INFO I211 - Information Infrastructure II

Spring 2014

Assignment 3

A *battleship* game in Javascript

Total: 100 points (to be completed in your I211 Student Team)

Due Date:

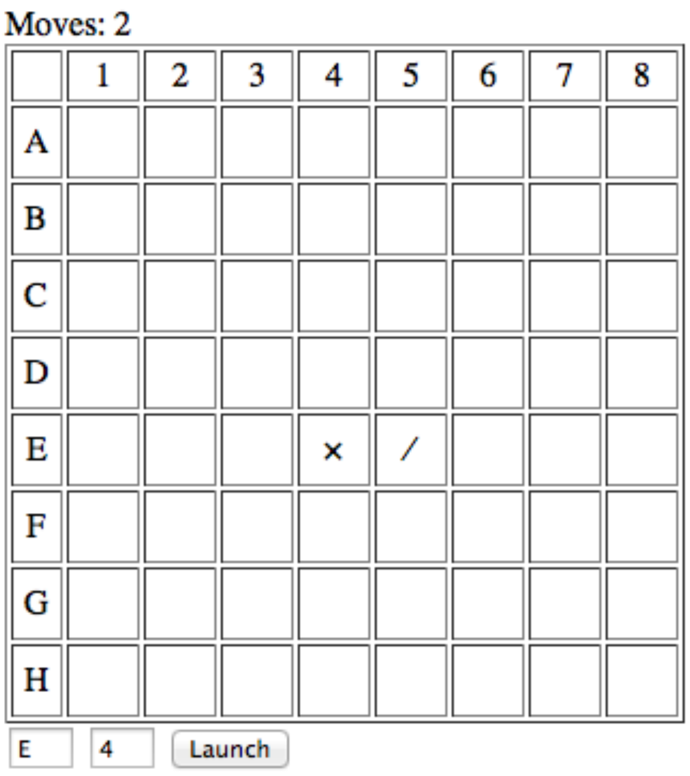
Sunday, April 27 2014, 11:59PM, on IU OnCourse. Free extension until Tuesday, April 29 2014, 11:59PM. If you need more time, please contact I211 Instructors by sending an Oncourse message with CC to recipients' email addresses.

How to work on your Assignment 3:

The Assignment 3 is to be completed in your I211 Student Team, where each team member is to pick *one* of three/four tasks to implement the "*battleship game*", by adapting the Assignment 2 code. The tasks are:

1. Task 1 = implement the "*opponent*" in the battleship game, as one single 1-square "*boat*" (if there was a team member who had the Task 2 *-to implement the* "*chaser*" *character-* in I211 Assignment 2, that person is assigned Task 1 for I211 Assignment 3).
2. Task 2 = implement the "*player*" in the battleship game.
3. Task 3 = implement the playing field for the battleship game.
4. Task 4 = if there are four (4) students in your I211 Student Team, the task for the fourth team member is to provide the implementation of a second 1-square "*boat*" for the "*opponent*", as from Task 1.

Assignment Tasks:


Moves: 2

	1	2	3	4	5	6	7	8
A								
B								
C								
D								
E				×	/			
F								
G								
H								

The goal for Assignment 3 is to complete the "*battleship*" game, by adapting the Assignment 2 code. Please start by adapting your solution for I211 Assignment 2 when you work on completing I211 Assignment 3.

Task 1

The first task is to complete the game logic for implementing the "*opponent*" in the "*battleship*" game.

1. The "*opponent*" is *not* visible on the playing field. For this version of the "*battleship*" game, you only have to implement one (1) "*boat*" of *size 1*, i.e. occupying one single square on the "*battleship*" game playing field.
2. At the beginning of the game, the "*boat*" has to be positioned at a position at random, chosen between rows *A* to *H*, and columns *1* to *8*. *(hint: a floating point value in the range [0.0...1.0] --as obtained from the* `Math.random()` *function-- can be converted to an integer value with the* `Math.round()` *function.)*
3. The *(x,y)* position of the "*boat*" on the playing field can be any coordinate pair, with *x* in the range *[1 ... W-1]* where *W* is the number of *columns* in the table, and with *y* in the range *[1 ... H-1]* where *H* is the number of *rows* in the table. Store the "*boat*" *(x,y)* position in two separate integer Javascript variables.
4. The *(x,y)* position of the "*boat*" character on the playing field also to be checked against the input provided by the user as from Task 2. If the two positions coincide, the "boat" has to be marked off as being hit: store the status of of the "*boat*" in one separate integer Javascript variable, which needs to keeps count of the number of squares occupied by the "*boat*" that are still intact. Since this "*boat*" only consists of one square on the playing field, this variable's value will be either *1* ("*boat*" still intact) or *0* ("*boat*" sunk).

Task 2

The second task is to complete the input controls and game logic for implement the "*player*" in the "*battleship*" game.

1. The positions in the game where the "*player*" has decided to "*launch*" their shots are represented on the playing field:
   - either by the "*multiplication sign*" letter symbol corresponding to the "*&times;*" HTML character entity, for any "*hits*",
   - or by the "*fraction sign*" letter symbol corresponding to the "*&frasl;*" HTML character entity, for any "*misses*".
2. At every move in the game, the "*player*" has to specify where to "*launch*" one shot, to be implemented in your Javascript+HTML. (*hint: input "launch" coordinates can be implemented with HTML* `input type="text"` *tags. HTML* `input type="text"` *tags and their processing have been presented in Lecture 16 and Lecture 17)*. Provide a 1-character text input with values ranging from *A* to *H* for rows, and a 1-character text input with values ranging from *1* to *8* for columns. The "*Launch*" button has to be implemented in your Javascript+HTML to confirm the entered *row* and *column* values.
3. At every move, update a Javascript variable keeping track of the total number of "*launches*".
4. The *(x,y)* position of each "*launch*" can be any coordinate pair, with *x* in the range *[1 ... W-1]* where *W* is the number of *columns* in the table, and with *y* in the range *[1 ... H-1]* where *H* is the number of *rows* in the table. Store the "*launch*" *(x,y)* position in two separate integer Javascript variables.
5. The *(x,y)* position of the "*launch*" at each move also needs to be used to memorize all the "*launches*" in the Javascript 2D array representing the playing field, at its corresponding *column* and *row* in the array.

Task 3

The third task is to complete the game logic *model* and the *view*.

1. The game playing field *view* is to be implemented as an HTML table with H (height) rows, and W (width) columns in every row. Please note that *row 0* and *column 0* in this game are reserved for labels for rows *A* to *H*, and columns *1* to *8*.

2. The game playing field *model* is to be implemented in a 2D array, i.e. in Javascript as an array of arrays: the main array needs to contain as many items as the HTML table W (width) size (except for the labels), and each of these items needs to be an array containing as many items as the HTML H (height) size (except for the labels). To implement this array of arrays, you may use the MyArray2D() function as from lecture 22 notes.
3. Initialize the 2D array to contain one single space " " character in each array item at index *(x,y)*, with *x* in the range *[0 ... W-1]* where *W* is the number of *columns* in the HTML, and with *y* in the range *[0 ... H-1]* where *H* is the number of *rows* in the table. Then obtain the "*boat*" *(x,y)* position from the two separate integer Javascript variables as implemented in Task 1, and keep track of the "*launch*" *(x,y)* position from the two separate integer Javascript variables as implemented in Task 2. Use these position pairs to write one "*B*" letter in the corresponding array item where the "*boat*" coordinates are (as from Task 1), and one "*X*" letter in the array item corresponding to where the "*launch*" coordinates are at each move in the game.
4. Write, into each cell of the HTML table, the character representing the "*hits*" and "*misses*" as described in Task 2 point (1), as stored in the corresponding Javascript 2D array items (i.e. for each *i,j,* write into the (j+1)th cell in the (i+1)th row of the HTML table).
5. At every move in the game, the 2D array needs to be updated thus:
    a. refresh the content of the entire HTML table as from point (4) above;
    b. if the "*boat*" *(x,y)* position and the "*launch*" *(x,y)* position coincide, display a Javascript alert "You won in *M* moves!", where *M* is the number of clicks on the input button, as from Task 2.

Task 4

If there are four (4) students in your I211 Student Team, the task for the fourth team member is to provide the implementation of a second 1-square "*boat*" for the "*opponent*", as from Task 1.

Assignment Submission:

When submitting the application, only one team member (of your choosing) is to submit all the code for the application, as well as a (common) team *README* file. All team members need to turn in their own (distinct) individual *README* file.

1. Include a team *README* plain-text file named `readme-a3-i211-`*`yourteamnumber`*`.text` in which you explain:
    - the parts of the assignment *your team* has completed
    - any extra functionality added to the code
    - any suggestions/enhancements you may have to improve the game
2. Include an individual *README* plain-text file named `readme-a3-i211-`*`yourusername`*`.text` in which you explain:
    - the parts of the assignment *you* have completed
    - the parts of the assignment *your team* has completed
    - any particularly clever code you added
    - (optional) any question you may have about the tasks or the provided template code
3. On the i211 / Spring 2014 site, turn in all your files by 11:59PM on Sunday, April 27 2014, on IU OnCourse.
   Free extension until Tuesday, April 29 2014, 11:59PM. If you need more time, please contact I211 Instructors by sending an Oncourse message with CC to recipients' email addresses.
4. P.S.: *yourusername* should in fact be your username, not the word *yourusername*... and *yourteamnumber* should in fact be your team number, not the word *yourteamnumber*...

Include the following information as a `<!--comment-->` at the top of every HTML file you turn in:

```
<!-- your name (First, Last) -->
<!-- your IU email address -->
<!-- your I211 team number -->
<!-- the names of all your I211 team members -->
```

Each team member also needs to turn in the "I211 Student Team Feedback Form.doc" (which you can find in the *Oncourse Resources->Assignments* folder) about your experience in the student team, and turn it in on Oncourse with your Assignment 3.

Good luck!