# Assignment: Kitchen Calculators (JavaScript functions and events)

This assignment is to be begun in class, and completed on your own, outside of class. Work through the following steps and upload your work to your wwwroot on misdemo.temple.edu when you are done.

HINT: If, at any point in this assignment, you do not get the expected result, the ***very first thing*** you should do is to open the web developer tools in Chrome and look at the console. Very often, the hint you need to fix your mistake can be found there!

## Getting the project file

1. Using Windows file explorer (or Mac OS finder) open your mis2402workspace
2. Your instructor will provide you with a zip file named: kitchen\_calculators.zip
3. Open the zip file. Drag the kitchen\_calculators folder in to the mis2402workspace folder.
4. Now, open VS Code. Navigate to the kitchen\_calculators folder using the VS Code Explorer panel.
5. Observe the contents of the files in the folder.

## Overview

1. Our goal in this assignment is to practice writing JavaScript functions and click events. Assuming you complete this assignment entirely you will define nine click events and nine functions. You also have the option of defining even more buttons with click events, and more functions if you want to.

We’ll also use this assignment as an opportunity to revisit one or two bootstrap concepts.

Nearly all the functions in the assignment follow the same pattern. Once you have one working function is written, you will find it very tempting to copy/paste it, and then alter the new function slightly to suit your purposes. An experienced programmer would most certainly do this. But, in this assignment, students are urged to resist this temptation! In typing the click event definitions and the function expressions in this exercise, you are reinforcing your own memory and understanding of the correct JavaScript syntax. The typing is part of the learning!

In this assignment we will create a simple application which converts units of measure commonly used in cooking. A screenshot of the completed application follows:



## Mathematic Trickery

1. In this assignment you will need to round a floating point number (that is, a number with decimal precision) to 3 digits. There’s more than one way to do this in JavaScript, but we’re going to employ a technique that uses functionality that we have already encountered. The toFixed method and the parseFloat() function. An example can be seen below:



See also rounding.html found in the start file.

## Data

1. You will find the following table helpful.

|  |  |  |
| --- | --- | --- |
| Unit | Common Abbreviation | Equivalency |
| Teaspoon | tsp | 1 teaspoon |
| Tablespoon | Tbl | 3 teaspoons |
| Ounce | oz | 6 teaspoons |
| Cup | cup | 48 teaspoons |
| Pint | pint | 96 teaspoons |

Instructions

1. Start by inspecting index.html. Notice that …
	1. There are two, 4 span bootstrap columns present. Later in the assignment you be instructed to add an extra 4-span column taking the total number of spans up to 12.
	2. For purely cosmetic purposes, the buttons in the start file are styled with the bootstrap classes of bg-primary and bg-secondary. Later in the assignment you will be instructed to add a button of the class btn-default.
	3. There is a 12-span column at the top of the document. It contains a div with an id of “message”. This is where we will eventually want to direct our output.
2. There is also an <h1> tag at the top of the document. Add an <i> tag with a font-awesome class of fa-calculator to the inner html of the <h1> tag
3. Now add the “document ready” event to the <script> tag. Notice that there are comments in the script tag to suggest that the script will have both a “document ready” section and a “supporting functions” section. This convention will be followed later in the course as well. Type the document ready definition carefully. Type:
	1. $(document).ready();
	2. Then, inside the parenthesis of ready add function(){}
	3. Then, add a line break between the curly braces { }
	4. Now you are ready to write your code!
4. Now, ***inside the document ready handler***, add in the click event handler for the button named button\_tsp\_tbl
	1. Let’s start with a comment. There are lot of buttons, so some comments are going to be helpful.

//teaspoons to tablespoons button

* 1. Now, we’ll add the click event handler. Type:
	$('#button\_tsp\_tbl').click();
	2. Now, inside the click parenthesis, define the function that will execute when the click event takes place: function(){}
	3. Now, back up. Add a line break between the curly braces { }
	4. You are now ready to write the code that handles the click event!
1. Here’s our code for the click event handler code for our first button.



Notice that:

* 1. There’s a certain amount of wishful thinking here. The function convertTspTbl does not exist yet. We still need to write it! But, what we are wishing for is a function that will take one parameter (the number of teaspoons) and return the number of tablespoons.
	2. We are using console.log here to report our answer. Once we are satisfied that we are getting the correct answers we will replace console.log with a command that will insert the answer into the inner html of the message div.
	3. When passing a parameter to a function, students often become overly concerned about the name of the variable being passed in. In these three lines of code, we have a variable named “tsp” … we could have named it anything! “teaspoons” “input” or even, simply, “x” would all have been suitable names. When we pass a parameter into a function, what matters is the ***value*** of the variable (e.g. 10, 3.5, 1, 0) not the variable’s identifier.
1. All right, it’s time to make all our wishes come true. ***Outside of the ready section***, in the “supporting functions” section of our code we can define the convertTspTbl function. We do so as follows:



Notice that:

* The name of our parameter here is “t” … it will receive the **value** passed into the function.
* We are using isNaN to validate the value of “t”. (So, any bad/nonsense input is handled here.)
* The “result” variable is used to store the output of the calculation and then returned.
* Instead of naming that variable “result” any number alternate names could be used: “y”, ”output”, “answer”, “tablespoons”, etc.
1. Let’s try it. Run your code and see what happens. Be sure to look in the web developer console for JavaScript error messages. The odds of typing many lines of code without any sort of mistake is not good. Knowing how to identify and fix your mistakes is just as important as writing the code in the first place!

## On your own

1. Now that you have one function and one button click handler working, write some more! As you do, try varying the names used for the “result” variable and the variable names used as parameters. This is *not* something you would do in work-setting (where consistency is highly prized!) but it is a good learning exercise to vary these names a little bit. By doing that you can better understand what the different variables are representing in each portion of code.

Here are some suggested names of the remaining functions.

convertTspOz

\* For the last three functions, you will need to add the HTML for the buttons themselves in addition to everything else!

convertTspCups

convertTspPints

convertTblTsp

convertOzTsp \*

convertCupsTsp \*

convertPintsTsp \*

1. Once you have all your calculations working correctly, consider how to clean up your output. If the answer to a calculation is “0.3333333333 Tablespoons” it would be nice if the repeated decimal was rounded accordingly (for example: “0.333 Tablespoons”) Also – we have a little trouble with the number 1 and the way the English language works. When our calculations result in the number one, it would be nice if the output of our program was “1 Tablespoon” not “1 Tablespoons”.

So, to clean up our output, consider the following supporting function code:



Below is an example of how this function can be used to clean up our output as written to the console log.

console.log(prepareOutput(answer,'tablespoon','tablespoons'));

1. Now it’s time to write our output into the message <div> tag. You can replace the console log statements with commands similar to the following:

$('#message').html(prepareOutput(answer,'tablespoon','tablespoons'));

## The last button

1. Add a new, 4 span, bootstrap column to your html and specify a button of class bg-default. (Again, this class assignment is purely for cosmetic purposes!) The new button is to convert cups to pints. A sample screenshot follows.



1. Add a click event handler for this new button. Notice that, in this case, you could write a specific cups-to-pints conversion function, ***but you don’t really need to***. You have already written all the supporting functions you need. To convert cups to pints, you can first convert cups to teaspoons, and then convert teaspoons to pints. And … ***you already wrote those functions***! Here is a sample click event handler for your new button. Be sure that you can follow the logic of it.



1. Be sure to upload your work before the due date / time. This assignment will be graded.

**Step back, think, simplify (OPTIONAL)**

There is always more than one way to solve a programming problem! In this challenge we created many functions that carried out very similar operations. Now that you have a working solution, you might want to step back and think about a more concise way of expressing the same logic with fewer lines of code.

One possible approach would be to consolidate the eight teaspoon-related functions into two functions. One function could be convertToTsp and the other could be convertFromTsp. You would need to pass in the number to convert as the first parameter of the function and then indicate the number to multiply or divide by (e.g. 3,6 -or- 48,96) as a second parameter. Give it a try!

**CONTINUED…**

## How will this assignment be graded?

|  |  |
| --- | --- |
| Item | Point Value |
| Uploaded OK?(solution must be ***exactly*** where requested!) | 10 |
| Primary button 1 works as expected | 10 |
| Primary button 2 works as expected | 10 |
| Primary button 3 works as expected | 10 |
| Primary button 4 works as expected | 10 |
| Secondary button 1 works as expected | 10 |
| Secondary button 2 works as expected | 10 |
| Secondary button 3 works as expected | 10 |
| Secondary button 4 works as expected | 10 |
| “Cups to Pints” works as expected and is written using other functions as seen in step 20 | 10 |