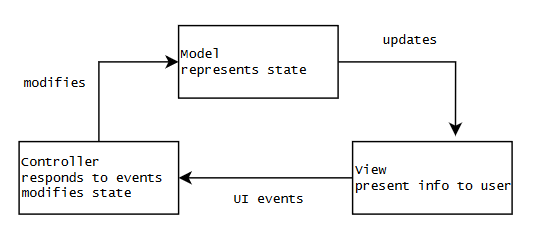
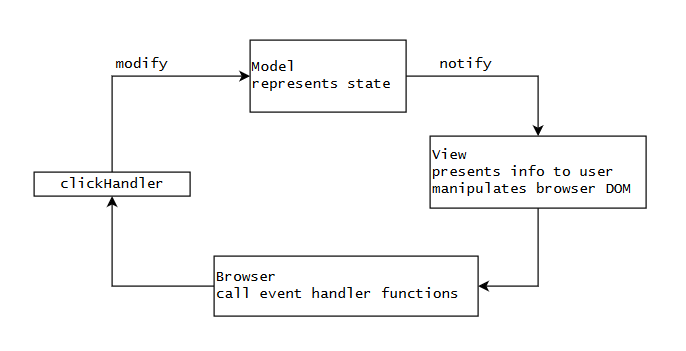
The Art Dealer Game Abstract Design

We start with the basic Model-View-Controller architecture.



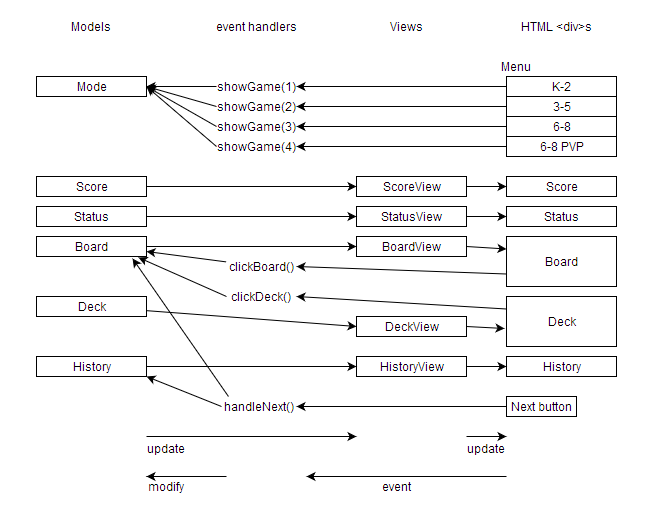
But due to the nature of the web app environment, some of this functionality is provided by the browser. So we dispense with the Controller as an encapsulated entity and our architecture builds upon the browser. Event handlers are just plain functions.



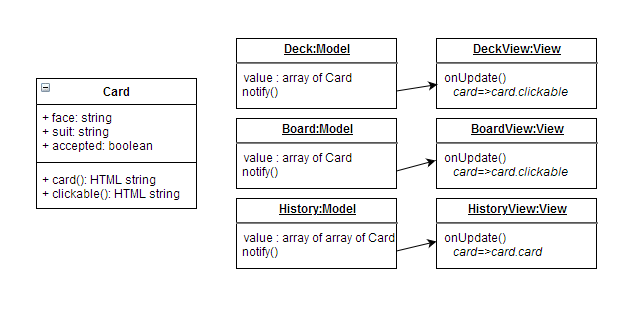
A Model communicates directly with its associated View using the Observer pattern, sending notifications whenever its state changes. The View:onUpdate(value) function receives the new state and transforms it into HTML objects and adds them to the Browser’s DOM. Each of these HTML objects may contain a reference to a clickHandler which the Browser will call in response to a click event. The click handler function modifies the Model’s state, possibly via subfunctions.

Thus the system is a hybrid of Object-Oriented and Procedural code. The OO objects encapsulate the “*nouns*” of the program, whereas a simple hierarchy of functions provide the “*verbs*”.

The source files are divided into .html which describes the visual structure of the app, with <div> elements defining the DOM portion for each View; .css which provides styling for the html elements, and .js which defines the Models; Views, Card class, click handler functions, and other functions.



The Deck, Board, and History models have as values arrays of Cards. The Card class has members describing the card’s *face* and *suit* and whether this card has been *accepted* by the Dealer. It also has pseudo-members (member functions pretending to be simple value members) which convert the card into an HTML entity, with or without wrapping the entity in a <span> with a click handler.



When the game starts, the Dealer is randomly selected from the Dealer Table and stored in the global variable dealer. The Dealer is an array of 2 elements: dealer[0] is a string description and dealer[1] is a function accepting an array of 4 cards and returning an array of 4 booleans. The handler for the ‘next’ button uses the dealer function to assign a value of .accepted for each card in the Board.

The Dealer Table is organized as a nested array. The first index selects one of the 3 game modes: K-2, 3-5, 6-8. The second index selects one of the dealer rules. We use 2 random numbers to select a dealer, so that the higher difficulty modes include the rules from the easier modes.