

---

# GOOGLE MAPS API

## POPIS A POUŽITÍ

---

[X36SWT] Softwarové technologie,  
Katedra počítačů,  
Fakulta elektrotechnická,  
České vysoké učení technické v Praze

Bc. Ondřej Čermák  
[cermak.cz@gmail.com](mailto:cermak.cz@gmail.com)  
[ondrejcermak.info](mailto:ondrejcermak.info)

Březen 2010

## **Abstrakt**

Dokument pojednává o službě Google Maps a k ní přidruženém Google Maps API. Je zde popsáno, co je to a hlavně jak se Google Maps API používá. Zároveň jsou zde také popsány jednotlivé verze a jejich odlišnosti.

Naleznete zde instalaci, základní použití a popis jednotlivých možností, které toto API nabízí. Jednotlivé možnosti jsou popsány pouze stručně, pro další studium se doporučuje projít oficiální dokumentaci. Ke všem příkladům jsou uvedeny obrázky a odkazy do oficiální dokumentace.

Ve většině textu se předpokládá základní znalost JavaScriptu.

# Obsah

<b>1</b>	<b>Úvod</b>	<b>4</b>
1.1	Google Maps	4
<b>2</b>	<b>Obecně o Google Maps API</b>	<b>5</b>
2.1	Možnosti použití	5
2.2	Verze	5
2.3	Alternativy	6
2.3.1	Mapy API, Seznam.cz	6
2.3.2	Atlas AMapy API, Atlas.cz	6
2.3.3	OpenStreetMap API	6
<b>3</b>	<b>Static Maps API</b>	<b>8</b>
3.1	Omezení	8
3.2	Parametry	8
3.2.1	Poziční parametry	8
3.2.2	Mapové parametry	8
3.2.3	Funkční parametry	9
3.2.4	Funkční parametry	10
3.2.5	Příklady	10
<b>4</b>	<b>Google maps API V3</b>	<b>12</b>
4.1	Rozdíly	12
4.2	Novinky	12
4.2.1	Geolokace	12
4.3	Základy	13
4.3.1	Načtení API	13
4.3.2	Lokalizace	13
4.3.3	Práce se souřadnicemi	14
4.3.4	Vytvoření mapy	14
4.4	Pokročilé použití	15
4.4.1	Možnosti	16
4.4.2	MVC model	17
4.4.3	Událostní model	17
4.5	Vrstvy	18
4.5.1	Ovládací prvky	18
4.5.2	Markery	19

4.5.3	Informační okna . . . . .	20
4.5.4	Lomené čáry . . . . .	21
4.5.5	Polygonální oblasti . . . . .	22
4.5.6	Vlastní vrstvy . . . . .	23
4.6	Služby . . . . .	27
4.6.1	Geocoding . . . . .	27
4.6.2	Hledání cest . . . . .	27
4.7	Knihovny pro V3 . . . . .	27
<b>5</b>	<b>Google maps API V2</b>	<b>28</b>
5.1	Co je navíc oproti V3? . . . . .	28
5.2	Knihovny pro V2 . . . . .	28
<b>6</b>	<b>Google maps API pro Flash</b>	<b>29</b>
<b>7</b>	<b>Další API spojené s Google Maps</b>	<b>30</b>
7.1	Google Maps Data API . . . . .	30
7.2	Google Mapplets . . . . .	30

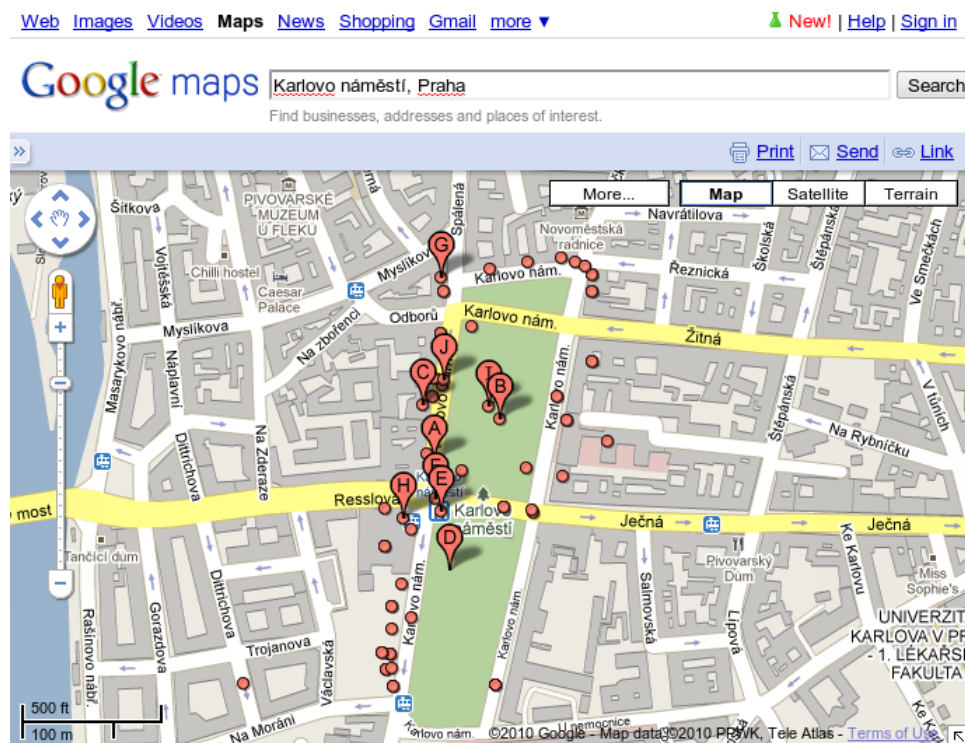
## Seznam obrázků

1	Služba Google maps . . . . .	4
2	Použití Static Maps API, <i>obrázek k příkladu 3.1</i> . . . . .	10
3	Pokročilé použití Static Maps API, <i>obrázek k příkladu 3.2</i> . .	11
4	Mapa vložená do webové stránky, <i>obrázek k příkladu 4.6</i> . .	16
5	Mapa s vloženými markery, <i>obrázek k příkladu 4.9</i> . . . . .	21
6	Mapa se zobrazeným overlayem, <i>obrázek k příkladu 4.13</i> . .	26

# 1 Úvod

## 1.1 Google Maps

Společnost Google nabízí mimo jiné vlastní mapovou službu, kterou naleznete na adrese <http://maps.google.com>. Tato služba poskytuje mnoho možností, které může uživatel využít. Jedná se hlavně o vyhledávání míst, prohlížení map s využitím různých mapových podkladů, vyhledávání cest atd.



Obrázek 1: Služba Google maps

Veřejnosti je však poskytováno také programové rozhraní<sup>[1]</sup> (dále jen API), díky kterému je možné využívat mapových služeb ve vlastních aplikacích. Toto API, jeho jednotlivé verze a jejich možnosti, je popsáno v následujícím textu.

## 2 Obecně o Google Maps API

Pomocí Google Maps API můžeme vložit okno s mapou do vlastní aplikace, ať už webové, či desktopové a využívat možnosti, které nám služba Google Maps poskytuje. Je tedy možné vytvářet vlastní mapové aplikace s dalšími rozšiřujícími možnostmi, nebo využít mapu pouze jako doplňkovou službu naší aplikace.

Google Maps API je vydáno ve více verzích o kterých se dozvíte v části [2.2 Verze](#). V tomto dokumentu bude nejvíce popsána aktuální JavaScriptová verze 3 popsaná v kapitole [4 Google maps API V3](#). Nicméně naleznete zde i zmínku o ostatních verzích a jejich možnostech.

### 2.1 Možnosti použití

Jak již bylo řečeno, Google Maps API můžeme využít na mnoho způsobů. Některé z těchto způsobů jsou popsány v této části.

**Jednoduché statické mapy** Tou nejjednodušší možností, jak využít API, je vložit do aplikace jednoduchou statickou mapu, tedy obrázek. K tomu slouží [Static Maps API](#) popsané v kapitole [3](#). Statické API nabízí vcelku pokročilé možnosti. Do mapy je možné vkládat markery, či vlastní ikony, kreslit lomené čáry a oblasti, určit si typ obrázku atd.

**Interaktivní mapové aplikace** Pokročilejší možností je použít API k tvorbě interaktivní mapové aplikace. K tomu slouží dynamické API, které je tvořené JavaScriptovou, případně Flash knihovnou. Výčet možností tohoto API je nepoměrně rozsáhlejší a protože dynamické API je vydáno ve třech verzích, možnosti jsou uvedeny vždy v příslušné kapitole.

**Velká databáze mapových podkladů a leteckých snímků** K našemu využití nám Google poskytuje velké množství mapových podkladů a leteckých snímků. Zároveň, pokud použijeme [Google maps API V2](#), můžeme využívat i služby Street View a jejích snímků přímo z ulice.

**Použití enginu pro vlastní mapové podklady** Poslední možností je použití mapového enginu pro vlastní mapové podklady, či velkoformátové obrázky. Hlavní výhodou je posouvání okna po mapě, či obrázku, a rychlé načítání díky rozřezání podkladu do dlaždic. Otázkou zůstává, zda není lepší k tomuto účelu (zobrazení velkých obrázků) využít některou z open-source implementací, jako například [OpenLayers](#).

### 2.2 Verze

Google Maps API je vydáno v několika verzích, které se od sebe liší. Podrobnější popis naleznete v příslušných kapitolách.

**Static Maps API** Statická verze API, slouží ke generování statických obrázků podle zadaných parametrů. Je popsána v kapitole [3 Static Maps API](#).

**Google Maps API V3** JavaScriptové API ve verzi 3. Od verze 2 se liší tím, že by mělo být optimalizované na rychlost a velikost knihoven. Zatím je však stále ve vývoji a nenabízí některé možnosti jako API verze 2. Je popsáno v kapitole [4 Google maps API V3](#).

**Google Maps API V2** JavaScriptové API ve verzi 2. Oproti verzi 3 nabízí více možností a existuje pro něj více knihoven, je ale větší a pomalejší. Je popsáno v kapitole [5 Google maps API V2](#).

**Google Maps API pro Flash** API vytvořené pro Flash. Nabízí zhruba stejné možnosti, jako V3. Je popsáno v kapitole [6 Google maps API pro Flash](#).

## 2.3 Alternativy

Společnost Google však není jedinou, která poskytuje mapovou službu s dostupným API. Pokud chceme vytvořit mapovou aplikaci, je vždy vhodné provést analýzu a rozhodnout se, které API bude našemu účelu vyhovovat nejvíce.

### 2.3.1 Mapy API, Seznam.cz

Společnost [Seznam.cz](#) také poskytuje veřejnou mapovou službu [mapy.cz](#) s využitelným API. Za nevýhodu lze považovat dostupnost pouze map Evropy, někdy to však nemusí vůbec vadit. Výhodou by mohly být lepší mapové podklady pro Českou republiku, na kterou jsou tyto mapy nejvíce zaměřené.

### 2.3.2 Atlas AMapy API, Atlas.cz

Další českou společností, která nabízí použití její mapové služby [amapy.cz](#) a přidruženého API je [Centrum holdings](#). Opět nabízí pouze mapy Evropy a další nevýhodou je nepříliš velká známost a oblíbenost této služby.

### 2.3.3 OpenStreetMap API

Posledním příkladem dostupného API je produkt [OpenStreetMap](#), který je zatím jedinou open-source alternativou k mapovým službám. Samotné mapové podklady jsou vytvářeny početnou komunitou, nicméně jejich přesnost je na vysoké úrovni a stále stoupá. Tato služba poskytuje podobné možnosti jako ostatní mapové služby, ale API je trochu odlišné. Ostatní služby mají

API většinou JavaScriptové, zde je však tzv. RESTful API, které poskytuje přístup k RAW mapovým datům.

Pokud bychom chtěli použít mapové podklady z OpenStreetMap ve vlastním okně s typickými vlastnostmi JavaScriptových API (posouvání mapy, markery atd.), je možné využít dalších open-source projektů jako je [OpenLayers](#) nebo [Mapstraction](#). Tyto projekty nabízí JavaScriptové knihovny, které umožňují zobrazení mapových podkladů z libovolného zdroje.

## 3 Static Maps API

Statické API slouží ke generování obrázků s mapou dle zadaných parametrů. Je to http rozhraní, na které posíláme dotazy, tedy http requesty, a dostává se nám odpovědi v podobě obrázků zadaného typu.

Využití nalezneme například v desktopové aplikaci, kde potřebujeme pouze zobrazit mapu a nemusíme s ní nijak hýbat. Některé dnešní fotoaparáty či mobilní telefony umí otagovat fotografii gps pozicí, kde byla vyfocena. Můžeme si tak představit aplikaci, kde se tato informace využije k zobrazení statické mapy, kde byla fotografie vytvořena.

### 3.1 Omezení

Oficiální dokumentace říká[7, Usage limits], že použití API je omezené na 1000 unikátních dotazů od jednoho uživatele za den. Jelikož se ale jedná o uživatele, tak by to pro vývojáře či provozovatele aplikace neměl být problém.

Zároveň je také omezena délka http dotazu na 2048 znaků. S tím se ale opět v praxi spíš nesetkáme.

### 3.2 Parametry

Rozhraní nalezneme na adrese <http://maps.google.com/maps/api/staticmap?parameters>, kde za **parameters** dosadíme naše vlastní parametry. Parametry jsou dle oficiální dokumentace [7, URL Parameters] následující.

#### 3.2.1 Poziční parametry

**center** (vyžadováno pokud nejsou přítomny markery) Pozice středu mapy. Zadává se v desetinném WGS84 [10] formátu jako dvojice hodnot zeměpisná délka a šířka oddělené čárkou (např. "50.076683,14.417794"). Nebo můžeme zadat adresu místa, které chceme zobrazit (např. "Karlovo+náměstí, Praha").

**zoom** (vyžadováno pokud nejsou přítomny markery) Určuje přiblížení mapy. Hodnoty jsou od **0** (je vidět celý svět) po **21**.

#### 3.2.2 Mapové parametry

**size** (vyžadováno) Určuje velikost vráceného obrázku v pixelech. Zadává se jako šířka×výška (např. "500x400").

**format** Určuje formát obrázku. Možné formáty jsou PNG, JPEG a GIF. Hodnoty, které je možné zadat jsou **png**, **png8**, **gif**, **jpg** a **jpg-baseline**. Pokud není uvedeno, použije se **png**.

**maptype** Určuje typ mapového podkladu. Možné hodnoty jsou **roadmap** (silniční mapa), **satellite** (satelitní snímky), **hybrid** (satelitní a silniční mapa) a **terrain** (terénní mapa).

**mobile** Určuje, zda bude mapa zobrazena na mobilním telefonu. Hodnota je **true** nebo **false**. Mapy na mobilní telefon mohou mít pozměněné podklady optimalizované pro tato zařízení.

**language** Určuje jazyk, kterým budou psány popisy na mapě.

### 3.2.3 Funkční parametry

**markers** Definuje jeden nebo více markerů (značek). Jednotlivé definice markerů se oddělují pomocí znaku roury (|). Markerům můžeme definovat styl zobrazení, pomocí parametrů:

**size** Velikost markeru. Možné hodnoty: **tiny**, **mid**, **small** a **normal**.

**color** Barva markeru. Možné hodnoty: 24-bitová barva, např. **0xFF00AA**, nebo předdefinovaná barva - **black**, **brown**, **green**, **purple**, **yellow**, **blue**, **gray**, **orange**, **red**, **white**

**label** Popis markeru. Možná hodnota je jedno písmeno abecedy.

Poté se určují souřadnice v desetinném WGS84 [10] formátu, nebo jako adresa. Celý zápis parametru **markers** pak může vypadat nějak takto:

```
markers=color:blue|label:F|50.076683,14.417794
```

**path** Definuje lomenou čáru pomocí dvou a více bodů. Souřadnice bodů se oddělují pomocí znaku roury (|). Opět můžeme definovat styl čáry pomocí parametrů:

**color** Barva čáry, stejně jako u markerů.

**weight** Tloušťka čáry v pixelech.

**fillcolor** Pomocí tohoto parametru říkáme, že se má oblast ohraničená čarou vyplnit danou barvou.

Souřadnice bodů se zapisují stejně jako u markerů.

**visible** Pomocí tohoto parametru můžeme specifikovat jednu nebo více lokací, které mají být na mapě viditelné. Nemusíme pak určovat parametr **center** a **zoom**

### 3.2.4 Funkční parametry

**sensor** (vyžadováno) Určuje, zda aplikace, která si vyžádala statickou mapu, používá GPS senzor ke zjištění pozice uživatele. Hodnota je **true** nebo **false**.

Pokud použijeme parametry **markers** nebo **path**, nemusíme specifikovat parametry **center** a **zoom** - ty se nastaví tak, aby byly vidět všechny markery, resp. celá čára.

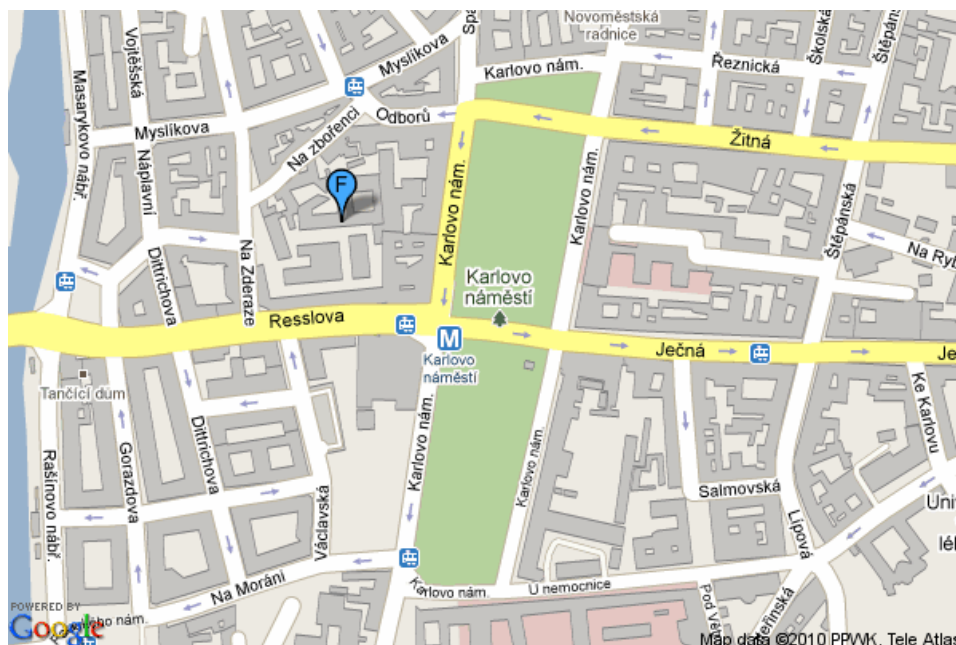
### 3.2.5 Příklady

---

*Příklad 3.1: Použití Static Maps API*

Celý dotaz může pak vypadat nějak takto:

```
http://maps.google.com/maps/api/staticmap?center=Karlovo+náměstí,  
Praha&zoom=16&size=600x400&maptype=roadmap&markers=color:blue|  
label:F|50.076683,14.417794&sensor=false
```



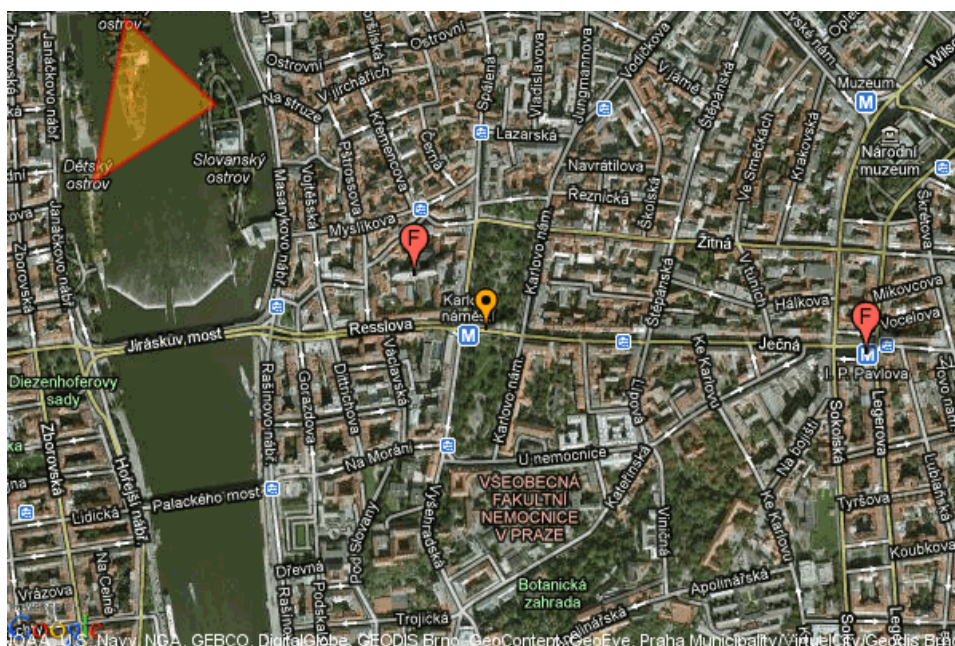
Obrázek 2: Použití Static Maps API, *obrázek k příkladu 3.1*

---

### Příklad 3.2: Pokročilé použití Static Maps API

Pokročilý dotaz může vypadat nějak takto:

```
http://maps.google.com/maps/api/staticmap?center=Karlovo+náměstí,  
Praha&zoom=15&size=600x400&maptype=hybrid&markers=color:red|  
label:F|50.076683,14.417794|I.P.Pavlova,Praha&markers=size:mid|  
color:orange|Karlovo+náměstí,Praha&path=weight:3|color:red|  
fillcolor:orange|Střelecký+ostrov|Slovanský+ostrov|Dětský+ostrov|  
Střelecký+ostrov&sensor=false
```



Obrázek 3: Pokročilé použití Static Maps API, *obrázek k příkladu 3.2*

## 4 Google maps API V3

Jedná se o JavaScriptové API, které poskytuje možnost vložit mapu do webové stránky a využívat všech možností klasických Google Maps, jako je například posouvání mapy v okně, přibližování, přepínání vrstev atd.

Verze 3 je nová a zatím je stále ve vývoji. Hlavním cílem této verze je optimalizace API na rychlost a velikost souborů, kvůli stále častějšímu používání API na mobilních zařízeních (Android, iPhone a další). Zatím však neposkytuje veškeré možnosti verze 2. Tato podpora je ale plánovaná.

### 4.1 Rozdíly

Hlavními rozdíly oproti druhé verzi je zrušení API klíče<sup>1</sup> a přejmenování tříd (resp. prototypů), které jsou nyní v `google.maps`. Třída se pak tedy jmenuje např. `google.maps.Marker`, oproti verzi 2, kde se jedná o `GMarker`.

### 4.2 Novinky

#### 4.2.1 Geolokace

Geolokace není ani tak novinka API V3, jako spíš HTML5. Jedná se o W3C Geolocation standard [9]. Jedná se o určení pozice uživatele s využitím GPS senzoru nebo síťové adresy. Je důležité poznamenat, že se jedná o API závislé na konkrétním přístroji a prohlížeči, protože ne všechny ho podporují.

Pro podrobnější popis a příklad je doporučeno podívat se do oficiální dokumentace [4, Geolocation]. Zde je uveden jen krátký příklad použití standardu.

---

*Příklad 4.1: Použití W3C Geolocation standardu*

Kód převzat z oficiální dokumentace [4, Geolocation].

```
1  if(navigator.geolocation) { //ověření podpory standardu
2      browserSupportFlag = true;
3      navigator.geolocation.getCurrentPosition(function(position) {
4          initialLocation = new google.maps.LatLng(
5                                  position.coords.latitude,
6                                  position.coords.longitude);
7          map.setCenter(initialLocation);
8      }, function() {
9          handleNoGeolocation(browserSupportFlag);
10     });
11 }
```

---

<sup>1</sup>API klíč slouží ve verzi 2 k autorizaci aplikace vůči Google Maps API.

Funkcí `navigator.geolocation.getCurrentPosition(function1, function2)` registrujeme dvě callback funkce. První se zavolá, pokud prohlížeč vrátí pozici uživatele. Druhá se zavolá v případě chyby. V příkladu jsou použity dvě anonymní funkce.

Zároveň se zde již používá objekt mapy a souřadnic, které budou popsány v následujícím textu.

## 4.3 Základy

V této části se dozvíme něco o základech použití Google Maps API. V první řadě musíme umět API načíst a poté vytvořit mapu ve webové stránce. K tomu je však nutné také zát něco o práci se souřadnicemi.

### 4.3.1 Načtení API

Tou nejzákladnější věcí vůbec, je načtení samotného API. To provedeme tak, že vložíme do stránky JavaScriptovou knihovnu pomocí tagu `script` s parametrem href nastaveným na URL s určitými parametry. Tag `script` se obvykle vkládá do hlavičky html dokumentu.

---

*Příklad 4.2: Načtení API*

Parametr `sensor` se nastavuje stejně, jako u statického API. Určuje, zda aplikace, která si vyžádala statickou mapu, používá GPS senzor ke zjištění pozice uživatele. Hodnota je `true` nebo `false`.

```
<script type="text/javascript"
  src="http://maps.google.com/maps/api/js?sensor=false">
</script>
```

### 4.3.2 Lokalizace

API nabízí možnost lokalizace a to jak jazykové, tak regionální. Tato možnost je přítomna již ve verzi 2.

**Jazyková** Jazyková lokalizace upravuje ovládání, texty a popisy v mapě, na základě jazykového kódu. Ten se specifikuje pomocí parametru `language` při vkládání knihovny do stránky. Čeština je podporovaná [2] a má jazykový kód `cs`.

---

*Příklad 4.3: Specifikace jazykové lokalizace*

Níže je uveden kód, který importuje API knihovnu s českou jazykovou lokalizací.

```
<script type="text/javascript" src="http://maps.google.com
/maps/api/js?sensor=false&language=cs">
</script>
```

**Regionální** API poskytuje různé mapové podklady a mění chování v závislosti na regionu. Pokud neurčíme jinak, bere se region podle domény, z jaké je API načítáno. Region využijeme hlavně pro geocoding<sup>2</sup>, kdy je nám přednostně nabízeno místo v daném regionu. Kódy regionů jsou určeny podle Unicode region identifikátorů [8], které odpovídají většinou ccTLD<sup>3</sup> dané země. Pro Českou republiku použijeme kód **cz**.

---

*Příklad 4.4: Specifikace regionální lokalizace*

Níže je uveden kód, který importuje API knihovnu s regionální lokalizací pro Českou republiku.

```
<script type="text/javascript"
src="http://maps.google.com/maps/api/js?sensor=false&region=CZ">
</script>
```

#### 4.3.3 Práce se souřadnicemi

Při práci s mapami musíme logicky pracovat se souřadnicemi. V Google Maps API je vytvořena třída `google.maps.LatLng`, která se používá napříč celým API, kdekoliv potřebujeme definovat souřadnice.

---

*Příklad 4.5: Definice souřadnic*

Souřadnice se zadávají v desetinném WGS84 [10] formátu jako dvojice hodnot zeměpisná délka a šířka oddělené čárkou (např. "50.076683,14.417794").

```
var myLatLng = new google.maps.LatLng(50.076683, 14.417794);
```

#### 4.3.4 Vytvoření mapy

Vytvoření mapy ve webové stránce je s Google Maps API velice snadná záležitost. V první řadě musíme načíst API, poté už stačí jen vytvořit v html stránce **div** s určitým **id**, které předáme konstruktoru mapy.

---

*Příklad 4.6: Vytvoření mapy ve webové stránce*

Kód html stránky.

---

<sup>2</sup>Geocoding je určení souřadnic na základě adresy.

<sup>3</sup>Country-code Top Level Domain

```

1 <html>
2 <head>
3   <title>Google Maps API Example</title>
4   <script type="text/javascript" src="http://maps.google.com
5     /maps/api/js?sensor=false&language=cs"></script>
6   <script type="text/javascript" src="gm.js"></script>
7 </head>
8 <body onload="gmInit()">
9   <div id="map_canvas" style="width: 800px; height: 600px;"></div>
10 </body>
11 </html>

```

Funkce `gmInit`, která se zavolá po načtení celého dokumentu (díky parametru `onload` tagu `body`), je uložena v souboru **gm.js**, který vypadá následovně:

```

1 var map;
2 function gmInit() {
3   var latlng = new google.maps.LatLng(50.075896,14.415073);
4   var myOptions = {
5     zoom: 17,
6     center: latlng,
7     mapTypeId: google.maps.MapTypeId.ROADMAP
8   };
9   map = new google.maps.Map(
10     document.getElementById("map_canvas"), myOptions);
11 }

```

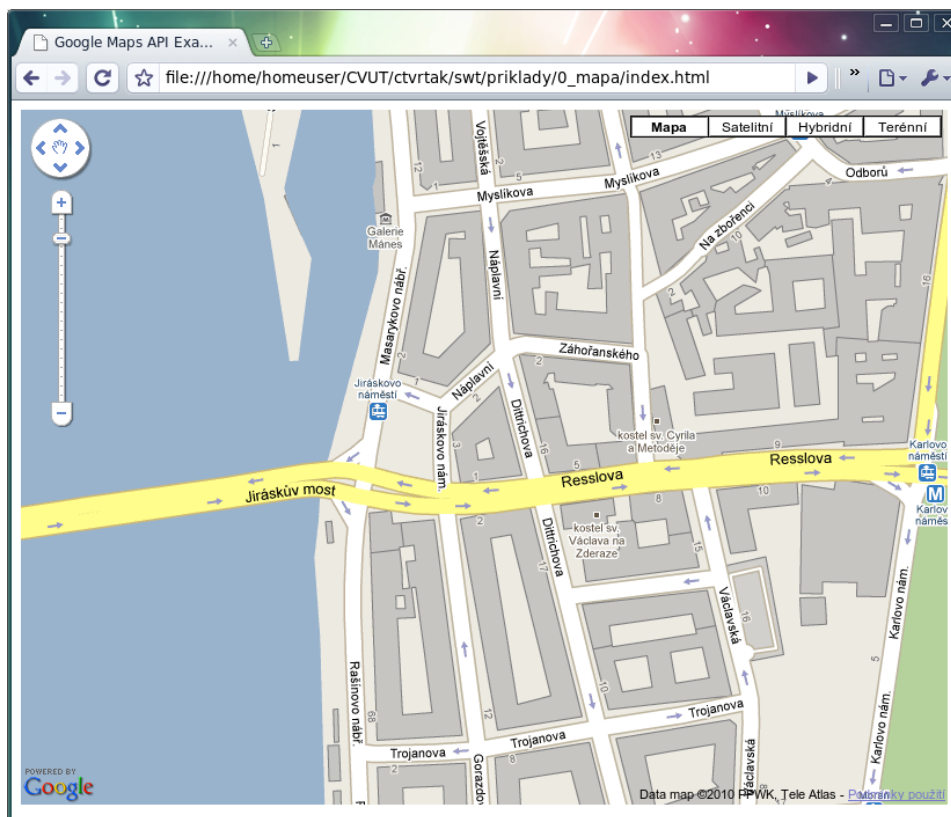
V `myOptions` můžeme samozřejmě definovat mnohem více parametrů, které jsou popsány v oficiální dokumentaci [6].

Parametr `zoom` určuje přiblížení mapy a má hodnoty v rozsahu od **0** po **21**. Parametr `center` určuje střed mapy a můžeme zadat buď objekt třídy `google.maps.LatLng` nebo řetězec obašující adresu. Pomocí konstant uložených v objektu `google.maps.MapTypeId` můžeme určovat, jaký mapový podklad se zobrazí. Konstanty jsou `ROADMAP`, `SATELLITE`, `HYBRID` a `TERRAIN`.

Po načtení stránky uvidíme mapu se základními ovládacími prvky.

## 4.4 Pokročilé použití

V předchozí kapitole jsme se dozvěděli, jak připojit API, nastavit lokalizaci a vložit mapu do webové stránky. V této kapitole se seznámíme s pokročilým použitím Google Maps API V3, s jeho vlastnostmi a možnostmi.



Obrázek 4: Mapa vložená do webové stránky, *obrázek k příkladu 4.6*

#### 4.4.1 Možnosti

Zde budou krátce představeny možnosti, které API V3 nabízí. Je jich sice méně, než v druhé verzi, avšak stále dostatečné množství, které postačí k libovольnému využití.

**Ovládací prvky** Máme možnost využívat základní ovládací prvky, měnit jejich vzhled a volit si, které použijeme. Pokud nám základní prvky nestačí, můžeme si pomocí API přidat vlastní.

**Vrstvy** Do map lze vykreslovat různé vrstvy. Jedná se o markery, lomené čáry, polygonální oblasti, informační okna a vlastní vrstvy či mapové podklady.

**Služby** Lze využívat různé služby. Ve třetí verzi API se jedná o geocoding a vyhledávání cest.

Jak vidíte, možností je mnoho a v dalším textu se s nimi podrobněji seznámíme.

#### 4.4.2 MVC model

Celé API je postavené na MVC<sup>4</sup> modelu, díky kterému můžeme intuitivně přistupovat k jednotlivým proměnným objektů. Veškeré vlastnosti objektů se nastavují pomocí tzv. setterů (např. `setCenter(latlng)`) a jejich hodnoty se získávají pomocí getterů (např. `getCenter()`). To nám umožňuje vždy korektně zvolit název metody pro přístup k dané vlastnosti.

Různé stavy objektů jsou uloženy v jejich proměnných a lze k nim tedy přistupovat pomocí getterů. Změny těchto stavů jsou propagovány a řízeny pomocí událostního modelu.

#### 4.4.3 Událostní model

Událostní model slouží k oznamování změn stavů objektů a k jejich řízení. Každá změna stavu vyvolá určitou událost, kterou je možné odchytnout a zajistit tak její zpracování.

Odchycení událostí se děje pomocí tzv. handlerů, které je možné připojit k jednotlivým objektům jako posluchače události.

Existuje mnoho typů událostí, které se dají rozdělit do dvou základních skupin.

**Uživatelské** Události vyvolané akcí uživatele. Např. kliknutí myši či najetí myši nad objekt.

**Změny stavu** Události vyvolané změnou stavu nějakého objektu v API. Např. změna zoomu mapy nebo načtení mapových podkladů.

---

##### *Příklad 4.7: Nastavení handleru události*

Takto lze nastavit handler události kliknutí do mapy. Prvním parametrem metody `addListener` je objekt, ke kterému připojujeme posluchače, a druhým je pak tzv. callback funkce (posluchač), která se provede při vyvolání události. Zde tato funkce pouze volá jinou funkci, ve které se děje samotná reakce na událost.

```
google.maps.event.addListener(map, 'click', function(event) {  
    placeMarker(event.latLng);  
});
```

Zároveň si zde můžeme všimnout, že objekt události `event` má určité vlastnosti, které si s sebou nese. V tomto případě využíváme proměnné `latLng`, která obsahuje objekt třídy `google.maps.LatLng`, o němž již víme, že představuje souřadnice na mapě.

---

<sup>4</sup>Návrhový vzor Model-View-Controller

## 4.5 Vrstvy

Do map lze vykreslovat mnoho zajímavých prvků v různých vrstvách. Zde se dozvíme jaké existují typy vrstev a něco o jejich použití.

Jednou z věcí, na které je třeba při používání vrstev pamatovat, je, že API se nijak nestará o uchovávání referencí na jednotlivé objekty, které do mapy kreslíme. Neposkytuje tedy žádné metody k odstranění všech našich objektů z mapy. O to se musíme postarat my a v příkladech si ukážeme, jak na to.

### 4.5.1 Ovládací prvky

První věcí, které si v okně s mapou všimneme, jsou ovládací prvky. Ty mají svou samostatnou vrstvu, do které jsou vykreslovány.

Základní ovládací prvky jsou rozdělené do tří skupin.

**navigationControl** Tyto prvky se starají o pohyb v mapě.

**mapTypeControl** Tyto prvky se starají o přepínání mapových podkladů.

**scaleControl** Tyto prvky se starají o změnu měřítka mapy.

Pomocí konfiguračního pole, které předáváme konstruktoru mapy, můžeme základní ovládací prvky vypínat. A to buď všechny, nebo jen některé.

---

*Příklad 4.8: Vypnutí základních ovládacích prvků*

Tento kód vypne základní ovládací prvky.

```
1 var map;
2 function gmInit() {
3   var latlng = new google.maps.LatLng(50.075896,14.415073);
4   var myOptions = {
5     zoom: 17,
6     center: latlng,
7     mapTypeId: google.maps.MapTypeId.ROADMAP,
8     disableDefaultUI: true,
9   };
10  map = new google.maps.Map(
11    document.getElementById("map_canvas"), myOptions);
12 }
```

Můžeme však také vypnout pouze některé ovládací prvky.

```
1 var myOptions = {
2   navigationControl: true,
3   mapTypeControl: false,
4   scaleControl: true
5 };
```

Je možné měnit vzhled a pozici základních ovládacích prvků pomocí předdefinovaných konstant. API nám také umožňuje přidat vlastní ovládací prvky. Tyto záležitosti jsou blíže popsány v oficiální dokumentaci [3].

#### 4.5.2 Markery

Markery slouží k označení místa v mapě pomocí ikonky. Bud' můžeme použít základní ikonku, nebo je možné vytvořit si vlastní. Jak již bylo řečeno, API neposkytuje metody pro vymazání všech markerů a proto si musíme reference na markery uchovávat sami.

K vytvoření markeru slouží třída `google.maps.Marker`. Parametry můžeme opět definovat v konstruktoru [5, Markers] pomocí konfiguračního pole, nebo potom přes settry. Marker se v mapě zobrazí automaticky pokud je mu pomocí konfigurace, nebo pomocí metody `setMap(map)` nastaven objekt mapy. Opačně, pokud zavoláme `setMap(null)`, se marker z mapy vymaže. Poté můžeme smazat samotnou referenci, abychom uvolnili paměť.

---

##### *Příklad 4.9: Vložení a smazání markerů*

Do webové stránky byly přidány dvě tlačítka, která vkládají a mažou markery z mapy.

```
1 <html>
2 :
3 <body onload="gmInit()">
4   <input type="button" value="Vložit markery" onclick="insertMarkers()"/>
5   <input type="button" value="Vymazat markery" onclick="deleteMarkers()"/>
6   <div id="map_canvas" style="width: 800px; height: 600px;"></div>
7 </body>
8 </html>
```

Zde je kód `gm.js`, který nyní obsahuje dvě nové metody `insertMarkers` a `deleteMarkers`.

```
1 var map;
2 var markers = [];
3 function gmInit() {
4   :
5   map = new google.maps.Map(
6     document.getElementById("map_canvas"), myOptions);
7   insertMarkers();
8 }
9
10 function insertMarkers() {
11   var image = 'beachflag.png';
12   var marker = new google.maps.Marker({
```

```

13     map: map,
14     position: new google.maps.LatLng(50.076683,14.417794),
15     icon: image
16 });
17 markers.push(marker);
18
19 marker = new google.maps.Marker({
20     map: map,
21     position: new google.maps.LatLng(50.075583,14.415894),
22 });
23 markers.push(marker);
24 }
25
26 function deleteMarkers() {
27     for(var i = 0; i < markers.length; i++) {
28         markers[i].setMap(null);
29     }
30     markers = [];
31 }

```

Markerům můžeme definovat vlastní ikonky a stíny poměrně komplexněji [5, Icons], než jak je tomu na řádce 15. Obrázek uvedený na řádce 11 je převzatý z oficiální dokumentace [5, Icons].

Zároveň si také můžeme všimnout, jak probíhá mazání markerů z mapy, na řádce 28.

### 4.5.3 Informační okna

Informační okno slouží k zobrazení dodatečných informací a vzhledově připomíná komixovou bublinu. Obsah okna se píše v HTML, takže není problém vložit například obrázek nebo video. K jeho vytvoření použijeme třídu `google.maps.InfoWindow`.

Místo, kde se informační okno zobrazí, se definuje buď pomocí objektu `LatLng`, nebo můžeme okno připojit k markeru.

---

*Příklad 4.10: Zobrazení informačního okna*

Kód zobrazí informační okno u daného markeru.

```

1  :
2  var infowindow = new google.maps.InfoWindow({
3      content: 'Zde je nějaký obsah v <strong>HTML</strong>.'
4  });
5
6  var marker = new google.maps.Marker({
7      position: myLatLng,
8      map: map

```

```

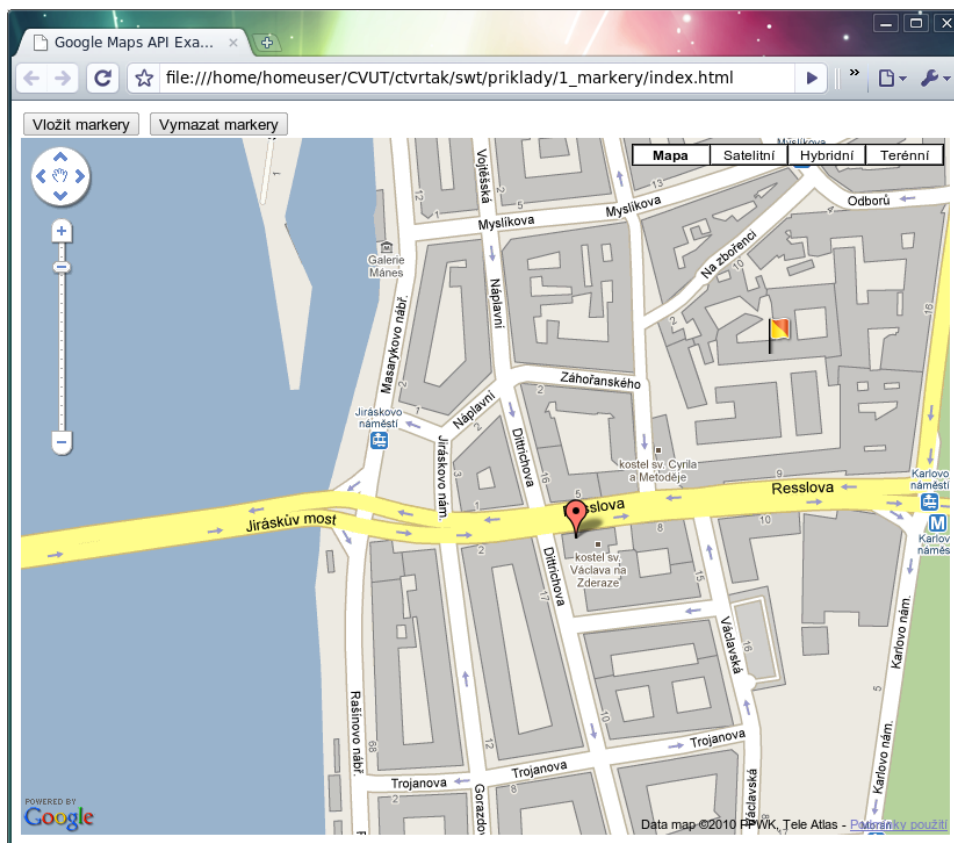
9   });
10
11   google.maps.event.addListener(marker, 'click', function() {
12       infowindow.open(map, marker);
13   });

```

Také si můžeme povšimnout registrace handleru na událost kliknutí na marker, který pak informační okno u daného markeru zobrazí. Metoda `open`, volaná na řádce 12, přijímá jako první parametr objekty mapy, a jako druhý objekt markeru. Pokud marker nepředáme, zobrazí se okno na definovaných souřadnicích (které zde ale nejsou žádné).

#### 4.5.4 Lomené čáry

Lomené čáry slouží k vykreslování čar do mapy. Čáre můžeme nastavit tloušťku, barvu a průhlednost. K vytvoření čáry použijeme třídu `google.maps.Polyline`.



Obrázek 5: Mapa s vloženými markery, *obrázek k příkladu 4.9*

Souřadnice, na které se čára vykreslí, se zadávají jako pole objektů `LatLng`. Čára se v mapě zobrazí, stejně jako u markeru, voláním metody `setMap(map)`.

---

*Příklad 4.11: Vykreslení lomené čáry*

Kód vykreslí lomenou čáru.

```
1  :
2  var myCoordinates = [
3      new google.maps.LatLng(37.772323, -122.214897),
4      new google.maps.LatLng(21.291982, -157.821856),
5      new google.maps.LatLng(-18.142599, 178.431),
6      new google.maps.LatLng(-27.46758, 153.027892)
7  ];
8  var myLine = new google.maps.Polyline({
9      path: myCoordinates,
10     strokeColor: "#FF0000",
11     strokeOpacity: 1.0,
12     strokeWeight: 2
13 });
14
15 myLine.setMap(map);
```

Parametr `strokeColor` udává barvu čáry, `strokeOpacity` průhlednost čáry a `strokeWeight` tloušťku čáry v pixelech.

S objektem čáry můžeme dále pracovat. Pokud zavoláme jeho metodu `getPath()`, získáme pole typu `MVCArray`, ve kterém jsou uloženy objekty `LatLng`. Pole poskytuje metody `getAt(index)`, `insertAt(index, latLng)` a `removeAt(index)`, jejichž význam je asi jasný. Tímto způsobem můžeme čáru dále upravovat.

#### 4.5.5 Polygonální oblasti

Polygonální oblasti se způsobem vytváření podobají lomeným čarám, ale mají několik odlišností. Jsou to uzavřené oblasti, které jsou navíc vyplněné nějakou barvou. K jejich vytvoření slouží třída `google.maps.Polygon`.

Souřadnice se zadávají stejně, jako u lomené čáry. Mezi první a poslední souřadnicí se čára doplní a oblast se tak uzavře. Zobrazení oblasti probíhá stejně, pomocí metody `setMap(map)`.

---

*Příklad 4.12: Vykreslení polygonální oblasti*

Kód vykreslí polygonální oblast.

```

1  :
2  var myCoordinates = [
3      new google.maps.LatLng(37.772323, -122.214897),
4      new google.maps.LatLng(21.291982, -157.821856),
5      new google.maps.LatLng(-18.142599, 178.431)
6  ];
7  var myPolygon = new google.maps.Polygon({
8      paths: myCoordinates,
9      strokeColor: "#FF0000",
10     strokeOpacity: 1.0,
11     strokeWeight: 2,
12     fillColor: "#990000",
13     fillOpacity: 0.5,
14 });
15
16 myPolygon.setMap(map);

```

Oproti čarám však přibyly dva parametry. `fillColor`, který slouží k nastavení barvy výplně a `fillOpacity`, který nastavuje průhlednost výplně.

API nám nabízí ještě další možnosti, jak pracovat s polygony. V příkladu 4.12 na řádce 8 je vidět, že se souřadnice předávají jako `paths`. To je proto, že je možné předat pole polí s objekty `LatLng`. Potom bude mít jeden polygon více oblastí, které bude vykreslovat. Např. vytvoříme polygon se jménem *CVUT*, který bude vykreslovat oblasti okolo všech budov ČVUT. Pokud budeme chtít vytvořit pouze jednu oblast, stačí předat jedno pole, stejně jako v příkladu. API si ho samo převede na pole polí, aby vše fungovalo tak, jak má.

Spolu s tím přibývá také metoda `getPaths()`, která vrací pole typu `MVCArray`, které obsahuje další pole typu `MVCArray` s objekty `LatLng`. Zároveň je však také možné volat na jednoduché polygony pouze metodu `getPath()`, která funguje stejně, jako u lomených čar.

#### 4.5.6 Vlastní vrstvy

Pomocí vlastních vrstev (overlayů) můžeme do map vykreslovat různé obrázky nebo vlastní mapové podklady. API nám umožňuje využít dvou přístupů, jak zobrazit vlastní obrázky. První jednodušší cesta spočívá ve vykreslení celého obrázku přes mapu. To je použitelné, pokud je obrázek dostatečně malý. V opačném případě je nutné použít druhou cestu, kterou je vytvoření vlastních mapových dlaždic a jejich následné vykreslení, dle potřeby. Zde si popíšeme pouze první způsob [5, CustomOverlays], druhý ponecháme na oficiální dokumentaci [5, CustomMapTypes].

K vytvoření overlaye slouží prototyp (třída) `google.maps.OverlayView`, který je nutné zdědit a implementovat některé metody. Těmi jsou `onAdd()`,

kterou volá API, když je připraveno overlay zobrazit, `onRemove()`, která se volá při smazání overlaye z mapy a `draw()`, která slouží k samotnému vykreslování overlaye. Objekt `OverlayView` také poskytuje metody k překladi souřadnic na mapě na pozici na obrazovce. Obrázek se pak vykresluje jako klasický html element `img`, který je vložený do elementu `div`. Vše bude nejlépe vidět na příkladu.

---

*Příklad 4.13: Vykreslení vlastního overlaye*

Kód ukazuje základní použití třídy `google.maps.OverlayView`, její zdědění a vykreslení vlastního overlaye.

```
1  var map;
2  var overlay;
3
4  LightMapOverlay.prototype = new google.maps.OverlayView();
5
6  function gmInit() {
7      var latlng = new google.maps.LatLng(37.0625,-95.677068);
8
9      var myOptions = {
10         zoom: 4,
11         center: latlng,
12         mapTypeId: google.maps.MapTypeId.ROADMAP
13     };
14     map = new google.maps.Map(
15         document.getElementById("map_canvas"),
16         myOptions);
17
18     var neBound = new google.maps.LatLng(53.46763, -57.97461);
19     var swBound = new google.maps.LatLng(7.04088, -127.18634);
20     var bounds = new google.maps.LatLngBounds(swBound, neBound);
21
22     var srcImage = 'lightmap.png';
23     overlay = new LightMapOverlay(bounds, srcImage, map);
24 }
25
26 function LightMapOverlay(bounds, image, map) {
27     this.bounds_ = bounds;
28     this.image_ = image;
29     this.map_ = map;
30     this.div_ = null;
31
32     this.setMap(map);
33 }
34
35 LightMapOverlay.prototype.onAdd = function() {
36     var div = document.createElement('DIV');
37     div.style.borderStyle = "none";
```

```

38     div.style.borderWidth = "0px";
39     div.style.position = "absolute";
40
41     var img = document.createElement("img");
42     img.src = this.image_;
43     img.style.width = "100%";
44     img.style.height = "100%";
45     div.appendChild(img);
46
47     this.div_ = div;
48
49     var panes = this.getPanes();
50     panes.overlayLayer.appendChild(div);
51 }
52
53 LightMapOverlay.prototype.draw = function() {
54     var overlayProjection = this.getProjection();
55
56     var sw = overlayProjection.fromLatLngToDivPixel(
57         this.bounds_.getSouthWest());
58     var ne = overlayProjection.fromLatLngToDivPixel(
59         this.bounds_.getNorthEast());
60
61     var div = this.div_;
62     div.style.left = sw.x + 'px';
63     div.style.top = ne.y + 'px';
64     div.style.width = (ne.x - sw.x) + 'px';
65     div.style.height = (sw.y - ne.y) + 'px';
66 }
67
68 LightMapOverlay.prototype.onRemove = function() {
69     this.div_.parentNode.removeChild(this.div_);
70     this.div_ = null;
71 }

```

V tomto kódu si můžeme všimnout hned několika zajímavých věcí. Na řádce 4 dochází k nastavení prototypu naší třídy `LightMapOverlay` na nový objekt třídy `OverlayView`. Na řádce 20 dochází k vytvoření objektu `LatLngBounds`, který bude určovat hranice okolo vykreslovaného overlaye. Hranice se zadávají jako souřadnice severozápadního a jihovýchodního rohu. Overlay je pak vytvořen na řádce 23. V konstruktoru si můžeme všimnout nastavení instančních proměnných, které budeme později potřebovat při vykreslování. Na řádce 30 je vytvořena proměnná `div_`, ve které bude následně uložen html element `div` s naším obrázkem. Nyní se podívejme na implementace metod prototypu.

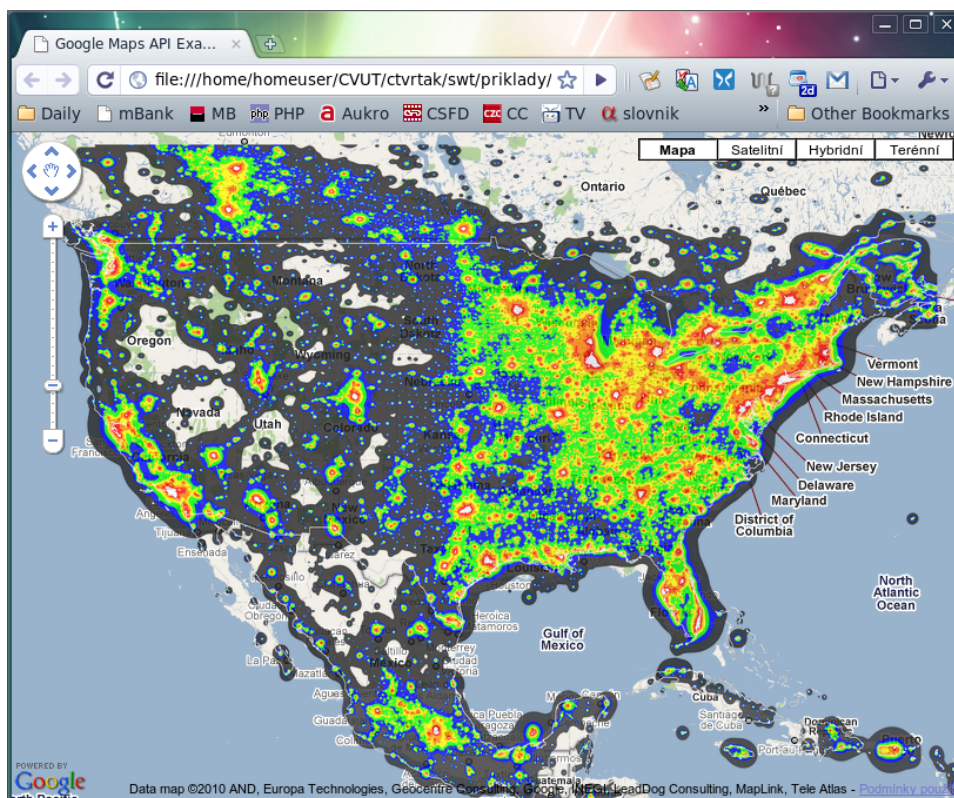
Ve funkci `onAdd` je na řádce 36 vytvoření elementu `div` a nastavení základních parametrů nutných ke správnému vykreslení. Poté je na řádce

41 vytvořen element `img`, je nastavena cesta k obrázku a nastaveny rozměry. Poté je element vložen do `divu`. Na řádce 49 jsou získány jednotlivé vrstvy, do kterých je možné náš `div` vložit. Ten je poté přidán do vrstvy `overlayu`. Vrstvy, do kterých je možné `div` přidat, jsou popsány v oficiální dokumentaci [5, Initializing].

Funkce `onRemove` slouží pouze k odstranění námi vložených elementů a úklidu.

Zajímavější je funkce `draw`. Nejprve na řádce 54 získáme objekt `MapCanvasProjection`, který nám následně umožní konvertovat souřadnice na pozici na obrazovce. Takto získané údaje nastavíme našemu `divu` jako absolutní pozici v rámci okna s mapou na řádce 62 a následujících.

K tomuto příkladu je nutno podotknout, že zobrazený `overlay` zcela neodpovídá požadavkům na malý obrázek a celá aplikace je pak náležitě pomalá.



Obrázek 6: Mapa se zobrazeným `overlayem`, *obrázek k příkladu 4.13*

## **4.6 Služby**

### **4.6.1 Geocoding**

### **4.6.2 Hledání cest**

## **4.7 Knihovny pro V3**

## **5 Google maps API V2**

Obečné kecy.

### **5.1 Co je navíc oproti V3?**

### **5.2 Knihovny pro V2**

## 6 Google maps API pro Flash

## **7 Další API spojené s Google Maps**

### **7.1 Google Maps Data API**

### **7.2 Google Mapplets**

## Reference

- [1] Google. Google maps api.  
<http://code.google.com/apis/maps/>, 2010.
- [2] Google. Google maps api coverage, languages.  
<http://spreadsheets.google.com/pub?key=p9pdwsai2hDMsLkXsoM05KQ&gid=1>, 2010.
- [3] Google. *Google Maps API V3 Controls*, 2010.  
<http://code.google.com/apis/maps/documentation/v3/controls.html>.
- [4] Google. *Google Maps API V3 Documentation*, 2010.  
<http://code.google.com/apis/maps/documentation/v3/basics.html>.
- [5] Google. *Google Maps API V3 Overlays*, 2010.  
<http://code.google.com/apis/maps/documentation/v3/overlays.html>.
- [6] Google. *Google Maps API V3 Reference*, 2010.  
<http://code.google.com/apis/maps/documentation/v3/reference.html>.
- [7] Google. *Static Maps V2 API Documentation*, 2010.  
<http://code.google.com/apis/maps/documentation/staticmaps/>.
- [8] Unicode. Unicode locale data markup language, unicode language and locale identifiers.  
[http://www.unicode.org/reports/tr35/#Unicode\\_Language\\_and\\_Locale\\_Identifiers](http://www.unicode.org/reports/tr35/#Unicode_Language_and_Locale_Identifiers), 2010.
- [9] W3C. Geolocation api specification.  
<http://dev.w3.org/geo/api/spec-source.html>, 2010.
- [10] Wikipedia. World geodetic system.  
[http://cs.wikipedia.org/wiki/World\\_Geodetic\\_System](http://cs.wikipedia.org/wiki/World_Geodetic_System), 2009.