Ross Studtman

CIS131 – Web Development 2

Project 3: Transaction Recorder

Shane May

11/08/2012

Problem Scope

Program provides for user to keep track of a money account’s withdrawals and deposits. The program allows user to input a date, select a transaction type (withdrawal or deposit), and input an amount. The program tracks and displays to the user: starting balance, running balance, and ending balance; as well as total deposits and withdrawal amounts. User must supply password to be granted access to this functionality.

Data Gathering

User’s dates should be in the format of mm/dd/yyyy exactly, and fall between the years of 1,000 through 3,000. Leap years and appropriate days per month, for all months, should be adhered to.

User’s input for amount should not be less than 0.0 or greater than 100,000.

User password must be one or more alphanumeric characters.

Functionality

Log in web page:

A single button for submitting user’s name and password form inputs.

Main Web page:

A single button allows the user to input their chosen data: date, transaction type, and amount. This data is used to perform calculations and update the following displays:

* Transaction display: showing date of, amount of, and running balance.
  + Withdrawals denoted by negative number.
* Starting Balance display.
* Total Deposits display.
* Total Withdrawals display.
* Ending Balance display.

Concerns

Program requested use of “array of arrays.” An object literal is basically an associative array. And objects are essentially collections of properties. Hence it was figured objects stuffed into an associative array met the requirement of an “array of arrays.”

Pseudo-Code

User Validation:

* Determine if user has permission to view web page.
  + Check cookies for password.
  + If no password, send user to log in form.
* Log in form, user inputs user name and password
  + Make cookie from password.
  + Cookie expires in 10 minutes.
  + Send user to account page.

Obtain user input:

* Date = new Date (user’s input on form)
  + User’s input in proper format of mm/dd/yyyy (exactly).
  + Year in range of 1000 to 3000.
* Type of transaction = user’s selection from form select.
* Amount of transaction = users input on form.
  + Amount in range of 0.0 to 100,000.

Functionality:

* User inputs data, clicks submit button on form.
* Validate data.
* Create new Transaction object.
* Add Transaction object to Account literal.
* Perform Account calculations.
* Update displays.



Operational Requirements

Program requires a computer with typical input and output devices. Program was written in Notepad++ with the aid of the JSLint plugin and Mozilla Firefox’s Firebug add on. Program’s Javascript, HTML, and CSS was ran and tested in Mozilla Firefox version 16.0.2.

Known Bugs or Limitations

Program may experience errors in functionality and or display if run in older or non-standards compliant browsers. This most notably may effect Microsoft Internet Explorer browsers older than version 9; for example, event handlers were written with “addEventListener” rather than “attachEvent”.

Program uses an object literal to represent a user’s account rather than an object constructor. The decision to do this was mostly due to programmer limitations regarding how to effectively and efficiently use aggregate composition, i.e., how to have methods of an aggregate Transaction object reference members of the parent Account object. There was also a desire to avoid using closures within the constructors for objects.

User password has to only meet one requirement: it needs to be at least single alphanumeric in length. This is not very practical.

The program currently only checks for the password cookie at page load and page blur. While the cookie will expire in its given time (currently 10 minutes) the user will not be aware of this unless they reload or leave the page and come back. Programmer is uncertain if this meets common functionality standards.

Program currently makes no use of the “user name” field.

Lastly, formatting of the display for each transaction becomes more askew with larger numbers.

Testing

Much testing and refinement of code occurred while programming the user’s date to be in a proper format (using regular expression) and adhere to known date limitations (leap years and months not having too many days in them). Test dates included years that don’t fit within range (less than 1,000 or greater than 3000), that had the wrong format, that had too many days for a given month. February days per month were tested for 2011 (a non-leap year) and for 2012 (a leap year). Adding to the initial challenge was Javascript’s Date object automatically correcting user input that gave too many days to a given month. For instance, if user put in 2/31/2011, Javascript would change that to 3/3/2011 but this kind of automation, it was felt, would not benefit the user attempting to input exact dates; the user would likely rather know of the input error rather than have it fixed behind the scenes.

Amounts less than zero and greater than 100,000 were correctly handled in the “amount” input field.

Cookie testing involved setting the time to expire to 1 minute and ensuring the password cookie was checked on an attempt to load the main page and on main page losing focus (blur). Cookie passwords were tested and it was discovered no entry into the password box or simply hitting the spacebar satisfied the creation of a password cookie, this was undesirable and the password requirement became a regular expression denoting at least one alphanumeric character be used (“\w+”).

As noted, there was a lot of testing (and failing) during the coding process regarding proper handling of dates and cookies. Eventually the program appears to handle these as it should.

Formatting of transaction display: numbers both positive and negative, big and small were used. A few ternary operators were used to enhance the resistance to the skewing but a more robust solution should be implemented (a function created to handled padding. Research showed a few solutions to tis problem but time limitations meant this enhancement could not be implemented).