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*.

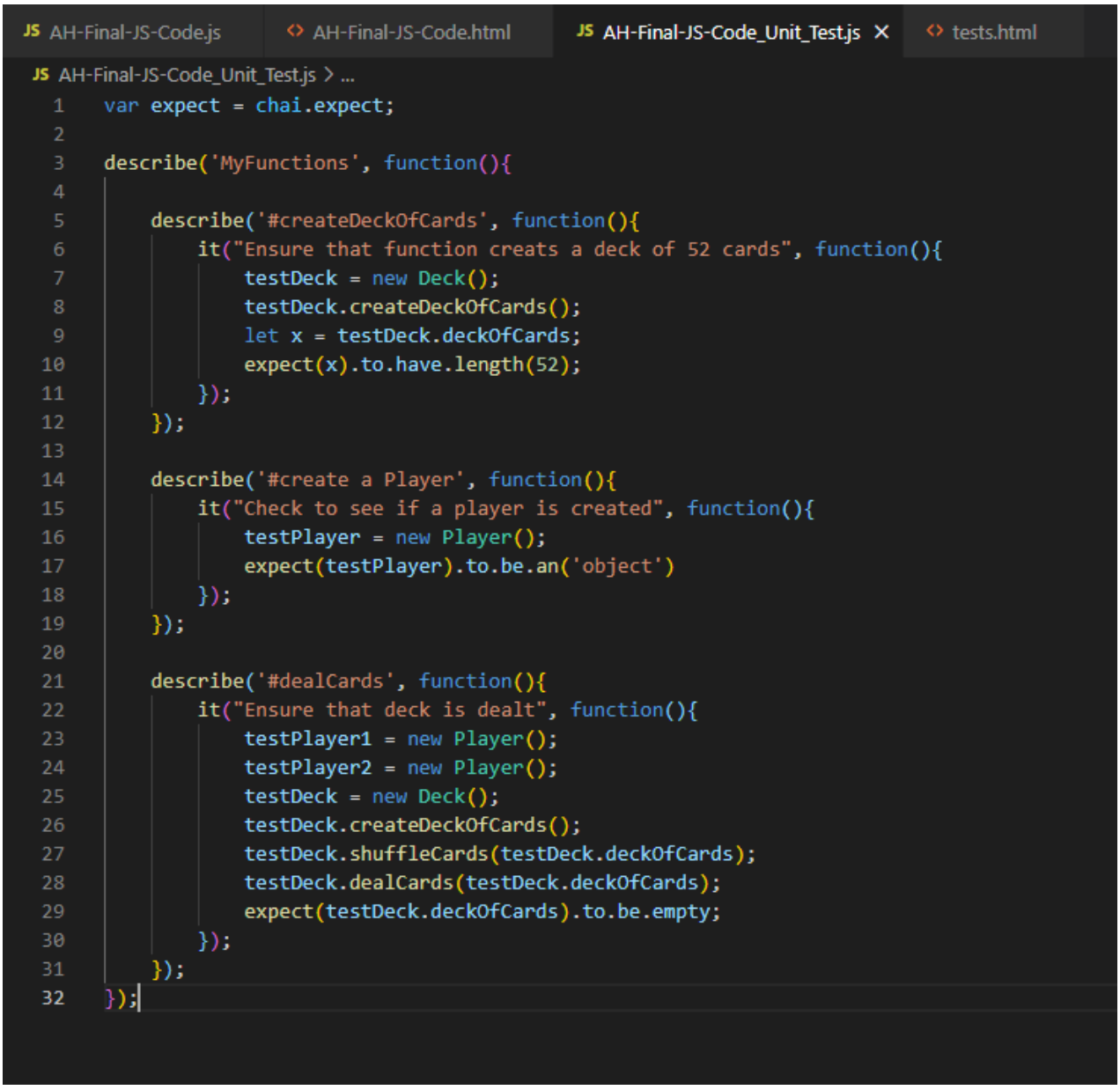
Think about how you would build this project and write your plan down. Consider classes such as Card, Deck, and Player and what fields and methods they might each have. You can implement the game however you'd like (i.e. printing to the console, using alert, or some other way). The completed project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

Unit Test:



War Game Code:

PROMINEO TECH

JS AH-Final-JS-Code.js | AH-Final-JS-Code.html | JS AH-Final-JS-Code_Unit_Test.js | tests.html

JS AH-Final-JS-Code.js > ...

```
class Card{                             //an object called "card" with a value, face, and suit
    constructor(value, suit, face){
        this.suit = suit;
        this.value = value;
        this.face = face;
    };


    describe(){                         //describes the card's suit and face
        return `${this.face} of ${this.suit}`;
    };
};


class Deck{                             //an object called "deck" that can create a deck of cards,
    constructor(){                      //shuffle them, and then deal them
        this.deckOfCards = [];
    };

    createDeckOfCards(){                //creates the cards for the deck
        for (let i = 2; i <= 14; i++){
            this.deckOfCards.push(new Card(i, "Hearts", i));

            this.deckOfCards.push(new Card(i, "Spades", i));

            this.deckOfCards.push(new Card(i, "Clubs", i));

            this.deckOfCards.push(new Card(i, "Diamonds", i));

        }

        for(let card of this.deckOfCards){      //fixes the faces of the cards
            if(card.value === 14){
                card.face = "Ace";
            }else if(card.value === 11){
                card.face = "Jack";
            }else if(card.value === 12){
                card.face = "Queen";
            }else if(card.value === 13){
                card.face = "King";
            }
        };
    };


    shuffleCards(){                         //shuffles the deckOfCards array
        this.deckOfCards.sort(function(){
            return Math.random() - 0.5;
        });
```

```
            return Math.random() - 0.5;
        });
    };


    dealCards(){                            //assigns the cards to the player's hands
        for (let i = 51; i >= 0; i--){
            if(i % 2 == 0){
                player1.hand.push(this.deckOfCards[i]);
            }else{
                player2.hand.push(this.deckOfCards[i]);
            }
            this.deckOfCards.pop();
        };
    };
};


class Player{                //an object called "player" that has a hand to hold cards and a score
    constructor(){
        this.hand = [];
        this.score = 0;
    };
};

function playCards(firstPlayer, secondPlayer){      //plays a game by comparing the cards at the different indexes
    let scoreString = [""];                         //in both players' hand arrays
    for (let i = 0; i<= 25; i++){
        if (firstPlayer.hand[i].value > secondPlayer.hand[i].value){
            firstPlayer.score++;
            let y = `Round ${i+1})  Player 1 wins round:  ${firstPlayer.hand[i].describe()}  beats  ${secondPlayer.hand[i].describe()} `;

            scoreString.push(y);
        }else if(secondPlayer.hand[i].value > firstPlayer.hand[i].value){
            secondPlayer.score++;
            let x = `Round ${i+1})  Player 2 wins round:  ${secondPlayer.hand[i].describe()}  beats  ${firstPlayer.hand[i].describe()} `;

            scoreString.push(x);
        }else{
            let z = `Round ${i+1})  Tie:  ${firstPlayer.hand[i].describe()}  is the same as  ${secondPlayer.hand[i].describe()} `;

            scoreString.push(z);
        };
    };

alert(`
```

```
92
93    alert(`
94    Game Over!
95    Final Score:
96    Player 1:  ${firstPlayer.score}
97    Player 2:  ${secondPlayer.score}
98
99    Round Recap:
100   ${scoreString.join("\n")}
101   `);
102   };
103
104
105   let player1 = new Player();
106   let player2 = new Player();
107   let myDeck = new Deck();
108
109
110   myDeck.createDeckOfCards();
111   myDeck.shuffleCards(myDeck.deckOfCards);
112   console.log(myDeck.deckOfCards);
113   myDeck.dealCards(myDeck.deckOfCards);
114   console.log(player1.hand);
115   console.log(player2.hand);
116   playCards(player1, player2);
117
118
```

Screenshots of Running Application:

PROMINEO TECH

file://

Game Over!
Final Score:
Player 1: 9
Player 2: 12

Round Recap:

Round 1) Tie: Jack of Clubs is the same as Jack of Diamonds
Round 2) Player 1 wins round: King of Clubs beats Jack of Spades
Round 3) Player 2 wins round: King of Hearts beats Jack of Hearts
Round 4) Player 2 wins round: King of Spades beats Queen of Clubs
Round 5) Player 1 wins round: Queen of Diamonds beats 10 of Hearts
Round 6) Player 1 wins round: Ace of Clubs beats 10 of Spades
Round 7) Player 1 wins round: Queen of Hearts beats 10 of Clubs
Round 8) Player 1 wins round: Ace of Diamonds beats 10 of Diamonds
Round 9) Player 2 wins round: Ace of Hearts beats King of Diamonds
Round 10) Player 2 wins round: Ace of Spades beats Queen of Spades
Round 11) Tie: 9 of Spades is the same as 9 of Diamonds
Round 12) Player 2 wins round: 9 of Clubs beats 8 of Diamonds
Round 13) Tie: 8 of Spades is the same as 8 of Hearts
Round 14) Player 1 wins round: 9 of Hearts beats 3 of Diamonds
Round 15) Player 2 wins round: 8 of Clubs beats 3 of Spades
Round 16) Tie: 3 of Clubs is the same as 3 of Hearts
Round 17) Player 2 wins round: 6 of Diamonds beats 2 of Diamonds
Round 18) Player 2 wins round: 5 of Spades beats 2 of Clubs
Round 19) Tie: 5 of Hearts is the same as 5 of Clubs
Round 20) Player 2 wins round: 7 of Clubs beats 5 of Diamonds
Round 21) Player 2 wins round: 6 of Clubs beats 4 of Diamonds
Round 22) Player 1 wins round: 4 of Clubs beats 2 of Spades
Round 23) Player 2 wins round: 4 of Hearts beats 2 of Hearts
Round 24) Player 1 wins round: 7 of Spades beats 6 of Hearts
Round 25) Player 1 wins round: 7 of Hearts beats 4 of Spades
Round 26) Player 2 wins round: 7 of Diamonds beats 6 of Spades

Don't allow this site to prompt you again

OK

PROMINEO TECH

Index

MyFunctions

#createDeckOfCards

- ✓ Ensure that function creats a deck of 52 cards

#create a Player

- ✓ Check to see if a player is created

#dealCards

- ✓ Ensure that deck is dealt

URL to GitHub Repository:

https://github.com/aheiser2/War-Game-Week-6.git