Ross Studtman

CIS131 – Web Development II

Second Project: Student Scores

Shane May

Due: 18 October 2012

Problem Scope

Summary:

Obtain and display students’ names and scores (grades). Be able to calculate average, standard deviation, and variance of scores. Have methods for sorting students by last name and for clearing all student records.

Data gathering:

* Obtain each student’s first and last names, and their score.
* Student scores (grades) are in the range of 0.0 to 100.0.

Functionality:

* Button: “Add Student Score” adds gathered data to “Student Scores” display.
* Student Scores display: “last name, first name: score”, one per line.
* Dynamically display: calculated average of scores.
* Dynamically display: calculated standard deviation of scores.
* Dynamically display: calculated variance of scores.
* Button: “Clear Student Scores” clears student names and scores from “Student Scores” display. Note that this will cause the dynamically calculated displays to be empty.
* Button: “Sort By Last Name” sorts students by last name.

Notes:

1. This application should use one or more arrays to store gathered data.
2. Do not assume user will enter valid numerical data.
3. Do not assume user will enter valid numerical range.

Concerns:

Standard deviation and variance are rusty topics for this programmer. Understanding standard deviation and variance and then coding the logic behind the calculations, and then being able to test if the logic is working correctly is of concern.

Pseudo-Code

Gather user input:

* firstName = #id of firstName textbox, get value;
* lastName = #id of lastName textbox, get value;
* score = #id of score textbox, get value

Functionality:

* user clicks “Add Student Score” button
  + ensure score is a number
    - if(score is not a number)
      * prompt user for proper number.
  + ensure score is within range 0.0 to 100.0
    - if(score is less than 0.0 or greater than 100.0)
      * prompt user for proper range.
  + score = parseFloat(score)
  + create student: use firstName, lastName, and score as arguments for Student class. (consider doing error checking in here instead).
  + Store created objects in studentArray
  + Iterate over studentArray, pulling properties from objects, and insert into HTML display.
  + numberStudents = studentsArray.length;
  + calculate average
    - average = Student.getScore() / numberStudents;
  + Calculate variance
    - Loop through array, each iteration do:
      * sqrDiff = (score – mean)^2;
    - variance = sqrDiff / (array.length – 1)
  + calculate standard deviation
    - standardDev = square root of variance.
  + update dynamic display for average scores.
  + Update dynamic display for standard deviation.
  + Update dynamic display for variance.
* User clicks “Clear Student Scores”
  + Assign studentArray to empty.
  + Clear HTML student display.
* User clicks “Sort By Last Name”
  + Clear HTML student display.
  + Sort studentArray by Student.getLastName() property.
  + Repopulate HTML student display



Operation Requirements

Javascript, HTML and CSS were tested on Firefox 16.01, though any modern browser ought to work well. Website and program require a display and input device.

Issues

Event handlers are written with “addEventListener” and should work with most modern browsers, excepting pre-Interned Explorer 9.

Significant digits appear to display too great of precision. This program is being modeled to match an example program where the significant digits for standard deviation and variance were four, though it is unlikely student grades will be entered with such precision.

Currently user can omit entering a first or last name, or any name at all; the only field required is a score. With no names, this program basically becomes a calculator of mean, variance, and standard deviation given a set of input data. If this “functionality” isn’t desired, place constraints on the first and last name fields, i.e., require them.

Testing

Testing data input:

* since there are no data restrictions on user’s first and last names, they didn’t require testing.
* User’s score needs to be A) a number, and B) within the range of 0.0 to 100.0. Program was tested with values that were not numbers and were not within range (both above and below range) and there were no issues.

Testing sorting of last names:

* data entry for user names was purposefully entered out of alphabetical order to test the sorting algorithm. Additionally, two identical last names were used to test if the sort would then sort by user’s first name. The program was able to handle these without incident.

Testing resetting of all data fields:

* all fields are displayed as empty after the “Clear Student Scores” button is pressed. An issue occurred during testing where the student array was not being zeroed out along with the display but this has been resolved.