Tallinna Ülikool

Digitehnoloogiate Instituut

Informaatika õppekava

Andmete visualiseerimine D3 abil, õppematerjal

Bakalaureusetöö

Autor: Matthias Johann Kurs

Juhendaja: Jaagup Kippar

Autor: ....,, ...,, 2018
Juhendaja: ....,, ...,, 2018
Instituudi direktor: ....,, ...,, 2018

Tallinn 2018

Autorideklaratsioon

Deklareerin, et käesolev bakalaureusetöö on minu töö tulemus ja seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud.

.....	.....
(kuupäev)	(autor)

Sisukord

Sissejuhatus

Andmete hulk kasvab iga päevaga aina suuremaks ning neid andmeid töödeldakse iga päev, et kätte saada ainult kõige olulisim info. Üheks viisiks, kuidas andmeid kätte saada ja välja lugeda, oleks andmete visualiseerimise abil. Andmete visualiseerimine võimaldab vajalikke andmeid kuvada pildina ning anda infot edasi mugavamalt ja arusaadavamalt. Käesoleva bakalaureusetöö käigus tutvustab autor andmete visualiseerimise vahendit nimega D3.js (edaspidi D3).

Bakalaureusetöö eesmärgiks on anda ülevaade D3'e teegi kohta ning luua D3'le õppematerjal, mida saaks kasutada andmete visualiseerimisel. Õppematerjal sisaldab erinevaid D3 funktsionaalsusi, mille abil on võimalik teostada erinevaid andmete visualiseerimisi. Lisaks õppematerjalile on autor toonud näiteid teistest erinevatest andmete visualiseerimiste vahenditest ning teiste poolt loodud õppematerjale D3 kohta.

Autor valis teema, kuna autor on eelnevalt D3'ga kokku puutunud ning samuti puudub ka eestikeelne õppematerjal D3 kohta. Autor leidis ka, et enamik ingliskeelseid materjale on loodud D3 vanema versiooni peale, mistõttu ei leidu palju õppematerjale, mis oleksid ajakohased.

Töö on koosneb neljast peatükist. Esimeses peatükis antakse ülevaade D3 ja tema võimaluste kohta. Teises peatükis toob autor välja D3'le alternatiive, mida saaks kasutada andmete visualiseerimiseks. Kolmandas peatükis uurib autor erinevaid teiste poolt loodud õppematerjale, et tutvustada neid. Neljas peatükk käsitleb õppematerjali loomiseks vajalikke samme ning õppematerjali testimist, viies läbi sissejuhatava tunni Tallinna Ülikooli Informaatika õpilastele.

Mõisted

D3.js (Data-Driven Documents või lihtsalt D3) – JavaScripti teek, mille abil saab andmeid visualiseerida veebilehitsejas.

HTML (HyperText Markup Language) – keel, millega märgendatakse veebilehti.

CSS (Cascading Style Sheets) – küljenduskeel, milles märgitakse üles peamiselt veebilehtede kujundust.

JavaScript – objektorienteeritud programmeerimiskeel, mida kasutatakse peamiselt veebilehtede skriptimiseks.

SVG (Scalable Vector Graphics või skaleeruv vektorgraafika) – on XML'il põhinev keele spetsifikatsioonide hulk, mis kirjeldab nii staatilist kui ka dünaamilist kahemõõtmelist vektorgraafikat.

SMIL (Synchronized Multimedia Integration Language) – XML'il põhinev keel, mis kirjeldab multimeedia esitamist.

Kanvas (Canvas) – HTML'i element, mis võimaldab joonistada graafikat ning luua animatsioone.

Plugin – lisakomponent, mis täiustab olemasolevat programmi lisades programmile uut funktsionaalsust.

Teek (ingl *library*) – kollektsioon erinevaid funktsioone, klasse ja muid komponente, mis on mõeldud korduvkasutuseks programmides.

1 D3.js ülevaade

D3.js (või lihtsalt D3 ehk Data-Driven Documents) on JavaScripti teek andmete visualiseerimiseks veebilehitsejas. Teek kasutab SVG, HTML5 ja CSS standardeid, et luua dünaamilisi ja interaktiivseid visualiseeringuid. (Wikipedia, kuupäev puudub)

D3 esimene versioon ilmus välja 2011. aastal ning arendajateks olid Mike Bostock, Jeff Heer ja Vadim Ogievetsky. D3'e eelkäijaks on Provotis, mis genereerib SVG graafikat andmetele vastavalt. Provotis'e arendamine lõpetati 2011. aastal, sest Provotis'e arendajad otsustasid keskenduda rohkem D3 arendamisele. (Stanford Visualization Group, 2009)

D3 sisaldab endas erinevaid funktsioone, mis võimaldavad valida DOM elemente, luua SVG elemente ning tekitada neile erinevaid efekte. D3 abil saab ka andmeid siduda SVG elementide külge, et genereerida erinevaid jooniseid. D3 on avatud lähtekoodiga ning tasuta, mis tähendab, et igaüks võib panustada D3 arendamisele kaasa. (Wikipedia, kuupäev puudub)

D3 suurimateks eelisteks on tema paindlikkus ehk arendaja saab kasutada D3'e koos teiste veebitehnoloogiatega ilma probleemideta. D3 on samuti ka dünaamiline ehk arendaja saab luua mistahes animatsioone ja interaktsioone oma joonistele. Täielik kontroll on ka DOM elementide üle, mis annab rohkem võimalusi oma joonise manipuleerimiseks. D3 suudab töödelda suurtes hulkades andmeid ilma, et koormaks hulgaliselt ressursse või jõudlust. Selleks, et säästa rohkem resursse, ei pea kasutama tervet teeki, vaid võib kasutada neid funktsioone, mis kasutajal vaja läheb. Lisaks on D3 jaoks olemas palju pluginaid, et lihtsustada andmete visualiseerimist ning leidub hulgliselt teiste poolt tehtud näiteid. D3 on ka avatud lähtekoodiga (ingl *open source*) ja kõigile tasuta. (Cabot Technology Solution, 2017)

D3 puudusteks on see, et tal puuduvad valmistehtud funktsioonid mis genereeriks automaatselt joonised vastavalt andmetele. D3 nõuab ka vähemalt algteadmisi HTML/CSS ja JavaScriptist, et luua lihtsamaid joonised, keerulisemate jooniste puhul peab rohkem süvenema D3 dokumentatsiooni või omama rohkem teadmisi JavaScripti kohta. D3 puuduseks on ka see, et ta töötab ainult nendel brauseritel, mis on nn. modernsed ja toetavad SVG'd ja CSS siirdeid (ingl *CSS transistions*). Näiteks

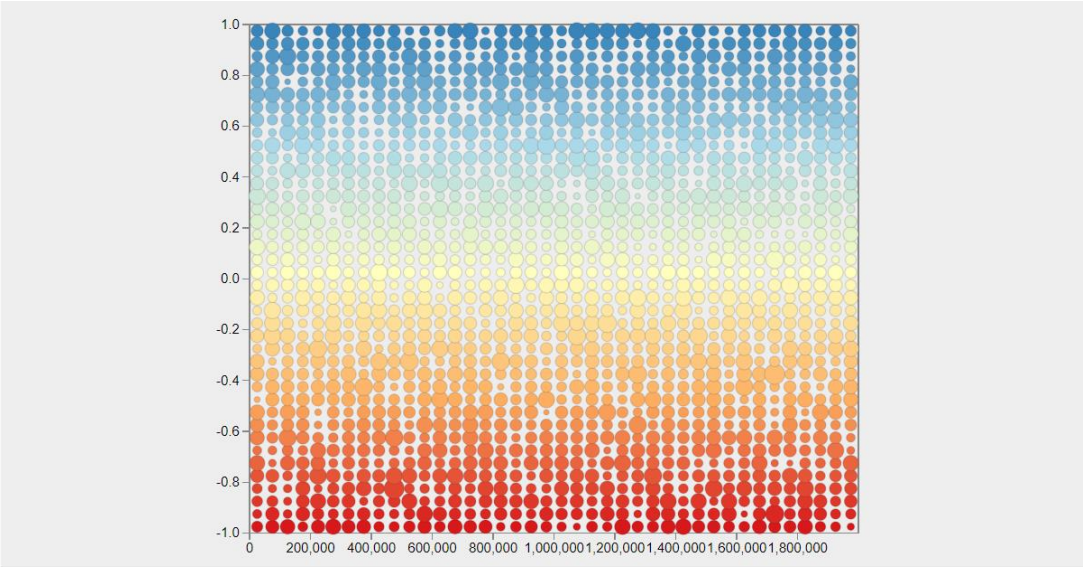
ei tööta D3 veebibrauseri Internet Explorer versiooni 8 peal ning tema vanematel versioonidel. (Selikoff, 2014)

1.1 D3 Pluginad

D3'l leidub palju erinevaid pluginaid, mis lihtsustavad andmete visualiseerimist. D3 pluginad on loodud vabatahtlike poolt ning D3 autor on loonud ka juhise kuidas luua ise D3 pluginaid. Kõik pluginad on saadaval aadressil https://github.com/d3/d3/wiki/Plugins

Näited erinevatest pluginatest:

- 2-dimensionaalne histogramm (Joonis 1) - https://github.com/herkulano/d3-hist2d
- 3D visualiseerimine - https://github.com/Niekes/d3-3d
- Ristküliku joonistamine - https://github.com/vijithassar/d3-rect
- D3 Legendid (jooniste seletused) - https://github.com/susielu/d3-legend



Joonis 1. 2D Histogramm (Campos, 2017)

2 Alternatiivid D3'le

Lisaks D3'le leidub palju teisi JavaScripti teeke, mis suudavad visualiseerida andmeid jooniste või diagrammide kaudu. Järgnevates alapeatükkides tuuakse näited erinevatest JavaScripti teekidest, mida saaks kasutada D3'e asemel. (Kesavan, 2017)

2.1 Chart.js

Chart.js on avatud lähtekoodiga JavaScripti teek, mis keskendub diagrammide loomisele. Chart.js kasutab HTML5 canvase elementi, et luua diagramme lihtsasti ja kiiresti. Hetkel toetab Chart.js kaheksat erinevat diagrammi tüüpi. Erinevaid diagramme on võimalik ka animeerida ja omavahel kombineerida. (Chart.js, kuupäev puudub)

Chart.js toetab järgmisi diagrammide tüüpe:

- Tulpdiagrammid
- Joondiagrammid
- Ala graafik (ingl *area chart*)
- Hajuvusdiagrammid
- Sektordiagrammid
- Radardiagrammid
- Polaar aladiagrammid (ingl *polar area chart*)
- Mullidiagrammid (ingl *bubble chart*)

Chart.js on võrreldes D3'ga palju lihtsam, mistõttu on jooniseid võimalik palju kiiremini luua. Chart.js abil saab ilma vaevata luua erinevaid diagramme, et oma andmeid visualiseerida. Chart.js ei nõua isegi JavaScripti teadmisi, et luua lihtsaid diagramme, mistõttu isegi algajad suudavad ilma vaevata omale diagramme luua.

Chart.js võimaldab luua diagramme lihtsalt ja kiiresti, kuid võrreldes D3'ga on tal võimalusi palju vähem. Näiteks on Chart.js limiteeritud ainult 8 erineva diagrammi tüübiga, D3 võimaldab palju rohkem vastavalt oskuste tasemele. D3 toetab ka andmete manipuleerimist ning sisaldab rohkem erinevaid interaktiivsuse võimalusi. (Slant, kuupäev puudub)

Chart.js tuleb kasuks siis kui on vaja kiiresti ja ilma vaevata luua diagramme. Keerulisemate ja rohkem interaktiivsete graafide puhul tuleks kasutada D3'e.

2.2 Chartist.js

Chartist.js on avatud lähtekoodiga JavaScripti teek, mida kasutatakse ka diagrammide loomiseks. Sarnaselt Chart.js'ile võimaldab Chartist.js luua diagramme lihtsalt ja kiiresti. Chartist.js kasutab jooniste tegemiseks SVG'd ning animatsioone saab luua CSS'i kui ka SMIL'i (W3C, 2008) abil. Chartist.js on ka kergekaaluline ehk ta võtab ainult 10KB ruumi, mis on väga kasulik kui on vaja luua veebisaite, mis reageeriksid kiiresti. Leidub ka erinevaid pluginaid, et oma diagramme täiustada. (Chartist.js, kuupäev puudub)

Chartist.js abil on võimalik luua järgnevaid diagramme:

- Joondiagrammid
- Sektordiagrammid
- Tulpdiagrammid

Chartist.js eelis D3'e ees on tema lihtsus, kõike diagramme on võimalik luua kiiresti ja ilma vaevata. Samuti on olemas ka dokumentatsioon, mis on lihtsasti loetav ja arusaadav kõigile.

Sarnaselt Chart.js'le jääb Chartist.js oma funktsioonalsuse poolest D3'le alla. Näiteks puuduvad sisseehitatud interaktsiooni võimalused või siis peab kasutama pluginaid, et luua interaktsiooni oma diagrammide jaoks. Chartist.js on mõeldud ainult andmete kuvamiseks diagrammide ning tema animatsioonide kaudu. Võrreldes D3'ga on näiteid palju vähem. (Slant, kuupäev puudub)

2.3 Google Charts

Google Charts on Google'i poolt loodud veebiteenus, mis võimaldab luua erinevad diagramme ja joonised. Google Charts kasutab JavaScripti, HTML'i ja SVG'd, et luua interaktiivseid ja dünaamilisi jooniseid. Jooniseid on võimalik ka kohendada vastavalt vajadustele ning on võimalik luua ka erinevaid animatsioone. (Google, kuupäev puudub)

Google Charts toetab palju erinevaid andmete visualiseerimise võimalusi, nagu näiteks:

- Histogrammid
- Sektordiagrammid (seal hulgas sõõrikugraafid)
- Joondiagrammid
- Mullidiagrammid
- Tabelid
- Ja palju muud.

Google Charts eelis nagu eelnevatelgi teekidel on tema lihtsus. Piisab andmete sisestamisest ja seejärel vastava graafifunktsiooni kasutamisest. Google Charts kasutamiseks ei pea tundma JavaScripti, mis muudab andmete visualiseerimise kõigile lihtsaks.

Google Charts võrreldes D3'ga ei ole võimeline töötlema andmeid suurtes hulkades. Google Charts'i puhul on interaktsiooni võimalused piiratud vastavalt joonise tüübile. D3'e puhul saad luua oma jooniseid vastavalt vajadusele ning lisada mistahes interaktsiooni võimalusi. (Sumanth, 2014)

3 Olemasolevad õppematerjalid

Selles peatükis analüüsitakse erinevaid olemasolevaid D3 õppematerjale, mis on teiste autorite poolt loodud. Internetis leidub palju erinevaid D3 näited ja õppematerjale, mistõttu otsustas autor välja valida 2 e-õppematerjali ning 2 e-raamatut, millest üks on kättesaadav ka tavalise raamatuna.

3.1 D3.js Tutorial (TutorialsTeacher)

Tüüp	E-õppematerjal
Sihtrühm	Algajad, edasijõudnud
Aadress	http://www.tutorialsteacher.com/d3js
Autorid	TutorialsTeacher

Tabel 1. D3.js Tutorial (TutorialsTeacher)

Õppematerjal on loodud veebilehena ehk tegemist on e-õppematerjaliga. Õppematerjali sihtrühmaks on algajad, kes soovivad D3 kohta rohkem teada. Autorite sõnul on materjali sihtrühmaks ka professionaalid, kes tahavad oma oskusi arendada D3 alal. Autorid on märkinud ka, et lugejad peaksid omama algteadmisi HTML'i, CSS'i ja JavaScripti kohta. (TutorialsTeacher, kuupäev puudub)

Materjal on jagatud erinevateks peatükkideks, kus iga peatükk tutvustab uut funktsioonaalsust või on täiendavaks materjaliks eelnevatele peatükkidele. Autori arvates on õppematerjal hea alustuskoht algajatele, kes pole ennem D3'ga kokku puutunud. Autori enda õppematerjal põhineb enamjaolt antud õppematerjali põhjal. Autor valis selle õppematerjali oma töö aluseks kuna selles õppematerjalis antakse ainult olulisim info edasi, mis on D3 sissejuhtavaks õppematerjaliks ideaalne.

Õppematerjal on oma sisult lühike, võetakse läbi ainult need teemad, mis on lihtsamate jooniste loomiseks vajalikud. Autori arvates võiks olla rohkem näiteid erinevate jooniste kohta, sest antud õppematerjal katab ainult kolme erinevat diagrammi tüüpi. Õppematerjal on samuti loodud D3 versiooniga 4, mistõttu osa infot pole seal uuendatud.

3.2 D3.js Tutorial (TutorialsPoint)

Tüüp	E-õppematerjal
Sihtrühm	Algajad, edasijõudnud
Aadress	https://www.tutorialspoint.com/d3js
Autorid	TutorialsPoint

Tabel 2. D3.js Tutorial (TutorialsPoint)

Õppematerjal käsitleb erinevaid teemasid D3'e kohta ning toob teemades välja erinevaid näited. Autor leiab, et see materjal sobib nii algajatele kui ka edasijõudnutele. Autori arvates on õppematerjali kerge lugeda kuna igat teemat on põhjalikult käsitletud ning koodinäited on sobivad.

Sarnaselt TutorialsTeacher õppematerjalile on näited jooniste kohta vähe. Eelkõige keskendub õppematerjal D3'e erinevate funktsioonide tutvustamisele ning nende selgitamisele. Õppematerjal on loodud D3 verisoonile 4, mis tähendab osa infot pole enam korrektne. (TutorialsPoint, kuupäev puudub)

3.3 Interactive Data Visualization for the Web, 2nd Ed.

Tüüp	Raamat, e-õppematerjal
Sihtrühm	Algajad, edasijõudnud
Aadress	http://alignedleft.com/work/d3-book-2e
Autorid	Scott Murray, Kirjasta: O'Reilly Media

Tabel 3. Interactive Data Visualization for the Web, 2nd Ed.

Tegemist on 457-leheküljelise raamatuga, mis on välja antud 2017.aastal. Raamat ise on tasuline, kuid registreerides end aadressil https://www.safaribooksonline.com on võimalik raamatut 10 päeva tasuta lugeda. Kõik koodinäited on kõigile kättesaadavad aadressil https://github.com/alignedleft/d3-book/releases juhul kui ole soovi raamatut osta ning on huvi ainult näidete vastu.

Raamat koosneb 16-st peatükist, mis omakorda on jagatud väiksemateks osadeks, et lugejal oleks võimalikult kerge materjali jälgida. Raamatu autor on õppematerjali üles ehitanud nii, et isegi nendel kellel puudub igasugune kogemus programmeerimise alalt on võimeline materjalist aru saama ja kaasa teha. Õppematerjal koosneb ka üle 130 erinevast koodinäitest, mis tulevad koos selgitustega. Raamatu lõpus on välja toodud ka erinevaid juhtumuuringuid (ingl *case studies*) ning on toodud välja ka teiste poolt loodud näiteid D3'e kohta. (Murray, 2017)

3.4 D3 Tips and Tricks v3.x

Tüüp	E-raamat
Sihtrühm	Algajad
Aadress	https://leanpub.com/D3-Tips-and-Tricks
Autorid	Malcolm Maclean

Tabel 4. D3 Tips and Tricks v3.x

Raamat on eelkõige mõeldud neile, kes pole varem D3'ga kokku puutunud ning soovivad sellest rohkem teada. Raamat ise koosneb 600-st leheküljest, mis katab erinevaid teemasid D3'e kohta. Raamat on täiesti tasuta ning sisaldab erinevaid koodinäiteid, et lugeja saaks kaasa teha.

Antud raamatu eeliseks on see, et raamatu autor ei kasuta oma näidetes ainult D3'e vaid kasutab ka teisi erinevaid tehnoloogiaid (nagu näiteks BootStrap, leaflet.js, jne), et tutvustada veel rohkem võimalusi D3'e kasutamiseks.

Puuduseks on hetkel see, et raamat on kirjutatud D3'e versioonile 3, mis on aegunud. Raamatu autor on hetkel kirjutamas uut õppematerjali uuemale D3 versioonile, kuid praeguseks ajaks pole see valmis saanud. (Maclean, viimati uuendatud 2018.aastal)

4 Õppematerjali koostamine

Bakalaureusetöö eesmärgiks on luua õppematerjal, mis annaks õppurile algteadmised D3'e kohta ning nende teadmiste rakendamine ülesannetes. Õppematerjali põhieesmärgiks on tutvustada D3'e funktsionaalsust ning visualiseerida andmeid lihtsate graafide abil. Eesmärgi täitmiseks on autor võtnud eeskujuks erinevaid teiste poolt loodud õppematerjale (enamik õppematerjalist on loodud http://www.tutorialsteacher.com/d3js/ materjali põhjal) ja samuti on võetud aluseks ka D3'e dokumentatsioon. Õppematerjali läbimiseks peaks õppuril olema eelnevaid teadmisi HTML'i, CSS'i ja JavaScripti kohta. Kasuks tulevad ka teadmised SVG kohta.

Bakalaureusetöös valminud ei käsitle D3'e kõiki võimalusi vaid pigem keskendub rohkem põhiteadmistele, mis on vajalikud D3'e kasutamiseks. Õppematerjalis käsitletakse eelkõige lihtsamate jooniste loomist, et materjali lugejal tekiks arusaam, kuidas neid luua.

Õppematerjal koosneb 12 peatükist, mis kirjeldavad mingit D3 funktsioonaalsust või demonstreerib erinevaid funktsioone koos. Osade peatükkide lõpus on olemas ka ülesanded, mis testivad õppuri teadmisi.

Autor otsustas ka õppematerjali testimiseks läbi viia ka tunni, et saada tagasisidet kaasõpilaste käest. Kõik õppematerjaliga seostuvad koodinäited ja ülesannete lahendused on leitavad aadressilt: http://www.tlu.ee/~mats96/d3/

4.1 Õppematerjali vajadus

Õppematerjali vajadusest lähtus autor eelkõige juhendaja soovist luua õppematerjal D3'e jaoks, et tuvustada erinevaid vahendeid andmete visualiseerimiseks. Hetkel on Tallinna Ülikoolis peamisteks andmete visualiseerimise vahendiks R-keel, mistõttu otsustas autor luua õppematerjali D3'e kohta, mida saaks rakendada õppekavasse.

Hetkel puuduvad ka eestikeelsed materjalid D3 kohta ning enamik ingliskeelseid materjale on loodud D3'e versioonile 3 või versioonile 4.

4.2 Sihtrühm

Selles peatükis kirjeldab autor kellele jaoks õppematerjal loodud on ning vajalikud oskused õppematerjali läbimiseks.

Peamised sihtrühmad:

- Õpilased
- Andmete visualiseerimise huvilistele
- D3 algajatele, või neile kellel on huvi D3 kohta

Vajalikud oskused:

- HTML – D3 abil muudetakse HTML elemente, et andmeid visualiseerida või neid kuvada
- CSS – kujundus, eelkõige kasutatakse CSS'i, et kujundada graafe vastavalt vajadusele
- JavaScript – andmete visualiseerimine teostatakse erinevate funktsioonide abil, mistõttu on vaja teada kuidas neid luua või rakendada.
- SVG – kujundite/graafide joonistamine. D3 kasutab SVG'd, et luua jooniseid.

Kasuks tuleb ka teadmised teiste JavaScripti teekide kohta, sest suur tõenäusus on, et neid saab rakendada koos D3'ga.

4.3 Õppematerjali ülesehitus

Selles alapeatükis annab autor ülevaate õppematerjali ülesehitustest. Õppematerjali alguses on väike sissejuhatus D3'e kohta ning peatükkide kohta. Õppematerjali sissejuhatuses on toodud välja ka link, kus on olemas kõik õppematerjaliga seonduvad materjalid nagu näiteks koodinäited ja ülesannete lahendused. Seejärel on kirjeldatud kuidas D3'e paigaldada, et oleks võimalik õppematerjalis olevaid koodinäiteid kasutada ja ülesandeid lahendada.

Peale sissejuhatust on õppematerjal jaotatud 12 peatükiks, mis käsitlevad erinevaid teemasid. Enamik peatükkide lõpus on olemas ka ülesanded, et lugeja saaks rakendada oma uusi teadmisi. Ülesanded on varieeruva raskusega ehk mõned ülesanded on lihtsad ning teised vajavad veidi mõtlemist.

4.4 Õppematerjali ülevaade

Selles alapeatükis annab autor lühikese ülevaate iga peatüki kohta ning nende vajalikkuse kohta. Õppematerjal on jaotatud järgmiselt:

- Selection
- Massiivid
- Andmete sidumine
- Event
- Animatsioonid
- Kujundid (SVG elemendid)
- Lihtsamad diagrammid
- Skaleerimine
- Teljed
- Tulpdiagramm
- Sektordiagramm
- Joondiagramm

Esimese peatükis on kirjeldatud D3 **selection** funktsiooni, mis võimaldab elemente manipuleerida ning neid muuta. Peatüki lõpus on ka 2 ülesannet, mis hõlmavad endas HTML elementide muutmist D3 abil.

Teises peatükis on kirjeldatud erinevaid viise massiivide töötlemiseks kasutades D3 valmisloodud funktsioone. Peatüki lõpus on olemas, mis hõlmab massiivide töötlemist kui ka esimese peatüki teadmisi.

Kolmas peatükk keskendub andmete sidumisele ja nende sisselugemisele erinevatest failidest. Peatükis on kirjeldatud **data(), enter()** ja **exit()** funktsioone, mis võimaldab andmeid siduda erinevate elementidega. Lisaks on peatükkis kirjeldatud ka andmete sisse lugemist eritüüpi andmefailidest. Peatüki lõpus on ka ülesanded uute teadmiste rakendamiseks.

Neljas peatükk keskendub **event()** funktsioonile, mis võimaldab lisada erinevaid interaktsiooni võimalusi D3 abil. Peatüki lõpus on ka ülesandeid, mis nõuab erinevate interaktsiooni funktsioonide loomist.

Viies peatükk katab animatsioonide teemat ning kuidas luua lihtsaid animatsioone D3 abil. Peatükk tutvustab **transition()** funktsiooni ning tema erinevaid võimalusi.

Kuuendas peatükis käsitletakse SVG elementide loomist kasutades D3'e. Peatükis on toodud erinevaid näited kujundite loomisest.

Seitsmes peatükk kirjeldab kuidas luua lihtsaid diagramme, kasutades eelnevates peatükkides omandatuid teadmisi ning SVG atribuute.

Kaheksas peatükk keskendub erinevate skaleerimisfunktsioonidele, mis on D3's olemas. Eelkõige on skaleerimine vajalik jooniste tegemiseks, et nad veebilehele ära mahuksid.

Üheksas peatükk keskendub telgede loomisele, et oleks võimalik neid rakendada erinevatele joonistele. Antud peatükk ei sisalda ülesandeid.

Viimased kolm peatükki keskenduvad erinevate diagrammide loomisele kasutades eelnevates peatükkides rakendatud funktsioone. Diagrammid sisaldavad ka osa funktsioone, mis ei ole eelnevates peatükkides kirjeldatud, sest neid funktsioone kasutatakse väga ainult teatud juhtumitel.

Kõik koodinäited on autori poolt ära kirjeldatud ning on lisatud ka autori kommentaarid erinevate koodiosade kohta. Viimased kolm peatükki on autor sammude kaupa kirjeldanud, et kõik oleks arusaadav

4.5 Õppematerjali testimine

Õppematerjali testimiseks otsustas autor läbi viia sissejuhatava tunni D3'e kohta. Tund sai läbiviidud aine „Üldotstarbelised arendusplatvormid" raames. Autor viis tunni läbi 18. aprillil 2018. aastal ning tund viidi läbi Informaatika eriala esmakursuslastele.

Autor alustas tundi D3'e tutvustamisega ning seejärel hakkas autor koos esmakursuslastega õppematerjali läbi võtma. Tunni läbi viimiseks oli aega poolteist tundi, mistõttu otsustas autor osade teemade puhul ainult need üle rääkida ja kirjeldada koodinäiteid. Testimisel võeti läbi peatükid 1-6 ja peatükk 10. Teemade "Selection" ja "Kujundid" puhul lahendasid õpilased ka ülesandeid, et rakendada

oma uusi teadmisi. Tunni lõpus küsis autor õpilaste käest tagasisidet ning soovitusi õppematerjali täiendamiseks.

Õpilaste tagasiside:

- Õppematerjal oli enamikele arusaadav.
- Ülesanded olid õpilaste arvates jõukohased ning tehtavad. Enamik said ülesannetega hakkama ning
- Diagrammide peatükkide puhul tekkis osadel raskusi ning ajanappuse tõttu ei suudetud graafide teemat korralikult käsitleda.

Õpilastel ei olnud soovitusi anda õppematerjali täiendamise kohta.

Testimisel tekkinud probleemid:

- Ajanappuse tõttu jäid enamus teemad korralikult käsitlemata ning autor pidi selgitama kiirelt üle erinevaid funktsioone. Näiteks ainuüksi esimesele teemale kulus pool tundi, et kõik saaksid aru mida antud teemas käsitletakse.
- Testimisel käsitleti ainult ühte joonise tüüpi.

Kokkuvõte

Käesoleva bakalaureusetöö põhieesmärgiks oli koostada õppematerjal, mille abil oleks võimalik teostada andmete visualiseerimist D3'e abil. Õppematerjali tegemise põhjusteks oli eestikeelse õppematerjali puudumine ning autori enda huvi D3'e ja selle võimaluste kohta. Lisaks antakse bakalaureustöös ülevaade D3'st ja tema alternatiividest ning tuuakse erinevaid näiteid olemasolevatest õppematerjalidest D3'e kohta.

Bakalaureusetöö käigus valminud õppematerjali eesmärgiks oli tutvustada D3'e erinevaid funktsioone, millega oleks võimalik teostada andmete visualiseerimist või jooniste loomist. Õppematerjali eeldusteks oli algteadmised HTML/CSS'i ning JavaScripti kohta. Õppematerjali loomisel lähtuti teiste poolt loodud õppematerjalidest ning ka D3'e dokumentatsioonist.

Õppematerjali testimiseks otsustas autor läbi viia sissejuhatava tunni, et tutvustada D3'e ja tema võimalusi Tallinna Ülikooli Informaatika eriala esmakursuslastele. Testimise käigus lugesid õpilased õppematerjali ning lahendasid õppematerjalile loodud ülesandeid. Tunni lõpuks andsid õpilased tagasisidet õppematerjali kohta. Autor leidis, et materjali läbimiseks ei olnud piisavalt aega, mistõttu jäid osad teemad läbimata ja enamik ülesandeid tegemata. Sellest hoolimata oli õpilaste arvates enamik õppematerjal arusaadav ning ülesanded olid jõukohased.

Õppematerjali koostamine andis autorile rohkem teadmisi D3'e ning tema võimaluste kohta. Lisaks sellele sai autor teadmisi õppematerjali loomise kohta ning kuidas tulevikus õppematerjali paremini luua ja seda testida.

Summary

This Bachelor Thesis is titled “Data Visualization Using D3, Study Material“

Main purpose of this Bachelor Thesis was to create study material, which would help to create data visualizations using D3 library. Reasons for creating this study material were, that there is no Estonian study material for D3 and also author’s personal interest in D3 and it’s possibilities. In Bachelor Thesis, author also gives overview of D3 and it’s alternatives, as well as, gives examples of study materials created by other authors.

The purpose of the study material, which was created for this Bachelor Thesis, was to introduce different functionalities, which is used in creating data visualizations. The prerequisites for completing study material, were basic knowledge about HTML/CSS and JavaScript. Author based his study material from D3 documentation as well as using study materials created by others as inspiration.

For testing purposes, author decided to give an introductory lesson to Tallinn University Informatics first-year students about D3. During testing, students read the study material and tried to complete the tasks, which were made for study material. At the end of the lesson students gave feedback about the study material. In author’s opinion the time meant for testing study material was too short and some topics weren’t covered at all also most tasks were not completed due to time constraint. Despite of the issues, students said that study material was understandable for most part, and tasks were doable.

Creating study material, gave author more knowledge about the D3 and it’s possibilities. In addition to that author also gained more experience in creating study materials and how to test them better in the future.

Kasutatud kirjandus

(D3.js. (kuupäev puudub). *Wikipedia.* Loetud 22. aprill 2018 aadressil: https://en.wikipedia.org/wiki/D3.js)

(D3.js. (kuupäev puudub). *Wikipedia.* Loetud 22. aprill 2018 aadressil: https://et.wikipedia.org/wiki/D3.js)

(Stanford Visualization Group. (2009). *Provotis.* Loetud aadressil: http://mbostock.github.io/protovis/)

(Cabot Tecnology Solution. (2017). *D3.js: the Perfect Dynamic Platform to Build Amazing Data Visualizations*. Loetud aadressil: https://hackernoon.com/d3-js-the-perfect-dynamic-platform-to-build-amazing-data-visualizations-ebe930f0648f)

(Selikoff, S. (2014). *An Introduction to D3*. Loetud aadressil: http://www.samselikoff.com/writings/intro-to-d3/)

(Kesavan, R. (2017). *Top 5 Best Open Source Javascript Chart Library*. Loetud aadressil: https://dzone.com/articles/top-5-best-open-source-javascript-chart-library)

(Ahlin, T. (2017). *Data visualization with Chart.js: An introduction*. Loetud aadressil: http://tobiasahlin.com/blog/introduction-to-chartjs/)

(Slant. (kuupäev puudub). *D3.js vs Chart.js.* Loetud aadressil: https://www.slant.co/versus/10577/10578/~d3-js_vs_chart-js)

(Chartist.js. (kuupäev puudub). Loetud aadressil: https://gionkunz.github.io/chartist-js/)

(W3C. (2008). *SMIL 3.0*. Loetud aadressil: https://www.w3.org/TR/SMIL3/)

(Slant. (kuupäev puudub). *D3.js vs Chartist.js*. Loetud aadressilt: https://www.slant.co/versus/10577/10579/~d3-js_vs_chartist-js)

(Sumanth, K. (2014). *D3.js vs. Google Charts: A Data Scientist's Review*. Loetud aadressil: https://blog.socialcops.com/engineering/d3-js-versus-google-charts/)

(Google. (kuupäev puudub). *Google Charts*. Loetud aadressil: https://google-developers.appspot.com/chart/)

(TutorialsTeacher. (kuupäev puudub). *D3.js Tutorial*. Loetud aadressil: http://www.tutorialsteacher.com/d3js/)

(TutorialsPoint. (kuupäev puudub). *D3.js Tutorial*. Loetud aadressil: https://www.tutorialspoint.com/d3js/)

(Murray, S. (2017). *Interactive Data Visualization for the Web, 2nd Ed.* Ameerika: O'Reilly Media, E-raamat: http://alignedleft.com/work/d3-book-2e)

(Maclean, M. (viimati uuendatud 2018.aastal). *D3 Tips and Tricks v3.x.* Loetud aadressil: https://leanpub.com/D3-Tips-and-Tricks)

(HTML. (kuupäev puudub). *Wikipedia.* Loetud 22. aprill 2018 aadressil: https://et.wikipedia.org/wiki/HTML)

(CSS. (kuupäev puudub). *Wikipedia.* Loetud 22. aprill 2018 aadressil: https://et.wikipedia.org/wiki/CSS)

(JavaScript. (kuupäev puudub). *Wikipedia.* Loetud 22. aprill 2018 aadressil: https://et.wikipedia.org/wiki/JavaScript)

(SVG. (kuupäev puudub). *Wikipedia.* Loetud 22. aprill 2018 aadressil: https://et.wikipedia.org/wiki/SVG)

(Synchronized Multimedia Integration Language. (kuupäev puudub). *Wikipedia.* Loetud 22. aprill 2018 aadressil: https://en.wikipedia.org/wiki/Synchronized_Multimedia_Integration_Language)

(Canvas. (kuupäev puudub). *Wikia.* Loetud 22.aprill 2018 aadressil: http://htmlcss.wikia.com/wiki/Canvas)

(Plug-in (computing). (kuupäev puudub). *Wikipedia.* Loetud 22.aprill 2018 aadressil: https://en.wikipedia.org/wiki/Plug-in_(computing))

(D3.js (kuupäev puudub). *Tutorials.* Loetud addressilt: https://github.com/d3/d3/wiki/Tutorials)

(D3.js (kuupäev puudub). *Plugins*. Loetud addressilt: https://github.com/d3/d3/wiki/Plugins)

(Teek. (kuupäev puudub). *Wikipedia*. Loetud addressilt: https://et.wikipedia.org/wiki/Teek)

(Chart.js. (kuupäev puudub). Loetud addressil: http://www.chartjs.org/)

(Campos H. (2017) *2 Dimensional Histogram*. Pilt võetud addressilt: https://bl.ocks.org/herkulano/4f43dbf3473dc5503052)

Lisad

Lisa 1. Õppematerjal

Tallinna Ülikool

Digitehnoloogiate instituut

Andmete visualiseerimine D3 abil

Õppematerjal

Autor: Matthias Johann Kurs

Tallinn 2018

Sisukord

Sissejuhatus

Käesolev õppematerjal annab lühiülevaate andmete visualiseerimiseks D3 abil. Õppematerjali eesmärgiks on tutvustada erinevaid D3 vahendeid, mida saaks kasutada andmete visualiseerimiseks. Õppematerjal keskendub eelkõige põhiteadmiste andmisele ning lihtsate graafide loomisele. Õppematerjal on eelkõige loodud neile, kes on huvitatud erinevatest viisidest kuidas andmeid kuvada kasutades Javascripti ja tema teeke. Materjal on koostatud D3 versiooniga 5.0.0.

Selleks, et õppematerjalist kõige paremini aru saada, peaks lugeja omama algteadmisi HTML'i, CSS'i ja JavaScripti kohta. Kasuks tuleb ka teadmised SVG'st, sest graafid luuakse SVG abil. Õppematerjal ei kata kõiki D3 võimalusi, keskendudes rohkem põhilistele vahenditele, mida kasutatakse andmete visualiseerimiseks ning on pigem sissejuhatus D3'e.

Õppematerjal koosneb 12 peatükist, millest viimased 3 keskenduvad rohkem eelnevate peatükkide teadmiste rakendamisele:

- Selection
- Massiivid
- Andmete sidumine
- Event
- Animatsioonid
- Kujundid (SVG elemendid)
- Lihtsamad diagrammid
- Skaleerimine
- Teljed
- Tulpdiagramm
- Sektordiagramm
- Joondiagramm

Osad peatükid sisaldavad ka ülesandeid, et rakendada teadmisi õpitud peatükist. Kõik koodinäited ja ülesannete lahendused on leitavad aadressil: http://www.tlu.ee/~mats96/d3/

Töö alustamine

Selleks, et D3'e kasutada oleks vaja tekstiredaktorit ja D3'e teeki. Tekstiredaktorina võib kasutada mistahes enda eelistustele sobivat tekstiredaktorit. Õppematerjali koodinäited on loodud Visual Studio Code (https://code.visualstudio.com) abil.

Teegi saab endale alla laadida D3 kodulehelt https://d3js.org/ või lisada oma scripti sisse antud koodirida:

```
<script src="https://d3js.org/d3.v5.min.js"></script>
```

Soovitatav on oma koodifailid üles panna serverisse, sest võib juhtuda, et osad brauserid ei pruugi andmeid sisse lugeda. (Näiteks Google Chrome võib keelduda andmete sisse lugemisest).

1 Selection

D3 selection võimaldab valida HTML elemente ning muuta nende omadusi, CSS'i ning rakendada erinevaid funktsioone.

1.1 Elementide valimine

- **d3.select("valik")** – valib esimese elemendi, mis vastab valikule. Kui elementi ei leita tagastatakse tühi valik.

```
<html>
<head>
   <script src="https://d3js.org/d3.v5.min.js"></script>
</head>
<body>
   <h1>
     Punane tekst
   </h1>
   <h1>
     Ei ole punane tekst
   </h1>
</body>
<script>
   var selection = d3.select("h1");
   selection.style("color", "red");
</script>
</html>
```

- **d3.SelectAll("valik")** – valib kõik elemendid, mis vastavad valikule. Kui elementi ei leita jääb valik tühjaks.

```
<html>
<head>
    <script src="https://d3js.org/d3.v5.min.js"></script>
</head>
<body>
   <h1>
     Punane tekst
   </h1>
   <h1>
    Nüüd on ka punane
   </h1>
</body>
<script>
   var selection = d3.selectAll("h1");
   selection.style("color", "red");
</script>
```

```
</html>
```

Järgnevas näites on toodud erinevaid viise elementide valimiseks.

```
<html>
<head>
<script src="https://d3js.org/d3.v5.min.js"></script>
</head>
<body>
   <div class="esimene">
   <h1>tekst</h1>
   <p>tekst</p>
   <h1>tekst</h1>
   </div>
</body>
<script>
var n2ide1 = d3.selectAll(".esimene").select("h1") // valib esimese h1
elemendi, mis kuuluvad klassi esimene
var n2ide2 = d3.selectAll(".esimene").select("h1") // valib kõik h1
elemendid, mis kuuluvad klassi esimene
</script>
</html>
```

1.2 Elementide muutmine

D3 võimaldab ka peale elementide valimise ka neid muuta. D3'i abil on võimalik muuta valitud elementide omadusi, stiili (CSS), teksti, HTML'i. D3 võimaldab ka elemente eemaldada.

- **d3.*selection*.text("tekst") –** muudab antud valiku/elemendi teksti sisu
- **d3.*selection*.attr("nimi", "väärtus") –** lisab antud valikule/elemendile atribuudi
- **d3.*selection*.remove() –** eemaldab valitud elemendi
- **d3.*selection*.insert("elemendi nimi") –** lisab uue elemendi valitud elemendi sisse
- **d3.*selection*.append("elemendi nimi") -** lisab uue elemendi valitud elemendi sisse
- **d3.*selection*.property("nimi", "väärtus") –** lisab antud valikule/elemendile atribuudi (kasutatakse nuppude puhul)
- **d3.*selection*.style("nimi", "väärtus") –** muudab antud elemendi stiili

- **d3.*selection*.classed("css klass", bool) –** eemaldab või lisab css klasse antud valikule

```
<html>
<head>
<script src="https://d3js.org/d3.v5.min.js"></script>
</head>
<body>
 <div class="esimene">
   <h1>tekst</h1>
   <p>tekst</p>
   <h1>tekst</h1>
 </div>
 <div class="teine">
   <p>tekst</p>
   <h2>tekst</h2>
 </div>
</body>
<script>
var attribuut = d3.select(".teine").select("p").attr("nimi", "Madis")
//     attribuudi lisamine
var style = d3.select(".esimene").selectAll("h1").style("color", "red")
// CSSi muutmine
var tekst = d3.select(".esimene").select("p").text("tere"); // muudab
valitud elemendi sisu
var                          uus_element                          =
d3.select("body").append("div").append("h1").text("Tere!") // loob uue
elemendi
var   eemalda_element   =   d3.select(".teine").select("p").remove()   //
elemendi eemaldamine
</script>
</html>
```

Rohkem infot selectioni kohta leiab aadressilt: https://github.com/d3/d3-selection

Ülesanded:

1. Luua 2 div elementi. Esimeses divis on suvaline tekst mis on punast värvi, teises divis on tekst sinist värvi. Elemendid luua scripti kaudu.

2. Luua 3-realine tabel, kus 1.rida on kollast värvi ning viimasel real on piirid(borderid) rohelist värvi.

2 Massiivid

D3 raamistikul on olemas erinevaid tööriistu, mis aitavad kaasa massiividega töötamisel.

2.1 Statistika

- **d3.min(massiiv)** – tagastab antud massiivi väikseima väärtuse. Stringide puhul tagastatakse massiivi esimene element.

  Siin juhul peab mainima, et antud meetod ignoneerib *undefined*, *null* ja *NaN* väärtusi.

```
<script src="https://d3js.org/d3.v5.min.js"></script>
<script>
var array = [23,4,5,6]
var min=d3.min(array);
console.log(min);
</script>
```

- **d3.max(massiiv)** – tagastab antud massiivi suurima väärtuse. Stringide puhul tagastatakse massiivi viimane element.

  Siin juhul peab mainima, et antud meetod ignoneerib *undefined*, *null* ja *NaN* väärtusi.

```
<script src="https://d3js.org/d3.v5.min.js"></script>
<script>
var array = [23,4,5,6]
var max=d3.max(array);
console.log(max);
</script>
```

- **d3.extent(massiiv)** – tagastab antud massiivi vähima ja suurima väärtuse, paneb nad massiivi ning järjestab nad kasvavas järjekorras. Stringide puhul tagastatakse esimene ja viimane massiivi element.

```
<script src="https://d3js.org/d3.v5.min.js"></script>
<script>
var array = [23,4,5,6]
var maxmin=d3.extent(array);
console.log(maxmin);
</script>
```

- **d3.sum(massiiv)** – tagastab massiivis olevate arvude summa. Tühja massiivi puhul tagastab väärtuse 0. Antud meetod ignoneerib *undefined* ja *NaN* väärtusi.

```
<script src="https://d3js.org/d3.v5.min.js"></script>
<script>
var array = [23,4,5,6]
var summa=d3.sum(array);
console.log(summa);
</script>
```

- **d3.mean(massiiv)** – tagastab massiivis olevate arvude keskmise. Tühja massiivi puhul tagastab väärtuse 0. Antud meetod ignoneerib *undefined* ja *NaN* väärtusi.

```
<script src="https://d3js.org/d3.v5.min.js"></script>
<script>
var array = [23,4,5,6]
var keskmine=d3.mean(array);
console.log(keskmine);
</script>
```

- **d3.median(massiiv)** – tagastab massiivis olevate arvude mediaani. Tühja massiivi puhul tagastab väärtuse 0. Antud meetod ignoneerib *undefined* ja *NaN* väärtusi.

```
<script src="https://d3js.org/d3.v5.min.js"></script>
<script>
var array = [23,4,5,6]
var mediaan=d3.median(array);
console.log(mediaan);
</script>
```

- **d3.quantile(massiiv, p)** – tagastab p-kvantiili väärtuse. Meetod võtab sisendiks massiivi ning väärtuse p. Siin juhul peab p väärtus jääma vahemikku 0-1.

```
<script src="https://d3js.org/d3.v5.min.js"></script>
<script>
var array = [23,4,5,6]
d3.quantile(array, 0);
d3.quantile(array, 0.5);
```

```
d3.quantile(array, 1);
</script>
```

- **d3.variance(massiiv) –** tagastab antud massiivis olevate arvude variatsiooni väärtuse. Kui massiivis on vähem kui 2 elementi tagastatakse väärtuseks *undefined*. Antud meetod ignoneerib *undefined* ja *NaN* väärtusi.

```
<script src="https://d3js.org/d3.v5.min.js"></script>
<script>
var array = [23,4,5,6]
var variatsioon = d3.variance(array);
console.log(variatsioon);
</script>
```

- **d3.deviation(massiiv)** – tagastab antud massiivis olevate arvude standardhälbe. Kui massiivis on vähem kui 2 elementi tagastatakse väärtuseks *undefined*. Antud meetod ignoneerib *undefined* ja *NaN* väärtusi.

```
<script src="https://d3js.org/d3.v5.min.js"></script>
<script>
var array = [23,4,5,6]
var standardhalve = d3.deviation(array);
console.log(standardhalve);
</script>
```

2.2 Transformatsioonid

- **d3.cross(a,b) –** võtab sisendiks kaks massiivi a ja b. Tekitab massiivi paarid, mille esimeseks elemendiks on esimesest massiivist võetud väärtus ja teiseks elemendiks on teisest massiivist võetud väärtus.

```
<script src="https://d3js.org/d3.v5.min.js"></script>
<script>
var array = [23,4,5,6];
var array2 = [10,12,45,69];
var cross = d3.cross(array, array2);
console.log(cross);
</script>
```

- **d3.merge([massiivid]) –** võtab sisendiks massiivid ja muudab nad üheks massiiviks.

```
<script src="https://d3js.org/d3.v5.min.js"></script>
```

```
<script>
var array = [23,4,5,6];
var array2 = [10,12,45,69];
var kokku = d3.merge([array, array2]);
console.log(kokku);
</script>
```

- **d3.pairs(massiiv)** – tagastab massiivis olevate elementide paarid kui elementide arv massiivis on alla 2 siis tagastatakse tühi massiiv.

```
<script src="https://d3js.org/d3.v5.min.js"></script>
<script>
var array = [23,4,5,6];
var array2 = [10,12,45,69];
var paarid = d3.pairs(array);
console.log(paarid);
</script>
```

- **d3.shuffle(massiiv)** – paigutab massiivis olevad elemendid suvalisse järjekorda.

```
<script src="https://d3js.org/d3.v5.min.js"></script>
<script>
var array = [23,4,5,6];
var array2 = [10,12,45,69];
var suvaline = d3.shuffle(array);
console.log(suvaline);
</script>
```

Rohkem infot massiivide kohta leiab aadressilt: https://github.com/d3/d3-array

Ülesanded:

1. Looge kaks massiivi, ühendage nad omavahel kokku ja leidke seejärel summa ja keskmine.

2. Looge suvaline massiiv ning tagastage tema miinimum, maksimum ja mediaan. Tulemused kuvada veebilehel.

3 Andmete sidumine

D3 võimaldab andmeid siduda DOM elementide külge kasutades **data()** funktsiooni. Antud funktsioon võtab andmeid sisse massiivina ning neid on võimalik manipuleerida.

- **d3.*selection*.data("andmed") -** paneb andme massiivi antud elemendi külge

```
<body>

<script src="https://d3js.org/d3.v5.min.js"></script>
<p></p>            <!-- 4 paragrahvi kuna minu andmetes on 4 arvu -->
<p></p>
<p></p>
<p></p>
<script>
var minuandmed = [10, 20, 30 , 40]    // andmed

var tulemus = d3.select("body").selectAll("p")
                .data(minuandmed)
                .text(function (d, i){
                    return d;  // funktsioon, mis tagastab andmed
                });
</script>
</body>
```

Eelnev näide töötab korralikult, aga sellega esineb üks probleem. Näiteks kui lisada andmeid juurde, siis tuleks lisada ka vastavaid HTML elemente juurde, mis muutub suure hulga andmete puhul väga tüütuks. Samuti võib tekkida probleeme, et osa andmeid lähevad kaotsi või lihtsalt ei kuvata.

Selle probleemi lahendamiseks on D3-l olemas funktsioon **enter()**. Enter funktsioon tuleb kasutusele siis kui me ei tea palju meil andmeid on ja me ei tea palju DOM elemente läheb tarvis.

- **d3.*selection*.data("andmed").enter()** – lisab puuduvatel elementidel oma selectioni ja viidad

```
<body>
<script src="https://d3js.org/d3.v5.min.js"></script>
<script>
var minuandmed = [10, 20, 30 , 40 , 50 , 60 , 70, 80]    // andmed
var tulemus = d3.select("body")
                .selectAll("p")
                .data(minuandmed)
                .enter()
                .append("p")
                .text(function(d){
                    return d;
                });
</script>
</body>
```

Kui **enter()** funktsioon lisab viitasid siis **exit()** funktsioon eemaldab neid. See tuleb kasuks näiteks siis kui meil on DOM elemente rohkem kui andmeid mille küljes nad on.

- **d3.*selection*.data("andmed").exit()** – eemaldab viited ja paneb nad exit valikusse, mida saab hiljem eemaldada

```
<body>
<script src="https://d3js.org/d3.v5.min.js"></script>
<p></p>          <!--- massiivis on ainult 2 numbrit kuid paragrahve on
3 -->
<p></p>
<p>SEDA TEKSTI EI KUVATA</p>  <!-- paragrahv mis eemaldatakse -->
<script>

var myData = [1, 2];

var p = d3.select("body")
          .selectAll("p")
          .data(myData)
          .text(function (d, i) {
              return d;
          })
          .exit()  /*  viimane  paragrahv  pannakse  exit  valikusse
(selection) */
          .remove();       /* exit valikus olev paragrahv eemaldatakse
*/
</script>
</body>
```

Täpsemat infot leiab aadressil:

https://github.com/d3/d3-selection/blob/master/README.md#joining-data

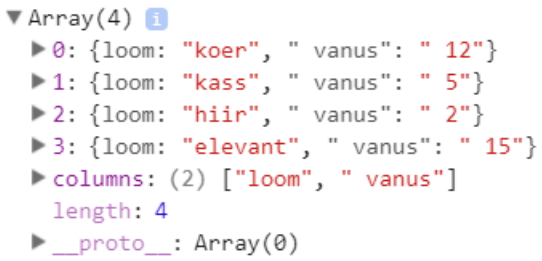
3.1 Andmete laadimine (CSV, JSON, TSV, XML, TXT)

Eelnevalt sai töödatud andmetega, mis oli lokaalsetes muutujates defineeritud. Õnneks võimaldab D3 ka andmeid siduda mujalt võetud andmetega ning nende erinevate tüüpidega. D3 tagastab tulemuse objektidena.

- CSV andmete kättesaamine: **d3.csv("csv faili asukoht")**

```
<body>
<script src="https://d3js.org/d3.v5.min.js"></script>
<script>
d3.csv("andmed.csv").then(function(data){
console.log(data);
});
</script>
</body>
```

Tulemus:

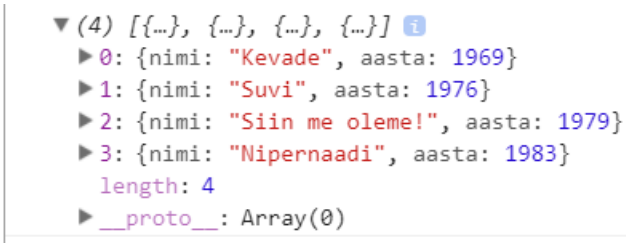


Joonis 1: CSV faili tagastus

- JSON andmete kättesaamine: **d3.json("json fail")**

```
<body>
<script src="https://d3js.org/d3.v5.min.js"></script>
<script>
d3.json("eestifilmid.json", function(data){

      console.log(data);

});
</script>
</body>
```

Tulemus

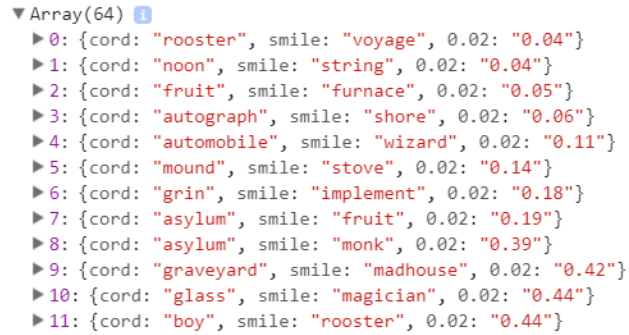


Joonis 2: JSON faili tagastus

- TSV andmete kättesaamine: **d3.tsv("tsv fail")**

```
<body>
<script src="https://d3js.org/d3.v5.min.js"></script>
<script>
d3.tsv("https://raw.githubusercontent.com/pilehvar/ADW/master/resources
/datasets/dataset.tsv").then(function(data){
console.log(data);
});
</script>
</body>
```

Tulemus



Joonis 3: TSV faili tagastus

- XML andmete kättesaamine: **d3.xml("xml fail")**

```
<body>
<script src="https://d3js.org/d3.v5.min.js"></script>
<script>
d3.xml("aastaajad.xml").then(function(data){
        console.log(data);
});
```

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

Tulemus:

```
▼#document
  <aastaajad>
  ▼<aastaaeg>
      <aastaajanimi>Talv</aastaajanimi>
      <v2rvus>Hall</v2rvus>
    </aastaaeg>
  ▼<aastaaeg>
      <aastaajanimi>Kevad</aastaajanimi>
      <v2rvus>Roheline</v2rvus>
    </aastaaeg>
  ▼<aastaaeg>
      <aastaajanimi>Suvi</aastaajanimi>
      <v2rvus>Kollane</v2rvus>
    </aastaaeg>
  ▼<aastaaeg>
      <aastaajanimi>Sygis</aastaajanimi>
      <v2rvus>Punane</v2rvus>
    </aastaaeg>
  </aastaajad>
```

Joonis 4: XML faili tagastus

- TXT andmete kättesaamiseks: **d3.text("txt fail")**

```
<body>
<script src="https://d3js.org/d3.v5.min.js"></script>
<script>
d3.text("andmed.txt").then(function(data){

      console.log(data);

});

</script>
</body>
```

Rohkem infot andmete sisse lugemiseks failidest leiab aadressil: https://github.com/d3/d3-fetch

Ülesanded:

1. Looge suvaline JSON fail ja kuvage selle väärtused veebilehel.

2. Looge muutuja, mis sisaldab endas kolme massiivi. Kuvada iga massiivi väärtused tabelina, kus igale massiivile kuulub üks rida.

4 Event

D3 on olemas erinevad sisseehitatud evendid, mida saab valikutele (*selections)* külge panna.

Eventide meetodid:

- ***selection*.on() –** lisab eventide kuulajaid (*listeners),* et püüda erinevaid eventide tüüpe
- **d3.event –** tagastab sündmuse teabe
- **d3.mouse("konteiner") –** tagastab hiire x ja y koordinaadid antud DOM elemendi (konteiner) sees.
- **d3.touch() –** tagastab puudutuse koordinaadid

```
<html>
<head>
<style>
div {
height: 100px;
width: 100px;
background-color: steelblue;
margin:5px;
}
</style>
<script src="https://d3js.org/d3.v5.min.js"></script>
</head>
<body>
<div> </div>    <!-- element millele lisame evendi -->
<script>
  d3.selectAll("div")     // valime kõik divid
    .on("mouseover", function(){
// funktsioon mis käivitub kui hiir on antud elemendi peal
  d3.select(this)
    .style("background-color", "orange");
    console.log(d3.event);                    // tagastab evendi info
konsoolis
    console.log(d3.mouse(this));    // hiire koordinaadid konsoolis
    })
    .on("mouseout", function(){ // funktsioon mis käivitub kui hiirt
pole peal
    d3.select(this)
      .style("background-color", "steelblue")
    });
</script>
</body>
```

```
</html
```

Rohkem infot eventide kohta leiab aadressil:

https://github.com/d3/d3-selection/blob/master/README.md#handling-events

Ülesanded

1. Sarnaselt koodinäitele looge konteiner (või mingi muu element, mis oleks nähtav), seejärel looge event, mis muudaks terve lehe tausta siniseks kui hiirega antud elemendile peale minna.

2. Looge 3 konteinerit (või elementi, mis oleksid nähtavad). Looge evendid, kus hiirega ühe elemendile peale minnes kaotab teised elemendid ära, hiire ära võtmisel elemendid taastatakse.

5 Animatsioonid

D3 võimaldab ka teha erinevaid animatsioone valitud elementidele. D3 puhul tähendab animatsioon seda, et element läheb ühest olekust teise teatud aja jooksul. Kõik animatsioonid tehakse kasutades ***selection*.transition()** meetodit. Õppematerjal ei käsitle kõiki animatsiooni võimalusi.

- ***selection*.transition()** – määrab valitud elementidele animatsiooni alguse ning funktsioonid
- ***selection*.transition().duration("aeg")** – määrab animatsiooni kestvuse millisekundites
- ***selection*.transition().ease()** – defineerib erinevaid animatsiooni sujuvuse funktsioone
- ***selection*.transition().delay("aeg")** – lisab elemendile viivituse millisekundites

Duration näide:

```
<html>
<head>
<style>
#container {                    /* defineerime konteineri */
height: 200px;
width: 200px;
background-color:blue;

}
</style>
<script src="https://d3js.org/d3.v5.min.js"></script>
</head>
<body>
   <div id="container"></div>
<script>
 d3.select("#container")
   .transition()
   .duration(10000)                              // aja lisamine
   .style("background-color", "black");
</script>
</body>
</html>
```

Erinevate sujuvuse(ease) funktsioonide näited leiab aadressilt: https://bl.ocks.org/d3noob/1ea51d03775b9650e8dfd03474e202fe

Delay näide koos ease funktsiooniga (kujundite loomist käsitletakse järgmises peatükis):

```
<body>
<script src="https://d3js.org/d3.v5.min.js"></script>
<script>
var svg = d3.select("body")
            .append("svg")
            .attr("width", 500)
            .attr("height", 500);



var bar1 = svg.append("rect")
              .attr("fill", "red")
              .attr("x", 100)
              .attr("y", 20)
              .attr("height", 20)
              .attr("width", 10)
var bar2 = svg.append("rect")
              .attr("fill", "blue")
              .attr("x", 120)
              .attr("y", 20)
              .attr("height", 20)
              .attr("width", 10)

update();
function update() {
bar1.transition()
    .ease(d3.easeLinear)
    .duration(2000)
    .attr("height",100)
bar2.transition()
    .ease(d3.easeSin)
    .duration(2000)  // animatsiooni pikkus
    .delay(2000)//  aeg  millal  animatsioon  käivitub  ehk  siis  2000
millisekundi pärast hakkab alles animatsioon käima
    .attr("height",100)
}
</script>
</body>
```

Animatsioonide kohta leiab rohkem infot aadressil: https://github.com/d3/d3-transition

1. Looge 3 elementi ning nendele vastavad animatsioonid:

- Esimene element muudab 10 sekundi pärast värvi ja muutma 5 sekundi pärast tagasi.
- Teine element peaks iga teatud aja tagant suurenema. (vähemalt 2 korda peaks suurenema)
- Kolmas element peaks iga teatud aja tagant vähenema. (vähemalt 2 korda peaks vähenema)

6 Kujundid (SVG elemendid)

D3 abil on võimalik luua erinevaid SVG elemente, sealhulgas erinevaid kujundeid. Kuigi SVG endaga on võimalik kujundeid luua, siis D3 võimaldab neid elementide külge panna ja neid andmete visualeerimiseks manipuleerida.

Algul võib jääda mulje, et D3 abil on kujundite loomine keerulisem ja kulukam kui lihtsalt SVG abil, kuid D3 abil on neid hiljem kergem muuta.

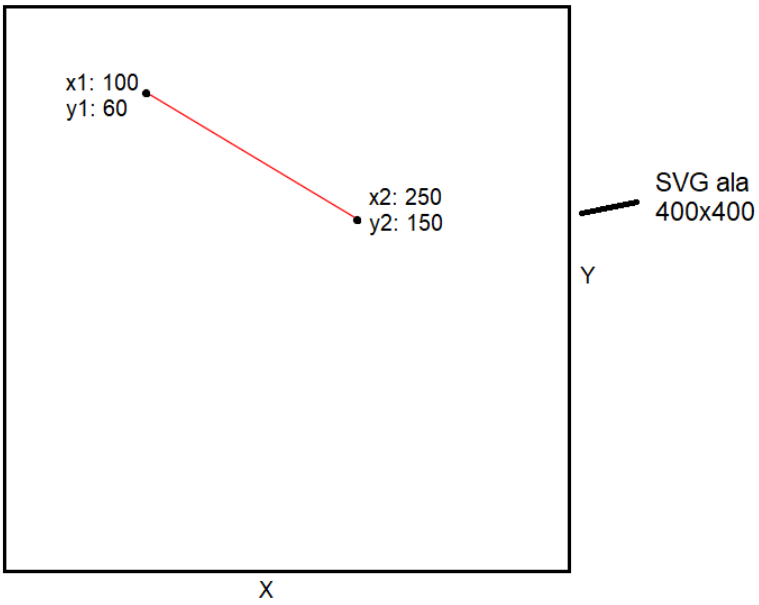
6.1 Joon

SVG abil joone loomine:

```
<svg width="500" height="500">
<line x1="100" y1="50" x2="500" y2="50" stroke="black"/>
</svg>
```

Joone loomine D3 abil:

```
<html>
<body>
<script src="https://d3js.org/d3.v5.min.js"></script>
<script>
var width = 400;  // svg laius
var height = 400;   // svg pikkus
var svg = d3.select("body")         // luuakse svg element body sisse
            .append("svg")
            .attr("width", width)
            .attr("height", height)
            .style("border-style", "solid");
// näidata svg pikkus ja laiust visuaalselt
svg.append("line") // svg elemendile lisatakse joone element
   .attr("x1", 100)                          // algus x koordinaat
   .attr("x2", 250)                          // lõpu x koordinaat
   .attr("y1", 60)                           // algus y koordinaat
   .attr("y2", 150)                         // lõpu y koordinaat
   .attr("stroke", "red")                  // joone värvus
</script>
</body>
</html>
```



Joonis 5: Joon

6.2 Ristkülik

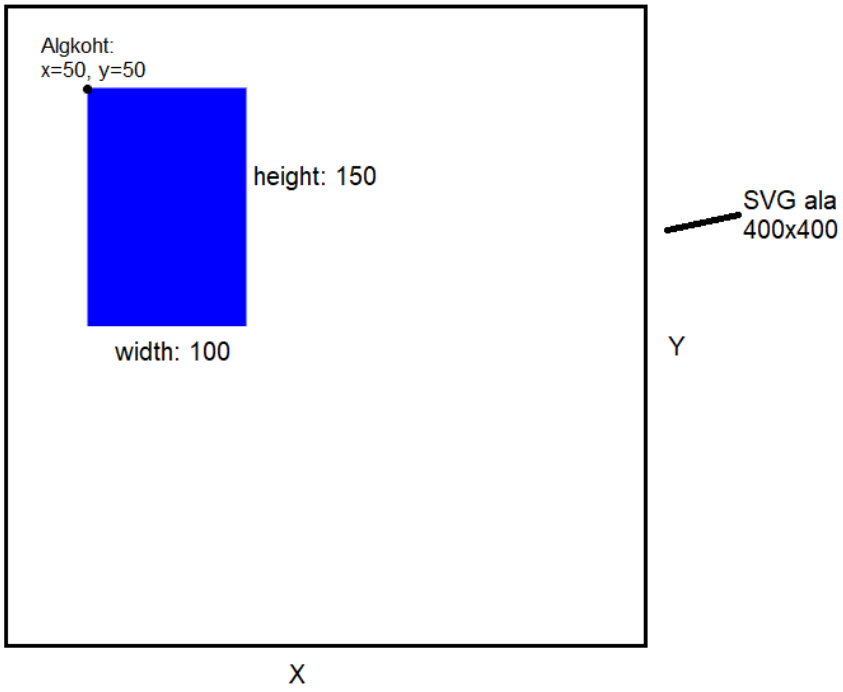
SVG abil ristküliku loomine:

```
<svg width="500" height="500">
<rect x="0" y="0" width="200" height="100"></rect>
</svg>
```

D3 abil ristküliku loomine:

```
<html>
<body>
<script src="https://d3js.org/d3.v5.min.js"></script>

<script>

var width = 400;    // svg suurus ja laius
var height = 400;


var svg = d3.select("body")     // svg elemendi loomine
            .append("svg")
            .attr("width", width)
            .attr("height", height)
            .style("border-style", "solid");
// kood jätkub

svg.append("rect")              // rect elemendi lisamine
   .attr("x", 50)           // ristküliku algus x-koordinaat
```

```
   .attr("y", 50)           // ristküliku algus y-koordinaat
   .attr("width", 100)     // ristküliku laius
   .attr("height", 150)    // ristküliku pikkus
   .attr("fill", "blue");
</script>
</body>
</html>
```



Joonis 6: Ristkülik

6.3 Ring

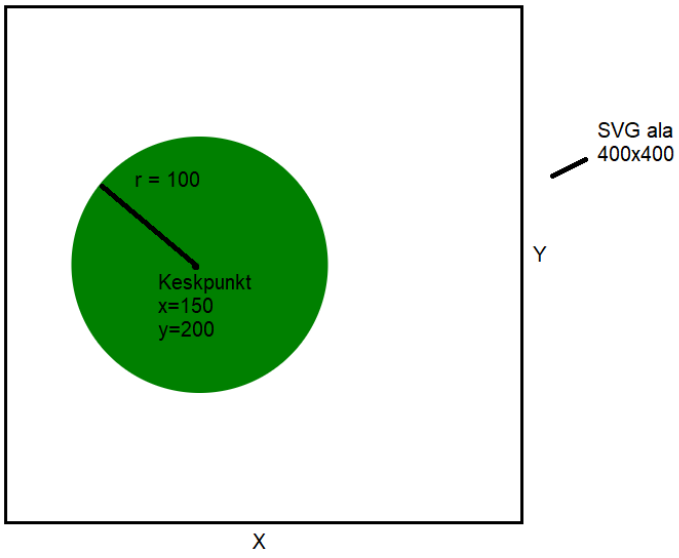
SVG abil ringi joonistamine:

```
<svg width="500" height="500">
<circle cx="250" cy="50" r="50"/>
</svg>
```

D3 abil ringi joonistamine:

```
<html>
<body>
<script src="https://d3js.org/d3.v5.min.js"></script>
```

```
<script>

var width = 400;    // svg suurus ja laius
var height = 400;


var svg = d3.select("body")     // svg elemendi loomine
            .append("svg")
            .attr("width", width)
            .attr("height", height)
            .style("border-style", "solid");

svg.append("circle")              // circle elemendi lisamine
   .attr("cx", 150)  // ringi algus x-koordinaat (keskpunkt)
   .attr("cy", 200) // ringi algus y-koordinaat  (keskpunkt)
   .attr("r", 100)     // ringi raadius
   .attr("fill", "green");
</script>
</body>
</html>
```



Joonis 7: Ring

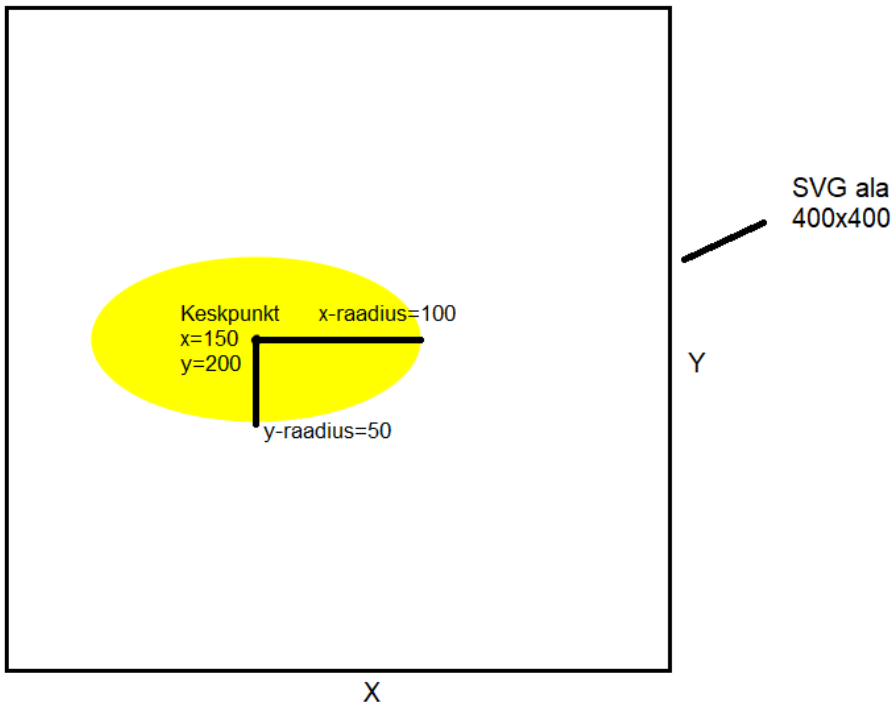
6.4 Ellips

SVG abil ellipsi joonistamine:

```
<svg width="500" height="500">
<ellipse cx="250" cy="25" rx="100" ry="25"/>
</svg>
```

D3 abil ellipsi joonistamine:

```
<html>
<body>
<script src="https://d3js.org/d3.v5.min.js"></script>

<script>

var width = 400;    // svg suurus ja laius
var height = 400;


var svg = d3.select("body")     // svg elemendi loomine
            .append("svg")
            .attr("width", width)
            .attr("height", height)
            .style("border-style", "solid");

svg.append("ellipse")              // ellipsi elemendi lisamine
   .attr("cx", 150) // ellipsi algus x-koordinaat(keskpunkt)
   .attr("cy", 200) // ellipsi algus y-koordinaat (keskpunkt)
   .attr("rx", 100)     // ellipsi x raadius
   .attr("ry", 50)      // ellipsi y raadius
   .attr("fill", "yellow");
</script>
</body>
</html>
```



Joonis 8: Ellips

Lisainfot erinevate kujundite joonistamise kohta leiab aadressil:
https://github.com/d3/d3-shape

Ülesanded:

1. Looge kriipsujuku D3 abil.

2. Lisage kriipsujuku joonisele animatsioonid, mis suurendaksid või vähendaksid kriipsujuku suurust (või kehaosa)

7 Lihtsamad diagrammid (SVG)

Selleks, et luua diagramme või erinevaid graafe D3 abil tuleks aru saada, kuidas käib lihtsate diagrammide loomine SVG abil. Seejärel saame rakendame D3, et luua samaväärne joonis.

7.1 Lihtne tulpdiagramm

SVG näide:

```
<html>
<head>
<script src="https://d3js.org/d3.v5.min.js"></script>
</head>
<style>
svg rect {
fill: blue;
}

svg text {
fill:black;
font: 10px sans-serif;
text-anchor: end;
}
</style>
<body>
<svg class="chart" width="420" height="120">  // svg ala
<g transform="translate(0,0)">        //    transform    määrab    tulba
alguskoha
<rect width="50" height="19"></rect>        // tulba pikkus ja laius
<text x="47" y="9.5">5</text>  // teksti kuvamine
</g>
<g transform="translate(0,20)">
<rect width="100" height="19"></rect>
<text x="97" y="9.5">10</text>
</g>
<g transform="translate(0,40)">
<rect width="120" height="19"></rect>
<text x="117" y="9.5">12</text>
</g>
</svg>
</body>
</html>
```

Selleks, et luua tulpdiagramm SVG abil tuleks:

- Luua SVG ala
- Defineerida palju tulpe meil vaja läheb
- Määrata tulpade pikkused ja laiused ning kuvada nende väärtus.

Vaatame lähedalt järgmist koodijuppi:

```
<g transform="translate(0,20)">       // näitab ära kust tulpa kuvada
<rect width="100" height="19"></rect> // tulba pikkus ja laius
<text x="97" y="9.5">10</text> // teksti kuvamine
</g>
```

<g transform="translate(x,y)"> - g element hoiab endas ühte tulpa ning transform on atribuut, mis määrab antud elemendi asukohta/asendit.

<**text**> **-** on element millega saab teksti joonistel kujutada vastavalt x ja y koordinaatidel ning saab lisada ka muid atribuute.

Antud diagrammi probleemiks on see, et antud diagramm ei võtta andmeid kuskilt sisse vaid kõik on manuaalselt ära paigutatad. Vaatame D3 näidet sarnase joonise tegemiseks.

D3 abil tulpdiagramm (ainult scripti osa):

```
<script>
var data = [5, 10, 12];  // Andmete defineerimine

var width = 200,         // Svg laius
scaleFactor = 10,  // Andmete mahutamise muutuja
barHeight = 30;          // tulba pikkus

var graph = d3.select("body")
              .append("svg")          // lisame body elemendile svg
              .attr("width", width)   // svg laiuse attribuut
              .attr("height", barHeight * data.length); // svg pikkus
vastavalt andmete hulgale

var bar = graph.selectAll("g")  //tulpade asukohtade loomine
               .data(data)
               .enter()
               .append("g")
            .attr("transform", function(d, i) {
```

```
            return "translate(0," + i * barHeight + ")";
        });


bar.append("rect")                  // tulba joonistamine
   .attr("width", function(d) {     // tulba laiuse arvutamine
       return d * scaleFactor;
   })
   .attr("height", barHeight - 1);  // tulba pikkus

bar.append("text")             // teksti lisamine igale tulbale
   .attr("x", function(d) { return (d*scaleFactor); })
   .attr("y", barHeight / 2)
   .text(function(d) { return d; });

</script>

scaleFactorist nii palju, et kuna SVG laius on 200, siis ei pruugi
suurte andmete puhul nende väärtusi kuvada kuna nad ei mahu joonisele
ära. Suurte andmete puhul alandada scaleFactori muutujat. (või
suurendada SVG laiust)
```

Antud kood annab sama tulemuse, mis SVG näite puhul, aga suur erinevus on selles, et me ei pea igat tulpa hakkama eraldi defineerima. D3 näite puhul piisab uue väärtuse lisamisest massiivi ning defineeritud funktsioonid loovad uue tulba automaatselt.

7.2 Lihtne ringdiagramm

Ringdiagrammi loomine käib väga sarnaselt tulpdiagrammi omale. Algselt tuleb luua SVG ala ning seal tuleb iga väärtuse kohta luua oma ring.

```
<script>
var width = 500;  // svg laius
var height = 500;   // svg pikkus
var data = [10, 15, 20, 25, 30]; // andmed
var  colors  =  ['#ffffcc','#c2e699','#78c679','#31a354','#006837']; //
ringide värvid, iga väärtuse kohta oma värv
var svg = d3.select("body")                 // svg loomine
            .append("svg")
            .attr("width", width)
            .attr("height", height);

var g = svg.selectAll("g")                          // andmete külge
panemine
           .data(data)
```

```
            .enter()
            .append("g")
            .attr("transform", function(d, i) {
               return "translate(0,0)";            // alguskoht
            })

g.append("circle")                          /*ringi joonistamine   */
 .attr("cx", function(d, i) {
     return i*100 + 50;// ringi x koordinaadi määramine vastvalt ta
väärtusele
  })
 .attr("cy", function(d, i) {
     return 100;                             // ringi y-koordinaat
  })
 .attr("r", function(d) {
   return d*1.5;                             // ringi raadius määramine
vastavalt ta väärtusele
  })
 .attr("fill", function(d, i){
     return colors[i];                       // ringi värvus
  })

g.append("text")                          // teksti lisamine koos tema
atribuutidega
  .attr("x", function(d, i) {
     return i * 100 + 40;
  })
 .attr("y", 105)
 .attr("stroke", "teal")
 .attr("font-size", "12px")
 .attr("font-family", "sans-serif")
 .text(function(d) {
    return d;
 });
</script>
```

Ülesanded:

1. Võtke tulpdiagrammi näide ja pange tulbad kuvama ülevalt alla poole.

2. Looge ringdiagramm, mis kuvatakse ülevalt alla poole ning looge nupp, mis kahekordistab iga ringi väärtus ning nende suurust.

8 Skaleerimine

D3 sisaldab endas erinevaid skaleerimisfunktsioone, et andmeid paremini visualiseerida. Eelkõige kasutatakse skaleerimist selleks, et andmeid graafile ära mahuks. Näiteks on osade andmete väärtused liiga väikesed ja neid pole graafil näha. Suurte andmetel ei pruugi joonis piisavalt suur olla.

Vaatame skaleerimisfunktsiooni **d3.scaleLinear()** ning kuidas ta töötab.

```
<html>
<head>
<script src="https://d3js.org/d3.v5.min.js"></script>
</head>
<script>
var myScale = d3.scaleLinear()  // funktsiooni defineerimine
                .domain([0, 10]) // väärtused, mis võtab vastu
                .range([0,    100]);    //    väärtuste    vahemik,    mis
tagastatakse

console.log(myScale(3)); // tagastab väärtuse 30
</script>
</body>
</html>
```

- **d3.scaleLinear.domain([min, max]) -** domain määrab ära, mis väärtusi funktsioon vastu võtab ehk antud näites võtab vastu kõik arvud, mis on vahemikus 0 – 10.
- **d3.scaleLinear.range([min, max]) –** range võtab domain vahemikus olevad väärtused ja kaardistab need vastavalt range vahemikule. Koodinäites tähendab see seda, et väärtustele 0 – 10 vastab uus väärtus vahemikust 0 – 100.

Veel näited erinevatest skaaladest:

```
<script>
var  timeScale  =  d3.scaleTime()                          //  kuupäevade
skaaleerimine
.domain([new Date(2016, 0, 1), new Date(2017, 0, 1)])
.range([0, 700]);

console.log(timeScale(new Date(2016, 6, 1)));
```

```
var myData = [0, 5, 7, 10, 20, 30, 35, 40, 60, 62, 65, 70, 80, 90,
100];

var quantileScale = d3.scaleQuantile() // iga myData andmele vastab
range's olev väärtus
.domain(myData)
.range(['lightblue', 'orange', 'lightgreen']);

quantileScale(0);   // tagastab 'lightblue'
quantileScale(20);  // tagastab 'lightblue'
quantileScale(30);  // tagastab 'orange'
quantileScale(65);  // tagastab 'lightgreen'
</script>
```

Skaaleerimise kohta leiab rohkem infot aadressil: https://github.com/d3/d3-scale

Ülesanded:

1. Uuri erinevaid skaaleerimisnäited aadressil http://d3indepth.com/scales/ ning katseta erinevaid skaaleerimisfunktsioone.

9 Teljed

D3 abil on ka luua nii x-telgesid kui ka y-telgesid, kasutades **d3.Axis** funktsioone. Telgede loomiseks on neli funktsiooni:

- **d3.axisTop() –** loob ülemise horisontaalse x-telje
- **d3.axisBottom() –** loob alumise horisontaalse x-telje
- **d3.axisRight() –** loob y-telje, mis on parempoolse orientatsiooniga
- **d3.axisLeft() –** loob y-telje, mis on vasakpoolse orientatsiooniga

X-telje näide:

```
<html>
<head>
<script src="https://d3js.org/d3.v5.min.js"></script>
</head>
<body>
<script>
var width = 400,  // svg laius
height = 100;     // svg pikkus
var data = [0, 10, 15, 20, 25, 50, 100]; // andmed

var svg = d3.select("body")             // svg külge panemine
            .append("svg")
            .attr("width", width)
            .attr("height", height);

var scale = d3.scaleLinear()       // Skaleerimine et telg svg alasse
mahuks
              .domain([d3.min(data), d3.max(data)])
              .range([0, width - 100]);

var  x_axis  =  d3.axisBottom()         //  x-telje  loomine  ja  selle
skaleerimine
               .scale(scale);

svg.append("g")                 // x-telje kuvamine
   .call(x_axis);


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

X-telg ja Y-telg koos:

```
<html>
<head>
<script src="https://d3js.org/d3.v4.min.js"></script>
</head>
<body>
<script>
var width = 400,  // svg laius
height = 200;     // svg pikkus

var data = [0, 10, 15, 20, 25, 50, 100]; // andmed

var svg = d3.select("body")             // svg külge panemine
            .append("svg")
            .attr("width", width)
            .attr("height", height);

var xscale = d3.scaleLinear()             // x-telje skaleerimine
               .domain([d3.min(data), d3.max(data)])
               .range([0, width - 100]);

var yscale = d3.scaleLinear()               // y-telje skaleermine
               .domain([d3.min(data), d3.max(data)])
               .range([height/2, 0]);

var  x_axis  =  d3.axisBottom()  //  x-telje  loomine  ja  skaleerimise
rakendamine
               .scale(xscale);

var  y_axis  =  d3.axisLeft()  //  y-telje  loomine  ja  skaleerimise
rakendamin
               .scale(yscale);

svg.append("g")                 // y-telje kuvamine
   .attr("transform", "translate(50, 10)")   // y-telje asukoht lehel
   .call(y_axis);

svg.append("g")                             // x-telje kuvamine
   .attr("transform",  "translate(50,110)")         //  x-telje  asukoht
lehel
   .call(x_axis);
</script>
</body>
</html>
```

Täpsemat info telgede kohta leiab aadressil: https://github.com/d3/d3-axis

10 Tulpdiagramm

Antud peatükis luuakse tulpdiagramm kasutades eelnevates peatükkides kasutatud funktsioone.

Loome tulpdiagrammi, mis kuvaks inimeste nimesid ja nende kasvu. Selleks loome csv faili, mis hoiab meie andmeid.

Inimesed.csv:
nimi,kasv,kaal
Madis,180,80
Kalle,170,65
Tiina,175,67
Mari,165,60
Vadim,190,90

Kõige pealt loome endale sobiva SVG ala

```
<svg width="1000" height="1000"></svg>
```

Scripti sisse loome muutuja **margin ,** et me saaks SVG ala hiljem muuta kui vaja.

```
<script>

var margin = 300;   // margin muutuja abil saame muuta svg ala suurust

var svg = d3.select("svg")
var width = svg.attr("width") - margin;
var height = svg.attr("height") - margin;
</script>
```

Loome x-telje ja y-telje skaleerimisfunktsioonid, et andmed mahuksid SVG alale ära.

```
var x_scale = d3.scaleBand().range([0, width]); //scaleBand kasutatakse
tulpade puhul

var y_scale = d3.scaleLinear().range([height, 0]);
```

Graafi alustuskoha loomine:

```
var g = svg.append("g")      // graafi alustuskoht
                   .attr("transform", "translate(" + 100 + "," +  100 +
")");
```

Andmete sisse lugemine ja nendele vastavalt skaala väärtuse rakendamine.

```
d3.csv("inimesed.csv").then(function(data){    // loeb andmed sisse,
kui saada kätte siis tagastatakse error
x_scale.domain(data.map(function(d){return d.nimi;})); // nimed
skaaleeritakse
y_scale.domain([0, d3.max(data, function(d){return d.kasv})]);//kasvud
skaaleeritakse
```

Telgede loomine:

```
g.append("g")                                          // x-telje
loomine
 .attr("transform", "translate(0," + height + ")")       // x-telje
asukoht lehel
 .call(d3.axisBottom(x_scale))        // x-telje joonistamine
 .append("text")                     // teksti loomine
 .attr("y", height - 675)            // teksti koordinaadid
 .attr("x", width)
 .attr("text-anchor", "end")
 .attr("stroke", "black")
 .text("Nimed")                                        // teksti
nimetus

g.append("g")                                          // y-telg
 .call(d3.axisLeft(y_scale).tickFormat(function(d){
      // funktsioon tagastab pikkused 10-liste vahedega
                return d;
            })
  .ticks(10))
  .append("text")    // teksti lisamine
  .attr("transform", "rotate(-90)") // ümberpööramine
  .attr("y", 6)
  .attr("dy", "-5.1em")
  .attr("text-anchor", "end")
  .attr("stroke", "black")
  .text("Kasvud");
```

Tulpade loomine:

```
g.selectAll(".bar")  // tulpade joonistamine
 .data(data)                        // andmete sisestamine
 .enter().append("rect")
 .attr("class", "bar")
 .attr("x", function(d){ return x_scale(d.nimi);}) // x-koordinaat
vastavalt skaleeritud nime väärtusele
 .attr("y", function(d){ return y_scale(d.kasv);}) // y- koordinaat
vastavalt skaleeritud kasvu väärtusele
 .attr("width", x_scale.bandwidth() - 15)  // tulba laius
```

```
 .attr("height", function(d){return height - y_scale(d.kasv);}); //
tulba pikkus
```

Ülesanded:

1. Looge teine tulpdiagramm, kuid seekord kuvage kaalude kaupa.

2. Täiendage esimese ülesande graafi lisades ühe animatsiooni ja evendi graafis olevatele tulpadele.

11 Sektordiagramm

Loome sektordiagrammi, mis kuvaks iga inimese kaalu ja selle osakaalu graafis.

Esimesena tuleks luua SVG ala:

```
<svg width="500" height="400"></svg>
```

Loome SVG laius ja pikkuse muutujad ning loome ka muutuja raadiuse jaoks.

```
var svg = d3.select("svg")  // valime svg ala

var width = svg.attr("width");  // svg laius muutujas
var height = svg.attr("height"); // svg pikkus muutujas
var radius = Math.min(width,height) / 2;
// kuna tegemist on  sektordiagrammiga on meil vaja ka raadiust
```

Graafi alustuskoha loomine:

```
var g = svg.append("g")      // graafi alustuskoht
          .attr("transform", "translate(" + width / 2 + "," + height /
2 +")");
```

Selleks, et eristada iga inimest sektordiagrammil määrame igale ühele oma värvi:

```
var                              color                                =
d3.scaleOrdinal(['#4daf4a','#377eb8','#ff7f00','#984ea3','#e41a1c']);
// iga inimese jaoks erivärv, et saaks eristada
```

Järgmiseks loome muutuja, mis võimaldaks meil joonistada kaari, kasutades **arc()** funktsiooni. Lisaks loome ka muutuja, et hiljem nimed sektoritele paigutada.

```
var path = d3.arc()                     // kaarte joonistamine
            .outerRadius(radius-10)     // kaare välimine raadius
            .innerRadius(0);            // kaare siseraadius
```

Nüüd oleks vaja luua muutuja, mis võimaldaks joonistada kaari vastavalt väärtustele. Selleks kasutame **pie()** funktsiooni. **Pie()** funktsioon genereerib kaarte algus- ja lõppkohad vastavalt andmetele.

```
var pie = d3.pie().value(function(d) {
           return  d.kaal;  //  genereerime  kaarte  pikkused  vastavalt
andmetele
          });
```

Loeme andmed sisse ja lisame need kaarte elementide sisse:

```
d3.csv("inimesed.csv").then(function(data){  // andmete kätte saamine

var arc = g.selectAll(".arc") // loome arc elemendi ja sisestame talle
andmed
.data(pie(data))
.enter().append("g")
.attr("class", "arc");  // rakendab arc classi kaart elementidele
```

Joonistame kaared:

```
arc.append("path")    // joonistame kaared
   .attr("d", path)
   .attr("fill", function(d) { return color(d.data.nimi); });
// kaarte värvid vastavalt nimedele
   });
```

Viimasena lisame sektoritele nimetused:

```
arc.append("text")            // teksti lisamine igale sektorile
   .attr("transform", function(d) {
    return "translate(" + label.centroid(d) + ")";
// teksti asetamine sektori keskele
    })
   .text(function(d){return d.data.nimi;}) // nimede lisamine
```

Ülesanded:

1. Looge sõõriku graaf (donut chart, ehk sektordiagramm, mis on keskelt tühi).

12 Joondiagramm

Antud peatükis loome joondiagrammi, kasutades D3 vahendeid.

Koodinäites kasutavad andmed leiab aadressil:
http://www.tlu.ee/~mats96/d3/joondiagramm/naited/bitcoin.json

Esimesena tuleks luua muutujad SVG ala jaoks ning ääriste jaoks, et joonis SVG alast välja ei jääks.

```
var margin = {top:20, right: 20, bottom: 30, left:50} // äärised

var width = 1000 - margin.left - margin.right; // laiuse muutuja
var height = 1000 - margin.top - margin.bottom; // pikkuse muutuja

var svg = d3.select("body").append("svg")  // svg ala joonistamine
          .attr("width", width + margin.left + margin.right)  // svg
ala laius
          .attr("height", height + margin.top + margin.bottom) // svg
ala pikkus
          .append("g")
        .attr("transform",  "translate("  +  margin.left  +  ","  +
margin.top + ")") // joonise alustuskoht
```

Järgmisena loome skaaleerimismuutujad telgede jaoks ning loome muutuja, mis aitab kuupäevasid sisse lugeda ning neid vormistada.

```
var parseTime = d3.timeParse("%d-%b-%y"); // kuupäevade vorming (%d-
päev, %b-kuu, %y-aasta)

var x = d3.scaleTime().range([0, width]); // x-telje skaleerimine
var y = d3.scaleLinear().range([height, 0]); // y-telje skaleerimine
```

timeParse() – funktsioon, mis aitab vormindada kuupäevasid

Nüüd loome muutuja joone jaoks, mida me hiljem graafile joonistame.

```
var valueline = d3.line()    // joone muutuja
            .x(function(d) {return x(d.date);}) // x-koordinaat sõltub
kuupäevast
            .y(function(d)  {return  y(d.price);})    //  y-koordinaat
sõltub hinnast
```

Andmete sisselugemine ja iga väärtusele vormistuse rakendamine.

```
d3.json("bitcoin.json").then(function(data){// andmete sisse lugemine
```

```
data.forEach(function(d){     // iga andme vormistamine
d.date = parseTime(d.date);
d.price = +d.price;
});
```

Telgede miinimum ja maksimumväärtused skaaleerimise abil.

```
x.domain(d3.extent(data, function(d){return d.date})); // skaleerimine
x-teljel
y.domain([5000    ,    d3.max(data,    function(d){return    d.price})]);
//skaleerimine y-teljel
```

Joone joonistamine

```
svg.append("path")    // joone joonistamine
   .data([data])
   .attr("class", "line")
   .attr("d", valueline)
```

X-telje ja y-telje joonistamine

```
svg.append("g")          // x-telg
   .attr("transform", "translate(0," + height  + ")")
   .call(d3.axisBottom(x));

svg.append("g")    // y-telg
   .call(d3.axisLeft(y));
```

Teksti lisamine

```
svg.append("text")    // teksti lisamine
   .attr("x", width/2)
   .attr("y", height/2)
   .style("fill", "grey")
   .text("BITCOIN GRAPH")
   .style("font-size", "40px")
```

Kasutatud kirjandus

(TutorialsTeacher. (kuupäev puudub). *D3.js Tutorial*. Loetud aadressil: http://www.tutorialsteacher.com/d3js/)

(Cook, P. (2014). *D3's enter() and exit(): Under the Hood*. Loetud aadressil: http://animateddata.co.uk/lab/d3enterexit/)

(Bostock, M. (2017). *Data-Driven Documents*. Loetud aadressil: https://d3js.org/)

(Bostock, M. (2018). *D3 Wiki*. Loetud aadressil: https://github.com/d3/d3/wiki)

(d3noob. (2017). *Transition Easing Comparison in v4*. Loetud aadressil: https://bl.ocks.org/d3noob/1ea51d03775b9650e8dfd03474e202fe)

(pilehvar (2014). *ADW.* Andmed võetud aadressilt: https://raw.githubusercontent.com/pilehvar/ADW/master/resources/datasets/dataset.tsv)

(Cook, P. (2016). *Scale functions*. Loetud aadressil: http://d3indepth.com/scales/)

(Murray, S. (2015). *Axes.* Loetud aadressil: http://alignedleft.com/tutorials/d3/axes)

(Bostock, M. (2013). *Let's Make a Bar Chart, III.* Loetud aadressil: https://bost.ocks.org/mike/bar/3/)

(d3noob. (2018). *Simple line graph with v4*. Loetud aadressil: https://bl.ocks.org/d3noob/402dd382a51a4f6eea487f9a35566de0)