

Mape

Interaktivnost je jedna od osnovnih osobina modernih web sajtova. Tako se danas kompletna zajednica uključena u razvoj weba trudi da čak i jednostavni, prezentacioni web sajtovi korisnicima omoguće nešto više od običnog teksta i fotografija. Jedan od primera interaktivnosti sajtova jesu i mape.

Prva asocijacija na mape na web sajtovima za većinu je verovatno kontakt sekcija ili zasebna strana koja je popunjena mapom koja ukazuje na lokaciju na kojoj se nalazi predmet interesovanja sajta. Moderni pristupi za integraciju digitalnih mapa u sajtove omogućavaju pretvaranje statičkih geografskih mapa u potpuno interaktivne elemente web dizajna, koji korisnicima mogu da pruže mnogo više od pukog uvida u adresu na kojoj se nalazi neka kompanija, organizacija ili pojedinac.

Lekcija pred vama baviće se osnovnim načelima integracije digitalnih, interaktivnih mapa u HTML dokumente. Biće predstavljeno korišćenje jednog potpuno besplatnog, open-source sistema koji omogućava interakciju sa mapama različitih provajdera. Pored integracije, u ovoj lekciji biće reči i o načinima za programabilnu kontrolu mapa i prilagođavanje mapa sopstvenim potrebama.

Razvoj web mapa

Razvoj digitalnih tehnologija nije zaobišao ni geografske mape. Tako su tokom vremena svi tipovi konvencionalnih, papirnih mapa i atlasa pretvoreni u digitalni oblik, koji omogućava mnogo jednostavnije čuvanje, ali i rukovanje. Digitalne mape našle su svoj put i do weba.

Web mape su digitalne mape koje se korisnicima prezentuju korišćenjem web pregledača. Ipak, za razliku od digitalnih mapa koje korisnicima ne obezbeđuju bilo kakvu interaktivnost, web mape se karakterišu mogućnošću veće ili manje kontrole.

Prvi servis koji je ponudio web mape bio je MapQuest, 1996. godine. Reč je o servisu koji je po svojim osobinama bio veoma daleko od današnjih, modernih web mapa. MapQuest je na primer, zahtevao potpuno osvežavanje stranice kako bi se obavili osnovni tipovi interakcije – kretanje kroz mapu (*scroll*) i približavanje i udaljavanje (*zoom in, zoom out*).

Moderne web mape, nalik na one kakve poznajmo danas, prvi je ponudio Google, sa servisom Google Maps, 2005. godine. Tokom godina, Google Maps je konstantno unapređivan, pa danas pored mogućnosti pregleda mapa gotovo kompletne planete omogućava i brojne dopunske servise, kao što su navigacija, interaktivne panoramske slike ulica (*Street View*), uvid u realno stanje saobraćaja itd.

Pored Google Maps, vredan pomena je i neprofitni projekat OpenStreetMap. Reč je o mapama u čijem stvaranju može da uzme učešće bilo ko. Osnovni cilj je kreiranje potpuno besplatnih mapa kompletne planete. OpenStreetMap je danas veoma ozbiljan projekat u čijem uređivanju u većoj ili manjoj meri učestvuje oko dva miliona registrovanih korisnika. Takođe, i mnoge velike kompanije koriste OpenStreetMap servis: Facebook, Craigslist, Seznam, Foursquare...

Upravo spomenuti sistemi Google Maps i OpenStreetMap mogu se nazvati provajderima mapa. Pored njih, danas postoje i mnogi drugi provajderi web mapa (Bing Maps, Mapbox, MAPCAT...).

Anatomija web mape

U osnovni modernih web mapa jeste pojam pločica (*tiles*). Tako su današnje web mape sačinjene iz mnoštva malih, pravougaonih slika, koja se drugačije nazivaju pločice. Sve pločice imaju identične dimenzije, tipično 128x128 ili 256x256px. Sastavljanjem većeg broja pločica, dobija se kompletan prikaz mape.

Razlog za korišćenje ovakvog pristupa je i više nego jednostavan – više malih pločica učitava se mnogo brže nego jedna velika slika, koja bi predstavljala kompletну mapu. Takva osobina je posebno korisna ukoliko se zna da korisnici u jednom trenutku uglavnom pregledaju samo određeni, limitirani deo mape. Pločice omogućavaju da se učita samo onaj deo mape koji je trenutno vidljiv na displejima korisnika.

Jedna od osnovnih funkcionalnosti web mapa jeste mogućnost kontrolisanja razmere, odnosno približavanje ili udaljavanje (*zoom in*, *zoom out*). Takva funkcionalnost zasniva se na određenom broju nivoa uvećanja. Uglavnom je taj broj 18. Manji brojevi označavaju širu sliku, odnosno manje uvećanje i obrnuto. Svaki od nivoa uvećanja poseduje sopstveni skup pločica.

Šta je Leaflet?

Leaflet je jedna od najpopularnijih JavaScript biblioteka otvorenog koda za rukovanje interaktivnim web mapama. Odmah na početku je potrebno razumeti da Leaflet nisu mape, odnosno da nije reč o sistemu koji je provajder mape. Leaflet obezbeđuje potrebne JavaScript funkcionalnosti kako bi mape različitih provajdera mogle da se integrišu u HTML i kontrolišu korišćenjem JavaScript jezika.

U ovom kursu odlučili smo se za korišćenje Leaflet biblioteke otvorenog koda za rukovanje potpuno besplatnim OpenStreetMap mapama. Iako je donedavno prvi izbor većine frontend programera uglavnom bio Google Maps servis, Google je nedavno uveo naplaćivanje korišćenja ovog sistema po broju zahteva za prikazom mape, što se približno može prevesti u broj poseta web sajta. Određeni broj zahteva (10.000–20.000) na mesečnom nivou Google poklanja, dok je nakon toga za svakih 1000 zahteva potrebno izdvojiti sedam dolara.

Zvanična Leaflet web prezentacija nalazi se na sledećoj adresi:

<https://leafletjs.com/>

Pitanje

Odabratи provajdere web mapa:

- **Google Maps**
- Leaflet
- Bootstrap

Objašnjenje:

Provajderi mapa i biblioteke koje omogućavaju njihov prikaz i interakciju su odvojeni pojmovi. Primeri provajdera mapa su Google Maps, Mapbox, OpenStreetMaps. Leaflet je jedna od biblioteka za integraciju i manipulaciju mapama različitih provajdera.

Integracija Leafleta u HTML dokument

Leaflet je u HTML dokument moguće integrisati na dva načina:

- preuzimanjem fajlova Leaflet biblioteke i ručnom integracijom u HTML dokument
- korišćenjem online resursa, dodavanjem referenci na neophodne fajlove hostovane na CDN serverima

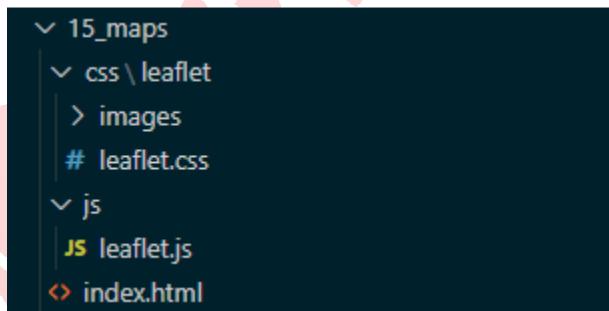
Mi ćemo u nastavku preuzeti fajlove Leaflet biblioteke i reference na njih ručno smestiti unutar HTML dokumenta. Preuzimanje fajlova se može obaviti sa sledeće web adrese:

<https://leafletjs.com/download.html>

Sa prikazane adrese je potrebno preuzeti poslednju stabilnu verziju Leaflet biblioteke, što podrazumeva dobijanje nekoliko fajlova, spakovanih unutar zip arhive. Proces integracije Leaflet biblioteke podrazumeva korišćenje sledećih fajlova i foldera:

- `leaflet.js` – fajl sa JavaScript kodom Leaflet biblioteke
- `leaflet.css` – fajl sa stilizacijom Leaflet biblioteke
- `images` – folder sa slikama koje koristi `leaflet.css` fajl; ovaj folder mora biti u istom folderu kao i `leaflet.css` fajl

Nakon postavljanja prikazanih fajlova i foldera na lokaciju web sajta, dobija se struktura kao na slici 15.1.



Slika 15.1. Struktura sajta nakon dodavanja Leaflet fajlova

Uključivanje Leaflet fajlova u HTML dokument

Unutar HTML dokumenta je potrebno uključiti dva prikazana `.js` i `.css` fajla. CSS fajl je potrebno postaviti unutar `head` sekciјe HTML dokumenta na sledeći način:

```
<link rel="stylesheet" href="css/leaflet/leaflet.css" />
```

Bitno je JavaScript fajl uključiti nakon upravo prikazanog CSS fajla, na sledeći način:

```
<script src="js/leaflet.js"></script>
```

Mi ćemo ovakav `script` element postaviti na sam kraj `body` dela dokumenta, ali pre bilo kog JavaScript koda koji ćemo samostalno pisati.

Kreiranje HTML dokumenta za prikaz mape i definisanje njegovih dimenzija

Nakon uključivanja dva prikazana fajla u HTML dokument, neophodno je kreirati element unutar koga će mapa biti prikazana. Najbolje je u tu svrhu kreirati jedan `div` element:

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

Bitno je primetiti da je `div` element obeležen `id` atributom sa vrednošću `map`. Ovu vrednost ćemo kasnije koristiti za referenciranje ovakvog elementa, kako bi se unutar njega pojavila mapa.

Nakon dodavanja HTML elementa, potrebno je definisati i njegove dimenzije. S obzirom na to da je reč o `div` elementu, on će zauzimati kompletну dostupnu širinu unutar svog roditelja, odnosno u ovom slučaju kompletног web pregledača. Ipak, neophodno je ručno definisati visinu:

```
#map {  
    height: 580px;  
}
```

Prikazanim CSS opisom visina `div` elementa se postavlja na 580px.

Definisanje JavaScript logike za prikaz mape

Sada se može preći na korišćenje specifičnih JavaScript funkcionalnosti Leaflet biblioteke. Prvo je neophodno kreirati objekat mape, što se postiže upotrebom metode `map()`, specifičnog Leaflet objekta koji je imenovan kao `L`:

```
var myMap = L.map('map');
```

Metodi `map()` prosleđuje se `id` vrednost elementa unutar koga će biti prikazana mapa. Nakon kreiranja objekta mape, koji je smešten unutar promenljive `myMap`, neophodno je definisati šta će mapa prikazivati i koji će biti početni nivo približenja:

```
myMap.setView([37.974, 23.72], 13);
```

Metodom `setView()` obavljaju se dve operacije:

- definišu se koordinate inicijalnog prikaza mape, korišćenjem niza sa dva elementa; prvi element niza je geografska širina (*latitude*), a drugi geografska dužina (*longitude*, *engl.*)
- definiše se početni nivo približenja, jednom od numeričkih vrednosti u rasponu od 1 do 18; veći brojevi označavaju viši nivo približenja

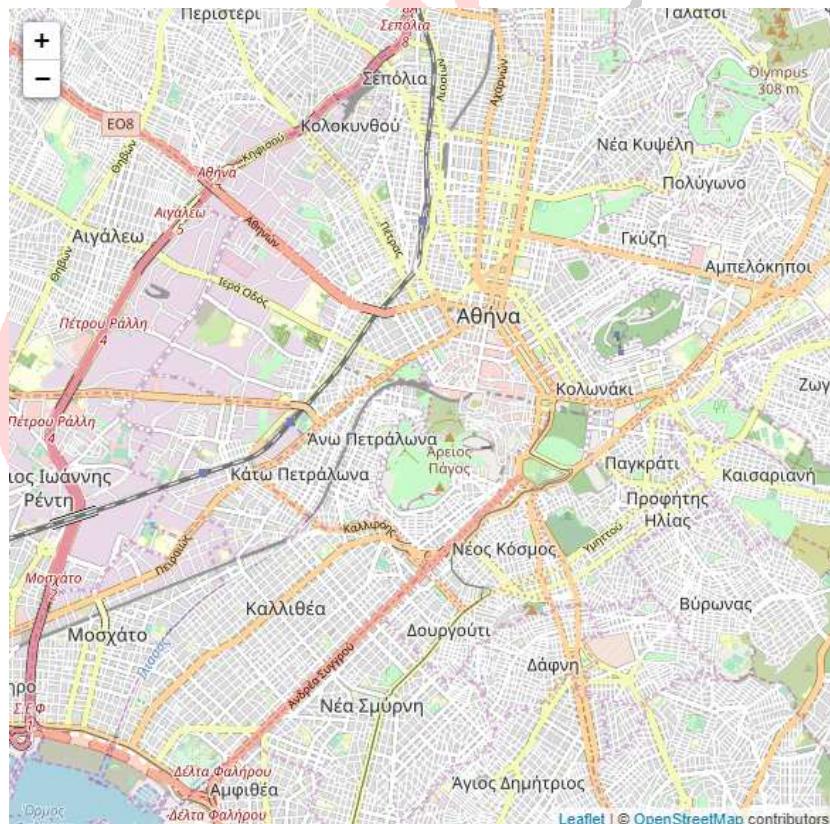
Do sada prikazanim kodom obavljeno je kreiranje prostora za prikaz mape unutar HTML dokumenta. Ipak, prethodne dve linije neće učiniti da mapa uistinu i postane vidljiva na stranici. Naime, unutar kreiranog prostora za mapu, potrebno je smestiti pločice mape (*map tiles*). Nešto ranije je objašnjeno da se web mape zapravo sastoje iz slojeva pločica, pri čemu je svaki nivo približenja zapravo jedan sloj. Upravo zbog toga Leaflet metoda za obavljanje takvog posla ima slikoviti naziv `tileLayer()`:

```
L.tileLayer('https://tile.openstreetmap.org/{z}/{x}/{y}.png',
{ attribution:
'&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a>
contributors' }).addTo(myMap);
```

Prikazanom naredbom obavlja se nekoliko stvari:

- metodom `tileLayer()` definiše se provajder mape, odnosno preciznije, provajder slojeva pločica mape, ali i propratni tekst koji će stajati u podnožju mape; reč je o tekstu kojim se navodi provajder mape; iako se ovaj tekst može izostaviti, to je protivno pravilima koja su propisali provajderi mape – u ovom slučaju OpenStreetMap;
 - metodom `addTo()` mapa odabranog provajdera se smešta unutar već kreiranog objekta mape, čime se obavlja prikaz mape unutar HTML dokumenta.

Nakon definisanja prikazanog koda, prvi put će na stranici moći da se vidi mapa (slika 15.2).



Slika 15.2. OpenStreetMap mapa unutar HTML dokumenta

Osnovna kontrola mape

U prethodnim redovima, jedina kontrola koja je obavljena nad mapom bila je definisanje koordinata i nivoa približenja. U narednim redovima će pored navedenih, biti prikazani načini za postizanje osnovne kontrole nad mapom.

Za početak, moguće je koristiti nekoliko različitih svojstava za uticanje na osobine mape (tabela 15.1).

Svojstvo	Značenje
attributionControl	boolean vrednost kojom se kontroliše da li će atribucija biti prikazana ili ne
zoomControl	boolean vrednost kojom se kontroliše da li će se na mapi prikazati kontrole za približavanje/udaljavanje
doubleClickZoom	boolean vrednost kojom se definiše da li će približavanje moći da se obavi korišćenjem duplog klika
scrollWheelZoom	boolean vrednost kojom se definiše da li će točkić miša moći da se koristi za približavanje/udaljavanje
dragging	boolean vrednost kojom se definiše da li će korisnik moći da se prevlačenjem kreće po mapi
center	LatLang vrednost kojom se definiše inicijalni geografski centar mape
zoom	number vrednost kojom se definiše inicijalni nivo približenja
minZoom	number vrednost kojom se definiše minimalni nivo približenja
maxZoom	number vrednost kojom se definiše maksimalni nivo približenja

Tabela 15.1. Svojstva za kontrolisanje mapa

Sva prikazana svojstva iz tabele 15.1. najefikasnije je koristiti prilikom kreiranja objekta mape. Tako se već prikazani pristup za kreiranje objekta mape može izmeniti da izgleda ovako:

```
var myMap = L.map('map', {  
    center: [37.974, 23.72],  
    zoom: 13,  
    attributionControl: false,  
    zoomControl: false,  
    doubleClickZoom: false,  
    scrollWheelZoom: false,  
    dragging: false,  
    minZoom: 10,  
    maxZoom: 17  
});
```

Na ovaj način su sva svojstva ilustrovana tabelom 15.1. iskorišćena za inicijalno podešavanje mape. Mapa koja se na ovaj način dobije neće biti naročito upotrebljiva, s obzirom na to da je namerno obavljeno isključivanje većine podrazumevanih mehanizama za kontrolisanje mape. Ipak, to je učinjeno samo kako biste videli efekte ovih svojstava i način za njihovo definisanje.

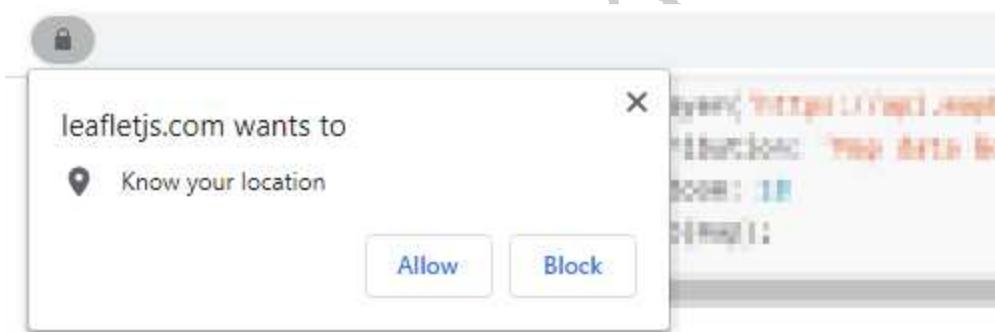
Geolociranje

U prethodnim redovima ste mogli da vidite dva načina za definisanje inicijalnih koordinata mape, kojima se definiše ono što će mapom biti prikazano, odmah nakon njenog učitavanja. U nekim situacijama se može javiti potreba za dinamičkim prikazom lokacije na kojoj se nalazi posetilac web sajta. U takvoj situaciji moguće je osloniti se na funkcionalnost geolociranja koju web pregledači stavljuju na raspolaganje web sajtovima. Sve što je potrebno uraditi kako bi se geolociranje aktiviralo jeste sledeće:

```
myMap.locate({ setView: true, maxZoom: 16 });
```

Metodom `locate()` aktivira se geolociranje. Ovoj metodi se prosleđuju opcije za podešavanje geolociranja. Postavljanjem svojstva `setView` na `true` rečeno je da nakon uspešnog geolociranja želimo da mapa prikaže korisničku lokaciju. Svojstvo `maxZoom` definiše maksimalni nivo uvećavanja prilikom prikaza korisničke lokacije.

Preduslov za uspešno obavljanje geolociranja jeste dobijanje dozvole od korisnika. Tako će korisnik biti upitan da li želi da web sajtu sa ovakvim kodom dozvoli pristup lokaciji (slika 15.3).



Slika 15.3. Primer prozora u kome web pregledač od korisnika traži dozvolu da njegovu lokaciju prosledi web sajtu

Crtanje oblika na površini mape

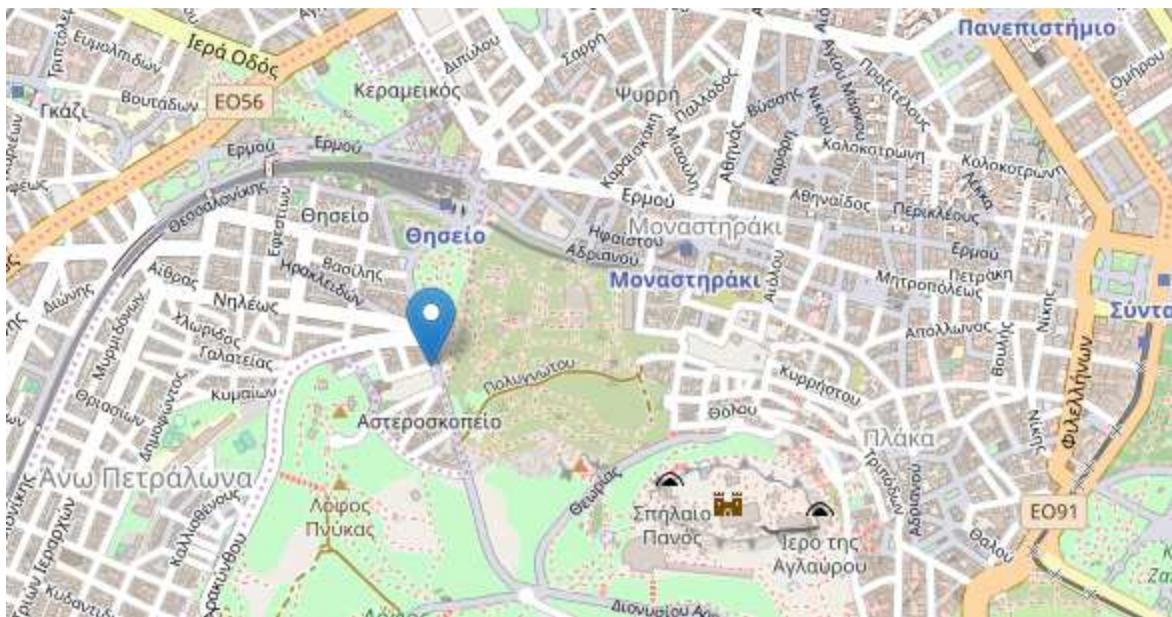
Verovatno najčešća intervencija nad mapama koje se dodaju na web sajtove predstavlja crtanie markera, oznaka ili nekih drugih obeležja kojima se korisniku žele prezentovati neke lokacije od značaja. Leaflet poseduje bogat skup funkcionalnosti kojima se tako nešto može postići.

Markeri

Na površinu mape je najjednostavnije dodati marker:

```
var marker = L.marker([37.974, 23.72]).addTo(myMap);
```

Korišćenjem metode `marker()`, obavlja se kreiranje objekta tipa `Marker`. Ovoj metodi se prosleđuju koordinate na kojima je potrebno da se marker nađe. Efekat prikazane linije koda je kao na slici 15.4.



Slika 15.4. Primer podrazumevanog Leaflet markera

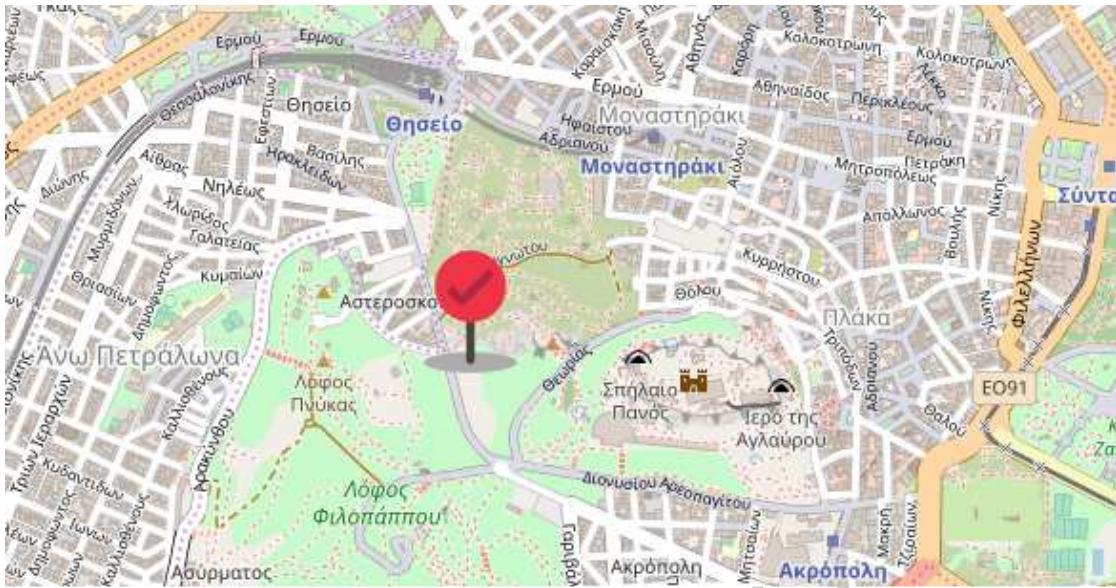
Ipak, pored koordinata, metodi `marker` je moguće proslediti još neke opcije kojima se marker može prilagoditi sopstvenim potrebama. Svakako najkorisnija je mogućnost definisanja sopstvene ikonice markera:

```
var myIcon = L.icon({
  iconUrl: 'img/marker.png'
});

var marker = L.marker([37.974, 23.72], {
  icon: myIcon
}).addTo(myMap);
```

Prvo se obavlja definisanje sopstvene ikonice markera, korišćenjem metode `icon()`. Svojstvom `iconUrl`, definiše se putanja na kojoj se nalazi fajl ikonice. Tako kreiran `Icon` objekat se postavlja za vrednost svojstva `icon`, drugog parametra metode `marker()`.

Marker koji se na ovaj način kreira izgleda kao na slici 15.5.



Slika 15.5. Primer korisnički definisanog Leaflet markera

Vektorski oblici

Pored markera, Leaflet omogućava crtanje i različitih vektorskih oblika na mapi (tabela 15.2).

Vektorski oblik	Opis
Izlomljena linija	predstavlja se objektom <code>Polyline</code> , a kreira korišćenjem metode <code>polyline()</code>
Pravougaonik	predstavlja se objektom <code>Rectangle</code> , a kreira korišćenjem metode <code>rectangle()</code>
Krug	predstavlja se objektom <code>Circle</code> , a kreira korišćenjem metode <code>circle()</code>
Poligon	predstavlja se objektom <code>Polygon</code> , a kreira korišćenjem metode <code>polygon()</code>

Tabela 15.2. Najznačajniji vektorski oblici koje je moguće crtati na mapi

Svi prikazani vektorski oblici crtaju se na sličan način. Oni se definišu tačkama na mapi. Svaka tačka se definiše koordinatama, odnosno geografskom širinom i dužinom. Na primer, jedna izlomljena linija se može nacrtati na sledeći način:

```

var latLngs = [
    [38.00729, 23.73508],
    [37.99992, 23.73328],
    [38.00084, 23.72757],
    [37.98917, 23.72671]
];

var polyline = L.polyline(latLngs, { color: 'red' })
).addTo(myMap);

```

Prikazanim kodom prvo je obavljeno definisanje tačaka od kojih će biti sastavljena izlomljena linija. Definisano je ukupno četiri tačke. Svaka tačka je zapravo niz od dve vrednosti – geografske širine i geografske dužine.

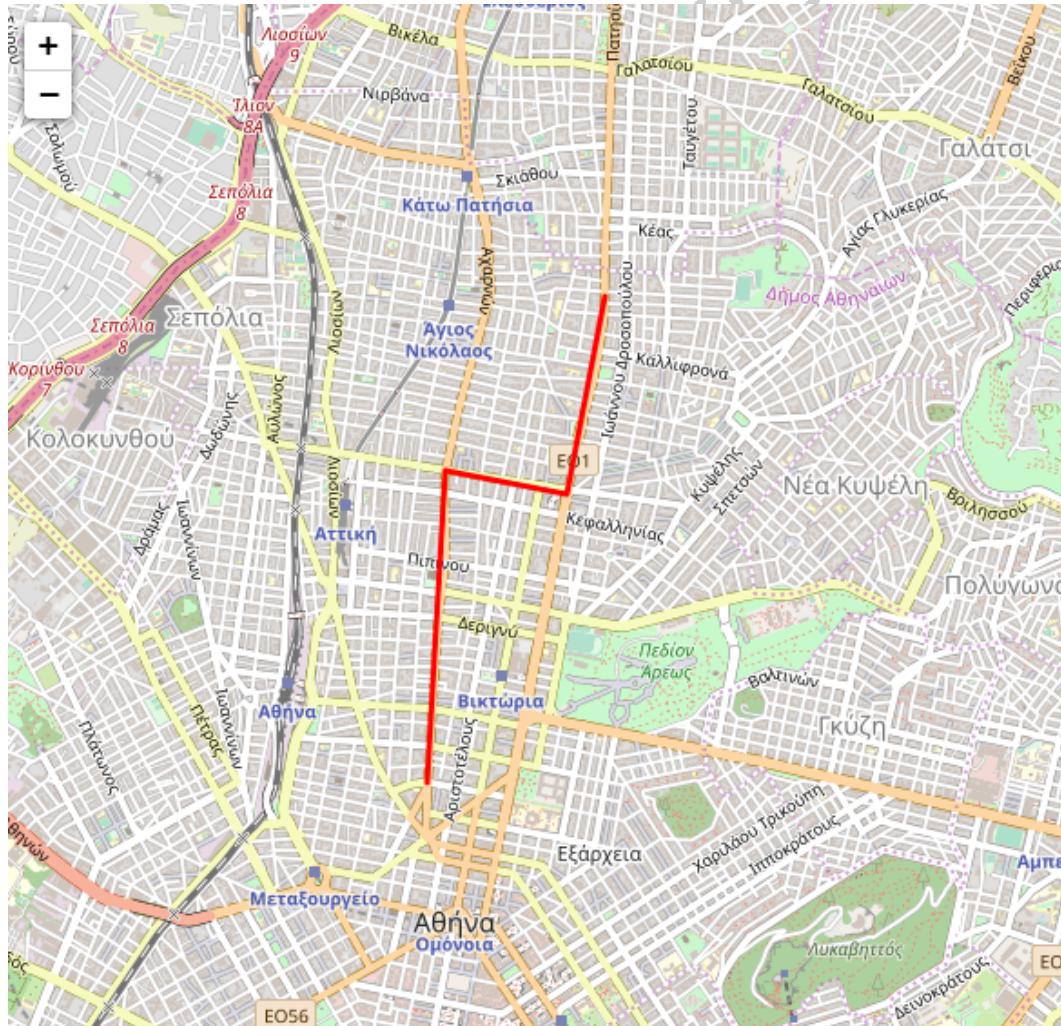
Ovako kreiran skup tačaka prosleđuje se metodi `polyline()`, kojom se obavlja kreiranje objekta tipa `Polyline`. Tako kreiran objekat se dodaje objektu mape, korišćenjem metode `addTo()`.

Pored tačaka, metodi `polyline()` se prosleđuje i objekat za konfigurisanje izlomljene linije, kao drugi parametar. Takvim objektom je definisana boja linije, koja je postavljena na crveno.

Nakon kreiranja i crtanja bilo kog oblika na mapi, veoma često se obavlja i centriranje mape na lokaciju na kojoj se nalazi nacrtani oblik. To se može postići na sledeći način:

```
myMap.fitBounds(polyline.getBounds());
```

Na kraju, efekat prikazanog koda biće kao na slici 15.6.

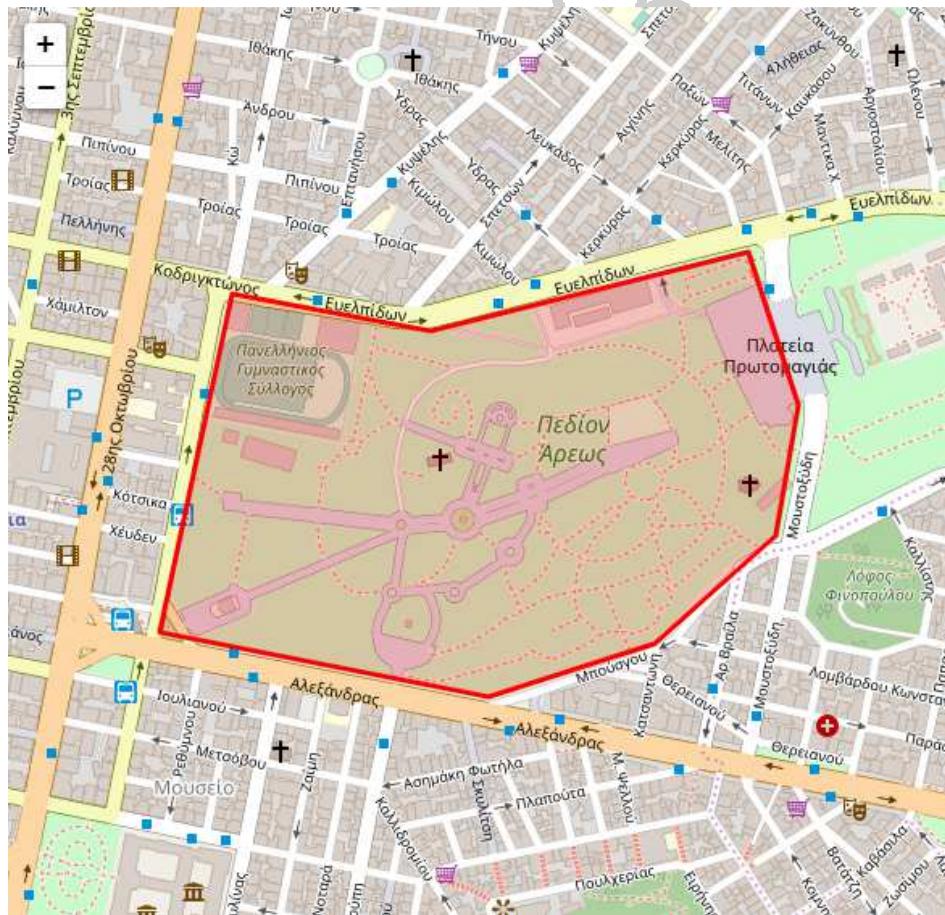


Slika 15.6. Primer crtanja izlomljene linije na površini mape

I ostali vektorski oblici se crtaju na sličan način. U nastavku će biti prikazano crtanje poligona, koji su veoma korisni ukoliko je na mapi potrebno oivičiti neku oblast:

```
var latlngs = [  
    [37.99528, 23.73347],  
    [37.99174, 23.73250],  
    [37.99105, 23.73689],  
    [37.99162, 23.73910],  
    [37.99274, 23.74071],  
    [37.99411, 23.74101],  
    [37.99573, 23.74035],  
    [37.99489, 23.73612]  
];  
  
var polygon = L.polygon(latlngs, { color: 'red' }).addTo(myMap);  
  
myMap.fitBounds(polygon.getBounds());
```

Kao i prilikom crtanja izlomljenih linija, i sada se definiše skup tačaka. Reč je o tačkama koje će definisati ivice poligona. Definisane tačke se prosleđuju metodi `polygon()`. Pored tačaka, definiše se i boja okvira i ispune. Na kraju se obavlja i centriranje mape, korišćenjem metode `fitBounds()`. Efekat je kao na slici 15.7.



Slika 15.7. Primer crtanja poligona na površini mape

Obrada događaja

Svi do sada nacrtani elementi na površini mape su zapravo interaktivni. To znači da je moguće definisati logiku koja će se aktivirati kada korisnik klikne na neki od prikazanih oblika. Najjednostavniji način za obavljanje takvog posla jeste korišćenje ugrađenih metoda Leaflet biblioteke i njenih specijalnih elemenata za prikaz poruka na mapi:

- popup
- tooltip

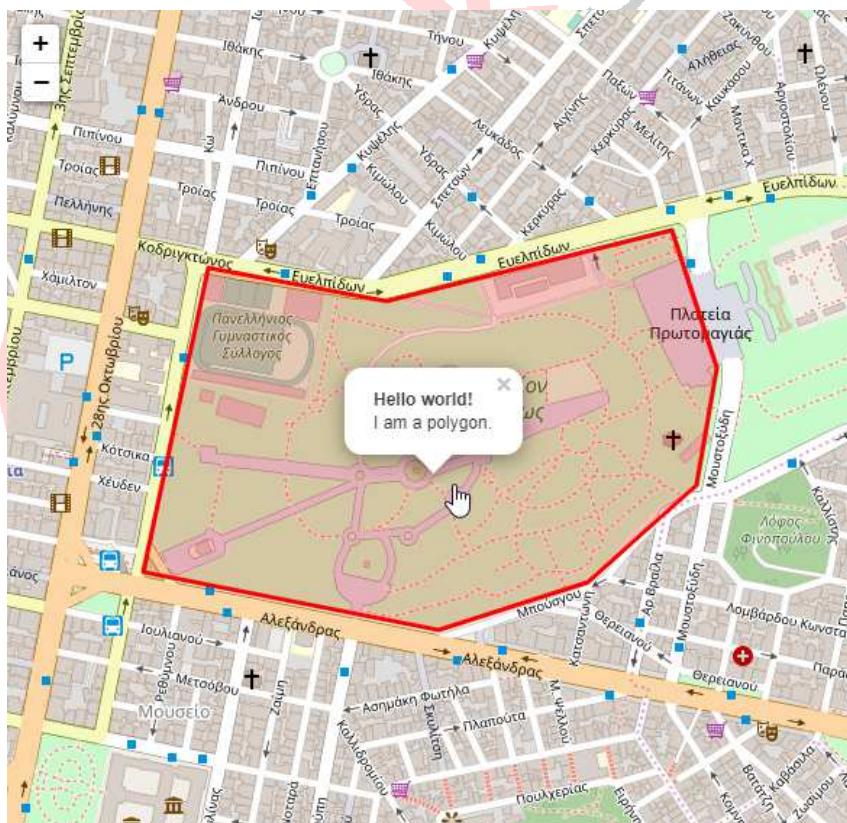
Popup i tooltip su ugrađeni elementi Leaflet biblioteke koji se mogu koristiti za prikaz poruka unutar mape.

Popup

Popup se može kreirati na sledeći način:

```
polygon.bindPopup( "<b>Hello world!</b><br>I am a polygon." );
```

Metodom `bindPopup()` obavlja se kreiranje popupa i njegovo povezivanje sa elementom nad kojim se ova metoda pozove. S obzirom na to da se u primeru metoda poziva nad `polygon` objektom, klikom na takav poligon biće prikazan popup. Sadržaj popupa se definiše parametrom koji se prosleđuje metodi `bindPopup()`. Parametar je običan tekst koji sadrži HTML kod koji će predstavljati sadržaj popupa. U ovakvoj situaciji, kada se klikne na poligon, biće prikazan popup (slika 15.8).



Slika 15.8. Primer popupa koji se prikazuje klikom na poligon

Popup je moguće i programabilno otvoriti, pozivanjem metode `openPopup()`:

```
marker.bindPopup("<b>Hello world!</b><br>I am a popup.").openPopup();
```

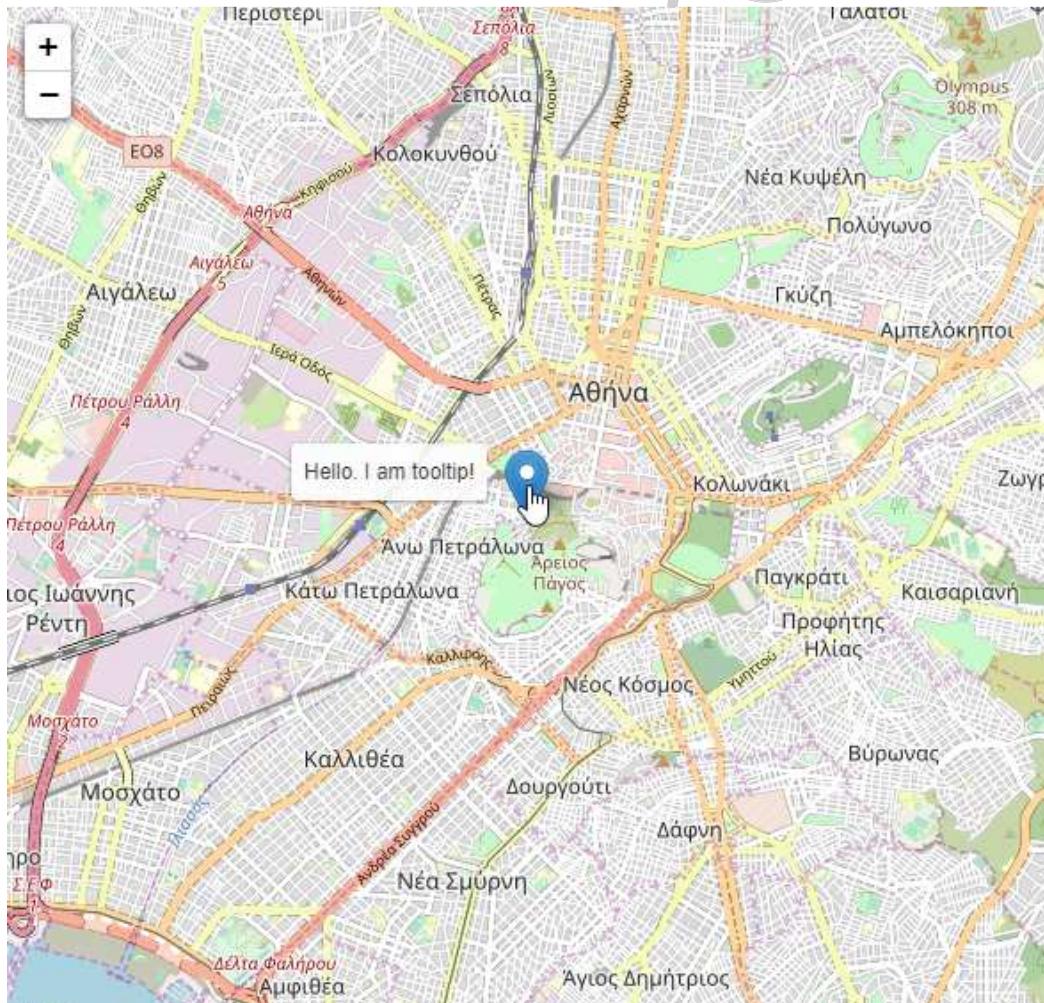
Tooltip

Tooltip je ugrađeni Leaflet element za prikaz poruka, nešto jednostavnijeg oblika. Ipak, osnovna osobina tooltipa jeste način na koji se inicira njegov prikaz. Tooltip se prikazuje kada se pokazivač miša nađe iznad elementa kome je tooltip pridružen:

```
var marker = L.marker([37.974, 23.72]).addTo(myMap);
marker.bindTooltip("Hello. I am tooltip!");
```

U prikazanom primeru je prvo kreiran jedan marker. Zatim je markeru pridružen tooltip, upotrebom metode `bindTooltip()`. Ovoj metodi je prosleđen tekst koji će se naći unutar tooltipa.

Prelaskom strelice miša preko kreiranog markera, doći će do pojave definisanog tooltipa, što je ilustrovano slikom 15.9.



Slika 15.9. Primer tooltipa koji se prikazuje prelaskom strelice miša preko markera

Samostalna obrada događaja

Prethodni redovi su ilustrovali primere obrade klik i hover događaja na elementima koji su nacrtani nad mapom, ali korišćenjem ugrađenih mehanizama koji omogućavaju prikaz popup i tooltip elemenata. Pored prikazanih pristupa, moguće je definisati i proizvoljnu logiku koja će se aktivirati prilikom dolaska do pojave nekog događaja na mapi. Na primer, kako bi se obradio click događaj na marker, dovoljno je napisati sledeće:

```
var marker = L.marker([37.974, 23.72]).addTo(myMap);

marker.on('click', function () {
    alert("You have clicked on marker!");
});
```

Za pretplatu na click događaj iskorišćena je on() metoda, koja je pozvana nad objektom markera. Prvim parametrom se definiše naziv događaja, a drugim parametrom logika koja će se aktivirati prilikom pojave takvog događaja. Logika je definisana korišćenjem anonimne funkcije. Na ovaj način, klikom na marker biva prikazan jedan modalni prozor sa porukom.

Elementi koji se crtaju nad mapom nisu jedini elementi čiji se događaji mogu slušati. Moguće je preplatiti se na događaje kompletne mape:

```
myMap.on('click', function (ev) {
    alert("Coordinates of clicked point is: " + ev.latlng.lat +
", " + ev.latlng.lng);
});
```

Sada je nad objektom mape (myMap) obavljena registracija logike koja će se aktivirati kada korisnik klikne bilo gde na površinu mape. Takođe, sada anonimna funkcija za obradu događaja prihvata i jedan parametar. Reč je o objektu događaja. Unutar objekta događaja mogu se naći neke dodatne informacije o događaju. Tako se u primeru koristi svojstvo latlng, unutar koga su objedinjene koordinate tačke na koju je korisnik kliknuo. Tako će svakim klikom na površinu mape biti prikazan modalni prozor sa informacijom o koordinatama klika.

Dinamička kontrola prikaza i nivoa približenja

Za kraj ove lekcije, biće prikazano i nekoliko pristupa kojima je moguće obavljati osnovnu kontrolu prikaza nivoa približenja dinamički, korišćenjem JavaScript koda. Metode čiji efekat će biti ilustrovan prikazane su tabelom 15.3.

Metoda	Opis
setZoom()	postavlja nivo približenja na osnovu prosleđene numeričke vrednosti u rasponu od 1 do 18
zoomIn()	povećavanja nivo približenja za onoliko koraka kolika je prosleđena numerička vrednost
zoomOut()	smanjuje nivo približenja za onoliko koraka kolika je prosleđena numerička vrednost
panTo()	centrira mapu na prosleđene koordinate
setView()	centrira mapu i postavlja nivo približenja
flyTo()	animirano centriira mapu i postavlja nivo približenja

Tabela 15.3. Metode za dinamičku kontrolu prikaza i nivoa približenja

Efekat metoda prikazanih tabelom 15.3. biće ilustrovan objedinjeno, u jednom primeru. Struktura HTML stranice će izgledati ovako:

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

<div>
    <button id="zoom">Set zoom to 5</button>
    <button id="zoom-in">Zoom in</button>
    <button id="zoom-out">Zoom out</button>
    <button id="pan">Show Belgrade</button>
    <button id="set-view">Show Tokyo with zoom level 13</button>
    <button id="fly">Fly to Athens</button>
</div>
```

Pored `div` elementa za prikaz mape, unutar HTML dokumenta je sada postavljen i skup `button` elemenata sa tekstrom koji oslikava namenu dugmeta. Klikom na neki od ovakvih `button` elemenata, biće obavljena odgovarajuća dinamička kontrola mape.

Kompletan JavaScript kod za prikaz i dinamičku kontrolu mape izgleda ovako:

```
var myMap = L.map('map', {
    center: [37.974, 23.72],
    zoom: 13
});

L.tileLayer('https://tile.openstreetmap.org/{z}/{x}/{y}.png', {
    attribution: '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors'
}).addTo(myMap);

let buttons = document.getElementsByTagName("button");

for (let i = 0; i < buttons.length; i++) {
    buttons[i].addEventListener("click", function () {
        let id = this.id;

        switch (id) {
            case "zoom":
                myMap.setZoom(5);
                break;
            case "zoom-in":
                myMap.zoomIn(1);
                break;
            case "zoom-out":
                myMap.zoomOut(1);
                break;
            case "pan":
                myMap.panTo([44.82556, 20.45112]);
                break;
            case "set-view":
                myMap.setView([35.69033, 139.77343], 13);
        }
    });
}
```

```

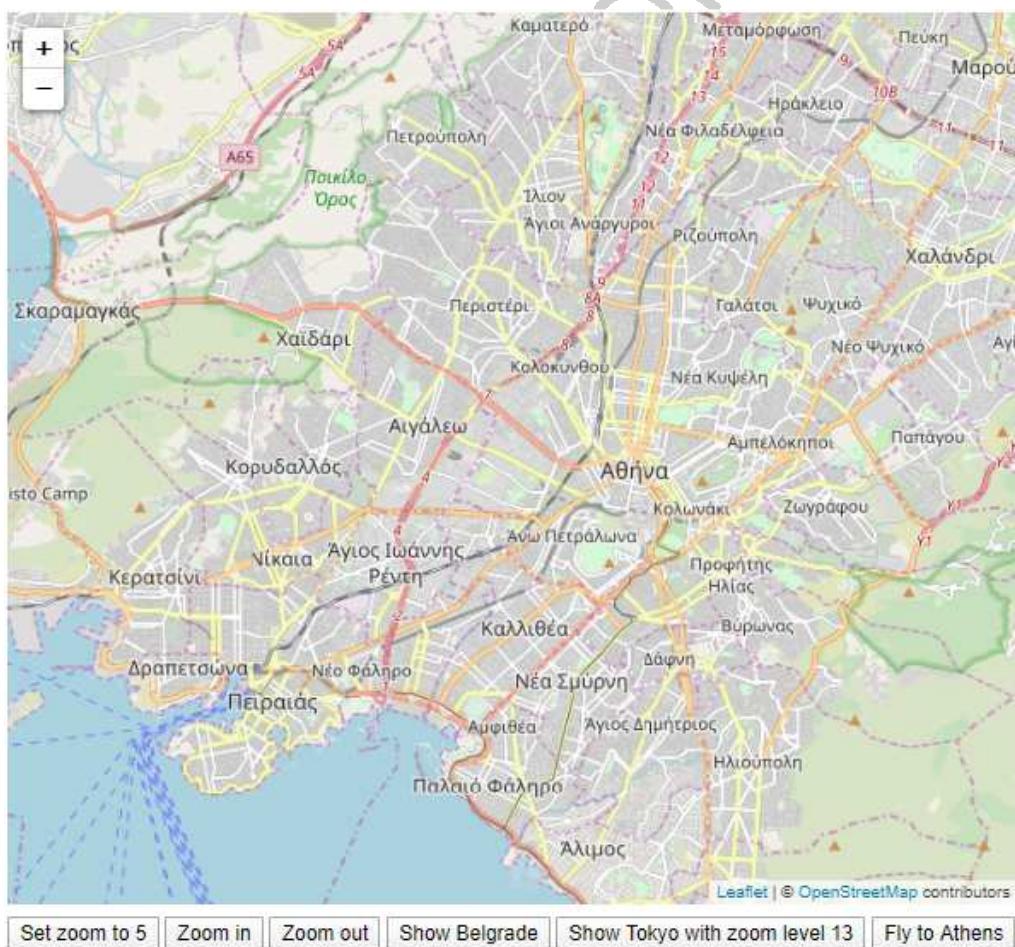
        break;
    case "fly":
        myMap.flyTo([ 37.95232, 23.72346 ], 15);
        break;
    }
}
);
}

```

JavaScript započinje već viđenim kodom za prikaz mape unutar `div` elementa sa `map` id-jem. Zatim su svi `button` elementi selektovani korišćenjem standardnog JavaScript pristupa i reference na takve elemente su smeštene unutar promenljive `buttons`.

For petljom je obavljen prolazak kroz niz `button` elemenata i preplata na `click` događaj za svako dugme. Prilikom klika na bilo koje dugme, aktivira se ista anonimna funkcija. Kako bi se znalo na koji `button` element je korisnik kliknuo, definisana je logika kojom se čita vrednost `id` atributa. Na osnovu vrednosti `id` atributa, unutar jedne switch uslovne konstrukcije, bira se odgovarajuća logika koja treba da se izvrši. Svaki od slučajeva unutar switch uslovne konstrukcije sadrži po jednu od metoda prikazanih tabelom 15.3.

Ovakav kod stvoriće prikaz kao na slici 15.10.



Slika 15.10. Izgled HTML dokumenta primera dinamičke kontrole mapa

Klikom na neki od button elemenata ispod mape prikazanih slikom 15.10, biće aktivirana odgovarajuća logika za dinamičko kontrolisanje mape.

Rezime

- Web mape digitalne mape koje se korisnicima prezentuju korišćenjem web pregledača.
- Prvi servis koji je ponudio web mape bio je MapQuest, 1996. godine.
- U osnovni modernih web mapa jeste pojam pločica (*tiles*), što su male, pravougaone slike čijim sastavljanjem se dobija kompletan prikaz mape.
- Leaflet je jedna od najpopularnijih JavaScript biblioteka otvorenog koda za rukovanje interaktivnim web mapama.
- Najpoznatiji provajderi web mapa su Google Maps, OpenStreetMap, Mapbox...
- Korišćenjem Leaflet biblioteke moguće je obaviti integraciju web mapa različitih provajdera.
- Leaflet poseduje bogat skup funkcionalnosti za crtanje po mapi, programabilnu kontrolu, geolociranje...