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.

URL to GitHub Repository: https://github.com/sw-dev-lisa-s-nh/JavaScript-Week6

Screenshots of Code:

PROMINEO TECH

JS week6_hw.js > ...

```
// **********************************
// JavaScript Week6 -- Game of War
// **********************************

class Deck {
    constructor() {
        this.cards =[];
    }

    populateCards() {
        var suits = ["Hearts", "Spades", "Diamonds", "Clubs"];
        var values = [2,3,4,5,6,7,8,9,10,11,12,13,14];
        var descriptions = ["Two", "Three", "Four", "Five", "Six", "Seven", "Eight", "Nine", "Ten", "Jack", "Queen", "King", "Ace"];
        for (let suit of suits) {
            var representation = '';
            const blackHeart = '\u2665';
            const blackspade = '\u2660';
            const blackclub = '\u2663';
            const blackdiamond = '\u2666';
            if (suit == "Hearts") {
                representation = blackHeart;
            } else if  (suit == "Spades") {
                representation = blackspade;
            } else if  (suit == "Diamonds") {
                representation = blackdiamond;
            } else if  (suit == "Clubs") {
                representation = blackclub;
            }
            for (let index = 0; index < values.length; index++) {
                let card = new Card(suit, values[index], `${descriptions[index]} of ${suit}`, representation);
                this.cards.push(card);
            }
        }
    }

    describe() {
        var counter = 1;
        for (let card of this.cards) {
            console.log(`${counter}) ${card.describe()}`);
            counter++;
            // debug code:
            //      card.describe();
        }
    }


}
```

PROMINEO TECH

```
49
50  class Card {
51      constructor(suit, value, description, representation) {
52          this.suit = suit;
53          this.value = value;
54          this.description = description;
55          this.representation = representation;
56      }
57
58      describe() {
59          // debug code:
60          //      console.log(`The ${this.description} has a value of ${this.value}`);
61          return(`the ${this.description} has a value of ${this.value} ${this.representation}`);
62      }
63
64
65  }
66
67  class Player {
68      constructor(name) {
69          this.name = name;
70          this.score = 0;
71          this.hand = [];
72      }
73
74      incrementScore() {
75          this.score++;
76      }
77
78      describe() {
79          //  do I want to return this, or console.log this??
80          // debug code:
81          //      console.log(`Player ${this.name} has a score of ${this.score}`);
82          return `\t${this.name} has a score of ${this.score}`;
83      }
84
85      flip() {
86          return this.hand.shift()
87      }
88
89      draw(deck) {
90          this.hand.push(deck.cards.shift());
91      }
92
93      setsScore(score) {
94          this.score = score;
95      }
96
97      getScore() {
98          return this.score;
99      }
00  }
01
```

PROMINEO TECH

```
// START OF GAME
// Populate a Deck with 52 cards and then sort them.
let deck = new Deck();
deck.populateCards();
console.log("\n-----------------------------------\n");
console.log("     SORTED DECK ");
console.log("-----------------------------------\n");
deck.cards.sort((a,b) => 0.5 - Math.random());
// debug code:
//      deck.describe();
console.log(`Deck has ${deck.cards.length} cards!`);


//  Prompt for userNames, and set to defaults if not provided!
let player1Name = prompt("The name for Player 1 is: ", "Player 1");
if (player1Name == null) {
    player1Name = "Player 1";
}

let player2Name = prompt("The name for Player 2 is: ", "Player 2");
if (player2Name == null) {
    player2Name = "Player 2";
}

let player1 = new Player(player1Name);
let player2 = new Player(player2Name);
console.log(`The game is starting with ${player1Name} and ${player2Name}`);

// deal the deck to both players
for (let index = 0; index < 26; index++) {
    player1.draw(deck);
    player2.draw(deck);
}

alert(`
-------------------------------------------
  Starting the GAME OF WAR: with 2 Players:
-------------------------------------------
${player1.describe()}
\t\tAND
${player2.describe()}
-------------------------------------------`);

// console.log(`Player 1: ${player1.describe()}`);
// for (let card of player1.hand) {
//     console.log(`${card.describe()}`);
// }

// console.log(`Player 2: ${player2.describe()}`);
// for (let card of player2.hand) {
//     console.log(`${card.describe()}`);
// }
```

PROMINEO TECH

```

// Each player flips a card
for (let index = 0; index < 26; index++) {
    let player1Card = player1.flip();
    let player2Card = player2.flip();

    // debug code:
    //      console.log(player1Card);
    //      console.log(player2Card);
    var win;
    // Compare the value
    // increment the score of the card with the highest value
    if (player1Card.value > player2Card.value) {
        player1.incrementScore();
        win = `Point goes to ${player1.name}!`;
    } else if (player2Card.value > player1Card.value) {
        player2.incrementScore();
        win = `Point goes to ${player2.name}!`;
    } else {
        win = `No score -- cards are equal!`;
        // debug code:
        //      console.log('No score -- cards are equal!');
    }
    alert(`
    ${player1.name} has ${player1Card.describe()}
                VS.
        ${player2.name} has ${player2Card.describe()}
    ------------------------------------------------------
            ${win}
    `)
}

if (player1.score > player2.score) {
    var winner = `The WINNER is: ${player1.name}!`
} else if (player2.score > player1.score) {
    var winner = `The WINNER is: ${player2.name}!`
} else {
    var winner = `This GAME OF WAR is a draw!`;
}

console.log(`-----------------------------------------
The game has ended.
\t${player1.describe()}
\t${player2.describe()}
The game has ended.  ${winner}
-----------------------------------------`);
```

```
// END OF GAME

alert(`
-------------------------------------------
        The GAME OF WAR is over:
-------------------------------------------
\t${winner}
-------------------------------------------
${player1.describe()}
\t\t AND
${player2.describe()}
-------------------------------------------`);

```

Test HTML (week6_tests.html):

week6_tests.html > html

```
<!DOCTYPE html>
<html>
<head>
    <link rel="stylesheet" href="node+modules/mocha/mocha.css">
</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>
    <script src="week6_hw.js"></script>
    <script>mocha.setup('bdd')</script>
    <script src="week6_test.js"></script>
    <script>mocha.run();</script>
</body>
</html>
```

Test File (week6_tests.js):

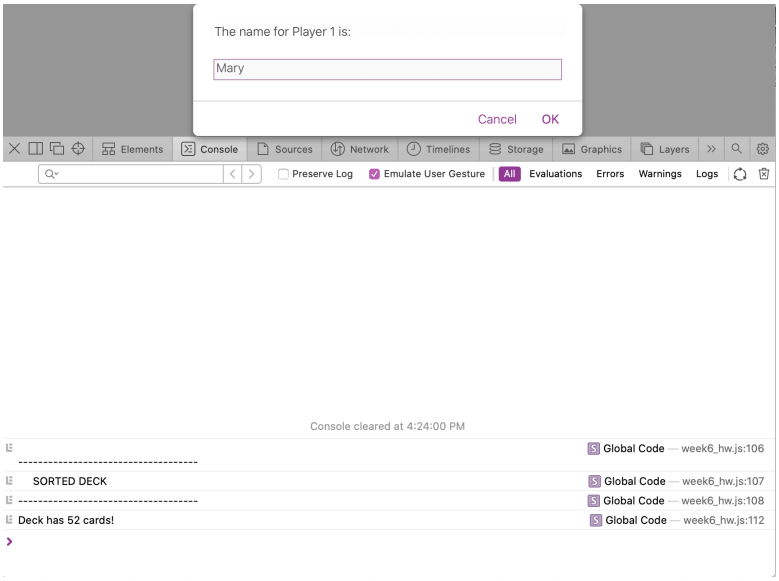
week6_tests.js > describe('LisasWeek6Functions') callback > describe('#populateCards') callback > it('should

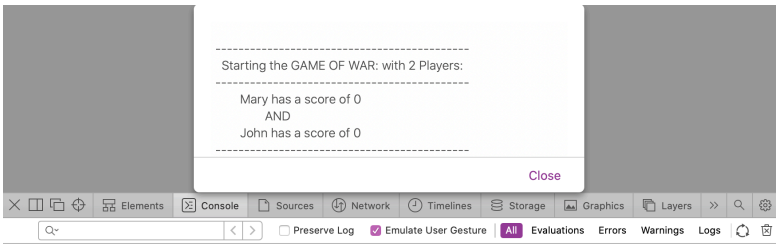
```
// imports the chai expect method
var expect = chai.expect;

// takes a "name", and a function that sets up the test
describe('LisasWeek6Functions', function() {
    describe('#populateCards', function() {

        it('should create a 52 card array', function() {
            // actual call to function
            var deck = new Deck();
            var cards = deck.populateCards();
            expect(cards).length(52);
        });

        it('should create 4 of each card value', function() {
            // create an array with a place for each card value.
            var cardCount = [0,0,0,0,0,0,0,0,0,0,0,0,0];
            var deck = new Deck();
            var cards = deck.populateCards();
            // count each value, and increment the array position
            for (let card of cards) {
                cardCount[card.value-2]++;
            };
            var number = cardCount.reduce(function(previous,current) {
                return previous + current;
            }, 0);
            number = number/13;
            expect(number).to.equals(4);
        });
```

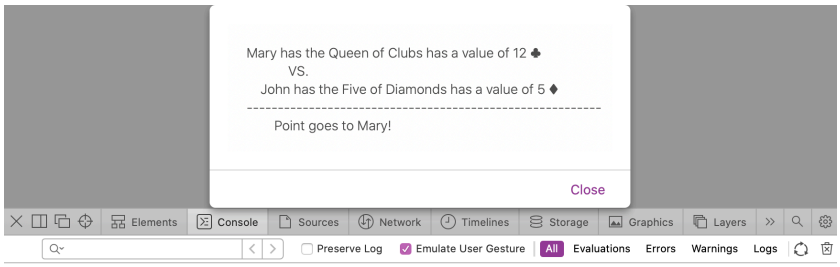
PROMINEO TECH

```
30
31          it('should create 13 of each suit value', function() {
32              // create an array with a place for each card value.
33              var heartCount = 0;
34              var spadeCount = 0;
35              var clubCount = 0;
36              var diamondCount = 0;
37              var deck = new Deck();
38              var cards = deck.populateCards();
39              // count each value, and increment the array position
40              for (let card of cards) {
41                  if (card.suit == "Hearts") {
42                      heartCount++;
43                  } else if (card.suit == "Spades") {
44                      spadeCount++;
45                  } else if (card.suit == "Diamonds") {
46                      diamondCount++;
47                  } else if (card.suit == "Clubs") {
48                      clubCount++;
49                  }
50              };
51              expect(heartCount).to.equals(13);
52              expect(spadeCount).to.equals(13);
53              expect(clubCount).to.equals(13);
54              expect(diamondCount).to.equals(13);
55          });
56
57          // it('should throw an error if first parameter is not a string', function() {
58          //     // wrapped in an expect, because we expect it to throw an error
59          //     expect(function() {
60          //         populateCards();
61          //     }).to.throw(Error);
62          // });
63      });
64
65  });
```

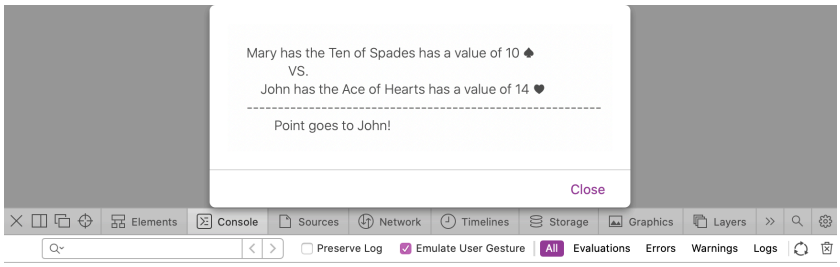
Screenshots of Running Application:

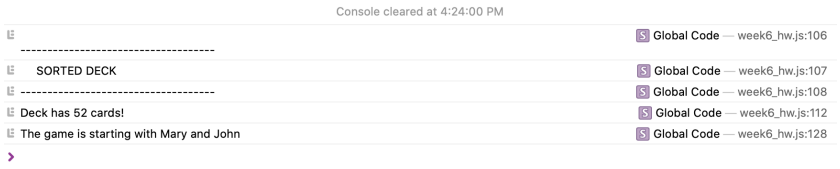


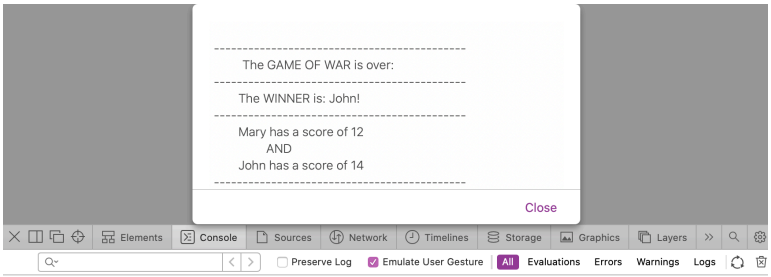


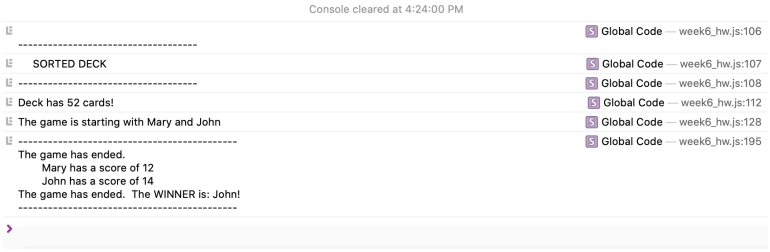

Mary has the Queen of Clubs has a value of 12 ♣
VS.
John has the Five of Diamonds has a value of 5 ♦
Point goes to Mary!
Close
Elements
Console
Sources
Network
Timelines
Storage
Graphics
Layers
Preserve Log
Emulate User Gesture
All
Evaluations
Errors
Warnings
Logs


Console cleared at 4:24:00 PM
Global Code — week6_hw.js:106
SORTED DECK
Global Code — week6_hw.js:107
Global Code — week6_hw.js:108
Deck has 52 cards!
Global Code — week6_hw.js:112
The game is starting with Mary and John
Global Code — week6_hw.js:128


Mary has the Ten of Spades has a value of 10 ♠
VS.
John has the Ace of Hearts has a value of 14 ♥
Point goes to John!
Close
Elements
Console
Sources
Network
Timelines
Storage
Graphics
Layers
Preserve Log
Emulate User Gesture
All
Evaluations
Errors
Warnings
Logs

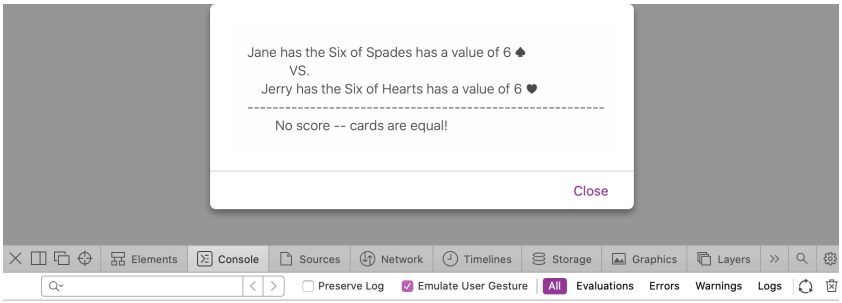

Console cleared at 4:24:00 PM
Global Code — week6_hw.js:106
SORTED DECK
Global Code — week6_hw.js:107
Global Code — week6_hw.js:108
Deck has 52 cards!
Global Code — week6_hw.js:112
The game is starting with Mary and John
Global Code — week6_hw.js:128

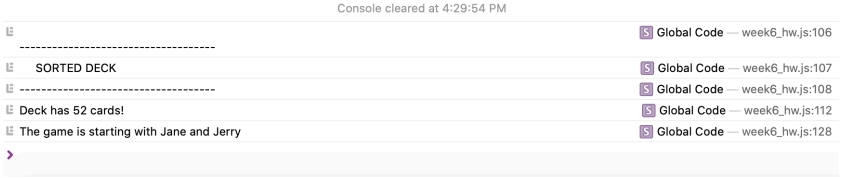


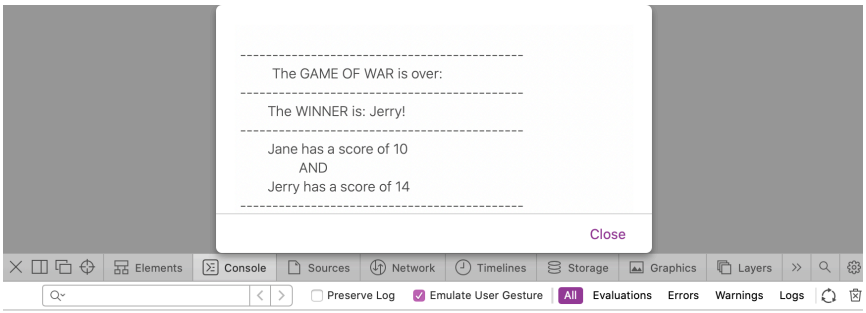


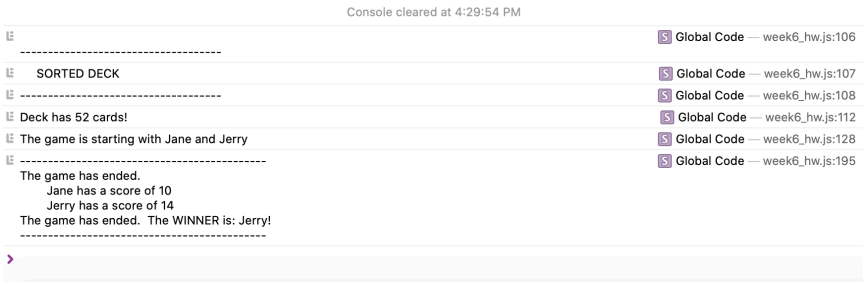
This run did not have any tied cards, so I ran the program again.

Here is a screenshot from that run of the game:

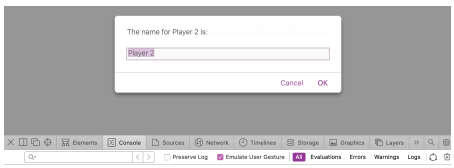


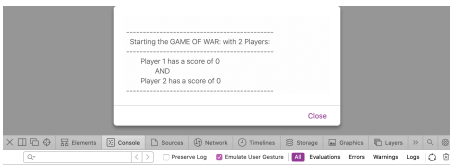


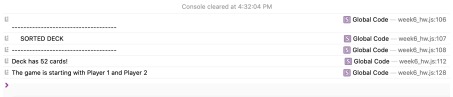


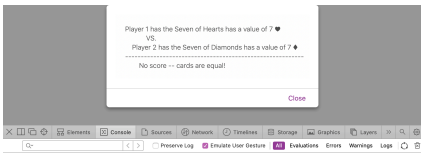


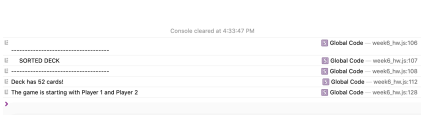
I added error checking for the Players Names. If no names are entered, then they resort to "Player 1" and "Player 2"

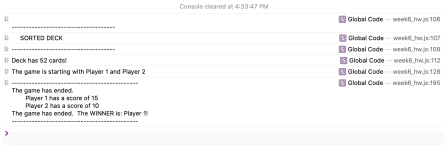












Unit Test Screen — three tests validating a new deck: