COSMIC SPACE LOTTERY

Documentation

Stephane Emptage

Table Of Contents

Overview

This project focuses on designing and creating a working lottery game.

To create this project, I used the following:

- JavaScript
- HTML
- CSS
- Handlebars (templating language)
- Express
- Bash
- Node.js
- Bootstrap

I used the following developer tools throughout my time on the project:

- Visual Studio Code
- Chrome Dev Tools
- Git for Windows terminal
-

Business Requirements

The lottery game had the following requirements:

- Players can select 6 individual numbers from the range 1-59
- 6 balls are randomly drawn from a set of 59 balls
- Prizes are awarded if 3, 4 ,5, 6 balls match
- 3 matches = 50 points, 4 matches = 100 points, 5 matches = 200 points and 6 matches = 500 points
- Wins should be highlighted and detailed to the player
- Player has manual pick – where they can pick 6 numbers
- Player can choose lucky dip – where 6 random balls will be drawn for them
- There should be a button that starts the game (instructs the 6 balls to be drawn)
- There should also be a reset button

- Drawn numbers should be matched immediately with the picked or lucky dip numbers and prizes should be awarded based on points scored.

Initial Concept

Due to the time limit on this project I decided to use technology I was comfortable with and decided to use node.js and express to create my backend for the project. I chose these technologies as I was familiar with them and because they were quick to set up, meaning I could spend more time on the logic of the game.

I opted to have a space theme for the project and downloaded some backgrounds and assets to use. I also used the following site to create png text backgrounds which I could use for logos/buttons, this is a really useful site for basic text asset creation:

https://www.picturetopeople.org/text_generator/others/transparent/transparent-text-generator.html

I decided to use Handlebars in the project (which is a templating language). This was useful mainly so I could store links to my CSS files, JavaScript files and bootstrap all in one place.

Express and Handlebars Setup

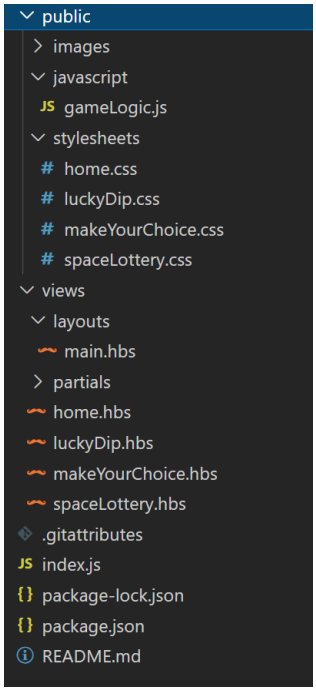
To create the backend, I first created a new project via GitHub and initialised the project. Next, I installed express into the project, using the command:

```
npm install express --save
```

Following this I installed Handlebars:

```
npm install express-handlebars --save
```

I created a folder structure and the files I would need to contain my code:



The first file I needed to create was the index.js file, which acts as the main code that handles the routing using node and allows the player to navigate to the different pages:

```
const express = require('express');
const exphbs = require('express-handlebars');
const app = express();
var path = require('path');

app.use(express.static(path.join(__dirname, 'public')));

// Configure template Engine and Main Template File
app.engine('hbs', exphbs({
    defaultLayout: 'main',
    extname: '.hbs'
}));
// Setting template Engine
app.set('view engine', 'hbs');

// routes
app.get('/', (req, res) => {
    res.render('home', { msg: 'This is home page'});
});
app.get('/spaceLottery', (req, res) => {
    res.render('spaceLottery');
});
app.get('/makeYourChoice', (req, res) => {
    res.render('makeYourChoice');
});
app.get('/luckyDip', (req, res) => {
    res.render('luckyDip');
});

// port where app is served
app.listen(3000, () => {
    console.log('The web server has started on port 3000');
});
```

Here, I basically set out the structure of how the server will function. I defined my extension name for handlebars, shortening it so that the filename is more readable than the standard .handlebars file extension. I set my routes and which views (handlebars file) should be rendered when the player visits specific pages. I also define the url which the game will be located on. In this case it will be located on localhost:3000.

Creating the Views

Setting up the ‘main’ view

I created a file called ‘main.hbs’ in my layouts sections within my views folder. I created this file so that for every view (or page) the player interreacts with, the information in this file would always be used on each page. This was useful to hold important references to various files, such as my stylesheets, Bootstrap and my JavaScript file. This falls in line with DRY principles (Do not repeat yourself), as it means this code does not have to be re-written on each new view.

Creating the ‘home’ view

The ‘home’ view is the landing page for the site. The HTML is fairly simple as I am just linking a few images and applying classes to them. A lot of the work around size, placement and interactivity are actually handled in the CSS file. The ‘home’ view looks like this:

```
<div class="logo"> <img class="logo" src="/images/logos/live5.png"></div>
  <div class="introducingTitle"> <img src="/images/logos/introducing.png"></div>
  <div class="newGameTitle"> <img src="/images/logos/aNewGame.png"></div>
  <div class="gameTitle"> <img src="/images/logos/slogan.png"></div>
  <a href="/makeYourChoice">
 <div class= "loginbutton">
    <button class="button" id="loginbutton"></button>
  </div>
</a>
```

The final results of the ‘home’ page is below:

The start button is interactive and changes when hovered:

Creating the ‘makeYourChoice’ view

I wanted the player to be able to select whether they wanted to play Lucky Dip or pick their own balls for the game, to do this I created a page that would act as a portal to the two game types. Similarly, I created minimal HTML and CSS did a lot of the heavy lifting in terms of button hover:

```
<div class="logo"> <img class="logo" src="/images/logos/live5.png"></div>

<div class="container">
    <div class="pickAChoice"> <img class="pickAChoice" src="/images/logos/yourChoice.png"></div>
    <div class="enterTheCosmos"> <img class="enterTheCosmos" src="/images/logos/enterTheCosmos.png"></div>
    <div class="row">
    <div class="col">
    <a href="/spaceLottery">
        <div class= "yourChoice">
            <button class="button" id="yourChoice"></button>
        </div>
    </a>
    </div>
        <div class="col">
            <a href="/luckyDip">
                <div class= "luckyDipChoice">
                    <button class="button" id="luckyDipChoice"></button>
                </div>
            </a>
        </div>
    </div>
</div>
```

Both the ‘pick your own’ and ‘lucky dip’ buttons changed when hovered over. I made the buttons circular to represent balls or potentially planets or suns. If I had more time, I would likely tweak these buttons to make them more visually appealing and in line with the space theme.

Creating the ‘luckyDip’ view

This page was somewhat more complex so I will divide up the HTML and discuss it in parts.

This page renders the Lucky Dip page. To start with I created various logos, a button to start the Lucky Dip and a table to hold the Lucky Dip results:

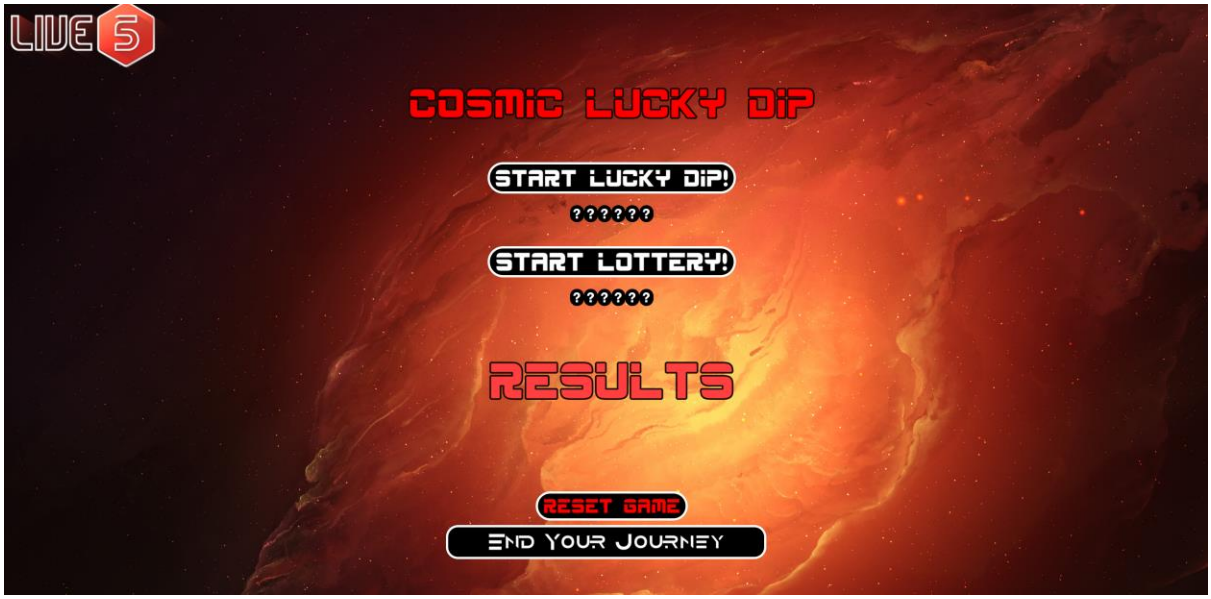
```
<div class="logo"> <img class="logo" src="/images/logos/live5.png"></div>
<div class="luckyDipLogo"> <img class="luckyDipLogo" src="/images/logos/luckyDip.png"></div>
<br>

<div id="table1">
<table>
<tr>
<div class= "startDip">
    <input type="button" class="button" id="startDip" onclick="luckyDip()"></button>
  </div>
  <br>
<tr>
<th width="10%" id="lucky0" class="numberResults">?</th>
<th width="10%"id="lucky1" class="numberResults">?</th>
<th width="10%"id="lucky2" class="numberResults">?</th>
<th width="10%"id="lucky3" class="numberResults">?</th>
<th width="10%"id="lucky4" class="numberResults">?</th>
<th width="10%"id="lucky5" class="numberResults">?</th>
</tr>
</div>
```

The table will be populated by a function within my JavaScript file which I will detail later on.

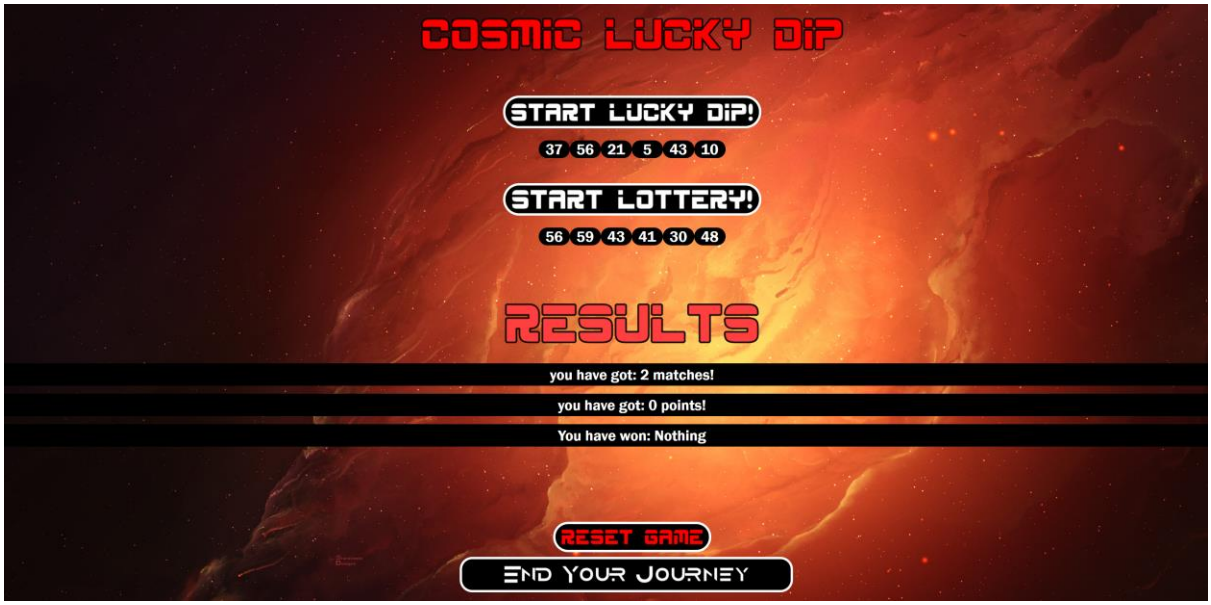
The next portion of HTML has a second table which is for the lottery, which will similarly have data applied when the ‘Start Lottery’ button is pressed.

```
<table id="table2">
<tr>
    <div class= "startLottery">
      <input type="button" class="button" id="startLottery" onclick="lotteryLuckyDip()"></button>
    </div>
    <br>
<tr>
<th width="10%" id="lotto0" class="numberResults">?</th>
<th width="10%"id="lotto1" class="numberResults">?</th>
<th width="10%"id="lotto2" class="numberResults">?</th>
<th width="10%"id="lotto3" class="numberResults">?</th>
<th width="10%"id="lotto4" class="numberResults">?</th>
<th width="10%"id="lotto5" class="numberResults">?</th>
</tr>
<table>

<div class="yourResults"> <img class="yourResults" src="/images/logos/results.png"></div>
<br>

<p id="result"></p>
<p id="points"></p>
<p id="prize"></p>

<a href="/luckyDip">
<div class= "resetGame">
    <button class="button" id="resetGame"></button>
</div>
</a>
<br>
<a href="/">
    <div class= "logoutbutton">
        <button class="button" id="logoutbutton"></button>
    </div>
</a>
```

I also have 3 paragraphs that are populated with results when the ‘Start Lottery’ button is pressed. These are populated with the result, the points gained and the prize gained. I also created a reset button, which simply refreshes the page and reset the game. The player can also return to the home page by clicking on the logout button. The final page looks like this pre-game:



And if the player has played a game, there results will show on the page:



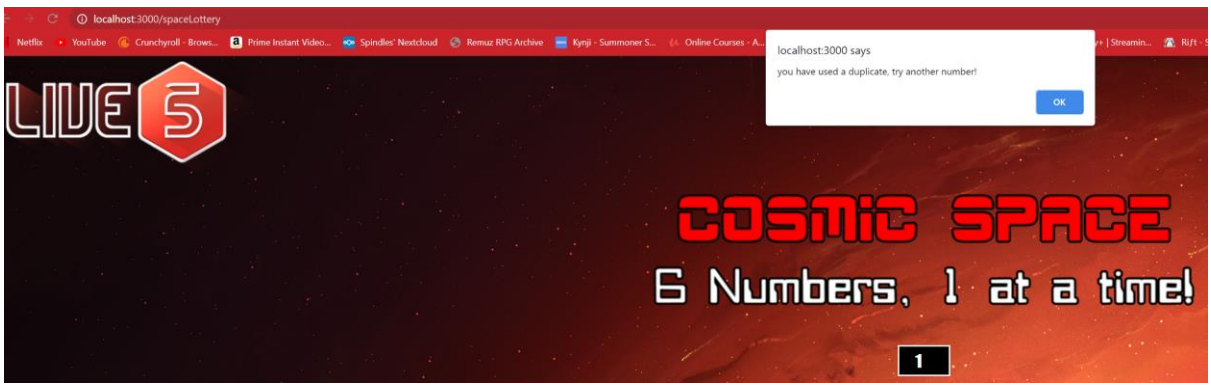
Creating the ‘spaceLottery’ view (or the pick your own page)

This page is fairly similar to the Lucky Dip page with a few differences. To start off with I created my logos and then created a user input so that the player could insert their number choices, followed by a table that would store their choices:

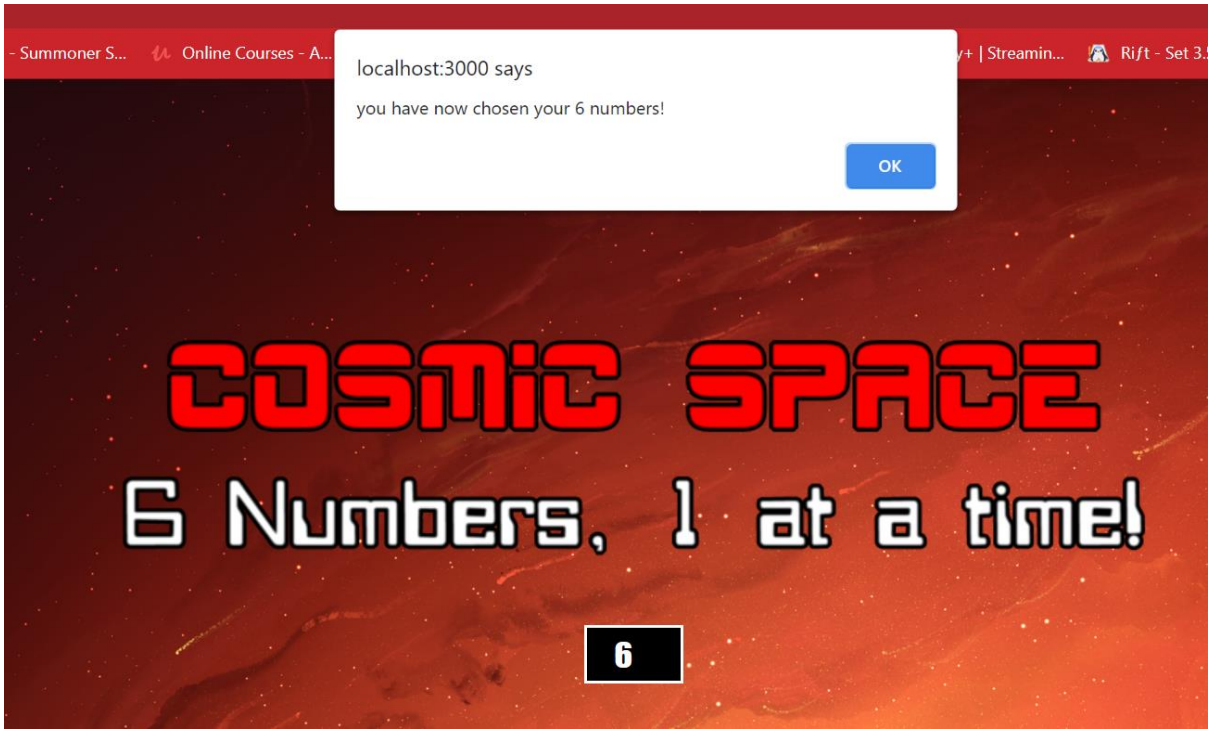
```
<div class="logo"> <img class="logo" src="/images/logos/live5.png"></div>
<div class="cosmicSpaceLogo"> <img class="cosmicSpaceLogo" src="/images/logos/cosmicSpace.png"></div>
<div class="chooseNumbers"> <img class="chooseNumbers" src="/images/logos/chooseNumbers.png"></div>
<br>

<div class="userinput">
  <input type="number" id="userinput" min="1" max= "59" required="yes"/>
<div>

<div class= "pickANumber">
  <input type="button" class="button" id="pickANumber" onclick="addTo()"></button>
</div>

<table id="table0">
<tr>
<br>
<tr>
<th width="10%" id="pick0" class="numberResults">?</th>
<th width="10%"id="pick1" class="numberResults">?</th>
<th width="10%"id="pick2" class="numberResults">?</th>
<th width="10%"id="pick3" class="numberResults">?</th>
<th width="10%"id="pick4" class="numberResults">?</th>
<th width="10%"id="pick5" class="numberResults">?</th>
</tr>
</table>
```

I have several JavaScript functions that handle this process which I will detail later.

I also wanted to make sure the user could not input the same number every time. I explain this in greater depth when detailing my functions, but essentially if a user picks a duplicate, it is not added to their choices and they receive an alert:



I also created an alert for when they have chosen their 6 numbers:



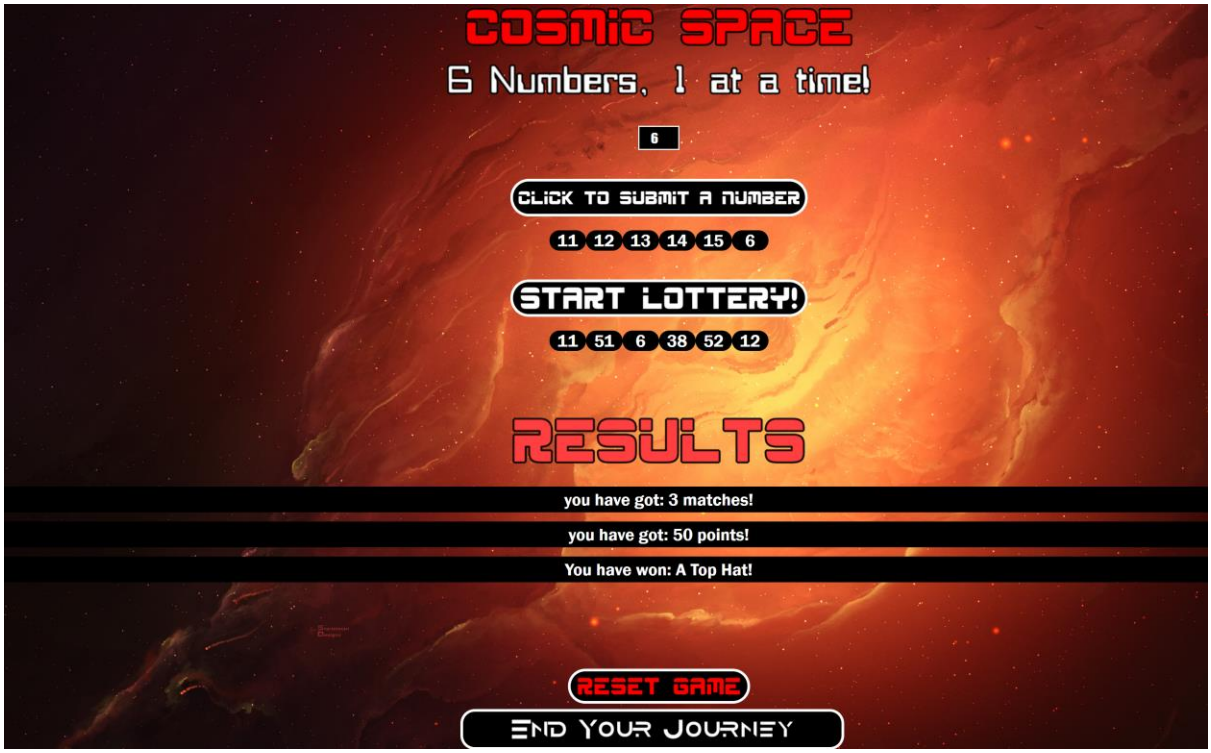
Once this has been pressed the table is filled with the results:

I decided to give the table entries round backgrounds to make them look like balls. If I had more time, I would look for another solution, as when the balls reach double figures they change in shape:

The rest of the page is the same as the Lucky Dip page:

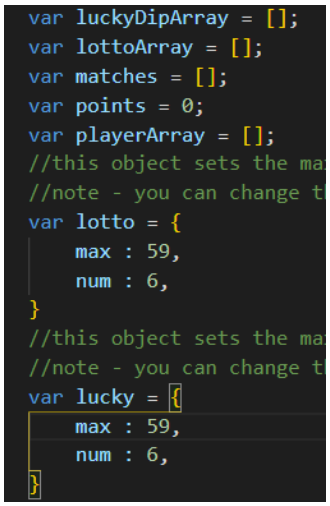
```
<table id="table2">
<tr>
    <div class= "startLottery">
      <input type="button" class="button" id="startLottery" onclick="lotteryPlayerChoice()"></button>
    </div>
  <br>
<tr>
<th width="10%" id="lotto0" class="numberResults">?</th>
<th width="10%"id="lotto1" class="numberResults">?</th>
<th width="10%"id="lotto2" class="numberResults">?</th>
<th width="10%"id="lotto3" class="numberResults">?</th>
<th width="10%"id="lotto4" class="numberResults">?</th>
<th width="10%"id="lotto5" class="numberResults">?</th>
</tr>
<table>

<div class="yourResults"> <img class="yourResults" src="/images/logos/results.png"></div>
<br>
<p id="result"></p>
<p id="points"></p>
<p id="prize"></p>

<a href="/spaceLottery">
<div class= "resetGame">
      <button class="button" id="resetGame"></button>
    </div>
</a>
<br>
<a href="/">
  <div class= "logoutbutton">
    <button class="button" id="logoutbutton"></button>
  </div>
```

And results display in the same way:


COSMIC SPACE
6 Numbers, 1 at a time!
6
CLICK TO SUBMIT A NUMBER
11 12 13 14 15 6
START LOTTERY!
11 51 6 38 52 12
RESULTS
you have got: 3 matches!
you have got: 50 points!
You have won: A Top Hat!
RESET GAME
END YOUR JOURNEY

Creating the game logic in the ‘gameLogic.js’ file

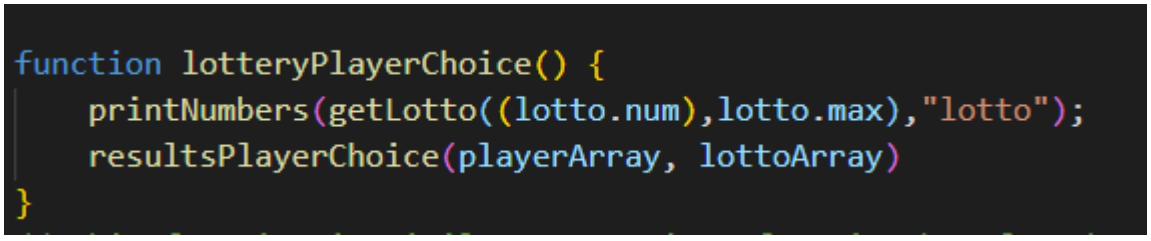
This file contained my main game logic and was where most of the complex code resides. I have various variables available outside of functions so that results from functions can then be used in other functions as seen below:



I created two objects ‘lotto’ and ‘lucky’. These were used to contained the max number of balls and the number of balls a player or the lucky dip could pick. These were useful as objects from a developer perspective, as it meant that I could change the max ball number easily, which made debugging a lot easier during the project, as seeing smaller max ball numbers was more useful when testing whether the features worked.

Next, I will break down what the individual functions do.

Creating the ‘lotteryPlayerChoice’, ‘lotteryLuckyDip’ and ‘luckyDip’ functions



This function is initialised when the player is on the 'pick your own' page. It is run when the player clicks on the 'Start Lottery' button. It is referenced in the HTML here:

```
<div class= "startLottery">
  <input type="button" class="button" id="startLottery" onclick="lotteryPlayerChoice()"></button>
</div>
```

This function runs two functions when initialised. First it runs the 'printNumbers' function. Which holds the mechanics of applying number picks to the specified table. I then pass the 'getLotto' function which manages the process of randomising lottery numbers.

```
printNumbers(getLotto((lotto.num),lotto.max),"lotto");
```

I then apply a reference to 'lotto.num' and 'lotto.max' which references my 'lotto' object and pulls in the max number of balls and the number of balls to pick. I then reference which table should be filled with the results. In this instance it is the lottery table.

Following this the 'resultsPlayerChoice' function is run:

```
resultsPlayerChoice(playerArray, lottoArray)
```

The 'resultsPlayerChoice' function handles the process of comparing matches and applying points and prizes. In this instance I pass it to my **playerArray** which is an array that hold the player chosen balls and I also pass it the **lottoArray**, which holds the lottery results so that the function knows which data to compare. This is then rendered on the page when the function is run.

The 'lotteryLuckyDip' and 'luckyDip' functions work in very much the same way so I won't explain these in great depth:

```
function lotteryLuckyDip(){
    printNumbers(getLotto((lotto.num),lotto.max),"lotto");
    resultsLuckyDip(luckyDipArray, lottoArray);
}
```

```
function luckyDip() {
    printNumbers(getLucky((lucky.num),lucky.max),"lucky");

}
```

Creating the ‘printNumbers’ and ‘printNumbersPlayer’ functions

The ‘printNumbers’ function takes two parameters, the numbers (which will be the maximum balls and the number of balls) and the type. The type relates to the reference location where the data will be stored. This function basically provides the mechanics of applying the **lotto** and **lucky** objects to their respective tables, as seen previously in the ‘lotteryPlayerChoice’ and ‘lotteryLuckyDip’ functions.

```
function printNumbers(numbers,type){
    for(var x in numbers){
        document.getElementById(type+x).innerHTML = numbers[x];
    }

}
```

Next there is the ‘printNumbersPlayer’ function. This function was made to transplant data from the **playerArray** and insert it into the player table. Essentially populating the player ball picks into the player table.

Creating the ‘getLotto’ and ‘getLucky’ functions

Both these functions essentially do the same thing so I will start off with explaining the ‘getLotto’ function:

```
function getLotto(totalBalls,balls) {
    var lottoNumbers = [];
    lottoArray = [];

    for (var i = balls; i > 0; i--){
        lottoNumbers.push(i);
    }
    lottoNumbers.sort(
        function(){
            return (Math.round(Math.random())-0.5);
        }
    );
    lottoArray = lottoNumbers.slice(0, totalBalls);

    return lottoNumbers.slice(0,totalBalls);
}
```

This function manages the process of randomising the lottery results. It pushes the maximum number of balls into the **lottoNumbers** array and then randomises the numbers. Following this I then slice the **lottoNumbers** array from 0 to the totalBalls (which would be 6) and insert it into a new array called **lottoArray**, which I can then use later. I then return the **lottoNumbers** array similarly sliced so it can be used by other functions, ready to be used to populate the lotto table.

The ‘getLucky’ function does essentially the same thing but for Lucky Dip numbers so I won’t detail this:

```
function getLucky(totalBalls,balls) {
    var luckyNumbers = [];
    luckyDipArray = [];
    for (var i = balls; i > 0; i--){
        luckyNumbers.push(i);
    }
    luckyNumbers.sort(
        function(){
            return (Math.round(Math.random())-0.5);
        }
    );
    luckyDipArray = luckyNumbers.slice(0,totalBalls);
    return luckyNumbers.slice(0,totalBalls);

}
```

Creating the ‘resultsLuckyDip’ and ‘resultsPlayerChoice’ functions

First, I will start with the ‘resultsLuckyDip’ function. I will break this down into sections as the function is quite long. In hindsight if I had more time, I would likely break this function into several smaller functions so it is less complicated.

```
function resultsLuckyDip(luckyDipArray, lottoArray){
    //finds matches between lucky dip and lotto
    var matches = luckyDipArray.filter(function(item){
        return lottoArray.indexOf(item) > -1
    })
    // concatonation of number of matches and applies to result element
    document.getElementById("result").innerHTML = "you have got: " + matches.length + " matches!";
    // switch statement to manage points based on number of matches
    switch(true){
      case matches.length == 3:
      points = points + 50;
      break;
      case matches.length == 4:
      points = points + 100;
      break;
      case matches.length == 5:
      points = points + 200;
      break;
      case matches.length == 6:
      points = points + 500;
      break;
    }
```

First, I create a variable (**matches**) to hold my matches. This variable manages the process of finding the matches between **luckyDipArray** and **lottoArray**. Following this I reference my “result” id and I print out the number of matches onto the webpage inside a concatenated string. The ‘result’ id is a reference to my result paragraph on both my ‘luckyDip’ and ‘spaceLottery’ pages:

```
<p id="result"></p>
```

Following this I created some switch statements. These were created to apply points if the matching numbers were 3 or more. I finish these statements off by referencing my points id (which is a paragraph which exists on my ‘luckyDip’ and ‘spaceLottery’ pages similar to the result id paragraph):

```
//presents points accumulated to the player
  document.getElementById("points").innerHTML = "you have got: " + points + " points!";
```

I create several more switch statements to handle prize allocation as seen below:

```
switch(true){
    case points >= 50 && points <= 99:
    prize = "A Top Hat!";
    break;
    case points >= 100 && points <= 199:
    prize = "A Rolex Watch!";
    break;
    case points >= 200 && points <= 499:
    prize = "A BMW!";
    break;
    case points >= 500:
    prize = "A Package Holiday to Ibiza!";
    break;
    default:
    prize = "Nothing";

}
// presents attained prizes to player
document.getElementById("prize").innerHTML = "You have won: " + prize;
```

I made sure to set a point cap to each prize apart from the final prize, to ensure a prize would always be allocated if points were gained and that the prizes would not override one another.

The ‘resultsPlayerChoice’ function is basically the same as the ‘resultsLuckyDip’ function so I won’t explain this in great depth:

```
function resultsPlayerChoice(playerArray, lottoArray){
 var matches = playerArray.filter(function(item){
    return lottoArray.indexOf(item) > -1
  })
  document.getElementById("result").innerHTML = "you have got: " + matches.length + " matches!";
  //switch statements for points
  switch(true){
    case matches.length == 3:
    points = points + 50;
    break;
    case matches.length == 4:
    points = points + 100;
    break;
    case matches.length == 5:
    points = points + 200;
    break;
    case matches.length == 6:
    points = points + 500;
    break;

  }

  document.getElementById("points").innerHTML = "you have got: " + points + " points!";
```

```
    document.getElementById("points").innerHTML = "you have got: " + points + " points!";
  //switch statements for prizes
    switch(true){
      case points >= 50 && points <= 99:
      prize = "A Top Hat!";
      break;
      case points >= 100 && points <= 199:
      prize = "A Rolex Watch!";
      break;
      case points >= 200 && points <= 499:
      prize = "A BMW!";
      break;
      case points >= 500:
      prize = "A Package Holiday to Ibiza!";
      break;
      default:
      prize = "Nothing";
  }
  document.getElementById("prize").innerHTML = "You have won: " + prize;
}
```

Creating the ‘addTo’ function

This function basically manages the process of adding player choices to the **playerArray**, to then be used in other functions. It also manages several alerts to inform the player.

```
function addTo() {
//had to parseInt so that player choices were converted to integars to be com
    playerChoice = parseInt(document.getElementById("userinput").value);
    //created to manage situations where the player may input a duplicate opt
    if (playerArray.length < 6 && playerArray.indexOf(playerChoice) === -1){
        playerArray.push(playerChoice);
    //highlights with an alert when the player has chosen all 6 numbers
    if(playerArray.length == 6){
        printNumbersPlayer()
        alert("you have now chosen your 6 numbers!");
    }

    return false;
    }
    //this else statement posts an alert if a duplicate has been used
    else {
        alert("you have used a duplicate, try another number!");
        return false;
    }
}
```

I first specify where the data will be retrieved from, which will be the ‘userinput’ id, which exists on the ‘spaceLottery’ page. This is the input box where the player can pick a number:

```
<div class="userinput">
  <input type="number" id="userinput" min="1" max= "59" required="yes"/>
<div>
```

I have to specify that I want the result to be an integer and I do this using ‘parseInt’:

```
playerChoice = parseInt(document.getElementById("userinput").value);
```

This is because the data I retrieved initially was a string and could not be compared with matches, as it was trying to match a string to an integer so the values would always not match. By using ‘parseInt’ I was able to convert the strings to integer so that matches could be correctly compared.

Following this I created several if statements.

```
if (playerArray.length < 6 && playerArray.indexOf(playerChoice) === -1){
    playerArray.push(playerChoice);
//highlights with an alert when the player has chosen all 6 numbers
if(playerArray.length == 6){
    printNumbersPlayer()
    alert("you have now chosen your 6 numbers!");
}

return false;
}
//this else statement posts an alert if a duplicate has been used
else {
    alert("you have used a duplicate, try another number!");
    return false;
}
```

These handled the process of several things. First, it checks if the **playerArray** length is less than 6 and that the player has not chosen a duplicate. It then pushes 'playerChoice' (our user input) into the **playerArray**.

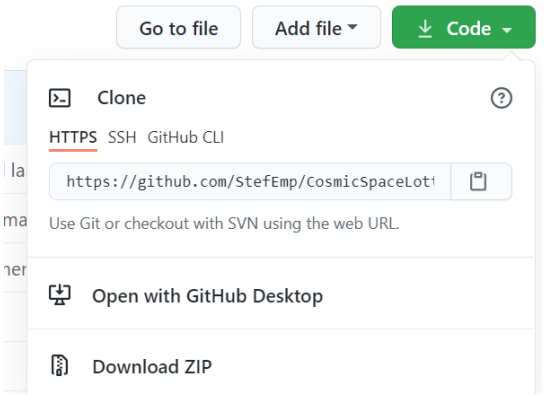
Following this I created an if statement that applies when the **playerArray** has 6 elements. If this is the case, then the numbers will be printed into the table so that the player can see the balls they have chosen. An alert will also pop up to tell them they have chosen all 6 balls.

I created an else statement to handle situations where a duplicate is applied.

Project Setup – setting the project up on your own PC

Firstly, make sure you have node.js installed. You can download it from here: https://nodejs.org/en/download/

Next, clone the project to your PC:



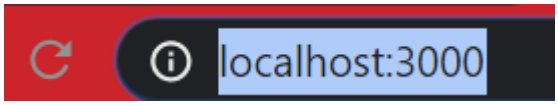
Once you have the project downloaded, open the project using VS Code or your IDE of choice. Now that the project is open an initial step must be taken to make sure the project will run. Simply open the terminal and type:

```
Stef@DESKTOP-8N7STNV MINGW64 /c/Projects/SpaceLottery (main)
$ npm install
```

This will install all the dependencies needed to run the project. Then to run the project simply type:

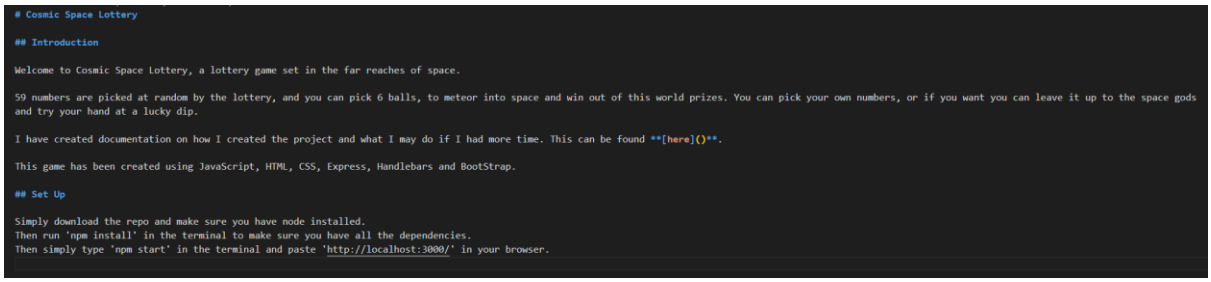
```
Stef@DESKTOP-8N7STNV MINGW64 /c/Projects/SpaceLottery (main)
$ node index
```

Then go to your browser and type:



You should now be able to navigate and use the application through your browser:

I also included setup documentation in the readme file using markdown:



```
# Cosmic Space Lottery

## Introduction

Welcome to Cosmic Space Lottery, a lottery game set in the far reaches of space.

59 numbers are picked at random by the lottery, and you can pick 6 balls, to meteor into space and win out of this world prizes. You can pick your own numbers, or if you want you can leave it up to the space gods
and try your hand at a lucky dip.

I have created documentation on how I created the project and what I may do if I had more time. This can be found **[here]()**.

This game has been created using JavaScript, HTML, CSS, Express, Handlebars and BootStrap.

## Set Up

Simply download the repo and make sure you have node installed.
Then run 'npm install' in the terminal to make sure you have all the dependencies.
Then simply type 'npm start' in the terminal and paste 'http://localhost:3000/' in your browser.
```

Conclusion

This was an interesting project to make. I enjoyed the process of figuring out the different functions as well as creating the front end. If I had more time, I would have spent more time on the front end and added more to the visual style. It would have been nice to have animated balls appear when the player picked a ball and perhaps some animations that related to the player gaining a prize and winning, to provide more visual feedback. Due to the time limitations, I also did not create tests for this project. Given more time I would have incorporated tests using Cypress.io to test the front-end usability. Other areas I would be keen to develop would be to host the game on Heroku.

There were are few interesting road blocks, such as my user input providing string values, which took a little time to debug and understand why I could not compare matches initially (the answer being to use parseInt to convert the input data into integers). I also enjoyed the process of figuring out how to retrieve player inputs and populate the **playerArray**. This was quite challenging at first and it took some time to figure out how to populate the player table with the results, but I was keen that this was coded in, as I felt from a player perspective, it would be confusing to input 6 numbers and not see the numbers you had entered.
The only problem with how the player choices are presented is that they are displayed after they have submitted all 6 numbers. If I had more time, I would change this, so that each time a number is entered, it would display in the table below. I did get this working at one point, however it would display the number entered and then each table element would display as ‘undefined’ which looked buggy. If I had more time, I would have debugged this issue, but due to time constraints I felt the best approach would be to display the results when all player number entries had been complete.

Another final improvement I would make, would be to make the webpage responsive. The page isn’t currently mobile friendly and was designed on 4k and 2k monitors.

Overall, I really enjoyed the project, it was a good challenge and I was pleased I could hit the technical requirements in the time provided.