**ReadMe Sections**

**Description**

After three weeks on the General Assembly Software Engineering Immersive course, I have completed my first project. So far, classes have focussed on HTML, CSS and JavaScript. Using these we were assigned a brief of creating a grid-based game. I chose Minesweeper as it was presented as one of the most challenging options available.

**Deployment link**

**Insert your Deployment link here:**

**Getting Started/Code Installation**

**Instructions**

*Explain how the reader accesses your code. Include a step by step approach.*

**Insert your Getting Started/Code Installation here:**

**Timeframe & Working Team (Solo/Pair/Group)**

Solo

I worked independently, and the project had a one-week deadline.

**Technologies Used**

JavaScript

HTML

CSS

**Brief**

**Instructions**

**Insert your Brief here:**

**Planning**

**Instructions**

The big challenge with this project would be how to manage tiles surrounding ‘zero’ tiles. I was aware of the potential problems that could occur especially when it came to infinite loops between two ‘zero’ tiles next to each other. I decided to streamline my planning focussing only the big parts of the game as if this didn’t work nothing else would. Once I had created an MVP which would manage the ‘zero’ tiles I would then plan how to add extra features.

My initial plan to make an MVP was to ->

1. Create a grid made up of cells
2. Add values to each cell
3. Randomly assign mines
4. Create a function to count the number of mines around each cell (count value)
5. Add a ‘lose’ function if a mine is clicked

My first idea was to find the count value of every cell as the game loads. I would then create arrays, and ‘zero’ cells touching and the surrounding numbers would be added to the array. I would start on the top left of the grid and make my way to the bottom right. If the user clicked on a cell, then the whole array would reveal.

The problem was I couldn’t find an efficient method to create the correct number of arrays.

I then changed my idea to include recursion. I knew that I needed to add extra features (win conditions, flags, timers) but my MVP just needed to show that the ‘zero’ tiles would show when they were supposed to.

Once this feature works, I could add features to check for win conditions (does number of cells minus revealed tiles = mines?). I could add flags (if flags are switched on it places a flag on the tile, or if there is already a flag it removes a flag. When there is a flag on a tile it cannot be clicked.) I could add a timer (the timer should start on the player’s first click and end at game over – win or lose). I could add a reset button (This would mean the player could restart without having to refresh the page). If I was feeling ambitious, I could also add difficulty levels which would add more mines. I also needed to add the extra rules Minesweeper has – for example a condition to check that the player’s first click is never a mine.

I drew out the process that every click would trigger, see below:

Diagram

Description automatically generated

**Build/Code Process**

My Code broke down into three main functions

1. createGrid
2. runGame (including createMines)

3)

i) checkArea (tile count > 0 || mine)

ii)checkArea (tile count === 0)

createGrid

Text

Description automatically generated

This function created our grid. The grid changed size based on the difficulty level with easy being 8 x 8 and challenge being 16 x 16.

It went through and gave each cell an ‘i’ value (the first cell had an i value of 0 and the last cell had an i value of width \* height– 1), an ‘x’ value (its x coordinate) and a ‘y’ value (its y coordinate)

runGame

When runGame was triggered, it checked multiple conditions. Is the game running? Has the tile already been revealed? Is it the players first click? Are flags on? Is there a Mine?

If the game is running, the process continued, otherwise it returned, ending the process.

If the tile had not yet been revealed, the process continued, otherwise it returned, ending the process.

If flags are on, if the tile clicked is a flag it turned the flag off then returned, and if the tile clicked is not a flag it became a flag, then returned.

If there is a mine on the tile selected, the game stopped running and the player lost.

If it is the players first click, it added the current tile to the createMineArray then ran the createMines function and then continued.

createMines

The createMines function ran whilst the length of the createMineArray did not equal the number of mines + 1. Each time it chose a random number and checked to see if it is already in createMineArray. If it is not already in the array, it added it to the array and then ran the function on that number. It then added the attribute mine to that randomly chosen cell value.

Text

Description automatically generated

checkArea

Within checkArea, x and y values are used. This was done to make sure that only tiles next to our selected tiles were checked and not tiles numerically next to our selected tile. Without this, tile 7 (end of row 0 – coordinate 7,0) would check tile 8 (start of row 1 – coordinate 0,1)

Text

Description automatically generated

I then used a filter to make sure only coordinates within our grid were accepted (preventing invalid coordinates such as -1,-1 from being checked.) and then used count to add 1 every time one of the surrounding cells contained a mine. I would add a ‘revealed’ class when the tiles had been checked, and added 1 to the ‘revealed’ count.

If the count value was greater than zero, the innerHTML would change to the tile’s count value, the check win function would check if number of cells minus revealed tiles = mines and if it did the game would end.

Text

Description automatically generated

checkArea -> count value === 0

If the count value === 0, the function would rerun, but instead of taking the event.target.value, it would take the surrounding coordinates as values. If the surrounding coordinate had already been revealed it would skip the process for that tile. This meant it would repeat until every touching ‘zero’ or number had been revealed.

Text

Description automatically generated

**Challenges**

The challenges were:

- having to add bounds to stop the cells counting the cells numerically next to them, instead of actually next to them  
- preventing endless loops during the recursion step – this was prevented by adding that it should only check surrounding tiles which had not been revealed.

- making sure only tiles within the grid were checked – this was managed by checking that only tiles within our filtered array were checked.

- making sure that tiles touching were checked instead of the numerically touching tiles – this was achieved using coordinates.

**Wins**

The design looks like classic Minesweeper

There are difficulty levels and the code will work regardless of size and number of mines.

Extra features – including a working timer and flags were added.

The zero sections work without any bugs

Added a restart button, so the player can reset without having to refresh the page

**Key Learnings/Takeaways**

I learnt how to use recursion.

I thought about how to tackle a problem and practised writing pseudocode before I started coding.

I worked out how to turn logical ideas into a working programme.

**Bugs**

doesn’t appear to work properly on mobile on challenge mode – adds extra tiles and the grid is the wrong size

**Future Improvements**

I would like to have a level system – when you beat the game on easy mode, it will automatically change the grid to challenge mode.