CO539 Assessment 2
Food Hygiene Ratings

Deadline: Tuesday 11th December 2018 (Moodle - 23:55)

Introduction

For this assessment, you are going to use jQuery and AJAX to make a dynamic web page that communicates with two server-side scripts that we have developed for you.

- The ***first script*** uses data from the Food Standards Agency. The Food Standards Agency makes the results of restaurants' (and other businesses serving food) **hygiene inspections** publicly available (see here: http://www.food.gov.uk), and we have developed a simple PHP script that will provide the results of the inspections for Medway in JSON format.
- The ***second script*** is uses data from Google Places. Google Places provides (among other data) **customer ratings** for restaurants and other businesses. We have developed a PHP script that will provide the customer ratings for some businesses in Medway in JSON format.

Requirements & constraints: You should contain all of your work on this assignment in a single HTML file. Use CSS embedded in the page's header for formatting, likewise for your JavaScript code. You must use the jQuery library (i.e., avoid plain JavaScript where jQuery can be used) and, for Part 3, the jQueryUI library (and its CSS if you wish). You cannot use any other jQuery plugins, external CSS, or any other JavaScript libraries. You should submit your single HTML file via Moodle before the deadline.

Part 1: Hygiene Ratings (50%)

Part 1 focuses on hygiene ratings. The main challenges are: setting up AJAX calls and pagination.

The Server-Side Script

The server-side script is hosted at the following URL:

`https://www.cs.kent.ac.uk/people/staff/sm951/co539/assessment2/hygiene.php`

The script can be manipulated using several GET parameters. Accessing the script without any parameters returns the first 10 businesses' ratings in JSON format for testing. The parameters accepted by the script are documented in the table below.

Parameter Name	Description
operation	This is the most important parameter, controlling what type of information the script is to return. The operation parameter can be set to one of the following values: *demo* — Returns the first 10 businesses' ratings (default operation) *get* — Returns a "page" of (20) results. Expects the **page** parameter *pages* — Returns how many pages of results are available *search* — Returns the first 20 records for businesses whose name contain the specified search term. Expects the **name** parameter.
page	Used only by the `get` operation. Expects an integer specifying which page of results to retrieve. For example, page=2 would return the 2nd page of results.
name	Used only by the `search` operation. Expects a string specifying the search term business

	names are to be matched against.
html	Shows the full data set as HTML, example: https://www.cs.kent.ac.uk/people/staff/sm951/co539/assessment2/hygiene.php?html

Examples:

https://www.cs.kent.ac.uk/people/staff/sm951/co539/assessment2/hygiene.php?operation=get&page=7

These parameters make the script retrieve the 7th page of results.

https://www.cs.kent.ac.uk/people/staff/sm951/co539/assessment2/hygiene.php?operation=search&name=Eat

These parameters make the script find businesses whose names contain the string "Eat".

JSON Format of Returned Data

The *demo*, `get`, and `search` operations all return data in the same JSON format. The format is an array of individual JavaScript objects, with each object having the following properties:

`fhrsId`	The ID of the inspection record.
`name`	The name of the business inspected.
`addressLine`	The address, including post code, of the business inspected.
`hygieneRating`	The rating (out of 5) awarded to the business.
`ratingDate`	The date on which the inspection was carried out.
`postcode`	The postcode of the business

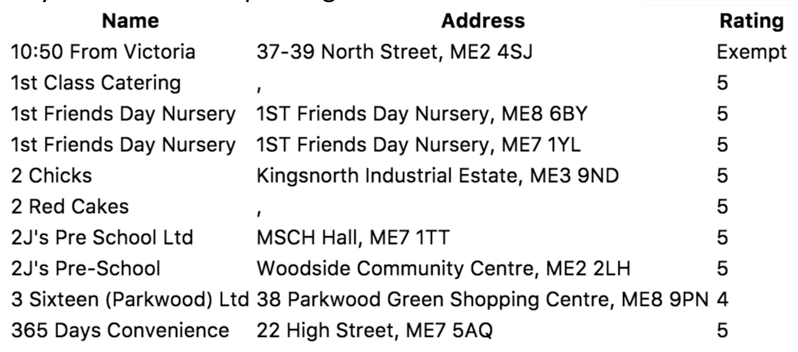
The *pages* operation returns a single JSON object, with a single "pages" property. This property contains the number of pages of results that are available.

Accessing the Script

The script outputs special HTTP headers, called Cross-Origin Resource Sharing (CORS, https://enable-cors.org/), which will allow your browser to access the script using AJAX requests regardless of where your HTML file is, in spite of the same origin policy. This means that you can complete this work from home without needing a webserver or database. We have hosted the script on a publicly-accessible webserver, so you don't need to use the VPN either. Because of the way these headers are interpreted by your browser, you **must** use the form of URL given above. Leaving out the *`www.`* will cause the CORS to fail.

Task 1.1 – Retrieving the First Page of Results When the Page Loads (25%)

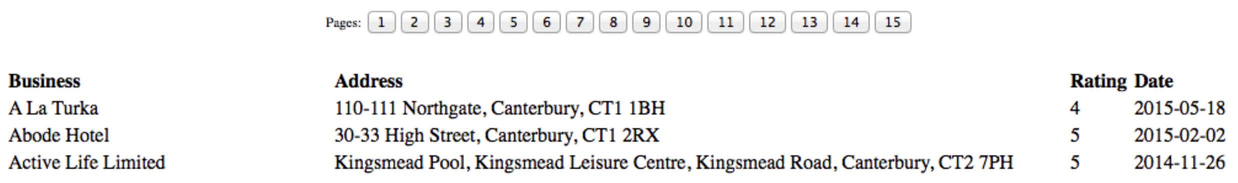
Write an HTML page containing a title, some text explaining what the page does, and an empty table. Use jQuery and an AJAX request (you *must* use the jQuery function `$.get` to do this) to the server-side script to populate the table with the *first page* (page 1) of results when the page loads. You may omit the id of the inspection record in your table. Furthermore, you will want to avoid visualising the postcode twice. The screenshot below shows you what the output might look like.



Name	Address	Rating
10:50 From Victoria	37-39 North Street, ME2 4SJ	Exempt
1st Class Catering	,	5
1st Friends Day Nursery	1ST Friends Day Nursery, ME8 6BY	5
1st Friends Day Nursery	1ST Friends Day Nursery, ME7 1YL	5
2 Chicks	Kingsnorth Industrial Estate, ME3 9ND	5
2 Red Cakes	,	5
2J's Pre School Ltd	MSCH Hall, ME7 1TT	5
2J's Pre-School	Woodside Community Centre, ME2 2LH	5
3 Sixteen (Parkwood) Ltd	38 Parkwood Green Shopping Centre, ME8 9PN	4
365 Days Convenience	22 High Street, ME7 5AQ	5

Task 1.2 – A Basic Paginator (25%)

Your page should now perform an additional AJAX request when the page loads (you *must* use the jQuery function `$.get` to do this). This request should find out the number of pages of inspection results that are available from the server-side script. Once you have found this out, create a row of buttons, one for each page. Each button should be labelled with a page number. The screenshot below shows you what your row of buttons might look like.



Pages: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

Business	Address	Rating	Date
A La Turka	110-111 Northgate, Canterbury, CT1 1BH	4	2015-05-18
Abode Hotel	30-33 High Street, Canterbury, CT1 2RX	5	2015-02-02
Active Life Limited	Kingsmead Pool, Kingsmead Leisure Centre, Kingsmead Road, Canterbury, CT2 7PH	5	2014-11-26

Clicking one of the paginator buttons should empty the results table, perform an AJAX request to the server-side script to fetch the requested page of results, and display them in the table.

Part 2: Customer ratings – mashup (25%)

Part 2 focuses on merging information from more than one source: the script you used in Part 1 and another script we provide, with information on customer ratings. As before, the script outputs CORS headers, so you can develop your solution from home.

The Server-Side Script

The server-side script is hosted at the following URL:
`https://www.cs.kent.ac.uk/people/staff/sm951/co539/assessment2/rating.php`
The script can be manipulated using several GET parameters, appended on to the end of the URL. The parameters accepted by the script are documented in the table below.

Parameter Name	Description
name	Expects a string specifying the search term business names are to be matched against. Returns all matching records for businesses whose name contains the specified search term. Each record will include the following information: • `'name'`: the full name of the matching item, • `'locationId'`: an identifier for the location, including the town name, • `'rating'`: the current average customer rating of the matching item.

Example: `https://www.cs.kent.ac.uk/people/staff/sm951/co539/assessment2/rating.php?name=kitchen`

This parameter make the script find businesses whose names contain the string “kitchen”.

JSON Format of Returned Data

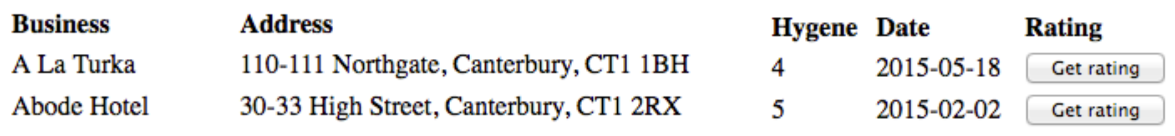
The data is returned in JSON format. The format is an array of individual JavaScript objects, with each object having the following properties:

`name`	The name of the business matched.
`locationId`	The town name, including country and some location id, of the business matched.
`rating`	A customer rating (out of 5).
`vicinity`	An approximate address.

If no business matched the given name the script returns an empty array. Note that for some businesses there may not be a location id or rating available (in which case key is associated to `null`).

Task 2.1 – Retrieving the Customer Rating when Requested by the User (25%)

Extend each row of the table with one button that allows the user to retrieve the customers’ ratings for the business in that row. The screenshot below shows you what the extended table might look like.



Business	Address	Hygene	Date	Rating
A La Turka	110-111 Northgate, Canterbury, CT1 1BH	4	2015-05-18	Get rating
Abode Hotel	30-33 High Street, Canterbury, CT1 2RX	5	2015-02-02	Get rating

Each button must trigger an AJAX request (you *must* use the jQuery function `$.get` to do this) directed at the server-side script `rating.php`. The customer rating for the (one) selected business is to be displayed in a pop-up window.

Notes

- buttons must be added to any table displayed, no matter if populated using the demo, page search, or name search functionality from Phase 1.
- The script `rating.php` may return more than one matching business. You need to find a way to match the business you are looking for with one (the right one!) of those returned by `rating.php`. You can, for example, try to match the town name.
- If the customer rating is not available (e.g., `rating.php` returns the empty array, none of the records returned matches the business you are looking for, or the matching business has empty rating) then you must notify the user, also using a pop-up window.
  Please note that the Google Place data set does not perfectly match the one of the Food Standard Agency. This makes it impossible to get 100% correct results. Thus, extra care is needed to avoid displaying wrongly attributed data.
- To help you testing your web application, you can view the complete data available on `rating.php`: https://www.cs.kent.ac.uk/people/staff/sm951/co539/assessment2/rating.php?html

Part 3: Search Functionality with Autocomplete (20%)

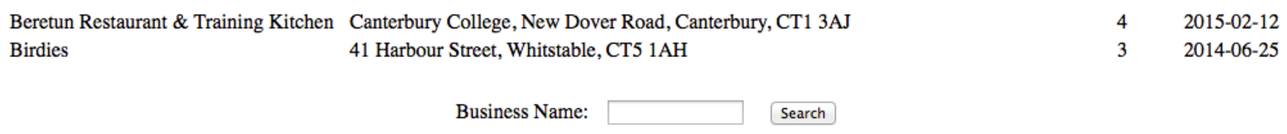
In this part you will use your knowledge of jQuery and AJAX, together with the available documentation of the jQueryUI API (https://jqueryui.com/autocomplete/), to learn to use a function that you may have not seen before: the autocomplete function of jQueryUI.

Task 3.1 – Add a search input with (20%)

You need to add a form to the bottom of your search page containing a text input and a search button. When the user enters text in the text input and clicks the button, your page should clear the results table, perform an AJAX request (you *must* use the jQuery function `$.get` to do this) to the server-side script form Part 1:

https://www.cs.kent.ac.uk/people/staff/sm951/co539/assessment2/hygiene.php

You need to retrieve any relevant results, and display them. The screenshot below shows you what your form might look like.



Beretun Restaurant & Training Kitchen	Canterbury College, New Dover Road, Canterbury, CT1 3AJ	4	2015-02-12
Birdies	41 Harbour Street, Whitstable, CT5 1AH	3	2014-06-25

After doing this, you need to apply the autocomplete function provided by jQueryUI to the input field. The autocomplete function will initially have a pre-defined array of *autocomplete tags*, which you have to define (initially, create an array containing *at most four* names of businesses at your choice). The autocomplete function should now suggest the four default names when focusing on the input field.

Next, you need to let the array of autocomplete tags be extended with previous search information. Concretely, every time the user clicks the search button, the name of each business returned by the script `hygiene.php` should be added to the array of autocomplete tags, avoiding duplicates.

Recall that you need to import the jQueryUI libraries before using them (see URL below)
https://ajax.googleapis.com/ajax/libs/jqueryui/1.12.1/jquery-ui.min.js

Also, you may want to use the jQueryUI styles (see URL below)
https://code.jquery.com/ui/1.12.1/themes/smoothness/jquery-ui.css

Design & Code Quality (5%)

There are some marks available for the visual design of the page, which you should achieve using CSS. Pay particular attention to the table in terms of size, spacing and margins to maximise usability. You should also make sure that the paginator buttons are a suitable size. You should also ensure that your code is properly commented, organised and laid out.

Late or Non-Submission of Coursework

The penalty for late or non-submission of coursework is that a mark of zero is awarded for the missing piece of work and the final mark for the module is calculated accordingly.

Plagiarism and Duplication of Material

Senate has agreed the following definition of plagiarism: "Plagiarism is the act of repeating the ideas or discoveries of another as one's own. To copy sentences, phrases or even striking expressions without acknowledgement in a manner that may deceive the reader as to the source is plagiarism; to paraphrase in a manner that may deceive the reader is likewise plagiarism. Where such copying or close paraphrase has occurred the mere mention of the source in a bibliography will not be deemed sufficient acknowledgement; in each such instance it must be referred specifically to its source. Verbatim quotations must be directly acknowledged either in inverted commas or by indenting."
The work you submit must be your own, except where its original author is clearly referenced. We reserve the right to run checks on all submitted work in an effort to identify possible plagiarism, and take disciplinary action against anyone found to have committed plagiarism.
When you use other peoples' material, you must clearly indicate the source of the material using the Harvard style (see http://www.kent.ac.uk/uelt/ai/styleguides.html).
In addition, substantial amounts of verbatim or near verbatim cut-and-paste from web-based sources, course material and other resources will not be considered as evidence of your own understanding of the topics being examined.
The School publishes an on-line Plagiarism and Collaboration Frequently Asked Questions (FAQ) which is available at: http://www.cs.ukc.ac.uk/teaching/student/assessment/plagiarism.local
Work may be submitted to Turnitin for the identification of possible plagiarism. You can find out more about Turnitin at the following page: https://www.kent.ac.uk/elearning/turnitin/?tab=information-for-students

Version 6.0.1

Stefan Marr (originally by Kris Welsh & Ian Utting, Laura Bocchi)