Ross Studtman

CIS131 – Web Development II

First Project: “Change Calculator”

Shane May

Due: 4 October 2012

Document completion date: 19 September 2012

Disclaimer: I have coded a similar project from this site: <http://www.codecademy.com/courses/cash-register-mark-ii> . That project used recursion. Thus I will not be doing recursion in this project.

**Program Purpose:**

Receive a user selected quantity of pennies and return equivalent value in quarters, dimes, nickels, and pennies. The fewest number of coins possible should be returned.

**User Interaction with Program:**

1. Input quantity of pennies.

2. Click “calculate” button.

3. Observe program results.

**Program Input:**

Text box: quantity of pennies to convert.

**Program Output:**

Text box quarters: quantity of quarters to return.

Text box dimes: quantity of dimes to return.

Text box nickels: quantity of nickels to return.

Text box pennies: quantity of pennies to return.

**Valid Input:**

Quantity of pennies should be an integer. Instructions should request an integer. If an integer is not provided, show a popup and request a valid integer.

**Pseudo Code:**

1. User Input:
   1. Instructions, fill in textbox with integer quantity of pennies
2. User clicks “calculate” button
   1. Click event listener calls “calculate” handler function
3. “calculate” handler function
   1. Assign constants
      1. QUARTER = 25; DIME = 10; NICKEL = 5; PENNY = 1;
   2. Assign totalPennies = validInteger();
   3. “validInteger” function
      1. Obtain user’s input data
      2. While input is not a number, prompt for valid input
         1. totalPennies = valid user input
      3. Return totalPennies
   4. Obtain quantity of quarters:
      1. qtyQuarters = parseInt (totalPennies / QUARTER)
   5. pennies remaining after qtyQuarters
      1. totalPennies = totalPennies % QUARTERS
   6. obtain quantity of dimes:
      1. qtyDimes = parseInt ( totalPennies / DIMES)
   7. pennies remaining after qtyDimes:
      1. totalPennies %= DIME;
   8. obtain quantity of nickels:
      1. qtyNickels = parseInt( totalPennies / NICKELS)
   9. pennies remaining after qtyNickels:
      1. totalPennies %= NICKEL;
   10. obtain quantity of pennies:
       1. qtyPennies = totalPennies;
   11. assign coin quantities to text boxes and display to user

Repeating processes in above algorithm:

1. qtyCOIN = parseInt( totalPennies / coinNAME )
2. totalPennies %= coinNAME

functions can be distilled from these common processes:

function coinQuantiy( totalPennies, coinName){

return = parseInt( totalPennies / coinName);

}

Function remainingPennies( pennies, coinName){

Return = pennies % coinName;

}

Note: Doesn’t appear that repeating these function over and over in the code is any better than foregoing placing the functions in the code. Perhaps if the function calls were made from within a loop? This is left unexplored since it seems that would needlessly complicate the code. Likely will not implement these functions.

Addendum: creating two arrays of coin names and coin values and the looping over totalPennies while totalPenny quantity greater than coin value, this could be an option (however, I’m not sure if this is my idea or if I’m scavenging from the site noted in the disclaimer above). I will have to wait and see if the code looks inelegant when written as the pseudo code above describes.



**Programing Utilities:**

Program written with notepad++ with jsLint plugin.

Program tested with Firefox browser with firebug plugin.

Program designed for standards compliant web browsers.

**Issues:**

Click event handler is not safeguarded against pre-explorer 9.

**Testing:**

Since the program is intended to find the smallest number of coins to return given an amount of pennies, any number of pennies sufficiently greater than the largest coin value should be all that is required for testing valid values.

Test user input:

* with a string entered => expect prompt for valid input
* with spaces before and after integer input => expect proper function
* with alpha-numeric input => expect prompt for valid input

Test program algorithm:

* with input quantity greater than several times the largest coin’s value => expected results: never more than 2 dimes, never more than 1 nickel, never more than 4 pennies.

Testing Results:

Tests with numbers works as anticipated. Testing with non-numbers works as anticipated. Test with alphanumeric works as anticipated. Test with numeric-alpha works as if the letters didn’t exist and proper output, as if the number existed alone, is presented; this must be the handy work of parseInt.