# Assignment: ToDo list

We have been learning about using GET and POST communication to interact with API endpoints. In this assignment we will use the $.getJSON and $.post() to implement a simple “to do” list application.

Make note of the following URLs. This “todo” API is written in such a way that it will hold data for 24 hours. Data added to the “to do” list will automatically wash out of the server’s memory after one day. This is by design!

|  |  |
| --- | --- |
| **URL** | **Notes** |
| <https://misdemo.temple.edu/todo/>  | Describes the features.  |
| <https://misdemo.temple.edu/todo/addtask/>  | POST to this URL to add a task. You must pass “task” and “task\_owner” as URL encoded data. This API feature returns confirmation data in a JSON array. |
| <https://misdemo.temple.edu/todo/tasks/>  | GET from this URL to retrieve all tasks for a certain name. You must pass “task\_owner” as URL encoded data to get the tasks for a specific owner. The tasks are returned a JSON array. |

**IMPORTANT:** In this specific assignment is important that you put the "/" at the end of the URLs as shown above.

Recall the basic structure of the jQuery $.getJSON() method.

$.getJSON( *someurl* , *serialized\_data* , function(data){ } );

Recall the basic structure of the jQuery $.post() method.

$.post( *someurl* , *serialized\_data* , function(data){ } );

## Instructions

1. Starting with unit6\_07\_todo.zip, notice that there is a hidden field with an id of “task\_owner” and a value of “Marvin”.

Change “Marvin” to your TUID. The value you specify will be used to differentiate your “to do” list from everyone else’s.

1. Notice the “Add Task” button and the “Refresh Task List” button, already created.
2. Inspect the <script> tag found in index.html. Complete the supporting functions there.
	1. Write the code necessary to POST data to the addtask endpoint.
	2. When a task is added, show a confirmation message in the <div> tag provided. The message can be “task added OK”
	3. Write the code necessary to GET data from the tasks endpoint. Loop through the data returned by the API and append each task to the existing unordered list.
3. Upload your work to the class server when you are done. Test it!

**CONTINUED…**

## How will this assignment be graded?

|  |  |
| --- | --- |
| Item | Point Value |
| All files / folders uploaded correctly | 10 |
| Wrote the code necessary to POST data to the addtask endpoint. Works. | 30 |
| When a task is added, a confirmation message in the <div> tag is provided.  | 30 |
| Wrote the code necessary to GET data from the tasks endpoint. Looped through the data returned by the API and appended each task to the existing unordered list. | 30 |