Columns Assignment Testing Report

Author: Jackson Luff

The following document outlines my test plan, the methods of my design and intent of outcome.

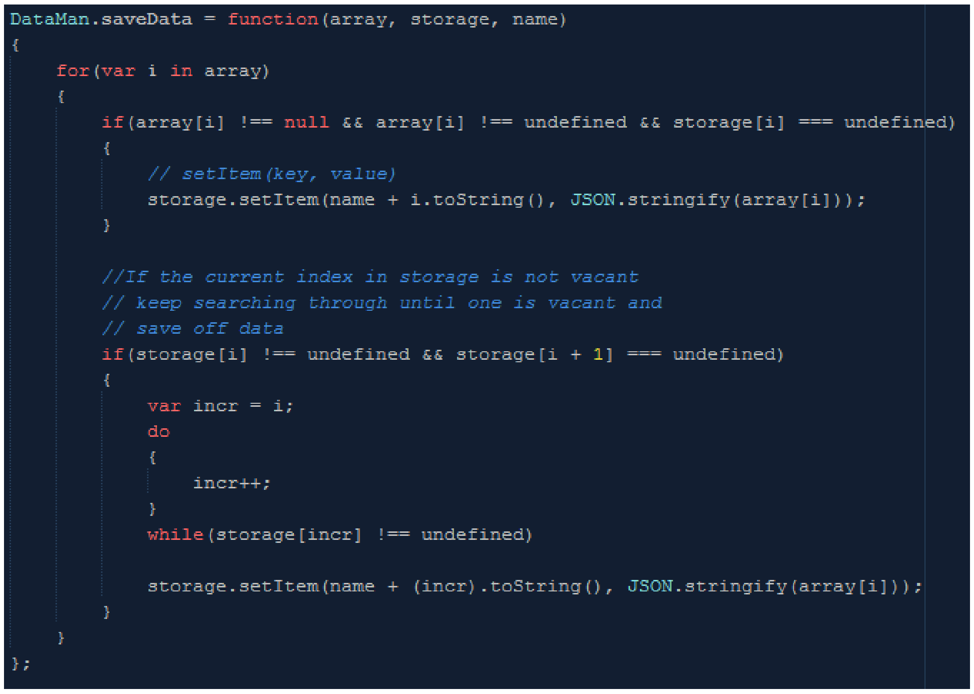
TEST PLAN

The main use of testing will apply inside the 'Column Manager'.js file where i continually check to see if a slot in the 2D array is either classified as null or a gem. Testing plays a large role when it comes to debugging. People often think 'programming can be done within hours because it's only a few thousand lines or so'.. But it's not the result that takes so long - it's the testing.
For example, i check to make sure i don't override data in local storage. If this procedure has to not have been executed every time prior to saving, this would result in loss of saves.

Another example of test implementation was checking all possible gems around a specific gem, branching off in attempt to recover a path greater than 3. Remove them if true, apply animation and remove them. I spent a lot of time figuring out the movement of the gems above the recently removed, and apart from a few minor errors, the current result seems to work just fine.

TEST DESIGN

The structural design of each test has been implemented in such a way that there is no duplication of jargon data and so that every unique test executed covers as much ground as possible in the grounds of which it specifies. (E.g. if a test in ran on managing data, no other result would log the same information about managing data unless the second test achieved a successful outcome. This process and/or structure assures a firm and clean logging system.)

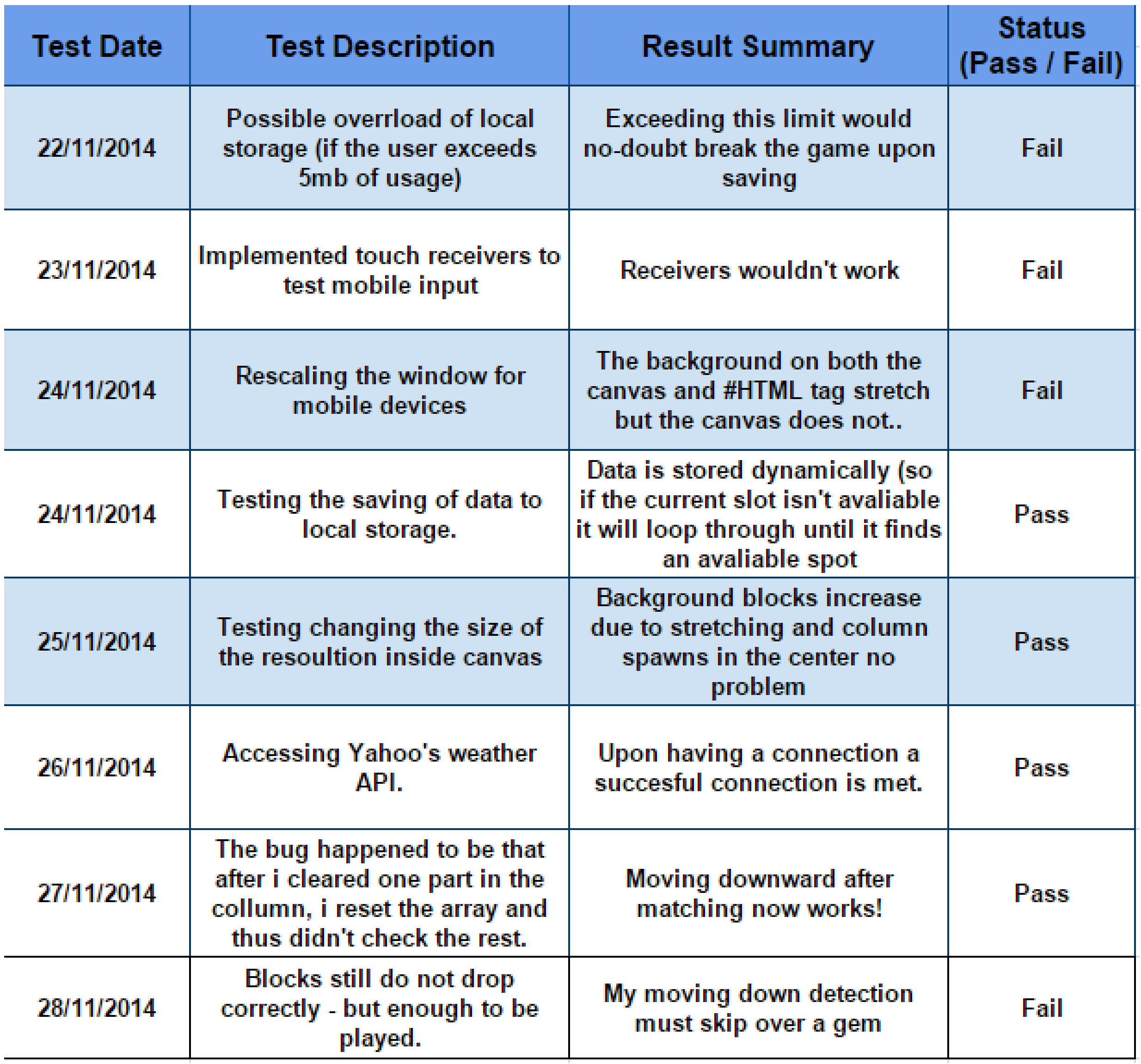


```
DataMan.saveData = function(array, storage, name)
{
    for(var i in array)
    {
        if(array[i] !== null && array[i] !== undefined && storage[i] === undefined)
        {
            // setItem(key, value)
            storage.setItem(name + i.toString(), JSON.stringify(array[i]));
        }

        //If the current index in storage is not vacant
        // keep searching through until one is vacant and
        // save off data
        if(storage[i] !== undefined && storage[i + 1] === undefined)
        {
            var incr = i;
            do
            {
                incr++;
            }
            while(storage[incr] !== undefined)

            storage.setItem(name + (incr).toString(), JSON.stringify(array[i]));
        }
    }
};
```

For example, i check to see if the index of the storage is available, if not, keep going through the data storage array until an available cell is found and add it to the array.

TEST RESULTS

The following table displays concluded results from the applied tests:



Test Date	Test Description	Result Summary	Status (Pass / Fail)
22/11/2014	Possible overrload of local storage (if the user exceeds 5mb of usage)	Exceeding this limit would no-doubt break the game upon saving	Fail
23/11/2014	Implemented touch receivers to test mobile input	Receivers wouldn't work	Fail
24/11/2014	Rescaling the window for mobile devices	The background on both the canvas and #HTML tag stretch but the canvas does not..	Fail
24/11/2014	Testing the saving of data to local storage.	Data is stored dynamically (so if the current slot isn't avaliable it will loop through until it finds an avaliable spot	Pass
25/11/2014	Testing changing the size of the resoultion inside canvas	Background blocks increase due to stretching and column spawns in the center no problem	Pass
26/11/2014	Accessing Yahoo's weather API.	Upon having a connection a succesful connection is met.	Pass
27/11/2014	The bug happened to be that after i cleared one part in the collumn, i reset the array and thus didn't check the rest.	Moving downward after matching now works!	Pass
28/11/2014	Blocks still do not drop correctly - but enough to be played.	My moving down detection must skip over a gem	Fail