Jasmine Godoy

Link:
https://www.awesomescreenshot.com/video/11707761?key=992eda8a33d108821e3b027dd93b3666

```html
<!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">
    <title>War Game</title>

</head>
<body>
    <script src="Week 6 Class Card.js"></script>
</body>
</html>
```

```
1
2  //here we construct how the player gets the hand, what they're score will be
   and the name.
3  import card from ./Class deck.js
4
5  class Player {
6    hand =[]
7    score = 0
8    constructor(name) {
9      this.name = name
10   }
11   // we return the name of the player on the screen
12   getPlayerName() {
13     return '$(this.name)'
14   }
15   // so the player can check they're cards we add a return for that too
16   checkHand() {
17     for (const card of this.hand) {
18         console.log(card)
19     }
20     return this.hand.length
21    //we use the push method to get an available card form the deck
22   }
23   getCard(card) {
24      this.hand.push(card)
25    }
26    //we use the pop method to element an item everytime the player uses a card
27   playCard() {
28      return this.hand.pop()
29   }
30   //everytime the player gets it right they get a score increase
31   increaseScore() {
32     this.score = 1
33   }
34   // and at the end of it they're able to recieve what they're score will be
   by the end of the game
35   getScore() {
36     return this.score
37   }
38 }
39 // all of these are used display whether the player can pick up,recieve,play,
   and check their hand.
40 console.log('$(player1.getPlayerName') received $(deck.deal())
```

```
export default class Card {
 constructor(suit,rank) {
  this.suit = suit
  this.rank = rank

 }

 getValue() {
 switch(this.rank.toLowerCase()) {
  case 'a':
    return 14
  case 'b':
    return 13
  case 'c':
    return 12
  case 'd':
    return 11
  default:
    return Number(this.rank)
 }
}
 getCard() {
   return `${this.rank} of ${this.suit}`
 }
}

const q = NewCard('club','0')
console.log(q.getvalue())
console.log(q.getCard())


```

```
// construction of the deck such as the name, number is listed as. We also push the nak and suit to display witht he new cards
// everytime we make a game the structure goes like this
//make the players, make the enviroment by stating the constructors and then add the actions what will the player do,what weapons will he have
//if theirs no weapons what do the characters do

import card from './Week 6 Classic Card.js'

export default class Deck {
    cards = []
    suits = ['clubs', 'diamonds', 'hearts','spades']
    ranks = ['2','3','4','5','6','7','8','10','J','Q','K','A']
    constuctor() {
        for (const suit of this.suits) {
            for (const rank of this.ranks) {
                this.cards.push(new Cards(suit,rank))
            }
        }
    }
    //if the const card is this.cards it will console.log "card" to display to check through the cards.
    checkCards() {
        for (const card of this.cards) {
            console.log(card)
        }
        return this.cards.length
    }

    //we want the cards to loop through and give us a random number to do that we'll loop through

    shuffleCards() {
       for (i < 0; i < this.cards.length; i++)
         const randomIndex = Math.floor(Math.random() * 52)
         const tempCard = this.cards[i]
         this.cards[1] = this.cards[randomIndex]
         this.cards[randomIndex] = tempCard
        }
    }

   dealCard(){
     return this.cards.pop();
   }
}



//const newDeck = new Deck();
//newDeck.shuffleCards();
//console.log(`card dealt is: `,newDeck.dealCard())

//console.log(newDeck.checkCards())


```

```
// we create a class for what the card would hold which is the suit, the ranking, and value within the contructor meaning where we construct the parts of the cards

import card ./

export class Card [] {
  constructor(rank,suit,value)
    this.suit = suit;
    this.rank = rank;
    this.value = value;
  }

 // we create another class for the deck it will at "this.cards" insert the set of cards

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

  // at getCards the player can choose a card from the deck which will be returned to them.

  getCards() {
    return this.cards;
  }

  //at buildDeck we want it to run and populate through the deck and also shuffle
  //we return the cards right after

  buildDeck() {
    this.populate();
    this.shuffle();
    return this.cards;
  }
 //Populate is defined as the cards names, ranking and value we have them all listed here.
 //We define the suits as the type of cards, we define the ranks as the tier, and we define the value as the number it holds

  populate() {
   const suits = ['Spades', 'Hearts', 'Clubs','Diamonds']
    const ranks = ['A','2','3','4','5','6','7','8','9','10','J','Q','K'];
    const values = ['1','2','3','4','5','6','7','8','9','10','11','12','13'];

    // let the indx be equal to 0, then i if I is less then the length og the suit array ('Spades',etcs) it will increment by 1
    //Same goes with the ranks
    // we want to loop through the amount of type of cards and also loop through the ranks of the cards then push the variable newcards as the rank suit and values.

    for (let i = 0; i < suits.length;i++) {
      for (let j = 0; j < ranks.length; j++) {
       this.cards.push(newCard(ranks[j],suits[j],values[j]));
      }
    }
```

```
  }

  // we want the deck to shuffle through the cards so we implement a statement thats says
  // if the length of the cards is greater then 0 then the deck being shuffled with include sorting the cards,randomizing it by subtracting by 0.5

  shuffle() {
     if (this.cards.length > 0) {
     const shuffleDeck = this.cards.sort(() => Math.random() - 0.5)
     this.cards = [shuffleDeck];
    }
   }
}

//when it comes to the players we want to construct by using a constucter that would name the score,deck, and name of the player
//we write each variable to then display. By writing this.(name of variable,action,object) we can then equal it to the name,object or action.

class Player {

    constructor(name) {
     this.playerName = name;
     this.playerScore = 0;
     this.playerDeck = [];
    }
    get name() {
        return this.playerName;
    }
    get playerScore () {
        return this.playerScore;
    }
    get deck(newDeck) {
      if(Array.isArray(newDeck)) {
        return this.playerDeck = newDeck;
      }
    }
    set score(newScore) {
      if (!isNaN(newScore)) {
        this.playerScore = newScore;
      }
    }

}


//Now this is where we name off the Game it self and construct what we will be implementing and displaying
// we create a constructor to place the players and deck in that we created before

class WarGame {

    constructor () {
      this.players = [];
      this.deck = [];
    }

```

```
 //Heres where we actually start. This is where we actually implement all fo the variables,objects,actions that we created by making arrays using methods and listing off objects and names.
 // We start off by displaying the tile where it says console.log,then we want it to display a prompt we do this by wriitng it out in brackets
 //we implement a switch to list all of the options ^ means its an exit click on that key to initiate it and so on.
 //at the end our input should equal our prompts

  start() {


    console.log('*** War Game ***')
    let input = prompt('0 - Exit; 1 - Play; 2 - Look at game instructions');
     while (input != '^') {
        switch (input) {
            case '^':
              exit;S
            case '1':
              this.createGame();
            case '2':
              this.instructions();
              break;
         }
         input = prompt(`0 - Exit; 1 - Play; 2 - click here to learn how to play`):
       }

}

//now that we've listed all the actions,named the players, and objects we need to initate the players.
//player 1 will equal to a new player same goes for two by using this.(variable name).
//we create a var called cards and equal it to a new that that will be built everytime
//we use the slice method to split my deck in half. Then we connsole log the title (dealing hands)
//we create a statement if we let equal 0 and  i is less than this.(our players) but also if ourplayers,deck and value are great then the others players deck and value are we
//we will get an output of our players score being equal to 1
//let the winning hand equal it

createGame() {
    this.players[0] = new player ('Player 1')
    this.players[1] = new player ('Player 2')

    const cards = new this.Deck().buildDeck();

    this.players[0].deck = [cards.slice(0,26)];
    this.players[1].deck = [cards.slice(26,52)];

    console.log('******* Dealing Hands ********')
    for (let i = 0; i < this.players[0].deck.length; i++) {
        if (this.players[0].deck[i].value > this.players[i].deck[i].value) {
            this.players[0].score +=1;

        let winningHand = '${this.players[0].deck[i].rank} of ${this.players[0].deck[i].suit}';
```

```
        console.log('Player 1 won with a ${winnningHand}');
        }
    }
}

    console.log('******* Hands Finished ******')


    if(this.players[0].score > this.players[1].score) {
    console.log('$(this.players[0].name.toUppercase()} WON THE WAR with a score of $(this.players[0].score}');
    } else if (this.players[0].score < this.players[1].score) {
    console.log('$(this.players[1].name.toUpperCase()} WON THE WAR with a score of $(this.players[1].score}');
    } else {
    console.log('Player one and Player tied');
  }
 }

// a const war game where it starts

const game = new WarGame();
game.start();


```