PROMINEO TECH

Intro to JavaScript Week 6 Coding Assignment

Points possible: 100

URL to GitHub Repository:

https://github.com/pat-in-a-hat/week_6_war_card_game

URL to Your Coding Assignment Video:

https://drive.google.com/file/d/1KDmoc-Juv2wVvyAWAl8bEVDQDPaFpZJS/view?usp=sharing

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.

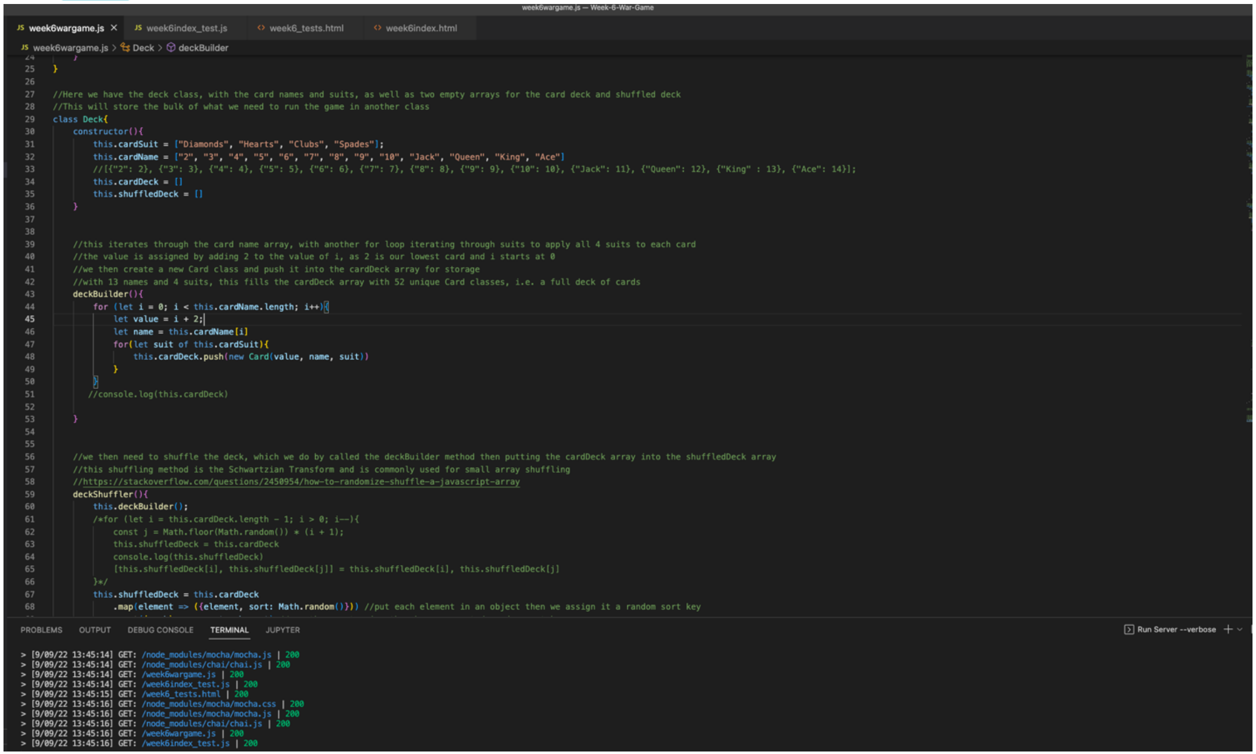
There are many versions of the game *WAR,* but in this version there are only 2 players and you don't need to do anything special when there is a tie on a round.

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.

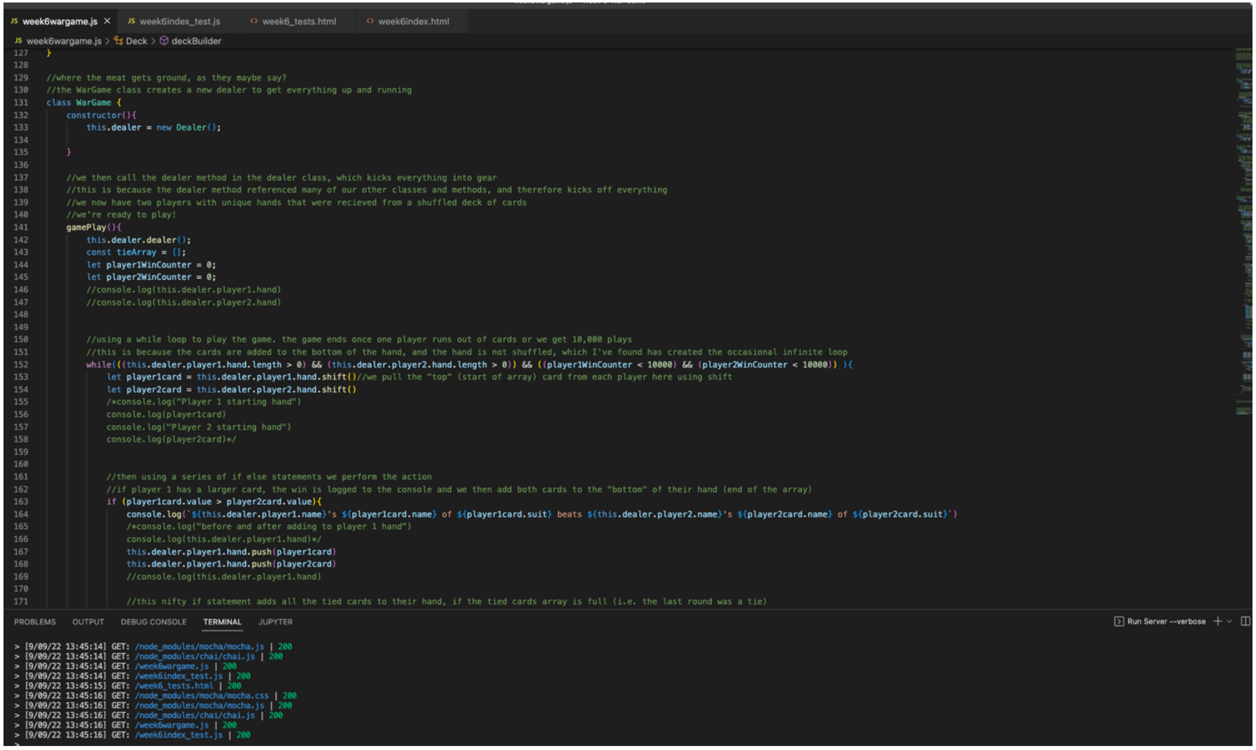
Write a Unit Test using Mocha and Chai for at least one of the functions you write.

Screenshots of Code:



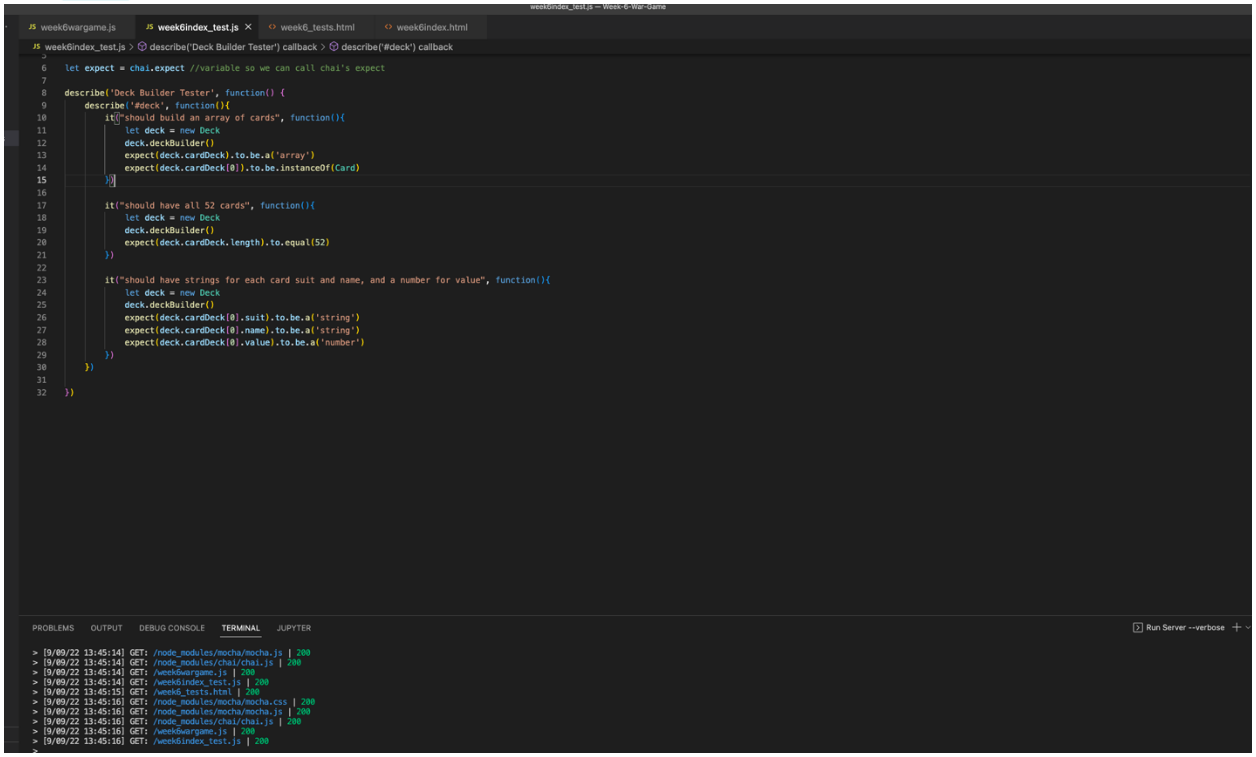
```javascript
}

//Here we have the deck class, with the card names and suits, as well as two empty arrays for the card deck and shuffled deck
//This will store the bulk of what we need to run the game in another class
class Deck{
    constructor(){
        this.cardSuit = ["Diamonds", "Hearts", "Clubs", "Spades"];
        this.cardName = ["2", "3", "4", "5", "6", "7", "8", "9", "10", "Jack", "Queen", "King", "Ace"]
        //[{"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}];
        this.cardDeck = []
        this.shuffledDeck = []
    }


    //this iterates through the card name array, with another for loop iterating through suits to apply all 4 suits to each card
    //the value is assigned by adding 2 to the value of i, as 2 is our lowest card and i starts at 0
    //we then create a new Card class and push it into the cardDeck array for storage
    //with 13 names and 4 suits, this fills the cardDeck array with 52 unique Card classes, i.e. a full deck of cards
    deckBuilder(){
        for (let i = 0; i < this.cardName.length; i++){
            let value = i + 2;
            let name = this.cardName[i]
            for(let suit of this.cardSuit){
                this.cardDeck.push(new Card(value, name, suit))
            }
        }
       //console.log(this.cardDeck)

    }


    //we then need to shuffle the deck, which we do by called the deckBuilder method then putting the cardDeck array into the shuffledDeck array
    //this shuffling method is the Schwartzian Transform and is commonly used for small array shuffling
    //https://stackoverflow.com/questions/2450954/how-to-randomize-shuffle-a-javascript-array
    deckShuffler(){
        this.deckBuilder();
        /*for (let i = this.cardDeck.length - 1; i > 0; i--){
            const j = Math.floor(Math.random()) * (i + 1);
            this.shuffledDeck = this.cardDeck
            console.log(this.shuffledDeck)
            [this.shuffledDeck[i], this.shuffledDeck[j]] = this.shuffledDeck[i], this.shuffledDeck[j]
        }*/
        this.shuffledDeck = this.cardDeck
            .map(element => ({element, sort: Math.random()})) //put each element in an object then we assign it a random sort key
```

```
> [9/09/22 13:45:14] GET: /node_modules/mocha/mocha.js | 200
> [9/09/22 13:45:14] GET: /node_modules/chai/chai.js | 200
> [9/09/22 13:45:14] GET: /week6wargame.js | 200
> [9/09/22 13:45:14] GET: /week6index_test.js | 200
> [9/09/22 13:45:15] GET: /week6_tests.html | 200
> [9/09/22 13:45:16] GET: /node_modules/mocha/mocha.css | 200
> [9/09/22 13:45:16] GET: /node_modules/mocha/mocha.js | 200
> [9/09/22 13:45:16] GET: /node_modules/chai/chai.js | 200
> [9/09/22 13:45:16] GET: /week6wargame.js | 200
> [9/09/22 13:45:16] GET: /week6index_test.js | 200
```



```javascript
}

//where the meat gets ground, as they maybe say?
//the WarGame class creates a new dealer to get everything up and running
class WarGame {
    constructor(){
        this.dealer = new Dealer();

    }

    //we then call the dealer method in the dealer class, which kicks everything into gear
    //this is because the dealer method referenced many of our other classes and methods, and therefore kicks off everything
    //we now have two players with unique hands that were recieved from a shuffled deck of cards
    //we're ready to play!
    gamePlay(){
        this.dealer.dealer();
        const tieArray = [];
        let player1WinCounter = 0;
        let player2WinCounter = 0;
        //console.log(this.dealer.player1.hand)
        //console.log(this.dealer.player2.hand)


        //using a while loop to play the game. the game ends once one player runs out of cards or we get 10,000 plays
        //this is because the cards are added to the bottom of the hand, and the hand is not shuffled, which I've found has created the occasional infinite loop
        while(((this.dealer.player1.hand.length > 0) && (this.dealer.player2.hand.length > 0)) && ((player1WinCounter < 10000) && (player2WinCounter < 10000)) ){
            let player1card = this.dealer.player1.hand.shift()//we pull the "top" (start of array) card from each player here using shift
            let player2card = this.dealer.player2.hand.shift()
            /*console.log("Player 1 starting hand")
            console.log(player1card)
            console.log("Player 2 starting hand")
            console.log(player2card)*/


            //then using a series of if else statements we perform the action
            //if player 1 has a larger card, the win is logged to the console and we then add both cards to the "bottom" of their hand (end of the array)
            if (player1card.value > player2card.value){
                console.log(`${this.dealer.player1.name}'s ${player1card.name} of ${player1card.suit} beats ${this.dealer.player2.name}'s ${player2card.name} of ${player2card.suit}`)
                /*console.log("before and after adding to player 1 hand")
                console.log(this.dealer.player1.hand)*/
                this.dealer.player1.hand.push(player1card)
                this.dealer.player1.hand.push(player2card)
                //console.log(this.dealer.player1.hand)

                //this nifty if statement adds all the tied cards to their hand, if the tied cards array is full (i.e. the last round was a tie)
```

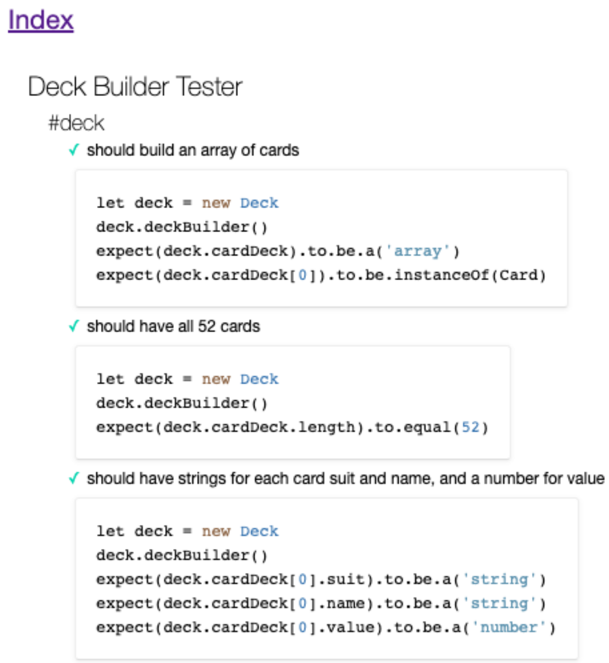
```
> [9/09/22 13:45:14] GET: /node_modules/mocha/mocha.js | 200
> [9/09/22 13:45:14] GET: /node_modules/chai/chai.js | 200
> [9/09/22 13:45:14] GET: /week6wargame.js | 200
> [9/09/22 13:45:14] GET: /week6index_test.js | 200
> [9/09/22 13:45:15] GET: /week6_tests.html | 200
> [9/09/22 13:45:16] GET: /node_modules/mocha/mocha.css | 200
> [9/09/22 13:45:16] GET: /node_modules/mocha/mocha.js | 200
> [9/09/22 13:45:16] GET: /node_modules/chai/chai.js | 200
> [9/09/22 13:45:16] GET: /week6wargame.js | 200
> [9/09/22 13:45:16] GET: /week6index_test.js | 200
```


week6index_test.js — Week-6-War-Game
week6wargame.js
week6index_test.js
week6_tests.html
week6index.html
week6index_test.js > describe('Deck Builder Tester') callback > describe('#deck') callback
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
JUPYTER
Run Server --verbose

```javascript
let expect = chai.expect //variable so we can call chai's expect

describe('Deck Builder Tester', function() {
    describe('#deck', function(){
        it("should build an array of cards", function(){
            let deck = new Deck
            deck.deckBuilder()
            expect(deck.cardDeck).to.be.a('array')
            expect(deck.cardDeck[0]).to.be.instanceOf(Card)
        })

        it("should have all 52 cards", function(){
            let deck = new Deck
            deck.deckBuilder()
            expect(deck.cardDeck.length).to.equal(52)
        })

        it("should have strings for each card suit and name, and a number for value", function(){
            let deck = new Deck
            deck.deckBuilder()
            expect(deck.cardDeck[0].suit).to.be.a('string')
            expect(deck.cardDeck[0].name).to.be.a('string')
            expect(deck.cardDeck[0].value).to.be.a('number')
        })
    })

})
```

```
> [9/09/22 13:45:14] GET: /node_modules/mocha/mocha.js | 200
> [9/09/22 13:45:14] GET: /node_modules/chai/chai.js | 200
> [9/09/22 13:45:14] GET: /week6wargame.js | 200
> [9/09/22 13:45:14] GET: /week6index_test.js | 200
> [9/09/22 13:45:15] GET: /week6_tests.html | 200
> [9/09/22 13:45:16] GET: /node_modules/mocha/mocha.css | 200
> [9/09/22 13:45:16] GET: /node_modules/mocha/mocha.js | 200
> [9/09/22 13:45:16] GET: /node_modules/chai/chai.js | 200
> [9/09/22 13:45:16] GET: /week6wargame.js | 200
> [9/09/22 13:45:16] GET: /week6index_test.js | 200
>
```

Screenshots of Running Application:



Index

Deck Builder Tester

#deck

✓ should build an array of cards

```
let deck = new Deck
deck.deckBuilder()
expect(deck.cardDeck).to.be.a('array')
expect(deck.cardDeck[0]).to.be.instanceOf(Card)
```

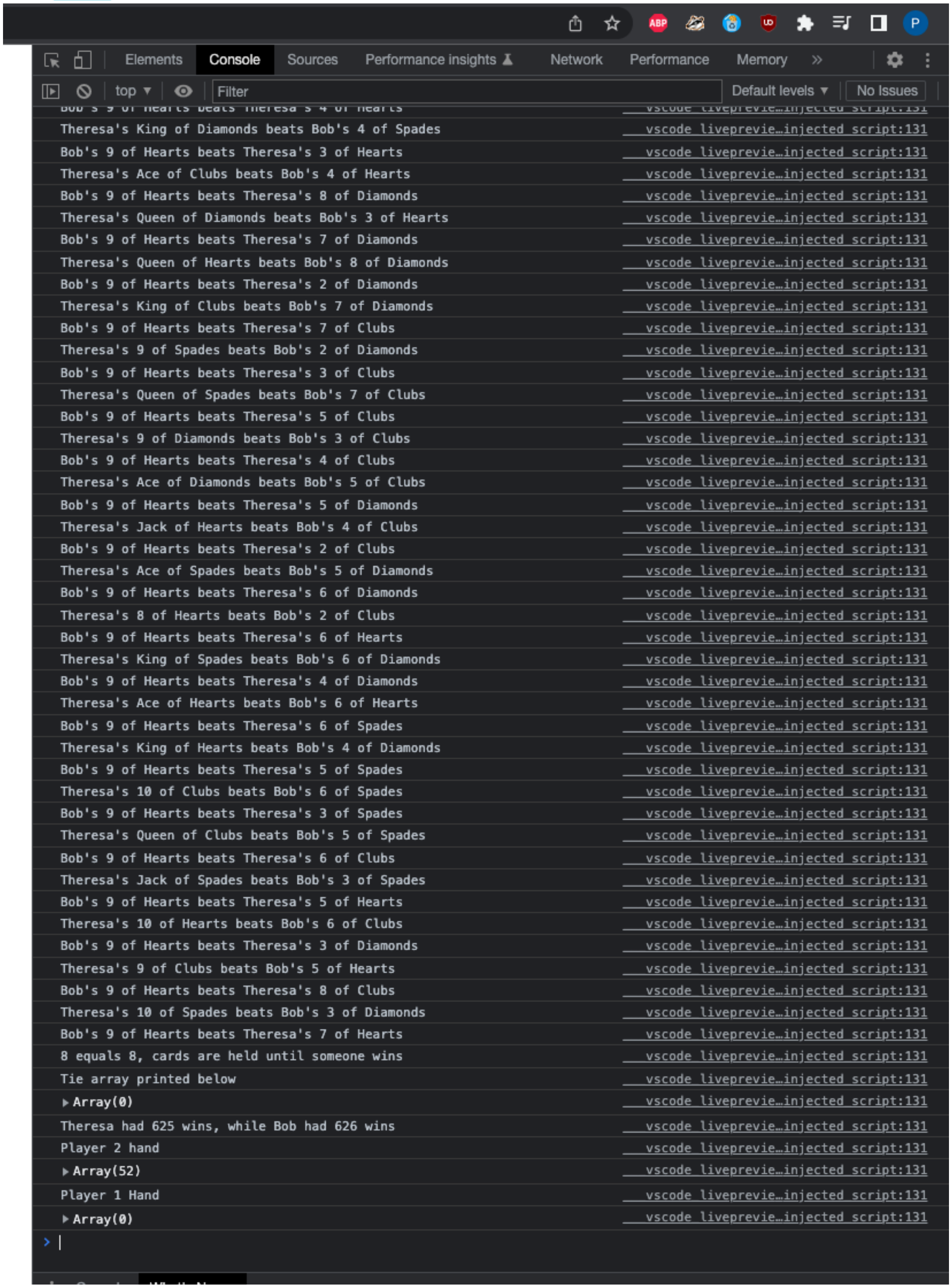
✓ should have all 52 cards

```
let deck = new Deck
deck.deckBuilder()
expect(deck.cardDeck.length).to.equal(52)
```

✓ should have strings for each card suit and name, and a number for value

```
let deck = new Deck
deck.deckBuilder()
expect(deck.cardDeck[0].suit).to.be.a('string')
expect(deck.cardDeck[0].name).to.be.a('string')
expect(deck.cardDeck[0].value).to.be.a('number')
```

PROMINEO TECH



```
Bob's 9 of Hearts beats Theresa's 4 of Hearts                   vscode liveprevie…injected script:131
Theresa's King of Diamonds beats Bob's 4 of Spades              vscode liveprevie…injected script:131
Bob's 9 of Hearts beats Theresa's 3 of Hearts                   vscode liveprevie…injected script:131
Theresa's Ace of Clubs beats Bob's 4 of Hearts                  vscode liveprevie…injected script:131
Bob's 9 of Hearts beats Theresa's 8 of Diamonds                 vscode liveprevie…injected script:131
Theresa's Queen of Diamonds beats Bob's 3 of Hearts             vscode liveprevie…injected script:131
Bob's 9 of Hearts beats Theresa's 7 of Diamonds                 vscode liveprevie…injected script:131
Theresa's Queen of Hearts beats Bob's 8 of Diamonds             vscode liveprevie…injected script:131
Bob's 9 of Hearts beats Theresa's 2 of Diamonds                 vscode liveprevie…injected script:131
Theresa's King of Clubs beats Bob's 7 of Diamonds               vscode liveprevie…injected script:131
Bob's 9 of Hearts beats Theresa's 7 of Clubs                    vscode liveprevie…injected script:131
Theresa's 9 of Spades beats Bob's 2 of Diamonds                 vscode liveprevie…injected script:131
Bob's 9 of Hearts beats Theresa's 3 of Clubs                    vscode liveprevie…injected script:131
Theresa's Queen of Spades beats Bob's 7 of Clubs                vscode liveprevie…injected script:131
Bob's 9 of Hearts beats Theresa's 5 of Clubs                    vscode liveprevie…injected script:131
Theresa's 9 of Diamonds beats Bob's 3 of Clubs                  vscode liveprevie…injected script:131
Bob's 9 of Hearts beats Theresa's 4 of Clubs                    vscode liveprevie…injected script:131
Theresa's Ace of Diamonds beats Bob's 5 of Clubs                vscode liveprevie…injected script:131
Bob's 9 of Hearts beats Theresa's 5 of Diamonds                 vscode liveprevie…injected script:131
Theresa's Jack of Hearts beats Bob's 4 of Clubs                 vscode liveprevie…injected script:131
Bob's 9 of Hearts beats Theresa's 2 of Clubs                    vscode liveprevie…injected script:131
Theresa's Ace of Spades beats Bob's 5 of Diamonds               vscode liveprevie…injected script:131
Bob's 9 of Hearts beats Theresa's 6 of Diamonds                 vscode liveprevie…injected script:131
Theresa's 8 of Hearts beats Bob's 2 of Clubs                    vscode liveprevie…injected script:131
Bob's 9 of Hearts beats Theresa's 6 of Hearts                   vscode liveprevie…injected script:131
Theresa's King of Spades beats Bob's 6 of Diamonds              vscode liveprevie…injected script:131
Bob's 9 of Hearts beats Theresa's 4 of Diamonds                 vscode liveprevie…injected script:131
Theresa's Ace of Hearts beats Bob's 6 of Hearts                 vscode liveprevie…injected script:131
Bob's 9 of Hearts beats Theresa's 6 of Spades                   vscode liveprevie…injected script:131
Theresa's King of Hearts beats Bob's 4 of Diamonds              vscode liveprevie…injected script:131
Bob's 9 of Hearts beats Theresa's 5 of Spades                   vscode liveprevie…injected script:131
Theresa's 10 of Clubs beats Bob's 6 of Spades                   vscode liveprevie…injected script:131
Bob's 9 of Hearts beats Theresa's 3 of Spades                   vscode liveprevie…injected script:131
Theresa's Queen of Clubs beats Bob's 5 of Spades                vscode liveprevie…injected script:131
Bob's 9 of Hearts beats Theresa's 6 of Clubs                    vscode liveprevie…injected script:131
Theresa's Jack of Spades beats Bob's 3 of Spades                vscode liveprevie…injected script:131
Bob's 9 of Hearts beats Theresa's 5 of Hearts                   vscode liveprevie…injected script:131
Theresa's 10 of Hearts beats Bob's 6 of Clubs                   vscode liveprevie…injected script:131
Bob's 9 of Hearts beats Theresa's 3 of Diamonds                 vscode liveprevie…injected script:131
Theresa's 9 of Clubs beats Bob's 5 of Hearts                    vscode liveprevie…injected script:131
Bob's 9 of Hearts beats Theresa's 8 of Clubs                    vscode liveprevie…injected script:131
Theresa's 10 of Spades beats Bob's 3 of Diamonds                vscode liveprevie…injected script:131
Bob's 9 of Hearts beats Theresa's 7 of Hearts                   vscode liveprevie…injected script:131
8 equals 8, cards are held until someone wins                   vscode liveprevie…injected script:131
Tie array printed below                                         vscode liveprevie…injected script:131
▸ Array(0)                                                      vscode liveprevie…injected script:131
Theresa had 625 wins, while Bob had 626 wins                    vscode liveprevie…injected script:131
Player 2 hand                                                   vscode liveprevie…injected script:131
▸ Array(52)                                                     vscode liveprevie…injected script:131
Player 1 Hand                                                   vscode liveprevie…injected script:131
▸ Array(0)                                                      vscode liveprevie…injected script:131
> |
```