MODEL VIEW CONTROLLER
IN JAVASCRIPT APPLICATIONS

Dr. Brian Durney
Model
Includes the rules of the game and the state of the game

Controller
Connects the model and the view

View
Displays the state of the game to players
Model

Includes the rules of the game and the state of the game

Controller

Connects the model and the view

View

Displays the state of the game to players

REUSABLE!
ADVANTAGES OF MVC

• Reusability: Use the same model with different views and use views with different models.

• Encourages well-defined interfaces.

• Localized changes: A change to the view doesn’t require a change to the model, and vice versa.

• Simplicity: Each part is simpler and easier to understand, implement and test.
FEN includes all of the state information for a game. It’s enough information for a chess player to decide on the next move or who’s winning.

This model could be used with any number of different views...
or any number of other chess sets or images of chess sets, including animated renderings of 3D models.
NO INFORMATION ABOUT THE VIEW

Model

NO HTML
NO CSS
NO IMAGES
If the model doesn’t have any information about the view, it’s much easier to use the model with different views.

It also localizes changes so that it’s often possible to make changes to the model without changing the view, and vice versa.
EXAMPLE: **TIC-TAC-TOE**

There are many ways we could represent a Tic-Tac-Toe board.

Two-Dimensional Array

\[
\begin{array}{ccc}
[0,0,0], [0,0,0], [0,0,0] \\
\end{array}
\]

Let’s start by looking at what happens when we don’t separate the model and view.
X’s turn

{players: 2,
 rows: 3, cols: 3,
 board:
  [[0,0,0],[1,0,0],[0,2,0]]

board[1][2] = 1
X’s turn

{players: 2, rows: 3, cols: 3, board: [[0, 0, 0], [1, 0, 1], [0, 2, 0]]}

board[1][2] = 1
NEW BOARD
One-dimensional array

{players: 2,
 rows: 3, cols: 3,
board: [0,0,0,1,0,0,0,2,0]}

X’s turn

board[1][2] = 1

Tic-Tac-Toe

X

O

X's turn

New Game
NEW BOARD
One-dimensional array

Tic-Tac-Toe

X

O

X’s turn

{players: 2,
 rows: 3, cols: 3,
 board:
 [0,0,0,1,0,0,0,2,0]}

board[1][2] = 1
Change is not localized

{players: 2,
 rows: 3, cols: 3,
 board: 
 [[0,0,0,1,0,0,0,2,0]]}
DEFINING A MODEL

Model

{players: 2, 
rows: 3, cols: 3, 
board: 
[[[0,0,0],[1,0,0],[0,2,0]]]}


By separating the model and the view, we will be free to change to a different board representation later.
MAKING THE MODEL A BLACK BOX

The only way to access the model is by calling methods.

If we change the board representation, only methods in the model need to change.

Game(rows, cols)

{players: 2, rows: 3, cols: 3, board: [[0,0,0],[1,0,0],[0,2,0]]}

markSquare(row, col, pn)

getSquare(row, col)

isGameOver()
Game(rows, cols)

getPlayerInfo()

isGameOver()

markSquare(row, col, pn)

generateBoard()
markSquare(row, col, pn)

getSquare(row, col)

isGameOver()

{players: 2, rows: 3, cols: 3, board: [[0,0,0],[1,0,0],[0,2,0]]}

Game(rows, cols)
Game(rows, cols)

isGameOver()

getSquare(row, col)

markSquare(row, col, pn)
Game(rows, cols)

getSquare(row, col)

isGameOver()

markSquare(row, col, pn)

click(1,2)
Game isGameOver()
getSquare(row, col)
markSquare(row, col, pn)

{players: 2,
 rows: 3, cols: 3,
board: 
[[0,0,0],[1,0,0],[0,2,0]]}

X's turn

X's turn

X’s turn

New Game
model:
- Game(rows, cols)
- isGameOver()
- getSquare(row, col)
- markSquare(row, col, pn)

controller:
- click(1,2)

view:
- {players: 2, rows: 3, cols: 3, board: [[0,0,0],[1,0,0],[0,2,0]]}
Game(rows, cols)

getPlayerInfo()

markSquare(row, col, pn)

generateBoard(players)

isGameOver()

click(1,2)

getSquare(row, col)

{players: 2, rows: 3, cols: 3, board: [[0,0,0],[1,0,0],[0,2,0]]}
NEW BOARD:
One-dimensional array

markSquare(row, col, pn)

getSquare(row, col)

isGameOver()

{players: 2, rows: 3, cols: 3, board: [0,0,0,1,0,0,0,2,0]}

Game(rows, cols)
NEW BOARD:
One-dimensional array

markSquare(row, col, pn)
getSquare(row, col)
isGameOver()

{players: 2,
 rows: 3, cols: 3,
board:
[0,0,0,1,0,0,0,2,0]}

Game(rows, cols)
Game(rows, cols)

markSquare(row, col, pn)

getSquare(row, col)

isGameOver()
```plaintext
Model

isGameOver()

getSquare(row, col)

markSquare(row, col, pn)

Controller

View

NEW VIEW:
Small phone screen

{players: 2,
 rows: 3, cols: 3,
 board:
 [[0,0,0],[1,0,0],[0,2,0]]
}

NO CHANGE

Game(rows, cols)
```