KTH Royal Institute of Technology
Urban Planning and Environment - Geoinformatics

Lab 1: Simple Web Maps

Due: See Assignment schedule on Canvas

Created by: Győző Gidófalvi

Teacher: Győző Gidófalvi

Updated by: Ehsan Saqib, Silvino Pedro Cumbane, Can Yang

Goal

The auxiliary goal of this lab is to set up an environment for commonly used web development tools. In this exercise, you will learn

1. Develop a web app to serve basic html files, images using node.js.
2. Create interactive web maps that uses the HTML imagemap to link areas of the map to various multimedia information on the web;
3. Use the Google Static Maps API to create static maps with pictorial representation of geographic information.
4. Create more dynamic and feature-rich web maps using the Google Maps JavaScript API.

Preparation - Setting up your Development Environment

The following tools and developments environments will be used in this or future labs and the project and hence this sections explain how to set them up.

- **Text editor:** Make sure that a proper text editor is installed on your computer. Notepad++, Sublime and Visual Studio Code are among the most popular and widely used text editors. Read the documentation for Notepad++ at http://notepad-plus-plus.org/, documentation for the Sublime3 at https://www.sublimetext.com/3 and documentation for Visual Studio Code at https://code.visualstudio.com.
- **Command line shell:** Familiarity with the command-line shell will be essential for the exercises. In Windows a cmd window or power shell with admin privileges would be needed. On a Mac or in Linux, a terminal window can be used. Please get familiar with the "sudo" command in OS X and Linux.
- **Browser:** In this course we will mainly use Chrome as the browser in all the exercises. However, you may use your preferred browser. Please note that not all browsers may support all the HTML5 features to the same extent. You might encounter problems when using other browsers. Therefore, we strongly urgue you to use the latest Chrome browser for the exercises in this course so that any problems are minimized.
- **Node.js and NPM:** NodeJS is a popular Javascript based server framework and NPM is Node Package Manager. To learn more about NodeJS, you can visit https://nodejs.org.
- To install Node on your machine, go to https://nodejs.org and click on the Download button. Depending on your computer's platform (Windows, MacOS or Linux), the appropriate installation package is downloaded.
- To ensure that your NodeJS setup is working correctly, type the following at the command prompt to check for the version of Node and NPM.

```
node -v
npm -v
```

  With npm, you can install a lot of powerful libraries in JavaScript.
- **Google Cloud Platform (GCP):** GCP is a suite of cloud services hosted on Google's infrastructure. GCP offers a wide variety of services and APIs that can be integrated with any cloud-computing application or project such as computing and storage, data analytics, machine learning, networking, etc.

  Register GCP go to https://cloud.google.com/free/.

  In this lab **Google Maps JavaScript API** will be used. You need to obtain a Google Maps JavaScript API key in order to use that service. Follow the tutorial at:
  https://developers.google.com/maps/documentation/maps-static/intro

  1. Click the *get started* button. You are directed to web page called 'Enable Google Maps Platform'. Tick the Maps box and continue.
  2. Select an existing project or create a new one. Click 'Next'.
  3. You are directed to the 'Set the billing account for project' dialog. Follow the instructions.
  4. Enable all the 7 APIs and create an API key. The key is a long string like AIzaSyC...

  Make sure that after you replace the YOUR_API_KEY in the following link with your API key, you see a static map.
  https://maps.googleapis.com/maps/api/staticmap?center=59.34849,18.07176
  &zoom=12&size=512x512&maptype=terrain&key=YOUR_API_KEY

References

- Express framework https://expressjs.com/
- An online tool to create imagemaps: https://www.image-map.net/
- An introduction to Google Static Maps:
  https://developers.google.com/maps/documentation/static-maps/intro

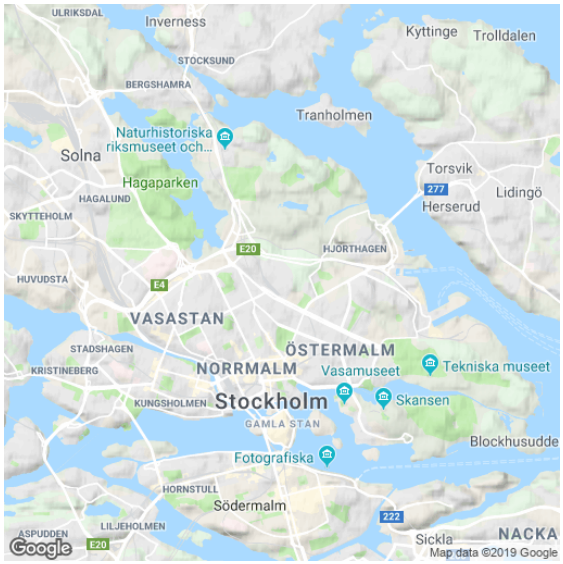

Figure 0.1: Static map

- W3Schools Google Maps JavaScript API tutorial:
  https://www.w3schools.com/graphics/google_maps_intro.asp
- Google Maps JavaScript API Guide:
  https://developers.google.com/maps/documentation/javascript/tutorial

1 My First App

- At a convenient location on your computer, create a folder named MyFirstAPP.
- Open a command prompt and move to the folder MyFirstAPP.
  1. To initialize a new npm web app, type the following command in the command prompt and fill up all the required information:

     ```
     1  npm init
     ```

  2. Install express, which is a lightweight and fast web framework for Node.js.
     Install it with

     ```
     1  npm install express --save
     ```

  3. Create a file called 'app.js' with the content of

     ```
     1  const express = require('express')
     2  const app = express()
     3  const port = 3000
     4  app.get('/', (req, res) => res.send('This is my
            first app part 1!'))
     5  app.listen(port, () => console.log(`Example app
            listening on port ${port}!`))
     ```

  4. With the above script entry, you can then start the server by typing the following command at the command prompt:

```
node app.js
```

Open the browser manually and type *http://localhost:3000* followed by Enter. Now you can see your page. Stop the server by pressing Ctrl+c.

5. Routing to an HTML file on your server. Add the following lines to the app.js file (between line 4 and 5) and paste the lab1part1.html to the project folder.

```
app.get('/lab1part1', (req, res) => {
  res.sendFile(__dirname + '/lab1part1.html');
})
```

Restart the server and visit http://localhost:3000/lab1part1.

6. Serving static files on your server. To access static files on your server, you need to specify a folder path.
   a) Create a folder called *public* under your project and paste the map.png into that folder.
   b) Modify the *app.js* file to add the following line

```
app.use(express.static('public'))
```

   c) In the *lab1part1.html* file. Add an extra element for the image inside the HTML element as

```
<img src="map.png">
```

   Hint: you can find more information about HTML elements at https://www.w3schools.com/html/html_images.asp

Finally, the content of *app.js* is

```
const express = require('express')
const app = express()
const port = 3000

app.use(express.static('public'))

app.get('/', (req, res) => res.send('Hello! This is my first app part 1!'))

app.get('/lab1part1', (req, res) => {
res.sendFile(__dirname + '/lab1part1.html');
})

app.listen(port, () => console.log(`Example app listening on port ${port}!`))
```

The *lab1part1.html* is

```
<html>
<h1>This is my first app!</h1>
<img src="map.png">
</html>
```

Run the server with

```
node app.js
```

You should be able to see a page like Figure 1.1.

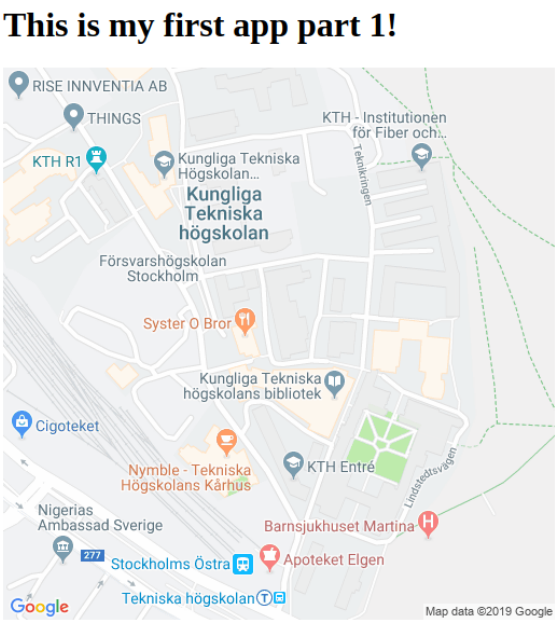

Figure 1.1: Web page of part 1.

Tasks:

- Briefly explain the content of the *app.js*, which will be useful for you to set up your own server.
- Create two files called *lab1part2.html* and *lab1part3.html* and update the scripts to support this functionality. Visit *http://localhost:3000/lab1part2* and *http://localhost:3000/lab1part3* we can see the same image but the header is updated with the new part number.

2 Imagemaps

In part 1, we have created a static image as a map shown on a web page. In this part, we will add some simple interactions to the web page using imagemap.

An imagemap is a graphic image where a user can click on different parts of the image and be directed to different destinations. See the example at https://www.w3schools.com/tags/tag_map.asp.

Imagemaps are made by defining each of the hot areas in terms of their x and y coordinates (relative to the top left hand corner). With each set of coordinates, you specify a link that users will be directed to when they click within the area. As an example, say you have a map of the World that you wish to act as an image map. Each country could have their hot areas defined on the map to take you to different pages.

The basic concepts of an imagemap is illustrated in the listing below. The `<map>` tag is used to define an imagemap with its `name` attribute set to `"image-map"`. Inside the `<map>` element using `<area>` tags a number of areas of different types of `shapes` (`"rect"`, `"circle"`, `"poly"`) are defined through their coordinates and are linked to various multimedia documents (HTML documents, images, sound and video files). Finally, the `usemap="#image-map"` inside the `<img>` tag associates the reference image with the imagemap.

Tasks:

Visit this link to download an image
https://maps.googleapis.com/maps/api/staticmap?center=59.34849,18.07176&zoom=16&size=512x512&key=key=YOUR_API_KEY

or use the *map.png* file provided in the lab to create an interactive web map using https://www.image-map.net/. Define areas for at least 3 important sites on the image and link to relevant web documents (pages or links).

Hint: You can copy the generated HTML code into the lab1part2.html file.

When you finish this part, make necessary screenshots and put in the report. You can also show your web page to the TA.

3 Google Static Maps API

In this part, you will get familiar with the Google Static Maps API, which enables you to embed a Google Maps *image* on your web page without requiring JavaScript or any dynamic page loading. The Google Static Maps API service creates your map based on URL parameters sent through a standard HTTP request and returns the map as an image you can display on your web page.

Open a web browser like Firefox of Chrome, copy-and-paste the following code as a single into the address bar or simply click this link:

```
https://maps.googleapis.com/maps/api/staticmap?center=59.34849,18.07176&zoom=12&
size=512x512&maptype=terrain&key=YOUR_API_KEY
```

You should see a map over the Stockholm city in your browser. This is a "static map" which is basically just a map image over the requested area.

What you pasted into the address bar is called an HTTP request. The HTTP *request* is sent to the Google *server* and based on the parameters that you specified in the request, the server sends back a *response* which in this example is the map image over the city of Stockholm. This process of forming a request → sending it to server → receiving a response is the very foundation of how a *client-server architecture* and the Internet works.

Now let's take a closer look at the HTTP request you just sent. In that request you can find a question mark (`?`) after which you specify different parameters. Several parameters can be specified but they should be separated from each other by the (`&`).

Look closely at that request, can you tell which parameters were sent to the server? Now try and change the zoom level to 13 and see what happens.

Can you now request a map image of your own city? (Tips: you can get the coordinates of a point on Earth by clicking anywhere on the Google maps, see here.

Now change the `maptype` parameter to `roadmap` and then to `satellite`. Observe the results and explain them briefly.

Tasks:
Read the introduction to Google Static Maps: https://developers.google.com/maps/documentation/static-maps/intro. Try the static map URL examples below and point out the meaning of the parameters in your report.

- `https://maps.googleapis.com/maps/api/staticmap?center=KTH+Biblioteket,Osquars+backe, Stockholm&zoom=15&size=800x800&markers=color:blue|label:G|Drottning+Kristinas+vag+30, SE-10044,Stockholm|&markers=icon:http://maps.google.com/mapfiles/ms/icons/info.png |59.346933,18.072188`
- `https://maps.googleapis.com/maps/api/staticmap?center=KTH+Biblioteket,Osquars+backe, Stockholm&zoom=15&size=800x800&path=color:0x0000ff|weight:5|Drottning+Kristinas+vag+30, SE-10044,Stockholm|KTH+Biblioteket,Osquars+backe,Stockholm|59.34651,18.07183`
- `https://maps.googleapis.com/maps/api/staticmap?center=KTH+Biblioteket,Osquars+backe, Stockholm&zoom=15&size=800x800&path=color:0x000000AA|weight:5|fillcolor:0xFFFF00AA |59.34651,18.07183|Drottning+Kristinas+vag+30,SE-10044,Stockholm|KTH+Biblioteket, Osquars+backe,Stockholm|59.34651,18.07183`

Now create your own static maps using the following features:

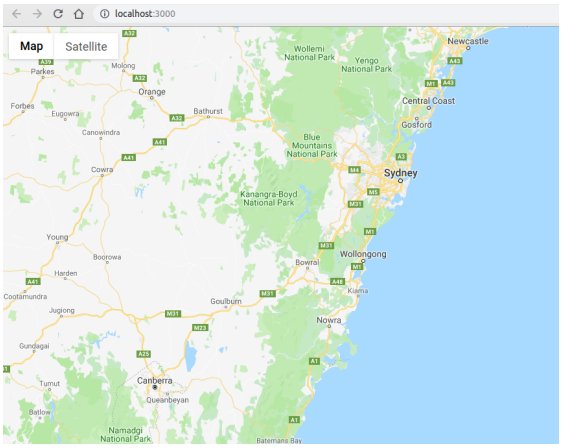

Figure 4.1: Screenshot of Google Map example on the node server

- **Markers**: Add a few markers in the map.
- **Path**: Add a path in the map.

4 Google Maps JavaScript API

The Google Maps JavaScript API allows you to make your web maps more dynamic and enables a wide rage of web GIS functionality. Exploring the complete list of these functionalities is beyond the scope of this lab exercise or the short course, but you are encouraged to practice all the available features based on the Google Maps JavaScript API Guide.

Tasks:

- **Hello Stockholm**: Create a Google map around Stockholm area and set the zoom level to 12 according to the tutorial here.

  You should be able to see a web map as shown in Figure 4.1.

  Hint: you can always copy the content of an HTML file provided by the tutorial into an HTML file in your project to run the example on your node server.
- **Google map events**: First read about Google map events in the tutorial here. Place a marker at KTH Library and add an event listener so that when it is clicked, the map is centered at KTH Library with certain zoom level 18.
- **Google data layer**: First, read about the Google map's data layer and finish the tutorial here. Second, a GeoJSON file representing simplified Metro lines in Stockholm is available in the lab folder - data.geojson. Load the GeoJSON file as a data layer into your map and style it according to the color attribute of each line. Finally, implement which event listener(s) that you might find interesting (e.g., highlight a feature when mouse hovering over it, display a line number while clicking on a feature).

  References:
  - Declarative style rules documentation
  - Change appearance dynamically example
  - Info windows documentation
  - Info windows example

5 Deliverables

Submit a written report in PDF format on Canvas for the following deliverables. Write the name of your group members on the report.

1. For the Imagemap part, show your interactive web map to the teaching assistant and put it in the report.
2. For the Google Static Maps part, write a short report in which you include your four URLs for your home city, briefly explain the parts of your URLs, and include the static map images that your HTTP requests result in. Show your report to the teaching assistant.
3. For the Google Maps JavaScript API, include some typical screenshots of your map and explain the relevant parts (code segments) that make up your map.