PROG2001 – WEB DESIGN AND DEVELOPMENT

A-04 : *HI-LO (REVISITED)* – DONE WITH ASP AND SERVER-SIDE LOGIC

OVERVIEW

In this assignment, you (and your partner) will revisit your original Hi-Lo game (A-01) but this time, you will implement the game using the **ASP server side** technology.

This is a partner-based assignment and you can have partner if you wish.

OBJECTIVES

This assignment supports the following course objectives:

- To demonstrate the operation of server side technologies (ASP)
- To demonstrate the ability to create user web forms
- Become familiar with client-side validation and feedback techniques

ACADEMIC INTEGRITY AND LATE PENALTIES

- Please refer to the SET Policies document regarding Academic Integrity Information
- Please refer to the SET Policies document regarding Late Policy

EVALUATION

- Please refer to the assignment weighting in the *Instructional Plan* for the course as well as the assignment's Rubric in the course shell.

PREPARATION

Review Module-03, Module-05 and Module-07 lesson content as well as each module's code samples – they will help you in this assignment. As well, review you're A-01 solution and feedback comments.

REQUIREMENTS

1. The user should have the exact same game experience as in A-01. Please revisit that assignment description for a reminder of what is required and expected …
2. Please call your starting page `hiloStart.html`
    a. As far as the ASP (server-side) functionality goes – you can choose to implement it in a single ASP page or multiple – the choice is yours.
3. Only **input-level validation** is to be done on the client-side using JavaScript – the **logic for the game** play is being moved to the **server-side**
    a. The range-check logic (i.e. user enters guess and the game checks if it is correct, too low or too high) is actually part of the game play and now belongs in the domain validation (on the server-side)
    b. The *grey-area* of validation that could be included on the client-side is the validation that a number has been input (as opposed to a letter, etc.). As discussed previously, this could also be argued that it should be done in the domain on the server … the choice is yours.
        - For the *maxNumber* input, the determination of whether this number is an integer greater than 1 is part of the game-play and belongs on the server
4. In this assignment, it is expected that you use a <form> element as well as other UI components for prompting the user and the gathering for information
    a. In A-01, you persisted (remembered) the user's name, the random number, etc. through the use of ***global variables*** in your client-side scripting
    b. In this assignment being as it is implemented on the server-side, you will need to use a different form of persistence (state management). You can refer to Module-07 for ideas on how to do this.
5. It is also expected by this point in the course that you can use and apply style to your solution through CSS (using internal styles (i.e. defined within the <HEAD> element) or external styles)
6. **The "game engine" logic must be executed on the server side.**
7. Make sure you comment appropriately (HTML header comment, JavaScript function comment blocks as well as inline comments – also don't forget to comment your ASP server-side logic)
8. Make sure that your Hi-Lo application runs properly and consistently within the Internet Explorer v11 and Chrome browsers

FILE NAMING REQUIREMENTS

As mentioned above in the Requirements, your solution can be comprised of any number of ASP pages, but your starting HTML page must be called `hiloStart.html`.

SUBMISSION REQUIREMENTS

When submitting your solution to this assignment, hand-in a single ZIP'd file containing:

1. Your starting HTML page (HTML files)
2. All of your ASP server-side source file(s)
3. Also remember that this solution will be tested using Internet Explorer v11 as well as Chrome
4. Please ZIP up these files and submit to the appropriate eConestoga Dropbox by the deadline
    a. Please give your ZIP submission the filename *lastName-firstInitial.zip* (e.g. if you are Sally Jones – then your ZIP should be named `jones-s.zip`
    b. If you are working with a partner, then include both your names in the ZIP filename (e.g. if Sally Jones is working with John Smith – then your ZIP should be named `jones-s-smith-j.zip`

NOTE: If working with a partner, only one partner need submit the solution

ADDITIONAL INFORMATION

Notes:

- You will already be exposed to everything you need to do this assignment, but you might find "better" ways if you do some research
- Be sure you understand what you are doing - otherwise, stick to the simple solutions that work
- Ensure that if you are using cookies, hidden input types and/or Session variables to hold state information that you clear them as needed if the user wishes to *Play Again*
- Also ensure that if the user selects a value outside of the allowable range, they are told of their error

About JavaScript in ASP source files

Over the years, a number of students have asked me if "...ASP files can contain client-side JavaScript..." The answer is definitely YES! (for example - see my `S02-Form-with-Postback.asp` example in Module-05 samples).

Remember that an ASP file is really just the same the HTML with the exception that it has some server-side scripting embedded within it (i.e. the <% . . . %> section(s)). The goal of an ASP file (like every other server-side framework) is to produce an HTML *response* to the client. And depending on the purpose of the code - it is very likely that the response will contain client-side JavaScript.

But when they asked me this question in conjunction with this assignment, I realized that perhaps they still didn't understand the line that existed between client-side (or input-level validation) and server-side (what this assignment calls *game-logic*).

Here are the highlights of what is client-side input-level validation and what is server-side *game logic*:

- Client-Side (input-level validation)
    - Getting the user's name - your validation here can ensure that cannot be blank. It is mandatory and can contain any characters
    - Entry of the *maxNumber* input - your validation here can ensure that the entry is **numeric** (doesn't contain any letters or spaces). It is also possible at the input-level validation side to ensure that the input is a **numeric integer** (if you want to). But the determination of whether the value in this integer is within the allowable range of game play (i.e. is greater than 1) is part of the rules of the game and therefore this logic belongs on the server-side as part of the game logic
    - Entry of the *guessNumber* input - this follows the same rules as the *maxNumber* above - the client-side validation can ensure that it is numeric and even go as far as ensuring it is a numeric integer. But checking if the value entered is in range of the min/max guessing range (and if it is the actual answer) is part of the rules of the game and belongs on the server-side
- Server-Side (game-logic)
    - The calculation of the final random number (*numberToBeGuessed*) is game-logic and needs to be done on the server-side
    - The determination of whether *maxNumber* is greater than 1 - part of the rules and belongs on the server-side
    - Any feedback from the *guessNumber* is to be done on the server-side
        - guess is out of allowable range
        - guess is in range and therefore changes the min/max values of the allowable range
        - guess matches the *numberToBeGuessed* - and the player WINS

Hopefully this little description clears up any confusion about (1) the presence of JavaScript in an ASP file and (2) what the game-logic is and therefore what needs to be done in VBScript in ASP on the server-side.