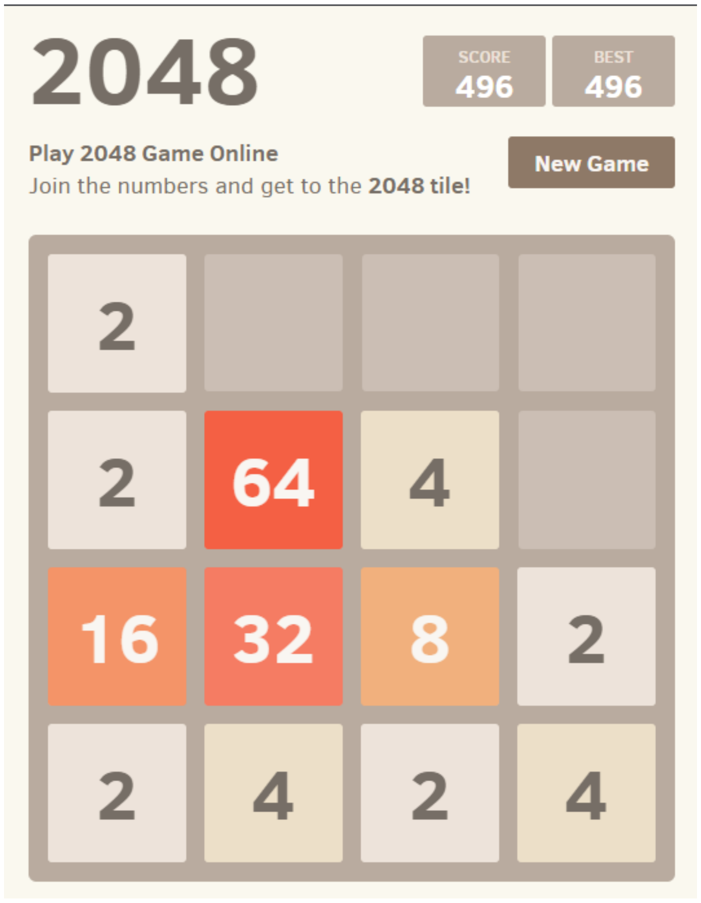
Problem Set 3

"Vue.js and Google Cloud Firestore"

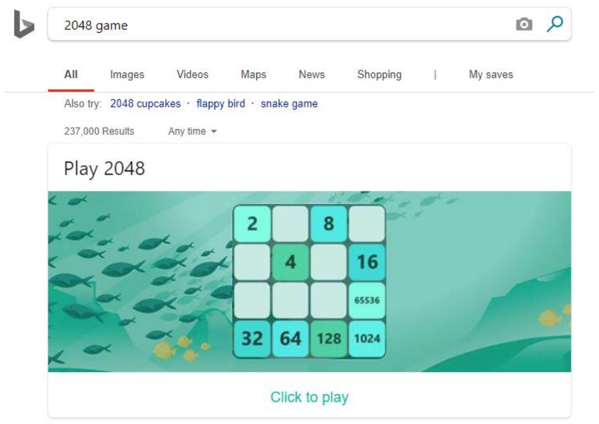
The following problem set is worth 100 points and is due by the date on the Canvas assignment. Please submit all relevant code in a zip file.

Problem 1: 2048 Problem (100 pts)

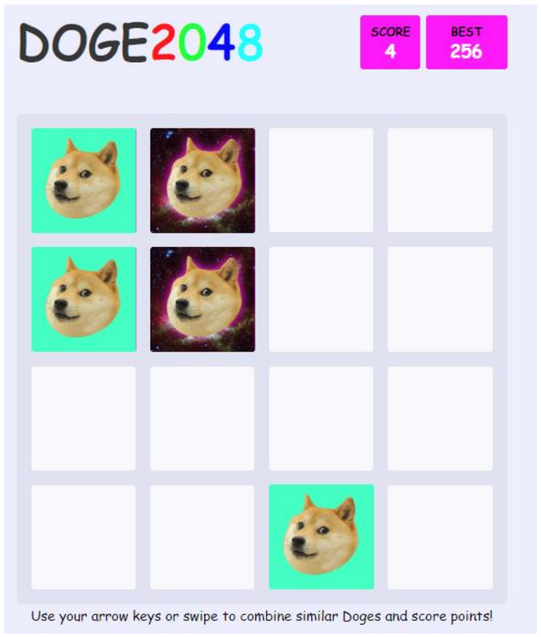
The game 2048 is a sliding puzzle block game which was incredibly popular a few years ago. There are numerous iPhone and Android implementations (clones) of this popular game. This game also has several JavaScript web applications as well. Here's one particular example http://2048game.com/



The search engine *Bing* also has their own playable version of this game as well, which can be played in browser simply by searching for the term:



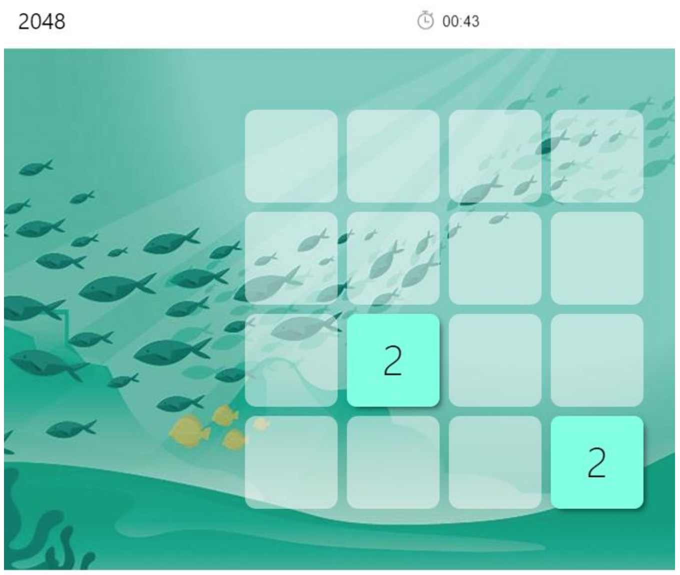
A developer also made a version of the game using a popular meme http://doge2048.com/


DOGE2048
SCORE 4
BEST 256
Use your arrow keys or swipe to combine similar Doges and score points!

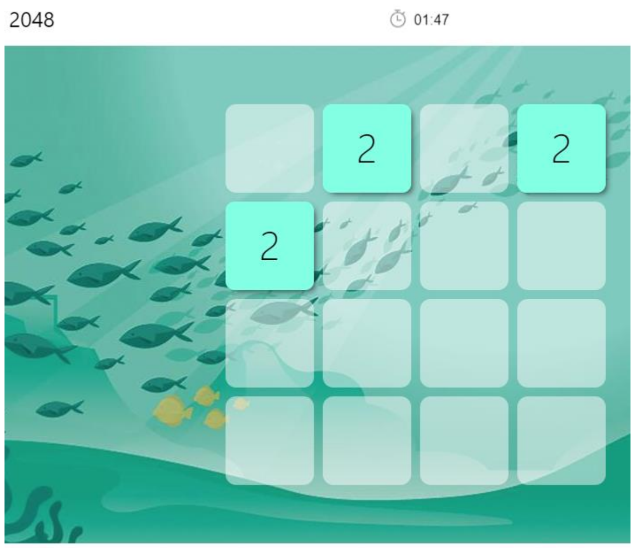
For this assignment you are to implement the popular 2048 game yourselves, with a few added requirements.

Rules of the game

2048 is a sliding puzzle game with sixteen tiles. When the game first loads two of those sixteen tiles are labeled with the value "2", as shown here:

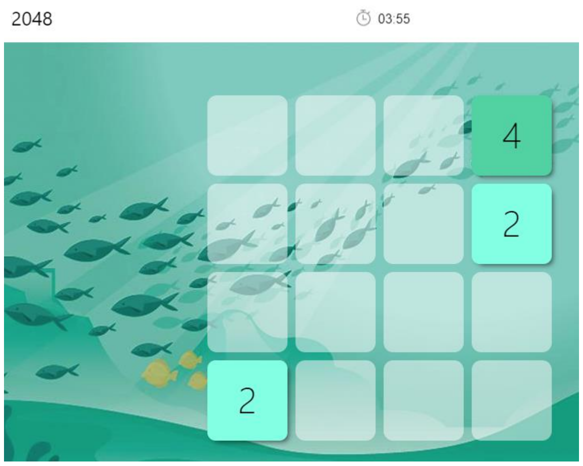


You can move all the tiles using the four “arrow” keys on your keyboard. For instance, entering the **up arrow** yields the following configuration:



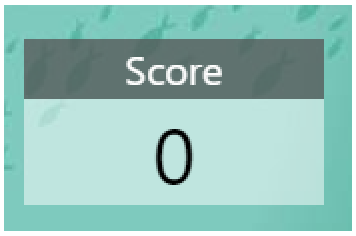
All the tiles immediately shifted to the top row in the grid. While doing this, a new (third) tile appeared with the number “2”.

The main rule of this game is to merge tiles with the same value. For instance, consider the configuration that occurs when entering the **right arrow** key:

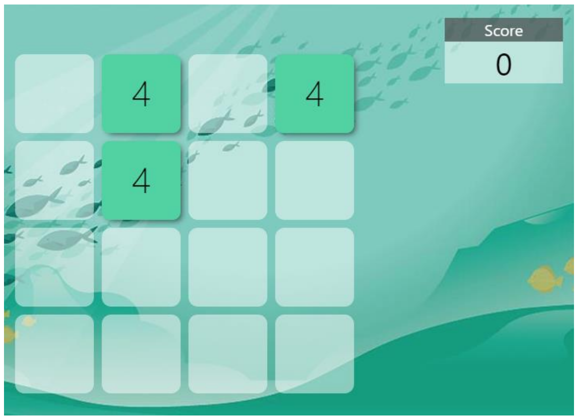


Upon doing this, the two upper tiles merged and formed "4". Note that after doing this, a new tile randomly spawned on the board with the value "2". Rule of thumb: merging two tiles each with value **x** yields a new tile with value ***2x***.

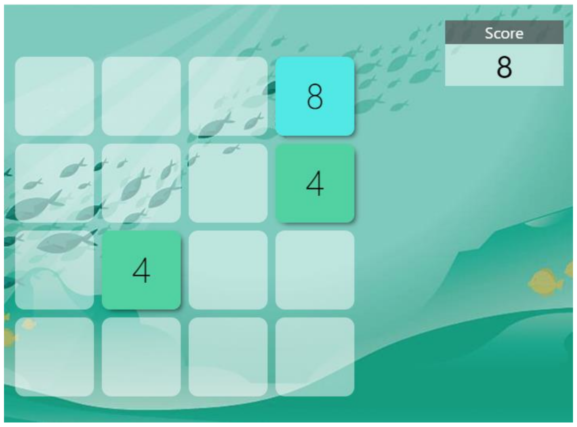
Take note that the game also includes a score field:



To update this value, consider the following:

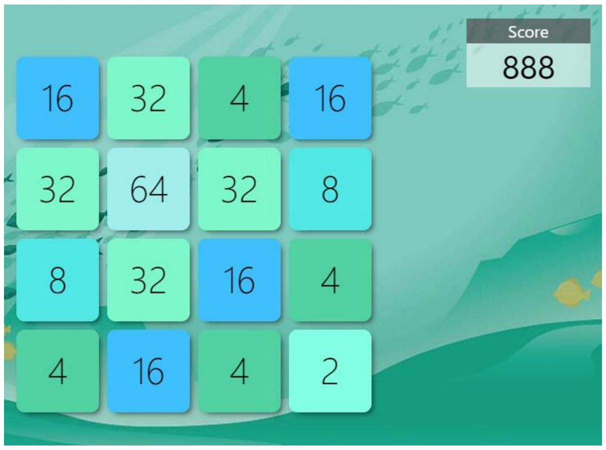


If you merge the two upper "4" cells like shown:



Your score should be incremented by adding the value of both tiles onto the previous score value.

The game ends when all 16 tiles have values and no more tiles can be shifted.

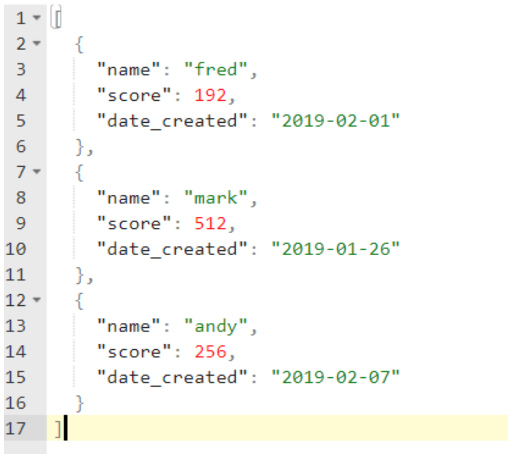


Assignment Implementation

For this assignment, you are to implement the 2048 game while incorporating both **Vue.js** and **Google Cloud Firestore**. Your implementation doesn't need all the "bells and whistles", and there are some extra credit features with this assignment as well.

High Score Feature

Consider the following JSON structure:



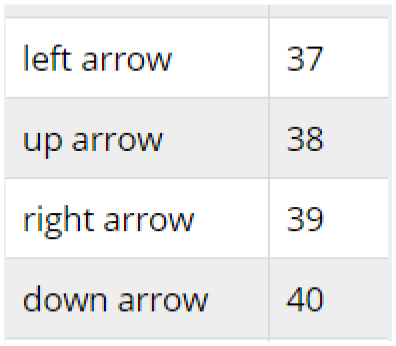
```
[
  {
    "name": "fred",
    "score": 192,
    "date_created": "2019-02-01"
  },
  {
    "name": "mark",
    "score": 512,
    "date_created": "2019-01-26"
  },
  {
    "name": "andy",
    "score": 256,
    "date_created": "2019-02-07"
  }
]
```

Your application should be able to keep track of which user recorded which score at any given moment in time. Store this data in a **Google Cloud Firestore** database, using the example code we went over in class.

The minimum requirements for the assignment are the following:

1. Present the user with a playable implementation of 2048 using **Vue.js**.
2. Allow user to play game to completion, until all 16 cells are filled.
3. Ask user for name, store the high score document in a **Google Cloud Firestore** database.
4. Retrieve and display the list of high scores to the user afterwards, and ask the user if the game would like to be played once more

For handling the arrow events on your keyboard, look for the following JavaScript keycodes:



left arrow	37
up arrow	38
right arrow	39
down arrow	40

Extra Credit

Animated Tiles (5 pts)

Animation is not a requirement for your submission. If you want extra points, animate the tiles moving to their new location each time the game board changes. jQuery might be a good approach for handling this.

Using the Vue.js CLI (5 pts)

The examples I'm showing in class initially for Vue.js will be using the CDN, similarly to how jQuery is being used in class. I will also be showing how to use the Vue.js CLI for creating single page applications. This approach involves Node.js and NPM, and requires you to use **.vue** template files for your components. Extra credit will be awarded to solutions which are pure single page applications.

Submission Requirements

For the submission, please include all code I need to run your applications. Feel free to use any libraries like jQuery, BootStrap, Materialize CSS in your submission since we've discussed them in class.

If using Node.js and NPM in your solution, do **NOT** include the *node_modules* directory in your submission (points will be deducted if otherwise.)

Submit your zip file via the Canvas drop box before the assignment due date.