

Course:	INFO3114
Professor:	Oscar Lara
Project:	Project #1 – Major League Baseball
Due Date:	Thursday, June 06, 2019 11:30 pm
Submitting:	Please see the last page for instructions

How will my lab be marked?

- This project counts for 15% of your final grade and will be evaluated using the following grid:

Marks Available	What are the Marks Awarded For?	Mark Assigned
2	Good coding style including proper indentation and use of variable and object naming conventions and suitable comments	
3	Display page correctly and with good styling	
3	Filter text characters correctly for all the text boxes	
3	Request and retrieve a JSON string from the MLB site using selected values from the 3 pulldowns	
3	Enable the user to select the properties for the “next” and “previous” games	
1	Proper zipped submission	
15	Total	

Lab Description

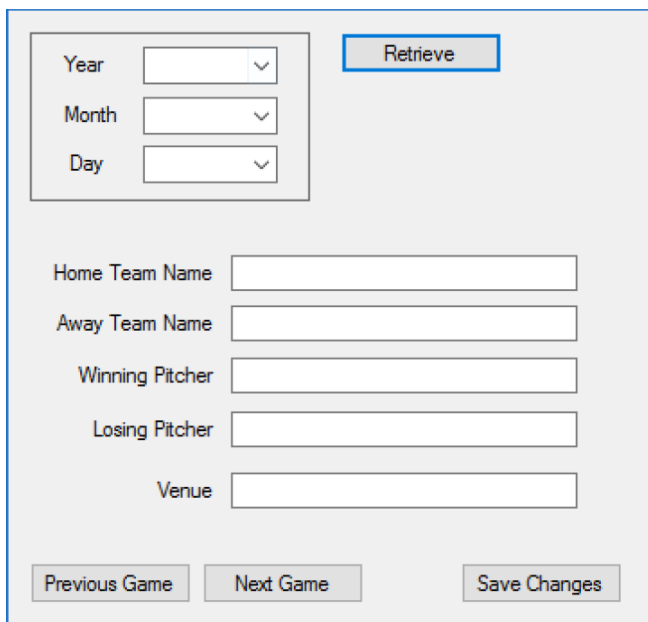
This project is based on the MLB Demo example that’s been provided to you recently. The demo page contains the JavaScript code required to retrieve data (JSON format) from the Major League Baseball site on the world wide web. Each retrieval is based on a specific day and requires a year, month and day to be specified. In addition, the demo example shows both synchronous and asynchronous strategies for retrieving the JSON data.

The objective of this project is to:

1. Retrieve a set of baseball data for a given date
2. Provide the user with the opportunity to edit a small subset of the data for each of the games
3. Save the changes to a local database (we won't actually save any changes with this project although you should have a fake button as part of the user interface)

Specific Requirements:

Create a basic user interface that looks somewhat like the following. You should use your own colours and fonts to create something that's professionally attractive.



The form is enclosed in a light gray border. At the top left, there are three stacked dropdown menus labeled 'Year', 'Month', and 'Day'. To their right is a blue 'Retrieve' button. Below these are five text input fields, each preceded by a label: 'Home Team Name', 'Away Team Name', 'Winning Pitcher', 'Losing Pitcher', and 'Venue'. At the bottom of the form are three buttons: 'Previous Game', 'Next Game', and 'Save Changes'.

When the page loads, the user will need to select a year, month and day from 3 <select> pulldown lists:

The Year pulldown should have 3 options: 2015, 2016 and 2017

The Month pulldown should have 12 options: 01, 02, 03 ... 12

The Day pulldown should have 31 options: 01, 02, 03 ... 31

We won't worry about validating any of the selected dates.

When the user clicks the "Retrieve" button, use an AJAX call to the MLB online site to retrieve the JSON for the specified day. Use a synchronous request to retrieve the data.

Your URL for issuing a request should look *something like* the following:

http://gd2.mlb.com/components/game/mlb/year_2015/month_07/day_12

When you receive the JSON data back from the site, you'll want to "parse" it into a JavaScript object in order to get access to the individual properties that you'll need to populate the form.

Something like:

```
var jsObject = JSON.parse(baseballJson);
```

In addition, you'll notice that the JSON string contains an array of objects where each element stores the set of properties for each of the games played on the specified day. This is called the "game" array which is part of the "games" object which is in turn part of the "data" object. See the "Notes" below for a detailed description of how to thoroughly analyze an unformatted JSON string.

As an example, given the declaration of the "jsObject" variable above:

```
jsObject.data.games.game[0]
```

gives us access to the "game[0]" object.

From here we want to allow the user to display and edit only 7 properties from the property data for each game:

home_team_name

away_team_name

winning_pitcher.first

winning_pitcher.last

losing_pitcher.first

losing_pitcher.last

venue

When you display the pitcher names, you will need to concatenate the first and last names for each pitcher.

Once you have the JSON data returned and converted to a JavaScript object, display the information for the first game of the day. The user may then load the set of properties for the next game in the game array by clicking the "Next Game" button. Similarly, if the user is not currently positioned at the first game, they can load the properties for the previous game into the game array by clicking the "Previous Game" button.

Once the properties are visible in the text boxes, you'll need to enforce text box level filtering to ensure that only appropriate characters are allowed for each text box. When you test your app, you will find that the "&" character is also required to allow editing of venue characters.

Remember as well, that the "Save Changes" button won't be operational for this project.

It is not a requirement that you create your JavaScript code and CSS code in separate files although you may choose to do so.

***** End of Requirements *****

Notes:

There are several strategies that can be employed to try and understand the contents of a complex JSON string:

1. Format a sample of the string using JSON Lint as described in the AJAX slide set.

The screenshot shows the JSONLint website interface. At the top, there's a header with the JSONLint logo and navigation links. The main area contains a text editor with a JSON object. Below the editor are buttons for 'Validate JSON', 'Clear', and a 'Support JSONLint for \$2/Month' button. The 'Results' section shows a green bar indicating 'Valid JSON'. Below this, there's an 'About JSONLint?' section and a 'Tips & Tricks' section with two bullet points.

```
{
  "subject": "master_scoreboard_mlb_2015_07_12",
  "copyright": "Copyright 2016 MLB Advanced Media, L.P. Use of any content on this page acknowledges agreement to the terms posted here http://gdx.mlb.com/components/copyright.txt",
  "data": {
    "games": {
      "next_day_date": "2015-07-13",
      "modified_date": "2016-03-31T15:34:22Z",
      "month": "07",
      "year": "2015",
      "game": [{
        "game_type": "R",
        "double_header_sw": "N",
        "location": "Flushing, NY",
        "away_time": "10:10",
        "broadcast": {
          "away": {
            "tv": "FSWI, MLBN (out-of-market only)",
            "radio": "WTMJ 620, Brewers Radio Network"
          },
          "home": {
            "tv": "YES, MLBN (out-of-market only)",
            "radio": "WFAN 660/101.9 FM, WADO 1280"
          }
        },
        "time": "1:05",
        "home_time": "1:05",
        "home_team_name": "Yankees",
        "description": "",
        "original_date": "2017/07/08",
        "home_team_city": "NY Yankees",
      ]
    }
  }
}
```

Results

Valid JSON

About JSONLint?

JSONLint is a validator and reformatter for JSON, a lightweight data-interchange format. Copy and paste, directly type, or input a URL in the editor above and let JSONLint tidy and validate your messy JSON code.

Tips & Tricks

- You can directly input a URL into the editor and JSONLint will scrape it for JSON and parse it.
- You can provide JSON to lint in the URL if you link to JSONLint with the "json" parameter. Here's an [example URL to test](#).
- JSONLint can also be used as a JSON compressor if you add ?reformat=compress to the URL.

2. Copy the formatted JSON into a text editor and save the new JSON file as <filename>.json. This step allows you to search the formatted JSON looking for specific properties.

The screenshot shows a text editor with the formatted JSON code. The code is well-indented and easy to read. A green oval highlights the 'home_team_name' property, which is 'Yankees'.

```
1 {
2   "subject": "master_scoreboard_mlb_2017_07_08",
3   "copyright": "Copyright 2017 MLB Advanced Media, L.P. Use of any content on this
4   page acknowledges agreement to the terms posted here
5   http://gdx.mlb.com/components/copyright.txt",
6   "data": {
7     "games": {
8       "next_day_date": "2017-07-09",
9       "modified_date": "2017-07-13T14:32:30Z",
10      "month": "07",
11      "year": "2017",
12      "game": [{
13        "game_type": "R",
14        "double_header_sw": "N",
15        "location": "Bronx, NY",
16        "away_time": "12:05",
17        "broadcast": {
18          "away": {
19            "tv": "FSWI, MLBN (out-of-market only)",
20            "radio": "WTMJ 620, Brewers Radio Network"
21          },
22          "home": {
23            "tv": "YES, MLBN (out-of-market only)",
24            "radio": "WFAN 660/101.9 FM, WADO 1280"
25          }
26        },
27        "time": "1:05",
28        "home_time": "1:05",
29        "home_team_name": "Yankees",
30        "description": "",
31        "original_date": "2017/07/08",
32        "home_team_city": "NY Yankees",
33      ]
34    }
35  }
36}
```

```

159         "away": "0"
160     }, {
161         "home": "0",
162         "away": "0"
163     }, {
164         "home": "3",
165         "away": "0"
166     }],
167     "h": {
168         "home": "6",
169         "away": "6"
170     }
171 },
172 "venue_w_chan_loc": "USNY0172",
173 "first_pitch_at": "",
174 "away_team_name": "Brewers",
175 "home_runs": {
176     "player": [{
177         "std_hr": "2",
178         "hr": "1",
179         "id": "640449",
180         "last": "Frazier",
181         "team_code": "nya",
182         "inning": "9",
183         "runners": "2",
184         "number": "30",
185         "name_display_roster": "Frazier",
186         "first": "Clint"
187     }, {
188         "std_hr": "15",
189         "hr": "1",
190         "id": "570267",
191
216     "away_preview":
217     "/mlb/gameday/index.jsp?gid=2017_07_08_milmlb_nyamlb_1&mode=preview
218     &c_id=mlb",
219     "tv_station": "YES, MLBN (out-of-market only)",
220     "home_audio":
221     "bam.media.launchPlayer({calendar_event_id:'14-491414-2017-07-08',m
222     edia_type:'audio'})",
223     "mlbtv":
224     "bam.media.launchPlayer({calendar_event_id:'14-491414-2017-07-08',m
225     edia_type:'video'})"
226 },
227 "home_ampm": "PM",
228 "game_pk": "491414",
229 "venue": "Yankee Stadium",
230 "home_league_id": "103",
231 "video_thumbnail":
232 "http://mediadownloads.mlb.com/mlbam/preview/milnya_491414_th_7_preview
233 .jpg",
234 "away_loss": "41",
235 "resume_date": "",
236 "away_file_code": "mil",
237 "losing_pitcher": {
238     "id": "608349",
239     "last": "Knebel",
240     "losses": "2",
241     "era": "1.76",
242     "number": "46",
243     "name_display_roster": "Knebel",
244     "first": "Corey",
245     "wins": "0"
246 },
247 "aw lg ampm": "PM",

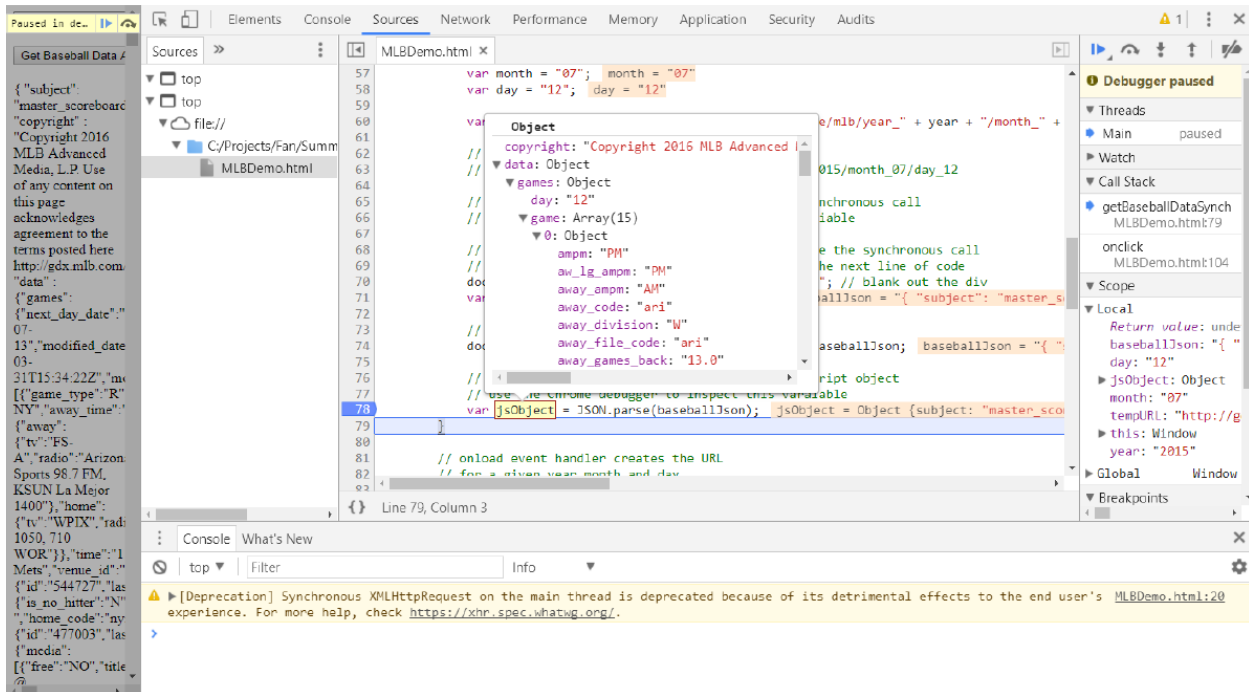
```

```

315     "time_hm_lg": "1:07",
316     "away_name_abbrev": "HOU",
317     "league": "AA",
318     "time_zone_aw_lg": "-4",
319     "away_games_back": "-",
320     "home_file_code": "tor",
321     "game_data_directory":
"/components/game/mlb/year_2017/month_07/day_08/gid_2017_07_08_houmlb_t
ormlb_1",
322     "time_zone": "ET",
323     "away_league_id": "103",
324     "home_team_id": "141",
325     "day": "SAT",
326     "time_aw_lg": "1:07",
327     "away_team_city": "Houston",
328     "tbd_flag": "N",
329     "tz_aw_lg_gen": "ET",
330     "away_code": "hou",
331     "winning_pitcher": {
332       "id": "573186",
333       "last": "Stroman",
334       "losses": "5",
335       "era": "3.28",
336       "number": "6",
337       "name_display_roster": "Stroman",
338       "first": "Marcus",
339       "wins": "9"
340     },
341     "game_media": {
342       "media": [
343         {
344           "free": "NO",
345           "title": "HOU @ TOR",

```

- Use the Chrome debugger to display the properties of the JavaScript object that is created when the JSON is “parsed.” This will allow you to drill-down and inspect the object in great detail. This is done by setting a break-point near or after you’ve “parse” the JSON string into a JavaScript object. You can then hover the mouse pointer over the object variable and Chrome will display the pop-up showing the detailed drill-down.



How should I submit my project?

Electronic Submission:

Submit your program files to the Info3114 "Project 1" electronic submission folder in FanshaweOnline. These files should be submitted as a single "zip" file containing your web application's complete website.

I strongly recommend that you test your own submission to ensure that nothing has been missed.

Submit your project on time!

Project submissions must be made on time! Late projects will be subject to divisional policy on missed test and late projects. In accordance with this policy, no late projects will be accepted without prior notification being received by the instructor from the student.

Submit your own work!

It is considered cheating to submit work done by another student or from another source. Helping another student cheat by sharing your work with them is also not tolerated. Students are encouraged to share ideas and to work together on practice exercises, but any code or documentation prepared for a project must be done by the individual student. Penalties for cheating or helping another student cheat may include being assigned zero on the project with even more severe penalties if you are caught cheating more than once. Just submit your own work and benefit from having made the effort on your own.