# Exercise: BlackJack

Grade: N/A

### Step 1: Arrange the table

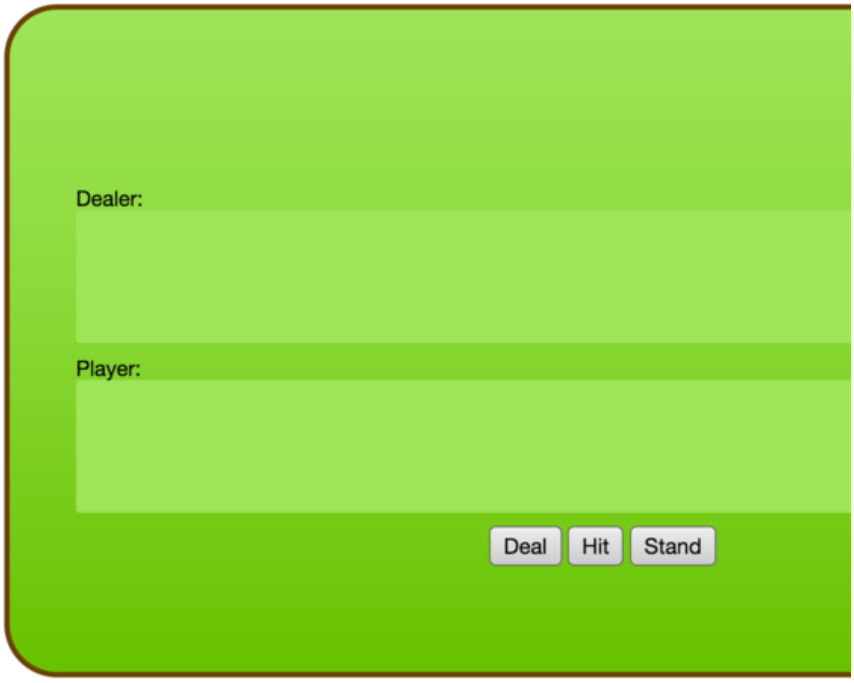
Create an HTML page and put the following markup in the body:

```
<div id="table">
<div id="messages"></div>
<label>Dealer: </label>
<label id="dealer-points" class="points"></label>
<div id="dealer-hand" class="hand">
</div>
<label>Player: </label>
<label id="player-points" class="points"></label>
<div id="player-hand" class="hand">
</div>
<div class="buttons">
<button id="deal-button">Deal</button>
<button id="hit-button">Hit</button>
<button id="stand-button">Stand</button>
</div>
</div>
```

Include a link to `blackjack.js` file of your html page.

### Step 2: Style the page

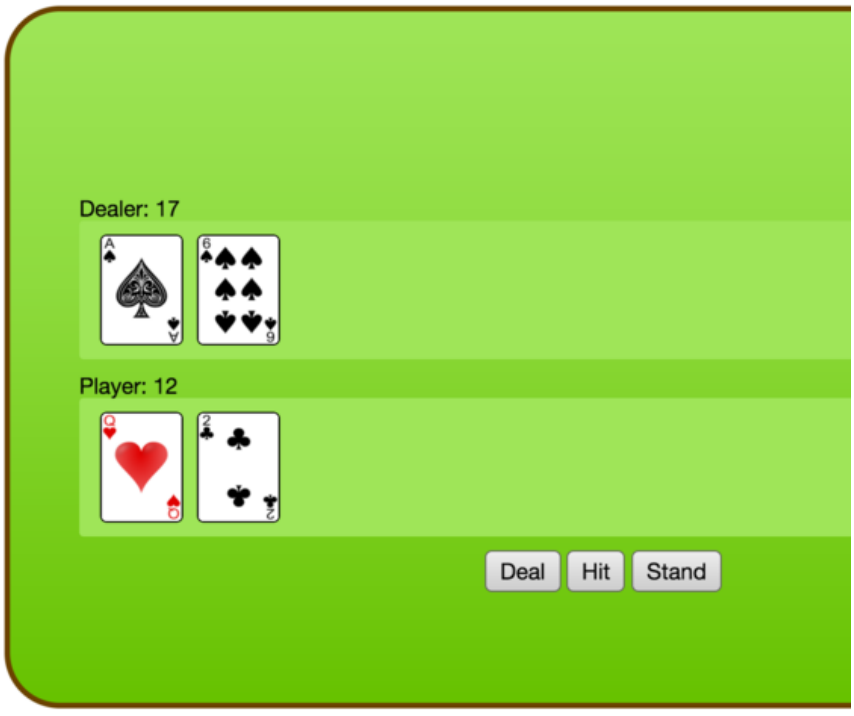
Style the page to look good, but not too good (this exercise is more about events than styling). Here is an example:




---

**Step 3: Cards on the table**

Put cards on the table by just adding `img` elements inside of the `#dealer-hand` or `#player-hand` elements (directly in the HTML) and have them link (set the src attribute) to card images in the images folder.




---

**Step 4: Dealing the cards**

Remove the cards you created in step 3. This time you will generate them dynamically using JavaScript. When the "Deal" button is clicked. Deal 2 cards each to the dealer's hand and the player's hand (It doesn't matter which cards at this point).

---

**Step 5: Hit me**

When the "Hit" button is clicked, deal one more card to the player (It doesn't matter which card).

---

**Step 6: A deck**

Create a deck of cards. A card has a point value and a suit. We will represent them as objects:

```
 var kingOfHearts = { point: 13, suit: 'hearts' };
var aceOfDiamonds = { point: 1, suit: 'diamonds' };
```

A deck of cards is thus represented as an array of these "card objects":

```
 var deck = [ { point: 13: suit: 'hearts' },
{ point: 1, suit: 'diamonds' }, ... ];
```

Generate a deck of 52 cards. A deck of cards has 4 cards for each point value from 1 to 13. For each point value, there are 4 different suits: diamonds, clubs, hearts and spades.

---

**Step 7: Deal the deck**

Now that you have a deck of cards, update your code to simulate dealing from that deck by

1. Taking away a card from it each time a card is dealt.
2. Saving the dealt card in two arrays variables in your program: `dealerHand` and `playerHand`. These arrays of cards will later be used to calculate the total points for the hand.
3. Displaying the card on the page corresponding to the card that was dealt.

For step 2 above, we have images in the images folder with the following naming scheme:

`_of_.png`

For example:

```
 5_of_hearts.png ace_of_spades.png
jack_of_diamonds.png
```

In order to put a card visually on the page, we need to insert an `<img>` tag, example:

`<img src="/page/images/5_of_hearts.png" alt="5 of hearts" />` We can use Javascript swap out the pictures in the `<img>` to the dealer's hand or the player's hand.

Except that we want to dynamically generate that `<img>` based on a card object, which is an object of the form: `{ point: 13: suit: 'hearts' }`.

Write a `function getCardImageUrl(card)`. It will take a card object as its first argument, and it will return a string containing the correct image URL for that card. For example, with the following code:

```
 var url = getCardImageUrl({ point: 13, suit: 'hearts' });
url should contain the string: "images/king_of_hearts.png".
```

Then use the `url` to generate an img element to display the image.

---

**Step 8: Calculate points for a hand**

Write a function calculatePoints that takes in an array of card objects and returns the points for that hand. Example:

```
 > calculatePoints([
     { point: 10, suit: 'diamonds' },
     { point: 12, suit: 'spades'}
])

20
> calculatePoints([
     { point: 10, suit: 'diamonds' },
     { point: 1, suit: 'clubs'}
])

21
```

---

**Step 9: Shuffle the deck**

Write a function to shuffle the deck. Here are some strategies for shuffling the deck:

1. For fixed number of times, randomly choose 2 cards within the deck and swap them.
2. Create a new array, randomly choose one card after another from the original array and push it to the new array.

3. Use the shuffle function to shuffle the deck.

---

**Step 9: Display points**

After dealing any card, display the current points for the dealer and the player.

---

**Step 10: Busts**

Check for busts after each card is dealt. A bust is when either the dealer or the player's hand go over 21 points. When that happens, they lose. If there is a bust, display a message on the page saying that they busted.

---

**Step 10: Player stands**

When the "Stand" button is clicked, the player doesn't want any more cards. Deal cards to the dealer until he reaches 17 points or more.

---

**Step 11: Determine winner**

When the dealer's turn is over, whoever has the most points wins. Determine the winner.

---

**Step 12: Restart game**

Allow for restarting the game.

---

**Extra Credit**

You can choose one or more of these extra credit features to implement:

- Play the game with 3 decks of cards (156 cards) or six deck of cards (312 cards).
- Use an animation to reveal a card when it is drawn.
- Hide dealer's hole card, and reveal it before dealer's turn.
- Keep track of wins vs losses.
- Add betting structure instead of wins vs losses. The player with start out with a certain amount of money - say $500. There will be a minimum bet of $5. The player can choose an amount to bet before each hand.

- Solutions

Posted Today at 8:59 am