**COMP 2511 Fall 2015**

**Assignment 4**

**Programming a Simple Game using JavaScript/jQuery**

**Due: December 4, 2015, before 3:59 p.m.**

***Outcomes***

The intent of this assignment is to develop a simple client-side game, building upon the CSS and HTML from assignment #3.

In particular students will be able to:

* Include and use jQuery and JavaScript.
* Handle JavaScript events via listeners.
* Write functions that take parameters and can be reused.
* Process a POST request using jQuery.
* Interrupt default form submission behaviour.
* Handle JSON objects.
* Change/insert/delete HTML elements using jQuery.
* Build the basic logic for a two-person game.

***Submitting***

Submit all your html, graphic and style files in a single folder or zip archive that includes your username at the start of the folder/file name, for instance nkhemka\_assignment4. When confident it is ready, drag your correctly named submission to the Submit Drive (I:).

Note: If submitting from off campus, use https://secure.mtroyal.ca to access your files.

**You will lose marks if you do not follow these submission instructions.**

***Grading***

Your assignment will be graded based on its gameplay and code quality. Gameplay includes the core gameplay logic, form posting, listener setup, handler authoring, and use of CSS to style the game as the gameplay progresses. Since you have already been awarded grades for layout and design in assignment 3, this assignment will focus on the gameplay itself, with only a small amount of grades allocated for (re)-design and using CSS with JavaScript.

***Requirements:***

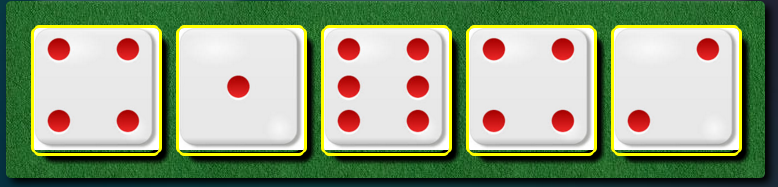
* ***Make and handle an AJAX request for a new game*** 
  + The form to enter new names and request a new game MUST have amethod of POST, and an action of <http://ins.mtroyal.ca/~nkhemka/2511/process.php>
  + Fields should include the namesplayer1, player2.
  + Interrupt the synchronous request and post the form asynchronously using jQuery.
  + Handle the response that will be simply five new dice encoded using JSON.
  + Display the dice returned as ordered, three-dimensional generic image that you created in A3 and markup you require for your dice.
    - You will need loops, parseJSON(), and object access to complete this task.
* ***Gameplay***
* When the page is loaded:
  + The area for player names allows the users to enter their names and start a new game.
  + “Current score area” and “move area” may be blank or may include a message of your choice for the players since the game is not yet started.
  + The dice are **not** displayed at this time.
  + The title and footer are displayed.
  + For example:



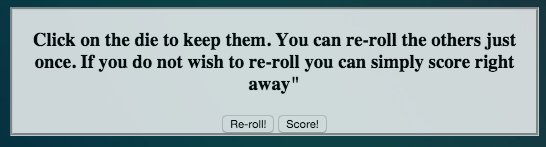
* When a new game starts:
  + The five dice are displayed.
  + The “move area” displays a message for the current player.
  + The “score area” displays 0 points for both the players, along with their names from the “new game” area.
  + The “roll the dice” area is also displayed.
  + For example:



* Gameplay consists of each player completing the following series of steps with the completion of these steps ending a player’s turn.
  + Step 1 (initial roll): The active player clicks on the “roll the dice” button. Player 1 is always the first active player. Pressing this button returns a data string that consists of 5 values, each ranging from 1 to 6. These values represent rolling 5 dice (5 separate results) and the results of those rolls (the value has to range between 1 and 6 inclusive).
    - The following php link: <http://ins.mtroyal.ca/~nkhemka/2511/process.php>   
      simulates this process, returning a different data string each time it is called. You **must** utilize this link for all of the dice rolls within your assignment. Here is an example of what this link returns (you will need to use parseJSON() to parse it): {"Roll":[{"value":4},{"value":1},{"value":6},{"value":4},{"value":2}]} represents having rolled 5 dice and having the dice show a result of 4, 1, 6, 4, 2. The dice faces are displayed in the order the values are returned from the server. The above example string shows that we “rolled” a 4, 1, 6, 4 and 2 and as such we would see the following displayed in your game area matching the values from left to right.



* Note that these values are random every time you “roll” the dice. If you clicked on the above link again you may end up with another string such as:  
  {"Roll":[{"value":6},{"value":1},{"value":1},{"value":2},{"value":4}]} which would represent rolling a 6, two 1s, a 2 and a 4.
  + Step 2 (optional reroll): Upon completing a “roll” action the active player is given the choice of immediately scoring (Step 3) or instead of being able to reroll some, or all, of their dice.



To use the above example of having rolled a result of 4, 1, 6, 4 & 2, the player should be able to

1. immediately score the results
2. reroll all of their dice or
3. to reroll some of the dice.

To continue the example, the player should be able to select both of the “4”s as dice that they want to keep and then when they press the “re-roll” button only the dice displaying 1, 6 and 2 would be rerolled. Both the 4 values must be retained.   
  
You must have some graphical means of displaying which dice are selected to be “kept” (i.e. which dice are not going to be rerolled). This can be an arrow, different shading, a different angle, or whatever you come up with! Let your imagination run wild. Each player is only given a single reroll on any given turn. For example, in the figure below, face values 5, 6 and 5 (the last three) are being kept and both the 3’s will be re-rolled.



* + Step 3 (scoring): After the active player has taken their re-roll, or chosen to forgo it and proceed directly to scoring, the resulting dice are scored according to some popular poker hands. The active player should be able to select which of the scoring results they wish to use. The scoring can be in its separate area, or a drop down box, or any how you prefer.
    - Two of a kind: Two dice are face up with the same values (i.e., you have two 2’s or two 3’s at the same time). Worth 20 points.
    - Three of a kind: Three dice are face up with the same values (i.e., you have three 1’s or three 4’s face up at the same time). Worth 30 points.
    - Four of a kind: Four dice are face up with the same values (i.e., you have four 1’s). Worth 40 points.
    - Five of a kind: Five dice are face up with the same values (i.e., you have five 1’s). Worth 50 points.
    - Full house: The active player has both a Two of a Kind and a Three of a Kind face up at the same time (i.e. two 5s and three 6s at the same time). Worth 75 points.
  + The result of scoring is recorded and added to the active player’s score after which play proceeds to the next player.
  + Please note that you are NOT expected to actually program in any fact checking. I do not expect you to ensure that a player actually has three of a kind when they choose three of a kind as their scoring option as that is beyond the scope of this assignment (although it might make for a compelling argument for bonus marks). However, you are expected to have the “Sum of all dice” option accurately add the correct amount to the active player’s score.
  + Step 4 (final scoring): After both players have completed steps 1-3, scores are compared and the player with most points wins the game. The winning player must be congratulated by name and a button must appear prompting to play again. In the event of a tie, both players are congratulated. Upon resetting the game, the display must return to a neutral setting with the previous scores/values being cleared out.
* ***HINTS***
* Consider adding some global variables to your program to handle state, such as scores.
* Selectors are your friends. You can set up many listeners at once using a selector that matches multiple items. Once triggered, the method being executed can reference the particular object that triggered the event by using $(this).
* Delegated event handlers (<https://learn.jquery.com/events/event-delegation>) are A Good Thing…a best practice that you should be comfortable using.
* You can animate using CSS# or using jQuery animate functions. Add to the quality of the game with thoughtful animations that illustrate concepts and add to the overall usability.