



Intro to JavaScript Week 6 Coding Assignment

Points possible: 70

Category	Criteria	% of Grade
Functionality	Does the code work?	25
Organization	Is the code clean and organized? Proper use of white space, syntax, and consistency are utilized. Names and comments are concise and clear.	25
Creativity	Student solved the problems presented in the assignment using creativity and out of the box thinking.	25
Completeness	All requirements of the assignment are complete.	25

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*.

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 ran, 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 either Player
- After all cards have been played, display the score.

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



PROMINEO TECH

Screenshots of Code:

```
VS Code - Visual Studio Code
index.js - JS War

File Edit Selection View Go Run Terminal Help

EXPLORER
  JS WAR
    > picomatch
    > randombytes
    > readrip
    > require-directory
    > safe-buffer
    > serialize-javascript
    > string-width
    > strip-ansi
    > strip-json-comments
    > supports-color
    > to-regex-range
    > type-detect
    > which
    > workerpool
    > wrap-ansi
    > wrapappy
    > y1ln
    > yargs
    > yargs-parser
    > yargs-unparser
    > yoscto-queue
  ( package-lock.json
  ( index_test.html
  ( index_test.js
  ( index.html
  ( index.js
  ( JS war.js
  ( package-lock.json
  ( package.json
  ( README.md
  > OUTLINE
  > TIMELINE

index.js
1  setupMatch
2  const suits = ['Spades', 'Hearts', 'Clubs', 'Diamonds']
3  const values = ['Ace', '2', '3', '4', '5', '6', '7', '8', '9', '10', 'Jack', 'Queen', 'King']
4  const valueMap = {
5    '2': 2,
6    '3': 3,
7    '4': 4,
8    '5': 5,
9    '6': 6,
10   '7': 7,
11   '8': 8,
12   '9': 9,
13   '10': 10,
14   'Jack': 11,
15   'Queen': 12,
16   'King': 13,
17   'Ace': 14
18 }
19 // Creates the entire possible combination for the cards but haven't created the cards yet.
20 class Card {
21   constructor(suit, value) {
22     this.suit = suit;
23     this.value = value;
24   }
25   // Here we identify every card has a suit and a value but they aren't assigned to each other yet.
26 }
27
28 class Player {
29   constructor(name) {
30     this.name = name; // tells that there will be a naming factor to this.
31     this.score = 0; // here to indicate that the starting score is 0
32     this.playerDeck = []; // begins to label this saying the player deck will be an array under Player.
33   }
34   addPlayerDeck(deck) {
35     this.playerDeck = deck; // add's the deck that will be cut in half after being shuffled to the Player class.
36   }
37 }
38
```

```
VS Code - Visual Studio Code
index.js - JS War

File Edit Selection View Go Run Terminal Help

EXPLORER
  JS WAR
    > picomatch
    > randombytes
    > readrip
    > require-directory
    > safe-buffer
    > serialize-javascript
    > string-width
    > strip-ansi
    > strip-json-comments
    > supports-color
    > to-regex-range
    > type-detect
    > which
    > workerpool
    > wrap-ansi
    > wrapappy
    > y1ln
    > yargs
    > yargs-parser
    > yargs-unparser
    > yoscto-queue
  ( package-lock.json
  ( index_test.html
  ( index_test.js
  ( index.html
  ( index.js
  ( JS war.js
  ( package-lock.json
  ( package.json
  ( README.md
  > OUTLINE
  > TIMELINE

index.js
39   }
40 }
41 // Gives names to our two players
42 var player1 = new Player('sand');
43 var player2 = new Player('glass');
44
45 class Deck {
46   constructor(cards = new newDeck()) { // stating that cards will be added to what's being called newDeck
47     this.cards = cards;
48   }
49   get numberOfCards() { // get is like a getter and setter from backend, gets the total number of cards they'll be
50     return this.cards.length; // returns the amount as a number due to .length
51   }
52   shuffle() { // time to shuffle
53     this.numberOfCards
54     for (let i = this.numberOfCards - 1; i >= 0; i--) { // tells it to count down from the total number of cards
55       const newCard = Math.floor(Math.random() * (this.numberOfCards)); // Every randomized combination of cards is added as a new card
56       const oldCard = this.cards[newCard]; // states the previous turn's card is overwritten by the new one to play
57       this.cards[newCard] = this.cards[i]; // declares that the new card previously discussed is i number in the cards array.
58       this.cards[i] = oldCard; // after being declared and used that card is changed and labeled as the old card.
59     }
60   }
61 }
62
63 function newDeck() { // returns the deck by creating every combination of the cards by mapping all of the suits and then mapping all the values
64   // onto them.
65   return suits.flatMap(suit => {
66     return values.map(value => {
67       return new Card(suit, value); // and returns each individual card
68     });
69   });
70 }
71
72 // we begin the match
73 function setupMatch(player1, player2) { // begins the match requiring two players
74
```



PROMINEO TECH

```
JS index.js X index.html JS index_test.js index_test.html
JS index.js > setupMatch
74 function setupMatch(player1, player2) { //begins the match requiring two players
75     const deck = new Deck(); //the deck constant for this function will be a new Deck from the class;
76     deck.shuffle(); //we actually do the implemented shuffle
77     console.log(deck.cards.length); // printed the length just to make sure all the cards are properly there.
78     player1.addPlayerDeck(deck.cards.slice(0, 26)); //implements the player one deck, starting with the shuffled deck array object 0-25,
79     //being the ending one actually isn't included in it
80     player2.addPlayerDeck(deck.cards.slice(26)); // make sure you start where the previous one ended with the same number so it's 26-51 objects
81     console.log(player1.playerDeck.length); // a test to make sure the player decks were the same size.
82     console.log(player2.playerDeck.length);
83 }
84
85
86 // implement the rounds of the game and to repeat until all cards from the player decks are gone.
87 function roundEnd(player1, player2) {
88     for (let i = 0; i < 26; i++) {
89         if (ValueMap[player1.playerDeck[i].value] > ValueMap[player2.playerDeck[i].value]) { //if the value of player 1's card from the player
90             //deck is higher than player 2's, player 1 wins.
91             console.log(`${player1.name} throws down the ${player1.playerDeck[i].value} of ${player1.playerDeck[i].suit}`);
92             console.log(`${player2.name} throws down the ${player2.playerDeck[i].value} of ${player2.playerDeck[i].suit}`);
93             console.log(`----- ${player1.name} wins the round -----`);
94             player1.score += 1;
95             console.log(`The current score is ${player1.score} to ${player2.score} \n`);
96         } else if (ValueMap[player1.playerDeck[i].value] < ValueMap[player2.playerDeck[i].value]) { //if the value of player 1's card from the
97             //player deck is lower than player 2's, player 2 wins.
98             console.log(`${player1.name} throws down the ${player1.playerDeck[i].value} of ${player1.playerDeck[i].suit}`);
99             console.log(`${player2.name} throws down the ${player2.playerDeck[i].value} of ${player2.playerDeck[i].suit}`);
100             console.log(`----- ${player2.name} wins the round -----`);
101             player2.score += 1;
102             console.log(`The current score is ${player1.score} to ${player2.score} \n`);
103         } else {
104             console.log(`----- Yarr, it be a tie -----`);
105             console.log(`The current score is ${player1.score} to ${player2.score} \n`);
106         }
107     }
108 }
109
```

```
JS index.js X index.html JS index_test.js index_test.html
JS index.js > setupMatch
100 console.log(`----- ${player2.name} wins the round -----`);
101 player2.score += 1;
102 console.log(`The current score is ${player1.score} to ${player2.score} \n`);
103 } else {
104     console.log(`----- Yarr, it be a tie -----`);
105     console.log(`The current score is ${player1.score} to ${player2.score} \n`);
106 }
107 }
108 }
109
110 //end of game results
111 function endResults(player1, player2) {
112     if (player1.score > player2.score) {
113         console.log(`${player1.name} wins! Good job ye land lubber`);
114     } else if (player1.score < player2.score) {
115         console.log(`${player2.name} wins! Now get back to work before the Capt'n see ye`);
116     } else {
117         console.log(`Tie Game! Well fancy ye that`);
118     }
119 }
120
121 //The calling of the three functions to actually run the game at the very end.
122 setupMatch(player1, player2);
123 roundEnd(player1, player2);
124 endResults(player1, player2);
```



PROMINEO TECH

```
JS index.js  <> index.html X  JS index_test.js  <>

<> index.html > html
1  <!DOCTYPE html>
2  <html lang="en">
3    <body>
4      <script src=index.js></script>
5    </body>
6  </html>
```

```
JS index.js  <> index.html  JS index_test.js X  <> index_test.html

JS index_test.js > describe('MyFunctions') callback > describe('#Name check') callback > it('Is looking for the name of the first player and tests them to match with what's
1  var expect = chai.expect;
2
3
4  describe('MyFunctions', function() {
5
6    describe('#Name check', function() {
7      it('Is looking for the name of the first player and tests them to match with what's in the code', function() {
8        let player1 = new Player('Sand'); //expect Sand
9        expect(player1.name).to.be.a('string');
10      });
11    });
12  });
```



PROMINEO TECH

```
JS index.js  index.html  JS index_test.js  index_test.html X
index_test.html > html > body > div#mocha
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8">
5    <meta http-equiv="X-UA-Compatible" content="IE=edge">
6    <meta name="viewport" content="width=device-width, initial-scale=1.0">
7    <link rel="stylesheet" href="node_modules/mocha/mocha.css">
8    <title>Week 6 Tests</title>
9  </head>
10 <body>
11   <div id="mocha">
12     <p><a href=".">Index</a></p>
13   </div>
14
15   <div id="messages"></div>
16   <div id="fixtures"></div>
17   <script src="node_modules/mocha/mocha.js"></script>
18   <script src="node_modules/chai/chai.js"></script>
19   <!-- The line below will reference your JS code you are looking to test against -->
20   <script src="index.js"></script>
21
22   <script>mocha.setup('bdd')</script>
23
24   <!-- The line below will reference your code where your test(s) are set up -->
25   <script src="index_test.js"></script>
26
27   <script>mocha.run()</script>
28
29 </body>
30 </html>
```

Screenshots of Running Application:

```
index.html
C:\Users\traps\OneDrive\Documents\JS\2020\Week 6\index.html
Elements Console Sources Network Performance Memory Application Security Lighthouse
top Filter Default levels No issues
52 index.js:52
25 index.js:25
26 index.js:26
Sand throws down the 6 of Spades index.js:108
Glass throws down the Jack of Clubs index.js:109
----- Glass wins the round ----- index.js:110
The current score is 0 to 1 index.js:111
----- Yarr, it be a tie ----- index.js:112
The current score is 0 to 1 index.js:113
Sand throws down the 10 of Spades index.js:114
Glass throws down the 7 of Hearts index.js:115
----- Sand wins the round ----- index.js:116
The current score is 1 to 1 index.js:117
----- Yarr, it be a tie ----- index.js:118
The current score is 1 to 1 index.js:119
Sand throws down the Jack of Diamonds index.js:120
Glass throws down the 3 of Clubs index.js:121
----- Sand wins the round ----- index.js:122
The current score is 2 to 1 index.js:123
Sand throws down the 4 of Diamonds index.js:124
Glass throws down the 8 of Diamonds index.js:125
----- Glass wins the round ----- index.js:126
The current score is 2 to 2 index.js:127
Sand throws down the 4 of Hearts index.js:128
Glass throws down the 2 of Hearts index.js:129
----- Sand wins the round ----- index.js:130
The current score is 3 to 2 index.js:131
Sand throws down the 9 of Clubs index.js:132
Glass throws down the King of Clubs index.js:133
----- Glass wins the round ----- index.js:134
The current score is 3 to 3 index.js:135
Sand throws down the 10 of Diamonds index.js:136
Glass throws down the 5 of Spades index.js:137
----- Sand wins the round ----- index.js:138
```

[illegible]

The screenshot shows a web browser window with the address bar displaying 'C:\Users\traps\OneDrive\Documents\55620War\index.html'. The console shows the following log:

```

The current score is 10 to 5
Sand throws down the 7 of Diamonds
Glass throws down the 6 of Clubs
-----
Sand wins the round -----
The current score is 11 to 5
Sand throws down the 6 of Diamonds
Glass throws down the 4 of Clubs
-----
Sand wins the round -----
The current score is 12 to 5
Sand throws down the 4 of Spades
Glass throws down the 9 of Diamonds
-----
Glass wins the round -----
The current score is 12 to 6
Sand throws down the 9 of Hearts
Glass throws down the 2 of Clubs
-----
Sand wins the round -----
The current score is 13 to 6
-----
Yar, it be a tie -----
The current score is 13 to 6
Sand throws down the 7 of Clubs
Glass throws down the Ace of Hearts
-----
Glass wins the round -----
The current score is 13 to 7
Sand throws down the 10 of Hearts
Glass throws down the 8 of Spades
-----
Sand wins the round -----
The current score is 14 to 7
Sand throws down the Queen of Diamonds
Glass throws down the 3 of Hearts
-----
Sand wins the round -----
The current score is 15 to 7
Sand wins! Good job ye land lubber

```

The right side of the console shows the source file 'index.html' and the line numbers corresponding to each log entry.

[Index](#)

MyFunctions

#Name check

- ✓ Is looking for the name of the first player and tests them to match with what's in the code

URL to GitHub Repository:



PROMINEO TECH

<https://github.com/trepvox/War-Card-Game-in-JavaScript>