ŽILINSKÁ UNIVERZITA V ŽILINE
FAKULTA RIADENIA A INFORMATIKY

BAKALÁRSKA PRÁCA

Študijný odbor: **Informatika**

Oľga Chovancová

Vizualizácia dát získaných pomocou SCADA systémov s využitím HTML 5 štandartov

Vedúci: **Ing. Juraj Veverka**

Reg.č. xxx/2015 Máj 2015

Abstrakt

CHOVANCOVÁ OĽGA: *Vizualizácia dát získaných pomocou SCADA systémov s využitím HTML 5 štandartov* [Bakalárska práca]
Žilinská Univerzita v Žiline, Fakulta riadenia a informatiky, Katedra softvérových technológií.
Vedúci: Ing. Juraj Veverka
Stupeň odbornej kvalifikácie: Inžinier v Telecommunication, Electrical Engineering 1995 – 2000 todo Solution Design Architect v Ipesofte TODOO

Obsahom práce je vzorová sada grafických komponentov na vizualizáciu technologických procesov s využitím HTML 5 štandardov. Jedná sa o grafické komponenty, ktoré nie sú bežne dostupné na tvorbu interaktívnych webových aplikácii ako napríklad vizualizácie mechanických súčasti hydraulických systémov, technologických liniek, silových a výkonových častí automatizačných sústav. Návrh interface, pomocou, ktorého budú tieto komponenty komunikovať so serverovou časťou SCADA systému. Cieľová platforma pre výslednú webovú aplikáciu bude kompatibilná s rodinou štandardov HTML 5 pre každý webový prehľadávač.

Abstract

CHOVANCOVÁ OĽGA: *Data visualization acquired by SCADA systems using HTML5 standarts*
[Bacalar thesis]
University of Žilina, Faculty of Management Science and Informatics, Department of TODO.
Tutor: Ing. Juraj Veverka.
Qualification level: Engineer in field ..... Žilina: TODO
TODO

The main idea of this ... TODO

Prehlásenie

Prehlasujem, že som túto prácu napísala samostatne a že som uviedola všetky použité pramene a literatúru, z ktorých som čerpala.

V Žiline, dňa DD.MM.2015 Oľga Chovancová

Obsah

Úvod

Téma práce je vizualizácia technologických dát zo SCADA systémov na webe. Produktom Bakalárskej práce je vzorová sada grafických komponent na vizualizáciu technologických procesov s využtím HTML 5 štandartov. Jedná sa o grafické komponenty, ktoré nie sú bežne dostupné na tvorbu interaktívnych webových aplikácií ako napríklad vizualizácie mechanických súčastí hydraulických systémov, alebo technologických liniek, vizualizácie silových a výkonových častí automatizačných sústav. Návrh interface, pomocou ktorého budú tieto komponenty komunikovať so serverovou časťou SCADA systému.

V súčasnosti je v IPESOFT s.r.o. software, ktorý dokáže vizualizovať dáta z technológii pomocou ”hrubých klientov”, čo sú natívne (exe) Windows aplikácie a je technológia, ktorá dokáže rovnaké dáta zobrazovať na webe.

Aktuálna webová prezentácia takýchto dát nespĺňa súčasne štandardy pre moderne webové aplikácie a preto je potrebne nájsť nový spôsob vizualizácie na webe, ktorá bude v budúcnosti použiteľná na rôznych platformách, nielen na PC.

Cieľová platforma pre výslednú webovú aplikáciu bude každý web prehliadač kompatibilný s rodinou štandardov HTML 5. Riešenie bude využívať výhradne open-source knižnice s licenciami typu MIT, GNU GPL, BSD. Zdrojové kódy práce budú udržiavané v Git repository.

Predbežný postup práce:

1. Analýza požiadaviek, prieskum možnosti využitia wyswing editorov na tvorbu grafických komponent s možnosťou exportu do formátov SVG, JSON, XML, alebo JavaScript.

2. Výber vhodných open-source knižníc na tvorbu grafických komponent kompatibilných s HTML 5.

3. Návrh REST API na prepojenie grafických komponent so SCADA serverom.
4. Analýza možnosti automatického mapovania API grafických prvkov pomocou metadát na existujúce API dostupné pre SCADA server D2000.
5. Implementácia vzorovej sady grafických komponent.
6. Analýza výkonnosti a výkonnostné obmedzenia.

Kapitola 1

Analýza požiadaviek

Editory, ktoré umožnujú tvorbu grafických komponentov:

- Adobe Illustrator,
- CorelDraw,
- Inkscape.

Nástroj, ktorý najviac vyhovuje mojim požiadavkam je Inkscape.

Na internete sa nachádzajú tieto JavaScriptové knižnice na tvorbu grafických komponentov:

- D3.js (Data Driven Document),
- Raphael.js,
- Snap.svg.js,
- Svg.js.

Rozhodla som sa použiť knižnicu Snap.svg.js hlavne pretože dokáže načítať a ovládať SVG komponenty.

1.1 JavaScript knižnice SVG

1.1.1 D3

D3 - **D**ata **D**riven **D**ocument - http://d3js.org/

D3.js is a JavaScript library for manipulating documents based on data. D3 helps you bring data to life using HTML, SVG and CSS. D3's emphasis on web standards gives you the full capabilities of modern browsers without tying yourself to a proprietary framework, combining powerful visualization components and a data-driven approach to DOM manipulation.

D3 allows you to bind arbitrary data to a Document Object Model (DOM), and then apply data-driven transformations to the document. For example, you can use D3 to generate an HTML table from an array of numbers. Or, use the same data to create an interactive SVG bar chart with smooth transitions and interaction.

D3 is not a monolithic framework that seeks to provide every conceivable feature. Instead, D3 solves the crux of the problem: efficient manipulation of documents based on data. This avoids proprietary representation and affords extraordinary flexibility, exposing the full capabilities of web standards such as CSS3, HTML5 and SVG. With minimal overhead, D3 is extremely fast, supporting large datasets and dynamic behaviors for interaction and animation. D3's functional style allows code reuse through a diverse collection of components and plugins.

1.1.2 Raphaël

http://raphaeljs.com/

Raphaël is a small JavaScript library that should simplify your work with vector graphics on the web. If you want to create your own specific chart or image crop and rotate widget, for example, you can achieve it simply and easily with this library.

Raphaël uses the SVG W3C Recommendation and VML as a base for creating graphics. This means every graphical object you create is also a DOM object, so you can attach JavaScript event handlers or modify them later. Raphaël's goal is to provide an adapter that will make drawing vector art compatible cross-browser and easy.

Raphaël currently supports Firefox 3.0+, Safari 3.0+, Chrome 5.0+, Opera 9.5+ and Internet Explorer 6.0+.

1.1.3 Snap.svg

http://snapsvg.io/

Snap.svg is a brand new JavaScript library for working with SVG. Snap provides web developers with a clean, streamlined, intuitive, and powerful API for animating and manipulating both existing SVG content, and SVG content generated with Snap.

Currently, the most popular library for working with SVG is Raphaël. One of the primary reasons Raphaël became the de facto standard is that it supports browsers all the way back to IE 6. However, supporting so many browsers means only being able to implement a common subset of SVG features. Snap was written entirely from scratch by the author of Raphaël (Dmitry Baranovskiy), and is designed specifically for modern browsers (IE9 and up, Safari, Chrome, Firefox, and Opera). Targeting more modern browsers means that Snap can support features like masking, clipping, patterns, full gradients, groups, and more.

Another unique feature of Snap is its ability to work with existing SVG. That means your SVG content does not have to be generated with Snap for you to be able to use Snap to work with it (think "jQuery or Zepto for SVG"). That means you create SVG content in tools like Illustrator, Inkscape, or Sketch then animate or otherwise manipulate it using Snap. You can even work with strings of SVG (for example, SVG files loaded via Ajax) without having to actually render it first which means you can do things like query specific shapes out of an SVG file, essentially turning it into a resource container or sprite sheet.

Finally, Snap supports animation. By providing a simple and intuitive JavaScript API for animation, Snap can help make your SVG content more interactive and engaging.

Snap is free and open-source (released under an Apache 2 license).

1.1.4 SVG.JS

http://www.svgjs.com/

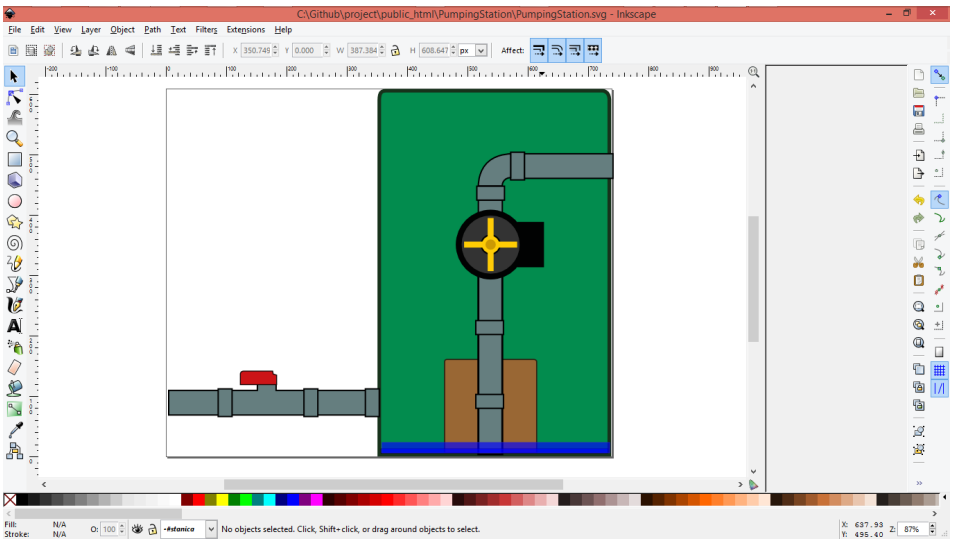
A lightweight library for manipulating and animating SVG.

- easy readable uncluttered syntax
- animations on size, position, transformations, color, ...
- painless extension thanks to the modular structure
- various useful plugins available
- unified api between shape types with move, size, center, ...
- binding events to elements
- full support for opacity masks and clipping paths
- text paths, even animated
- element groups and sets
- dynamic gradients

Kapitola 2

Popis vytvorenia SVG v Inkscape

Vytvorenie SVG v programe Inkscape .



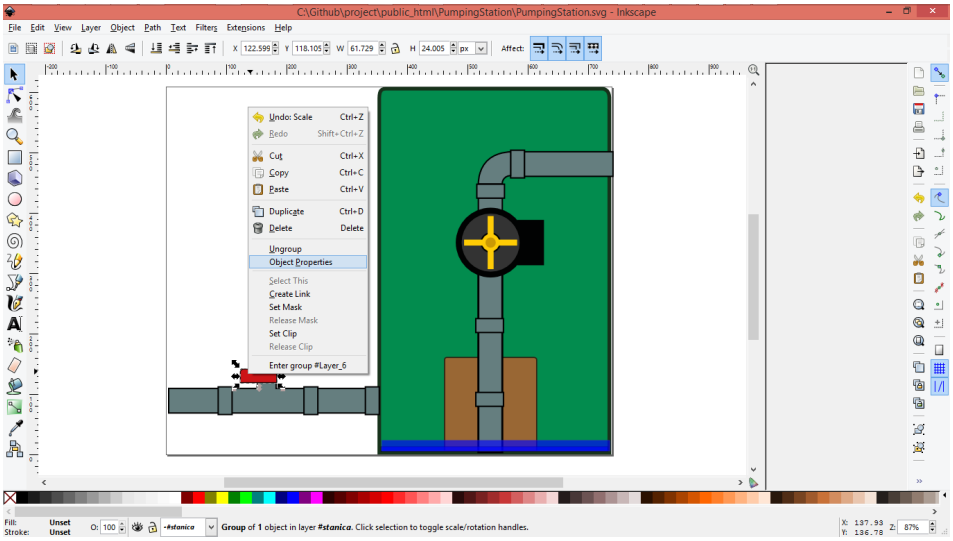
Obr. 2.1: obrázok 1

Nakreslenie jednotlivých častí komponentov pomocou bočného panela.

Pre ovládanie JavaScriptom je nutné si pozrieť jednotlivé ID SVG Klikneme pravým tlačidlom na daný komponent, časť, a potom na Objekt Properties.

Zobrazí sa nám nasledovné okno obrázok č.x.

Z obrázka možno vyčítať aké je ID, predvolené sú tam napr. desc3072. Hodnoty je možné zmeniť tlačidlom Set. Pre nás je dôležitá hodnota v kolónke Label - #ventil. Toto nám umožni



Obr. 2.2: obrázok 2

potom neskôr ako CSS selektor, cez ktorý budeme môcť ovládať danú časť. Spravidla hodnoty ID a Label sú rovnaké, a líšia sa iba v #. ID je unikátny názov pre danú vetvu SVG

Alebo ďalší spôsob zistenia ID SVG je priamo nájsť tú hodnotu v nazovSuboru.SVG Je to označené ako ID=”ventil”.

Plná nádrž ma nasledovne parametre:

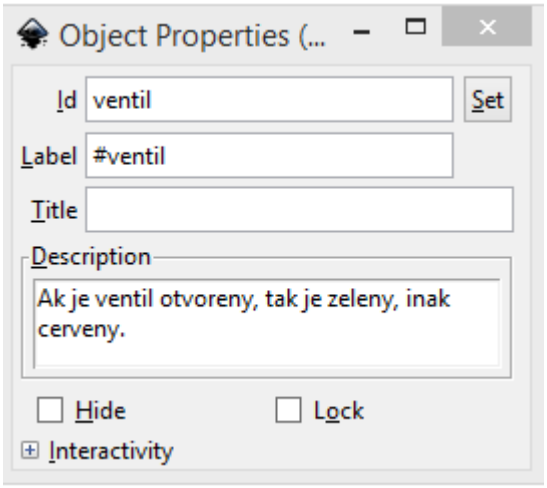
V SVG súbore je to

```
Inkscape:label="\#hladina"
y="1320.1689"
x="2507.8459"
height="605.83868"
width="797.04492"
id="hladina"
```

Prázdna nádrž

v SVG to je nasledovne

```
<rect
inkscape:label="\#hladina"
```



Obr. 2.3: obrázok 3

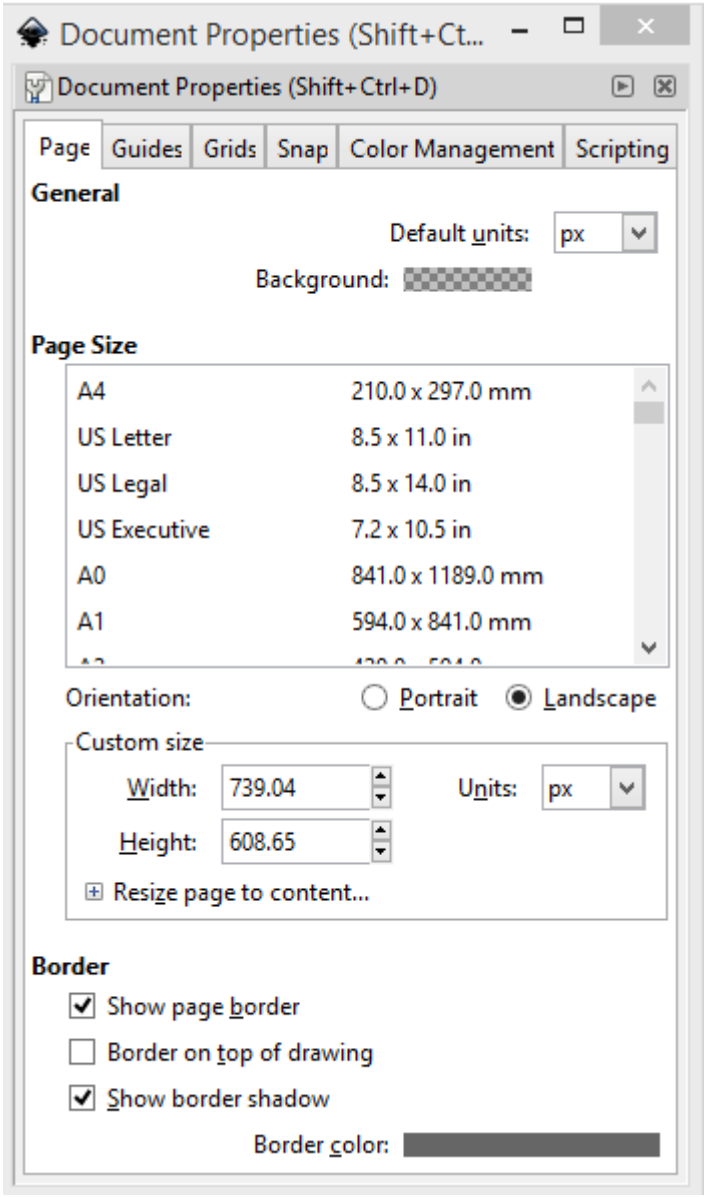
```
y="1916.3605"
x="2507.8459"
height="9.6471272"
width="797.04492"
id="hladina"
...
```

Hladina nádrže: vykreslená ako obdĺžnik

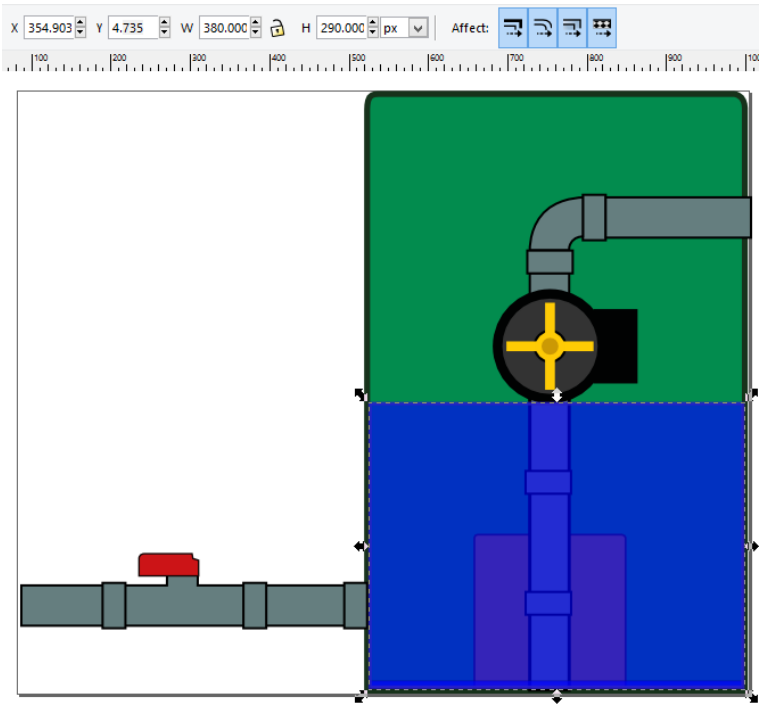
```
<rect
        inkscape:label="#hladina"
        y="1320.1689"
        x="2507.8459"
        height="605.83868"
        width="797.04492"
        id="hladina"
        style=
                "fill:#0000ff;
                fill-opacity:0.65098039;
                fill-rule:evenodd;
                stroke:#2c20c8;
                stroke-width:7.42523718px;
                stroke-linecap:butt;
```

```
                stroke-linejoin:miter;
                stroke-opacity:0.80952382">
<desc id="desc3119-4">hladina</desc>
<title id="title3117-0">hladina</title>
</rect>
```

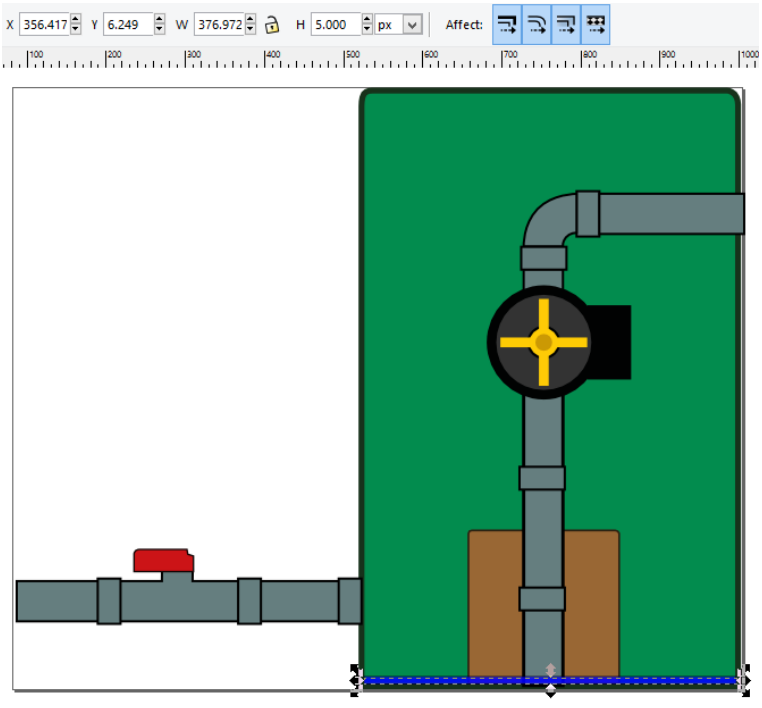
Parametre ako stroke, fill, a iné sa dajú meniť prostredníctvom attr v Snap. . . . TODO



Obr. 2.4: obrázok 4



Obr. 2.5: obrázok 5



Obr. 2.6: obrázok 6

Kapitola 3

js

Keď už máme nakreslený komponent pomocou Inkscape, tak postupujem dalej. TODO

Pridáme do HTML

```
<svg
        id="svgStanica"
        viewBox="0 0 750 600"
        width="40%"
        height="40%"
        >
</svg>
```

id / toto pouzijeme pri vykresleni viewBox - je atribút, ktorý povoľuje špecifikovať danú množinu grafick.. aby to fit a vošlo do kontajnera elementu. Hodnoty atribútov v viewBox sú štyri čísla - min-x, min-y, width a height. Width a height je šírka a výška - a je možné ich uviesť aj relatívne v percentách, alebo absolútne v pixeloch.

```
<body onload="onPageLoad();">

function onPageLoad() {
        PumpingStation("PumpingStation.svg", "#svgStanica" );
}
```

Parametre pre PumpingStation je nazov svg suboru, a tag v html.

```
var PumpingStation = function(nazovFileSVG, nameHTMLidSVG) \{
        paper = Snap(nameHTMLidSVG);
        Snap.load(nazovFileSVG, function (f) \{
                paper.append(f);
\});
\};
```

paper - bude globalna premenna. Vytvorí plochu na kreslenie, alebo wraps existujúci SVG element. Ako parametre môžu byť buď šírka, výška, alebo DOM element.

Pomocou load nacitam svg subor, ktory som vytvorila a pomocou append ho zobrazim na danu plochu.

3.0.5 Tank

Zanimovanie stupania a klesania hladiny nadrze.

```
var Tank = {
        idTank: "#hladina",
        tank: function(){
                return  paper.select(this.idTank);},
        animateComponentTank: function(fillPerc) {
                if (fillPerc === undefined || fillPerc < 0) {
                        fillPerc = 0;
                }
                var perHeight = 600 * (fillPerc / 100);
                var perY = 1912 - perHeight;
                this.tank().animate  (  {
                        height: perHeight,
                        y: perY
                }, 800);
                return console.log("animacia tanku " + fillPerc);
        }
};
```

Vytvorila som objekt Tank medzi jeho atributy patria: idTank, funkcia tank, a animateComponentTank. IdTank - je stringové - je to id, ktoré som získala zo svg súboru, alebo cez Inkscape

ako Label. Funkcia tank - vyberie dany objekt, ktory chcem ovladat. Pomocou Tank.tank() mozem volat funkcie z Snap kniznice. n

Zanimovanie tanku je realizovane v funkcii animateComponentTank - kde parametrom je v percentách udane o koľko sa ma zdvihnut hladina nadrze. Využívam funkciu animate. Kde v prvom parametri - mením výšku a os y. Hodnotu perHeight je výška 600, ktoru vynásobím percentom o ktore sa ma posunut. PerY je hodnota, o ktoru sa posunei po y-osi. Je vypocitana ako 1912 co je y prázdnej nádrže a je od nej odpocitana hodnota vysky. Dalsi parameter pri funkcii animate() je rýchlosť animácie vyjadrená v milisekundach.

3.0.6 Ventil

```
var Valve = {
        idValve: "#ventil",
        valve: function (){ return paper.select(this.idValve);},
        colorValve: "red",
        changeIsOpen: function (isOpened) {
                isOpened = (isOpened) ? 0 : 1;
                this.colorValve = (isOpened) ?   "red" : "green";
                this.valve().attr({fill: this.colorValve});
                return;);
        }
}
```

Farba sa dá zmeniť aj príkazom
`Valve.valve().attr({fill: \green"});.`

Názov farby môže byť uvedený slovne, alebo ako RGB.

Zmena farby Valve -

```
this.valve().attr  ({fill: this.colorValve});
```

Kapitola 4

Analýza požiadaviek

Kapitola 5

Analýza požiadaviek

Literatúra

[1] Dawber D., *Learning Raphael JS Vector Graphics* , Packt Publishing 2013, ISBN 978-1-78216-916-1.

[2] Wilson CH., *RaphaelJs Graphic and visualization on the web* , O'Reilly Media 2013, ISBN 978-1-449-36536-3.

[3] Haverbeke M., *Eloquent Javascript* 2 edition, No Starch Press 2014, ISBN 978-1-59327-584-6.

[4] Zakas N. Z., *JavaScript pro webové vývojáře Programujeme profesionálně*, vydanie prvé, Brno, Computer Press, a.s., 2009, ISBN 978-80-251-2509-0.

[5] Suehring S., *JavaScript krok za krokem*, vydanie prvé, Brno, Computer Press, a.s., 2008, ISBN 978-80-251-2241-9.

[6] Zakas N. C., McPeak J., Fawcett J., *Profesionálně Ajax*, Zoner Press 2007, ISBN 978-80-86815-77-0.

[7] Eisenberg D. J., *SVG Essentials*, O'Reilly Media 2002, ISBN 978-0-596-00223-7, dostupné na `http://commons.oreilly.com/wiki/index.php/SVG_Essentials`

[8] The JavaScript SVG library for the modern web, `http://snapsvg.io/`.

[9] `http://raphaeljs.com/`

[10] `http://d3js.org/`

[11] Inkscape is a professional vector graphics editor for Windows, Mac OS X and Linux. It's free and open source. `http://www.inkscape.org/en/about/features/`

Zoznam obrázkov