SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA
Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek

Priručnik za laboratorijske vježbe iz kolegija

Vizualizacija podataka

Nositelj kolegija:
doc.dr.sc. Josip Job

suradnik:
doc.dr.sc. Časlav Livada

Osijek, 2016.

Sadržaj

1 Uvod u HTML i CSS. JavaScript i DOM. Uvod u D3.js i SVG

1.1 Uvod u HTML i CSS

1.1.1 HTML

HTML je kratica od engleskog pojma HyperText Markup Language što je naziv jezika za izradu međusobno povezanih dokumenata. HTML nije programski jezik već je skup pravila kako renderirati stranicu. Sastoji se od tekstualnih oznaka (engl. tags) koje pregledniku daju upute o strukturi i rasporedu elemenata na stranici koju je potrebno prikazati. Elementi stranice sastoje se od oznake početka elementa, sadržaja elementa i oznake kraja elementa, npr.:

```
<p>Ovo je HTML element paragraf koji se sastoji od oznake
za početak elementa, sadržaja elementa i oznake za kraj
elementa.</p>
```

Tablica 1: Primjeri oznaka često korištenih HTML elemenata.

Oznaka (tag)	Značenje
$\langle html \rangle$	Definira HTML dokument
$\langle body \rangle$	Definira tijelo dokumenta
$\langle h1 \rangle$ ... $\langle h6 \rangle$	Definira zaglavlje 1 to zaglavlje 6
$\langle p \rangle$	Definira paragraf
$\langle br \rangle$	Umeće prijelom retka
$\langle hr \rangle$	Stavlja horizontalnu crtu
$\langle ! -- \rangle$	Definira komentar

HTML oznake pišu se unutar znakova $<$ i $>$, a unutar kojih, osim samog imena oznake, mogu biti navedeni različiti atributi koji dodatno opisuju specifična svojstva pojedinog elementa.

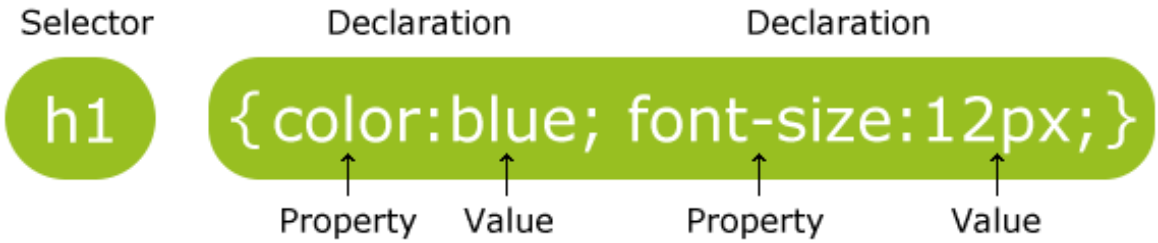
Svi HTML elementi mogu imati svojstva. Neka od najčešće upotrebljavanih svojstava su *id, name* i *class*, a njihova namjena je omogućiti odabir određenog elementa prema jedinstvenom identifikatoru, prema imenu, odnosno definirati pripadnost određenoj klasi elemenata.

1.1.2 CSS

CSS je kratica engleskog pojma Cascading Style Sheets. CSS je jezik koji služi za manipulaciju prezentacijom HTML elemenata, odnosno omogućava promjenu izgleda kako pojedinih elemenata tako i cjelokupne stranice. Najčešće se koristi za promjene boje pozadine stranice ili boje samih elemenata, stilova teksta, dimenzija, položaja elemenata i ostalog.

Slika 1.1: Primjeri HTML naredbi

```
<!DOCTYPE html>
<html>
```



Slika 1.2: Primjeri CSS deklaracije

```
<head>
 <style>
  body {
          background-color:#d0e4fe;
  }
  h1 {
           color:blue;
           text-align:center;
           font-size:20px; }
  p {
           font-family:"Times New Roman";
           font-size:12px;
  }
 </style>
</head>
<body>
 <h1>CSS example!</h1>
 <p>This is a paragraph.</p>
</body>
```

Odabiranje HTML elemenata pomoću CSS-a:

– odabir prema nazivu elementa – omogućava odabir određenog elementa prema njegovu nazivu (oznaci):

```
p
{
text-align:center;
color:red;
font-size:12px;
}
```

– odabir prema id-u elementa – omogućava odabir određenog elementa prema njegovom jedinstvenom identifikatoru:

```
#para1
{
text-align:center;
```

```
color:red;
}
```

– odabir prema klasi elementa – omogućava odabir određenog elementa prema klasi kojoj element pripada:

```
.center
{
text-align:center;
color:red;
}
```

Odabir je dozvoljen i na način da se kombiniraju navedeni pristupi pa je tako moguće odabrati samo elemente određenog tipa koji pripadaju nekoj klasi:

```
p.center
{
text-align:center;
color:red;
}
```

Postoje tri dozvoljena načina umetanja CSS-a:

– eksterni stilovi (External Style Sheet):

```
<head>
 <link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
```

– interni stilovi (Internal Style Sheet):

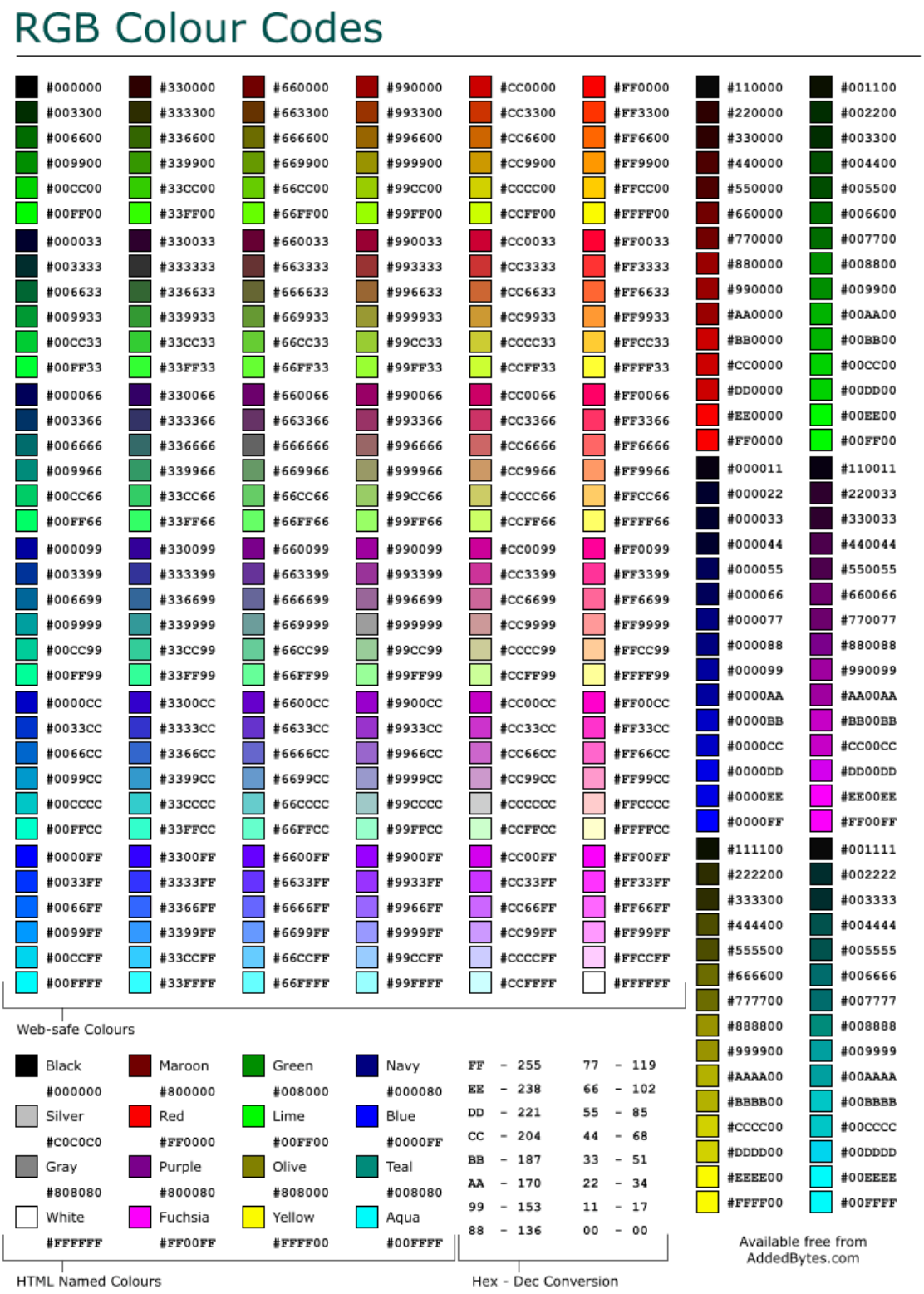
```
<head>
 <style>
  hr {color:sienna;}
  p {margin-left:20px;}
  body {background-image:url("images/background.gif");}
 </style>
</head>
```

– jednoredni stilovi (engl. Inline Styles):

```
<p style="color:sienna;margin-left:20px;">This is a paragraph.</p>
```

Redoslijed primjene stilova:

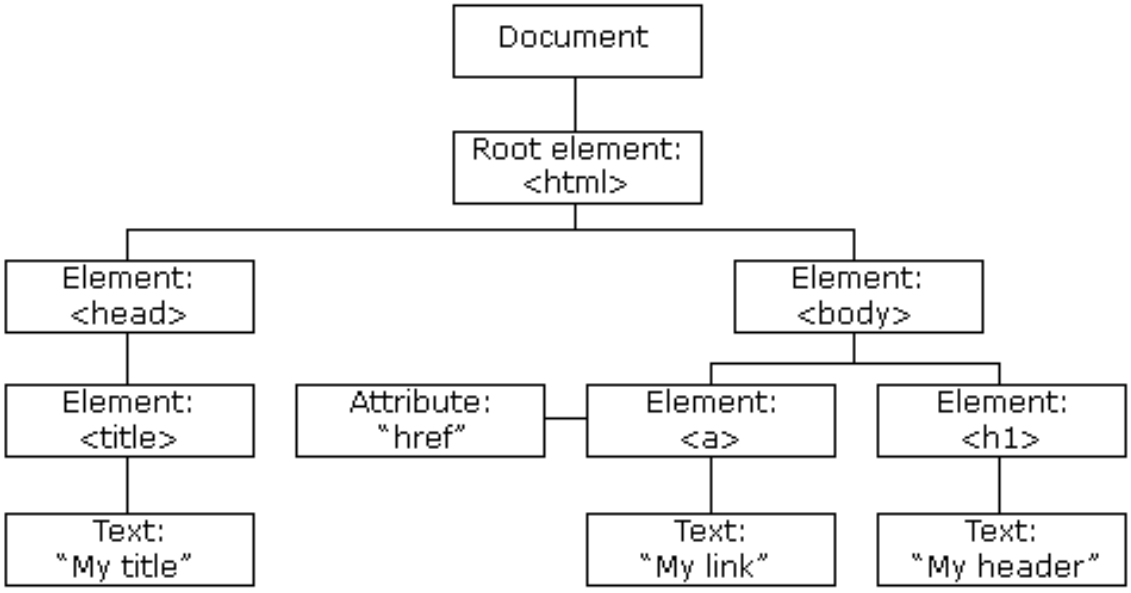
1. Browser default
2. External style sheet
3. Internal style sheet (in the head section)
4. Inline style (inside an HTML element)



Slika 1.3: RGB kodovi boja

1.2 HTML DOM i JavaScript

HTML DOM (Document Object Model) ili objektni model dokumenta jest stablo objekata neke HTML stranice. Nakon učitavanja stranice, preglednik izrađuje stablo objekata prema strukturi navedenoj u samom HTML dokumentu.



Slika 1.4: Stablo objekata HTML stranice

Osim navedenog, DOM je i programsko sučelje za HTML te definira:

- HTML elemente kao objekte
- svojstva (engl. properties) svih HTML elemenata
- metode za pristup svim HTML elementima
- događaje (engl. events) za sve HTML elemente

Objektni model dokumenta, DOM, omogućava pristup svim elementima i atributima elemenata HTML dokumenta te njihovu izmjenu, dodavanje novih i brisanje postojećih. Programski jezik za manipulaciju objektima u DOM-u je JavaScript.

JavaScript omogućuje:

- promjenu svih elemenata stranice
- promjenu svih atributa elemenata stranice;
- promjenu svih stilova stranice;
- brisanje postojećih HTML elemenata i atributa;
- dodavanje novih HTML elemenata i atributa;
- reakciju na sve postojeće HTML događaje (engl. events) stranice;
- dodavanje novih HTML događaja.

1.3 Uvod u D3.js

Data driven documents (D3.js) je Javascript biblioteka za manipulaciju HTML elementima prvenstveno u smislu jednostavne i brze izrade vizualizacija za web. Autori su Mike Bostock, Jeffrey Heer, Vadim Ogievetsky i dr., a prva javno dostupna verzija objavljena je 2011. godine. Izrada vizualizacija temelji se na jednostavnoj manipulaciji HTML5, CSS i SVG elemenata/objekata putem Javascripta. D3.js biblioteka nastala je kao rezultat rada na razvoju različitih alata:

- Prefuse, Java + Java plug-in [2005]
- Flare, ActionScript + Flash plug-in [2007]
- Protovis, JavaScript [2009]

D3.js je, zapravo, API koji se sastoji od nekoliko stotina funkcija koje se mogu svesti na nekoliko kategorija vidljivih u tablici 2.

Tablica 2: Primjeri kategorija D3.js-a.

Kategorija	Opis
Selections	Odabir elementa
Transitions	Prijelazi/animacije
Arrays	Polja
Math	Matematičke funkcije
Color	Boje
Scales	Mjerila
SVG	SVG
Time	Vrijeme
Layouts	Prikazi
Geography	Rad s geografskim podacima, izrade karata
Geometry	Geometrijske transformacije
Behaviours	Ponašanje

1.3.1 Odabir elemenata

Odabir elemenata vrši se pomoću dvije metoda *select()* i *selectAll()*.

- *select()* - odabir prvog elementa koji odgovara parametru (znakovnom nizu).
- *selectAll()* - odabir svih elemenata koji odgovaraju parametru.

Traženi parametar može biti:

a) "ime_elementa"

b) "#id_elementa"

c) ".klasa"

d) atribut - "[color=red]"

e) "parent child"

a) "ime_elementa" – potrebno je navesti ime HTML elementa kojeg se želi dohvatiti i metoda će vratiti polje s dohvaćenim elementom, odnosno polje s dohvaćenim elementima kod korištenja metode *selectAll*.

```
d3.select("body")
 [Array[1]
        0: body
        length: 1
        parentNode: html__proto__: Array[0]]
```

b) "#id_elementa" – prosljeđivanjem ID-a metoda *select* u obliku polja vraća pronađeni element. Moguće je i korištenje metode *selectAll* iako bi ID trebao biti jedinstven pa samim time i rezultat obje metode jednak. Ispred samog ID-a, potrebno je navesti znak '#'.

```
d3.select("#bodyID")
```

c) ".klasa" – za dohvat elemenata koji pripadaju određenoj klasi potrebno je, kao parametar, navesti naziv klase, a ispred naziva potrebno je navesti znak '.'

```
d3.select(".myClass")
```

d) atribut - "[color=red]" – dohvat elemenata moguć je i preko željenog atributa: [naziv atributa=vrijednost atributa]

```
d3.selectAll("[background-color=red]")
```

e) "parent child" – dohvat koji se zasniva na odnosu roditelj-dijete radi se tako da se imena HTML elemenata za traženi odnos odvoje razmakom i to tako što se navodi prvo roditelj, a potom dijete.

```
d3.selectAll("body div")
```

Važno je i napomenuti kako će svaka od gore navedenih primjena metoda *select* i *selectAll* vratiti polje kao rezultat, a razlika između uspješnog i neuspješnog dohvata je u tome što će u slučaju neuspjeha polje sadržavati *null* vrijednost.

```
d3.select("body")
 [Array[1]
        0: null
        length: 1
        parentNode: html__proto__: Array[0]]
```

1.3.2 Atributi

Izmjena i dohvat vrijednosti pojedinog atributa nekog elementa vrši se pomoću metode *attr()*. Većina metoda u biblioteci D3.js ima dvostruku funkcionalnost koja ovisi o broju parametara pa tako metoda attr može dohvatiti vrijednost pojedinog atributa (engl. *getter method, accessor method*), ali i postaviti vrijednost pojedinog atributa (engl. *setter method*):

- *attr(name[,value])* - ako je parametar value zadan - mijenja vrijednost atributa.
- *attr(name)* - ako parametar value nije zadan - vraća vrijednost atributa odabranog elementa, odnosno vrijednost atributa prvog elementa (ako je odabrano više od jednog elementa).

1.3.3 Stilovi

Definiranje i izmjena stila vrši se preko metode *style()* ili preko metode *attr()*:

- style("property", "value")

```
d3.select("body").style("background", "lightblue");
```

- attr("style", "property:value;")

```
d3.select("body").attr("style", "background:lightblue;");
```

1.3.4 Dodavanje elemenata

Dodavanje novih elemenata i sadržaja obavlja se metodama *append()*, *insert()* i *html()*:

- *append()* - dodaje novi element kao posljednje dijete svakog elementa iz odabira. Vrsta elementa ovisi o predanom parametru.

```
d3.select("body")
        .append("div");
```

- *insert()* – postavlja element kao prvo dijete nekog roditeljskog čvora.

```
d3.select("body")
        .insert("div")
```

- *html()* - postavlja tekst između oznaka (tagova) novog elementa, npr. tekst kod naslova ili paragrafa (innerHTML).

```
d3.select("body")
        .append("h1")
        .html("Naslov 1");
```

- *text()* - kao i *html()*, postavlja tekst izmedu oznaka (tagova) novog elementa, ali se primjenjuje za dodavanje teksta unutar SVG elemenata. Razlika u nekim preglednicima neće postojati, ali kod drugih će pogrešno mjesto primjene metoda *text()* i *html()* rezultirati neispravnim prikazivanjem teksta.

1.3.5 Anonimne funkcije

Javascript dozvoljava upotrebu tzv. *anonimnih funkcija*, odnosno funkcija bez naziva. Riječ je o funkcijama koje se koriste samo na mjestu gdje su napisane, tj. ne pozivaju se iz ostatka programa. Potreba za njihovom primjenom javlja se kada je potrebno obaviti istu radnju nad skupom podataka.

Kao argumente može imati vrijednosti:

- d - prvi parametar; predstavlja podatak
- i - drugi parametar; indeks (redni broj) podatka
- j – treći parametar.

```
var fs=["10px","20px","30px"];
d3.selectAll("p")
        .data(fs)
        .style("font-size",function(d) {return d;});

d3.selectAll("p")
        .style("font-size",function(d,i){return 10*(i+1)+"px";});

//oba primjera, ako postoje samo tri paragrafa, daju isti rezultat,
↪  tj. veličina slova unutar paragrafa će biti redom 10, 20 i 30
↪  px.
```

1.3.6 Metode enter(), update() i exit()

Dodavanje i izmjena elemenata u ovisnosti o podacima obavlja se pomoću metoda *enter()*, *update()* i *exit()*:

- *enter()* - vraća "spremnike" za nepostojeće elemente koji će se dodati (append).
- *update()* - običan *select()* ili *selectAll()*.
- *exit()* - vraća elemente za koje nema podataka, tj. koji su višak. Često se koristi u paru s funkcijom *remove()* kako bi se višak elemenata obrisao.

Metode *enter()* i *exit()* koriste se nakon pozivanja metode *.data()*, a ne nakon metode *.select()* jer iste ovise o broju elemenata i broju podataka, odnosno ovise o tome postoje li elementi za nove podatke i ima li podataka za neke postojeće elemente.

```
//dohvat tijela dokumenta i svih div elemenata
//pridruživanje podataka div elementima
//dodavanje novih div elemenata i pripadajućeg teksta
d3.select("body").selectAll("div")
        .data([4, 8, 15, 16, 23, 42])
        .enter()
        .append("div")
        .html(function(d) { return d; });
```

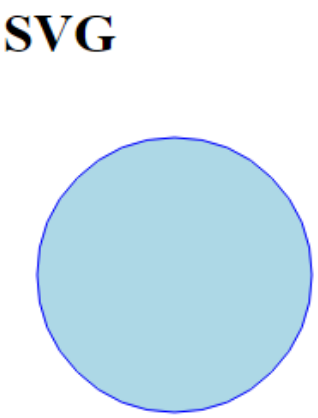
Dodatne informacije:

- Three Little Circles - http://mbostock.github.io/d3/tutorial/circle.html
- Selections - https://github.com/mbostock/d3/wiki/Selections
- Mike Bostock - http://bost.ocks.org/mike/d3/workshop/
- How Selections Work - http://bost.ocks.org/mike/selection/

1.4 SVG

SVG (engl. *Scalable Vector Graphics*) je vektorski zapis slike koji se koristi za izradu dvodimenzionalne grafike, a zasnovan je na XML jeziku (engl. *Extensible Markup Language*) te podržava interakciju i animacije. SVG zapis je podržan u većini modernih web preglednika.

```
<!DOCTYPE html>
<html>
 <body>
  <h1>SVG</h1>
   <svg width="200" height="200">
    <circle cx="100" cy="100" r="80" stroke="blue" stroke-width="1"
     ↪ fill="lightblue" />
   </svg>
 </body>
</html>
```



Slika 1.5: Primjer SVG elementa

1.4.1 SVG koordinatni sustav

Koordinatni sustav SVG dokumenata ishodište ima u gornjem lijevom kutu dokumenta pa je tako točka (0, 0) smještena lijevo i gore dok je točka (100, 100) smještena desno i dolje.

Slika 1.6: SVG koordinatni sustav

1.4.2 Osnovni SVG oblici

Sliku u SVG zapisu moguće je brzo i jednostavno izraditi korištenjem osnovnih SVG oblika koji se zadavanjem pripadajućih parametara vrlo jednostavno prilagođavaju željama korisnika. Na raspolaganju je šest osnovnih oblika i oni se mogu dobiti umetanjem elementa s imenom prema ovim ključnim riječima:

a) rect
b) circle
c) ellipse
d) line
e) polyline
f) polygon

Za prikazivanje pravokutnika koristi se *rect*, a osnovni parametri su mu:

1. 'x' – x položaj, odnosno udaljenost od lijevog ruba
2. 'y' – y položaj, odnosno udaljenost od vrha SVG elementa
3. 'width' – širina pravokutnika
4. 'height' – visina pravokutnika

Kod potreban za iscrtavanje jednostavnog pravokutnika izgledao bi ovako:

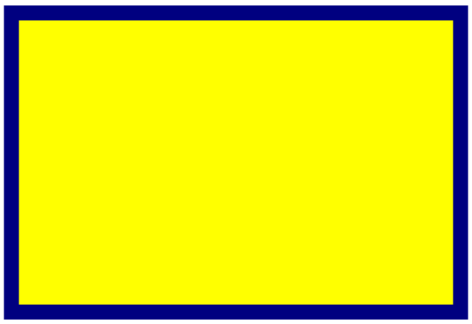
```
<svg width="500" height="500">
 <rect x="100" y="100" width="300" height="200"></rect>
</svg>
```

Ako se želi izmijeniti boje i obrub, potrebno je zadati vrijednosti parametara. Kod potreban za iscrtavanje jednostavnog pravokutnika žute boje s plavim obrubom izgledao bi ovako:

```
<svg width="500" height="500">
 <rect x="100" y="100" width="300" height="200" fill="yellow" stroke="navy"
  ↪  stroke-width="10"></rect>
</svg>
```

Biblioteka D3.js osim što pruža mogućnost manipulacije HTML elementima, omogućava jednostavnu i učinkovitu izradu interaktivnih vizualizacija izradom SVG elemenata te manipulacijom nad njima (dodavanje, brisanje i izmjena). Prethodne primjere moguće je napraviti i pomoću biblioteke D3.js:

Slika 1.7: Primjer jednostavnog pravokutnika



Slika 1.8: Modificirani pravokutnik

```
<!DOCTYPE html>
<html>
 <head>
  <script src="http://d3js.org/d3.v3.min.js" charset="utf-8"></script>
 </head>

 <body>
  <h1>SVG</h1>
  <script>
    var svg = d3.select("body")
                .append("svg")
                .attr("width","200")
                .attr("height","200");

        svg.append("circle")
                .attr("cx","100")
                .attr("cy","100")
                .attr("r","80")
                .attr("style","stroke:blue; stroke-width:1; fill:lightblue;");
  </script>
 </body>
```

```
</html>
```

Za izradu SVG pravokutnika može se upotrijebiti:

```
svg.append("rect")
      .attr("x","10")
      .attr("y","10")
      .attr("width","40")
      .attr("height","90")
      .attr("style","stroke:blue;stroke-width:1;fill:lightblue;");
```

Slika 1.9: SVG pravokutnik koristeći D3.js

Literatura i korisne poveznice:

1. http://www.w3schools.com/svg/
2. https://www.dashingd3js.com/svg-basic-shapes-and-d3js
3. http://en.wikipedia.org/wiki/Scalable_Vector_Graphics

2 Formatiranje podataka, JSON, jednostavne vizualizacije

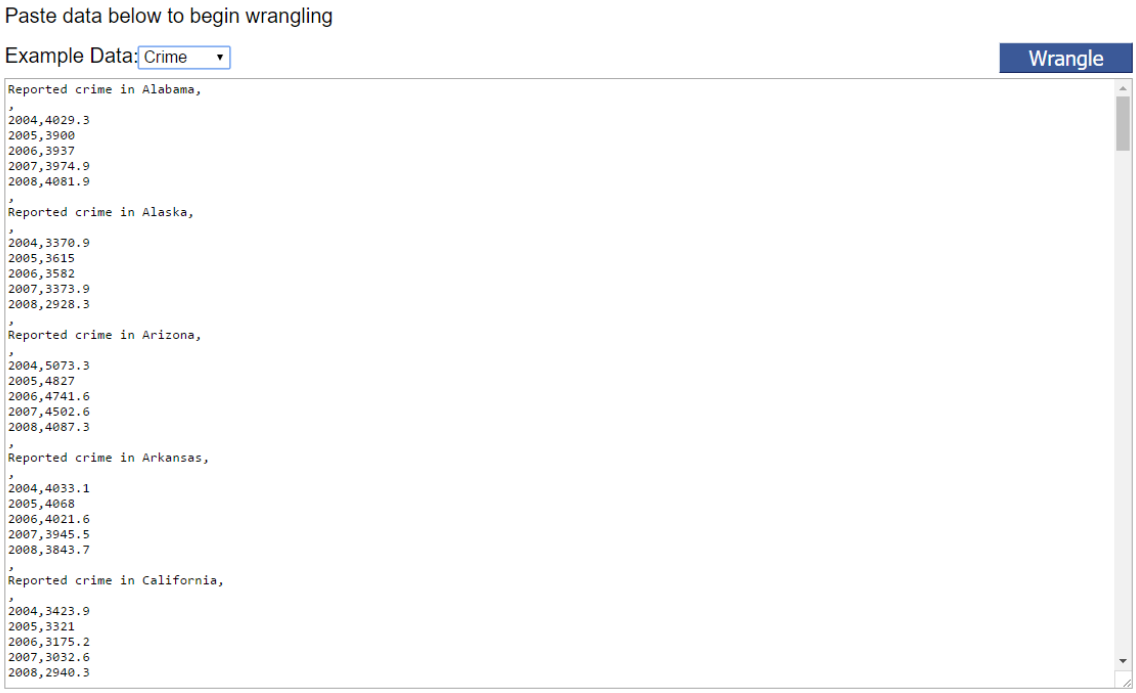
2.1 DataWrangler

DataWrangler je interaktivni alata za transformaciju i pročišćavanje velike količine podataka. Dostupan je na adresi http://vis.stanford.edu/wrangler/app.

Slika 2.1: Početna stranica DataWrangler aplikacije

Na slici 2.1 vidljiva je početna stranica na kojoj je moguće kopirati neformatirane podatke i pritiskom na *Wrangle* otvara se interaktivno sučelje koje omogućuje manipulaciju podacima, Slika 2.3. Za primjer će se obraditi priložen primjer podataka – Crime.

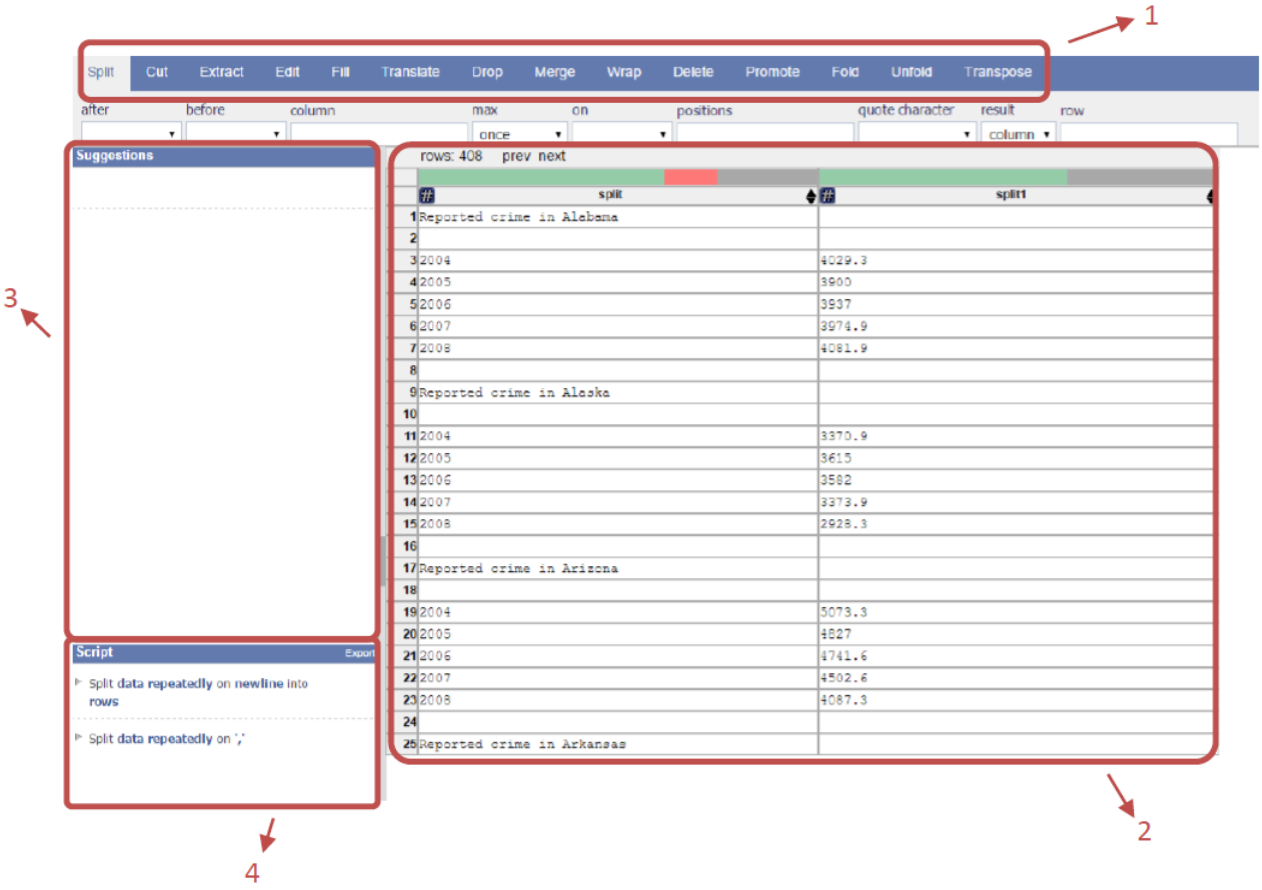
Nakon potvrde podataka otvara se interaktivno sučelje koje omogućuje manipulaciju podacima, Slika 2.3.



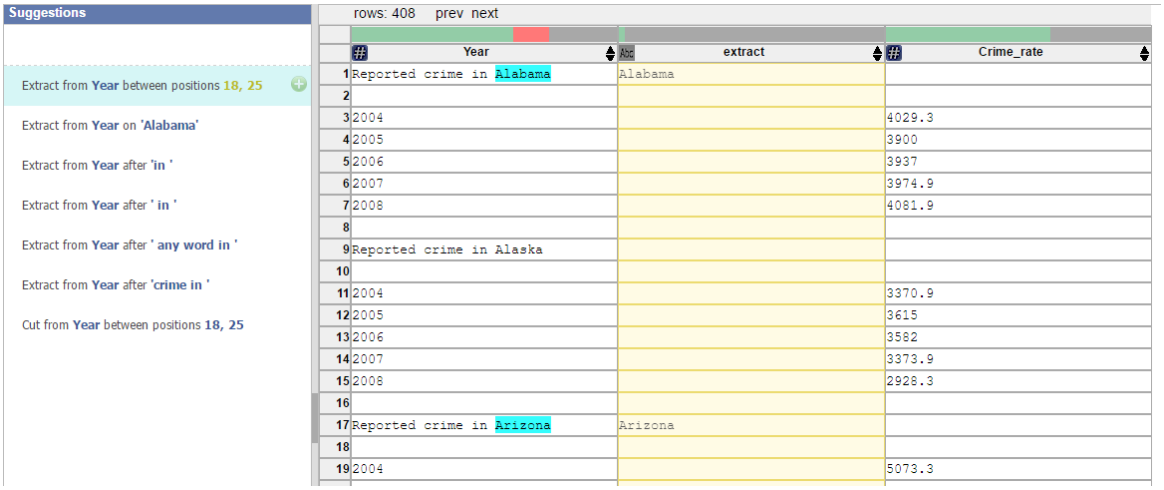
Slika 2.2: Primjer neformatiranih podataka

Na slici 2.3 označeni su pojedini dijelovi DataWrangler aplikacije:

1. Moguća djelovanja nad podacima
2. Uneseni podaci kao *split* i *split1*, prvi redak prikazuje popunjenost tablice
   a) Zelena boja predstavlja količinu važećih i iskoristivih podataka
   b) Crvena boja predstavlja količinu podataka koju nije moguće parsirati
   c) Siva boja predstavlja količinu podataka koja nedostaje
3. Nakon što se odabere podatak unutar tablice aplikacija daje prijedloge za manipulaciju podacima u tablici na temelju odabranih podataka 2.4
   a) Prelaskom miša preko ponuđenih opcija podaci u tablici se automatski formatiraju kako bi korisnik uočio promjene
   b) Promjene se primjenjuju na tablicu tek nakon što korisnik stisne zeleni gumb
4. Prozor s primijenjenim skriptama



Slika 2.3: Manipulacija podacima

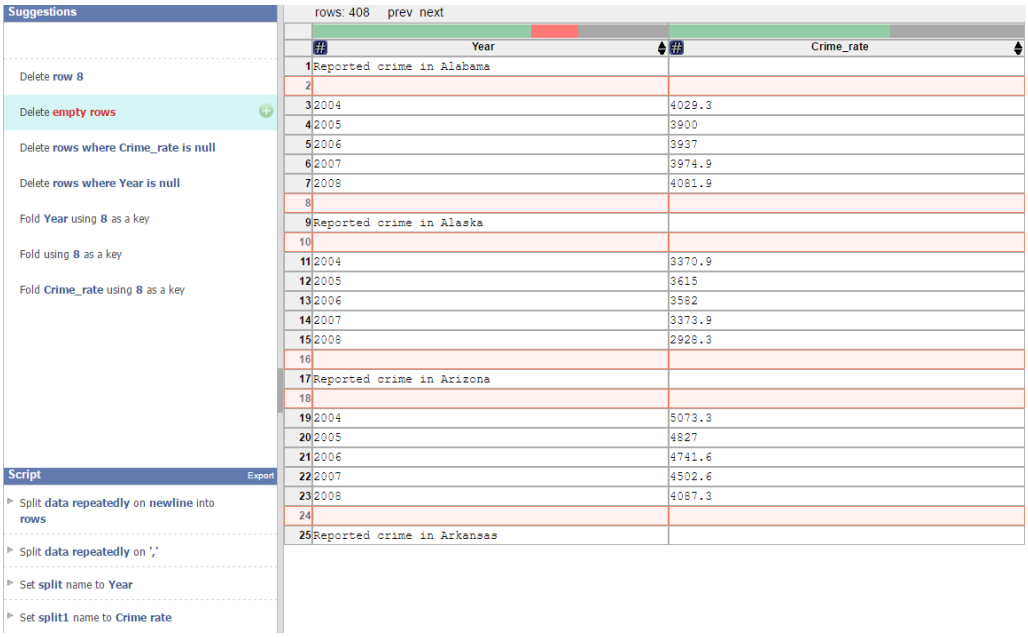


Slika 2.4: Primjer s prijedlozima

2.1.1 Primjer manipulacijom podatka - Crime

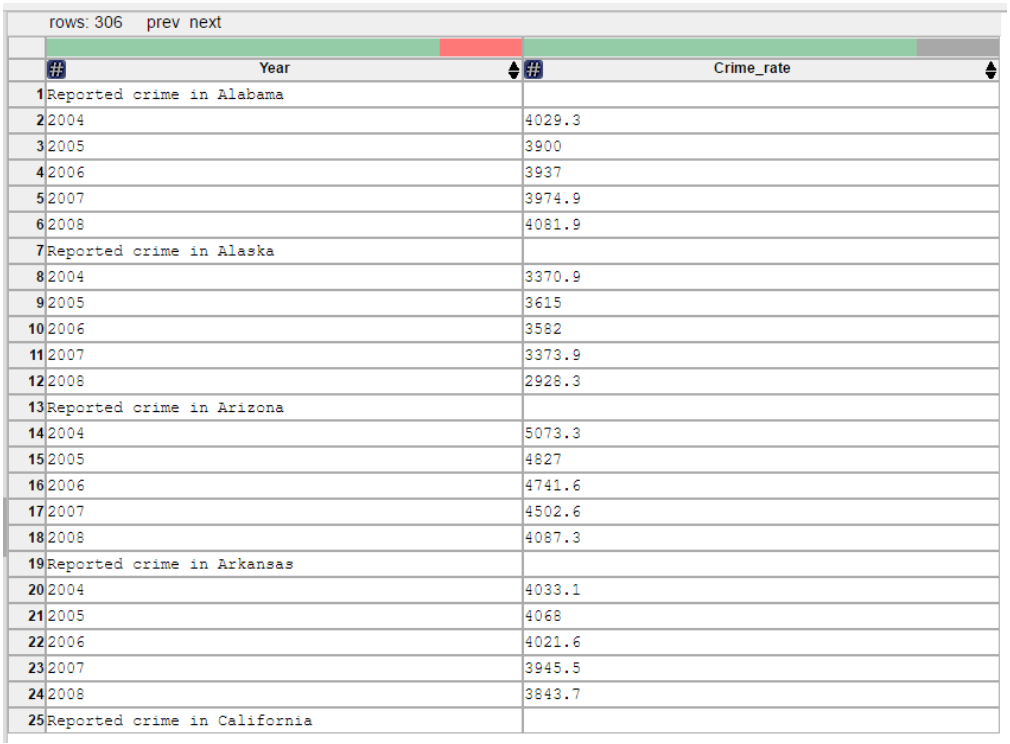
Dvostrukim klikom na *split* i *split1* promijeniti nazive stupaca u Year i Crime_rate. Nakon toga potrebno je obrisati prazne retke – odabrati 8. redak te na prijedlozima primijeniti Delete empty

rows, Slika 2.5.



Slika 2.5: Prijedlog nakon odabiranja *Delete empty rows*

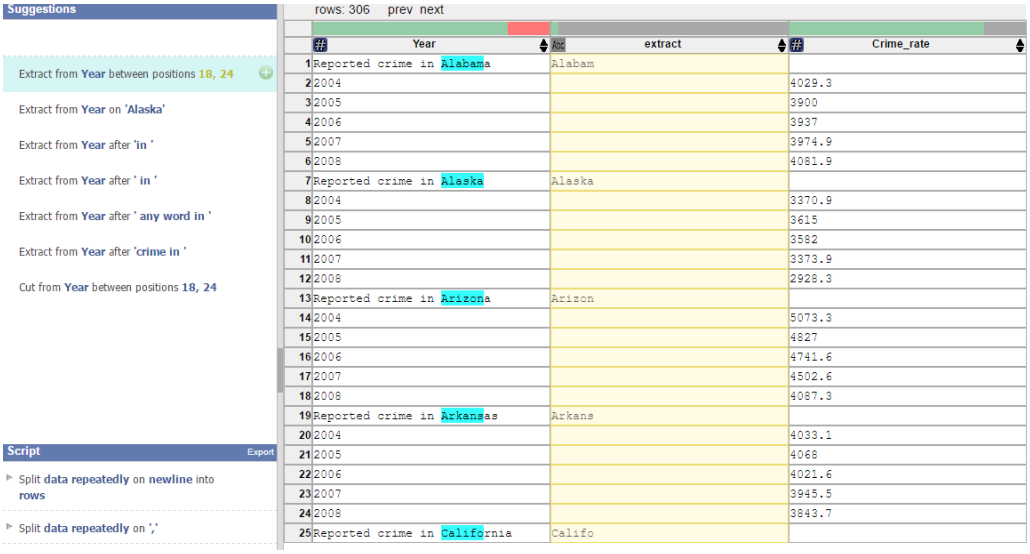
Na Slici 2.6 vidljiv je rezultat nakon primjene Delete empty rows.



Slika 2.6: Obrisani prazni retci u cijeloj tablici

U sljedećem koraku potrebno je izdvojiti imena država iz prvog stupca na taj način da se označi država *Alaska*, na što nam DataWrangler daje prijedlog manipulacije *Extract* što je vidljivo na Slici

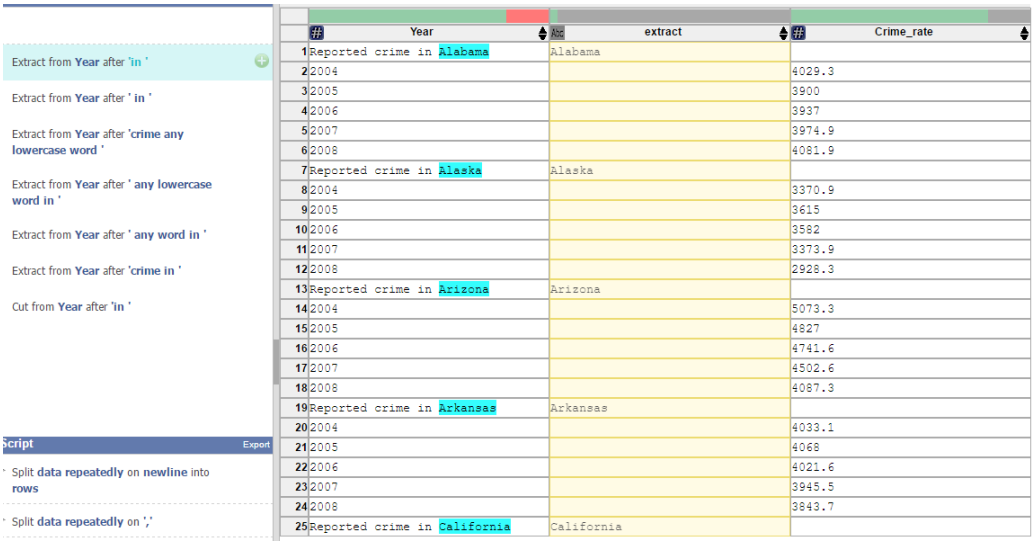
2.7. Kako bi DataWrangler generalizirao odabir i obavio točnije izdvajanje država, potrebno je označiti neku drugu državu, npr. *Arizona*.



	Year	extract	Crime_rate
1	Reported crime in Alabama	Alabam	
2	2004		4029.3
3	2005		3900
4	2006		3937
5	2007		3974.9
6	2008		4081.9
7	Reported crime in Alaska	Alaska	
8	2004		3370.9
9	2005		3615
10	2006		3582
11	2007		3373.9
12	2008		2928.3
13	Reported crime in Arizona	Arizon	
14	2004		5073.3
15	2005		4827
16	2006		4741.6
17	2007		4502.6
18	2008		4087.3
19	Reported crime in Arkansas	Arkans	
20	2004		4033.1
21	2005		4068
22	2006		4021.6
23	2007		3945.5
24	2008		3843.7
25	Reported crime in California	Califo	

Slika 2.7: Prijedlog operacije nakon označavanja imena grada *Alaska*

Nakon odabira drugog grada, DataWrangler je shvatio da želimo izdvojiti imena država te predlaže sljedeću operaciju: „Extract from Year after 'in ' “, što je vidljivo na Slici 2.8.



	Year	extract	Crime_rate
1	Reported crime in Alabama	Alabama	
2	2004		4029.3
3	2005		3900
4	2006		3937
5	2007		3974.9
6	2008		4081.9
7	Reported crime in Alaska	Alaska	
8	2004		3370.9
9	2005		3615
10	2006		3582
11	2007		3373.9
12	2008		2928.3
13	Reported crime in Arizona	Arizona	
14	2004		5073.3
15	2005		4827
16	2006		4741.6
17	2007		4502.6
18	2008		4087.3
19	Reported crime in Arkansas	Arkansas	
20	2004		4033.1
21	2005		4068
22	2006		4021.6
23	2007		3945.5
24	2008		3843.7
25	Reported crime in California	California	

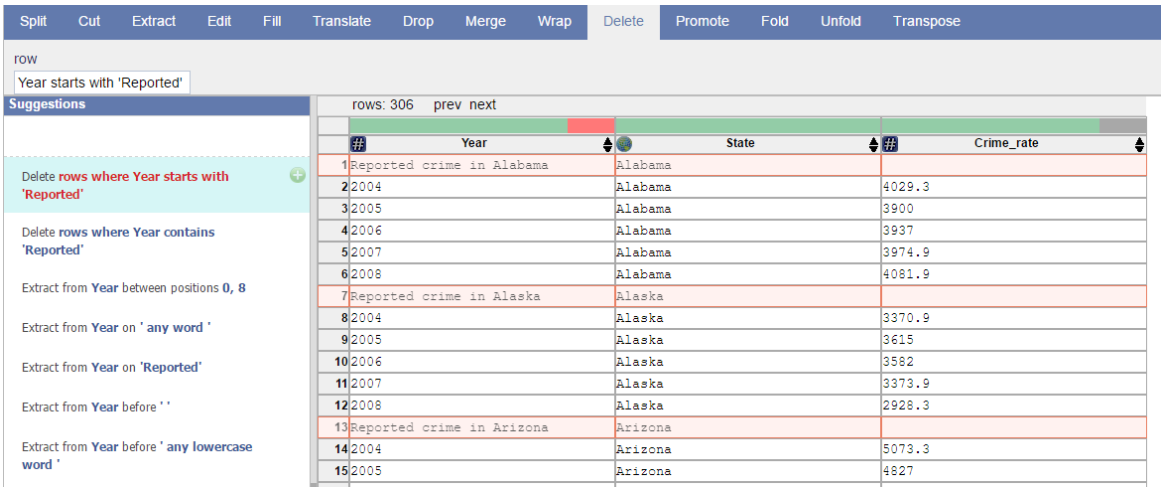
Slika 2.8: Prijedlog operacije nakon označavanja imena drugog grada

Pritiskom na predloženu operaciju, države se izdvajaju u novi stupac kojem je potrebno promijeniti ime iz *extract* u State. Budući da je to slabo popunjena tablica potrebno je popuniti prazne ćelije pritiskom na sivo područje u prvome retku stupca State te tada DataWrangler nudi neke opcije, a odabiremo „Fill State with values from above“ čiji je rezultat vidljiv na Slici 2.9.

U sljedećem koraku potrebno je obrisati retke u kojima se ne nalaze godine, a to se radi na taj način da se označi riječ *Reported* i odabere opcija *Delete* na gornjem izborniku te od ponuđenih operacije primjeni se „Delete rows where Year starts with 'Reported', vidljivo na Slici 2.10.

	Year	State	Crime_rate
1	Reported crime in Alabama	Alabama	
2	2004	Alabama	4029.3
3	2005	Alabama	3900
4	2006	Alabama	3937
5	2007	Alabama	3974.9
6	2008	Alabama	4081.9
7	Reported crime in Alaska	Alaska	
8	2004	Alaska	3370.9
9	2005	Alaska	3615
10	2006	Alaska	3582
11	2007	Alaska	3373.9
12	2008	Alaska	2928.3
13	Reported crime in Arizona	Arizona	
14	2004	Arizona	5073.3
15	2005	Arizona	4827
16	2006	Arizona	4741.6
17	2007	Arizona	4502.6
18	2008	Arizona	4087.3
19	Reported crime in Arkansas	Arkansas	
20	2004	Arkansas	4033.1
21	2005	Arkansas	4068
22	2006	Arkansas	4021.6
23	2007	Arkansas	3945.5
24	2008	Arkansas	3843.7
25	Reported crime in California	California	

Slika 2.9: Rezultat nakon kopiranja imena država

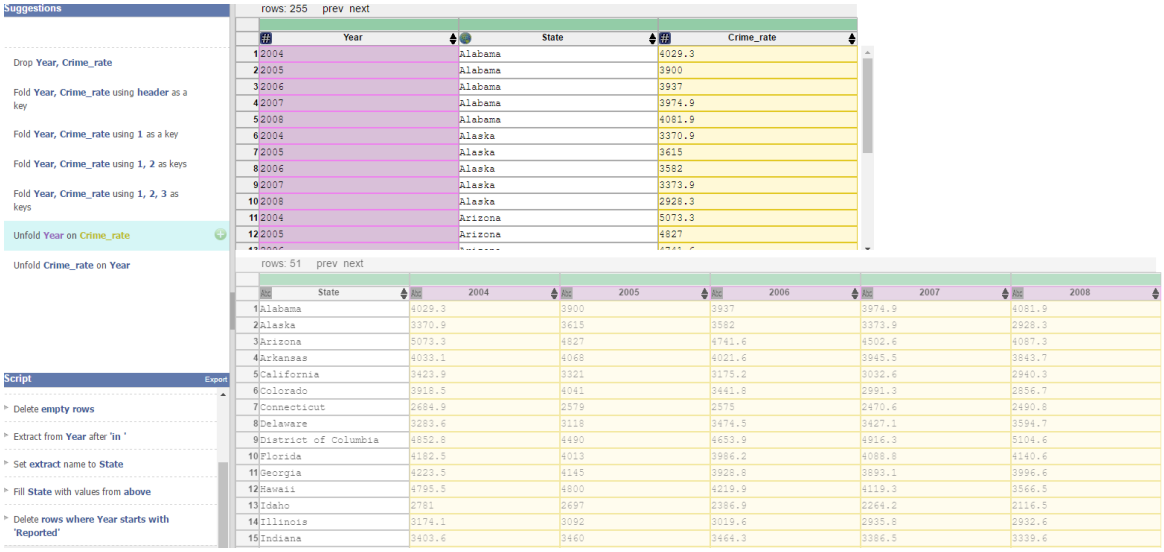


Slika 2.10: Način brisanja unosa

Nakon primjene navedene operacije u donjem dijelu ekrana nazire se gotovo upotrebljiva tablica, no potrebno je napraviti još jednu operaciju kako bi tablica bila upotrebljiva. Potrebno je označiti prvi i treći stupac (Year, Crime_rate) te od opcija odabrati „Unfold Year on Crime_rate“, Slika 2.11, a čiji je rezultat vidljiv na Slici 2.12. Na Slici 2.12 može se uočiti Export pomoću kojeg dobivamo upotrebljive podatke za vizualizaciju.

Nakon obavljene manipulacije podacima, DataWrangler omogućuje izvoz podataka u nekoliko tipova podataka:

- CSV – Comma separated values



Slika 2.11: Odabir željenih podataka za konačni prikaz

Slika 2.12: Uređena tablica

- TSV – Tab separated values
- Row-oriented JSON
- Column-oriented JSON
- Lookup table

Projekt DataWrangler više nije aktivan te je prešao u novi komercijalni program naziva Trifacta: http://www.trifacta.com/

Vrlo dobar tutorial za rad s DataWrangler-om: http://www.visualcinnam on.com/2013/11/data-wrangler-tutorial-turning -ugly.html

Alat za pretvaranje CSV, TSV, pa čak i xls tablice u JSON: http://shancar nter.github.io/mr-data-converter/

2.2 JSON

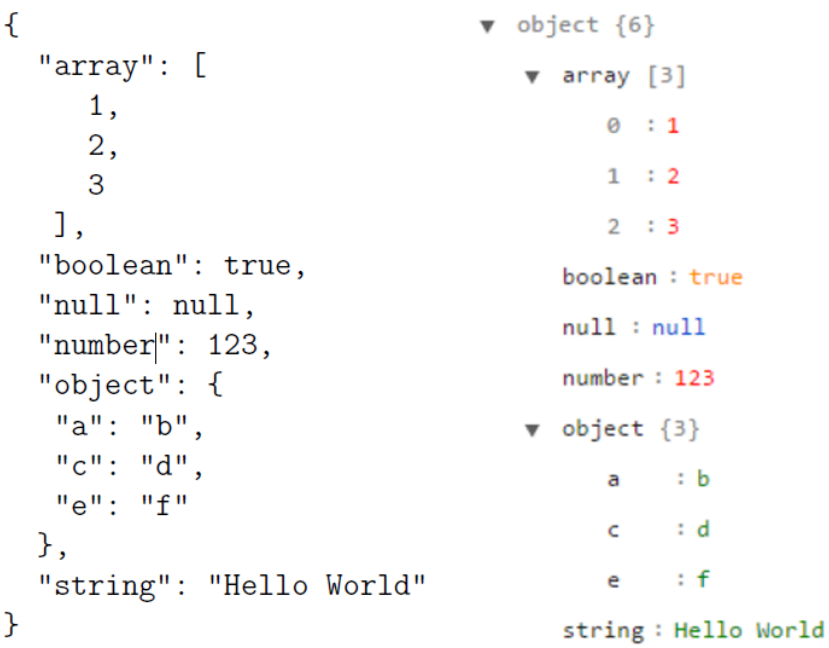
JSON ili JavaScript Object Notation – lako čitljiv format za prijenos podataka koji se sastoje od para *atribut-vrijednost*. Koristi se kao alternativa XML-u.

Osnovni tipovi JSON podataka su:

- Broj – decimalni broj koji može koristiti E notaciju
- String – niz unicode znakova unutar dvostrukih navodnika
- Boolean – true ili false.
- Polje – lista nekoliko vrijednosti koje mogu biti bilo kojeg tipa. Koriste uglate zagrade, a elementi se odvajaju zarezom
- Objekt – lista parova ime-vrijednost unutar vitičastih zagrada, elementi također odvojeni zarezom
- null

Primjer objekata s više elemenata:

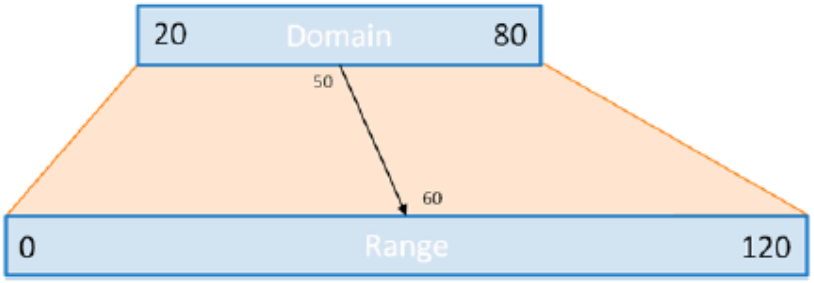
```
var djelatnici = [
   {"ime":"Josip", "prezime":"Job"},
   {"ime":"Hrvoje", "prezime":"Leventic"},
   {"ime":"Kresimir","prezime":"Romic"}
]

djelatnici[1].ime + \ \ + djelatnici[1].prezime -> vraća vrijednost Hrvoje
↪  Leventic
```



Provjeru ispravnosti JSON zapisa možete napraviti na stranici:
https://www.jsoneditoronline.org/

2.3 Skale

Prilikom izrade vizualizacije uobičajeni problem jest nepodudaranje dimenzija SVG elementa i podataka koji se žele prikazati. Primjerice, ako SVG element ima širinu 600 px, a visinu 200 px, jednostavno je prikazati deset podataka čije se vrijednosti kreću od 0 do 500 pomoću stupčastog grafa kojemu će visina svakog stupca u pikselima prikazivati vrijednost pojedinog podatka jer se dimenzije SVG elementa podudaraju s rasponom podataka. Međutim, ako se te vrijednosti kreću od 0 do 1000 ili od 0 do 1, tada je potrebno napraviti izračun odgovarajuće vrijednosti koja će predstavljati podatak u koordinatnom sustavu SVG elementa, tj. željene visine stupaca. Rješenje problema je jednostavna funkcija koja će za domenu imati minimalnu i maksimalnu vrijednost skupa podataka, a za kodomenu minimalnu i maksimalnu visinu stupaca izraženu u pikselima.



Slika 2.13: Način funkcioniranja skala

Kako bi se ovaj postupak olakšao D3 nudi gotove funkcije za rad sa skalama. Skale (engl. scales) su funkcije koje podatke iz ulazne domene preslikavaju u odgovarajući izlazni raspon pa su tako na raspolaganju:

- Kvantitativne skale
- Ordinalne ili redne skale
- Vremenske skale

2.3.1 Kvantitativne skale

Za gore navedeni primjer prikladna je kvantitativna linearna skala:

```
d3.scale.linear()
```

Navedena naredba će napraviti novu skalu s domenom [0, 1] i kodomenom [0, 1]. Kako je u primjerima redom navedeno, odgovarajuće skale bile bi:

```
var skala1 = d3.scale.linear()
  .domain([0, 500])
  .range([0, 500]);
```

```
var skala2 = d3.scale.linear()
 .domain([0, 1000])
 .range([0, 500]);

var skala3 = d3.scale.linear()
 .domain([0, 1])
 .range([0, 500]);
```

Svaka od njih će određenu vrijednost iz ulaznog skupa preslikati u odgovarajuću vrijednost iz izlaznog raspona.

```
var x1 = skala1(250); //x1 postaje 250;
var x2 = skala2(500); //x2 postaje 250;
var x3 = skala3(0.5); //x3 postaje 250;
```

Ako se pak želi dobiti inverznu funkciju, potrebno je metodi invert() predati vrijednost iz izlaznog intervala, a ona će vratiti odgovarajuću vrijednost iz ulaznog intervala:

```
var skala3 = d3.scale.linear()
 .domain([0,1])
 .range([0,500]);

var x = skala3.invert(500); //x postaje 1;
```

Navedeno je korisno kada se želi očitati vrijednost prema položaju pokazivača miša iznad same vizualizacije jer je tada poznat podatak u pikselima (izlazni raspon), a koji je potrebno pretvoriti u podatak iz intervala ulaznih vrijednosti. Osim linearnih podržane su i logaritamska skala, skala potencija, kvantizirana skala i skala kvantila.

2.3.2 Ordinalne ili redne skale

Razlika između kvantitativnih i rednih skala je u tome što potonje imaju diskretnu domenu, tj. domena im može biti skup imena ili kategorija. Primjerice ako se želi preslikati vrijednosti [0,1,2,3,4,5] u odgovarajuću skalu od 0 do 100, onda se to može napraviti pomoću ovog koda:

```
var skala1 = d3.scale.ordinal()
 .domain([0, 1, 2, 3, 4, 5])
 .rangePoints([0, 100]);
skala1.range();
//[0, 20, 40, 60, 80, 100]
```

ili pomoću ranije korištenih metoda:

```
var skala1 = d3.scale.ordinal()
 .domain([0, 1, 2, 3, 4, 5])
 .range([0, 20, 40, 60, 80, 100][0, 100]);
```

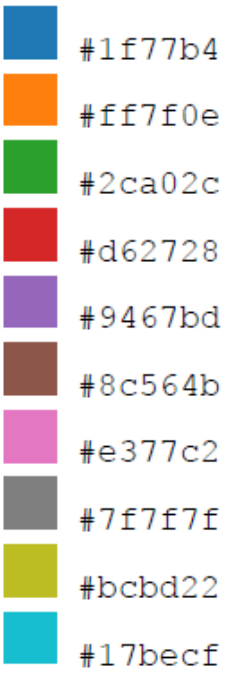
Razlika je što u prvom slučaju dajemo interval izlaznog raspona, a u drugom slučaju dajemo konkretne vrijednosti. Najčešća primjena rednih skala jest kod pridruživanje odgovarajućih RGB komponenata boje nekom podatku.Ugrađene skale za rad s bojama su:

```
d3.scale.category10();
d3.scale.category20();
d3.scale.category20b();
d3.scale.category20c();
```

Prva ugrađena skala

```
d3.scale.category10();
```

napravit će novu rednu skalu s 10 izlaznih kategorija iz skupa:



Slika 2.14: RGB komponenete skale *d3.scale.category10()*

2.3.3 Vremenske skale

Vremenske skale rade s JavaScript objektom Date kao domenom, a proširenje su linearne skale. Detaljnije o radu s vremenom i vremenskim skalama može se pročitati na poveznicama:
https://github.com/mbostock/d3/wiki/Time-Intervals
https://github.com/mbostock/d3/wiki/Time-Scales
Više detalja o svim skalama može se pronaći na:
https://github.com/mbostock/d3/wiki/Scales

2.3.4 Stupčasti grafovi

U prethodnom poglavlju spominju se stupčasti grafovi (engl. bar chart). Njihova implementacija pomoću biblioteke D3 je jednostavna jer se realiziraju pomoću <rect> elemenata koji su pravilno razmaknuti jedan od drugoga, imaju jednaku širinu, a visinu ovisno o podatcima u slučaju vertikalnih stupaca ili jednaku visinu, a širinu ovisnu o podatcima u slučaju horizontalnih stupaca.

Primjer: Izraditi jednostavan stupčasti graf za 10 podataka zadanih u polju.

```
var data = [1, 2, 3 , 4, 5, 6, 7, 8, 9, 10];
var width = 500;
var height = 500;
var svg = d3.select("body")
 .append("svg")
  .attr("width", width)
  .attr("height", height);
var barchart = svg.selectAll("rect")
 .data(data)
  .enter()
 .append("rect")
  .attr("x", function(d, i) { return 50 * i; })
  .attr("y", function(d) { return height - d * 50; })
  .attr("width", 40)
  .attr("height", function(d) { return d * 50; })
  .attr("fill", "blue");
```

Prve tri linije koda inicijaliziraju varijable koje predstavljaju polje podataka, širinu i visinu SVG elementa.

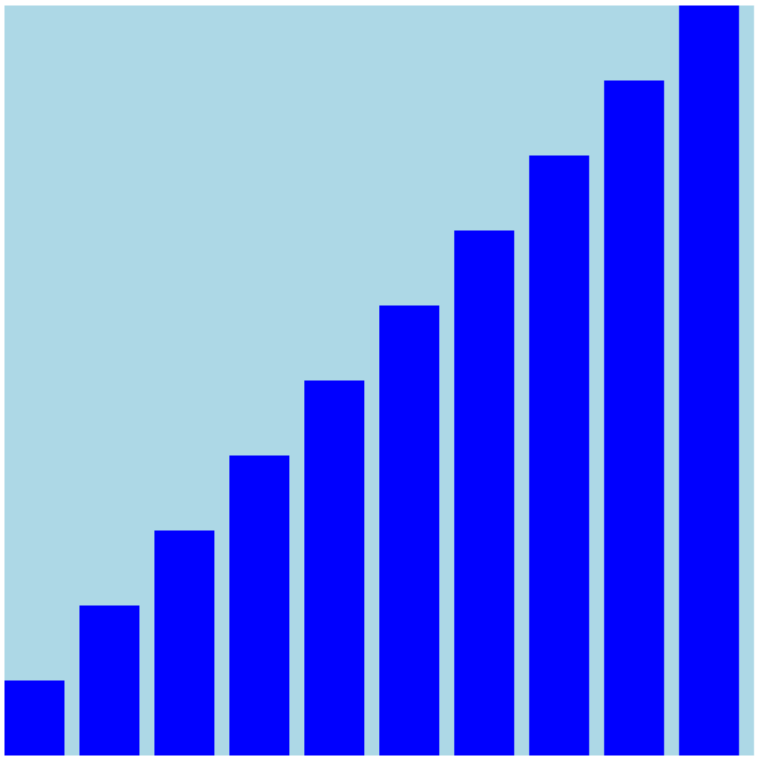
```
var data = [1, 2, 3 , 4, 5, 6, 7, 8, 9, 10];
var width = 500;
var height = 500;
```

Nakon toga inicijalizira se varijabla svg koja predstavlja D3 odabir (selektor) SVG elementa, tj. radi se odabir tijela dokumenta (body) te se u njega umeće SVG element, a potom se poziva metoda *attr()* pomoću koje se definiraju širina i visina SVG elementa.

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

Ako se želi dohvatiti SVG element pohranjen u svg varijabli, to je moguće napraviti preko indeksa, ali kako selektor pohranjuje čvorove kao polje unutar polja, potrebno je navesti oba indeksa ([0][0]):

Varijabla *barchart* je selektor svih pravokutnika *(svg.selectAll("rect"))*. Svakom podatku za koji ne postoji <rect> element, a to su svi podaci jer unutar SVG elementa ne postoje drugi elementi, pridružit će se po jedan <rect> element. <rect> elementima pristupiti se može preko indeksa elementa unutar polja, tj. barchart[0][0] dohvaća prvi pravokutnik, a barchart[0][9] vraća deseti pravokutnik.



Slika 2.15: Primjer stupčastog grafa

```
> svg
< [▶ Array[1]]
> svg
< [▼ Array[1] i                ]
    ▶ 0: svg
      length: 1
    ▶ parentNode: html
    ▶ __proto__: Array[0]
> svg[0]
< [  <svg width="500" height="500"></svg>]
> svg[0][0]
<   <svg width="500" height="500"></svg>
```

```
var barchart = svg.selectAll("rect")
 .data(data)
 .enter()
.append("rect")
 .attr("x", function(d, i) { return 50 * i; })
 .attr("y", function(d) { return height - d * 50; })
 .attr("width", 40)
 .attr("height", function(d) { return d * 50; })
 .attr("fill", "blue");
```

Metode *attr()* koriste se za postavljanje vrijednosti parametara <rect> elemenata i to redom x i y koordinata, širine, visine i boje ispune. Za izračun horizontalnog položaja koristi se indeks svakog elementa koji je pomnožen brojem 50, dakle lijevi bridovi pravokutnika smješteni su na mjesta 0, 50, 100, 150, 200, 250, 300, 350, 400, 450 i 500. Vertikalni položaj izračunava se tako što se od ukupne visine (maksimalne visine) oduzima vrijednost podatka pomnožena brojem 50, tj. pomoću izraza height – d * 50. Visina pojedinog elementa je dobivena množenjem vrijednosti podatka i broja 50, tj. izrazom d * 50. Boja ispune je plava ("blue").

Ovaj primjer je jednostavan način rješavanja problema i ne koristi sve funkcionalnosti koje nudi D3. Ranije su spominjane skale koje omogućavaju preslikavanje iz skupa podataka u odgovarajući skup iz zadanog. Isto tako prikladno bi bilo dodati i koordinatne osi kako bi se omogućilo očitanje vrijednosti koje graf prikazuje. Kako prvotni primjer zauzima cijelu površinu SVG elementa poželjno je graf smanjiti kako bi se mogle dodati koordinatne osi. Uobičajeni pristup jest da se cijeli graf, odnosno svi pravokutnici, stave unutar <g> elementa te da se onda taj element translatira prema dolje i udesno. Kako bi kod bio čitljiv, uvodi se objekt *margin* koji sadrži vrijednosti sve četiri margine, a za čije vrijednosti se umanjuje veličina grafa. Potrebno je uvesti i varijable barWidth i barPadding koje predstavljaju širinu stupaca i razmak između njih.

```
var margin = {top: 20, bottom: 70, left:40, right: 20};
var width = 500 - margin.left - margin.right;
var height = 500 - margin.top - margin.bottom;
var barPadding = 4;
var barWidth = width / data.length - barPadding;
```

Odgovarajuće skale za prethodni primjer su:

```
var x = d3.scale.ordinal()
 .domain(d3.range(data.length))
 .rangeRoundBands([0, width]);

var y = d3.scale.linear()
 .domain([0, d3.max(data)])
 .range([height, 0]);
```

Dodavanje i translatiranje SVG elementa nešto je izmijenjeno pa sada izgleda:

```
var svg = d3.select("body")
 .append("svg")
  .attr("width", width + margin.left + margin.right)
  .attr("height", height + margin.bottom + margin.top)
  .style("background-color", "lightblue")
 .append("g")
  .attr("transform", "translate(" + margin.left + "," + margin.top +
  ↪  ")");
```

Za umetanje koordinatnih osi potrebno je:

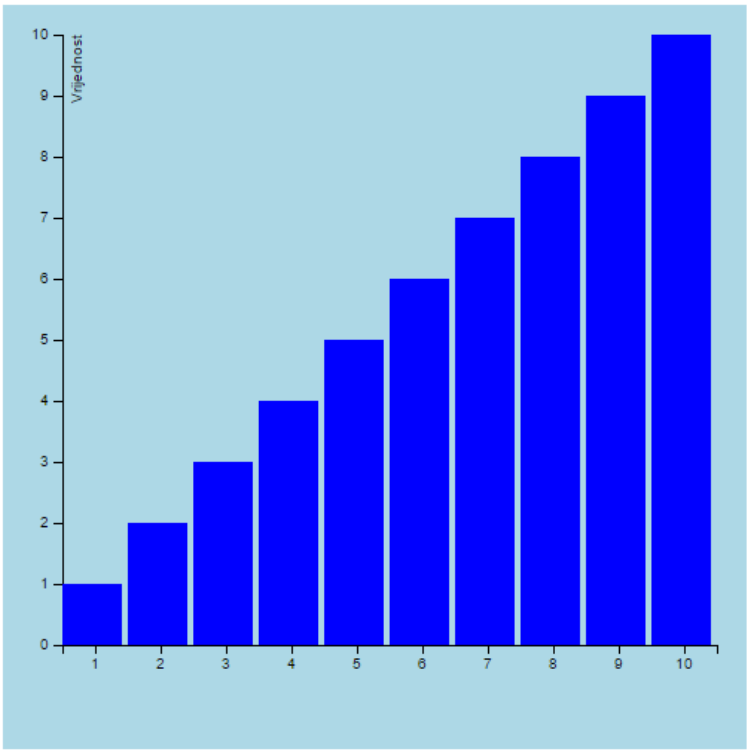
```
var xAxis = d3.svg.axis()
 .scale(x)
 .orient("bottom")
 .tickFormat(function(d, i) { return i + 1; });

var yAxis = d3.svg.axis()
 .scale(y)
 .orient("left")
 .ticks(10);

svg.append("g")
 .attr("class", "x axis")
 .attr("transform", "translate(0," + height + ")")
 .call(xAxis)
 .selectAll("text")
 .style("text-anchor", "middle");

svg.append("g")
 .attr("class", "y axis")
 .call(yAxis)
.append("text")
 .attr("transform", "rotate(-90)")
 .attr("y", 6)
 .attr("dy", ".71em")
 .style("text-anchor", "end")
 .text("Vrijednost");
```

Napravljene su i promjene u dijelu koda koji radi s <rect> elementima pa sada izgleda ovako:

```
var barchart = svg.selectAll("rect")
 .data(data)
 .enter()
.append("rect")
 .attr("x", function(d, i) { return x(i); })
 .attr("y", y) .attr("height", function(d) { return height - y(d); })
 .attr("width", barWidth)
 .attr("fill", "blue");
```

2.3.5 Linijski grafovi

Sljedeća vrsta grafa koja će se obraditi u okviru ove vježbe je linijski graf. Linijski graf može se izraditi iz istih podataka kao i stupčasti, ali sam način implementacije ipak se ponešto razlikuje od prethodnog slučaja. Prvi primjer obuhvatit će izradu jednostavnog linijskog grafa, a nakon toga će se grafu dodati dodatni elementi kako bi se prikazale mogućnosti biblioteke D3. Definiranje varijabli potrebnih za izradu SVG elementa unutar kojega će se iscrtati linijski graf ne razlikuju se od onih iz prethodnog primjera osim što su izostavljene varijable specifične za stupčasti graf:



Slika 2.16: Primjer stupčastog grafa s dodatnim elementima

```
var data = [1, 2, 3 , 4, 5, 6, 7, 8, 9, 10];
var margin = {top: 20, bottom: 70, left:40, right: 20};
var width = 500 - margin.left - margin.right;
var height = 500 - margin.top - margin.bottom;
```

Skala korištena za x-os je izmijenjena pa se sada upotrebljavaju vrijednosti domene u intervalu [0, 9] dobivene pomoću metode *d3.range(data.length)* koja će vratiti vrijednosti iz navedenog intervala, a koje ovise o broju elemenata polja *data*. Raspon točaka kodomene bit će u intervalu [0, width]. Skale sada izgledaju ovako:

```
var x = d3.scale.ordinal()
 .domain(d3.range(data.length))
 .rangePoints([0, width]);

var y = d3.scale.linear()
 .domain([0, d3.max(data)])
 .range([height, 0]);
```

Promjena u dijelovima koda za umetanje SVG elementa i koordinatnih osi nema:

```
var xAxis = d3.svg.axis()
 .scale(x)
 .orient("bottom")
 .tickFormat(function(d, i) { return i + 1; });
```

```
var yAxis = d3.svg.axis()
   .scale(y)
   .orient("left")
  .ticks(10);

var svg = d3.select("body")
  .append("svg")
  .attr("width", width + margin.left + margin.right)
  .attr("height", height + margin.bottom + margin.top)
  .style("background-color", "lightblue")
  .append("g")
  .attr("transform", "translate(" + margin.left + "," + margin.top +
  ↪  ")");

svg.append("g")
  .attr("class", "x axis")
  .attr("transform", "translate(0," + height + ")")
  .call(xAxis)
  .selectAll("text")
  .style("text-anchor", "middle");

svg.append("g")
  .attr("class", "y axis")
  .call(yAxis)
.append("text")
  .attr("transform", "rotate(-90)")
  .attr("y", 6)
  .attr("dy", ".71em")
  .style("text-anchor", "end")
  .text("Vrijednost");
```

Razlika između stupčastog i linijskog grafa je u tome što je stupčasti izveden korištenjem <*rect*> elemenata dok se za linijski koristi element <*path*>. Navedeni element umetnut je pomoću ovih linija koda:

```
var valueline = d3.svg.line()
  .x(function(d, i) { return x(i); })
  .y(function(d) { return y(d); });

var linechart = svg.append("path")
  .attr("class", "line")
  .attr("d", valueline(data))
  .style("stroke", "blue");
```

Metoda *d3.svg.line()* je interpolator za zadani skup vrijednosti, a kao rezultat vraća skup točaka potrebnih za atribut d elementa <*path*>. Dakle metoda line() uzima prvu i drugu vrijednost iz

zadanog skupa i izračunava točke za te dvije vrijednosti, potom uzima drugu i treću vrijednost te izračunava dvije točke. Opisani postupak se ponavlja, a rezultat iz gore navedenog primjera, za slučaj kada se kao polje podataka preda varijabla *data*, je element <path> s odgovarajućom vrijednošću atributa d:

```
<path class="line"
d="M0,369L8.148148148148147,362.1666666666667C16.29629629629629
4,355.3333333333333,32.59259259259259,341.66666666666663,48.888
888888888886,327.99999999999994C65.18518518518518,314.333333333
3333,81.48148148148147,300.66666666666663,97.77777777777777,287
C114.07407407407406,273.3333333333333,130.37037037037035,259.66
666666666663,146.66666666666666,245.99999999999997C162.96296296
296293,232.33333333333331,179.25925925925924,218.66666666666666
,195.55555555555551,205C211.85185185185182,191.33333333333331,2
28.1481481481481,177.66666666666666,244.4444444444444,164C260.7
407407407407,150.33333333333331,277.037037037037,136.6666666666
6666,293.3333333333333,123C309.62962962962956,109.3333333333333
3,325.92592592592587,95.66666666666666,342.2222222222221,81.999
99999999999C358.5185185185185,68.33333333333331,374.81481481481
48,54.66666666666666,391.11111111111103,40.99999999999999C407.4
074074074074,27.33333333333333,423.7037037037037,13.66666666666
6664,431.8518518518518,6.833333333333332L440,0" style="stroke:
blue;"></path>
```

Dobiveni graf je linijski graf plave boje, koordinatne osi s intervalima [0, 9] i [0, 10]:

2.3.6 Atribut fill

Ako vrijednosti polja izmijenimo na neke druge koje ne leže na istom pravcu:

```
var data = [11, 2, 31 , 4, 5, 6, 27, 8, 9, 10];
```

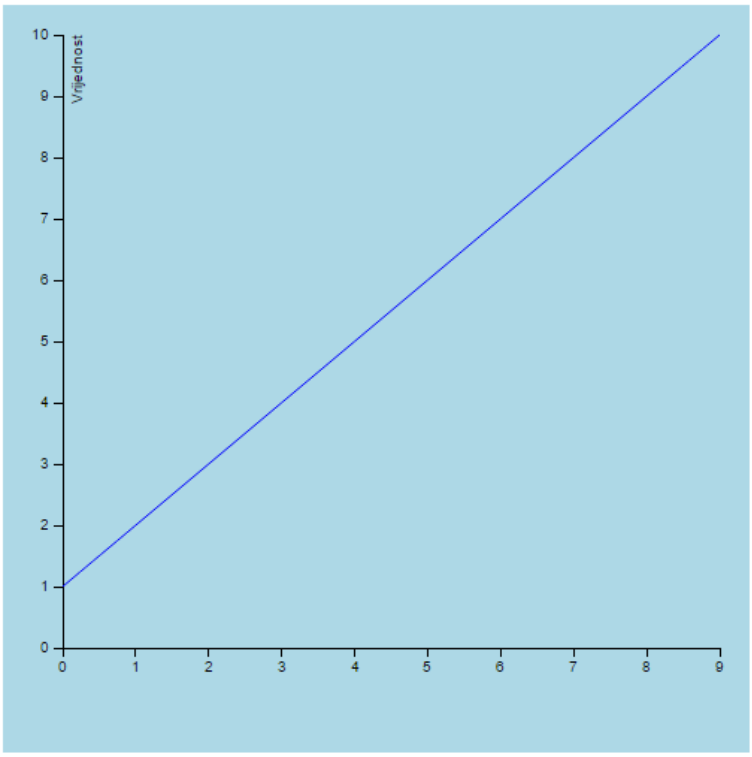
dobiti ćemo neobičan efekt:

Budući da element <*path*> osim stroke atributa ima i fill atribut, a njegova vrijednost nije definirana pa je primijenjena pretpostavljena vrijednost koja iznosi 0 i predstavlja crnu boju dobit će se gore prikazana slika. Jednostavno rješenje spomenutog problema jest postaviti vrijednost atributa *fill* na "none" što će spriječiti neželjeno ponašanje.

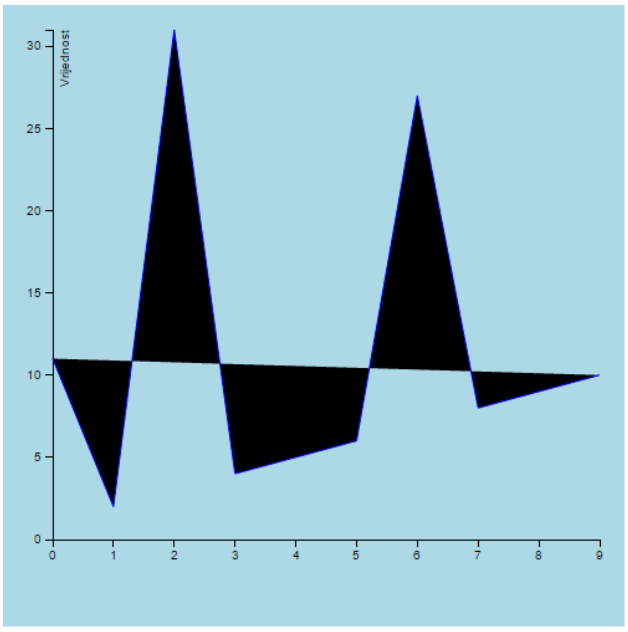
2.3.7 Metode interpolacije

Osim linearne interpolacije između točaka, koja je podrazumijevana metoda za slučaj kada interpolator nije definiran, moguće je koristiti i neke druge interpolacijske metode.

```
var valueline = d3.svg.line()
 .interpolate("linear")
 .x(function(d, i) { return x(i); })
 .y(function(d) { return y(d); });
```



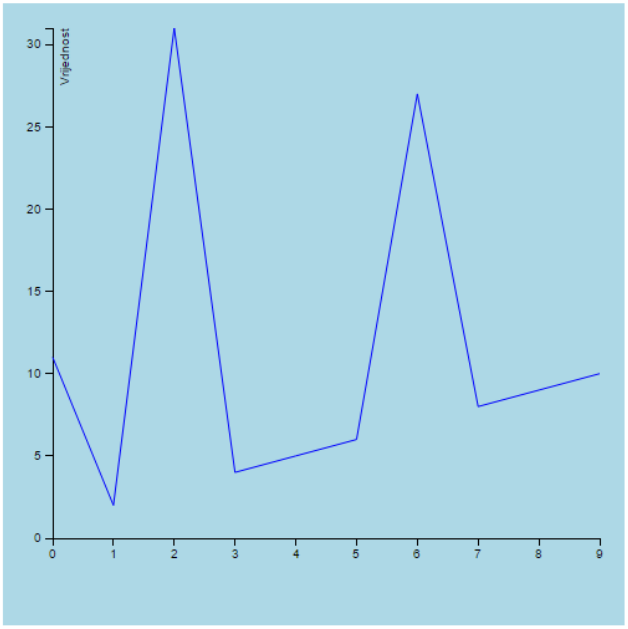
Slika 2.17: Primjer linijskog grafa s dodatnim elementima



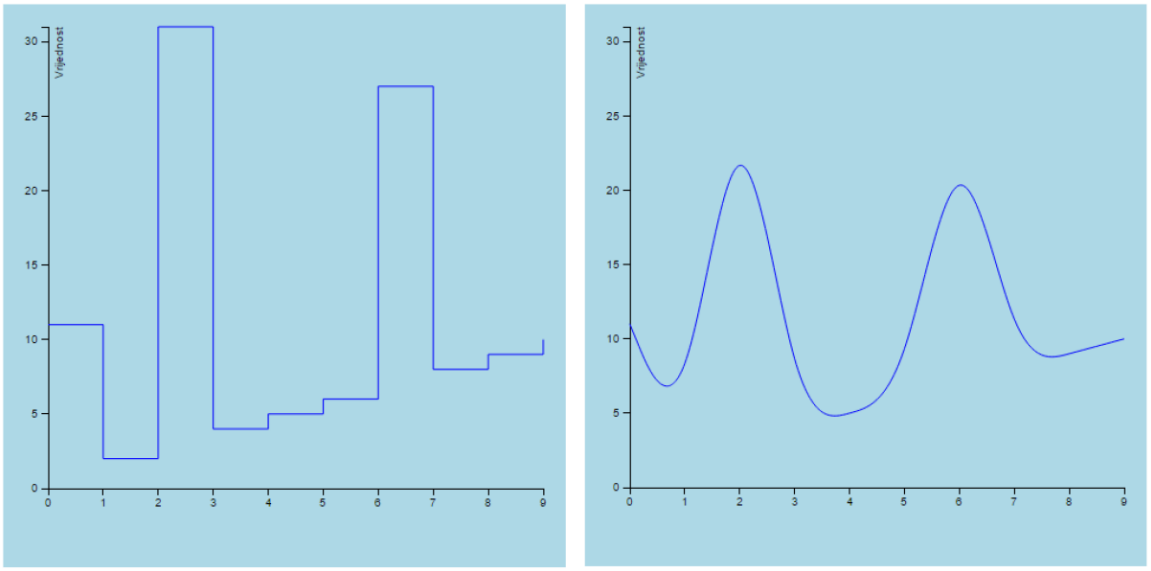
Slika 2.18: Primjer linijskog grafa s izmijenjenim vrijednostima polja

Od velikog broja ugrađenih interpolatora u biblioteci D3 izdvojeni su *step-after* i *basis*.

Prvi će omogućiti izradu stepenastog prikaza podataka, a drugi koristi B-spline kao interpolacijsku metodu. Više detalja o interpolacijskim metodama i elementu <*path*> moguće je pronaći u službenoj



Slika 2.19: Primjer linijskog grafa uz modificiranje fill atributa



Slika 2.20: Step-after i basis interpolacije

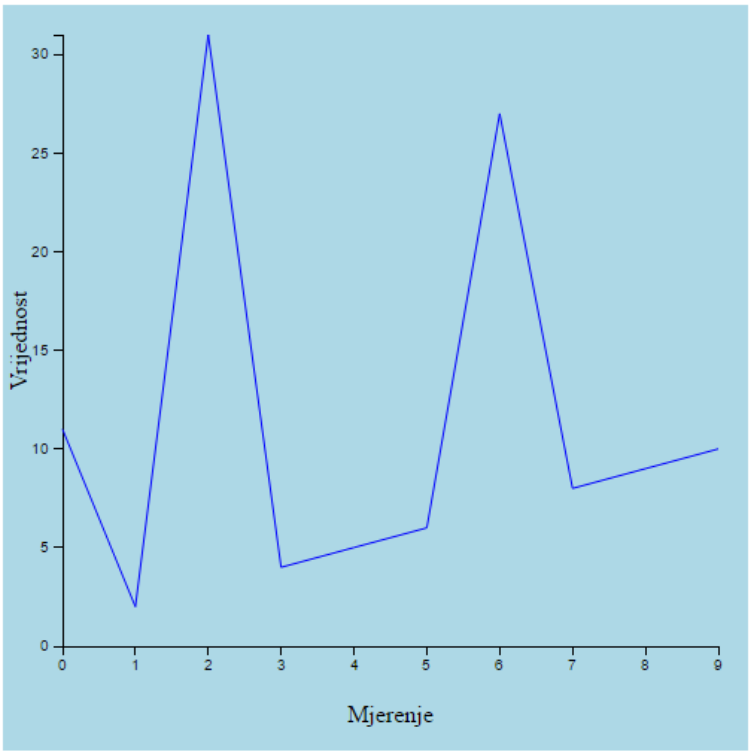
dokumentaciji biblioteke D3.

2.3.8 Nazivi koordinatnih osi

Svaki graf treba imati nazive koordinatnih osi kako bi korisnik mogao razumjeti što taj graf prikazuje. Nazivi koordinatnih osi dodaju se u obliku elementa <*text*> kojeg je potrebno postaviti na odgovarajući položaj unutar SVG slike. U dosadašnjem primjeru za izradu linijskog grafa svi elementi grafa postavljeni su unutar elementa <*g*> koji je translatiran za vrijednosti *margin.left* i *margin.top*. Prva vrijednost predstavlja pomak u smjeru x-osi i ostavlja dovoljno prostora lijevo od grafa za naziv

y-osi, a pomak za *margin.top* spušta cijeli graf prema dolje kako bi se osigurao prostor za naziv grafa. Za smještanje naziva x-osi potrebno je uzeti vrijednost visine samog prikaza grafa, odnosno varijablu *height* kojoj je potrebno dodati (*margin.bottom* / 2) kako bi se naziv smjestio u središtu donje margine. Nije potrebno dodavati *margin.top* jer je element <*g*> već pomaknut za navedenu vrijednost. Tekst naziva y-osi rotiran je za -90° kako bi bio napisan uspravno. Izmijenjeni kod s nazivima koordinatnih osi sada izgleda ovako:

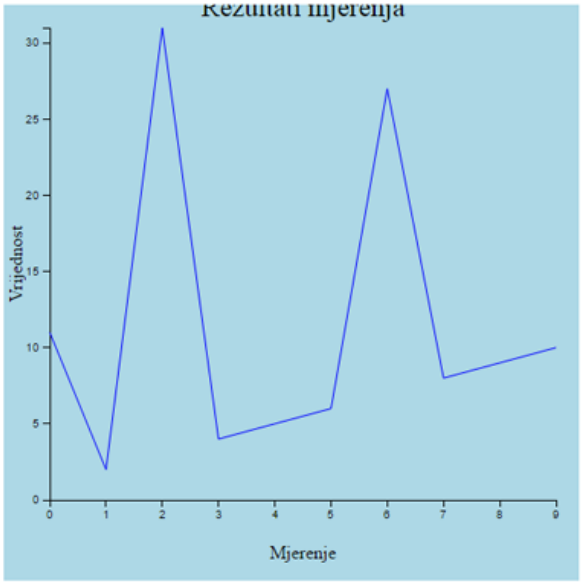
```
svg.append("g")
 .attr("class", "x axis")
 .attr("transform", "translate(0," + height + ")")
 .call(xAxis);

svg.append("text")
 .attr("x", (width / 2))
 .attr("y", (height + (margin.bottom / 2)))
 .attr("dy", "1em")
 .style("text-anchor", "middle")
 .text("Mjerenje");

svg.append("g")
 .attr("class", "y axis")
 .call(yAxis);

svg.append("text")
 .attr("transform", "rotate(-90)")
 .attr("x",0 - (height / 2))
 .attr("y", 0 - margin.left)
 .attr("dy", "1em")
 .style("text-anchor", "middle")
 .text("Vrijednost");
```



Slika 2.21: Linijski graf s dodanim nazivima osi

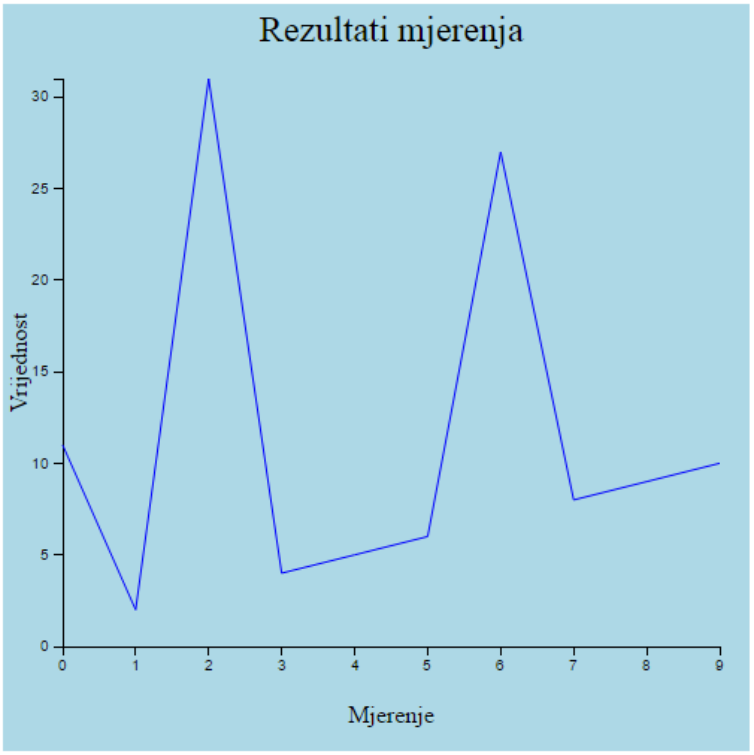
Ako ispod gore navedenog koda dodamo još jedan element <*text*> kojega ćemo pomaknuti prema gore od elementa <*g*> za vrijednost *margin.top* dobit će se tekst naziva grafa na odgovarajućem položaju:

```
svg.append("text")
  .attr("x", (width / 2))
  .attr("y", 0 - (margin.top / 2))
  .attr("text-anchor", "middle")
  .style("font-size", "1.5em")
  .text("Rezultati mjerenja");
```



Slika 2.22: Linijski graf s dodanim nazivima osi i naslovom

Kako bi se ispravila pogreška s odrezanim naslovom grafa, potrebno je izmijeniti vrijednosti gornje margine u varijabli margin čime će se napraviti dovoljno prostora za postavljanje naziva iznad grafa:



Slika 2.23: Linijski graf s dodanim nazivima osi i ispravljenim naslovom

3 Animacija i pridruživanje podataka

3.1 Tranzicije

Tranzicija je proces promjene iz jednog stanja u drugo stanje. U D3.js tranzicija (engl. *transition*) je poseban oblik odabira (engl. *selection*) kod kojega se operatori primjenjuju kroz određeni vremenski period umjesto trenutno. Važno je napomenuti da nije moguće koristiti sve operatore preko tranzicije pa je tako primjerice prvo potrebno dodati novi element (engl. *append*), a tek onda je moguće primijeniti tranziciju:

```
d3.select("body")
 .append("p")
 .html("neki tekst")
 .transition()
 .style("color","red");
```

Gore navedeni odsječak koda dodat će paragraf teksta te će boju izmijeniti iz crne u crvenu. Navedena promjena kreće bez vremenske zadrške (engl. *delay*), a istu je moguće definirati pozivanjem metode *delay(x)* kojoj se predaje parametar x, tj. iznos ms za koji se želi odgoditi početak. Ako se drugačije ne navede, podrazumijevana vrijednost je 0 ms, ali potrebno je imati na umu da stvarna vrijednost iznosi 17 ms zbog mjerenja vremena.

Uobičajeno trajanje promjene iznosi 250 ms, a navedeno se mijenja predajom parametra metodi *duration* s vrijednošću željenog trajanja u ms.

Budući da se tranzicije trebaju obavljati glatko (engl. *smoothly*) potrebno je interpolirati vrijednosti između dva krajnja stanja, početnog i završnog. Ako je riječ o promjeni boje, potrebno je izračunati odgovarajuće boje koje će biti prikazane između dva krajnja stanja kako bi se postigao efekt prijelaza, umjesto da isti bude trenutna promjena. Isto tako, ako je riječ o promjeni položaja, dimenzija ili bilo kojeg drugog atributa, potrebno je izračunati vrijednosti svakog atributa u međukoracima prijelaza. Sve potrebne interpolacijske vrijednosti D3 će izračunati samostalno. D3 ima metodu (ease) koja će na osnovu proteklog vremena odabrati vrijednost atributa kako bi se postigao efekt nelinearnosti transformacija. Uobičajena vrijednost parametra je „cubic-in-out“ koja omogućava usporeni početak i kraj animacije.

```
var colors = d3.scale.category10();
.attr("class", "x axis")
.attr("transform", "translate(0," + height + ")")
.call(xAxis);

var svg = d3.select("body")
 .append("svg")
 .attr("width",1024)
 .attr("height",768)
 .style("background","black");

 var rects = svg.append("rect")
  .attr("x", 100)
  .attr("y", 100)
```

```
        .attr("width", 100)
        .attr("height", 200)
        .style("fill", colors(2));

      d3.select("rect")
        .transition()
        .delay(250)
        .duration(1000)
        .attr("transform", function(d){ return "translate(400,100)"; });
```

Ako se želi translatirati pravokutnik i da ta promjena izgleda glatko, tj. s postepenim ubrzavanjem i usporavanjem, kao parametar ease operatora može se koristiti „linear-in-out“ ili „quad-in-out“, tj. potrebno je prije definiranja atributa pozvati operator ease("quad-in-out")

Podržani parametri su:

- linear – linearno preslikavanje iz vremenske domene u odgovarajuću vrijednost
- poly(k) – potencira vrijeme, t zadanom potencijom, k
- quad – ekvivalentno s poly(2)
- cubic - ekvivalentno s poly(3)
- sin – primjena trigonometrijske funkcije sin
- exp – broj 2 potenciran na vrijednost vremena, t
- circle – četvrtina kružnog odsječka
- elastic(a, p) – simulira elastičnost i može prijeći izvan 0 i 1
- back(s) – kreće u suprotnom smjeru pa se vraća u odredišni položaj
- bounce – simulira sudar s odbijanjem

Osnovne tipove promjena moguće je dodatno proširiti različitim modovima:

- in – uobičajeno ponašanje
- out – vrijeme se koristi u obrnutom smjeru [1,0]
- in-out – duplicira i zrcali funkciju ease od [0,.5] i [.5,1]
- out-in - duplicira i zrcali funkciju ease od [1,.5] and [.5,0]

3.2 Transformacije

Transformacija je postupak u kojem se nekom elementu izmijeni njegov položaj, orijentacija, veličina ili oblik. Transformiranje je moguće vršiti kako nad HTML elementima tako i nad SVG elementima. HTML elemente se transformira putem CSS-a dok za SVG elemente postoji atribut *transform*. Kod SVG transform atributa transformacije se manifestiraju na nešto drugačiji način, nego je to slučaj kod transformacija postignutih primjenom CSS atributa.

3.2.1 Transformacije nad SVG elementima

Nad SVG elementima moguće je raditi različite vrste transformacija kao što su translacija, rotacija, skaliranje i slično. Popis mogućih transformacija je

- translate() – translacija po x i y osima
- rotate() – rotiranje
- scale() – povećanje SVG elementa (položaja, dimenzija i debljine linija);
- skew() – iskrivljavanje za određeni kut (skewX, skewY)
- matrix() – transformacije zadane u matričnom obliku

Važno je naglasiti kako, osim što je transformacije moguće izvršiti postavljanjem naziva transformacije atributu *transform* zajedno s pripadajućim parametrima, transformaciju je moguće definirati i putem matrice transformacije kojom se postiže učinak jednak onome gore spomenutom, npr. translaciju je moguće obaviti preko:

```
transform="translate(75,25)"

transform="matrix(1,0,0,1,75,25)"
```

Primjeri transformacija:

1. translate()

```
<rect x="20" y="20" width="50" height="50" style="fill:
↪ #cc3333"/>

<rect x="20" y="20" width="50" height="50" style="fill: #3333cc"
↪ transform="translate(75,25)" />
```

Slika 3.1: Translacija SVG kvadrata

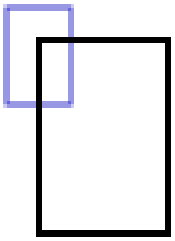
2. rotate()

```
<rect x="20" y="20" width="40" height="40" style="stroke:
↪ #3333cc; fill:none;"/>

<rect x="20" y="20" width="40" height="40" style="fill: #3333cc"
↪ transform="rotate(15)" />
```

3. scale()

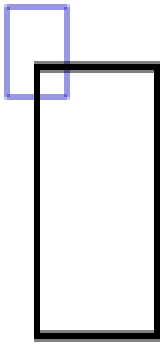
Slika 3.2: Rotacija SVG kvadrata

```
<rect x="10" y="10" width="20" height="30" style="stroke:
↪ #3333cc; fill:none;" />

<rect x="10" y="10" width="20" height="30" style="stroke:
↪ #000000; fill:none;" transform="scale(2)"/>
```



Slika 3.3: Skaliranje SVG pravokutnika - 1 metoda

```
<rect x="10" y="10" width="20" height="30" style="stroke:
↪ #3333cc; fill:none;" />

<rect x="10" y="10" width="20" height="30" style="stroke:
↪ #000000;fill:none;"transform="scale(2,3)" />
```



Slika 3.4: Skaliranje SVG pravokutnika - 2 metoda

4. skew()

```
<rect x="10" y="10" width="20" height="30" style="stroke:
↪ #3333cc; fill:none;" />

<rect x="50" y="10" width="20" height="30" style="stroke:
↪ #000000; fill:none;" transform="skewX(10)" />

<rect x="100" y="10" width="20" height="30" style="stroke:
↪ #000000; fill:none;" transform="skewX(45)" />

<rect x="150" y="10" width="20" height="30" style="stroke:
↪ #000000; fill:none;" transform="skewX(60)" />
```

Slika 3.5: Iskrivljavanje SVG pravokutnika

5. matrix()

```
<rect x="20" y="20" width="50" height="50" style="fill:
↪ #cc3333"/>
<rect x="20" y="20" width="50" height="50" style="fill: #3333cc"
↪ transform="matrix(1,0,0,1,100,20)"/>
```

Slika 3.6: Translacija SVG kvadrata preko matričnog zapisa

Inače ako se transformacije želi definirati u matričnom obliku onda su odgovarajući zapisi ranije opisanih transformacija:

- translacija: matrix(1 0 0 1 x y),
- skaliranje: matrix(x 0 0 y 0 0),
- rotacija oko točke (0, 0): $matrix(\cos\alpha - \sin\alpha\ 0\ \sin\alpha\ \cos\alpha\ 0\ 0\ 0\ 1)$,
- skewX: $matrix(1\ \tan\alpha\ 0\ 0\ 1\ 0\ 0\ 1)$,
- skewY: $matrix(1\ 0\ 0\ \tan\alpha\ 1\ 0\ 0\ 0\ 1)$.

3.2.2 Transformacije SVG elemenata pomoću D3

Primjena transformacija na SVG elemente pomoću biblioteke D3 je krajnje jednostavna jer se u suštini ne razlikuje od primjene transformacija na navedene elemente izravno unutar SVG-a. Ako se želi translatirati pravokutnik za (75,25), onda to u SVG zapisu izgleda ovako:

```
<rect x="0" y="0" width="50" height="50" transform="translate(75,25)"
↪  />
```

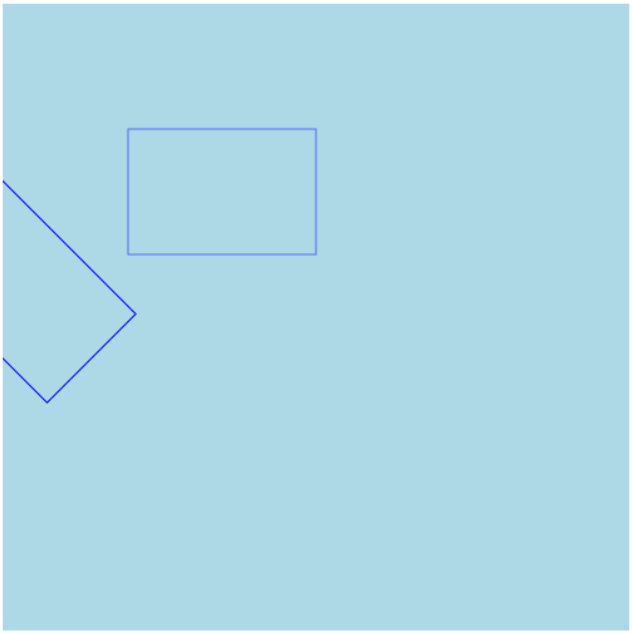
dok bi to pomoću D3, za neki ranije dodani *¡rect¿* element, izgledalo ovako:

```
d3.select("rect")
 .attr("transform", "translate(75,25)");
```

Primjer: Rotacija pravokutnika za 45°

```
var width = 500;
var height = 500;

var svg = d3.select("body")
 .append("svg")
 .attr("width", width)
 .attr("height", height)
 .style("background-color", "lightblue");

var rect1 = svg.append("rect")
 .attr("x", 100)
```

```
  .attr("y", 100)
  .attr("width", 150)
  .attr("height", 100)
  .style("fill", "none")
  .style("stroke", "blue")
  .style("opacity", 0.5);

var rect2 = svg.append("rect")
  .attr("x", 100)
  .attr("y", 100)
  .attr("width", 150)
  .attr("height", 100)
  .style("fill", "none")
  .style("stroke", "blue")
  .attr("transform", "rotate(45)");
```



Slika 3.7: Pravokutnik rotiran za 45°

Ishodište koordinatnog sustava je gornji lijevi kut SVG elementa pa je rotacija obavljena obzirom na tu točku. Ako se rotaciju želi napraviti u odnosu na središte pravokutnika onda je prilikom deklaracije rotacije potrebno navesti i točku oko koje se rotacija vrši pa će posljednja linija gore navedenog primjera biti:

```
.attr("transform", "rotate(45 175 150)");
```

Za točku oko koje se vrši rotacija uzete su redom: x koordinata i pola vrijednosti širine te y koordinata i pola visine. Rezultat gore izmijenjene transformacije je rotacija pravokutnika oko vlastitog središta:

Glavno pitanje prilikom transformiranja jest ishodište koordinatnog sustava u odnosu na koji se transformacija vrši. Navedeno predstavlja problem kada postoji više elemenata koje se želi rotirati oko iste osi. Jedno od rješenja jest da se elementi stavljaju unutar elementa za grupiranje, <g>, koji je translatiran na željeni položaj, a sami elementi unutar grupe imaju položaj (0,0). Dakle, prema podacima se dodaju elementi za grupiranje:

```
var groups = svg.selectAll("g")
 .data(data)
 .enter()
.append("g")
 .attr("transform", function(d){ return "translate(" + d.x + "," + d.y
 ↪  + ")"; });
```

Nakon što su napravljene grupe, dodaju se ostali elementi unutar svake od njih:

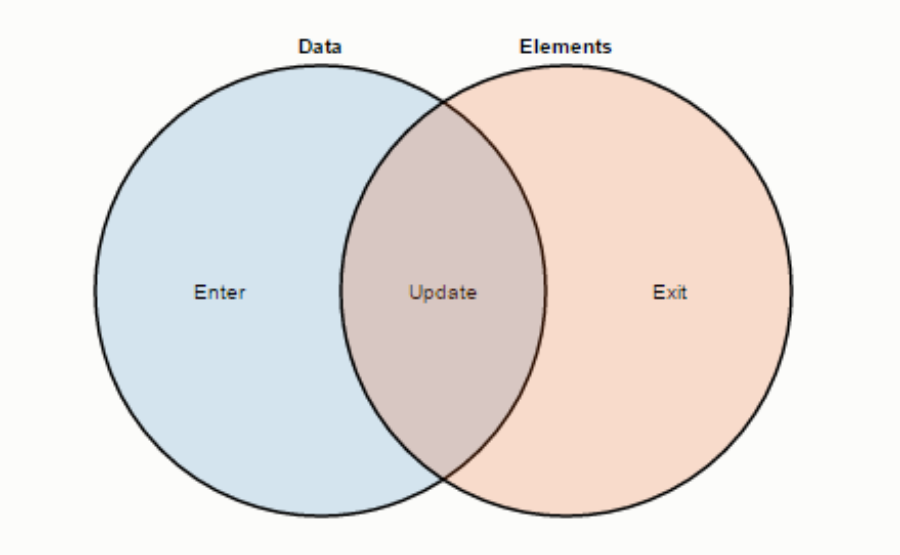
```
var texts = groups
 .append("text")
  .attr("x", function(d){return 0;})
  .attr("y", function(d){return 0;})
  .text((function (d) {return d.x + ", " + d.y;}))
  .style("fill",function (d,i) {return colors(i);})
  .style("opacity",1)
  .on("mouseover",animate);

var rects = groups// svg.selectAll("rect")
 .append("rect")
  .attr("x", function(d){return 0;})
  .attr("y", function(d){return 0;})
  .attr("width", function(d){return d.width;})
  .attr("height", function(d){return d.height;})
  .attr("class", function(d){return d.x;})
  .style("fill", function (d,i) {return colors(i);})
  .style("opacity",0.8)
  .on("mouseover",animate);
```

3.3 Povezivanje elemenata i podataka

Prethodna vježba pokrivala je korištenje statičnih podataka, odnosno podatkovnih skupova koji se definiraju unutar koda i više se ne mijenjaju. Takav pristup dostatan je za veliki broj podatkovnih skupova koji su nepromjenjivi, ali prilikom izrade vizualizacije podataka čest je slučaj kod kojega podatci nisu statični tj. moguće su promjene pojedinih ili svih vrijednosti unutar skupa u različitim vremenskim trenutcima ili drugi slučaj može biti situacija u kojoj se korisniku želi omogućiti interakcija s grafom prilikom koje se u određenom trenutku prikazuju podatci za jedan od više različitih skupova podataka, a primjer čega bi bio odabir prikaza različitih kategorija, npr. odabir između prikaza temperature zraka i vlažnosti zraka i sl.

Zadatak ove vježbe jest upoznati studente načinom kako dinamički prilagoditi vizualizaciju svakoj promjeni koja se dogodi nad, odnosno u podatcima. D3 ima implementiran mehanizam odabira elemenata HTML dokumenta prema stanju i količini s njima povezanih podataka, tj moguće je razlikovati tri vrste elemenata:

- elementi za novonastale podatke - elementi koji još ne postoje unutar DOM-a, tj. elementi koje je potrebno izraditi i povezati ih s pripadajućim podatcima
- elementi za već postojeće podatke – elementi već postoje unutar DOM-a, ali su dobili nove podatke pa ih je potrebno izmijeniti
- elementi za koje podatci ne postoje – postojeći elementi unutar DOM-a za koje podataka više nema pa ih je stoga poželjno ukloniti iz DOM-a, odnosno sa stranice



Slika 3.8: Tri načina dohvaćanja elemenata

Navedeno se naziva pridruživanje podataka i predstavlja učinkovit mehanizam za dodavanje novih elemenata, modificiranje postojećih i uklanjanje elemenata za koje podatci više ne postoje. D3 razlikuje tri različite kategorije elemenata, ovisno o njihovim podatcima pa stoga postoje i tri različita načina kako dohvatiti elemente pojedine kategorije, a navedeni selektori zovu se:

- enter – vraća spremnike za elemente za koje podatci postoje, ali elemenata u DOM-u još nema, odnosno za tzv. novonastale elemente

- update – slučaj kada elementi postoje unutar DOM-a, ali su se podatci izmijenili
- exit – elementi postoje unutar DOM-a, ali više ne postoje podatci koji se mogu spojiti s navedenim elementima

Način pridruživanja podataka elementima i dodavanje novih elemenata prikazan na prethodnim LV zapravo je samo pojednostavljeni slučaj onoga što će biti predstavljeno u ovoj vježbi.

Varijabla *data* sadržavat će 10 parova x i y vrijednosti koje će predstavljati koordinate unutar SVG elementa, tj. položaj na koji će se smjestiti <*circle*> elementi.

```
var data = d3.range(10).map(function() {
 return {
  x: Math.random()*500,
  y: Math.random()*500
  };
});
```

Varijabla *circles* sadržavat će sve <circle> elemente:

```
var circles = svg.selectAll("circle");
```

Metoda *selectAll("circle")* vratit će prazno polje jer ne postoji niti jedan <*circle*> element unutar SVG-a. Ako se nakon pozivanja *selectAll()* metode pozove i *data()* metoda, onda će se navedenom odabiru pridružiti podatci te će nastati tri odabira ovisno o tri moguća stanja (enter, update i exit):

```
var circles = svg.selectAll("circle");
 .data(data);
```

Budući da je prilikom prvog poziva SVG element prazan, update i exit odabiri bit će prazni, a enter odabir vratit će spremnike za nove elemente <*circle*>. Logičan slijed jest dohvatiti novonastale elemente putem enter() metode te postaviti im željene atribute kao u prethodnoj vježbi:

```
var circles = svg.selectAll("circle")
 .data(data)
 .enter()
.append("circle");
 .attr("cx", function(d) { return d.x; })
 .attr("cy", function(d) { return d.y; })
 .attr("r", 15)
 .attr("height", 100)
 .style("fill", "none")
 .style("stroke", "blue");
```

Ovo je uobičajen i sasvim ispravan način izrade statičnih vizualizacija, ali za vizualizacije kod kojih podaci mogu nastajati i nestajati, poželjno je upoznati se s sva dodatna selektora, update i exit. Dio koda za dodavanje elemenata u SVG smjestit će se u izdvojenu funkciju kako bi ga se moglo pozivati svaki put kada se podatkovni skup izmijeni kako bi se osigurala i prikladna izmjena vizualizacije.

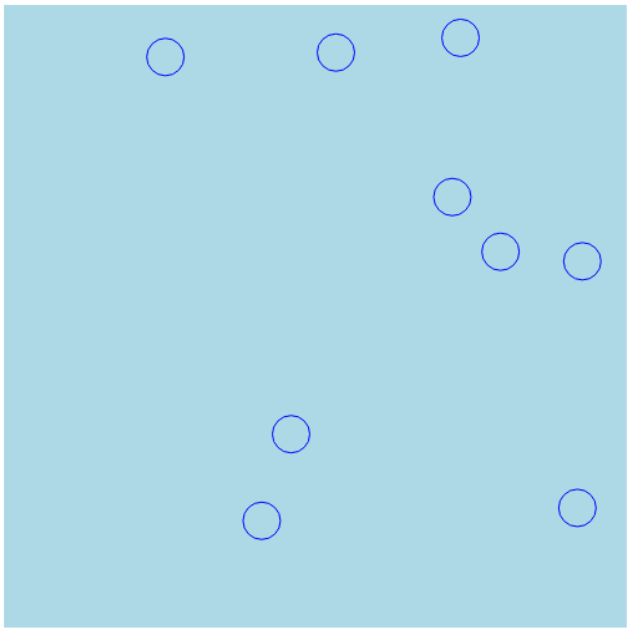
```
function update() {
 var randomNumber = Math.ceil(Math.random() * 10);
 var data = d3.range(randomNumber).map(function() { return {x:
  ↪  Math.random()*500, y: Math.random()*500}; });

//pridruzivanje podataka te ujedno i update odabir

 var circles = svg.selectAll("circle")
  .data(data)
  .style("fill", "blue");

//enter odabir - dodavanje novih elemenata nakon kojega slijedi
 ↪  pridruzivanje vrijednosti atributima

circles.enter()
 .append("circle")
 .attr("cx", function(d) { return d.x; })
 .attr("cy", function(d) { return d.y; })
 .attr("r", 15)
 .attr("height", 100)
 .style("fill", "none")
 .style("stroke", "blue");

//exit odabir - uklanjanje elemenata za koje podaci vise ne postoje
circles.exit().remove(); }

//pozivanje funkcije update() kako bi se dodali odgovarajuci
 ↪  elementi u SVG
update();
```

Pozivanje opisane funkcije update() dodat će broj krugova prema slučajnoj vrijednosti između 1 i 10. Kako bi se vidjelo promjene u podatcima, potrebno je omogućiti pozivanje funkcije update ciklički, a najprikladniji način za ostvariti navedeno jest definirati unutar funkcije setInterval() željeno vrijeme između dva uzastopna poziva i naziv funkcije koju se želi pozvati:

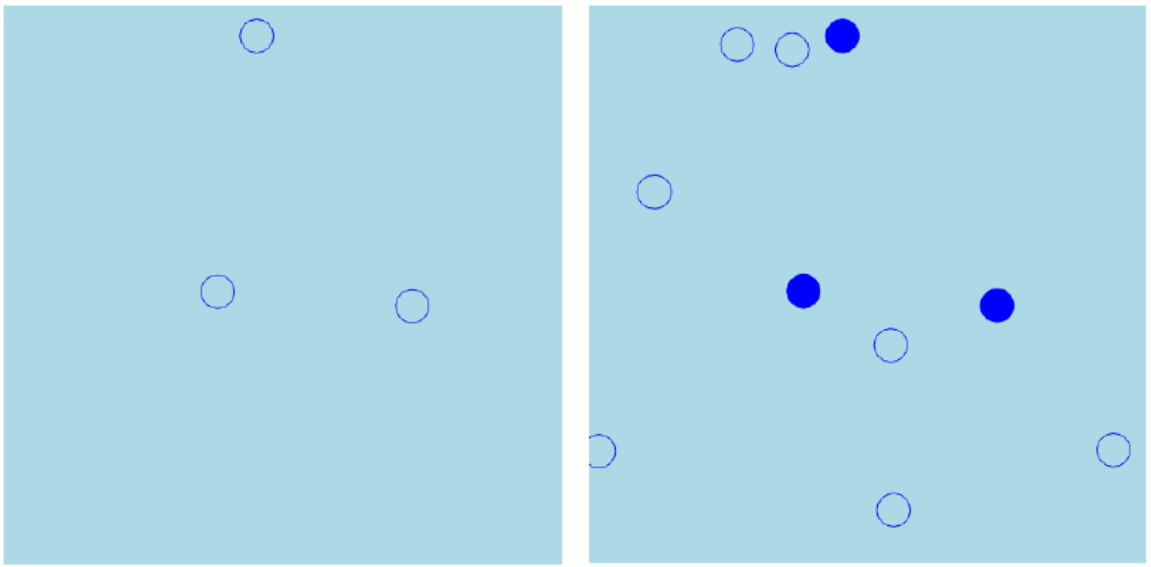
```
setInterval(update, 2000);
```

Dva uzastopna poziva funkcije update pokazat će promjenu u podacima:

Novonastali podatci prikazani su obrubom u boji (3 kruga na lijevoj slici), dok su u slučaju update odabira takvi podaci dodatno ispunjeni plavom bojom (3 kruga na desnoj slici). Novonastali podatci pridružiti će se novim <circle> elementima pa je stoga na lijevoj slici moguće vidjeti još sedam dodanih krugova.



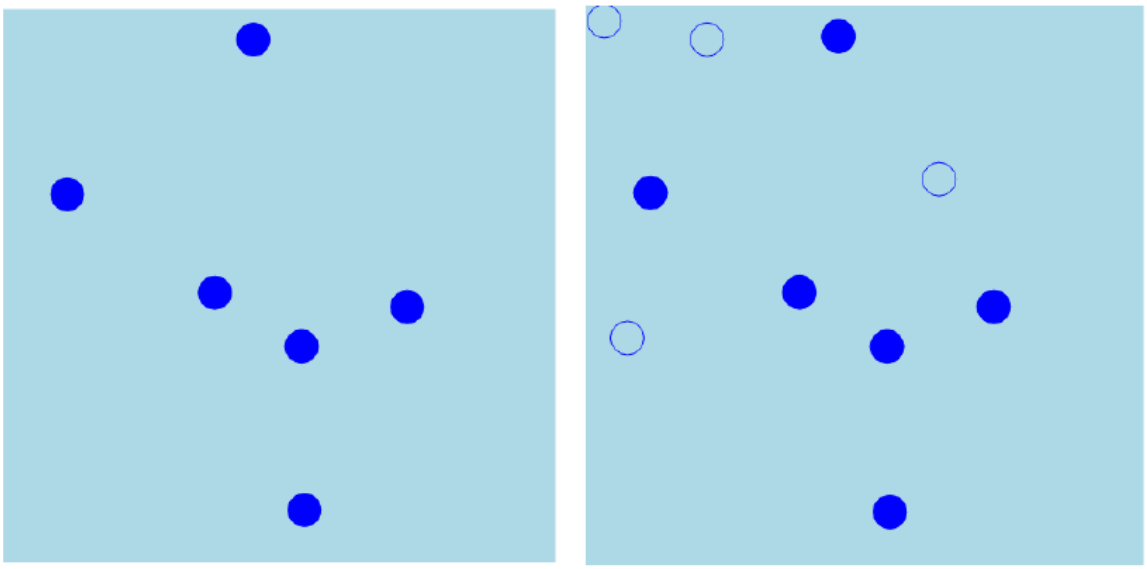
Slika 3.9: Statična vizualizacija



Slika 3.10: Prikaz promjene u podatcima nakon dva poziva funkcije update

Trećim pozivom update funkcije, generirano je samo šest podataka što je manje od ranijih 10 pa je stoga exit odabirom dohvaćeno četiri kruga za koje podatci ne postoje te su isti uklonjeni iz SVG-a. Četvrti poziv funkcije update ponovno je generirao 10 podataka pa su dodana još četiri nova <circle> elementa koja su vidljiva na desnoj slici obojeni plavim obrubom i bez ispune.

Važno je promotriti položaj krugova s ispunom koji se ne mijenja između dva uzastopna poziva jer enter selektor vraća isključivo elemente za koje podatci prethodno nisu postojali, tj. vraća novonastale elemente dok update selektor vraća postojeće elemente. Kako bi se položaj postojećih elemenata izmijenio potrebno je prilikom update odabira izmijeniti vrijednosti atributa položaja:



Slika 3.11: Prikaz nakon trećeg poziva update funkcije

```
//update odabir s ispravnim pozicioniranjem elemenata
var circles = svg.selectAll("circle")
 .data(data)
 .style("fill", "blue")
 .attr("cx", function(d) { return d.x; })
 .attr("cy", function(d) { return d.y; });
```

Ako se rezultat različitih odabira želi animirati, onda je za svaki odabir može definirati odgovarajuća tranzicija. Za olakšano praćenje promjena između različitih poziva funkcije update dodan je i paragraf u tijelo HTML dokumenta kako bi se vidio ukupan broj parova unutar varijable *data*:

```
var myText = d3.select("body").append("p");
```

Konačni izgled funkcije update je:

```
function update() {
 var randomNumber = Math.ceil(Math.random() * 10);
 var data = d3.range(randomNumber).map(function() { return {x:
  ↪  Math.random()*500, y: Math.random()*500}; });

 myText.html("Duljina podataka: " + randomNumber);

 //pridruzivanje podataka te ujedno i update odabir
 var circles = svg.selectAll("circle")
  .data(data)
  .style("fill", "blue");

 circles
  .transition()
```

```
    .duration(1000)
     .attr("cx", function(d) { return d.x; })
     .attr("cy", function(d) { return d.y; });

  //enter odabir - dodavanje novih elemenata nakon kojega slijedi
   ↪  pridruzivanje vrijednosti atributima
  circles.enter()
   .append("circle")
   .attr("cx", function(d) { return d.x; })
   .attr("cy", function(d) { return d.y; })
   .attr("r", 15)
   .attr("height", 100)
   .style("fill", "none")
   .style("stroke", "blue");

  //exit odabir - uklanjanje elemenata za koje podaci vise ne postoje
  circles.exit()
   .transition()
   .duration(1000)
   .attr("r", 0)
   .each("end", function() { d3.select(this).remove(); });
}

setInterval(update, 2000);
```

4 Prikazi

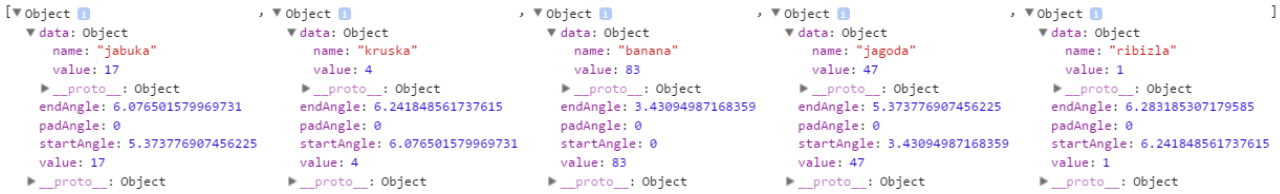
Biblioteka D3, osim što nudi sučelje za izradu i manipulaciju osnovnim HTML i SVG elementima te njihovim atributima, posjeduje i sučelje za kreiranje i manipulaciju različitim prikazima. Riječ je o jednostavnom i brzom načinu izrade prikaza koji se često susreću u brojnim vizualizacijama podataka. Navedene prikaze moguće je i prilagoditi prema željama korisnika, odnosno prema zahtjevima projekta, ali ono što je moguće dobiti kroz osnovnu funkcionalnost, dovoljno je za izradu zanimljivih i smislenih vizualizacija podataka sukladno dobroj praksi.

Izrada uobičajenih prikaza počinje odabirom vrste prikaza koji se želi izraditi. Podržan je rad s brojnim vrstama prikaza:

- Bundle – primjena Holtenovog algoritma hijerarhijskog usnopljavanja
- Chord – povezivanje kružnih odsječaka odgovarajućim putanjama
- Cluster – grupiranje elemenata u stablastu strukturu
- Force – pozicioniranje čvorova korištenje fizikalnog modela
- Hierarchy – izrada vlastitih hijerarhijskih prikaza
- Histogram – histogramski prikaz
- Pack – hijerarhijski prikaz primjenom metode pakiranja kružnica (engl. *circle-packing*)
- Partition – rekurzivno particioniranje čvorova u sunburst prikaz
- Pie – izračun početnih i završnih kutova kružnih odsječaka potrebnih za kružne prikaze (pie chart, donut chart)
- Stack – izračun točaka za stack prikaze
- Tree – pozicioniranje hijerarhijskih čvorova u kružni stablasti prikaz korištenjem Reingold–Tilfordovog algoritma
- Treemap – primjena rekurzivne podjele prostora kako bi se omogućila izrada stablastih karata

Nabrojani prikazi, suprotno svom imenu, ne omogućavaju izravnu izradu odgovarajućeg prikaza, nego služe za preslikavanje iz domene podataka u podatke prikladne za izradu željenih prikaza korištenjem osnovnih vizualnih elemenata. Tako će, primjerice, d3.layout.pie() iz podatkovnog skupa koji sadrži samo jedan broj, koji može predstavljati količinu, izračunati početne i krajnje kutove kako bi se izradili kružni odsječci pomoću d3.svg.arc() metode.

```
var data = [{name: 'jabuka', value: 17}, {name: 'kruska', value: 4},
↪  {name: 'banana', value: 83},  {name: 'jagoda', value: 47}, {name:
↪  'ribizla', value: 1}];
var pie = d3.layout.pie()
 .value(function(d) { return d.value; });

pie(data);
```



Slika 4.1: Prikaz data varijable

4.1 Kružni prikaz

Kružni prikazi su vrsta vizualizacija koja podatke prikazuje preko odgovarajućih kružnih odsječaka. Sam podatak predstavlja kut pripadajućeg kružnog odsječka. Za izradu kružnog prikaza koristit će se *d3.layout.pie()* i *d3.svg.arc()* kao generator odgovarajućih putanja, odnosno <path> elemenata. U gornjem primjeru vidljivo je da je kružni layout samo izračunao relativne odnose između primljenih podataka, ali nije izradio vizualnu reprezentaciju podataka koje je potrebno prikazati na ekranu. Kako bi se to ostvarilo, potrebno je izraditi odgovarajuće SVG elemente na osnovu izračunatih vrijednosti. Za izradu kružnog prikaza koristi se *d3.svg.arc()* generator koji omogućuje jednostavnu izradu kružnih odsječaka putem odgovarajućih <path> elemenata.

Primjer: Izrada kružnog prikaza Potrebno je definirati veličinu SVG elementa te veličinu kružnog prikaza putem vanjskog radijusa kružnih odsječaka.

```
var width = 500;
var height = 500;
var outerRadius = 200;
var innerRadius = 0;
```

Kružni odsječci za međusobno razlikovanje trebaju biti obojeni različitim bojama pa je potrebno pripremiti odgovarajuću skalu za kasnije potrebe bojenja odsječaka:

```
var color = d3.scale.category20();
```

Navedeni odsječci će se izraditi pomoću d3.svg.arc() generatora kojem je potrebno definirati unutrašnji i vanjski radijus:

```
var arc = d3.svg.arc()
  .innerRadius(innerRadius)
  .outerRadius(outerRadius);
```

Podaci koji će se prikazati kružnim prikazom su:

```
var data = [
  {name: 'jabuka', value: 17},
  {name: 'kruska', value: 4},
  {name: 'banana', value: 83},
  {name: 'jagoda', value: 47},
  {name: 'ribizla', value: 1}
];
```

Anonimna funkcija koja se predaje d3.layout.pie().value() metodi vratiti će baš value svojstvo svakog elementa iz polja s podacima jer inače pie-layout ne bi znao koji na osnovu kojeg podatka treba izračunati kutove, tj. ne bi znao treba li koristiti *name* ili *value* svojstvo:

```
var pie = d3.layout.pie()
 .value(function(d) { return d.value; });
```

Potrebno je dodati SVG element u tijelo dokumenta s pripadajućom širinom i visinom:

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

U SVG element dodaju se <g> elementi kojima će se pridružiti podaci putem pie-layouta te su svi translatirani u središte SVG-a:

```
var pieArcs = svg.selectAll("g.pie")
 .data(pie(data))
 .enter()
.append("g")
 .attr("class", "pie")
 .attr("transform", "translate(" + (width / 2) + ", " + (height / 2) +
 ↪  ")");
```

Na kraju se dodaju <path> elementi koji će preko funkcije arc() poprimiti oblik kružnih odsječaka koji zajedno čine kružni prikaz:

```
pieArcs.append("path")
 .attr("fill", function(d, i) { return color(i); })
 .attr("d", arc);
```

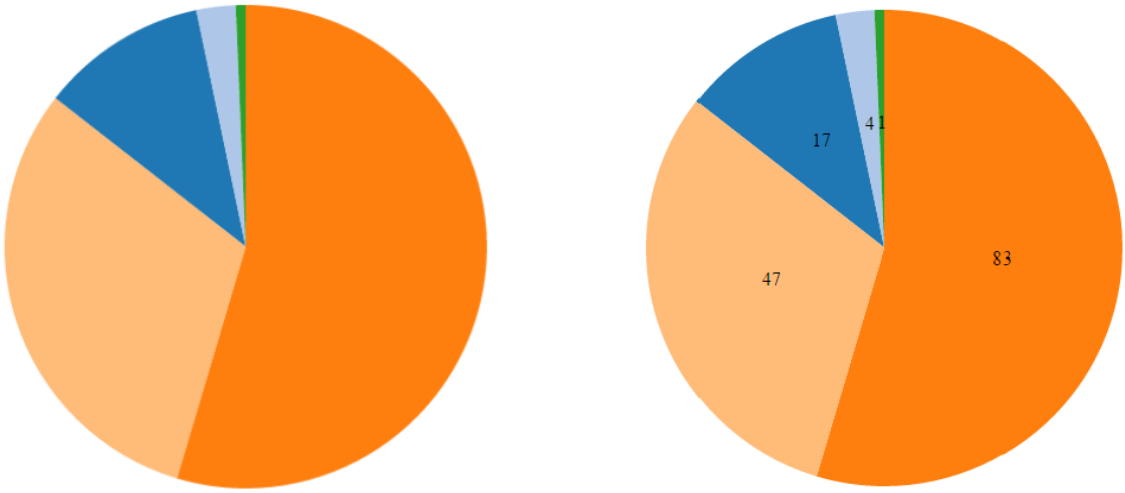
Tekst s vrijednostima moguće je obaviti pomoću:

```
pieArcs.append("text")
 .attr("transform", function(d) { return "translate(" +
 ↪  arc.centroid(d) + ")"; }) .attr("text-anchor", "middle")
 .text(function(d) { return d.value; });
```

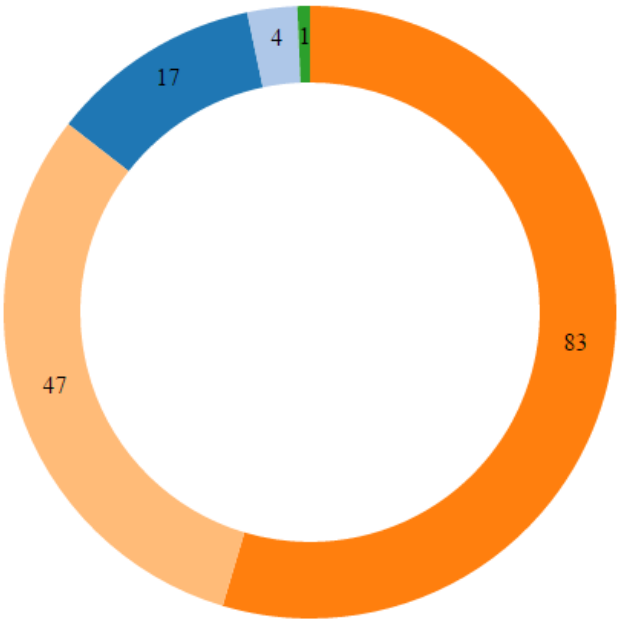
4.1.1 Prstenasti prikaz

Ako se želi izraditi kružni prikaz prstenastog oblika, u gore navedeni kod potrebno je dodati vrijednost za unutrašnji radijus koja će biti u intervalu između 0 i vanjskog radijusa:

```
var outerRadius = 200;
var innerRadius = 150;
```



Slika 4.2: Kružni prikaz



Slika 4.3: Kružni prikaz

4.2 Hijerarhijski prikazi

Prikaz hijerarhijskih podataka moguće je ostvariti korištenjem različitih layouta, ovisno o tome koji konkretan prikaz se želi izraditi. Za izradu stabla potrebno je pripremiti stilove unutar ¡style¿¡/style¿ oznaka koji će definirati izgled svih čvorova i veza među njima:

```
.node circle {
 fill: #fff;
 stroke: steelblue;
```

```
 stroke-width: 1.5px;
}

.node {
 font: 10px sans-serif;
}

.link {
 fill: none;
 stroke: #ccc;
 stroke-width: 1.5px;
}
```

Podaci koje će se prikazati nalaze se u varijabli *data*:

```
var data = {
 "name": "flare",
 "children": [
  {
   "name": "analytics",
   "children": [
    {
     "name": "cluster",
     "children": [
      {"name": "AgglomerativeCluster", "size": 3938},
      {"name": "CommunityStructure", "size": 3812},
      {"name": "MergeEdge", "size": 743}
     ]
    },
    { "name": "graph",
          "children": [
           {"name": "BetweennessCentrality", "size": 3534},
           {"name": "LinkDistance", "size": 5731}
          ]
         }
        ]
  }
 ]
};
```

Dimenzije SVG elementa su:

```
var width = 500;
var height = 500;
```

Definiranje layouta i generatora krivulja napravljeno je pomoću:

```
var cluster = d3.layout.cluster()
 .size([height, width - 160]);

var diagonal = d3.svg.diagonal()
 .projection(function(d) { return [d.y, d.x]; });
```

SVG element dodaje se pomoću:

```
var svg = d3.select("body").append("svg")
 .attr("width", width)
 .attr("height", height)
.append("g")
 .attr("transform", "translate(40,0)");
```

Izračun položaja čvorova i veza među njima obavlja layout, u ovome slučaju d3.layout.cluster() i to metode nodes() i links():

```
var nodes = cluster.nodes(data),
 links = cluster.links(nodes);
```

Prvo se dodaju veze između čvorova:

```
var link = svg.selectAll(".link")
 .data(links)
.enter().append("path")
 .attr("class", "link")
  .attr("d", diagonal);
```

a tek nakon veza dolaze i čvorovi:

```
var node = svg.selectAll(".node")
 .data(nodes)
.enter().append("g")
 .attr("class", "node")
 .attr("transform", function(d) { return "translate(" + d.y + "," +
 ↪  d.x + ")"; })

node.append("circle") .attr("r", 4.5);
```

Razlog takvog rasporeda jest kako bi se osiguralo da se sve veze između čvorova smjeste ispod čvorova. Na kraju se dodaje i tekst imena svakog od čvorova kako bi se, suprotno ranijem, osiguralo da svi nazivi budu iznad ostalih elemenata, tj. kako ih ne bi prekrivali čvorovi ili njihove veze:

```
node.append("text")
 .attr("dx", function(d) { return d.children ? -8 : 8; })
 .attr("dy", 3)
 .style("text-anchor", function(d) { return d.children ? "end" :
 ↪  "start"; })
 .text(function(d) { return d.name; });
```

Dobiveni prikaz bit će struktura stabla koja prikazuje podatke i veze između njih:

5 Projekcije

Za izradu projekcije pomoću D3.js potrebno je:

- razumjeti princip rada s funkcijama za projekcije
- razumjeti princip rada sa SVG <path> elementom
- odabrati i pripremiti topološke podatke

Projekcija je transformacija položaja (geografske dužine i širine) na sferi ili elipsoidu u položaje u ravnini (x,y) koordinate. Neke od funkcija koje se koriste prilikom rada s projekcijama su:

- scale: određuje stupanj zumiranja
- rotate: određuje koji položaj se trenutno prikazuje (rotacija globusa)
- translate: određuje položaj projekcije unutar SVG elementa
- clipAngle: skrivanje država koje su s druge strane globusa, a koje trebaju biti nevidljive

Postoje i druge funkcije koje se mogu koristiti prilikom rada s određenim vrstama projekcija, a za što je neophodno proučiti dokumentaciju.

5.1 Funkcija path

Funkcija *d3.geo.path()* preslikava GeoJSON svojstva u podatke o SVG putanjama (element <path>). GeoJSON je JSON zapis za kodiranje geografskih podatkovnih struktura (svojstava). Komprimirani GeoJSON podaci se pohranjuju u TopoJSON zapisu koji je u pravilu i do nekoliko puta manji u odnosu na odgovarajući GeoJSON, ali po cijenu nešto manje preciznosti.

Neka GeoJSON svojstva su:

- Point - jedna točka [geografska širina i dužina]
- MultiPoint - lista točaka
- LineString - lista točaka namijenjenih za spajanje
- MultiLineString - lista LineStringova
- Polygon - lista LineStringova namijenjenih za zatvaranje
- MultiPolygon - lista Polygona

Za pronalaženje zemljopisnog položaja bilo kojeg mjesta na Zemlji dovoljno je u tražilicu Google upisati naziv mjesta i „longitude latitude“, npr. „Osijek longitude latitude“. Postoje i različiti servisi/portali koji nude istu funkcionalnost:

- Google Maps
- Google Earth
- http://teczno.com/squares
- ...

Primjer: Izrada projekcije SAD-a.

Kao i kod svake izrade vizualizacije potrebno je definirati dimenzije SVG elementa unutar kojega će se projekcija prikazati:

```
var width = 960;
var height = 500;
```

Potrebno je definirati i vrstu projekcije kojom će se koordinate iz geografskih preračunati u koordinate Kartezijevog koordinatnog sustava (x,y):

```
var projection = d3.geo.albers()
```

Funkcija koja će se GeoJSON svojstva preobličiti u instrukcije (generator putanje) pomoću kojih će se izraditi SVG path element je *d3.geo.path()*, a njoj je potrebno navesti s kojom projekcijom će se raditi:

```
var path = d3.geo.path()
 .projection(projection);
```

Nakon što su definirani projekcija i generator putanje postavlja se SVG element unutar tijela dokumenta (<body>):

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

Sljedeći korak je učitavanje JSON podataka o SAD-u. Funkcija *d3.json()*, prima dva parametra: dokument koji treba učitati i funkciju u kojo je navedeno što s učitanim podacima učiniti:

```
d3.json("us.json", function(error, us) {
 svg.append("path")
  .datum(topojson.feature(us, us.objects.land))
  .attr("class", "land")
  .attr("d", path);
});
```

Navedena funkcija je asinkrona, tj. paralelno s učitavanjem JSON datoteke će se izvršavati i ostatak skripte izvan funkcije *function(error, us)* {...}. Po učitavanju podataka unutar SVG elementa će se iscrtavati putanje (<path> elementi). Ako je pojedini dio koda cjelokupne skripte ovisan o učitanim podacima, tj ako se treba izvršiti nakon učitavanja podataka, potrebno ga je staviti ili unutar funkcije ili ga staviti u zasebnu funkciju koju je potrebno pozvati iz funkcije *function(error, us)* {...}. Dakle, koristi se dokument *us.json*, a funkcija koja se poziva nakon učitavanja podataka, *us.json*, koristi TopoJSON svojstva us i us.objects.land, *.datum(topojson.feature(us, us.objects.land))*. Sve iscrtane putanje su klase .land, čime im se može pristupiti, tj. obaviti select. Atribut „d“ poprima vrijednost koju vraća generator putanja var path = d3.geo.path().

Gore navedeni kod potrebno je:

- pohraniti u odvojenu skriptu koju će se učitati unutar HTML dokumenta ili

- dodati u obliku skripte unutar tijela HTML dokumenta oblika:

```
<!DOCTYPE html>
<html>
 <head>
  <script src="http://d3js.org/d3.v3.min.js"></script>
  <script src="http://d3js.org/topojson.v1.min.js"></script>
 </head>
 <body>
  <script>

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

HTML, JS i JSON dokumente potrebno je pohraniti u direktorij kojem se pristupa preko web poslužitelja, tj. potrebno je pokrenuti lokalni web poslužitelj (web server: dodatak za Google Chrome, mongoose ili bilo što drugo) i odabrati navedeni direktorij te otvoriti HTML dokument preko internetskog preglednika.

Otvaranjem, na gore opisani način izrađene, stranice dobiva se sljedeći prikaz:

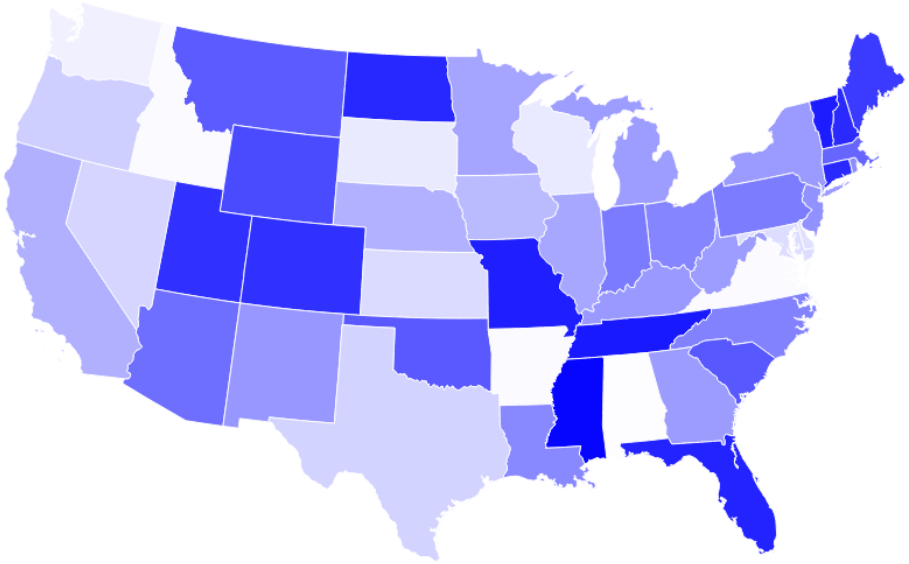
Slika 5.1: Prikaz SAD-a

Navedeno predstavlja prikaz SAD-a, ali bez pojedinih država, tj. unutrašnje podjele. Kako bi se ostvario prikaz svih država, unutar d3.json funkcije potrebno je koristiti drugi TopoJSON property pa je tako umjesto *topojson.feature(us, us.objects.land)* potrebno koristiti *topojson.feature(us, us.objects.states)*:

```
d3.json("us.json", function(error, us) {
 var statesData = topojson.feature(us, us.objects.states);
 var states = svg.selectAll('path.state')
```

```
        .data(statesData.features)
      .enter()
      .append('path')
      .classed("state", true)
      .attr("d", path)
      .style("fill", "blue")
      .style("stroke", "white")
      .style("stroke-width", 2)
      .style("fill-opacity", function() { return Math.random(); });
  });
```

Varijabla land predstavlja cijelo područje SAD-a i obojena je crnom bojom, a varijabla states predstavlja pojedine države unutar SAD-a i one su obojene plavom bojom, ali uz varijabilnu vrijednost *fill-opacity* svojstva pa tako dobiveni prikaz predstavlja kartogram:



Slika 5.2: Obojene države SAD-a

Korištena datoteka ne posjeduju nazive država niti neke druge korisne podatke pa je će sljedeći primjer prikazati izradu projekcije za RH uz korištenje datoteke s topografskim podacima kao i podacima o nazivima objekata koji će se iscrtati na karti.

Primjer: Izrada projekcije Republike Hrvatske

Kod izrade karte za RH potrebno je odabrati željenu projekciju te postaviti odgovarajuće vrijednosti parametara koje se razlikuju od onih iz prethodnog primjera. Potrebno je definirati path generator te mu naznačiti koju će projekciju koristiti prilikom konverzije zemljopisnog položaja u SVG koordinate. U ovom primjeru, koriste se podaci o županijama RH koji, osim što sadrže koordinate granica županija, sadrže i nazive istih. Osnovne postavke su veličina SVG elementa, definiranje projekcije, generatora krivulja i sam SVG element. Navedeno je napravljeno pomoću:

```
var width = 960;
var height = 700;

var projection = d3.geo.mercator()
 .center([0, 10])
 .scale(6000)
  .translate([17600, 4500])
 .rotate([-180, 0]);

var path = d3.geo.path()
 .projection(projection);

var svg = d3.select("body").append("svg")
 .attr("width", width)
 .attr("height", height)
 .style("background", "black");
```

Budući da je topološke podatke potrebno učitati iz zasebne datoteke, dodavanje svih ¡path¿ elemenata u SVG element obavlja se unutar funkcije d3.json() koja će asinkrono učitati podatke, a nakon učitavanja će pozvati funkciju koja će dodati sve gore navedeno:

```
d3.json("cro_regv3.json", function(error, cro) {
 var data = topojson.feature(cro, cro.objects.layer1);

 var states = svg.selectAll("path.county")
  .data(data.features)
  .enter()
 .append("path")
  .attr("class", "county")
  .attr("id", function(d) { return d.id; })
  .attr("d", path) .style("fill", "red")
  .style("stroke", "gray")
  .style("stroke-width", 1)
  .style("stroke-opacity", 1);
 });
```

Rezultat:

Slika 5.3: Prikaz Republike Hrvatske

5.2 GeoJSON i TopoJSON zapisi

Podaci u GeoJSON zapisu često sadržavaju suvišne detalje te podatke koji se ponavljaju pa je osmišljena nova vrsta zapisa topoloških podataka koja smanjuje količinu detalja i izbacuje redundanciju iz podataka kako bi se podatke komprimiralo. TopoJSON zapis može biti i do 80% manji od odgovarajućeg GeoJSON zapisa. Svaki objekt u GeoJSON zapisu ima vlastite koordinate (putanju) koje ga opisuju, tj. opisuju njegove oblik/granice. Budući da velik broj država ima susjede, s tim susjedima dijeli i granice pa je u TopoJSON zapisu ta činjenica iskorištena za smanjenje količine podataka potrebne za opis objekata. Kod TopoJSON zapisa za svaki objekt se navedu redni brojevi elemenata polja na kojima se nalaze koordinate, a onda su koordinate navedene samo jedanput te se mogu koristiti za iscrtavanje dvije države koje imaju zajedničku granicu.

Primjer GeoJSON zapisa:

```
{
 "type": "FeatureCollection",
 "features": [
  {
   "type": "Feature",
        "id": "01",
        "properties": { "name": "Alabama" },
        "geometry": {
```

```
                "type": "Polygon",
               "coordinates": [[[-87.359296,35.00118],
                 [-85.606675,34.984749],[-85.431413,34.124 869],
                [-85.184951,32.859696],[-85.069935,32.580 372],
                [-84.960397,32.421541],[-85.004212,32.322 956],
                [-84.889196,32.262709],[-85.058981,32.136 74] ...
                ]]
          }
        },
        {
         "type": "Feature",
         "id": "02",
         "properties": { "name": "Alaska" },
         "geometry": {
          "type": "MultiPolygon",
          "coordinates": [[[[-131.602021,55.117982],
           [-131.569159,55.28229],[-131.355558,55.183705],
           [-131.38842,55.01392],[-131.645836,55.035827],
           [-131.602021,55.117982]]],[[[-131.832052,55.42469],
           [-131.645836,55.304197],[-131.749898,55.128935],
           [-131.832052,55.189182], ...
          ]]]
         }
}
```

Primjer TopoJSON zapisa:

```
        {
        "scale":[0.0015546403012701271,0.0010939367048704803],
        "translate":[-13.69131425699993,49.90961334800009]},
        "objects":{
        "subunits":{
         "type":"GeometryCollection",
         "geometries":[
         {
          "type":"MultiPolygon",
          "id":"ENG",
          "arcs":[[[0]],[[1]],[[2]],[[3]],[[4]],[[5]], [[6,7,8,9]]],
          "properties":{"name":"England"}},
         {
          "type":"MultiPolygon",
          "id":"IRL",
          "arcs":[[[10]],[[11]],[[12]],[[13]],[[14]], [[15]],[[16,17]]],
          "properties":{"name":"Ireland"}},
         ...
         ]},
        "places":
```

```
 {
 "type":"GeometryCollection",
 "geometries":[
  {"type":"Point",
  "coordinates":[5868,5064],
  "properties":{"name":"Ayr"}},
  {"type":"Point",
  "coordinates":[7508,6637],
  "properties":{"name":"Aberdeen"}},
  {"type":"Point",
  "coordinates":[8776,1455],
  "properties":{"name":"London"}
  }]}},
"arcs":
 [[[4787,4],[-6,-4],[-8,3],[1,13],[6,7],[4,-3],[5,-8]
  ↪  ,[-2,-8]],[[4762,36],[-4,-7],[-5,3],[-5,10],[0,8]
  ↪  ,[4,-1],[3,-3],[2,-1],[4,-3],[1,-6]],[[8015,789]
  ↪  ,[98,-36],[16,0],[7,-1],[8,-5],[0,-3],[-1,-7],
  ↪  [1,-3],[2,0],[5,1],[2,-1],[13,-19],[0,-6],[-3,0],
  ↪  [-2,-2],[-1,-2],[-2,-3],[-24,-6],[-11,-5],
  ↪  [-18,-14],[-3,-4],[-2,-5],[0,-20],[0,-10],[-2,-4],
  ↪  [-72,-19],[-24,6],[-84,54],[-4,5],[-22,14],
  ↪  [-7,3],[-25,-3],[-10,-4],[-12,-6],[16,25],[25,20],
  ↪  [57,24],[-2,-3],[-1,-6],[-2,-4],[16,2],
  ↪  [15,11],[22,25],[17,10],[14,1]],
...
...
...
[[6135,3097],[8,-1],[9,1],[14,3],[8,1],[4,2],[6,9],
 ↪  [4,2],[4,-1],[8,-4],[3,-1],[19,0],[9,-1],[8,-6],
 ↪  [-13,-12],[-23,-36],[-15,-7],[-16,-3],[-17,-7],
 ↪  [-15,-10],[-12,-12],[-3,-3],[-1,-2],[0,-4],[0,-7],
 ↪  [-1,-7],[-2,-1],[-3,1],[-23,-14],[-10,-11],
 ↪  [-41,-21],[-7,-2],[-8,2],[-16,6],[0,-2],[-4,-3],
 ↪  [-5,-2],[-4,1],[-1,6],[0,8],[1,8],[2,3],[5,4],
 ↪  [7,9],[3,8],[-6,4],[-10,-4],[-14,-18],[-10,-3],
 ↪  [-1,3],[-21,16],[-2,1],[-5,-2],[-2,1],[0,2],
 ↪  [1,8],[-1,2],[0,3],[0,7],[-3,7],[-13,10],[-21,36],
 ↪  [-12,12],[0,7],[4,9],[1,14],[0,20],[3,31],[-1,13],
 ↪  [-7,14],[12,7],[82,24],[9,0],[44,-6],[13,-1],
 ↪  [24,-12],[7,-5],[3,-11],[1,-11],[2,-5],[0,-2],
 ↪  [8,-5],[9,-5],[4,0],[0,-5],[0,-6],[-1,-6],[3,-2],
 ↪  [0,-2],[11,-10],[3,-11],[1,-8],[3,-5]]]}
```


Izvor:

http://blog.james-lafa.fr/how-to-display-in-10-minutes-worldwide-data-on-a-globe-with-d3-js/

http://maori.geek.nz/post/d3_js_geo_fun

Primjeri:

https://github.com/mbostock/d3/wiki/Geo-Projections

https://www.npmjs.org/package/d3-geo-projection

Korisno:

http://teczno.com/squares