Course:	**INFO3144**
Professor:	**Jim Cooper**
Project:	**Project #1 – JavaScript Spreadsheet**
Due Dates:	**Tuesday, November 10, 2020, 11:59 PM**
Submitting:	**Please see the last page for instructions.**

How will my project 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	Clear all literals and formulas from the spreadsheet with “Clear” button	
2	Ability to enter text or numbers into the spreadsheet using single text box	
6	Ability to enter and validate the =SUM(Ax:By) formula	
4	Automatic recalculation on change	
2	Numeric row labels and alpha column labels (20 x 10)	
2	Cell mouse selection similar to Excel Sheet	
2	Spreadsheet table created dynamically at run-time	
2	Proper final submission	
5	Use of Open Broadcaster Debut Video Recorder or similar software to create a demo video	
30	Total	

Project Description

Build a web application that allows users to create and edit simplified spreadsheets running within a web browser. Your solution can be created using Visual Studio or a standalone HTML web page (using as well CSS and JavaScript). The application must be written to manipulate the spreadsheet client using JavaScript, CSS and the DOM.

Much of the work of rendering the spreadsheet is already done for you with the DHTMLTable.html example on FOL although it is not a requirement that you use this code. See the DHTML_Table_API.html example for an alternative.

Specific Requirements:

The following is a list of constraints and requirements for your spreadsheet application:

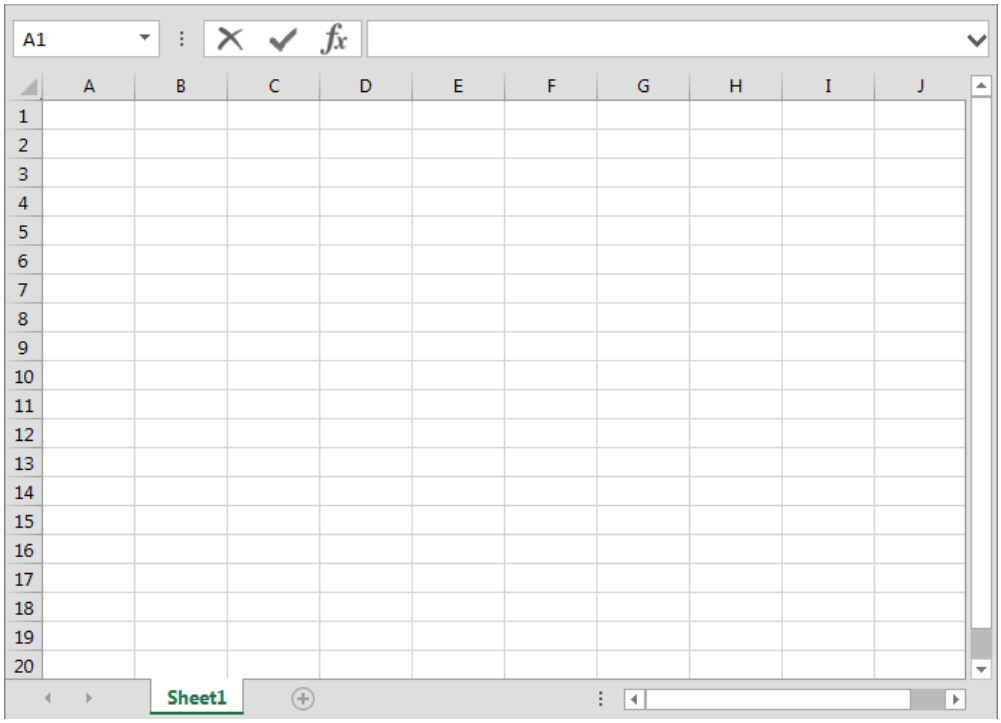
1. You may build your web application to run on any of the following browsers: Edge, FireFox, Chrome.

2. All spreadsheet processing; i.e. cell entry, interpretation and recalculation must be done using JavaScript, CSS and DOM calls within the browser environment.

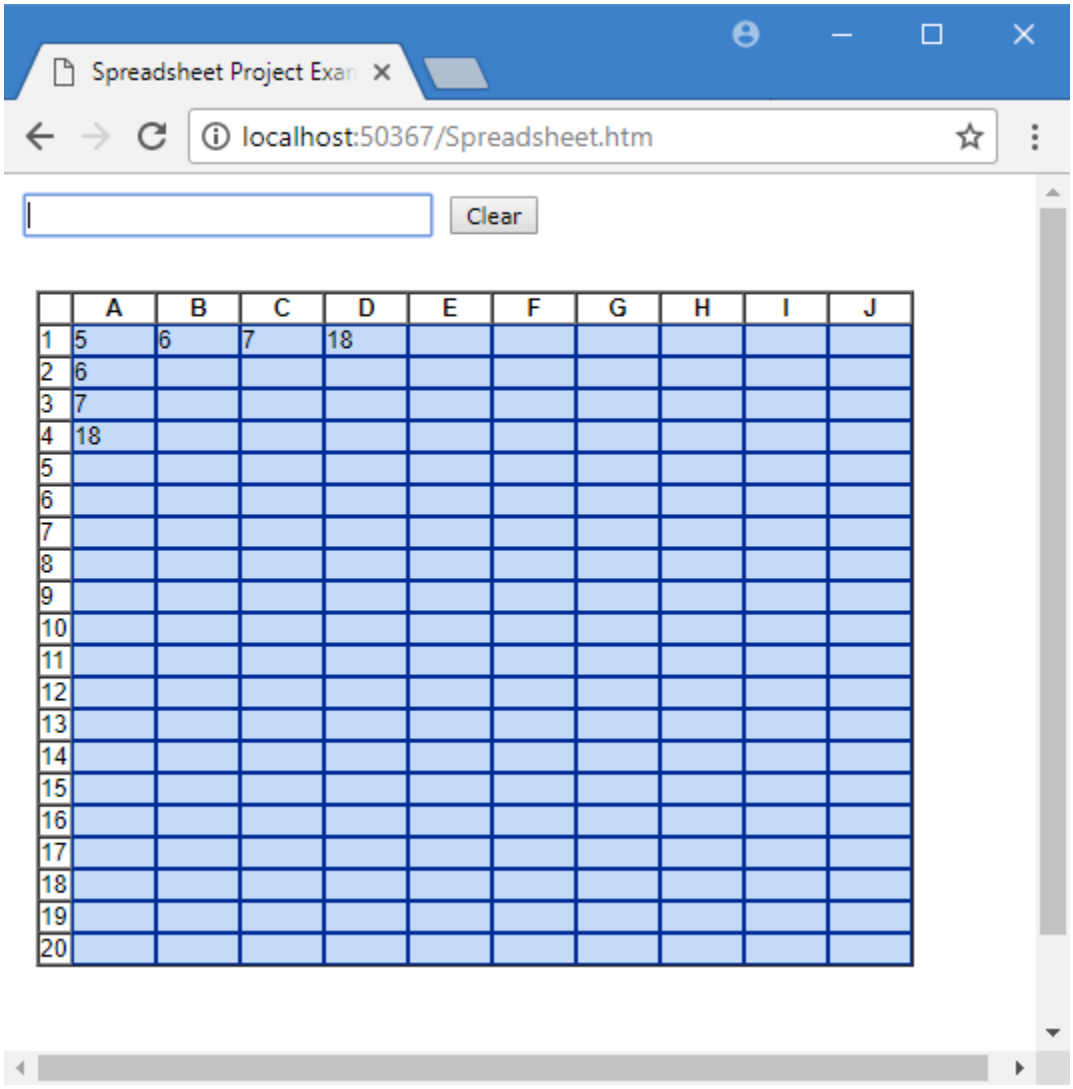
3. You must be able to handle data entry at the cell level, as text, literal numbers or formulas.

4. The only formula you will need to handle will be =SUM(Ax:By) where “A” and “B” represent *from* and *to* columns and “x” and “y” represent *from* and *to* rows – as with MS Excel or Google Sheets. The objective being to sum a set of cells and display the result. For the purpose of this project, you can assume that all Ax and Bx coordinates will represent a column or row of numbers.

5. Your spreadsheet should display row numbers and column letters (MS Excel), somewhat like the following screen capture from Excel 2019 however other than the spreadsheet itself, you only need a text box to enter data for the currently selected cell and a clear button.



6. The following represents a minimalistic version of what's required.



7. Your spreadsheet should support 10 columns (A – J) and 20 rows (1 – 20).

8. Depict clearly to the user, which cell is currently selected using a rendering strategy similar to Excel; i.e. border change, background colour change, etc.

9. You will also need to have a text box where the user can edit the contents of the currently selected cell.

10. Your application must also provide a means of clearing all the cells with a single user event (a "Clear" button for example).

11. As with Excel, we want your spreadsheet to automatically recalculate whenever the user updates a literal numeric cell or changes a formula cell.

This recalculation processing should be done whenever the user presses the enter key within the text box. See the note at the end regarding this feature.

12. Your JavaScript code must determine during recalculation, what type of data is in each cell; i.e. a literal text string (numeric or alpha) or a formula. If you detect a formula, you must perform the designated (sum) calculation and output the result.

13. The normal way to render this kind of 2D array of data within an HTML (or ASPX) page is to use an HTML table. One of the requirements for this project is that you build the table at run-time rather than hard-code the table into your page. This should be done using dynamic HTML practices following the techniques discussed in class.

14. You can use third party UI libraries such as jQuery to enhance your ability to render controls etc. however the core solution must be your own JavaScript code. It would defeat the purpose of the project to simply use an open source JavaScript spreadsheet library.

15. Create a brief video recording which demonstrates that your submission meets the requirements. Ideally, your recording should include audio as well.

Notes:

The normal way to solve this kind of problem is to have an HTML table to render the output and a 2D JavaScript array to store the editable data, including formulas. When a user clicks on a cell, the data is copied to the text box from the table. In this way users edit formulas, but the output from a formula is always placed in the HTML table cell, rather than the JavaScript array.

In order to check for the enter key, you'll need to process each keystroke separately for the check box. See the "Filtered_Textbox_Example" code from INFO-3114.

In addition, you'll need to add an "onSubmit" event handler to your form tag in the main web page to prevent the page from refreshing each time the user presses the enter key:

```
<form id="frmSpreadsheet" onSubmit="return false;">
```

How should I submit my project?

Submissions Folder (FOL):

Submit your program files to the *INFO3144 "Project 1"* dropbox in *FanshaweOnline*. These files should be submitted as a single "zip" file containing your web application's complete website.

When grading your submission, I will want to unzip your web site and find either a single standalone HTML page or a combination of HTML page, CSS and JavaScript files that I can run using the browser you designate.

In addition, you should also include a single video recording file as a separate submission.

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.