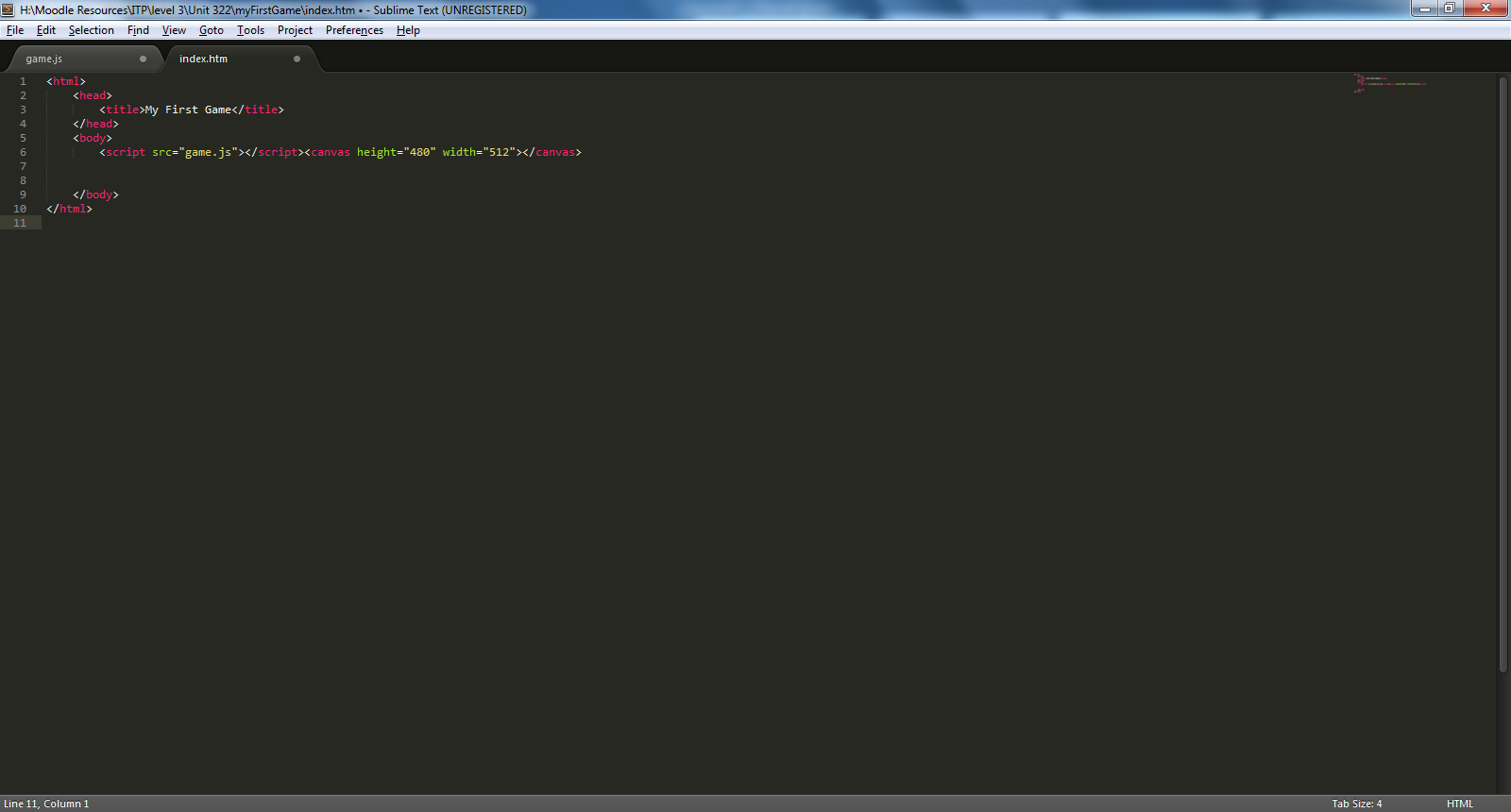
**My first JavaScript game - Catch the goblin**

This documentation will go through how to create a web based computer game called Catch the Goblin using JavaScript. Once you have created the game, you will be able to make changes to it to improve the game.

**Step 1 – Setting up the canvas**

Create a new HTML in your favourite text editor and type in the following information:

****

1. Inside the body of the HTML file a link has been made to an external JavaScript file which is where all of our game code is going to go.
2. The canvas tag has been used to specify how high and wide we wish the area for our game to be, as well denote where on the screen the game is going to be. In this case, the game will be displayed in the top left corner of the screen.
3. Notice that in the above we first specify a title for our game inside the head of the HTML file.
4. Once you are done, create a new folder called “My first Game”
5. Save your file as index.html into a folder “My first Game”

**Step 2 – Creating the JavaScript file**

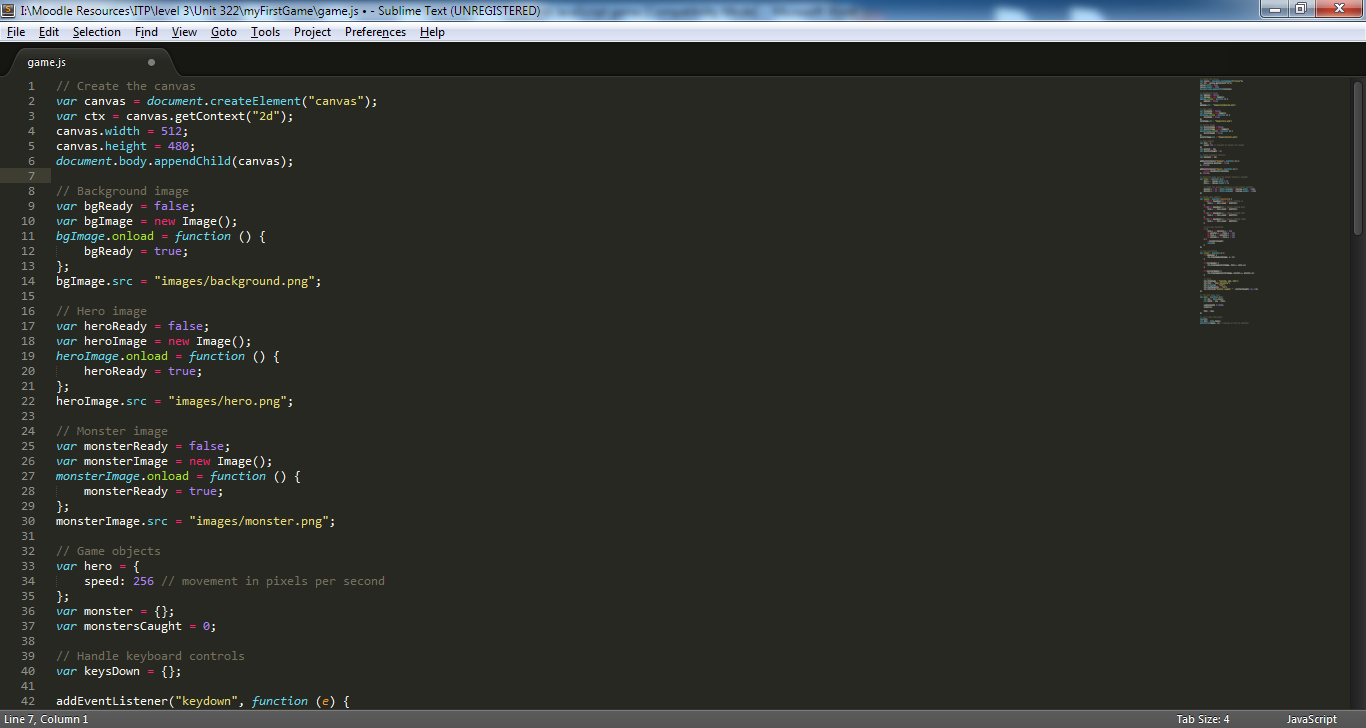
Using your favourite text editor create a new file, this file will be where we type in all of our JavaScript code for the game. Before we begin creating the file, save your project as game.js which was the name that we chose within the link inside the HTML file. Saving the file now will also allow you to see how the JavaScript is constructed more than saving it later on.

Inside your JavaScript file type in the following code:

The code shown will create a canvas element on the screen and add it to a variable called canvas

The context is set to 2d as we will be working in 2d and the width and height of the canvas has also been specified.

Finally the canvas is appended to the body of the document as a child object i.e. becomes part of the web page



**Step 3 – Inserting the images into the game**

Shown below is how to set up a background image for use in the game, type in the following code below the canvas creation code:

First a variable is created to check if the background is ready

A variable is then created which will act as a new image

When the script loads we are wanting the image to become ready and so its value changes to true

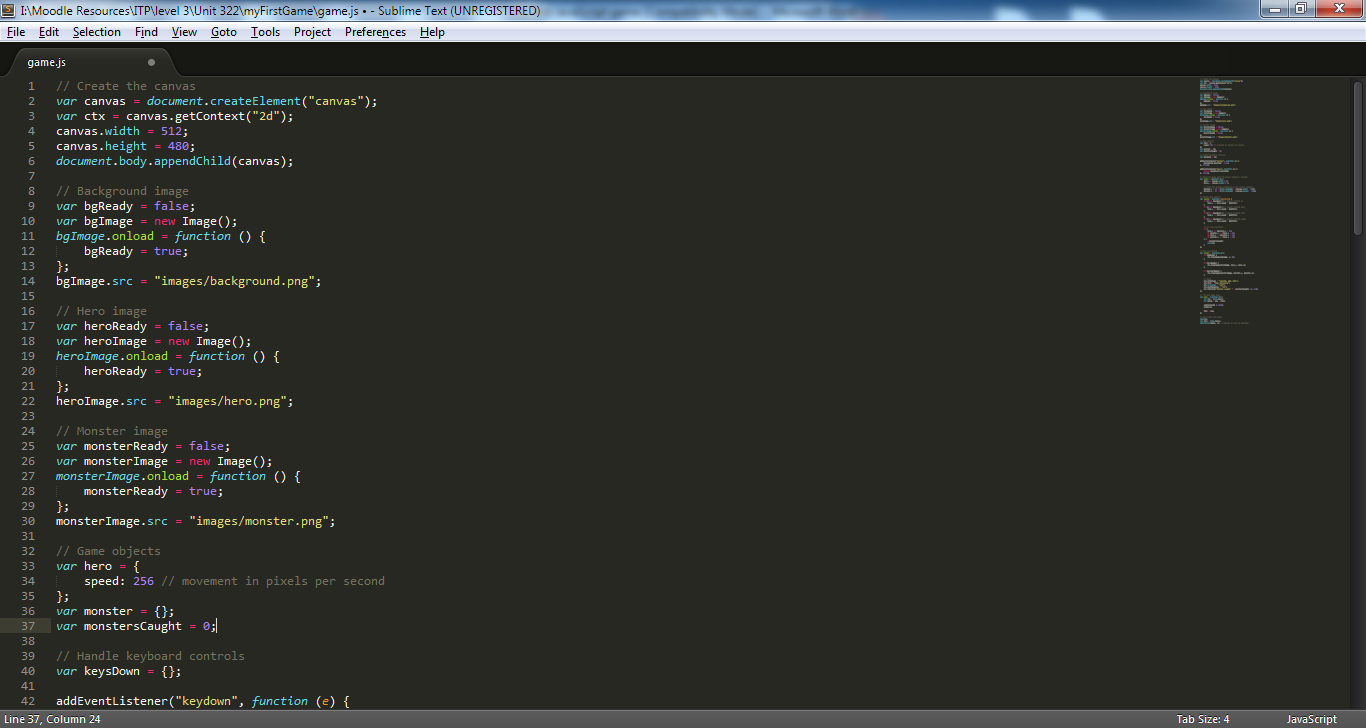
Finally the image source is stated to where the image is stored

First a variable is created to check if the background is ready

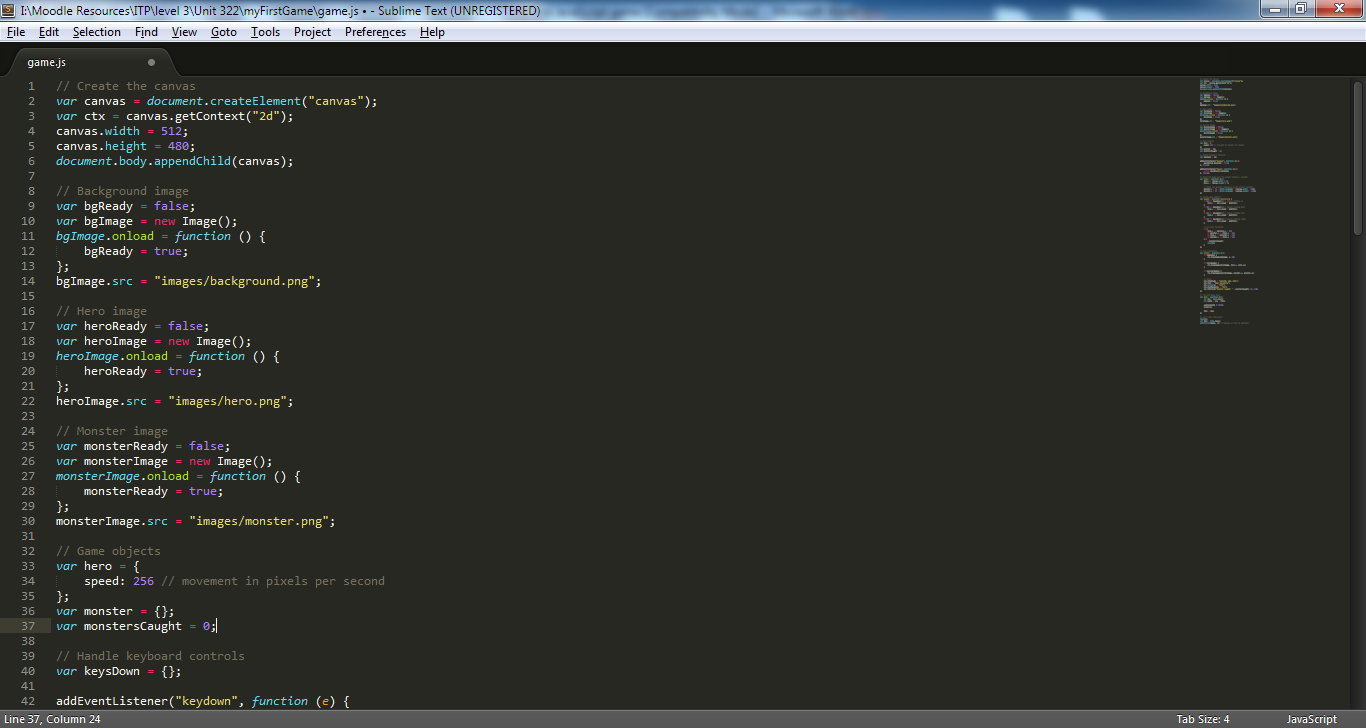
A variable is then created which will act as a new image

When the script loads we are wanting the image to become ready and so its value changes to true

Finally the image source is stated to where the image is stored



Next we create another image but this time for the main character, in this case we are going to call the main character hero and so we create an image for the hero under the background image code as shown:



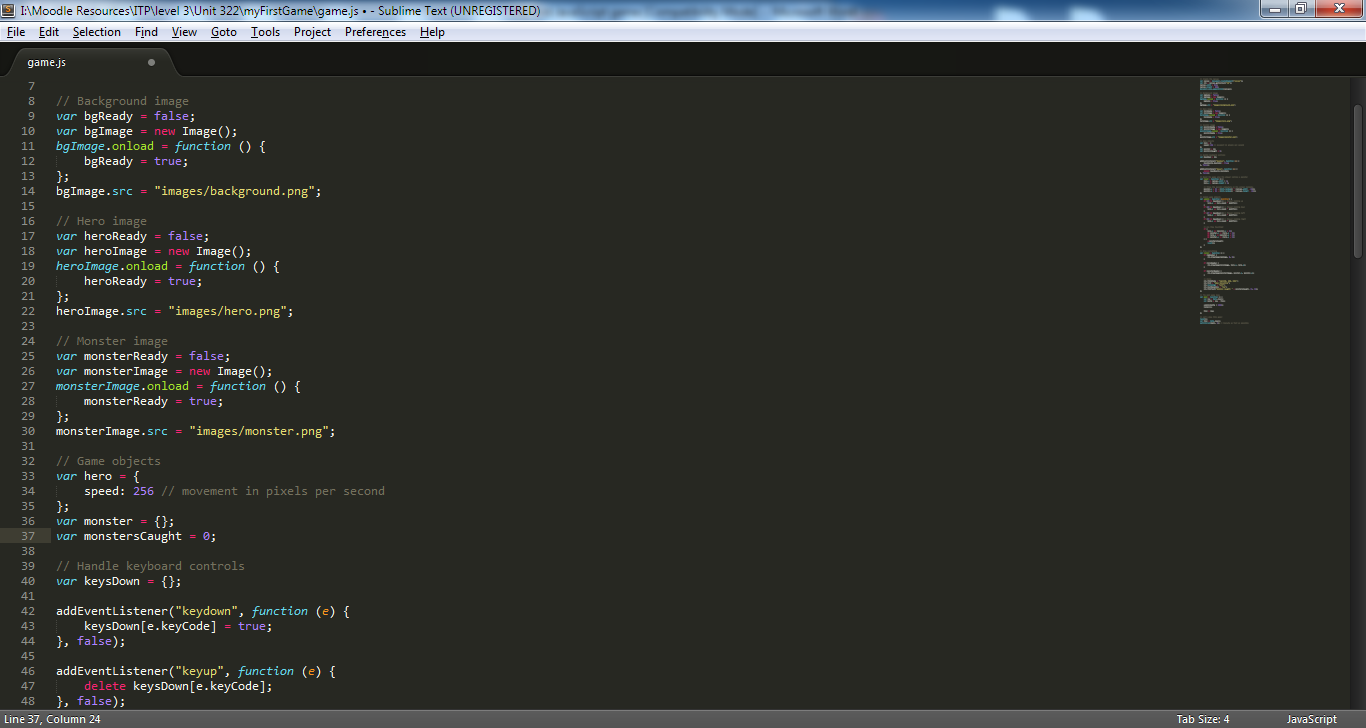
The next image we need to set up is for the enemy, in this case we have decided to call the enemy monster and so the image for the monster is set up directly below the hero image code as shown below:

First a variable is created to check if the background is ready

A variable is then created which will act as a new image

When the script loads we are wanting the image to become ready and so its value changes to true

Finally the image source is stated to where the image is stored



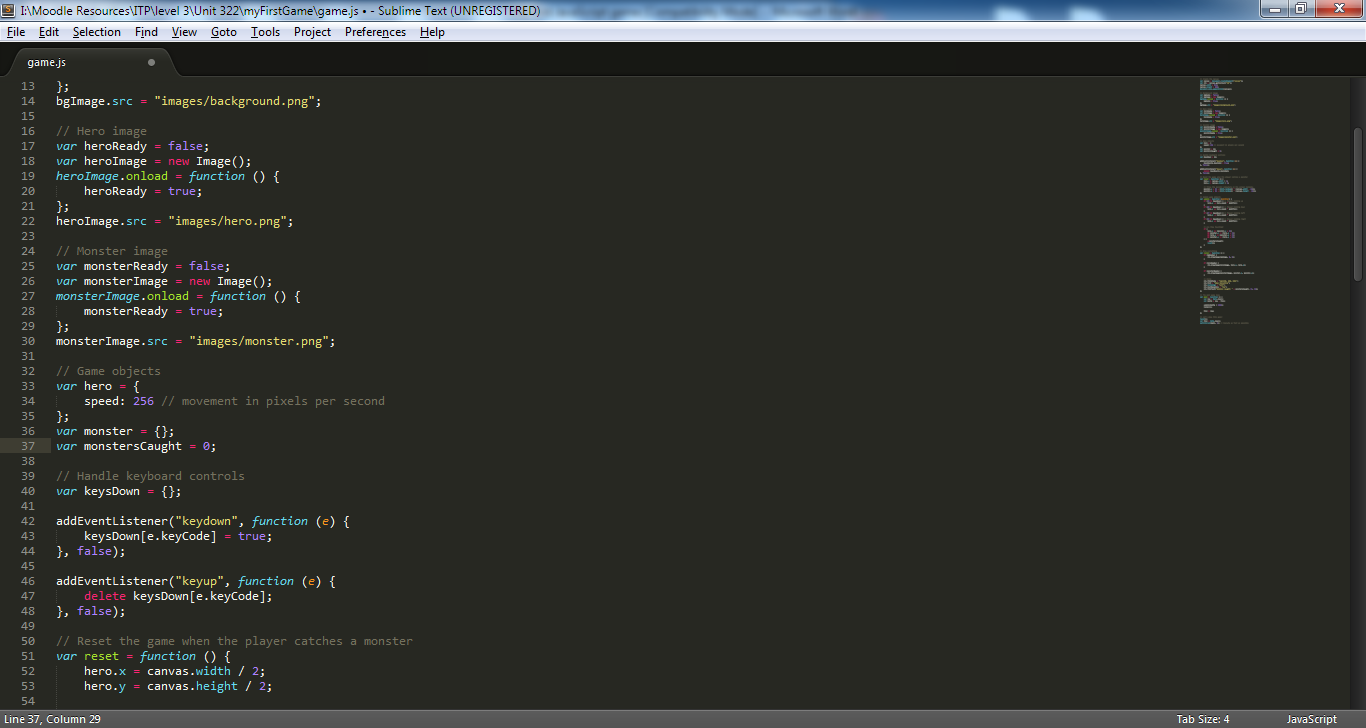
**Step 4 – Creating the game objects**

Underneath the creation of the monster we will now set up the objects which are to be used in the game. Since we are going to have a player and an enemy in the game we will need two objects, one for the player i.e. the hero and one for the enemy i.e. the monster. The code which we will use is created under the creation of the monster image as shown below:

Here we set up a variable called hero and tell it to become a list so that we can store properties within it. We have created an item called speed and told it to equal 256 e.g. pixel movement per second for the object

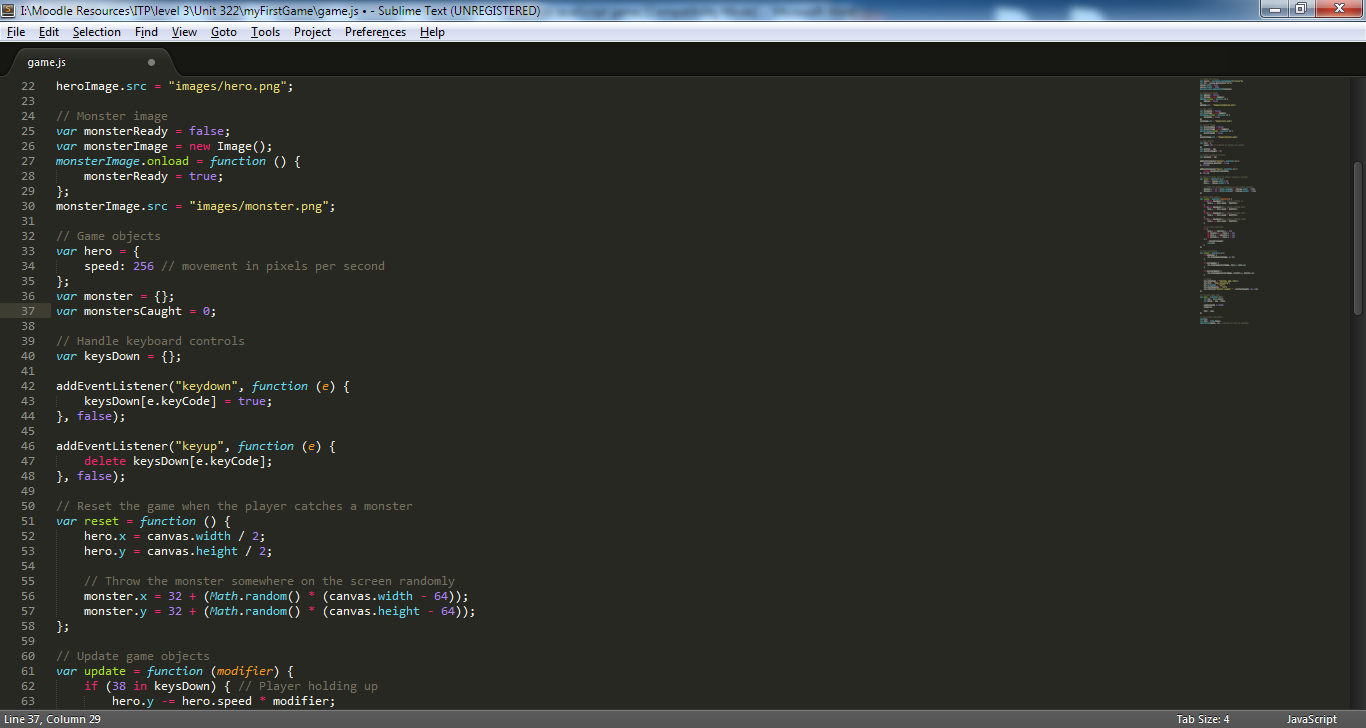
The monster has also been set up which currently has no properties about it

A variable called monstersCaught is set up and equal to 0. This variable will increment by one each time a monster is caught



**Step 5 – Setting up keyboard handler events**

The code after setting up the game objects shows how to set up keyboard handler events so that the keyboard can interact with the game. The below defines two event listeners, one for when the user presses down on a key and one for when the user releases the key. The key that the user pressed down is then stored inside an array called keysDown and is deleted once they release the key as shown:



Set up a new array called keysDown

Add the key code that the user pressed to the keysDown array once they press a key

Remove the key code pressed from the array once the user releases the key

**Step 6 – Resetting the game**

The code below resets the game by placing the player into the middle of the screen and the monster at a random position on the screen. The code should be typed underneath the key event listener code:

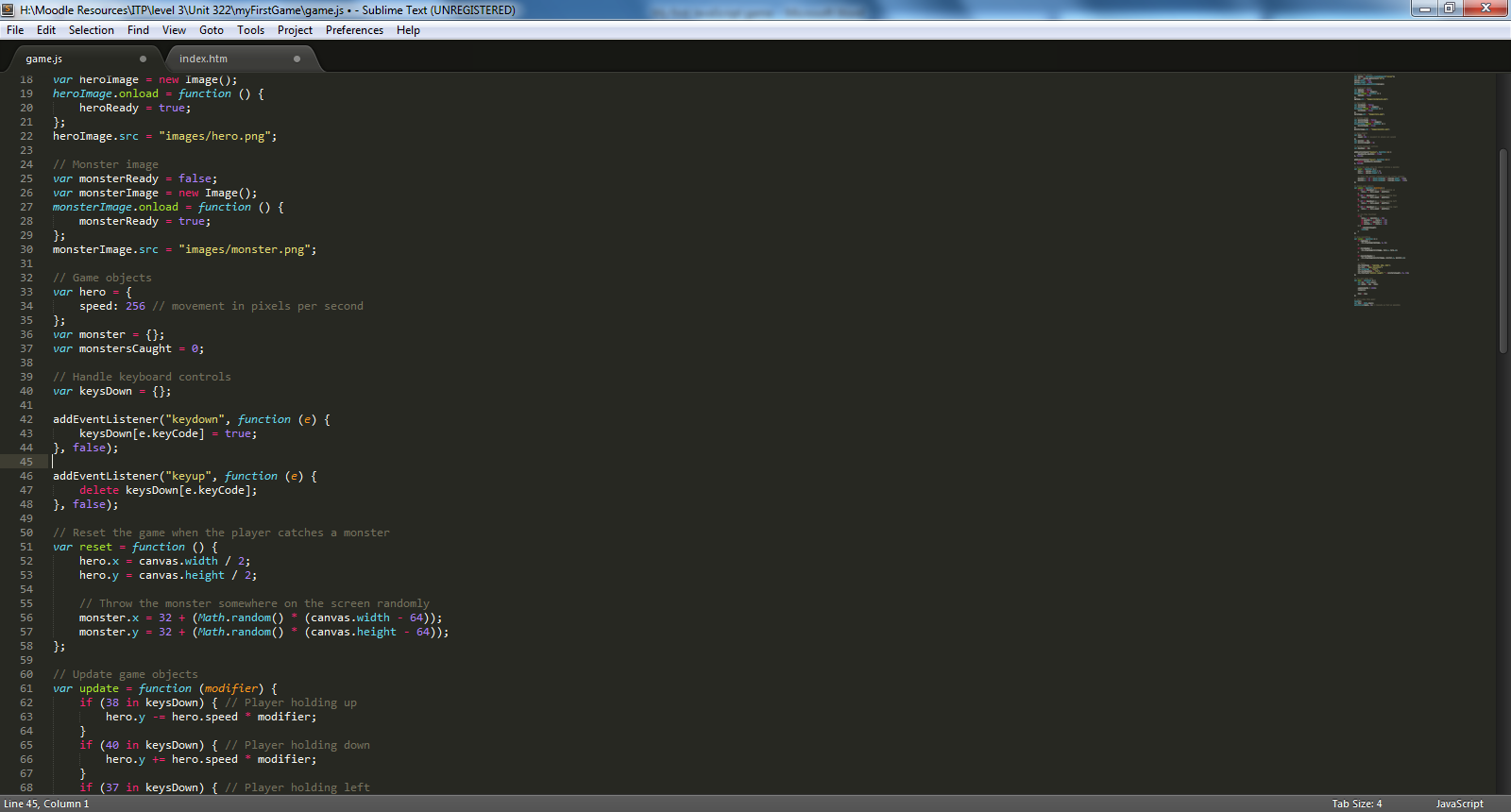
Set up a new function called reset

Tell the x coordinate of the player to be the width of the canvas divided by 2

Twell the y coordinate of the player to be the height of the canvas divided by 2

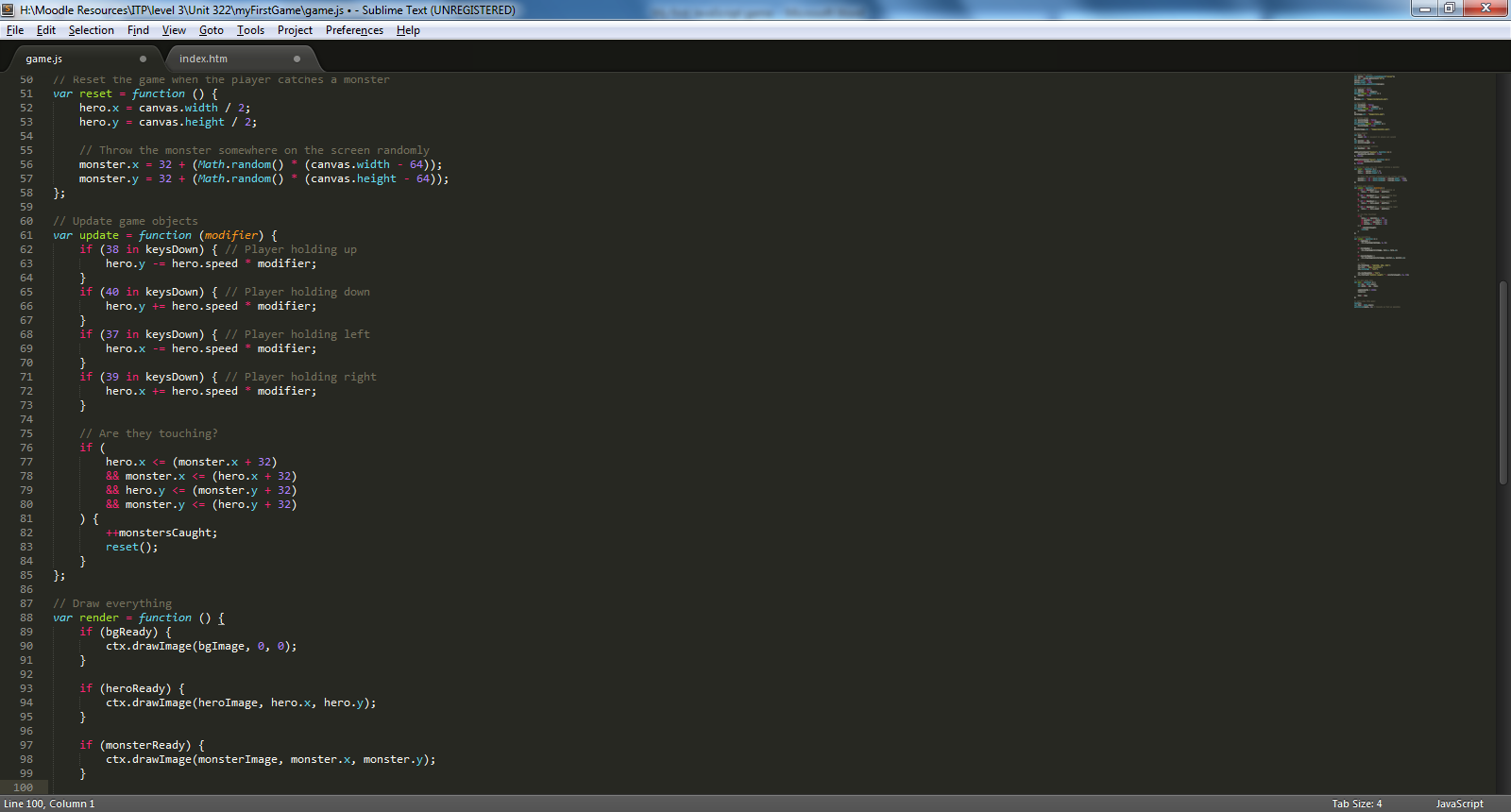
Tell the x and y coordinates of the enemy to equal to a random position on the canvas

Notice that the monster can only start from a minimum of 32 pixels from the left and 64 pixels from the right. This is so that it doesn’t hit the border



**Step 7 – Updating the players location**

The code below shows how to update the players’ location depending on the key they press on the keyboard. The player will move at their speed times by the modifier. The monsters location is also changed if they touch the player. The code for the update function is put underneath the reset code you have just created:



Update the players location on the x and y axis depending on the key that they press

If the monster is within range of the player then we tell monstersCaught to increment by one and the reset function is called to reset the game. Notice that the monster is caught if the image collides with any part of the players image i.e. is touching on the left, right, top or bottom or the monster. The second the two collide the if statement becomes try and the game resets

**Step 8 – Rending the images on the screen**

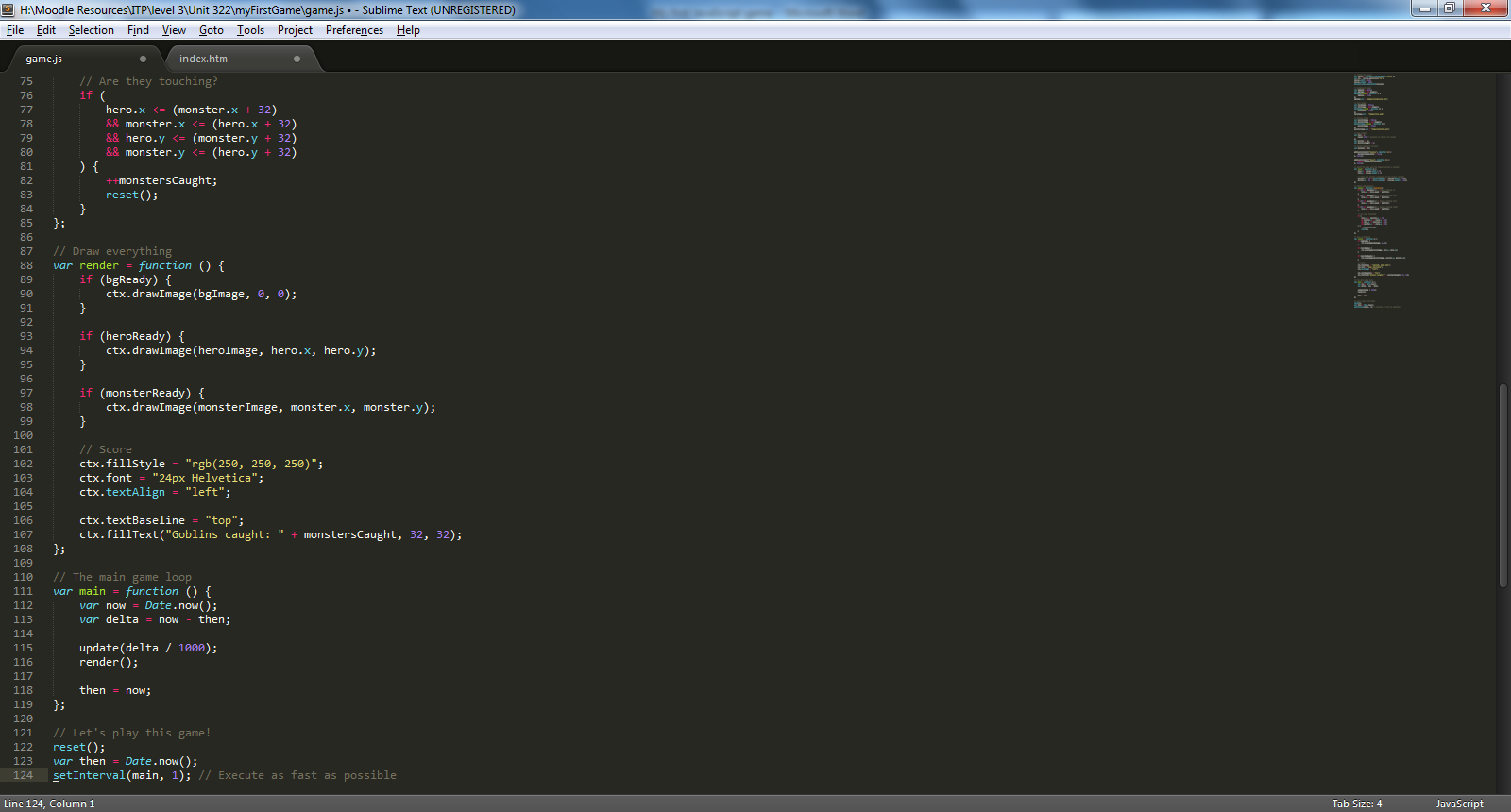
The screenshot below shows how to render the images drawn on the screen, by rendering the images it will change the location of the image depending on where the object is on the screen. The render code below should go below the update function:

Here we draw the background onto the screen if the background is ready

Here we draw the player onto the screen if the background is ready depending on the players current x and y coordinates

Here we draw the enemy onto the screen if the background is ready depending on the players current x and y coordinates

Here we select our font type, the side and write down how many monsters have currently been caught onto the canvas



**Step 9 – Setting up the main part of the game**

The code shown below demonstrates how to set up the main part of the game. This will cause the update function to be called every 1000 milliseconds so that it is constantly scanning for changes in the game. The render function is also called within the main part of the game to re-draw the objects locations

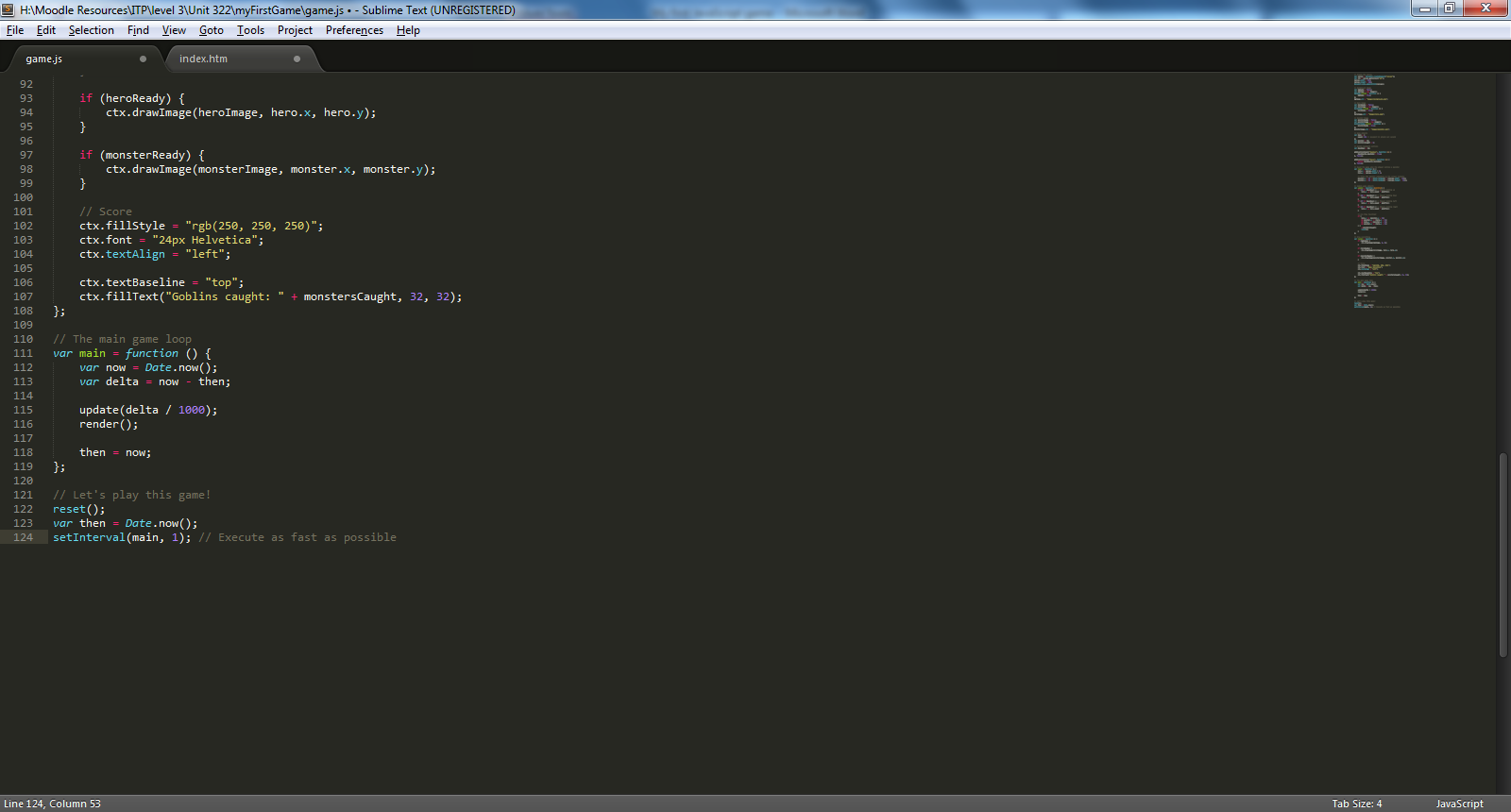
The code underneath the main will reset the game into it’s default state and call the main function to begin the game as shown below. The main code and the calling of it should always go at the bottom of the program as shown:

Set up the main part of the game

Get the current time difference between the time now and then in delta ticks

The update function is to be called at the amount of delta ticks we have divided by 1000 i.e. the update function is called really quickly in milliseconds

The code below shows that we reset the game when we first start it and tell the main code to be executed as soon as the code starts i.e. after 1 millisecond



**Step 10 – Gathering of the images**

Gather the images to use in your game and put them inside an images folder within your main game folder. These images are what will be used for when playing the game. You can get both background and character images from Google images if you would rather use your own images

**Scenario**

You have been asked to modify a game which has currently just been created to make it look more effective. The game will need to be changed in many different ways as the client it was originally created for has changed their need.

After talking to your line manager, it has been agreed that you will undertake this project and make any changes which are necessary to meet the clients need. Using your prior knowledge of developing objects on a canvas in JavaScript you have been asked to make the following changes:

1. Change the collision path so that the player cannot walk out of the canvas as currently the player can move out of the playing area
2. Create 2 other monsters which can be used in the game
3. Change the game so that different monsters are included in the game and that different scores are included for hitting the different monsters
4. Change the program so that multiple monster can be created at one time and not just one of each monster
5. Allow the monsters to move randomly around an area of the screen so that it makes it more challenging for the user to hit them

Once all changes have been implemented you will need to test your game and make sure that it noe works. If you get stuck with anything then do not hesitate to contact your tutor for help and support during the creation of your game.

Your deadline is the end of the day to get the game completed, changes made and for it to be fully working.