1 Introduction

If you look in your **vscode** tab, you will find three main files, *index.html*, *styles.css*, *snake.js* and *main.js*. For this project, we will be focussing on the **snake.js** file.

If you open up the **snake.js** file, you will find a whole bunch of code already written. This is the skeleton of the code that will make the snake in the snake game work. The snake is represented as a class, which is a programming feature that lets us create things, known as objects. A class contains functions that let us interact with our objects (in this case, our snake).

In the **snake.js** file you will find a class with a number of functions. A function looks like this:

```
myFunction() {
  let a = 2;
  let b = 3;

  return a + b;
}
```

There are five main functions that make up our **Snake**:

- **constructor()**: This is a function that gets run when we create our snake
- **move()**: This function will move our snake forwards one square
- **grow()**: This function will increase the size of our snake when it eats some food
- **draw()**: This function gets called to draw our snake onto our website canvas.
- **checkCollision()**: This function has been written for you, and checks to see if our snake has collided with anything, like food, walls, etc...

In this project we will be adding code to create, move and grow our snake. this will involve writing code for the **constructor()**, **move()**, **grow()** and **grow()** functions.

2 Creating the Snake

Open up the **snake.js** file and find the **constructor()** function. Here we will create the snakes body, and setup its initial direction.

Our snake will be made up of a line of 1x1 squares. We will create the snakes body by declaring a list of coordinates. Each coordinate will represent one of the squares of the snakes body. We can create a snake with one square in its body like so:

```
constructor() {
  this.body = [ { x: 1, y: 1 } ];
}
```

This will place the head of our snake at coordinates **(1, 1)**.

We also need to specify the snakes initial direction to travel in. There are four possible directions our snake could move in, **Up**, **Down**, **Left** and **Right**. We start our snake off by moving right like so:

```
constructor() {
  ...

  this.direction = Direction.Right;
}
```

Add the code for to create your snakes body and initial direction to the **constructor()** function. You constructor should now look something like this:

```
constructor() {
  this.body = [{ x: 1, y: 1 }];
  this.direction = Direction.Right;
}
```

3 Drawing the Snake

If you refresh the website and press the start button, you will see... nothing. That's because while our snake has been created now, we aren't **drawing** it onto our website.

We can draw shapes onto our website using the **context** variable. Here's an example of drawing a green square onto the website's canvas:

```
context.fillStyle = "green";
context.fillRect(x, y, width, height);
```

To draw our snake onto the website, we will be adding code to the **draw()** function. You can find this in the Snake class in the **snake.js** file. To start, we first need to specify what colour we want to draw with. For now we will just use the colour that has been setup: **snakeColor**.

```
draw() {
  context.fillStyle = snakeColor;
}
```

Remember the list of square(s) we created before that make up the snakes body? We now need to loop through each of these square and draw each one using **context**. We can do this using a **forEach** loop like this:

```
draw() {
  ...

  this.body.forEach(square => {
    let x = square.x * resolution;
    let y = square.y * resolution;

    context.fillRect(x, y, resolution, resolution);
  });
}
```

Note that we do a little maths when determining the **x** and **y** coordinates of each square since our game grid and the website canvas grid use different coordinates. **resolution** is the width of one grid square, hence we also use it as the width and height of each square.
Add the above code to set the colour of/draw each square onto our website. Your **draw()** function should now look like this:

```
draw() {
  context.fillStyle = snakeColor;

  this.body.forEach(square => {
    let x = square.x * resolution;
    let y = square.y * resolution;

    context.fillRect(x, y, resolution, resolution);
  });
}
```

If you refresh the game website and press start, you should now see at least 2 white squares appear on the game board. One of these will be your snake, and the other will be the piece of food that has been spawned now that a snake exists.

4 Moving the Snake

Now that we can see our snake, we want it to be able to move around the board. We will do this using the **move()** function. Open your **snake.js** file and look for the **move()** function. Every time this function gets run, we want to move our snake forwards one square.

We move each square individually into the position of the square in front of it, so that the snake slithers along the path laid out by the head. To accomplish this, we need to loop through the squares again — this time starting at the back and ending at the head. This is because if we start updating the position of the squares in the front, then the squares behind them won't know where to move to. Add a **for** loop to move the squares:

```
move() {
  for (let i = this.body.length - 1; i > 0; i--) {
    this.body[i] = { ...this.body[i - 1] };
  }
}
```

Now we want to move the head of the snake one square in the direction the snake is travelling in (remember the direction we specified in the **constructor()**?). In your **move()** function, below the **for** loop, add code to check which direction the snake is travelling in and move the position of the first square forwards in that direction:

```
move() {
  ...

  switch(this.direction) {
    case Direction.Up:
      this.body[0].y--;
      break;
    case Direction.Down:
      this.body[0].y++;
      break;
    case Direction.Left:
      this.body[0].x--;
      break;
    case Direction.Right:
      this.body[0].x++;
      break;
  }
}
```

Add the code to move the snake to your **move()** function in your **snake.js** file. The **move()** function should now look something like this:

```
move() {
  for (let i = this.body.length - 1; i > 0; i--) {
    this.body[i] = { ...this.body[i - 1] };
  }

  switch(this.direction) {
    case Direction.Up:
      this.body[0].y--;
      break;
    case Direction.Down:
      this.body[0].y++;
      break;
    case Direction.Left:
      this.body[0].x--;
      break;
    case Direction.Right:
      this.body[0].x++;
      break;
  }
}
```

Reload the website and you should find that your snake will now move across the game board! You should find that you can also use the **arrow keys** to control the direction of the snake.

5 Growing the Snake

You may notice that our snake doesn't grow when it eats some food. That's because we need to add code to our **grow()** function. All we need to do now is add another square to the end of our snakes body in the same position as the current tail of the snake. We can add squares to the end of our snake with the **.push()** function like so:

```
grow() {
  this.body.push({
    ...this.body[-1]
  });
}
```