PROMINEO TECH

Intro to JavaScript Week 6 Coding Assignment

Points possible: 100

URL to GitHub Repository:

https://github.com/harberts01/Week_6.git

URL to Your Coding Assignment Video:

https://youtu.be/dX7vVEGxkNk

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.

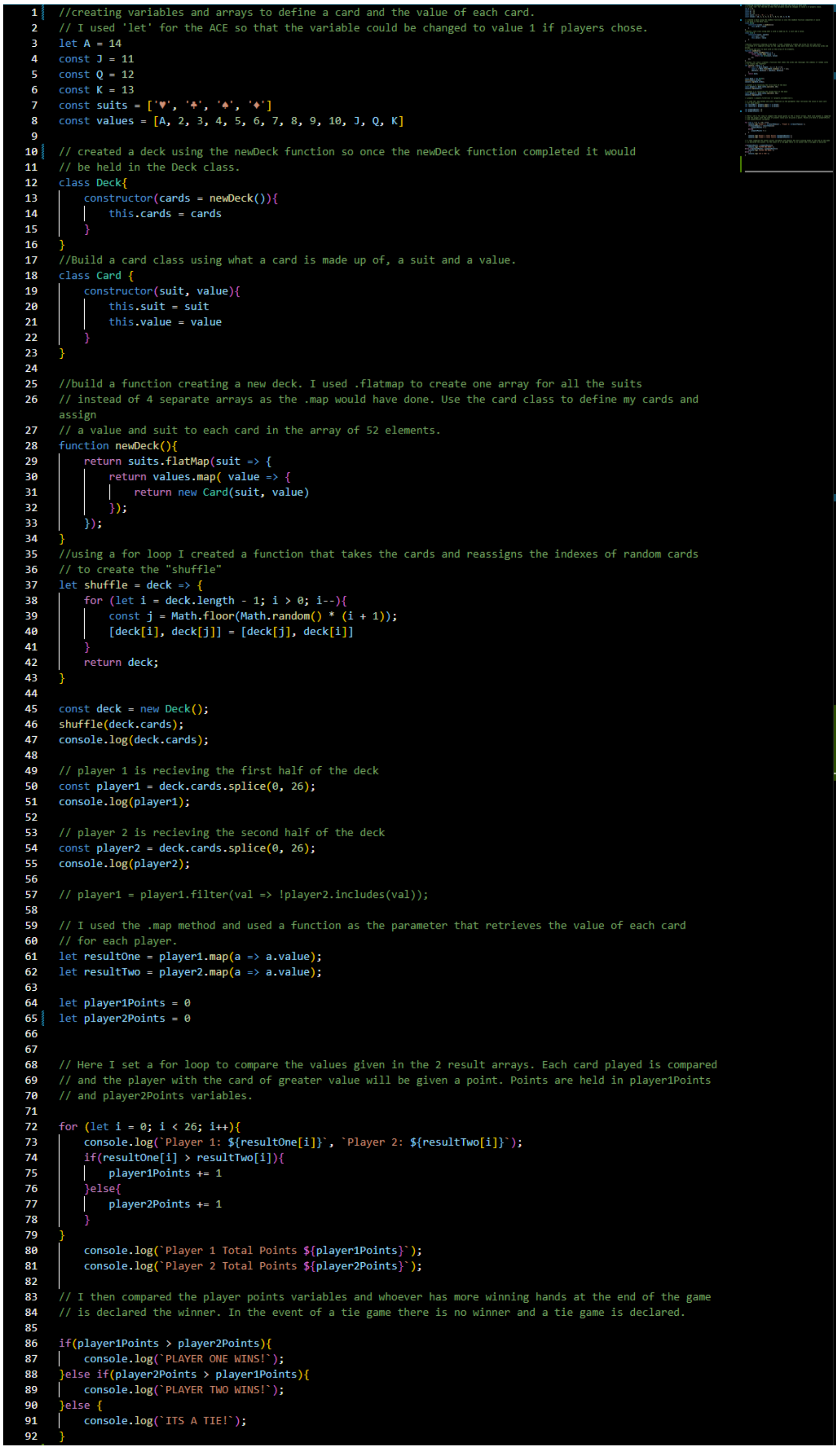
There are many versions of the game *WAR*, but in this version there are only 2 players and you don't need to do anything special when there is a tie on a round.

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
  - 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.

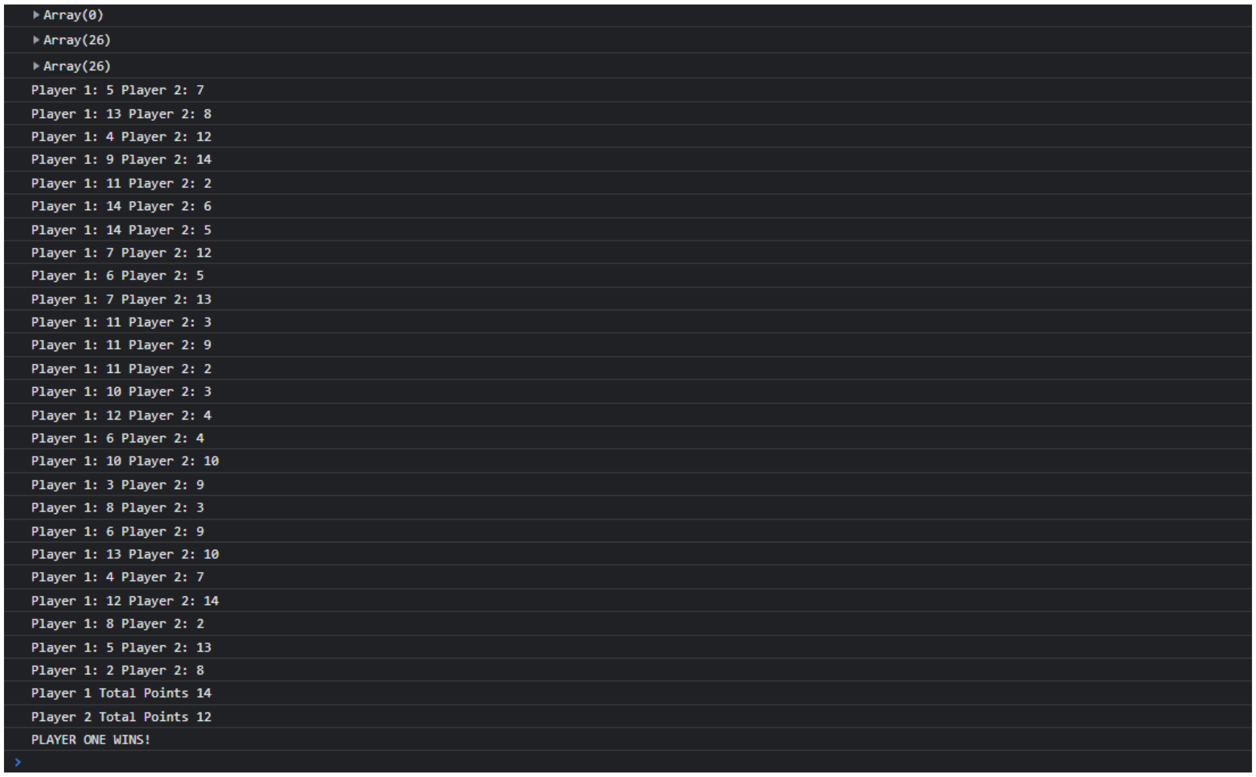
Screenshots of Code:



```javascript
//creating variables and arrays to define a card and the value of each card.
// I used 'let' for the ACE so that the variable could be changed to value 1 if players chose.
let A = 14
const J = 11
const Q = 12
const K = 13
const suits = ['♥', '♣', '♠', '♦']
const values = [A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K]

// created a deck using the newDeck function so once the newDeck function completed it would
// be held in the Deck class.
class Deck{
    constructor(cards = newDeck()){
        this.cards = cards
    }
}
//Build a card class using what a card is made up of, a suit and a value.
class Card {
    constructor(suit, value){
        this.suit = suit
        this.value = value
    }
}

//build a function creating a new deck. I used .flatmap to create one array for all the suits
// instead of 4 separate arrays as the .map would have done. Use the card class to define my cards and
assign
// a value and suit to each card in the array of 52 elements.
function newDeck(){
    return suits.flatMap(suit => {
        return values.map( value => {
            return new Card(suit, value)
        });
    });
}
//using a for loop I created a function that takes the cards and reassigns the indexes of random cards
// to create the "shuffle"
let shuffle = deck => {
    for (let i = deck.length - 1; i > 0; i--){
        const j = Math.floor(Math.random() * (i + 1));
        [deck[i], deck[j]] = [deck[j], deck[i]]
    }
    return deck;
}

const deck = new Deck();
shuffle(deck.cards);
console.log(deck.cards);

// player 1 is recieving the first half of the deck
const player1 = deck.cards.splice(0, 26);
console.log(player1);

// player 2 is recieving the second half of the deck
const player2 = deck.cards.splice(0, 26);
console.log(player2);

// player1 = player1.filter(val => !player2.includes(val));

// I used the .map method and used a function as the parameter that retrieves the value of each card
// for each player.
let resultOne = player1.map(a => a.value);
let resultTwo = player2.map(a => a.value);

let player1Points = 0
let player2Points = 0


// Here I set a for loop to compare the values given in the 2 result arrays. Each card played is compared
// and the player with the card of greater value will be given a point. Points are held in player1Points
// and player2Points variables.

for (let i = 0; i < 26; i++){
    console.log(`Player 1: ${resultOne[i]}`, `Player 2: ${resultTwo[i]}`);
    if(resultOne[i] > resultTwo[i]){
        player1Points += 1
    }else{
        player2Points += 1
    }
}
    console.log(`Player 1 Total Points ${player1Points}`);
    console.log(`Player 2 Total Points ${player2Points}`);

// I then compared the player points variables and whoever has more winning hands at the end of the game
// is declared the winner. In the event of a tie game there is no winner and a tie game is declared.

if(player1Points > player2Points){
    console.log(`PLAYER ONE WINS!`);
}else if(player2Points > player1Points){
    console.log(`PLAYER TWO WINS!`);
}else {
    console.log(`ITS A TIE!`);
}
```

PROMINEO TECH

Screenshots of Running Application:



```
▸ Array(0)
▸ Array(26)
▸ Array(26)
Player 1: 5 Player 2: 7
Player 1: 13 Player 2: 8
Player 1: 4 Player 2: 12
Player 1: 9 Player 2: 14
Player 1: 11 Player 2: 2
Player 1: 14 Player 2: 6
Player 1: 14 Player 2: 5
Player 1: 7 Player 2: 12
Player 1: 6 Player 2: 5
Player 1: 7 Player 2: 13
Player 1: 11 Player 2: 3
Player 1: 11 Player 2: 9
Player 1: 11 Player 2: 2
Player 1: 10 Player 2: 3
Player 1: 12 Player 2: 4
Player 1: 6 Player 2: 4
Player 1: 10 Player 2: 10
Player 1: 3 Player 2: 9
Player 1: 8 Player 2: 3
Player 1: 6 Player 2: 9
Player 1: 13 Player 2: 10
Player 1: 4 Player 2: 7
Player 1: 12 Player 2: 14
Player 1: 8 Player 2: 2
Player 1: 5 Player 2: 13
Player 1: 2 Player 2: 8
Player 1 Total Points 14
Player 2 Total Points 12
PLAYER ONE WINS!
›
```

PROMINEO TECH

Index

MyFunctions

New hand

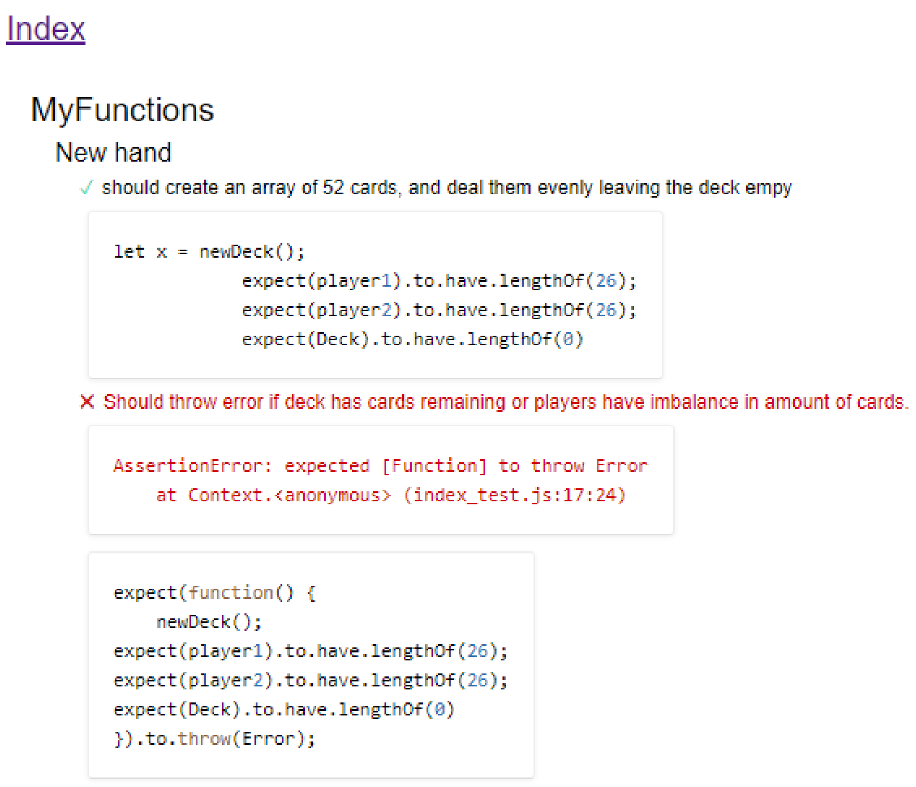
✓ should create an array of 52 cards, and deal them evenly leaving the deck empy

```
let x = newDeck();
            expect(player1).to.have.lengthOf(26);
            expect(player2).to.have.lengthOf(26);
            expect(Deck).to.have.lengthOf(0)
```

✓ Should throw error if deck has cards remaining or players have imbalance in amount of cards.

```
expect(function() {
    newDeck();
expect(player1).to.have.lengthOf(26);
expect(player2).to.have.lengthOf(25);
expect(Deck).to.have.lengthOf(1)
}).to.throw(Error);
```

PROMINEO TECH



Index

MyFunctions

New hand

✓ should create an array of 52 cards, and deal them evenly leaving the deck empy

```
let x = newDeck();
            expect(player1).to.have.lengthOf(26);
            expect(player2).to.have.lengthOf(26);
            expect(Deck).to.have.lengthOf(0)
```

✕ Should throw error if deck has cards remaining or players have imbalance in amount of cards.

```
AssertionError: expected [Function] to throw Error
    at Context.<anonymous> (index_test.js:17:24)
```

```
expect(function() {
    newDeck();
expect(player1).to.have.lengthOf(26);
expect(player2).to.have.lengthOf(26);
expect(Deck).to.have.lengthOf(0)
}).to.throw(Error);
```