COMP 2511 Winter 2017

Assignment 4

Programming a Simple Game using JavaScript/jQuery

Due: April 12, 2017, before 11:59 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 a method of `POST`, and

an action of `http://ins.mtroyal.ca/~nkhemka/test/process.php`

- Fields should include the names `player1, player2.`
- Interrupt the synchronous request and post the form asynchronously using jQuery.
- Handle the response that will be ten new card values encoded using JSON along with the first value which is the sum to beat.
- Display the cards as returned, using the images that you created in A3 and markup you require for your cards.
    - You will need loops and `parseJSON()` to complete this task.

- ***Gameplay***

1) 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 all the other areas may be blank or may include a message of your choice for the players since the game is not yet started.
    - The cards are **not** displayed at this time.
    - The title and footer are displayed.
    - For example:

2) The players enter the names and click on the “new game” button.
`<form action="http://ins.mtroyal.ca/~nkhemka/process.php" method="POST">`

   When a new game starts:

   - The sum to beat is displayed. The ten cards are displayed. The sum to beat is the first value that is received from the JSON script. The next ten values are the ten cards to be displayed. The order of these cards are received from the JSON script.
   - The “move area” displays a message for the current player. It also displays the “Sum to Beat” field.
   - The “score area” displays 0 points for both the players, along with their names from the “new game” area.
   - The “end the turn” and “end the game” buttons area is also displayed.
   - For example:

3) Gameplay consists of each player completing the following series of steps with the completion of these steps ending a player's turn.

    - Player 1 is always the first active player.

    - When the game is loaded or at the start of a players turn, ten cards are displayed to the players. The following php link:
      http://ins.mtroyal.ca/~nkhemka/test/process.php
      simulates this process, returning a different data string each time it is called. You **must** utilize this link for all of the player turns within your assignment. Here is an example of what this link returns (you will need to use parseJSON() to parse it and this corresponds to the above image): {"Cards":[{"value":8},{"value":3},{"value":4},{"value":4},{"value":4},{"value":3},{"value":4},{"value":0},{"value":0},{"value":0},{"value":2}]}
      The first value in this string is the "Sum to Beat" (in our example: 8) and the remaining values represents the order of the cards. The cards are displayed in the order the values are returned from the server. The above example string shows that the order is 3, 4, 4, 4, 3, 4, 0, 0, 0, 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 the “end the turn” button is clicked. If you clicked on the above link again you may end up with another string such as: {"Roll":[{"value":7},{"value":0},{"value":2},{"value":1},{"value":1},{"value":3},{"value":3},{"value":2},{"value":0},{"value":4},{"value":4}]} which would represent cards in the order of 0, 2, 1, 1, 3, 3, 2, 0, 4, 4 and the sum to beat value of 7.

- After the cards are displayed to the active player, the player has the ability to select as many cards (0 or more) as they like in order to exceed the “sum to beat” value. For example:

The “sum to beat” value is 8. In this case, the active player selects 4 + 4 + 0 = 8. The

player

You must have some graphical means of displaying which cards are selected to be added towards the “sum to beat” value. This can be an arrow, different shading, a different angle, or whatever you come up with! Let your imagination run wild.

- Scoring: After the active player has selected all the cards they wish to add towards their “sum to beat”, the active player can either “End the turn” or “End the Game”.

  “End the turn”: When the active player hits “End the Turn” the value of each of the selected cards are added. If the total value of the cards selected is greater than or equal to the “Sum to Beat”, a message is displayed to the player saying that “You are right, you score 10 points, and it is now Namrata’s turn”.

  On the other hand, if the total sum of the cards is less than the “sum to beat” the player gets no points and the control is passed to the next player.

- The game ends when either of the players choses to end the game. The 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 may choose to make modifications to your HTML and CSS files.