Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Bakalářská práce

Mapové API v prostředí WWW

Tomáš Steska

! ! !

Místo této strany vložíte zadání bakalářské práce.

(Do jedné vazby originál a do druhé kopii)

! ! !

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Mapové API v prostředí WWW" jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 11. 3. 2015 _____

Poděkování

Rád bych touto cestou poděkoval Ing. Pavlu Šimkovi, Ph.D. za cenné rady a připomínky, které mě poskytl při zpracovávání této bakalářské práce.

Mapové API v prostředí WWW

Map API in WWW environment

Souhrn

Se stálým nárůstem počtu webových stránek se na internetu objevují stále častěji weby, které obsahují vlastní mapové API pro vyznačení své adresy, případně nejrůznějším způsobem tohoto vyznačení rozšiřují. A to za pomoci jak map statických, tak s využitím javascriptu.

Hlavní náplní teoretické části je vytvoření přehledu jak základních funkcí statických map, tak i javascriptových mapových API od Google, včetně podrobného popisu a ukázek jednotlivých funkcí, které se nejčastěji využívají. Méně využívané funkce jsou popsané stručněji.

Praktická část se v první etapě zabývá srovnáním tuzemského mapového API od Seznamu se zmiňovaným Google. V druhé fázi je pak vytvořená ukázková mapa za použití javascriptu, včetně využití složitějších metod s jejich následným rozborem a popisem.

Summary

With constantly increasing number of websites, websites containing their own maps API for marking their location or websites extending this marking in different ways are more likely to be found on the Internet. Extension can be made by means of both statistical maps and javascript.

Theoretical part mainly contains overview of basic functions of statistical maps as well as of javascript API mapping by Google, including detailed description and examples of particular functions, which are most commonly used. Less common functions are described just briefly.

First part of practical part consists of comparison between domestic API mapping by Seznam and Google, which was already mentioned above. In the second part, there is

map example created using javascript, including application of more complex methods and their subsequent analysis and description

Klíčová slova: Seznam, Mapové API, Javascript, Mapy, Google API, Internet, Statické mapy, Prvky mapových API

Keywords: Seznam, Maps API, Javascript, Maps, Google API, Internet, Static maps, Elements of maps API

Obsah

1 Úvod

Google je v současné době největším poskytovatelem internetových služeb, včetně mapových API. Díky kvalitnímu zpracování, možnostem jejich praktického využití a adaptací zastává jednoznačně dominantní postavení a nenachází mezi ostatními firmami konkurenty. Především díky jejich bohaté rozmanitosti a detailně zpracované dokumentaci. Z toho pak vyplývá uživatelská oblíbenost, díky které existují spousty diskuzních fór, které v případě problémů umožňují uživatelům najít správné řešení.

Cílem práce je shrnutí základních funkcí mapových API, se kterými by se měl zájemce detailněji seznámit. Pro lepší pochopení jednotlivých funkcionalit a jejich detailního nastavení je u funkcí kromě jejich popisu uvedena praktická ukázka zápisu. V rámci popisu jsou zmíněny dvě možnosti, které slouží k vytvoření mapy. Jedná se o mapy statické, jež využívají pouze příkaz ve formátu internetové adresy, tak i o mapy využívající javascript, u nichž je již potřeba základních znalostí jazyků HTML a CSS. Oba typy map mají své klady i zápory.

U statické verze zpracování se jedná o jednoduchou ovladatelnost a její snadné přizpůsobení. Naproti tomu druhá varianta přináší výrazně širší nabídku funkcionalit s velkou možností povolených úprav. To ale za cenu složitějšího zápisu jednotlivých částí kódu a větších potřebných znalostí na její zpracování.

V oblasti mapových API je vhodné zmínit českou firmu Seznam, která provozuje vlastní API. Díky umožnění komerčního využití a kvalitním mapovým podkladům pro Českou republiku, se stalo oblíbenou variantou pro některé tuzemské uživatele. Proto je zpracováno srovnání zaměřené na obecnou funkčnost obou API a jednotlivých mapových podkladů, které tyto firmy nabízejí.

Pro komplexnější pochopení celkové problematiky je v rámci javascriptu od Google, vytvořena praktická ukázka s přehledem vybraných funkcí, které mapy nabízejí, včetně jejich následného vysvětlení a vzájemného propojení jednotlivých funkcí.

2 Cíl a metodika práce

2.1 Cíl práce

Bakalářská práce je tematicky zaměřena na problematiku mapových API v prostředí WWW. Hlavním cílem práce je vytvoření přehledu základních funkcionalit Google API včetně reálného využití a následné srovnání s API Seznamu. Dílčím cílem práce je vytvoření příkladu mapového API s použitím funkcí Google API

2.2 Metodika

Metodika řešené problematiky bakalářské práce je založena na studiu a analýze odborných informačních zdrojů. Vlastní práce spočívá v seznámení se s prostředím Google API, ve vytvoření přehledu základních funkcionalit, především za pomoci technické dokumentace stránek Google Developers a knižního podkladu Beginning Google API 3 (Svennerberg), jejich popisu, rozboru a vytvoření ukázek z jednotlivých funkcionalit.

Součástí práce bude také srovnání možností nabízených prostředím Google API s českým ekvivalentem Seznam API, kterému bude předcházet studium kvality nabízených služeb a specifických funkcí.

V rámci praktické části bude vytvořen vzorový příklad API na základě získaných informací a to za použití vybraných funkcí pro základní nastavení map, vytvoření mapových značek včetně jejich animací a plánovaní trasy mezi dvěma body s využitím současné polohy. Veškeré vzniklé problémy budou řešeny na základě studia literatury a za použití odborných diskuzních fór.

Na základě syntézy teoretických poznatků a výsledků praktické části budou formulovány závěry bakalářské práce.

3 Literární rešerše

Následující literární rešerše je parafrázována z anglického jazyka a založena převážně na knihách *Beginning Google Maps API 3* [1] a *Google Maps JavaScript API Cookbook* [2], následně na technické dokumentaci z Google Developers [3] [4] a dalších odborně založených webů a knih [5] [6] [7].

3.1 Seznámení s API

Zkratka API vyjadřuje application programming interface neboli programovací rozhraní pro aplikace. V rámci své činnosti využívá softwarové komponenty, jejich vstupy a výstupy a základní typy. Hlavním cílem API je definovat sadu funkcí, které nezávisí na jejich implementacích a umožňují definice a implementace měnit, aniž by došlo k narušení chodu.

Kromě přístupu k databázím nebo počítačovému hardware, jako jsou pevné disky či grafické karty, můžeme API využít k usnadnění práce při programování grafického uživatelského prostředí. S jeho pomocí můžeme začlenit nové funkce do již existujících aplikací nebo sdílet data mezi jinak zdánlivě odlišnými aplikacemi. Obvykle se API vyskytuje v podobě souboru knihoven, funkcí, procedur, tříd a protokolů.

API může nabývat mnoha podob, včetně mezinárodních forem jako je POSIX, či knihoven pro programovací jazyky, kam například patří Standard Template Library v C++, nebo Java API. [8] [9] [10]

3.2 Historie a vývoj Google maps

Samotné Google mapy se poprvé objevily v roce 2004. Byly vytvořeny v jazyce C++ bratry Lars a Jens Rasmussen. První ideou bylo, aby si každý uživatel mohl stáhnout mapy přímo z internetu. Tento princip následně nahradil nápad, že se mapy stanou čistě webovým základem.

V polovině následujícího roku vzniká Google maps API a Google Earth, které zobrazovaly satelitní snímky celého světa. Dokonce včetně katastrofy, kterou způsobil hurikán Katrina (v následujících letech byly pak snímky nahrazeny).

V roce 2006 byl pak Earth implementován přímo do Google maps, které začaly zároveň plnit funkci cestovních map, ale pouze pro světové giganty jako byla Velká Británie, Japonsko, USA. Ve třetině téhož roku vychází druhá verze Maps API a o několik měsíců později byla tato novinka rozšířena o schopnost Geokódování. Ke konci roku se objevily první 3D modely budov a také dopravní zastávky.

V roce 2007 se začaly objevovat i dopravní situace v reálném čase, zatím jen ve větších městech v USA. Mapové API se postupně rozrůstalo o nové funkce, jako například plánování tras. V květnu stejného roku Google představil první Street View, které umožňovalo výhled 360°, opět pouze ve Spojených Státech. Ve druhé polovině roku byly mapy rozšířeny o dalších 54 států, s možností plánování tras s využitím MHD a s možností přepínání zobrazení různých druhů map.

V následujících letech byly mapy několikrát aktualizovány a Street View bylo rozšířeno o další významné státy (Japonsko, Austrálie,…) a možnost javascriptového využití označované jako verze 3. Na přelomu desetiletí byla doprogramována podpora pro Google Chrome k již podporovaným prohlížečům a API bylo ke komerčním účelům zpoplatněno. Od roku 2011 až do současnosti se pokrytí funkcí Street View dále rozšiřovalo a došlo k mnoha aktualizacím již zmapovaných částí.

3.3 Statické mapy

Prvním druhem map, které jsou i v současnosti velmi využívány, jsou mapy statické. Tyto mapy jsou přístupné na webu bez jakýchkoliv dalších požadavků, jako je například javascript, nebo dynamické načítaní. Celá mapa se všemi parametry je vytvořena pouze na základě URL linku, který již obsahuje všechny náležitosti jako velikost mapy, její polohu, či upřesňující informace a styly. Celý proces funguje zavoláním HTTP požadavku, který navrátí vzhled mapy, včetně všech jejich náležitostí a stylů, které link obsahuje, na stránku. Celková délka linku nesmí přesáhnout při zadávání 2048 znaků. Po zaslání requestu pomocí URL se vrátí obraz ve formátu GIF, PNG či JPG.

Příklad číslo 1 - Statické mapy - Přehled validních znaků [4]

Set	characters	URL usage
Alphanumeric	a b c d e f g h i j k l m n o p q r s t u v w x y z A B C D E F G H I J K L M N O P Q R S T U V W X Y Z 0 1 2 3 4 5 6 7 8 9	Text strings, scheme usage (`http`), port (`8080`), etc.
Unreserved	- _ . ~	Text strings
Reserved	! * ' ( ) ; : @ & = + $, / ? % # [ ]	Control characters and/or Text Strings

Následující kapitoly se zaměřují na jednotlivé prvky, které lze u statických map nastavit.

3.3.1 Parametry URL

Samotné statické mapové API jsou jednoduché na vytváření a používání. Jednotlivé mapy se volají pomocí URL linků. Všechny typy parametrů, které lze u URL využít, se dají rozdělit do tří kategorií.

První z nich jsou parametry pozice, které se umisťují hned za označení staticmap. Po jejich skončení se zbytek odděluje čárkou, na rozdíl od dalších kategorií. Základem těchto parametrů je funkce CENTER, jež určuje bod, na který bude mapa vycentrována a funkci ZOOM, pod kterou nalezneme přiblížení (oddálení) mapy.

<*https://maps.googleapis.com/maps/api/staticmap?parameters*> [4]

Další větší kategorií je samotné nastavení map umožňující přizpůsobení většiny ostatních dat, které se mají na mapě vyskytovat. A to od velikosti mapy, kde se zadává vertikální a horizontální velikost v pixelech. Dále pak možnost SCALE, pod kterou si lze představit počet pixelů, které daná mapa bude mít. Defaultně s hodnotou 1, ale i s možností zadání hodnoty 2 nebo 4. Tato možnost říká, že daná mapa se vrátí s dvakrát, či čtyřikrát více pixely než je tomu u defaultního nastavení. Další položkou na výběr je MAPTYPE, který umožňuje zobrazit klasickou, satelitní, hybridní nebo čistě terénní mapu. Metoda „Formát“ následně určuje typ grafického formátu, v jakém se mapy budou zobrazovat. Vybrat lze z jednoho z již dříve zmíněných formátů JPG, PNG a GIF. Předposlední metodou, která se dá nastavit, je LANGUAGE určující v jakém jazyku budou popisky mapy. Poslední metodou je funkce REGION vymezující hranice pro zobrazení. Na rozdíl od první varianty, je nyní potřeba oddělovat veškeré parametry symbolem ampersand (&) a jedinou povinnou složkou mapových parametrů je SIZE. Všechny ostatní jsou nepovinné.

Poslední kategorií jsou funkce, které umožňují dotvořit detaily přímo na mapě, jako například připojení obrázku k určitému místu. Tuto funkci nám umožní MARKERS.

Samotných markers je možné mít i větší množství a mohou mít jak stejné stylování tak i nabývat zcela rozdílného. Jednotlivé definice se pak oddělují pomocí značky roura (|).

Celkem zajímavou funkcí je příkaz PATH, pomocí něhož lze vyznačit oblast přímo mezi určitými body nebo přímo udělat trasu propojením míst pomocí zadaných bodů. Další funkcí je VISIBLE, přispěním které lze docílit stavu, že specifická místa zůstanou stále vyznačená na mapě. Poslední v seznamu je STYLE. Ten umožňuje vytvořit vlastní styl pro jednotlivé objekty, jako jsou parky, cesty atd.

Následující podkapitoly jsou věnovány zmiňovaným funkcím, detailním hodnotám a ukázkám jejich využití v praxi.

3.3.2 Nastavení lokace

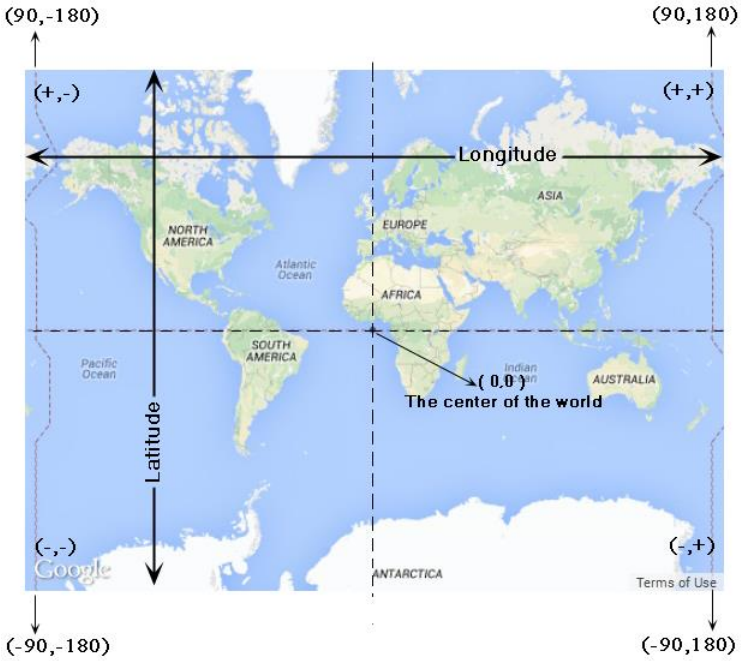
Statické mapy musí mít jednoznačně zadanou výchozí polohu ve správném tvaru. Toto zadání lze provést třemi různými způsoby.

První je pomocí vyznačeného bodu za pomoci funkce MARKERS nebo využitím numerického tvaru (zeměpisné délky a šířky).

Poslední možností určení je stringový řetězec, pomocí kterého se zadává přesná adresa požadované lokality, která se má zobrazit.

Samotná šířka a délka je pak ve tvaru čísla. Pro šířku v rozmezí od -90 až po 90 a pro šířku pak hodnoty od -180 do 180. Jelikož toto zadání by bylo velmi nepřesné, k udání polohy se využívá číslo s upřesněním na šest desetinných míst. Ostatní desetinná místa jsou pak již ignorována. Výsledné zadání polohy pak může vypadat například takto: 50.088535, 14.423739 (tyto souřadnice označují pozici Prahy).

Příklad číslo 2 - Mapa ukazující Lattitude a Longitude [6] [11]



Další možností je určení přímo podle adresy. Tuto volbu ocení uživatelé méně zběhlí v orientování se pomocí souřadnic LatLng. Operaci přímého zadání adresy nazýváme jako Geocoding. API je schopné geokódování provést samo, ale pouze za předpokladu zadání existující a platné adresy. Samotné kódování pak využívá URL-scaped, které zadanou adresu například z tvaru „Praha 1“ převede na „Praha+1“ nebo „Karlův most, Praha 1“ na podobu „Karluv+most,Praha+1“.

V případě zápisu přesné adresy je potřeba použít zvolený typ zadávání do všech dalších funkcí, které určení polohy vyžadují. Výsledný kód pak může vypadat například takto.

<https://maps.googleapis.com/maps/api/staticmap?center=Praha+1&zoom=12&size=800x800>

Příklad číslo 3 - Statické mapy - nastavení vycentrování mapy [11]

3.3.2.1 Zoom

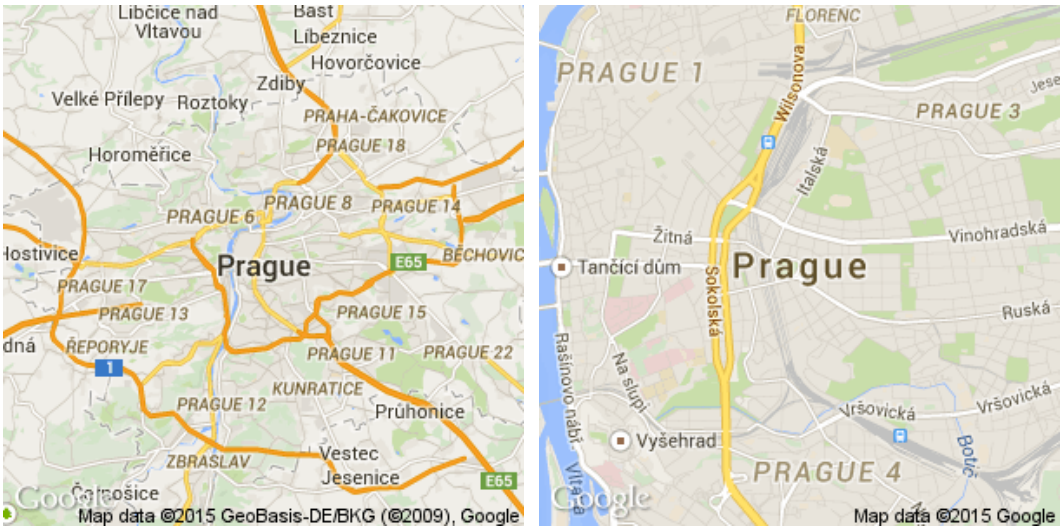
Zoom nám udává, jak moc budeme mít náhled přiblížený. Samotná hodnota se zadává jako integer, který nabývá hodnot 0 až 22, kde 0 je nejvzdálenější pohled, zatímco 22 je maximální možné přiblížení. Pokud bude zadána větší hodnota než 22, mapa, která se vrátí jako odpověď, již nebude mít větší přiblížení než u hodnoty 22. V některých oblastech může být maximální hodnota ještě nižší v závislosti na kvalitě map.

Na příkladu je zaznamenán projev hodnoty zoomu 10 a 13.

<https://maps.googleapis.com/maps/api/staticmap?center=Praha&zoom=10&size=300x300>

<https://maps.googleapis.com/maps/api/staticmap?center=Praha&zoom=13&size=300x300>

Příklad číslo 4 - Statické mapy - ukázka nastavení různých úrovní přiblížení (zoom 10 a zoom 13) [11]



3.3.3 Nastavení map

Tato kapitola se věnuje nastavení vzhledu mapy, která se z URL vrací. První část se zaměřuje na nastavení měřítka, dále pak na typy a formátování vzhledu mapy. Je to jediný povinný parametr, který je nutno u každé mapy uvést k získání adekvátní odpovědi.

3.3.3.1 Velikost mapy a měřítko

Velikost udává, jak velká mapa se vrátí. Hodnota se zadává jako VERTIKÁLNÍ x HORIZONTÁLNÍ (LATTITUDE x LONGITUDE) a uvádí se v pixelech. Velikost je úzce spjatá s měřítkem, které určuje, kolik pixelů se vrátí. Měřítku (SCALE) je možno přiřadit hodnotu 1, 2 nebo 4. Hodnota 4 funguje jen ve verzi „for work“.

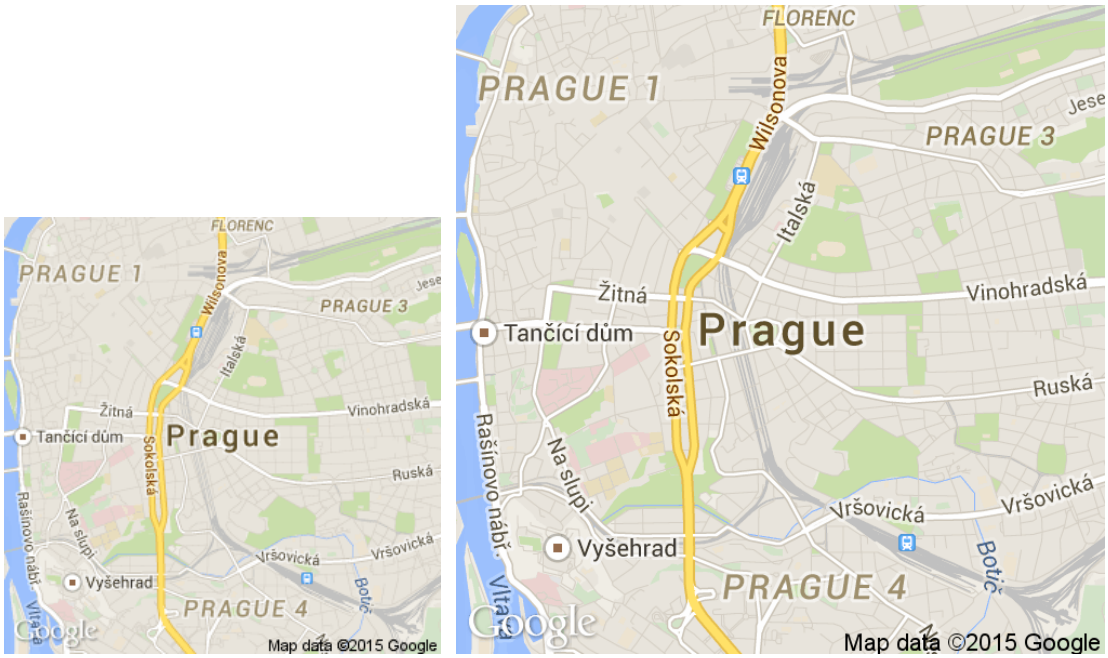
V praxi funguje tak, že pokud je zadán požadavek na mapu 300 x 300 s měřítkem hodnoty 2, vrátí se mapa o velikosti 600 x 600, ovšem tato mapa bude zobrazovat stejnou oblast, jako kdyby byla zvolená velikost 300 x 300, ale bude na ní dvakrát tolik pixelů, čímž bude ve finále větší. Pokud bude přímo zadáno 600 x 600, vrátí se mapa, která zachytí pouze větší zobrazovanou oblast.

<https://maps.googleapis.com/maps/api/staticmap?center=Praha&zoom=13&scale=1&size=300x300>

<https://maps.googleapis.com/maps/api/staticmap?center=Praha&zoom=13&scale=2&size=300x300>

<https://maps.googleapis.com/maps/api/staticmap?center=Praha&zoom=13&scale=1&size=600x600>

Příklad číslo 5 - Statické mapy - rozdíl mezi hodnotami scale [11]



3.3.3.2 Typy map

Následně lze zvolit, v jaké podobě se mapy zobrazí. Lze je nechat vrátit v klasickém provedení (ROADMAP), eventuálně jako mapy terénní nebo hybridní. K nastavení vzhledu se používá funkce MAPTYPE a přiřazuje se mu typ mapy, který bude zobrazovat.

První možností je ROADMAP, jenž je zobrazení, které je defaultně nastaveno. Adaptací tohoto provedení jsou mapy hybridní (HYBRID), zobrazující satelitní mapy, avšak se zvýrazněním a popsáním komunikací, parků a dalších objektů.

Dalším druhem jsou mapy satelitní (SATELLITE), kde již žádné zvýraznění ani popisky nejsou zobrazeny.

Posledním druhem jsou terénní mapy (TERRAIN). Základ mají stejný jako mapy klasické, ale na rozdíl od nich přesně zobrazují klenutí terénu v dané lokalitě.

3.3.3.3 Formáty map

K nastavení typu vraceného obrázku pak slouží funkce FORMAT. Pomocí ní se zvolí, zda se má mapa vrátit ve tvaru JPG, PNG, nebo GIF. Mimo to je nutno u některých

formátů rozlišit i další možnosti. Formát png lze zadávat buď PNG (PNG8) pro vrácení 8 bitového formátu a nebo PNG32 pro návrat 32 bitové hodnoty. U dalších možností již další upřesnění nejsou.

Díky rychlé odezvě v rámci webu je stále nejoblíbenější možností využívání formátu JPG.

3.3.4 Funkce map

Poslední sadou úprav jsou funkce, které umožňují co nejvíce změnit vrácenou mapu podle přání uživatele. Pomocí funkcí je možno dodělat různé vyznačení, ale také do určité míry přímo změnit vzhled zobrazovaných map.

3.3.4.1 Značky

Značky umožnují vytvořit označení na místo, které má být na mapě zvýrazněné. K tomuto slouží funkce MARKERS. Pokud se má zobrazit pouze jeden styl značek, používá se jeden marker a do něho se přiřadí pouze více souřadnic jednotlivých míst pro vytvoření jedné nebo více značek stejného stylu. Pokud je požadavek na rozdílný styl značek, je potřeba použít markerů více a u každého nastavit jeho vlastní styl, pod jakým bude viditelný.

Samotný zápis se provede v následujícím pořadí. Nejprve se do markers přiřazuje styl, jak bude vypadat a následně znak roury, za který se již vypisuje seznam adres. U samotného stylu lze upřesnit tři parametry. Velikost, barva a popisek.

Pro velikost používáme příkaz SIZE, který může nabývat pouze tří hodnot. Nejmenší možnou variantou je TINY, následuje hodnota SMALL a největší, která se používá je MID. Poslední varianta se zobrazí také v případě, že možnost SIZE nebude uvedena.

Další kategorií je nastavení barvy, neboli COLOR. K výběru lze použít seznam všech 24-bitových barev, nikoliv však 32-bitových, které jsou v dnešní době obvyklé. Barva je zapisována v hexadecimálním tvaru (např. 0xFFFFFF). Druhou možností je výběr jedné z již přednastavených barev ze seznamu (black, white, red, green, blue, yellow, purple, gray a brown).

Poslední funkcí je tvorba vlastního pojmenování neboli štítku. K tomuto se využívá příkaz LABEL. Název pak může obsahovat jeden znak ze škály {0-9, A-Z}. K použití

popisu značky je třeba využít nastavení velikosti na MID, jelikož při zvolení jiné možnosti bude značka příliš malá a písmeno/číslo se nezobrazí.

Po vytvoření typu štítku, je potřeba zadat jednu nebo více lokací na mapě, kam se mají nadefinované štítky umístit. Adresu lze opět zadat oběma způsoby. Jak pomocí souřadnic, tak zadáním přesné adresy, které stále musí splňovat určitý formát zápisu. Při zadávání více adres, je potřeba oddělovat jednotlivé místa rourou. V případě, že nebudou nastaveny základní parametry CENTER a ZOOM, značky se na mapě nezobrazí.

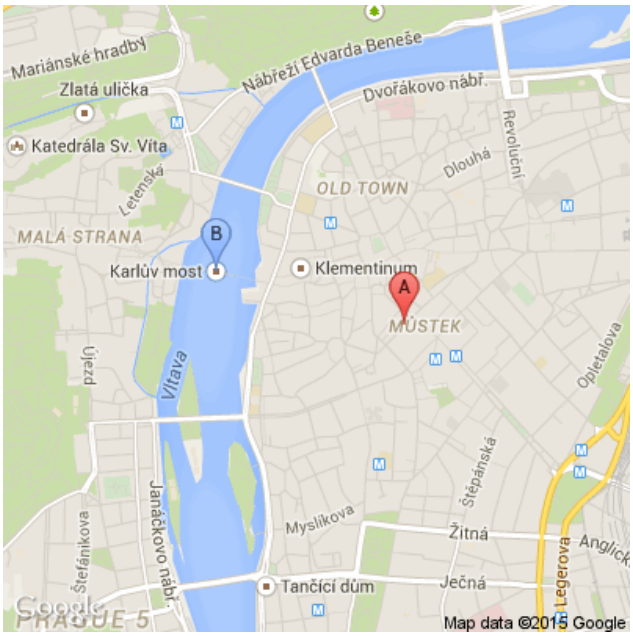
<https://maps.googleapis.com/maps/api/staticmap?center=50.0848576,14.4169059&zoom=14&size=400x400&markers=color:red|label:A|50.0846108,14.4216786|50.086477,14.411437>

Příklad číslo 6 - Statické mapy - vytvoření více značek na mapě [11]

Další příklad ukazuje zápis štítků při použití volby více druhů stylů.

<https://maps.googleapis.com/maps/api/staticmap?center=50.0848576,14.4169059&zoom=14&size=400x400&markers=color:red|label:A|50.0846108,14.4216786&markers=color:blue|label:B|50.086477,14.411437>

Příklad číslo 7 - Statické mapy - vytvoření více značek různých typů na mapě [11]



3.3.4.2 Cesta

Cesta umožňuje propojit zvolenou sadu bodů, dokonce i vyznačit celou oblast mezi danými body. K nastavení cesty se využívá funkce PATH, která má stejný tvar zápisu jako předchozí štítky. Na prvním místě se zapíše styl PATH a pak se postupně zadávají adresy, které se oddělují znakem roury. Ve stylu lze upřesnit hmotnost, barvu, výplň a zakřivení.

Hmotnost určuje, jak bude výsledná čára mezi body silná. Síla spojnice se zadává pomocí číselné hodnoty v pixelech. V případě, že nebude žádná hodnota zadaná, nastaví se defaultní hodnota, která činí 5 pixelů.

Další možností nastavení je barva. U tohoto nastavení lze použít stejné postupy jako u štítků, protože nastavení je stejné jako v předchozím příkladu. Také se využívá pouze 24 bitová hodnota pro zadání barvy, anebo je možno využít přednastavených barev, které zůstávají ve stejném rozsahu, jako v předchozím případu. Na konec nastavení barvy je nutné přidat ještě jednu hexa hodnotu od 00 do FF, která určuje průsvitnost čáry. 00 je pro 0% krytí a naopak FF je pro neprůhlednou čáru.

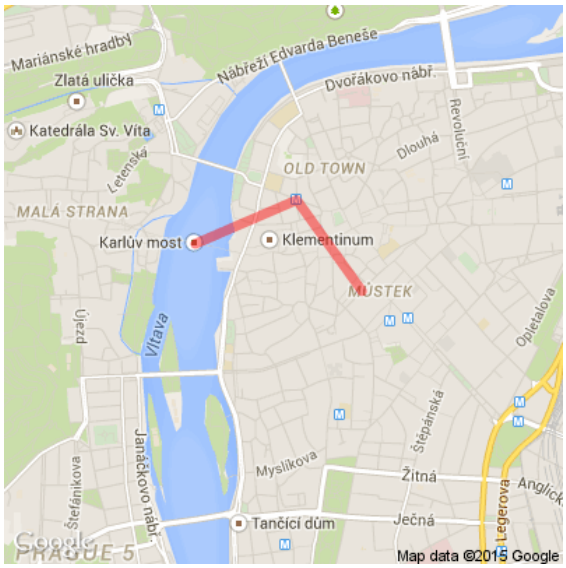
Novou vlastností, kterou mapy umožňují upravit je funkce FILLCOLOR, jež umožní nastavit vlastní barvu výplně mezi propojenými body, ale pouze pokud je tvar uzavřený (3 a více bodů).

Další novou funkcí jsou GEODESIC. Fungují jako propojení bodů, které není tvořené pouze linií. Jsou ale mnohem sofistikovanější a kopírují i zakřivení země na trase mezi zadanými body. Funkce pak nabývá jednoduchých parametrů a to hodnoty TRUE nebo FALSE.

Pro lepší představu a pochopení všech funkcí a parametrů následuje ukázka.

<https://maps.googleapis.com/maps/api/staticmap?center=50.0848576,14.4169059&zoom=14&size=400x400&path=color:red|weight:7|50.0846108,14.4216786|50.088203,14.417671|50.086477,14.411437>

Příklad číslo 8 - Statické mapy - vytvoření cesty na mapě spojující 3 body [11]



V případě požadavku na vymezení oblasti, je potřeba definovat i barvu výplně. Jinak se oblast nezobrazí. Je potřeba vybrat jednu ze dvou možností. Pokud má být v oblasti zobrazena čára spojení, musí se zadat výchozí bod na konec adres a tím se útvar uzavře.

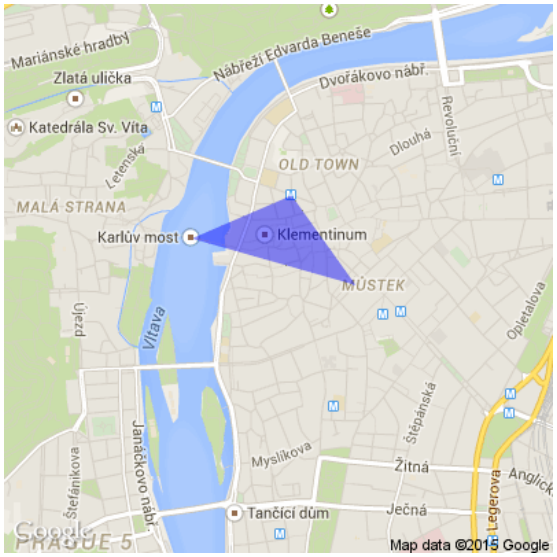
Pokud nemá být v oblasti zobrazena čára spojení, nastaví se barva čáry s nulovým krytím, aby se vůbec nezobrazila.

<https://maps.googleapis.com/maps/api/staticmap?center=50.0848576,14.4169059&zoom=14&size=400x400&path=fillcolor:blue|weight:7|50.0846108,14.4216786|50.088203,14.417671|50.086477,14.411437|50.0846108,14.4216786>

Příklad číslo 9 - Statické mapy - vykreslení tvaru na mapě se zvýrazněným okrajem [11]

<https://maps.googleapis.com/maps/api/staticmap?center=50.0848576,14.4169059&zoom=14&size=400x400&path=color:0x00000000|fillcolor:blue|weight:7|50.0846108,14.4216786|50.088203,14.417671|50.086477,14.411437>

Příklad číslo 10 - Statické mapy - vykreslení tvaru na mapě bez zvýrazněného okraje [11]



3.3.4.3 Styl

Styl je poslední možností, kterou lze nastavit u vlastních stylů mapy. Pod tímto označením se skrývá změna zobrazení jednotlivých elementů, jako jsou silnice, parky a budovy, aby vypadaly jinak než v případě defaultního zobrazení. Využívá se především k zvýraznění určitých prvků na mapě, které mají být zobrazeny.

U každé mapy lze použít libovolné množství stylů. Samotný styl se potom musí skládat z pravidel a výběru, pro které budou pravidla platit. Jednotlivé typy deklarací se oddělují pomocí znaku roury. Všechny funkce jsou volitelné a nemusí se uvádět v deklaraci.

První možností, kterou lze nastavit je funkce FEATURES, která určuje oblast výběru. V případě neuvedení upřesnění bude automaticky nastavena původní hodnota, která obsahuje ALL (feature:all). Mezi nejčastěji používané patří výběr cesty (feature:road) nebo pozadí mezi cestami (feature:landscape).

Možností výběru prvků je obrovské množství a mnoho položek má svoje vlastní podkategorie. Například LANDSCAPE lze upřesnit na LANDSCAPE.NATURAL pro výběr přírody, či LANDSCAPE.MAN_MADE, pomocí kterého lze zvolit postavené budovy. Zajímavým upřesněním je položka s názvem POI, která vybere pouze body zájmu a se správným upřesněním může vyselektovat do výběru nemocnice, turistická lákadla, školy, sportovní centra a mnoho dalších. Například škola se vybere následovně.

[style:poi.school]

Rozšířením skupiny FEATURE jsou prvky, které upřesnění upravovaný výběr v dané kategorii. K modifikaci se využívá funkce ELEMENT. Ta umožňuje změnit vzhled popisků, které jsou například u silnic nebo grafických čar. Pro první verzi se volá upřesnění ELEMENT:LABELS a pro druhou variantu pak ELEMENT:GEOMETRY. V případě, že není upřesněný výběr, je defaultně vybrána možnost ALL a úpravy se budou vztahovat na všechny typy prvků na mapě. Například výběr popisků u přírodních ploch.

[Style:landscape.natural|element:labels]

Následující kapitola se zaměřuje na styl a pravidla formátování. Každý druh stylu může obsahovat jednu a více úprav. Jednotlivé úpravy se mezi sebou dělí pomocí roury a k upřesnění u jednotlivých funkcí se využívá znak dvojtečka. Následuje přehled jednotlivých funkcí, které lze používat.

První možností je HUE, která se zadává v podobě RBG hexa řetězce a určuje barvu zvoleného výběru. Tento druh barevného určení využívá takzvaného barevného kruhu, který zobrazuje druhy barev. Na rozdíl od klasického výběru barevný kruh RGB nevyužívá sytosti a světelnosti daných barev. Proto v případě zvolení dvou hexa kódů lze dostat jako návratovou hodnotu dvě stejné barvy. Další rozlišení barev se provádí využitím speciálních funkcí, které určí sytost a světelnost.

Následně je potřeba se zaměřit na světelnost a sytost. Pro první možnost se používá funkce s označením LIGHTNESS, která může nabývat hodnot od -100 až do 100. Hodnota určuje procentuální světelnost elementu. V případě, že je použita hodnota nižší, vrátí se jako odpověď tmavý odstín (v případě zadání -100 černá) a naopak pokud bude hodnota nasvícení kladná, vrátí se odstín světlejší (pro bílou se zadává hodnota 100). Další možností je funkce INVERSE_LIGHTNESS, která původní lesklost jednoduše převrátí na opačnou.

V případě sytosti, neboli SATURATION, kde lze zadat taktéž hodnoty od -100 do 100. Zesilují nebo zeslabují intenzitu zvolené barvy. V praxi to znamená, jak moc bude daná barva výrazná. V případě zadání -100 vrátí pouze kombinaci bílé a černé barvy a naopak při hodnotě 100, zobrazí čistou barvu, která je bez jakýchkoliv příměsí bílé a černé.

Dále lze nastavit gamma záření. Tuto možnost lze najít pod heslem GAMMA, a dá se jí přiřadit hodnota od 0.01 až do 10.0. Defaultní hodnotou je 1.0. Gama nám umožňuje upravit světelnost nelineárním způsobem mezi bílými a černými hodnotami. Například pokud je zobrazeno příliš světlé barvy je možnost pomocí gammy upravit kontrast mezi okraji jednotlivých uskupení. Za předpokladu, že hodnoty budou menší než 1, pak kontrast vzroste a naopak pokud zadaná hodnota bude větší než 1, pak kontrast klesne.

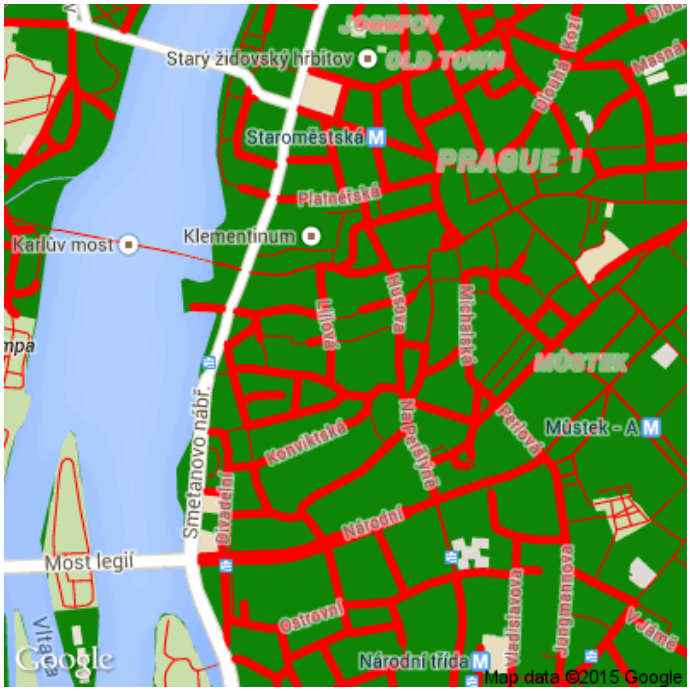
Poslední možností je funkce VISIBILITY, která může nabývat pouze hodnot ON, OFF nebo hodnoty SIMPLIFIED. Viditelnost určuje, jak se určitý prvek na mapě objeví. V případě použití možnosti SIMPLIFIED může mapa zjednodušit zobrazení některých prvků tak, aby se stále zobrazovaly správně.

Po zadání požadavku na mapu jsou pak všechny funkce zobrazovány v pořadí, v jakém byly styly deklarovány. Proto si je potřeba dávat pozor na správné pořadí zapsání jednotlivých prvků.

Pro dokonalejší pochopení budou funkce předvedeny na příkladu za použití extrémních hodnot pro kontrastní rozlišení jednotlivých elementů.

<https://maps.googleapis.com/maps/api/staticmap?center=50.0848576,14.4169059&zoom=15&size=400x400&style=feature:road.local|element:geometry|color:0xff0000|visibility:on&style=feature:landscape|element:geometry|color:0x0E8607|saturation:50|visibility:on>

Příklad číslo 11 - Statické mapy - ukázka použití stylů s extrémními hodnotami [11]



3.4 Mapové API Javascript v3

Samotné API, za použití javascriptu (dále jen JS), se liší tím, že vyžadují alespoň základní znalost HTML a taky schopnost psaní a porozumění JS kódu. Samotný skript je pak umístěn přímo do HTML nebo do jiného souboru, který je však s tímto HTML provázán. Výhoda JS oproti již popisovaným statickým mapám spočívá v jeho rozsáhlých možnostech úprav a prvků, které jsou uživatelům poskytnuty k vytvoření téměř jakékoliv mapy.

Prvním krokem, který tato metoda vyžaduje, je vytvoření vlastního unikátního identifikačního klíče, bez kterého není možné API vůbec psát. Samotné vygenerování klíče je zdarma, avšak obsahuje určité denní limity pro provolávání. Pro vygenerování je nutné mít platný účet na Google Developers, kde již stačí jen vybrat požadované API, pro které klíč požadujeme.

3.4.1 Základní nastavení

Jako první krok je potřeba založit spouštěcí HTML soubor a nastavit mu základní formátování. Dalším krokem je samotná deklarace JS a jeho provázání s HTML v případě externího souboru. Deklarace skriptu by mohla vypadat následně.

Příklad číslo 12 - Deklarace JS - kód [3]

```
<html>
  <head>
    <script type="text/javascript"
      src="https://maps.googleapis.com/maps/api/js?key=API_KEY">
    </script>
```

„API_KEY“ se nahrazuje vlastním klíčem, který se vygeneroval. Tento tag je pro fungování celého API zcela zásadní, jelikož načítá všechny symboly a definice. Tento URL je však ještě možné použít ve více adaptacích a to například s upřesněním národní volby. Ta určuje, v jakém jazyce se na mapě budou zobrazovat popisky. Také lze rozhodnout, jak se bude zobrazovat text. Jestli zleva do prava (LTR) nebo zprava do leva (RTL). Tuto volbu lze využít například u arabštiny a to přidáním parametru „DIR=“ a směr.

Příklad číslo 13 – Nastavení národní volby - kód

```
<script type="text/javascript" src="https://maps.googleapis.com/maps/api/js?language=en">
</script>
```

V případě, že nebude skript přímo v HTML, je potřeba přidat propojení na umístění souboru. Samotný soubor by pak měl byt ve formátu „NAME.js“. V uvedeném případě je použitý název „javascript.js“.

Příklad číslo 14 - Propojení HTML s JS - kód

```
<script type="text/javascript" src="javascript.js"></script>
```

Skript pak běží pod oběma typy zabezpečení, tedy jak pod HTTPS, tak i pod HTTP. Další možností zjednodušení pro uživatele je možnost načtení vytvořených knihoven, které se automaticky nenačítají, ale musí se dodatečně jednotlivě volat.

Správné vytvoření maps API začíná již v HTML, kde je pro mapu potřeba rezervovat dostatečný prostor, aby se správně zobrazovala a nebyla nijak deformovaná. Následné nastavení map je pak již velmi podobné použitému nastavení ve statických mapách avšak tentokrát za použití JS kódu.

Samotné škály zadávaných hodnot se nemění, rozdíl je jen ve způsobu zadávání. Pro nastavení vycentrování mapy na určitý bod se použije zeměpisné šířky a délky, kde hodnoty jsou zadané na maximálně šest desetinných míst. Při zadání více čísel za desetinou čárkou, nebude poloha již více upřesněna. Toto zadání polohy lze dělat hned několika metodami.

Příklad číslo 15 - Nastavení polohy - kód [3]

```
center: new google.maps.LatLng(-34.397, 150.644)
center: {lat: -34.397, lng: 150.644}
center: {lng: 150.644, lat: -34.397}
```

Další položkou, kterou je potřeba zadat hned na začátku, je přiblížení mapy, jehož škála se pohybuje od 0 do 22, kdy při zadání hodnoty 22 a větší je mapa maximálně přiblížena. Maximální přiblížení avšak nemusí fungovat vždy. V mnoha případech se setkáme s menší kvalitou map, a tedy i s menší možností přiblížení map, pokud mají zůstat zaostřené. Samotné zapracování do JS je pomocí příkazu „Zoom:“.

Posledním prvkem pro dokončení mapy je vytvoření nové instance třídy.

Příklad číslo 16 - Vytvoření nové instance třídy - kód [3]

```
var map = new google.maps.Map(document.getElementById("map-canvas"),
    mapOptions);
```

Takto lze vytvořit jednu mapu na stránku. Stejným způsobem lze vytvořit libovolné množství instancí pro danou třídu (funguje stejně jako objektové programování např. v C#). Následuje prvek, který mapu přiřadí ke správnému objektu (místo na stránce které je vymezeno) v podobě document.getElementById(). Posledním prvkem bude nadefinováno nastavení mapy (název je volitelný ale musí být uvedený všude stejně). Takto vytvořená mapa se musí ještě načíst zapomoci příkazu google.maps.event.addDomListener(window, 'load', initialize), který zjednodušeně říká, ve kterém kroku se kód zpracuje a předá do BODY v HTML. Alternativou pak může být použití „Onload“, který může vypadat například takto: window.onload = function(), které bude naopak na začátku JS.

Pomocí těchto poznatků lze již vytvořit mapu, která je možné vycentrovat na určité místo s určitým přiblížením a s těmito parametry ji vykreslit. V první části je vidět HTML kód, který je následovaný JS.

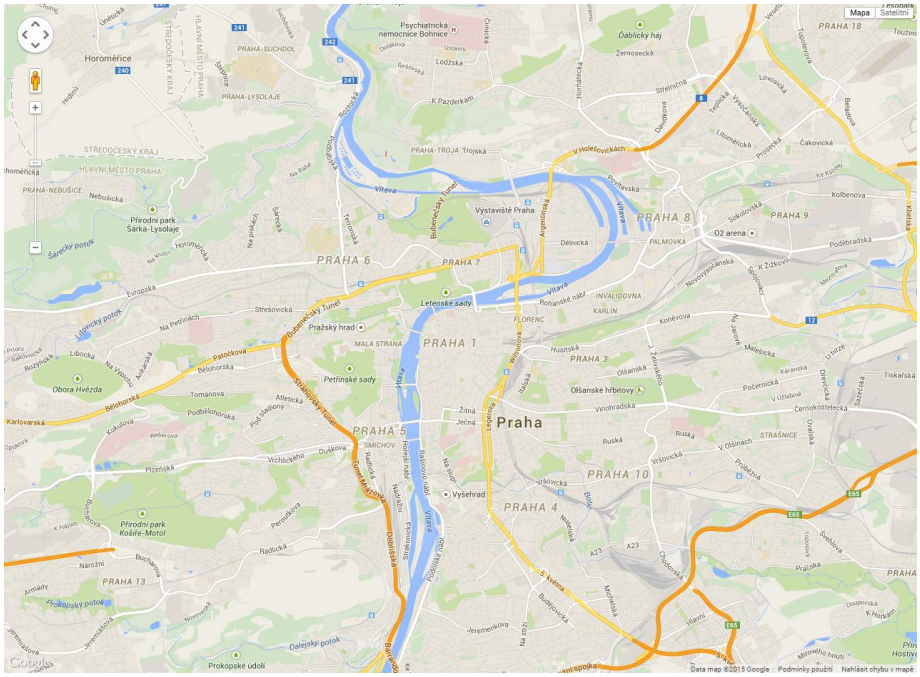
Příklad číslo 17 – Vytvoření nové mapy - kód

```
<!DOCTYPE html>
<html>
  <head>
    <style type="text/css">
      html, body, #map-canvas { height: 100%; margin: 0; padding: 0;}
    </style>
    <script type="text/javascript"
      src="https://maps.googleapis.com/maps/api/js?key=API_KEY">
    </script>
    <script type="text/javascript" src="javascript.js"></script>
  </head>
  <body>
<div id="map-canvas"></div>
  </body>
</html>
```

```
function initialize() {
   var nastaveniMapy = {
     center: { lat: 50.088535, lng: 14.423739},
     zoom: 13
   };
   var map = new google.maps.Map(document.getElementById('map-canvas'),
       nastaveniMapy);
 };
google.maps.event.addDomListener(window, 'load', initialize);
```

Jako výsledek se vrátí požadovaná mapa, která při tomto nastavení zaujímá plochu celé stránky, je vycentrovaná na Prahu 1 a úroveň přiblížení je nastavená na stupeň 13.

Příklad číslo 18 - Výsledná mapa – ukázka [11]



V případě, že se mapa nemá zobrazovat na celou obrazovku, stačí pouze adaptovat tag DIV v HTML a pomoci vlastnosti STYLE dodat pevnou velikost zobrazující se mapy. Pomocí WIDTH lze nastavit šířku a využitím HEIGHT pak naopak výšku. Obě hodnoty se obvykle zadávají v pixelech (PX), které se jsou uvedeny těsně za hodnotu.

Příklad číslo 19 – Nastavení pevné velikosti mapy - kód

```
<div id="map-canvas" style="width: 600px; height: 600px;"></div>
```

3.4.2 Typy map

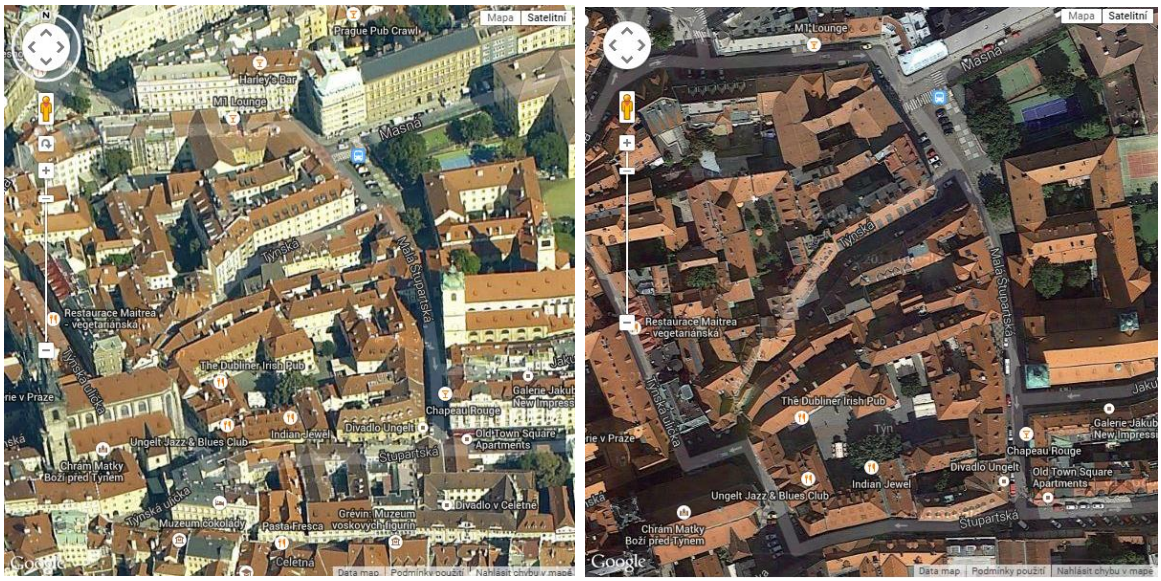
Tak jako u statických map lze změnit vzhled načítané mapy na takový, jaký je požadován. V základu jsou čtyři typy map. Jedná se o mapy ROADMAP, jež jsou pod základním vyobrazením většiny map. Následuje mapa SATELLITE, neboli satelitní. Dále pak mapa hybridní (HYBRID) a nakonec terénní (TERRAIN). Všechny tyto typy jsou detailněji vysvětleny u statických map. Jejich implementace v JS se pak provádí pomocí příkazu MapTypeId.TYP_POŽADOVANÉ_MAPY.

Oproti statickým mapám přibývá u satelitních a hybridních map možnost pohledu pod úhlem 45°, což je umožněno pouze za předpokladu menšího oddálení mapy. Při větším oddálení se mapa sama přepne na pohled z vrchu. Naklonění lze dosáhnout příkazem “setTilt(45)” (tento pohled je defaultně nastaven od Google). Náhled lze vypnout zadáním příkazu “setTilt(0)”.

Příklad číslo 20 - Hybridní mapa se zkoseným pohledem - kód

```
var nastaveniMapy = {
    center: { lat: 50.088535, lng: 14.423739},
    zoom: 18,
    mapTypeId: google.maps.MapTypeId.HYBRID
};
var map = new google.maps.Map(document.getElementById('map-canvas'),
   nastaveniMapy);
map.setTilt(45);
};
```

Příklad číslo 21 - Vykreslení mapy setTilt(45) a setTilt(0) - ukázka [11]



3.4.3 Styly map

Jak již bylo uvedeno u statických map, lze pomocí stylů přizpůsobit prezentovanou mapu podle nároků uživatele. Je možno změnit veškeré vizuální prvky (plochy, cesty…).

Úpravy lze rozdělit do dvou větších kategorií. Jedná se o FEATURES, což jsou grafické prvky, které můžeme na mapě vybrat (vodní plochy, silnice, budovy…) a STYLERS, které naopak upravují samotný vzhled mapy (barva, gama, odstín…).

3.4.3.1 Features

Tato kategorie obsahuje celou řadu objektů, jako například silnice nebo vodní plochy. Celkově objekty tvoří strom, kde každá větev je upřesněním určité kategorie. Globálně jsou pak všechny prvky zahrnuty pod MapTypeStyleFeatureType. Samotný zápis funguje na stylu rodičů a dceřiných kategorií, kde platí, že změna rodiče ovlivní všechny podkategorie, ale naopak nikoliv. Jednou z větších kategorií jsou cesty či body zájmu.

Příklad číslo 22 – Nastavení stylu s výběrem kategorie - kód [3]

```
{
  featureType: "road"
}
```

V případě podkategorie se za hlavní kategorii píše tečka „.” a za ni bez mezer podkategorie, která má byt změněna.

Příklad číslo 23 – Nastavení stylu s výběr podkategorie - kód

```
{
    featureType: "road.local"
}
```

Po výběru upravované kategorie následuje volba jednotlivého prvku, u kterého má změna nastat. Například u silnic lze změnit jak geometrii (GEOMETRY) tak i zobrazované popisky (LABELS), které se v ní vyskytují. V případě neupřesnění výběru je automaticky přednastaveno “ALL”, neboli změny všech možností.

Příklad číslo 24 Nastavení stylu s popiskem lokálních silnic - kód [3]

```
{
  featureType: "road.local",
  elementType: "labels"
}
```

3.4.3.2 Stylers

Po vybrání prvku, následuje jeho upravení pomocí stylů. Můžeme upravit vše, co se týká vzhledu nebo daný prvek úplně vypnout, aby se na mapě zobrazoval. Veškeré možnosti úprav stylů zůstávají stejné jako u “Statických map” a to včetně zadávaných škál.

Jedná se o položky:

- HUE upravující barvu. Zadáváno pomocí RGB
- LIGHTNESS (světelnost), kterou zadáváme z rozmezí hodnot -100 (pro převážně černé) až 100 (pro převážně bílé)
- SATURATION (sytost), hodnoty jsou stejné jak u světelnosti, kde -100 značí nejmenší sytost a 100 maximální
- GAMMA kde se zadávají hodnoty od 0.1 do 10.0 a výchozí hodnotou 1.0
- INVERT_LIGHTNESS, které nabývá hodnoty TRUE v případě použití
- VISIBILITY, kde se vyskytují hodnoty ON, OFF nebo SIMPLIFIED podle toho jestli objekt bude nebo nebude zobrazen, nebo případně se zobrazí bez určitých stylů
- COLOR se zadává pomocí RGB stringu nebo předdefinovanou barvou
- WEIGHT se zadává jako číslo větší nebo rovné nule. Hodnota je zadána v pixelech a určuje tloušťku

Detailní popis těchto prvků byl řešený v kapitole 3.4.4.3 Styl u statických map. Samotný zápis by pak měl být ve tvaru.

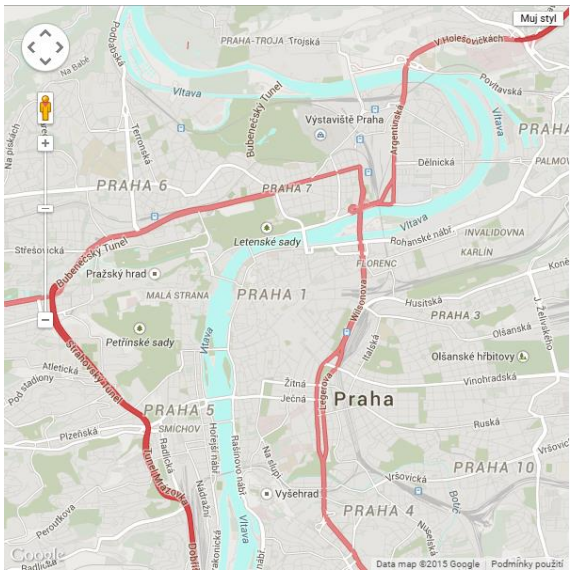
Příklad číslo 25 - Jak správně zapsat Features a Stylers v JS - kód [3]

```
var stylesArray = [
  {
    featureType: '',
    elementType: '',
    stylers: [
      {hue: ''},
      {saturation: ''},
      {lightness: ''},
      // etc...
    ]
  },
  {
    featureType: '',
    // etc...
  }
]
```

Za pomoci těchto parametrů lze vytvořit vlastní adaptaci mapy. Veškeré měnitelné položky lze najít u podrobnější dokumentace Features, přímo na stránkách Google Developers. Výsledný kód může být například v tomto tvaru.

Příklad číslo 26 - Ukázka vlastního stylu mapy u JS – kód a ukázka [11]

```
  var styles = [
 {
   stylers: [
     { saturation: -50 }
   ]
 },{
   featureType: "road.highway",
   elementType: "geometry",
   stylers: [
     { hue: "#754d4d" },
     { saturation: 25 }
   ]
 },{
   featureType: "water",
   elementType: "geometry",
   stylers: [
     { hue: "#00ffe6" },
     { saturation: 50 }
   ]
 }
];
```



Aby bylo možno tohoto stylu dosáhnout, musí se do kódu přidat položka, která vytvořený styl umožňuje na danou mapu aplikovat.

Příklad číslo 27 - Aplikování stylu na mapu - kód

```
var styledMapType = new google.maps.StyledMapType(styles,
{ name: 'Muj styl' });
  map.mapTypes.set('Muj styl', styledMapType);
```

3.4.4 Přizpůsobení zobrazení

Mimo úpravu vzhledu samotné mapy, existuje možnost úpravy ovládacích prvků, které jsou na ní umístěny. Tyto prvky lze přesunout, vypnout nebo dokonce vytvořit prvky zcela nové. Do základního zobrazení patří celkem osm prvků, kterými jsou:

- ZOOM zobrazuje posuvník, nebo v případě menších map pouze tlačítko +/- a slouží k nastavování úrovně přiblížení mapy.
- PAN zobrazuje tlačítko pro posun po mapě v levém horním rohu.
- SCALE umožňuje nastavit měřítko mapy. Defaultně je tato možnost na mapě vypnutá.
- MAPTYPE je položka umístěná v prvém horním rohu a umožňuje přepínání mezi jednotlivými typy map (satelitní, cestovní…).
- STREET VIEW je na mapě nad položkou “Zoom” v podobě žluté postavy. Přetáhnutím postavy na vyznačenou část mapy se zapne funkce Streetview.

- ROTATE umožňuje otáčet mapu o 90°. Samotná volba je defaultně zapnutá, ale zobrazuje se až při přepnutí do režimu 45°, do té doby je invisible.
- OVERVIEW MAP zobrazuje náhled mapy, který je zobrazen jako výřez ve větší zobrazovací oblasti. Defaultně je obsažen v levém spodním rohu pod znakem šipky.

Pro povolení vlastních úprav je nezbytné, nejprve vypnout defaultní přednastavení. Provádí se pomocí příkazu disableDefaultUI, kterému se musí přiřadit hodnota TRUE.

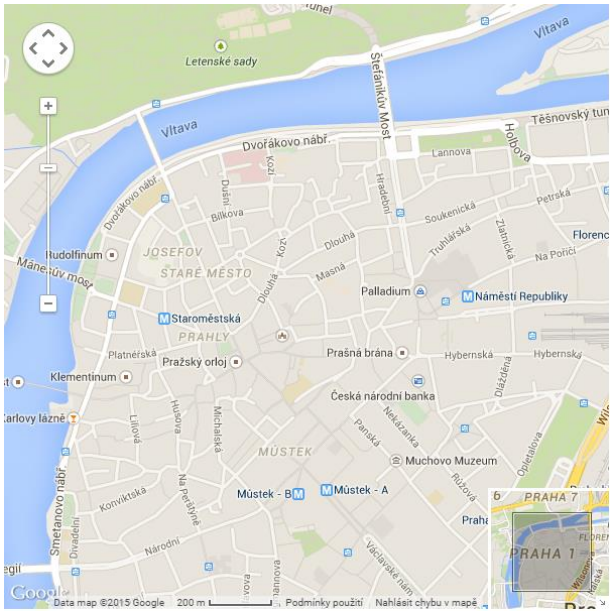
Příklad číslo 28 - Vypnutí Default control u Google maps pomocí JS – kód [12]

```
disableDefaultUI:true
```

Jednotlivé položky jsou ve tvaru "Název položky"Control. Všechny jsou typu Boolean, takže zadáváme pouze hodnoty TRUE (pro zapnutí elementu) či FALSE (pro jeho vypnutí).

Příklad číslo 29 - Přizpůsobení ovládacích prvků mapy v JS - kód a ukázka [11]

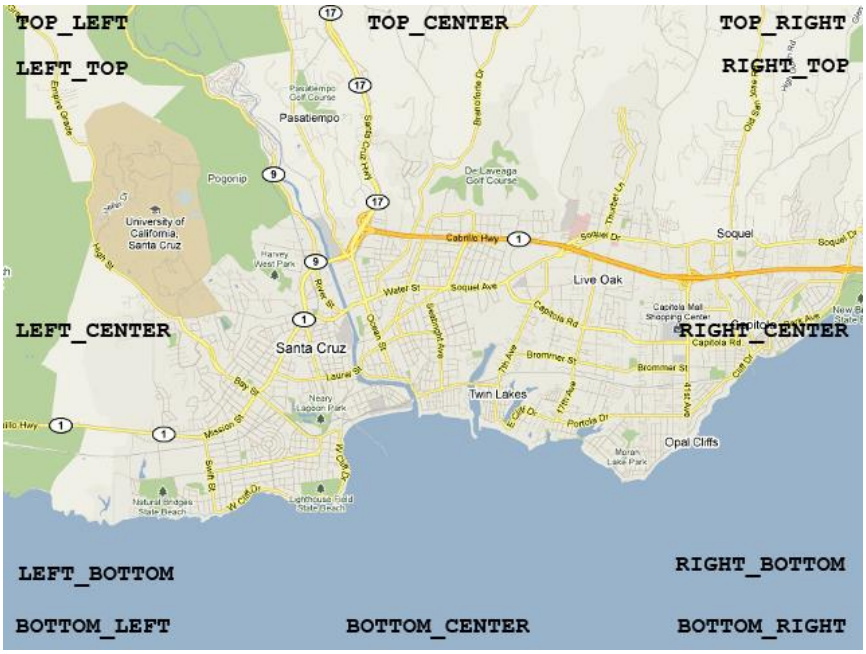
```
disableDefaultUI:true ,
panControl:true,
zoomControl:true,
mapTypeControl:false,
scaleControl:true,
streetViewControl:false,
overviewMapControl:true,
rotateControl:true
```



Mimo tyto úpravy lze také zasáhnout do vlastností jednotlivých položek a upravit například jejich velikost (pouze u ZOOM) nebo jejich rozložení na mapě, kde se budou zobrazovat. Nastavení tlačítka Zoom se provádí pomocí příkazu google.maps.ZoomControlStyle. za který se přidává typ zobrazení. Může být použito SMALL, LARGE, nebo DEFAULT.

Nastavení pozice se dělá pomocí příkazu google.maps.ControlPosition. kde následuje string řetězec s polohou (viz obrázek).

Příklad číslo 30 - Možnosti umístění ovládacích prvků na mapě - ukázka [3]



Výsledný zápis JS by měl, pro zobrazení zmenšeného ZOOM a jeho přesunutí doprostřed vrchní části, vypadat následovně.

Příklad číslo 31 - Umístění zmenšeného Zoomu na TOP_CENTER - kód

```
zoomControlOptions: {
    style:google.maps.ZoomControlStyle.SMALL,
    position: google.maps.ControlPosition.TOP_CENTER
     },
```

3.4.5 Značky

Značka vytváří zvýraznění místa na mapě. Samotná značka může být zobrazena mnoha způsoby. Formou klasického „špendlíku“ na určitém místě, lineárním vyznačením či vyznačením celé oblasti na místě pomocí kruhu nebo n-úhelníkového útvaru.

U jednotlivých značek je pak možno nastavit jejich vlastní vzhled, ať už se jedná o barevnou úpravu nebo vytvoření úplně nového obrázku (u klasických značek). Jednotlivým vyznačením pak lze přidat jakýkoliv popisek jak textový, tak obrazový. Velkou výhodou oproti statickým mapám je možnost přiřazení události „click“, jenž umožní po kliknutí na značku provést určitou akci (například click na marker způsobí přiblížení do dané oblasti a vycentrování mapy na marker).

3.4.5.1 Marker

Vytvoření klasické značky se provádí pomocí příkazu google.maps.Marker a určují se u ní dvě základní eventuality. První možností je pozice (POSITION), která určuje, kde bude značka zobrazena. Tuto položku je nutné vyplnit. Další možností je map (MAP), která říká, ke které mapě bude daná značka přiřazena. Položka sice povinná není, ale pokud nebude uvedena je potřeba na konci skriptu volat funkci Jmeno_znacky.setMap(Mapa_kam_chceme_pridat_znacku), která k dané mapě značku přidá. Pokud by nebyla uvedena ani jedno možnost, značka se nebude zobrazovat.

Příklad číslo 32 - Nastavení pozice marker - kód

```
var znacka = new google.maps.Marker({
    position: LatLng
});

znacka.setMap(map);
```

V případě odstranění značky z mapy, se využije opět funkce setMap(), avšak tentokrát ve tvaru funkce Jmeno_znacky.setMap(NULL). Použitím tohoto příkazu se značka nesmaže, ale nebude se pouze nadále zobrazovat na mapě.

Další výhoda oproti statickým mapám se skrývá v grafickém zpracování, přesněji animaci značek. Funkčnost se neovlivní, ale zlepší se estetický dojem. Pro animaci se u vlastností značky přidává možnost “ANIMATION“, které lze přiřadit jednu ze dvou možností.

První je funkce “DROP“, kdy značka „přiletí“ z horní části mapy. Samotná akce se provede pouze jednou a to při prvním načtení značky. Pak se animace stává neaktivní.

Další možností je “BOUNCE“, která vytvoří efekt „skákání“ značky na místě.

Příklad číslo 33 - Nastavení animace k marker - kód

```
var znacka = new google.maps.Marker({
    position: LatLng,
    animation: google.maps.Animation.BOUNCE
});

znacka.setMap(map);
```

3.4.5.2 Polyline

Funkce "POLYLINES" nalezne uplatnění při propojování více míst. Vytvoří mezi těmito místy lineární spojnici, která může procházet přes x bodů. Lze nastavit celkem tři vlastnosti.

- strokeColor – umožňuje nastavit barvu zobrazované čáry. Barva se zadává pomocí hexadecimálního formátu (#FF0000). Volba přednastavených barev u polyline nefunguje.
- strokeOpacity – určuje sílu barevného krytí čáry. Zadávají se hodnoty od 0.0 až po 1.0, s tím že čím menší hodnota, tím je úroveň krytí menší.
- strokeWeight – udává tloušťku spojovací linky. Hodnota této proměnné se udává v pixelech.

Na mapu spojnici přidáme opět použitím funkce setMap() a odebereme stejným způsobem jako markers.

Příklad číslo 34 - Nastavení spojnice přes 3 body bez uzavření - kód

```
var Cesta = [
    new google.maps.LatLng(-22.906847,-43.172896),
    new google.maps.LatLng(41.902783,12.496366),
    new google.maps.LatLng(50.075538,14.437800)
    ];

var Let = new google.maps.Polyline({
 path: Cesta,
 strokeColor:"#FF0000",
 strokeOpacity:0.5,
 strokeWeight:7
});
```

3.4.5.3 Polygon

Oproti polylines představuje troj a více úhelník, který vytvoří uzavřenou oblast. Uzavřená oblast pak obsahuje výplň. U polygonů lze nadefinovat stejné vlastnosti jako u polylines co se týče čáry, ale navíc obsahují prvky, které ovlivňují barvu a intenzitu výplně mezi body.

- fillColor – nastavuje, jakou barvu bude mít výplň mezi zvolenými body. Barva se zadává opět pomocí hexadecimálního kódu.
- fillOpacity – nabývá stejných hodnot jako strokeOpacity, ale určuje intenzitu výplně.

Samotný obrazec se vytvoří jako array pole obsahující LatLng souřadnic jednotlivých bodů, které budou tvořit vrcholy objektu. Pro uzavření objektu není potřeba znovu uvádět první souřadnici, ale uzavření se provede automaticky propojením poslední a první souřadnice.

Příklad číslo 35 - Nastavení spojnice tří bodů s uzavřením - kód

```
var Trojuhelnik = [
    new google.maps.LatLng(48.856614,2.352222),
    new google.maps.LatLng(41.902783,12.496366),
    new google.maps.LatLng(50.075538,14.437800)
    ];

var Oblast = new google.maps.Polygon({
 path: Trojuhelnik,
 strokeColor:"#FF0000",
 strokeOpacity:0.5,
 strokeWeight:3,
 fillColor: '#0000FF',
 fillOpacity: 0.2
});

Oblast.setMap(map);
```

Odebrání zůstává stejné jak u předchozích kapitol.

3.4.5.4 Circle

Je to specifický typ polygonu, u kterého je možné nastavit barvu, tloušťku okraje a průhlednost barvy, která se zobrazuje. Následně pak barvu a intenzitu výplně. Veškeré barvy se zadávají pomocí hexadecimální kombinace.

Dalšími prvky, které se musí zadat je LatLng (pomocí vlastnosti center) a rádius kruhu (poloměr).

Příklad číslo 36 - Nastavení kruhu kolem zvolené lokace - kód

```
var mesto = new google.maps.Circle({
 center:LanLng,
 radius:10000,
 strokeColor:"#FF0000",
 strokeOpacity:0.6,
 strokeWeight: 4,
 fillColor: "#0000FF",
 fillOpacity: 0.5
});
```

3.5 Porovnání s mapovým API Seznam

Průkopníkem, hlavním autorem a v současnosti také nejvýznamnějším hráčem na trhu poskytování webových a mapových služeb je společnost Google. Dá se říct, že od roku 2005 Google postupně pokryl mapovými službami většinu států světa.

V podmínkách České republiky na filozofii map od Google navazuje společnost Seznam. Velké množství nabízených funkcí je inspirováno přímo Googlem, takže fungují spolehlivě. Stejně jako u Google je pro používání samotného API a přístup ke službám nutné vlastnit účet spojený s unikátním vygenerovaným klíčem. Seznam mapové služby zpočátku umožnil používat pouze pro nekomerční využití, za což byl kritizován. Proto Seznam své podmínky pro používání API změnil.

Zásadně se Seznam oproti Google odlišuje v poskytnutí velké rozmanitosti map pro Českou republiku. Ostatní funkcionality jsou více méně srovnatelné.

Co se týká příkazů v javascriptu se Seznam odlišuje pouze formou zápisu, ale funkčnost zůstává velmi podobná.

Z nabídky knihoven využívá většina uživatelů maximálně 10 až 20% veškerých nabízených funkcí. V tomto rozsahu použití obě firmy kvalitou nabídky excelují a určitě všechny své uživatele dostatečně uspokojí.

Samotné výhody Seznamu jsou patrné až při využití nabízených podkladů pro Českou republiku a to díky široké nabídce a kvalitě map. Naopak Google dominuje v globálním měřítku, kde již lze docenit všechny jeho nabízené výhody.

Co naopak od Seznamu odrazuje je komunikace jednotlivých funkcí. Zatímco Google využívá ke svému Geocoding API moderní návratovou hodnotu ve tvaru JSON, která je schopna zobrazit i podrobné územní členění, Seznam využívá stále metodu XML.

Při hodnocení zbývajících 80% funkcí se ukáže, že mnoho funkcí Google je výrazně lépe propracovaných a podložených kvalitnější dokumentací. Například je to práce s liniemi a křivkami kde Google používá zmiňovaný GeoJSON a nabízí mnohem větší množství podporovaných formátů. [13] [14]

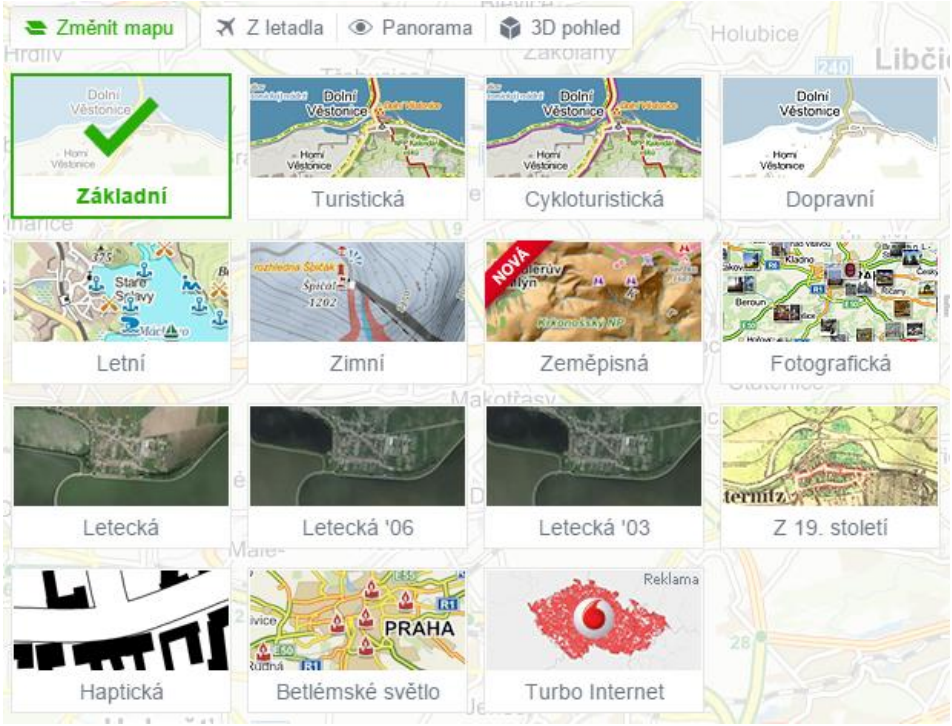
Další nevýhodou Seznamu je určitě jeho méně rozsáhlá dokumentace ke všem funkcím a vlastnostem. Naopak tím, že je Google používaný celosvětově, existuje mnohem více odborných článků a rad, které umožňují snadnější práci s odladěním chyb a případných nefunkčností.

Významným uživatelským nedostatkem Seznamu je, že umožňuje implementaci statických map pouze za použití javascriptu. Pro uživatele neznalé tohoto jazyka Google nabízí funkci statických map s jednoduchým využitím URL linku.

3.5.1 Mapy pro ČR

Jak bylo výše zmíněno, vyniká Seznam v prostředí České republiky v propracovanosti a velké škále map. Zatímco Google nabízí pouze klasickou mapu, u Seznamu je možno využít nepřeberné množství nabídky map. V tomto rozsáhlém výběru lze objevit například mapy cyklistické a turistické, dále pak mapu zimní a letní, či dokonce výběr leteckých map z různých let. Dokonce existuje možnost zobrazit mapu z 19. století nebo typ mapy „Haptická“, která slouží pro lidi se zrakovým postižením.

Příklad číslo 37 - Ukázka nabízených map od Seznam [15]



3.5.2 Panoramata

K významným událostem pro nabídku Seznamu patřilo zveřejnění vlastního produktu „Panoramata“, jehož funkci do té doby plnily Street view od Google. Do „Panoramat“ byla vkládána velká očekávání. Zástupci české firmy vsadili na jejich dokonalé zpracování a kvalitu. Výsledkem mělo být kvalitnější zpracování oproti světovému gigantu, který mapoval cca každých 20 metrů. Seznam se rozhodl mapovat každé 3 metry, aby dosáhl lepšího výsledku. Výsledek těmto očekáváním určitě neodpovídal. Snímky nevynikly ani ostrostí a fakt, že mapování probíhalo téměř za jakéhokoliv počasí, také kvalitě nepřidal. Výsledkem jsou panoramata, která mnohdy vyznívají pochmurně. Díky tomu jsou některé úseky tmavé a špatně osvětlené.

Vedení Seznamu přiznalo, že výsledek nesplnil jejich očekávání a většina úseků bude přesnímána za lepšího počasí a jediné části kde zůstanou současné snímky, budou silnice nižších tříd.

Příklad číslo 38 - Porovnání Streetview a Panoramat - ukázka [11] [15]

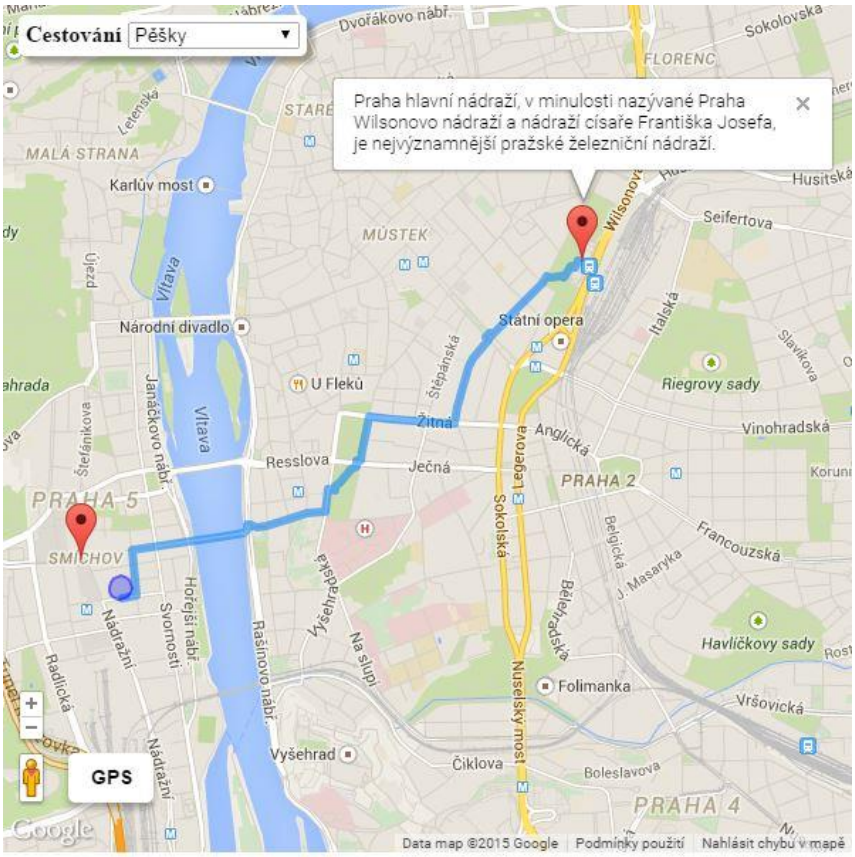
4 Praktická část

V rámci praktické části je předveden program za využití zmiňovaných funkcí s detailnějším využíváním nabídky prvků, které jsou nejrůzněji adaptovány na konkrétní problém. V příkladu je využíváno i pokročilejších funkcí, ke kterým je možno zařadit například tvoření eventů, využívání současné pozice nebo vytváření a nastavování nových tlačítek.

Vytvořený program obsahuje zpracování mapy s umístěnými ukázkovými značkami s nastavenou animací DROP a BOUCE a jejich vlastní popisek.

Dále mapa obsahuje nově vytvořené tlačítko a nabídku s výběrem dopravy. Kliknutím na konkrétní značku se ukáže nejkratší spojení ze současné polohy k danému marker s možností volby způsobu dopravy.

Příklad číslo 39 - Vykreslení trasy od GPS polohy k marker se zobrazeným popiskem - ukázka [11]



4.1 Popis kódu

V první části jsou nadefinované markers ve formátu TITLE, LAT, LNG a testu infowindow. Všechny jsou definovány v rámci pole.

Příklad číslo 40 - Nastavení seznamu značek a informací o nich – kód praktická [16] [17] [18] [19]

```
var znackySeznam = [
    ['Hlavní nádraží, Praha', 50.083505, 14.434138,"Praha hlavní \n\
    nádraží, v minulosti nazývané Praha Wilsonovo nádraží a nádraží \n\
    císaře Františka Josefa, je nejvýznamnější pražské železniční \n\
    nádraží."],

    ['Anděl, Praha', 50.071944, 14.403889,"Anděl je velmi významná \n\
    levobřežní dopravní křižovatka v Praze, faktické centrum zdejší \n\
    městské čtvrti Smíchov. "],

    ['ČZU, Praha', 50.131026, 14.373271,"Česká zemědělská univerzita v \n\
    Praze je veřejná vysoká škola univerzitního typu se sídlem v \n\
    Praze-Suchdole. Založena byla v roce 1952 a skládá se z šesti fakult \n\
    a jednoho institutu. Do roku 1995 nesla název Vysoká škola \n\
    zemědělská v Praze."]
];
```

Samotná úvodní deklarace nezaručí přidání značek na mapu. Přidání se provede až v rámci inicializace mapy pomocí FOR cyklu, pomocí kterého se každému marker nadefinuje řada vlastností. Proměnná „i“ se čísluje od 0, a pak se již vybírá pouze pozice v rámci předchozí deklarace. V příkladu se jedná o proměnnou Seznam a LatLng. Dále je vidět nastavení vlastností Marker a to pozici na mapě, animaci „příletu na mapu“, přiřazení marker k mapě. Následují dvě dodatečně definované proměnné. Jednou z nich je TITLE, který se zobrazí v případě najití kurzoru na marku a následující POPIS, který se zobrazuje do infowindow.

Příklad číslo 41 - Nastavení vlastností marker - kód praktická

```
for (var i in znackySeznam) {
    var Seznam = znackySeznam[i];
    var LatLng = new google.maps.LatLng(Seznam[1], Seznam[2]);
    var marker = new google.maps.Marker({
    position: LatLng,
    animation: google.maps.Animation.DROP,
    map: map,
    title: Seznam[0],
    popis: Seznam[3]
});
```

Samotné nadefinování a vytvoření infowindow se prování v rámci tohoto cyklu pro každý marker zvlášť. K tomuto účelu slouží předdefinovaná proměnná THIS, která zaručí,

že se deklarace provede pro všechny markers ze seznamu. Jak je vidět, pomocí „.“ lze provázat nastavenou proměnou POPIS a tento marker.

Příklad číslo 42 - Nastavení infoWindow k marker - kód praktická [20]

```
    google.maps.event.addListener(marker, 'click', function() {
    infowindow.setContent(this.popis);
    infowindow.open(map, this);
});
```

Další funkcí u značky je nastavení animace BOUNCE, která se vytvoří stejným způsobem jako u předchozí funkce. Při kliknutí na marker, se spustí animace s časovým omezením a v případě opětovného kliknutí, ještě před vypršením animace, se animace opět zruší. Doba jedné animace je dlouhá 700 ms a samotný stopAnimation je nastaven na čas 4200 ms.

Příklad číslo 43 - Nastavení animace BOUNCE k marker - kód praktická

```
google.maps.event.addListener(marker, 'click', function (){
if (this.getAnimation() !== null) {
    this.setAnimation(null);
} else {
    this.setAnimation(google.maps.Animation.BOUNCE);
    stopAnimation(this);
}
});

function stopAnimation(marker) {
    setTimeout(function () {
        marker.setAnimation(null);
    }, 4200);
}
```

Před poslední deklarovanou funkcí u marker je potřeba nejprve zdůraznit funkci, kterou následně využívá a tou je GPS poloha. Pro tuto funkci se využívá navigator.geolocation, která určuje, zda je možné GPS polohu získat či nikoliv. Následuje pak přiřazení Latitude a Longitude souřadnic do zvolené proměnné a nastavení vycentrování mapy na tuto pozici. V ukázce je v rámci GPS nadefinovaný modrý CIRCLE, který se zobrazí kolem stávající polohy. V poslední části se pomocí handleNoGeolocation zaznamenává, zda vše proběhlo v pořádku či nikoliv.

Příklad číslo 44 - Nastavení GPS polohy a circle kolem ní - kód praktická [21]

```
if(navigator.geolocation) {
   navigator.geolocation.getCurrentPosition(function(position) {
   GPS = new google.maps.LatLng(position.coords.latitude,
                                position.coords.longitude);

   map.setCenter(GPS);
   var GPSkruh = new google.maps.Circle({
       center:GPS,
       radius: 50,
       strokeColor: '#0000FF',
       strokeOpacity: 0.5,
       strokeWeight: 2,
       fillColor: '#0000FF',
       fillOpacity: 0.3,
       map: map
    });

    }, function() {
       handleNoGeolocation(true);
    });
   } else {
       handleNoGeolocation(false);
    }
```

Nyní se lze vrátit k poslední funkci obsažené u markers a tou je vykreslení cesty ze současné polohy k marker po kliknutí na ni. V první části se v rámci HTML musí nadefinovat zobrazovaný panel a přednastavit mu hodnoty, kterých může nabývat.

Příklad číslo 45 - Deklarace hodnot u výběru dopravního prostředku v HTML - kód praktická

```
<select id="mode" onchange="calcRoute();">
  <option value="DRIVING">Autem</option>
  <option value="WALKING">Pěšky</option>
  <option value="TRANSIT">Veřejná doprava</option>
</select>
```

Následně se pak vybraná položka importuje do proměnné selectedMode. Hodnoty u value jsou přímo deklarovány od Google. Po získání hodnoty obsahující způsob dopravy následuje request, který udává startovací pozici (origin), konečnou pozici (destination) a typ dopravního prostředku, který se získá právě z proměnné selectedMode. Následná funkce pak vyhodnotí, jestli zadaný požadavek na trasu je validní a v případě, že ano, vrátí se status OK. V tomto případě se zašle trasa, která se zobrazí na mapě.

Příklad číslo 46 - Vytvoření cesty pomocí zvoleného dopravního prostředku - kód praktická

```
google.maps.event.addListener(marker, 'click', function() {
var selectedMode = document.getElementById('mode').value;
var request = {
    origin: GPS,
    destination: this.position,
    travelMode: google.maps.TravelMode[selectedMode]
};
directionsService.route(request, function(response, status) {
if (status == google.maps.DirectionsStatus.OK) {
    directionsDisplay.setDirections(response);
}
}); });
```

Poslední potřebnou částí je pak přiřazení trasy k definované mapě. K tomu lze použít klasicky setMap(map). V ukázce je zvoleno, aby se nezobrazovaly značky na začátku a konci trasy. Důvodem je zabránění vzniku duplicit pod aktuálními značkami při vykreslení trasy.

Příklad číslo 47 - Přidání cesty na mapu a vypnutí počáteční a koncové značky - kód praktická

```
directionsDisplay = new google.maps.DirectionsRenderer();
directionsDisplay.setMap(map);
directionsDisplay.setOptions( { suppressMarkers: true } );
```

Posledním krokem je vytvoření vlastního tlačítka. Samotné formátování se provádí pomocí funkcí známých z CSS. Je třeba nastavit vše. Od rámečku, odsazení až po velikost zobrazovaného tlačítka a nastavení textu, který se bude v tlačítku zobrazovat. Lze taky nastavit stín tlačítka pro vytvoření plastičtějšího efektu. Na závěr je třeba nadefinovat jeho funkci po kliknutí. V tomto případě se jedná o návrat na GPS polohu a nastavení přiblížení na hodnotu 13.

Příklad číslo 48 - Nastavení vlastního tlačítka a následné přiřazení funkce [21]

```
function HomeControl(controlDiv, map) {
      controlDiv.style.padding = '35px';
      var controlUI = document.createElement('div');
      controlUI.style.backgroundColor = 'white';
      controlUI.style.borderRadius = '5px';
      controlUI.style.boxShadow = '0 4px 6px rgba(0,0,0,.3)';
      controlUI.style.cursor = 'pointer';
      controlUI.style.marginLeft = '-60px';
      controlUI.style.cursor = 'pointer';
      controlUI.style.textAlign = 'center';
      controlUI.title = 'Zpátky na Vaši polohu';
      controlDiv.appendChild(controlUI);
      var controlText = document.createElement('div');
      controlText.style.fontFamily='Arial';
      controlText.style.lineHeight = '35px';
      controlText.style.fontSize='14px';
      controlText.style.paddingLeft = '5px';
      controlText.style.paddingRight = '5px';
      controlText.innerHTML = '<b>GPS<b>';
      controlUI.appendChild(controlText);

      google.maps.event.addDomListener(controlUI, 'click', function() {
        map.setCenter(GPS),
        map.setZoom(13);
      });
    }
```

Samotné umístění tlačítka se vytváří přímo v rámci inicializace. Ostatní prvky už byly popisovány v předchozích ukázkách. Celý kód je umístěn do příloh, včetně celého HTML kódu.

4.2 Problémy a jejich řešení

Samotná tvorba a funkčnost kódu se samozřejmě neobešla bez problémů a jejich následného řešení tak, aby vadné části neovlivnily funkčnost následujících částí kódu.

První problém nastal hned při nastavování značek, kdy problém nebyl s nastavením vlastností, ale s přiřazováním jednotlivých funkcí. První varianty obsahovaly funkce ve tvaru, který využíval přímo proměnnou marker.

```
google.maps.event.addListener(marker, 'click', function() {
        infowindow.setContent(marker.popis);
        infowindow.open(map, marker);
    });
```

Veškeré funkce se přiřadily pouze až do poslední nadefinované značky a ostatní zůstaly pouze se základním nastavením pozice, animace DROP a titulku.

Prvním možným způsobem řešení bylo nadefinování indexu k markers. Tuto variantu se v kombinaci s ostatními funkcionalitami nepodařilo vyřešit. Po následném studiu podkladů se ukázalo jako výhodné použití proměnné „this“, která umožnila nadefinovat sadu funkcí přesně pro určitý marker. Následně pak již stačilo funkci adaptovat na pozice, které nastavují vlastnosti a nahradit proměnou marker právě zvolenou this.

```
google.maps.event.addListener(marker, 'click', function() {
        infowindow.setContent(this.popis);
        infowindow.open(map, this);
    });
```

Tento postup umožnil vyřešit i následující problémy s nastavováním ostatních funkcí, které byly přiřazovány.

Další problém, který nastal, byl se zvýrazňující animací BOUNCE. V původní verzi programu bylo uvažováno s tím, že se animace nebude zastavovat. Po následné zkoušce, kdy všechny značky měly spuštěnu animaci, neexistoval způsob jak je zastavit. Proto byla přidána podmínka na zastavení animace pro případ, že již jiná animace na daném marker běží.

```
google.maps.event.addListener(marker, 'click', function (){
        if (this.getAnimation() !== null) {
           this.setAnimation(null);
        } else {
           this.setAnimation(google.maps.Animation.BOUNCE);
        }
        });
```

Tento způsob se stal také nepraktickým po přidání cesty. Díky němu nastal problém z neustálého přeplánovávání cesty při vypnutí již spuštěných animací. Proto bylo nutno nalézt jiné řešení.

Okamžitě se objevila možnost přidání funkce, která by zastavila všechny probíhající animace před tím, než by se spustila animace, na jiném marker. Toto elegantní řešení ovšem muselo být zamítnuto a to z důvodu zvolení neindexovaných markers, především z důvodu, že by se kód musel od začátku předělat kvůli animaci.

Na závěr byla zvolena kompromisní varianta, a to ve formě nastavení časovače, který by animaci zastavil po určitém čase.

```
google.maps.event.addListener(marker, 'click', function (){
        if (this.getAnimation() !== null) {
           this.setAnimation(null);
        } else {
           this.setAnimation(google.maps.Animation.BOUNCE);
           setTimeout(function(){ this.setAnimation(null); }, 3000);
        }
        });
```

Bohužel ani toto jednoduché řešení nakonec nefungovalo a nastala otázka proč? Bylo třeba se obrátit na diskuzní fóra [22] s vyhledáním řešení. Po prostudování velkého množství nefunkčních rad se objevilo řešení. V rámci příspěvku byl navržen postup na vytvoření funkce setTimeout mimo nastavení animace, což se ukázalo jako účinné řešení tohoto problému.

```
function stopAnimation(marker) {
   setTimeout(function () {
      marker.setAnimation(null);
   }, 3000);
}
```

Posledním detailem, který bylo nutno dořešit, bylo zjištění doby trvání jednotlivých animací tak, aby se nastavil setTimeout na správnou hodnotu v ms a aby animace neskončily předčasně. Z technické dokumentace bylo zjištěno, že animace má trvání 700 ms a nebylo problémem následně hodnotu přenastavit.

Další problém byl spojený s funkcí Circle, která slouží na vyznačení GPS polohy. Její původní umístění přímo v rámci inicializace se ukázalo jako nepraktické, jelikož se do něj nedokázala propsat souřadnice uložená v proměnné GPS. V reakci na to se vůbec nezobrazovala. Po otestování na pevné pozici, zda se nejedná o špatné nastavení, se potvrdilo, že problém je opravdu s GPS. Po několika pokusech, které skončily nezdarem, kdy se zvýrazňující kruh nezobrazil, byla vyzkoušena možnost přesunutí celé proměnné přímo do nadefinování GPS. Po přesunutí se již proměnná začala standardně propisovat přímo do nastavení Center a problém byl odstraněn.

```
GPS = new google.maps.LatLng(position.coords.latitude,
                             position.coords.longitude);
    map.setCenter(GPS);
    var GPSkruh = new google.maps.Circle({
      center:GPS,
      radius: 50,
      ...
     });
```

Posledním problém, který se však nepovedlo vyřešit, byl také spojen s GPS polohou. Z neznámých příčin nastal problém, kdy internetový prohlížeč přestal být schopen zobrazovat současnou polohu zjišťovanou přes GPS. Problém nenastal pouze u programu, ale také u ukázkových příkladů přímo na Google Developers. Nepomohlo ani následné vyčištění souborů cookies. Další možnost v podobě doinstalování dalších prohlížečů také problém nevyřešilo. Jediný prohlížeč, který byl schopen zobrazit získávanou polohu, byl IE. Ten se ale ukázal pro následné napojení GPS na ostatní funkce zcela nepoužitelný vzhledem k tomu, že nebyl schopen upřesnit polohu přesněji než pouze na Českou republiku.

Dalším krokem k vyřešení tohoto problému bylo otestovat, zda problém nastává i na jiných zařízeních v rámci sítě. Bohužel toto ověření ukázalo, že oba testované nootebooky nemají sebemenší problém se získáním polohy v rámci tohoto kódu.

Poslední možnost byla obrátit se na online diskuze a přímo oslovit programátory. I přesto zůstal problém nadále nevyřešen, tak bylo nezbytné dodělat propojení GPS a kódu na jiném zařízení.

5 Závěr

V práci byly přehledně shrnuty základní funkce, které umožňují vytvářet vlastní funkční mapové API. Byl vytvořen podrobný přehled jednotlivých funkcí, které mohou být využity, se zdůrazněním povinných položek u jednotlivých prvků statických i javascriptových map.

Celkové zpracování slouží především k pochopení a pro přehled nejnovější funkcí a jejich aktualizací je potřeba navštívit přímo technickou dokumentaci na Google Developers.

V rámci této bakalářské práce bylo provedeno také porovnání mapových API Google a Seznam, především jednotlivých mapových podkladů, které jsou uvedenými firmami nabízeny a jejich základní funkčnost. Na základě tohoto srovnání se ukázalo, že i přes vlastní kvalitní mapové podklady Seznamu pro Českou republiku, zůstává Google díky nabízeným funkcím mapového API stále všeobecně využitelnější i pro českého uživatele.

V rámci běžněji využívaného celosvětového API byl zhotoven praktický příklad, jako ukázka vybraných funkcí, které javascript od Google nabízí. Jedná se na jedné straně o služby, jež statické mapy neumožňují, tak i o nabídku nadstandardních funkcí javascriptu. V praktickém příkladu je použita funkce současné polohy, její využití při plánování optimálních tras, kdy je možno pomocí eventů jednoduše najít cestu k vyznačenému bodu, ať už se jedná o bod zájmu nebo konkrétní adresu. Následuje zapracování těchto funkcí k markers, které jsou na ukázkové mapě vytvořeny a využití současné polohy, pro nově vytvořené tlačítko s funkcí vycentrování mapy na tuto polohu.

Při psaní praktického příkladu, byly postupně řešeny problémy, které se objevily u jednotlivých použitých funkcí. Většina problémů se objevila při propojování částí příkladu do větších celků. Problémy byly řešeny a následně odstraněny s přispěním studia odborných podkladů a za pomoci návrhů nalezených na diskuzních internetových fórech.

6 Citovaná literatura

1. **SVENNERBERG, Gabriel.** *Beginning Google Maps API 3.* New York : Apress, 2010. 978-143-0228-028.

2. **URAZ, Balkan a Alper DINCER.** *Google Maps JavaScript API Cookbook.* Birmingham : Packt Publishing, 2013. 1849698821.

3. **Google.** Google Maps JavaScript API v3. *Google Developers.* [Online] Google, 15. 1 2015. [Citace: 1. 2 2015.] https://developers.google.com/maps/documentation/javascript/.

4. —. Static maps. *Google Developers.* [Online] Google, 1. 10 2014. [Citace: 15. 10 2014.] https://developers.google.com/maps/documentation/staticmaps/.

5. **Petroutsos, Evangelos.** *Google Maps: Power Tools for Maximizing the API.* New York : McGraw-Hill, 2014. 0071823026 / 9780071823029.

6. **W3resource.** Google Maps API V 3 - Tutorial. *w3resource.* [Online] w3resource, 1. 2 2015. [Citace: 5. 2 2015.] http://www.w3resource.com/API/google-maps/index.php.

7. **W3schools.** Google API. *W3schools.com.* [Online] W3schools, 1. 2 2015. [Citace: 5. 2 2015.] http://www.w3schools.com/googleAPI/.

8. **FOLDOC.** Application Program Interface. *FOLDOC.* [Online] FOLDOC, 15. 2 1995. [Citace: 2014. 10 20.] http://foldoc.org/Application+Program+Interface.

9. **Pcmag.** Definition of:API. *Pcmag.* [Online] PCmag, 1. 10 2015. [Citace: 20. 10 2015.] http://www.pcmag.com/encyclopedia/term/37856/api.

10. **Webopedia.** API - application program interface. *Webopedia.* [Online] Webopedia, 1. 10 2015. [Citace: 20. 10 2015.] http://www.webopedia.com/TERM/A/API.html.

11. **Google.** Google Maps. *Google Maps.* [Online] Google, 1. 2 2015. [Citace: 2015. 2 5.] Mapové podklady Google. https://www.google.cz/maps.

12. **W3school.** Google Maps Controls. *w3school.com.* [Online] w3school, 1. 2 2015. [Citace: 1. 2 2015.] http://www.w3schools.com/googleapi/google_maps_controls.asp.

13. **Vyleťal, Martin.** Souboj mapových titánů. *Lupa.cz.* [Online] Lupa, 4. 3 2014. [Citace: 15. 1 2015.] http://www.lupa.cz/clanky/souboj-mapovych-titanu-vyrovnaji-se-nove-mapy-cz-mapam-googlu/.

14. **Seznam.** Mapy API. *Mapy.cz.* [Online] Seznam, 1. 2 2015. [Citace: 1. 3 2015.] http://api.mapy.cz/.

15. —. Mapy.cz. *Mapy.cz.* [Online] Seznam, 1. 2 2015. [Citace: 1. 2 2015.] http://www.mapy.cz/.

16. **Google.** Anděl (Praha). *Google.* [Online] Google, 20. 1 2015. [Citace: 1. 3 2015.] https://www.google.com/webhp?sourceid=chrome-instant&ion=1&espv=2&ie=UTF-8#q=praha%20and%C4%9Bl.

17. **Praha.eu.** Hlavní nádraží. *Praha.eu Portál hlavního města Prahy.* [Online] Praha.eu, 22. 2 2015. [Citace: 1. 3 2015.] http://www.praha.eu/jnp/cz/doprava/vlakova/nadrazi-praha_hlavni_nadrazi.html.

18. **ČZU.** Základní informace. *Česká zemědělská univerzita v Praze.* [Online] ČZU, 1. 3 2015. [Citace: 1. 3 2015.] http://www.czu.cz/cs/?r=298.

19. **Latlong.** Get Latitude Longitude. *latlong.net.* [Online] 1. 3 2015. [Citace: 1. 3 2015.] http://www.latlong.net/.

20. **Theoryapp.** Java. *Theoryapp.* [Online] Theoryapp, 1. 3 2015. [Citace: 1. 3 2015.] http://theoryapp.com/java/.

21. **W3schools.** CSS Reference. *W3schools.com.* [Online] W3schools, 1. 3 2015. [Citace: 1. 3 2015.] http://www.w3schools.com/cssref/default.asp.

22. **Stackoverflow.** stackoverflow - Questions. *Stackoverflow.* [Online] Stackoverflow, 1. 3 2015. [Citace: 1. 3 2015.] http://stackoverflow.com/questions/.

7 Seznam obrázků

8 Přílohy

HTML:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <style type="text/css">
      html, body, #map-canvas { height: 100%; margin: 0; padding: 0;}
          #panel {
        position: fixed;
        top: 10px;
        left: 12px;
        background-color: #fff;
        padding: 5px;
        border-radius: 5px;
        box-shadow: 5px 5px 5px rgba(0,0,0,.3);
      }
    </style>
    <script type="text/javascript"
src="https://maps.googleapis.com/maps/api/js?language=cs&key=API_KEY">
    </script>
    <script type="text/javascript" src="javascript.js"></script>
  </head>
  <body>
<div id="map-canvas" style="width: 600px; height: 600px;"></div>
<div id="panel">
    <b>Cestování </b>
    <select id="mode" onchange="calcRoute();">
      <option value="DRIVING">Autem</option>
      <option value="WALKING">Pěšky</option>
      <option value="TRANSIT">Veřejná doprava</option>
    </select>
    </div>
  </body>
</html>
```

JavaSript:

```
    var znackySeznam = [
       ['Hlavní nádraží, Praha', 50.083505, 14.434138,"Praha hlavní \n\
       nádraží, v minulosti nazývané Praha Wilsonovo nádraží a nádraží \n\
       císaře Františka Josefa, je nejvýznamnější pražské železniční \n\
       nádraží."],

       ['Anděl, Praha', 50.071944, 14.403889,"Anděl je velmi významná \n\
       levobřežní dopravní křižovatka v Praze, faktické centrum zdejší \n\
```

```
      městské čtvrti Smíchov. "],

      ['ČZU, Praha', 50.131026, 14.373271,"Česká zemědělská univerzita v \n\
      Praze je veřejná vysoká škola univerzitního typu se sídlem v \n\
      Praze–Suchdole. Založena byla v roce 1952 a skládá se z šesti fakult \n\
      a jednoho institutu. Do roku 1995 nesla název Vysoká škola \n\
      zemědělská v Praze."]
   ];

   var directionsDisplay, GPS;
   var directionsService = new google.maps.DirectionsService();


function HomeControl(controlDiv, map) {
    controlDiv.style.padding = '35px';
    var controlUI = document.createElement('div');
    controlUI.style.backgroundColor = 'white';
    controlUI.style.borderRadius = '5px';
    controlUI.style.boxShadow = '0 4px 6px rgba(0,0,0,.3)';
    controlUI.style.cursor = 'pointer';
    controlUI.style.marginLeft = '-60px';
    controlUI.style.cursor = 'pointer';
    controlUI.style.textAlign = 'center';
    controlUI.title = 'Zpátky na Vaši polohu';
    controlDiv.appendChild(controlUI);
    var controlText = document.createElement('div');
    controlText.style.fontFamily='Arial';
    controlText.style.lineHeight = '35px';
    controlText.style.fontSize='14px';
    controlText.style.paddingLeft = '5px';
    controlText.style.paddingRight = '5px';
    controlText.innerHTML = '<b>GPS<b>';
    controlUI.appendChild(controlText);

    google.maps.event.addDomListener(controlUI, 'click', function() {
      map.setCenter(GPS),
      map.setZoom(13);
    });
   }
function inicializace() {

    if(navigator.geolocation) {
      navigator.geolocation.getCurrentPosition(function(position) {
      GPS = new google.maps.LatLng(position.coords.latitude,
                                   position.coords.longitude);

      map.setCenter(GPS);
      var GPSkruh = new google.maps.Circle({
```

```
      center:GPS,
      radius: 50,
      strokeColor: '#0000FF',
      strokeOpacity: 0.5,
      strokeWeight: 2,
      fillColor: '#0000FF',
      fillOpacity: 0.3,
      map: map
    });

    }, function() {
      handleNoGeolocation(true);
    });
   } else {
      handleNoGeolocation(false);
    }
var optionsEdit = {
    zoom: 13,
    disableDefaultUI:true ,
    panControl:false,
    zoomControl:true,
    zoomControlOptions: {
        style:google.maps.ZoomControlStyle.SMALL,
        position: google.maps.ControlPosition.LEFT_BOTTOM
         },
    mapTypeControl:false,
    scaleControl:false,
    streetViewControl:true,
    streetViewOptions: {
        position: google.maps.ControlPosition.BOTTOM_LEFT
         },
    overviewMapControl:false,
    rotateControl:false,
    mapTypeId: google.maps.MapTypeId.ROADMAP
};
    var map = new google.maps.Map(document.getElementById('map-canvas'),
    optionsEdit);

    var infowindow = new google.maps.InfoWindow({
      maxWidth: 300
    });

    directionsDisplay = new google.maps.DirectionsRenderer();
    directionsDisplay.setMap(map);
    directionsDisplay.setOptions( { suppressMarkers: true } );

    var homeControlDiv = document.createElement('div');
    var homeControl = new HomeControl(homeControlDiv, map);
```

```
        map.controls[google.maps.ControlPosition.BOTTOM_LEFT].
        push(homeControlDiv);

        for (var i in znackySeznam) {
          var Seznam = znackySeznam[i];
          var LatLng = new google.maps.LatLng(Seznam[1], Seznam[2]);
          var marker = new google.maps.Marker({
          position: LatLng,
          animation: google.maps.Animation.DROP,
          map: map,
          title: Seznam[0],
          popis: Seznam[3]
        });
          google.maps.event.addListener(marker, 'click', function() {
          infowindow.setContent(this.popis);
          infowindow.open(map, this);
        });
          google.maps.event.addListener(marker, 'click', function() {
          var selectedMode = document.getElementById('mode').value;
          var request = {
             origin: GPS,
             destination: this.position,
             travelMode: google.maps.TravelMode[selectedMode]
          };
          directionsService.route(request, function(response, status) {
          if (status == google.maps.DirectionsStatus.OK) {
             directionsDisplay.setDirections(response);
          }
          }); });

          google.maps.event.addListener(marker, 'click', function (){
          if (this.getAnimation() !== null) {
             this.setAnimation(null);
          } else {
             this.setAnimation(google.maps.Animation.BOUNCE);
             stopAnimation(this);
          }
          });
    }
}
function stopAnimation(marker) {
    setTimeout(function () {
       marker.setAnimation(null);
    }, 4200);
}
google.maps.event.addDomListener(window, 'load', inicializace);
```