



Intro to JavaScript Week 6 Coding Assignment

Points possible: 100

URL to GitHub Repository:

https://github.com/lbell/bootcamp_wk6_coding_assignment

URL to Your Coding Assignment Video:

<https://drive.google.com/file/d/1SX-gltKa0jFWrFZ0g-2HgeXjXgnD1gHh/view?usp=sharing>

Instructions: In Visual Studio Code, or an IDE of your choice, write the code that accomplishes the objectives listed below. Ensure that the code compiles and runs as directed. Take screenshots of the code and of the running program (make sure to get screenshots of all required functionality) and paste them in this document where instructed below. Create a new repository on GitHub for this week's assignments and push this document, with your JavaScript project code, to the repository. Add the URL for this week's repository to this document where instructed and submit this document to your instructor when complete.

Coding Steps:

For the final project you will be creating an automated version of the classic card game *WAR*. You do not need to accept any user input, when you run your code, the entire game should play out instantly without any user input.

There are many versions of the game *WAR*, but in this version there are only 2 players and you don't need to do anything special when there is a tie on a round.

Think about how you would build this project and write your plan down. Consider classes such as Card, Deck, and Player and what fields and methods they might each have. You can implement the game however you'd like (i.e. printing to the console, using alert, or some other way). The completed project should, when run, do the following:

- Deal 26 Cards to two Players from a Deck.
- Iterate through the turns where each Player plays a Card
- The Player who played the higher card is awarded a point
 - o Ties result in zero points for both Players
- After all cards have been played, display the score and declare the winner.

Write a Unit Test using Mocha and Chai for at least one of the functions you write.



Video Steps:

Create a video, up to five minutes max, showing and explaining how your project works with an emphasis on the portions you contributed. This video should be done using screen share and voice over. This can easily be done using Zoom, although you don't have to use Zoom, it's just what we recommend. You can create a new meeting, start screen sharing, and start recording. This will create a video recording on your computer. This should then be uploaded to a publicly accessible site, such as YouTube, Dropbox, or Google Drive. **MAKE SURE THE LINK YOU SHARE IS PUBLIC or UNLISTED.** If it is not accessible by your grader, your project will be graded based on what they can access. The link should be pasted in the submission text box after the GitHub repo link. **REQUIRED: PUBLIC link to video, and GitHub repo link with everything listed above!**

Screenshots of Code:

S

```
JS index.js > buildDeck
1 // Set up the scoreboard
2 let player1Score = 0;
3 let player2Score = 0;
4
5 // Gather the cards
6 let deck = buildDeck();
7
8 // Shuffle the cards
9 deck = shuffle(deck);
10
11 // Split deck in half between players (26 cards to each). Since the deck is
12 // randomly shuffled, no need to deal 1x1
13 let player1Stack = deck.slice(0, 26);
14 let player2Stack = deck.slice(26);
15
16 // Play the game
17 playGame(player1Stack, player2Stack);
18
19 /**
20  * Builds a deck of 52 cards - no suites needed, so keeping things simple.
21  * Of course simpler would be to just hard-code the array of values, but what's
22  * the fun in that?
23  *
24  * @returns deck of 52 cards
25  */
26 function buildDeck() {
27   let output = [];
28
29   // Need 4 suites of cards, so loop through 4 times
30   for (i = 0; i < 4; i++) {
31     // build an array 1-13 (well, 0-14, then hack off the 0). .keys() gets the
32     // keys (indexes) and ditches the values (undefined).
33     let suite = [...Array(14).keys()].slice(1);
34
35     // tack on the new suite with the old one
36     output = [...output, ...suite];
37   }
38   return output;
39 }
40
41 /**
42  * Elegant fisher-Yates shuffling algorithm found here: https://stackoverflow.com/a/2450976/1061836
43  * With an excellent explainer: https://bost.ocks.org/mike/shuffle/
44  *
45  * Walks through the array and pulls a random index number and swaps it with the
46  * current index number.
47  *
48  * @param {arr} array Array to shuffle
49  * @returns shuffled array
50  */
51
```



PROMINEO TECH

```
91 /**
92  * Loops through each player's hand 26 times, calculates the score, and outputs
93  * the running totals.
94  *
95  * @param {arr} player1Stack All of player 1's cards
96  * @param {arr} player2Stack All of player 2's cards
97  */
98 function playGame(player1Stack, player2Stack) {
99   let turns = 0;
100   let winner = "";
101
102   while (turns < 26) {
103     console.log("\n");
104
105     console.log(`--- Round ${turns + 1}! ---`);
106
107     console.log(`Player 1 card: ${player1Stack[turns]} | Player 2 card: ${player2Stack[turns]}`);
108     calculateScore(player1Stack[turns], player2Stack[turns]);
109     console.log(`Current Score: ${player1Score} to ${player2Score}`);
110
111     turns++;
112   }
113   declareWinner();
114 }
115
116 /**
117  * Declares the winner based on player scores.
118  */
119 function declareWinner() {
120   console.log("-----");
121   console.log(`FINAL SCORE: ${player1Score} to ${player2Score}`);
122   if (player1Score > player2Score) {
123     winner = "Player 1";
124   } else if (player1Score < player2Score) {
125     winner = "Player 2";
126   } else {
127     winner = "Tie! Everybody";
128   }
129   console.log(`${winner} is the winner!!!`);
130 }
131
```



PROMINEO TECH

JS index_test.js > describe("MyFunctions") callback

```
1  const expect = chai.expect;
2
3  describe("MyFunctions", function () {
4    const unshuffledDeck = [
5      1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 1
6      11, 12, 13, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,
7    ];
8    describe("#buildDeck", function () {
9      it("should create a deck of cards with 1-13 repeating 4 times", function () {
10        expect(buildDeck()).to.eql(unshuffledDeck);
11      });
12    });
13
14    // Test the shuffle function. Note need a new array to test rather than point
15    // to the original array which mutates with the shuffle.
16    describe("#shuffle", function () {
17      const newDeck = [
18        1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,
19        10, 11, 12, 13, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,
20      ];
21      it("should return an array not equal to unshuffledDeck", function () {
22        expect(shuffle(newDeck)).to.not.eql(unshuffledDeck);
23      });
24    });
25  });
26
```

Screenshots of Running Application:



PROMINEO TECH

```

  ... Round 19! ...
Player 1 card: 7 | Player 2 card: 2
Player 1 wins!
Current Score: 11 to 8

--- Round 20! ---
Player 1 card: 3 | Player 2 card: 6
Player 2 wins!
Current Score: 11 to 9

--- Round 21! ---
Player 1 card: 7 | Player 2 card: 2
Player 1 wins!
Current Score: 12 to 9

--- Round 22! ---
Player 1 card: 6 | Player 2 card: 4
Player 1 wins!
Current Score: 13 to 9

--- Round 23! ---
Player 1 card: 5 | Player 2 card: 4
Player 1 wins!
Current Score: 14 to 9

--- Round 24! ---
Player 1 card: 2 | Player 2 card: 5
Player 2 wins!
Current Score: 14 to 10

--- Round 25! ---
Player 1 card: 8 | Player 2 card: 1
Player 1 wins!
Current Score: 15 to 10

--- Round 26! ---
Player 1 card: 3 | Player 2 card: 9
Player 2 wins!
Current Score: 15 to 11
-----
FINAL SCORE: 15 to 11
Player 1 is the winner!!!
>
```