

# DELETE THIS PAGE

## 0.1 Poznámky - informačný účel

Táto strana slúži na poznámky, TODO, toho čo mám ešte urobiť. Bude samozrejme zmazaná. Dany text, ktorý sa bude dať použiť v práci bude presunutý do jadra práce.

doplniť do praktickej časti mapy: využívajúc tento príklad ovládania.. <http://raphaeljsvectorgraphics.com/the-graphical-web/turtle-graphics-logo> pozrieť si funkciu snap - inAnim()...

Pridať postup ako sa získava Snap zo stránky a implementuje do html dokumentu. .  
ujednotniť názov atribút a parameter

## 0.2 Kapitoly by mali byť nasledovne

1. cieľ práce
2. metodika práce
3. definovanie základných pojmov / čo je html5, čo je scada system, a čo je svg vektorová grafika (takmer hotové, len nemám scada systémy definované),
4. analýza nástrojov - predtým som sa rozhodla použiť inkscape na tvorbu svg obrázkov, //este tam musím spomenúť možnosť exportu do formátov...
5. postup na vytvorenie OBRAZKA SVG V INKSCAPE
6. príklad vytvorenia precepvacej stanice v Inkscape (toto je už takmer hotové)
7. ANALÝZA javascriptových knižníc / (toto už je takmer hotové, este tam pridať jquery )
8. Postup **tvorby** grafických komponentov cez Snap...
9. Postup **animácie** už vytvorených grafických elementov

10. Príklad v kóde animovanie Precerpavacej stanice... (Takmer hotové) toto je vlastne  
ta implementácia vzorovej sady grafických komponentov
11. Navrh REST API na prepojenie grafických komponentov so SCADA SERVEROM  
*toto no nemam / mam iba príklad kódu jednoducheho*
12. Analýza možnosti automatickeho mapovania API grafických prvkov pomocou me-  
tadat na existujúce API dostupné pre SCADA server D2000 *toto nemam vôbec*
13. analýza výkonnosti a výkonnostné obmedzenia - toto iba okrajovo spomením -  
všeobecne prečo je lepšie SVG, a prečo nie je vhodné, v prvej kapitole som písala  
rozdiel medzi canvas a SVG - istým spôsobom to je analýza obmedzení.. ?
14. zhrnutie

## 0.3 Learning Raphael JS Vector Graphics

## 0.4 Porovnanie spôsobu vykreslenia cez SVG SMIL a Snap

Kreslenie vektorov je jednoduchšie cez Snap ako čisto písanie SVG.

Príklad kreslenia obdĺžníka a animovanie šírky z 50 pixlov na 100 pixelov cez SVG SMIL:[2, p. 9]

```
<svg>
<rect x="10" y="10" width="50" height="30">
  <animate attributeType="XML"
    attributeName="width"
    to="100"
    fill="freeze"
    dur="10s" />
</rect></svg>
```

Nakreslíme obdĺžnik na súradniciach (10, 10) s šírkou 50, a výškou 30 použitím elementu `<rect>`. Zoskupený element `<animate>` definuje animáciu zmeny šírky obdĺžníka na šírku 100 px, ktorá trvá desať sekúnd. Kde `fill="freeze"` je použité na zachovanie stavu obdĺžníka po ukončení animácie. Inak by bola nastavená na 50.

Ekvivalent k animácii cez Snap API v nasledujúcom príklade:

```
paper = Snap();
var rect = paper.rect(10, 10, 50, 30);
rect.animate({
  width: 100
}, 10000);
```

Syntax metód `animate` a `rect` je výstižnejšia a lepšia na pochopenie. Snap sa tiež dobre integruje s inými knižnicami, ako napríklad jQuery.

## 0.5 Krok 1: Inicializácia plátna na kreslenie

Na to, aby sme boli schopní kresliť grafické komponenty, tak potrebujeme definovať miesto, kde budú vykreslené. Viditeľná oblasť okna prehliadača, alebo viewport, definuje oblasť, v ktorej sa vykreslí komponent na plátno. SVG špecifikácia referuje ako miesto vykreslenia seba ako viewport. Inak povedané viewport je akákoľvek obdĺžniková oblasť. Okno prehliadača je referencia na viewport a kresliaca oblasť je plátno. [2]

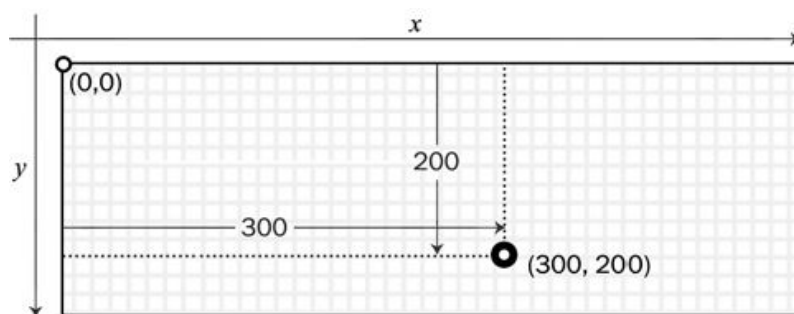
Vytvorenie plátna cez Snap konštruktor sa dá urobiť viacerými spôsobmi.

### 0.5.1 Súradnice plátna

Nasledujúci príkaz zadefinuje pláno s rozmermi šírka je 300 a výška 200.

```
var paper = Snap(300, 200);
```

Na obrázku 1 je znázornená východzí súradnicový systém plátna vytvoreného cez Snap konštruktor. Začiatok súradníc na osi x, y je rovné nule. Bod na plátne so súradnicami  $x = 300$ ,  $y = 200$  alebo  $(300, 200)$  vo vektorovom zápise je bod 300px vpravo od začiatku x-ovej osi a 200px dole od počiatku y-ovej osi.



Obr. 1: Súradnicový systém plátna s bodom  $(300, 200)$

### 0.5.2 DOM element

Dost často je potrebné použiť existujúci DOM element ako kontajner pre plátno než viewport. Ako element môžeme použiť napríklad:

```
<div id="mojePlatno"></div>
```

Nasledujúcim kódom vytvorím 500px široké a 300px vysoké plátno.

```
var paper = Snap("mojePlatno", 500, 300);
```

Keď využívam túto formu konštruktora, tak prvý element je ID elementu. Alternatívne sa dá prvý parameter DOM element napísať nasledovným spôsobom:

```
Snap(document.getElementById('mojePlatno'), 500, 300);
```

## SVG v HTML dokumente

SVG môže byť zobrazená buď ako inline v HTML dokumente, alebo ako vložený samostatného .SVG súboru. V tabuľke 1 sú vymenované HTML tagy na zobrazenie SVG.

Technika	Popis
<embed> tag	Načíta vytvorený SVG súbor.
<object> tag	Nepovoľuje skriptovanie.
<iframe> tag	Zobrazí SVG v rámci
Inline <svg> tag	Vytvorí Svg

Tabuľka 1: Spôsoby vytvorenia SVG v HTML dokumente

## 0.6 Kreslenie základných tvarov cez Snap

Snap API poskytuje metódy na kreslenie jednoduchých tvarov.

Tvar	SVG element	Snap metoda	Atribúty
Obdĺžnik	<rect>	.rect()	x, y, šírka, výška, rx, ry
Kruh	<circle>	.circle()	r, x, y, cx, cy, rx, ry
Elipsa	<ellipse >	.ellipse()	x, y, cx, cy, rx, ry
Čiara	<line>	.line()	x1, y1, x2, y2
Polyline	<polyline>	.polyline()	pole x, y suradnic bodov
Polygon	<polygon>	.polygone()	pole x, y suradnic bodov
Path	<path>	.path()	vid tabuľka 4

Tabuľka 2: Zoznam tvarov, ktoré podporuje SVG a Snap API, a TODO TODO PORO-ZMYSLAT NAD NAZVOM a atribúty pre definovanie tvaru

Tvar, ktorý je vykreslený cez Snap API má nasledovnú syntax:

```
var paper = Snap (...);  
var tvar = paper.NazovSnapMetody({  
    nazovAtributu: "hodnotaAtributu",  
    ...  
});
```

Tvar, ktorý je vykreslený priamo na HTML webovej stránke má vo vnútri elementu <svg> definované atribúty nasledujúcim spôsobom:

```
<ElementTvar nazovAtributu = "hodnotaAtributu" ... />
```

### 0.6.1 Popis atribútov tvarov

Názvy atribútov a ich význam pre obdĺžnik, kruh, elipsu sú vyjadrené v tabuľke 3

Pre útvary polyline, polygon sú atribúty dvojice súradníc osi x, y, ktoré určujú body, ktoré sa spoja.

Parameter	Poznámka
x, y	súradnica x-osi, y-osi
cx	x-os súradnica centra kruhu, alebo elipsy
cy	y-os súradnica centra kruhu, alebo elipsy
r	polomer kruhu, elipsy alebo okruhlých rohov na obdĺžniku
rx	horizontálny polomer elipsy
ry	vertikálny polomer elipsy
x1, y1	začiatočné x, y súradnice
x2, y2	konečné x, y súradnice

Tabuľka 3: Názvy atribútov a ich význam

### Path tvar

V Snap API je to metóda `Paper.path([pathString])`, ktorá vytvorí `<path>` element podľa daného reťazca. Parameter `pathString` pozostáva reťazca skladajúceho sa z jedno písmenkových príkazov, nasledovaných bodkami a oddelovaný argumentami a číslami. Príkazy sú uvedené v tabuľke 4.

Napríklad: "M10,20L30,40 obsahuje príkazy: M s argumentami (10, 20) a L (30, 40). Rozdiel vo veľkosti písma vyjadruje to, či ide o absolútnu, alebo relatívnu cestu. Ak sú malé znaky jedná sa o relatívne, v prípade veľkých znakov absolútna cesta.

Príkaz	Názov	Parametre
M	moveto	(x y)+
Z	closepath	(none)
L	lineto	(x y)+
H	horizontal lineto	x+
V	vertical lineto	y+
C	curveto	(x1 y1 x2 y2 x y)+
S	smooth curveto	(x2 y2 x y)+
Q	quadratic Bézier curveto	(x1 y1 x y)+
T	smooth quadratic Bézier curveto	(x y)+

Tabuľka 4: Niekoľko príkazov na tvorbu Path elementu



TOTO BUDE PRIKLAD KED BUDEM MAT UZ NAPISANE ATTRIBUTY NA ZMENU STYLU

## PRIKLAD TVORBY KRUHU A NASTAVENIE ATRIBUTOV

Kód vytvoreného kruhu:

```
<svg width="100" height="100">
<circle cx="50" cy="50" r="40" stroke="black" stroke-width="2"
fill="silver" />
</svg>
```

SVG obrázok začína s `<svg>` elementom. Atribúty elementu `<svg>` sú `width` a `height`. Definujú šírku a výšku SVG obrázka. Element `<circle>` je použitý na nakreslenie kruhu.

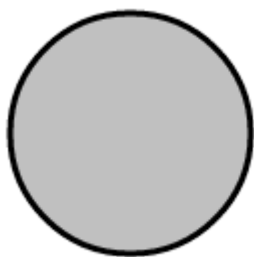
TODO TODO TODO

Atribúty `stroke` a `stroke-width` určujú to ako bude vyzerat obrys útvaru. Kruh má nastavený 2px čierny okraj. Atribút `fill` vyplní vnútro kruhu. V príklade je vyplnený sivou farbou. Tag, ktorý uzavrie SVG obrázok je `</svg>`. Keďže SVG je validné XML, tak všetky elementy musia byť správne zatvorené. [13]

Kruh vytvorený cez Snap API má nasledovný kód:

```
var paper = Snap(100, 100);
var kruh = paper.circle(50, 50, 40);
kruh.attr({
  stroke: "black",
  strokeWidth: 2,
  fill: "silver"
});
```

Vykreslí sa na HTML stránku obrázok 2. Obidva spôsoby vykreslili kruh na webovej stránke úplne rovnako.



Obr. 2: Vykreslený kruh vytvořený cez SVG, a Snap API

ŽILINSKÁ UNIVERZITA V ŽILINE  
FAKULTA RIADENIA A INFORMATIKY

## BAKALÁRSKA PRÁCA

Študijný odbor: **Informatika**

**Olga Chovancová**

**Vizualizácia dát získaných pomocou  
SCADA systémov s využitím HTML 5  
štandardov**

Vedúci: **Ing. Juraj Veverka**

Tútor **Ing. Patrik Hrkút, PhD.**

Reg.č. 5/2014

Máj 2015

## **Abstrakt**

CHOVANCOVÁ OĽGA: *Vizualizácia dát získaných pomocou SCADA systémov s využitím HTML 5 štandardov* [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

Tútor Ing. Patrik Hrkút, PhD.

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 aplikácií 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 prehliadač.

## Abstract

CHOVANCOVÁ OEGA: *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 uviedla všetky použité  
pramene a literatúru, z ktorých som čerpala.

V Žiline, dňa DD.MM.2015

Olga Chovancová

# Obsah

0.1	Poznámky - informačný účel . . . . .	1
0.2	Kapitoly by mali byť nasledovne . . . . .	1
0.3	Learning Raphael JS Vector Graphics . . . . .	3
0.4	Porovnanie spôsobu vykreslenia cez SVG SMIL a Snap . . . . .	3
0.5	Krok 1: Inicializácia plátna na kreslenie . . . . .	4
0.5.1	Súradnice plátna . . . . .	4
0.5.2	DOM element . . . . .	4
0.6	Kreslenie základných tvarov cez Snap . . . . .	6
0.6.1	Popis atribútov tvarov . . . . .	6
<b>Úvod</b>		<b>2</b>
<b>1</b>	<b>Základné pojmy</b>	<b>4</b>
1.1	HTML5 štandard . . . . .	4
1.2	Čo je SVG? . . . . .	5
1.2.1	Podpora v webovom prehliadači . . . . .	5
1.2.2	Rozdiely medzi SVG a Canvas . . . . .	5
1.2.3	Príklad použitia SVG v HTML dokumente s inline SVG . . . . .	6
<b>2</b>	<b>Analýza požiadaviek</b>	<b>8</b>
2.1	Nástroje na tvorbu grafických komponentov . . . . .	8
2.2	JavaScriptové knižnice pre grafické komponenty . . . . .	8
2.2.1	D3.js . . . . .	9

2.2.2	Raphaël.js . . . . .	9
2.2.3	Snap.svg.js . . . . .	9
2.2.4	SVG.JS . . . . .	10
2.3	Zhodnotenie požiadaviek . . . . .	10
<b>3</b>	<b>Knižnica Snap.svg.js</b>	<b>11</b>
3.1	append() . . . . .	11
3.2	Element.attr(...) . . . . .	11
3.2.1	Parametre: . . . . .	11
3.2.2	Použitie: . . . . .	11
3.3	Element.animate() . . . . .	12
3.4	Gradients, Path String, Colour Parsing . . . . .	12
3.4.1	Paper.path([pathString]) . . . . .	12
<b>4</b>	<b>Postup vytvorenia komponentov</b>	<b>14</b>
4.1	Použitie SVG v HTML dokumente . . . . .	14
4.1.1	Vytvorenie SVG . . . . .	15
4.2	JavaScript . . . . .	16
<b>5</b>	<b>Príklad vytvorenia grafického komponentu v Inkscape</b>	<b>17</b>
5.1	Vytvorenie SVG v programe Inkscape . . . . .	17
5.2	Definovanie id SVG . . . . .	17
5.2.1	XML Editor . . . . .	19
5.3	Zistenie atribútov SVG . . . . .	19
5.3.1	Prázdna nádrž . . . . .	19
5.3.2	Plná nádrž . . . . .	19
<b>6</b>	<b>Integrácia grafického komponentu pre dynamické ovládanie SVG objektu</b>	<b>23</b>
6.1	HTML súbor . . . . .	23



6.1.1	Kód . . . . .	24
6.1.2	Vysvetlenie kódu . . . . .	24
6.2	PumpingStation.js . . . . .	24
6.2.1	onPageLoad() . . . . .	25
6.2.2	PumpingStation(par1, par2) . . . . .	25
6.2.3	Tank . . . . .	26
6.2.4	Ventil . . . . .	27
<b>7</b>	<b>Návrh REST API</b>	<b>28</b>
7.1	JSON . . . . .	28
<b>8</b>	<b>Automatické mapovanie API</b>	<b>29</b>
<b>9</b>	<b>TODO — Analýza výkonnosti a obmedzení SVG</b>	<b>30</b>

# Zoznam obrázkov

1	Súradnicový systém plátna s bodom (300, 200) . . . . .	4
2	Vykreslený kruh vytvorený cez SVG, a Snap API . . . . .	10
1.1	HTML 5 API . . . . .	4
1.2	Podpora SVG vo webových prehliadačoch . . . . .	7
5.1	Grafické prostredie programu Inkscape s nakreslenou prečerpávacou sta- nicou . . . . .	18
5.2	Zobrazenie menu pre daný objekt v Inkscape . . . . .	18
5.3	Object Properties . . . . .	20
5.4	Xml Editor v Inkscape . . . . .	20
5.5	Prázdna nádrž prečerpávacej stanice . . . . .	21
5.6	Plná nádrž . . . . .	21

# Zoznam tabuliek

1	Spôsoby vytvorenia SVG v HTML dokumente . . . . .	5
2	Zoznam tvarov, ktoré podporuje SVG a Snap API, a TODO TODO PO- ROZMYSLAT NAD NAZVOM a atributy pre definovanie tvaru . . . . .	6
3	Názvy atribútov a ich význam . . . . .	7
4	Niekoľko príkazov na tvorbu Path elementu . . . . .	8
1.1	Podpora HTML <code>&lt;svg&gt;</code> elementu v webových prehliadačoch . . . . .	7
1.2	Porovnanie Canvas a SVG . . . . .	7
3.1	Výber možných parametrov pre funkciu <code>Element.attr(...)</code> . . . . .	13
4.1	Mapovacia tabuľka . . . . .	14

# Úvod

Téma práce je vizualizácia technologických dát zo SCADA systémov na webe. Cieľ práce je nájsť postup tvorby a vizualizácie grafických komponentov. Produktom bakalárskej práce je grafický komponent na vizualizáciu technologických procesov s využitím HTML 5 štandardov. V práci je animovaná prečerpávacía stanica. Navrhnutý bol aj interface, pomocou ktorého komponenty komunikujú so serverovou časťou SCADA systému.

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 technológií 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 potrebné nájsť nový spôsob vizualizácie na webe, ktorá bude v budúcnosti použiteľná na rôznych platformách, nielen na PC.

Výsledná webová aplikácia je kompatibilná s štandardmi HTML 5. Riešenie využíva výhradne open-source knižnice s licenciami typu MIT, GNU GPL, BSD, Apache 2. Zdrojové kódy práce sú udržiavané v Git repository. <https://github.com/chovancova/project>

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. Popis postupu vizualizácie grafického komponentu
6. Implementácia vzorovej sady grafických komponent.
7. Analýza výkonnosti a výkonnostné obmedzenia.

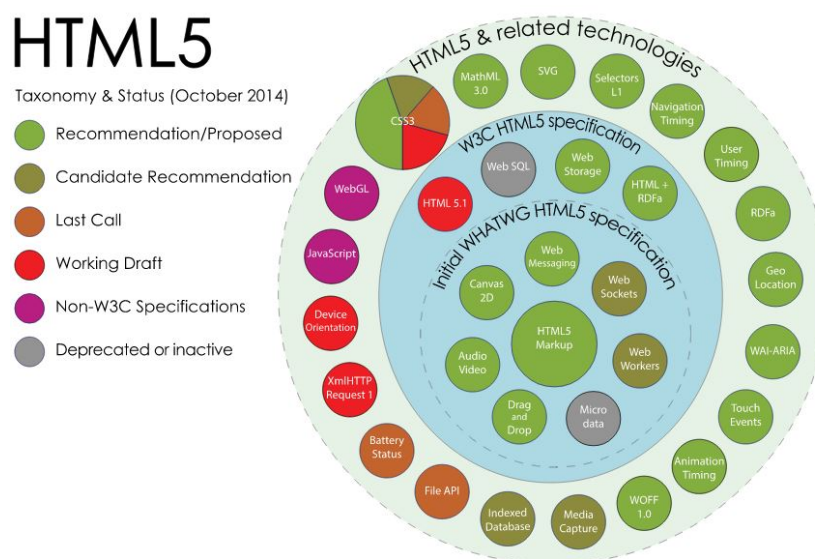
# Kapitola 1

## Základné pojmy

V kapitole sú popísané základné pojmy.

### 1.1 HTML5 štandard

World Wide Web Consortium (W3C) vydalo štandard HTML5 dňa 28. októbra 2014. HTML5 je podporovaný vo všetkých moderných webových prehliadačoch. Na obrázku 1.1 je HTML5 API a súvisiace taxonómia technológií a ich status.



Obr. 1.1: HTML 5 API

HTML5 Graphics definuje dva spôsoby vykreslenia využívajúc:

- `<canvas>` - JavaScript
- `<svg>` - SVG

## 1.2 Čo je SVG?

Scalable Vector Graphics (SVG) je štandardný formát pre vektorovú grafiku. Vektorová grafika je definovaná cez body, priamky, mnohoúhelníky, elipsy, krivky, alebo iné geometrické tvary.

SVG je jazyk na opísanie dvojrozmernej grafiky v EXtensible Markup Language (XML). Vďaka tomu, umožňuje reprezentáciu grafických informácií v kompaktnom a prenositeľnom tvare.

SVG povoľuje tieto tri typy grafických objektov: vektorové grafické tvary, obrázky a text. Grafické objekty môžu byť zoskupené, štylizované, zmenené, a kombinované do predošlých vrstiev objektov.

SVG obrázky môžu byť dynamické a interaktívne.

Prispôsobiteľnosť SVG umožňuje zmeniť veľkosť grafického komponentu bez straty kvality vzhľadu. Čo umožňuje zobraziť responzívne na viacerých možných zariadení. SVG sa bude zobrazovať rovnako na rôznych platformách. Je kompatibilná s štandardmi HTML5, ktoré navrhla W3C.

### 1.2.1 Podpora v webovom prehliadači

Súčasný prehliadač plne podporujú `<svg>` elementy. Čísla v tabuľke 1.1 špecifikujú prvé verzie webových prehliadačov, ktoré sú schopné zobraziť `<svg>` element.[12]

### 1.2.2 Rozdiely medzi SVG a Canvas

SVG patrí do vektorovej grafiky a Canvas zase do raster bitmap grafiky. SVG je jazyk na opísanie dvojrozmernej grafiky v XML. Canvas kreslí dvojrozmernú grafiku za behu programu cez JavaScript. SVG je XML založený, čo znamená, že každý element je dostupný cez SVG DOM. JavaScript umožňuje ovládanie udalostí elementov. V SVG je každý tvar zapamätaný ako objekt. V prípade zmeny `<svg>` elementu sa automaticky prekreslí.

Canvas je prekresľovaný pixel za pixelom. Bitmapová grafika je zložitejšia pre dynamické prekresľovanie, a má menšie pamäťové nároky a je rýchlejšie.

Zariadenia ako moderné smartfóny majú veľmi vysokú hustotu pixelov. Niektoré potláčajú 300 Pixels Per Inch (PPI) s tým, že sa spoliehajú na obmedzenosť ľudských očí rozoznávať jemné detaily. Pixel nemá v reálnom živote equivalent vo veľkosti až pokým je na obrazovke s fixovaným rozmerom a rozlíšením. Text s veľkosťou 16 pixelov bude veľmi malý pre oko. Pre tento dôvod zariadenia jednoducho nezobrazujú 1 CSS pixelovú jednotku na 1 pixel zariadenia. Namiesto toho zdvoja svoju veľkosť.

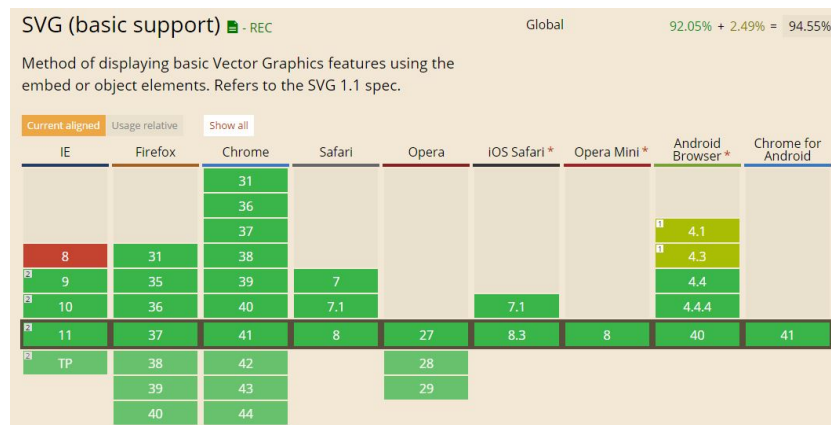
Tabuľka 1.2 zobrazuje niekoľko dôležitých odlišností medzi Canvas a SVG. TODO  
DPI

### 1.2.3 Príklad použitia SVG v HTML dokumente s inline SVG



Element	Chrome	Internet Explorer	Firefox	Safari	Opera
< svg >	4.0	9.0	3.0	3.2	10.1

Tabuľka 1.1: Podpora HTML < svg > elementu v webových prehliadačoch



Obr. 1.2: Podpora SVG vo webových prehliadačoch

Canvas	SVG
Závislé na rozlíšení a DPI	Nezávislé na rozlíšení a DPI
Nepodporuje dynamické zmeny	Podporuje dynamické zmeny
Obmedzené možnosti na vykresľovanie	Vhodné pre aplikácie s veľkými plochami na vykresľovanie
	Väčší výpočtový výkon pri komplexnom obrázku
Vhodné pre grafické-intenzívne hry	Nevhodné pre dynamické hry

Tabuľka 1.2: Porovnanie Canvas a SVG

# Kapitola 2

## Analýza požiadaviek

Kapitola popisuje výber z dostupných nástrojov a knižníc.

### 2.1 Nástroje na tvorbu grafických komponentov

WYSIWYG editory, ktoré umožňujú tvorbu grafických komponentov sú:

- Adobe Illustrator,
- CorelDraw,
- Inkscape,
- Sketch,
- <http://www.drawsvg.org/> .

Nástroj, ktorý najviac vyhovuje požiadavkam je Inkscape. Inkscape is Free and Open Source Software licensed under the GPL. Adobe Illustrator, CorelDraw, Sketch boli vylúčené pretože nie sú open-source.

### 2.2 JavaScriptové knižnice pre grafické komponenty

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

- D3.js,
- Raphael.js,
- Snap.svg.js,
- Svg.js.

Popis jednotlivých JavaScriptových knižnic.

### 2.2.1 D3.js

D3.js je JavaScriptová knižnica určená na manipuláciu dokumentov založených na dátach. Pomocou HTML, SVG a CSS umožňuje vizualizáciu dát. Je vhodná na vytváranie interaktívnych SVG grafov s hladkými prechodmi a interakciami.

D3 rieši efektívnu manipuláciu dokumentov zakladajúcich si na dátach. Využíva webové štandardy ako HTML, SVG a CSS3. [16]

### 2.2.2 Raphaël.js

Raphaël je malá JavaScriptová knižnica, ktorá umožňuje jednoducho pracovať s vektorovou grafikou na webe. Umožňuje pomocou jednoduchých príkazov vytvárať špecifické grafy, obrázky.

Raphaël využíva SVG W3C odporúčania a VML na tvorbu grafických komponentov. Z toho vyplýva, to že každý vytvorený grafický objekt je zároveň aj DOM objekt. To umožňuje cez JavaScriptové pridávať manipuláciu udalostí, alebo upravovať ich neskôr. Momentálne podporuje Firefox 3.0+, Safari 3.0+, Chrome 5.0+, Opera 9.5+ and Internet Explorer 6.0+.[15] Autor knižnice je Dmitry Baranovskiy. Raphael API má široké spektrum používateľov. Knižnica neumožňuje load SVG do dokumentu zo súboru.

### 2.2.3 Snap.svg.js

Snap.svg.js je JavaScriptová knižnica na prácu s SVG. Poskytuje pre webových developerov API, ktoré umožňuje animáciu a manipulovanie s buď existujúcim SVG, alebo

programátorsky vytvorene cez Snap API.

Tvorca Snap knižnice je rovnaký ako pri Raphael knižnici. Bola navrhnutá špeciálne pre moderné prehliadače (IE9 a vyššie, Safari, Chrome, Firefox, and Opera). Z toho vyplýva, že umožňuje podporu maskovania, strihania, vzorov, plných gradientov, skupín.

Snap API je schopné pracovať s existujúcim SVG súborom. To znamená, že SVG obsah sa nemusí generovať cez Snap API, aby sa mohol oddelene používať. Obrázok vytvorený v nástroji Inkscape sa dá animovať, alebo inak manipulovať cez Snap API. Súbory načítané cez Ajax sa dajú vykresliť, bez toho, aby boli renderované.

Snap podporuje animácie. Poskytuje jednoduché a intuitívne JavaScript API pre animáciu. Snap umožňuje urobiť SVG obsah viac interaktívnejší a záživnejší. [14]

#### 2.2.4 SVG.JS

SVG.JS je ďalšia knižnica umožňujúca manipulovať a animovať SVG.

Medzi hlavné výhody knižnice patrí to, že je má ľahko čitateľnú syntax. Umožňuje animovanie veľkosti, pozície, transformácie, farby. Má modulárnu štruktúru, čo umožňuje používanie rôznych rozšírení. Existuje množstvo užitočných pluginov dostupných na internete. [17]

TODO NAPISAT NIECO PRECO SOM HO VYLUCILA

### 2.3 Zhodnotenie požiadaviek

Grafické komponenty sa budú vytvárať v programe Inkscape. Ovládanie a animovanie prostredníctvom knižnice Snap.svg.js. TODO PRESTYLIZOVAT.. Hlavný dôvod, prečo som sa rozhodla pre túto knižnicu bol, že dokáže načítavať SVG súbor a potom s ním manipulovať.

Spĺňa požiadavku kompatibility pre moderné webové prehliadače. Je to open-source knižnica a má licenciu Apache 2.

# Kapitola 3

## Knižnica Snap.svg.js

### 3.1 append()

### 3.2 Element.attr(...)

Vráti alebo nastaví dané atribúty elementu.

#### 3.2.1 Parametre:

- objekt - obsahuje pár kľúč-hodnota atribútov, ktoré chcem nastaviť.
- string - názov atribútu

Niekoľko možných dvojíc parametrov sú v tabuľke 3.1. Vráti buď súčasný element alebo stringovú hodnotu atribútu.

#### 3.2.2 Použitie:

```
el.attr({  
  fill: "#fc0",  
  stroke: "#000",  
  strokeWidth: 2,
```

```
});
```

### 3.3 Element.animate()

Snap.animation = function (attr, ms, easing, callback)

- attr (object) attributes of final destination - duration (number) duration of the animation,

in milliseconds - easing (function) #optional one of easing

functions of @mina or custom one - callback (function) #optional callback

function that fires when animation ends

## 3.4 Gradients, Path String, Colour Parsing

### 3.4.1 Paper.path([pathString])

Vytvorí <path> element podľa daného reťazca. Parameter pozostáva z jedno písmenkových príkazov, nasledovaných bodkami a oddelovaný argumentami a číslami. Napríklad: "M10,20L30,40" obsahuje príkazy: M s argumentami (10, 20) a L (30, 40). Rozdiel vo veľkosti písma vyjadruje to, či ide o absolútnu, alebo relatívnu cestu. Ak sú malé znaky jedná sa o relatívne, v prípade veľkých znakov absolútna cesta.

Parameter	Príklad použitia	Poznámka
cx	cx: 50	x-os súradnica centra kruhu, alebo elipsy
cy	cy: 90	y-os súradnica centra kruhu, alebo elipsy
r	r: 40	polomer kruhu, elipsy alebo okruhlých rohov na obdĺžniku
rx	r: 50	horizontálny polomer elipsy
ry	r: 40	vertikálny polomer elipsy
x, y	x: 50, y: 100	súradnica x-osi, y-osi
width, height	width: 500, height: 10	šírka, výška
"fill-opacity"	"fill-opacity": 0.5	neprehľadnosť, 0-1
fill	fill: "blue"	vyplnenie farbou, gradientom, obrázkom
stroke	stroke: "blue"	farba výplne okraja
strokeWidth	strokeWidth: 2	šírka okraja v px, default je 1
strokeLinecap	strokeLinecap: "butt"	["butt", "square", "round"]
strokeLinejoin	strokeLinejoin: "round"	["bevel", "round", "miter"]
viewBox		napr. viewBox: [0, 0, 800, 600]
strokeDasharray	strokeDasharray: "5 3"	pole čiarok, bodiek, pomlčiek
font	font: "20px Source Sans Pro, sans-serif"	zmena písma, rodiny písma, veľkosti v pixeloch, a weight
transform	transform: "t" + [0, 5] + "r" + 20	t - zmena súradníc, r - otočenie
path	path: "M10,10 210,10"	SVG cesta
text	text: "snap"	zmení text elementu

Tabuľka 3.1: Výber možných parametrov pre funkciu `Element.attr(...)`

# Kapitola 4

## Postup vytvorenia komponentov

UML diagram...

Spôsob vytvorenia grafických komponentov je nasledovný. Najprv používateľ vytvorí SVG súbor, následne ho načíta, a vytvorí funkcie v JavaScripte na ovládanie atribútov SVG elementu. Alternatívna možnosť je vytvoriť SVG elementy prostredníctvom JavaScriptovej knižnice a nenačítavať súbor.

akcia	SVG akcia	JavaScript akcia	Popis
Animácia			
Nastavenie farby			
Transformácia			
Skryvanie		attr(visibility: true)	

Tabuľka 4.1: Mapovacia tabuľka

### 4.1 Použitie SVG v HTML dokumente

SVG sa dá použiť a vytvoriť viacerými spôsobmi:

- priamo v HTML dokumente - inline,



- načítanie z oddeleného SVG súboru,
- načítanie pomocou JavaScriptovej knižnice.

#### 4.1.1 Vytvorenie SVG

Cez program WYSWING

načítanie JavaScriptovú knižnicu

Postup (načítanie súboru v tele JavaScriptovej metóde):

1. Načítať knižnicu Snap.svg.js do HTML súboru.
2. Pridať atribút `onPageLoad()`; do definície body.
3. Pridať HTML tag `<svg>` do tela HTML a nastaviť v ňom požadovanú veľkosť cez `viewBox`.
4. Vytvoriť JavaScriptový súbor, alebo tag `<script>`, ktorý bude obsahovať funkcie.
5. Vytvorenie funkcie na načítanie Snap API, a .SVG súboru.
6. V tele funkcie inicializácia Snap Canvasu. To znamená, kde konkrétne v HTML stránke sa zobrazí.
  - `s = Snap()` - najbližšie voľné miesto
  - `s = Snap(šírka, výška)`
  - `s = Snap(HTMLtag)` - id tagu `<svg>`, ktoré sa pridalo v bode č. 3
7. Načítanie .SVG súboru cez funkciu `Snap.load()`, s parametrami: názov súboru a funkcie s parametrom `f`.
8. Zobrazenie súboru cez príkaz `s.append(f);`, ekvivalenté zápisy: `s.appendAll(f);`, `s.add(f);`.
9. V HTML stránke sa zobrazí daný .SVG súbor.

Postup ovládania SVG elementu:

1. Vytvorenie novej funkcie.
  - anonymná funkcia
  - pomenovaná funkcia
  - objekt, v ktorom bude zadaná funkcia.
2. Nová premenná var, ktorá obsahuje id SVG elementu, ktorý sa ide ovládať. (Na zistenie id SVG vid Postup krokov na zistenie id.)
3. Vytvorenie funkcie cez ktorú sa bude pristupovať k API Snap knižnice.
4. `s.select(id SVG)`
5. V tejto chvíli je možné volať funkcie z Snap API príkazom: `funkcia().funkciaAPISnap..`
  - `.animate()` - animácia
  - `.attr()` - nastavenie atribútu
  - `.add()`
  - TODO

TODO - odkazat tuto na vsetky mozne atributy, ktore sa daju zmeniť.

## 4.2 JavaScript

## Kapitola 5

# Príklad vytvorenia grafického komponentu v Inkscape

### 5.1 Vytvorenie SVG v programe Inkscape

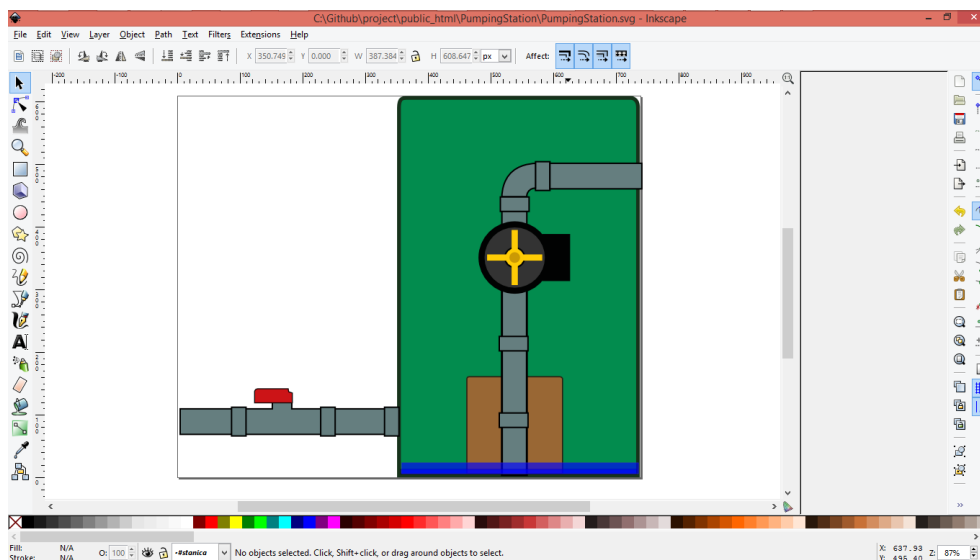
Nakreslenie jednotlivých častí komponentov prečerpávacej stanice bolo realizované pomocou ľavého bočného panela. Prečerpávacia stanica sa skladá z potrubí, indikátora úrovne hladiny vody, motora, a symbolu rotora čerpadla. Ako je možné vidieť na obrázku 5.1.

### 5.2 Definovanie id SVG

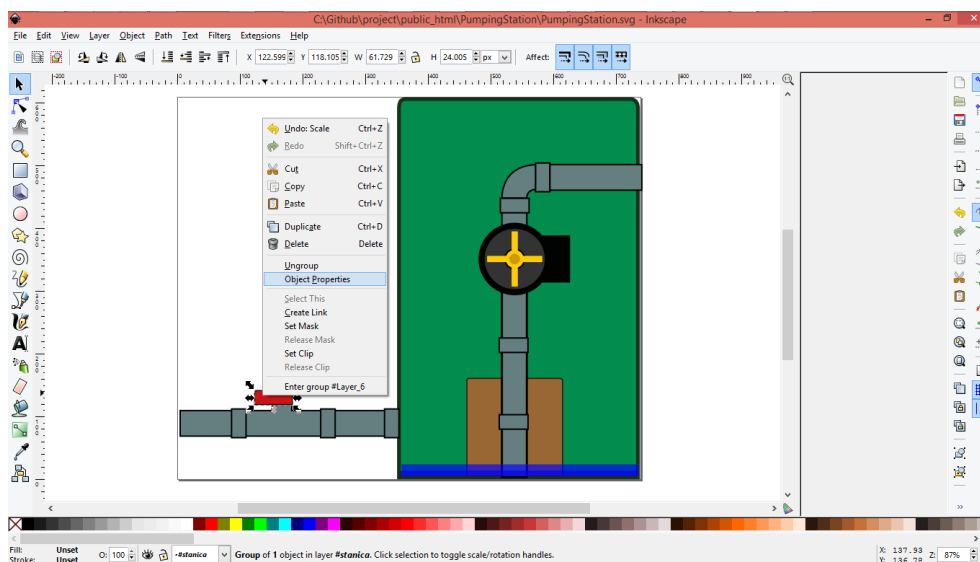
Pre ovládanie JavaScriptom je nutné si pozrieť jednotlivé jedinečné identifikačné názvy. V SVG sú označované ako id. Zistenie id je pomerne jednoduché. Klikneme pravým tlačidlom na daný komponent, ktorého id chceme vedieť, a potom na Objekt Properties. Ako je možné vidieť na obrázku 5.2.

Po kliknutí sa nám zobrazí okno s názvom Object Properties.

Z obrázka č.5.3 možno vyčítať aké je ID, predvolené sú tam napr. desc3072. Hodnoty je možné prepísať a zmeniť stlačením tlačidla Set. Pre nás je dôležitá hodnota v kolónke Label - #ventil. Na ovládanie časti svg elementu cez CSS selektor je potrebné



Obr. 5.1: Grafické prostredie programu Inkscape s nakreslenou prečerpávacou stanicou



Obr. 5.2: Zobrazenie menu pre daný objekt v Inkscape

si zapamätať hodnotu Labelu.

Alebo ďalší spôsob zistenia id SVG časti tvarov je priamo nájsť tú hodnotu v PumpingStation.svg. Je to označené ako id="ventil".

V okne Object Properties je možné nastaviť script na animovanie. Po kliknutí na Interactivity sa zobrazia ďalšie kolónky, kde je možné zadať akciu, ktorá má nastať.

### 5.2.1 XML Editor

Ďalší spôsob získania informácií o svg cez Inkscape je cez zabudovaný XML Editor. Stlačením klávesovej skratky SHIFT + CTRL + X, alebo v hornej lište v menu vybrať ponuku Edit a na spodu je XML Editor. Následne sa zobrazí okno, ktoré je na obrázku 5.4.

## 5.3 Zistenie atribútov SVG

### 5.3.1 Prázdna nádrž

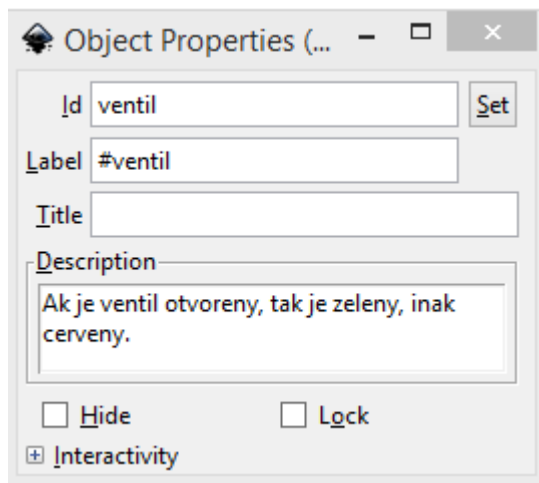
Prázdna nádrž je zobrazená na obrázku 5.5 Vyjadruje to, že do nádrže nevchádza tekutina. Hladina má výšku nastavenú na 9,64px. Šírka a súradnica x je rovnaká ako pri plnej nádrži. Súradnice osi y, x sú 1916,36 a 2507,84. Pri plnej nádrži sa zmení výška a os y. Pre animáciu zdvihnutia hladiny nádrže bude potrebné si zapamätať tieto súradnice.

### 5.3.2 Plná nádrž

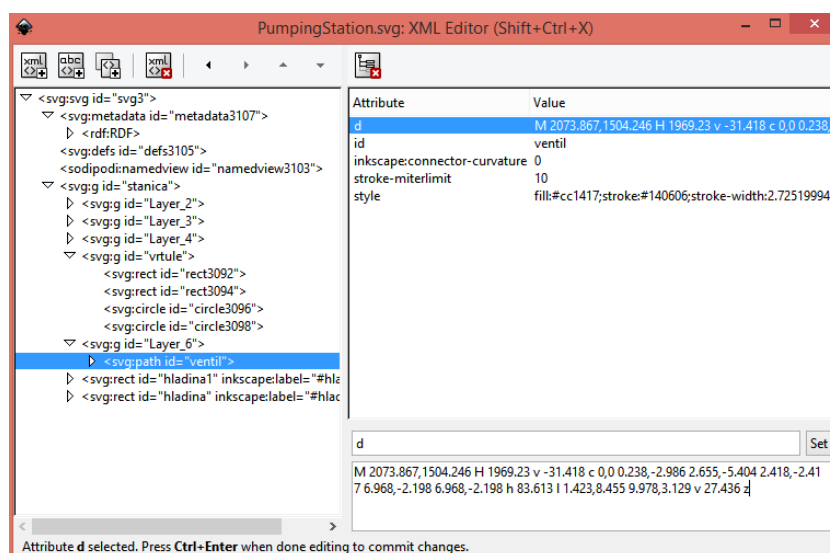
Na obrázku č. 5.6 je vykreslená plná nádrž. Hladina má súradnice x, a šírku rovnakú ako v prázdnej nádrži. Zmenila sa os y, a výška - na 1320,16 a na 605,92.

V SVG súbore je to zapísané nasledovným kódom.

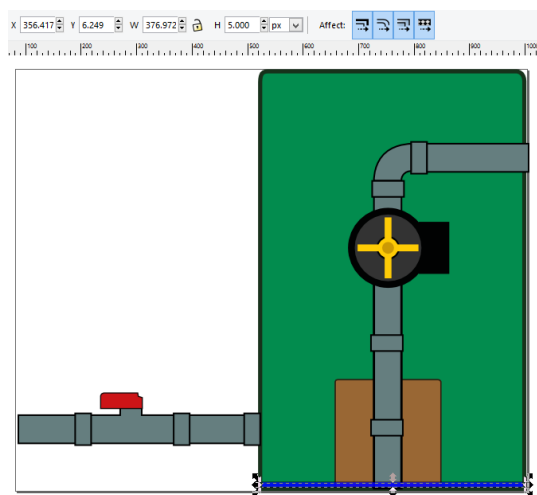
```
<rect
  inkscape:label="#hladina"
  y="1320.1689"
```



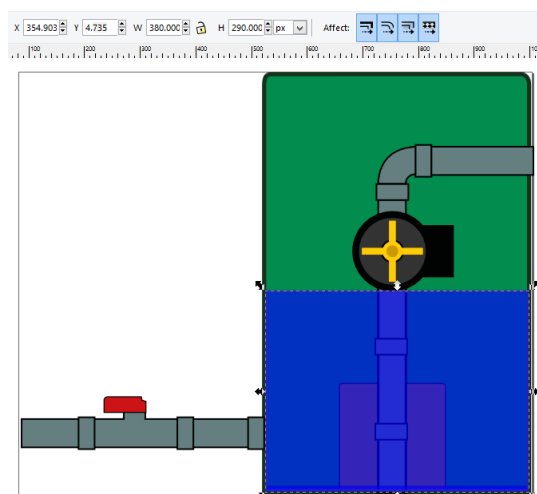
Obr. 5.3: Object Properties



Obr. 5.4: Xml Editor v Inkscape



Obr. 5.5: Prázdná nádrž přečerpávací stanice



Obr. 5.6: Plná nádrž

```
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">  
</rect>
```



## Kapitola 6

# Integrácia grafického komponentu pre dynamické ovládanie SVG objektu

Súborová štruktúra príkladu:

- index.html
- PumpingStation.js
- PumpingStation.svg
- TestPumpingStation.js
- exec.js

### 6.1 HTML súbor

Do HTML súboru index.html pridáme párový tag `<svg>`. Na toto miesto sa neskôr vykreslí SVG načítané zo súboru cez JavaScript. Môže sa tu uviesť i celý kód SVG obrázka. V prípade, že nebude v dokumente dané kde presne sa nachádza SVG tag tak sa pridá na najbližšie voľné miesto.

### 6.1.1 Kód

```
<svg
  id="svgStanica"
  viewBox="0_0_750_600"
  width="40%"
  height="40%">
</svg>
```

### 6.1.2 Vysvetlenie kódu

- **id** - jedinečný identifikátor, cez ktorý meníme vlastnosti.
- **viewBox** - je virtuálne okno, ktorým sa užívateľ uvidí svg obrázok. Je atribút, ktorý povoľuje špecifikovať danú množinu grafických komponentov, aby sa zobrazili v daných súradniciach x, y a šírke, výške. Hodnoty atribútov v viewBox sú štyri čísla - min-x, min-y, width a height.
- **width** a **height** je šírka a výška. Hodnoty atribútov je možné uviesť relatívne v percentách, alebo absolútne v pixloch.

Musíme sa uistiť, aby sa načítali všetky JavaScriptové knižnice, pred spustením funkcií. To zabezpečíme pridaním onload do tagu <body>.

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

A ešte jedna vec pri HTML súbore - TODO.

```
<script type="text/javascript" src="../js/snap.svg-min.js">
</script>
<script type="text/javascript" src="PumpingStation.js">
</script>
```

## 6.2 PumpingStation.js

TODO V súbore PumpingStation.js sú funkcie na animovanie.. TODO

### 6.2.1 onPageLoad()

onPageLoad() sa spustí pri načítaní tela HTML súboru index.html. Funkcia spustí funkciu PumpingStation(). Prvý parameter je udaný konkrétny svg súbor, ktorý chcem načítať. Druhý parameter je id tagu svg, ktorý je v html. TODO

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

### 6.2.2 PumpingStation(par1, par2)

Funkcia inicializuje daný svg súbor, a vykreslí ho. Parametre pre PumpingStation sú názov svg súboru, a id, ktoré sa nachádza v tagu <svg> html súbore.

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

**paper** - globálna premenná. TODO REFERENCIA NA PLOCHU ... Vytvorí plochu na kreslenie, alebo wraps existujúci SVG element. Ako parametre môžu byť buď šírka, výška, alebo DOM element. Napríklad Snap(600, 800), alebo Snap("#svgStanica"), resp Snap().

**load** - TODO funkciu z knižnice Snap. Cez ňu TODO načítam svg súbor. Ako parametre funkcie je názov súboru svg, prípadne môže byť prázdny, ak TODO nenačítavam súbor, ale beriem ho priamo z html súboru. Druhý parameter je funkcia, v ktorej TODO volám funkciu na zobrazenie obsahu svg súboru do daného tagu svg s id, ktorý bol daný pri funkcii paper. Na plochu TODO ho zobrazím pomocou príkazu append. TODO

### 6.2.3 Tank

Zanimovanie stúpania a klesania hladiny nádrže.

```
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);
  }
};
```

TODO PRESTYLIZOVAT VYHODIT RODY MOJE TODO TODO Vytvorila som objekt Tank medzi jeho atribúty 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 daný objekt, ktorý chcem ovládať. Pomocou Tank.tank() môžem volať funkcie z Snap knižnice.

Zanimovanie tanku je realizované v funkcii animateComponentTank - kde parametrom je v percentách udané o koľko sa ma zdvihnúť hladina nadrž. Využívam funkciu animate. Kde v prvom parametri - mením 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 vypočítaná ako 1912 čo je y prázdnej nádrže a je od nej odpočítaná

hodnota výšky. Další parameter pri funkcii `animate()` je rýchlosť animácie vyjadrená v milisekundách.

### 6.2.4 Ventil

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

TODO TRANSFORM .. TODO

# Kapitola 7

## Návrh REST API

### 7.1 JSON

TODO PREROBIT CELE - TODO -

API k Pumping station schéme.

```
var updateData = {  
  "valve": "true",  
  "tank": "20",  
  "engine": "20"  
};
```

Tento kód definuje objekt s názvom updateData, ktorá má tri vlastnosti.

Interface funkcia k REST API.

```
function updateSchema(updateData){  
  updateSchema01(updateData.valve, updateData.tank, updateData  
    .engine);  
}
```

## Kapitola 8

# Automatické mapovanie API

Analyzujte možnosti automatického mapovania API grafických prvkov pomocou meta-dát na existujúce API dostupné pre SCADA server D2000.

## Kapitola 9

# TODO — Analýza výkonnosti a obmedzení SVG

<http://caniuse.com/#feat=svg-html>

\*\*\*\*\*

Animácia pozície - `transform: translate(npx, npx);`

Animácia škály - `transform: scale(n);`

Animácia otáčania - `transform: rotate(ndeg);`

Animácia neprehľadnosti - `opacity: 0..1;`

\*\*\*\*\*

**Transformation** *scale*(sx, sy) - zmením veľkosť tvaru na dané súradnice, *translate*(tx, ty) - presuním na iné miesto - zmením súradnice

Podpora svg v prehliadačoch <http://caniuse.com/#feat=svg>

<http://www.schepers.cc/svg/blendups/embedding.html>



# Záver

V mojej práci som sa snažila nájsť najjednoduchšie riešenie pre vizualizáciu komponentov. Vykresľovanie podľa súradníc, a priamo kreslenie cez JavaScript sa mi zdalo nepraktické z dôvodu, že priamo nevidím to čo kreslím. Preto som hľadala také riešenie, ktoré by mi umožňovalo ovládať už vytvorený obrázok ovládať cez JavaScript.

Cez Inkscape sa nakreslí komponent, a cez JavaScript pomocou knižnice Snap.svg.js sa manipuluje. Podarilo sa mi aj to, aby bol výsledný prvok responzívny aj na iných platformách ako napríklad tablety. SVG podporujú všetky moderné webové prehliadače.

Moje riešenie používa knižnicu a softvér, ktorý je open-source. Všetky zdrojové kódy práce sú v Git repository na GitHubu.

# Literatúra

- [1] Mavrody S., *Sergey's HTML5 & CSS3 Quick Reference: HTML5, CSS3 and APIs*, 3rd edition, Belisso, 2012, ISBN 978-0-98338-674-2
- [2] Dawber D., *Learning Raphael JS Vector Graphics*, Packt Publishing, 2013, ISBN 978-1-78216-916-1
- [3] Wilson CH., *RaphaelJs Graphic and visualization on the web* , O'Reilly Media, 2013, ISBN 978-1-449-36536-3
- [4] Haverbeke M., *Eloquent Javascript* 2. vyd. No Starch Press, 2014, ISBN 978-1-59327-584-6
- [5] Zakas N. Z., *JavaScript pro webové vývojáře Programujeme profesionálně*, 1. vyd. Brno:Computer Press, a.s., 2009, ISBN 978-80-251-2509-0
- [6] Suehring S., *JavaScript krok za krokem*, 1. vyd. Brno:Computer Press, 2008, ISBN 978-80-251-2241-9
- [7] Zakas N. C., McPeak J., Fawcett J., *Profesionálně Ajax*, Zoner Press, 2007, ISBN 978-80-86815-77-0
- [8] 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](http://commons.oreilly.com/wiki/index.php/SVG_Essentials)
- [9] Richardson L., Amundsen M., *RESTful Web APIs* 1. vyd. O'Reilly Media, 2013, ISBN 978-1-449-35806-8

- [10] Allamaraju S., *RESTful Web Services Cookbook* 1. vyd. O'Reilly Media, 2010, ISBN 978-0-596-80168-7
- [11] <http://www.w3.org/TR/html/>
- [12] [http://www.w3schools.com/html/html5\\_svg.asp](http://www.w3schools.com/html/html5_svg.asp)
- [13] [www.w3schools.com/svg/svg\\_inhtml.asp](http://www.w3schools.com/svg/svg_inhtml.asp)
- [14] The JavaScript SVG library for the modern web, <http://snapsvg.io/>.
- [15] [http://raphaeljs.com/](http://dmitrybaranovskiy.github.io/raphael/)
- [16] <http://d3js.org/>
- [17] <http://www.svgjs.com/>
- [18] 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

**JPEG** Join Photographic Experts Group

**GIF** Graphics Interchange Format

**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

**VML** Vector Markup Language

**WYSIWYG** What You See Is What You Get

**DPI** Dots Per Inch

**PPI** Pixels Per Inch

**SMIL** Synchronized Multimedia Integration Language

## Zoznam termínov