CSCI 135

Prof. Mariam Salloum January 23, 2018

Problem Set 2

1 Instructions

Create a folder on your computer named **ps2_lastnamefirstname**. You will include all files required for this assignment in this folder. When you are ready to turn-in your assignment, you will compress (ZIP UP) your folder and submit the ZIP file to the 'Assignments' area on Sakai.

2 Part 1

Red-Blue Tic Tac Toe is based on regular Tic Tac Toe. Here are the rules:

- The game is played on a 3x3 grid. Initially each of the nine squares in the grid is colored white.
- The two players are called red and blue. The players alternate making moves, with red moving first.
- Until there are three squares of each color on the board, red (or blue) selects a white square and marks it with red (or blue).
- After there are three squares of each color on the board, red (or blue) selects a white square and marks it with red (or blue). As soon as this happens, the square that was least recently marked red (or blue) reverts to white.
- The game continues in this fashion, with the players alternating moves, until red (or blue) wins by having the three red (or blue) squares on the same row, column, or diagonal.

You are to implement the Tic Tac Toe game using HTML, CSS, and JavaScript. Your game should be called tictactoe.html, and it should work properly in Chrome. At a minimum, your game should:

- have a game grid consisting of nine squares colored white, red, and blue as appropriate,
- allow the players to make legal moves by clicking on the game grid,
- prevent players from making illegal moves,
- report when a player has won, and
- have a way to start a new game at any point.

Feel free to customize the game beyond this minimum specification to your hearts content.

Hints: One way to represent the game board is as a 3x3 array of images, where each image is either completely white, completely blue, or completely red. With them you can use a JavaScript image swapping function. Alternatively, you can use a table to represent the board and use JavaScript to modify the CSS backgroundcolor element of a $< td >$ tag. This would be simpler and display faster in the browser, not to mention it wouldn't require a single image. You might find document.getElementById().style.background useful for doing it this way. I might suggest that you try to use CSS as much as possible.

You might want to keep the CSS and JavaScript code as external files rather than keeping them as internal (i.e., all in one file) even though those external files will not be shared by any other pages. So, for this problem, you would create at least three files: tictactoe.html, tictactoe.css, and tictactoe.js.

3 Part 2

For part 2, you will design a HTML form to gather information from a user, and utilize JavaScript to validate the form's input. The Micosoft account signup page (`https://signup.live.com/?uaid=5af504a1f26d4658897be771f711c67c&lic=1`) is a good example of form validation - a complicated form that would be unfriendly without validation.

The main objective of this part of the assignment is to use JavaScript and jQuery as much as possible to validate form input. The form should not be submitted if there are any errors in the user input. Of course, since we haven't written the server part of this page, submission means nothing at this time.

3.1 Requirements

The code files for form validation are validate.html, validate.css and validate.js.

You should implement the function validateField in validate.js. The function *validateField* should take in three parameters, *fieldElem*, *infoMessage*, and *validateFn*.

- fieldElem can be a CSS selector, jQuery element, or DOM element that represents a single form text field.
- infoMessage should be a string that gives a human-readable description of the fields requirements.
- validateFn should be a function that validates the fields value.

When the function is invoked, you should insert a ¡span¿ notification element immediately after the form field. The notification element should initially be hidden.

- When the field is currently being edited, the notification elements text should be infoMessage, its class should be info, and it should be visible.
- When the field is not being edited:
  - If the field is empty, the notification element should be hidden.
  - If the field is non-empty and the form field validates, the notification elements text should be OK, its class should be ok, and it should be visible.

  - If the field is non-empty and the form field does not validate, the notification elements text should be Error, its class should be error, and it should be visible.

Additionally, use this function to validate the form fields on the web page. You may use any sensible information messages of your choosing. The validations are as follows:

- The username field must contain only alphabetical or numeric characters.
- The password field should be at least 8 characters long and contain 1 number.
- The phone number should only have numeric characters and of the format xxx xxx xxxx.
- The email address field should be validated using a simple regular expression pattern (to match an email along the lines of [a-z]@[a-z].[com, edu, gov])
- The radio buttons should have an option selected.
- The checkboxes should have at least 2 selections.

You should use the form events (`http://api.jquery.com/category/events/form-events/`) functions to detect when a form field gains or loses focus. You should not use mouse and keyboard events such as click or keydown: there are multiple ways to give a form field focus, and it is difficult to detect all of them reliably using mouse and keyboard events.

The information span element can be added using element creation and DOM insertion (`http://api.jquery.com/jQuery/#jQuery2`).

You can design the form on any topic you like as long as you include items similar what is included in the sample file : validate.html. Alternatively, you can maintain a generic form as provided in the sample.

4 Submission

- Your final submission should be a ZIP file that consists of all your files.
- Additional files, such as, additional HTML/CSS/JS files, image files, etc. should be included in the ZIP file as well.
- It is your responsibility to verify that the ZIP file was uploaded correctly, and contains all your files. Double-check what you upload just in case!
- Be sure to VALIDATE your HTML before submission and ensure your game works properly on Chrome!