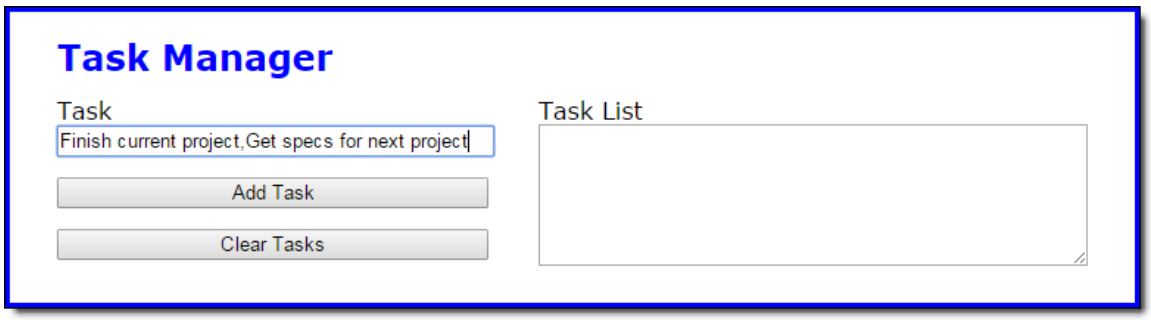
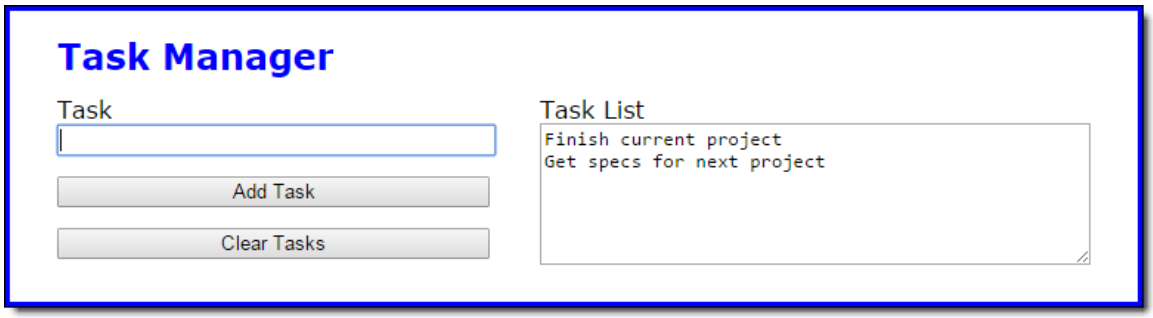
CST Lab9 Allow multiple task entries in the Task Manager app

In this application, you'll make an enhancement that allows you to enter multiple tasks separated by commas in a single entry.

Here is the enhanced application, with multiple tasks about to be entered:



And here is the application after the multiple tasks have been entered:



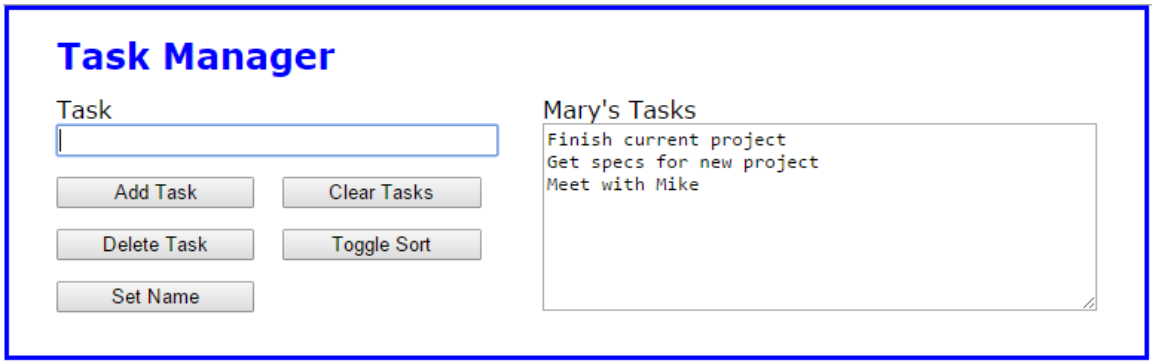
1. Open the HTML and JavaScript files on Moodle
2. Run the application and add two tasks, separated by a comma. Note that the tasks are stored as one task, exactly as you entered it.
3. In the JavaScript file, find the code in the addToTaskList function that adds the task entered by the user to the tasks array. Comment out that code, and replace it with code that works for one task in an entry and also for more than one task in an entry.

   To do that, the split method of the String object may be useful.

   You may also want to use the array concat method.

Part 2 – Implement the new buttons

The enhanced version of the Task Manager has 3 new buttons:



1. Open the files on Moodle in the `lab9/task_manager_buttons/` folder
2. Test the application in Chrome. Only the 'Add Tasks' and 'Clear Tasks' buttons are working.
3. Press the F12 key to review the local storage items.
4. Review the JavaScript file for the application. First, there's another global variable named sortDirection with an initial value "ASC" (ascending). Second, there are 3 empty function expressions called deleteTask, toggleSort and setName. Last, the onload event handler attaches these functions as the event handlers for the click events of the 3 new buttons.

Code the deleteTask event handler

5. Code the deleteTask function so it uses the prompt method to ask the user for the index number of the task to delete. Assume the entry is valid, and use the splice method of the tasks array to delete the element at the specified index. Then, use the join method and the pipe separator ("|") to create a string from the tasks array, update the localStorage value with the string, and call the displayTaskList function to re-display the tasks. Now, test the function.
6. Add data validation to this function so the user's entry has to be a number, but don't display a message if it isn't. Then, test this change to make sure that nothing is done if you enter an index value that isn't in the array or a value that isn't a number.

Code the toggleSort event handler

7. Code the toggleSort function so it sets the value of the global sortDirection variable based on its current value. So, if the current value is "ASC", change it to "DESC", and vice versa. Then call the displayTaskList function to re-display the tasks in the page.

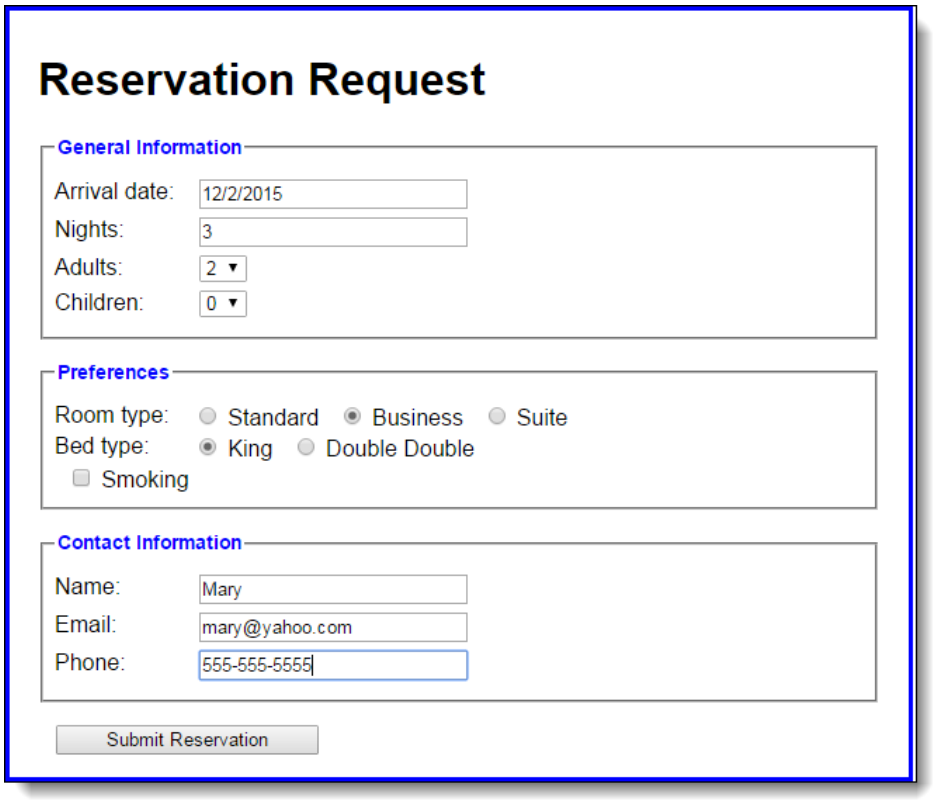
8. In the displayTaskList function, find the line of code that calls the tasks array's sort method, and comment it out. Below this line, add an if statement that checks the value of the sortDirection variable. If the value is "ASC", call the tasks array's sort method. Otherwise, call the tasks array's reverse method. Test the change.

Code the setName event handler

9. Code the setName function so it uses the prompt method to ask the user for a name. It should then store the name in session storage and call the displayTaskList function to re-display the tasks in the page.
10. In the displayTaskList function, add code that gets the name value from session storage or an empty string if there's nothing in session storage. Then, set the value of the span element with "name" as its id to either the name from session storage or an empty string if there's nothing in session storage. Add an apostrophe and s ('s) after the name as shown above.

Part 3 Save a reservation in session storage

In this exercise, you'll develop an application that stores data in session storage. The interface looks like this:



Then, when you click on the Submit Reservation button, a new page gets the data from session storage and displays it like this:

4. Open the HTML and JavaScript files in this folder.
5. In the index.html file, note the coding for the form element. It's action attribute calls the response.html page. Also, at the bottom of the form element, a button element (not a submit button) is used for the button that submits the form.
6. In the response.html file, note that there is an embedded script tag within the main element in the body of the document, and it contains document.write methods that get the data that's submitted from session storage. This is the page that's called when the form is submitted.

7. In the JavaScript file, note that three functions are supplied. The $ function. The start of a saveReservation function that ends by submitting the form. And an onload event handler that attaches the saveReservation function to the click event of the Submit Reservation button and sets the focus on the first textbox on the page.

8. Run this application without entering any data and click the submit button to see how the response.html file will look when no data has been saved to session storage.

9. In the saveReservation function of the JavaScript file, get the values from the controls on the page and store them in session storage using the same key names as the response.html file uses. But don’t bother doing any data validation because that isn’t the point of this exercise.