# Assignment 3--Model and view for game grid and pieces

**Due:** 11:59pm February 16, 2015

In Assignment 2, you wrote JavaScript code to generate HTML for a table to use as the grid for your game. In this assignment, you will make a model that keeps track of the state of your game and then write JavaScript code that displays model information in a view, which is the HTML table.

We'll be using a simple version of the Model-View-Controller (MVC) design pattern, which is an important and commonly used software design pattern. The controller for your project will be the JavaScript code that gets data from the model (JavaScript arrays and objects, with some JavaScript code) and displays it in the view (an HTML table). In this assignment page I'll talk about models and views but won't say anything more about the controller.

## Models and views

Chess can be played with any number of different chess sets, or even no set at all (if moves are recorded on paper, as in a play-by-mail game), but every game makes use of a similar **model**. The model for a chess game includes information about the location of pieces, which pieces have been captured, and whose turn it is. It also includes information about how the pieces can move and capture. The chess set (board and pieces) are the **view** of the game. The same kind of model is used for games played on a small pocket set with a magnetic board and pieces, or a large board where people act as the pieces.

The same idea applies to other games. For instance, in Battleship, the state of the game (or model) includes what shots have been taken, which shots were missed, and which shots were hits. The model also includes information about ships: size, location, and orientation, and damage. The model does **not** include any information about how the game is displayed. That means that the same model could be used with any kind of view, ranging from a simple text display on a phone to a big-screen 3D display with surround sound.

By separating the model and the view, we can make the model simpler and easier to understand. We can also make it possible to use the same model with different kinds of views.

## Make a model for your game using JavaScript objects and arrays

The first thing to do for this assignment is to decide what data and functions will be part of the model. Your preliminary software design from Assignment 1 should be helpful in doing that, although you don't need to implement all of the functions for this assignment.

Most grid games will include a two-dimensional JavaScript array in the model. For example, in Minesweeper each cell in the 2D array could indicate whether that square in the grid has a mine, and if not, how many mines are in adjacent squares. It can also include information about whether the square has been revealed or whether the user has flagged it.

Many kinds of grid games can also make good use of JavaScript objects in the model. For Battleship, each ship in the game can be represented by a JavaScript object that tells the size, location, and orientation of the ship, as well as how many hits it has taken.

Write JavaScript code that creates and initializes the data in your model. You should use alerts, output to

debugging divs, and browser debugging tools like Firebug to make sure that your model is properly created.

Your model should not contain any HTML, CSS, or images. Those things should be in the view part of your program. It's helpful to make a JavaScript file just for the model so that it's easy to make sure that you haven't put things in the model that should be in the view or controller. (For this assignment, don't worry about separating the controller from the view.)

For some games (like Tic-Tac-Toe or Connect 4) the initial state is not very interesting. If you are working on that kind of game, you should set up a model state that represents a game in progress so that you have some game pieces to display in the grid. Setting up various test cases takes time, but it can save a lot of debugging time later on.

## Display game objects in your HTML grid

After you have written the code for your model, write code that will display game objects (like chess pieces) in the HTML grid. The code that displays game objects should be executed after the code that generates the HTML grid.

In many cases, the code that displays game objects will have nested loops that go through each cell of a 2D array in the model and use the model cell's contents to add the appropriate content to a cell in the HTML table.

If you have two loop counters that are used in nested loops, it's easy to use those counters to index into a 2D JavaScript array in the model. It's possible to use those same counters to access cells in an HTML array.

The first thing you need to do is to use getElementById to get a pointer to the HTML table element:

```
var gridTable = document.getElementById("myTableId");
```

Next, use the loop counters as the row and column to index into the elements of the HTML table and get a pointer to the table cell (td element) like this:

```
var cell = gridTable.rows[row].cells[col];
```

The previous line of code assumes that `row` is the counter for the outer loop and `col` is the counter for the inner loop.

Once you have a pointer the HTML table cell element, you can use the innerHTML property to set the contents of the cell appropriately.

**Important:** I need to be able to look at your HTML table and see that it correctly displays the model. For that reason you might need to display something other than the initial state of the game. In Concentration or Minesweeper, for example, the initial display will be a grid where all squares of the grid look the same. For games like those you should display an intermediate state or the final state of the game so that I can see that information from the model is correctly displayed.

## Update your software design

Update your software design from Assignment 1 to include the new information about your model that you produced as part of this assignment. Your updated design should include descriptions of the arrays and objects used in the model and examples.

Use literals for examples of arrays and objects. Here are some examples of objects from my preliminary software design example for the card game Hearts:

```
{"rank": 2, "suit": "CLUBS"}
{"rank": "QUEEN", "suit": "SPADES"}
{"rank": "ACE", "suit": "HEARTS"}
```

(In my preliminary software design I have variables named CLUBS, QUEEN, ACE, etc., but here they are shown as string values for simplicity.

Your design should be displayed on the project home page (index.html) or in an HTML document that has a link on the project home page.

## Turn in

Put your JavaScript file, along with your game desription HTML file, grid HTML file, CSS file, and image file(s) in a zip file and turn the zip file in as specified on the main Assignment page.

### Points

| | |
|---|---|
| 40 | Creates and initializes a model that uses JavaScript arrays and objects |
| 40 | Displays information from the model in the HTML table. The display shows game objects placed in the grid. |
| 20 | Updated software design describes models and includes sample arrays and objects |
| 100 | TOTAL |