Haaga-Helia
ammattikorkeakoulu Oy

Luvuista kuviksi

D3.js-kirjasto osana journalistista datavisualisointiprosessia.

Lasse Leipola

Opinnäytetyö
Tietojenkäsittelyn koulutusohjelma
2016

Tekijä(t)
Lasse Leipola

Koulutusohjelma
Tietojenkäsittelyn koulutusohjelma

Opinnäytetyön otsikko
Luvuista kuviksi: D3.js-kirjasto osana journalistista datavisualisointiprosessia.

Sivu- ja liitesivumäärä
49 + 20

Opinnäytetyön otsikko englanniksi
From Figures to Figures: D3.js-library as a part of journalistic process of data visualization.

Tämän opinnäytetyön tavoitteena on selvittää, miten hyvin datavisualisointiin tarkoitettu D3.js-kirjasto soveltuu journalistiseen käyttöön Vihreä Lanka -lehden kaltaisessa pienessä toimituksessa. Opinnäytetyön ulkopuolelle on rajattu datan jalostaminen, visualisointien esteettisyys sekä sisällöllisen merkittävyyden tarkempi journalistinen pohdinta.

Selvityksen perusteella kokenut D3.js-osaaja pystyy tekemään näyttäviäkin visualisointeja muutamassa tunnissa. Järkevän kokonaisuuden kannalta tämä kuitenkin edellyttää, että D3.js-toteutuksen koetaan tuovan jotain lisäarvoa. D3.js ei ole helppo tai nopea työkalu visualisointien tekemiseen vaan yksinkertainen pylväs- tai viivadiagrammi on huomattavasti helpompaa ja nopeampaa tehdä Excelillä tai Illustratorilla ja liittää se juttuun kuvana.

Teknisen haastavuutensa vuoksi D3.js soveltuukin melko huonosti pienen toimituksen käyttöön, ellei toimittajilla ole sattumalta tarvittavaa osaamista esimerkiksi harrastuneisuuden tai aiempien työtehtävien myötä.

Yleisesti ottaen ja edellä mainituista haasteista huolimatta D3.js soveltuu kuitenkin hyvin verkkosivuilla julkaistavien visualisointien toteuttamiseen, jos niihin halutaan interaktiivista toiminnallisuutta tai mahdollisuus hyödyntää päivittyvää dataa. D3.js-visualisoinneista saa melko helposti responsiivisia, jolloin ne joustavat saumattomasti verkkosivun muun sisällön mukana.

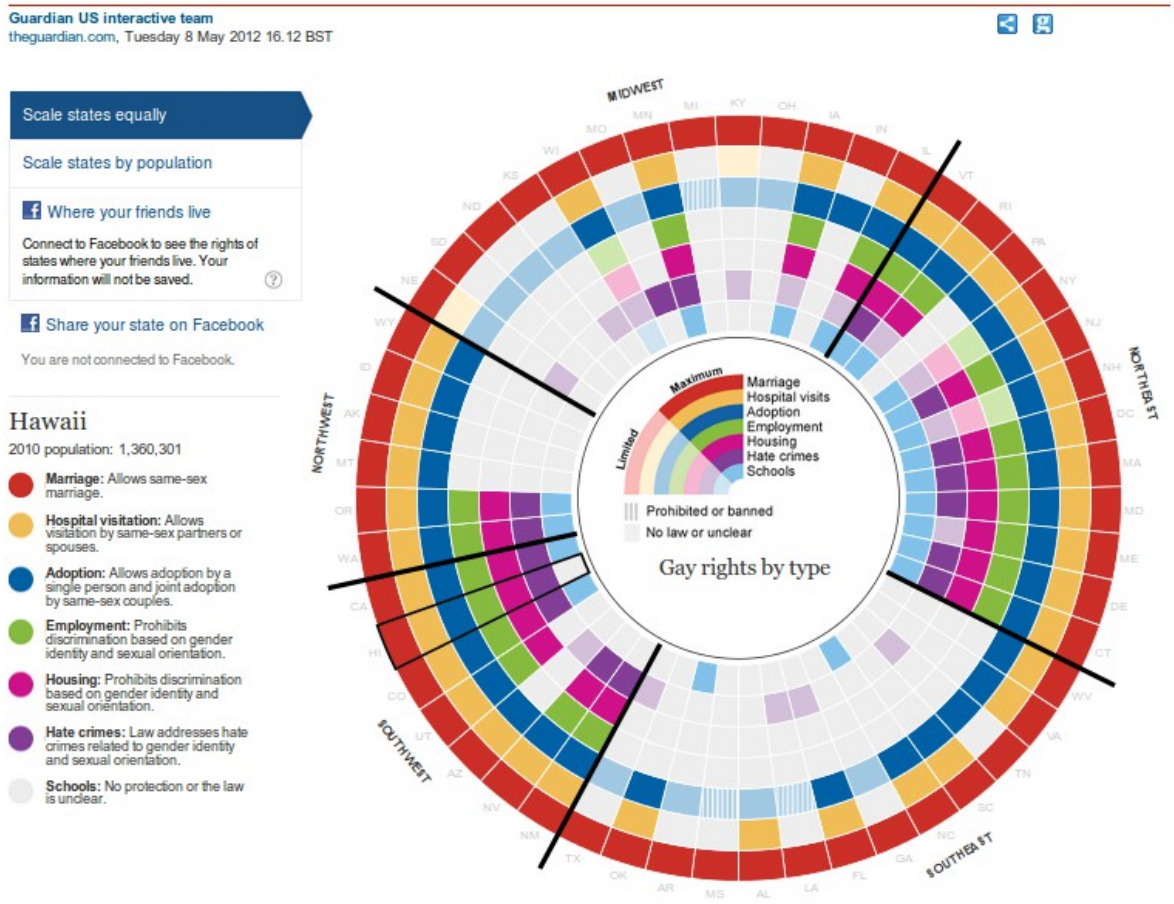
Asiasanat
datajournalismi, visualisointi, ohjelmointi, verkkojulkaiseminen, tilastot

Sisällys

1 Johdanto

Erilaisten tietoaineistojen visualisointi on ollut jo pitkään osa journalismia (Klein 2016). Esimerkiksi taloussivuilla on perinteisesti julkaistu pörssikursseja tai korkotason muutoksia kuvaavia viivadiagrammeja. Työvälineiden ja työtapojen kehittyminen sekä julkaisutoiminnan siirtyminen internetiin on kuitenkin vienyt datajournalismin ja -visualisoinnit uudelle tasolle (Kuutti & Uskali 2016, 22).

Suuret kansainväliset sanomalehdet, kuten Guardian (2012) ja New York Times (Ashkenas, Bloch, Carter & Cox 2012), ovat verkossa julkaistavan datajournalismin ja datavisualisointien edelläkävijöitä. Vähitellen visualisointeja on alettu tekemään pienemmissä toimituksissa (Vihreä Lanka 2016a) ja blogeissa (Kosonen 2016). Kuvassa 1 on esimerkki Guardianissa julkaistusta datavisualisoinnista, joka kertoo, miten seksuaalivähemmistöjen oikeudet ovat toteutuneet Yhdysvaltojen eri osavaltioissa.



Kuva 1. Guardianin interaktiivinen visualisointi. (Guardian 2012.)

Yksi tehokkaimmista ja suosituimmista apuvälineistä tehtäessä verkkosivuilla julkaistavia datavisualisointeja on nimeltään D3.js tai lyhyemmin D3 (Skau 2013). Esimerkiksi New

York Times käyttää D3:a päivittäin (Wilson 2015). D3 on JavaScript-kirjasto, jonka nimi on lyhenne sanoista Data Driven Documents eli vapaasti suomennettuna ”datan muodostamat dokumentit”. Yksinkertaistetusti voisi kuvata, että D3:lle syötetään numeerista dataa, josta se tekee ohjelmoijan ohjeiden mukaista grafiikkaa verkkosivuille. (Bostock, Heer & Ogievetsky 2011.)

1.1 Opinnäytetyön tarkoitus ja tavoite

Tämän opinnäytetyön tavoite on kartoittaa D3.js-kirjaston hyödyllisyyttä pienen toimituksen käytössä. Tavoitteen saavuttamiseksi tässä opinnäytetyössä pyritään vastaamaan seuraaviin tutkimuskysymyksiin:

- Millainen on D3:n hyödyntämiseen tarvittava työprosessi?
    - Miten D3 toimii?
    - Millainen työprosessi sopii parhaiten D3:n hyödyntämiseen?
- Millaista osaamista tai muita resursseja D3:n hyödyntäminen edellyttää?
    - Tarvittavien tekniikoiden, ohjelmistojen ja välineiden hinta?
    - Tarvittavat osaamisresurssit?
    - Visualisointien toteuttamiseen kuluva aika?
- Soveltuuko D3 pienen toimituksen käyttöön?
    - Miten hyvin D3 sopii Vihreän Langan työprosesseihin ja tarpeisiin?
    - Miten D3:lla toteutettujen visualisointien julkaiseminen onnistuu Vihreän Langan verkkopalvelussa?
    - Tarvitaanko visualisointien toteuttamisessa ulkopuolista apua vai pystyykö ”koodaava” toimittaja toteuttamaan ne itsenäisesti?

Opinnäytetyön ulkopuolelle on rajattu visualisointien esteettisyys sekä sisällöllisen merkittävyyden tarkempi journalistinen pohdinta. Koska pääpaino on datan jalostamisessa ohjelmallisesti visuaaliseksi, tiedonhankintaa ja datan jalostamista käsitellään varsin pintapuolisesti.

Empiirisen osan tavoitteena on hahmotella D3.js-kirjaston hyödyntämiseen journalistisessa työssä käytettävä prosessi kolmen esimerkin avulla. Esimerkit on pyritty sovittamaan Vihreä Lanka -lehden tarpeisiin aihevalintojen ja esitystavan osalta. Case-esimerkeistä kaksi on julkaistu syksyn 2016 aikana Vihreän Langan sivuilla. Kolmas case-esimerkki julkaistaan huhtikuussa 2017. Visualisointien journalistista arvoa on arvioitu yhdessä toimituksen kanssa opinnäyteprosessin edetessä.

Henkilökohtaisena tavoitteena on perehtyä D3.js-kirjastoon ja sen käyttömahdollisuuksiin syvällisemmin, mikä auttaa hahmottamaan sen käyttökelpoisuutta työelämässä.

1.2 Toimeksiantaja ja työelämäyhteys

Vihreä Lanka on Vihreä Liitto rp:n enemmistöomistama ja rahoittama, kahdeksan kertaa vuodessa ilmestyvä aikakauslehti, jonka toinen julkaisukanava on verkkosivusto (Kuva 2). Vihreän Langan verkkopalvelulla on kuukausittain 50 000-100 000 kävijää. (Vihreä Lanka 2016b.)

Kuva 2. Vihreä Lanka -lehden verkkosivusto. (Vihreä Lanka 2016c.)

Vihreä Lanka -lehden palveluksessa työskentelee toimitusjohtaja, graafikko, päätoimittaja, toimituspäällikkö sekä kolme toimittajaa (Vihreä Lanka 2016d). Vihreä Lanka on jo vuosien ajan julkaissut paperilehtensä sivuilla datavisualisointeja Graafinen maailmanselitys -nimellä (Vihreä Lanka 2016a).

Case-esimerkeissä tuotettujen visualisointien aiheet ja toteutus on sovittu Vihreä Lanka -lehden toimituksessa. Niiden tarpeellisuutta on arvioitu samalla tavalla kuin muun julkaistavan journalistisen sisällön tarpeellisuutta. Visualisoinneille on suunniteltu myös relevantti julkaisuaikataulu.

1.3 Menetelmä

Kyseessä on toiminnallinen eli produktityyppinen opinnäytetyö (Haaga-Helia 2016), jonka tavoitteena on tuottaa käyttökelpoisia datavisualisointeja Vihreä Lanka -lehden käyttöön. Opinnäytetyö jakautuu neljään osaan, joista ensimmäinen on johdantokappale.

Toinen osa, eli luvut kaksi ja kolme, esittelevät aiheeseen liittyvää teoriaa, ensin datajournalismin ja datavisualisointien ja sitten D3.js-kirjaston osalta. Teoriaosuus perustuu enimmäkseen verkkolähteisiin ja D3:n dokumentaatioon. Teorialukujen tavoitteena on vastata ensimmäiseen pääkysymykseen: millainen on D3:n hyödyntämiseen tarvittava työprosessi? Muihin tutkimuskysymyksiin vastataan case-esimerkkien pohjalta luvussa seitsemän.

Empiirinen osa kattaa luvut 4, 5 ja 6. Niissä käydään läpi kolmen erilaisen datavisualisoinnin toteutukset D3.js-kirjastoa käyttäen. Esimerkit käydään läpi seikkaperäisesti havainnollistavia koodiesimerkkejä ja kuvia hyödyntäen. Kukin esimerkissä syntynyt visualisointi on julkaistu tai tullaan julkaisemaan Vihreä Lanka -lehden verkkosivuilla.

Viimeinen osa, eli luku 7, sisältää johtopäätökset sekä aiheeseen liittyvää pohdintaa. Luvussa 7 vastataan toiseen ja kolmanteen pääkysymykseen: millaista osaamista tai muita resursseja D3:n hyödyntäminen edellyttää ja soveltuuko D3 pienen toimituksen käyttöön? Peilaan luvussa 7 omia kokemuksiani tämä opinnäytetyön case-esimerkeistä sekä teoriapohjasta kokemuksiini Vihreän Langan toimittajana (elokuusta 2007 lähtien) ja uutispäällikkönä (touko-kesäkuu 2015).

1.4 Käytettävät käsitteet ja lyhenteet

CSS	HTML:ssä käytettävä tyylitiedosto
CSV	tietotaulukko, jossa rivin arvot on eroteltu pilkuin
data	tietoaineistoa
datan jalostaminen	tietoaineiston saattaminen koneellisesti käsiteltävään muotoon
datan puhdistaminen	tietojen saattaminen johdonmukaiseen ja määriteltyyn (esim. CSV) muotoon
datajournalismi	journalistinen tiedonvälitys, jossa hyödynnetään tietotekniikkaa
datavisualisointi	ohjelmallisesti tuotettu visualisointi (usein taulukkomuotoisesta) datasta
HTML	verkkosivuilla käytettävä hypertekstikieli
interaktiivinen visualisointi	visualisointi, jonka ulkoasu muuttuu käyttäjän valintojen myötä
JavaScript	käyttäjän omassa selaimessa pyörivä ohjelmointikieli
journalismi	ammattilaisten tuottamaa tietoperusteista joukkoviestintää

JSON	Java Script Object Notation, tapa tallentaa hierarkisesti järjestettyä tietoa
kehittäjä	kts. ohjelmoija
käyttäjä	verkkopalvelun loppukäyttäjä eli lukija tai asiakas
ohjelmoija	henkilö tai ryhmä, joka tekee visualisoinnin
skaalautuva	Elementin koko muuttuu käyttäjän selainikkunan koon muuttuessa
staattinen	elementti, joka ei reagoi käyttäjän toimintaan
tietopyyntö	viranomaiselle jätettävä pyyntö tietyn tiedon saamiseksi
SVG	skaalautuva vektorigrafiikka
TSV	tietotaulukko, jossa rivin arvot on eroteltu sarkaimella

Yleisimmät kaaviotyypit, joihin tekstissä viitataan, on esitelty liitteessä 4:

bar chart	pylväsdiagrammi
line chart	viivadiagrammi
pie chart	piirakkadiagrammi
scatter plot	parvidiagrammi

2 Datajournalismi ja datavisualisoinnit

Datajournalismilla ja datavisualisoinneilla on pitkä historia, mikä johtuu osittain siitä, että niiden määritelmät ovat varsin löyhiä. Esimerkiksi perinteisiä säätiedotuksia voi pitää datajournalismina (Jugon 2011) tai taloussivujen osakekurssikuvaajia datavisualisointeina (Koning 2012).

2.1 Datajournalismi

Datajournalismi tarkoittaa journalismia, jossa käytetään hyväksi dataa. Data voi olla toimittajalle joko lähde tai työkalu. Tämä tarkoittaa, että datajournalismissa voi olla kyse esimerkiksi automatisoidusta tiedonhankinnasta, olemassa olevan datan käsittelystä ymmärrettävään muotoon tai monimutkaisten kokonaisuuksien esittämisestä visuaalisesti. (Bradshaw 2012.)

Suomessa asiaa tutkineet Kuutti ja Uskali (2016, 60–62) jakavat datajournalismin kolmeen alalajiin: yleiseen datajournalismiin, tutkivaan datajournalismiin ja reaaliaikaiseen datajournalismiin. Yleisessä datajournalismissa ei välttämättä tarvita ohjelmointitaitoja vaan data-aineistoa käsitellään esimerkiksi taulukkolaskentaohjelmassa. Yleisessä datajournalismissa data-aineisto on saatu valmiina julkisista lähteistä, kun taas tutkivassa datajournalismissa aineisto on saatu vuodettuna tai julkisuuslakiin pohjautuvalla tietopyynnöllä. Tutkivassa datajournalismissa data-aineiston käsittelyyn tarvitaan ohjelmointitaitoja. Reaaliaikainen datajournalismi tarkoittaa niin sanottua robottijournalismia, jossa jonkinlainen algoritmi tuottaa automaattisesti esimerkiksi pörssi- tai rikosuutisia.

Kuutti & Uskali (2016, 22) ajoittavat modernin datajournalismin synnyn vuosiin 2005–2006, jolloin saatiin ensimmäiset esimerkit siitä, että suuria datamääriä voidaan tehokkaasti ja ymmärrettävästi uutisoida, tarinallistaa ja visualisoida. Varsinaisena läpimurtona he pitävät vuotta 2010, kun Wikileaks julkaisi suuria datamääriä ja niiden käsittelemiseen tarvittiin toimituksissa uusia keinoja.

Datajournalismi rantautui Suomeen kunnolla vuonna 2011. Kuutti ja Uskali mainitsevat edelläkävijöinä Helsingin Sanomien Tuomo Pietiläisen ja Esa Mäkisen sekä freelancer Jens Finnäsin. Myös myöhemmin Yleisradioon päätynyt Teemo Tebest kiinnostui datajournalismista vuonna 2011. (Kuutti & Uskali 2016, 71–77.)

Julkista avointa dataa on julkaistu viime vuosina runsaasti, mutta sitä ei vielä juurikaan hyödynnetä toimituksissa. Kuutin ja Uskalin haastattelema Wall Street Journalin toimittaja Rob Barry arvioi, ettei avoin data kiinnosta toimittajia, koska se on kaikkien saatavilla,

minkä vuoksi siitä puuttuu ”never before seen -elementti”, joka yleensä tarvitaan uutisjuttuihin. Barryn mielestä arvokasta dataa saadaan esimerkiksi tietopyynnöillä. (Kuutti & Uskali 2016, 107.)

Guardianin datajournalisti Simon Rogers (2011b) kuvaa edustamansa lehden prosessia viisivaiheiseksi. Hänen mallissaan lähdetään raakadatan hankinnasta tai vastaanottamisesta. Tämän jälkeen pureudutaan dataan arvioimalla muun muassa, voiko sitä esittää aikasarjana tai pitäisikö sitä täydentää jollain muulla aineistolla. Kolmannessa vaiheessa data puhdistetaan usein taulukkomuodossa. Vasta neljännessä vaiheessa päästään tekemään laskelmia ja pureutumaan dataan ja vasta tässä vaiheessa selviää, onko kyseessä uutinen vai ei. Viides ja Rogersin mallin viimeinen vaihe on esitystavan valinta: tehdäänkö visualisointi vai perinteinen tekstimuotoinen juttu.

Kuutin & Uskalin (2016, 96) kuvaama datajournalistinen työprosessi on hieman erilainen. Periaatteessa se keskittyy lähinnä Rogersin mallin neljänteen vaiheeseen. Heidän mukaansa kyse on eräänlaisesta karusellista, jossa datalle esitetään journalistisia kysymyksiä ja saatuja vastauksia arvioidaan journalistisesti.

Jos tässä prosessissa päädytään journalistisesti kiinnostavaan tietoon, edetään datan puhdistamiseen ja analysointiin sekä vähitellen kohti julkaisemista. Kuutin ja Uskalin (2016, 165) mukaan datan puhdistuksen yhteydessä on tärkeää arvioida, onko data riittävää vai puutteellista. Heidän mukaansa ”datan likaisuudessa” on omat aste-eronsa, eli vaikka datassa olisi puutteita, sitä voidaan käyttää, kunhan puutteet huomioidaan. Heidän mukaansa data-aineiston puutteellisuudesta on kerrottava yleisölle.

Kuutin ja Uskalin mukaan (2016, 97) datajournalistiseen prosessiin sisältyy vuorovaikutus yleisön kanssa eli palaute ja kommentit esimerkiksi sosiaalisessa mediassa. Samaa korostaa teknologialehti Wiredin datajournalisti Duncan Geere (2012), jonka mukaan yleisö on tärkeä voimavara, sillä he voivat paitsi antaa palautetta myös löytää aivan uusia näkökulmia. Hänen mukaansa datajournalististen juttujen yhteydessä olisi aina julkaistava raakadata.

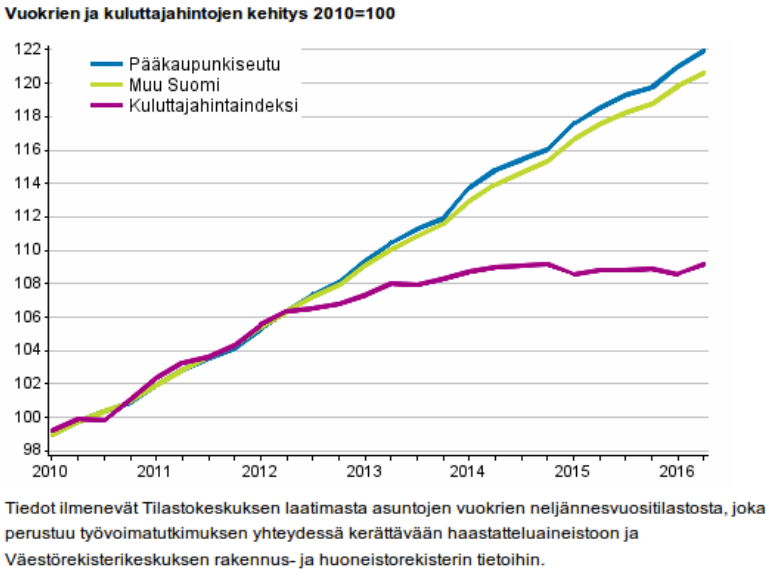
Suurissa toimituksissa datajournalismiin ja datavisualisointeihin käytetään usein ammattiohjelmoijia. Esimerkiksi Yleisradion datajournalismiin erikoistuneessa Plus-deskissä ei ole yhtäkään varsinaista toimittajaa vaan tuottajia, koodareita ja graafikkoja, jotka yhdistävät osaamisensa Ylen toimittajien kanssa tarpeen mukaan. (Kuutti H. & Uskali T. 2016, 79.)

Maailmalla datajournalismia tehdään suurissa yksiköissä. Esimerkiksi Washington Postilla oli vuonna 2014 pelkästään ohjelmoijia 12–13, joiden lisäksi datajournalismia teki vaihteleva määrä toimittajia. Washington Postin toimittajilla ei ole juurikaan ohjelmointiosaamista, kun taas USA Todayn yksikkö edustaa toisenlaista suuntausta: heidän yksikkönsä koostuu toimittajista, jotka ovat myös taitavia ohjelmoijia. Vaikka kysely on kohdistettu varsin rajalliselle joukolle toimituksia, kertoo se siitä, että maailmalla monissa toimituksissa datajournalismin tekemiseen varataan merkittäviä resursseja. (Crucianelli S. & Zanchelli M. 2014.)

Toisaalta datajournalismia voi moderneilla työkaluilla tehdä kuka tahansa ilman koodaustaitoja. Kyse on siitä, että dataa lähestytään journalistisesti miettien, mikä on kiinnostavaa ja merkittävää. Ennen kaikkea kyse on tarinoiden kertomisesta, kuten kaikessa journalismissa. (Rogers 2011a.)

2.2 Datavisualisoinnit

Datavisualisoinnit ovat yksinkertaisesti numeerisen datan esittämistä visuaalisessa muodossa, eli esimerkiksi pylväs-, piiras- tai viivadiagrammeina (Kuva 3). Niiden merkitys on korostunut 2000-luvulla, koska tietokoneiden ja sovellusten yleistyminen ja kehittyminen on mahdollistanut aiempaa kattavamman tiedonkeruun ja analysoinnin. Toinen merkittävä muutos on ollut internet, jossa on mahdollista julkaista monipuolisempia visualisointeja kuin paperilla. Kuutti & Uskali (2016, 48) ajoittavat tällaisen ”digitaalisen infografiikan” nousun vuosiin 2007–2012, jonka jälkeen tahti on hieman hiipunut.

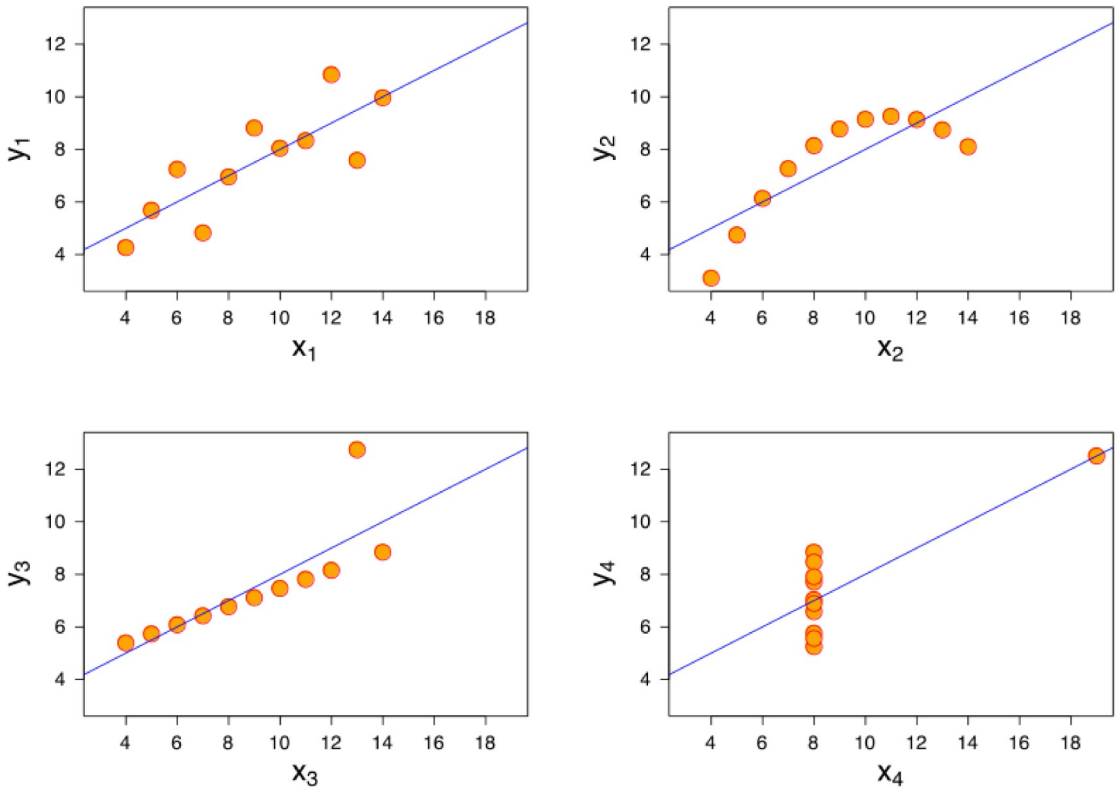


Kuva 3. Tilastokeskuksen julkaisemat grafiikat, kuten tämä vuokratason nousua esittävä kuvaaja, ovat datavisualisointeja perinteisimmillään. (Tilastokeskus 2016.)

Kosara (2007) määrittelee kolme kriteeriä, jotka tuotoksen on täytettävä ollakseen datavisualisointi: Ensinnäkin sen on perustuttava näkymättömään dataan eli johonkin, mikä ei

ole suoraan nähtävissä. Toisekseen visualisoinnin on oltava kuvallinen ja kuvan on oltava kerronnan päämuoto, eli tekstillä voidaan vain tuoda lisätietoa. Kolmanneksi lopputuloksen on oltava luettavissa: ”Jokainen merkityksellisen datan muuntaminen hukkaa osan informaatiosta, mutta ainakin jokin olennainen datan ominaisuus on oltava kuvassa näkyvillä” (Kosara 2007).

Datan visualisoimisen hyödyllisyyttä perustellaan tyypillisesti (Grönroos 2015, 20; Nieminen 2013, 3) niin sanotulla Anscomben nelikolla (Kuva 4). Tilastotieteilijä Francis Anscombe osoitti vuonna 1973, kuinka neljä erilaista tietoaineistoa voivat tuottaa täysin erilaiset visualisoinnit, vaikka niiden matemaattinen vertailu (keskiarvo, korrelaatio ja varianssi) antaa lähes identtiset tulokset. (Kosara 2011.)



Kuva 4. Anscomben nelikko. (Kosara 2011.)

Tyypillisimmät tavat esittää dataa ovat perinteiset piiras-, pylväs- ja viivadiagrammit sekä parvidiagrammi, josta esimerkkinä edellä esitetty Anscomben nelikko. Nämä sekä muutamia muita tyypillisiä datavisualisointeja löytyvät Andrew Abelan (2006) kehittämästä kaaviosta, jonka avulla voi valita kuhunkin tarpeeseen parhaiten sopivan kaavion (liite 4).

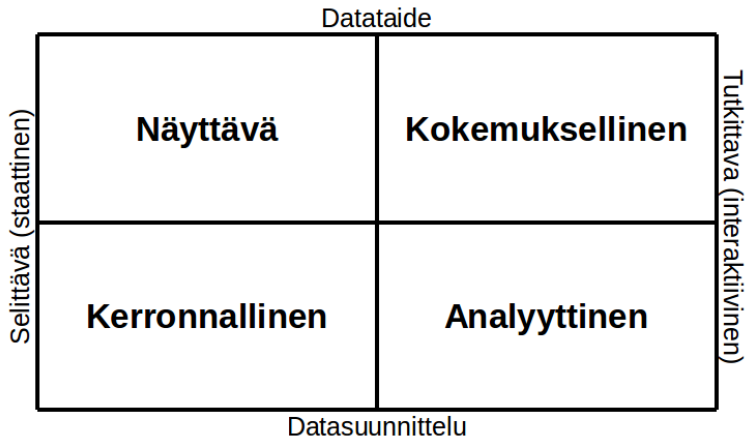
Datavisualisoinneissa on lähtökohtaisesti kyse todellisen ja luotettavan tiedon esittämisestä kuvallisessa muodossa, mutta erilaisilla valinnoilla voidaan vaikuttaa siihen, antaako visualisointi totuudenmukaisen kuvan käsiteltävästä tietoaineistosta. Nämä vaihtoehdot on tärkeää tuntea, jotta niistä voidaan valita tarkoituksenmukaisin. Käytännössä valintaan vaikuttavat erilaiset rajoitukset kuten käytettävissä olet taidolliset, teknologiset, ajalliset ja taloudelliset resurssit sekä esimerkiksi toteuttajan ideointikyvyn rajallisuus. Yleisön rakenne ja konteksti, jossa visualisointi esitetään, on niin ikään huomioitava. Se, että visualisointi on kiinnostava edellyttää sitä, että esitettävä tieto on yleisölle uutta ja että se esitetään ymmärrettävällä tavalla. (Tulep 22.4.2016.)

Taulukko 1. Tulepin nelikenttä auttaa hahmottamaan, mikä tekee visualisoinnista kiinnostavan eli onnistuneen.

Taulukon 1 mukaan huonoin mahdollinen, suorastaan epäonnistunut kuvio on sellainen, jossa esitetään ennestään tuttua tietoa vaikeaselkoisessa muodossa. Visuaalisesti hieno tai helposti ymmärrettävä esitystapa ei tee visualisoinnista kiinnostavaa, jos esitettävä asia on ennestään tuttua. Onnistuneen visualisoinnin on tarjottava uutta tietoa ymmärrettävässä muodossa.

Visualisointivaihtoehtoja voidaan jaotella myös käsitteellisesti. Tulep käyttää verkkosivuilla esitettävien visualisointien lajitteluun nelikenttää (Taulukko 2), jossa visualisoinnit jaetaan kahdella akselilla neljään eri ryhmään: näyttäviin, koettaviin, kerronnallisiin ja analyyttisiin. Näyttävien tavoitteena on tehdä kuvaaja, joka on ensisijaisesti näyttävä eli esimerkiksi data-aineiston ymmärrettävyyden edistäminen on toissijaista. Kyse on datataiteesta, kuten kokemuksellisten visualisointien kohdalla. Nämä ovat interaktiivisia esityksiä, joiden kulkuun käyttäjä voi vaikuttaa. Datan ymmärtämisen edistäminen on jälleen toissijaista. (Tulep 22.4.2016.)

Tulepin (22.4.2016) mukaan datasuunnittelussa sen sijaan pyritään esitystapoihin, jotka auttavat oikeasti ymmärtämään datan sisältöä. Tällaisia ovat kerronnalliset datavisualisoinnit kuten kuvat 3 (Tilastokeskuksen viivadiagrammi) ja kuva 4 (Anscomben nelikön parvidiagrammit) edellä: ne eivät ole kauniita tai interaktiivisia, mutta antavat selkeän kuvan siitä, millaisesta datasta on kyse. Datasuunnitteluun kuuluvat myös analyyttiset työkalut, joissa käyttäjä voi omilla valinnoilla vaikuttaa esitystapaan. Esimerkki tällaisesta on myöhemmin kuvassa 6, jossa esitetyssä New York Timesin visualisoinnissa käyttäjä voi nostaa hiiren klikkauksella esiin tarkempia tietoja.



Taulukko 2. Tulepin nelikenttä datavisualisointien jaotteluun.

Datasuunnittelun kahden lajityypin, kerronnallisten ja analyyttisten visualisointien eroja, on pohtinut myös Alberto Cairo (2014). Hän puhuu infografiikoista ja datavisualisoinneista käyttäen jälkimmäistä termiä huomattavasti suppeammassa mielessä kuin esimerkiksi tässä opinnäytetyössä. Cairon mukaan infografiikat kertovat tarinan, jonka suunnittelija haluaa kertoa, kun taas datavisualisoinnit antavat ihmisille mahdollisuuden luoda omat näkemyksensä datan perusteella.

Visualisointitekniikoita voidaan arvottaa sillä perusteella, kuin helposti niiden ilmentävät erot ovat omaksuttavissa. Tulepin (22.4.2016) mukaan esittämiseen käytettävät tekniikat voivat laittaa järjestykseen sen perusteella, kuinka helppo niitä on hahmottaa. Järjestystä tai numeerisia arvoja esittävien tekniikoiden osalta järjestys helpoimmasta vaikeimpaan on:

1. Sijainti samalla asteikolla.
2. Sijainti eri asteikoilla.
3. Yksiulotteisen kuvion pituus.
4. Asento/kulma.
5. Kaksiulotteisen kuvion pinta-ala.
6. Syvyys eli sijainti kolmiulotteisessa koordinaatistossa.

7. Värisävyn voimakkuus tai tummuus.
8. Kaarevuus.
9. Kolmiulotteisen kappaleen tilavuus.

Parhaita visualisointeja ovat janat ja koordinaatistot, joissa kaikki arvot esitetään samalla asteikolla. Heti tämän jälkeen seuraavat esimerkiksi erilliset mutta rinnakkaiset koordinaatistot. Esimerkki yksiulotteisen kuvion pituudesta voisi olla pylväsdiagrammi. Perinteinen ja yleinen piirakkadiagrammi on yhdistelmä kohtia neljä ja viisi, eli kulman ja pinta-alan hahmottamista. Tulepin (22.4.2016) mukaan piirakkaa ei tästä syystä pitäisi käyttää kahta suurempien arvojoukkojen esittämiseen. Värisävyt, kaarevuudet sekä kaikki kolmiulotteinen hahmottaminen on Tulepin (22.4.2016) mukaan liian vaikeaa käytettäväksi.

Luokiteltavien arvojen, eli sellaisten joita ei voida yksiselitteisesti laittaa järjestykseen, osalta vastaava järjestys on hieman erilainen:

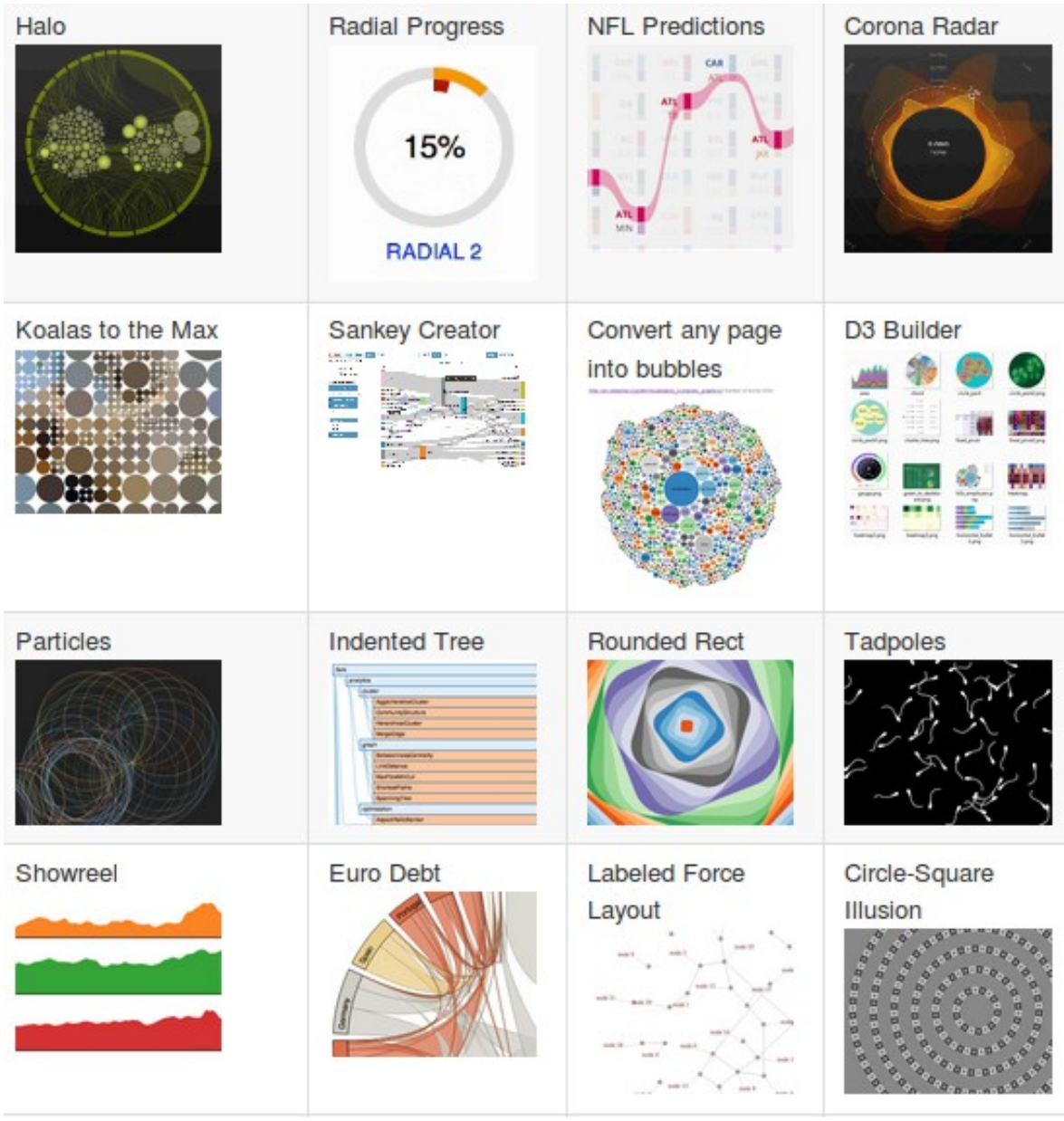
1. Sijoittuminen.
2. Väri.
3. Liike.
4. Muoto.

Jo listalla toiseksi parhaaksi sijoitetusta väristä tulee Tulepin (22.4.2016) mukaan vaikeasti erotettava, jos vertailtavia kohteita on yli seitsemän. Eli jos halutaan esimerkiksi erotella automerkkejä toisistaan on selvintä erotella ne sijoittamalla ne eri ryhmiin sen sijaan, että käytettäisiin väriä tai muotoa (kuten logoa).

3 D3.js-kirjasto

D3.js on JavaScript-kirjasto, joka auttaa visualisoimaan dataa verkkojulkaisuissa HTML:n, SVG:n ja CSS:n avulla. Koska D3 perustuu yleisesti käytettyihin standardeihin, se toimii käytännössä kaikilla moderneilla selainohjelmilla (muun muassa Chrome, Firefox, Safari sekä versiota kahdeksan uudemmat Internet Explorer -selaimet). Tarvittaessa D3:lla voidaan toteuttaa visualisointeja vanhemmille selaimille, mutta tämä vaatii erityisiä toimenpiteitä. (D3 Wiki 2016a.)

D3:n moninaisia käyttötapoja on esitelty kuvassa 5.



Kuva 5. Esimerkkejä D3:n käyttötavoista. (Github 2016.)

D3:n hyötyihin kuuluu vaihtoehtoisia visualisointitekniikoita kuten Flashia lyhyempi latausaika (Bostock, Heer & Ogievetsky 2011). Toisaalta merkittävin heikkous on Skaun (2013) mukaan hitaus isoja datamääriä käsiteltäessä. Hän tosin toteaa, että oikein toteutetussa visualisoinnissa suuria datamassoja ei yleensä tarvitse käsitellä enää piirtovaiheessa.

3.1 D3:n juuret

D3:n ja sen edeltäjä Protovisin kehittämisen taustalla oli ajatus aiempia visualisointityökaluja paremmin ja saumattomammin verkon standardeja hyödyntävästä ratkaisusta. Aiemmat visualisointityökalut, kuten Prefuse ja Flare, vaativat käyttäjältä Javan tai Flashin kaltaisten ohjelmistojen asentamista. Käytännössä tällaisista riippuvuuksista irtautuminen tarkoitti sitä, että erillisen grafiikan piirtämisen sijaan visualisoinnit perustuvat JavaScriptin kykyyn muokata HTML-verkkosivun niin sanottuja dokumenttioliomalli- eli DOM-elementtejä. Datavisualisoinnissa vaaditaan kuitenkin kykyä luoda uusia ja poistaa tarpeettomia elementtejä, mikä on kehittäjien mukaan mahdotonta CSS:llä ja työlästä jQueryllä, jossa mahdollisuudet sitoa dataa DOM-elementteihin ovat rajalliset. (Bostock, Heer & Ogievetsky 2011.)

Cukier (2015) suosittelee, että D3:n käyttö kannattaisi aloittaa opettelemalla ensin kunnolla JavaScriptiä, HTML:n logiikkaa ja SVG-grafiikkaa ja vasta sen jälkeen perehtymällä itse D3.js-kirjastoon. Cukier itse aloitti D3:n käytön tuntematta kovin tarkasti tukevien teknologioiden mahdollisuuksia, jolloin hän päätyi käyttämään D3:a silloinkin, kun pelkkä JavaScript olisi riittänyt.

3.2 Toimintaperiaate

D3.js-kirjasto on kokoelma JavaScript-metodeista ja funktioista koostuvia moduuleja, joiden avulla voidaan luoda HTML-sivulle SVG-elementtejä ja muokata niiden ominaisuuksia. Kirjasto sisältää myös matematiikkaan, geometriaan, väreihin, animoituihin siirtymiin ja muihin efekteihin liittyvää toiminnallisuutta. Kirjaston avulla voi sitoa suuriakin tietoaineistoja SVG-elementteihin. (D3 Wiki 2016b.)

D3.js-kirjastolle voi syöttää ulkoisia tietoaineistoja seuraavissa tiedostomuodoissa: txt, json, html, xml, csv ja tsv. Ulkoisen datan osalta D3 tukee tietoaineistojen hakemista XML-muotoisella HTTP-pyynnöllä. (D3 Wiki 2016b.)

Koska D3 on suunniteltu erityisesti tilastodatan visuaaliseen esittämiseen, siinä on mukana monia tähän käyttöön hyödyllisiä toimintoja. Esimerkiksi scale-funktioilla voidaan sitoa data-aineiston skaala kuten esimerkiksi maiden väkiluvut nollasta miljardiin graafisen esi-

tyksen skaalaan eli esimerkiksi pylväsgrafiikan korkeuteen. Tällaiselle funktion avulla esimerkiksi pylväsdiagrammin pylväiden korkeus saadaan skaalautumaan automaattisesti aina tietoaineiston muuttuessa. (Cook 2016.)

Teknisesti ottaen D3 on sukua jQuerylle: molemmat ovat JavaScript-kirjastoja, joiden avulla voidaan valita ja muokata verkkosivujen dokumenttioliomallin niin sanottuja DOM-elementtejä. Kahta kirjastoa on mahdollista käyttää yhdessä tai erikseen (Möller 2015).

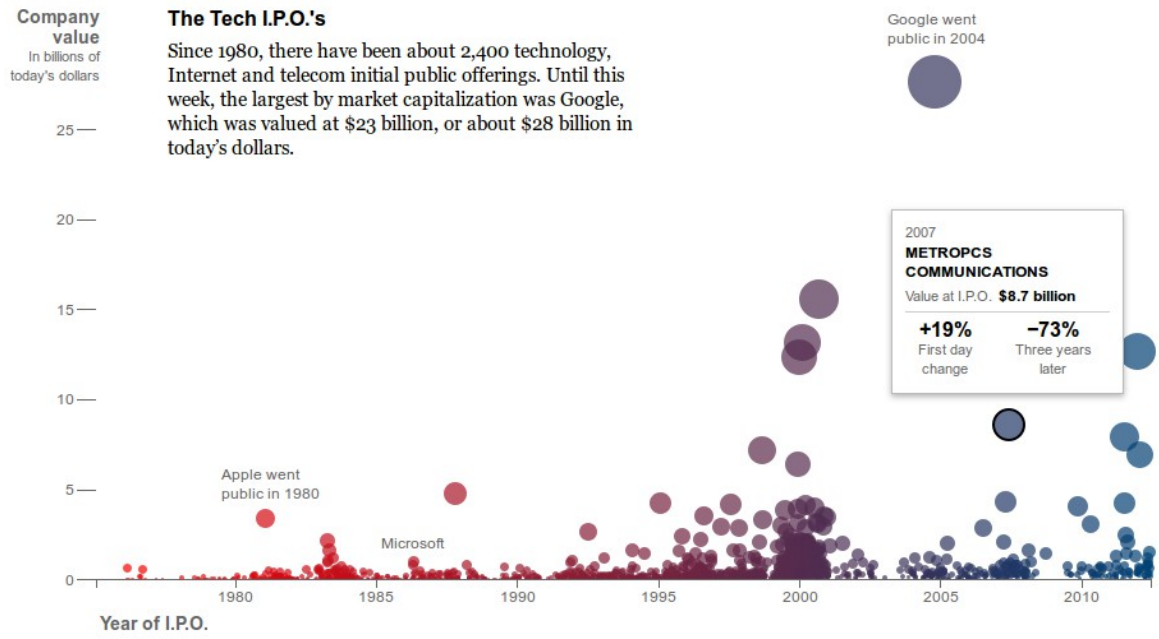
D3 toimii siten, että HTML-dokumenttiin sisällytetyllä JavaScript-koodilla kutsutaan D3.js-kirjaston funktioita ja metodeja muokkaamaan DOM-elementtejä. Käytännössä tämä tarkoittaa sitä, että D3 luo, muokkaa ja poistaa elementtejä niihin sidotun datan mukaisesti (Bostock, Heer & Ogievetsky 2011.). Ensimmäisessä case-esimerkissä (luku 4) käydään seikkaperäisesti läpi D3:n toimintaa ja perusominaisuuksia.

Kesällä 2016 julkaistun päivityksen myötä D3 siirtyi versiosta kolme versioon neljä. Päivityksen suurin uudistus oli D3:n rakenteen muuttaminen modulaariseksi. Käyttäjän ei enää tarvitse välttämättä ladata koko kirjastoa vaan hän voi vaihtoehtoisesti valita, mitkä kirjastot tai moduulit hän tarvitsee käyttöönsä. Pienemmät kirjastot nopeuttavat D3:n toimintaa ja tekevät siitä joustavampaa. Vanhojen visualisointien päivittäminen versiosta kolme versioon neljä on melko työlästä, sillä suuri osa funktioiden ja metodien nimistä on muuttunut. (Bostock 2016.)

3.3 Käyttöesimerkkejä

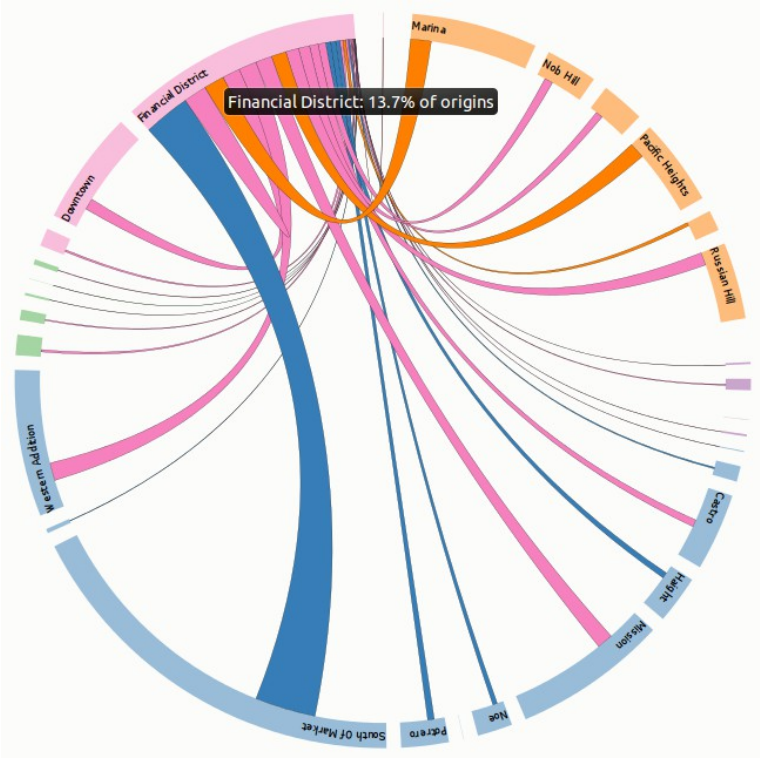
Lukuisat suuret mediatalot käyttävät D3.js-kirjastoa journalistisiin datavisualisointeihin, jotka vaihtelevat pienistä grafiikoista suuriin tarinallisiin kokonaisuuksiin. Näitä on esitelty kuvissa 6–8.

Kuvassa 6 on New York Timesin D3:lla toteutettu visualisointi IT-yritysten listautumisista pörssiin. Parvidiagrammissa aika on esitetty X-akselilla ja yrityksen arvo Y-akselilla. Kun yksittäisen pallon valitsee hiiren osoittimelle, saa lisätietoa kyseisestä yrityksestä. (Ashkenas, Bloch, Carter & Cox 2012.)



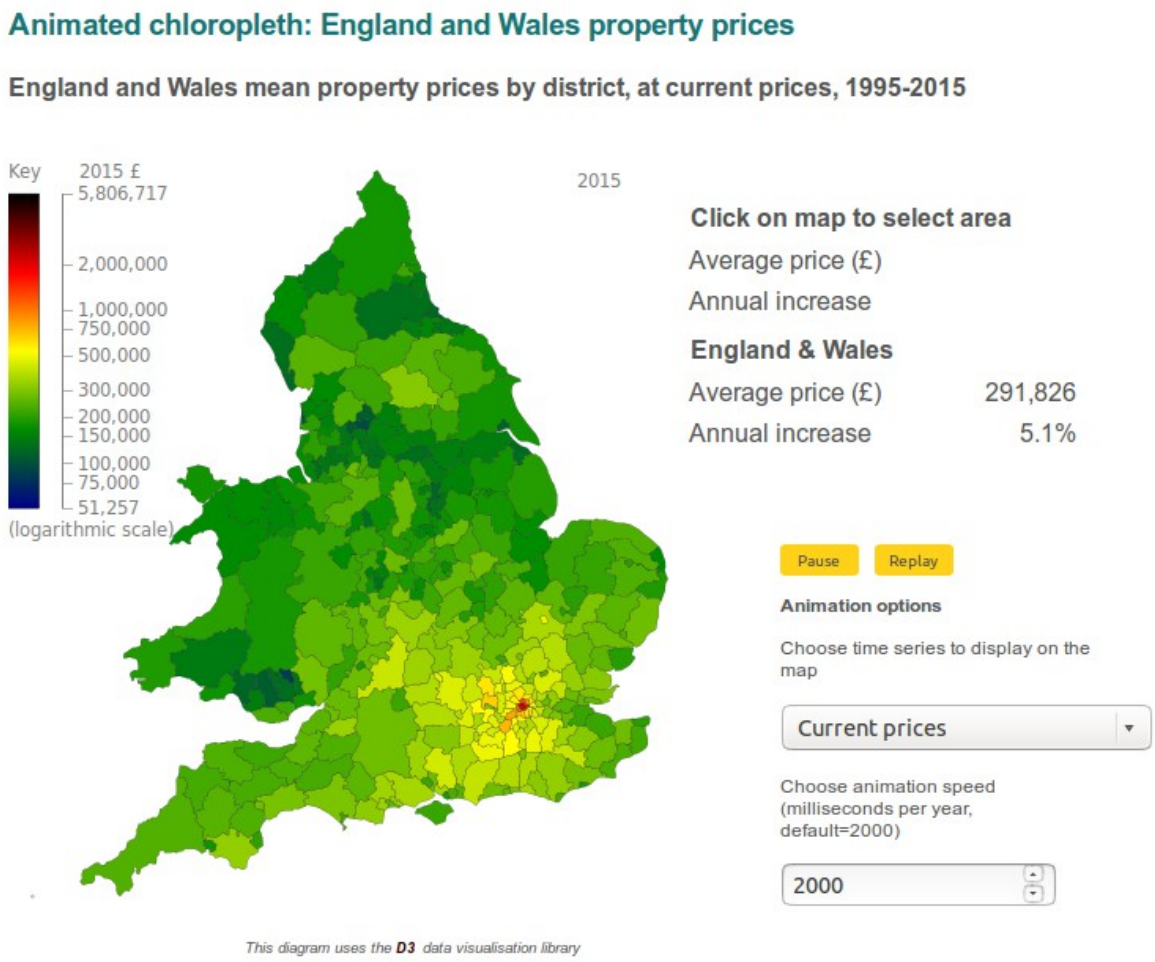
Kuva 6. New York Timesin visualisointi IT-yritysten listautumisista pörssiin.

Kuvassa 7 on D3:n kehittäjä Mike Bostockin tekemä visualisointi Über-kyydeistä San Franciscossa. Diagrammissa on mahdollista valita hiirellä mikä tahansa kehällä oleva kaupunginosa, jolloin visualisointi näyttää, mihin kyseisestä kaupunginosasta on tilattu kyytejä. Visualisointia on selvennetty värien käytöllä.(Bostock 2012.)



Kuva 7. Mike Bostockin visualisointi Über-kyydeistä San Franciscossa.

Kuvassa 8 on muun muassa Guardianilla työskennelleen Helen Jacksonin visualisointi asuntojen hintakehityksestä Englannissa ja Walesissa. Se havainnollistaa, miten D3 soveltuu karttavisualisointien toteuttamiseen. Kehitys esitetään animaationa, jonka nopeuden käyttäjä voi itse valita. (Jackson 2015.)



Kuva 8. Visualisointi asuntojen hintakehityksestä Englannissa ja Walesissa.

3.4 SVG

SVG on CSS:n tapaan olennainen osa HTML5-standardia (W3 Schools 2016b). SVG on lyhenne sanoista ”scalable vector graphics” eli skaalautuva vektorigrafiikka ja sillä tarkoitetaan grafiikan esittämistä XML-muodossa (Mozilla 2016a).

Mozilla-yhtiön (2016b) ohjelmoijille tarkoittamilla sivuilla on esimerkkejä, jotka havainnollistavat esimerkiksi viivan piirtämistä SVG:llä. Aluksi määritellään alue, jolle SVG-grafiikkaa piirretään. Tässä on syytä huomata, että SVG:ssä (ja siten myös D3:ssa) käytettävän koordinaatiston nollapiste on vasemmassa yläkulmassa, eli – toisin kuin perinteisessä matemaattisessa koordinaatistossa – Y-akselin arvon kasvaminen tarkoittaa, että koordinaatistossa siirrytään ylhäältä alas eikä päinvastoin.

```
<svg width="120" height="120"
     viewPort="0 0 120 120" version="1.1"
     xmlns="http://www.w3.org/2000/svg">
```

Varsinainen viiva piirretään koodilla, jossa määritellään pisteet, joiden kautta sen on kuljettava. Ensimmäisen piste on 20 pikseliä piirtoalueen vasemmasta laidasta oikealle ja 100 pikseliä yläreunasta alaspäin. Toinen pikseli on 100 pikseliä vasemmasta reunasta oikealle ja 20 pikseliä yläreunasta alas. Lopputuloksena on lyhyt viiva alhaalta vasemmalta oikealle ylös.

```
    <line x1="20" y1="100"
          x2="100" y2="20"
          stroke="black"
          stroke-width="2"/>

</svg>
```

D3:n etu on siinä, että se tekee tällaista SVG-koodia automaattisesti datan pohjalta. Kirjasto käsittelee datan ohjelmoijan antamien ohjeiden mukaan ja tuottaa siitä SVG-grafiikkaa selaimen näytettäväksi ilman, että ohjelmoijan tarvitsee välttämättä osata tai edes nähdä SVG-koodia missään vaiheessa.

3.5 Vertailu vaihtoehtoisiin työkaluihin

Lähimmin D3.js-kirjastoon vertautuvat muut visualisointiin käytetyt JavaScript-kirjastot kuten Processing.js, Paper.js, Graph.js ja Raphael.js. Pienempiin JavaScript-kirjastoihin verrattuna D3:n vahvuus on Skaun (2013) mukaan sen joustavuus ja laajuus. Tämä juontaa juurensa siihen, että D3:n avulla voi muokata mitä tahansa dokumenttioliomallin osasta eikä ainoastaan SVG- tai kanvas-elementtejä. Toiset kirjastot saattavat olla helppokäyttöisempiä – esimerkiksi Processing.js:n käyttö ei vaadi edes JavaScript-osaamista – mutta ne eivät taivu yhtä hyvin monimutkaisiin visualisointeihin (Dormehl 2014).

Kaupallisista ratkaisuista ehkä tunnetuin ja suosituin on Tableau. Sen käyttö maksaa halvimmillaan 500 dollaria vuosi (Tableau 2016). Wilsonin (2015) mukaan Tableau on yhtä helppokäyttöinen kuin Excel-taulukkolaskenta ja sillä kuka tahansa voi tehdä helposti ja nopeasti oikeaoppisia visualisointeja.

Interaktiivisia tai animoituja datavisualisointeja on perinteisesti toteutettu Flashilla, joka oli ensimmäinen laajalle levinnyt ja suosittu tapa yhdistää verkkosivuille videoita ja interaktiivisia elementtejä. Flashin käyttö vaatii kuitenkin erillisen lisäosan asentamisen. Flash on kärsinyt myös tietoturvaongelmista. Kaikkien modernien selaimien noudattama HTML5-standardi, johon D3:kin nojaa, poistaa suurilta osin tarpeen käyttää Flashin kaltaisia lisäosia. Esimerkiksi videota, ääntä ja erilaisia interaktiivisia visualisointeja voidaan toteuttaa nykyisin ilmankin. (Nixon 2015.)

Verkossa on perinteisesti julkaistu vanhanaikaisilla kaupallisilla visualisointityökaluilla tehtyjä kuvaajia. Esimerkiksi Adobe Illustratoria ja kaikkein arkisimmillaan Microsoft Exceliä tai vastaavaa taulukkolaskentaohjelmaa käyttäen voi taulukkomuotoisesta datasta tuottaa staattisia visualisointeja. (Mitevski 2010.)

4 Case: Puolueiden kannatuksen muutos 2015–2016

Ensimmäinen case-esimerkki (Kuva 9) on hyvin yksinkertainen D3:n animointi- ja interaktiivisuusominaisuuksia hyödyntävä visualisointi: pylväsgraafi, jonka pylväiden pituus vaihtuu nappia painamalla. Pylväät kuvaavat puolueiden kannatusta käyttäjän valinnasta riippuen joko eduskuntavaaleissa 2015 tai Ylen gallupissa kesäkuussa 2016. Käytännössä tämä animaatio on niin yksinkertainen, että se olisi helpompi toteuttaa esimerkiksi huomattavasti yksinkertaisemmalla Graph.js-kirjastolla, mutta tässä tapauksessa yksinkertainen esimerkki on helppo tapa päästä sisälle D3:n logiikkaan.

Kuva 9. Valmis grafiikka Vihreän Langan sivuilla. (Vihreä Lanka 2016e.)

Grafiikka on julkaistu Vihreä Lanka -lehden verkkopalvelussa 8. kesäkuuta 2016 gallup-lukuja käsittelevän uutisen kuvituksena. Esimerkki on hyvin yksinkertainen ja sen tekee noin tunnissa, jos D3 ja sen vaatimat tekniikat (erityisesti JavaScript) ovat tekijälle tuttuja.

Tämä opinnäytetyö on tehty touko-lokakuussa 2016. Projektin aikana julkaistiin D3:n neljäs versio. Kolmesta case-esimerkistä tämä ensimmäinen on tehty kolmannella versiolla, mutta kahdessa muussa käytetään nelosversiota.

4.1 Tiedonhankinta, tiedonkäsittely ja esitystapa

Vaalien tulostiedot on hankittu oikeusministeriöstä (Oikeuministeriö 2015) ja gallup-tiedot Ylen aihetta koskevasta uutisesta (Yle 2016). Koska tietomäärä on tässä visualisoinnissa hyvin pieni, on tiedot poimittu ja muotoiltu käsin. Käytännössä kyseessä on kaksi arvosarjaa, joista piirrettävä valitaan käyttäjän valinnan mukaan. Tiedot on kerätty yhteen CSV-tiedostoon, jonka sisältö on alla:

```
puolue,vanha,uusi
Kokoomus,16.2,17.0
Keskusta,24.9,20.4
Vihreät,8.8,13.5
SDP,16.8,21.5
Perussuomalaiset,15.4,8.5
Vasemmisto,8.8,8.5
RKP,4.0,4.5
KD,3.3,3.6
```

Pilkuilla erotellun taulukon ensimmäisessä sarakkeessa on puolueen nimi, toisessa sarakkeessa kannatus vaaleissa ja kolmannessa sarakkeessa kannatus toukokuussa 2016. Suomalaisesta käytännöstä poiketen desimaalierottimena on käytetty JavaScript standardin mukaan pistettä (W3 Schools 2016a).

Puolueiden kannatusta tavataan esittää monella eri tavalla. Koska puolueiden kannatuksen summa on sata prosenttia, olisi luontevaa käyttää piirasdiagrammia. Piirasdiagrammin käyttöä tulisi kuitenkin välttää, kun vertailtavia arvoja on enemmän kuin kaksi (Tulep 22.4.2016). Viivadiagrammi puolestaan soveltuu tilanteeseen, jossa halutaan esittää pitkiä aikasarjoja esimerkiksi kuukausittain vaalien välillä (Abela 2006).

Tässä visualisoinnissa halutaan esittää puolueiden kannatukset kahdella ajanhetkellä – vaaleissa 2015 ja kesäkuussa 2016. Tähän tarkoitukseen soveltuu parhaiten pylväsgrafiikka, koska siinä verrataan keskenään erillisiä asioita eikä samaa asiaa eri ajan hetkillä (Abela 2006). Jos visualisointi toteutettaisiin staattisena paperilehdessä, pitäisi joko piirtää kaksi erillistä pylväikköä tai jokaiselle puolueelle kaksi erillistä pylvästä. D3 mahdollistaa kuitenkin interaktiivisen esitystavan.

4.2 Tekninen toteutus

Toteutus on mukaelma D3:n sivuilta löytyvästä esimerkistä, jossa niin sanotun donitsikaavion sisältö muuttuu animoidusti nappia painamalla (Buezas 2016). Esimerkistä poiketen tässä visualisoinnissa käytetään pylväitä ja nappeja on kaksi, yksi kummankin tietosarjan esittämiseen.

HTML:n osalta välttämätöntä koodia on perusrakenteen lisäksi ainoastaan napit:

```
<button class="vanha">Vaalit 2015</button>
<button class="uusi">Gallup 5/2016</button>
```

D3:n koodi sijoitetaan nappien jälkeen <script>-tägien sisälle. D3-osio alkaa halutun piirtoalueen muodostamisella:

```
var svg = d3.select("body").append("svg")
          .attr("viewBox", "0 0 500 400")
          .attr("preserveAspectRatio", "xMidYMid meet");
```

Käytännössä koodissa muodostetaan muuttuja ”svg” funktiolla, joka valitsee HTML:n body-elementin ja tekee sen alle skaalautuvan SVG-elementin, jonka leveys on 500 pikseliä ja korkeus 400 pikseliä. Käytännössä SVG-elementti skaalautuu niin suureksi kuin ympäröivä HTML-rakenne tai käyttäjän ikkunan koko sallii. Syötetyt arvot (500 ja 400 pikseliä) määrittävät kuvasuhteen sekä piirtoalueen koon, joka pitää huomioida koodissa siten, että piirrettävät elementit mahtuvat sen sisälle. Jos halutaan muodostaa kiinteä eli skaalautumaton svg-elementti annetaan vain arvot leveys- ja korkeus-attribuuteille:

```
          .attr("width", 500)
          .attr("height", 400)
```

Varsinaisen piirtämisen tekee vaihda-funktio, joka ottaa vastaan paitsi itse datan, myös tiedon siitä, kumman ajankohdan mukainen kannatus piirretään. Funktion alussa tehdään

omat muuttujat väritykselle, joka haetaan D3:n valmiista väripaletista, sekä pylväille, joihin sidotaan samalla funktiolle syötetty data:

```
function vaihda(data, aika) {

          var color = d3.scale.category10();
          var palkit = svg.selectAll("rect").data(data);
```

Seuraavaksi aloitetaan itse piirtäminen. Ensimmäisessä ns. enter-vaiheessa kerrotaan, millaisena palkki piirretään, kun se ilmestyy ensimmäisen kerran.

```
palkit.enter()
          .append("rect")
          .attr("x", 0)
          .attr("y", function (d, i){ return i * 50; })
          .attr("width", 0)
          .attr("height", 30)
          .style("fill", function (d, i){ return color(i); })
```

Yllä olevan koodin toisella rivillä kerrotaan, että halutaan piirtää suorakulmio. Kolmas rivi määrittää sijainnin X-akselilla nollaksi, eli suorakulmio alkaa vasemmasta reunasta. Neljäs rivi määrittää suorakulmion vasemman yläkulman etäisyyden ”piirtoalustan” eli svg-elementin ylälaidasta. Tämä määritetään funktiolla kunkin dataelementin indeksin perusteella, jotta yksittäiset pylväät tulevat eri korkeuksille. Seuraavat kaksi riviä määrittävät suorakulmion ulkoiset mitat: Leveys on nolla, jotta palkki ikään kuin kasvaa siinä vaiheessa, kun data ladataan. Korkeudeksi on valittu 30 pikseliä. Viimeisellä rivillä pyydetään täyttämään suorakaide värillä, joka haetaan kunkin dataelementin indeksilukua vastaavasta kohdasta color-muuttujan paletista.

Seuraavaksi kerrotaan, millaiseksi haluamme palkkien muodostuvan:

```
palkit
          .transition().duration(1000)
          .attr("x", 0)
          .attr("y", function (d, i){ return i * 50; })
          .attr("width", function(d){
                    var dAika = eval("d." + aika);
                    return dAika * 20;
                    })
```

```
            .attr("height", 30)
            .style("fill", function (d, i){ return color(i); });
```

Toinen rivi määrittää muodostumisajaksi 1000 millisekuntia eli yhden sekunnin. Kukin palkki kasvaa yhdessä sekunnissa tyhjästä täyteen mittaansa. Kolmas ja neljäs rivi määrittävät palkin sijainnin piirtoalustalla aivan kuten edellisessä vaiheessa, sillä palkkien ei haluta siirtyvän toiseen paikkaan.

Riveillä 5-8 muodostetaan palkin leveys: Ensin luodaan uusi muuttuja dAika, jossa aikamuuttujan alkuun lisätään dataa kuvastava kirjain d sekä piste. Eval-funktiolla muutetaan sulkujen sisällä olevan string muuttujan nimeksi. Lopulta palautetaan pylvään leveydeksi dAika kerrottuna kahdellakymmenellä. Muuttujan dAika sisällä on joko d.vanha tai d.uusi, eli kun D3 käy tietoaineistoa läpi, se käsittelee kultakin riviltä ainoastaan halutun sarakkeen arvon.

Seuraavaksi haetaan palkeissa esitettävät tekstit:

```
var tekstit = svg.selectAll("text").data(data);

tekstit.enter().append("text");

tekstit
            .attr("fill", "white")
            .attr("x", 2)
            .attr("y", function (d, i) { return i * 50 + 20; })
            .text(function (d) {
                        var dAika = eval("d." + aika);
                        var etiketti = d.puolue + " " + dAika;
                        var etiketti = etiketti.replace(/\./g, ',');
                        return etiketti;
            });
```

Tekstien osalta koodin rakenne noudattaa samaa logiikkaa: Ensin muodostetaan teksteille muuttuja, johon sidotaan käsiteltävä data. Sen jälkeen kerrotaan, millaisena tekstit tulisi esittää: väriksi on syötetty valkoinen, sijainti x-akselilla on kaksi pikseliä piirtoalueen vasemmasta laidasta ja sijainti y-akselilla määräytyy indeksin mukaan siten, että palkkien väli on 50 pikseliä ja jokaista tekstiä siirretään 20 pikseliä alaspäin, jotta se olisi pystysuunnassa keskellä kutakin palkkia. Lopuksi muodostetaan tekstin sisältö, johon haetaan arvot samalla tavalla kuin aiemmin pylvään leveyteen. Tosin tällä kertaa mukana on myös

d.puolue, eli kultakin datariviltä näytetään tekstinä puolueen nimi sekä halutun ajankohdan kannatus. Replace-funktiolla muutetaan desimaalierotin pisteestä pilkuksi.

Skriptiosion lopussa ovat funktiot, jotka ajetaan joko sivun latautuessa tai kun nappeja painetaan:

```
d3.csv("ontcase1_data.csv", function(data) {
          var tiedot = data;
          var aika = "uusi";
          vaihda(tiedot, aika);
});
```

Ensimmäinen näistä funktioista ladataan automaattisesti sivun latautuessa. Siinä CSV-tiedostojen käsittelyyn tarkoitetulle d3.csv-funktiolle annetaan ohjeeksi lukea ontcase1_data.csv-tiedosto. Sen jälkeen tallennetaan data tiedot-muuttujaan, muodostetaan aika-muuttuja ja lopuksi lähetetään data ja haluttu ajankohta vaihda-funktion käsiteltäviksi.

Lopuksi on vastaavat koodipätkät erikseen molemmille napeille:

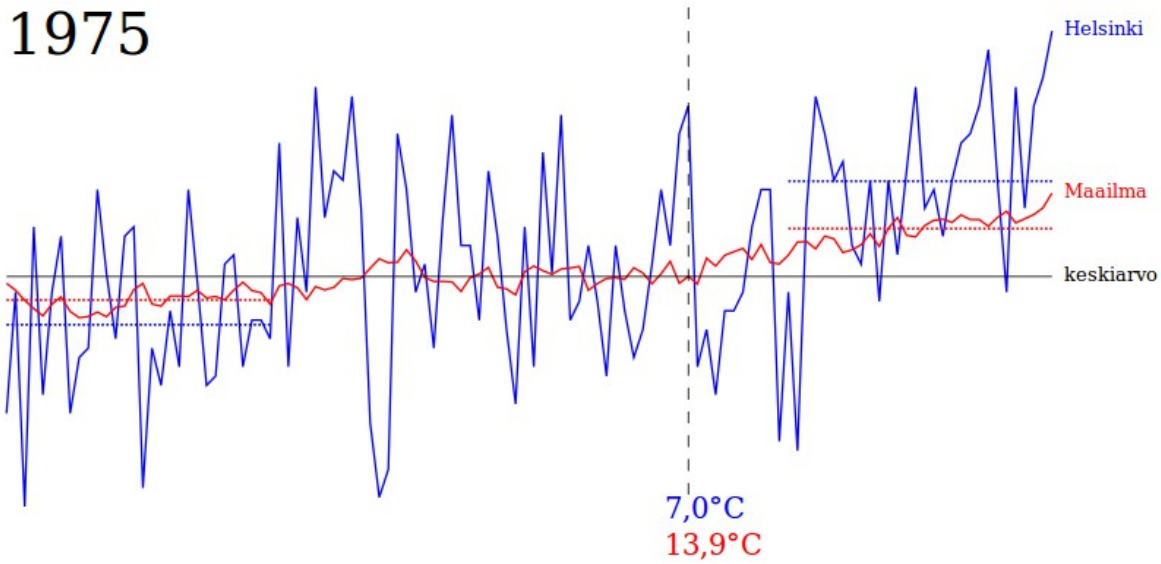
```
d3.select(".vanha")
          .on("click", function(){
                    d3.csv("ontcase1_data.csv", function(data) {
                              var tiedot = data;
                              var aika = "vanha";
                              vaihda(tiedot, aika);
                    });
          });

d3.select(".uusi")
          .on("click", function(){
                    d3.csv("ontcase1_data.csv", function(data) {
                              var tiedot = data;
                              var aika = "uusi";
                              vaihda(tiedot, aika);
                    });
          });
```

Funktio d3.select määrittää, mihin HTML-elementtiin toiminnallisuus lisätään. Seuraavaksi määritellään, että edellä esiteltyä, automaattisesti ajettavaa funktiota vastaava funktio ajetaan nimenomaan silloin kun määriteltyä HTML-nappulaa painetaan.

Täydellinen koodi sekä kuva toteutuksesta löytyvät liitteestä 1.

5 Case: Ilmaston lämpeneminen meillä ja muualla

Toisen case-esimerkin taustalla on juttuidea siitä, että ilmasto lämpenee Suomessa kaksi kertaa nopeammin kuin maailmassa keskimäärin. Tämä jää kuitenkin suomalaisilta huomaamatta, koska sää on aina vaihdellut täällä niin paljon. Ilmiötä käsittelevän jutun visualisoinniksi ideoitiin viivadiagrammi, joka esittää vuosittaiset keskilämpötilat Helsingissä ja maailmassa keskimäärin.



Kuva 10. Valmis visualisointi lämpötiloista.

Kuvassa 10 näkyvä lopullinen versio visualisoinnista näyttää lämpötilat valitun vuoden kohdalla. Vuoden valintaa on helpotettu hiiren mukana liikkuvalla pystyviivalla. Vuosien 1900–1929 ja 1986–2015 keskiarvot on merkitty vaakakatkoviivoin. Värienkäytöllä on pyritty tekemään visualisoinnista helpommin luettava: sininen kuvaa aina Helsinkiä ja punainen koko maailmaa. Visualisointi on julkaistu Vihreä Lanka -lehden verkkosivuilla marraskuussa 2016.

Case 2 on selvästi haastavampi kuin yksinkertainen puoluekannatusten esittäminen pylväinä, koska siinä esitetään useita eri aikasarjoja ja interaktiivisuus perustuu nappien sijaan hiiren liikkeisiin. Visualisointi on kuitenkin mahdollista toteuttaa muutamassa tunnissa.

5.1 Tiedonhankinta, tiedonkäsittely ja esitystapa

Maailmaa koskeva tilasto noudettiin Yhdysvaltain ilmatieteen laitoksen NOAA:n (2016) palvelusta. Helsingin keskilämpötilat ladattiin Suomen Ilmatieteen laitoksen sivuilta (2016a).

Tiedot ladattiin edellä mainituista palveluista taulukkomuodossa. Kaksi erillistä tietosarjaa liitettiin yhteen taulukkolaskentaohjelmassa. Taulukossa on kolme saraketta: vuosi, Helsingin lämpötila ja globaali lämpötila.

Koska kyseessä on perättäisten aikajaksojen – tässä tapauksessa vuosien – vertailu, on oikea kaaviotyyppi viivadiagrammi. Rinnakkaiset pylväät olisivat väärä valinta, koska lämpötila ei ala vuoden alussa nollasta vaan on vuodesta toiseen jatkuva suure. (Abela 2006.)

5.2 Tekninen toteutus

Tekninen toteutus on esitetty tässä suppeammin kuin ensimmäisessä, puoluekannatusta koskeneessa case-esimerkissä, koska perusrakenteen osalta koodit ovat hyvin samankaltaiset. Osa kahden esimerkin eroavaisuuksista johtuu siitä, että case 1 on toteutettu D3:n versiolla kolme ja caset 2 ja 3 versiolla neljä.

Toisin kuin huomattavasti yksinkertaisemmassa case 1:ssä tässä tapauksessa on käytetty scaleLinear-funktiota, jonka avulla graafi saadaan automaattisesti istumaan määriteltyyn tilaan, vaikka tietoaineisto päivittyisi ja ääriarvot kasvaisivat aiempaa suuremmiksi.

```
var yScale = d3.scaleLinear()
          .domain(d3.extent(data, function(d) { return d.hki; }))
          .range([210, 10]);
```

Esimerkiksi muuttujaan yScale tallenetaan funktio, jolle on määritelty domain (tietosisällön skaala alkuperäisellä mittayksiköllä) ja range (esityksessä käytettävä skaala pikseleinä). Tämän jälkeen tietoaineisto voidaan sitoa viivaelementteihin:

```
var line1 = d3.line()
          .x(function(d) { return xScale(d.year); })
          .y(function(d) { return yScale(d.hki); });
```

Muuttujasta line1 muodostetaan line-funktiolla SVG-koordinaatteja sisältävä objekti, jossa X-arvot määräytyvät xScale-funktion läpi ajetuista vuosiluvuista ja Y-akselin arvot määräytyvät yScale-funktion läpi ajettavista Helsingin lämpötiloista. Varsinainen kuvaajan piirtäminen tehdään näin:

```
svg.append("path")
          .datum(data)
          .attr("class", "lineHki")
          .attr("d", line1);
```

Luokan määritteleminen kolmannella rivillä mahdollistaa tyylien säätämisen CSS:llä.

Koska visualisoinnista halutaan sellainen, jota käyttäjä voi tutkia, lisätään siihen mahdollisuus tarkastaa hiirtä käyttämällä tiettyjen vuosien lämpötiloja. Tämä tehdään näkymättömällä suorakaiteella, joka muodostetaan kuvaajan päälle. Suorakaiteelle asetettava overlay-luokka määritellään seuraavasti CSS-tiedoissa:

```
.overlay {
          fill: none;
          pointer-events: all;
}
```

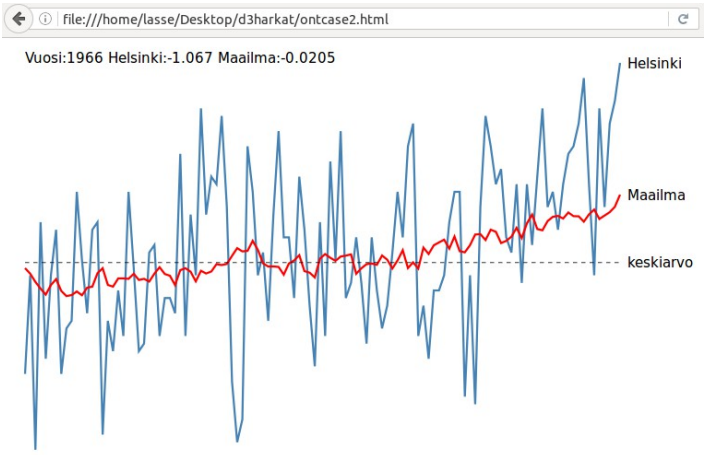
Itse suorakaide määritellään näin:

```
svg.append("rect")
          .attr("class", "overlay")
          .attr("x", 0)
          .attr("y", 0)
          .attr("width", 450)
          .attr("height", 210)
          .on("mouseover", function() {
                    yearLabel.style("display", null);
                    hkiLabel.style("display", null);
                    gloLabel.style("display", null);
                    valitsin.style("display", null);
          })
          .on("mousemove", mousemove);
```

Kun hiiren osoitin liikutetaan kuvion päälle (mouseover), poistetaan etikettien muodostamisessa käytetty displayn none-arvo. None-arvo on alussa tarpeen, ettei automaattisesti näytetä lämpötilaa, jota käyttäjä ei ole valinnut. Kun hiirtä liikutetaan (mousemove), ajetaan mousemove-funktio:

```
function mousemove() {
          var sijainti = Math.floor(xScale.invert(d3.mouse(this)
[0])-1899.5);
          var hkiLuku =
((+data[sijainti].hki+5.167).toFixed(1)+"°C").replace(/\./g, ',');
          var gloLuku =
((+data[sijainti].glo+13.9).toFixed(1)+"°C").replace(/\./g, ',');
          yearLabel.text(data[sijainti].year);
          hkiLabel
                    .text(hkiLuku)
                    .attr("x", xScale(data[sijainti].year)-10);
          gloLabel
                    .text(gloLuku)
                    .attr("x", xScale(data[sijainti].year)-10);
          valitsin
                    .attr("x1", xScale(data[sijainti].year))
                    .attr("x2", xScale(data[sijainti].year));
}
```

Mousemove-funktio hakee hiiren sijainnin ja laskee sen perusteella, mitä viivadiagrammin arvoa se vastaa. Lämpötilatietoja ja valitsin-pystyviivaa liikutetaan X-akselilla hiiren osoittimen sijainnin mukana.



Kuva 11. Keskeneräinen versio visualisoinnista.

Kuvassa 11 on keskeneräinen versio graafista, jossa näkyvät molemmat lämpötilakäyrät sekä keskiarvoakseli. Valitsimen pystyviiva ei vielä näy ja lämpötila- sekä vuositiedot ovat tyylittelemättöminä vasemmassa yläkulmassa.

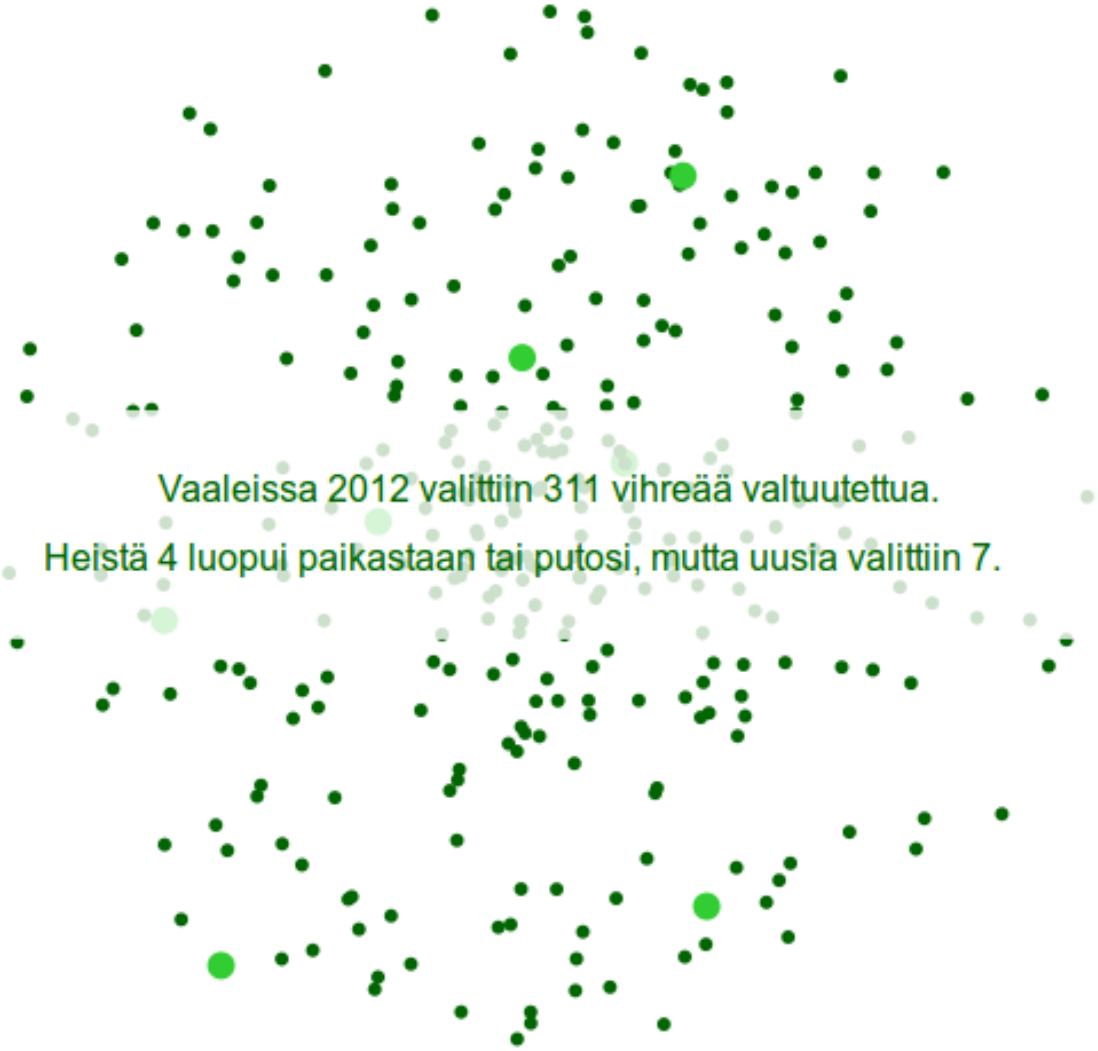
Yleisen lämpenevän trendin osoittamiseksi visualisointiin on lisätty katkoviivat, jotka osoittavat sekä Helsingin että koko maailman keskilämpötilat mittausdatan ensimmäisiltä ja viimeisiltä 30 vuodelta. Ilmatieteen laitoksen (2016b) mukaan 30 vuoden ajanjakso on riittävän pitkä kertomaan ilmaston keskilämpötilasta ilman jaksoittaisten vaihteluiden aiheuttamaan vääristymää. Tämä tehdään viivaelementeillä, joita ei sidota dataan vaan joille annetaan kiinteät arvot.

```
svg.append("line")
          .style("stroke", "blue")
          .style("stroke-dasharray", ("1, 1"))
          .attr("x1", xScale(1900))
          .attr("y1", yScale(-0.517))
          .attr("x2", xScale(1929))
          .attr("y2", yScale(-0.517));
```

Täydellinen koodi sekä kuva toteutuksesta löytyy liitteestä 2.

6 Case: Vihreiden kuntavaalituloksen visualisointi

Kolmas visualisointi on valmis pohja käytettäväksi vuoden 2017 kuntavaalien tuloksen ratkettua. Toteutusvaiheessa on käytetty vuoden 2012 vaalitulosdataa. Mukana ei ole noissa vaaleissa valtuustosta pudonneita vaan osa 2012 valituista on merkitty tuolloin pudonneiksi, jotta visualisoinnin toiminnallisuutta voidaan testata. Tästä syystä esimerkkikuvissa näkyvät luvut eivät pidä paikkaansa. Julkaisuvaiheessa luvut korjataan vuoden 2017 vaalituloksen mukaisiksi. Lopputulos on lyhyt intro-animaatio, joka näyttää aluksi kaikki vaaleissa 2012 valitut vihreät valtuutetut, etenee sitten automaattisesti niin, että pudonneita valtuutettuja edustavat pallot poistuvat kuvasta. Samalla uusia edustajia kuvaavat, hieman isommat ja eri väriset, pallot tulevat esiin. Näitä siirtymiä selitetään tekstein.



Kuva 12. Introvaiheessa valittuja symboloivat pallot näkyvät taustalla ja valtuutettumäärän muutoksia selitetään tekstein.

Lopulta visualisointi päätyy karttatilaan, jossa käyttäjä voi itse tutkia, keitä on valittu missäkin kunnassa.

Kuva 13. Karttatilassa voi valita yksittäisen kunnan ja tarkastella siinä valittuja ehdokkaita.

Visualisointi on määrä julkaista Vihreä Lanka -lehden sivuilla huhtikuussa 2017 mahdollisimman pian vaalituloksen ratkettua.

Visualisointi oli datan hankinnan ja jalostuksen sekä erityisesti varsinaisen toteutuksen osalta kolmesta case-esimerkeistä työläin. Rutinoitunut D3:n käyttäjä pystyy kuitenkin toteuttamaan tällaisen visualisoinnin muutamassa päivässä.

6.1 Tiedonhankinta, tiedonkäsittely ja esitystapa

Tarvittava data koostuu kahdesta erillisestä aineistosta. Kuntien esittämiseen maantieteellisellä koordinaatistolla on tarvittu kuntakeskusten sijaintitiedot. Olennaisin tietoaineisto on kuntavaalien tulostiedot, joka tarkoittaa tässä kohdin vuoden 2012 vaalitulosta, mutta julkaistavaan versioon päivitetään vuoden 2017 tulos.

Kuntakeskuksien koordinaattitiedot on hankittu MapQuest-palvelusta Yleisradion Teemo Tebestin (2013) Datajournalismi-blogissa esittelemää keinoa hyväksikäyttämällä. Suomen Kuntaliiton (2016a) ajantasaisen listan kunnat muutettiin ensin CSV-muotoon lisäten jokaiselle riville toiseksi arvoksi "Finland", jottei joukkoon tule vahinko-osumia ulkomailta. Tämä CSV-muotoinen lista ajettiin GPS Visualizerin Geocoder-palvelun läpi ja lopputuloksena saatiin lista koordinaateista. Tämän jälkeen on käsin korjattu ne kolme kuntaa (Suomen Kuntaliitto 2016b), jotka liittyvät toisiinsa vuoden 2017 alussa. Käytännössä tämä tarkoittaa, että liitoksien pienemmät kunnat on poistettu listalla.

Vuoden 2012 jälkeen tapahtuneiden kuntaliitoksien kohdalla 2012 vaaleissa valitut on merkitty uuteen kuntaansa. Vaikka tämä ei pidä tarkalleen ottaen paikkaansa, haitta on vähäinen, koska vuoden 2012 vaalitulosta ei julkaistavassa toteutuksessa esitetä kuntatasolla vaan ainoastaan intro-vaiheen taustalla. Eri kuntien valtuustoihin vuosien 2012 ja 2017 vaaleissa valitut esitetään sekä pudonneiden että uusien joukossa, koska oman kuntansa kohdalla he ovat juuri tätä.

Koska toteutuksessa on tarkoitus esittää dataa useammalla eri tasolla, ylemmällä tasolla kuntatason tietoja ja alemmalla ehdokastason tietoja, muunnettiin CSV-muotoinen lista kuntien nimistä ja koordinaateista JSON-muotoon.

Vaalien tulostiedot on poimittu käsin Ylen Kuntavaalit 2012 -palvelusta (Yle 2012). Statuskenttä kertoo, onko kyseinen valtuutettu pudonnut (0), jatkava (1) vai uusi (2). Mukaan on poimittu myös kaikkien kussakin kunnassa annettujen äänien määrä sekä valittujen äänimäärät, vaikkei näitä tietoja lopulta visualisoinnissa käytetty. Tämä mahdollistaa visualisoinnin jatkokehittämisen.

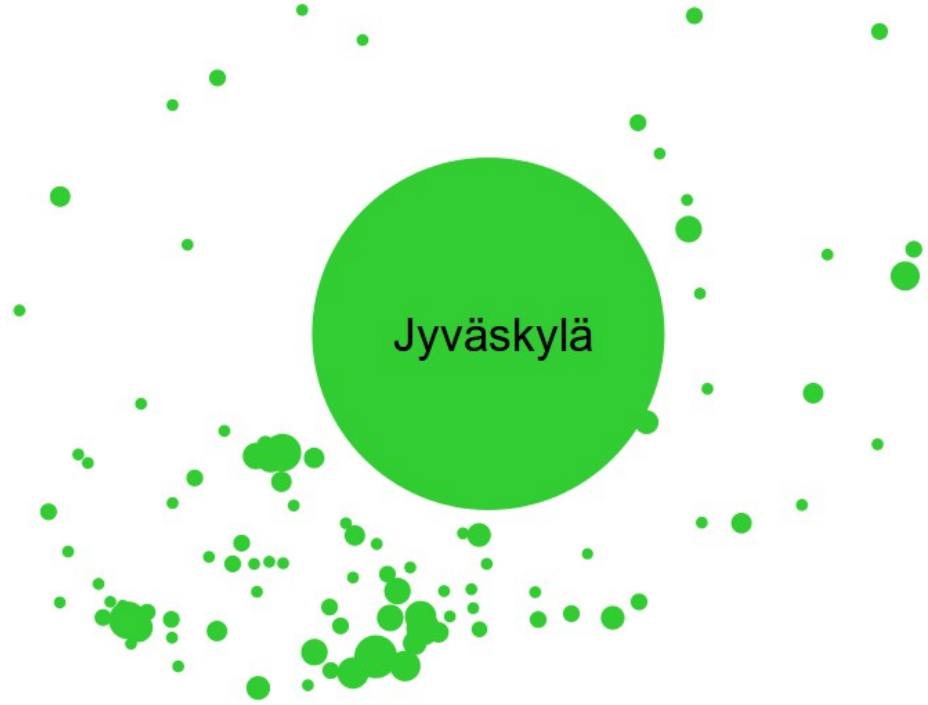
Yksittäisen kunnan data näytti lopulta tältä:

```
{
     "name": "Enontekiö",
     "totvotes": 1137,
     "lat": 68.38549,
```

```
        "lon": 23.643798,
        "children": [
            {"name": "Birgitta Eira", "status": 1, "pervotes": 48},
            {"name": "Miliza Kimmel", "status": 1, "pervotes": 32}
        ]
},
```

Koska kyseessä ei ole varsinaisesti tilastotyyppisen datan esittäminen (ei aikasarjoja eikä keskenään vertailtavia arvoja), eivät perinteiset käytännöt ja säännöt (Liite 4) rajoittaneet esitystavan valintaa.

Karttanäkymässä esitetään kunkin kuntakeskuksen kohdalla pallo, jonka pinta-ala on suoraan verrannollinen kyseisessä kunnassa valittujen vihreiden valtuutettujen määrään. Tämä esitystapa antaa yhdellä silmäyksellä kuvan siitä, mihin vihreiden valtuutettujen valitseminen maantieteellisesti painottuu.



Kuva 14. Suurennos Etelä-Suomesta siten, että Jyväskylä on valittuna.

Koska toteutuksessa pyrittiin äärimmäiseen yksinkertaisuuteen, ei karttanäkymässä käytetä taustakarttaa vaan Suomen muoto hahmottuu valittujen edustajien mukaan (Kuva 14). Eli sitä osaa Suomea, jonne vihreitä ei ole valittu ei ikään kuin ole olemassa.

6.2 Tekninen toteutus

Kuten case-esimerkissä 2 tässäkään tapauksessa ei käydä läpi koodin jokaista riviä vaan olennaisimmat erot aiempiin esimerkkeihin. Täydellinen koodi löytyy liitteestä 3.

Data tuodaan visualisointiin kaksi erillistä kertaa: introa varten ehdokastasolla ja karttanäkymää varten kuntatasolla. Ehdokastasolla kuntatason tiedot liitetään jokaisen ehdokkaan yhteyteen forEach-funktiolla.

```
d3.json("oikeakuntadata.json", function(error, json) {
          if (error) throw error;
          json.tulokset.forEach(function(d) {
                    var kuntaNimi = d.name;
                    var kuntaVotes = d.totvotes;
                    var kuntaLon = d.lon;
                    var kuntaLat = d.lat;
                    d.children.forEach(function(d) {
                              if (d.status == 0) {
                                        pudokkaat = pudokkaat + 1;
                              } else if (d.status == 1) {
                                        jatkajat = jatkajat + 1;
                              } else {
                                        uudet = uudet + 1;
                              };
                              d.kname = kuntaNimi;
                              d.totvotes = kuntaVotes;
                              d.lon = kuntaLon;
                              d.lat = kuntaLat;
                              data.push(d);
```

Kuntatasolla jokaisen kunnan tietoihin lisätään lista valittujen nimistä ja tämän listan pituus eli valittujen lukumäärä:

```
d3.json("oikeakuntadata.json", function(error, json) {
          if (error) throw error;
          json.tulokset.forEach(function(d) {
                    var kunta = {name: d.name, totvotes: d.totvo-
tes, lon: d.lon, lat: d.lat};
                    var nimet = [];
```

```
                    d.children.forEach(function(d) {
                              if (d.status != 0) {
                                        nimet.push(d.name);
                              };
                    });
                    kunta.nimet = nimet;
                    kunta.kuntaPaikat = nimet.length;
                    kuntaData.push(kunta);
          });
          kaupungit(kuntaData);
});
```

Introssa valitut esitellään ympyrän muotoisena parvena, jossa heidän sijaintinsa on sattumanvarainen. Tämä on saatu arpomalla jokaiselle etäisyys ja suunta kuvion keskustasta. Kasautumisen välttämiseksi etäisyyden minimiarvoksi on laitettu 15.

```
var angle = (Math.random() * Math.PI * 2);
var radius = (15 + Math.random() * 225);
```

Varsinaiset koordinaatit on laskettu kulman ja säteen perusteella sin- ja cosini-funktioita käyttäen esimerkiksi näin:

```
var deltax = Math.sin(angle)*radius;
var deltay = -Math.cos(angle)*radius;
```

Introvaiheen selitystekstit esitetään läpikuultavassa laatikossa, jotta taustalla olevat pallot eivät haittaa tekstin lukemista. Animoinnissa on käytetty transition-ominaisuutta, jolle voi antaa keston millisekunteina. Tällä on saatu esimerkiksi pudonneiden ehdokkaiden pallot putoamaan (käytännössä siirtymään SVG-kanvaksen alalaitaan ja muuttumaan taustan värisiksi) sulavasti näkyvistä:

```
svg.selectAll(".status0")
          .transition()
                    .duration(2000)
                    .style("fill", "white")
                    .attr("cy", 500);
```

Intro-animaation kaksi vaihetta etenevät samaa transition-ominaisuutta käyttäen automaattisesti ilman, että käyttäjän tarvitsee tehdä mitään. Siirryttäessä karttaan kunkin eh-

dokkaan sattumanvarainen paikka vaihtuu aitoon sijaintiin kartalla. Samalla kuitenkin piirretään kullekin kunnalle paikkamäärää vastaavan kokoinen pallo. Tämä antaa vaikutelman saumattomasta siirtymästä, vaikka oikeasti siirrytään uuteen dataan ja uusiin elementteihin vanhojen jäädessä alle piiloon.

Kun hiiri viedään jonkun kaupunkia tai kuntaa edustavan pallon kohdalle, pallo suurenee ja sen sisälle ilmestyy kyseisen paikan nimi:

```
.on("mouseover", function(d) {
          d3.select(this)
                    .transition()
                              .duration(200)
                              .attr("r", 50);
          svg.append("text")
                    .text(d.name)
                    .attr("class", "kuntaNimi")
                    .attr("x", x(d.lon)-d.name.length*3)
                    .attr("y", y(d.lat)+5)
                    .style("fill", "black");
```

Samalla lista valittujen valtuutettujen nimistä esitetään kartan oikealla puolella. Vastaavasti, kun hiiren osoitin viedään pallon ulkopuolelle, pallo palaa alkuperäiseen kokoonsa ja nimilista poistetaan näkyvistä:

```
.on("mouseout", function(d) {
     d3.select(this)
          .transition()
               .duration(400)
               .attr("r", function(d) {
                    return 2*Math.sqrt(d.kuntaPaikat/Math.PI);
               });
     d3.selectAll(".nimiLista").remove();
     d3.selectAll(".kuntaNimi").remove();
});
```

Karttanäkymä jatkaa toimintaansa, kunnes käyttäjä sulkee sivun.

Täydellinen koodi sekä kuva toteutuksesta löytyy liitteestä 3.

7 Pohdinta

Opinnäytetyön tavoitteena oli kartoittaa D3.js-kirjaston hyödyllisyyttä pienen toimituksen käytössä. D3 on kiinnostava, koska se on ilmainen ja se on kehitetty nimenomaan journalististen datavisualisointien toteuttamiseen.

Opinnäytetyön lopputuloksena syntyi Vihreä Lanka -lehden verkkosivujen käyttöön kolme datavisualisointia, joista yksi julkaistiin kesäkuussa 2016 ja toinen marraskuussa 2016. Kolmas visualisointi tullaan julkaisemaan huhtikuussa 2017. Kukin visualisointi tukee ja auttaa ymmärtämään kirjallista journalistista ilmaisua. Puolueiden kannatusta kuvaava pylväikkö näyttää visuaalisesti muutoksen edellisten vaalien jälkeen. Ilmaston lämpenemistä kuvaava graafi antaa käyttäjälle mahdollisuuden tutkia yksittäisten vuosien lämpötiloja, mutta tarjoilee ennen kaikkea numeromuotoista dataa visuaalisessa ja siten helpommin omaksuttavassa muodossa. Kuntavaalituloksen visualisoinnissa on mukana kerronnallisuutta ja käyttäjälle annetaan mahdollisuus tehdä omia tulkintoja data-aineistosta.

Kaikkia kolmea visualisointia yhdistää se, että ne ovat jatkoa ajatellen käyttökelpoisia pohjia, joita on helppo soveltaa toisenlaisiin käyttötarkoituksiin. Esimerkiksi animoitu pylväsgrafiikka voisi puoluekannatuksien sijaan esittää esimerkiksi tiettyyn aihepiiriin liittyvien väittämien kannatuksen muutoksia. Lämpötilakäyrä soveltuu puolestaan lähes sellaisenaan esimerkiksi valuuttakurssien tai muun tasaisesti dataa tuottavan ilmiön esittämiseen. Kuntavaalivisualisoinnin ilmiselviä käyttötapoja ovat tietysti muut vaalit, mutta samaa pohjaa voisi soveltaa esimerkiksi tulotason tai veroäyrin esittämiseen kuntakohtaisesti.

Luonnollisin ja yksinkertaisin uudelleenkäyttötapa on tietojen päivittäminen uusimmilla gallup-kannatuksilla tai lämpötilatiedoilla. Tällöin syntyneitä grafiikkoja voidaan käyttää sellaisenaan jatkuvasti uudelleen.

Yksi tutkimuksen tavoitteista oli selvittää, millainen on D3:n hyödyntämiseen tarvittava työprosessi. Kolmen esimerkin toteuttamisessa saamani kokemuksen perusteella työprosessi on teoriassa ja lähtökohtaisesti hyvin samanlainen kuin missä tahansa journalismissa: ideasta edetään tiedonhankintaan ja lopulta itse toteutukseen (Reynolds 2012). D3:n, kuten muidenkin monipuolisten ja monimutkaisten työkalujen, kohdalla prosessi ei kuitenkaan ole aina näin suoraviivainen. Tämä vastaa muiden kokemuksia: Mattingly et al. (2015) ovat tutkineet D3:n soveltuvuutta biolääketieteellisten ilmiöiden visualisointiin ja todenneet, että D3:a tulee hyödyntää iteratiivisen, yrityksen ja erehdyksen kautta etenevän työprosessin kautta.

On tärkeää hahmottaa, mihin D3 soveltuu ja mihin ei. Cukier (2015) korostaa, että monet niistä asioista, jotka voidaan tehdä D3:lla, voidaan tehdä pelkällä JavaScriptillä tai jollain kevyemmällä kirjastolla. Esimerkiksi pelkkään DOM-elementtien valitsemiseen ja attribuuttien kuten koon tai värin muuttamiseen riittää JavaScript. Esimerkiksi puoluekannatuksien visualisoinnin olisi todennäköisesti pystynyt tekemään vähintään yhtä helposti pelkällä JavaScriptillä.

D3:a ei pidä nähdä Chart.js:n kaltaisena valmiina ohjelmana, joka tekee suoraan luvuista visualisointeja vaan D3:ssa on kyse datavisualisointityökalusta, jossa on kyse niin datan kuin visualisoinnin työstämisestä (Sun 2014). Tämän opinnäytetyön kohdalla datan työstäminen D3:n avulla jäi vähäiseksi, koska kahdessa ensimmäisessä esimerkissä käytetty data oli melko valmista visualisoitavaksi. Kolmannen esimerkin vaalidataa olisi jälkikäteen arvioiden voinut yrittää käsitellä D3:n avulla, kun nyt se tehtiin käsin.

Kolmesta esimerkkitapauksesta saamani kokemuksen perusteella työprosessin on hyvä olla – kuten esimerkiksi Sharma (2015) kannustaa – sillä tavoin osallistava, että keskeneräisestä työstä pyydetään ja saadaan palautetta. Tällöin ongelmakohdat voidaan havaita ajoissa ja niihin on mahdollisesti helpompi puuttua. Toisaalta D3-koodin rakenne osoittautui siinä määrin selkeäksi ja joustavaksi, että suuriakin muutoksia on melko helppo tehdä vielä loppusuoralla.

7.1 Resurssien kysyntä ja tarjonta

D3:n suurimmat vahvuudet ovat sen avoimuus, ilmaisuus ja yleisyys sekä tekninen yksinkertaisuus, joiden vuoksi verkosta löytyy runsaasti dokumentaatiota, ohjeita ja esimerkkejä (Skau 2013). D3:n hyödyntäminen ei myöskään vaadi erityistä laitteistoa tai ohjelmistoa, mikä tekee siitä taloudellisessa mielessä helposti lähestyttävän.

Toisaalta D3:n vaatimat tekniset taidot tekevät sen melko haastavaksi. Kolme melko yksinkertaista esimerkkiä osoittivat, että D3:n tehokas hyödyntäminen edellyttää hyvää HTML:n, CSS:n, SVG:n sekä erityisesti JavaScriptin tuntemusta (Cukier 2015). Tämän lisäksi kirjasto itsessään on niin laaja ja monipuolinen, että sen oppimiseen kuluu aikaa. Tämän tyyppiset taidot eivät kuulu tavallisen toimittajan työkaluihin, joten D3:n hyödyntäminen edellyttää joko alan koulutusta tai harrastuneisuutta. Toimittajien hallitsemat taidot saattavat olla kuitenkin muuttumassa, sillä eurooppalaisilla toimittajilla tehdyn tutkimuksen mukaan datajournalismitaidoille on kysyntää (ECJ 2011).

Esimerkit havainnollistivat myös sitä, miksi datavisualisointien tekeminen keskittyy suurimpiin toimituksiin, joilla on usein omat asiaan erikoistuneet toimittajat tai ohjelmoijat. Oman

kokemukseni perusteella Vihreän Langan kaltaisessa pienessä toimituksessa yhden toimittajan irrottaminen muusta työstä visualisoinnin toteuttamiseen on erittäin haastavaa. Tämä korostuu tapauksissa, joissa visualisoinnit jäävät varsinaisten juttujen kuvitukseksi tai tueksi.

D3 ei ole helppo ja nopea työkaluvisualisointien tekemiseen (Cukier 2015). On huomattavasti nopeampaa tehdä yksinkertainen pylväs- tai viivadiagrammi Excelillä tai Illustratorilla ja liittää se juttuun kuvana taikka hyödyntää Chart.js:n kaltaista valmiimpaa visualisointikirjastoa. Rutinoitunut D3:n taitava tekijä pystyy kuitenkin tekemään näyttäviäkin visualisointeja muutamassa tunnissa, mutta tämä edellyttää, että D3-toteutuksen koetaan tuovan jotain lisäarvoa. (Sun 2014.)

Lisäarvon syntymistä voidaan pohtia tämän opinnäytteen kolmen case-esimerkin kohdalla. Pylväsgrafiikassa puolueiden vaalikannatuksen olisi voinut esittää staattisesti gallupkannatuksen rinnalla, jolloin lukija olisi saanut täsmälleen saman tiedon ilman tarvetta painaa nappia. Samoin lämpötilavisualisoinnin kohdalla voisi väittää, että olennaisin tieto (eri tahtia nousevat käyrät) näkyisi myös staattisessa versiossa. Interaktiivisuus antaa mahdollisuuden tutkia yksittäisten vuosien lämpötiloja, mutta tämä tieto on melko satunnaista ja siten hieman irrelevanttia. Toki molemmissa tapauksissa lisäarvo syntyisi välittömästi, jos tietolähteenä käytettäisiin staattisen datan sijaan automaattisesti päivittyvää dataa – tällaiseen ei perinteinen kuva luonnollisestikaan pysty. Toisaalta interaktiivisuus voi saada lukijan tutkimaan grafiikkaa hieman pidemmäksi aikaa, jolloin pääviestikin saatetaan sisäistää paremmin (Bee, Kiok & Zhecheng 2014).

Ainoastaan kuntavaalidatan kohdalla voidaan puhua todellisesta lisäarvosta. Intro-animaatio saattaa pysäyttää lukijan ja toimia siten vahvempana houkuttimena kuin pelkkä kuva. Pelkkään kuvaan olisi toki mahdollista saada kartta, jossa kuntakeskuksien kohdalla olevat pallot kertovat vihreiden valtuutettujen määrän, mutta satojen valtuutettujen nimien listaaminen yhdessä kuvassa olisi epäkäytännöllistä ellei mahdotonta.

Visualisointien toteuttamiseen tarvittava aika vaihtelee suuresti riippuen toteuttajan taidoista sekä toteutettavan visualisoinnin monimutkaisuudesta ja ainutlaatuisuudesta. Ainutlaatuisuudella tarkoitetaan tässä sitä, että omaperäisten visualisointien – kuten kolmannen casen kuntavaalituloksen – toteuttaminen on huomattavasti hitaampaa kuin sellaisten, joihin löytyy verkosta melko suoraan sovellettavia esimerkkejä. Tämän opinnäytetyön esimerkkeihin kului aktiivista työaikaa muutamasta tunnista muutamaan päivään.

Koska D3 vaatii datan jalostamisen tiettyyn muotoon eikä koodi ole missään tapauksessa ainoastaan muutamaa riviä, vie yksinkertaisimmankin visualisoinnin – kuten ensimmäisen casen puoluekannatuspylväiden – noin tunnin. Vertailuna esimerkiksi Vihreän Langan verkkosivuilla kuvituksien hankkimiseen tai toteuttamiseen käytetään oman kokemukseni perusteella tyypillisesti vain joitakin kymmeniä minuutteja, joten D3-visualisoinnin toteuttaminen olisi aina huomattavasti hitaampi ja työläämpi prosessi.

7.2 Johtopäätökset

Vihreän Langan tarpeet liittyvät kokemuksieni perusteella enimmäkseen pieniin ja yksinkertaisiin visualisointeihin, joita julkaistaan juttujen yhteydessä. Taidolliset resurssit ovat tärkeä tekijä, sillä Vihrellä Langalla ei ole palkattua ohjelmoijaa vaan visualisoinnit pitäisi saada aikaan toimittajien ja graafikon taidoilla.

D3 soveltuukin huonosti pienen toimituksen käyttöön, ellei toimittajilta ole sattumalta – esimerkiksi harrastuksen tai aiempien työtehtävien myötä – tarvittavaa osaamista. Toinen vaihtoehto on ostaa koodaaminen ulkoisena palveluna. Kansainvälisten esimerkkien perusteella datajournalismin tekeminen edellyttää suurempia erikoisyksikköjä, joissa on joko erillisiä ohjelmoijia tai taitoon jatkokoulutettuja toimittajia (Crucianelli & Zanchelli 2014).

Oman kokemukseni perusteella mahdollisuudet hyödyntää D3-visualisointeja Vihreän Langan kaltaisen pienen toimituksen arkisessa toimitustyössä ovat sen verran rajalliset, ettei liene tarkoituksenmukaista ryhtyä kouluttamaan toimittajille tällaisia taitoja, jos niille ei ole muuta tarvetta. Sama rajallisuus tehnee ulkopuolisen koodausavun ostamisesta todennäköisesti kannattamatonta.

Kaikista näistä haasteista huolimatta D3 soveltuu hyvin verkkosivuilla julkaistavien visualisointien toteuttamiseen, jos niihin halutaan interaktiivista toiminnallisuutta tai mahdollisuus hyödyntää päivittyvää dataa. Kyseessä on lopulta yksinkertainen HTML-elementti sekä siihen liittyvä skripti, jotka on helppo sijoittaa mille tahansa verkkosivulle. D3-visualisoinneista saa melko helposti responsiivisia, jolloin ne joustavat saumattomasti verkkosivun muun sisällön mukana.

Yhtenä opinnäytetyön tavoitteena oli edistää henkilökohtaista D3-tuntemustani. Tässä tapauksessa prosessi on palvellut tarkoitustaan hyvin, sillä olen oppinut ymmärtämään aiempaa paremmin D3:n käyttömahdollisuuksia ja toimintaperiaatetta. Luulin aluksi, että kyseessä olisi suoraviivaisempi visualisointityökalu, mutta ymmärrän nyt paremmin, että D3:n ytimessä on datan käsittely. D3:n avulla on mahdollista toteuttaa monenlaista vi-

suaalisuutta verkkosivuille, mutta tarkoituksenmukaiseksi sen käyttäminen tulee vasta silloin, kun visuaalisuus perustuu suuriin data-aineistoihin.

Visualisointien toteuttaminen D3:lla osoittautui melko ohjelmoijaystävälliseksi, sillä console.log-funktiolla ja selaimen JavaScript-consolilla oli helppo pitää itsensä tietoisena siitä, mikä toimii ja mikä ei. Kuntavaalituloksen visualisoinnissa pääsin kokeilemaan D3:n rajoja suorituskyvyn näkökulmasta, sillä alkuvaiheessa testasin visualisointia aineistolla, johon oli generoitu yli tuhat valittua ja tällöin käyttöliittymä alkoi selvästi hidastua. Tämä on itselleni arvokasta tietoa jatkoa ajatellen, sillä nyt osaan paremmin ennakoida, millaisissa tilanteissa Skaun (2013) varoittama rajallinen suorituskyky tulee vastaan.

Uskon, että tämä kokemus auttaa tulevaisuudessa käyttämään D3.js-kirjastoa tarkoituksenmukaisemmin. D3:n mahdollisuuksista tuli tämän opinnäytetyön puitteissa käytyä läpi vain pieni osa ja itseäni kiinnostuvat jatkossa muun muassa karttadataan ja reaaliaikaiseen aineistoon perustuvat sovellukset.

Lähteet

Abela, A. 2006. Choosing a good chart. Luettavissa: http://extremepresentation.typepad.com/blog/2006/09/choosing_a_good.html. Luettu: 13.8.2016.

Ashkenas, Bloch, Carter & Cox. 2012. The Facebook Offering: How It Compares. Luettavissa: http://www.nytimes.com/interactive/2012/05/17/business/dealbook/how-the-facebook-offering-compares.html?_r=0. Luettu: 16.8.2016.

Bee, Kiok & Zhecheng. 2014. Interactive Data Visualization to Understand Data Better: Case Studies in Healthcare System. Luettavissa: http://www.igi-global.com/article/interactive-data-visualization-to-understand-data-better/147300. Luettu: 12.11.2016.

Bostock, M. 2012. Uber Rides by Neighborhood. Luettavissa: https://bost.ocks.org/mike/uberdata/. Luettu: 23.8.2016.

Bostock, M. 2016. Changes in D3 4.0. Luettavissa: https://github.com/d3/d3/blob/master/CHANGES.md. Luettu: 15.8.2016.

Bostock M., Heer J. & Ogievetsky V. 2011. D3: Data-Driven Documents. Luettavissa: http://idl.cs.washington.edu/papers/d3/ Luettu: 5.5.2016.

Bradshaw, P. 2012. What Is Data Journalism? Luettavissa: http://datajournalismhandbook.org/1.0/en/introduction_0.html. Luettu: 8.9.2016.

Buezas D. 2016. Pie chart labels. Luettavissa: http://bl.ocks.org/dbuezas/9306799. Luettu: 18.7.2016.

Cairo, A. 2014. Infographics to explain, data visualizations to explore. Luettavissa: http://www.thefunctionalart.com/2014/03/infographics-to-reveal-visualizations.html. Luettu: 20.8.2016.

Cook, P. 2016. What is D3? Luettavissa: http://animateddata.co.uk/articles/d3/whatisd3/. Luettu: 28.8.2016.

Crucianelli S. & Zanchelli M. 2014. Integrating Data Journalism into Newsrooms. International Center for Journalists.

Cukier, J. 2015. You may not need d3. Luettavissa: http://www.jeromecukier.net/blog/2015/05/19/you-may-not-need-d3/. Luettu: 16.10.2016.

D3 Wiki. 2016a. Browser / Platform Support. Luettavissa: https://github.com/d3/d3/wiki. Luettu: 30.5.2016.

D3 Wiki. 2016b. API Reference. Luettavissa: https://github.com/d3/d3/wiki/API-Reference Luettu: 30.5.2016.

Dormehl, L. 2014. The Five Best Libraries For Building Data Visualizations. Luettavissa: https://www.fastcompany.com/3029760/the-five-best-libraries-for-building-data-vizualizations. Luettu: 8.9.2016.

ECJ. 2011. Training data driven journalism: Mind the gaps. Luettavissa: http://datadrivenjournalism.net/news_and_analysis/training_data_driven_journalism_mind_the_gaps. Luettu: 16.10.2016.

Geere, D. Engaging People Around Your Data. Luettavissa: http://datajournalismhandbook.org/1.0/en/delivering_data_10.html. Luettu: 15.10.2016.

Github. 2016. Gallery. Luettavissa: https://github.com/d3/d3/wiki/Gallery. Luettu: 16.8.2016.

Grönroos, J. 2015. Datavisualisointi prosessi. Metropolia Ammattikorkeakoulu.

Guardian. 2012. Luettavissa: http://www.theguardian.com/world/interactive/2012/may/08/gay-rights-united-states. Luettu: 16.8.2016.

Haaga-Helia. 2016. Ohje pitkien raporttien laatimiseen. Haaga-Helia Ammattikorkeakoulu.

Ilmatieteen laitos. 2016a. Vuositilastot. Luettavissa: http://ilmatieteenlaitos.fi/vuositilastot. Luettu: 23.8.2016.

Ilmatieteenlaitos. 2016b. Ilmastollinen vertailukausi 1981–2010. Luettavissa: http://ilmatieteenlaitos.fi/ilmastollinen-vertailukausi-1981-2010. Luettu: 28.8.2016.

Jackson, H. 2015. Animated chloropleth: England and Wales property prices. Luettavissa: http://helenjacksonanalytic.co.uk/EW_prop_price.html. Luettu: 23.8.2016.

Jugon, R. 2011. When data journalism gets it wrong... Luettavissa: https://drivenbydata.wordpress.com/2011/02/22/when-data-journalism-goes-wrong. Luettu: 15.10.2016.

Klein, S. 2016. The Forgotten History of Visualization in News. Luettavissa: http://www.tapestryconference.com/blog/2016/scott-klein-forgotten-history-visualization-news. Luettu: 12.11.2016.

Koning, J.P. 2012. Who charts the chartists? Luettavissa: http://jpkoning.blogspot.fi/2012/10/data-visualization-breaking-down.html. Luettu: 15.10.2016.

Kuutti, H. & Uskali, T. 2016. Datajournalismin työkäytännöt. Vastapaino. Tallinna.

Kosara, R. 2007. Visualization Criticism – The Missing Link Between Information Visualization and Art. Proceedings of the 11th International Conference on Information Visualisation. Luettavissa: http://kosara.net/papers/2007/Kosara_IV_2007.pdf. Luettu: 20.8.2016.

Kosara, R. 2011. Anscombe's Quartet. Luettavissa: https://eagereyes.org/criticism/anscombes-quartet. Luettu: 16.8.2016.

Kosonen, I. 2016. Asylum applicants in Europe & Finland in 2015. Luettavissa: http://inkakosonen.com/work/datavisualisointi-turvapaikan-hakijat-euroopassa-vuonna-2015/. Luettu: 18.7.2016.

Mattingly, W et al. 2015. An iterative workflow for creating biomedical visualizations using Inkspace and D3.js. 14th Annual UT-KBRIN Bioinformatics Summit 2015.

Mitevski, G. 2010. Creating Graphs With Adobe Illustrator. Luettavissa: https://www.smashingmagazine.com/2010/09/creating-graphs-with-adobe-illustrator/. Luettu: 15.10.2016.

Mozilla. 2016a. SVG. Luettavissa: https://developer.mozilla.org/en-US/docs/Web/SVG. Luettu: 23.8.2016.

Mozilla. 2016b. <line>. Luettavissa: https://developer.mozilla.org/en-US/docs/Web/SVG/Element/line. Luettu: 23.8.2016.

Möller, C. 2015. Replacing jQuery with D3. Luettavissa: http://blog.webkid.io/replacing-jquery-with-d3/. Luettu: 18.8.2016.

Nixon, R. 2015. Why HTML5 must replace Flash. Luettavissa: http://www.computerweekly.com/opinion/Why-HTML5-must-replace-Flash. Luettu: 8.9.2016.

Nieminen, J. 2014. Avoimen datan visualisointi verkkoselaimessa. Metropolia Ammattikorkeakoulu.

NOAA. 2016. Climate at a Glance. Luettavissa: http://www.ncdc.noaa.gov/cag/time-series/global. Luettu: 23.8.2016.

Oikeusministeriö 2015. Luettavissa: http://tulospalvelu.vaalit.fi/E-2015/fi/tulos_koko-maa.html. Luettu: 13.8.2016.

Reynolds, R. 2012. Data Visualization Design Process: A Primer. Luettavissa: https://constructive.co/insights/data-visualization-design-process-a-primer/. Luettu: 12.11.2016.

Rogers, S. 2011a. Data journalism at the Guardian: what is it and how we do it? Luettavissa: https://www.theguardian.com/news/datablog/2011/jul/28/data-journalism. Luettu: 8.9.2016.

Rogers, S. 2011b. Datajournalism broken down: what we do to the data before you see it. Luettavissa: https://www.theguardian.com/news/datablog/2011/apr/07/data-journalism-workflow. Luettu: 15.10.2016.

Sharma, N. 2015. 7 most common data visualization mistakes. Luettavissa: http://thenext-web.com/dd/2015/05/15/7-most-common-data-visualization-mistakes/. Luettu: 12.11.2016.

Skau, D. 2013. Why D3.js is So Great for Data Visualization? Luettavissa: http://www.scribblelive.com/blog/2013/01/29/why-d3-js-is-so-great-for-data-visualization/. Luettu: 18.7.2016.

Sun, R. 2014. What D3.js is Not. Luettavissa: http://ruoyusun.com/2014/05/26/what-d3js-is-not.html. Luettu: 16.10.2016.

Suomen Kuntaliitto. 2016a. Kuntapohjaiset aluejaot, kuntanumerot ja kuntien lukumäärät. Luettavissa: http://www.kunnat.net/fi/tietopankit/tilastot/aluejaot/Sivut/default.aspx. Luettu: 25.9.2016.

Suomen Kuntaliitto 2016b. Ajankohtaistilanne. Luettavissa: http://www.kunnat.net/fi/palvelualueet/kuntaliitokset/ajankohtaistilanne/Sivut/default.aspx. Luettu 25.9.2016.

Tableau. 2016. Buy Tableau. Luettavissa: https://buy.tableau.com/. Luettu: 21.8.2016.

Tebest. 2013. Suomen kuntien koordinaattitiedot. Luettavissa: http://datajournalismi.blogspot.fi/2013/03/suomen-kuntien-koordinaattitiedot.html. Luettu: 25.9.2016.

Tilastokeskus. 2011. Luettavissa: http://www.stat.fi/til/asvu/2016/02/asvu_2016_02_2016-08-08_tie_001_fi.html. Luettu: 16.8.2016.

Tulep, J. W. 22.4.2016. Data experience designer. How to design interesting data visualizations. TULP Interactive. Seminaariesitys. Helsinki.

Vihreä Lanka 2016a. Luettavissa: http://www.vihrealanka.fi/graafit/. Luettu 18.7.2016.

Vihreä Lanka 2016b. Luettavissa: http://www.vihrealanka.fi/mika. Luettu 18.7.2016.

Vihreä Lanka 2016c. Luettavissa: http://www.vihrealanka.fi/. Luettu: 16.8.2016.

Vihreä Lanka 2016d. Luettavissa: http://www.vihrealanka.fi/yhteystiedot. Luettu: 18.7.2016.

Vihreä Lanka 2016e. Luettavissa: http://www.vihrealanka.fi/uutiset-kotimaa/ylen-gallup-t%C3%A4r%C3%A4ytti-vihre%C3%A4t-kaikkien-aikojen-enn%C3%A4tykseen. Luettu: 16.8.2016.

W3 Schools 2016a. JavaScript Data Types. Luettavissa: http://www.w3schools.com/js/js_datatypes.asp. Luettu: 13.8.2016.

W3 Schools 2016b. HTML5 Introduction. Luettavissa: http://www.w3schools.com/html/html5_intro.asp. Luettu: 21.8.2016.

Wilson, M. 2015. What Killed The Infographic? Fast Company. Luettavissa: http://www.fastcodesign.com/3045291/what-killed-the-infographic. Luettu: 21.8.2016.

Yle. 2012. Kuntavaalit 2012. Luettavissa: http://vaalit.yle.fi/tulospalvelu/2012/kuntavaalit/puolueiden_kannatus.html. Luettu: 6.10.2016.

Yle. 2016. Luettavissa: http://yle.fi/uutiset/keskusta_palasi_niukasti_piikkipaikalle__vihreat_porskutti_ennatyslukemiin/8938031. Luettu: 13.8.2016.

Liitteet

Liite 1. Datavisualisointi puolueiden kannatuksen muutoksesta

Puoluekannatusgrafiikka osana Vihreän Langan verkkopalvelua.

Puoluekannatusgrafiikan koodi:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>TEST</title>
<style>
text {font-size:0.9em;}
</style>
</head>
<body>
<h1>Puoluekannatukset</h1>
<button class="vanha">Vaalit 4/2015</button>
<button class="uusi">Gallup 5/2016</button>
<hr />
<script src="d3-master/d3.min.js"></script>
<script>
```

```
var svg = d3.select("body").append("svg")
          .attr("viewBox", "0 0 500 400")
          .attr("preserveAspectRatio", "xMidYMid meet");

function vaihda(data, aika) {

          var color = d3.scale.category10();
          var palkit = svg.selectAll("rect").data(data);

          palkit.enter()
                              .append("rect")
                              .attr("x", 0)
                              .attr("y", function (d, i){ return i
* 50; })
                              .attr("width", 0)
                              .attr("height", 30)
                              .style("fill", function (d, i){ re-
turn color(i); })

          palkit
                              .transition().duration(1000)
                              .attr("x", 0)
                              .attr("y", function (d, i){ return i
* 50; })
                              .attr("width", function(d){
                                        var dAika = eval("d." +
aika);
                                        return dAika * 20;
                                        })
                              .attr("height", 30)
                              .style("fill", function (d, i){ re-
turn color(i); });

          var tekstit = svg.selectAll("text").data(data);

          tekstit.enter().append("text");

          tekstit
                              .attr("fill", "white")
```

```
                                .attr("x", 2)
                                .attr("y", function (d, i) { return
i * 50 + 20; })
                                .text(function (d) {
                                        var dAika = eval("d." +
aika);
                                        var etiketti = d.puolue + "
" + dAika;
                                        var etiketti = etiket-
ti.replace(/\./g, ',');
                                        return etiketti;
                                });
}

d3.csv("ontcase1_data.csv", function(data) {
          var tiedot = data;
          var aika = "uusi";
          vaihda(tiedot, aika);
});

d3.select(".vanha")
          .on("click", function(){
                    d3.csv("ontcase1_data.csv", function(data) {
                              var tiedot = data;
                              var aika = "vanha";
                              vaihda(tiedot, aika);
                    });
          });

d3.select(".uusi")
          .on("click", function(){
                    d3.csv("ontcase1_data.csv", function(data) {
                              var tiedot = data;
                              var aika = "uusi";
                              vaihda(tiedot, aika);
                    });
          });

</script>
```

```
</body>
</html>
```

Puoluekannatusgrafiikan datatiedoston sisältö:

```
puolue,vanha,uusi
Kokoomus,16.2,17.0
Keskusta,24.9,20.4
Vihreät,8.8,13.5
SDP,16.8,21.5
Perussuomalaiset,15.4,8.5
Vasemmisto,8.8,8.5
RKP,4.0,4.5
KD,3.3,3.6
```

Liite 2. Datavisualisointi ilmaston lämpenemisestä

Lämpötilavisualisoinnin täydellinen koodi:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>ONT2</title>
    <script src="d3v4/d3.min.js"></script>
          <style>

.lineHki {
  fill: none;
  stroke: blue;
  stroke-width: 0.8px;
}

.lineGlo {
  fill: none;
  stroke: red;
  stroke-width: 0.8px;
}
```

```
.lineZer {
  fill: none;
  stroke: black;
  stroke-width: 0.5px;
}

.lineSel {
  fill: none;
  stroke: black;
  stroke-width: 0.5px;
}

.overlay {
  fill: none;
  pointer-events: all;
}

          </style>
</head>

<body>
          <script>

var svg = d3.select("body").append("svg")
          .attr("viewBox", "0 0 500 250")
          .attr("preserveAspectRatio", "xMidYMid meet");

d3.csv("ontcase2_data.csv", type, function(error, data) {

          if (error) throw error;

          var xScale = d3.scaleLinear()
                    .domain(d3.extent(data, function(d) { return
d.year; }))
                    .range([10, 450]);

          var yScale = d3.scaleLinear()
                    .domain(d3.extent(data, function(d) { return
d.hki; }))
```

```
          .range([210, 10]);

var line1 = d3.line()
          .x(function(d) { return xScale(d.year); })
          .y(function(d) { return yScale(d.hki); });

var line2 = d3.line()
          .x(function(d) { return xScale(d.year); })
          .y(function(d) { return yScale(d.glo); });

var line3 = d3.line()
          .x(function(d) { return xScale(d.year); })
          .y(function(d) { return yScale(0); });

svg.append("path")
          .datum(data)
          .attr("class", "lineHki")
          .attr("d", line1);

svg.append("path")
          .datum(data)
          .attr("class", "lineGlo")
          .attr("d", line2);

svg.append("path")
          .datum(data)
          .attr("class", "lineZer")
          .attr("d", line3);

svg.append("line")
          .style("stroke", "blue")
          .style("stroke-dasharray", ("1, 1"))
          .attr("x1", xScale(1900))
          .attr("y1", yScale(-0.517))
          .attr("x2", xScale(1929))
          .attr("y2", yScale(-0.517));

svg.append("line")
          .style("stroke", "blue")
```

```
            .style("stroke-dasharray", ("1, 1"))
            .attr("x1", xScale(1986))
            .attr("y1", yScale(1.023))
            .attr("x2", xScale(2015))
            .attr("y2", yScale(1.023));

svg.append("line")
            .style("stroke", "red")
            .style("stroke-dasharray", ("1, 1"))
            .attr("x1", xScale(1900))
            .attr("y1", yScale(-0.252))
            .attr("x2", xScale(1929))
            .attr("y2", yScale(-0.252));

svg.append("line")
            .style("stroke", "red")
            .style("stroke-dasharray", ("1, 1"))
            .attr("x1", xScale(1986))
            .attr("y1", yScale(0.514))
            .attr("x2", xScale(2015))
            .attr("y2", yScale(0.514));

var yearLabel = svg.append("text")
            .attr("font-size","24px")
            .attr("x", 10)
            .attr("y", 20)
            .style("display", "none")
            .text("testi");

var hkiLabel = svg.append("text")
            .attr("font-size","12px")
            .style("fill", "blue")
            .attr("x", 0)
            .attr("y", 215)
            .style("display", "none")
            .text("testi");

var gloLabel = svg.append("text")
            .attr("font-size","12px")
```

```
            .style("fill", "red")
            .attr("x", 0)
            .attr("y", 230)
            .style("display", "none")
            .text("testi");

var valitsin = svg.append("line")
            .style("stroke", "black")
            .style("stroke-dasharray", ("5, 5"))
            .style("display", "none")
            .attr("class", "lineSel")
            .attr("x1", 0)
            .attr("y1", 0)
            .attr("x2", 0)
            .attr("y2", 210);

svg.append("text")
            .attr("font-size","8px")
            .style("fill", "blue")
            .attr("x", 455)
            .attr("y", yScale(data[115]["hki"])+2)
            .text("Helsinki");

svg.append("text")
            .attr("font-size","8px")
            .style("fill", "red")
            .attr("x", 455)
            .attr("y", yScale(data[115]["glo"])+2)
            .text("Maailma");

svg.append("text")
            .attr("font-size","8px")
            .attr("x", 455)
            .attr("y", 115)
            .text("keskiarvo");

svg.append("rect")
            .attr("class", "overlay")
            .attr("x", 0)
```

```
                    .attr("y", 0)
                    .attr("width", 450)
                    .attr("height", 210)
                    .on("mouseover", function() {
                              yearLabel.style("display", null);
                              hkiLabel.style("display", null);
                              gloLabel.style("display", null);
                              valitsin.style("display", null);
                    })
                    .on("mousemove", mousemove);

function mousemove() {
          var sijainti = Math.floor(xScale.invert(d3.mouse(this)
[0])-1899.5);
          var hkiLuku =
((+data[sijainti].hki+5.167).toFixed(1)+"°C").replace(/\./g, ',');
          var gloLuku =
((+data[sijainti].glo+13.9).toFixed(1)+"°C").replace(/\./g, ',');
          yearLabel.text(data[sijainti].year);
          hkiLabel
                    .text(hkiLuku)
                    .attr("x", xScale(data[sijainti].year)-10);
          gloLabel
                    .text(gloLuku)
                    .attr("x", xScale(data[sijainti].year)-10);
          valitsin
                    .attr("x1", xScale(data[sijainti].year))
                    .attr("x2", xScale(data[sijainti].year));
}

});

function type(d) {
          d.year = +d.year;
          d.hki = +d.hki;
          d.glo = +d.glo;
          return d;
};
```

```
        </script>
</body>
</html>
```

Havainnollistava katkelma tietosisällön alusta:

```
year,hki,glo
1900,-1.467,-0.0725
1901,-0.167,-0.1464
1902,-2.467,-0.2545
1903,0.533,-0.3429
```

Liite 3. Datavisualisointi kuntavaalituloksesta

Kuntavaalivisualisoinnin täydellinen koodi:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>Kuntavaalit</title>
<style>
        .nimiLista {
                pointer-events: none;
                font-family: Arial;
                font-size: 0.8em;
        }

        .kuntaNimi {
                pointer-events: none;
                font-family: Arial;
                font-size: 0.8em;
        }

        .boldattu {
                font-weight: 700;
        }

        .narrator {
```

```
                font-family: Arial;
                font-size: 1.0em;
        }

        .otsake {
                font-family: Arial;
                font-size: 1.3em;
        }

        .numba {
                font-family: Arial;
                font-size: 4.3em;
                font-weight: 700;
        }

</style>
</head>
<body>
<script src="d3v4/d3.min.js"></script>
<script>

var svg = d3.select("body").append("svg")
        .attr("viewBox", "0 0 500 500")
        .attr("preserveAspectRatio", "xMidYMid meet");

var x = d3.scaleLinear()
        .domain([20, 37])
        .range([0, 500]);

var y = d3.scaleLinear()
        .domain([59, 70])
        .range([500, 0]);

var data = [];
var pudokkaat = 0;
var jatkajat = 0;
var uudet = 0;

function aloitus(data) {
```

```
          svg.selectAll(".dot")
                    .transition()
                              .duration(1500)
                              .style("fill", "darkgreen")
                              .attr("r", function(d) {
                                        if (d.status == 2) {
                                                  return 0;
                                        } else {
                                                  return 3;
                                        };
                              })
                              .attr("transform", function() {
                                        var angle = (Math.random()
* Math.PI * 2);
                                        var radius = (15 +
Math.random() * 225);
                                        var deltax = 100;
                                        var deltay =  -100;
                                        if (angle < Math.PI/2) {
                                                  var deltax =
Math.sin(angle)*radius;
                                                  var deltay =
-Math.cos(angle)*radius;
                                        } else if (angle < Math.PI)
{
                                                  angle = angle -
Math.PI/2;
                                                  var deltax =
Math.sin(angle)*radius;
                                                  var deltay =
Math.cos(angle)*radius;
                                        } else if (angle <
Math.PI*1.5) {
                                                  angle = angle -
Math.PI;
                                                  var deltax =
-Math.sin(angle)*radius;
                                                  var deltay =
Math.cos(angle)*radius;
```

```
                                        } else {
                                                angle = angle -
Math.PI*1.5
                                                var deltax =
-Math.sin(angle)*radius;
                                                var deltay =
-Math.cos(angle)*radius;
                                        }
                                        var tulos = deltax + "," +
deltay;
                                        return "translate(" + tulos
+ ")";
                                });
          svg.append("rect")
                    .attr("class", "narrator")
                    .style("fill", "white")
                    .transition(1000)
                              .attr("x", 0)
                              .attr("y", 200)
                              .attr("width", 500)
                              .attr("height", 100)
                              .style("opacity", 0.8);
          svg.append("text")
                    .style("opacity", 0)
                    .attr("class", "narrator")
                    .style("fill", "darkgreen")
                    .attr("x", 80)
                    .attr("y", 240)
                    .transition(1000)
                              .style("opacity", 1)
                              .text(function() {
                                        var luku = pudokkaat + jat-
kajat;
                                        return "Vaaleissa 2012 va-
littiin " + luku + " vihreää valtuutettua.";
                              });
};

function jatko(data) {
```

```
		svg.selectAll(".status0")
				.transition()
					.duration(2000)
					.style("fill", "white")
					.attr("cy", 500);
		svg.selectAll(".status2")
				.transition()
					.duration(2000)
					.style("fill", "limegreen")
					.attr("r", 6);
		svg.append("text")
				.style("opacity", 0)
				.attr("class","narrator")
				.style("fill", "darkgreen")
				.attr("x", 30)
				.attr("y", 270)
				.transition(2500)
					.style("opacity", 1)
					.text(function() {
						var lause = "Heistä " + pu-
dokkaat + " luopui paikastaan tai putosi, mutta uusia valittiin "
+ uudet + ".";
						return lause;
					});
}

function kartta(data) {
		d3.selectAll(".narrator").remove();

		svg.append("text")
				.style("opacity", 0)
				.text("Suomen")
				.attr("class", "otsake")
				.attr("x", 340)
				.attr("y", 20)
				.style("fill", "black")
				.transition()
					.duration(1000)
					.style("opacity", 1);
```

```
svg.append("text")
          .style("opacity", 0)
          .text(function() {
                    return jatkajat + uudet;
          })
          .attr("class", "numba")
          .attr("x", 338)
          .attr("y", 77)
          .style("fill", "limegreen")
          .transition()
                    .duration(1400)
                    .style("opacity", 1);
svg.append("text")
          .style("opacity", 0)
          .text("vihreää")
          .attr("class", "otsake")
          .attr("x", 340)
          .attr("y", 100)
          .style("fill", "black")
          .transition()
                    .duration(1800)
                    .style("opacity", 1);
svg.append("text")
          .style("opacity", 0)
          .text("valtuutettua")
          .attr("class", "otsake")
          .attr("x", 340)
          .attr("y", 120)
          .style("fill", "black")
          .transition()
                    .duration(2200)
                    .style("opacity", 1);

svg.selectAll(".dot")
          .data(data)
          .transition()
                    .duration(1000)
                    .attr("transform", "translate(0,0)")
```

```
                                .attr("cx", function(d) { return
x(d.lon); })
                                .attr("cy", function(d) { return
y(d.lat); })
                                .attr("r", 0);

          var kuntaData = [];

          d3.json("oikeakuntadata.json", function(error, json) {
                    if (error) throw error;

                    json.tulokset.forEach(function(d) {
                              var kunta = {name: d.name, totvotes:
d.totvotes, lon: d.lon, lat: d.lat};
                              var nimet = [];
                              d.children.forEach(function(d) {
                                        if (d.status != 0) {
                                                  nimet.push(d.na-
me);
                                        };
                              });
                              kunta.nimet = nimet;
                              kunta.kuntaPaikat = nimet.length;
                              kuntaData.push(kunta);
                    });
                    kaupungit(kuntaData);
          });

          function kaupungit(kuntaData) {
                    svg.selectAll(".city")
                    .data(kuntaData)
                    .enter().append("circle")
                              .attr("class", "city")
                              .attr("cx", function(d) { return
x(d.lon); })
                              .attr("cy", function(d) { return
y(d.lat); })
                              .attr("r", function(d) { return
3*Math.sqrt(d.kuntaPaikat/Math.PI); })
```

```
                        .style("fill", "limegreen")
                .on("mouseover", function(d) {
                        d3.select(this)
                                .transition()
                                        .duration(200)
                                        .attr("r", 50);
                        svg.append("text")
                                .text(d.name)
                                .attr("class", "kuntaNimi")
                                .attr("x", x(d.lon)-d.na-
me.length*3)
                                .attr("y", y(d.lat)+5)
                                .style("fill", "black");
                        svg.append("text")
                                .text("Valitut:")
                                .attr("class", "nimiLista
boldattu")
                                .attr("x", 340)
                                .attr("y", 150)
                                .style("fill", "black");
                        d.nimet.forEach(function(d, i) {
                                svg.append("text")
                                        .text(d)
                                        .attr("class",
"nimiLista")
                                        .attr("x", 340)
                                        .attr("y",
166+16*i)
                                        .style("fill",
"black");
                        });

                })
                .on("mouseout", function(d) {
                        d3.select(this).transition().dura-
tion(400).attr("r", function(d) { return 2*Math.sqrt(d.kuntaPai-
kat/Math.PI); });
                        d3.selectAll(".nimiLista").remove();
                        d3.selectAll(".kuntaNimi").remove();
```

```
                    });
          };

}

d3.json("oikeakuntadata.json", function(error, json) {
          if (error) throw error;
          json.tulokset.forEach(function(d) {
                    var kuntaNimi = d.name;
                    var kuntaVotes = d.totvotes;
                    var kuntaLon = d.lon;
                    var kuntaLat = d.lat;
                    d.children.forEach(function(d) {
                              if (d.status == 0) {
                                        pudokkaat = pudokkaat + 1;
                              } else if (d.status == 1) {
                                        jatkajat = jatkajat + 1;
                              } else {
                                        uudet = uudet + 1;
                              };
                              d.kname = kuntaNimi;
                              d.totvotes = kuntaVotes;
                              d.lon = kuntaLon;
                              d.lat = kuntaLat;
                              data.push(d);
                    });
          });

          svg.selectAll(".dot")
                    .data(data)
                    .enter().append("circle")
                              .attr("class", function(d) {
                                        var luokka = "dot status" +
d.status;
                                        return luokka;
                              })
                              .attr("cx", 250)
                              .attr("cy", 250)
                              .attr("r", 0)
```

```
                                .style("fill", "darkgreen");
          aloitus(data);
          setTimeout(function() {
                    jatko(data);
          }, 3000);
          setTimeout(function() {
                    kartta(data);
          }, 9000);

});

</script>
</body>
</html>
```

Havainnollistava katkelma tietosisällön alusta:

```
{"tulokset":[
   {
   "name": "Akaa",
   "totvotes": 7590,
   "lat": 61.168006,
   "lon": 23.868675,
   "children": [
      {"name": "Heli Piirainen", "status": 1, "pervotes": 135}
   ]},
   {
   "name": "Asikkala",
   "totvotes": 4426,
   "lat": 61.21606,
   "lon": 25.500376,
   "children": [
      {"name": "Tuula Wikström", "status": 1, "pervotes": 63}
]},
```

Liite 4. Yleisimmät kaaviotyypit

Andrew Abela julkaisi vuonna 2006 blogissaan kaavion, jota voi käyttää apunaan oikeaa esitystapa valittaessa. (Abela 2006.)

Chart Suggestions—A Thought-Starter