä, kyseisen tyypin TrueType-kirjasintiedosto voidaan lukea palvelimen hakemistosta ja käyttää sitä. Tulosteen vasempaan alareunaan rakennetaan mittakaavajana, joka pituus mukautuu tulosteen koon ja karttatyypin mukaan. Kaikkien tekstien taustalle, ennen itse tekstin lisäämistä, piirretään tekstin kokoa mukaileva suorakaidelaatikko.



Kuva 10. Tulostussivu opaskarttanäkymästä, kun on haettu osoitetta

8.3 Palautetoiminto

OpenLayersin toimintovalikoimaan ei kuulu palautetoimintoa, mutta vanhassa Pääkaupunkiseudun opaskartassa on hyvin toimiva oma toteutus, jonka koodipohjaa on käytetty uudessa sovelluksessa suoraan. Palautelomake on hyvin yksinkertainen html-lomakesivu, joka POST-protokollaa käyttäen lähettää asiakkaan palauteviestin postinlähetysskriptin kautta palautteita käsittelevän henkilön hallinnoimaan sähköpostilaatikkoon. Poiketen lähes kaikesta muusta Pääkaupunkiseudun opaskartan sisältämistä palvelimella ajettavista skripteistä, postinlähetysskripti on kirjoitettu ASP-VBScript-kielellä. Tämä johtuu siis siitä, että toiminto on siirretty lähes sellaisenaan vanhasta sovelluksesta. Periaatteessa myös postinlähetystoiminto oltaisi voitu toteuttaa PHP-kielellä yhtenäisyyden vuoksi, koska Windows-palvelimen postipalvelukirjastot voidaan ottaa komponenttiolioina (COM-object) käyttöön myös PHP-kielessä. Koska

sovellusta ajetaan IIS-palvelimella ja näin ollen vbscript-skriptit ovat suoraan ajokelpoisia, kielimuutosta ei ole ainakaan toistaiseksi tarpeellista tehdä.

9 Suorituskyky ja ylläpidettävyys

Kuten aiemmin todettiin, suorituskykyä voidaan hieman parantaa tiivistämällä OpenLayersin kirjastohakemiston tiedostot yhdeksi suureksi tiedostoksi, josta on siistitty ylimääräiset merkit kuten rivinvaihdot, ylimääräiset välilyönnit, sarkainsisennykset ja ennen kaikkea kommentit. Tiivistykseen voi käyttää jakelupaketin mukana tulevaa python-kielistä skriptiä. Skriptin ajo vaatii, että Python-ajoympäristö on asennettuna työasemaan. Pääkaupunkiseudun opaskartan kirjasto on tiivistetty juuri kyseisellä menetelmällä. Tällä hetkellä tiivistetty kirjastotiedosto on alle 400 kilotavun suuruinen. Version 2.11 valmiiksi tiivistetty versio, jossa on mukana koko toiminnallisuus, on kooltaan hieman alle megatavun kokoinen.

Toinen tapa suorittaa tiivistys on käyttää OpenLayerer-verkkosovellusta. Se saadaan osoitteesta

`http://openlayerer.appspot.com/`

Siinä tarvittavat kirjaston luokat voidaan valita rastittamalla. Sovellus tietää myös kirjastojen riippuvuudet, eli se ottaa mukaan automaattisesti myös ne kirjastot, jotka vaaditaan mukaan riippuvuussuhteiden perusteella. Lopuksi sopivan yhdistelmän valittuaan käyttäjä voi nappia painamalla aloittaa prosessin, jossa palvelin paketoi ja yhdistelee valitut asiat kirjastoista yhdeksi tiiviiksi tiedostoksi ja sen valmiiksi saatuaan tarjoaa sen ladattavaksi. Valitun paketointikonfiguraation voi myös tallettaa tekstitiedostoksi ja ladata omaan koneeseen myöhempää tarvetta varten. Uuden OpenLayers–version ilmestyttyä voi konfiguraatiotiedoston ladata taas Openlayereriin ja suorittaa uuden tiivistyksen. Openlayerer tekee tällä hetkellä paketoinnin 2.9.1-versiosta, joten tuoreinta vakaata versiota ei toistaiseksi tueta.

Hyvään käytäntöön kuuluu, että palvelimelle ei viedä kuin ne tiedostot, joita sovellus vaatii. OpenLayersin koko asennuspakettia ei tietoturvasyistäkään kannata kopioida palvelimelle, vaan ainoastaan ne osat ja kirjastot, joita tarvitaan.

10 Tietoturva

Karttasovelluksen kannalta mahdolliset tietoturva-aukot liittyvät lähinnä lomakesovellusosiin. Tietoturvan kohtuulliselle tasolle saattaminen edellyttää palvelinkäyttöjärjestelmän päivitysten ajan tasalla pitämisen, levyoikeuksien määrittämisen, palvelinohjelmiston asetusten tietoturvallisiksi säätämisen ja tietokantayhteyksien oikeuksien rajoittamisen. Jos Windows Server-palvelimessa käytetään muita palvelimella ajettavia skriptikieliä kuin ASP VBScript, nämä skriptiajoympäristöt on myös pidettävä päivitettyinä ja huolehtia niiden asetuksista siten, että käyttö on turvallista. Pääkaupunkiseudun opaskarttasovellus käyttää palvelimessa ajettavia PHP-skriptejä ja ODBC-rajapintaa Microsoft Access -tietokantaan. PHP-ohjelmiston ajan tasalla pitäminen ja ODBC-ajurin asettaminen vain lukutilaan saattaa perustason tietoturvalle. Näiden perusasioiden jälkeen tietoturvan taso riippuu siitä, miten koodin laatija rakentaa ohjelmiston. PHP:ssä on oletusasetuksena esimerkiksi parametrikäsittely, register_globals, poistettuna. Skriptitiedostot, jotka sisältävät tietokantayhteyden muodostamisia, sisältävät usein myös parametreina annettavia arvoja hakutoiminnallisuutta varten. Osoitehakuskriptit, joissa käyttäjä antaa suoraan hakusanan, parsii skriptin käyttämän hakusanan suoraan SQL-lauseeseen. Koska hakusanaksi voidaan antaa mielivaltainen merkkijono, annetaan myös potentiaaliselle hyökkääjälle mahdollisuus sisällyttää SQL-lausetta jatkavaa koodia sekaan (SQL-injektio), jos hakusanasyötettä ei ajeta tarkistusrutiinin läpi. Tarkistusrutiinina on Pääkaupunkiseudun opaskartassa oma funktionsa, joka analysoi osoitesyötteen ja riisuu siitä ylimääräiset ja eritoten tietoturvan kannalta arveluttavat merkit.

11 Sovelluksen jatkokehitys ja tulevat muutokset

11.1 Koordinaatisto

Sisäisesti suurin vuoden 2012 aikana tuleva muutos lienee sovelluksen sisäisen koordinaattijärjestelmän muuttaminen. Helsingin, Espoon ja Vantaan kaikkien ylläpidettävien paikkatietoaineistojen yhteiseksi koordinaattijärjestelmäksi otetaan vuoden 2012 loppuun mennessä ETRS-GK25. Tämä aiheuttaa väkisinkin useampia

muutoksia sovelluksessa. Tärkeintä on, että muutokset voidaan hoitaa hallitusti, siksi ympäristö pitää dokumentoida riittävällä tarkkuudella.

11.2 Karttarajapinta

Koska OpenLayers tukee WMS-rajapintaa suoraan, koordinaattijärjestelmän muutos avaa myös mahdollisuuden käyttää suoraan EPSG-koodilla varustetulla koordinaattijärjestelmällä tarjottavaa kartta-aineistoa. Riippuu siis WMS:n suorituskyvystä, kuinka käyttökelpoiseksi sen käyttö osoittautuu. Suoran WMS-saannin käyttö myös vapauttaa vain yhden julkaisukoneen rajoituksesta. Tarvittaessa julkaisukone voidaan hyvin nopeasti vaihtaa muuttamalla sovelluksen asetuksiin julkaisevan palvelinkoneen nimi, mikäli toisessa WMS-palvelimessa ilmenee käyttökatkoja tai häiriöitä.

11.3 Osoiterajapinta

Keväällä 2012 Kaupunkimittausosasto julkaisi uuden osoiterajapinnan. Pääkaupunkiseudun opaskarttasovelluksen osoitehaku voitaisi toteuttaa rajapinnan kautta, koska se tuottaa xml-rakenteista syötedataa hakusanan perusteella. Lisäksi koordinaatteja tarjoillaan useammassa järjestelmässä, myös ETRS-GK25:ssä. Rajapinnasta on tätä kirjoitettaessa olemassa beetaversio, joka täydentyy kesään mennessä tuotantokäyttöön sopivaksi versioksi. Mikäli Pääkaupunkiseudun opaskartta käyttää jo syksyllä rajapintaa osoitehakujensa kanssa, vain POI-kohteet, kuten museot, sairaalat, kirkot ja liikennepaikat, joudutaan hakemaan edelleen ODBC-haun avulla. Tosin nekin voidaan liittää osoiterajapintaan tai vastaavanlaiseen omaan rajapintaansa.

12 Yleisöpalaute

Pääkaupunkiseudun opaskartan yleisöpalaute on ollut vaihtelevaa. Yleisö on antanut sekä kiitoksia että negatiivista palautetta. Kritiikkiä on saatu mm. matkanmittaustoiminnon puutteista. Myös osoitehakuun toivotaan suoraa hakua kokonaisella osoitteella. Tämä toiminnallisuuden laajennus onkin työn alla tätä

kirjoittaessa. Toisinaan palautteesta on hyvin vaikea tulkita, mitä asiakas sovelluksessa kritisoi. Valtaosa palautteesta kuitenkin liittyy itse aineistoihin, niiden ajantasaisuuteen sekä puutteellisuuksiin. Joskus palautteesta selvästi huomaa, että asiakas ei ole lukenut käyttöohjetta ennen palautteen lähettämistä. On kyllä totta, että ihanteellisen web-sovelluksen pitäisi toimia täysin intuitiivisesti ilman hankalaa käyttöohjeistuksen läpikäyntiä. Kaikkia käyttäjiä miellyttävää ja universaalisti ymmärrettävää web-sovellusta hyvin harva on kyennyt luomaan. Kaikenlainen palaute koetaan kuitenkin tervetulleeksi ja arvokkaaksi, jotta parasta mahdollista sovellusta kohden voitaisiin pyrkiä.

13 Muita OpenLayers-sovelluksia

13.1 Sovellukset Helsingin kaupungilla

13.1.1 Pääkaupunkiseudun ulkoilukartta

Kaupunkimittausosasto on Liikuntaviraston kanssa yhteistyönä kehittänyt Pääkaupunkiseudun ulkoilukarttasovelluksen (http://www.ulkoilukartta.fi). Vanhastaan paperiversiona jo pitkään olemassa ollut karttatuote on ollut sangen suosittu ilmaisjakelutuote. 2000-luvun ensimmäisen vuosikymmenen puolivälissä kartasta tehtiin internet-versio perustuen silloisen Centroidin, nykyisen SITO:n SpatialWeb Karttapaikka -tuotteeseen. Karttapaikan tuen loputtua vanhat karttapaikkasovellukset päätettiin joko lakkauttaa tai toteuttaa muilla tavoin. Ulkoilukarttaan päätettiin vuoden 2010 aikana tehdä OpenLayers-ratkaisu omin voimin. Alkuvaiheessa sovelluksen toiminnallisuus rajoittuu vanhaan SpatialWeb Karttapaikkaa vastaaviin toimintoihin. Jatkossa sovellusta jatkokehitetään joko konsulttivoimin tai haetaan kokonaisratkaisua ostopalveluna.

13.1.2 WLAN–kartta

Centroid Oy:n vanhat karttapaikka-sovellukset korvanneista OpenLayers-ratkaisuista ulkoilukartan lisäksi elämään jäi vain Helsingin langattomia verkkopisteitä ja niiden sijainteja esittänyt WLAN-kartta (http://ptp.hel.fi/wlan). Kaupunkimittausosaston

resurssit WLAN-pistekohteiden ja niiden attribuuttitietojen ylläpitoon ovat valitettavasti rajalliset. Lisäksi palvelun käyttäjämäärät ovat tällä hetkellä melko vähäiset. Kaupungin omien WLAN-pisteiden sijainnit ovat saatavissa nykyään myös Palvelukartasta (http://www.hel.fi/palvelukartta), joten palvelun jatkuvuus on tästä päätellen melko epävarma.

13.1.3 Karttalaatikko-projekti

Vuonna 2007 aloitettiin yhteistyössä Helsingin yliopiston ja Opetusviraston kanssa projekti, jossa tarkoituksena oli luoda täysin avoimeen lähdekoodiin perustuva ympäristö opetustarkoituksiin (http://karttalaatikko.hel.fi/kallahti). Tarkoituksena oli opettaa koululaisia käyttämään vuorovaikutteisia paikkatietosovelluksia webissä. Testikouluna toimi Kallahden ala-asteen koulu. Projekti tuotti Linux/Apache/Postgres/OpenLayers-sovelluksen, jota on käytetty aktiivisesti oppilaiden ja opettajien toimesta vuoteen 2010 saakka. Tästä eteenpäin käytöstä on vain hajahavaintoja. Karttalaatikkoprojektissa mukana on ollut myös Helsingin kaupunginosayhdistys HELKA sekä Helsingin kaupunginkirjasto Tarinoiden Helsinki -projektin osalta. Myös joitakin ulkopuolisia hallintokuntien edustajia on ollut mukana toiminnassa. Kaupunkimittausosaston osuus on ollut järjestää projektille palvelin ja toimintaympäristö ja toimittaa tarvittavat tausta-aineistot. Koulujen paikkatietosovellukset teki ulkopuolinen projektityöntekijä yhteistyössä Kaupunkimittausosaston kanssa. Toimintaa aiotaan jatkaa edelleen.

13.1.4 Karttakioski

Vuoden 2012 alussa Helsingin rakennusvalvontavirasto tiedusteli seuraajaa Kaupunkimittausosaston intranetistä aiemmin löytyneelle kiinteistökarttasovellukselle. Vanha kiinteistökartta oli ollut yksinkertainen ASP/HTML-sovellus, joka pohjautui vanhaan pääkaupunkiseudun opaskarttasovelluksen koodiin. Helsingin uuden paikkatietopalvelun myötä intranet-kiinteistökartta tulikin tarpeettomaksi. Koska vanha sovellus oli todettu käyttökelpoiseksi Rakennusvalvontaviraston yleisöpäätteellä, tehtiin tätä korvaamaan Karttakioski. Käytännössä toteutus oli hyvin nopea, koska tarvittiin vain helposti web-selaimella toimiva, mieluummin koko näyttöalan hyväksikäyttävä

sovellus näyttämään kiinteistökarttaa. Mitään toimintoja ei sovellukseen juuri tarvinnut sisällyttää. Osoitehaku kuitenkin lisättiin, koska koodi siihen löytyi valmiina. Työaikaa ei sovelluksen koodaamiseen tarvittu juuri paria tuntia enempää.

13.2 OpenLayers–sovellukset Suomessa

OpenLayers- ja Ext Js 4 -kirjastot ovat Suomen julkishallinnollisen paikkatietoportaalin, Paikkatietoikkunan, karttasovelluksen perustana (http://www.paikkatietoikkuna.fi/web/fi/kartta).
Karttasovelluksen (Karttaikkuna) ulkoasu poikkeaa huomattavasti OpenLayersin perusnäkymästä ja toiminnallisuutta on kehitetty huomattavan monipuoliseksi. Paikkatietoikkunan koko lähdekoodi on julkaistu vapaasti ladattavaksi ja käytettäväksi GPL v3 -lisenssillä.

13.3 OpenLayers-sovellukset muualla maailmassa

Maailmalla lienee lukuisia kunnallisia, jopa valtiollisiakin paikkatietopalveluita joita on toteutettu OpenLayers-ratkaisuin. Seuraavassa on lueteltu esimerkillisesti toteutettuja karttasovelluksia, joissa on käytetty runsaasti sekä omaa koodityötä että hyödynnetty tehokkaasti OpenLayersin ominaisuuksia.

Ehkä maailmanlaajuisesti tunnetuin OpenLayers-alustalle tehty sovellus on OpenStreetMap (OSM) (http://www.openstreetmap.org/), joka on kansalaisten vapaaehtoisvoimin GPS-perustaisesti kartoitus- ja paikkatietoa keräävä avoin kartta kattaen koko maailman.

Veloland on sveitsiläinen pyöräilykarttasovellus (http://map.veloland.ch/), josta löytyy myös pistetietoa turistikohteista. Erikoisuutena sovellukseen on toteutettu liukusäätimellä toimiva himmentävä karttatasovalinta.

IDEC WMS Viewer on Katalonian aluehallinnollisen maanmittauslaitoksen (IDEC) karttasovellus (http://mapaidec.icc.cat/idecwebservices/mapawms/index.jsp?lang=en_UK).

Palvelussa on erittäin kattava aineistotarjonta. Sovellukseen voi lisätä jopa omia WMS-tasoja sekä siitä voi ladata dataa KML-rajapinnan kautta. Myös GeoRSS-syötettä tarjotaan kattaen erilaisia Katalonian alueen kuntien teemoja.

14 OpenLayersin tulevaisuus

14.1 HTML5

OpenLayers-sovellukset ovat tähän saakka käytännössä toimineet HTML 4.0- tai XHMTL 1.0 -versioihin rakennetuissa sivustoissa. Periaatteessa OpenLayersin kirjaston toiminta ei riipu niinkään mistään merkintäkielen versiosta, vaan pikemminkin selaimen JavaScript- ja Ajax-tuesta sekä CSS-muotoilujen onnistuneesta käytöstä. HTML-merkintäkielen uusin versio, HTML5, on vielä tätä kirjoitettaessa luonnosvaiheessa [7]. Uusimmissa web-selaimissa luonnosversioiden tuki kuitenkin jo on laajasti käytössä. Kokeiluja HTML5-sivuun tehdystä OpenLayers-sovelluksesta on tehty. Saksalaisen Tobias Sauerweinin Kaiserslauternin ammattikorkeakoulussa tekemä opinnäytetyö käsittelee HTML5:n käyttöä OpenLayers-sovelluksen pohjana. Päätelmässään Sauerwein toteaa, ettei HTML5 varsinaisesti tuo uutta OpenLayersin peruskäyttöön, mutta uuden canvas-elementin avulla voidaan kyllä visualisoida uudella tavalla maantieteellistä aineistoa esimerkiksi alueellisten analyysien muodossa. Tulevaisuudessa saatetaan HTML5:n ominaisuuksia käyttää hyväksi myös laitteistokiihdytettynä, mikä tähän saakka on aina vaatinut ulkoisten lisäosien käyttöä. [8, s. 64]

14.2 Kehitys ja integraatio

JavaScriptiin perustuvat ohjelmointitekniikat ovat pitäneet pintansa erittäin hyvin ajan saatossa. Selainvalmistajat kilpailevat JavaScriptin ajamisen nopeudessa ja CSS:n tulkkaamisen oikeellisuudessa. Tämä luonnollisesti tukee ajatusta, että web-sovelluksia kannattaa ennen kaikkea laatia JavaScriptin, CSS:n ja Ajax-tekniikan ympärille.

Avoin ohjelmointiarkkitehtuuri mahdollistaa myös lähes rajattomat integrointimahdollisuudet. Integroiminen avoimiin sisällönhallintajärjestelmiin tai kehitysympäristöihin on jo nyt todellisuutta, esimerkkinä Drupal sekä Plone.

Avoimen lähdekoodin ohjelmistojen tulo markkinoille on muuttanut voimakkaasti asetelmia tietotekniikkamarkkinoilla. Aiemmin useimmiten täysin suljetuissa koodiympäristöissä toimineet kehittäjät ja konsultit ovat saattaneet laajentaa toimintaansa siten, että avoimienkin järjestelmien tukea tarjotaan. Nykyään voidaan tarjota jopa ratkaisuja, jotka on toteutettu kokonaan tai suurimmaksi osaksi avointen sovellusten pohjalta. Avoimen lähdekoodin paikkatietosovelluksia on viime aikoina ilmaantunut runsaasti. Nykyään on mahdollista jopa luoda kokonaisia paikkatiedon tuotantoympäristöjä avoimilla ohjelmistoilla. Yksi avoimien ohjelmistojen vahvuuksista ilmaisuuden lisäksi on se, että niistä tehdään usein käännökset suosituimmille käyttöjärjestelmille. Jollei valmiita käännöksiä löydy, niiden lähdekoodeista voi ainakin yrittää kääntää ajokelpoisia ohjelmia mihin hyvänsä käyttöjärjestelmään ja suoritinarkkitehtuuriin, jos tahtoa ja taitoa riittää. Se, voidaanko julkisen organisaation paikkatietojärjestelmää perustaa kokonaan avoimen lähdekoodin ympäristöihin, saattaa olla hankalaa vielä tänään, mutta ainakin osajärjestelmiä voidaan hyvin toteuttaa vaikkapa juuri OpenLayersin, PostGisin, MapServerin tai GeoServerin kaltaisilla sovelluksilla.

14.3 OpenLayers, versio 3.0

OpenLayersin uusi versiosukupolvi on toistaiseksi vasta määrittelyasteella. Siihen on ideoitu paitsi runsaasti uusia toimintoja, myös uusi tapa käsitellä ja luoda olioita. Suunnitelman mukaan nykyisestä tavasta hoitaa luokkaperintä luovutaan.

Uusi versio taipuu myös paremmin mobiilipäätelaitteille, eli tämän myötä toiminta sekä älypuhelimissa että tablettimikroissa ja niiden kosketusnäytöissä paranee.

Kesäkuussa 2010 Tim Schaub, OpenGeon teknologiajohtaja ja yksi OpenLayersin aktiivisimmista kehittäjistä, luetteli OpenLayers-blogissa tavoitteita versiolle 3.0. Ehdotetut muutokset koskevat paitsi ohjelmointiympäristöä sekä suorituskykyä, mutta myös käyttöliittymän tekoon luvataan parannuksia. Riippuvuudet eri kirjastojen välillä

pyritään järjestämään siten, että koodia, jota ei sovelluksessa käytetä, ei myöskään ladata. Pyrkimyksenä on tarjota mahdollisuus käyttää vaikkapa OpenLayersin oman koodin sijaan ulkoista koodia ja sen myötä sivuuttaa kyseiset OpenLayersin koodilohkot suorituskyvyn parantamiseksi. Käyttöliittymän puolella uudistukset koskevat vierityksen virtaviivaistamista sekä mahdollisuutta lisätä karttakohteisiin tapahtumakäsittelijöitä. Tämä mahdollistaisi esimerkiksi raporttien tai älykkäiden työkaluvihjeiden luomisen. Käyttöliittymäkontrollien osalta parannetaan myös mahdollisuutta käyttää ulkoisia kontrollielementtejä karttanäytön ja tasojen käsittelyyn. Myös mobiilikäyttöliittymien tekemistä helpotetaan. Tavoitteiden perusteella OpenLayersin käyttöliittymä kehittyy vähitellen täysverisen paikkatieto-ohjelmiston käyttöliittymän suuntaan. Mikäli kehitys jatkuu onnistuneesti ja esimerkiksi HTML5:n ja OpenLayersin yhdistäminen lunastaa odotukset, voidaan hyvällä syyllä uskoa, että tulevaisuudessa nähdään vapaita web-pohjaisia paikkatietosovelluksia, jotka ovat täysin kilpailukykyisiä kaupallisten tuotteiden kanssa.

Toistaiseksi kehitystyön alla on vielä versio 2.13, joka tulee olemaan tämänhetkisen tiedon mukaan viimeinen 2. sarjan julkaistu versio.

15 Pääkaupunkiseudun opaskarttasovelluksen tulevaisuus

Kuten tämän insinöörityön alkusanoissa jo todettiin, Pääkaupunkiseudun opaskartta oli ensimmäinen laajempaa suosiota saavuttanut seudullinen opaskarttapalvelu Suomessa. Koska Helsingin kaupungin ylläpitämät opaskarttasovellukset ovat palvelinliikenteen tunnuslukujen perusteella edelleen melko suosittuja sovelluksia, voidaan otaksua, että kansalaiset hyötyvät niistä ja että niitä kannattaa edelleen ylläpitää ja kehittää. Vaikka kaupalliselta pohjalta toimivat yksityisen sektorin kilpailijat ottanevat käyttöliittymien suhteen kehityksen aallonharjan haltuunsa, on tämän palvelun vahvuuksina erityisesti ajantasaisuus sekä aineiston laatu. Lisäksi melko nopea reagointikyky yleisöpalautteeseen antaa oman lisänsä sovelluksen käyttökelpoisuuteen ja palveluvalmiuteen.

16 Päätössanat ja pohdintaa

Pääkaupunkiseudun opaskarttasovelluksen alustusskripti, XHTML-koodi sekä CSS-tyylimääritykset kommentoituna ovat kokonaisuudessaan tämän insinöörityön liitteissä lähempää tarkastelua silmällä pitäen. Kaikkea sovelluksen koodia en ole katsonut kuitenkaan järkeväksi liittää paitsi tilakysymysten vuoksi, mutta myös koska tarkoituksena on viime kädessä pitää tämä dokumentti olemassa olevan ohjelmakoodin tulkintaan soveltuvana käsikirjana. Mikäli koodia on tarkoitus opiskella taikka muokata ylläpitäjän ominaisuudessa, pitää siis koko koodi ottaa käsiteltäväkseen editorissa tai kehitysympäristössä ja tutkia sitä tämän dokumentin välittämän tiedon valossa.

Kokonaisuutena Pääkaupunkiseudun opaskartan ja OpenLayers-ympäristön kanssa on ollut mielenkiintoista toimia. Sen myötä olen joutunut ensimmäistä kertaa syvällisemmin perehtymään JavaScript-ohjelmointiin. Erityisesti muista kielistä poikkeavan olioajattelunsa vuoksi kielen omaksumisessa on ollut omat hankaluutensa, mutta jos uuteen ajatteluun ja toimintatapaan oppii, ymmärtää myös paremmin syyt siihen, miksi kyseinen kieli on saavuttanut käytännön standardiaseman web-maailmassa. Aika näyttää myös, säilyttääkö OpenLayers johtoaseman avoimen lähdekoodin karttakäyttöliittymien kehityksessä. Joka tapauksessa HTML5, JavaScript sekä Ajax ovat tekniikoita, jota web-ohjelmistosuunnittelijoiden ja sellaiseksi opiskelevien kannattaa ehdottomasti omaksua. Paikkatietotekniikan ammattilaisten ja opiskelijoiden, jotka haluavat erikoistua web-sovelluksiin, edellä mainittujen tekniikoiden sekä OpenLayers-osaamisen yhdistelmä olisi erittäin hyödyllinen tulevaisuuden työmarkkinoiden ja paikkatietotekniikan kehityksen kannalta. Mielestäni aihetta olisi myös hyvä käsitellä paikkatietoalan oppilaitosten opinto-ohjelmissa.

Lähteet

1 Kokousmuistiot 1997–1999. Helsingin kaupunki, Kaupunkimittausosasto.

2 Arponen, Matti. 2011. Muistiinpanot haastattelusta 17.11.2011.

3 Kostet, Juhani. 1992. Helsingin kaupunkimittauksen vaiheita 1892-1992. Jyväskylä: Gummerus.

4 OpenLayers Graduates Incubation. 2007. Verkkodokumentti. Osgeo.org. <http://www.osgeo.org/node/461>. Luettu 19.8.2011.

5 Schmidt, Christian. 2010. OpenLayers 3.0 on GitHub (30.6.2010). Verkkodokumentti <http://openlayers.org/blog/2010/06/30/openlayers-3-on-github/> Luettu 1.2.2012.

6 OpenLayers Milestones 2012. Verkkodokumentti. Osgeo.org <http://trac.osgeo.org/openlayers/roadmap>. Luettu 4.2.2012

7 HTML5 Editors draft 7 March 2012. Verkkodokumentti. W3C-consortium. <http://dev.w3.org/html5/spec/Overview.html>. Päivitetty 3/2012. Luettu 9.3.2012.

8 Sauerwein, Tobias. 2010. Evaluation of HTML5 for its Use in the Web Mapping Client OpenLayers. Verkkodokumentti. <http://www.tobias-sauerwein.de/files/Tobias%20Sauerwein%20-%20Evaluation%20of%20HTML5%20for%20its%20Use%20in%20the%20Web%20Mapping%20Client%20OpenLayers.pdf>.

9 UTF-8 and unicode standards. 2011. Verkkodokumentti. Unicode Consortium. <http://www.utf8.com/>. Luettu 10.3.2012.

10 PHP General Information: Manual. 2012. Verkkodokumentti. Php.net. <http://fi.php.net/manual/en/faq.general.php>. Luettu 15.4.2012.

index.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" <html
xmlns="http://www.w3.org/1999/xhtml" lang="fi" xml:lang="fi">
<head>
<title>Pääkaupunkiseudun opaskartta</title>
<!-- Seuraava IE 8 -yhteensopivuuden takaamiseksi matkanmittaustoiminnon kanssa -->
<meta http-equiv="X-UA-Compatible" content="IE=EmulateIE7"/>
<meta http-equiv="Content-type" content="text/html; charset=UTF-8" />
<link rel="stylesheet" href="./theme/default/style.css" type="text/css" />
<script src="lib/styleswitcher.js" type="text/javascript"></script>
<link href="style1minus.css" rel="stylesheet" type="text/css" title="small" />
<link href="style.css" rel="alternate stylesheet" type="text/css" title="medium" />
<!--[if IE]><link href="" rel="alternate stylesheet" type="text/css"/><![endif]-->
<link href="style1plus.css" rel="alternate stylesheet" type="text/css" title="large"
/>
<script type="text/javascript" src="./OpenLayers.js"></script>
<script type="text/javascript" src="./lib/KmoLayerSwitcher.js"></script>
<script type="text/javascript" src="init.js"></script>
</head>
<body onload="init()">

<!-- Yläosa -->
<table border="0" width="100%" cellpadding="0" cellspacing="0" align="center">
<tr>
<td valign="top" >
<div class="wpsPortletBody">
<div id="top">
<div id="topLinks">
<div id="topLinksLeft">
<div id="textSize"> Tekstin koko <a href="#"  class="aText"
onclick="javascript:enlarge(); return false;">A+</a> | <a href="#"  class="aText"
onclick="javascript:reduce(); return false;">A-</a> </div>
 Suomeksi |<a href="index_sv.html" > På svenska</a> | <a href="index_en.html" >In
English</a>| <a
href="http://www.hel.fi/hki/hs/Selkosivut/Selkosivut/Etusivu">Selkokielellä</a> | <a
href="http://www.infopankki.fi/" >Muut kielet</a>  </div> <div id="topLinksRight">
<a
href="http://www.helsinginseutu.fi/wps/portal/HelsinginSeutu?WCM_GLOBAL_CONTEXT=/HS/He
lsingin+Seutu/Aihealueet/Yhteystiedot" >Yhteystiedot</a> | <a
href="http://www.helsinginseutu.fi/wps/portal/HelsinginSeutu?WCM_GLOBAL_CONTEXT=/HS/He
lsingin Seutu/Aihealueet/Osallistuminen, demokratia, hallinto/Palaute" >
Palaute</a> | <a
href="http://www.helsinginseutu.fi/wps/portal/HelsinginSeutu?WCM_GLOBAL_CONTEXT=/HS/He
lsingin Seutu/Etusivu_1/Sivukartta" >Sivukartta</a> </div></div>
</div><div id="topLogo">
<img src="img/logo_suomi.jpg" alt="Helsingin Seutu - Otsikkokuva" />
</div><a name="paanavigaatio"></a><div id="navTop">
<div class="innerContent">
<ul>
<li><a
href="http://www.helsinginseutu.fi/wps/portal/HelsinginSeutu?WCM_GLOBAL_CONTEXT=/hs/He
lsingin+Seutu/Etusivu_1/"><span>Etusivu</span></a></li>
<li><a
href="http://www.helsinginseutu.fi/wps/portal/HelsinginSeutu?WCM_GLOBAL_CONTEXT=/hs/He
lsingin+Seutu/Aihealueet/"><span>Palvelut</span></a></li>
<li><a
href="http://www.helsinginseutu.fi/wps/portal/HelsinginSeutu?WCM_GLOBAL_CONTEXT=/hs/He
lsingin+Seutu/Neuvonta/"><span>Neuvonta</span></a></li>
<li><a
href="http://www.helsinginseutu.fi/wps/portal/HelsinginSeutu?WCM_GLOBAL_CONTEXT=/hs/He
lsingin+Seutu/S_hk_inen+asiointi+ja+lomakkeet/"><span>Verkkoasiointi</span></a></li>
<li><a
href="http://www.helsinginseutu.fi/wps/portal/HelsinginSeutu?WCM_GLOBAL_CONTEXT=/hs/He
lsingin+Seutu/Kaupunkitieto+ja+tilastot/"><span>Kaupunkitieto</span></a></li>
```

```
<li><a
href="http://www.helsinginseutu.fi/wps/portal/HelsinginSeutu?WCM_GLOBAL_CONTEXT=/hs/He
lsingin+Seutu/Uutishuone/"><span>Uutishuone</span></a></li>
<li><a
href="http://www.helsinginseutu.fi/wps/portal/HelsinginSeutu?WCM_GLOBAL_CONTEXT=/hs/He
lsingin+Seutu/Yhteisty_elimet/"><span>Yhteistyöelimet</span></a></li>
<li><a class="selected"><span>Kartta</span></a></li>
<li><a
href="http://www.helsinginseutu.fi/wps/portal/HelsinginSeutu?WCM_GLOBAL_CONTEXT=/hs/He
lsingin+Seutu/Seutuselvitykset/"><span>Seutuselvitykset</span></a></li>
</ul>
</div>
</div>
</div>
</td>
</tr>
</table>
<!-- Yläosan loppu -->

<!-- Karttakehikko -->

<table border="0" cellpadding="0" cellspacing="0" style="vertical-align: top; margin-
left: 10px; height: 80%" id="body">
<tr style="vertical-align: top; height: 80%">
  <td style="vertical-align:top;">
    <div id="search" style="margin-top: 10px">
    <form name="frmHaku" id="frmHaku" action="" autocomplete="off" onsubmit="return
false;">
    Hae:<br />
    <select name="kohde" id="lang" title="valitse hakutapa" onchange="if
(this.selectedIndex>1) {addrSearch('',(document.getElementById('lang').value))}" >
      <option selected="selected" value="1">Haku suomenk. osoitteilla</option>
      <option value="2">Hae ruotsink. osoitteilla</option>
      <option value="3">Liikennepaikat</option>
      <option value="4">Museot</option>
      <option value="5">Kirkot</option>
      <option value="6">Sairaalat</option>
      <option value="7">Yritykset ja yhteisöt</option>
    </select>
    <input type="text" name="q" id="q1" size="20" class="haku_txt" title="Kirjoita
hakukenttään vähintään neljä merkkiä"
onkeyup="addrSearch(this.value,(document.getElementById('lang').value))" />
    </form>
    <div id="list" class="list"></div>
    </div>
    <br />
    <div class="services">
    <h2>Kartta</h2>
     <ul>
     <li><div id="layerswitcher">
      </div></li>
     </ul>
    <h2>Mittaus</h2>
     <ul>
     <li><input type="radio" name="type" value="none" id="noneToggle"
onclick="toggleControl(this);" checked="checked" /><label for="noneToggle">Liikuta
karttaa</label></li>
     <li><input type="radio" name="type" value="line" id="lineToggle"
onclick="toggleControl(this);" /><label for="lineToggle">Mittaa matkaa  <a
href="./help/#matkanmittaus" target="_blank">Ohje</a></label></li>
     <!--<input type="radio" name="type" value="polygon" id="polygonToggle"
onclick="toggleControl(this);" /><label for="polygonToggle">Mittaa pinta-ala</label>--
>
     <li><div id="output" style="margin-left: 22px;"></div></li>
     <li><div id="coords" style="margin-left: 22px;"></div></li>
     </ul>
    <h2>Ohjeita</h2>
```

```
     <ul>
     <li><a href="#" title="Palvelun käyttöohje" onclick="javascript:
window.open('./help/','Ohjeet','width=500,height=500,top=10,left=100,toolbar=no,
location=no,
directories=no,status=no,menubar=no,resizable=yes,scrollbars=yes')">Tietoja/ohje</a></
li>
     <li><a href="#" title="Selitteet karttamerkeille ja väreille"
onclick="javascript:
window.open('./help/legenda_fi.html','Selitteet','width=500,height=500,top=10,left=100
,toolbar=no, location=no,
directories=no,status=no,menubar=no,resizable=yes,scrollbars=yes')">Karttamerkkien
selitteet</a></li>
     <li><a href="#" title="Anna palautetta koskien karttoja, osoitteita tai kohteita"
onclick="javascript:
window.open('./fb/','Karttapalaute','width=900,height=500,top=10,left=100,toolbar=no,
location=no,
directories=no,status=no,menubar=no,resizable=yes,scrollbars=yes')">Karttapalaute</a><
/li>
     <li><a href="#" title="Tulostus"
onclick="javascript:printwin();">Tulostus</a></li>
     </ul>
    </div>
  </td>
  <td style="width: 100%; height: 100%; vertical-align: top">
    <!-- Karttakehys -->
    <div id="map" style="position: relative; vertical-align: top; height:100%;
width:100%"></div>
  </td>
</tr>
</table>

<div id="footer">
  <div id="footerLinks">
    <div id="footerLinks_left"> <a
href="http://www.helsinginseutu.fi/wps/portal/HelsinginSeutu?WCM_GLOBAL_CONTEXT=/HS/He
lsingin+Seutu/Aihealueet/Yhteystiedot" >Yhteystiedot</a> | <a
href="http://www.helsinginseutu.fi/wps/portal/HelsinginSeutu?WCM_GLOBAL_CONTEXT=/HS/He
lsingin Seutu/Aihealueet/Osallistuminen, demokratia, hallinto/Palaute" >Palaute</a> |
<a
href="http://www.helsinginseutu.fi/wps/portal/HelsinginSeutu?WCM_GLOBAL_CONTEXT=/HS/He
lsingin Seutu/Etusivu_1/Sivukartta" >Sivukartta</a> | <a
href="http://www.helsinginseutu.fi/wps/portal/HelsinginSeutu?WCM_GLOBAL_CONTEXT=/HS/He
lsingin+Seutu/Etusivu_1/Palveluseloste" >Palveluseloste</a> | <a
href="http://www.helsinginseutu.fi/wps/portal/HelsinginSeutu?WCM_GLOBAL_CONTEXT=/HS/He
lsingin+Seutu/Etusivu_1/Rekisteriseloste" >Rekisteriseloste </a> </div>
    <div id="footerLinks_right"> Copyright Seutuportaali 2008 </div>
  </div>
</div>

</body>
</html>
```

init.js

```
/**
 ** init.js OpenLayers Initialization script by Kaupunkimittausosasto, 2009 Helsinki
City Survey Department
 ** Lisätty koordinaattien laskenta 2010
**/
var map, markers, measureControls, layerSwitcher;
// Hakumerkkikuvakkeen määritys (punainen rengas)
var addriconsize = new OpenLayers.Size(32,32);
var addriconoffset = new OpenLayers.Pixel(-15,-15);
var addriconimage = new OpenLayers.Icon ('img/ring32.png',
addriconsize,addriconoffset);
var xaddr, yaddr, addrtitle;
xaddr=0;
yaddr=0;
addrtitle="";

function init(lang){
// Tekstit valitaan kieliversion mukaan
if(lang==2){
        var maplabel1 = 'Guidekarta';
        var maplabel2 = 'Guidekarta (utan företag)';
        var maplabel3 = 'Flygbild';
        var copyright = ' &copy; Esbo, Helsingfors och Vanda städer';
}else if(lang==3){
        var maplabel1 = 'Guide map';
        var maplabel2 = 'Guide map (w/o business/society)';
        var maplabel3 = 'Aerial photo';
        var copyright = ' &copy; Cities of Espoo, Helsinki and Vantaa';
}else{
        var maplabel1 = 'Opaskartta';
        var maplabel2 = 'Opaskartta (ei yritysnimiä)';
        var maplabel3 = 'Ilmakuva';
        var copyright = ' &copy; Espoon, Helsingin ja Vantaan kaupungit';
}
/** ************** Karttaolio, layerit, kontrollit alustetaan ********* **/
/** Map -olion luonti **/
map = new OpenLayers.Map( 'map',
    { controls:[],
                    maxExtent: new OpenLayers.Bounds(27000,65000,69000,100000),
                    //maxResolution: 2,
                    projection: "Helsinki",
                    units: "m",
                    numZoomLevels: 4
    }
);
/** Layer-olioiden luonti **/
// Opaskartta-layer
var op_map = new OpenLayers.Layer.WMS( maplabel1,
    "map.php",
    { layers: "Opaskartta2" },
    { buffer: 0,
                    attribution: (maplabel1+copyright),
                    maxResolution: 2,
                    minResolution: 16,
                    numZoomLevels: 4
    }
);
// Opaskartta+yritysnimitaso -layer
var op2_map = new OpenLayers.Layer.WMS( maplabel2,
    "map.php",
    { layers: "Opaskartta" },
    { buffer: 0,
                    attribution: (maplabel2+copyright),
                    maxResolution: 2,
                    minResolution: 2,
```

```
                    numZoomLevels: 1
    }
);
// Ortokuva -layer
var ok_map = new OpenLayers.Layer.WMS( maplabel3,
    "map.php",
    { layers: "ortokuva" },
    { buffer: 0,
                    attribution: (maplabel2+copyright),
                    maxResolution: 0.5,
                    minResolution: 2,
                    numZoomLevels: 3
    }
);
// Staattinen rasterikuva -layer indeksikarttaa varten
var ov_map = new OpenLayers.Layer.Image("Yleiskartta",
    "seutu.png",
    new OpenLayers.Bounds(27296,63740,70144,99804),
    new OpenLayers.Size(1339,1127),
    { maxResolution: 32,
                    minResolution: 16,
                    numZoomLevels: 2,
                    maxExtent: new OpenLayers.Bounds(27296,63740,70144,99804),
                    units: "m",
                    projection: "Helsinki"
    }
);
// Lisätään layerit karttaolioon
map.addLayers([op_map,op2_map,ok_map]);

// Lisätään markers-taso, toistaiseksi tyhjänä.. Estetään näkyminen switcherissä.
markers = new OpenLayers.Layer.Markers('osoite',{displayInLayerSwitcher: false});
map.addLayer (markers);

//Tehdään yleiskarttaikkunan säätötoimet. Ikkunan koko säädetään ensin arrayssa
(ov_map)
var ov_options = { size: new OpenLayers.Size(200,220), layers:[ov_map] };
var ov_contr = new OpenLayers.Control.OverviewMap(ov_options);

/** Kontrollien lisääminen karttaolioon **/
// Lisätään näyttöön yleiskarttaruudun kontrollina
map.addControl(ov_contr);
// suurennetaan yleiskartta oletuksena!
ov_contr.maximizeControl();
//Panorointi- ja zoomikontrolli, jossa zoomisäätimen pitkä malli
map.addControl( new OpenLayers.Control.PanZoomBar() );
//Otetaan näkyviin karttatasojen valitsin
map.addControl( new OpenLayers.Control.KmoLayerSwitcher({'div':
OpenLayers.Util.getElement('layerswitcher')}) );
//Alareunan mittakaavajana
map.addControl( new OpenLayers.Control.ScaleLine({maxWidth:150}) );
//Karttalayerin metatieto. Tähän voidaan laittaa copyrightit jne.
map.addControl(new OpenLayers.Control.Attribution() );
//Näppäimistön peruskontrolli
map.addControl(new OpenLayers.Control.KeyboardDefaults() );
//Hiiren peruskontrolli
map.addControl(new OpenLayers.Control.MouseDefaults() );
//Permalink -linkitys
map.addControl(new OpenLayers.Control.Permalink());
/** ******** Karttaolion, layerien, kontrollien alustus päättyy tähän ** **/

/************** Matkanmittaustoimintojen alustus ********************/
// Tässä asetetaan kartanliikutusmoodi oletuksena päälle (erotuksena mittaa matkaa-
moodista)
document.getElementById('noneToggle').checked = true;
// Malli toimintoon on pitkälti mukailtu OpenLayersin dokumentaatiosta löytyneestä
```

```
// skriptiesimerkistä, ks http://dev.openlayers.org/releases/OpenLayers-
2.10/examples/measure.html
// Ulkoasun määritykset mittakursorille, mittajanalle sekä pinta-alajanoille
sketchSymbolizers = {
        "Point": {
                    pointRadius: 4,
                    graphicName: "circle",
                    fillColor: "white",
                    fillOpacity: 1,
                    strokeWidth: 1,
                    strokeOpacity: 1,
                    strokeColor: "#fe0500"
        },
        "Line": {
                    strokeWidth: 3,
                    strokeOpacity: 1,
                    strokeColor: "#0078ff",
                    strokeDashstyle: "dash"
        },
        "Polygon": {
                    strokeWidth: 2,
                    strokeOpacity: 1,
                    strokeColor: "#00ff00",
                    fillColor: "white",
                    fillOpacity: 0.3
        }
};
var style = new OpenLayers.Style();
style.addRules([
        new OpenLayers.Rule({symbolizer: this.sketchSymbolizers})
]);
var styleMap = new OpenLayers.StyleMap({"default": style});
var options = {
        // !!! Määritetään että vain METRIT näytetään! Päällekirjoittaa Measure.js:n
vastaavan kohdan.
        displaySystemUnits: { metric: ['m']},
        handlerOptions: {
                    style: "default",
                    layerOptions: {styleMap: styleMap},
                    persist: true
        }
};
measureControls = {
        line: new OpenLayers.Control.Measure(
        OpenLayers.Handler.Path, options
        ),
        polygon: new OpenLayers.Control.Measure(
        OpenLayers.Handler.Polygon, options
        )
};
var control;
for(var key in measureControls) {
        control = measureControls[key];
        control.events.on({
                    "measure": handleMeasurements,
                    "measurepartial": handleMeasurements
        });
        map.addControl(control);
}
/************** Matkanmittaustoimintojen alustus päättyi tähän **********/

/*************Koordinaattinäytön alustus alkaa tästä *****************/
/** Luodaan uusi olio coords **/
var coords = new OpenLayers.Control();
OpenLayers.Util.extend(coords, {
        draw: function () {
                    this.point = new OpenLayers.Handler.Point( coords,
```

```
                                {"done": this.notice},
                                {keyMask: OpenLayers.Handler.MOD_CTRL});
                                this.point.activate();
                        },
                        notice: function (e) {
                        var yvvj = e.x;
                        var xvvj = e.y;
                        yvvj = yvvj.toFixed();
                        xvvj = xvvj.toFixed();
                        innerCoord(xvvj,yvvj);
                        var coordtext = ("Koordinaattien mittaus");
                        document.getElementById('coords').innerHTML = coordtext;
                        // avataan popup-ikkuna, jossa koordinaatit näytetään
                        window.open("./search/coord.php?q="+xvvj+","+yvvj, "koordinaatit",
"toolbar=1,width=400,height=300");
        }
        });
// lisätään kontrolli map-olioon!
map.addControl(coords);
/*************Koordinaattinäytön alustus päättyy tähän*****************/
// Lopuksi suoritetaan karttaelementin keskistys haluttuihin koordinaattehin ja zoom-
tasoon
map.setCenter(new OpenLayers.LonLat(52440, 73600), 3);

}
/************* Init-funktion päättyy tähän *****************/

/* ***************** Muut funktiot ***************************** ****/
//*** Locate-funktiolla keskistetään kartta haluttuun pisteeseen. mm osoitehaku
käyttää tätä.
function Locate(y,x) {
        map.panTo(new OpenLayers.LonLat (y, x))
}
//*** Show point näyttää haetun kohdan kartalla keskimerkkikuvakkeen avulla (punainen
rengas)
function showPoint(y,x,title){
        mark = new OpenLayers.Marker(new OpenLayers.LonLat (y, x), addriconimage);
        mark.setOpacity (1);
        markers.addMarker(mark);
        yaddr = (y);
        xaddr = (x);
        addrtitle = (title);
}
/************* Ajax-toiminnot *************************/
var xmlHttp,q,language;
/**** Osoitehakufunktio ********/
// Parametit:q=hakusana l=katunimen kieli (su,ru) tai poi-haun
tyyppi(museot,sairaalat,kirkot..) language=su,ru,en
function addrSearch(q,l,language){
        xmlHttp=GetXmlHttpObject();
        if (xmlHttp==null){
                alert ("Ei riittävää javascript-tukea! No AJAX-support!");
                return;
        }
        if (l==2){
                var url="./search/as2.php?";
        }else if(l>2){
                var url="./search/poi.php?l="+l+"&";
        }else{
                var url="./search/as.php?";
        }
        url=url+"q="+q+"&language="+language+"&sid="+Math.random();
        // tapahtumakuuntelija (onreadystatechange) määrittää jokaisen tapahtuman
jälkeen
        // että hakufunktion tuloksesta päivitetään list-(div)elementin sisältö
        xmlHttp.onreadystatechange=function(){
                if (xmlHttp.readyState==4){
```

```
        document.getElementById("list").innerHTML=xmlHttp.responseText;
                    }
        };
        xmlHttp.open("GET",url,true);
        xmlHttp.send(null);
}

// selainten erojen vuoksi luodaan XML-kyselyolio eri tavoilla
function GetXmlHttpObject(){
        var xmlHttp=null;
        try
        {
                    // Firefox, Opera 8.0+, Safari, IE 7->
                    xmlHttp=new XMLHttpRequest();
        }
        catch (e)
        {
                    // Internet Explorer 6 ja <6
                    try
                    {
                                xmlHttp=new ActiveXObject("Msxml2.XMLHTTP");
                    }
                    catch (e)
                    {
                                xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
                    }
        }
        return xmlHttp;
}

/******** Matkanmittauskontrollifunktiot ****************************/
function handleMeasurements(event) {
        var geometry = event.geometry;
        var units = event.units;
        var order = event.order;
        var measure = event.measure;
        var element = document.getElementById('output');
        var out = "";
        if(order == 1) {
                                out += measure.toFixed(0) + " " + units;
                    } else {
                                out += "Pinta-ala: " + measure.toFixed(0) + " " +
units + "<sup>2</" + "sup>";
                    }
        element.innerHTML = out;
}

function toggleControl(element) {
        for(key in measureControls) {
        var control = measureControls[key];
                    if(element.value == key && element.checked) {
                                control.activate();
                    } else {
                                control.deactivate();
                    }
        }
}

function calcVincenty(geometry) {
        var dist = 0;
        for (var i = 1; i < geometry.components.length; i++) {
                    var first = geometry.components[i-1];
                    var second = geometry.components[i];
                    dist += OpenLayers.Util.distVincenty(
                                {lon: first.x, lat: first.y},
                                {lon: second.x, lat: second.y}
```

```
                                );
                        }
        return dist;
}

/********** Tulostusikkunafunktio ********************/
function printwin() {
        var xy = map.getCenter();
        var pZoom = map.getZoom();
        var printx = (xy.lat);
        var printy = (xy.lon);
        var lyr = map.getLayerIndex(map.baseLayer);
        var res = map.getResolution();
        // haetaan koordinaatit keskusmerkille ja muut tiedot
        var printurl =
"print.php?x="+printx+"&y="+printy+"&z="+pZoom+"&l="+lyr+"&r="+res+"&px="+xaddr+"&py="
+yaddr+"&t="+addrtitle;
        window.open(printurl,"mywindow",
"menubar=1,toolbar=1,resizable=1,width=1020,height=820");
}

/**** Koordinaattienhakufunktio ******************/
function innerCoord(x,y){
        xmlHttp=GetXmlHttpObject();
        if (xmlHttp==null){
                        alert ("Ei riittävää javascript-tukea! No  JavaScript/AJAX-
support!");
                        return;
        }
        var url="./search/icoord.php?"
        url=url+"q="+x+","+y;
        url=url+"&sid="+Math.random();
        xmlHttp.onreadystatechange= function(){
                        if (xmlHttp.readyState==4){

        document.getElementById("list").innerHTML=xmlHttp.responseText;
                        }
        }
        xmlHttp.open("GET",url,true);
        xmlHttp.send(null);
}
```

style.css

```
/*      YLEISET     */
html, body {
        height:100%;
        margin:0px;
        padding:0px;
        font-family: arial, helvetica, sans-serif;
        font-size: medium;
}
a {
        color:#c32e30;
        text-decoration:none;
}
h2 {
        border-bottom: 3px solid #000000;
        margin-bottom:15px;
        font:normal 24px times, times new roman, helvetica, sans-serif;
        padding-top:10px;
        padding-bottom:10px;
}
h4 {
        font-size: 110%;
}
a:hover {
        text-decoration:underline;
}
img {
        display:block;
        margin-bottom:15px;
}
a img {
        border-width:0px;
}
.innerContent {
        /**width:965px; **/
        padding-left:23px;
}
th {
        text-align:left;
}
#body .tdSpacer {
        padding:0px;
        font-size:2px;
        width:8px;
        height:8px;
}
.imgRight {
        float:right;
        margin:0px 0px 0.5em 0.5em;
}
/*      OTSIKKOTASOT*/
#header h1 {
        margin:18px 0px 0px 90px;
}
.aakkoset h3 {
        border-bottom: 3px solid #000000;
        margin-bottom:15px;
        font:normal 26px times, times new roman, helvetica, sans-serif;
        padding-top:10px;
        padding-bottom:10px;
}
#body .tdContentPalsta img {
        padding:5px;
}
```

```
#body .tdPalsta img {
	padding-top:5px;
}
#body .tdPalsta h3 {
	font:bold 26px times, times new roman, helvetica, sans-serif;
	color:#c32e30;
	padding: 0px;
	margin-top: 0px;
	margin-right: 0px;
	margin-bottom: 5px;
	margin-left: 0px;
}
#body .cityContent h3 {
	color:#4E5760;
	margin:0px 0px 2px 0px;
	padding:0px 0px 0px 0.25em;
	border-bottom:1px solid #333333;
	font-size:120%;
}
.tblPalaute td h3 {
	font:bold 75% verdana, arial, sans-serif;
	margin:1em 0px 0.5em 0px;
}
.tblMain td h3 {
	font:normal 26px times, times new roman, helvetica, sans-serif;
	padding-top:10px;
	margin:0px;
}
.tdContentNosto .contentNosto h2 {
	font:bold 80% verdana, arial, sans-serif;
	color:#0B3F70;
	display:block;
	background:url(img/bgContentNosto.gif) left top repeat-y #D2DBDE;
	padding:3px 6px;
	margin:0px;
}
/*** STYLES FOR TOP LINKS ***/

#top {
	background-color: #eeeeee;
	height:20px;
	color:#606060;
	font:normal 12px arial, helvetica, sans-serif;
	/**width:975px;**/
	padding-left:25px;
}
#top a {
	text-decoration:none;
	padding: 0px 4px 0px 4px;
}
#top a:hover {
	text-decoration:underline;
}
#topLinks {
	/** width:960px; **/
	padding-right:30px;
	display:block;
}
#topLinksLeft {
	float:left;
}
#topLinksRight {
	float:right;
	padding-right:50px;
}
#topLinksLeft #textSize {
	float:left;
```

```
        padding-right:25px;
}
#topLogo {
        background-color:#c32e30;
        padding: 0px 60px 0px 20px;
        /** width:920px; **/
}
#header #tblHeader {
        /**width:965px;**/
        margin-left:23px;
}
.tdHaku {
        vertical-align:top;
        padding-top:9px;
}
.tdHaku table {
        float:right;
        font-size:80%;
}
.tdHaku label {
        padding:2px 4px;
        background-color:#6F859C;
        display:block;
        color:#FFFFFF;
}
.tdHaku input {
        border-width:1px;
        margin:0px 2px;
        padding:0px 2px;
}
#navTop ul li a {
        display: block;
        height: 30px;
        float: left;
        font-family: arial, helvetica, sans-serif;
        font-size: 12px;
        font-weight: bold;
        color: #fff;
        background-image: url(img/TopNavi_right_two.png);
        background-repeat: no-repeat;
        background-position: right top;
        text-decoration: none;
        margin-left:4px;
        margin-right:4px;
}
.tdHaku a {
        text-decoration:none;
        color:#0B3F70;
        font:bold 90% verdana, arial, sans-serif;
}
#navTop {
        background-color:#c32e30;
        height:30px;
        /**width:1000;**/
        padding: 0px 60px 0px 20px;
}
html>body #navTop {
        margin-top:-17px;
}
#navTop ul li {
        display: inline; /* Fix IE Step Down */
}
#navTop ul li a span {
        display: block;
        height: 22px;
        float: left;
        background-image: url(img/TopNavi_left_two.png);
```

```
        background-repeat: no-repeat;
        background-position: left top;
        padding-top: 8px;
        padding-right: 10px;
        padding-bottom: 0px;
        padding-left: 10px;
}
#navTop .selected {
        background-position: bottom right;
}
#navTop .selected span {
        background-position: bottom left;
        color: #8D2123;
}
#body {
        position:relative;
        clear:both;
}
#body a:hover {
        /**         color:#000000;*/
        text-decoration:underline;
}
#body .tblMain {
        width:100%;
}
.tblMain td {
        vertical-align:top;
}
#body .headerMargin {
        height:26px;
        width:225px;
}
#body .breadcrumb a {
        color:#c32e30;
        text-decoration:none;
}
#body .breadcrumb span {
        color:#000000;
        font-weight:normal;
}
#body .tdContentPalsta {
        padding-bottom:15px;
        color:#000000;
        line-height:140%;
        width:100%;
}
.tdContentPalsta .linkList {
        list-style-type:none;
        margin:6px 0px 0px 0px;
        padding:0px;
        line-height:0.9em;
}
.linkList li {
        margin-bottom:0.5em;
}
#body .tdContentNosto {
        border-top:1px solid #FFDB14;
        background-color:#FFFFFF;
        padding:15px 0px 15px 15px;
}
.tdContentNosto .contentNosto {
        width:250px;
        border:1px solid #1983A6;
        background-position:left bottom;
        background-repeat:no-repeat;
        padding-left:0px;
}
```

```
.tdContentNosto .contentNosto p {
        margin:0.5em 0px 0.5em 85px;
        width:150px;
}
.contentNosto .lueLisaa {
        margin-bottom:0.5em;
}
#body .ulTopics {
        list-style-image:url(img/bulletArrow.gif);
        padding:0px;
        margin:3px 0px 0px 18px;
}
.ulTopics li {
        line-height:15px;
}
.ulTopics li a {
        font:normal 70% verdana, arial, sans-serif;
        color:#0B3F70;
        text-decoration:none;
        vertical-align:middle;
}
#body .tdEmpty {
        width:425px;
}
#body .tdEmptyBlue {
        width:425px;
        background-color:#F9FAFB;
}
#body .cityContent {
        float:left;
        width:225px;
}
#body .cityContent p {
        margin:0px;
        color:#000000;
        line-height:140%;
}
#body .cityContent a {
        color:#c32e30;
        text-decoration:none;
}
#body .cityContent a:hover {
        color:#000000;
        text-decoration:underline;
}
#body .services {
        float:left;
        width:225px;
        padding-right:20px;
        vertical-align: top;
}
#body .services p {
        margin:0px;
        color:#000000;
        line-height:140%;
}
#body .services a {
        color:#c32e30;
        text-decoration:none;
}
#body .services a:hover {
        color:#000000;
        text-decoration:underline;
}
#body .services ul {
        margin:0px 0px 25px 0px;
        padding:0px;
```

```
}
#body .services li {
        list-style-type:none;
        border-bottom:1px solid #cacaca;
}
#ajank {
        float:right;
        width:225px;
        padding-left: 20px;
}
#ajank .tblPalstat {
        width:100%;
        clear:both;
}
#ajank .tblPalstat a:hover {
        color:#000000;
        text-decoration:underline;
}
#body .tblPalstat .tdPalsta {
        width:100%;
}
#body .tdPalsta ul {
        padding:0px;
        margin-top: 0px;
        margin-right: 0px;
        margin-bottom: 25px;
        margin-left: 0px;
}
#body .tdPalsta .emptymargin ul {
        padding:0px;
        margin-top: 0px;
        margin-right: 0px;
        margin-bottom: 0px;
        margin-left: 0px;
}
#body .tdPalsta .date {
        line-height:15px;
        display:inline;
}
#body .tdPalsta li {
        list-style-type:none;
        border-bottom:1px solid #cacaca;
}
#body .tdPalsta p {
        margin:0px;
        color:#000000;
        line-height:140%;
}
#body .tdPalsta li a {
        color:#c32e30;
        text-decoration:none;
}
#body .tdPalsta .lisaa {
        border-bottom:0px;
}
#body .palstaRight {
        width:190px;
}
#body .divPalsta {
        width:225px;
        float:right;
        padding-left:20px;
}
.divPalsta .palstaContent {
        width:225px;
        float:right;
}
```

```
.divPalsta .palstaContent p {
	margin:0px;
}
#body .divPalsta a {
	font:normal 22px times, times new roman, helvetica, sans-serif;
	color:#c32e30;
	display:block;
	border-bottom: 1px solid #cacaca;
}
#body #lastUpdate {
	color:#0F6094;
	font-size:60%;
}
.tblPalaute td {
	padding:0px;
	vertical-align:top;
}
.tblPalaute label {
	font:normal 70% verdana, arial, sans-serif;
	margin-right:0.75em;
}
.tblPalaute .tf {
	width:15em;
	margin-bottom:6px;
	font:normal 70% verdana, arial, sans-serif;
}
.tblPalaute .btn {
	border:1px outset #6F859C;
	background-color:#6F859C;
	color:#FFFFFF;
	font-size:75%;
}
/*	ALAPALKKI	*/
#footer {
	display:block;
	clear:both;
	background-color:#eeeeee;
	padding-top:10px;
	padding-left:25px;
	/** width:975px;**/
}
/*	SEARCH & BROWSE : hakutulossivun raidan v‰ri (jos liian kirkas, kokeile :
#F6E57E;		*/
.topbar {
	background-color: #F9DD3C;
}
/* STYLES FOR LEFT PANE */
#leftPane {
	width:225px;
	float:left;
}
#leftPane ul {
	margin:0px 0px 25px 0px;
	padding:0px;
}
#leftPane ul li {
	list-style-type:none;
	border-bottom:1px solid #cacaca;
}
#leftPane ul li .subNavi_selected {
	color:#8d2325;
}
#leftPane ul li ul {
	margin:5px 0px 0px 10px;
}
#leftPane ul li .lvl2 {
	margin:5px 0px 0px 10px;
```

```
}
#leftPane ul li ul li {
        border-bottom:0px;
}
#leftPane ul li ul li .subNavi_selected {
        color:#000000;
}
#leftPane a {
        display:block;
        padding:2px 0px 2px 0px;
}
#leftPane p {
        margin-bottom:10px;
}
.tdNavLeft .aakkoset {
        float:left;
        width:225px;
}
.aakkoset a {
        font:normal 22px times, times new roman, helvetica, sans-serif;
        color:#c32e30;
        display:inline !important;
}
#footer_sitemap {
        display:block;
        height:161px;
        border-bottom:1px solid #cacaca;
        /** width:960px; **/
}
#footer_sitemap a {
        display:block;
        padding-top:3px;
}
#footer_sitemap_left {
        float:left;
        display:block;
        width:478px;
        border-right: 1px solid #cacaca;
        margin-right:10px;
        height:131px;
}
#footer_sitemap_right {
        display:block;
}
#footer_sitemap #column1 {
        float:left;
        width:235px;
        border-right: 1px solid #cacaca;
        margin-right:10px;
        height:131px;
}
#footer_sitemap #column2 {
        padding-left:10px;
}
#footer_sitemap #column3 {
        float:left;
        width:235px;
        border-right: 1px solid #cacaca;
        margin-right:10px;
        height:131px;
}
#footer_sitemap #column4 {
        float:left;
}
#footerLinks {
        display:block;
        /** width:960px; **/
```

```
        font:normal 12px arial, helvetica, sans-serif;
        color:#666666;
        padding-top:5px;
        padding-bottom:5px;
        background-color:#eeeeee;
        height:25px;
}
#footerLinks a {
        padding:0px 3px 0px 3px;
}
#footerLinks_left {
        float:left;
}
#footerLinks_right {
        float:right;
        width:160px;
}
hr {
        border-top: 3px solid #000000;
        color:#000000;
        height:3px;
}
#body .tdNavLeft {
        float:left;
        width:225px;
        padding-right:20px;
}
.tdNavLeft ul {
        padding:0px;
}
.tdNavLeft li {
        list-style-type:none;
        border-bottom:1px solid #cacaca;
        padding-left:20px;
}
/* STYLE FOR RIGHT PANE */
#search {
        float:left;
        padding-top: 0px;
        padding-right: 0px;
        padding-bottom: 0px;
        padding-left: 0px;
}
#search .radiobutton {
        display:block;
        font-weight:normal;
}
#search_textbox {
        width:205px;
}
#search form {
        background-color:#eeeeee;
        width:220px;
        height:70px;
        font:bold 13px arial, helvetica, sans-serif;
        padding:5px 0px 5px 5px;
        margin-bottom:10px;
}
#search_button {
        display:block;
        cursor:pointer;
        float:right;
        border:0px;
        margin:5px 10px 0px 0px;
        background-color:#c32e30;
        font:bold 12px arial, helvetica, sans-serif;
        color:#FFFFFF;
```

(12)

```
        width:45px;
        height:20px;
}
#events {
        float:right;
        width:225px;
        padding-left:20px;
}
#events .city {
        border-bottom: 1px solid #cacaca;
        padding-bottom:10px;
}
#events .city p {
        margin-top:0px;
        padding-top:0px;
}
#events .city h4 {
        font:bold 12px arial, helvetica, sans-serif;
        margin-bottom:0px;
}
.tdNavLeft ul {
        margin:0;
        width:225px;
}
.tdNavLeft li {
        width:225px;
}
.tdNavLeft li a {
        color:#c32e30;
}

.tdNavLeft a {
        color:#c32e30;
        display:block;
}
.tdNavLeft a.selected {
        color:#8d2325;
}
.tdNavLeft a.selected2 {
        color:#000000;
}
.tdNavLeft a.selected3 {
        color:#000000;
}
.tdNavLeft a.selected4 {
        color:#000000;
}
.tdNavLeft a.subnavi2 {
        padding-left:10px;
}
.tdNavLeft a.subnavi3 {
        padding-left:20px;
}
.tdNavLeft a.subnavi4 {
        padding-left:30px;
}
.tdNavLeft p {
        margin:0px;
        padding-top:2px;
        padding-bottom:2px;
        display:block;
        border-bottom: 1px solid #cacaca;
}
.divPalsta .palstaContent font {
        font:normal 22px times, times new roman, helvetica, sans-serif;
```

```
        color:#c32e30;
        display:block;
        border-bottom: 1px solid #cacaca;
}
#body .newsroom {
        width:977px;
}
.newsroom .tdPalsta {
        width:100%;
}
.newsroom td {
        vertical-align:top;
}
.tblPalaute .btn {
        display:block;
        cursor:pointer;
        float:left;
        border:0px;
        margin:5px 10px 0px 0px;
        background-color:#c32e30;
        color:#FFFFFF;
        padding-left:3px;
        padding-right:3px;
        height:20px;
        font:bold 12px arial, helvetica, sans-serif;
}
.tblMain .tblPalaute td h3 {
        margin-bottom:0px;
        font-family: arial, helvetica, sans-serif;
        font-size: medium;
        font-weight: bold;
}
#body .teematieto a {
        font-family: arial, helvetica, sans-serif;
        font-size: medium;
        display:inline;
        border-bottom: none;
}

/** Osoitelistaelementin muotoilu **/
.list { height: 100px;          width: 225px; overflow: auto; background-color: white}

/** OL-elementtien muotoilut **/
.olControlScaleLineBottom { display: none }
.olControlScaleLineTop { background-color: white }
.olControlAttribution { left: 1.0em; background-color: white; width: 120px }
.viewportframe { position: absolute }
.viewport { position: absolute; left:1px; top:1px; right:1px; bottom:1px; height:100%;
width:100%; min-height:600px; min-width:800px}
/** Permalinkin (ikilinkki) muotoilut jotka korvaavat natiivit **/
.olControlPermalink {
    left: 10px;
    bottom: 9em;
        background-color: white;
        width: 120px
}

#coordwindow{
        font-family: arial, helvetica, sans-serif;
        font-size: small;
        color: #000000;
        background-color:#ffffff;
}
#coordwindow td{
        font-family: arial, helvetica, sans-serif;
```

(12)

```
        font-size: small;
        background-color:#eeeeee;
}
#coordwindow th{
        font-family: arial, helvetica, sans-serif;
        font-size: small;
        color: White;
        background-color:#c32e30;
}
```