SER421 Fall 2015 Lab3: HTML5 and DOM Events  
Assigned 9/25/15, due 10/6/22/15 at 11:59pm via online submission to Blackboard

**Objectives:**

1. Continued competency building in proficiency of constructing self-contained (single-page) HTML5/Javascript applications (SPA) using HTML DOM manipulation.
2. Gain competency in managing application state.
3. Gain familiarity with new HTML5 features
4. Understand the impact of DOM loading events

***This lab may be completed and submitted individually or in pairs. There will be a Lab 2 solution made available, and you may use this code as a starting point if you do not want to use your own Lab 2 solution.***

**Overview:**

More fun with Clue! Last lab we created a SPA textual version of the Clue game. This lab, let’s enhance the same game with some visual features, and take advantage of advanced HTML Javascript capabilities to build a more robust game.

*One important functional difference from last lab. Instead of the human user and the computer holding cards in her/his hand, all suspects, weapons, and rooms will be available visually as specified in Task 1. Gameplay will merely be the user and the computer taking turns making guesses until a player wins.*

**Task 1: Create a visual version of the Clue game**

For this task you will port the Clue game, specifically Functional Requirements #2 and #4 thru #6, to a visual version using and HTML5 canvas and/or Drag and Drop. Requirement #3 is superseded by the statement in italics above. For this version, instead of using text and UI widgets to display the state of the game and accept user interaction, you will use HTML5 features and images to achieve gameplay. Specifically:

1. Req #1 from Lab2 becomes visual:
   1. A game board should be displayed on the screen. A game board image is given to you. It should be 1024x1024 pixels. This is new, but I have given you an image.
   2. ALL Suspects should be displayed visually at the TOP of the board.
   3. ALL Weapons should be displayed visually to the BOTTOM of the board.
2. The functional requirement #2 from Lab2 stays the same, as does the logic of #5 (however the UI elements change as described next). These DOM elements should be rendered below the game board area.
3. The Lab2 requirements #4 should now be accomplished by dragging and dropping one Suspect and one Weapon into one Room.
   1. The human makes a guess by drag and drop, then clicking a submit button which should be able to determine what guess was made. You should textually indicate the guess in the bottom area.
   2. The computer makes its own guess, but this guess should be shown visually on the board game.
   3. Your UI logic should detect if the suspect and weapon and not in the same room, inform the end user this is an invalid guess, and restart the guess process.
   4. Once a guess is made, a message at the bottom should dynamically indicate whether the guess is correct or not, and if not displays one of the ways it is wrong. Note this logic will have to change from your Lab 2.

One very important constraint has been lifted – you will use the classic Clue game pieces, meaning you do in fact know who the suspects, weapons, and rooms are. I have given you all the media on Blackboard. So, there will be no global arrays at the top of the Javascript as with Lab 2.

At the end of the lab specification is a mock-up of what the board may look like. The trick here is to decide if you are going to implement Drag and Drop using DOM manipulation as shown in the 2 examples from Chapter 6 of the Modern Web book examples (see your javascript3 examples), or if you are going to use an HTML5 Canvas with Drag and Drop. This is a little different as you have to do some of the geometry work yourself. Yes I am fine with you looking up examples of how the canvas way is done, but always attribute someone else’s code. And NO, you cannot use a 3rd party library like Kinetic for Canvas-based Drag and Drop.

**Task 2: Save state using browser-based storage**

This task asks you to port functional requirements #7 and #8 from Lab 2 to browser-based storage. That is, these requirements read like this in Lab 2:

1. Provide a button (outside the form) named “Show History” that dynamically shows the guesses you and the Computer have made so far in the current game. When shown, the “Show History” button should become “Hide History” (you are essentially creating a toggle button).
2. Provide a button named “Show Record” that when pressed, dynamically displays the won-loss record for the Computer, and a history of who the Computer played, the date, and the outcome.

Now they should read like this:

1. Provide a button named “Show History” that dynamically shows the guesses you and the Computer have made so far in the current game. When shown, the “Show History” button should become “Hide History” (you are essentially creating a toggle button).
2. Provide a button named “Show Record” that when pressed, dynamically displays the won-loss record for the Computer, and a history of who the Computer played, the date, and the outcome.

The constraint is that you have to use Session or Local storage and not IndexedDB, Cookies, or SQLite for these requirements.

Second, you should modify functional requirement #2 from Lab 2 (the sequence where you ask for the user’s name) and instead remember the user’s name if s/he has played it before in this browser (even if the browser was closed between plays). You may NOT implement this using Session or Local storage. Functionally, if the user has played the game before you go straight to the welcome message. However, you should also provide the capability to switch user names.

**Task 3: Detect and control page loading based on events**

Now that you have the game set, let’s assume your game will load image resources for the weapons and suspects from a remote server. Assume the base URL is <http://dcs.asu.edu/faculty/KevinGary/ser421/clue/>; so for example, to get the image for Mrs. White, you would use URL <http://dcs.asu.edu/faculty/KevinGary/ser421/clue/MrsWhite.jpg>.

Modify your code to ensure that drag-and-drop behavior cannot happen until all of the visual resources for the game have completely loaded and rendered. Other behavior that happens before gameplay, such as requirement #2, should happen at the earliest point possible during game load. You should assume that I will introduce artificial delays in the loading of remote images into your code for testing (see the tricks from the javascript3 sample code) .

**Overall Assumptions**

1. Again, you will use only the classic Clue pieces as represented in the media I have given you, not 3 global arrays.
2. NFRs 1 and 2 from Lab2 remain, that is:
   1. Do not load a new file using something like window.location.href assignment. The application is to be a SPA.
   2. Do not use alerts. “Dynamic display” in the functional requirements means you are to dynamically manipulate the DOM to substitute new content as needed.
3. NFRs #3 and #4 from Lab 2 no longer apply, they have been superceded above.

Grading hints – I suggest you decompose the functional requirements and complete in increments. I will award partial credit, so you are welcome to submit 2 files, one representing your latest working version and one your latest non-working version. I view requirements 1-4 as straightforward, 5 and 6 a little more complex, and 7 and 8 as related.

**Submission:**

You are to implement your solution **by yourself or with a single partner**! If you partner please name your zipfile submission <asurite1>\_<asurite2>\_lab3.zip, if it is just you use <azurite>\_lab3.zip. The zipfile you submit should be self-contained; that is we should be able to unzip it in an empty directoy, and load a file name lab3.html into our browser and have it work. As before you may submit unlimited number of times, so I suggest getting the application to work one task (one requirement) at a time, submit, and then go back and complete more functionality. We will always grade your last submitted attempt. You may add a README.txt in the zipfile if there is anything you want us to take into consideration. There will be no extensions for this lab!

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Macintosh HD:Users:kevinagary:work:asu:classes:ser421:fall2015:labs:lab3:ColMustard.jpg | Macintosh HD:Users:kevinagary:work:asu:classes:ser421:fall2015:labs:lab3:ProfPlum.jpg | Macintosh HD:Users:kevinagary:work:asu:classes:ser421:fall2015:labs:lab3:MrGreen.jpg | Macintosh HD:Users:kevinagary:work:asu:classes:ser421:fall2015:labs:lab3:MissScarlet.jpg | Macintosh HD:Users:kevinagary:work:asu:classes:ser421:fall2015:labs:lab3:MrsPeacock.jpg | Macintosh HD:Users:kevinagary:work:asu:classes:ser421:fall2015:labs:lab3:MrsWhite.jpg |
| Macintosh HD:Users:kevinagary:work:asu:classes:ser421:fall2015:labs:lab3:board.png | | | | | |
|
|
|
|
|
|  | Macintosh HD:Users:kevinagary:work:asu:classes:ser421:fall2015:labs:lab3:leadpipe.jpg | Macintosh HD:Users:kevinagary:work:asu:classes:ser421:fall2015:labs:lab3:revolver.jpg | Macintosh HD:Users:kevinagary:work:asu:classes:ser421:fall2015:labs:lab3:rope.jpg | Macintosh HD:Users:kevinagary:work:asu:classes:ser421:fall2015:labs:lab3:candlestick.jpg | Macintosh HD:Users:kevinagary:work:asu:classes:ser421:fall2015:labs:lab3:wrench.jpg |

THE REST OF THE DOM-RELATED TEXT APPEARS UNDERNEATH THE GAMEPLAY BOARD AREA ABOVE