Pedagogická fakulta
Faculty of Education

Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

Jihočeská univerzita v Českých Budějovicích

Pedagogická fakulta

Katedra informatiky

Možnosti využití SVG ve webovém designu

Possibilities of using SVG in web design

Bakalářská práce

Vypracoval: Tomáš Včelák

Vedoucí práce: PaedDr. Petr Pexa, Ph.D.

České Budějovice 2019

JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH
Fakulta pedagogická
Akademický rok: **2016/2017**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Tomáš VČELÁŘ**
Osobní číslo:	**P15677**
Studijní program:	**B7507 Specializace v pedagogice**
Studijní obor:	**Informační technologie a e-learning**
Název tématu:	**Možnosti využití SVG ve webovém designu**
Zadávající katedra:	**Katedra informatiky**

Zásady pro vypracování:

Cílem bakalářské práce je představit možnosti využití jazyka SVG (Scalable Vector Graphics) ve webovém designu. Jazykem SVG vzniká škálovatelná vektorová grafika, která vzhledem k problematické podpoře ve starších verzích prohlížeče MS Explorer nenašla dlouho na webu uplatnění. V současné době je již možné plně využít výhod SVG při tvorbě front-end webu, jako je např. nezávislost na rozlišení displeje (mj. v souvislosti s nástupem Retina displejů), možnost libovolné změny velikosti bez ztráty vizuální kvality grafiky, strojová čitelnost obsahu a především minimální datová náročnost, díky nimž byl jazyk SVG také zařazen do rodiny HTML5 technologií. Autor se ve své práci zaměří na způsoby získání SVG k využití pro webový design včetně generování PNG alternativ a na varianty implementace SVG do kódu webu. Základem práce bude demonstrace využití SVG pro práci s webovým textem (výplně, přechody, filtry, animované efekty), s obrázky (výřezy a filtry), pro animace zatržítek a ikon, animované interakce s formulářovými input poli a tlačítky, interakční přechody, tvorbu bannerů a map s využitím animací cest apod. Vše bude představeno na vlastní webové prezentaci s mnoha originálními praktickými příklady se zdůrazněním výhod použitých technik v porovnání s tradiční webovou grafikou bitmapovou.

Rozsah grafických prací: **CD ROM**

Rozsah pracovní zprávy: **40**

Forma zpracování bakalářské práce: **tištěná**

Seznam odborné literatury:

1. **TRYTHALL, Joni. Pocket Guide to Writing SVG [online]. 2016 [cit. 2017-04-11]. Dostupné z: http://svgpocketguide.com/book/**
2. **MICHÁLEK, Martin. SVG: vektorový formát, který na webu chybél [online]. 2017 [cit. 2017-04-11]. Dostupné z: http://www.vzhurudolu.cz/prirucka/svg**
3. **COYIER, Chris. Using SVG [online]. 2013 [cit. 2017-04-11]. Dostupné z: https://css-tricks.com/using-svg/**
4. **ZASTROW, Philip. A bit about SVG [online]. 2014 [cit. 2017-04-11]. Dostupné z: https://seesparkbox.com/foundry/a_bit_about_svg**
5. **W3Schools. HTML5 SVG [online]. [cit. 2017-04-11]. Dostupné z: https://www.w3schools.com/html/html5_svg.asp**

Vedoucí bakalářské práce: **PaedDr. Petr Pexa, Ph.D.**
Katedra informatiky

Datum zadání bakalářské práce: **24. dubna 2017**

Termín odevzdání bakalářské práce: **30. dubna 2018**

Mgr. Michal Vančura, Ph.D.
děkan

L.S.

doc. PaedDr. Jiří Vaníček, Ph.D
vedoucí katedry

V Českých Budějovicích dne 24. dubna 2017

Prohlášení

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích dne 10. dubna 2019

Tomáš Včelák

Abstrakt/Anotace

Cílem bakalářské práce je představit možnosti využití jazyka SVG (Scalable Vector Graphics) ve webovém designu. Jazykem SVG vzniká škálovatelná vektorová grafika, která vzhledem k problematické podpoře ve starších verzích prohlížeče MS Explorer nenašla dlouho na webu uplatnění. V současné době je již možné plně využít výhod SVG při tvorbě front-end webu, jako je např. nezávislost na rozlišení displeje (mj. v souvislosti s nástupem Retina displejů), možnost libovolné změny velikosti bez ztráty vizuální kvality grafiky, strojová čitelnost obsahu a především minimální datová náročnost, díky nimž byl jazyk SVG také zařazen do rodiny HTML5 technologií. Autor se ve své práci zaměří na způsoby získání SVG k využití pro webový design včetně generování PNG alternativ a na varianty implementace SVG do kódu webu. Základem práce bude demonstrace využití SVG pro práci s webovým textem (výplně, přechody, filtry, animované efekty), s obrázky (výřezy a filtry), pro animace zatržítek a ikon, animované interakce s formulářovými input poli a tlačítky, interakční přechody, tvorbu bannerů a map s využitím animací cest apod. Vše bude představeno na vlastní webové prezentaci s mnoha originálními praktickými příklady se zdůrazněním výhod použitých technik v porovnání s tradiční webovou grafikou bitmapovou.

Klíčová slova

SVG, HTML5, webový design, front-end webu, vektorová grafika

Abstract

The goal of the thesis is to introduce possibilities of SVG (Scalable Vector Graphics) in web design. SVG is a scalable vector graphic format which didn't find application on the web for a long time due to poor support in older versions of Microsoft Explorer. Nowadays, it is finally possible to fully benefit advantages of SVG in front-end web design, e.g. independence on a screen resolution (especially with wide-spread Retina displays), resizing without visual quality loss, machine readability of embedded text content, and most importantly small file size. These features allowed SVG to be recognized as an HTML5 family member. Author of the thesis will focus on methods of bringing SVG to web design, including methods of generating PNG alternatives and variants of SVG inclusion into web page code. The foundation of the thesis will be a web page presentation, which will demonstrate use of SVG on text (fillings, gradients, filters, and animations), images (cropping and filters), animations of checkboxes and icons, animated interactions with form inputs and buttons, animated transitions, banners and maps with animated paths, etc. The presentation will emphasize advantages of used techniques in comparison with traditional bitmap graphics.

Keywords

SVG, HTML5, web design, front-end web design, vector graphics

Poděkování

Děkuji panu PaedDr. Petru Pexovi, Ph.D. za jeho rady. Byly pro mne velmi cenné při realizaci této práce. Také děkuji všem, kteří svými poznámkami ovlivnili výslednou práci.

Obsah

1 Úvod

Bakalářská práce se zabývá formátem SVG, který popisuje dvojrozměrnou vektorovou grafiku jazykem XML. Vzhledem k současné podpoře prohlížečů je možné obrázky SVG plně využít při tvorbě webových stránek.

1.1 Východiska práce

Vetšina webových vývojářů a designerů chce, aby všechna grafika, kterou užívají, vypadala hezky na každém rozlišení displeje a přitom její velikost byla co nejmenší. Dnešní pixelová doba pro ně může být správná, ale také může představovat veliké problémy.[1] Existuje možnost dosáhnout kvalitního řešení a to díky SVG (Scalable Vector Graphic). SVG může vypadat ostře na všech rozlišeních. Právě responzitiva je dnes velmi aktuální kvůli mobilním zařízením, ale také v souvislosti s nástupem Retina displejů.[2] Zároveň může mít velmi malou velikost a nebude tedy datově náročná. Dá se snadno vytvářet a upravovat a proto je SVG vhodné k využití při tvorbě front–end webu.[1]

1.2 Cíle práce

Cílem bakalářské práce je představit možnosti využití jazyka SVG ve webovém designu. Teoretická část bude zaměřena na způsoby získání SVG k využití pro webový design včetně generování PNG alternativ a na varianty implementace SVG do kódu webu. Základem práce bude demonstrace využití SVG pro práci s textem, obrázky a animacemi. Praktická část představí SVG v rámci webové prezentace s praktickými příklady se zdůrazněním výhod použitých technik v porovnání s tradiční webovou bitmapovou grafikou.

1.3 Metody práce

V úvodu práce popíši základy formátu SVG. Představím, jaké jsou výhody SVG obrázků pro využívání ve webovém designu a jaký je rozdíl oproti bitma-

pové grafice. Zaměřím se na způsoby přípravy SVG s vhodným výběrem optimalizace a na možné náhrady bitmapovými obrázky. Uvedu, jaké jsou možnosti implementace a jejich výhody pro určité použití. Vysvětlím správnou strukturu SVG dokumentu a správné zobrazení v prohlížeči. Dále se zaměřím na manipulaci pomocí CSS a JavaScriptu, na práci s SVG textem a obrázky s použitím výřezů, masek a filtrů. Závěrem popíši způsoby animace SVG.

2 Základy formátu SVG

SVG neboli škálovatelná vektorová grafika (Scalable Vector Graphics) je dvourozměrný vektorový grafický formát založený na značkovacím jazyku XML. Dodržuje shodné standardy, jak by měl být napsán a jak by na něj měl klientský software reagovat. Vše v jazyku XML je napsáno v textovém rozhraní a lze jej obecně číst nejen pomocí strojů, ale lidé se v textu také mohou bez problému orientovat. Umožňuje manipulaci s objekty pomocí CSS a JavaScriptu a to stejným způsobem jako používáme CSS a JavaScript ve spojení s HTML.[3]

Všechny grafické prvky SVG jsou vykresleny v souřadnicovém systému na osách x a y. Při vytváření SVG dáváme prohlížečům pokyny o tom, kam vykreslit jednotlivé body souřadnicového systému a navzájem je propojit. Spojením vykreslených bodů můžeme vytvářet tvary. Existuje několik základních tvarů (čára, kružnice, mnohoúhelník apod.).

2.1 Rozlišení SVG obrázku

Existuje mnoho výhod proč využívat právě SVG pro grafické prvky ve webovém designu. Můžeme očekávat, že se SVG bude chovat stejně jako bitmapová grafická podoba PNG s nekonečným rozlišením, ale tak to opravdu není.[4] Největším rozdílem a výhodou oproti tradičním obrázkům formátu PNG nebo JPEG je nezávislost na rozlišení. Vzhledem k tomu, že je SVG formát založen na vektoru, při zvětšování či zmenšování nedochází k žádné ztrátě kvality, takže můžeme použít jakoukoliv velikost, jak je ukázáno na obrázku 1. To je obzvláště užitečné pro responzivní webové stránky, které musí vypadat a fungovat dobře i v široké škále velikostí na různých typech zařízení.[5]

2.2 Velikost SVG souboru

Výhodou při použití SVG je velikost souborů. Obrázky na responzivních webových stránkách ve formátu PNG nebo JPEG mohou mít za následek velkou

velikost souboru. Tyto obrázky jsou přizpůsobeny největší možné kvalitě, často jsou větší než je třeba, což znamená, že je prohlížeč nucen stáhnout zbytečně velké soubory.[4] Vzhledem k tomu, že je vektorová grafika škálovatelná, můžeme mít velmi malé velikosti souborů bez ohledu na to, jak velké budeme obrázky zobrazovat. Bitmapová grafika musí nést informace o každém pixelu zvlášť. U obrázku 100×100 pixelů máme 10000 pixelů s informací, oproti tomu SVG má jen instrukce, jak obrázek vykreslit a díky tomu mohou mít menší velikost. Tento faktor má pozitivní vliv na rychlost stahování a čas načítání.[5]

2.3 Modifikace SVG obrázku

Vhledem k tomu, že SVG není bitmapová grafika, můžeme soubory jednoduše modifikovat.[4] U bitmapové grafiky lze úpravy provést v editovacím softwaru, exportovat a znovu nahrávat, ale úpravy jsou omezené. SVG můžeme upravit jak v editovacím softwaru tak přímo. Při použití „inline SVG“ lze kód přidat do HTML stránky, kdy je grafika vykreslována prohlížečem na základě kódu. Není třeba provádět žádost pro načtení souboru obrázku a při potřebě záměny jen upravíme kód.[5] Výhodou je, že můžeme SVG upravovat jednoduše pomocí CSS nebo JavaScriptu.[4]

2.4 Animovaná a dynamická grafika

Možnosti stylování SVG pomocí CSS a JavaScriptu je možné použít i pro vytváření animací. Například můžeme vytvořit hodiny, které ukazují aktuální čas nebo třeba ikonu, které bude odpovídat barvě textu, který bude umístěn vedle ní. SVG obrázek může reagovat a měnit se různými způsoby.[6]

2.5 Spritesheets a ikony

Na webových stránkách jsou často užívané ikony. Pomocí SVG můžeme do jedné grafiky vložit celou sadu ikon, z toho vyplývá pouze jeden malý soubor

místo několika souborů. Výhodou je rychlejší načtení stránky. Tento druh sady ikon nazýváme spritesheet.[6]

SVG obrázky mají využití i v řadě dalších případů, avšak nejsou vhodné pro každou situaci. Samozřejmě nemohou nahradit fotografie které na webových stránkách často používáme. U fotografií budou muset být stále používány formáty jako PNG nebo JPG. V ideálním případě bychom použili obrázky formátu SVG pro ikony, ilustrace, loga, grafy a u obrázků, které se na webových stránkách častokrát opakují.[7]

2.6 Podpora SVG v prohlížečích

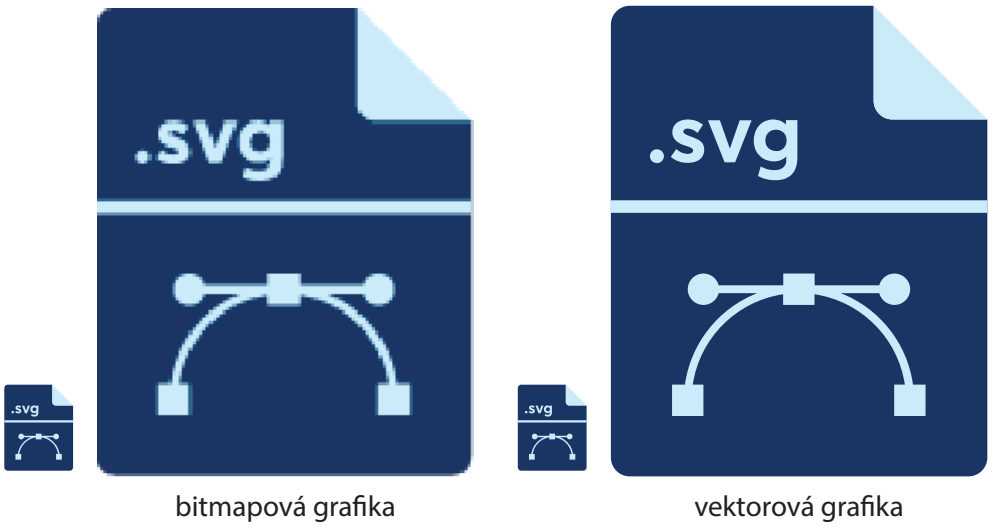
Většina dnešních moderních prohlížečů dokáže formát SVG zobrazit. Jediné prohlížeče, které skutečně postrádají podporu, jsou starší verze Internetu Explorer (verze 8 a nižší) a starší verze systému Android. Celkově velmi malé procento populace používá právě tyto prohlížeče a číslo použitelnosti se i nadále snižuje. Proto lze dnes na webových stránkách SVG bezpečně využít. Pokud bychom ale měli obavy, můžeme využít alternativy SVG a pro starší prohlížeče využít záložní PNG, pro jehož pořízení máme několik nástrojů, jako například Grumpicon ze skupiny Filement, knihovny Modernizr nebo SVGeezy.[5]

2.7 Rozdíl SVG oproti bitmapové grafice

Bitmapová grafika Bitmapová grafika, které se také říká rastrová, je oproti SVG založena na jednotlivých obrazových bodech, které nazýváme pixely. Pixely jsou uspořádány do pravidelné sítě (rastr). Každý takový pixel má určenou svou přesnou polohu, barvu, jas a případně průhlednost. Kvalitu bitmapového obrázku určuje rozlišení a barevná hloubka. Rozlišení je počet pixelů na jeden palec čtvereční (DPI – Dots per inch, který často používáme při tisku nebo PPI – Pixels per inch, který používáme na zobrazovacím zařízení jako je třeba monitor počítače). Barevná hloubka je počet bitů, které nesou informaci o barvě

pixelu. Pokud rozlišení u bitmapové grafiky neodpovídá rozlišení zobrazovacího zařízení, dochází k degradaci kvality. Ve webovém designu má optimalizovaná bitmapová grafika stále uplatnění a to zejména pro fotografie.[8]

Vektorová grafika Vektorová grafika je vytvářena na základě přesných geometrických tvarů. Používáme u ní právě formát SVG nebo formáty odvozené od vektorových editorů (AI, CDR, ...) případně PostScriptového EPS, PS. Tvary jsou tvořeny z bodů, linek a především z křivek, díky kterým je možné vytvořit jakýkoliv tvar. Tyto geometrické tvary jsou pomocí matematických funkcí vyjádřeny na osách x, y. Díky matematickému vyjádření nedochází při zvětšování ke ztrátě kvality, což se považuje za velkou výhodu. Další výhodou je práce s jednotlivými objekty v obrázku nezávisle na objektech ostatních.[9]



Obrázek 1: Porovnání SVG a bitmapové grafiky, s ukázkou nezávislosti rozlišení

3 Příprava a optimalizace SVG

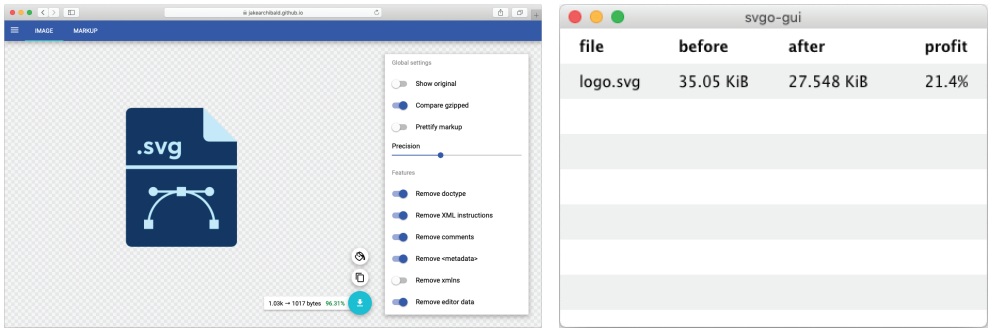
SVG obrázky můžeme vytvořit přímo v textovém rozhraní napsáním kódu nebo je získat exportováním z grafického editoru. Pro práci s SVG se používají editory pro vektorovou grafiku jako jsou například Adobe Illustrator, Sketch, CorelDraw, Inkscape apod. Výběr editoru záleží na preferencích uživatele. [10]

Úspora velikosti souboru je u SVG znatelná. Stejně tak jako potřebujeme ve webovém designu optimalizovat bitmapovou grafiku, měla by být před použitím optimalizována i grafika v SVG. Máme několik postupů, jak dosáhnout nejlepšího výsledku.[11]

Nejprve musíme optimalizovat obrázek ve vektorovém grafickém editoru. Odstraníme ze souboru vše zbytečné. Musíme se ujistit, že používáme co nejméně kotevních bodů na našich tvarech či cestách. Smažeme neviditelné vrstvy. Pokud vrstvy pojmenujeme, budou v SVG uvedeny jako ID daného elementu. To sice přidá na celkové velikosti, ale může to pomoci při stylování. Aby se správně zobrazoval text, musíme ho převést na obrysy (tedy pokud nezamýšlíme stylování pomocí webového písma). Při kombinaci tvarů, cest a tahů bychom měli vše převést na pevné tvary. Žádný obsah nesmíme maskovat. Při skrytí, případně při přetečení grafiky mimo plátno, se obsah v kódu také zobrazí. Je vhodné zjednodušit skupiny a uděláme nejlépe, když se používání skupin budeme snažit vyhnout. Před exportem bychom měli zajistit, aby obrázek seděl na celých obrazových bodech (20,4×20,8 není vhodné) kvůli skutečně ostrému obrazu. Pokud máme dva nebo více tvarů, které se překrývají, je vhodné odstranit nepotřebné překrytí. Poté se může zdát, že se mezi tvary objeví bílá čára. Abychom tomu zabránili, můžeme objekty lehce překrýt. Nakonec ořízneme plátno. Po exportu musíme zkontrolovat v textovém editoru, že atribut `viewBox` začíná na 0 0. Pokud souřadnice x a y začínají v jiném bodě, je lepší vytvořit nový soubor, obsah vložit do středu a oříznout plátno.[12]

Po optimalizaci v grafickém editoru je vhodné soubor prohnat nástrojem pro optimalizaci. Jedním z nich je například SVGO[1]. SVGO je optimalizační open source projekt založený na Node.js. Při používání SVGO bychom měli brát na vědomí, že nástroj může soubor zoptimalizovat až příliš a tím ztížit plánované manipulace s CSS nebo s JavaScriptem. Možné je použít i pohodlnější verzi SVGO–GUI, do které lze soubory jednoduše přetáhnout. Je skvělá pro hromadnou a rychlou optimalizaci, ale neobsahuje detailní nastavení. Pokud chceme získat menší soubor budeme muset použít jinou variantu.

SVGOMG[2] je perfektní optimalizační nástroj vytvořen Jakem Archibaldem, který lze použít online i offline. Obsahuje mnoho nastavení se kterým lze dosáhnout skvělého výsledku.[11]



Obrázek 2: Ukázka optimalizace v SVGOMG a SVGO-GUI

Dále je vhodné soubor otevřít v textovém editoru a doladit poslední detaily, zkontrolovat a případně odstranit zbývající nepotřebný kód. Můžeme také vhodně aplikovat běžné atributy a tím vytvořit lépe čitelný kód.[1]

Neměli bychom zapomenout zapnutí komprese gzip pro SVG na webovém serveru v souboru .htaccess.[1]

Pokud se ve webovém designu naskytne situace dvou stejných obrázku, můžeme použít `<use>` pro opakování, což kompletní objekt také zmenší.[1]

[1]https://github.com/svg/svgo
[2]https://jakearchibald.github.io/svgomg/

4 Možné alternativy SVG

Měli bychom zajistit, aby se všem grafika zobrazila. Pokud tedy máme obavy z toho, že někteří uživatelé stále používají starší prohlížeče, můžeme SVG nahradit bitmapovým obrázkem. Prvním krokem je pro každý SVG vytvořit náhradu. Vhodnou možností je například PNG. Můžeme PNG exportovat přímo z grafického editoru nebo použít jiné, případně online, nástroje. Jedním z nich je například nástroj pro generování PNG alternativ grunt-svg-2png[3] pomocí modulu Grunt.[13]

Dále potřebujeme zajistit detekci SVG v prohlížeči. Existují různá řešení využívající JavaScript. Jednou z možností je knihovna Modernizr[4], která detekuje podporu SVG pomocí příkazu `document.createElementNS`. Další možností pro kontrolu prohlížeče je odkaz na objekt pomocí `window.SVGSVGElement`.[14] Spolehlivou detekcí SVG je testování prohlížeče pomocí `document.implementation.hasFuture`. Kód na příkladu 1 zajistí, že v prohlížeči bez podpory SVG bude u SVG objektů nastavena třída `.no-svg`.[13]

```
1 if (!document.implementation.hasFeature("http://www.
    w3.org/TR/SVG11/feature#Image", "1.1")) {
2   document.documentElement.className = "no-svg";}
```

Příklad 1: Metoda detekce podpory SVG pomocí funkce `document.implementation.hasFuture`

Po zajištění detekce podpory SVG přejdeme ke vkládání alternativ. Máme několik způsobů, které můžeme použít. Než budeme náhrady implementovat, je vhodné vědět jaký typ potřebujeme.[15]

[3] https://github.com/dbushell/grunt-svg2png
[4] https://modernizr.com

Skrytí obrázku Pokud máme SVG obrázek, u kterého je smysl v textu jasně vyjádřen a jeho skrytí neohrozí funkčnost webu. Můžeme ho skrýt.[15]

Nahrazení textem Pokud SVG obrázek může být vyjádřen textem, pouze zajistíme, aby se místo obrázku objevil daný text.[15]

Nahrazení bitmapou Jedná se o PNG nebo GIF obrázek, který vypadá stejně jako SVG, jen s větší velikostí souboru a horším rozlišením.[15]

Nahrazení jiným aktivním prvkem V případě náhrady animované a interaktivní grafiky SVG je PNG nevhodné. Je za potřebí grafický jazyk s interaktivním DOM. Můžeme SVG převést na Flash a přepsat kód v jazyce ActionScript[5]. Nebo můžeme použít JavaScript knihovnu Raphaël[6], která pracuje s vektorovou grafikou a pokud jí prohlížeč nepodporuje, použije VML (Vector Markup Language od společnosti Microsoft). Výsledek by tedy měl fungovat ve všech prohlížečích stejně.[15]

Alternativa SVG jako `<img>` Pokud pro vložení SVG používáme značku `<img>`, můžeme náhradu zajistit výměnou koncovky pomocí JavaScriptu, jak je ukázáno na příkladu 2.[13]

```
1 if (!document.implementation.hasFeature("http://www.
    w3.org/TR/SVG11/feature#Image", "1.1")) {
2   var imgs = document.getElementsByTagName('img');
3   var endsWithDotSvg = /.*\.svg$/
4   var i = 0;
5   var l = imgs.length;
6   for(; i != l; ++i) {
7       if(imgs[i].src.match(endsWithDotSvg)) {
```

[5] http://svg2swf.sourceforge.net
[6] http://dmitrybaranovskiy.github.io/raphael/

```
8             imgs[i].src = imgs[i].src.slice(0, -3) + '
                  png';
9         }
10    }
11 }
```

Příklad 2: Zajištění výměny koncovky z SVG na PNG

Stejně to dělá i knihovna SVGeezy[7], která zajistí detekci a obstará změnu koncovky dle potřeby na PNG. Je to jednoduché a funkční, ale prohlížeče, které ji nepodporují mohou stahovat dva soubory, náhradní PNG a SVG, i když je nepotřebují.

Pokud potřebujeme použít inline metodu, můžeme použít kód uvedený v příkladu 3.

```
1 <img src="obrazek.svg" onerror="this.src='obrazek.png
    '; this.onerror=null;">
```

Příklad 3: Metoda zajištění inline náhrady

Na příkladu 4 je ukázána stejná metoda za použití knihovny Modernizr a jQuery..[15]

```
1 if (!Modernizr.svg) {
2   $("img[src$='.svg']")
3     .attr("src", fallback);}
```

Příklad 4: Metoda detekce pomocí Modernizr

[7] https://github.com/benhowdle89/svgeezy

Další knihovnou pro náhradu je SVGInjector[8], která pomáhá se záměnou `<img>` s pomocí inline SVG a Javascriptu. Nebo jQuery plugin SVGMagic[9], který nachází SVG soubory (na pozadi a inline) a vytváří pro ně PNG alternativy.[15]

Element `<picture>` umí detekovat SVG a díky němu můžeme záměnu dělat bez JavaScriptu. V současné době většina prohlížečů element `<picture>` podporuje. Pro prohlížeče, které tento element nepodporují, musíme v hlavičce dokumentu zahrnout JavaScript picturefill od Filement Group.[15]

```
1 <picture>
2   <source type="image/svg+xml" srcset="obrazek.svg">
3   <img src="nahrada.png" alt="">
4 </picture>
```

Příklad 5: Náhrada SVG zajištěná prvkem `<picture>`

Alternativa SVG jako `<object>` Nejjednodušší způsob, jak poskytnout náhradu, je `<object>`. Funguje tak, že pokud obsah elementu `<object>` nelze zobrazit, zobrazí vložený obsah. Tím může být jiný obrázek, ale i formátovaný text nebo jiný objekt. Tato možnost je ukázána v příkladu 6.[15]

```
1 <object type="image/svg+xml" data="svg-ok.svg">
2   <img src="svg-no.png" alt="Nepodporuje SVG">
3 </object>
4 <object type="image/svg+xml" data="svg-ok.svg">
5   <p class="warning">Nepodporuje SVG!</p>
6 </object>
```

Příklad 6: Náhrada SVG zajištěná prvkem `<object>`

[8] https://github.com/iconic/SVGInjector
[9] http://dirkgroenen.nl/SVGMagic/

Alternativa SVG jako obrázek na pozadí Když použijeme k detekci SVG již zmíněný JavaScript z příkladu 1, můžeme obrázek nahradit pomocí CSS nastavením vlastnosti `background-image`, jak je zobrazeno na příkladu 7[13]

```
1 .obrazek {
2   background-image: url('obrazek.svg');
3 }
4 .no-svg .obrazek {
5   background-image: url('obrazek.png');
6 }
```

Příklad 7: Náhrada SVG jako obrázek na pozadí

Rozšiřujícím řešení je použití vícenásobných pozadí a gradientů. Pokud v CSS prohlížeč detekuje chybu v kódu, tuto hodnotu ignoruje a použije hodnotu deklarovanou dříve. Cílem tedy je uvést do souboru CSS deklarace, které jsou podporovány všemi prohlížeči.[15]

```
1 background: url(pozadi_alternativa.png);
2   background: url(pozadi.svg),
3         linear-gradient(transparent, transparent);
```

Příklad 8: Náhrada SVG jako obrázek na pozadí s použitím překrývání

Tehdy, když prohlížeč podporuje vícenásobné pozadí a gradienty, podporuje i SVG. Pokud nepodporuje, použije dříve deklarovanou PNG alternativu. Použití je ukázáno na příkladu 8.[15]

Alternativa SVG pro inline `<svg>` Na SVG odkazujeme jako na soubor nebo je grafika zakreslena přímo pomocí jazyka. Jelikož je inline SVG v DOM,

můžeme s ním interagovat s pomocí CSS nebo JavaScriptu. K tomu, abychom detekovali podporu prohlížeče, můžeme použít Modernizr[10]. Ukázka použití je zobrazena na příkladu 9.[15]

```
1 define(['Modernizr', 'createElement'], function(
    Modernizr, createElement) {
2   Modernizr.addTest('inlinesvg', function() {
3     var div = createElement('div');
4     div.innerHTML = '<svg/>';
5     return (typeof SVGRect !== 'undefined' && div.
        firstChild && div.firstChild.namespaceURI) ===
        'http://www.w3.org/2000/svg';
6   });
```

Příklad 9: Detekce podpory inline SVG pomocí Modernizr

Dále můžeme SVG zaměnit za použití prostého textu. Protože u `<svg>` elementu musí být text vnořen do elementu `<text>`, když prohlížeč bude SVG podporovat, prostý text bude ignorovat. Pokud prohlížeč SVG podporovat nebude zobrazí pouze prostý text, jak je ukázáno na příkladu 10.[15]

```
1 <svg>
2 prosty text
3 <text>text, ktery se u SVG zobrazi</text>
4  <circle fill="red" r="20" />
5 </svg>
```

Příklad 10: Náhráda SVG pro inline `<svg>` se zobrazením textu

[10] https://modernizr.com

Další variantou je záměna za HTML obsah. Můžeme použít HTML značku `<desc>` (description), která slouží k popsání obsahu čtečkám pro nevidomé. Proto můžeme použít textový obsah v alternativní podobě a tak bude sloužit i pro prohlížeče bez podpory SVG. Aby byl `<desc>` rozpoznán, měl by být umístěn v `<svg>` na začátku. Do `<desc>` můžeme také zahrnout `<img>` náhradu, která poskytne alternativní obrázek. Nevýhodou je, že ho stáhnou všechny prohlížeče včetně těch, které by obsah SVG mohly vykreslit.[13]

Pro inline `<svg>` je možné použít náhradu pomocí obrázku na pozadí. Pokud prohlížeč nebude SVG podporovat (to můžeme zajistit již zmíněným JavaScriptem s přidáním `.no-svg` z příkladu 1), zamění grafiku třídy v CSS za pozadí.[15]

Další možnost náhrady v inline `<svg>` je použití značky `<image>`, u které není za potřebí JavaScript. SVG `<image>` je možné použít v inline `<svg>`, ačkoliv uvnitř HTML ho prohlížeče mají za nestandardní synonymum `<img>`. V SVG specifikujeme URL pro soubory pomocí atributu `xlink:href`, ale v HTML pomocí atributu `src`. Použitím elementu `<image>` s atributem `src` bude mít za následek, že prohlížeče, které SVG nepodporují, náhradu stáhnou, ale ty, které SVG podporují ji nestáhnou. Může se stát, že by některý prohlížeč, který SVG podporuje, mohl náhradu stáhnout. Proto elementu nastavíme prázdný atribut `xlink:href`. Ke správné funkčnosti musíme vhodně zajistit atributy `width` a `height`. Výsledné řešení je ukázáno v příkladu 11.[15]

```
1 <svg width="96" height="96">
2 <image src="nahrada.png" xlink:href="" width="96"
    height="96" />
3 </svg>
```

Příklad 11: Náhrada SVG pro inline `<svg>` s použitím `<image>`

Také můžeme použít JavaScript knihovny.

- SVG for Everybody[11] s použitím `<use>` a definicí náhradního souboru.
- Plugin Grunt s JavaScript knihovnu Grunticon[12], která obstará celý průběh pro práci s SVG, včetně generování náhrady. Pro užití s inline `<svg>` JavaScriptová část Grunticonu vloží grafiku do DOM a Grunt plugin vytvoří náhrady a přegeneruje CSS.

Alternativa pro ikony Alternativu SVG ikon můžeme zajistit pomocí náhrady jako obrázek na pozadí, ale s vytvořením CSS ikon. Varianta je zobrazena na příkladu 12. Nahrazení lze také zajistit s knihovnou Grunticon nebo za pomoci ikon vytvořených z písma, `@fontface`. Pro funkčnost těchto metod je většinou zapotřebí JavaScript.[15]

```
1  <svg class="icon">
2    <a class="svg-status"></a>
3    <use xlink:href="#svg-status" />
4  </svg>
5  <style>
6  .icon a, .icon + a {
7    /* styly pro vsechny ikony*/
8  }
9  .icon .svg-status,
10 .icon + .svg-status {
11   /* styly pro konkretni ikonu */
12 }
13 </style>
```

Příklad 12: Náhrada s vytvořením CSS ikon

[11] https://github.com/jonathantneal/svg4everybody
[12] http://www.grunticon.com

5 Implementace SVG do webové stránky

Pro implementaci SVG do webové stránky existuje řada možností. SVG můžeme přidat pomocí HTML5 tagu `<svg>`, běžně používané techniky pro vkládání obrázků `<img>`, vkládání pomocí tagů `<object>` a `<embed>`, použitím `<iframe>` nebo jako obrázek na pozadí CSS.[16]

Aby bylo možné správně vytvořit responzivní SVG, měli bychom odstranit atributy výšky a šířky. Pokud necháme pevnou výšku nebo šířku nebude SVG správně reagovat. Atribut `viewBox` bychom měli ponechat.[16]

5.1 Vložení pomocí `<img>`

Do webové stránky lze SVG vložit pomocí tagu `<img>`. Když je SVG vloženo pomocí `<img>`, výška a šířka je zadaná v tagu a v prohlížeči se tak z SVG vytváří výřez. Obsah SVG je tak umístěn ve výřezu v závislosti na `viewBoxu`, který je specifikován přímo v `<svg>`.[17]

```
1 <img src="obrazek.svg" alt="obrazek_img">
```

Příklad 13: Vložení SVG pomocí `<img>`

SVG vždy vyplní šířku nadřazeného prvku v DOM tak, aby byl zachován poměr stran obrázku. Naopak `img` obrázky se zobrazují ve výchozí velikosti a nadřazený prvek vyplní pouze pokud v CSS explicitně nastavíme šířku na 100 %.[16]

5.2 Vložení pomocí `<object>`

Použití elementu `<object>` je nejvíce flexibilní způsob, jak přidat SVG do HTML dokumentu. SVG u `<object>` se chová podobně jako při vkládání pomocí `<img>`. V rámci elementu můžeme deklarovat náhradní grafiku, ačkoli bude u prohlížečů stažena, i když to nebude potřeba.[16]

```
1 <object data="obrazek.svg" type="image/svg+xml">
2         <img src="nahrada.png"> <!--nahrada-->
3 </object>
```

Příklad 14: Vložení SVG pomocí `<object>`

Stejně jako u tagu `<img>` prohlížeče předpokládají, že šířka objektu SVG bude 100% a výška bude zachovávat poměr stran. Nastavená šířka a výška u `<object>` bude výřezem pro vykreslení grafiky.[16]

5.3 Vložení pomocí `<picture>`

V responzivním webdesignu umožňuje element `<picture>` zobrazit různé varianty obrázků v závislosti na rozlišení. SVG obrázky jsou na rozlišení nezávislé a tak není potřeba více variant. Vložení pomocí `<picture>` můžeme použít v případě, kdy budeme chtít zobrazit jiné SVG obrázky, případně jen část SVG obrázku na různých typech zařízení. K zobrazení je vyžadován tag `<img>`, který zároveň slouží jako náhrada SVG.[18]

```
1 <picture>
2         <source media="(max-width: 640px)" srcset="
            maly-obrazek.svg" type="image/svg+xml">
3         <source media="(max-width: 1024px)" srcset="
            stredni-obrazek.svg" type="image/svg+xml">
4         <source srcset="cely-obrazek.svg" type="image
            /svg+xml">
5         <img src="nahrada.jpg" alt="nahrada">
6 </picture>
```

Příklad 15: Vložení SVG pomocí `<picture>`

5.4 Vložení pomocí `<embed>`

SVG lze také vložit pomocí `<embed>`, který je podobný jako `<object>`. Je určen k implementaci interaktivního obsahu a podporuje ho většina prohlížečů.[19]

```
1 <embed type="image/svg+xml" src="obrazek.svg" />
```

Příklad 16: Vložení SVG pomocí `<embed>`

5.5 Vložení pomocí `<iframe>`

Vzhledem k tomu, že prohlížeče mohou samostatně vytvářet dokumenty SVG, je možné je načíst v rámci `<iframe>`. Tato možnost vkládání SVG je podobná jako `<object>`.[16]

```
1 <iframe src = "obrazek.svg">
2         <img src="nahrada.png"> <!--nahrada-->
3 </ iframe>
```

Příklad 17: Vložení SVG pomocí `<iframe>`

Prohlížeče, které `<iframe>` zpracovávají odkazují na SVG bez výšky a šířky. Standardní výchozí hodnota šířky je `300px` a výšky `150px`. Tyto hodnoty prohlížeče pevně nastaví. Pokud nastavíme šířku na 100 %, výška zůstane beze změny a tím vznikají prázdná místa. Způsob, jak změnit výšku, je nastavením výřezu výšky a šířky prvku `<iframe>` ke stejnému poměru stran, jako má SVG obrázek. Poměr stran SVG plátna je výška a šířka `viewBoxu`.

V CSS je možné specificky nastavit poměr stran pro `<iframe>` pomocí techniky „Padding Hack“.[16]

5.5.1 Padding Hack – `<iframe>`

Prvním krokem je zbavit se `width` a `height` z prvku `<svg>` a ujistit se, že jsme nastavili `viewBox`, který potřebujeme.[16]

Aby Padding Hack fungoval, musíme ho obalit kontejnerem. Použijeme obecný div s `<iframe>` odkazujícím na SVG, který bude zmenšen tak, aby se do kontejneru vešel.[16]

Vzhledem k tomu, že se vnitřní okraj prvku dopočítává pomocí šířky. V kontejneru vynutíme šířku na 0. Pro šířku nebo výšku obvykle nastavujeme procentuální hodnotu. Pro `padding` použijeme hodnotu podle vzorce *(height : width)* · *procentuální hodnota šířky* = *padding-top*. Pozici nastavíme na relativní a prvku `<iframe>` musíme nastavit absolutní pozicování, šířku a výšku.[16]

5.6 Vložení jako obrázek na pozadí

SVG můžeme vložit jako obrázek na pozadí v CSS. SVG jako obrázek na pozadí se chová stejně jako při vkládání rastrových obrázků.[16]

```
1 .element {
2     background-image: url(obrazek.svg);}
```

Příklad 18: Vložení SVG jako obrázek na pozadí v CSS

5.7 Vložení pomocí inline `<svg>`

Do teď jsme do webové stránky vkládali SVG tak, že jsme připojili samostatný soubor, který obsahoval SVG XML. U inline `<svg>` vkládáme XML kód přímo do HTML stránky. Díky tomu, že je obrázek přímo v dokumentu, nepotřebujeme další žádost HTTP. Výhodou je možnost přístupu přes DOM a možnost úprav pomocí CSS a JavaScriptu. Nevýhodou může být rozsáhlý kód v dokumentu a neschopnost ukládání do mezipaměti, jako je to u rastrových obrázků

nebo při vkládání jinými metodami.[4]

```
1 <svg viewBox="0 0 100 100">
2     <rect width="50" height="50"/>
3 </svg>
```

Příklad 19: Vložení SVG pomocí inline `<svg>`

5.7.1 Struktura SVG dokumentu

Všechny podrobnosti SVG se nacházejí uvnitř elementu `<svg>`. Tento prvek obsahuje potřebné atributy ke správné vykreslení grafiky. Atributy jsou nezbytné a vynecháním můžeme způsobit problémy jak pro nás, tak pro načítání v prohlížečích. Pokud budeme vkládat grafiku ručně nebo pomocí elementů, vždy musíme zajistit správnou organizaci a strukturu SVG kódu, která je klíčová především pro správné uspořádání a funkčnost grafických prvků.[20]

Element svg Hlavním prvkem SVG je element `<svg>`, uvnitř kterého se nacházejí veškeré potřebné elementy. Je klasifikován jako kontejner a tvoří konstrukční prvek pro SVG v dokumentu s vlastním souřadnicovým systémem. Nejduležitější atributy používané v rámci elementu jsou `width`, `height`, `preserveAspectRatio` a `viewBox`. Tyto atributy definují plátno pro grafické prvky. Pokud budeme SVG získávat za pomoci grafických editorů, budou v rámci elementu použity i další atributy jako číslo verze SVG, `DOCTYPE`, `xmlns` apod.[20]

Element g Prvek `<g>` slouží k seskupení souvisejících grafických prvků dohromady. Díky tomu můžeme manipulovat jak s celkem, tak s jednotlivými částmi grafiky a to se hodí zejména při animaci. U prvků, které nejsou obsaženy v prvku `<g>` se předpokládá, že jsou samostatná skupina.[20]

Element use Prvek `<use>` umožnuje uvnitř SVG duplikovat element na jiné místo s použitím odlišných vlastností. Pokud používáme opakující se symboly jako jsou např. ikony na které můžeme odkazovat a tím minimalizujeme kód. Pro použití stačí, když existuje prvek `<g>` s atributem `ID`, který můžeme zavolat s pomocí `xlink:href`. Prvku můžeme přidat atributy jako například `x`, `y`, `width`, `height`, `fill` apod.[20]

Element defs V rámci prvku `<defs>` je definována grafika, která není vykreslena, ale je odkazována a vykreslena za pomoci `xlink:href`. S atributem ID může být použita v celém dokumentu. Definovat můžeme buď úplnou grafiku nebo jen atributy.[20]

```
1 <svg>
2         <defs>
3         <linearGradient id="prechod">
4             <stop offset="0%" stop-color="red" />
5             <stop offset="100%" stop-color="blue" />
6         </linearGradient>
7     </defs>
8     <rect x="0" y="0" width="100" height="100" fill=
        "url(#prechod)" />
9 </svg>
```

Příklad 20: Definování v rámci prvku `<defs>`

Element symbol Prvek `<symbol>` je podobný prvku `<g>`, protože slouží k seskupení grafických prvků, avšak prvky v rámci `<symbol>` nemají vizuální výstup (stejně jako `<defs>`) a objeví se až po odkazování za pomoci prvku `<use>`. Na rozdíl od prvku `<g>` má `<symbol>` vlastní souřadnicový systém oddělen od výřezu, do kterého je grafika vykreslena.[20]

Pořadí elementů Pořadí elementů v SVG nelze ovlivnit v CSS s pomocí `z-index`, jako je to u elementů HTML, ale zcela závisí na umístění. Objekt v `<svg>` elementu, který je v kódu umístěn níže, se zobrazí v popředí a naopak.[20]

5.7.2 Tvary a cesty

Často při vytváření SVG využíváme grafický software, ve kterém můžeme vytvořit složité grafické prvky a poté kompletní grafiku implementovat. SVG obsahuje několik základních tvarů, jako jsou obdélníky, kruhy, elipsy, čáry, křivky a mnohoúhelníky. Všechny uzavřené tvary se dají vyplnit pomocí `fill` a všechny tahy nastavit pomocí `stroke`. Každý tvar má své specifické vlastnosti, které si představíme. [20]

Obdélník Obdélník definujeme pomocí tagu `<rect>`. Velikost obdélníku určují atributy `width` a `height` a výplň nastavíme pomocí `fill`. Pokud bychom atribut `fill` neurčili, obdélník bude výchozí černé barvy. Souřadnice, které mohou být zahrnuty, určíme pomocí `x` a `y`. Tyto hodnoty určují umístění podél příslušné osy dle rozměrů nastavených v `<svg>`. U obdélníku také můžeme nastavit zaoblené rohy a to pomocí zadání atributů `rx` a `ry`, kdy `rx` určuje horizontální poloměr strany rohů a `ry` vertikální.[20]

```
1 <svg>
2     <rect width="200" height="100" fill="red" />
3 </svg>
```

Příklad 21: Vykreslení obdélníku

Kruh Kruh definujeme pomocí tagu `<circle>` na základě středového bodu a vnějšího poloměru. Souřadnice `cx` a `cy` určují umístění středu a atribut `r` určuje velikost vnějšího poloměru.[20]

```
1 <svg>
2     <circle cx="75" cy="75" r="75" fill="red" />
3 </svg>
```

Příklad 22: Vykreslení kruhu

Elipsa Elipsu definujeme pomocí tagu `<ellipse>` na základě středového bodu a dvou poloměrů. Zatímco souřadnice `cx` a `cy` určují umístění středu elipsy, hodnoty `rx` a `ry` definují poloměry stran tvaru.[20]

```
1 <svg>
2     <ellipse cx="100" cy="100" rx="100" ry="50" fill=
        "red" />
3 </svg>
```

Příklad 23: Vykreslení elipsy

Čára Čáru se začátkem a koncem definujeme pomocí tagu `<line>`, kdy atributy `x1` a `y1` určují začátek a hodnoty `x2` a `y2` konec čáry. Tloušťku čáry určíme pomocí atributu `stroke-width`.[20]

```
1 <svg>
2     <line x1="5" y1="5" x2="100" y2="100" stroke="red
        " stroke-width="5"/>
3 </svg>
```

Příklad 24: Vykreslení čáry

Lomená čára Tag `<polyline>` definuje několik vzájemně propojených bodů s neuzavřeným počátečním a konečným bodem. Tento tvar nazýváme lomenou čárou a pomocí hodnot v atributu `points` určujeme umístění bodů na osách `x` a `y`, které jsou seskupeny v seznamu hodnot jako `x`, `y`. Vlastnosti lomené čáry určujeme stejnými atributy jako u ostatních tvarů.[20]

```
1 <svg>
2     <polyline points="0,40 40,40 40,80 80,80 80,120
        120,120 120,160" fill="white" stroke="red"
        stroke-width="5" />
3 </svg>
```

Příklad 25: Vykreslení lomené čáry

Mnohoúhelník Uzavřený tvar ze vzájemně propojených bodů definujeme tagem `<polygon>`, kdy jsou body určeny v seskupeném seznamu hodnot za pomoci `x`, `y`. Tímto prvkem můžeme vytvářet různé tvary v závislosti na počtu definovaných bodů.[20]

```
1 <svg>
2     <polygon points="50,5 100,5 125,30 125,80 100,105
        50,105 25,80 25,30" fill="red" />
3 </svg>
```

Příklad 26: Vykreslení mnohoúhelníku

Cesta Cesta u SVG představuje obrys tvaru, který se skládá z mnoha křivek. Cestu definujeme pomocí tagu `<path>`, který obsahuje atribut `d`. V tomto atributu je definován obrys tvaru. Atribut `d` obsahuje instrukce jako jsou `moveto`, `line`, `curve`, `arc` a `closepath`.[20]

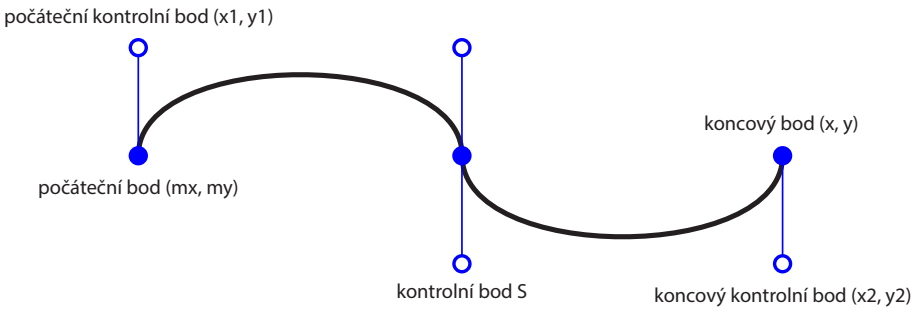
moveto K vytvoření nového počátečního bodu se využívá příkaz `M` nebo `m`. Používá se vždy, když chceme začít kreslit na novém místě. Příkazy `moveto`, které navazují, představují začátek cesty a vytváří složenou cestu. S použitím malého písmena `m` se označují relativní souřadnice, zatímco s použitím velkého písmena `M` označujeme absolutní souřadnice, které budou navzájem navazovat.[20]

closepath K uzavření cesty se používá příkaz `Z` nebo `z`. Pokud použijeme tento příkaz, cesta se v aktuálním bodě ukončí a vytvoří čáru k bodu počátečnímu. Pokud hned za příkazem `closepath` následuje příkaz `moveto`, tak reprezentuje začátek nové složené cesty, ale pokud následuje jiný příkaz, cesta začíná ze stejného bodu jako uzavřená cesta. Při použití velkého `Z` nebo malého `z` je výsledek stejný.[20]

lineto Příkazy `lineto` nakreslí úsečku z aktuálního bodu do nového bodu. Při použití `L` a `l` se vykreslí úsečka z aktuálního bodu do následujícího bodu s novými souřadnicemi. Příkazy `H` a `h` vykreslí vodorovnou úsečku z aktuálního bodu a příkazy `V` a `v` úsečku vertikální. Velká písmena signalizují návaznost s absolutní polohou a malá písmena pozici relativní.[20]

Křivky K vykreslení křivky můžeme použít tři typy křivek. Jsou jimi Cubic Bézier (C, c, S, s), Quadratic Bézier (Q, q, T, t) a Elliptical arc (A, a).[20]

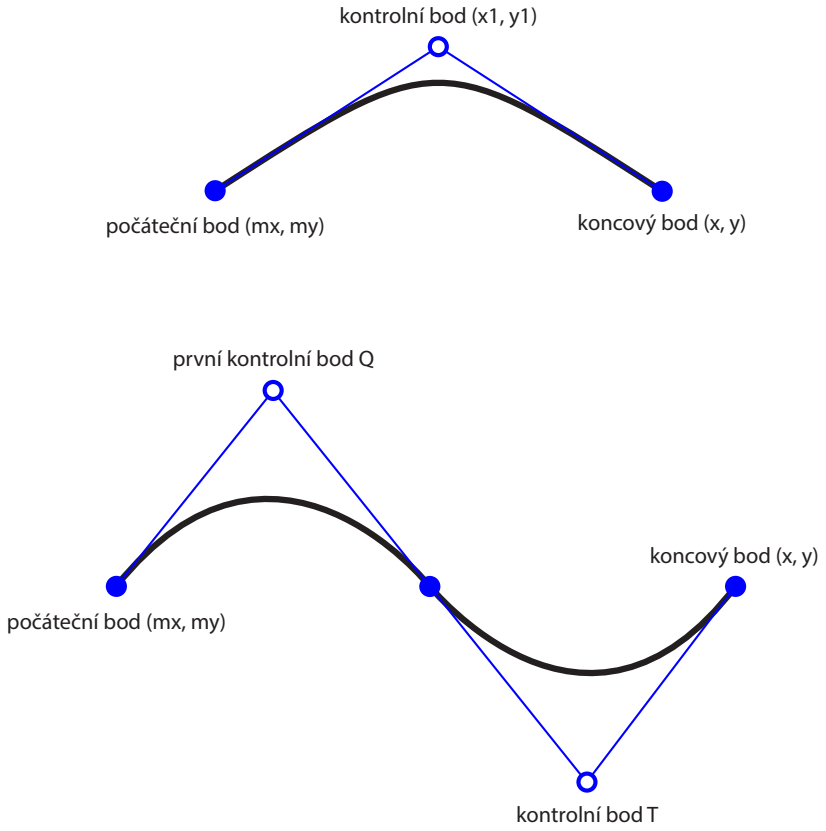
Cubic Bézier Příkazy C a c vykreslí křivku z aktuálního bodu a pomocí řídících parametrů x1, y1 nad tímto počátečním kotevním bodem získáme kontrolu. Parametry x2, y2 jsou řídícími parametry pro koncový kotevní bod. Manipulace prvních a posledních hodnot určuje počáteční a koncovou polohu, zatímco manipulace středových hodnot ovlivňuje tvar křivky. Přidáním parametrů S a s, se také vykreslí Cubic Bézier, ale s tím, že první kontrolní bod je odrazem posledního řídícího bodu z předchozího příkazu C. Tento odraz je k počátečnímu bodu příkazu S relativní. Velká písmena C signalizují, že bude následovat absolutní pozice, u malého písmena c bude pozice relativní. Stejně tak je to u písmen S a s. Význam bodů je znázorněn na obrázku 3.[20]



Obrázek 3: Vykreslení křivky Cubic Bézier

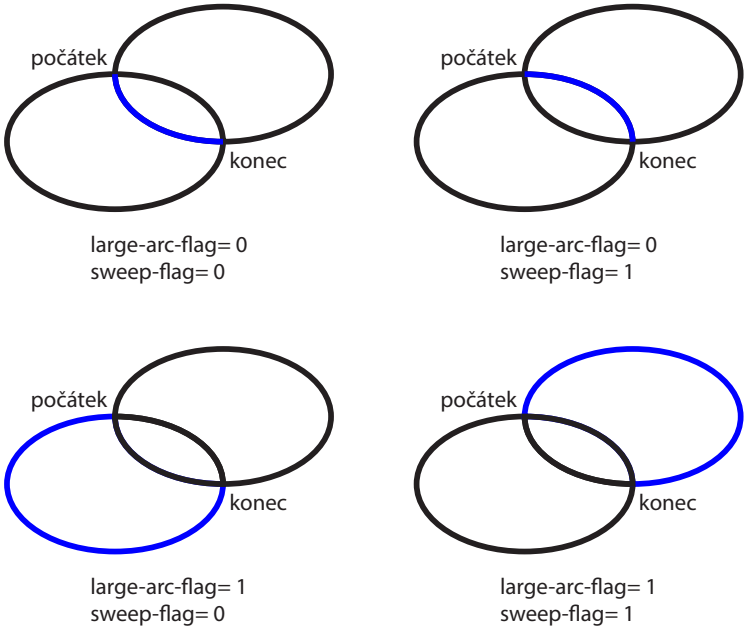
Quadratic Bézier Quadratic Bézier (Q, q, T, t) je podobná jako Cubic Bézier s tím rozdílem, že má pouze jeden kotevní bod. První a poslední hodnota určuje počáteční a koncovou polohu. Středová hodnota Q definuje řídící bod křivky a určuje její tvar. Q a q vykreslí křivku z počátečního bodu do koncového pomocí kotevních kontrolních bodů x1, x2. T a t vykreslí křivku za

předpokladu, že bude odrazem z předchozího kotevního bodu, vzhledem k novému počátečnímu bodu příkazů T nebo t. Velká písmena signalizují absolutní polohu, malá písmena polohu relativní. Znázorněno na obrázku 4.[20]



Obrázek 4: Vykreslení křivky Quadratic Bézier

Elliptical Arc Eliptical Arc definuje část elipsy za pomoci příkazu A nebo a. Tyto příkazy vykreslují oblouk určený počátečním a koncovým bodem, rotací, směrem a poloměry x, y. První a poslední sada hodnot určuje počáteční a koncové souřadnice. Druhá sada hodnot určuje poloměry. Oblouk lze vykreslit čtyřmi různými možnostmi za pomocí hodnot 1 a 0 (large-arc-flag a sweep-flag). Možnosti vykreslení je ukázáno na obrázku 5.[20]



Obrázek 5: Vykreslení Elliptical Arc

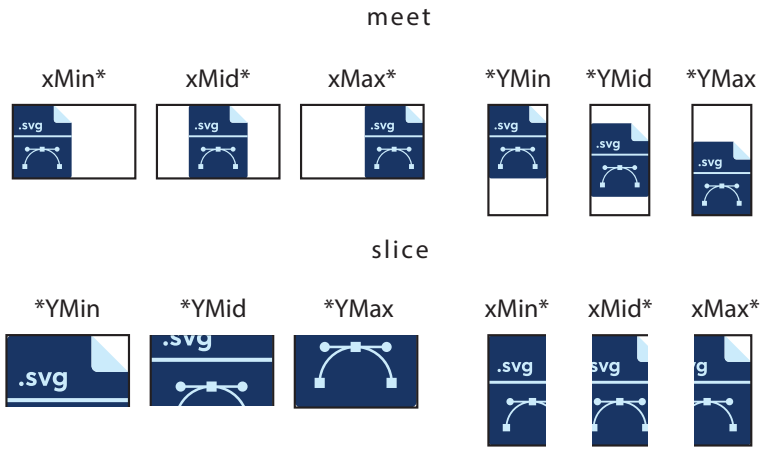
5.8 SVG viewport a viewBox

Pro práci s velikostí SVG, používáme viewport a viewBox.[10]

viewport Atribut `viewport` (výřez) je výška a šířka SVG elementu. Definuje viditelnou část SVG obrázku. Často je nastavena stejně jako atributy `width` a `height` přímo uvnitř `<svg>` nebo přes CSS. Přestože může mít SVG dokument různou šířku a výšku, bude se zobrazovat jen určitá část obrázku. Pokud nastavíme atributy `width` a `height` uvnitř `<svg>` elementu a nepoužijeme atribut `viewBox`, zachová se stejně, jako kdybychom pracovali s CSS.[10]

viewBox Atribut `viewBox` určuje souřadný systém a poměr stran. Má čtyři parametry, `min-x`, `min-y`, `width` a `height`. Parametry `min-x` a `min-y` definují, v jakém místě by měl viewBox začínat (levý horní roh), zatímco `width` a `height` určují výšku a šířku viewBoxu.[10]

preserveAspectRatio Pokud `viewport` a `viewBox` nemají stejný poměr stran, atribut `preserveAspectRatio` nasměruje prohlížeč jak má obrázek zobrazit. Určujeme u něj zarovnání a poměr stran. Hodnoty x a y za kterými následuje Min, Mid nebo Max určí zarovnání obrázku. Výchozí hodnota je xMidYMid. Poměr stran je volitelný a můžeme u něj použít tři hodnoty: `meet`, `slice` nebo `none`. Zarovnání je znázorněno na obrázku 6.[10]



Obrázek 6: Možnosti zarovnání obrázku s atributem `preserveAspectRatio`

5.9 Transformace u SVG

Transformace v SVG umožňuje manipulaci s SVG prvky různými způsoby za pomoci atributů `translate`, `scale`, `rotate` a `skew`. Při použití transformace nemusíme zasahovat do kódu SVG, což je důležité pro dynamickou práci s SVG.[21]

translate Transformace `translate` zajistí posunutí SVG elementu dle zadaných souřadnic `x` a `y` z počátečních souřadnic prvku. Argument `y` je volitelný a pokud ho neuvedeme, je ekvivalentní k `x`.[21]

scale Transformace `scale` mění velikost SVG elementu dle zadaných `x` a `y`. Hodnoty určují relativní zvětšení ve směru os. Stejně jako u `translate` je `y` volitelné a pokud ho neuvedeme, je ekvivalentní k `x`.[21]

rotate Transformace `rotate` otočí prvek o zadaný počet stupňů. Tato transformace má tři argumenty. První argument určuje počet stupňů, o kolik se má prvek otočit. Druhým a třetím argumentem jsou souřadnice `x` a `y`, který definují počáteční bod rotace. Pokud pro prvek nezadáme počáteční bod, použije se počáteční bod výřezu.[21]

skew Transformace `skew` zkosí prvek o úhel podél zadané osy. Stejně jako u `rotate` a `scale`, je `skew` založeno na počátečním bodu. Podle zadání atributu `skewX` a `skewY` se určí podél jaké osy se má SVG prvek zkosit.[21]

6 Manipulace s SVG

S SVG obrázky můžeme manipulovat na za pomoci CSS nebo JavaScriptu.

6.1 SVG manipulace použitím CSS

Zkratka z Cascading Style Sheets (česky kaskádové styly) jsou obecně používány k formátování vizuálního vzhledu HTML dokumentu. [22] CSS může být ke stylování prvků u SVG používán úplně stejně, jako při použití HTML dokumentu. SVG je s CSS vzájemně propojený. Podporuje deklarativní styl prvků, ale používá základní model rozvržení, který je na CSS zcela nezávislý. Při užívání SVG není nutné CSS použít, ale je to možnost, jak mít nad SVG kontrolu a definovat kompletní grafický výstup za pomoci atributů, které je později možné jednoduše měnit.[22]

CSS v SVG Existují čtyři možnosti, jak definovat CSS atributy v SVG elementu.[22]

Presentation attributes Většina stylů, které jsou užité v SVG, mohou být specifikovány jako XML atributy. Výsledek je obvykle stejný, jako když bychom použili CSS. Jen XML překladač rozlišuje velikost písmen a hodnoty, které by byly nastaveny za pomoci CSS, mají přednost.[22]

Inline styles Stejně jako u HTML elementů, mohou být styly nastaveny přímo uvnitř prvků.[23] Například vybarvení a obrys u čtverce je zobrazeno na příkladu 27.[21]

```
1 <rect style="fill: red; stroke: blue;" />
```

Příklad 27: Nastavení stylů uvnitř prvku

Internal stylesheet Pokud nechceme mít styly přímo v SVG, můžeme je také umístit do těla HTML dokumentu.[23]

External stylesheets Styly také mohou být umístěny v externím CSS souboru, díky němuž můžeme k SVG přistupovat z více dokumentů. Máme čtyři možnosti, jak externí styly připojit.[22]

- Na začátku jiného CSS souboru.
  `<style type="text/css"> @import "style.css"</style>`
- Těsně před otevřeným `<svg>` tagem.
  `<?xml - stylesheet href="style.css"type="text/css"?>`
- V hlavičce HTML dokumentu pomocí elemetnu `<link>`.
  `<link href="style.css"rel="stylesheet"type="text/css">`
- Uvnitř SVG souboru s použitím HTML `<link>` elementu.
  `<html:link href="style.css"rel="stylesheet"type="text/css">`

Selektory Většinu selektorů, které jsou užívány v CSS, můžeme použít i pro SVG elementy. Kromě obecného výběru pomocí tříd a ID mohou být na SVG prvky přidány vlastnosti dynamických pseudotříd (`:hover`, `:active`, `:focus`) a pseudotříd (`:first-child`, `:visited`, `:link`, `:lang`). Zbývající pseudotřídy (např. `::before` a `::after`) nejsou součástí jazyka SVG, takže nemají na prvky vliv.[23]

Kaskáda stylů Může se stát, že budeme mít SVG nastylováno pomocí více stylů. V tomto případě se pořadí přepisuje podle obrázku 7. Styly, které jsou uvedeny níže, přepíší všechny styly uvedené výše.[23]

Obrázek 7: Pořadí kaskádových stylů u SVG

SVG v CSS Existují dva způsoby, jak můžeme odkazovat na SVG v rámci CSS souboru.

- Použitím kompletního SVG souboru jako obrázku.
- Použitím SVG prvku, na který aplikujeme grafické efekty.

Kompletní SVG obrázek může být použitý jako jakýkoliv jiný obrázek v CSS (`background-image`). Odkazování na SVG elementy pro stylování ostatních prvků, které obsahují atributy jako `fill`, `stroke`, `filter`, `mask`, `clip-path`, se mohou použít v obsahu CSS souboru.[22]

6.2 SVG manipulace použitím JavaScriptu

SVG, stejně jako HTML, je reprezentováno pomocí rozhraní DOM (Document Object Model), které umožňuje manipulaci, aktualizaci, vytváření a mazání elementů, vlastností a obsahu založeném na XML dokumentech. [1] Díky tomu máme přístup k jednotlivým objektům XML, a tak s nimi můžeme manipulovat

za pomoci JavaSriptu. Přistupovat můžeme buď k vnitřnímu nebo externímu SVG. Elementy, se kterými budeme pracovat, musí mít jedinečný identifikátor a můžeme k nim přistupovat s vnitřním nebo externím JavaScriptem.[24]

Vnitřní SVG K získání inline SVG elementu lze použít `document.getElementById()`. Script lze umístit uvnitř nebo mimo tag `<svg>`.[24]

```
1 <script type="text/javascript">
2 var svg = document.getElementById('inline-1');
3 svg.setAttribute("fill", "red");
4 </script>
```

Příklad 28: Získaní inline SVG s metodou `document.getElementById()`

Externí SVG a vnitřní JavaScript Pro externí SVG můžeme použít stejný kód, jako když přidáváme `<script>` do vnitřního SVG. Skript bychom měli obalit do `<![CDATA[...]]>`, z důvodu zajištění správné funkčnosti XML překladače.[24]

```
1 <script type="text/javascript"><![CDATA[
2 var svg = document.getElementById('external-1');
3 svg.setAttribute("fill", "red");
4 ]]></script>
```

Příklad 29: Získaní externího SVG s metodou `document.getElementById()`

Externí SVG a externí JavaScript V tomto případě nemůžeme k prvku SVG přistupovat přímo, protože je uvnitř objektu. Nejprve musíme získat ob-

jekt a poté přístup k jeho `contentDocument`. Po získání SVG dokumentu můžeme k prvkům přistupovat.[24]

```
1 <script type="text/javascript"> window.
    addEventListener("load", function() { var svgObject
     = document.getElementById('svg-object').
    contentDocument; var svg = svgObject.getElementById
    ('external-1'); svg.setAttribute("fill", "red"); })
    ;
2 </script>
```

Příklad 30: Získání externího SVG v externím JavaScriptu

K jednotlivým prvkům přistoupíme díky `ID`, za pomoci metody `getElementById()`. Tato metoda je ukázána na příkladu 31.

```
1 var element = document.getElementById('item');
```

Příklad 31: Přístup k prvku metodou `getElementById()`

Na příkladu 32 je případ použití s externí SVG, kdy musíme nejprve přistoupit k objektu.

```
1 {window.addEventListener("load", function() { var
    svgObject = document.getElementById('svg-object').
    contentDocument; var element = svgObject.
    getElementById('item'); });}
```

Příklad 32: Přístup k prvku s externím SVG v externím JavaScriptu

7 Práce s SVG textem

Práce s textem v SVG se provádí za pomoci elementu `<text>`. Text, který chceme zobrazit musí být umístěn mezi tagy `<text>` a `</text>` uvnitř elementu `<svg>`. Na `<text>` lze také aplikovat přechody, výplně, výřezy, masky nebo filtry. Prvek `<text>` vytváří text jako grafiku, kterou můžeme snadno upravit přímo v kódu. [2] S textem se v SVG zachází obdobně jako s tvary, takže viditelnost textu určuje `viewport`. Text SVG nefunguje jako HTML text, který se zalomí podle šířky nadřazeného kontejneru. V SVG je každý element nezávislý a je umístěn přímo tam, kam ho umístíme, i když překrývá jiný obsah.[22]

Prostřednictvím atributů uvnitř elementu `<text>` nastavíme, jak bude text ve výsledku vypadat. Používáme atributy `x`, `y`, `dx`, `dy`. Hodnota `x` určuje, kde začne text na ose x. Hodnota `y` určuje umístění spodní části textu na ose y. Zatímco `x` a `y` určuje souřadnice v absolutním prostoru, `dx` a `dy` stanovuje relativní souřadnice.[20]

```
1 <svg width="600" height="100">
2     <text x="30" y="90" fill="red" font-size="50" >
        SVG text</text>
3 </svg>
```

Příklad 33: Vložení SVG textu

Atribut `textLength` a `lengthAdjust`

`textLength` Délku textu určuje atribut `textLength`. Hodnota v atributu určí délku celého textu a délku zajistí tím, že mezi jednotlivé znaky nastaví mezeru.

`lengthAdjust` Přidáním atributu `lengthAdjust` můžeme specifikovat rozteč a velikost znaků, které se přizpůsobí hodnotám délky.[20] Specifikovat jej můžeme dvěma možnými hodnotami. Hodnota `spacing` zachová velikost znaků a délka textu se upraví pomocí mezer mezi znaky. U hodnoty `"spacingAndGlyphs"` je celý text roztažen nezávisle na proporci znaků. [25]

```
1 <svg width="600" height="100">
2     <text x="30" y="90" fill="red" font-size="50"
        textLength="500" >SVG text</text>
3 </svg>
```

Příklad 34: Specifikace délky SVG textu

Atribut `rotate` Atribut `rotate` zajišťuje manipulaci jednotlivých znaků textu. Jedna hodnota v atributu vede k tomu, že každý znak rotuje dle stejné hodnoty. Řetězec hodnot nastaví odlišné rotace jednotlivých znaků. Pokud neuvedeme dostatečný počet hodnot, které odpovídají počtu znaků, poslední hodnota nastaví rotaci pro zbývající znaky. Pokud bychom chtěli nastavit rotaci pro celý textový element, použijeme atribut `transform`.[20]

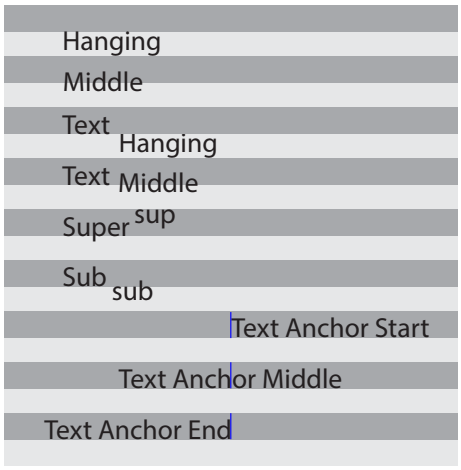
```
1 <svg width="600" height="300">
2     <text x="30" y="90" fill="red" font-size="50"
        rotate="20,0,5,30,10,50,5,10,65,5" transform="
        rotate(8)" >SVG text</text>
3 </svg>
```

Příklad 35: Rotace znaků v SVG textu

Prvek `tspan` SVG text nepodporuje automatické zalamování řádku a tak se k zalamování používá `<tspan>`. Tím, že oddělíme určitá slova, můžeme pomocí `<tspan>` manuálně vytvořit víceřádkový text. Nemusíme definovat nový souřadnicový systém, jen další řádky `<tspan>` umístíme do elementu `<text>` a řádek převezme parametry z řádku předchozího. Pomocí hodnoty `dy` se řádek umístí podél osy `y` ve vztahu s předchozím řádkem. Každému znaku můžeme nastavit vlastní pozici, když do `dy` uvedeme seznam hodnot (`dy="-30 30 60"`).[20]

Vlastnosti textu U textu můžeme nastavovat základní textové vlastnosti jako `font-family`, `font-size`, `font-weight`, `font-style` a také vlastnosti `fill` a `stroke`.[21]

Zarovnání textu Vlastnosti, které nastavují vertikální zarovnání textu, jsou `dominant-baseline`, `alignment-baseline`, `baseline-shift`. Atribut `dominant-baseline` se používá k nastavení základní hodnoty. Atribut `alignment-baseline` určuje, jak je objekt zarovnán vzhledem k jeho nadřazenému objektu. Atribut `baseline-shift` umožňuje dolní a horní index přesunem výchozího prvku nahoru nebo dolu. Atribut `text-anchor` se používá pro zarovnání horizontálního textu vzhledem k danému bodu. Obsahuje počáteční (`start`), střední (`middle`) a koncové (`end`) zarovnání. Vlastnosti zarovnání jsou zobrazeny na obrázku 8. [21]



Obrázek 8: Vertikální a horizontální zarovnání SVG textu

Vlastnosti `kerning` a `letter-spacing` Kerning umožnuje nastavení rozestupů mezi znaky na základě typografických pravidel nebo na základě vlastní zadané hodnoty. Hodnota auto udává, že mezery mezi znaky určí tabulka z písma. Vlastní hodnotu určíme pomocí číselné hodnoty (`kerning="30"`). Také můžeme použít `inherit` hodnotu, která dědí hodnotu od rodičovského elementu. Do `letter-spacing` můžeme uvést hodnoty `normal`, `length` nebo `inherit`. Pokud uvedeme číselnou hodnotu, výsledek bude stejný jako u atributu `kerning`. Vlastnost je určena jako doplněk k vlastnosti `kerning` zajišťující mezery mezi jednotlivými znaky.[20]

Vlastnost `word-spacing` Hodnota ve vlastnosti `word-spacing` určuje mezery mezi slovy. Můžeme také použít hodnoty `auto`, která je hodnotou výchozí, anebo hodnotu `inherit`.[20]

Vlastnost `text-decoration` Vlastnost `text-decoration` umožnuje použít hodnotu pro podtržení `underline`, nadtržení `overline` a přeškrtnutí `line-through`. Hodnota `underline` a `overline` jsou vykreslené předtím, než je vykreslený text. U hodnoty `line-through` je to naopak, nejdříve se vykreslí text a poté se v popředí vykreslí vlastnost.[20]

Text pomocí křivky Text na křivku zrealizujeme pomoci elementu `<textPath>`, který umístíme do prvku `<text>`. Element `<textPath>` se odkáže pomocí `ID` na křivku (`<path>`), kterou umístíme do elementu `<defs>`. Pro odkazovani na křivku užijeme atribut `xlink:href`, který umístíme do elementu `<textPath>`.[20]

Pro nastavení začátku textu na křivce můžeme nastavit atribut `startOffset`, který představuje délku posunu v procentech. Hodnota "0 %" označuje počáteční bod křivky a hodnota "100 %" koncový bod. Pokud chceme křivku vykreslit, použijeme prvek `<use>`, u kterého nadefinujeme parametry křivky.[20]

7.1 Výplně SVG

Text můžeme vyplnit stejně tak jako základní tvary. Vnitřní barvu nastavíme za pomoci vlastnosti `fill`. Vlastnost `fill` přijímá platnou hodnotu barvu s možností nastavit průhlednost pomocí vlastnosti `fill-opacity` s hodnotou, jako je hodnota `alpha` v definici barev rgba. Hodnota "0" nastaví úplnou průhlednost, hodnota "1" neobsahuje průhlednost žádnou.

Do vlastnosti `fill` můžeme také uvést reference na elementy jako jsou `hatch`, `linearGradient`, `meshgradient`, `pattern`, `radialGradient` a `solidcolor`.

Barvu tahu nastavíme vlastností `stroke`, do které uvedeme hodnotu barvy. Kromě toho můžeme nastavit i tloušťku tahu vlastností `stroke-width`. Zobrazení tahu můžeme specifikovat vlastnostmi `stroke-dasharray`, `stroke-dashoffset`, `stroke-linecap` a `stroke-linejoin`.

Vlastnost `stroke-dasharray` nastavuje délku čar a mezer pro přerušovaný obrys. Liché hodnoty určují délku vykreslených čar, hodnoty sudé určují délku mezer.

K předchozí vlastnosti se vztahuje atribut `stroke-dashoffset` s kladnou nebo zápornou hodnotou. Tato vlastnost specifikuje vzdálenost posunu od za-

čátku vykreslení čáry.

Atribut `stroke-linecap` určuje, jakým způsobem má být vykreslen konečný bod čáry. Hodnoty jsou `butt`, `round`, `square` a `inherit`.

Poslední vlastnost `stroke-inerjoin` definuje zaoblení rohů. Možné hodnoty jsou `miter` (ostré hrany s pouze jedním úhlem), `round` (hladké zaoblení rohů) a `bevel` (přidá jeden nový úhel a vykreslí složený roh).[21]

Výplň pomocí `<clipppath>` Objekty lze vyplnit pomocí ořezové cesty. Ořezová cesta určuje oblast, ve které bude pozadí, případně barva, zobrazena. Pozadí, které leží mimo definovaný tvar, nebude zobrazeno a to, které se nachází uvnitř, zobrazeno bude. Ořezová cesta může mít několik hodnot, z nichž jednou je element `<text>`. Když nastavíme ořezovou cestu na textový prvek, pozadí se ořízne na tvar dle použitého textu. Tato cesta může být odkazována buď pomocí ID nebo pomocí CSS vlastnosti daného obrázku.[26]

Výplň pomocí `<pattern>` Element `<pattern>` umožňuje definovat opakující se malý grafický vzor složený z různě barevných tvarů a cest. [1] Vzor, případně více vzorů, definujeme v prvku `<defs>`.[20]

Výplň pomocí přechodů Existují dva typy přechodů, `linearGradient` a `radialGradnient`. Přechody definujeme v prvku `<defs>`, který umožňuje opakované použití. Tyto definice nemají vizuální výstup, dokud nejsou odkazovány pomoci jedinečného `ID` v rámci atributu `fill` nebo `stroke` uvnitř `<svg>`.[20]

Lineární přechod Lineární přechod mění barvu podél přímky. Je definován v elementu `<linearGradient>` pomocí bodů `<stop>`. Na každém z těchto bodů je barva se 100% sytostí. V mezeře mezi body se vytvoří přechod z jedné barvy na druhou. Existují dva druhy atributů, `offset` a `stop-color`.

Atribut `offset` přijímá buď číslo od 0 do 1 nebo procentuální hodnotu a informuje, v jakém bodě má přiřadit `stop-color`.

Do atributu `stop-color` uvedeme jakoukoliv platnou hodnotu barvy, případně může také obsahovat průhlednost.

Hodnoty atributů `x1`, `y1`, `x2`, `y2` představují počáteční a koncové body, na kterých se projeví změna barvy. Pokud v atributu `y` uvedeme hodnotu "100 %" a v atributu `x` hodnotu "0", výsledkem bude horizontální přechod. V případě prohození hodnot vznikne vertikální přechod. Jakákoliv jiná hodnota způsobí přechod v daném úhlu.

Atribut `gradientUnits` definuje souřadný systém pro hodnoty `x1`, `y1`, `x2`, `y2`. Zadat můžeme dvě hodnoty `"userSpaceOnUse"`, která nastaví systém souřadnic v přechodu v absolutních jednotkách, nebo `"objectBoundingBox"`, která vytvoří systém dle daného SVG tvaru. Tato hodnota je nastavena jako výchozí.

Hodnota v atributu `spreadMethod` určuje, jak se má přechod zachovat v případě rozšíření do nevyplněného prostoru v tvaru. Výchozí hodnotou je `pad`, která na zbytek prostoru rozloží první a poslední barvu.

Hodnota `repeat` zařídí, aby se vzor nepřetržitě opakoval. Poslední hodnota, `reflect` bude vzor střídavě zrcadlit v pořadí od začátku do konce a od konce na začátek.[20]

Radiální přechod Radiální přechod obsahuje podobné atributy jako lineární s výjimkou jiného souřadnicového systému. Atributy `cx`, `cy` definují středovou souřadnici a atribut `r` nastavuje poloměr přechodu. Atributy `fx` a `fy` představují souřadnice středového bodu.[20]

Výplň pomocí `<mask>` Maskování na rozdíl od ořezové cesty bere v úvahu atributy, které jsou z tvaru vyříznuty pryč. Lze použít průhledný, poloprůhledný nebo zcela plný rozsah výplně. [1]

Masku můžeme přidat do SVG `<text>` tím, že nejdříve definujeme do elementu `<defs>` masku a místo atributu `fill` na ní budeme odkazovat pomoci atributu `mask`. V SVG bude element ve výchozím nastavení vykreslen pouze

tam, kde se v masce nachází bílá barva, tedy naopak než CSS. Jakákoliv hodnota mezi bílou a černou barvou vytvoří průhlednost. Přiblížení hodnot k černé barvě zajistí průhlednější vykreslení v masce.[27]

Výplň pomocí filtrů Filtry SVG jsou definovány v elementu `filter` přímo uvnitř SVG na rozdíl od CSS filtrů, které lze použít pro libovolný HTML element prostřednictvím vlastnosti `filter`. Nutností je uvést `ID`, díky kterému můžeme SVG filtr aplikovat na text. Případně můžeme použít metody prostřednictvím CSS.[28]

8 Výřezy u SVG

Mnoho funkcí, které používáme v CSS, bylo odvozeno z SVG. Jednou z těch funkcí je ořezávání. CSS i SVG poskytuje výřez do vlastních tvarů. Grafická operace ořez umožňuje úplné nebo částečné skrytí části elementu. Oříznutým prvkem může být libovolný kontejner nebo grafický element. Části prvků, které jsou zobrazeny nebo skryty, jsou určeny pomocí ořezové cesty.[29]

Ořezová cesta definuje oblast, uvnitř které je vše vidět a zbylá vnější plocha se nezobrazuje. Tato oblast se označuje jako ořezová oblast. Použitím vlastnosti `clip-path` specifikujeme výřez, který bude použitý na prvek. V SVG `clip-path` můžeme ořez definovat pomocí základních tvarů jako jsou `polygon()`, `circle()`, `inset()` a `ellipse()`. Můžeme také použít SVG prvek `<clipPath>`, ve kterém nadefinujeme ořezovou oblast a pomoci `ID` ji odkážeme na SVG `clip-path`. Prvky `<clipPath>` se nikdy nevytvoří přímo. Jejich jediné použití je s pomocí vlastnosti `clip-path`.[29]

```
1 .element {clip-path: url(#svgClipPathID);}
2 .element {clip-path: polygon(...);}
```

Příklad 36: Použití ořezové oblasti odkazované pomocí url(ID)

Prvek `<clipPath>` může obsahovat libovolný počet základních tvarů jako `<circle>`, `<ellipse>`, `<line>`, `<polygon>`, `<polyline>`, `<rect>`, ale může obsahovat také prvky `<path>` a prvky `<text>`. Může obsahovat popisek (`<title>`, `<desc>`, `<metadata>`) a také element `<use>` nebo `<script>`. Element `<use>` může v `<clipPath>` odkazovat pouze na jednoduché tvary. Prvek `<clipPath>` také může obsahovat animace a to s pomocí `<animate>`, `<animateColor>`, `<animateMotion>`, `<animateTransform>` nebo `<set>`. Skupiny zahrnuté pomocí prvku `<g>` u `<clipPath>` nebudou fungovat.[29]

Ořez můžeme také aplikovat přímo s elementem `<image>`. Element `<image>` je v SVG používán k zahrnutí grafiky, která obsahuje celé SVG nebo rastrový obrázek. Pokud je to obrázek SVG, atributy `width` a `height` se použijí z `viewportu`. Při použití rastrového obrázku se obrázek zmenší tak, aby odpovídal nastaveným atributům `width` a `height`. U nastavení atributů musíme zajistit správným poměr stran.[29]

V CSS se u samotné ořezové cesty může ve vlastnostech `clip-path` definovat referenční prostor za použití `<basic-shape>`. Pro SVG `<clipPath>` je referenční prostor určen podle rámečku HTML elementu. Pokud aplikujeme `<basic-shape>` na SVG, referenční prostor může být nastaven jednou ze tří hodnot:

- Hodnota `fill-box` nastaví hranici samotného objektu.
- Při použití `stroke-box` bude hranice nastavena dle okraje objektu.
- Třetí hodnotou je `view-box`. Pokud není `viewBox` specifikován, tato hodnota použije nejbližší možný `viewport`. Pokud je `viewBox` uveden, pak je souřadnicový systém založen na původních rozměrech specifikovaných `viewBoxem`.

U prvku `<clipPath>` můžeme užívat řadu atributů jako `ID`, `class`, `transform` a atributy jako `fill`, `stroke` a další. Jeden z užitečných atributů je `clipPathUnits`. Tento atribut specifikuje souřadnicový systém pro obsah prvku `<clipPath>`. Může obsahovat jednu ze dvou hodnot. Hodnota `objectBoundingBox` nebo `userSpaceOnUse`, která je použita jako výchozí.[29]

Hodnota `userSpaceOnUse` představuje hodnoty v uživatelském souřadnicovém systému v místě prvku `clipPath` odkazující přes vlastnost `clip-path`. Lokální souřadnicový systém slouží k definování, jak jsou souřadnice a délky umístěny. Pro HTML prvky s CSS se model liší od modelu SVG. Prvky asociované s CSS jsou v souřadnicovém systému umístěny v levém horním rohu svého referenčního prostoru a každý bod se rovná jednomu pixelu. SVG prvky, které

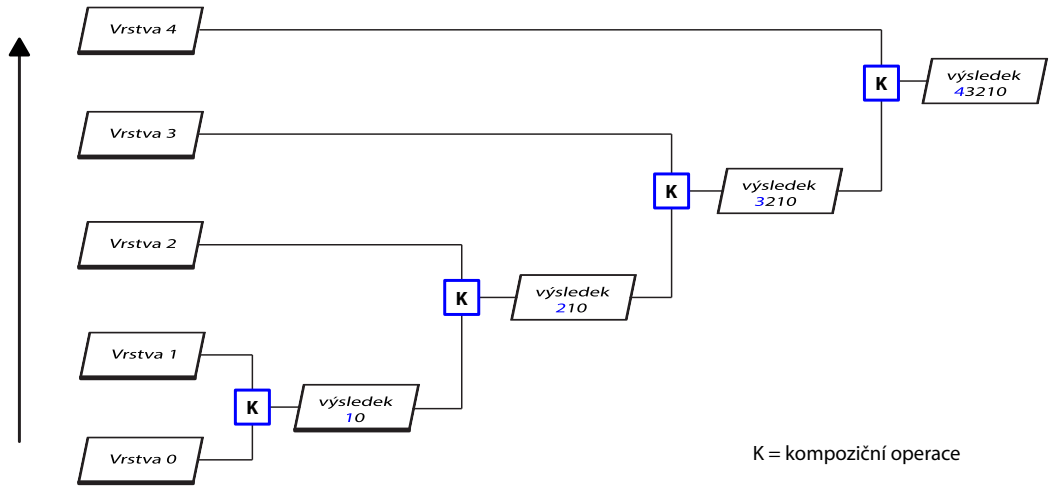
nejsou asociovány s CSS, jsou umístěny v nejbližším levém horním rohu výřezu (`viewport`) daného prvku. Ve většině případů je nejbližší výřez určen šířkou a výškou v prvku `<svg>`. U SVG lze také upravit souřadný systém pomocí atributu `viewBox`.

Hodnota `objectBoundingBox` vytvoří nový souřadnicový systém v levém horním rohu objektu, na který se vztahuje ořezová cesta stejné šířky a výšky jako má objekt. Tato hodnota je pro prvky SVG obzvláště užitečná, protože umožňuje použít ořezovou cestu jako hranici samotného objektu. Pokud použijeme `objectBoundingBox`, souřadný systém se stává jednotkovým systémem, takže v obsahu `<clipPath>` musí být rozmezí 0 až 1.[29]

Ořezové cesty jsou užitečné pro rychlé zahrnutí nebo vyloučení konkrétních oblastí grafiky, případně pro změnu pravoúhlých tvarů. Ořezové cesty jsou však omezené. Musíme je definovat pouze za pomoci jednoduchých vektorových tvarů. Pro detailní efekty potřebujeme masky.[22]

9 Masky u SVG

Maska funguje jinak než ořezová cesta. Maskovací kompozice umožňuje kombinovat různé vrstvy prvku `mask` do jedné pomocí různých operací. Kombinujeme je pomocí jednotlivých pixelů. Pokud zkombinujeme dvě vrstvy prvku `mask`, z každé vrstvy se vezme odpovídající pixel a na těchto pixelech se aplikuje určitá kompoziční operace, ze které získáme třetí pixel pro výslednou vrstvu. Výsledná vrstva se pak kombinuje s další vrstvou. Celý proces je zobrazen na obrázku 9.[30]



Obrázek 9: Proces kompozice masky jednotlivých vrstev

Masky definované v SVG jsou známe jako masky jasu (angl. luminance masks). Maska určuje neprůsvitnost výsledné maskované grafiky. Obrazová vrstva masky je převedena na stupně šedi a dle intenzity šedi se určuje průhlednost masky. Bílá barva odpovídá 100% jasu (`opacity 1`), což znamená že je maska průhledná. Černá barva odpovídá 0% jasu (`opacity 0`), takže všechny černé části budou zcela oříznuty.[22]

V SVG je element `<mask>` aplikován na jiný element buď pomocí CSS vlastnosti `mask`, nebo přímo pomocí `ID`. Vlastnost `mask` funguje podobně jako

vlastnost `clip-path`. Hodnota je buď `none` (výchozí hodnota) nebo jako odkaz `url()`. Tentokrát však musí být prvek v odkazu `<mask>`. Stejně jako u `<clipPath>` je element `<mask>` kontejnerem pro jinou grafiku, která může být zmenšena jako `objectBoundingBox` nebo jako `userSpaceOnUse`.[31]

Uvnitř masky mohou být použity všechny platné grafické prvky SVG včetně skupin, symbolů a vložených obrázků. Také mohou být použity všechny SVG styly jako jsou obrysy, výplně, vzory, přechody a průhlednost. Všechny tyto prvky se převedou jako obrazová vrstva na masku jasu. Element `<mask>` používá stejné atributy jako `<pattern>`, jeden pro rozměr a druhý pro obsah. Atribut `maskUnits` (výchozí hodnota `objectBoundingBox`) řídí rozměr celkové oblasti masky obdélníkem definovaným v prvku `<mask>` atributy `x`, `y`, `width` a `height`. Atribut `maskContentUnits` (výchozí hodnota `userSpaceOnUse`) řídí rozměr prvku uvnitř masky.[22]

Pro rozměr masky je vhodné vytvořit masku s přidáním "10 %" paddingu, která zakryje okraje maskované grafiky na všech stranách. To je dobré pro všechny obrázky, které se přesně nevejdou do určeného rámce a pro tvary a text s tenkým obrysem.[22]

10 Filtry u SVG

Stejně tak jako přechody, masky, vzory a další grafické efekty v SVG, tak i filtry mají svůj vyhrazený prvek `<filter>`. Prvek `<filter>` nikdy není vykreslen přímo. Můžeme ho použít v prvku SVG pomocí reference `ID`, nebo s CSS funkcí url(). Prvek `<filter>` vyžaduje k funkčnosti zdroj obrázku.[32]

K ovládání filtrů existuje několik atributů. Atributy pozice `x`, `y` s výchozí hodnotou -10 %. Atributy `width` a `height` s výchozí hodnotou "120 %". K předdefinování rozlišení slouží `filterRes`. Atribut definující systém pro atributy `x`, `y`, `width` a `height` se nazývá `filterUnits` s výchozí relativní hodnotou `ObjectBoundBox` nebo absolutní hodnotou `userSpaceOnUse`.[28]

```
1 <svg width="500" height="500" viewBox="0 0 500 500">
2     <filter id="myFilter">
3         <!-- filtry -->
4     </filter>
5     <image xlink:href="..." width="100\%" height="
        100\%" x="0" y="0" filter="url(\#myFilter)"></
        image>
6 </svg>}
```

Příklad 37: Aplikace SVG filtru

Prvek `<filter>` je kontejner pro řadu filtrů, které dohromady vytvářejí efekt. Tyto operace se v SVG nazývají „Filter Primitives“. Každý základní filtr provádí jeden grafický proces na jednom nebo více vstupech. V rámci specifikace SVG filtrů je v současné době definováno 17 základních filtrů, které jsou schopné grafických efektů jako jsou šumy, textury, světelné efekty, manipulace s barvami apod.[32]

Základní filtry pracují tak, že se vezme zdrojová grafika a na tu se aplikuje

filtr. Výstup jednoho efektu lze použít jako vstup do dalšího. To znamená, že na sebe můžeme kombinovat nespočet grafických efektů. Každý základní filtr může mít jeden nebo dva vstupy a pouze jeden výsledek. Vstup je definován v atributu `in` a výsledek operace je definován v atributu `result`. Pokud efekt filtru zadá druhý vstup, bude nastaven v atributu `in2`. Výsledek operace může být použit pro jakoukoliv jinou operaci. Pokud není vstup specifikován v atributu `in`, výsledek předchozí operace se automaticky použije jako vstup.

Kromě toho základní filtry přijímají i další vstupní hodnoty (například obrázek nebo text). `SourceGraphic` určuje, že na prvek je použit celý filtr. `SourceAlpha` je totéž jako `SourceGraphic`, ale grafika prvku může obsahovat jen kanál alfa.[33]

11 Způsoby animace SVG

Animaci SVG lze použít pro oživení webových stránek a je základem interaktivního designu. Hodí se zejména pro animaci ikon, tlačítek, formulářových polí, interakčních přechodů apod. Existují tři různé způsoby jak animovat SVG. Můžeme použít CSS, SMIL nebo JavaScript. Tyto způsoby lze kombinovat a tím se navzájem doplňují.[10]

11.1 Animace s použitím CSS

Stejné techniky, které se používají pro animaci a přechody u HTML prvků, lze použít i pro animaci SVG. CSS animace umožňuje přechod mezi jednotlivými stavy za pomoci `@keyframes`. Při použití animace můžeme používat jen atributy, které označujeme jako Presentation attributes.[21]

```
1 @keyframes animate-circle {
2   0%   {transform: translateX(0)}
3   100% {transform: translateX(800px)}
4 }
5 #circle {
6   animation: animate-circle 3s linear;
7   animation-fill-mode: forwards;
8 }
```

Příklad 38: CSS animace s použitím `@keyframes`

11.2 Animace s použitím SMIL

Další možností jak animovat SVG je pomocí SMIL (Synchronized Multimedia Integration Language). Tuto možnost nepodporují prohlížeče od firmy Microsoft, přestože má více možností než CSS. Smil je deklarativní způsob animace

s přehlednou syntaxí, který je umístěn přímo uvnitř <svg> elementu.[21]

Hlavním tagem je `<animate>`, uvnitř kterého definujeme animaci pomocí základních atributů, díky nimž můžeme vytvořit různé animace, které bychom s pomocí CSS vytvořit nemohli.[10]

- `attributeName` *požadováno* Definuje platný atribut na element, který bude animován.
- `From` *volitelný* Označuje začátek animace. Pokud atribut vynecháme, animace začne od aktuální hodnoty.
- `To` *požadováno* Označuje konečnou hodnotu, do které animujeme.
- `dur` *požadováno* Označuje trvání animace. Podporuje časové hodnoty jako 2s nebo 1250ms.
- `xlink:href` *požadováno* Specifikuje ID elementu, který bude animován. Definuje odkaz v případě, že není animace uvnitř animovaného elementu.
- `begin/end` *volitelný* Specifikuje, kdy má animace začít nebo končit. Možným atributem je `click`.
- `fill` *volitelný* Specifikuje, co se stane po skončení animace. Například můžeme použít atributy `freeze` nebo `remove`.

Kromě použití tagu `<animate>` můžeme použít také `<animateTransform>`, `<set>` nebo `<animateMotion>`.[21]

```
<animate
        xlink:href="#circle"
        attributeName="cx"
        from="75"
        to="900"
        dur="3s"
        begin="0s"
        fill="freeze" />
```

Příklad 39: SMIL animace s použitím tagu `<animate>`

11.3 Animace s použitím JavaScriptu

Ještě předtím, než se animace vytvářela za pomoci CSS `keyframes` a přechodů, využíval se JavaScript. JavaScript umí animovat SVG, protože umí manipulovat s objekty v DOM. Pro animaci je přistupováno k JavaScript cyklům pomocí metody `requestAnimationFrame`, díky které vypadá animace v prohlížeči plynule.[21]

Metoda `requestAnimationFrame` nejdříve zjistí vhodnou snímkovou frekvenci pro animaci v závislosti na zařízení, které je používáno. Například, pokud budeme na mobilním zařízení, nebude používat tak vysokou snímkovou frekvenci jako na počítači. Tato metoda je velice výkonná a používá ji většina externích JavaScript knihoven.[34]

Díky externím JavaScriptovým knihovnám je animace mnohem snazší. Nejznámější knihovny, které můžeme použít jsou, GreenSock (GSAP), VelocityJS, jQuery, Snap.svg, apod.[21]

12 Media Queries u SVG

Media Queries umožňují definovat CSS pravidla k zobrazení webového obsahu v závislosti na rozlišení nebo velikosti různých zařízení.

Pokud nebudeme u SVG používat Media Queries, jednoduše se obrázek zvětší nebo zmenší tak, aby se vešel do výřezu nebo do kontejneru.

Pokud použijeme inline SVG, výřez HTML a výřez SVG bude stejný, tak se SVG dokument bude chovat stejně jako jakýkoliv jiný HTML prvek. Pokud bychom SVG odkazovali pomocí elementů jako `object` nebo `img`, jedná se o výřez z SVG dokumentu. Media Queries pracují v obou případech, ale když je SVG odkazován, jeho výřez je na HTML dokumentu nezávislý. V takovém případě neurčuje velikost výřezu prohlížeč, ale určují je dané elementy.

S pomocí Media Queries můžeme nastavit viditelnost jen části obrázku. Na velkých zařízení může být zobrazeno např. celý logotyp. Na menších zařízení může být zobrazeno pouze logo, část loga nebo symbol.[35]

13 Praktická část

V praktické části této bakalářské práce jsem vytvořil webovou prezentaci, která demonstruje použití SVG prvků v rámci webové stránky.

Webová prezentace začíná úvodním rozcestníkem, který je rozdělen na dvě části. V jedné části je umožněno stažení textu bakalářské práce. Druhá část je odkazem dokumentační sekce.

V dokumentační sekci sděluji získané znalosti z teoretické části bakalářské práce. V úvodu představuji, co SVG je a jaká je jeho současná podpora v prohlížečích, proč je vhodné SVG ve webovém designu používat a jaké jsou rozdíly oproti grafice bitmapové. Dále uvádím možné varianty implementace SVG do webové stránky a strukturu SVG dokumentu se zajištěním umístění a správného vykreslení grafických prvků v prohlížeči. Popisuji manipulaci s SVG za pomoci CSS a JavaScriptu. Následně ukazuji jakým způsobem můžeme SVG animovat a jak se SVG chová při použití Media Queries.

V závěru praktické části jsou jednotlivě uvedeny příklady pro práci s SVG texty, obrázky, animacemi apod., ze kterých je zhotovena ukázková stránka s možnostmi využití SVG ve webovém designu.

Pro zpracování webové prezentace jsem použil vývojové prostředí Visual Studio Code[13]. Zhotovenou webovou prezentaci naleznete na:

- http://svg.tomasvcelak.cz/

[13]https://code.visualstudio.com/

14 Závěr

Možnosti využití SVG ve webovém designu jsou obrovské, avšak nejsou vhodné pro každou situaci. Pomocí SVG nemůžeme nahradit fotografie a některou rastrovou grafiku. V případě vhodného použití je SVG díky zmíněným výhodám opravdu skvělý formát při využití na webových stránkách.

Moderní prohlížeče SVG formát podporují a nemusíme mít obavy z jeho využití. V případě, že budeme chtít podporovat starší prohlížeče, budeme muset SVG grafiku nahradit grafikou bitmapovou.

Praktická část bakalářské práce byla vytvořena jako pomůcka pro výuku SVG. Z dokumentační části s příklady se lze naučit, jak SVG správně využívat a z jakého důvodu jsou právě obrázky SVG formátu přínosné. Z příkladů je zhotovena ukázková webová stránka, která demonstruje jejich praktické využití.

Zajímavým pokračováním bakalářské práce by mohlo být použití SVG pro vizualizaci informací například pomocí interaktivních grafů. Pozoruhodné by také bylo využití SVG spolu s JavaScriptovým rozhraním WebGL pro vytvoření interaktivní 3D grafiky.

Seznam použité literatury a zdrojů

[1] GILTSOFF, Jake. *A Practical Guide to SVGs on the web* [online]. 2015 [cit. 2018-04-22]. Dostupné z: <https://svgontheweb.com>.

[2] MICHÁLEK, Martin. *SVG: vektorový formát, který na webu chyběl* [online]. 2014 [cit. 2018-04-22]. Dostupné z: <https://www.vzhurudolu.cz/prirucka/svg>.

[3] W3C. *An SVG Primer for Today's Browsers* [online]. 2010 [cit. 2018-04-29]. Dostupné z: <https://www.w3.org/Graphics/SVG/IG/resources/svgprimer.html>.

[4] IRELAN, Ryan. *SVG Beginners Tutorial* [online]. 2016 [cit. 2019-02-10]. Dostupné z: <https://svgontheweb.com>.

[5] GIRARD, Jeremy. *Why You Should Be Using SVG on Your Website* [online]. 2019 [cit. 2019-03-31]. Dostupné z: <https://www.lifewire.com/using-svg-in-web-design-3470014>.

[6] NOWELL, Peter. *How Designers Should Think About SVG* [online]. 2017 [cit. 2019-02-10]. Dostupné z: <https://medium.com/sketch-app-sources/how-designers-should-think-about-svg-b2b92efc4d77>.

[7] SOUEIDAN, Sara. *An Overview Of SVG Sprite Creation Techniques* [online]. 2014 [cit. 2019-02-15]. Dostupné z: <https://www.sarasoueidan.com/blog/overview-of-svg-sprite-creation-techniques/>.

[8] FRAŠKO, Josef. *Bitmapová grafika* [online]. 2018 [cit. 2019-02-15]. Dostupné z: <https://www.kspa-grafika.cz/l/bitmapova-grafika/>.

[9] FRAŠKO, Josef. *Vektorová grafika* [online]. 2018 [cit. 2019-02-16]. Dostupné z: <https://www.kspa-grafika.cz/l/vektorova-grafika/>.

[10] COYIER, Chris; HEAD, Val. *Practical SVG*. 1. vyd. A Book Apart, 2016. ISBN 978-1-937557-43-0.

[11] SOUEIDAN, Sara. *Tips For Optimizing SVG Delivery For The Web* [online]. 2014 [cit. 2019-02-16]. Dostupné z: <https://www.sarasoueidan.com/blog/svg-optimization/>.

[12] SERIC, Jayden. *How to optimize SVG* [online]. 2018 [cit. 2019-02-16]. Dostupné z: <https://jaydenseric.com/blog/how-to-optimize-svg>.

[13] MICHÁLEK, Martin. *SVG fallbacky* [online]. 2016 [cit. 2019-02-17]. Dostupné z: <https://www.vzhurudolu.cz/prirucka/svg-fallbacky>.

[14] COYIER, Chris. *Test for Support of SVG as img* [online]. 2015 [cit. 2019-03-17]. Dostupné z: <https://css-tricks.com/test-support-svg-img/>.

[15] BELLAMY-ROYDS, Amelia. *A Complete Guide to SVG Fallbacks* [online]. 2019 [cit. 2019-03-20]. Dostupné z: <https://css-tricks.com/a-complete-guide-to-svg-fallbacks/>.

[16] SOUEIDAN, Sara. *Making SVGs Responsive with CSS* [online]. 2014 [cit. 2019-02-18]. Dostupné z: <https://tympanus.net/codrops/2014/08/19/making-svgs-responsive-with-css/>.

[17] BELLAMY-ROYDS, Amelia. *How to Scale SVG* [online]. 2017 [cit. 2019-03-18]. Dostupné z: <https://css-tricks.com/scale-svg/>.

[18] SOUEIDAN, Sara. *Better SVG Fallback and Art Direction With The <picture> Element* [online]. 2015 [cit. 2019-04-09]. Dostupné z: <https://www.sarasoueidan.com/blog/svg-picture/>.

[19] BUCKLER, Craig. *How to Add Scalable Vector Graphics (SVG) to Your Web Page* [online]. 2012 [cit. 2019-03-20]. Dostupné z: <https://www.sitepoint.com/add-svg-to-web-page/>.

[20] TRYTHALL, Joni. *Pocket Guide to Writing SVG* [online]. 2016 [cit. 2019-03-22]. Dostupné z: <http://svgpocketguide.com/book/>.

[21] LARSEN, Rob. *Mastering Svg*. 1. vyd. Packt Publishing Ltd., 2018. ISBN 978-1-78862-674-3.

[22] BELLAMY-ROYDS, Amelia; CAGLE, Kurt; STOREY, Dudley. *Using SVG with CSS3 and HTML5: Vector Graphics for Web Design*. 2. vyd. O'Reilly Media, 2018. ISBN 978-1-491-92197-5.

[23] SOUEIDAN, Sara. *Styling And Animating SVGs With CSS* [online]. 2014 [cit. 2019-03-22]. Dostupné z: <https://www.smashingmagazine.com/2014/11/styling-and-animating-svgs-with-css/>.

[24] COLLINGRIDGE, Peter. *Using Javascript with SVG* [online]. 2018 [cit. 2019-03-23]. Dostupné z: <http://www.petercollingridge.co.uk/tutorials/svg/interactive/javascript/>.

[25] MDN. *<text> - SVG: Scalable Vector Graphics* [online]. 2019 [cit. 2019-03-23]. Dostupné z: <https://developer.mozilla.org/en-US/docs/Web/SVG/Element/text>.

[26] SOUEIDAN, Sara. *Techniques For Creating Textured Text* [online]. 2013 [cit. 2019-03-23]. Dostupné z: <https://www.sarasoueidan.com/blog/textured-text-techniques/>.

[27] SOUEIDAN, Sara. *CSS vs. SVG: Graphical Text Effects* [online]. 2015 [cit. 2019-03-29]. Dostupné z: <https://theblog.adobe.com/css-vs-svg-graphical-text>.

[28] WEBER, Dirk. *The Art Of SVG Filters And Why It Is Awesome* [online]. 2015 [cit. 2019-03-30]. Dostupné z: <https://www.smashingmagazine.com/2015/05/why-the-svg-filter-is-awesome/>.

[29] SOUEIDAN, Sara. *Clipping in CSS and SVG — The clip-path Property and <clipPath> Element* [online]. 2014 [cit. 2019-03-30]. Dostupné z: <https://www.sarasoueidan.com/blog/css-svg-clipping/>.

[30] TUDOR, Ana. *Mask Compositing: The Crash Course* [online]. 2019 [cit. 2019-03-31]. Dostupné z: <https://css-tricks.com/mask-compositing-the-crash-course/>.

[31] GAEBEL, Dennis. *A Comprehensive Guide to Clipping and Masking in SVG* [online]. 2018 [cit. 2019-03-29]. Dostupné z: <https://webdesign.tutsplus.com/tutorials/a-comprehensive-guide-to-clipping-and-masking-in-svg--cms-30380>.

[32] SOUEIDAN, Sara. *SVG Filters 101* [online]. 2019 [cit. 2019-03-31]. Dostupné z: <https://tympanus.net/codrops/2019/01/15/svg-filters-101/>.

[33] ROBERTS, Steven. *The complete guide to SVG* [online]. 2018 [cit. 2019-03-30]. Dostupné z: <https://www.creativebloq.com/features/the-complete-guide-to-svg/4>.

[34] DRASNER, Sarah. *A Comparison of Animation Technologies* [online]. 2018 [cit. 2019-04-01]. Dostupné z: <https://css-tricks.com/comparison-animation-technologies/>.

[35] BROWN, Tiffany. *Using SVG with Media Queries* [online]. 2018 [cit. 2019-04-01]. Dostupné z: <https://www.sitepoint.com/using-svg-with-media-queries/>.

Seznam obrázků

Seznam příkladů

A Příloha

CD

- Celé znění bakalářské práce v PDF a zdrojový kód ve formatu LaTeX
- Zdrojové kódy webové prezentace

B Příloha

Webová prezentace

- http://svg.tomasvcelak.cz/