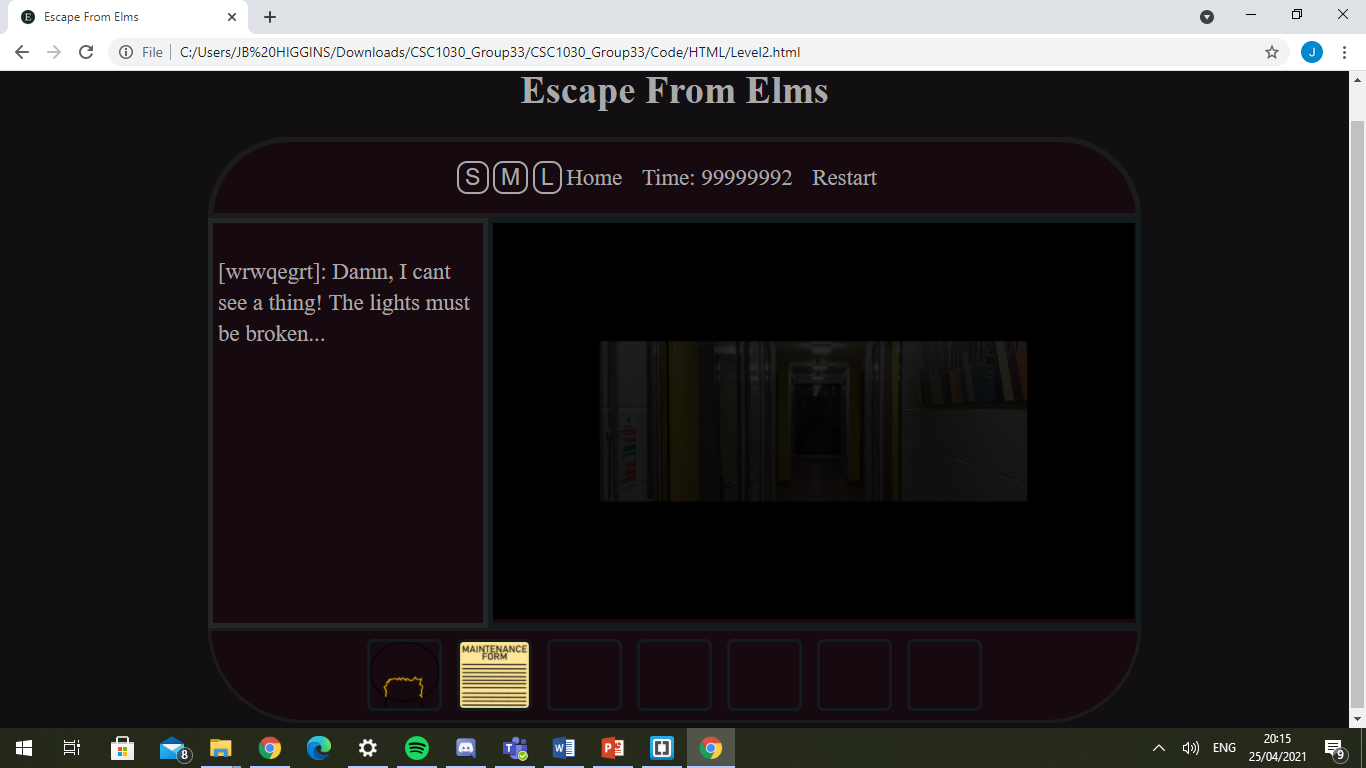
**Individual Summary Report**

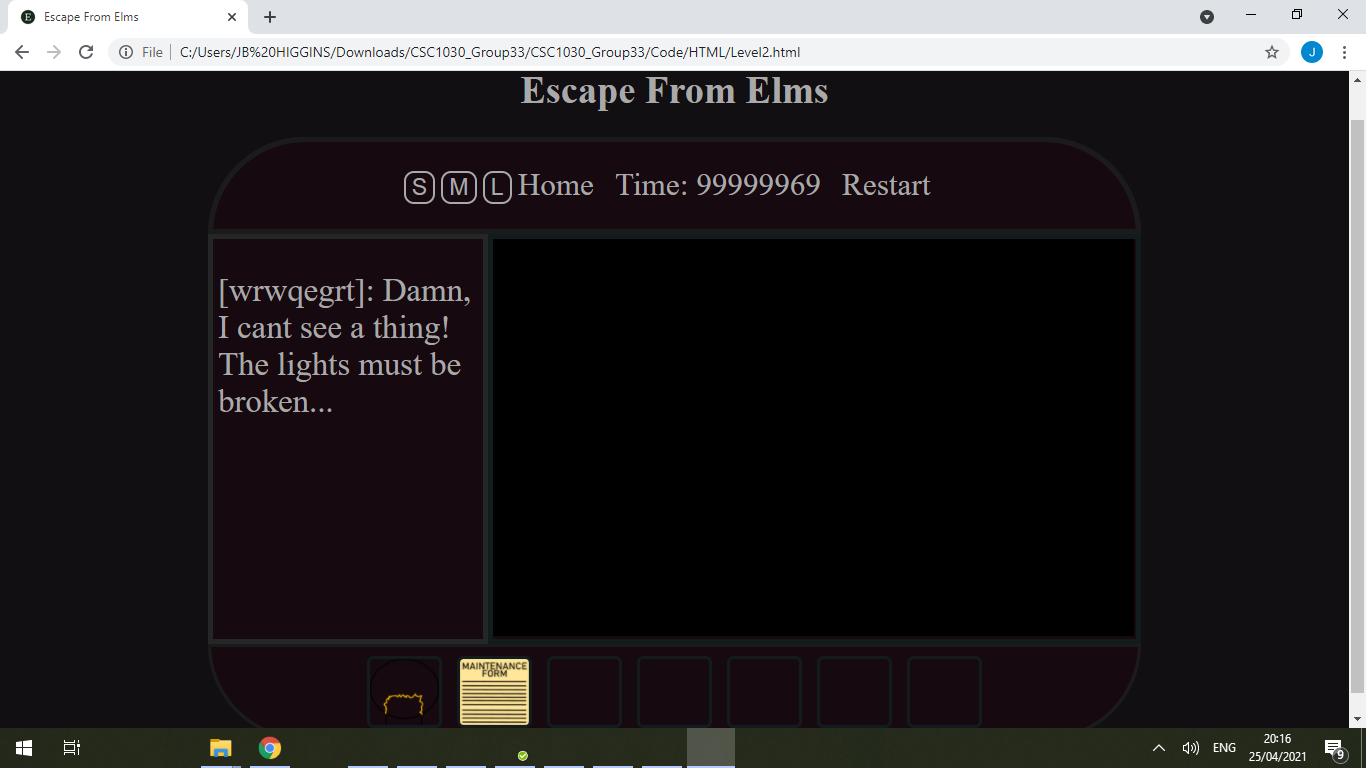
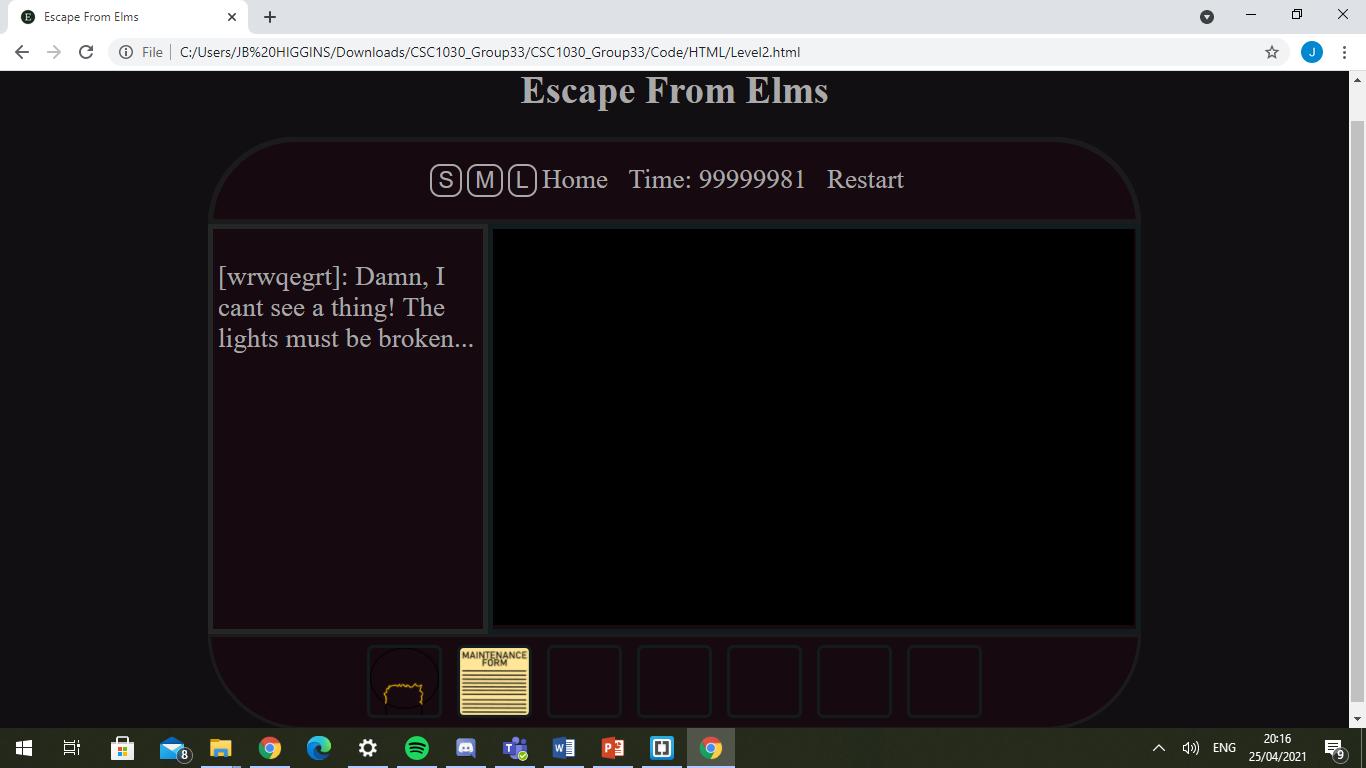
***TEXT RESIZER***

For one of our accessibility features I implemented a way of being able to change text sizes. This is presented to the user as 3 buttons within the navigation bar, labelled ‘S’, ‘M’ and ‘L’ (for small, medium and large). Each of these buttons calls to a function in JavaScript which in turn alters the CSS font size properties for the text. The medium function shares the same values as the default values for the page, and the other functions use the ‘larger’ and ‘smaller’ values. *I would consider this a great contribution to the overall project as it has been implemented in everyones’ level.*



**Buttons In Navbar**

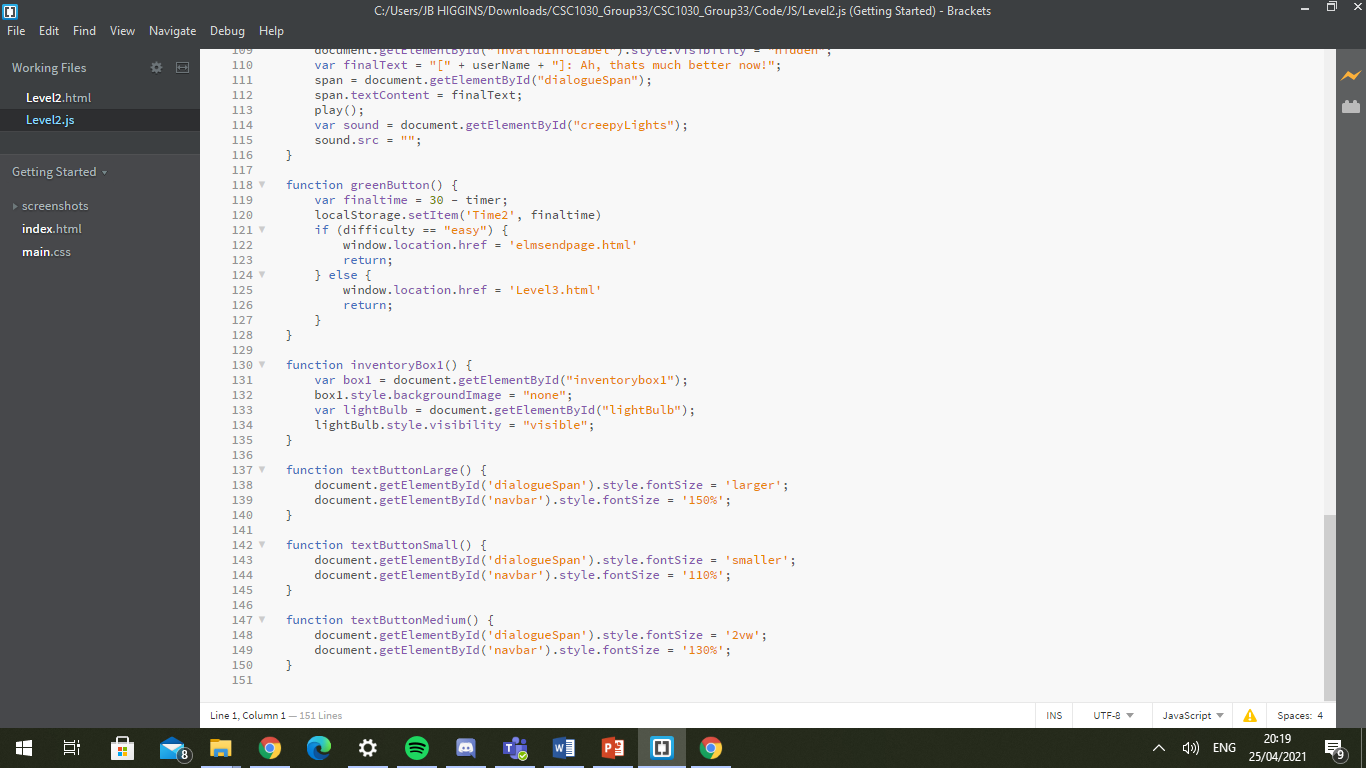




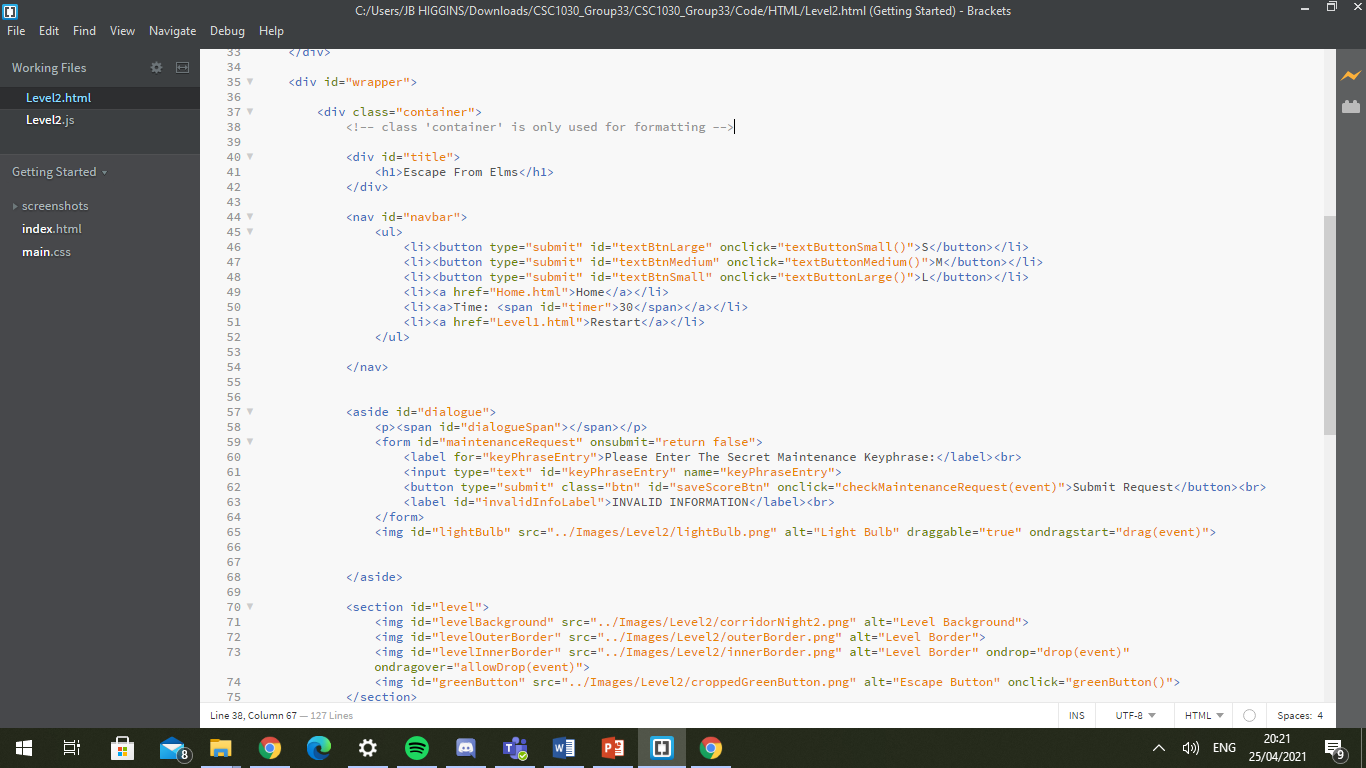
**Large**

**Small**

**Medium**



**JavaScript Showing how the ‘larger’ and ‘smaller values are used**



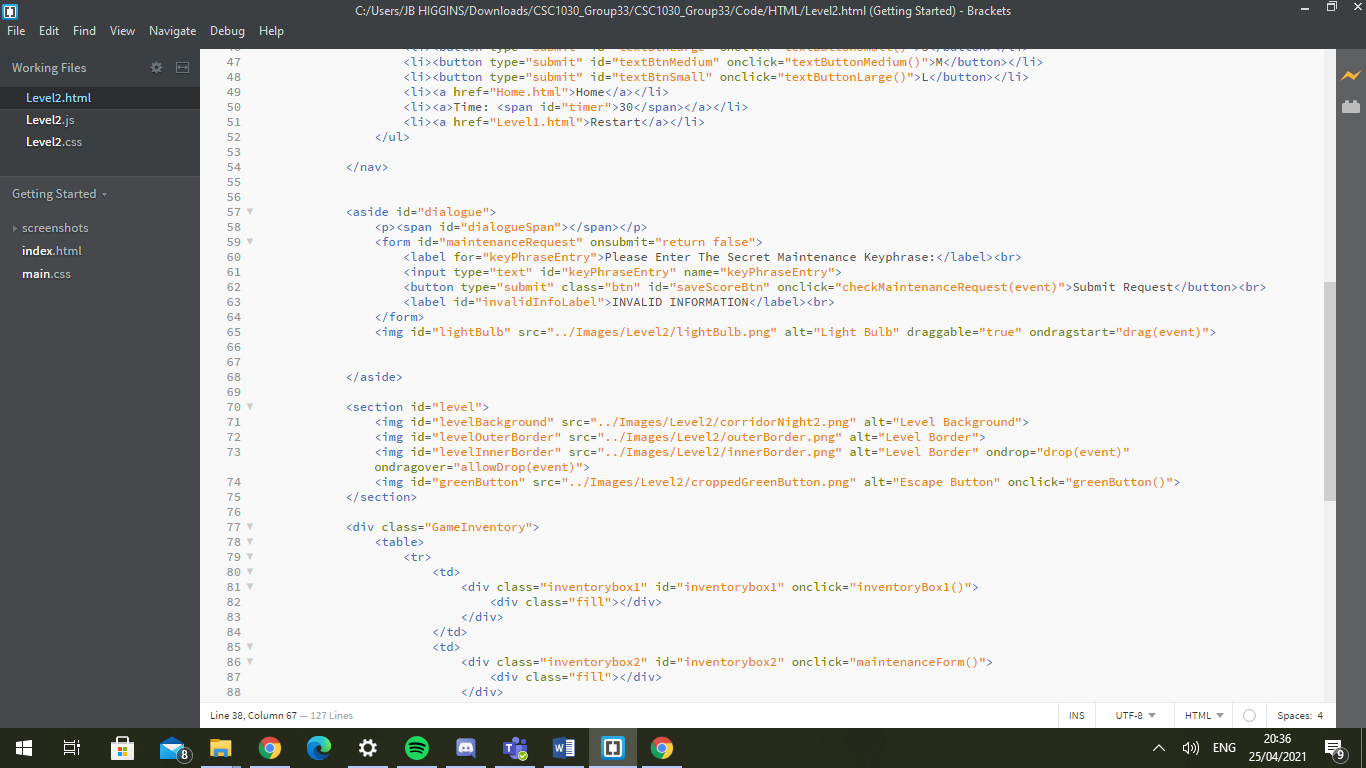
**HTML showing buttons in Navbar**

***DRAG & DROP***

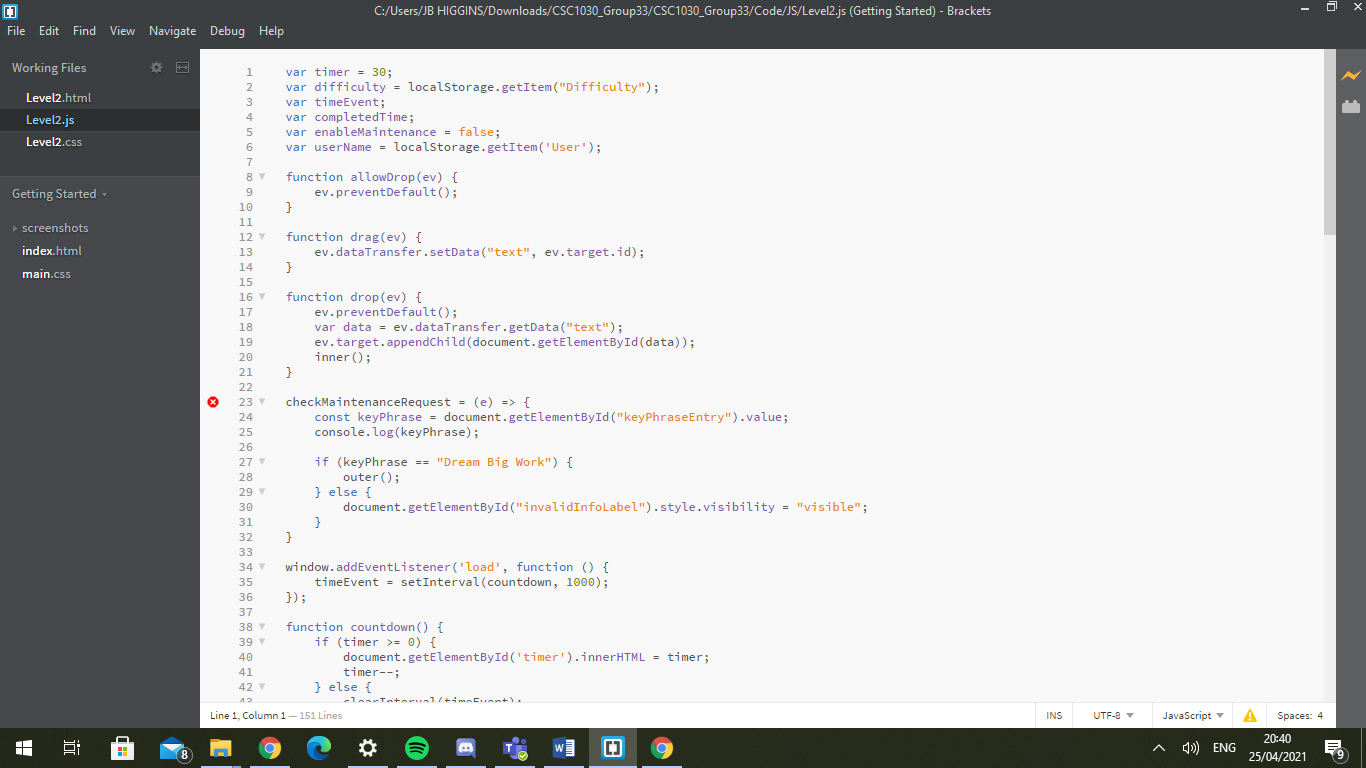
In my level the user is tasked with escaping a corridor with flickering lights that reduce visibility. The way the user fixes the initial flickering light is by dragging a light bulb into the obscured area. This has been implemented using the ‘draggable’ attribute in HTML. By setting this to be equal to true, the browser recognizes the image should be drag-able and alters the cursor appropriately. We then also use the ‘ondragstart’ attribute to call to the ‘drag’ function in JavaScript. Similarly, additional features had to be implemented for the area onto which the user drops the object. This has been done by using the ‘ondrop’ attribute to call to the appropriate JavaScript method. *I would consider this a top technical feature of the game.*

**‘draggable’ & ‘ondragstart’ attributes**

**Light Bulb is drag-able from Dialogue box**



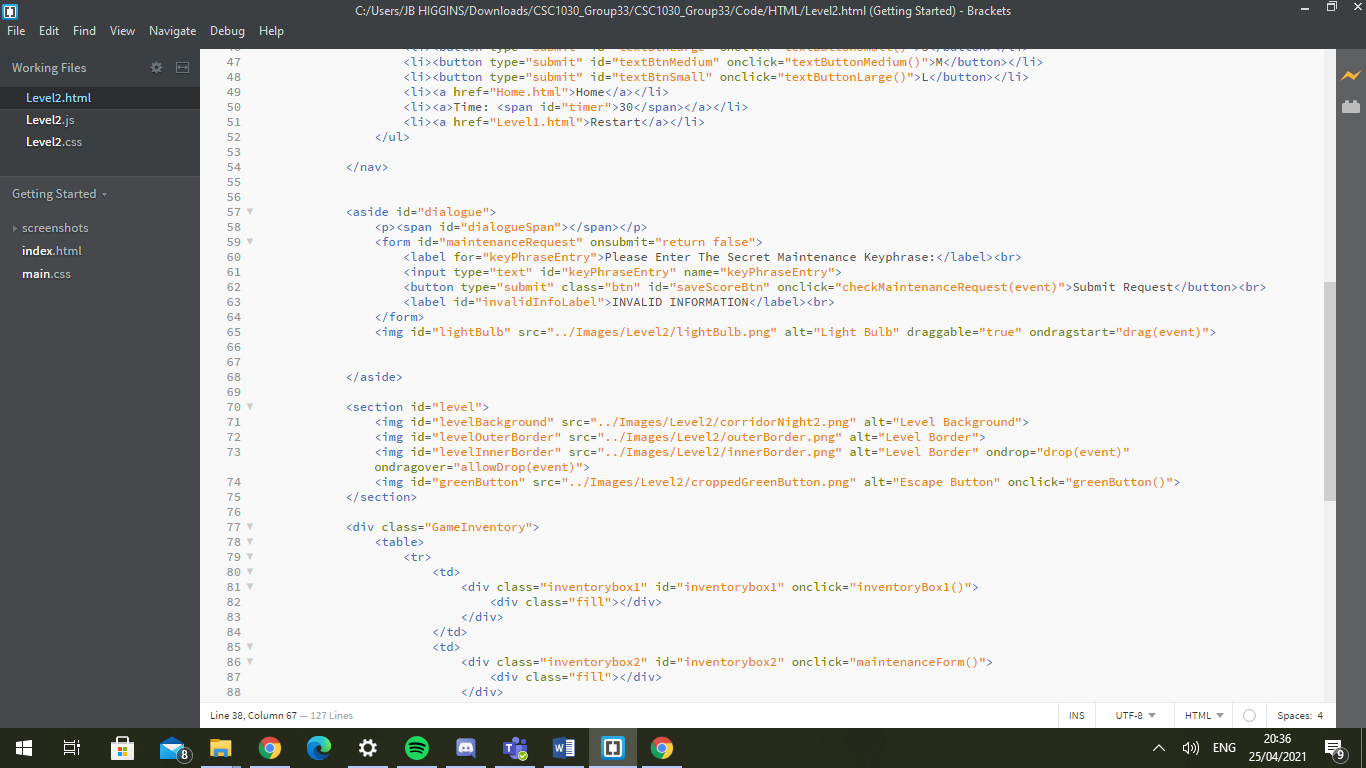
**‘ondrop’ attribute**



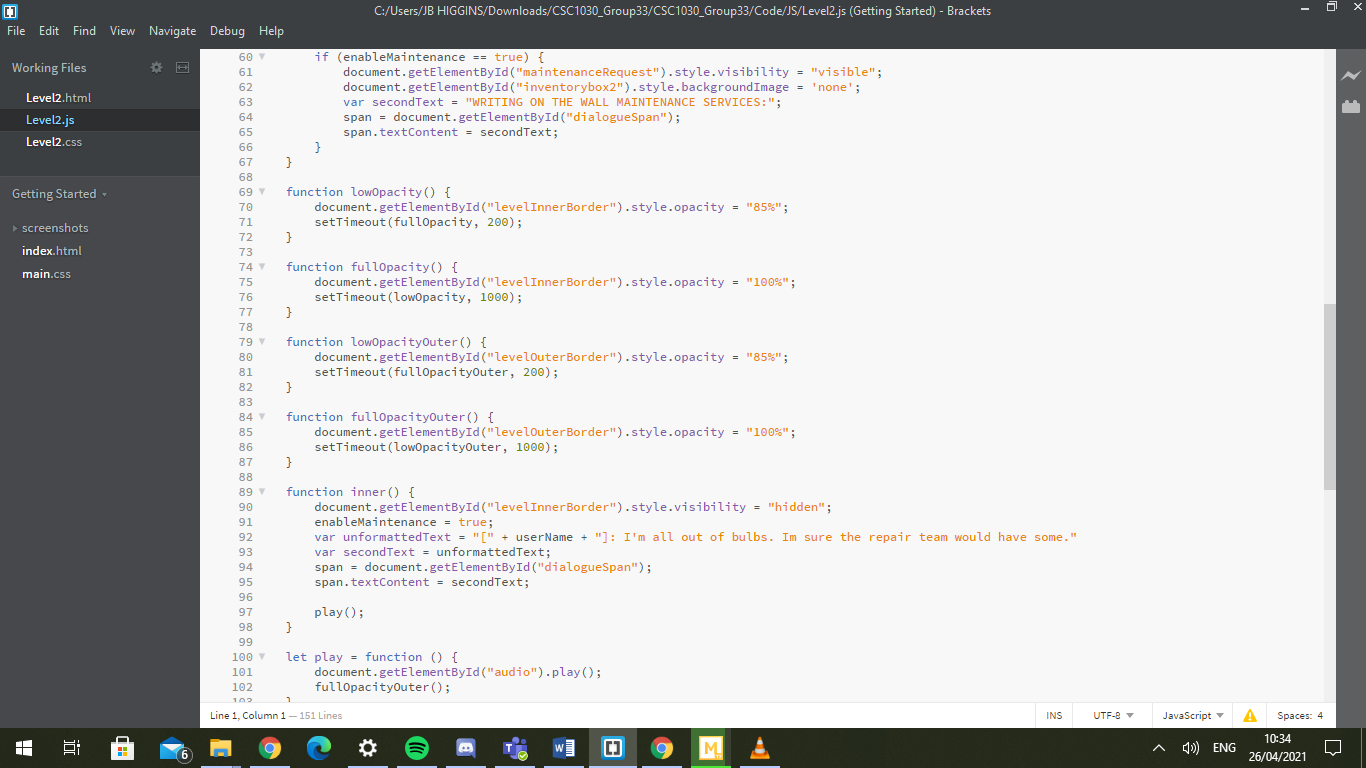
**JavaScript that facilitates Drag & Drop API**

***LIGHT FLICKER***

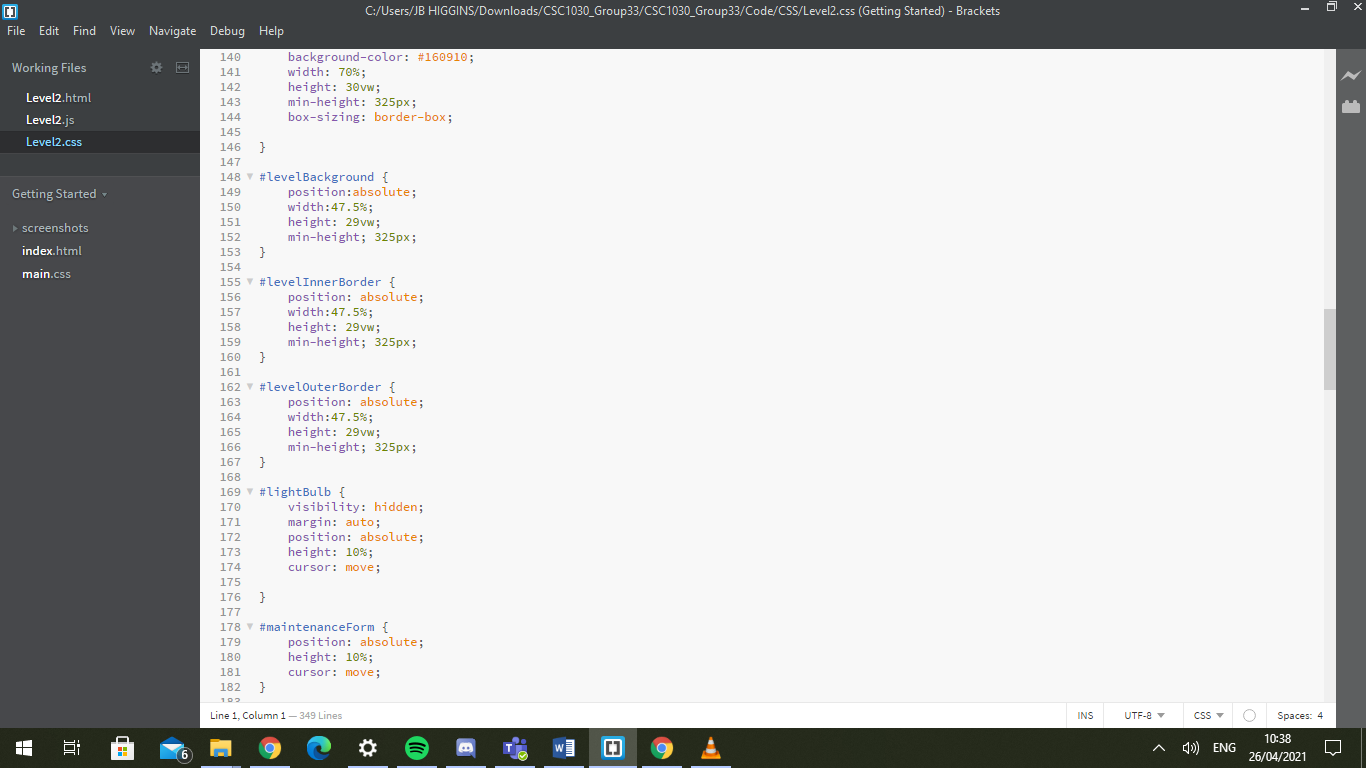
In my level I have used the ‘setTimeout’ function in JavaScript to alternate between functions that change the opacity of the level image. This gives the illusion of a flickering light. *I am quite proud of this feature, as it provides a visually engaging experience to the user and is a prime example of the use of time functions in a context outside of a timer.*



**Image Placement**



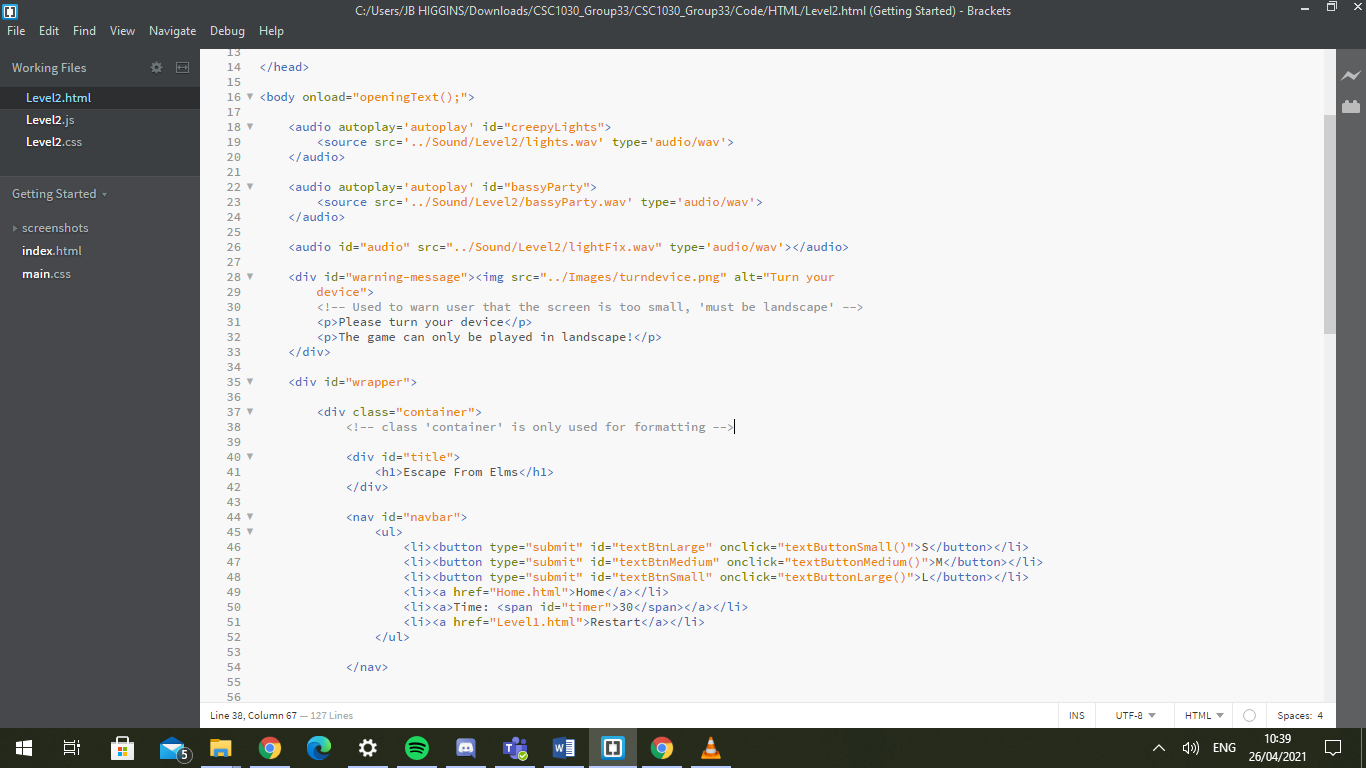
**JavaScript Methods which show how Timeout is used to provide a delay and how the functions call on each other to provide the illusion of flickering lights**



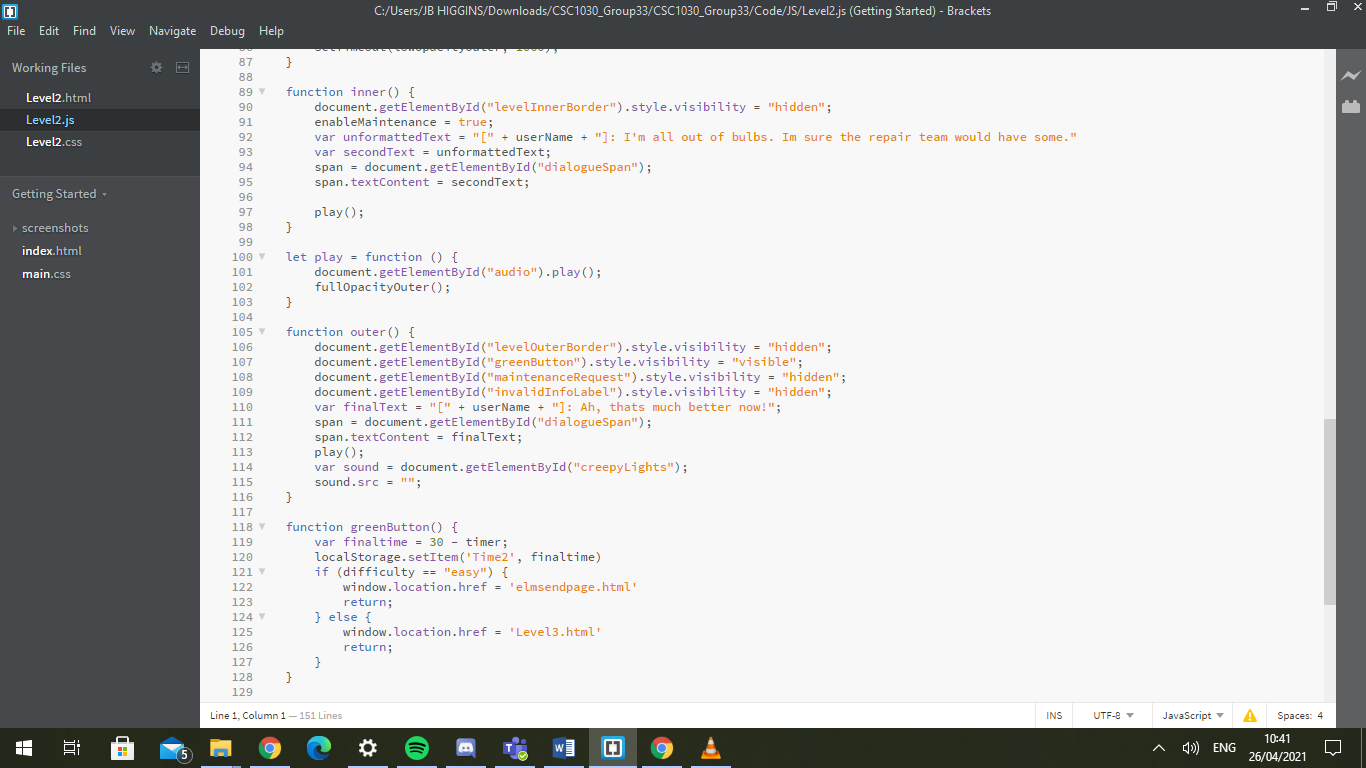
**The CSS across the level background and the two borders is consistent so that they sit flush with one another**

***MUSIC***

I constructed the music for my level by obtaining two royalty free tracks from ‘freesounds.org’. I overlaid the sound of lights flickering on top of a muffled party. This track plays automatically on compatible browsers. I also have switch sound effects for when the user successfully fixes a light. *I feel like this gives off an appropriately creepy tone for the level that is very consistent with the rest of the game.*



**The audio is set up with AutoPlay for compatible browsers**



**JavaScript that stops the flickering sound effect once the player has fixed the lights**