5CCS2INS - Internet Systems: Coursework 2: HTML, XML, and JavaScript

Department of Informatics

2022-11-07

The objective of this coursework is to improve your learning about HTML, JSON, JavaScript, and AJAX. This coursework will be marked, and it counts 22% of the final mark for this module.

1 General Description

In this coursework, you are asked to create a webpage to view the crowding at London Underground ('tube') stations by weekday.

Your webpage should show a greeting level-1 header to the users. There should then be a drop-down choice to choose the tube station. There should then be a drop-down choice to choose the day of the week. Then, when the user clicks a button, they should be able to see a bar-chart of the crowding (as a percentage of total station occupancy) on the y-axis and time-bands (i.e., 15 minute intervals) on the x-axis. **Below** this the AM peak time frame, and PM peak time frame, should be noted. **Below** this there should be a table showing the crowding in 15-minute increments.

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

1. The user is presented with a greeting level-1 header (`<h1>`) and two drop-down lists: the first to select the tube station, the second to select the day of the week.
2. The button-click triggers a JavaScript function which reads the user's selection and extracts the selected tube station NaPTAN[2] ID.[3]

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

[2] NaPTAN is the national dataset for uniquely identifying all public transport access points in England, Scotland and Wales.

[3] See section sec. 5.6 for the tube stations you must include and their NaPTAN ID.

3. The next step is to query the Transport for London (TfL) API for the crowding information on the selected tube station on the selected day of the week.
    1. The code constructs a URL[4] which holds the details of your query.
    2. The query (constructed URL) is then sent to the TfL API,[5] through an asynchronous AJAX call, and the response is handled *by another function*.
4. A bar chart, using chart.js version 3, is created showing the capacity levels *as percentages* (i.e., multiply by 100) in 15-minute increments throughout the selected day of the week.
5. The code then displays the AM and PM peak time frame, as returned by the API.
6. The code then displays the capacity levels as percentages in 15-minute increments throughout the selected day of the week.

2 Deliverables

Please submit it via KEATS as a **single HTML** file with any JavaScript scripts in it (i.e., in `<script>` tags 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 able to be run from both Firefox and Chrome browsers. Before submitting, please make sure that you test your deliverable with both Firefox and Chrome.

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 are detailed below:

1. User Interface: 10%
2. Code structure: 5%
3. Comments: 5%
4. Read user selection: 10%
5. Construct query: 10%
6. Query TfL API and handle response: 20%
7. Show results in a table: 15%
8. Show AM/PM peak time: 5%
9. Show bar chart using chart.js: 20%

[4] See section sec. 5.1 for details.
[5] See section sec. 5.1 for details.

4 Deadline

The deadline is the 30th November 2022 16:00 GMT (UK time).

5 Technical details

5.1 TfL API

TfL provide an API useful for querying many things about their service. For this coursework, we will use the crowding API. TfL allow for 500 requests per minute, clearly far more than you'll need for the coursework. **Do not share your key with your colleagues**: each person needs their own.

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

1. Go to https://api-portal.tfl.gov.uk/signup and signup.
2. Verify your signup with your email.
3. Go to https://api-portal.tfl.gov.uk/signin and sign in.
4. Go to Products in the top bar (https://api-portal.tfl.gov.uk/products).
5. Click on "500 Requests per min".
6. Type "crowding" in "Your new product subscription name" and click "Subscribe".
7. This should open your profile (https://api-portal.tfl.gov.uk/profile).
8. Click show next to "Primary Key" – this is your API key.
9. Go here https://api-portal.tfl.gov.uk/api-details#api=crowding&operation=dayofweek to try out the API. The URL is in the format `https://api.tfl.gov.uk/crowding/{Naptan}/{DayOfWeek}`. Naptan is the ID given in sec. 5.6. DayOfWeek is the first 3 letters of the day of the week in lowercase (i.e., 'mon', 'tue', etc.). `app_key` must be supplied as a *header* – this is your API key.
10. In order to query the server through JavaScript (AJAX call) with the URL you need to follow the example of week 7's slides on AJAX. Both techniques on "Reading a JSON" slide or "jQuery AJAX" slide work equally – choose as you prefer.
11. You will get a response like so:

```
{
    "naptan": "940GZZLUPAC",
    "dayOfWeek": "MON",
    "amPeakTimeBand": "07:45-09:45",
    "pmPeakTimeBand": "17:00-19:00",
    "timeBands": [{
        "timeBand": "00:00-00:15",
        "percentageOfBaseLine": 0.027325633
    }, {
        "timeBand": "00:15-00:30",
        "percentageOfBaseLine": 0.020355553
    }, ...
    ]
}
```

12. You will need `amPeakTimeBand` and `pmPeakTimeBand` for the AM and PM peak time frame, respectively. You will need `timeBands` for the bar chart and table.
13. For the bar chart you will need two arrays: one of `timeBands.timeBand`, which **must** called `labels` and one of `timeBands.percentageOfBaseLine * 100`, which **must** be called `crowding`. To do this you may want to do look into `Array.prototype.map`, though other solutions are acceptable.
14. For the table, display each `timeBand` and `percentageOfBaseLine` (multiplied by 100) as a row.

5.2 Scripts

You must include the `chart.js` script, and you probably want to include `jQuery` also. You can do this by adding the following to your document:

```
<script
  src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/3.9.1/chart.min.js">
</script>
<script
  src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.6.1/jquery.min.js">
</script>
```

5.3 Displaying the AM and PM peak time frame

To do this, you initially need to have an empty `span` in your HTML page for AM and PM. Then you need, from JavaScript, to get the spans from their IDs and store them in variables. Then, you can update the span's text with the AM and PM period, as appropriate (from `amPeakTimeBand` and `pmPeakTimeBand` respectively).

Hint: In case the user has already run the site, you will want to overwrite any existing text in your span! (i.e., you don't want to show multiple periods).

5.4 Add rows to an HTML table through JavaScript.

To do this, you initially need to have an empty table in 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 (string interpolation / concatenation may help here), and then append this code to the table body's innerHTML. This example may be useful.

Hint: If the user has already run the site, you may need to clear the table first!

5.5 Displaying a bar chart of crowding levels

Firstly, you will need to have a `<canvas>` element with an `id` to store your chart.

For the bar chart you will need two arrays: one of `timeBands.timeBand` called `labels` and one of `timeBands.percentageOfBaseLine * 100` called `crowding`. To do this you may want to do look into `Array.prototype.map`, though other solutions are acceptable.

You will then need to create a `data` object, like so:

```
const data = {
    labels: labels,
    datasets: [{
        label: "Crowding",
        data: data,
    }]
};
```

You will then need to plug in this `data` object into the bar chart `config`. This documentation may come in handy to do this. You will then need to create a `Chart` object, supplying the `canvas` (*hint*: use `document.getElementById`) and the `config`. See this minimum example for how to create and display a chart.

The bar chart should start at 0 and end at 100 on the y-axis. This documentation will be useful for doing this. If you get stuck on this and consult other documentation, make sure that it is specific for chart.js version 3.

Hint: If the user has already run the site, you will need to `destroy` the chart first!

5.5.1 Relevant documentation

- Minimum working example of chart
- Bar chart
- Changing the range of axes

5.6 Tube stations

The six tube stations from which the user can choose from are:

Name	NaPTAN ID
Paddington	940GZZLUPAC
King's Cross St Pancras	940GZZLUKSX
Euston	940GZZLUEUS
Waterloo	940GZZLUWLO
London Bridge	940GZZLULNB
Liverpool Street	940GZZLULVT

Do not show the NaPTAN ID to the user, you will only need them for the API.

6 Example

6.1 Before being run

The follow page shows an example of what the page will look like before being run.

View crowding info at tube stations

Station: Paddington Day of week: Monday View crowding info!

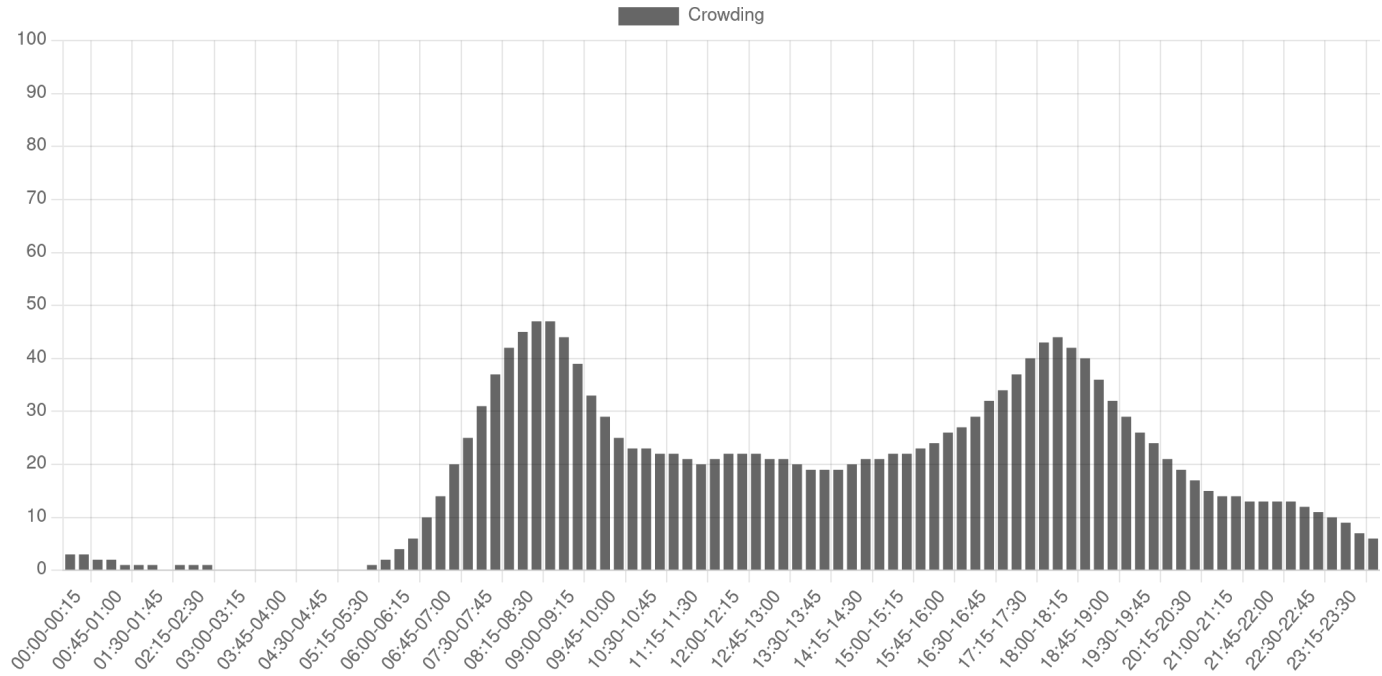
Table of crowding

6.2 After being run

The following pages show an example of what the page will look like after run.

View crowding info at tube stations

Station: Paddington Day of week: Monday View crowding info!



AM Peak: 07:45-09:45

PM Peak: 17:00-19:00

Table of crowding

Time Band	Crowding Percentage
00:00-00:15	3
00:15-00:30	3
00:30-00:45	2
00:45-01:00	2
01:00-01:15	1
01:15-01:30	1
01:30-01:45	1
01:45-02:00	0
02:00-02:15	1
02:15-02:30	1
02:30-02:45	1
02:45-03:00	0
03:00-03:15	0
03:15-03:30	0
03:30-03:45	0
03:45-04:00	0
04:00-04:15	0
04:15-04:30	0
04:30-04:45	0
04:45-05:00	0
05:00-05:15	0
05:15-05:30	0

05:30-05:45 1
05:45-06:00 2
06:00-06:15 4
06:15-06:30 6
06:30-06:45 10
06:45-07:00 14
07:00-07:15 20
07:15-07:30 25
07:30-07:45 31
07:45-08:00 37
08:00-08:15 42
08:15-08:30 45
08:30-08:45 47
08:45-09:00 47
09:00-09:15 44
09:15-09:30 39
09:30-09:45 33
09:45-10:00 29
10:00-10:15 25
10:15-10:30 23
10:30-10:45 23
10:45-11:00 22
11:00-11:15 22
11:15-11:30 21
11:30-11:45 20
11:45-12:00 21
12:00-12:15 22
12:15-12:30 22
12:30-12:45 22
12:45-13:00 21
13:00-13:15 21
13:15-13:30 20
13:30-13:45 19
13:45-14:00 19
14:00-14:15 19
14:15-14:30 20
14:30-14:45 21
14:45-15:00 21
15:00-15:15 22
15:15-15:30 22
15:30-15:45 23
15:45-16:00 24
16:00-16:15 26
16:15-16:30 27
16:30-16:45 29
16:45-17:00 32
17:00-17:15 34
17:15-17:30 37
17:30-17:45 40
17:45-18:00 43

18:00-18:15 44
18:15-18:30 42
18:30-18:45 40
18:45-19:00 36
19:00-19:15 32
19:15-19:30 29
19:30-19:45 26
19:45-20:00 24
20:00-20:15 21
20:15-20:30 19
20:30-20:45 17
20:45-21:00 15
21:00-21:15 14
21:15-21:30 14
21:30-21:45 13
21:45-22:00 13
22:00-22:15 13
22:15-22:30 13
22:30-22:45 12
22:45-23:00 11
23:00-23:15 10
23:15-23:30 9
23:30-23:45 7
23:45-00:00 6

Appendix

Detailed marking guide (rubrics)

Total marks: 100

User interface

- 10 marks – The greeting text is compact and clear, and the station and day of week selection is done easily.
- 7-9 marks – The greeting text is fairly easy to understand, not too extensive **and** the station and day of week selection is done fairly easily.
- 4-6 marks – The greeting text is fairly easy to understand, not too extensive, **or** the station and day of week selection is done fairly easily.
- 2-3 mark – (a) The greeting text is hard to understand, extensive, and the station and day of week selection is not easy **OR** (b) There is no greeting text, and the station and day of week selection is done fairly easily.
- 1 mark – There is no greeting text **and** station and day of week selection is not easy.
- If the NaPTAN ID is shown to the user reduce mark by 1.
- If not all stations or days of week shown reduce mark by 1.
- If the order of the page is not (from top to bottom): greeting, selection, chart, AM/PM peak time, table; then reduce mark by 1.

Code structure

- 5 marks – The code is exceptionally well organised, very easy to follow, and with descriptive variable names.
- 4 marks – The code is fairly easy to read.
- 3 marks – The code is readable only by someone who knows what it is supposed to be doing.
- 2 marks – The code is poorly organised and very difficult to read.

Comments

- 5 marks – The comments are well written and clearly explain what the code is accomplishing and how, especially in the complex parts.
- 4 marks – The comments are extensive but are not clear **or** they do not explain clearly the complex parts.
- 3 marks – The comments cover a good part of the code, but are not clear, neither do they explain well the complex parts.

- 1-2 marks – The comments are not enough, neither do they explain the complex parts clearly.
- 0 marks – There are no comments.

Read User Selection

- 10 marks – Submit button triggers a method call and the proper NaPTAN ID is used for the following program.
- 5 marks – Submit button triggers a method call and wrong or static NaPTAN ID are sued for the following program.
- 0 marks – Submit button does not trigger a method call and NaPTAN ID is not calculated by the user's choice.

Construct query

- 10 marks – URL constructed according to the pattern **and** the `app_key` header is set correctly.
- 7-8 marks – URL not constructed according to the pattern **or** the `app_key` header is not set correctly / set at all.
- 4-5 marks – URL not constructed according to the pattern **and** the `app_key` header is not set correctly / set at all.
- 0-1 – URL not constructed **and** the `app_key` header is not set correctly / set at all.

Query TfL API and handle response

- 20 marks – The query is sent to the server, the response is handled by a *separate* method, and the response is extracted and used.
- 15 marks – The query is sent to the server, the response is handled by a separate method, but the response is not extracted and used.
- 10 marks – The query is sent to the server, the response is **not** handled by a separate method and the response is **not** extracted and used.
- 5 marks – The query is prepared, but AJAX is not used to send it.
- 0 marks – The query is not prepared and sent to the server.

Show results in a table

- 15 marks – All results are shown in the table. Each line has the time band and crowding as a percentage.
- 7-10 marks – Only some results are shown in the table. Each line may show either the time band or crowding as a percentage.
- 5 – All results are shown but not as a table (i.e., as plain text).

- 2-3 – Only some results are shown but not as a table.
- 0 – Results are not shown textually.
- Subtract one mark if the crowding has not been multiplied by 100.
- Subtract three marks if the table is not cleared when the user re-runs the page (i.e., don't get multiple results showing at once).
- The total mark for this section cannot be lower than 0.

Show AM/PM peak time

- 5 marks – If both the AM & PM peak time is displayed correctly.
- 3 marks – If only one of the AM & PM peak time is displayed correctly.
- 0 marks – If neither the AM & PM peak time is displayed correctly.
- Subtract one mark if the AM & PM peak time is not cleared when the user re-runs the page (i.e., don't get multiple results showing at once).
- The total mark for this section cannot be lower than 0.

Show bar chart using chart.js

- 20 marks – The bar chart is displayed correctly.
- 17-19 marks – The bar chart is displayed correctly *except* that it doesn't start at 0, end at 100, or the crowding was not multiplied by 100 (i.e., not a percentage).
- 15 marks – The bar chart is not displayed correctly, but the `config` and `data` object was made correctly (excluding any `options` of the `config` object).
- 8-10 marks – The bar chart is not displayed correctly, and the `config` object was not made correctly (excluding any `options` of the config object) / at all, but the `data` object was created correctly.
- 5-6 marks – The bar chart is not displayed correctly, and the `config` object was not made correctly (excluding any `options` of the config object) / at all, and the `data` object was not created correctly / not at all, but the `crowding` and `labels` variables were created correctly.
- 1 mark – The bar chart is not displayed, the `config` and `data` objects were not made, but the `crowding` or `labels` variables were made.
- 0 marks – The bar chart is not displayed, the `config`, `data`, `crowding` and `labels` objects were not made,
- Subtract one mark each if the `crowding` and `labels` variables were not named correctly / were done in-line.
- Subtract three marks if the chart is not destroyed if the user has already ran the page.
- The total mark for this section cannot be lower than zero.