Web Programming
Web APIs

Krisztian Balog | University of Stavanger

Web APIs

- Set of methods and tools that can be used for building applications
- Server-side APIs
  - Making resources accessible by 3rd party systems
  - Based on request-response message system, typically HTTP-based
  - All major players provide APIs: Google, Facebook, Twitter, YouTube, etc.
  - Mashups: web applications that combine multiple server-side APIs
- Client-side web APIs
  - Commonly, browser extensions

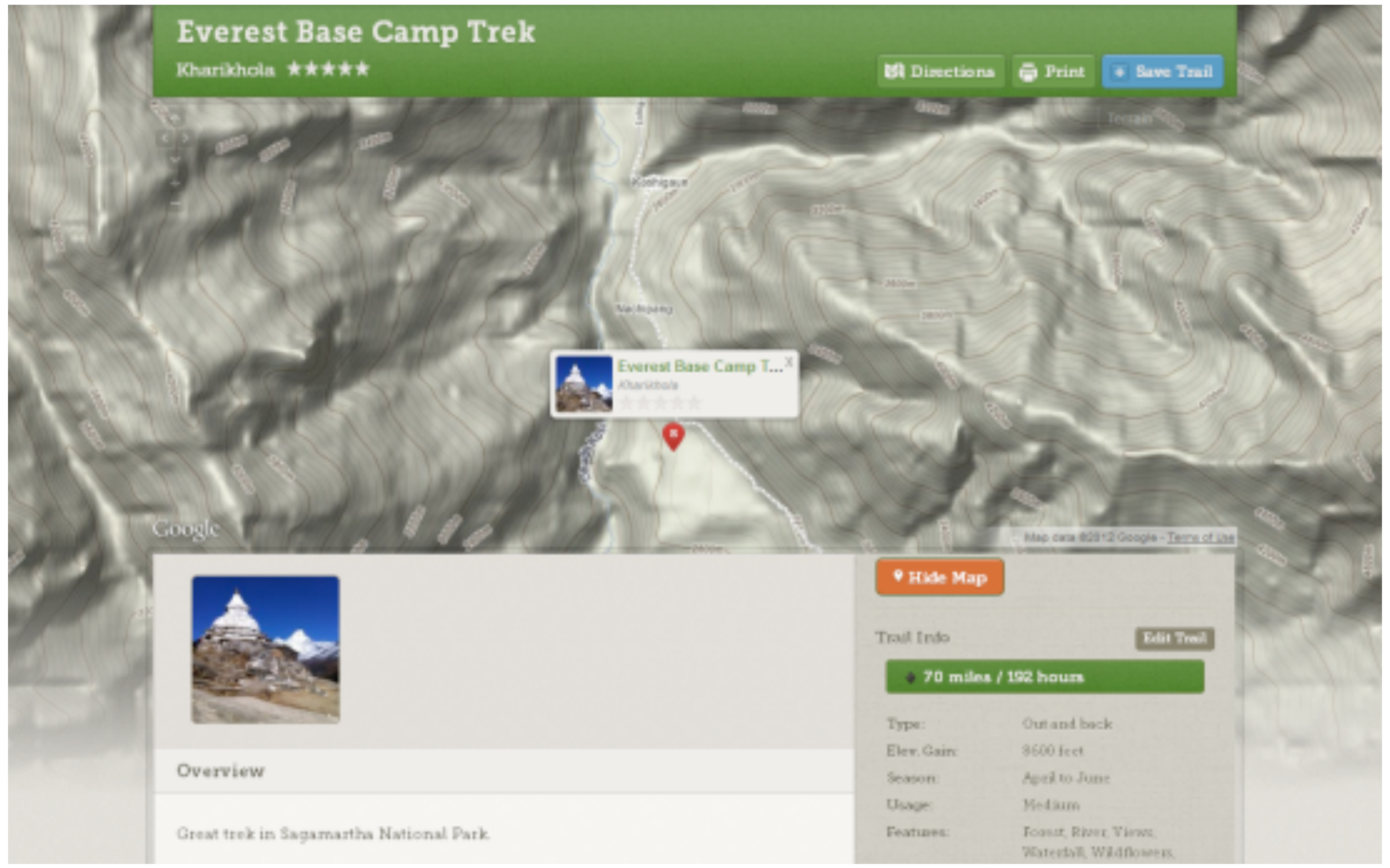
Server-side APIs

- JavaScript Web APIs
  - Inserting ready-made elements ("widgets") from a 3rd party
- RESTful Web APIs
  - Accessing data and/or services of a 3rd party

Google Maps

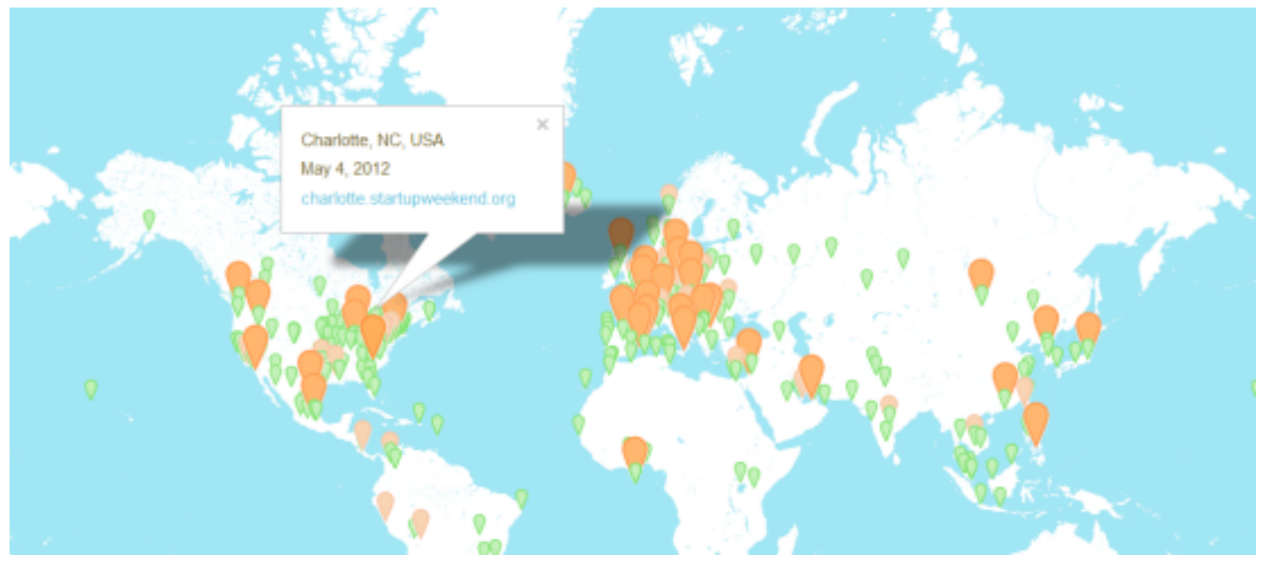
Google Maps API

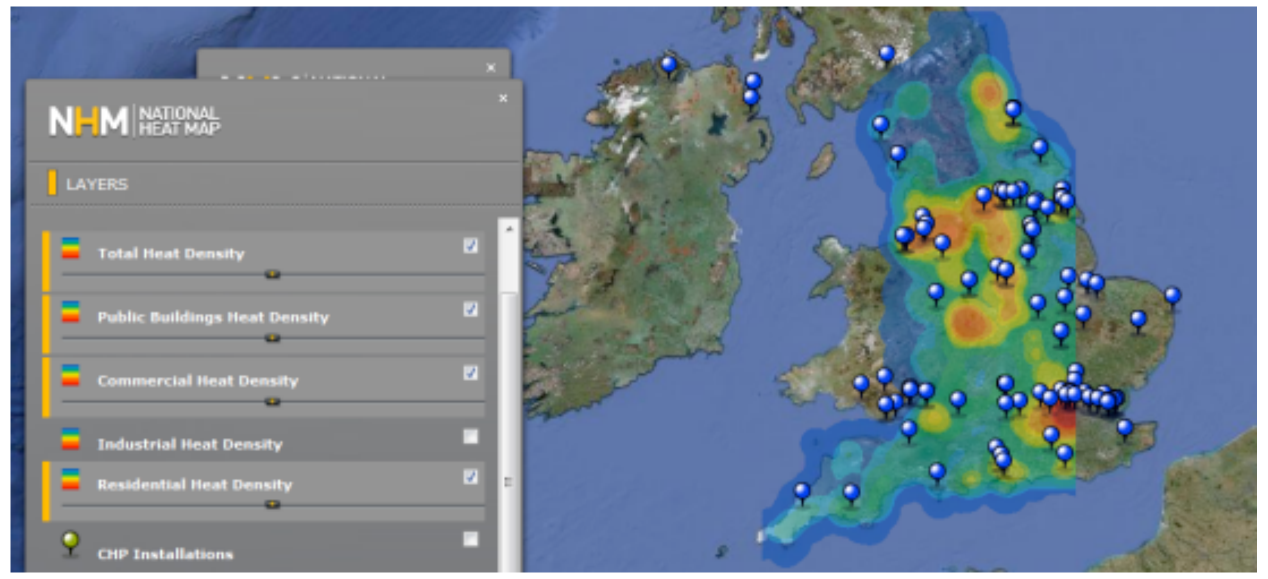
- Allows to customize maps and the information on the maps



Type: Out and back
Elev. Gain: 9600 feet
Season: April to June
Usage: Medium
Features: Forest, River, Views, Waterfall, Wildflowers

Great trek in Sagarmatha National Park

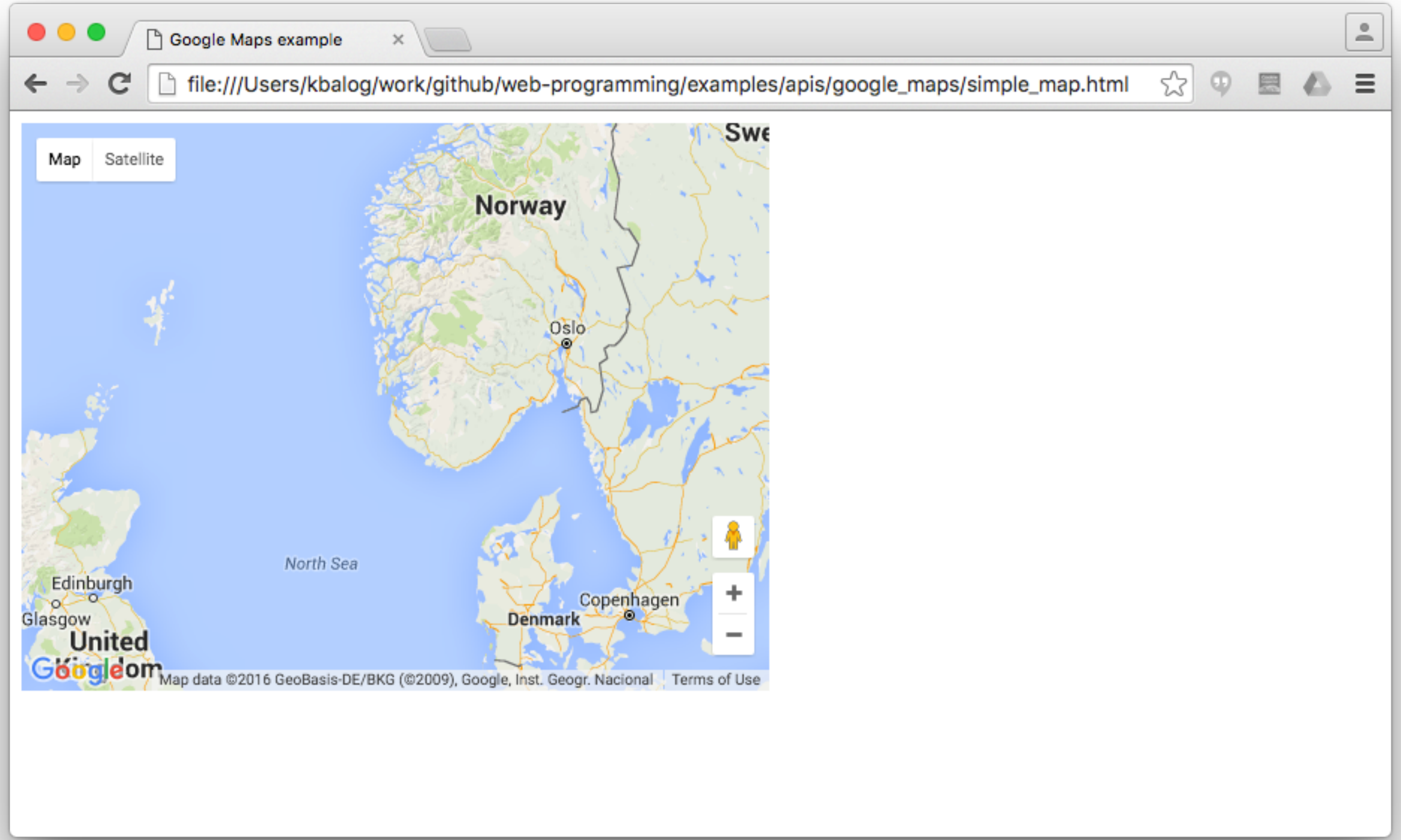




- See http://www.noupe.com/development/collection-of-the-coolest-uses-of-the-google-maps-api.html

Example

examples/apis/google_maps/simple_map.html


Google Maps example
file:///Users/kbalog/work/github/web-programming/examples/apis/google_maps/simple_map.html
Map
Satellite
Norway
Swe
Oslo
North Sea
Edinburgh
Glasgow
United Kingdom
Copenhagen
Denmark
Google
Map data ©2016 GeoBasis-DE/BKG (©2009), Google, Inst. Geogr. Nacional
Terms of Use

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Google Maps example</title>
    <script src="http://maps.googleapis.com/maps/api/js"></script>
    <script>
        function initialize() {
            var mapProp = {
                center: new google.maps.LatLng(58.9700, 5.7331),
                zoom: 5,
                mapTypeId: google.maps.MapTypeId.ROADMAP
            };
            var map = new google.maps.Map(
                      document.getElementById("googleMap"), mapProp);
        }
        google.maps.event.addDomListener(window, 'load', initialize);
    </script>
</head>
<body>
<div id="googleMap" style="width:500px;height:380px;"></div>
</body>
</html>
```

Load the Google Maps JavaScript library

Set map properties

Create a Map object

Execute the initialize() function upon page load

Div element to hold the map

Map types

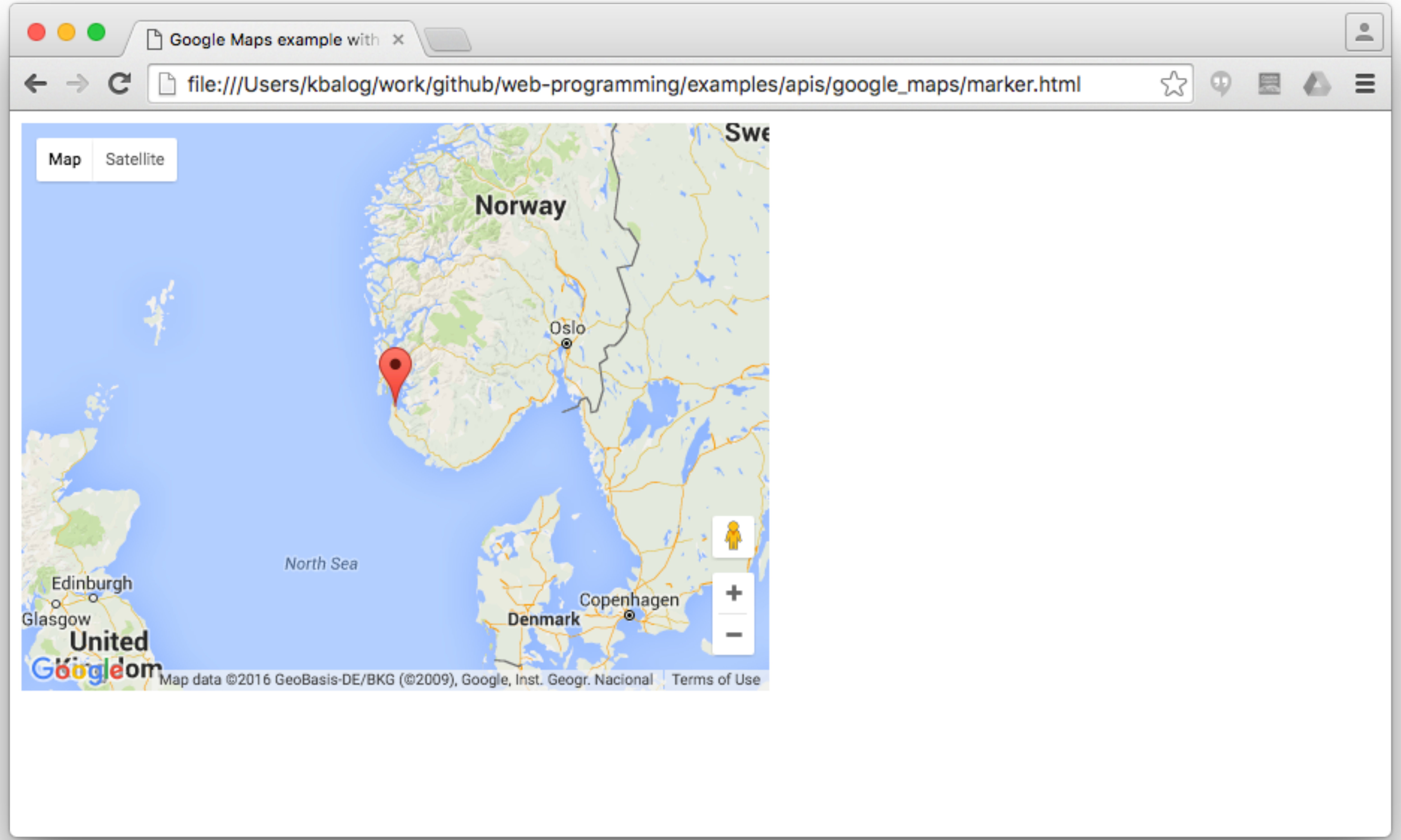
- **`mapTypeId`** specifies the map type to display
  - **`ROADMAP`** — normal, default 2D map
  - **`SATELLITE`** — photographic map
  - **`HYBRID`** — photographic map + roads and city names
  - **`TERRAIN`** — map with mountains, rivers, etc.

Drawing on the map

- Overlays are objects on a map that are bound to latitude/longitude coordinates
- Types of overlays
  - Marker — single locations; can also display custom icon images
  - Polyline — series of straight lines
  - Polygon — series of straight lines on a map, and the shape is "closed"
  - Circle and rectangle
  - Info windows — content within a popup balloon on top of a map
  - Custom overlays

Example

examples/apis/google_maps/marker.html


Google Maps example with
file:///Users/kbalog/work/github/web-programming/examples/apis/google_maps/marker.html
Map
Satellite
Norway
Swe
Oslo
North Sea
Edinburgh
Glasgow
United Kingdom
Copenhagen
Denmark
Map data ©2016 GeoBasis-DE/BKG (©2009), Google, Inst. Geogr. Nacional
Terms of Use

Example

examples/apis/google_maps/marker.html

```
<script>
    function initialize() {
        var locStavanger = new google.maps.LatLng(58.9700, 5.7331);
        var mapProp = {
            center: locStavanger,
            zoom: 5,
            mapTypeId: google.maps.MapTypeId.ROADMAP
        };
        var map = new google.maps.Map(
                  document.getElementById("googleMap"), mapProp);

        // marker for Stavanger
        var marker = new google.maps.Marker({
            position: locStavanger
        });
        marker.setMap(map);
    }
    google.maps.event.addDomListener(window, 'load', initialize);
</script>
```

The Marker constructor creates a marker (the position property must be set!)

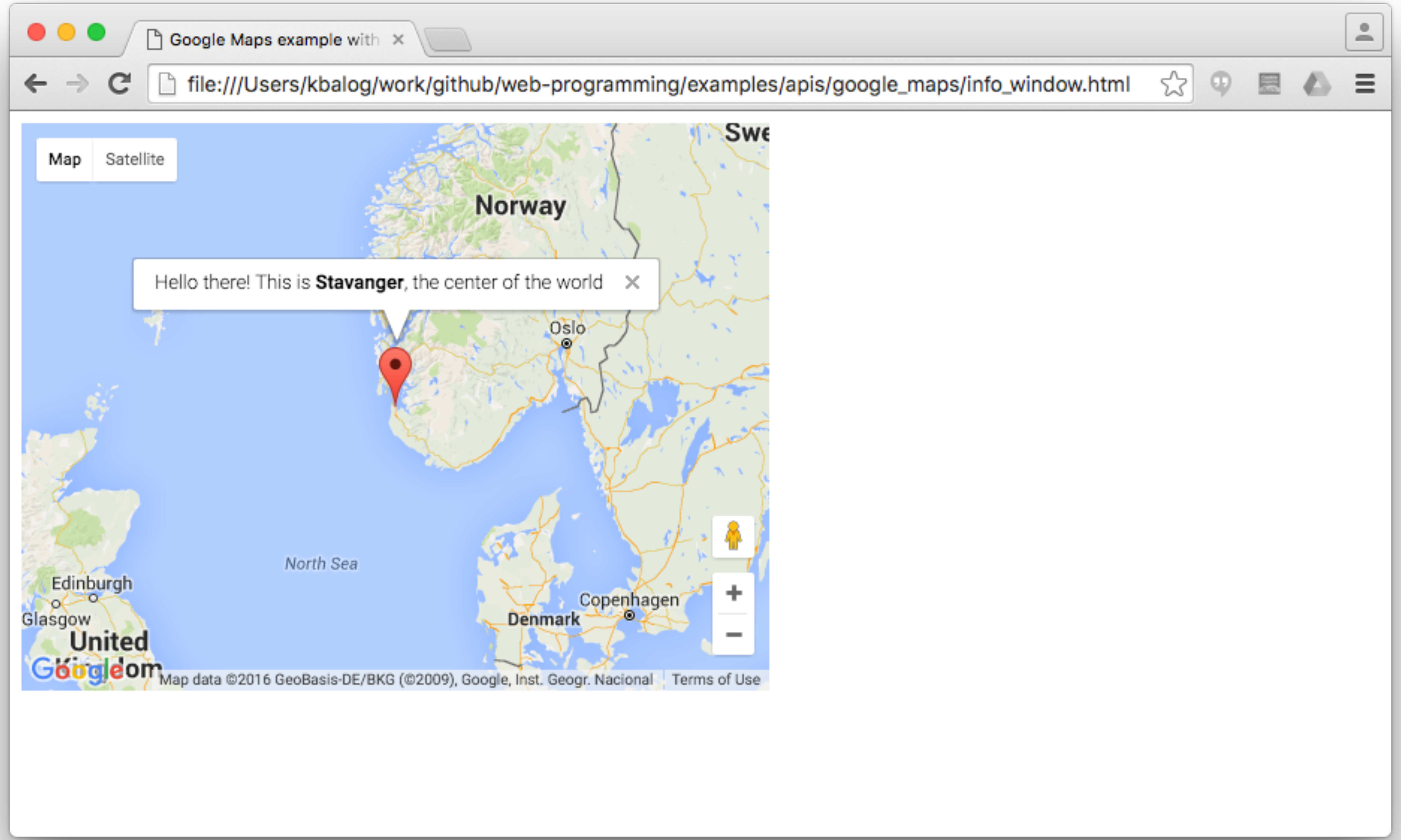
Add the marker to the map

Info window

- An **`InfoWindow`** diplays content (usually text or images) in a popup window above the map at a given location
- Typically, an info window is attached to a marker

Example

examples/apis/google_maps/info_window.html


Google Maps example with

file:///Users/kbalog/work/github/web-programming/examples/apis/google_maps/info_window.html

Hello there! This is **Stavanger**, the center of the world

Example

examples/apis/google_maps/marker.html

```
<script>
    function initialize() {
        var map = new google.maps.Map(…);

        var marker = new google.maps.Marker({
            position: locStavanger
        });
        marker.setMap(map);

        var contentString = "Hello there! This is <strong>Stavanger</strong>,
                             the center of the world";
        var infowindow = new google.maps.InfoWindow({
            content: contentString
        });
        marker.addListener('click', function() {
            infowindow.open(map, marker);
        });
    }
    google.maps.event.addDomListener(window, 'load', initialize);
</script>
```

Create info window

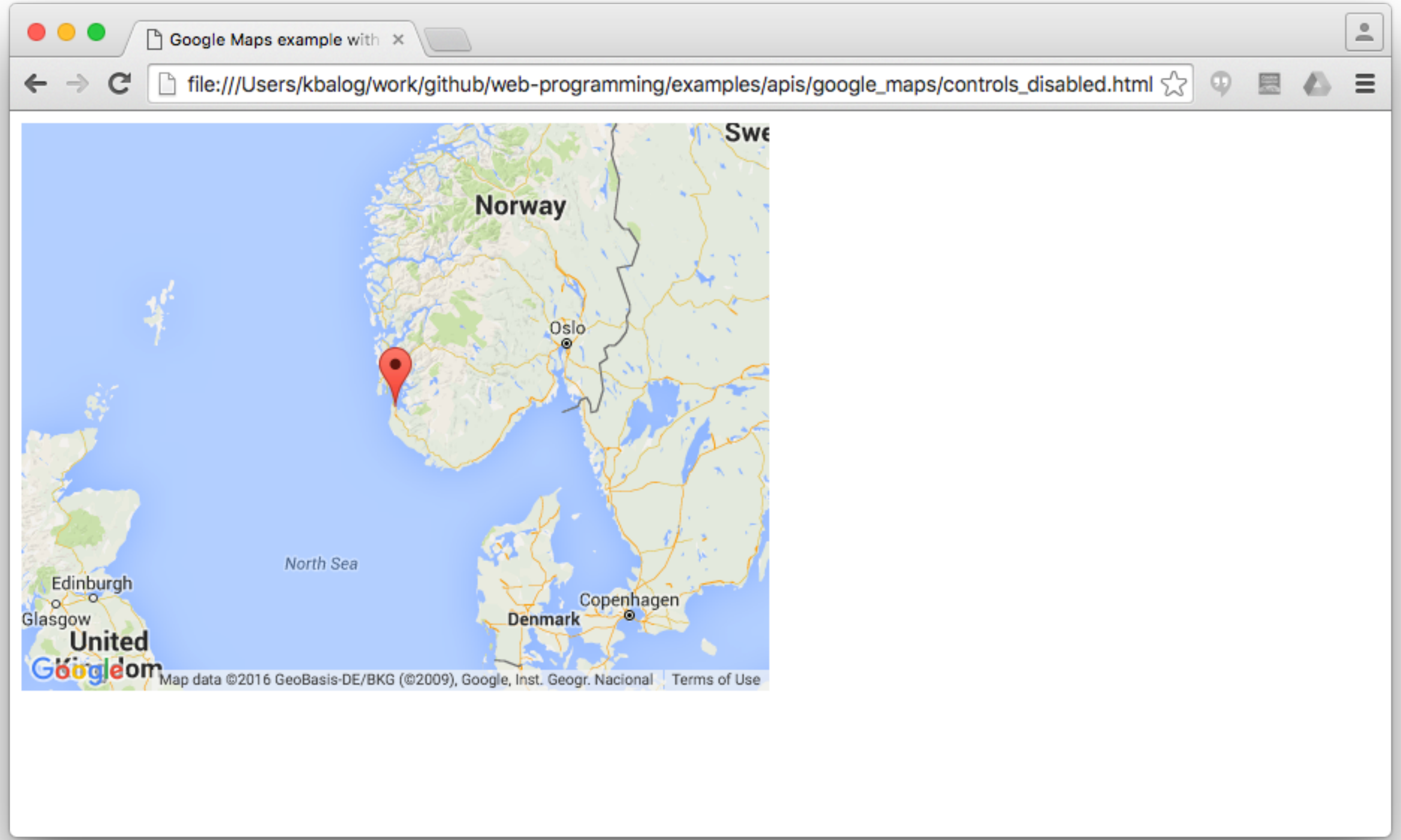
Assign info window to the marker’s click event

Controls

- Default control set:
  - Zoom — displays a slider or "+/-" buttons to control the zoom level
  - MapType — lets the user toggle between map types (roadmap/satellite)
  - Street view — icon which can be dragged to the map to enable Street view
- In addition to the default controls, Google Maps also has:
  - Scale — displays a map scale element
  - Rotate — allows you to rotate maps
  - Overview map — thumbnail overview map

Example

examples/apis/google_maps/controls_disabled.html


Google Maps example with
file:///Users/kbalog/work/github/web-programming/examples/apis/google_maps/controls_disabled.html
Norway
Swe
Oslo
North Sea
Edinburgh
Glasgow
United Kingdom
Copenhagen
Denmark
Google
Map data ©2016 GeoBasis-DE/BKG (©2009), Google, Inst. Geogr. Nacional
Terms of Use

Example

examples/apis/google_maps/controls_disabled.html

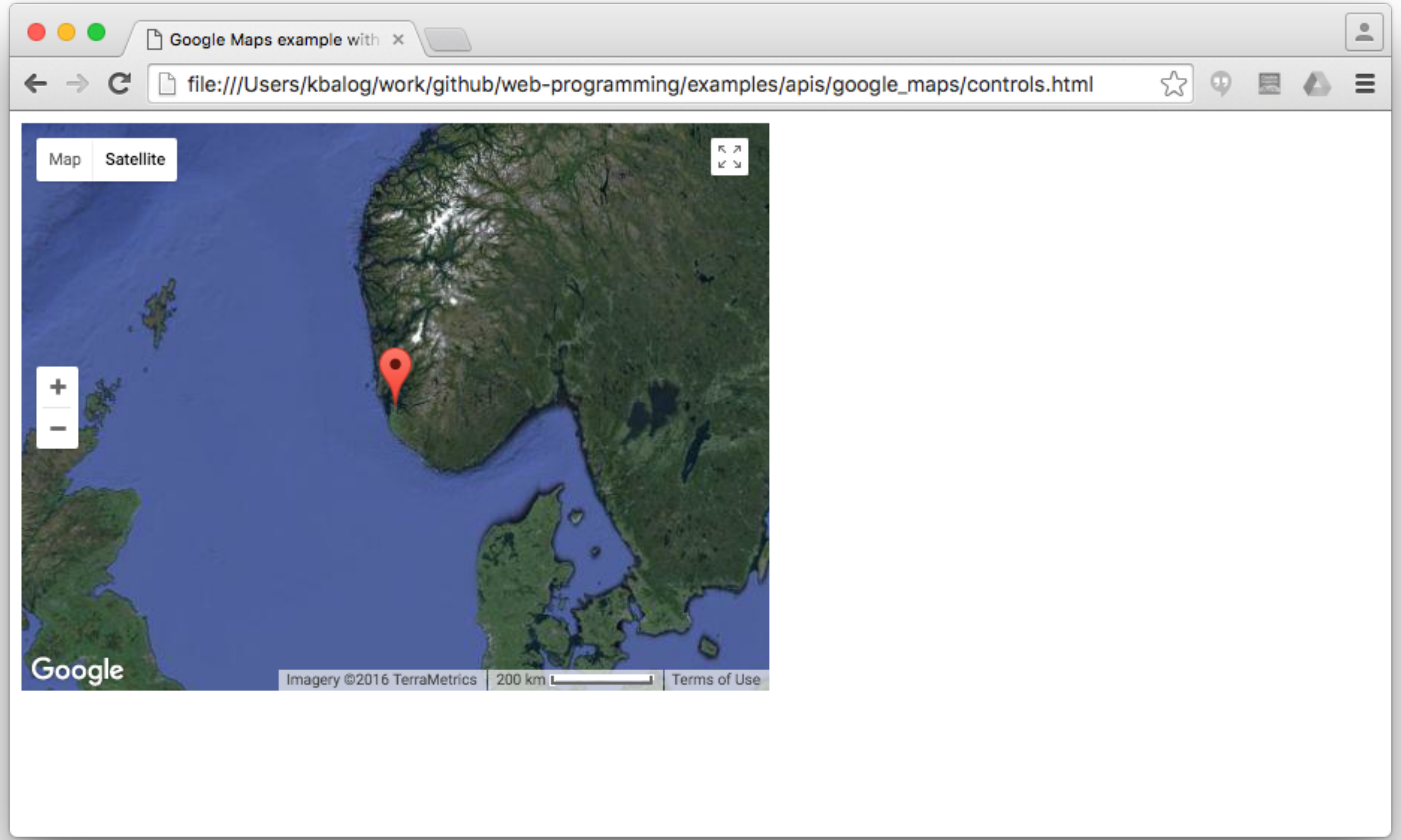
```
<script>
    function initialize() {
        var locStavanger = new google.maps.LatLng(58.9700, 5.7331);
        var mapProp = {
            center: locStavanger,
            zoom: 5,
            disableDefaultUI: true,
            mapTypeId: google.maps.MapT
        };
        var map = new google.maps.Map(
                document.getElementById("googleMap"), mapProp);
    }

    google.maps.event.addDomListener(window, 'load', initialize);
</script>
```

Default UI disabled

Example

examples/apis/google_maps/controls.html


Google Maps example with
file:///Users/kbalog/work/github/web-programming/examples/apis/google_maps/controls.html
Map
Satellite
+
−
Google
Imagery ©2016 TerraMetrics
200 km
Terms of Use

Example

examples/apis/google_maps/controls.html

```
<script>
    function initialize() {
        var locStavanger = new google.maps.LatLng(58.9700, 5.7331);
        var mapProp = {
            center: locStavanger,
            zoom: 5,
            mapTypeId: google.maps.MapTypeId.SATELLITE,
            zoomControl: true,
            zoomControlOptions: {
                position: google.maps.ControlPosition.LEFT_CENTER
            },
            scaleControl: true,
            streetViewControl: false,
            overviewMapControl: true,
            fullscreenControl: true
        };
        var map = new google.maps.Map(
                document.getElementById("googleMap"), mapProp);
    }
    google.maps.event.addDomListener(window, 'load', initialize);
</script>
```

Exercises #1, #2

https://github.com/kbalog/web-programming/tree/master/
exercises/apis

RESTful Web APIs

REST

- **RE**presentational **S**tate **T**ransfer
- REST is an architectural style (not a protocol)
  - Web service APIs are called RESTful
- Uniform interface separates clients from servers
  - Data storage is internal to the server
  - Servers are not concerned with the user's state
- Stateless
  - The client must provide all the information for the server to fulfill the request. No sessions.

Uniform interface

- Resources are identified by URIs
- Operations are performed on resources
- Resources are manipulated through representations
    - Representation contains enough information for the client to modify/delete it on the server
    - Representations are typically in JSON or XML format

RESTful web APIs

- HTTP based
- Resources are identified by URIs
  - E.g., http://example.com/resources/
- Operations correspond to standard HTTP methods
  - GET, PUT, POST, DELETE
- Media type is JSON

Typical RESTful API

	GET	PUT	POST	DELETE
Collection URI `http://example.com/resources`	**List** elements	**Replace** the entire collection	**Create** a new element in the collection	**Delete** the entire collection
Element URI `http://example.com/resources/item17`	**Retrieve** the representation of an element	**Replace** element or **create** if it doesn't exist	generally not used	**Delete** the element

Making HTTP requests

- How to make HTTP requests?
  - JavaScript: using **`XMLHttpRequest`** object
  - jQuery: **`$.ajax()`**, **`$.post()`**, **`$.get()`**
  - Python: **`requests.get()`**

HTTP requests in jQuery

- Using the **`$.ajax()`** method

```
$.ajax({
    url: '/script.cgi',
    type: 'DELETE',
    success: function(result) {
        // Do something with the result
    }
});
```

- This won’t work cross-domain because of the same-origin policy!

Same-origin policy

- A script in one page can only access data in a second web page, if both have the same origin
  - Same protocol, host, and port
- Workarounds
  - Request data from using a server-side script
  - JSONP

JSONP

- "JSON with padding"
- A workaround to be able to request data from a server in a different domain
  - Relies on the fact that browsers don’t enforce the same-origin policy on <script> tags
  - The server wraps the response with a callback function
- The server must know how to respond with JSONP-formatted results!
- JSONP is limited to GET requests!

JSONP

- Example request
  `http://www.example.net/sample.aspx?callback=mycallback`
- Normal JSON response

```
{ foo: 'bar' }
```

- JSONP response

```
mycallback({ foo: 'bar' });
```

  - Since the request came from inside a **`<script>`** tag, it will be executed

JSONP request in JavaScript

```
<!-- Request sent via a script tag -->
<script src="https://status.github.com/api/status.json?callback=apiStatus">
</script>

<!-- Data received as an execution of the predefined function. -->
<script> function apiStatus(data) {
    console.log(data.status);
}
</script>
```

JSONP request in jQuery

```
<script>
    $.getJSON("http://api_url.com?jsoncallback=?",
            {
                param1: "value1",
                param2: "value2"
            },
            function (data) {
                // processing data
            });
</script>
```

RESTful web APIs

Authentication

- Sending authorized requests to an API
- OAuth protocol
  http://oauth.net/
- Application-only authentication
  - Application makes API requests on its own behalf, without a user context
- Application-user authentication
  - Making API calls on behalf of a user
  - Identify the user’s identity (and permission) in addition to the application’s identity

Flickr

Flickr

- Flickr’s public feed is available as JSON
  - https://www.flickr.com/services/feeds/docs/photos_public/
- Typical Flickr JSON object

```
"items": [
         {
          "title": "View from the hotel",
          "link": "http://www.flickr.com/photos/33112458@N08/[...]",
          "media": {"m":"http://farm4.static.flickr.com/[...]4a6569750c_m.jpg"},
          "date_taken": "2008-12-04T04:43:03-08:00",
          "description": "Talk On Travel[…]",
          "published": "2008-12-04T12:43:03Z",
          "author": "nobody@flickr.com (Talk On Travel)",
          "author_id": "33112458@N08",
          "tags": "spain dolphins tenerife canaries lagomera aqualand […]"
         }
         …
```

Exercises #3, #4

https://github.com/kbalog/web-programming/tree/master/
exercises/apis

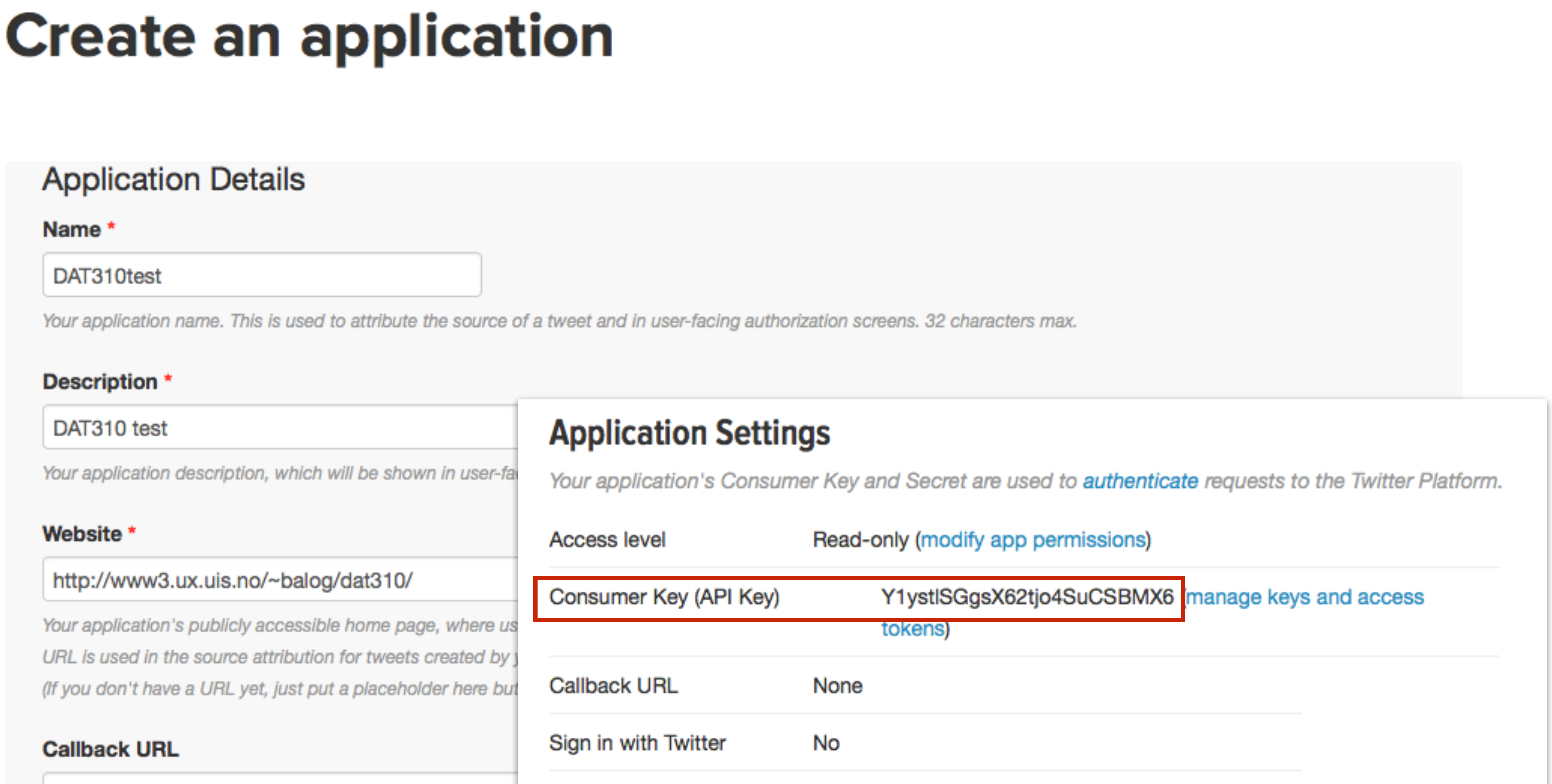
Twitter

Twitter Search API

- Behaves similarly like the search feature in Twitter mobile/web clients
- API endpoint: **`GET search/tweets`**
- Requires (application-only) authentication
  https://dev.twitter.com/oauth/application-only
- Create an application to get an API key
  https://dev.twitter.com/apps

GET API key

https://dev.twitter.com/apps



Create an application

Application Details

Name *

DAT310test

Your application name. This is used to attribute the source of a tweet and in user-facing authorization screens. 32 characters max.

Description *

DAT310 test

Your application description, which will be shown in user-fa

Website *

http://www3.ux.uis.no/~balog/dat310/

Your application's publicly accessible home page, where us
URL is used in the source attribution for tweets created by y
(If you don't have a URL yet, just put a placeholder here but

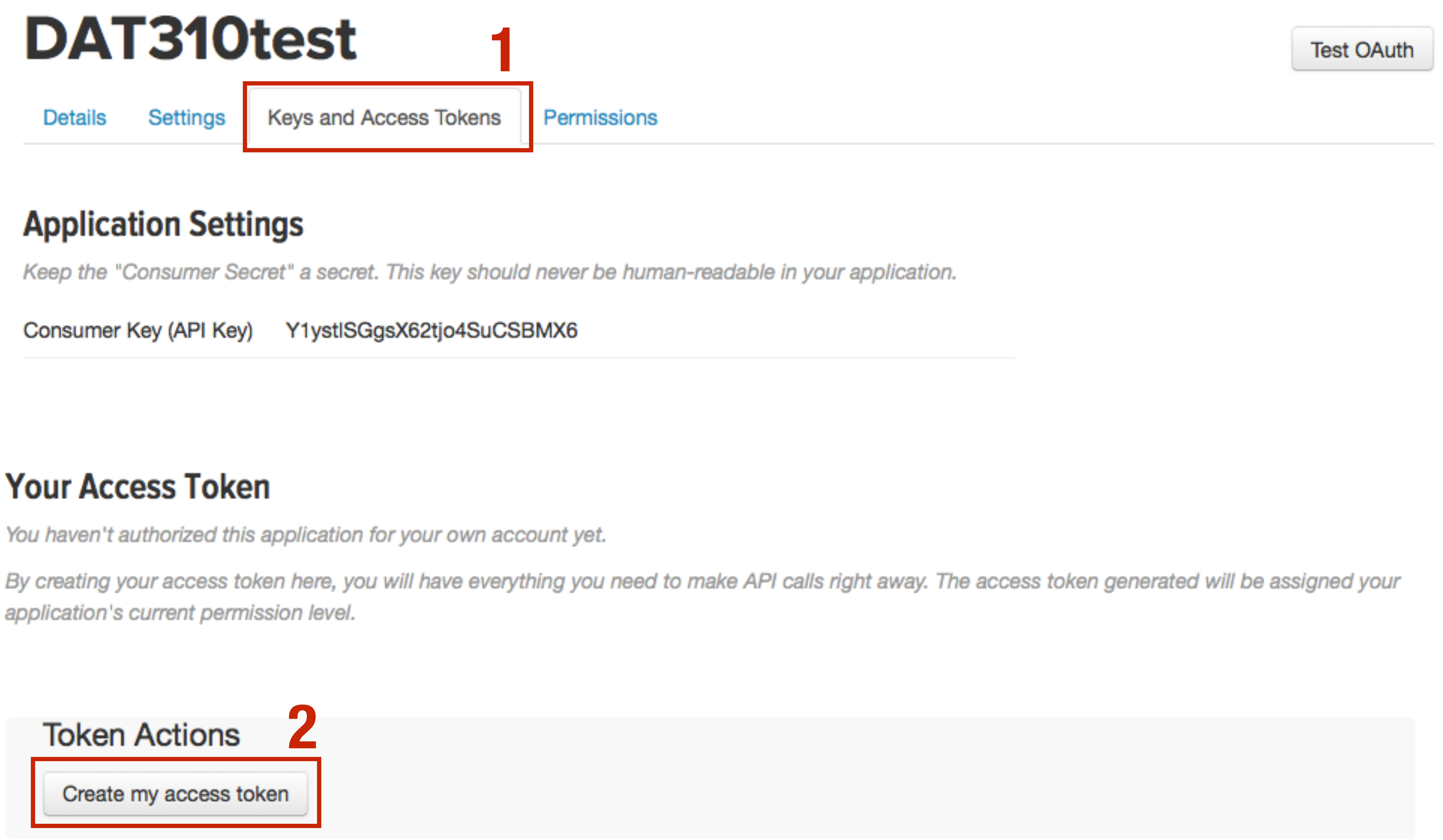
Callback URL

Application Settings

Your application's Consumer Key and Secret are used to authenticate requests to the Twitter Platform.

Access level	Read-only (modify app permissions)
Consumer Key (API Key)	Y1ystlSGgsX62tjo4SuCSBMX6 (manage keys and access tokens)
Callback URL	None
Sign in with Twitter	No

Get OAuth access token



Application Settings

Keep the "Consumer Secret" a secret. This key should never be human-readable in your application.

Consumer Key (API Key) Y1ystlSGgsX62tjo4SuCSBMX6

Your Access Token

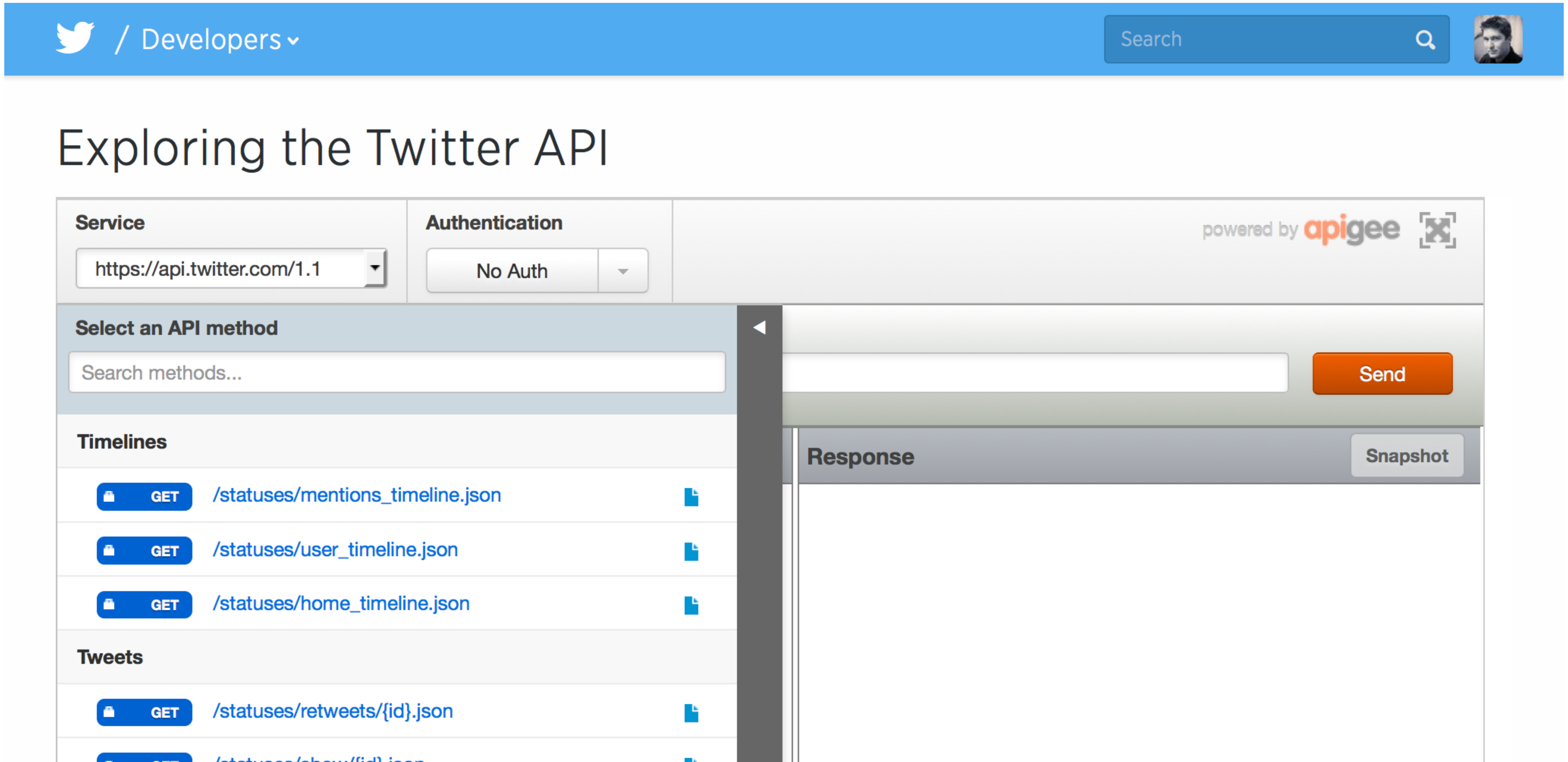
You haven't authorized this application for your own account yet.

By creating your access token here, you will have everything you need to make API calls right away. The access token generated will be assigned your application's current permission level.

Token Actions

Twitter API console tool

https://dev.twitter.com/rest/tools/console



Exploring the Twitter API

Service: https://api.twitter.com/1.1

Authentication: No Auth

Select an API method

Timelines

- GET /statuses/mentions_timeline.json
- GET /statuses/user_timeline.json
- GET /statuses/home_timeline.json

Tweets

- GET /statuses/retweets/{id}.json

References

- Google Maps API
  - https://developers.google.com/maps/documentation/javascript/
  - https://developers.google.com/maps/tutorials/
  - http://www.w3schools.com/googleapi/
- REST API tutorial
  - http://www.restapitutorial.com/
- OAuth (Twitter’s documentation)
  - https://dev.twitter.com/oauth