# MIS2402 – Assignment 02

## JavaScript Functions, Variables and Simple Math

### Instructions:

1. Connect to the class server. Use the username and password your instructor gave you. The class server’s hostname is: **misdemo.temple.edu**
2. After your successful login, click the “New terminal console” icon in Bitvise.
3. At the prompt, type:

cd wwwroot

1. At the prompt, type in this wget command:

wget **https**://misdemo.temple.edu/assignments/assignment02.zip
2. Unzip the file with an “unzip” command. This will create a file named piesbypatty.html.

unzip assignment02.zip

Now, in Chrome, type this URL into your browser:

https://misdemo.temple.edu/**tuz54321x**/piesbypatty.html

You should replace tuz54321x with your own account.

1. You will be prompted to provide your misdemo username and password. After you do that. You should see a web page that looks like this:



1. Now, back in Bitvise, open the piesbypatty.html file with nano.

nano piesbypatty.html

1. Look inside the <script> tag in our html file. We will edit the functions found there.

## Scenario

Patty is an entrepreneurial young woman. Patty is very good at baking, and especially good at baking cherry pies. We are going to help Patty with her pie baking business.

To keep our imaginary scenario simple, we are going to pretend that Patty has access to her parents’ kitchen, where she has unlimited access to a conventional oven, and that she does not have to worry about things like paying the gas or the electric bill to make that oven work!

On the form you will find on input that allows Patty to enter how many pies she needs to prepare.

Our JavaScript functions will then calculate:

* How many boxes of pie crust she needs to buy
* How many 32-ounce quarts of cherry pie filling she needs to buy
* How much her ingredients will cost
* How much time she needs to bake
1. **(This step is a repeat from the in-class activity!)** Find the function “calcBoxes” and complete it. Patty can make two pies with a box of premade pie crusts. So, that gets a little tricky… to make ***one*** pie, she needs a box of crusts. To make ***two*** pies, she needs a box of pie crusts. To make ***three*** pies, she needs ***two*** boxes of pie crusts (and she would have one crust left over!)

How will we solve this? We will need to divide the number of pies by two *and then round up*. How can we do that? JavaScript gives us a “ceiling” function that will round up to the next integer for us. Careful! The “M” in “Math.ceil()” is upper case and the method is spelled C – E – I - L

Here is some code to help you with the calcBoxes function.

let answer = pies / 2;

answer = Math.ceil(answer); //reassignment

return answer;

1. Test your work. Notice that we are taking it on faith that the user will enter a nice, reasonable number in the text box. That is ok for this week, but soon we will need to think about how to deal with incorrect or unexpected values that a user might provide as input.
2. Good news! It turns out that if Patty buys pie filling by the quart, it’s cheaper. Complete the “calcQuarts” function. A quart is 32 ounces. A pie requires 42 ounces of pie filling. This function should calculate how many quart containers of pie filling Patty needs to buy.
3. Complete the “calcCost” function using the existing “calcBoxes” and “calcQuarts” function. A box of pie crusts costs $4.29 and a quart container of pie filling costs $4.25.

Your calcCost function should return an answer that looks like a dollar value (that is, round your answer to two decimal places using toFixed and put a “$” in front of it!)

1. Complete the “calcHours” function. It takes an hour to bake a pie, and in Patty’s oven she can cook up to 4 pies at a time.
2. Save your work in nano. Refresh your web page. Check your work. Does your code run? Do your answers make sense? Here is some sample output:

 

1. Determine the URL for your work. Go find the corresponding assignment on canvas and put the URL to your work here.

## How will this assignment be graded?

100 – You provided a good, working URL, everything runs, and your answers are correct.

80 – Your code has one or two small problems (such as a miscalculation or a formatting problem)

50 – Your code has multiple problems, but it does not crash and you made an effort.

0 – Your code is missing, your code completely crashes, or you failed to provide a good URL.