JavaScript Assignment #5

Grading Note: Grading will be based on whether your page does what it is supposed to do, but also that it is written properly with correct HTML 5, CSS, and JavaScript. If any of your assignment parts isn’t working right but you are turning it in anyway, be sure to indicate that it doesn’t work properly. I will discover this anyway, but you might save me a lot of time and keep me in a better grading mood!

**Part 1: Hooray for Arrays!**(Intro to Arrays as a discounted price list – using A4P4 as a starting point)

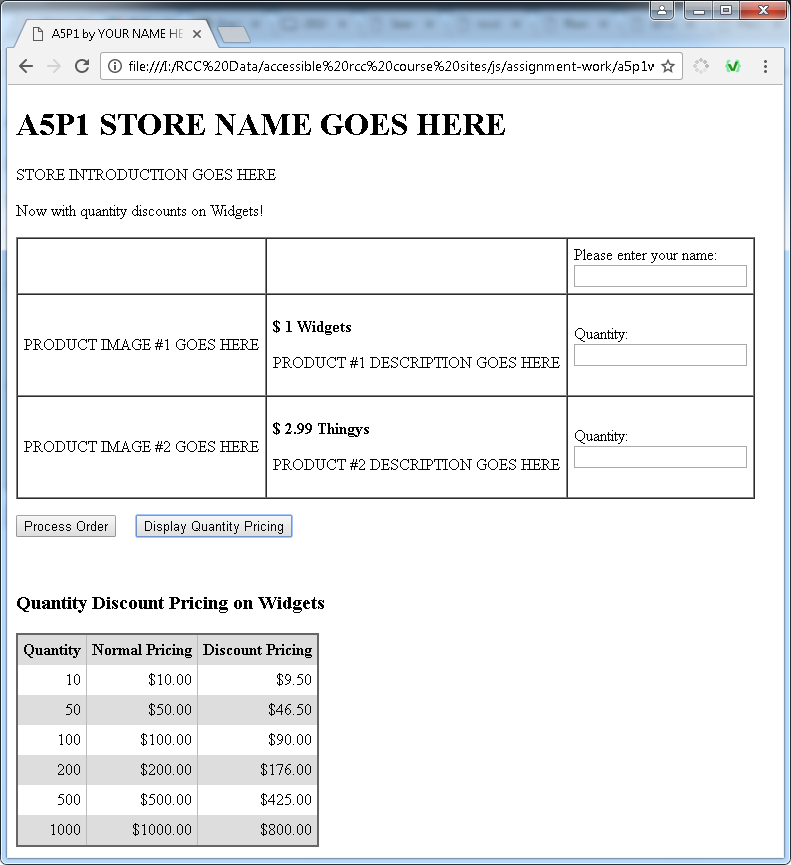
Begin by making a copy of your A4p4 file and call it A5P1. To refresh your memory, A4P4 was called "Table Display" and produced a nicely styled table of discounted prices for quantities of 10, 20, 30, 40, 50, 60, 70, 80, 90, and 100, all of which were discounted 10% (90% of normal pricing). But in most practical business situations, quantity discounts would vary based on the quantity being ordered. For example, you might expect to get a bigger percentage discount for buying 1000 than if you bought only 10. To accommodate these types of discount schemes, you are asked to produce a table of discounts that take their quantity price breaks from data stored in an array. The discount strategy is shown below:

|  |  |
| --- | --- |
| Quantity | Discount % |
| 10 | 5% |
| 50 | 7% |
| 100 | 10% |
| 200 | 12% |
| 500 | 15% |
| 1000 | 20% |

Using this table of price breaks, create 2 (parallel) arrays, one to hold the given quantities and another to hold the given discount percentages. Note that you probably do not want to use the "%" sign in your actual array – instead use a number like 5 or 0.05 or maybe 0.95 to represent the 5% discount, depending on how you want to use it in your calculations.

To make it easier to debug for you (and to grade for me) change your global price variable's value from its current price to an even dollar amount (zero cents). This is only for the product that is receiving the discounts. For example, set it to $1 or $10 or $100. Then you (and I) can more easily check to confirm that the proper discounts are being given. Your table of discounts should display using the same "div" and same CSS styling as in A4P4. Optionally you could add another column to the table shown below that indicates the discount % (using the % numbers from your array) to help customers understand the discounts.

As an example of what your results should display for a normal price of $1, see the screen shot below:



Use a “for” loop to work your way through the arrays of quantities and discounts to create your table display. Use the array’s length property to know how many items your array holds so you can control your for loop counter properly.

**What to turn in:**

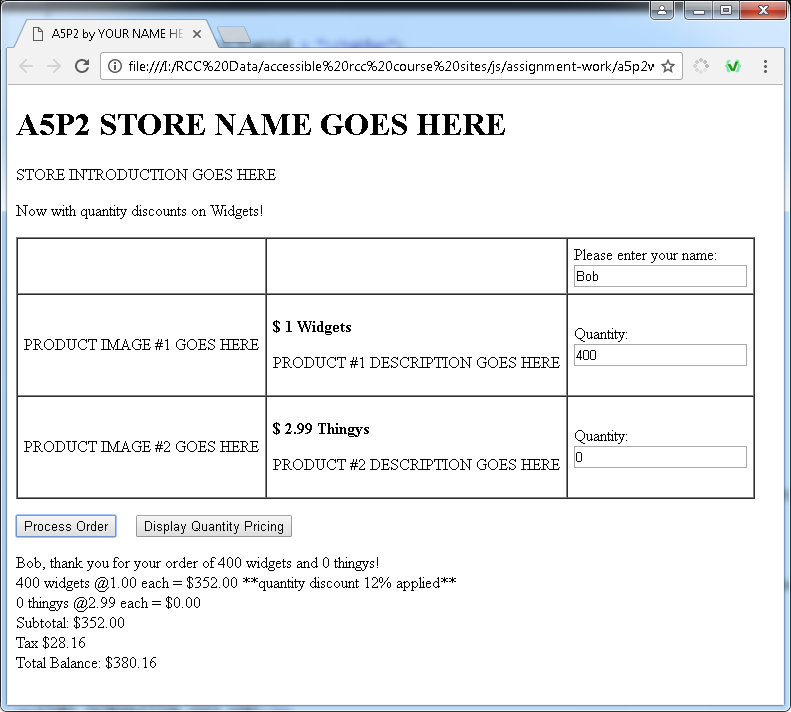
**Screen shot**: Into the given assignment 5 “turn it in” Word file's part 1 screen shot area, paste a copy of all of your web page displaying in the browser after clicking just the "Display Quantity Pricing" button. The result should be something very similar the sample screen shot above. Note that it may not all fit on one screen shot. If that is the case, use 2 screen shots as needed. All of the discount price list table should be visible.

**Code**: Into the given assignment 5 “turn it in” Word file, paste a copy of all of your web page’s code into the part 1 code section. It should contain **ALL** of your page’s code. Do NOT paste a screenshot of code, paste the actual text of the code. **Make sure all of your code is visible** within the Word file's margins! You can format the font, spacing, tabbing, or size of the code after pasting into Word to improve its appearance and readability, just don’t change any of the code itself.

**Part 2: Stop That False Advertising!**Start discounting prices when selling products! Calculate the discount with a function!

Begin by making a copy of your A5P1 (this assignment's part 1) page and calling your new copy “a5p2.html”. This part will now put the discount pricing into effect on purchases of large quantities based on the percentages advertised by the table shown at the bottom of your part 1 screenshot. For example, if the customer fills out the order form and orders 10 (or any number between 10 and 49) of your main discounted product item, they should be given the 5% discount that was quoted in the quantity pricing table display. If the customer wants between 50 and 99 of your item, a 7% discount should apply, and so on according to the above table from part 1.

To do so, write a new function called "calculateDiscount" or something similar, and it should be passed as parameters the quantity being ordered and the 2 arrays used to determine the discount percentage, and it should look through the array (using an appropriate loop) to figure out and return the percentage discount to apply. (For example, it might return a value of 0.15 if the person ordered 600 items and it would return 0.0 if the person ordered 9 or fewer.) When that function returns its discount percentage back to the "order processing" function that called it, the order processing can continue to calculate the cost and produce the receipt by applying the discount that this function returned to it. Place this new function above all other functions so that it is toward the top of your script. A screenshot appears below:



Notice that the receipt includes a note that indicates what specific % discount was applied (12% in this case since 400 items were ordered.) Your receipt should do the same (although different language can be used.)

**What to turn in:**

**Screen shot**: Into the given assignment 5 “turn it in” Word file's part 2 screen shot area, paste a copy of all of your web page displaying in the browser after clicking just the "Process Order" button. The result should be something very similar the sample screen shot above. Note that it may not all fit on one screen shot. If that is the case, use 2 screen shots as needed. All of the order form and receipt should be visible.

**Code**: Into the given assignment 5 “turn it in” Word file, paste a copy of all of your web page’s code into the part 2 code section of the document. It should contain **ALL** of your page’s code. Do NOT paste a screenshot of code, paste the actual text of the code. **Make sure all of your code is visible** within the Word file's margins! You can format the font, spacing, tabbing, or size of the code after pasting into Word to improve its appearance and readability, just be sure not to actually change any of the code itself.

**A5P3 Getting to know your new friend "Dom"  
Required Fields with Error Messaging and getElementsByClassName**

To begin this part, copy your previous a5p2 part's file, and name the new copy A5P3.html. In this part you will work with some built-in features that the browser provides to JavaScript programmers having to do with the current document being displayed by the browser (the current page). The Document Object Model (or DOM) is the browser's dynamically built object that contains a representation of every element on your page. Your ability to access and manipulate the objects, properties and methods of the DOM will largely affect your ability to successfully program in JavaScript. It is that important. You have already worked with "document.getElementById", and a few properties of a few element objects like the "value" property of text box objects, but that is just the tip of the iceberg. Here in this part (and again in part 4), we explore a little more of the DOM, with some continued emphasis on arrays.

For this version of the store, you will add JavaScript that uses "getElementsByClassName" method to examine the required textboxes to see if any have been left blank by the user and respond with some custom user friendly error messaging that grabs their attention and helps them to fill out the form completely before processing their order. Browsers can do some of this by themselves based on your HTML 5 code these days, but JavaScript allows you to control and customize the messages and reactions to any of your required fields that are not filled out properly. In our case you want to make both of your quantity textboxes to be required (but not the name textbox). So in your HTML, set the "class" of those 2 textboxes to "required" (or something similar) and then use "document.getElementsByClassName" to return an array of those (2) textboxes for you to work with. By doing it this way, your code should work even if we add more required textboxes in the future as long as we give them the same class.

Here is how your script should behave as it interacts with the user who is not filling things out properly at first but then gradually gets it right:

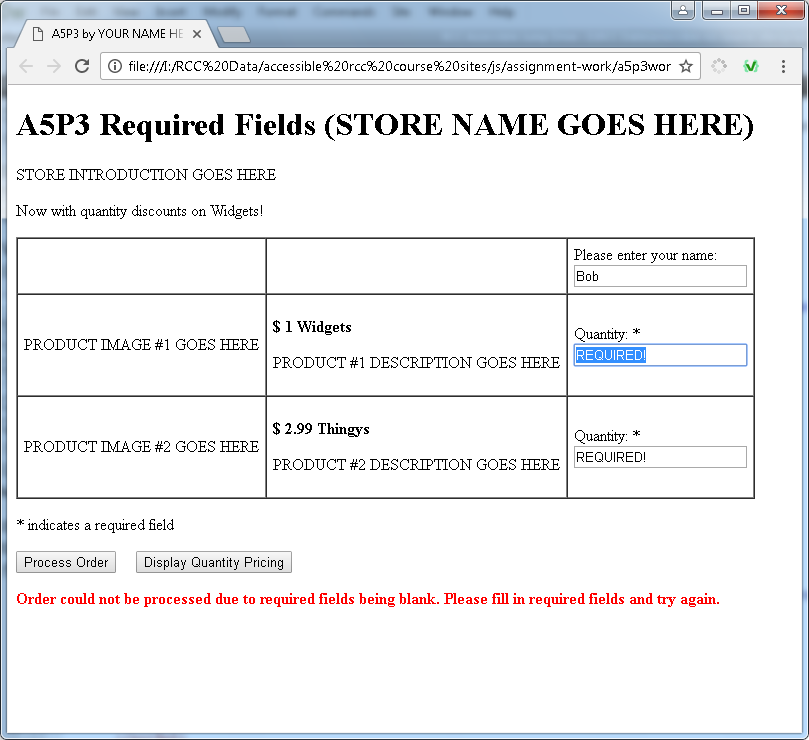
After filling out the form by entering only the name but no quantities and then clicking "Process Order", your script should detect the 2 empty required fields and…

1) fill both of them with the word "REQUIRED!",

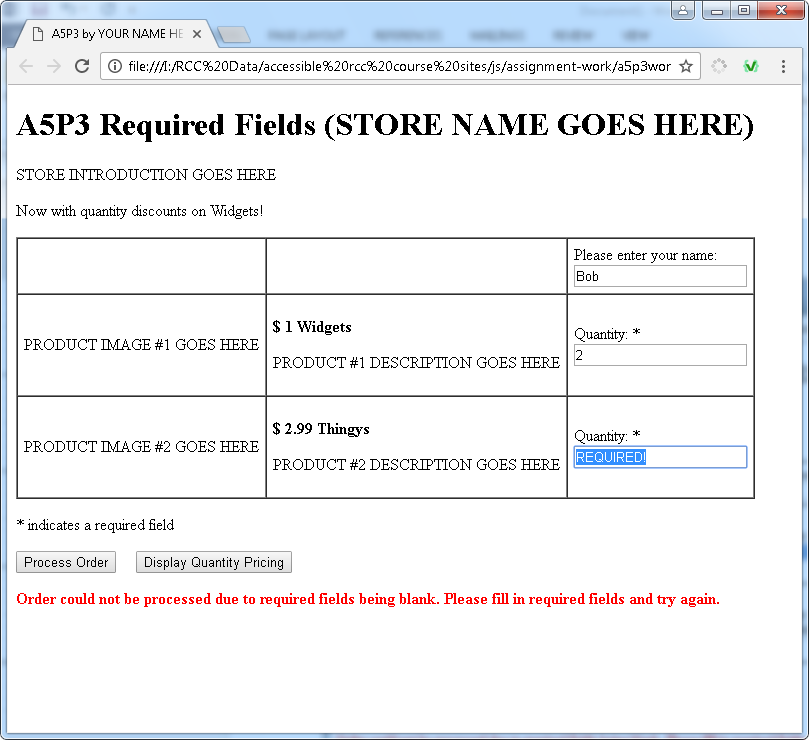
2) should fill the HTML for the receipt with an error message using bold and the color red (message should be similar to that shown below), and…

3) Should select the text of the first textbox that was not filled out properly, as shown below.

To select a textbox, use the "**.select()**" method of the textbox element.

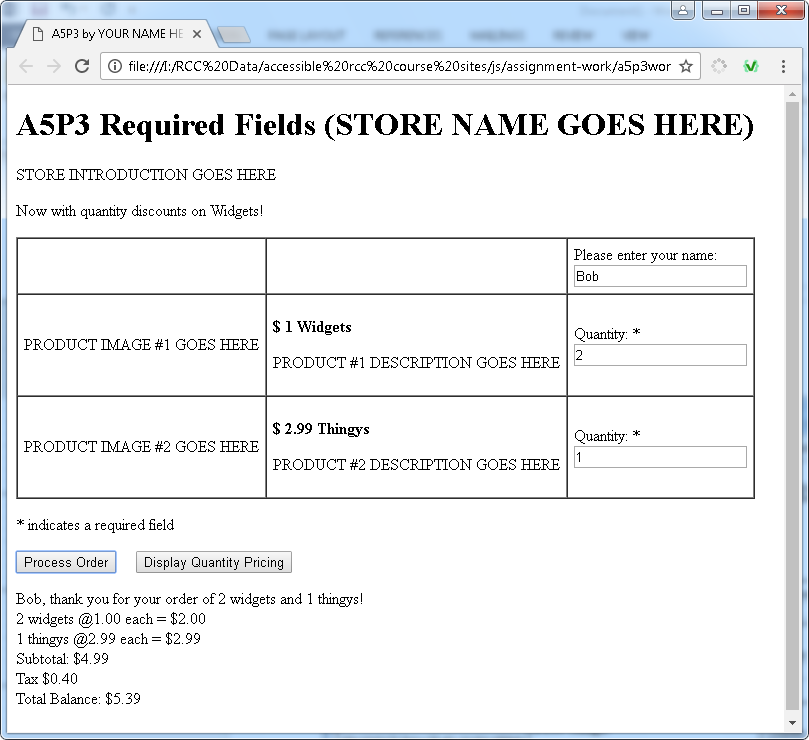


Next, if the user fills out the first required field properly (like entering a 2 as shown below) and clicks "Process Order", your script should detect that the second textbox still wasn't filled out properly and act accordingly (the textbox wasn't blank, but it was left with "REQUIRED!" still in it). The result is shown below.



(See next page for the last screen shot and description.)

Finally, if that second textbox is then filled out properly (by entering a number like 1 shown below) and the "Process Order" button is clicked, then your script should detect that all required fields have now been filled out properly and should replace the existing bold and red error message text with the appropriate black and "normal" (non-bold) receipt text as shown below.



Note that to change the style of any element within your HTML, just get your hands on that element somehow, such as with getElementById, and then change its "style" property. For example, to change the background color to yellow of a heading with an id of "main", this would work:

**document.getElementById('main').style.backgroundColor = 'yellow';**

Note that you can change any CSS property this way, where multiple word CSS properties that have dashes between words convert over to JavaScript style properties with no dashes but with the second word capitalized –so "background-color" becomes "backgroundColor", and "text-align" becomes "textAlign", and so on.

See last page for "what to turn in"…

**What to turn in:**

**3 Screen shots**: Into the given assignment 5 “turn it in” Word file's part 3 screen shots area, paste a copy of 3 screen shots, taken at the exact same times as the 3 above screen shots. :

1) Fill in your full name into the "name" textbox, but leave the other 2 quantity textboxes empty, then click your "Process Order" button and take a screen shot of the result. (It should similar to the 1st screen shot above.)

2) With the previous screen shot being your starting point for this one, fill in a quantity of 2 for the first product textbox but leave the second one empty and then click your "Process Order" button and take a screen shot of the result. (It should similar to the 2nd screen shot above.)

3) With the previous screen shot being your starting point for this one, fill in a quantity of 1 for the last product textbox and then click your "Process Order" button and take a screen shot of the result. (It should similar to the 3rd screen shot above.)

Note that if any of the 3 screen shots are tampered with (not produced by your actual code) you will be deducted a lot of points – for example, if you use your mouse to select the "REQUIRED!" message instead of doing it in JavaScript, that will result in a much more severe deduction than if you just left it the way it was in the browser.

Also realize that **these 3 screenshots must be obtained one after another after another while your same page is active in the browser the whole time!**

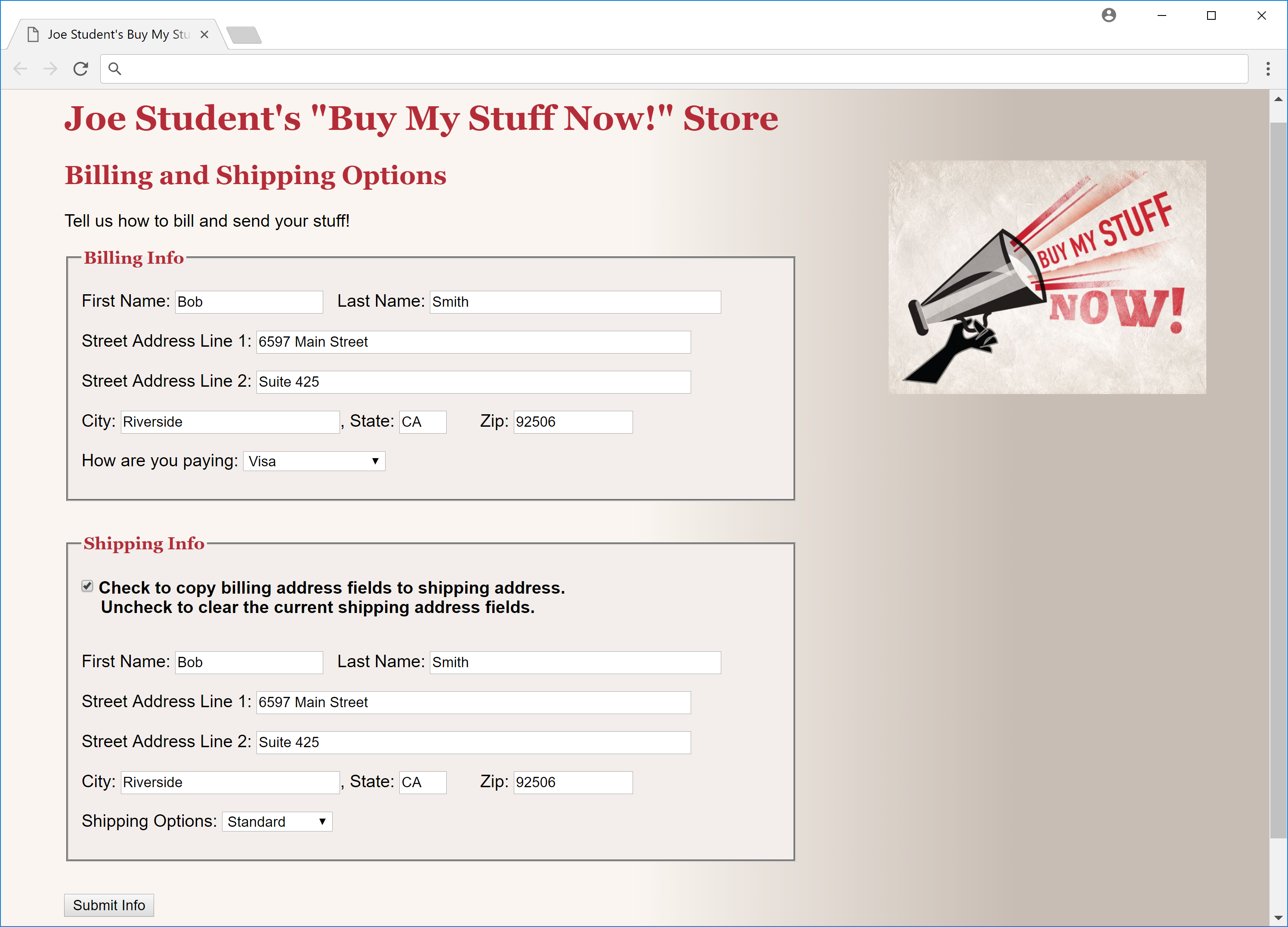
**Code**: Into the given assignment 5 “turn it in” Word file, paste a copy of all of your web page’s code into the part 3 code section of the document. It should contain **ALL** of your page’s code. Do NOT paste a screenshot of code, paste the actual text of the code. **Make sure all of your code is visible** within the Word file's margins! You can format the font, spacing, tabbing, or size of the code after pasting into Word to improve its appearance and readability, just be sure not to actually change any of the code itself.

**A5P4 More Form Work with getElementsByClassName and Arrays**

For this part you are going to download the a5p4 zip file and work with the files within the resultant folder. Once downloaded and extracted, you should find the following files: 2 style sheet files (ending in .css), 1 image (buymystuff.jpg), and 2 web pages, billing.html and billing-code.html. "Billing-code.html" shows you one way to code up the solution to this part in JavaScript, but the code does not use arrays. It does things "the long way" (more code than necessary). You are asked instead to use that example to help you write an alternative solution to the same problem that uses "getElementsByClassName" and arrays and "for" loops in a shorter way (less code). Your solution should be placed within the given "billing.html" page.

Essentially your code must process the shipping and billing address information when the user clicks the checkbox. Checking the checkbox should result in the address fields from the billing part of the form to be copied into each of the address fields within the shipping part of the form. Unchecking the checkbox should result in clearing out (setting to the empty string) each of the shipping address fields. This is all done for you by example in the "billing-code.html" page, so you can play around with that example page in the browser to clarify any questions you have about the desired behavior. Also, you can inspect its JavaScript and html code for ideas on how to react to the checkbox activity and the resulting function that is called to handle the aftermath of the checkbox clicks. Below is what your page should look like after the billing information section of the form has been filled out and then the checkbox is checked. Note that the drop down "select" lists are not part of any of the coding requirements since they do not contain address information.

Below is a screen shot of what your page should look like in the browser after filling out the billing info part of the form and clicking the checkbox.



**What to turn in:**

**Screen shot**: Into the given assignment 5 “turn it in” Word file's part 4 screen shot area, paste a copy of all of your web page displaying in the browser after filling out the billing information and then clicking the checkbox. The result should be something very similar the sample screen shot above. Make sure that your own name replaces all of the "Joe Students" that appear on the page. Note also that it may not all fit on one screen shot. If that is the case, use 2 screen shots as needed. All of the order form and receipt should be visible.

**Code**: Into the given assignment 5 “turn it in” Word file, paste a copy of all of your web page’s code into the part 4 code section of the document. It should contain **ALL** of your page’s code. Do NOT paste a screenshot of code, paste the actual text of the code. **Make sure all of your code is visible** within the Word file's margins! You can format the font, spacing, tabbing, or size of the code after pasting into Word to improve its appearance and readability, just be sure not to actually change any of the code itself.