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:

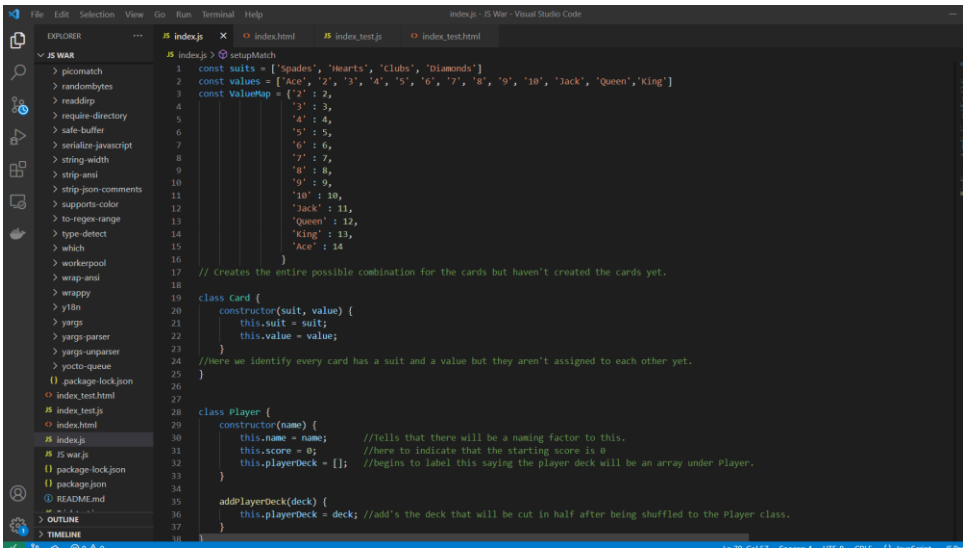
For the final project you will be creating an automated version of the classic card game *WAR*.

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:

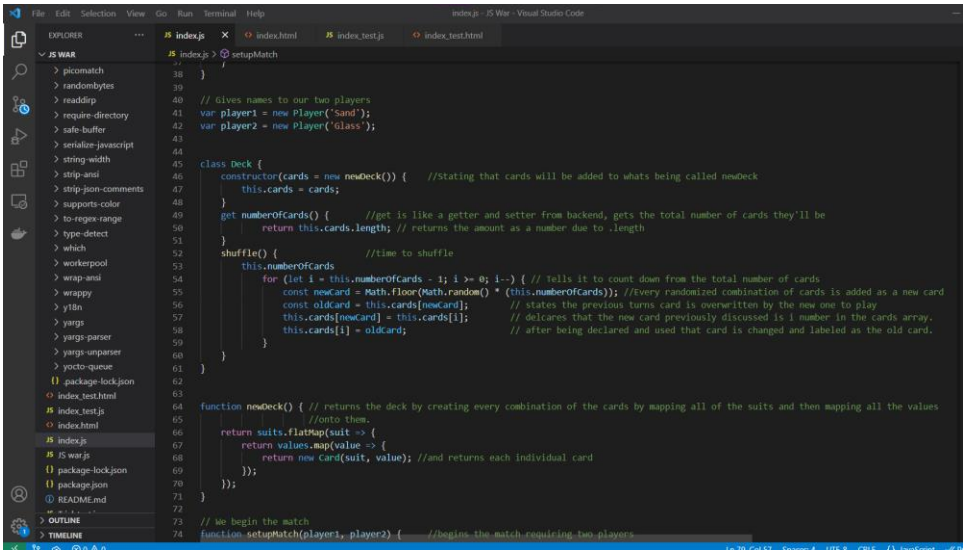
- Deal 26 Cards to two Players from a Deck.
- Iterate through the turns where each Player plays a Card
- The Player who played the higher card is awarded a point
    - Ties result in zero points for either Player
- After all cards have been played, display the score.

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

Screenshots of Code:



```
const suits = ['Spades', 'Hearts', 'Clubs', 'Diamonds']
const values = ['Ace', '2', '3', '4', '5', '6', '7', '8', '9', '10', 'Jack', 'Queen','King']
const ValueMap = {'2' : 2,
                  '3' : 3,
                  '4' : 4,
                  '5' : 5,
                  '6' : 6,
                  '7' : 7,
                  '8' : 8,
                  '9' : 9,
                  '10' : 10,
                  'Jack' : 11,
                  'Queen' : 12,
                  'King' : 13,
                  'Ace' : 14
                 }
// Creates the entire possible combination for the cards but haven't created the cards yet.

class Card {
    constructor(suit, value) {
        this.suit = suit;
        this.value = value;
    }
//Here we identify every card has a suit and a value but they aren't assigned to each other yet.
}


class Player {
    constructor(name) {
        this.name = name;       //Tells that there will be a naming factor to this.
        this.score = 0;         //here to indicate that the starting score is 0
        this.playerDeck = [];   //begins to label this saying the player deck will be an array under Player.
    }

    addPlayerDeck(deck) {
        this.playerDeck = deck; //add's the deck that will be cut in half after being shuffled to the Player class.
    }
}
```



```
}

// Gives names to our two players
var player1 = new Player('Sand');
var player2 = new Player('Glass');


class Deck {
    constructor(cards = new newDeck()) {    //Stating that cards will be added to whats being called newDeck
        this.cards = cards;
    }
    get numberOfCards() {       //get is like a getter and setter from backend, gets the total number of cards they'll be
        return this.cards.length; // returns the amount as a number due to .length
    }
    shuffle() {                 //time to shuffle
        this.numberOfCards
            for (let i = this.numberOfCards - 1; i >= 0; i--) { // Tells it to count down from the total number of cards
                const newCard = Math.floor(Math.random() * (this.numberOfCards)); //Every randomized combination of cards is added as a new card
                const oldCard = this.cards[newCard];        // states the previous turns card is overwritten by the new one to play
                this.cards[newCard] = this.cards[i];        // delcares that the new card previously discussed is 1 number in the cards array.
                this.cards[i] = oldCard;                    // after being declared and used that card is changed and labeled as the old card.
            }
    }
}


function newDeck() { // returns the deck by creating every combination of the cards by mapping all of the suits and then mapping all the values
                     //onto them.
    return suits.flatMap(suit => {
        return values.map(value => {
            return new Card(suit, value); //and returns each individual card
        });
    });
}

// We begin the match
function setupMatch(player1, player2) {     //begins the match requiring two players
```

index.js | index.html | index_test.js | index_test.html

index.js > setupMatch

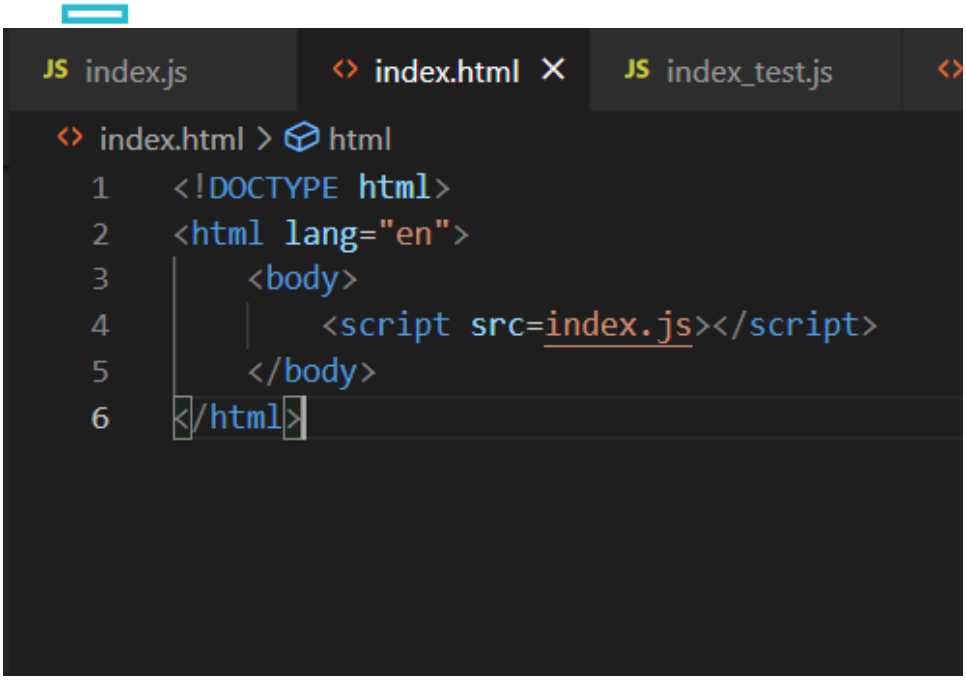
```
74  function setupMatch(player1, player2) {     //begins the match requiring two players
75      const deck = new Deck();                //the deck constant for this function will be a new Deck from the class;
76      deck.shuffle();                         //we actually do the implemented shuffle
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('---------- Yar, it be a tie ----------');
105             console.log(`The current score is ${player1.score} to ${player2.score} \n`) ;
106         }
107     }
108 }
109
```

index.js | index.html | index_test.js | index_test.html

index.js > setupMatch

```
101             player2.score += 1;
102             console.log(`The current score is ${player1.score} to ${player2.score} \n`);
103         } else {
104             console.log('---------- Yar, 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

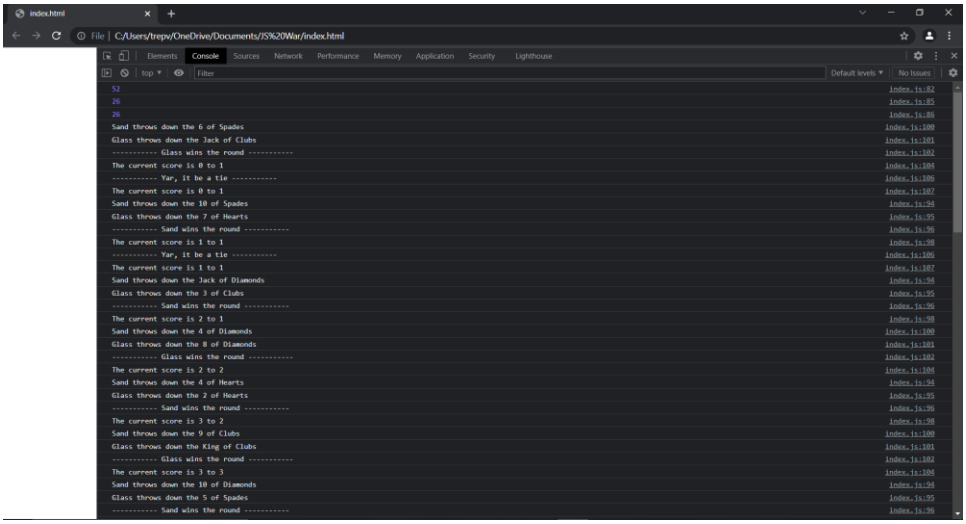


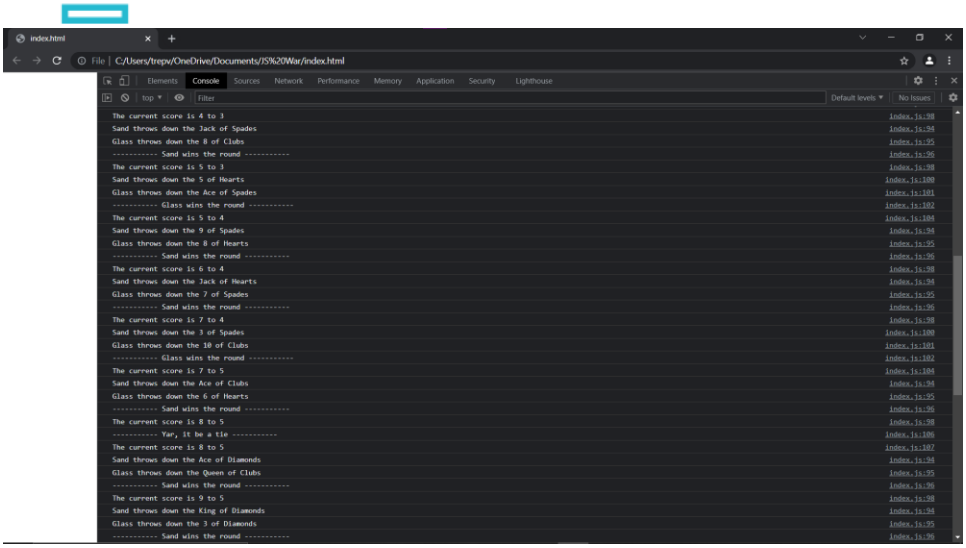
index.js | index.html | index_test.js | index_test.html

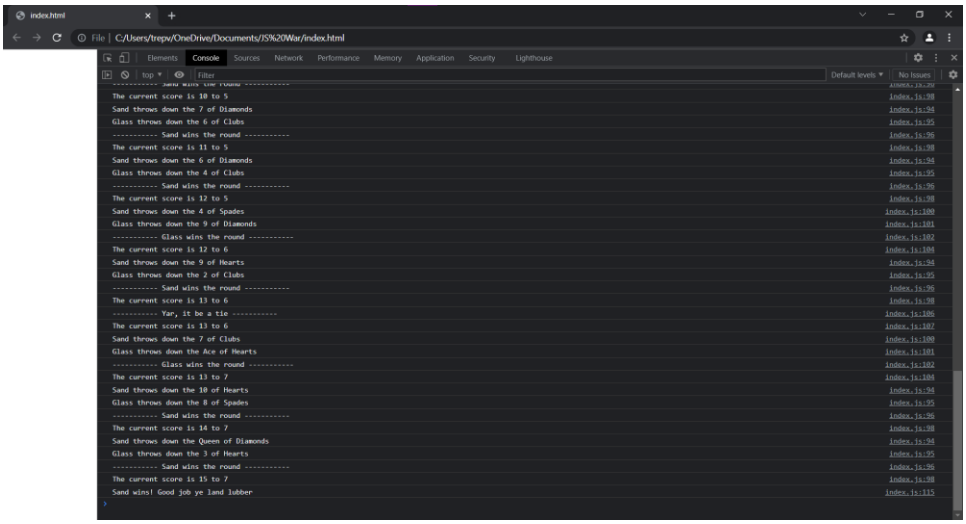
index_test.html > html > body > div#mocha

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="node_modules/mocha/mocha.css">
    <title>Week 6 Tests</title>
</head>
<body>
    <div id="mocha">
        <p><a href=".">Index</a></p>
    </div>

    <div id="messages"></div>
    <div id="fixtures"></div>
    <script src="node_modules/mocha/mocha.js"></script>
    <script src="node_modules/chai/chai.js"></script>
    <!-- The line below will reference your JS code you are looking to test against -->
    <script src="index.js"></script>

    <script>mocha.setup('bdd')</script>

    <!-- The line below will reference your code where your test(s) are set up -->
    <script src="index_test.js"></script>

    <script>mocha.run()</script>

</body>
</html>
```

Screenshots of Running Application:







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