D3 – Data-Driven Documents

JavaScript(JS) knižnica pre vizualizáciu dát, používa sa s HTML, CSS a často SVG. Stiahnuť si ju môžete na stránke http://d3js.org/ .

Na to aby sme mohli začať používať D3 potrebujeme ovládať základne koncepty HTML, CSS a JS.

HTML

HTML je skratka pre Hypertest Markup Language a používa sa na štruktúrovanie obsahu pre webové prehliadače. Jednoduchá stránka HTML vyzerá napríklad takto:

```
<html>
    <head>
        <title>Page Title</title>
    </head>
    <body>
        <h1>Page Title</h1>
        <p>This is a really interesting paragraph.</p>
    </body>
</html>
```

Vytvorte si túto jednoduchú stránku a uložte si ju pod menom ***index.html***, budeme na ňu následovne nabaľovať ďalší obsah.

DOM

Document Object Model, alebo DOM, je hierarchická štruktúra HTML. Každý *tag,* resp. *HTML značku* môžeme považovať za element. Medzi jednotlivými elementami potom môžeme určovať vzťahy ako parent (rodič), child (dieťa), sibling (súrodenec), atď.

V našej jednoduchej stránke vidíme niekoľko elementov, ako napr. <p>, <h1>, <head>. V našom prípade vidíme napríklad <body>, ktoré má dve deti <p> a <h1>. Rodičom <p> je teda <body> a súrodencom <p> je <h1>.

CSS

Cascading Style Sheets sa používajú na pridanie vizuálneho štýlu našej stránke. Naša jednoduchá stránka momentálne vyzerá následovne:

To nevyzerá úplne pekne. Preto si vytvoríme jednoduchý CSS súbor a uložíme ho pod menom ***mystyle.css*** do rovnakej zložky ako ***index.html***.

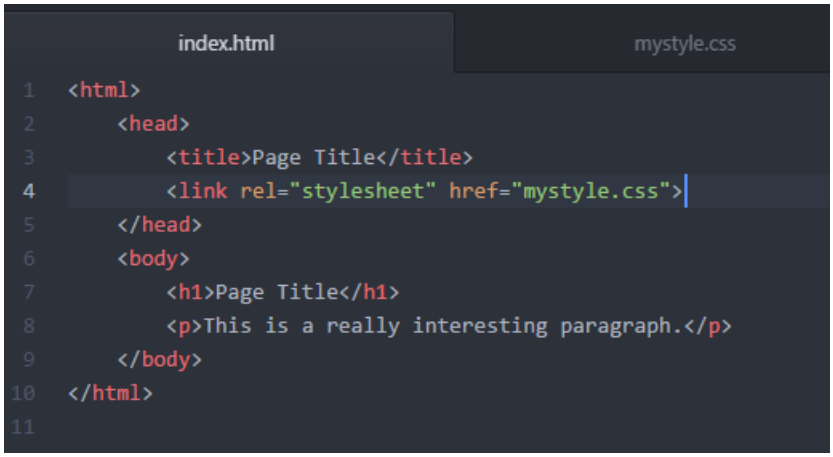
```
body {
      background-color: white;
      color: black;
}

p{
      font-size: 24px;
      line-height: 14px;
      color: orange;
}
```

Pokiaľ si teraz refreshneme našu stránku, nič sa nestane. Musíme jej ešte dať vedieť, že má používať nami vytvorený CSS štýl.

V *index.html* pridáme pod značku <title></title> následujúci riadok:

<link rel="stylesheet" href="mystyle.css">

Tak aby náš súbor vyzeral následovne:



Naša stránka by mala momentálne vyzerať takto:

Čo je o čosi lepšie ako to, ako vyzerala predtým.

JavaScript

JavaScript je dynamický skriptovací jazyk ktorý sa často používa pri tvorbe dynamických webových stránok. Vytvoríme si teraz jednoduchý JavaScript súbor a vložíme ho do našej stránky. Vytvorte si

súbor ***helloWorld.js*** a uložte ho do rovnakej zložky ako súbor ***index.html***. Tento súbor bude obsahovať jednoduchý kód.

helloWorld.js

```
alert("Hello, world!");
```

Aby naša stránka používala tento skript, musíme ešte pridať jeden riadok kódu do súboru *index.html,* hneď pod značku <link> v ktorej sme pridali CSS súbor:

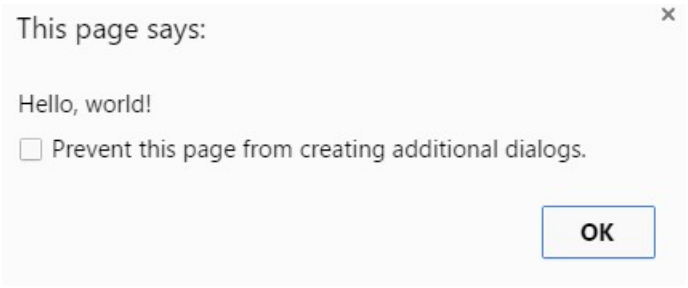
```
<script type="text/javascript" src="helloWorld.js"></script>
```

Súbor *index.html* by teda mal vyzerať následovne:

helloWorld.js | index.html

```
<html>
    <head>
        <title>Page Title</title>
        <link rel="stylesheet" href="mystyle.css">
        <script type="text/javascript" src="helloWorld.js"></script>
    </head>
    <body>
        <h1>Page Title</h1>
        <p>This is a really interesting paragraph.</p>
    </body>
</html>
```

Keď teraz otvoríme *index.html* v prehliadači vyskočí na nás následovné otravné okno:



Presne to sme chceli!

D3

Po tom, čo sme získali základy HTML a CSS, môžeme ísť používať D3. Pokiaľ ste si už stiahli D3 zo stránky http://d3js.org/ otvorte si .zip súbor a rozbaľte do zložky <project-folder>/d3, tak aby štruktúra vášho projektu vyzerala následovne:

project-folder/

d3/

d3.js

index.html

Môžete si samozrejme vytvoriť úplné novú zložku so súborom a taktiež nový *index.html*. Aby sme mohli začať používať D3 v našom HTML súbore, musíme mu, rovnako ako tomu bolo s JS skriptom a CSS súborom, dať vedieť, že má používať knižnicu D3.

Náš *index.html* prepíšeme do tohto stavu:

helloWorld.js | index.html

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <title>D3 Test</title>
        <script type="text/javascript" src="d3/d3.js"></script>
    </head>
    <body>

    </body>
    <script type="text/javascript" src="myJScode.js"></script>
</html>
```

Taktiež si vytvoríme prázdny JS skript s názvom ***myJScode.js*** do ktorého budeme písať príkazy D3. Tento skript uložte do rovnakej zložky, kde sa vyskytuje váš *index.html*.

Pokiaľ otvoríte súbor *index.html* veľa toho neuvidíte, takže poďme pridať nejaký obsah!

Pridanie jednoduchého odstavca

Ak by sme chceli pridať jednoduchý odstavec s textom pomocou JS skriptu stačí nám zavolať niekoľko príkazov. Napíšte do súbora *myJScode.js* následujúci príkaz:

```
d3.select("body").append("p").text("New paragraph!");
```

Ak teraz otvoríme *index.html* uvidíme tam nový odstavec. Rovnakým spôsobom môžete pridať aj nádpis typu <h1> a pridať do súboru *index.html* CSS súbor ako na začiatku a vytvoriť tak dynamicky rovnakú stránku ako predtým.

Ale čo vlastne kód v našom JS skripte spravil? Metóda ***select*** vybrala element <body> do ktorého sme pomocou metódy ***append*** pridali nový odstavec (element <p>) a za použita metódy ***text*** sme odstavcu pripradili text.

Pridávanie dát a ich vizualizácia

D3 ale nechceme používať na to, aby sme pridávali odstavce, ale na to, aby sme vizualizovali dáta. Takže sa do toho pustíme. V súbore *myJScode.js* si najprv vytvoríme dáta, ktoré budeme zobrazovať. Pre zjednodušenie nám bude stačiť päť hodnôt:

```
var dataset = [ 5, 10, 15, 20, 25 ];
```

Tieto dáta následne naviažeme na elementy v našom HTML súbore.

```
d3.select("body").selectAll("p")
      .data(dataset)
      .enter()
      .append("p")
      .text("New paragraph!");
```

JS kódy by mal vyzerať následovne:

myJScode.js

```
var dataset = [ 5, 10, 15, 20, 25 ];
d3.select("body").selectAll("p")
    .data(dataset)
    .enter()
    .append("p")
    .text("New paragraph!");
```

Čo tento kód robí? Rovnako ako pred chvíľou nájdeme body HTML súboru a označíme všetky odstavce <p> ktoré v ňom sú. V našom prípade tam však žiaden nie je. Preto musíme použiť metódu enter, ktorá naviaže naše dáta na nové elementy, aj pokiaľ ešte zatiaľ neexistujú. Vzhľadom na to, že v našom datasete máme 5 hodnôt vytvorí sa nám 5 odstavcov, každý obsahujúci text New paragraph! a odpovedajúce dáta (aj keď ich zatiaľ nevidíme).

Pokiaľ si teraz otvoríme náš HTML súbor vidíme:

New paragraph!

New paragraph!

New paragraph!

New paragraph!

New paragraph!

To nám veľmi nepomôže. Zobrazme si teda dáta, ktoré sa v jednotlivých odstavcoch nachádzajú. Zmeňte poslednú volanú funkciu na:

```
.text(function(d) { return 'Value in paragraph: ' + d; });
```

Týmto spôsobom sme vytvorili vlastnú funkciu, ktorá vytvorí textový reťazec, ktorý nám oznámi, aká hodnota sa momentálne v našom elemente nachádza. Táto hodnota je predávaná funkcii ako parameter ***d*** a môžeme ju použiť rovnakým spôsobom ako akýkoľvek iný parameter.

Pokiaľ si teraz otvoríme náš HTML súbor uvidíme, že každý odstavec zobrazuje svoje dáta.

Funkcie ktoré napíšeme môžu byť aj oveľa komplikovanejšie ako táto. Pokiaľ by sme napríklad chceli zmeniť farbu písma v odstavci podľa hodnoty môžeme použiť funkciu ***style*** následovne:

```
.style("color", function(d) {
```

```
     if (d > 15) {
          return "red";
     } else {
          return "black";
     }
});
```

Náš kód bude vyzerať následovne:

myJScode.js index.html

```
var dataset = [ 5, 10, 15, 20, 25 ];
d3.select("body").selectAll("p")
    .data(dataset)
    .enter()
    .append("p")
    .text(function(d) { return 'Value in paragraph: ' + d; })
    .style("color", function(d) {
    if (d > 15) {
        return "red";
    } else {
        return "black";
    }
});;
```

HTML stránka teraz zobrazuje odstavce, ktoré majú hodnotu väčšiu ako 15 v červenej farbe. Takýmto spôsobom môžeme zmeniť ľubovoľný atribút štýlu, rovnako ako by sme to robili za pomoci CSS súboru.

SVG

Mohli by sme teraz zobrať dáta, ktoré už máme a priamo ich vizualizovať za pomoci CSS štýlov. Avšak D3 je oveľa užitočnejšie pokiaľ používa SVG elementy. SVG je skratka pre Scalable Vector Graphics a dá sa používať podobne ako HTML. Výhodou SVG je to, že sa jedná o vektorovú grafiku, ktorá vyzerá oveľa lepšie pri zmene veľkosti jednotlivých elementov.

Na to, aby sme mohli vykresľovať jednotlivé SVG elementy, potrebujeme najprv SVG panel. Zmažte z myJScode.js všetok kód okrem hodnôt datasetu a pridajte tam nasledujúci kód:

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

var h = 50;

var w = 500;

svg.attr("width", w)

    .attr("height", h);
```

Teraz keď máme SVG panel o veľkosti 500x50 pixelov uložený v premennej, môžeme ho jednoducho zavolať a vykresľovať do neho naše dáta. SVG samotné obsahuje možnosť použiť jednoduché prvky, ako napríklad obdĺžnik (rect), kruh (circle), elipsa (ellipse), čiara (line) atď.

Každý z týchto elementov má svoje parametre. Pozrieme sa bližšie na kružnicu. Tak, ako by ste očakávali, kružnica je definovaná svojim stredom a polomerom. Preto tento SVG element obsahuje atribúty cx (x-ová súradnica stredu), cy (y-ová súradnica stredu) a r (polomer).

Využijeme dáta, ktoré máme a namapujeme ich na veľkosť polomeru.

```
var circles = svg.selectAll("circle")
                    .data(dataset)
                    .enter()
                    .append("circle");
circles.attr("cx", function(d, i) {
               return (i * 50) + 25;
          })
         .attr("cy", h/2)
         .attr("r", function(d) {
               return d;
         });
```

Náš kód by mal vyzerať následovne:

myJScode.js | index.html

```
var dataset = [ 5, 10, 15, 20, 25 ];
var svg = d3.select("body").append("svg");
var h = 50;
var w = 500;

svg.attr("width", w)
   .attr("height", h);

var circles = svg.selectAll("circle")
                 .data(dataset)
                 .enter()
                 .append("circle");
circles.attr("cx", function(d, i) {
            return (i * 50) + 25;
        })
       .attr("cy", h/2)
       .attr("r", function(d) {
            return d;
       });
```

Pokiaľ sa teraz pozrieme na našu webovú stránku uvidíme 5 kruhov, každý s polomerom odpovedajúcim jeho hodnote.

V kóde sme najprv vytvorili kolekciu kruhov, podobne ako tomu bolo v predchádzajúcom prípade s odstavcami. Nasledne sme každému kruhu priradili hodnotu atribútov za pomoci funkcie ***attr.*** Hodnota polomeru je braná z dát samotných, y-ová súradnica stredu je namapovaná na polovicu výšky SVG panelu, aby tak boli prvky vycentrované. Pri funkcii ktorá definuje x-ovú súradnicu si

môžeme všimnúť, že funkcia berie okrem parametra ***d*** aj parameter ***i***. Tento parameter označuje poradie elementu v kolekcii.

Silnou vlastnosťou D3 je to, že môžeme jednoducho pridávať nové elementy do datasetu. Vyskúšajte si pridať pár nových hodnôt a zistite, čo sa stane.

Bar Chart

Ako poslednú vec si vytvoríme z našich dát jednu zo základných vizualizácií, tzv. Bar Chart. Vymažte všetok kód z myJScode.js a nahraďte ho následovne:

```
var dataset = [ 5, 10, 13, 19, 21, 25, 22, 18, 15, 13,
                    11, 12, 15, 20, 18, 17, 16, 18, 23, 25 ];


var w = 500;
var h = 100;
var svg = d3.select("body")
            .append("svg")
            .attr("width", w)
            .attr("height", h);
```

Tento kód nám vytvorí nový ***dataset***. Následne vytvoríme SVG panel, do ktorého budeme vykresľovať vizualizáciu.

Aby sme mohli vidieť bar chart, pridáme do JS skriptu tento kód:

```
svg.selectAll("rect")
    .data(dataset)
    .enter()
    .append("rect")
    .attr("x", 0)
    .attr("y", 0)
    .attr("width", 20)
    .attr("height", 100);
```

Týmto zobrazíme všetky hodnoty, avšak všetky budú zobrazené na jedno miest a budú mať rovnakú výšku. Aby sme sa presvedčili, že máme skutočne všetky hodnoty zobrazené prepíšeme časť

```
.attr("x", 0)
```

na

```
.attr("x", function(d, i) {

     return i * 21;
})
```

Náš kód by mal momentálne vyzerať takto:

myJScode.js index.html

```
var dataset = [ 5, 10, 13, 19, 21, 25, 22, 18, 15, 13,
                11, 12, 15, 20, 18, 17, 16, 18, 23, 25 ];

    var w = 500;
    var h = 100;
    var svg = d3.select("body")
                .append("svg")
                .attr("width", w)
                .attr("height", h);

   svg.selectAll("rect")
   .data(dataset)
   .enter()
   .append("rect")
   .attr("x", function(d, i) {
    return i * 21;
    })
   .attr("y", 0)
   .attr("width", 20)
   .attr("height", 100);
```

Pokiaľ by sme ale chceli dynamicky meniť počet hodnôt, ktoré zobrazujeme, mohli by sme rýchlo vybehnúť z nášho panelu. Preto upravíme aj šírku jednotlivých stĺpcov tak, aby odzrkadľovala počet prvkov, ktoré zobrazujeme.

V JS skripte definujeme novú premennú

```
var barPadding = 1;
```

Ktorá bude určovať medzeru medzi jednotlivými stĺpcami. Následne zmeníme atribút ***width*** z fixnej hodnoty 20 na

```
.attr("width", w / dataset.length - barPadding)
```

A atribút ***x*** na

```
.attr("x", function(d, i) {
     return i * (w / dataset.length);
})
```

Týmto sme vytvorili závislosť pozície každého stĺpca a jeho šírky od počtu prvkov. Pridajte do datasetu nové prvky aby ste sa uistili, že váš kód funguje.

Teraz keď už vidíme, že všetky dáta, ktoré máme sa reálne zobrazujú, mali by sme ich vizualizovať tak, aby sa ich hodnota odzrkadľovala v našom grafe. Napamujeme teda hodnoty datasetu na výšku stĺpcov. Najjednoduchší spôsob, ako to docieliť je zmenov hodnoty atribútu height z fixnej hodnoty tak, aby odzrkadľovala naše dáta:

```
.attr("height", function(d) {
     return d * 4;
});
```

Naše hodnoty sme prenásobili hodnotou 4 aby boli rozdiely lepšie viditeľné. Každopádne, stĺpce ktoré sme dostali nie sú úplne to, čo by sme chceli docieliť.

Tento efekt nastal kvôli tomu, že SVG má súradnicu (0,0) v ľavom hornom rohu. Aby sme to napravili, musíme dáta vykresľovať od spodnej časti panela. Zmeňte atribút ***y*** na:

```
.attr("y", function(d) {
     return h – d * 4;
})
```

A následne výšku stĺpca na:

```
.attr("height", function(d) {
     return d * 4;
});
```

Výsledok už vyzerá o čosi lepšie. Pridáme teraz ďalšiu informáciu a hodnotu dát namapujeme aj na farbu stĺpca. Na to slúži atribút ***fill***. Pridáme následovný kód:

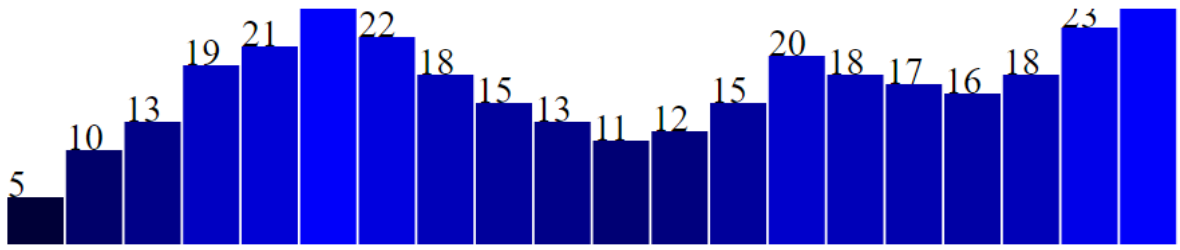
```
.attr("fill", function(d) {
     return "rgb(0, 0, " + (d * 10) + ")";
});
```

Toto nám zarúči, že čim väčšia je hodnota dát, tým saturovanejšia bude farba daného stĺpca. Kód na vykreslenie bar chartu by mal vyzerať následovne:

```
    svg.selectAll("rect")
    .data(dataset)
    .enter()
    .append("rect")
    .attr("x", function(d, i) {
     return i * (w / dataset.length);
     })
     .attr("y", function(d) {
      return h - d * 4;
     })
    .attr("width", w / dataset.length - barPadding)
    .attr("height", function(d) {
     return d * 4;
})
.attr("fill", function(d) {
    return "rgb(0, 0, " + (d * 10) + ")";
});
```

Aby sme videli aké hodnoty sa zobrazujú pridáme do SVG ďalší element, tentoraz to bude ***text***.

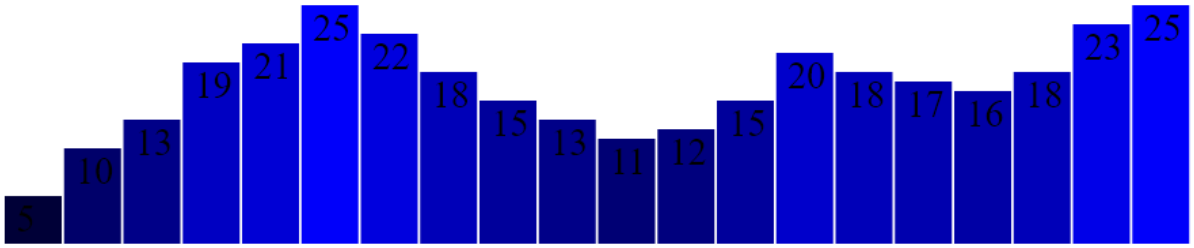
```
svg.selectAll("text")
    .data(dataset)
    .enter()
    .append("text")
    .text(function(d) {
          return d;
    })
   .attr("x", function(d, i) {
          return i * (w / dataset.length);
    })
    .attr("y", function(d) {
          return h - (d * 4);
    });
```



Niektoré hodnoty sú však orezané a ani jedna z nich nie je vycentrovaná. Poďme to napraviť. Zmeňte atribúty ***x*** a ***y*** na:

```
.attr("x", function(d, i) {

          return i * (w / dataset.length) + 5;
    })
    .attr("y", function(d) {
          return h - (d * 4) + 15;
    });
```

Vo výsledku sú hodnoty zobrazené na lepšom mieste, avšak nie sú veľmi dobré čitateľné.



Musíme ešte zmeniť farbu textu.

```
.attr("font-family", "sans-serif")

    .attr("font-size", "11px")
    .attr("fill", "white");
```

A finálny výsledok by mal vyzerať následovne.