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

Zoznam obrázkov

Zoznam tabuliek

Listings

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 využitím HTML 5 štandardov. 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 WYSIWYG

   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

Základné pojmy

1.1 HTML 5 štandardy

1.2 What is SVG?

Scalable Vector Graphics (SVG) je aplikácia XML, ktorá umožnuje reprezentáciu grafických informácii v kompaktnom, prenostiteľnom tvare.

- SVG stands for Scalable Vector Graphics
- SVG is used to define graphics for the Web
- SVG is a W3C recommendation

The HTML ¡svg¿ Element The HTML ¡svg¿ element (introduced in HTML5) is a container for SVG graphics. SVG has several methods for drawing paths, boxes, circles, text, and graphic images.

1.2.1 Browser Support

The numbers in the table specify the first browser version that fully supports the ¡svg¿ element. Element ¡svg¿ 4.0 9.0 3.0 3.2 10.1

1.2.2 Differences Between SVG and Canvas

SVG is a language for describing 2D graphics in XML. Canvas draws 2D graphics, on the fly (with a JavaScript). SVG is XML based, which means that every element is available within the SVG DOM. You can attach JavaScript event handlers for an element. In SVG, each drawn shape is remembered as an object. If attributes of an SVG object are changed, the browser can automatically re-render the shape. Canvas is rendered pixel by pixel. In canvas, once the graphic is drawn, it is forgotten by the browser. If its position should be changed, the entire scene needs to be redrawn, including any objects that might have been covered by the graphic.

1.2.3 Comparison of Canvas and SVG

The table below shows some important differences between Canvas and SVG: Canvas

1. Resolution dependent
2. No support for event handlers
3. Poor text rendering capabilities
4. You can save the resulting image as .png or .jpg
5. Well suited for graphic-intensive games

SVG

1. Resolution independent
2. Support for event handlers
3. Best suited for applications with large rendering areas (Google Maps)
4. Slow rendering if complex (anything that uses the DOM a lot will be slow)
5. Not suited for game applications

Kapitola 2

Analýza požiadaviek

Editory, ktoré umožňujú 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.

2.1 JavaScript knižnice SVG

2.1.1 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 acs{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 ac*{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.

2.1.2 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+.

2.1.3 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).

2.1.4 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 3

Knižnica Snap.svg

CSS selektory ...

a podrobnejší popis tejto knižnice.. .

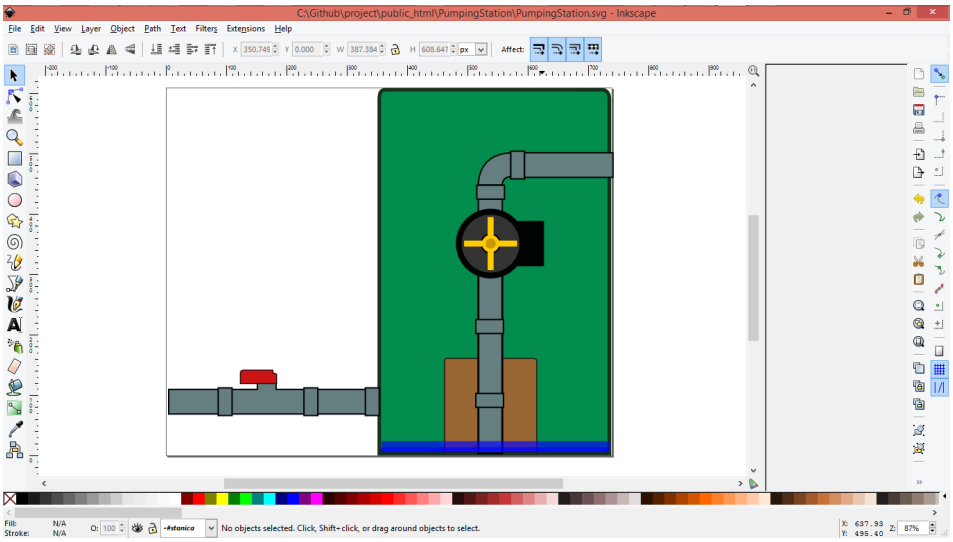
mapovacia tabuľka.. .

z dokumentácie Snap.svg vybrat zopár príkazov, funkcií - príkladov...

Kapitola 4

Tvorba grafických komponentov v Inkscape

Vytvorenie SVG v programe Inkscape .

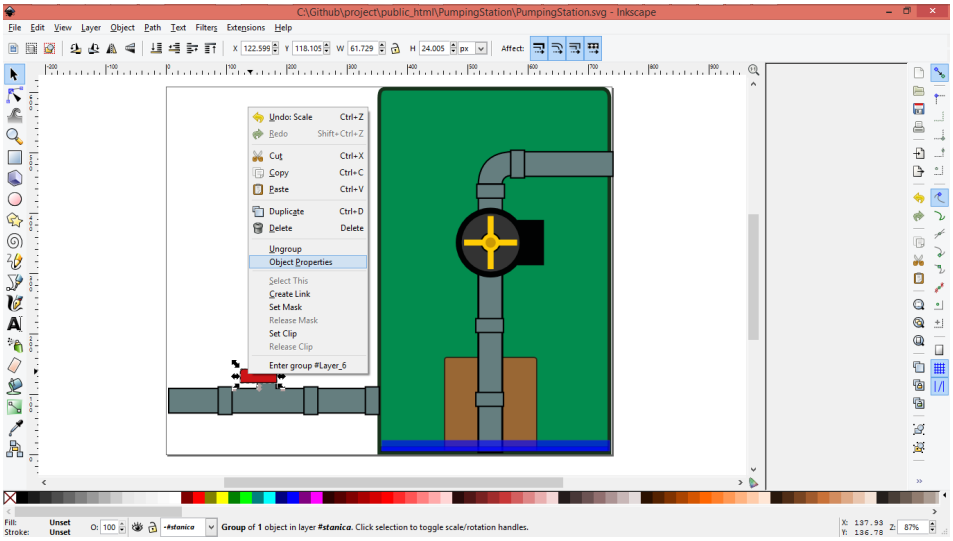


Obr. 4.1: obrázok 1

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.



Obr. 4.2: obrázok 2

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 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”.

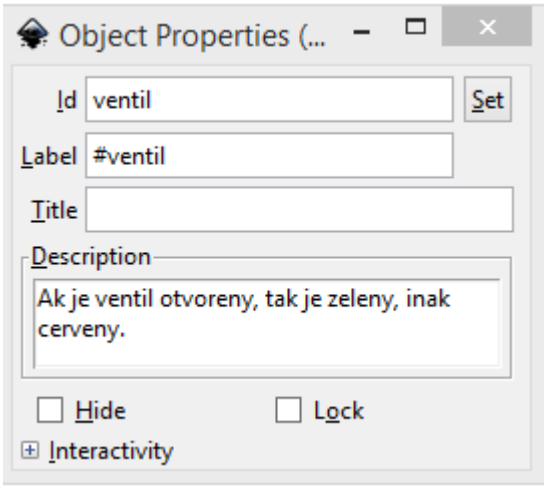
Plná nádrž ma nasledovne parametre:

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



Obr. 4.3: obrázok 3

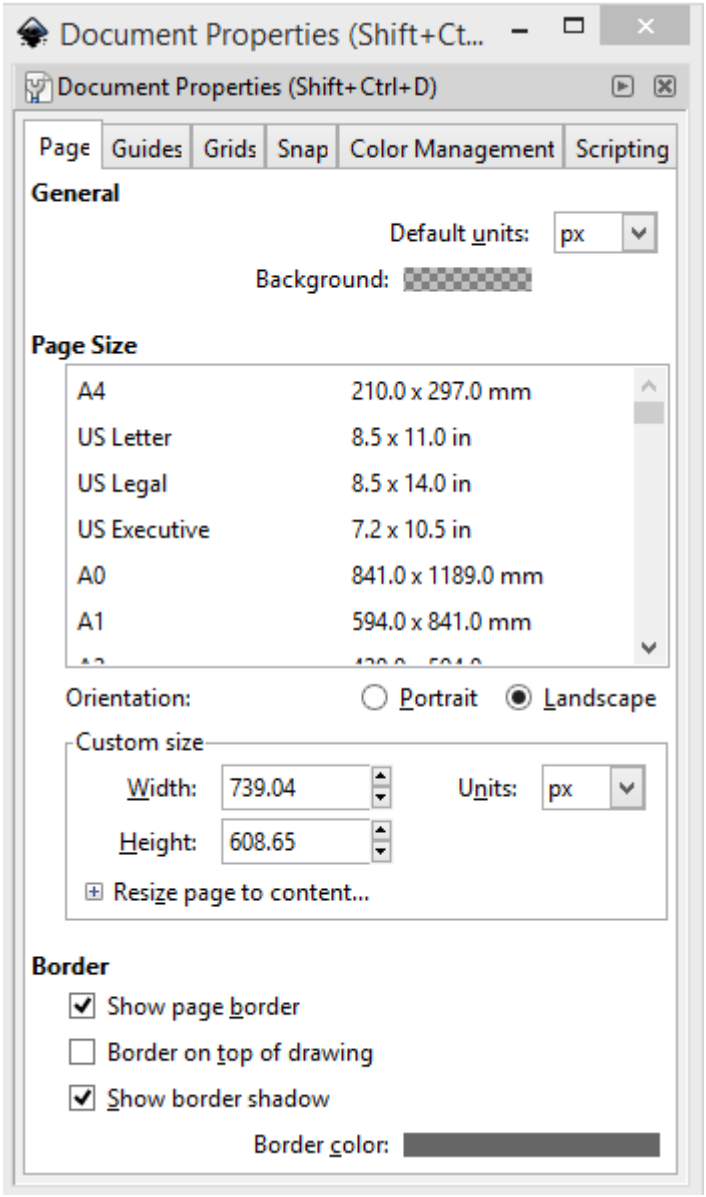
```
<rect
inkscape:label="\#hladina"
y="1916.3605"
x="2507.8459"
height="9.6471272"
width="797.04492"
id="hladina"
...
```

Hladina nádrže: vykreslená ako obdĺžnik

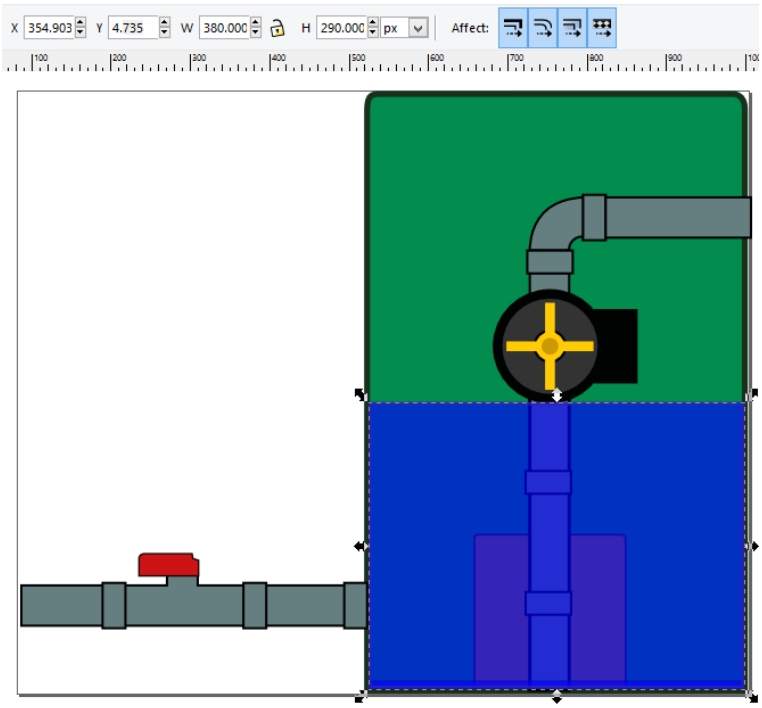
```
<rect
        inkscape:label="#hladina"
        y="1320.1689"
        x="2507.8459"
        height="605.83868"
        width="797.04492"
        id="hladina"
        style=
                "fill:#0000ff;
                fill-opacity:0.65098039;
                fill-rule:evenodd;
                stroke:#2c20c8;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

```
                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>
```

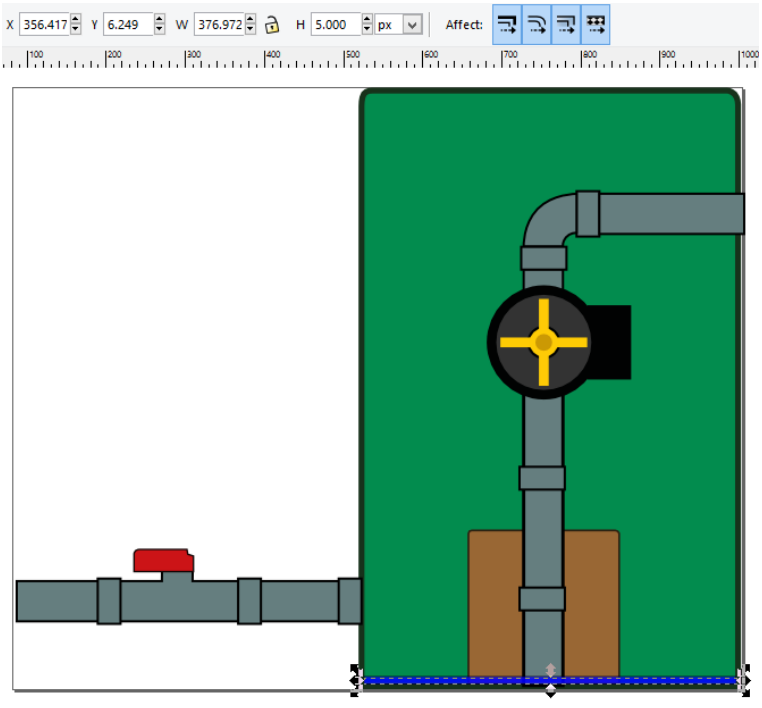
Parametre ako stroke, fill, a iné sa dajú meniť prostredníctvom attr v Snap. . . . TODO



Obr. 4.4: obrázok 4



Obr. 4.5: obrázok 5



Obr. 4.6: obrázok 6

Kapitola 5

Implementácia komponentov

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

Pridáme do HTML

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

id / toto použijeme pri vykreslení viewBox - je atribút, ktorý povoľuje špecifikovať danú množinu grafických. 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 pixloch

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

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

Parametre pre PumpingStation je názov svg súboru, a tag v html.

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

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

Pomocou load načítam vytvorený svg súbor. Na plochu ho zobrazím pomocou príkazu append.

5.0.5 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 atribúty patria: idTank, funkcia tank, a animate-ComponentTank. IdTank - je stringové - je to id, ktoré som získala zo svg súboru, alebo

cez Inkscape ako Label. Funkcia tank - vyberie daný objekt, ktorý chcem ovládať. Pomocou Tank.tank() môžem volať funkcie z Snap knižnice. n

Zanimovanie tanku je realizované v funkcii animateComponentTank - kde parametrom je v percentách udané o koľko sa ma zdvihnúť hladina nadrze. Využívam funkciu animate. Kde v prvom parametri - menim výšku a os y. Hodnotu perHeight je výška 600, ktorú vynásobím percentom o ktoré sa ma posunúť. PerY je hodnota, o ktorú sa posuniem po y-osi. Je vypočitaná ako 1912 co je y prázdnej nádrže a je od nej odpočítaná hodnota výšky. Ďalší parameter pri funkcii animate() je rýchlosť animácie vyjadrená v milisekundách.

5.0.6 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 6

Návrh REST API

JSON

23 strana knihy restful web apis

Kapitola 7

Automatické mapovanie

Analyzujte možnosti automatikcého mapovania API grafikcých prvkov pomocou metadát na existujúce API dostupné pre SCADA server D2000

Kapitola 8

Implementácia komponentov

Kapitola 9

Výkonnosť a obmedzenia SVG

Záver

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] Richardson L., Amundsen M., *RESTful Web APIs* vydanie prvé, O'Reilly Media 2013, ISBN 978-1-449-35806-8

[9] Allamaraju S., *RESTful Web Services Cookbook* vydanie prvé, O'Reilly Media 2010, ISBN 978-0-596-80168-7

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

[11] http://raphaeljs.com/

[12] http://d3js.org/

[13] 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 skratiek

RGB Red Green Blue

XML EXtensible Markup Language

SVG Scalable Vector Graphics

SCADA Supervisory Control and Data Acquisition

HTML Hyper Text Markup Language

API Application Programming Interface

REST Representational State Transfer

JSON JavaScript Object Notation

W3C World Wide Web Consortium

DOM Document Object Model

CSS Cascading Style Sheets

D3 Data Driven Document

WYSIWYG What You See Is What You Get

Zoznam termínov

IPESOFT D2000® je objektovo orientovaný SCADA (Supervisory Control And Data Acquisition) systém, ako aj platforma pre tvorbu komplexných MES (Manufacturing Execution System) aplikácií. V súhrne svojich vlastností predstavuje optimalizovaný nástroj triedy RAD (Rapid Application Development) pre informačné systémy pracujúce súčasne s údajmi technického charakteru v reálnom čase, technickými a obchodnými údajmi vo forme časových radov a obchodnými údajmi vo forme databázových tabuliek.