Jake Horn

Verkkokarttasovelluksen uusi ulkoasu ja sen hyödyntäminen

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Mediatekniikan koulutusohjelma

Insinöörityö

12.5.2015

Metropolia

Tekijä Otsikko Sivumäärä Aika	Jake Horn Verkkokarttasovelluksen uusi ulkoasu ja sen hyödyntäminen 30 sivua 12.5.2015
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Mediatekniikka
Suuntautumisvaihtoehto	Digitaalinen media
Ohjaaja	Lehtori Ilkka Kylmäniemi

Insinöörityössä oli tarkoituksena tutkia ja toteuttaa karttasovellus, jossa käytetään kartan ulkoasuna omaa karttaa. Työssä perehdyttiin karttojen historiaan verkossa, karttarajapintoihin ja niiden ominaisuuksiin.

Verkkokarttojen historia on syvästi yhteydessä Internetin kehitykseen. Niiden luomiseen kuuluu suunnittelu, luominen ja jakelu Internetissä. Ensimmäinen verkkokarttapalvelu julkaistiin vuonna 1993. Nykypäivänä karttarajapintapalveluiden tarjoajia on monia, ja niiden sisältö on laajentunut tekniikan kehittyessä.

Sovelluksen toteutuksessa käytettiin karttarajapintana Google Maps API:a. Käyttöliittymään toteutettiin responsiivinen ulkoasu ja kuvakaruselli Bootstrapia käyttäen. Karttaan sijoitettiin markkereita, joilla pystyi kontrolloimaan kuvakarusellia. Sovelluksessa käytettiin lisäksi HTML-, CSS- ja jQuery-ohjelmointikieliä.

Työssä tehdyn sovelluksen tarkoitus on lisätä tietoisuutta karttarajapinnoista ja olla mallina tuleviin karttasovelluksiin. Toteutettu sovellus vastasi suunniteltua sovellusta, ja se tuo hyvin esille, mitä ominaisuuksia karttasovelluksilla voidaan toteuttaa. Työstä saatujen kokemusten avulla on helpompi lähteä suunnittelemaan uutta karttasovellusta.

Avainsanat	Karttarajapinta, Google Maps

Author Title Number of Pages Date	Jake Horn A new layout for a web map application and how to utilize it 30 pages 12 May 2015
Degree	Bachelor of Engineering
Degree Programme	Media Technology
Specialisation option	Digital Media
Instructor	Ilkka Kylmäniemi, Senior Lecturer

The purpose of the final year project was to study and realize a map application where the user would use his/her own map as the layout of the map. As a result, the thesis deals with the history of maps on the web, map interfaces and their features.

The history of web maps is deeply ingrained in the development of the Internet. Their creation consists of planning, creating and distributing on the Internet. The first Web Map Service was released in 1993. Today there are many providers for web map interfaces and their content has widened as the technology has evolved.

The Google Maps API map interface was used in the realization of the application this thesis deals with. A responsive layout and automatic picture slide show were created into the user interface using Bootstrap. Markers were placed in the map to enable controlling the automatic picture slide show. Additionally, the HTML, CSS and jQuery programming languages were used in the application.

The purpose of the application is to increase awareness of map interfaces and to be a model for future map applications. The realized application matched the planned application and it clearly shows what features can be realized with map applications. With the experience gained from the project it is easier to start developing a new map application.

Keywords	Map interface, Google Maps

Sisällys

Lyhenteet

API	Application programming interface eli ohjelmointirajapinta. Ohjelmointirajapinnat ovat palveluja, joilla saa palvelun tarjoamia ominaisuuksia käyttöön.
SVG	Scalable Vector Graphics. Kaksiulotteisten vektorikuvien kuvauskieli, joka perustuu World Wide Web Consortiumin kehittämään avoimeen kuvatiedostostandardiin.
HTML	Hypertext Markup Language. Hypertekstin merkintäkieli. Avoimesti standardoitu kuvauskieli, jolla voidaan kuvata hypertekstiä.
CSS	Cascading Style Sheets. Tyyliohjeiden laji. Käytetään HTML-elemttien ulkoasun määrittelyyn.
AJAX	Asynchronous JavaScript and XML. Tekniikka, jonka avulla voidaan siirtää tietoa selaimen ja palvelimen välillä.

1 Johdanto

Insinöörityön tarkoitus on selvittää, kuinka toisen kartan voi lisätä jo valmiiksi tehtyyn karttarajapintasovellukseen. Työssä suunnitellaan ja toteutetaan karttasovellus, jonka on tarkoitus olla tekijälleen oppimista edistävä prosessi ja toimia pohjana seuraaville karttasovelluksille.

Työssä perehdytään yleisesti verkkokarttoihin ja siihen, mistä karttarajapinnat muodostuvat ja kuinka muun muassa latitude ja longitude selkeästi liittyvät verkkokarttoihin. Lisäksi selvitetään, miten verkkokartat rakennetaan sivustolle laatoitetuista kuvista.

Sovelluksen suunnitteluun kuuluvat sovelluksen tekniset vaatimukset ja sommittelu. Tässä vaiheessa otetaan myös kantaa responsiiviseen verkkosuunnitteluun. Suunnittelun jälkeen esitellään käytännön toteutus, eli itse sovellus. Toteutus sisältää sovelluksen käyttöliittymän rakenteen ja sen, kuinka sovelluksen voi siirtää Google Maps API -sivustolle. Lisäksi käsitellään yleisesti, kuinka karttaan voi tehdä muutoksia, etsiä kartan pisteitä ja muita huomioitavia asioita sovelluksesta.

Lopuksi käsitellään sovelluksen käytettävyyttä, mitä tulisi ottaa huomioon karttasovellusta suunniteltaessa ja kuinka insinöörityönä tehtyä sovellusta voisi hyödyntää tulevaisuudessa.

2 Verkkokartat

Verkkokarttojen luomiseen kuuluu niiden suunnittelu, luominen ja jakelu internetissä. Vuonna 1993 esiteltiin teknologia, joka mahdollisti karttojen esittämisen internetissä. Tämä teknologia mahdollisti dynaamisen sisällön ja hyperlinkitetyt kuvat. Verkkokarttojen kehitys oli suuri edistysaskel kartoitusalalla. Ennen verkkokarttojen läpimurtoa kartoitus oli rajoittunut vain muutamiin yrityksiin ja toiminta edellytti kalliita työkaluja ja osaavaa työvoimaa. Verkkokarttojen kehittymisen myötä kartoitusalalle on tullut enemmän toimijoita ja vanha ala on muuttunut. Kehittyneet ja helposti saatavilla olevat verkkokarttatyökalut ovat tehneet suurelle määrälle kehittäjiä mahdolliseksi tuottaa karttoja verkkoon, ja paperisten karttojen käyttö on vähentynyt. Verkkokartat mahdollistavat kustannustehokkaamman toiminnan ja helpon karttojen päivittämisen. Verkkokartoille pystytään myös luomaan henkilökohtaista sisältöä ja jakamaan muun muassa paikkatietoja. [1.]

Verkkokarttoja on kahta eri tyyppiä: staattiset ja dynaamiset kartat. Ensimmäiset verkkokartat olivat pääasiassa staattisia teknisten rajoitteiden vuoksi. Staattiset verkkokartat ovat pelkkiä kuvia. Niitä voi verrata tavallisiin paperikarttoihin, eikä niiden katselua voi optimoida näytöllä. Nykypäivänä verkkokartat voivat olla täysin interaktiivisia ja tuottaa tietoa monista tietolähteistä, eli ne ovat dynaamisia. Dynaamiset verkkokartat luodaan joka kerta uudelleen, kun verkkosivulla aloitetaan vierailu. Kartalla näytettävät tiedot tallennetaan tietokantaan, josta ne ovat haettavissa yhä uudelleen. Kartat ovat myös vuorovaikutteisia ja käyttäjät voivat esimerkiksi lähentää ja loitontaa karttaa ja vaihtaa tarkasteltavaa aluetta. [1.]

2.1 Web-karttojen historia

Interaktiiviset kartat ovat olleet yksi merkittävimmistä ja käytännönläheisimmistä internetin käyttökohteista. Web-karttojen historia on syvästi yhteydessä internetin kehitykseen. Ensimmäisen web-karttapalvelun julkaisi Xerox PARC vuonna 1993. Xerox PARC Map Viewer teki kehittäjille mahdolliseksi luoda omaa sisältöä karttoihin. [1.]

Seuraavina vuosina alkoi tulla enemmän karttapalveluita, jotka mahdollistivat oman sisällön lisäämisen karttasovellukseen. Tämä toiminta tunnetaan nykyään paremmin nimellä mashup. Yksi alkuaikojen tunnetuimmista karttapalveluista oli TIGER Mapping

Service, jonka loi Census Bureaus. Ensimmäinen laajalti käytetty vapaaseen lähdekoodin perustuva karttapalvelu julkaistiin vuonna 1995 Minnesotan yliopistossa. Seuraavana vuonna julkaistiin ensimmäinen kaupallinen verkossa toimiva karttapalvelu nimeltä MapQuest. Samoihin aikoihin julkaistiin myös Terraserver USGS:n, Microsoftin ja HP:n yhteistyönä. Terraserverissä käytettiin ilmakuvia ja USGS:n grafiikkaa. [1.]

2000-luvun alkupuolella esiteltiin OpenStreetMap, joka oli vapaaseen lähdekoodiin perustuva web-pohjainen maailmankarttaprojekti. Vuonna 2005 Google julkaisi Google Mapsin, joka mahdollisti helpon karttaintegraation olemassa oleville verkkosivuille. [1.]

2.2 Karttarajapinnat

Rajapinnat ovat palveluja, joilla saa palvelun tarjoamia ominaisuuksia käyttöön omalle sivustolle. Ne sisältävät muun muassa toimintoja ja luokkia, joita voi suoraan käyttää omassa koodissa. Verkkokartta-API:t tuovat kartan näyttämiseen tarvittavat toiminnot ja kuvat kartan näyttämiseen. Rajapintaa käytettäessä ei tarvitse kirjoittaa matalan tason toimintoja, vaan voidaan keskittyä vaikeampien toimintojen tekemiseen ja kartan ulkoasun suunnitteluun. [2.]

Google Maps

Google Maps on internetin karttapalveluista suosituin. Sillä on pelkästään mobiililaitteilla 250 miljoonaa käyttäjää, ja Google Mapsin sovellusliittymä on käytössä yli 800 000 sivustolla. Se tarjoaa ilmakuvaa, ja yli 120 kaupungissa käytössä ovat 45 asteen kuvat, jotka mahdollistavat tietojen katselun sopivalla tavalla. Google on kiinnittänyt erikoishuomiota mobiilikäyttöön. Se mahdollistaa sovellusten ja palveluiden kehittämisen perustuen käyttäjän sijaintiin. [3.]

Google Mapsista on kaksi versiota. Mapsin sovellusliittymä on tarkoitettu sivustoille, jotka ovat maksuttomia. Tätä versiota suositaan sovelluskehityksessä, testauksessa ja ei-kaupallisissa sovelluksissa. Mapsin sovellusliittymä on kutsurajoitettu, eikä sillä ole kaikkia Googlen tarjoamia moninaisuuksia. [3.]

Google Maps for Workin sovellusliittymä taas tarjoaa kaikki lisäominaisuudet ja tukipalvelun. Se tarjoaa myös tarkemman enimmäisresoluution Mapsin staattisessa sovellus-

liittymässä ja Street View -kuvasovellusliittymässä. Se myös antaa suuremman skaalaus kertoimen. [3.] Kuvassa 1 näkyy, mitä eroa sovellusliittymäversiolla on.

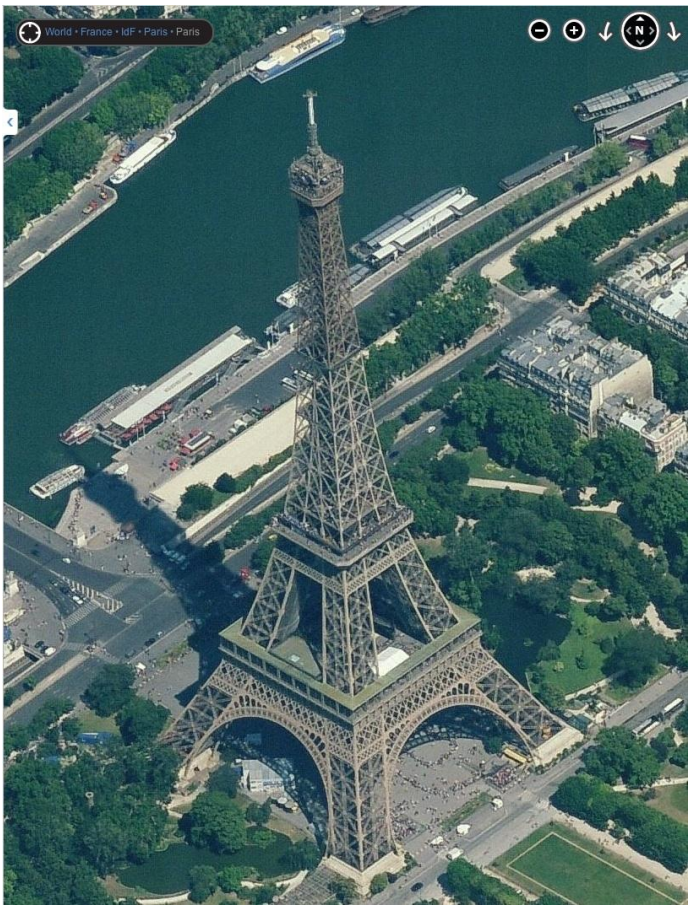
Ominaisuudet	Mapsin sovellusliittymä	Maps for Workin sovellusliittymä
Street View	✓	✓
Geokoodaus-verkkopalvelu	2 500 pyyntöä päivässä	Niin monta kuin vaaditaan. Voi muuttua ostetun määrän mukaan.
Reittiohjeet-verkkopalvelu	2 500 pyyntöä päivässä ja 10 reittipistettä pyyntöä kohden	Niin monta kuin vaaditaan. Voi muuttua ostetun määrän mukaan.
Etäisyysmatriisi-verkkopalvelu	100 elementtiä kyselyä kohden 100 elementtiä 10 sekunnissa 2 500 elementtiä päivässä	Niin monta kuin vaaditaan. Voi muuttua ostetun määrän mukaan.
Korkeusprofiili-verkkopalvelu	2 500 pyyntöä päivässä ja 25 000 näytettä päivässä	Niin monta kuin vaaditaan. Voi muuttua ostetun määrän mukaan.
Mapsin staattisen sovelluslıittymän enimmäisresoluutio	640 x 640	2048 x 2048
Mapsin staattisen sovelluslıittymän enimmäisskaalaus	2X	4X
Street View -kuvasovelluslıittymän enimmäisresoluutio	640 x 640	2048 x 2048
Tuki	**Mapsin sovellusliittymä**	**Maps for Workin sovellusliittymä**
Google Mapsin sovelluslıittymän kehittäjäresurssit	✓	✓
Palvelutasosopimus		✓
Tekninen tuki		✓
Tukiportaali ja käyttöraportointi		✓
Käyttötapoja	**Mapsin sovellusliittymä**	**Maps for Workin sovellusliittymä**
Maksuton ja yleisesti käytettävissä	✓	✓
Sisäiset käyttöönotot		✓
Upottaminen maksullisiin ohjelmistoihin ja sovelluksiin		✓
Google Mapsin sisältävien palveluiden jälleenmyynti		✓
Mainonnan hallinta		✓
Yksityinen omaisuuden seuranta		✓

Kuva 1. Google Mapsin sovelluslıittyminen ominaisuudet [3].

Microsoft Bing

Microsoft julkaisi oman Bing-karttansa kilpailemaan Googlen karttasovelluksen kanssa. Se on erityisesti keskittynyt tarjoamaan käyttäjille paikallista dataa. Sen uniikki ominaisuus on ’bird’s eye’ -näkymä. Se mahdollistaa ilmakuvien tarkastelun monista näkökulmista. Muita Bingin ominaisuuksia ovat muun muassa ’Streetside’, jossa kartan käyttökokemus on kuin itse seisoisi kartalla, liikennetiedotus ja kansainvälinen Geocodingin. [4; 5.]

Bingin karttarajapinnan ilmainen käyttö on rajattu 125 000 sessioon tai 500 000 dollarin liikevaihtoon vuodessa. Ilmainen versio ei tuo Bingin ’bird’s eye’ -näkymää eikä ’streetside’ -karttoja. ’Streetside’ on käytettävissä koulutus- ja ei-voittoa-tavoittelevissa käyttäjätileissä. [5.] Kuvassa 2 on esimerkki bird’s eye -näkymästä.



Kuva 2. Bird’s eye -näkymässä kohdetta voi tarkastella monesta kulmasta [4].

MapQuest

MapQuest oli yksi ensimmäisistä karttojen tarjoajista verkossa. Sen tavoitteena on rohkaista käyttämään ilmaisia karttoja. Se on ainoa yritys, joka antaa mahdollisuuden valita lisensoitujen karttojen ja ilmaisten karttojen väliltä. Lisensoituja karttoja voidaan käyttää myös ilmaiseksi ilman mitään rajoitteita karttanäkymässä. Se kuitenkin rajoittaa käyttäjien tekemät kutsut, haut ja paikannukset 5 000 kappaleeseen päivässä. [5.]

Ilmainen versio käyttää OpenStreetMapsia, ilmaisia satelliitteja ja ilmakuvadataa, eikä sitä ole rajoitettu millään tavalla. Siitä kuitenkin puuttuvat reititys- ja liikennepalvelut. Yleisimmistä ei-avoimen lähdekoodin kartta-API-rajapinnoista poiketen sitä voidaan myös käyttää salasanalla suojatuissa ja ei-julkisissa verkkosovelluksissa. [5.]

OpenLayers

Open Source Geospatial Foundation on OpenLayers API -hankkeen takana. OpenLayersia käytetään yleensä OpenStreetMapin karttojen ja datan kanssa, mutta se toimii myös Google Mapsin ja Microsoft Bingin karttojen kanssa. Se on erittäin tehokas ja joustava API-rajapinta, joka on suunniteltu käytettäväksi varsinkin kehittyneissä karttasovelluksissa. Käytettävyydeltään se on muita monimutkaisempi ja raskaampi. Sen API-rajapinta tarjoaa paljon ominaisuuksia, mutta sen käyttö mobiililaitteissa voi olla hankalaa, koska se on suunniteltu ennen niiden suosion kasvua. [5; 6.]

Modest Maps

Nimensä mukaisesti Modest Maps (kuva 3) on suunniteltu tarjoamaan karttasovellusten perusominaisuudet pienessä ja siistissä paketissa. Se on alun perin suunniteltu käyttämään Adoben Flashia, mutta se on myöhemmin käännetty JavaScriptille. [5.]

Kuva 3. Modest Maps -karttasovelluksen näkymä [5].

Polymaps

Toisin kuin yleisimmät kartta-API-rajapinnat, Polymaps käyttää SVG-kuvaformaattia karttojen piirtämiseen, ja se tuottaa datan suoraan selaimelle. Yleisimpien kartta-

API:ien kartat tehdään laatoista, joiden päälle vasta ladataan maantieteelliset tiedot, kuten merkit ja reitit. Polymaps-API:ssa tiedot voidaan ladata suoraan esimerkiksi GeoJSON-formaattia käyttäen ja sulauttaa se suoraan karttaan. [5.]

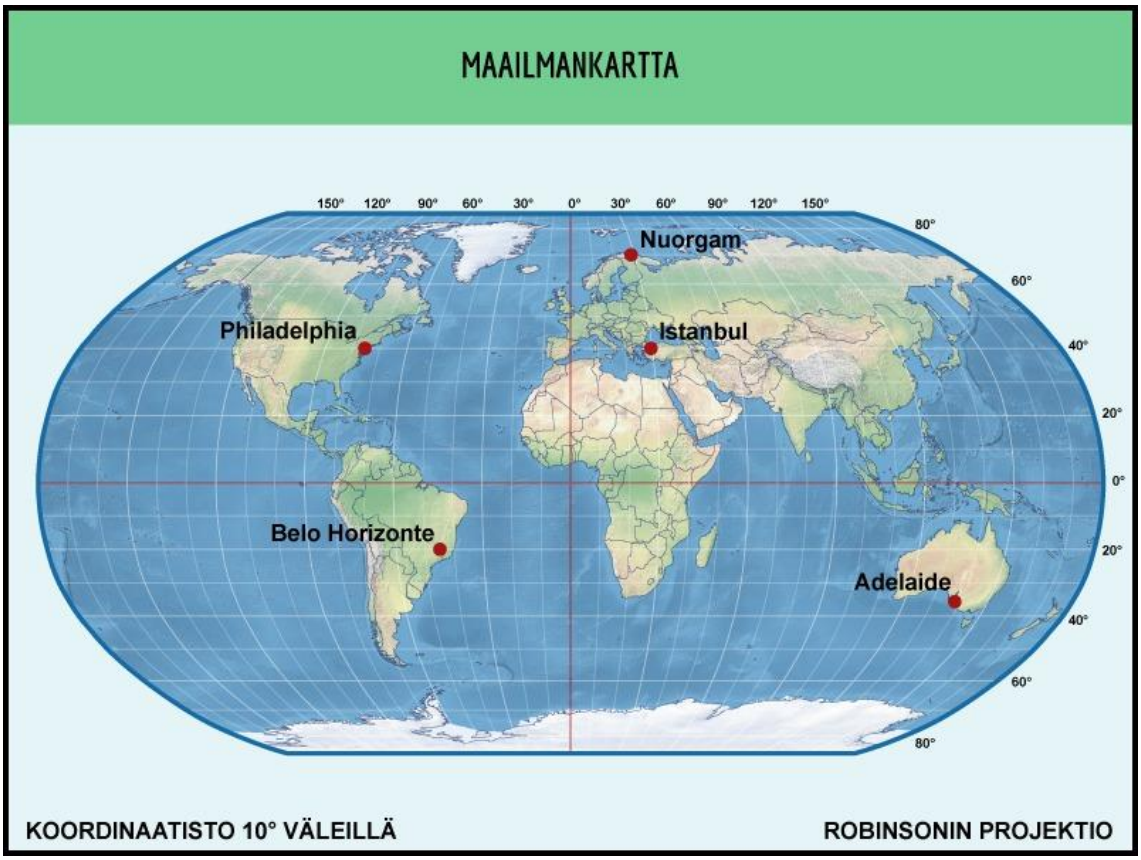
SVG:n käyttö mahdollistaa CCS-tyylien muokkauksen muun muassa kartalla oleville teille ja maarajoille. CCS-tyylien muuttaminen onnistuu myös niin sanotusti lennosta. SVG mahdollistaa myös siirtymien ja animaatioiden käyttämisen karttasovelluksessa. [5.]

Polymapsissa on helppo visualisoida suuria paikkatietoaineistoja. Esimerkiksi asukastiheys, rikostilastot ja vaalikäyttäytyminen voidaan helposti näyttää kartalla. Tässä karttasovelluksessa paikkadatatietojen käsittelytapa mahdollistaa kartan dynaamiset muutokset. Polymapsia eivät rajoita kiinteät lähennä- ja loitonna-tasot, kuten karttoja jotka, perustuvat laatoitettuihin kuviin. Koska SVG on resoluutiosta riippumaton, se toimii hyvin myös suurempiresoluutioisilla näytöillä. [5.]

Polymapsin heikkoutena voidaan pitää huonoa tukea mobiiliselaimilla. Vaikka mobiililaitteiden tuki kohentuisi, se vaatii silti paljon laskentatehoa käytettävältä laitteelta. Sillä ei myöskään ole tukea vanhemmille Internet Explorer -selaimille. [5.]

2.3 Leveys- ja pituusasteet

Suomeksi latitude ja longitude tunnetaan nimillä leveys- ja pituusaste, mutta ne ovat vakiintuneet englanninkielisillä nimillään verkkokarttoihin. Leveys- ja pituuspiirit muodostavat kartalle verkkomaisen pinnan kuvan 4 tapaan. Niitä hyödyntämällä jokaiselle paikalle kartalla voidaan antaa oma leveys- ja pituusaste, jotka muodostavat uniikin paikkatunnisteen. [7; 8.]



Kuva 4. Maailmankarttakoordinaatisto [7].

Kartalla vaakatasossa kulkevia viivoja kutsutaan leveyspiireiksi. Niiden kehät ovat samansuuntaisia, joten ne eivät koskaan kohtaa toisiaan. Pisin leveyspiiri sijaitsee päiväntasaajalla, ja se jakaa maapallon pohjoiseen (N) pallonpuoliskoon ja eteläiseen (S) pallonpuoliskoon. Päiväntasaajalla ollessa asteluku on 0, ja navoilla asteluvut ovat 90. [7.]

Kartalla pystytasossa eli navalta navalle kulkevat viivat ovat nimeltään pituuspiirejä. 0-pituuspiiri kulkee Lontoon Greenwichin kautta. Se jakaa maapallon läntiseen (W) pallonpuoliskoon ja itäiseen (E) pallonpuoliskoon. Pituuspiirin suurin mahdollinen asteluku on 180. Läntiset ja itäiset pituuspiirit saavuttavat suurimmat mahdolliset asteluvut Tyynellämerellä. [7.]

Latitude- ja longitude-pisteen määrittäminen

Uniikin pisteen määrittämisessä kartalla käytetään vielä tänäkin päivänä asteita, minuutteja ja sekunteja. Asteet kertovat millä, piirillä kuljetaan, ja minuutit ja sekunnit ker-

tovat metrit tältä piiriltä. Esimerkiksi Pariisin latitude olisi asteesityksenä 48°51'24"N. Koska leveyspiirit ovat eripituisia, myös minuuttien ja sekuntien pituudet vaihtelevat riippuen siitä, millä piirillä juuri sillä hetkellä ollaan. Esimerkiksi 48. leveyspiirin kohdalla yksi minuutti on noin 1,15 kilometriä ja yhden sekunnin pituus on noin 30 metriä. Pisteen koordinaatti voidaan myös antaa desimaaliasteena, joka Pariisin tapauksessa on 48,856. Longitude-piste saadaan saamaa tyyliä käyttäen, ja yhdessä latituden kanssa saadaan tarkka piste maapallolta. [9.]

2.4 Google Maps API:n ominaisuudet

Google Maps API on hyvin dokumentoitu kirjasto, ja siitä on paljon esimerkkejä internetissä. Googlen omilta kehittäjäsivuilta löytyy esimerkkejä, kuinka saada erinäisiä näkymiä ja toimintoja käyttöön kartalla. Tämän lisäksi erikoisempia ja pidemmälle vietyjä esimerkkejä on paljon, jo pelkästään etsimällä hakukoneella asiaa, jota haluaa kartalla tehtävän. [10.]

Peruskarttatyylejä Google Mapsissa on neljä: tiekartta, satelliittikuva, maastokartta ja hybrid, joka on sekoitus tavallisesta ja satelliittikuvasta. Peruskarttatyylien lisäksi voidaan luoda omia karttatyylejä. Google Maps tarjoaa myös 45 asteen perspektiivin ilmakuvia, joita voi katsella eri kuvakulmista. Tämä ei ole maailmanlaajuinen ominaisuus, vaan kohteet, joissa se on käytössä, on rajattu. [10.]

Tietoa sisältävät kerrokset lisätään Google Maps API:ssa ”layers”-ominaisuudella. Nämä kerrokset koostuvat joko yhdestä tai useammasta erillisestä osasta. Näitä osia kuitenkin muokataan aina yhtenä kokonaisuutena. Kokonaisuudet esitetään yleensä yhtenä uutena kerroksena kartan päällä, ja niitä muokataan aina kerroskokonaisuuksittain. Yksittäisiin osiin kerroskokonaisuudessa harvemmin pystytään vaikuttamaan. [10.]

Google Maps API tarjoaa seuraavat kerrokset:

- liikenne
- julkinen liikenne
- pyöräilyreitit
- sää ja pilvikartta
- lämpökartta
- Google Maps Data -kerros

- panoramio
- Fusion Table-kerros
- KML-kerros.

Kartalla tapahtumien seuraamiseen käytetään ”event listenereitä”, joiden suomenkielinen nimi on tapahtumankuuntelija. Ne ”seuraavat” sitä, jos tietty tapahtuma tapahtuu kartalla ja siihen tapahtumaan on kytketty jokin toiminto. Jos tapahtuma toteutuu kartalla, toiminto laukeaa. Esimerkiksi voidaan tarkastella kartalla tapahtuvaa hiiren liikettä. Seuraavassa muutama Google Maps API:n ’event listenereistä’:

- Center_changed seuraa kartan keskikohdan muutoksia.
- Zoom_changed seuraa lähennä- ja loitonna-tason muutoksia.
- Mousemove, mouseout ja mouseover seuraavat hiiren liikkeitä kartalla.
- Click, dblclick ja rightclick seuraavat painalluksia kartalla.

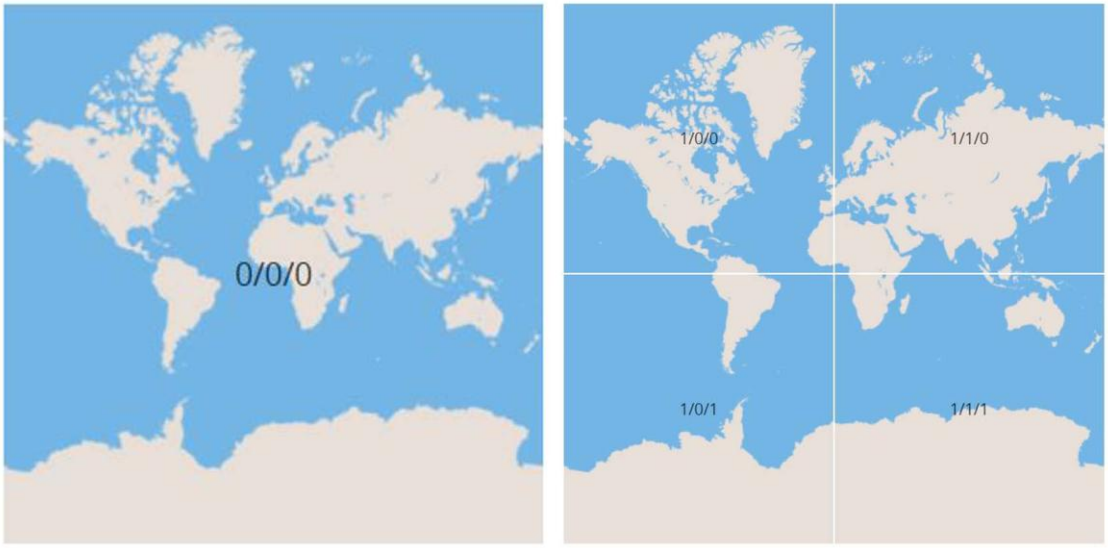
Näiden toimintojen käytön suunnittelussa on hyvä ottaa huomioon mobiililaitteet, koska esimerkiksi hiiren liikkeiden seuranta ei laukaise mitään toimintaa mobiililaitteilla. Keskikohdan muutosten seuranta taas toimii niin selainkäytössä kuin myös mobiililaitteilla. [10.]

Google Maps API:ssa on mahdollisuus lisätä informatiivisia objekteja. Lisätyt objektit voivat olla esimerkiksi yksittäisiä pisteitä, viivoja tai alueita. Ne esitetään kartalla peittokuvina. Objektiin lisätään paikkakoordinaatit antamalla sille latitude- ja longitude-arvot. Näin kohde pysyy paikoillaan, vaikka karttaa liikuttaa tai lähentää ja loitontaa. Yleisimmin käytössä on yksittäinen sijaintiin sidottu objekti, jota kutsutaan ’markkeriksi’. Tämä markkeri voi sisältää sellaisia tietoja kuin esimerkiksi paikan nimen ja linkin paikan sivustolle. Google Maps API:ssa on myös mahdollisuus käyttää symboleita, kuten neliö ja ympyrä. Kartalle on mahdollista myös piirtää mielivaltaisia kuvioita käyttämällä polygoneja kuvion piirtämiseen. [10.]

2.5 Laatoitettu karttakuva

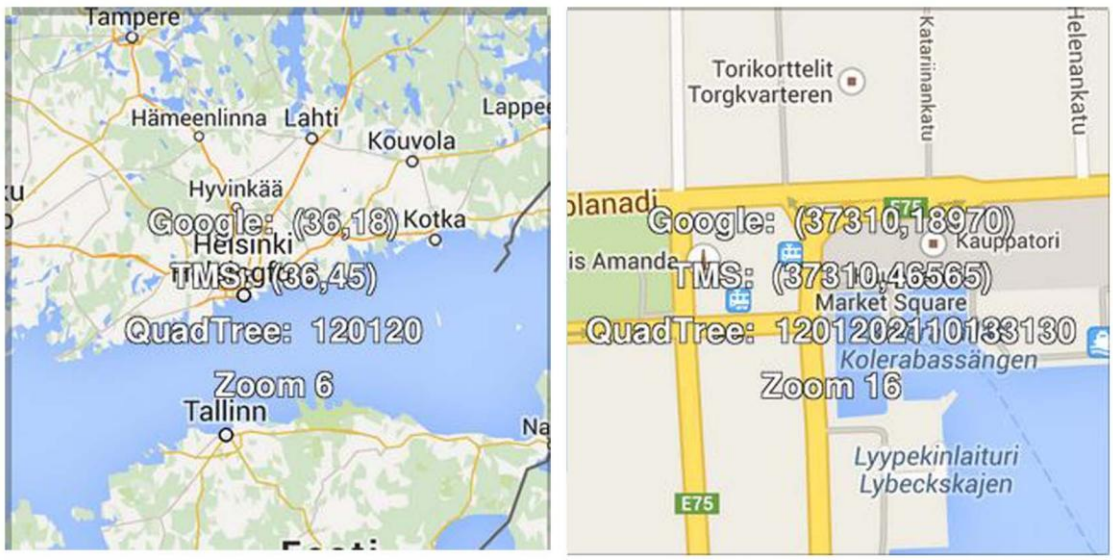
Yksittäinen katutasosta asti yksityiskohtaisesti tarkka maailmankartta olisi miljardien pikseleiden kokoinen. Tällainen kuva olisi liian suuri ladattavaksi tai pidettäväksi muistissa. Tästä syystä kartta on tehty pienemmistä laatoista. Tyypillisesti yhden laatan

koko on noin 256 * 256 pikseliä ja laatat on sijoitettu saumattomasti vierekkäin luomaan illuusion erittäin suuresta yksittäisestä karttakuvasta. [11.]



Kuva 5. Maailmankartta kahdessa eri zoomaustasossa [11].

Lähennä- ja loitonna-tasot määrittävät sen, kuinka monesta kuvasta kartta on muodostunut. Esimerkiksi kuvassa 5 on lähennä- ja loitonna-tasolla nolla yksi laatta ja lähennä- ja loitonna-tasolla yksi neljä laattaa. Tämä jatkuu siten, että lähennä- ja loitonna-tasolla kaksi on 16 laattaa ja tasolla 3 on 64 laattaa. Koska laattojen määrä kasvaa valtavasti suuremmilla lähennä- ja loitonna-tasoilla, näytetään vain ne laatat kerrallaan, jotka mahtuvat näytölle. Vain tarvittavien laattojen lataaminen nopeuttaa kartan latautumista, ja laattoja voidaan helposti tallentaa välimuistiin. [11.]



Kuva 6. Suuremmalla zoomaustasolla laatassa on enemmän yksityiskohtia [11].

Laattojen määrän kasvuun vaikuttaa myös se, että lähempänä katutasoa pitää pystyä kartalla näyttämään enemmän yksityiskohtia. Tämä näkyy hyvin kuvassa 6. Näiden eri tasojen ja miljoonien kuvatiedostojen hallintaan käytetään yksinkertaista koordinaatistoa z/x/y. Z tarkoittaa lähennä- ja loitonna-tasoa, x- ja y-koodinaatit sijaintia neliöverkostossa. [11.]

3 Karttasovelluksen suunnittelu

Sovelluksen suunnitteluvaiheessa käydään läpi ne ominaisuudet ja toiminnot, joita sovellukselta halutaan. Tässä vaiheessa suunnitellaan myös sivuston ulkoasu eli sivuston sivut sommitellaan. Näin ei tarvitse tehdä ”turhaa” työtä toteutusvaiheessa, vaan sivusto toteutetaan suunniteltujen ominaisuuksien, toimintojen ja sommittelun perusteella.

3.1 Sovelluksen määrittely

Tässä projektissa tehtävän karttasovelluksen on tarkoitus tutkia ja selvittää, kuinka Google Maps API:iin voidaan liittää omia karttoja ja sen toiminnoilla ohjata sivuston muita toimintoja. Tarkoitus on saada Google Maps API:n tarjoamia ominaisuuksia käyttöön omalla kartalla. Sovellukseen tulee kaksi esimerkkisovellukseen lisätyistä kartoista. Toisessa lisätään yksittäinen kuva ja toisessa esimerkissä kartta muodostetaan

laatoitetuista karttakuvista. Sovelluksessa käytetään valmista sivustopohjaa, ja sen tarkoitus on toimia niin selaimella kuin mobiililaitteilla.

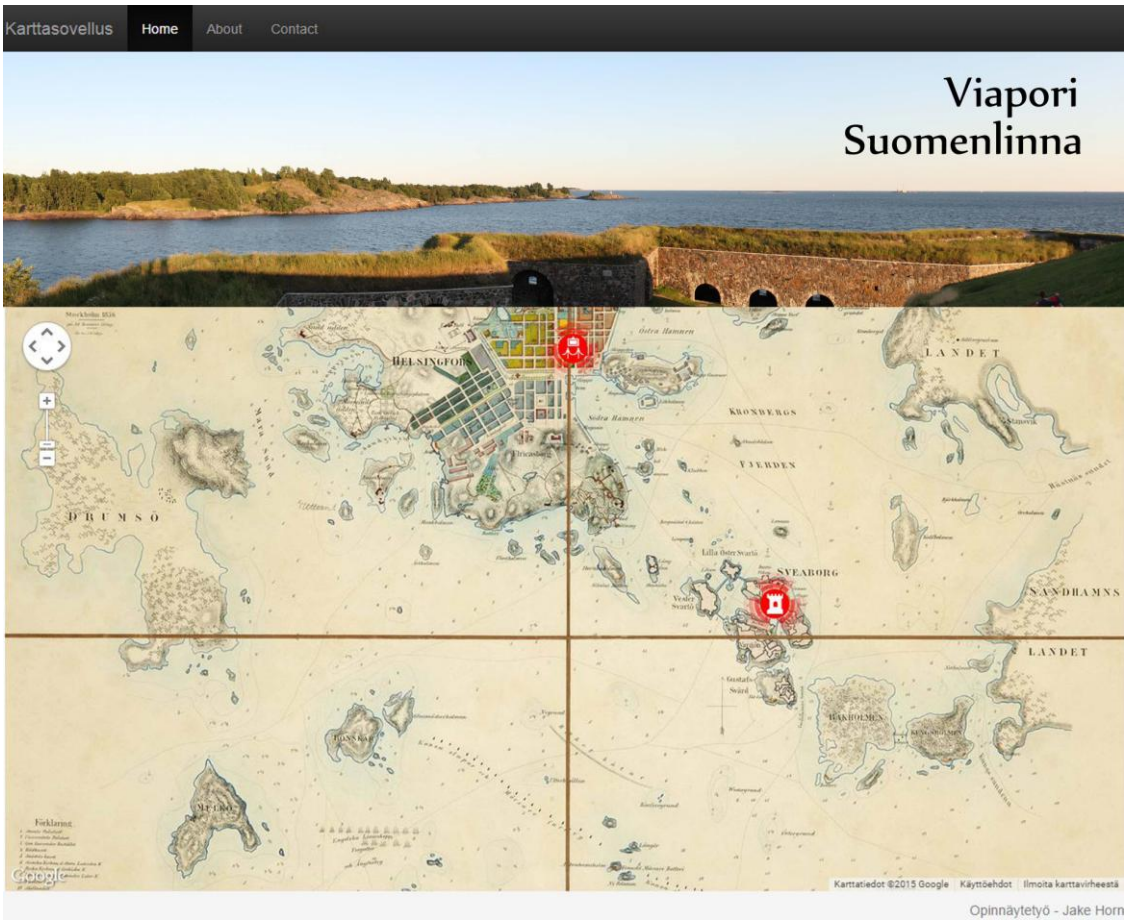
Karttasovellukselta halutaan seuraavia ominaisuuksia:

- oma kartta Google Mapsissa
- kartta-alueen rajaus oman kartan kokoiseksi
- ulkopuolisen alueen ilmoittaminen
- lähennä- ja loitonna-tasojen rajoittaminen
- markkereiden asettaminen kartan kohteisiin
- Responsiivinen ulkoasu
- mahdollisuus vaihtaa kartan näkymä
- navigaatiovalikko
- kuvakaruselli.

Kuvakaruselliin tulee kuvia kohteista, jotka on merkitty karttaan markkereilla. Karusellissa on automaattinen pyörintä, mutta diaa voi vaihtaa myös painamalla kartalta markkeria. Tällöin markkeri alkaa pomppia ja karusellin pyörintä pysähtyy. Karusellin saa jälleen pyörimään vaihtamalla diaa karusellista. Karusellin dioihin tulee linkit kohteiden sivuille.

3.2 Sommittelu

Sovelluksen sommittelu tehdään kuvankäsittelyohjelmalla. Sommittelussa määritellään sivuston värimaailma ja elementtien sijainnit sivustolla. Sen tavoite on saada aikaiseksi mahdollisimman hyvä kuvaus siitä, miltä sivuston valmiina tulisi näyttää. Tässä sovelluksessa käytetään valmista verkkosivuston pohjaa, joten se voidaan ottaa sommittelun pohjaksi. Kuvassa 7 näkyvät sivuston osat. Yläreunaan tulee valikkopalkki, keskelle jää alue, johon tulee kuvakaruselli sekä kartta ja alareunaan tulee alaviiteosuus.



Kuva 7. Suunnitelma karttasovelluksen sivun ulkoasusta.

3.3 Responsiivinen verkkosuunnittelu

Mobiililaitteiden käytön yleistymisen jälkeen verkkosuunnittelussa on tullut tärkeäksi ottaa huomioon mobiililaitteilla tapahtuva verkkosivujen selailu. Nykyään verkkosivuja selataan enemmän mobiililaitteilla kuin selaimilla. Huonolla mobiililaitetoimivuudella menetetään paljon käyttäjiä. Responsiivisella verkkosuunnittelulla tarkoitetaan sivuston skaalautumista kaikenkokoisille näytöille. Responsiivisen verkkosuunnittelun etuna on, että mobiililaitteille ei tarvitse tehdä erillistä sivustoa. Näin ollen sivuston ylläpito ja päivitys on helpompaa ja halvempaa. [12.] Kuvassa 8 on esimerkki sivustosta, jossa responsiviivisuus on otettu huomioon.

Kuva 8. Responsiivinen käyttöliittymä toimii kaikilla laitteilla [12].

4 Sovelluksen toteutus

Sovellus aloitetaan luomalla kuvan 9 kaltainen kansiorakenne ja tarvittavat perustiedostot. Index.html:ään tehdään verkkosivuston runko, ja siellä viitataan käytettäviin kirjastoihin, tyylitiedostoihin ja JavaScript-tiedostoihin. Kirjastoja voidaan käyttää suoraan verkosta, tai ne voidaan ladata omaan projektikansioon ja käyttää sieltä.

Kuva 9. Sovelluksen kansiorakenne ja käytettävien kirjastojen lisäys.

4.1 Käyttöliittymän HTML–rakenne

Sovelluksen sivuston rakenteena on käytetty Bootstrapin sivuilta ladattavaa verkkosivupohjaa, koska sillä saadaan aikaiseksi vähällä vaivalla responsiivinen käyttöliittymän runko. Bootstrap on tehokas käyttöliittymätyökalu, jolla sivuston kehittäminen on helppoa ja nopeaa. Se sisältää myös yhtenäisiä suunnittelumalleja käyttöliittymien osiin, kuten typografiaan, lomakkeisiin, nappeihin, taulukoihin ja valikoihin. Yhdessä JavaScriptin kanssa sillä voidaan myös luoda dynaamista sisältöä, kuten esimerkiksi kuvakaruselleja. [13.]

Kuvassa 10 näkyy Bootstrapin sivuilta haettu HTML–sivun rakenne. Tässä rakenteessa sivusto on jaettu kolmeen osaan, joilla on yhteinen luokka eli ”class”. ”Class”-tunniste ei ole yksilöllinen, vaan se voidaan antaa monelle elementille. Tässä tapauksessa luokalle kirjoitetut tyylit ovat voimassa jokaisessa elementissä, jossa luokka on määritelty. Sivuston yläosaan tulee navigointivalikko, keskelle muodostuu sivuston sisältöosuus ja sivuston alareunaan tulee alaviiteosio. Tyylitiedostossa näille kaikille osille on annettu perustyylit, joissa on muun muassa määritelty elementin koko, väri ja sijainti sivustolla. [14.]

index.html

```
<body>
  <nav class="navbar navbar-inverse navbar-fixed-top">
    <div class="container">
      <div class="navbar-header">
        <button type="button" class="navbar-toggle collapsed" data-toggle="collapse"
          data-target="#navbar" aria-expanded="false" aria-controls="navbar">
          <span class="sr-only">Toggle navigation</span>
          <span class="icon-bar"></span>
          <span class="icon-bar"></span>
          <span class="icon-bar"></span>
        </button>
        <a class="navbar-brand" href="kartta.html">Karttasovellus</a>
      </div>
      <div id="navbar" class="collapse navbar-collapse">
        <ul class="nav navbar-nav">
          <li class="active"><a href="#">Home</a></li>
          <li><a href="#about">About</a></li>
          <li><a href="#contact">Contact</a></li>
        </ul>
      </div><!--/.nav-collapse -->
    </div>
  </nav>

  <div class="container">
  </div><!-- /.container -->
  <footer class="footer">
    <div class="container">
      <p class="text-muted">Opinnäytetyö - Jake Horn</p>
    </div><!-- /.container -->
  </footer>
</body>
```

styles.css

```
html {
    height: 100%
}

body {
    height: 100%;
    margin: 0;
    padding: 50px 0 40px 0;
}

.container {
    height: 100%;
    padding: 0 !important;
}

.footer {
    position: absolute;
    bottom: 0px;
    width: 100%;
    height: 40px;
    background-color: #F5F5F5;
}

footer .container .text-muted {
    margin: 10px 0px;
    float: right;
}
```

Kuva 10. Bootsrapin sivuilta haettu html-rakenne ja CSS-tyylit.

Toimiakseen Bootstrap tarvitsee jQuery-kirjaston. JQuery on luotu helpottamaan JavaScriptin käyttämistä verkkosivustolla. Se yhdistää usein käytettyjä JavaScript-

funktioita, joita sitten voidaan kutsua yhdellä jQuery-funktiolla. Sillä esimerkiksi voidaan tehdä helposti AJAX-kutsuja ja muokata dynaamisesti sivuston rakennetta. [15.]

4.2 Google Maps API sivustolle

Ensimmäinen tavoite lähdettäessä tekemään karttasovellusta on saada kartta näkymään sivustolla. Jotta Google Maps API on käytettävissä, se pitää lisätä käytettävien kirjastojen joukkoon sivulla 15 olevan kuvan 9 osoittamalla tavalla. HTML:n tiedostoon tulee div, johon kartta tullaan asettamaan. Diville annetaan id, ja tässä se on ”map-canvas”. Script.js:ään tulee kartan alustus, jossa määritellään kartan aloitus lähennä– ja loitonna-taso ja kartan aloituskeskipiste. Keskipiste annetaan latitude- ja longitude-arvoilla. Toiminnon lopussa kartta asetetaan index.html:ssä määriteltyyn div-elementtiin. Tyylitiedostoon styles.css lisätään vielä karttaa koskevia tyylejä, jolla voidaan muun muassa vaikuttaa kartan kokoon. [11.] Koodiesimerkit ovat kuvassa 11.

index.html

```
12    <div id="map-canvas"></div>
```

script.js

```
1  var map;
2  function initialize() {
3    var mapOptions = {
4      zoom: 8,
5      center: new google.maps.LatLng(-34.397, 150.644)
6    };
7    map = new google.maps.Map(document.getElementById('map-canvas'),
8        mapOptions);
9  }
10
11 google.maps.event.addDomListener(window, 'load', initialize);
```

styles.css

```
1  #map-canvas {
2      height: 100%;
3      margin: 0px;
4      padding: 0px
5    }
```

Kuva 11. Tarvittavat koodit Google Mapsin saamiseksi sivulle.

4.3 Yksittäisen karttakuvan lisääminen

Staattisen kartan saamiseksi näkymään kartalla täytyy tehdä muutoksia JavaScript-koodiin. Skriptitiedoston alkuun lisätään muuttuja kuvan 12 rivin 1 mukaisesti. Se kuvaa kartalle lisättävää uutta ”kerrosta”. Tämän jälkeen rivillä 2 kerrotaan, että tämä sovellus käyttää Google Maps API:n overlayView’ta. [11.]

```
1  var overlay;
2  USGSOverlay.prototype = new google.maps.OverlayView();
3
4  function initialize() {
5    var mapOptions = {
6      zoom: 11,
7      center: new google.maps.LatLng(62.323907, -150.109291),
8      mapTypeId: google.maps.MapTypeId.SATELLITE
9    };
10
11   var map = new google.maps.Map(document.getElementById('map-canvas'), mapOptions);
12
13   var swBound = new google.maps.LatLng(62.281819, -150.287132);
14   var neBound = new google.maps.LatLng(62.400471, -150.005608);
15   var bounds = new google.maps.LatLngBounds(swBound, neBound);
16
17   var srcImage = 'kartta.jpg';
18
19   overlay = new USGSOverlay(bounds, srcImage, map);
20 }
21
22 function USGSOverlay(bounds, image, map) {
23
24   this.bounds_ = bounds;
25   this.image_ = image;
26   this.map_ = map;
27
28   this.div_ = null;
29
30   this.setMap(map);
31 }
```

```
32 USGSOverlay.prototype.onAdd = function() {
33
34   var div = document.createElement('div');
35   div.style.borderStyle = 'none';
36   div.style.borderWidth = '0px';
37   div.style.position = 'absolute';
38
39   var img = document.createElement('img');
40   img.src = this.image_;
41   img.style.width = '100%';
42   img.style.height = '100%';
43   img.style.position = 'absolute';
44   div.appendChild(img);
45
46   this.div_ = div;
47
48   var panes = this.getPanes();
49   panes.overlayLayer.appendChild(div);
50 };
51 USGSOverlay.prototype.draw = function() {
52
53   var overlayProjection = this.getProjection();
54
55   var sw = overlayProjection.fromLatLngToDivPixel(this.bounds_.getSo
56   var ne = overlayProjection.fromLatLngToDivPixel(this.bounds_.getNo
57
58   var div = this.div_;
59   div.style.left = sw.x + 'px';
60   div.style.top = ne.y + 'px';
61   div.style.width = (ne.x - sw.x) + 'px';
62   div.style.height = (sw.y - ne.y) + 'px';
63 };
64
65 google.maps.event.addDomListener(window, 'load', initialize);
```

Kuva 12. Koodiesimerkki staattisen kartan saamiseksi Google Mapsiin.

Seuraavaksi on hyvä määrittää uusi aloituspiste kartalle. Tämä tehdään kuvan 12 rivillä 7. Seuraavaksi määritellään alue, jolle kartta muodostetaan. Se annetaan latitude- ja longitude-pisteinä swBound- ja neBound-objekteihin. SwBoundin kaksi ensimmäistä kirjainta tulevat sanoista S = south ja W = west eli southwest, joka ilmansuunnissa suomeksi olisi lounas. Samoin neBoundin kaksi ensimmäistä kirjainta muodostavat N = north ja E = east eli northeast, joka ilmansuuntana suomeksi on koillinen. [11.]

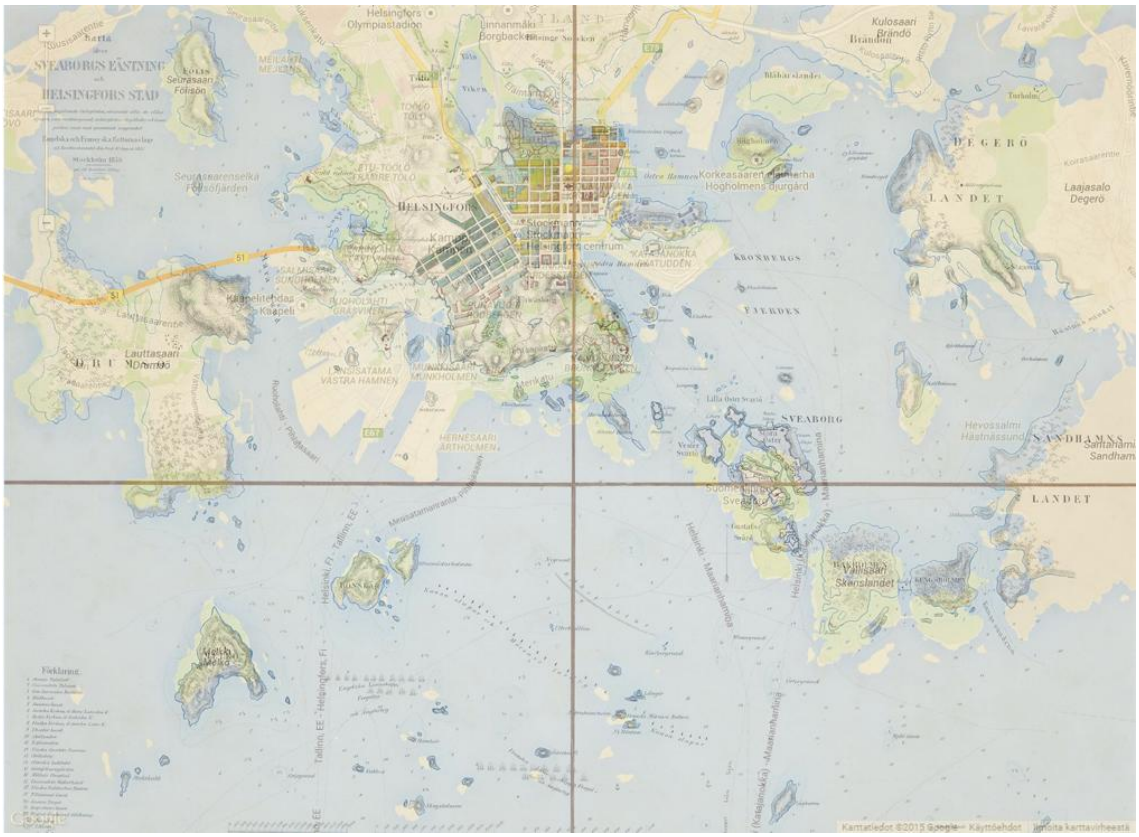
Kuvan 12 rivillä 17 määritellään käytettävän kuvatiedoston tiedostopolku. Tämän jälkeen rivillä 19 asetetaan rivillä 1 esiteltyyn objektiin tiedot, jotka lähetetään funktiolle. Tiedot sisältävät määritellyt rajat, kuvatiedoston ja viittauksen kartasta. Rivin 22 funktiossa alustetaan objektin arvot ja kutsutaan kuvan asettamista. Seuraavassa funktiossa määritellään kuvalle ja sitä ympäröivälle diville tyylit. Sitä seuraavassa funktiossa käytetään hyväksi määriteltyjä rajojen swBound ja neBound arvoja, joista lasketaan edellisessä funktiossa määritellyn divin sijainti ja koko. Tämän lisäyksen jälkeen lisätty kartta näkyy Google Mapsin päällä määritetyllä kohdalla. [11.]

4.4 Karttaan tehtävät muutokset

Vanhassa käytettävässä kartassa mittasuhteet ja paikkojen sijainnit eivät välttämättä täsmää täysin kohdalleen uuden ja tarkan kartan kanssa. Näin ollen käytettävää kart-

taa joudutaan mahdollisesti työstämään kuvankäsittelyohjelmalla, jotta kohteet vastaisivat paremmin oikeita kohteiden latitude- ja longitude-arvoja. Kartan korjailu mahdollistaa paremman käytettävyyden karttasovelluksen ominaisuuksiin, kuten oikeiden paikkatietojen käytön ja zoomauksen.

Paikkakohtien kohdistus aloitetaan menemällä karttasovellukseen ja hakemalla sieltä kohta, johon ollaan lisäämässä omaa karttaa. Tämän jälkeen kohdasta otetaan kuva esimerkiksi Windowsin Snipping-työkalulla. Tallentamisen jälkeen kuva viedään kuvankäsittelyohjelmaan. Tämän karttapohjan päälle tuodaan oma kartta, jota halutaan muokata. Omaan karttaan lisätään läpinäkyvyyttä siten, että nähdään pohjakartan kohteet kuvan 13 tapaan. Tämän jälkeen oma kartta kohdistetaan mahdollisimman hyvin pohjakarttaan. Pohjakartassa saatetaan joutua muuttamaan kartan kokoa. Tämän jälkeen nähdään mahdolliset eroavaisuudet ja kartan kohteita voidaan siirtää käyttämällä kuvankäsittelyohjelman työkaluja.



Kuva 13. Kaksi karttaa päällekkäin.

4.5 Karttapisteiden etsiminen

Kartan sijainnin määrittely saattaa olla hankalaa, koska oikeiden rajapisteiden määrittely arvaamalla on vaikeaa. Tähän voidaan hyödyntää JavaScript-funktiota, jolla latitude- ja longitude-pisteitä voidaan hakea ja näyttää HTML-elementissä. Kuvan 14 funktiossa määritellään kohta, jossa hiiren painallus tehdään kartalla ja lähetetään paikan latitude- ja longitude-arvot HTML-elementtiin. Tätä hyödyntämällä tarvittavat arvot swBound, neBound ja kartan keskikohta löytyvät helpommin. Tämän jälkeen tarvittavat pisteet voidaan sijoittaa JavaScriptiin oikeille kohdille.

script.js

```
17  google.maps.event.addListener(map,'click',function(event) {
18  document.getElementById('latlongclicked').value = event.latLng.lat() + ', ' + event.latLng.lng()
19  })
```

index.html

```
65  Latitude ja Longitude:
66  <input id="latlongclicked" type="text" style="width: 400px;">
```

Kuva 14. Koodiesimerkki latitude-ja longitude-arvojen saamiseksi kartalta.

4.6 Alueen rajaaminen

Koska käytettävä oma kartta on vain osa maailmankartan sovelluksesta, voidaan sovelluksen käyttämä alue rajata JavaScript-funktiolla. Rajauksen tekeminen kannattaa aloittaa karttapohjan asetusten tarkastelusta. Tässä vaiheessa tarkistetaan, mikä on aloitus lähennys- ja loitonnus-taso. Tähän on hyvä valita taso, jossa oma kartta on mahdollisimman iso siten, että alla olevaa karttaa ei näkyisi ollenkaan. Tämä arvo asetetaan zoom-parametriin. Tässä vaiheessa lisätään myös maxZoom ja minZoom. MinZoom on sama kuin aloitettava lähennys- ja loitonnus-taso ja maxZoom taso määritetään sellaiseksi, että käytettävä kartta on vielä tarkka.

Rajojen määrittelyyn voidaan käyttää samoja pisteitä, joita käytettiin oman kartan sijoittamisessa. Jotta pisteet saataisiin käyttöön JavaScript-koodissa, ne pitää lukea objektista ja katsoa, kuinka ne on sinne tallennettu. Tämä voidaan tehdä laittamalla koodiin console.log(swBound);. Console.log on hyödyllinen työkalu, kun halutaan tietää, mitä koodissa tapahtuu, ja se on erittäin suosittu työkalu virheenkorjauksessa. Kun console.log on asennettu koodiin ja sivusto suoritetaan uudelleen selaimella, selaimen kon-

soliin tulevat objektin sisältämät tiedot. Tästä nähdään, kuinka tieto on tallennettu objektiin. Tässä tapauksessa swBoundin latitude löytyy muuttujasta k ja longitude muuttujasta D. Näitä arvoja voidaan yhdessä swBoundin kanssa käyttää luettaessa arvoja objektista. Esimerkiksi swBound.k antaa latitude-arvon ja swBound.D longitude-arvon.

Funktion laukaisemiseen käytetään tapahtumankuuntelijaa. Kuvassa 15 näkyyvät valitut event listenerit ’center_changed’ ja ’zoom_changed’. Laukaistavan funktion alussa määritellään uudet raja-arvot ja keskipiste. Jos uudet arvot ovat alkuperäisten rajojen sisäpuolella, tulee uusi keskipiste käyttöön. Muussa tapauksessa edellinen pätevä keskiipiste jää voimaan.

```
28
29  google.maps.event.addListener(map, 'center_changed', checkBoundsCenter);
30  google.maps.event.addListener(map, 'zoom_changed', checkBoundsZoom);
31
32  function checkBoundsCenter() {
33
34    var currentBounds = map.getBounds();
35    var current_ne_lng = currentBounds.getNorthEast().lng();
36    var current_ne_lat = currentBounds.getNorthEast().lat();
37    var current_sw_lng = currentBounds.getSouthWest().lng();
38    var current_sw_lat = currentBounds.getSouthWest().lat();
39
40    var currentCenter = map.getCenter();
41    var centerX = currentCenter.lng();
42    var centerY = currentCenter.lat();
43
44    if (current_ne_lng > neBound.D) centerX = centerX-(current_ne_lng-neBound.D);
45    if (current_ne_lat > neBound.k) centerY = centerY-(current_ne_lat-neBound.k);
46    if (current_sw_lng < swBound.D) centerX = centerX+(swBound.D-current_sw_lng);
47    if (current_sw_lat < swBound.k) centerY = centerY+(swBound.k-current_sw_lat);
48
49    map.panTo(new google.maps.LatLng(centerY,centerX));
50
51  }
52
```

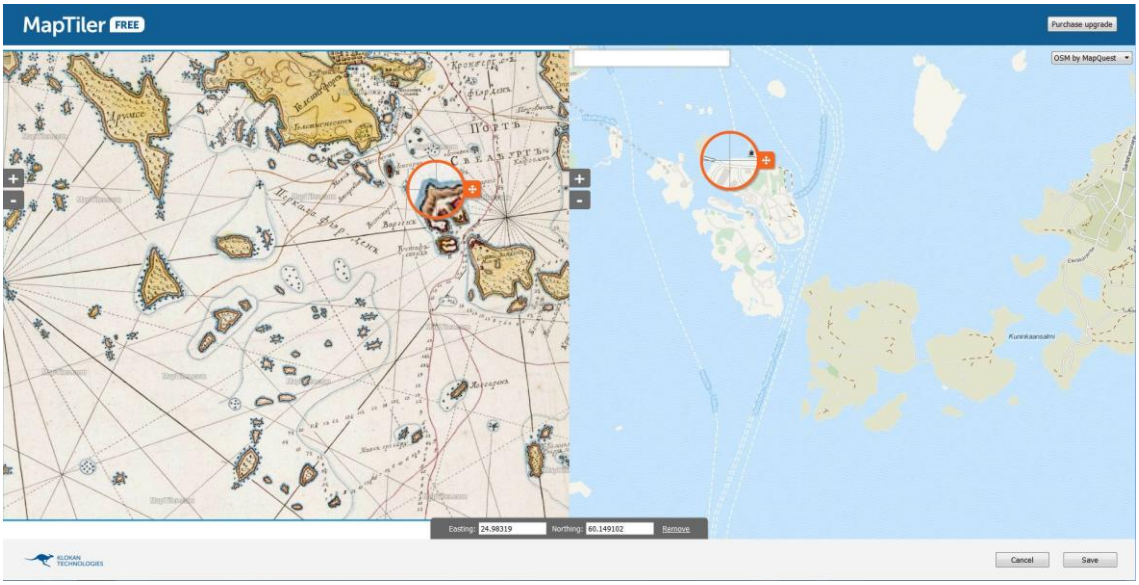
Kuva 15. Koodiesimerkki kartan rajaamiselle.

4.7 Kuvan osiin laatoittaminen

Isoissa karttakuvissa on syytä harkita käytettävän kartan laatoittamista. Laatoitettu kuva käyttäytyy samalla tavalla kuin Google Mapsin API:n oma kartta. Kartan laatoittamiseen on muutamia eri vaihtoehtoja. Yleisimmin käytössä oleva sovellus on nimeltään MapTiler. Siinä on ilmainen sovellus, mutta maksamalla saadaan enemmän ominaisuuksia käyttöön. Sovelluksessa kartasta luodaan laattoja ja saadaan esimerkkikoodit yleisimpiin karttasovelluksiin. [16.]

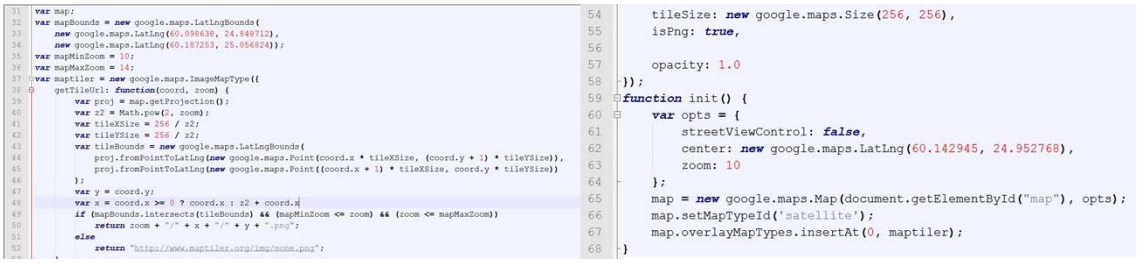
Kuvan laatoittaminen aloitetaan valitsemalla kuvatiedosto, josta laatat halutaan luoda. Seuraavassa vaiheessa valitaan tapa, jolla kartta kohdistetaan karttasovellukseen. Tässä tapauksessa halutaan kohdistaa kartta visuaalisesti karttasovelluksen kanssa (kuva 16). Visuaalisessa kohdistamisessa etsitään samoja pisteitä omasta kartasta ja

karttasovelluksesta. Mitä enemmän pisteitä asetetaan, sitä tarkempi käytettävästä kartasta tulee. Tärkeää on laittaa mahdollisimman paljon reunapisteitä, koska niillä varmistetaan, että kartta ei pääse kummasti leviämään. Pisteiden asettelun jälkeen seuraavaksi valitaan kansio, johon laattakuvat tallennetaan. Tämän jälkeen ohjelma luo laatat ja esimerkkikoodit yleisimmille karttasovelluksille. Tämän jälkeen laatoitettu kartta on katsottavissa selaimella. [16.]



Kuva 16. Kartan kohdistaminen visuaalisesti.

MapsTilerin esimerkkikoodissa kaikki HTML-, CSS- ja JavaScript-koodit ovat yhdessä tiedostossa. Laatoitetun kartan tapauksessa ei tuoda yksittäistä kuvaa div-elementin sisälle vaan luodaan kokonaan uusi Google Map-kuvatyyli, joka tuodaan peittokuvana kartan päälle. JavaScriptin alussa määritellään alue mapBounds, jolla laatat näytetään. Seuraavaksi määritellään lähennä- ja loitonna-tasoväli, jolla laatoitettu kartta näytetään. Se on määritelty MapTiler-sovelluksessa ennen kuvien luontia. Tässä tapauksessa minimi lähennä- ja loitonna-taso on 10 ja maksimitaso 14. Seuraavaksi on luotu getTileUrl, jossa on funktio, joka määrittää laattakuvan, joka missäkin kohtaa karttaa näytetään. Jos alue on rajatun alueen ulkopuolella, näytetään yleislaatta, joka tässä tapauksessa on tyhjä. Funktiossa määritellään lisäksi kuvalaattojen koko ja niiden peittävyys (opacity). Pienentämällä peittävyysarvoa saadaan alla oleva kartta näkyviin ja voidaan tehdä vertailua, kuinka hyvin oma kartta nyt vastaa pohjalla olevaa karttaa. [16.] Koodiesimerkki on kuvassa 17.



```
var map;
var mapBounds = new google.maps.LatLngBounds(
    new google.maps.LatLng(60.098638, 24.848712),
    new google.maps.LatLng(60.187253, 25.056824));
var mapMinZoom = 10;
var mapMaxZoom = 14;
var maptiler = new google.maps.ImageMapType({
    getTileUrl: function(coord, zoom) {
        var proj = map.getProjection();
        var z2 = Math.pow(2, zoom);
        var tileXSize = 256 / z2;
        var tileYSize = 256 / z2;
        var tileBounds = new google.maps.LatLngBounds(
            proj.fromPointToLatLng(new google.maps.Point(coord.x * tileXSize, (coord.y + 1) * tileYSize)),
            proj.fromPointToLatLng(new google.maps.Point((coord.x + 1) * tileXSize, coord.y * tileYSize))
        );
        var y = coord.y;
        var x = coord.x >= 0 ? coord.x : z2 + coord.x
        if (mapBounds.intersects(tileBounds) && (mapMinZoom <= zoom) && (zoom <= mapMaxZoom))
            return zoom + "/" + x + "/" + y + ".png";
        else
            return "http://www.maptiler.org/img/none.png";
    },
    tileSize: new google.maps.Size(256, 256),
    isPng: true,

    opacity: 1.0
});
function init() {
    var opts = {
        streetViewControl: false,
        center: new google.maps.LatLng(60.142945, 24.952768),
        zoom: 10
    };
    map = new google.maps.Map(document.getElementById("map"), opts);
    map.setMapTypeId('satellite');
    map.overlayMapTypes.insertAt(0, maptiler);
}
```

Kuva 17. MapTiler-sovelluksesta saatu koodiesimerkki.

4.8 Kartan ulkopuolinen alue

Karttasovelluksen rajoja ei välttämättä tarvitse määritellä, jos halutaan antaa mahdollisuus zoomata oman karttakuvan ulkopuolelle. Tällöin kartan ulkopuolinen alue voidaan joko näyttää sellaisenaan tai tehdä ilmoitus, että ollaan kartan ulkopuolella. Ilmoitus tehdään luomalla yksittäinen laatta, jota toistetaan kartan ulkopuolella.

Laatta luodaan kuvankäsittelyohjelmassa. Laattaan voidaan kirjoittaa esimerkiksi, että kartta ei ole tällä alueella käytettävissä. Kuvan läpinäkyvyyttä voidaan myös muuttaa siten, että alta näkyy Google Mapsin kartta. Kuva tulee tallentaa muodossa, joka tukee kuvan läpinäkyvyyttä.

Tapauksessa, jossa kartta on asetettu div-elementtiin, luodaan uusi Google Maps -kuvatyyli. Uudessa kuvatyylissä määritellään käytettävä laattakuvatiedosto ja laatan koko. Tämä tyyli asetetaan uudelle peittokerrokselle, joka asetetaan Google Maps -kartan päälle. Tämän kerroksen päälle tulee div-elementti, jossa oma kartta on asetettu. Tässä voidaan käyttää kuvaa, jossa tekstin lisäksi tausta voi olla esimerkiksi läpinäkyvä harmaa.

MapTiler-esimerkkikoodissa määritelty tyhjä laatta voidaan korvata omalla kuvalaatalla. Tässä tapauksessa kuvaan voidaan lisätä vain teksti, koska MapTiler-sovelluksesta saatujen tiilien ulkoalue on valmiiksi läpinäkyvä. Kuvassa 18 nähdään, mitä tapahtuu, jos käytetään laattaa, jolle on määritelty väri ja se on vain osittain läpinäkyvä.

Kuva 18. Ulkopuolisessa alueen ilmoittavassa laatassa vain on vain osittainen läpinäkyvyys.

4.9 Markkeri ja karuselli

Karusellin esimerkkirakenne löytyy Bootstrapin sivuilta. Karuselli koostuu kolmesta pääosasta: indikaattorit, diat ja kontrollerit. Indikaattorit ilmaisevat, mikä dia on valittu, ja niistä voi myös valita dian. Dia itsessään on sisältö, joka kulloinkin on näkyvissä. Kontrollereilla voidaan mennä dioissa eteen- ja taaksepäin. Karusellille voidaan myös asettaa diojen vaihtumisnopeus. Asetus tehdään kuvassa 19 vasemmanpuoleisessa koodiesimerkissä rivillä 2, jossa se on asetettu arvoon 10000.

Markkeri luodaan JavaScriptissä. Markkerille annetaan perustiedot, kuten sijainti ja ikonin polku. Perustietojen antamisen jälkeen markkeri näkyy kartalla. Seuraavaksi JavaScript-koodissa määritellään tapahtumankuuntelijat. Niistä toinen laukaisee dian valinnan ja toinen aktivoi markkerin pomppimisen. Kuvassa 19 oikeanpuoleisessa koodiesimerkissä goToSlide-funktio valitsee määritellyn dian ja pysäyttää karusellin rotaation. Rivillä 18 oleva toggleBounce aktivoi markkerin pomppimisen ja toinen painallus pysäyttää sen. Markkerin pomppimisen voi myös lopettaa karusellin indikaattoreista tai kontrollereista. Tämä myös aktivoi karusellin rotaation uudestaan.

```
<div class="bs-example">
    <div id="myCarousel" class="carousel slide" data-interval="10000" data-ride="carousel">
        <!-- Carousel indicators -->
        <ol class="carousel-indicators">
            <li data-target="#myCarousel" data-slide-to="0" class="active start"></li>
            <li data-target="#myCarousel" data-slide-to="1" class="start"></li>
            <li data-target="#myCarousel" data-slide-to="2" class="start"></li>
        </ol>
        <!-- Carousel items -->
        <div class="carousel-inner">
            <div class="active item moi">
                <h2>Slide 1</h2>
                <div class="carousel-caption">
                  <h3>First slide label</h3>
                  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>
                </div>
            </div>
            <div class="item">
                <h2>Slide 2</h2>
                <div class="carousel-caption">
                  <h3>Second slide label</h3>
                  <p>Aliquam sit amet gravida nibh, facilisis gravida odio.</p>
                </div>
            </div>
            <div class="item">
                <h2>Slide 3</h2>
                <div class="carousel-caption">
                  <h3>Third slide label</h3>
                  <p>Praesent commodo cursus magna, vel scelerisque nisl consectetur.</p>
                </div>
            </div>
        </div>
        <!-- Carousel nav -->
        <a class="carousel-control left start" href="#myCarousel" data-slide="prev">
            <span class="glyphicon glyphicon-chevron-left"></span>
        </a>
        <a class="carousel-control right start" href="#myCarousel" data-slide="next">
            <span class="glyphicon glyphicon-chevron-right"></span>
        </a>
    </div>
</div>
```

```
var markerLatLng = new google.maps.LatLng(60.168332, 24.951954);
var iconBase = 'images/icons/';
var marker = new google.maps.Marker({
    position: markerLatLng,
    draggable: false,
    raiseOnDrag: true,
    map: map,
    labelContent: "Kohde",
    labelAnchor: new google.maps.Point(22, 0),
    labelClass: "labels",
    title: 'Kohde on hieno',
    icon: iconBase + 'icon.png'

});
google.maps.event.addListener(marker, "click", function (e) { goToSlide(1);});
google.maps.event.addListener(marker, 'click', toggleBounce);

function toggleBounce() {

    if (marker.getAnimation() != null) {
        marker.setAnimation(null);
    } else {
        marker.setAnimation(google.maps.Animation.BOUNCE);
    }
}
function goToSlide(number) {
   $("#myCarousel").carousel(number);
   $("#myCarousel").carousel('pause');
}

$(".start").on('click', function() {
    $("#myCarousel").carousel('cycle');
    marker.setAnimation(null);
});
```

Kuva 19. Karuselli- ja markkeri-koodiesimerkit.

4.10 Käytettävyys

Insinöörityönä tehdystä karttasovelluksesta tuli suunnitellun mukainen, ja se soveltuu hyvin karttasovelluksen ominaisuuksien esittämiseen. Sovelluksen rakenne ei ole monimutkainen, mutta sillä on hyvä esitellä erinäisiä ratkaisumalleja ja sitä voi käyttää pohjana, kun lähdetään suunnittelemaan uutta karttasovellusta.

Sovelluksen testaaminen on syytä suorittaa mahdollisimman monella eri selaimella ja mobiililaitteella. Jokaisella selaimella on omia erikoisominaisuuksia, ja kaikille toiminnoille ei välttämättä ole tukea. Etenkin vanhemmista Internet Explorer -selaimista puuttuu usein tuki käytettyihin kirjastoihin ja toimintoihin.

Insinöörityössä sovelluksen selaintestaamiseen käytettiin Google Chromen versiota 41.0.2272.101 m, Firefoxin versiota 36.0.4 ja Internet Explorerin versiota 8. Chromessa ja Firefoxissa käytettävyydessä ei ollut minkäänlaisia ongelmia. Tähän suuri syy on se, että sovelluskehitysvaiheessa yleensä käytetään joko Chromea tai Firefoxia. Tällöin niiden toiminnallisuus tulee samalla varmistettua.

Internet Exploreria käytettäessä huomaa heti, että kyseessä on vanhempi selain. Heti sivustolle mentäessä ulkoasu ei vastaa suunniteltua ja latausajat ovat kasvaneet. Tämä johtunee siitä, että Internet Explorer 8 ei tue HTML5:tä, johon Bootstrap pohjautuu.

Tätä asiaa voidaan korjata käyttämällä html5shiv- ja respond-kirjastoja. Niiden tarkoitus on tuoda HTML5-ominaisuuksia sellaisille selaimille, jotka eivät muuten sitä tue. Sivuston ulkoasun korjaamiseen saatetaan myös joutua kirjoittamaan omia tyylejä, jotka koskevat vain Internet Explorerin selaimia. On yleistä, että Internet Explorerin ongelmat huomataan vasta testausvaiheessa, koska sovelluskehitysvaiheessa sitä harvemmin käytetään.

Mobiililaitteilla testaus suoritettiin Nokian 925-, Google Nexus 5- ja Samsung Galaxy S3 mini -puhelimilla. Testauksessa yllättävän huomion sai se, että sovellus toimi hyvin Nokian selaimella. Sivusto latautui yllättävän nopeasti ja oli hyvin käytettävissä. Uusimpana puhelinmallina Google Nexus 5 latasi sivuston nopeimmin ja käyttö oli jouhevaa. Samsung Galaxy S3 minillä testauksessa huomattavaa oli, että sivusto ei toiminut oikein puhelimen Chrome-selaimella, mutta omalla selaimella sivusto toimi moitteetta. S3 minin Chrome-selaimella ulkoasun responsiivisuus ja kartan toiminnot eivät toimineet. Lataamalla Play Storesta Chromen betaversio kartantoiminnot alkoivat toimia, mutta ulkoasuun jäi vielä hiottavaa.

Oman kartan päälle lisäämisessä on parempi käyttää laatoitettua karttaa. Tämä siksi, että varsinkin mobiililaitteilla yksittäisen ison kuvan lataaminen kestää kauan ja vie paljon muistitilaa. Tämän huomaa selvästi karttakuvaa ladattaessa.

4.11 Huomioitavaa

Kun karttasovellusta lähdetään tekemään, on hyvä tutustua eri karttarajapintavaihtoehtoihin. Tätä työtä aloitettaessa ei ollut kuin ”yksi” vaihtoehto, josta valita. Työn edetessä vaihtoehtojen määrä kasvoi. Jos työtä alettaisiin tekemään alusta, tulisi rajapinnan valinta vasta sen jälkeen, kun sovellukselta vaadittavat toiminnot on päätetty. On myös hyvä ottaa huomioon ilmaisten ja maksullisten versioiden erot. Esimerkiksi tässä projektissa openLayers-rajapinta olisi todennäköisesti tuonut aivan yhtä hyvän lopputuloksen kuin Google Maps.

Maptiler-sovelluksen käyttö oli yllättävän hyvä kokemus. Sen ilmaisversio on hyvä ja antaa selvän kuvan sovelluksen toiminnoista. Ilmaisversion käyttö ei tulisi kuitenkaan kysymykseen lopullista työtä tehtäessä, koska sitä käytettäessä lähennä- ja loitonnatasoja ei voida määrittää ja se lisää vesileiman laatoitettuihin kuviin. PhotoShop-

kuvankäsittelyohjelmaan on myös kartanlaatoitusskripti, mutta siinä lähennä- ja loitonna-taso valinnat ovat vieläkin huonommat.

Lisäämällä sivustolle karuselli saadaan lisää dynaamisuutta. Kun tieto esitetään karusellissa kartalla olevien informatiivisten ikkunoiden sijasta, kartan ulkoasu pysyy siistinä ja selkeänä. Tällöin jää myös pois informatiivisten ikkunoiden sulkeminen, mikä parantaa käytettävyyttä.

4.12 Hyödyntäminen

Omaa karttaa karttasovelluksessa voidaan hyödyntää yrityksen verkkosivuilla, jos halutaan tuoda personoitua ilmettä sivuille. Tämä voi myös esimerkiksi olla museo, joka haluaa esitellä vanhoja karttoja ja laittaa kohdetietoja kartalle niiden oikeille paikoille koordinaatteja hyväksi käyttäen. Oikeita kohdetietoja hyödyntämällä käyttäjän on esimerkiksi helppo navigoida haluttuun paikkaan.

Tulevaisuudessa tämäntyyppistä karttasovellusratkaisua voitaisiin käyttää tarinan kertomiseen kartalla. Tarinakarttasovelluksessa kartan ympärillä kerrottaisiin historiallisista tapahtumista ja kartta osoittaisi kulloisenkin tarinan kohdan tapahtumapaikan. Tämän lisäksi kartalla voisi myös navigoida ja tutustua historiallisiin kohteisiin tarkemmin.

5 Yhteenveto

Karttapintavaihtoehtoja on monia. Tämä seikka on hyvä ottaa huomioon, kun karttasovelluksen suunnittelu aloitetaan. Karttarajapinta tulee valita sen mukaan, mitä toimintoja sovellukselta odotetaan ja kuinka paljon käyttäjiä sovelluksella oletetaan olevan. Nykypäivänä on myös tärkeää ottaa huomioon mobiililaitteet suunnittelu- ja toteutusvaiheissa. Kaikki karttarajapinnan toiminnot eivät välttämättä toimi suoraan mobiililaitteilla. Responsiivisella käyttöliittymäsuunnittelulla säästetään aikaa ja rahaa sovelluksen ylläpitovaiheessa. Tällöin ei tarvitse päivitellä erikseen selain- ja mobiililaiteversioita.

Karttasovellukseen lisätty oma kartta antaa sivustolla olevalle kartalle uniikkia ilmettä ilman, että menetettäisiin karttarajapinnan toimintoja. Tällaisen toteutuksen tekemistä

kannattaa harkita sellaisissa tilanteissa, kun halutaan sivuston erottuvan muista tai tuoda alueen historiaa esille kartan muodossa. Sivuston dynaamisuutta voidaan lisätä karusellilla ja markkereilla. Karusellissa voidaan näyttää kuvia ja tekstiä, jotka vaihtuvat tietyin väliajoin. Markkereilla voidaan kartalla esittää kohteita ja niitä painamalla laukaista toimintoja. Näiden avulla käyttäjälle tulee rikkaampi käyttäjä kokemus sivustosta.

Google Maps API on hyvin dokumentoitu rajapinta, ja siitä on paljon esimerkkejä Googlen kehittäjäsivustolla. Niistä voi katsoa mallia ja hakea ideoita omaan karttasovellukseen. Koska Google Maps -kartta koostuu laatoitetusta karttakokonaisuudesta, on päälle lisättävä kartta hyvä myös laatoittaa. Yksittäisen karttakuvan lisäys onnistuu myös, mutta sen lataaminen ja tallentaminen välimuistiin on raskasta. Laatoitetussa kartassa välimuistiin tarvitsee ladata vain ne laatat, jotka sillä hetkellä ovat kartalla näkyvissä. Tällä tavalla kartan latautumista saadaan nopeutettua.

Insinöörityönä toteutettu karttasovellustyö onnistui suunnitelmien mukaisesti. Sitä voidaan hyödyntää pohjana uusille karttasovelluksille ja esittämään esimerkiksi karttasovellusten ominaisuuksia.

Lähteet

1 Web Mapping. Verkkodokumentti. The Google Map Makerpedia team. <https://sites.google.com/site/mapmakerpedia/maps-101/web-mapping>. Luettu 24.3.2015.

2 Sterling, Quinn, Dutton, John A. 2014. What is a web mapping API? Verkkodokumentti. <https://www.e-education.psu.edu/geog585/node/714>. Luettu 24.3.2015.

3 Google Maps for Working-sovellusliittymä. Verkkodokumentti. Google. <https://www.google.com/work/mapsearth/products/mapsapi.html>. Luettu 1.4.2015.

4 Bing Maps Platform Features. Verkkodokumentti. Microsoft. <http://www.microsoft.com/maps/product/features.aspx>. Luettu 1.4.2015.

5 Guide to Google Maps API – and great alternatives. 2014. Verkkodokumentti. Creativeblog. <http://www.creativebloq.com/web-design/google-maps-api-7122779?page=1>. Luettu 2.4.2015.

6 Ing Jim. 2012. Lightwight Maps for Mobile, Part 1: Introduction to Maps APIs and Libraries. Verkkodokumentti. <http://devblog.blackberry.com/2012/05/lightweight-maps-for-mobile-part-1>. Luettu 3.4.2015.

7 Karttatietoutta. 2014. Verkkodokumentti. Peda.net. <https://peda.net/kannonkoski/e-opin-oppikirjat/amerikka2/karttatietous>. 5.4.2015.

8 Rouse, Margaret. 2007. Latitude and longitude. Verkodokumentti. <http://whatis.techtarget.com/definition/latitude-and-longitude>. Luettu 5.4.2015.

9 Briney, Amanda. 2015. Latitude. Verkkodokumentti. <http://geography.about.com/od/locateplacesworldwide/a/latitude.htm>. Luettu 5.4.2015.

10 Google Developers. Verkkodokumentti. Google. <https://developers.google.com>. Luettu 7.4.2015.

11 How to make web maps work. Verkkodokumentti. Mapbox. <https://www.mapbox.com/guides/how-web-maps-work/#why-tiles>. Luettu 7.4.2015.

12 Olander, Ilkka. 2015. Verkkosuunnittelun trendit 2015 – Mitä sinun tulee tietää?. Verkkodokumentti. <http://sometek.fi/verkkosuunnittelun-trendit-2015-mita-sinun-tulee-tietaa/>. Luettu 8.4.2015.

13 Bootstrap Introduction. 2015. Verkkodokumentti. Tutorial republic. <http://www.tutorialrepublic.com/twitter-bootstrap-tutorial/bootstrap-introduction.php>. Luettu 10.4.2015.

14 Getting started. Verkkodokumentti. Bootstrap. <http://getbootstrap.com/getting-started/>. Luettu 10.4.2015.

15 jQuery Introduction. Verkkodokumentti. w3school. <http://www.w3schools.com/jquery/jquery_intro.asp>. Luettu 10.4.2015.

16 Go beyond Google Maps with your own maps. 2015. Verkkodokumentti. Klockan Technologies GmbH. <http://www.maptiler.com>. Luettu 10.4.2015.