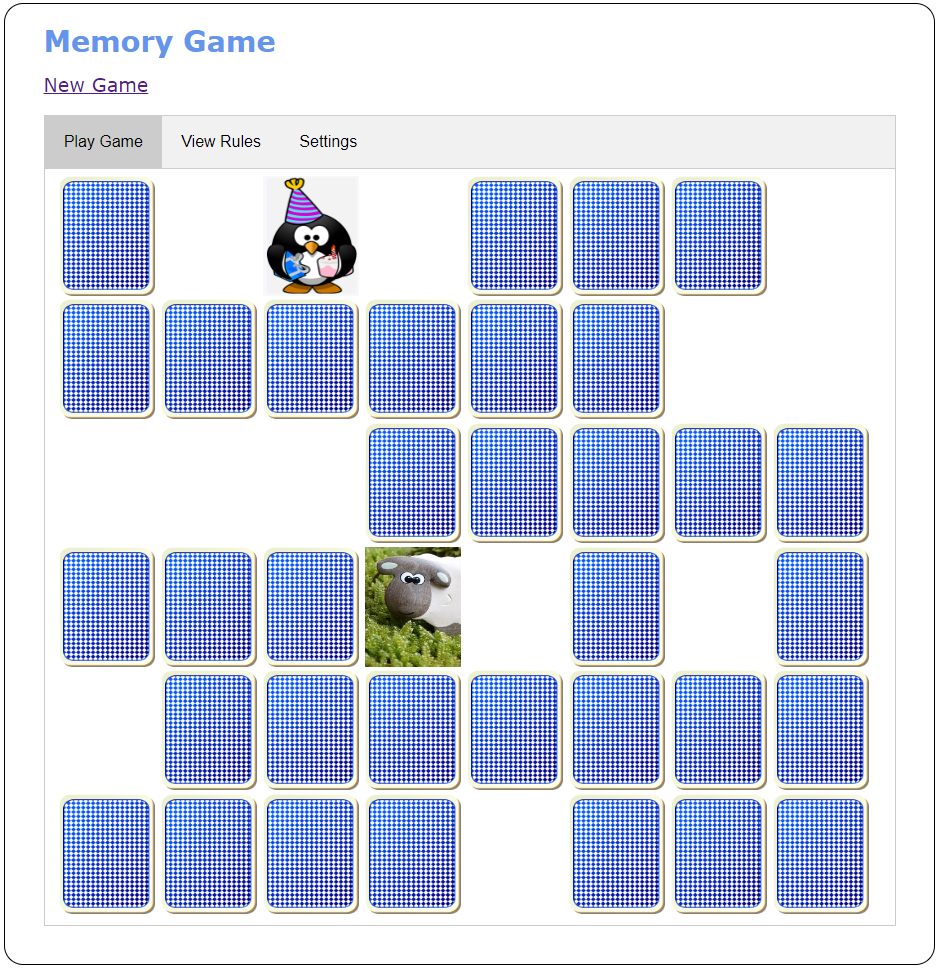
# Part 2: Modify the Memory game to use objects and modules

Part 2 of this case study has you modify the Memory game from part 1 so it uses objects and libraries.

If you haven’t already done part 1, you’ll need to do it first, unless it’s given to you as a starting point.

Prerequisites: Chapters 1 to 13.

## User interface



## Specifications

* Rework the code for the app so it uses objects. To do that, you’ll need to convert some existing functions to methods of an object, some existing variables to properties of an object, and so on.
* Organize the code that defines the objects in library files.
* If you need just a single instance of an object, and you don’t need to protect any private state, create the object with an object literal.
* Use a storage object to provide general methods that get a string from local storage, get an int from local storage, and set an item in local storage.
* Use a settings object to provide accessor properties that get and set the settings data (player name and number of cards) in local storage. It should use the general methods of the storage object to get and set the data.
* Use a Card class to provide methods for working with a card. It should include:
  + A constructor that creates a Card object from an <a> element for a card
  + An accessor property that determines whether the card is face down (the back of the card is displayed)
  + A method that accepts a Card object and checks if the current Card object is equal to the Card object that’s passed to the method
  + A method that flips the card using a fade effect
* Use a cards object to provide methods that return:
  + The src attribute for the card back
  + The src attribute for the blank card
  + The <div> object that represents the HTML for the cards
* Use a game object to track the number of turns, the number of matches, the correct percent, the first card of a turn, and whether the turn is over. This object should use the cards object to get data about the cards.

The game object should include accessor properties that return:

* + The <div> object that represents the HTML for the cards
  + The high score from local storage or a default value of -1
  + The correct percent value
  + A Boolean value indicating whether all the cards have matched

The game object should also include methods that:

* + Accept a Card object and use its properties to determine whether a turn is over
  + Accept a Card object, display the card, and determine whether it’s the first or second card of the turn and process the card accordingly
  + Calculate the correct percent and update the player’s high score if necessary
* The index.html file should include the library files in the proper order, so each library is included after any libraries it depends on. If necessary, use an import map for the library files that are modules so the files that import them can use module specifiers.