

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**

Tútor **Ing. Patrik Hrkút, PhD.**

Reg.č. 5/2014

Máj 2015

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

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 študijnom odbore Telekomunikačné siete, na Elektrotechnickej fakulte na Žilinskej univerzite v Žiline.

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 OLEGA: *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 University of Zilina, Faculty of Telecommunication,
Fixed networks. Solution Design Architect
TODO

The main idea of this ... TODO

Obsah

Úvod	2
1 Základné pojmy	4
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
2 Analýza požiadaviek	8
2.1 Nástroje na tvorbu grafických komponentov	8
2.2 JavaScriptové knižnice pre grafické komponenty	9
2.2.1 D3.js	9
2.2.2 Raphaël.js	9
2.2.3 Snap.svg.js	10
2.2.4 SVG.JS	10
2.3 Zhodnotenie požiadaviek	11
3 Learning Raphael JS Vector Graphics	12
3.1 Porovnanie spôsobu vykreslenia cez SVG SMIL a Snap	12
3.2 Krok 1: Inicializácia plátna na kreslenie	14
3.2.1 Súradnice plátna	14

3.2.2	DOM element	15
3.3	Kreslenie základných tvarov cez Snap API	16
3.3.1	Popis atribútov tvarov	16
3.4	Vykreslenie obrázku	18
3.5	Atribúty elementu	18
3.5.1	Výplň elementu - fill	19
3.5.2	Nastavenie okraja elementu - stroke	19
3.6	Zoskupovanie elementov	19
3.6.1	Maskovanie - Element.mask()	20
3.7	Práca s textom - Paper.text(x, y, text)	21
3.8	Transformácie - Element.transform(...)	22
3.8.1	Suradnicovy system	23
3.8.2	Jednoduché transformácie	23
3.8.3	Event handler	26
3.9	Animácie	26
3.10	Animacia - Element.animate(...)	28
3.10.1	Animácia atribútov	28
3.10.2	Animácia elementov, path	28
3.10.3	Animation easing	29
3.10.4	Animácia transformácií	31
3.10.5	Animacia vyuzivajuca vlastne atributy	31
3.10.6	Animacia podľa path	31
3.10.7	Pausing a resuming animation	31
4	Práca s Existujúcimi SVG	32
4.1	Inkscape	32
4.2	Definovanie id SVG	32
4.3	Zoznam id SVG atributov vyuzite v priklade	32

5	Postup vytvorenia komponentov	33
5.1	Použitie SVG v HTML dokumente	33
5.1.1	Vytvorenie SVG	34
5.2	JavaScript	35
6	Príklad vytvorenia grafického komponentu v Inkscape	36
6.1	Vytvorenie SVG v programe Inkscape	36
6.2	Definovanie id SVG	37
6.2.1	XML Editor	38
6.3	Zistenie atribútov SVG	39
6.3.1	Prázdna nádrž	39
6.3.2	Plná nádrž	39
7	Integrácia grafického komponentu pre dynamické ovládanie SVG objektu	41
7.1	HTML súbor	41
7.1.1	Kód	42
7.1.2	Vysvetlenie kódu	42
7.2	PumpingStation.js	42
7.2.1	onPageLoad()	43
7.2.2	PumpingStation(par1, par2)	43
7.2.3	Tank	44
7.2.4	Ventil	45
7.3	REST API	46
7.4	REST API pre graficke komponenty	46
7.5	Data binding pre system D2000	46
8	TODO — Analýza výkonnosti a obmedzení SVG	48
.1	Poznámky - informačný účel	58
.2	Kapitoly by mali byť nasledovne	59

Úvod

Téma práce je vizualizácia dát získaných pomocou SCADA systémov. 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 prečerpávacia 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 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 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.

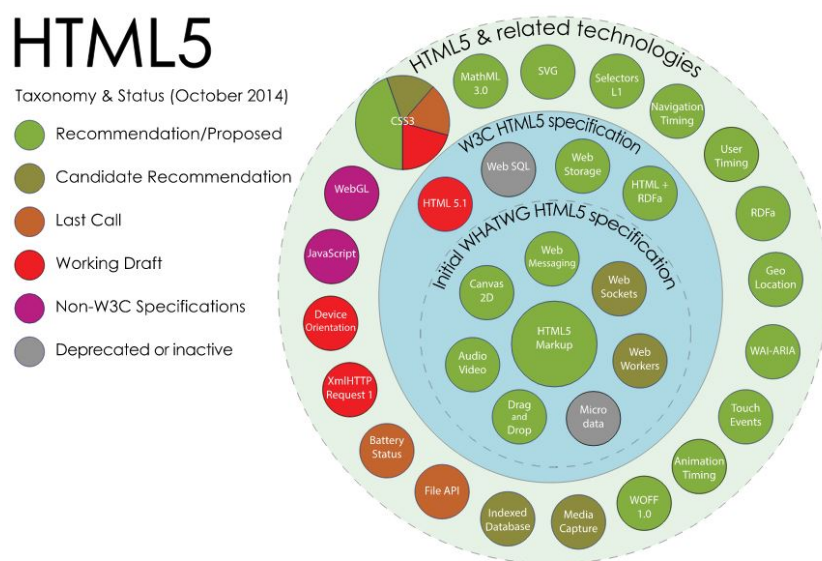
Kapitola 1

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, mnohouholníky, 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.[13]

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.

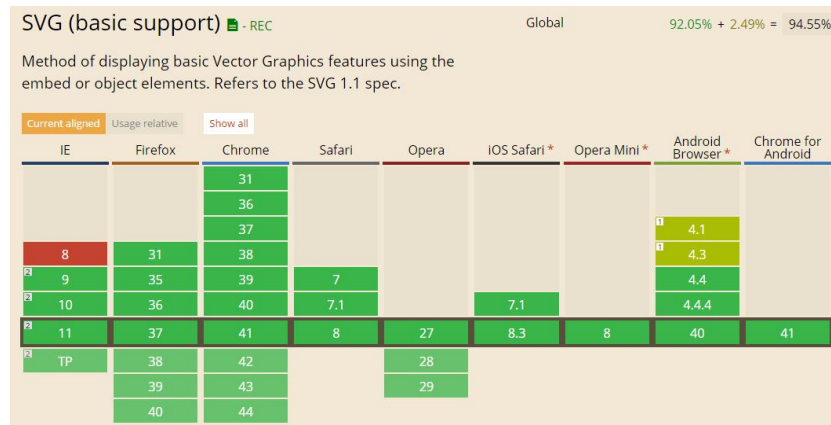
Zariadenia ako moderné smartfóny majú veľmi vysokú hustotu pixelov. Niektoré potláčajú 300 Pixels Per Inch (PPI) s tým, že sa spoliehajú na obmedzenosť ľudských očí rozoznávať jemné detaily. Pixel nemá v reálnom živote equivalent vo veľkosti až pokým je na obrazovke s fixovaným rozmerom a rozlíšením. 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
< <i>svg</i> >	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/> .

Volne dostupné online SVG editory:

- ScriptDraw - online SVG editor vyvinutý pomocou modernej technológie Adobe Flex.
- svg-edit - Rýchly, webový SVG editor založený na JavaScriptovej technológii, ktorá funguje v akékoľvek modernom webovom prehliadači.

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 knižníc.

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. [17]

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+.[16] 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 vytvorené 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. Súbor 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. [15]

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. [18]

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

Kapitola 3

Learning Raphael JS Vector Graphics

3.1 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 obdĺžnika 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.

3.2 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.

3.2.1 Súradnice plátna

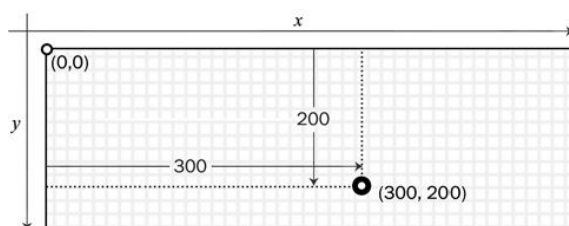
TODO Na to, aby bolo plátno responzívne vo webovom prehliadači musí byť nastavené tieto dve veci:

- definovaný viewBox
- výška a šírka plátna musí byť v relatívnych rozmeroch, najlepšie nastavená na 100%

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

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

Na obrázku 3.1 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. 3.1: Súradnicový systém plátna s bodom $(300, 200)$

3.2.2 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 vloženým samostatného .SVG súboru. V tabuľke 3.1 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 3.1: Spôsoby vytvorenia SVG v HTML dokumente

3.3 Kreslenie základných tvarov cez Snap API

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 3.4

Tabuľka 3.2: Zoznam tvarov, ktoré podporuje SVG a Snap API, a TODO TODO PO-ROZMYSLAT 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" ... />
```

3.3.1 Popis atribútov tvarov

Názvy atribútov a ich význam pre obdĺžnik, kruh, elipsu sú vyjadrené v tabuľke 3.3

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 3.3: Názvy atribútov a ich význam

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

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 3.4.

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 3.4: Niekoľko príkazov na tvorbu Path elementu

3.4 Vykreslenie obrázku

Snap povoľuje vloženie bitmapových obrázkov (.jpg alebo .png) na plátno. Používa metódu `image` z `Paper` objektu. Parametre metódy `image` sú: zdroj, x, y, šírka, výška. Príklad kódu, ktorý vkladá .jpg obrázok do plátna:

```
var paper = Snap("mojePlatno", 300, 200);
paper.image("obrazok.jpg", 15, 15, 100, 150);
```

3.5 Atribúty elementu

Tvary, ktoré sa dajú nakresliť sa môžu vyfarbiť, orámoviť alebo mnoho iných atribútov sa dá nastaviť. Keď sa vytvorí tvar, tak sa vráti `Element` objekt. Tento objekt má `attr` metódu, ktorá akceptuje key-value pár atribútov. V tomto odseku sa pozrieme na rôzne atribúty, ktoré môžu byť aplikované na naše grafické komponenty používajúc túto metódu.

3.5.1 Výplň elementu - fill

Pozadie pre element nastavím cez metódu `attr` použitím `fill` atribútu ako parameter. Pre jednofarebné výplne je formát vyjadrený cez CSS špecifikáciu. CSS špecifikácia farieb je nasledovná: `#rrggbb` alebo skrátené `#rgb`, `rgb(r, g, b)` reťazec alebo slovne. Napríklad:

```
var kruh = paper.circle(50, 50, 40).attr("fill", "red");
```

Ďalšie spôsoby výplne elementu sú obrázkom, gradientom, alebo vzorom. Pre nastavenie neprehľadnosti nastavíme atribút `"fill-opacity"` hodnotou čísla v rozsahu od 0-1. Element pri `"fill-opacity": 1` je neprehľadný.

3.5.2 Nastavenie okraja elementu - stroke

Elementy môžu mať niekoľko rôznych druhov okrajových atribútov. Prehľad tých najznámejších je v tabuľke 3.5.[12]

Atribút pre <code>attr()</code>	CSS atribút	Poznámka
<code>stroke</code>	<code>stroke</code>	farba výplne okraja
<code>strokeWidth</code>	<code>stroke-width</code>	šírka okraja v px, default je 1
<code>strokeOpacity</code>	<code>stroke-opacity</code>	neprehľadnosť, 0-1
<code>strokeLinecap</code>	<code>stroke-linecap</code>	["butt", "square", "round"], tvar - okraj konca
<code>strokeLinejoin</code>	<code>stroke-linejoin</code>	["bevel", "round", "miter"], tvar - okraj rohu
<code>strokeDasharray</code>	<code>stroke-dasharray</code>	pole čiarok, bodiek.., napr.5,3,2

Tabuľka 3.5: Výber možných stroke atribútov

3.6 Zoskupovanie elementov

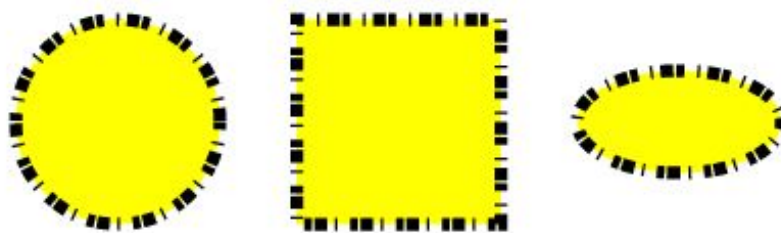
Niekedy je potrebné použiť rovnaké atribúty, transformácie, alebo animácie pre viacero elementov. V Snap API je možné využiť metódu `group` alebo `g`. Group zoskupí viacero elementov do množiny. Príkazom `add` sa dajú pridať ďalšie prvky. V množine sa dajú meniť atribúty pre viacero prvkov naraz volaním metódy `attr`.

Príklad zoskupenia elementov. Výsledné zoskupenie je zobrazené na obrázku 3.2.

```
var paper = Snap();
var kruh = paper.circle(50, 50, 40);
var obdlznik = paper.rect(120, 10, 80, 80);
var elipsa = paper.ellipse(270, 50, 40, 20);

var group = paper.g(kruh, obdlznik);
group.add(elipsa);

group.attr({
  fill: 'yellow',
  stroke: '#000',
  strokeWidth: 5,
  strokeDasharray: [3, 5, 1]
});
```



Obr. 3.2: Príklad zoskupenia elementov a následná zmena atribútov

3.6.1 Maskovanie - `Element.mask()`

TODO - PRE DANY ELEMENT VOLAM FUNKCIU ATTR, KDE NASTAVIM VLASTNOST MASK NAPR. `mask: ellipse`

```
/ Now more interesting stuff // Let's assign this group as a mask for our big circle
bigCircle.attr( mask: discs );
```

3.7 Práca s textom - Paper.text(x, y, text)

Vykreslenie textu na plátne namiesto HTML markup s CSS štýlovaným umožňuje animovať a transformovať v rovnakým spôsobom ako iné tvary. Text vytvorenie cez metódu `text`. Parametre metódy `text` sú súradnice `x`, `y` a `text`, ktorý sa vykreslí. Vlastnosti textu sa dajú zmeniť volaním metódy `attr`. V tabuľke sú atribúty, ktoré sa dajú zmeniť prostredníctvom metódy `attr`.

Snap atribút	CSS atribút	Poznámka
<code>font</code>	<code>font</code>	napr. "30px Helvetica, sans-serif",
<code>textAnchor</code>	<code>text-anchor</code>	pozícia textu, napr. "middle"
<code>fill</code>	<code>fill</code>	výplň textu farbou, gradientom, vzorom
<code>fontSize</code>	<code>font-size</code>	veľkosť textu
<code>fontFamily</code>	<code>font-family</code>	napr. "monospace"
<code>fontStyle</code>	<code>font-style</code>	štýl písma, napr. kurzíva
<code>fontVariant</code>	<code>font-variant</code>	napr. "small-caps"
<code>fontWeight</code>	<code>font-weight</code>	hrúbka písma, napr. normal, bold, bolder, lighter, 100-900

Tabuľka 3.6: Atribúty na zmenu vlastností elementu text

Príklad zmeny farby:

```
var paper = Snap();
var text = paper.text(30, 100, "Namestovo");

text.attr({
  textAnchor: "middle",
  fill: "#00b",
  fontSize: '16px',
  fontFamily: "Veranda",
  fontStyle: "italic",
```

```
fontVariant: "small-caps",
fontWeight: 800,
});
```

Na obrázku 3.3 je zobrazený text, ktorý bol uvedený v príklade.

NAMESTOVO

Obr. 3.3: Príklad na zmenu atribútov v texte.

3.8 Transformácie - `Element.transform(...)`

```
Element.transform(
```

tstr

```
)
```

`r-30, 50, 10t10, 20s1.5` Adds transformation to the element which is separate to other attributes, i.e. translation doesn't change x or y of the rectangle. The format of transformation string is similar to the path string syntax:

`"t100,100r30,100,100s2,2,100,100r45s1.5"` Each letter is a command. There are four commands: t is for translate, r is for rotate, s is for scale and m is for matrix.

There are also alternative `\absolute` translation, rotation and scale: T, R and S. They will not take previous transformation into account. For example, `...T100,0` will always move element 100 px horizontally, while `...t100,0` could move it vertically if there is `r90` before. Just compare results of `r90t100,0` and `r90T100,0`.

So, the example line above could be read like `\translate by 100, 100; rotate 30 stupnov around 100, 100; scale twice around 100, 100; rotate 45 stupnov around centre; scale 1.5 times relative to centre`. As you can see rotate and scale commands have origin coordinates as optional parameters, the default is the centre point of the element. Matrix accepts six parameters.

Usage

```

var el = paper.rect(10, 20, 300, 200);
// translate 100, 100, rotate 45, translate -100, 0
el.transform("t100,100r45t-100,0");
// if you want you can append or prepend transformations
el.transform("...t50,50");
el.transform("s2...");
// or even wrap
el.transform("t50,50...t-50-50");
// to reset transformation call method with empty string
el.transform("");
// to get current value call it without parameters
console.log(el.transform());

```

Keď vytváram element v Snap, tak efektívne vytváram Document Object Model (DOM) objekt. TODO... [2, page 51]

3.8.1 Suradnicovy system

3.8.2 Jednoduché transformácie

Basic transformations and event handling We perform transformations in Raphaël using the transform method on elements, which accepts as an argument a transformation string. Transformation strings define a sequence of transformations to take place on a particular element.

You can also transform elements and sets using the transform attribute. This is often convenient when we are changing multiple attributes at the same time. You will see examples of this in Chapter 5, Vector Animation. Basic transformations Like a path string, a transformation string is composed of a number of commands defined in the order in which they are to be interpreted. The syntax for the basic commands is as

follows, where brackets indicate optional parameters: 3.8.2

Príkaz	Parameter	Príklad
T, t	x, y	t50,100
R, r	uhol, (bod rotacie x, y)	r45,0,0
S, s	scale x, y, (scale bod x, y)	S 2,4.5,75,125

Tabuľka 3.7: Jednoduché príkazy na vytvorenie parametra stringu pre metodu v Element.transform(...) TODO

As with a path string, transformation strings have uppercase and lowercase variants. The uppercase variant means that we transform, irrespective of the previous transformation, while the lowercase variant takes previous transformations into account. The following examples demonstrate the basic transformations.[2, p. 52]

V tabulke 3.8.2 je prehľad transformácií umožňujúce pre SVG. TODO [8, p 73]

http://commons.oreilly.com/wiki/index.php/SVG_Essentials/Transforming_the_Coordinate_System

Transformation	Popis
translate(x,y)	Posunie súradnicový systém na dané x, y.
scale(xFactor, yFactor)	Vynasobí súradnice o daný násobok. Násobok môže byť aj zlomok, záporné číslo
rotate(uhol)	Zrotuje súradnice o daný uhol. Stred rotácie je (0,0).
rotate(uhol, centerX, centerY)	Zrotuje súradnice o daný uhol, s danými súradnicami stredu rotácie.
skewX(uhol)	Nakloní všetky x-ové súradnice v danom uhle.
skewY(uhol)	Nakloní všetky y-ové súradnice v danom uhle.

Tabuľka 3.8: SVG transformácie

TODO / TOTO JE Z JEDNEJ BAKALARKY O SVG ..

Všetky transformácie grafických prvkov sa realizujú pridaním požadovanej transformácie do atribútu "transform". Keď sa teda v zdrojovom kóde uvádza, že bol zmenený súradnicový systém vo vnútri nejakej skupiny g pridaním atribútu transform = "scale (2)", tak môžeme vyjadriť túto zmenu vetou: "Všetky prvky vo vnútri danej skupiny boli zväčšené 2x." Hodnotou atribútu transform je zoznam transformácií, z čoho vyplýva, že na ľubovoľný objekt možno súčasne aplikovať viac ako jednu transformáciu. Jednotlivé položky môžu byť oddelené medzerou alebo čiarkou. Matrix (a, b, c, d, e, f) - všeobecná transformačná matica

Translate (tx, ty) - translácie alias posun. Všeobecný zápis: 'matrix (1 0 0 1 tx ty)'

Scale (sx, sy) - zmena mierky, ak je uvedená len jedna hodnota, prehliadač predpokladá, že obidve hodnoty sú rovnaké. Všeobecný zápis: 'matrix (sx 0 0 sy 0 0)'

Rotate (uhol [stred-x stred-y]) - rotácia, otáčanie o zadaný uhol okolo ľubovoľného stredového bodu otáčania [súradnice v hranatých zátvorkách nemusia byť uvedené, potom sú ich predvolené hodnoty 0,0]. Všeobecne (so stredom rotácie v 0,0) možno zapísať ako 'matrix (cos (uhol) sin (uhol)-sin (uhol) cos (uhol) 0 0)'

SkewX (uhol) - skosenie pozdĺž osi X. Všeobecný zápis: 'matrix (1 0 tan (uhol) 1 0 0)'

SkewY (uhol) - skosenie pozdĺž osi Y. Všeobecný zápis: 'matrix (1 tan (uhol) 0 1 0 0)'

<animateTransform

attributeName="transform"

type="rotate"

values="0;360;0"

begin="2s" dur="4s"/>

Translation

Element.transform("T 300 0); todo [2][p 53]

Rotation

```
Element.transform("t 100 0 r 180 220 100");
```

Scaling

```
Element.transform("t 100 0 s 1.2 0.5");
```

TODO je tam este toho dost o tych maticiach.. a

3.8.3 Event handler

tuto by som porovnala ake su predefinovane v svg, a ake su v javascripte.. onstart, onend, onmove, draffing, dropping,

Transformation je dost rozsiahla..

3.9 Animácie

TUTO PRIBUDNE TABULKA TOHO CO SA DA REALIZAT CEZ SVG A CO CEZ SNAP.SVG. bud to bude na konci ako zhrnutie alebo na zacia <http://www.w3.org/1999/07/30/WD-SVG-19990730/animate.html> - zoznam dostupnych SVG animácií, prejdí si to prosím a skús najst analógie v snap.js knižnici. animateFlipbook nie je moc dôležitá, ale ostatné môžeš dať do bakalárky aj s príkladmi.

Animacia	SVG akcia	JavaScript akcia	Popis
Animácia			
Nastavenie farby			
Transformácia			
Skryvanie		attr(visibility: true)	
Atributy farby	animateColor	attr(fill: farba)	
Nastavenie atributov	set element	metoda attr	
zmena farby	animateColor element	attr(fill: farba)	
animovanie transformacie	animateTransform element	transform()	
	animateMotion element		zoznam atributov p

Tabuľka 3.9: Mapovacia tabuľka

V podstate SVG SMIL animacia elementy na menenie:

- The ‘animate’ element
- The ‘set’ element
- The ‘animateMotion’ element
- The ‘animateTransform’ element

O com bude táto časť:

- Animácia metód a jednoduchých atribútov animácie TODO ZMENA FARBY INDIKATORA
- Animovanie path TODO KAPITOLA O NADRZI
- Easing využívajúc kubickú Bézierovú syntax

- Animácie transformácie TODO KAPITOLA ROTOR
- Animácie využívajúce spoločné atribúty a animácie popri path TODO MAPA CESTY

3.10 Animacia - Element.animate(...)

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

- - attr (object) atribúty finalného produktu
- - duration (number) trvanie animácie v milisekundách ,
- - easing (function) vid tabulka, alebo kapitola o tom funkcie @mina
- funkcia, ktorá sa vykoná po skončení animácii

3.10.1 Animácia atribútov

TODO NAPR ZMENA FARBY INDIKATORA TOKU VODY

tabulka ake atributy sa daju menit napr

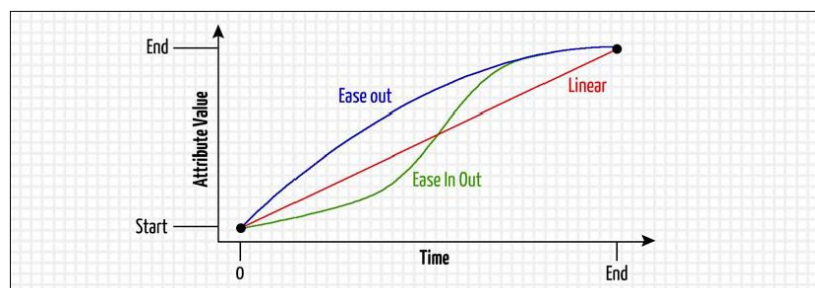
- visibility: / attr(visibility: true)
- okrajov: pozri tabulku c. 3.5
- farby: pozri tabulku
- textu: tabulka 3.7
- tvarov: tabulka 3.3

3.10.2 Animácia elementov, path

TODO NAPR ANIMOVANIE ZMENY PRITOKU VODY DO NADRZE

3.10.3 Animation easing

Easing describes how the value of an attribute varies with regard to time. By default, the value of an attribute changes consistently—that is, linearly—over the course of an animation but by specifying a particular easing type, we can change the way in which the attribute is animated. Consider the following graph that demonstrates three of the available easing types^{3.4: [2][p 70]}



Obr. 3.4: Graf troch easing typov

For each easing type, the rate at which the attribute changes varies along the graph in the time axis. Each easing type can be described as such:

- Linear: The value varies consistently from its start value to its end value over the course of the animation
- Ease Out: The value increases quickly towards its end point before slowing down towards the end of the animation
- Ease In Out: The value decreases slowly at first and then increases quickly before finally slowing down to its end point towards the end of the animation

Prazdna strana *vymaz*

3.10.4 Animácia transformácií

TODO ASI TEN VENTIL BY BOLO VHODNE ZANIMOVAT A TRANSFORMOVAT

3.10.5 Animacia vyuzivajuca vlastne atributy

3.10.6 Animacia podla path

toto uz v prikladoch mam ako mapu TODO / todo urobit pri nadrzi

3.10.7 Pausing a resuming animation

Kapitola 4

Práca s Existujúcimi SVG

V tejto kapitole sa budem zaoberať:

- Inkscape vektorovým grafickým editorom
- Manuál na vyšetrenie SVG paths
- Príklad zistenia SVG id

4.1 Inkscape

4.2 Definovanie id SVG

4.3 Zoznam id SVG atributov využite v príklade

Kapitola 5

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 5.1: Mapovacia tabuľka

5.1 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.

5.1.1 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 - odkazať tu na všetky možné atribúty, ktoré sa dajú zmeniť.

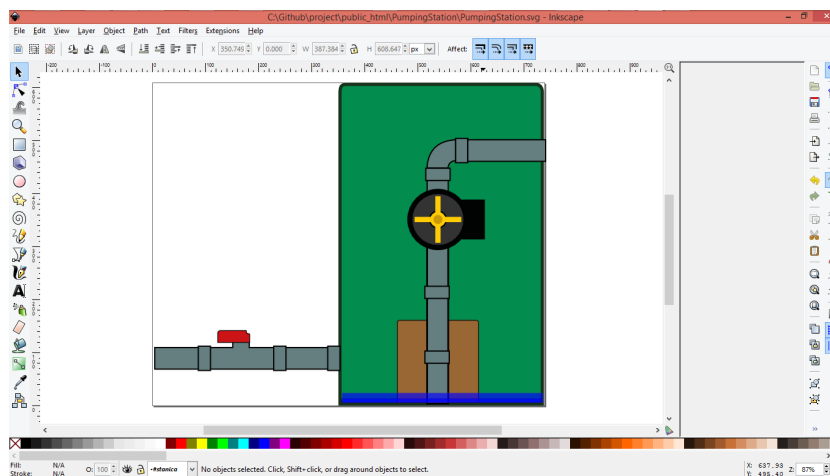
5.2 JavaScript

Kapitola 6

Príklad vytvorenia grafického komponentu v Inkscape

6.1 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 6.1.

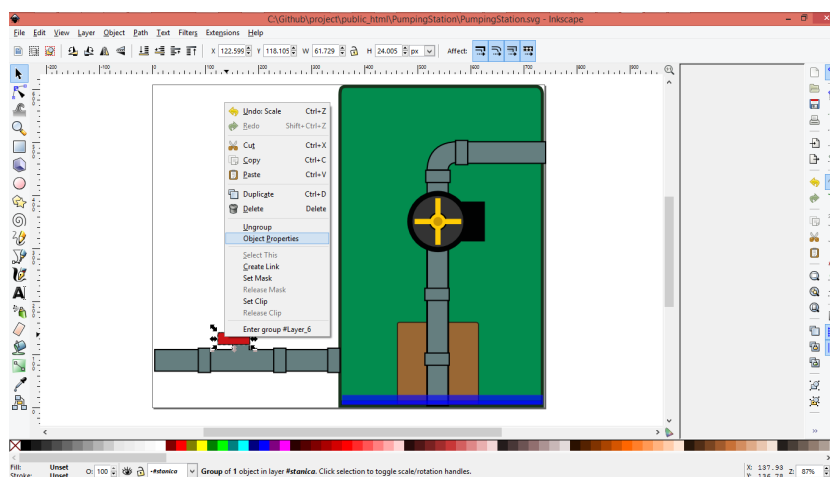


Obr. 6.1: Grafické prostredie programu Inkscape s nakreslenou prečerpávacou stanicou

=

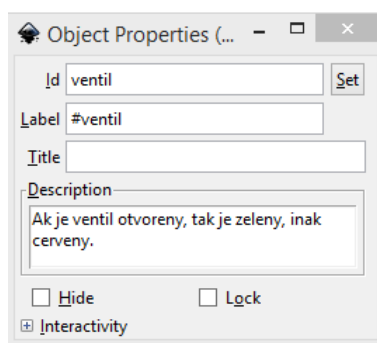
6.2 Definovanie id SVG

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 6.2.



Obr. 6.2: Zobrazenie menu pre daný objekt v Inkscape

Po kliknutí sa nám zobrazí okno s názvom Object Properties.



Obr. 6.3: Object Properties

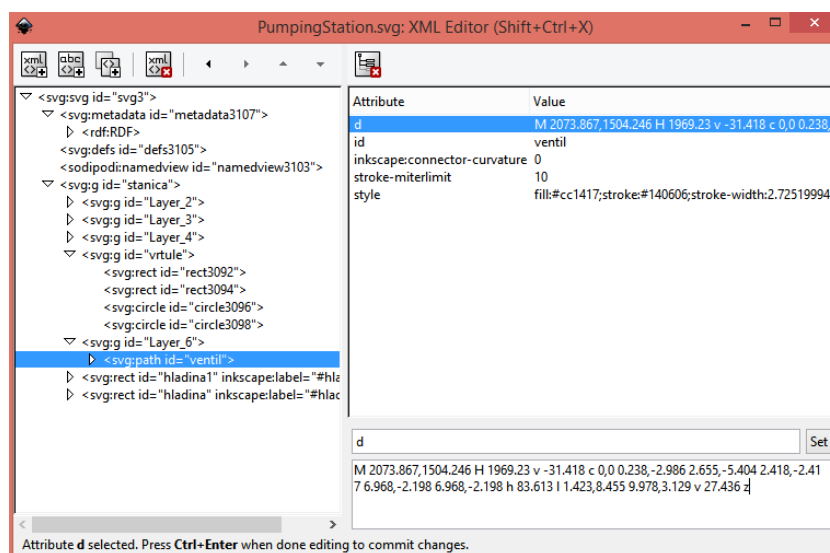
Z obrázka č.6.3 možno vyčítať aké je ID, predvolené sú tam napr. desc3072. Hodnoty je možné prepísať a zmeniť stlačením tlačidla Set. Pre nás je dôležitá hodnota v kolónke Label - #ventil. Na ovládanie časti svg elementu cez CSS selektor je potrebné si zapamätať hodnotu Labelu.

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.

6.2.1 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 6.4.

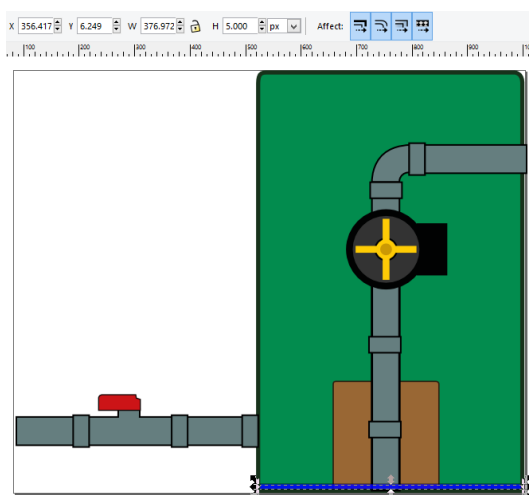


Obr. 6.4: Xml Editor v Inkscape

6.3 Zistenie atribútov SVG

6.3.1 Prázdna nádrž

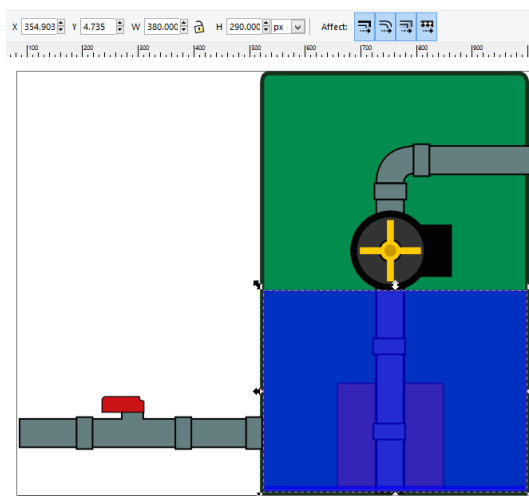
Prázdna nádrž je zobrazená na obrázku 6.5 Vyjadruje to, že do nádrže nevchádza tekutina. Hladina má výšku nastavenú na 9,64px. Šírka a súradnica x je rovnaká ako pri plnej nádrži. Súradnice osi y, x sú 1916,36 a 2507,84. Pri plnej nádrži sa zmení výška a os y. Pre animáciu zdvihnutia hladiny nádrže bude potrebné si zapamätať tieto súradnice.



Obr. 6.5: Prázdna nádrž prečerpávacej stanice

6.3.2 Plná nádrž

Na obrázku č. 6.6 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.



Obr. 6.6: Plná nádrž

V SVG súbore je to zapísané nasledovným kódom.

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

Kapitola 7

Integrácia grafického komponentu pre dynamické ovládanie SVG objektu

Súborová štruktúra príkladu:

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

7.1 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.

7.1.1 Kód

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

7.1.2 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>
```

7.2 PumpingStation.js

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

7.2.1 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");
}
```

7.2.2 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

7.2.3 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.

7.2.4 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

7.3 REST API

7.4 REST API pre graficke komponenty

Graficke komponenty pre vizualizáciu dát zo systému D2000 budú umiestňované na HTML stránkach používaných ako súčasť web rozhrania frameworku D2000 WebSuite. Tento framework je založený na technológii Java Enterprise Edition a Java Server Faces. Životný cyklus web stránky - načítanie stránky s technologickou schémou, pre prvé zobrazenie kompletnej schémy je potrebný plný dátový súbor. príklad REST URL

`http://localhost:8080/scada-demo/rest/pumpingstation/gatfulldataset` príklad JSON dát:

(možno uviesť dáta z pumpujúcej stanice) Zobrazenie stránky v rámci jednej HTTP session, typicky vo web aplikáciach kde sa užívateľ prihlási pomocou mena a hesla, trvanie jeho session je obmedzené na predom stanovený čas, napríklad jednu hodinu. Pri zobrazení zložitejšej technologickej schémy, je potrebné optimalizovať množstvo prenesených dát a interakciu s DOM stránkou. Preto je počas zobrazenia schémy výhodné implementovať čiastočné aktualizácie, ktoré menia len dotknuté časti schémy a nie celú schému ako je tomu pri načítaní stránky. príklad REST URL

`http://localhost:8080/scada-demo/rest/pumpingstation/getvalvestatus` príklad JSON dát:

(možno uviesť dáta z pumpujúcej stanice) príklad REST URL

`http://localhost:8080/scada-demo/rest/pumpingstation/getrotorstatus` príklad JSON dát:

(možno uviesť dáta z pumpujúcej stanice) príklad REST URL

`http://localhost:8080/scada-demo/rest/pumpingstation/getwaterlevel` príklad JSON dát:

(možno uviesť dáta z pumpujúcej stanice)

7.5 Data binding pre systém D2000

Priama väzba v prípade jednoduchých schém. Jednoduchá schéma predstavuje vizualizáciu meraného alebo počítaného bodu v systéme D2000. Takáto vizualizácia je realizovaná pomocou widgetu (možno uviesť príklad ako teplomer alebo ručičkový merací, obrázok) jedna sa o mapovanie 1:1. Zložitejšie schémy pozostávajúce z vizualizácie vy-

robneho procesu alebo komplexnej technologie su mapovane vo vzťahu n:1, teda jedna schena vizualizuje data z n meranych alebo pocitanych bodov, pripadne ziskava data cez asynchrone volania RPC (remote procedure call) systemu D2000. Ak to vyžaduje logika aplikacie, server implementuje datovy zdroj (service), ktory agreguje data a systemove udalosti. Pre dosiahnutie real-time odozvy web rozhrania vyhodne použiť obojstrannu komunikáciu medzi web browserom a serverom - technológiu web sockets. (možes uviesť niečo o web socketoch, skús pohľadať na webe), vo štvrtok to možme doladiť.

Kapitola 8

TODO — Analýza výkonnosti a obmedzení SVG

<http://www.sitepoint.com/advanced-snap-svg/> Performance Improvements

One way to improve performance when manipulating the DOM is using DocumentFragments. Fragments are minimal containers for DOM nodes. Introduced a few years ago, they allow you to inexpensively manipulate entire subtrees, and then clone and add a whole subtree with n nodes to our page with 2 method calls instead of n . The actual difference is explained in details on John Resig's blog.

Snap allows for native use of fragments as well, with two methods:

`Snap.parse(svg)` takes a single argument, a string with SVG code, parses it and returns a fragment that can be later appended to any drawing surface.

`Snap.fragment(varargs)` takes a variable number of elements or strings, and creates a single fragment containing all the elements provided.

Especially for large svg drawings, fragments can lead to a huge performance saving, when used appropriately.

<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) - zmenim veľkosť tvaru na danane suradnice, *translate*(tx, ty) - presuniem na ine miesto - zmenim suradnice

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
- [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.w3.org/TR/SVG/painting.html#StrokeProperties>
- [13] http://www.w3schools.com/html/html5_svg.asp
- [14] www.w3schools.com/svg/svg_inhtml.asp
- [15] The JavaScript SVG library for the modern web, <http://snapsvg.io/>.
- [16] <http://raphaeljs.com/>
- [17] <http://d3js.org/>
- [18] <http://www.svgjs.com/>
- [19] 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

1.1	HTML 5 API	4
1.2	Podpora SVG vo webových prehliadačoch	7
3.1	Súradnicový systém plátna s bodom (300, 200)	14
3.2	Príklad zoskupenia elementov a následná zmena atribútov	20
3.3	Príklad na zmenu atribútov v texte.	22
3.4	Graf troch easing typov	29
6.1	Grafické prostredie programu Inkscape s nakreslenou prečerpávacou stanicou	36
6.2	Zobrazenie menu pre daný objekt v Inkscape	37
6.3	Object Properties	37
6.4	Xml Editor v Inkscape	38
6.5	Prázdna nádrž prečerpávacej stanice	39
6.6	Plná nádrž	40

Zoznam tabuliek

1.1	Podpora HTML <code><svg></code> elementu v webových prehliadačoch	7
1.2	Porovnanie Canvas a SVG	7
3.1	Spôsoby vytvorenia SVG v HTML dokumente	15
3.2	Zoznam tvarov, ktoré podporuje SVG a Snap API, a TODO TODO PO- ROZMYSLAT NAD NAZVOM a atribúty pre definovanie tvaru	16
3.3	Názvy atribútov a ich význam	17
3.4	Niekoľko príkazov na tvorbu Path elementu	18
3.5	Výber možných stroke atribútov	19
3.6	Atribúty na zmenu vlastností elementu text	21
3.7	Jednoduché príkazy na vytvorenie parametra stringu pre metodu v Ele- ment.transform(...) TODO	24
3.8	SVG transformácie	24
3.9	Mapovacia tabuľka	27
5.1	Mapovacia tabuľka	33

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

DELETE THIS PAGE

.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

dalsie čo ma napadlo - je kapitola [2, p. 81] ktorá je o práci s existujúcimi svg subormi / keďže to takmer na konci, tak by som mohla prehodit taktiež kapitoly, že najprv budem písať o svg knižniciach, ako sa to tam tvorí, a tak ďalej, a potom prejdem na inkscape a príklad a animáciu ..

Making our SVG responsive

There are 2 key points to making our svg responsive. The width/height must be 100% and a viewBox MUST be defined.

Výška a šírka musí byť udaná relatívne, napríklad v percentách. Musí byť nastavený aj viewBox. To môžem realizovať dvoma spôsobmi: 1. spôsob

.2 Kapitoly by mali byt nasledovne

1. ciel prace
2. metodika prace
3. definovanie zakladnych pojmov / co je html5, co je scada system, a co je svg vektorova grafika (takmer hotove, len nemam scada systemy definovane),
4. ANALYZA javascriptovych kniznic / (toto uz je takmer hotove, este tam pridat jquery)
5. Postup **tvorby** grafickych komponentov cez Snap...
6. analyza nastrojov - preco som sa rozhodla pouzit inkscape na tvorbu svg obrazkov, //este tam musim spomenut moznost exportu do formatov...
7. postup na vytvorenie OBRAZKA SVG V INKSCAPE a integracia do snapu [2, p. 82-5]
8. priklad vytvorenia precepavacej stanice v Inkscape (toto je uz takmer hotove)
9. Postup **animacie** uz vytvorených grafických elementov
10. Příklad v kode animovanie Precerpavaciej stanice... (Takmer hotove) toto je vlastne ta implementacia vzorovej sady grafickych komponentov
11. Navrh REST API na prepojenie grafickych komponentov so SCADA SERVEROM *toto no nemam / mam iba priklad kodu jednoducheho*
12. Analyza moznosti automatickeho mapovania api grafickych prvkov pomocou metadat na existujuce api dostupne pre scada serverr D2000 *toto no nemam vobec*
13. analyza vykonnosti a vykonnostne obmedzenia - toto iba okrajovo spomeniem - vseobecne preco je lepsie svg, a preco nie je vhodne, v prvej kapitole som pisala rozdiel medzi canvas a svg - istym sposobom to je analyza obmedzeni.. ?
14. zhrnutie