[redacted]

Coursework 3: HTML, XML and JavaScript

The objective of this coursework is to improve your learning about HTML, JSON, JavaScript and AJAX. This coursework will be marked and it counts 15% of the final mark for this module.

1) General Description

In this coursework you are asked to create a webpage which will be showing traffic incidents on a map.

Your webpage should show a greeting message to the users and briefly let them know how they can use the page. The user should be able to select a city from a list and click a button to see the traffic incidents in the selected city.

The webpage should get the traffic data from an online repository, load a map of the city and show the traffic incidents both spatially on the map but also textually in a table.

The flow of the website's functionality should be the following[1]:

1. The user is presented with a greeting message and a dropdown list to select the city of which (s)he wants to see traffic incidents for. The user selects the city from a drop-down list and clicks a button to see the results.
2. The button-click triggers a JavaScript function which reads the user's selection and extracts the selected city's Longitude and Latitude[2].
3. The next step is to query the online repository for the traffic incidents on this city.
    a. The code constructs a URL[3] which holds the details of your query.
    b. The query (constructed URL) is then sent to the online repository (*mapquest)*[3]*,* through an asynchronous AJAX call, and the response is handled by another function.
4. The code then displays the traffic incidents in an HTML table. Each row of the table shows the number of the incident[4] and its short description.
5. A map is then loaded, using the *mapquest*[3] library, centred around the selected city.
6. The incidents are plotted in the map as dots[3] using JavaScript.

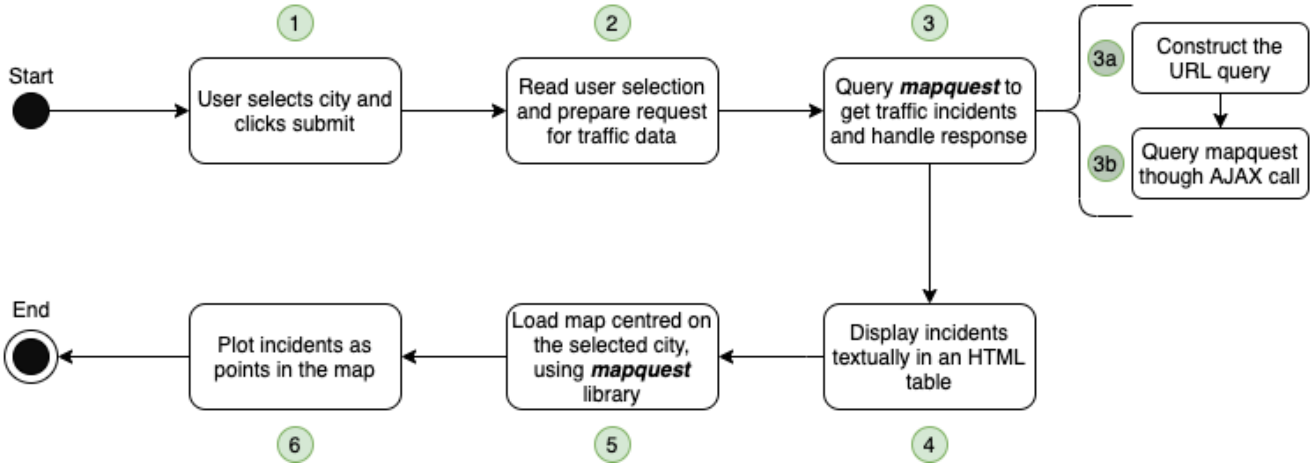
[1] Slight variations are acceptable as long as the final outcome is the same. Note that some of the steps are not necessarily district steps that the user can observe as the user sees only the final result (annotated map and table).

[2] See Section 5.4 for the cities you must include and their coordinates

[3] See Section 5 – Technical Details for details.

[4] A simple incremental number from 1 to the total number of events.

[redacted]


Start
1
User selects city and clicks submit
2
Read user selection and prepare request for traffic data
3
Query mapquest to get traffic incidents and handle response
3a
Construct the URL query
3b
Query mapquest though AJAX call
4
Display incidents textually in an HTML table
5
Load map centred on the selected city, using mapquest library
6
Plot incidents as points in the map
End

See section 6 for an example of how your webpage should look like – after the user has made a choice and the data have been loaded and showed.

2) Deliverables

Please submit it via KEATS as a SINGLE HTML file with any Javascript scripts in it (in the HTML). Please do not submit any instructions, the web page should be self-explanatory. Comment your code appropriately.

Your code will be checked against plagiarism and collusion, so make sure you only submit your original work (except where explicitly stated otherwise). Your webpage should be runnable from both Firefox and Chrome browsers.

3) Marking

We'll evaluate your code structure and comments, how clear and self-explanatory the user interface is and how the code runs. Markings will be as detailed below:

1. User interface: 10%
2. Code Structure : 5%
3. Comments: 5%
4. Read user selection: 10%
5. Construct query URL: 10%
6. Query mapquest and handle response: 20%
7. Show results in a table: 20%
8. Show map centred on selected city: 10%
9. Plot incidents on the map: 10%

For more on marking look at the Appendix section, to see the detailed marking guide (rubrics).

[redacted]

[redacted]

5) Technical Details

5.1) MapQuest API

MapQuest API is online webservice that provides information for map services. One of the resources that it provides is data on traffic incidents. MapQuest allow for 15000 free transactions per month (that means you can ask 15000 times per month for the traffic data – clearly more than what you'll need for the coursework). Do not share you key with your colleagues; ßeach one needs to have their own.

In order to get access to the API, you'll need to do the following:

1. Go to https://developer.mapquest.com/
2. Click "Get your free API key"
3. Fill in your data (obviously don't have to register for email updates)
4. Sign up and confirm email if needed.
5. After sign up, login and in your main profile page you will see a section called "my Keys". You will need the key listed there for querying the MapQuest server to get the data[5].
6. Go to https://developer.mapquest.com/documentation/traffic-api/incidents/get/ to find information for how to get the traffic data. Try to run the example request from this page, in your browser's address, replacing *KEY* with your unique key (step 5). This is an example of traffic data retrieved from the server for Denver US. To centre the results around another city you need to change the numbers after *"`boundingBox=`"* to *"`cityLat-0.22,cityLng-0.22,cityLat+0.22, cityLng+0.22`"*, where `cityLat` and `cityLng` are the city's latitude and longitude respectively. For example, a city that has *Latitude: 40.64* and *Longitude 22.94* the bounding box should be:
   "`boundingBox=40.42,22.72,40.86,23.16`"
   You will need to calculate these values and feed them on the appropriate place (BoundingBox) on the URL.
7. In order to query the server through JavaScript (AJAX call) with the URL you need to follow the example of week's 7 slides on AJAX. Both techniques on "Reading a JSON" slide or "jQuery AJAX" slide work the equally, chose whichever you prefer.
8. From these data (list of incidents) the columns that you'll need are *`lat`*, *`lng`*, and *`shortDescr`*. The first two are the coordinates and the later the short description of each specific incident.

[5] This key is unique to your profile and is there to ensure that the server identifies you when you ask for data; do not share you key with your colleagues, each one needs to have their own. We will not use the key in your code other than running your project for marking it. You can check the number of transactions you have done from your profile page.

[redacted]

[redacted]

5.2) Loading a map though JavaScript:

1. You need to add the **mapquest** library as script in the *head* of your HTML. This basically downloads the library and attaches it to the browser's tab, and it is ready to be used in the HTML you created. To do that you simply add the following at your HTML's Head tag, before other scripts:

```
<script src="https://api.mqcdn.com/sdk/mapquest-js/v1.3.2/mapquest.js"></script>
```

and

```
<link type="text/css" rel="stylesheet" href="https://api.mqcdn.com/sdk/mapquest-js/v1.3.2/mapquest.css" />
```

2. This example shows how you can load the map dynamically in your page. In order to achieve this you also need to have the following empty <div> in your code, in order for the JavaScript code to attach the map to:

```
<div id="map" style="width: 100%; height: 530px;"></div>
```

5.3) To add a marker (for an incident) on the map:

The following code allows you to place a circular marker on the map at the specific coordinates:

```
L.circle([latitude, longitude], { radius: 2 }).addTo(map)
```

`latitude` and `longitude` are the values of the incident's coordinates.
Bear in mind that `map` is the JavaScript variable that holds the map, created in the previous step.

5.4) Add rows to an HMTL table through JavaScript.

To make this work, you initially need to have an empty table to your HTML page. Then you need from JavaScript to get the table's body from its id and store it in a variable, then create the HTML code for the lines to be added as a JavaScript string (concatenation will help here) and then append this code to the table body's innerHTML. This example may help you.

5.5) Cities

The five cities[6] which the user can choose from are:

Name	Latitude	Longitude
New York	40.71	-74.00
Los Angeles	34.03	-118.15
Chicago	41.53	-87.38
Seattle	47.37	-122.20
Louisville	38.15	-85.46

- Do not show the Latitude and Longitude to the user, you will only need them to locate the map.
- You need to locate the map on the city when the user selects it.

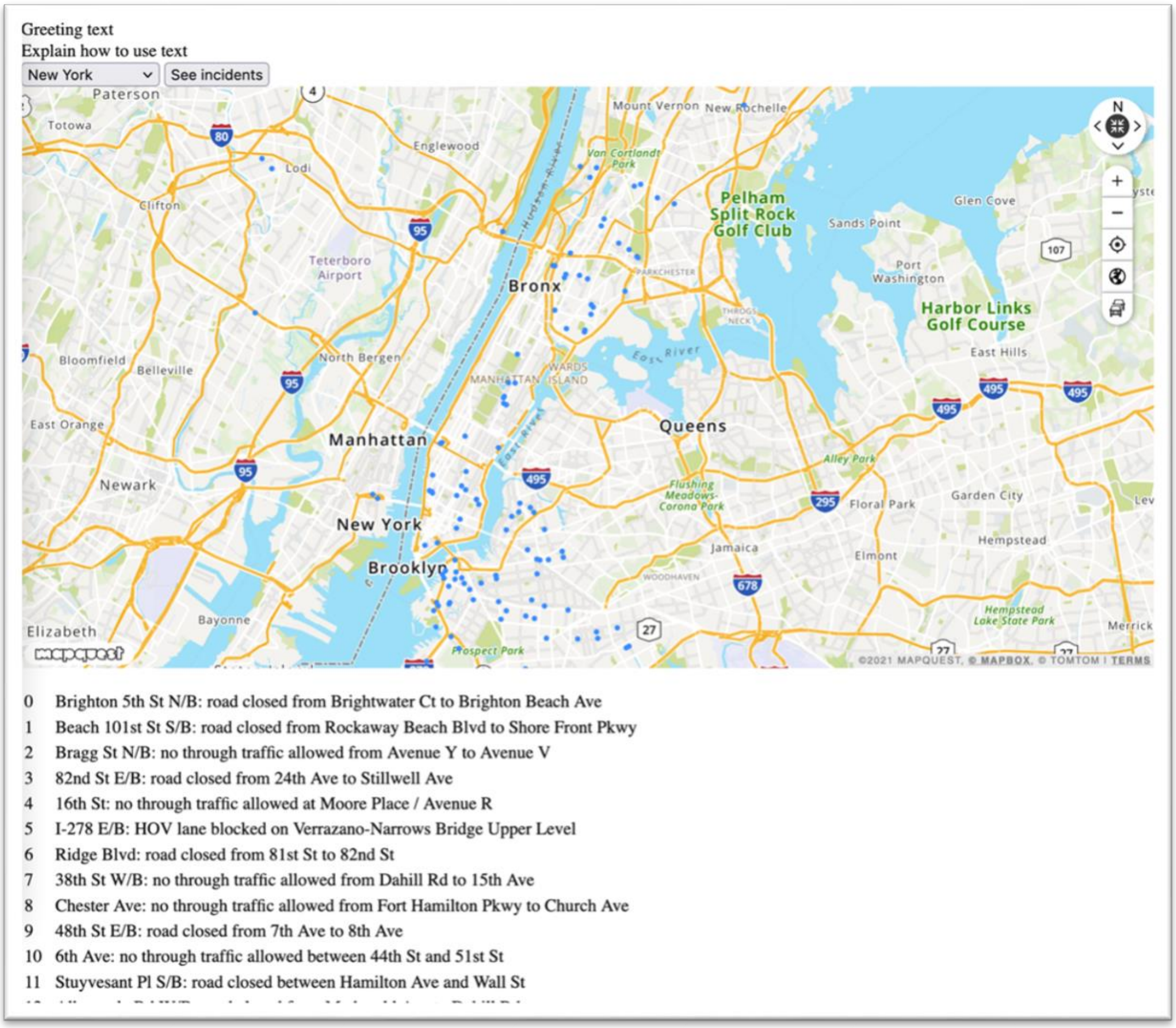
[6] Unfortunately, MapQuest provides data only for cities in the USA.

[redacted]

Note

For an extra challenge you can clear the map and the table whenever the user makes a new choice. This is not compulsory, nor will it give you more marks.

6) Example outcome



Greeting text
Explain how to use text

0 Brighton 5th St N/B: road closed from Brightwater Ct to Brighton Beach Ave
1 Beach 101st St S/B: road closed from Rockaway Beach Blvd to Shore Front Pkwy
2 Bragg St N/B: no through traffic allowed from Avenue Y to Avenue V
3 82nd St E/B: road closed from 24th Ave to Stillwell Ave
4 16th St: no through traffic allowed at Moore Place / Avenue R
5 I-278 E/B: HOV lane blocked on Verrazano-Narrows Bridge Upper Level
6 Ridge Blvd: road closed from 81st St to 82nd St
7 38th St W/B: no through traffic allowed from Dahill Rd to 15th Ave
8 Chester Ave: no through traffic allowed from Fort Hamilton Pkwy to Church Ave
9 48th St E/B: road closed from 7th Ave to 8th Ave
10 6th Ave: no through traffic allowed between 44th St and 51st St
11 Stuyvesant Pl S/B: road closed between Hamilton Ave and Wall St

[redacted]

Detailed marking guide (rubrics).

1) User Interface	The greeting text is compact and clear, and the city selection is done easily.	The greeting text is fairly easy to understand, not too extensive , **AND** the city selection is done fairly easily	The greeting text is fairly easy to understand, not too extensive , **OR** the city selection is done fairly easily	The greeting text is hard to understand, extensive , and the city selection is not easy.	There is no greeting text **AND** city selection is not easy
Marks	10	7-9	4-6	2-3	1
2) Code Structure	The code is exceptionally well organised, very easy to follow and with descriptive variable names	The code is fairly easy to read.	The code is readable only by someone who knows what it is supposed to be doing.	The code is poorly organized and very difficult to read.	
Marks	5	4	3	2	
3) Comments	The comments are well written and clearly explain what the code is accomplishing and how, especially in the complex parts	The comments are extensive but are not clear **OR** they do not explain clearly the complex parts	There comments cover a good part of the code, but are not clear neither do they explain well the complex parts.	The comments are not enough neither do the explain the complex parts clearly.	There are no comments.
Marks	5	4	3	1-2	0

If Longitude and latitude are shown to the user reduce level of achievement by 1. Same if not all cities are not present.

[redacted]

4) Read User Selection	Submit button triggers a method call and the proper latitude and longitude are used for the following program.	Submit button triggers a method call and wrong or static latitude and longitude are used for the following program.	Submit button does not trigger a method call and latitude or longitude are not calculated by the user's choice.		
Marks	10	5	0		
5) Construct query URL	URL consructed according to the pattern	URL constructed but without a valid key **OR** without the .22 calibration	URL constructed but without a valid key **AND** without the .22 calibration	URL not constructed	
Marks	10	8	5	0	
6) Query MapQuest and handle response	The query is send to server, the response is handled by a separate method and response is extracted and used.	The query is send to server, the response is handled by a seperate method but the response is not extracted and used.	The query is send to server, the response is **not** handled by a separate method and the response is **not** extracted and used.	The query is prepared but AJAX is not used to send it.	The query is not prepared and sent to the server
Marks	20	15	10	5	0
7) Show results in a table	All resutls are show in the table. Each line has a column with the incident number and the short description.	All resutls are show in the table. Each line has a column with only the incident 's short description	All resutls are show but not in a table (i.e. as plain text on the screen)	Resutls are not shown textually	
Marks	20	15	10	0	

[redacted]

[redacted]

8) Show map centred on the selected city	Map loads, it is centred around the selected city and the zoom covers most of the city	Map loads, it is centred around the selected city but the zoom does not cover most of the city	Map loads, but it is not centred around the selected city	Map does not load	
Marks	10	7-9	5	0	
9) Plot incidents on the map	**All** Incidents for the selected city are correctly plotted on the map	**Not all** Incidents for the selected city are correctly plotted on the map	Incidents **are not** plotted on the map		
Marks	10	5-9	0		
Total Marks	100				