**Description**

In my first project for the General Assembly Software Engineering Immersive course, I had one week to build a grid-based game using JavaScript, HTML and CSS. I wanted to challenge myself, so I chose Tetris.

**You will find the deployed app here:** [Tetris](https://dancfc84.github.io/Project_1/)

**Getting started**

1. Download source code via the 'Clone or download' button in GitHub.
2. In the root folder open the index.html file in a browser

**Timeframe & Working Team**

* Timeframe:
  + 2 weeks
* Working Team:
  + Solo project

**Technologies used**

* HTML
* CSS
* JavaScript (ES6)
* DOM
* Git
* GitHub Pages

**Brief**

* Render a game in the browser.
* Use JavaScript, CSS and HTML.
* Use JavaScript for DOM manipulation.
* Deploy the game using GitHub pages.
* Game requirements:
  + The game should stop if a Tetrimino fills the highest row of the game board.
  + The player should be able to rotate each Tetrimino around its own axis.
  + If a line is completed it should be removed and the pieces above should move down a row.

**Planning**

I mapped out all four rotations and the axis point for each shape so I could replicate how the Tetrimino’s should move in my game.

A picture containing text, crossword puzzle

Description automatically generated

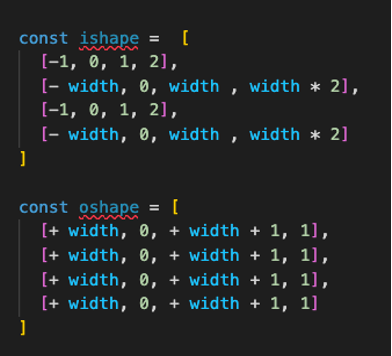
I also created a basic wireframe for how I wanted my game to look in the browser and on mobile.

Graphical user interface, shape

Description automatically generated with medium confidence

**How the game works**

* When the start button is pressed a 10 x 20 grid is created.
* 200 divs are then created to make up the grid, a class is added to each of them to style the grid. Each cell is then added to an array called cells.
* A current score, a lines cleared count, a top score (all stored in the browser using localstorage) and a display showing what level you are currently on is also created.
* The Tetriminos were created by setting the axis point of the Tetrimino to zero and then the other three cells were created from that point. Each shape has four rotations.

[](https://github.com/dancfc84/Project_1/blob/master/screenshots/TetriminoExampleArray.png)

* Before the game starts, two shapes need to be created, the first shape for the main grid and the other for the next shape. I used setTimeout to move the Tetrimino down one row at a time, the speed of the intervals between movements varies depending on the level you are on.
* The Tetrimino is added to the grid by assigning the specific class of the current shape to the appropriate cells in relation to its current location.
* The shape is removed from the grid (class removed from those cells) and re-added one row below (class added to the new cells).
* If arrow-left, arrow-right or arrow-down are pressed the Tetrimino will move in the direction pressed on the keyboard.
* If arrow-up is pressed the rotation of the shape will change and the next entry in the shapes array will be used to add the class back to the cells.
* When a shape is removed or rotated, there are checks to see whether the future position of the Tetrimino is allowed. If the future move cannot be made, the current cells are marked as full, and a new shape is created on the grid. If the shape cannot move right or left or be rotated but can be moved down, it continues its descent down the grid.

**Challenges**

**Featured Code: Removing Shapes and Re-adding them, checking whether the future move is possible**

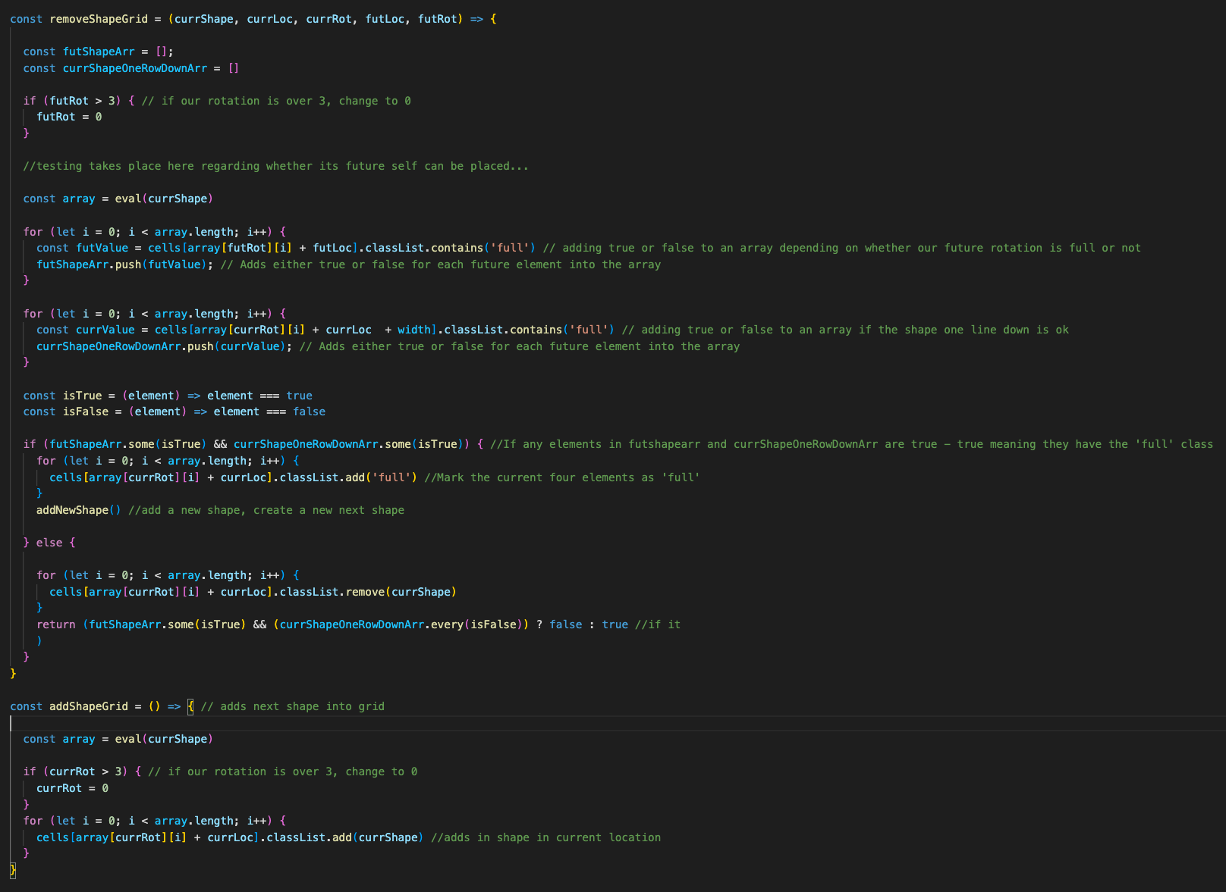
This function receives several arguments, including the current shape, location and rotation and also what the future location will be (for example, if the user presses the left arrow key, this will be the currLoc - 1).

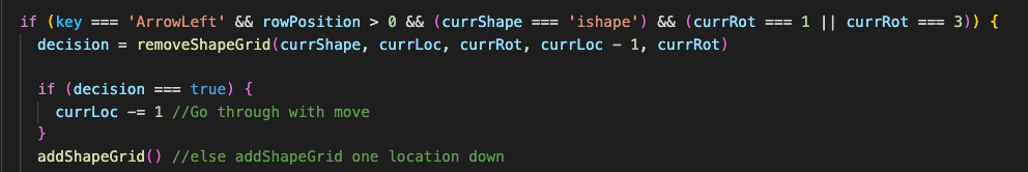
The function then finds the four future cells and iterates through them, checking whether the cells for the future location of the shape contain the "full" class.

It then finds the cells for the current location but one row down and iterates through the four cells to see whether they contain the "full" class.

If both the future shape array and the current shape one row down array have any cells with the full class, it marks the current cells as full and adds a new shape.

If that is not the case, it will check if the move left or right or a rotation is possible, if possible, it will allow that move and move the Tetrimino down a row, if the move is not possible, it will just move the Tetrimino down one row and ignore the key press.

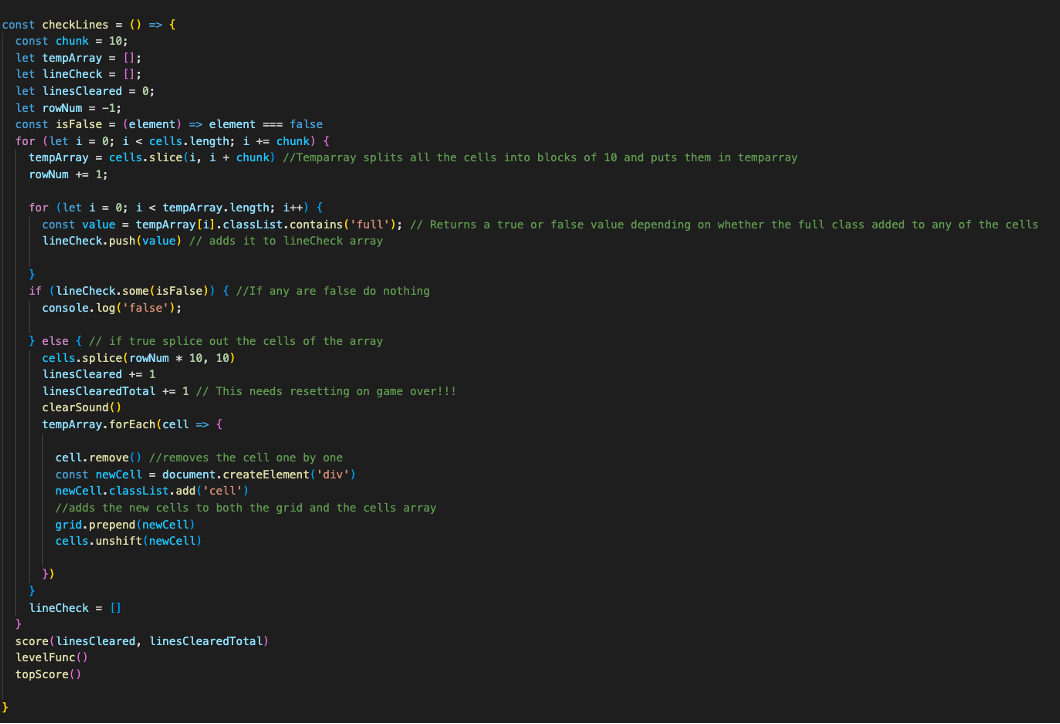
**[](https://github.com/dancfc84/Project_1/blob/master/screenshots/removeShapeGrid.png)**

**[](https://github.com/dancfc84/Project_1/blob/master/screenshots/ExampleMoveTetris.png)**

**Featured Code: Line Completion**

I split the cells array into chunks of 10 which signifies a line in my grid, then a for loop iterates through each row of the chunks of 10 cells, if all the cells contain the class ‘full’ the row of cells are removed from the array and the line of divs are removed from the grid.

I then used unshift to add the cells back to the array and then prepend to add the divs back to the grid at the top.

[](https://github.com/dancfc84/Project_1/blob/master/screenshots/CheckLines.png)

When a line is removed the score and lines cleared are updated. If you get the top score this is updated as soon as you surpass it. The current level is linked to how many rows you have cleared.

When a piece is spawned at the top of the grid which overlaps another piece the game ends.

**Wins**

This was the most complicated project I had undertaken up to this point. I was pleased with the way that I broke down the project into smaller pieces, working out the logic required in JavaScript for each part of the game, so it became less overwhelming.

**Key Learnings/Takeaways**

The project tested the JavaScript theory that I had been studying over the previous months, completing this project gave me confidence that I could successfully put in practice what I had been learning with general Assembly and Udemy to create a complex application.

It was my first responsive website; I had a lot of fun using Chrome’s developer tools and making the game playable on mobile.

**Bugs**

* The music stopped working when the site was deployed.
* I had issues with key presses registering too many times when a key was pressed, I implemented a workaround for this, but it stopped the ability to hold down the down-arrow for a faster descent.

**Future Improvements**

* Fix the bugs highlighted above.
* Improve styling.