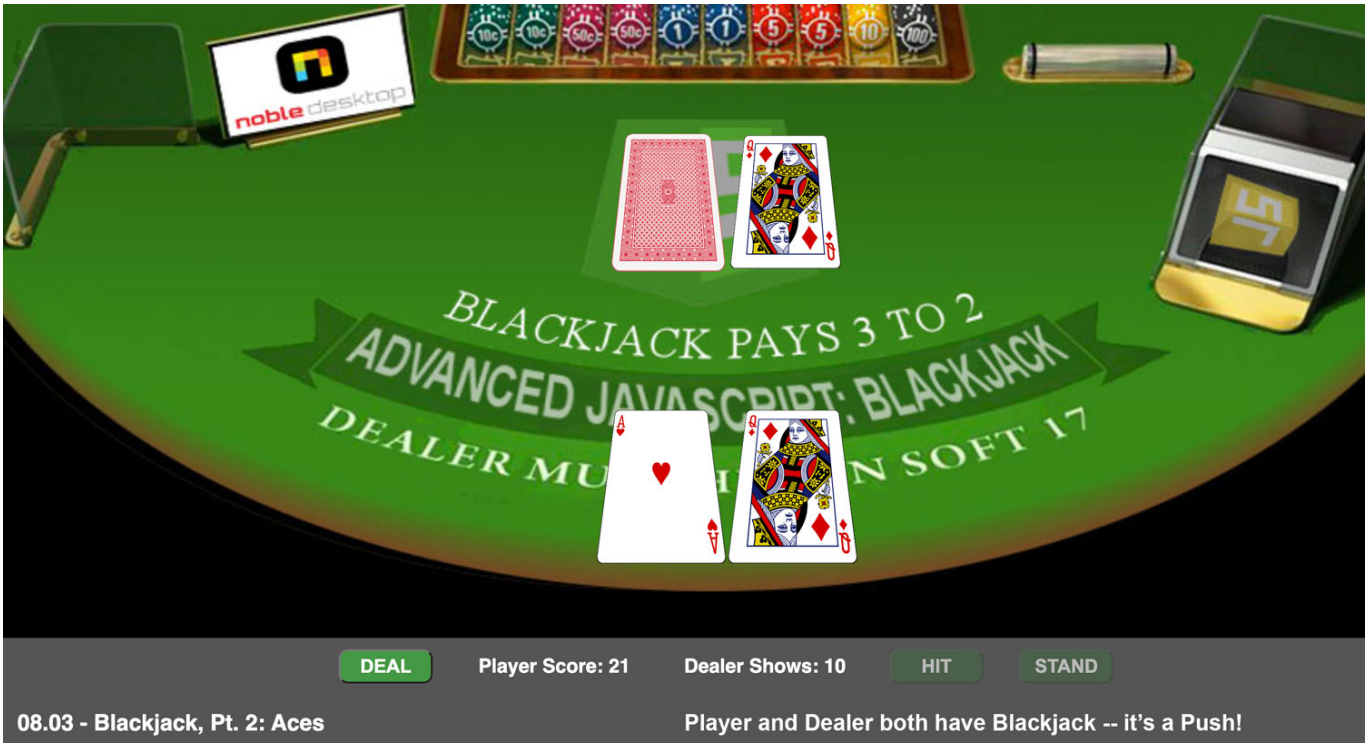

Blackjack - Pt. 2: Aces..!

evaluating Aces as worth either 11 or 1

testing the various Blackjack scenarios

The next thing we need to do for our blackjack application is attend to the tricky issue of how to handle Aces, which are worth either 11 or 1, depending on the score. For the purposes of this discussion, an "Ace 11" refers to an Ace whose value is 11, as opposed to 1.

evaluating an Ace: is it worth 11 or 1..?
If an "Ace 11" would bust the player or dealer, its value must be changed to 1:

- an Ace is initially assigned a value of 11 as its **valu** property
- if the player or dealer is dealt an Ace to start, the Ace is worth 11
- but if the player or dealer is dealt another Ace, the new Ace cannot also count as 11, as that would result in a score of 22, which is a bust. So, the second Ace counts as 1, which yields a score of 12.

keeping track of the Ace score separately
In order to set Ace values, we must keep track of Aces dealt. For this we will declare two new global variables, one for the dealer's Ace score, the other for the player's Ace score.

7. Above the **deal()** function, declare these two variables, **pAceScore** (player Ace score) and **dAceScore** (dealer Ace score):

```
    let pAceScore = 0; // player Ace score
    let dAceScore = 0; // dealer Ace score
```

Next, we need to go back into the DEAL function from the last lesson and account for Aces. If anyone has *one* Ace, that Ace counts as 11. Since you cannot bust on the deal, if anyone has *two* Aces, the second Ace is worth 1, for a score of 12.

8. Inside the **deal()** function, review the logic of the if-else block, where cards are dealt and scores are tallied. The numbering in the comments pertains to the steps of the *previous* lesson:

```
        // 24. Set up an if-else statement to handle the logic for dealing
two cards each to player and dealer, starting with the player.
        if(dealCounter % 2 == 1) {
            // 25. Output the card to the player's div
            playerCardsDiv.appendChild(pic);
            // 26. Push the card into the player's hand
            playerHand.push(card);
```

9. Right before "old step 27", check to see if the card is an Ace:

```
            if(card.kind == "Ace") {
            }

            // 27. Increment the player's score
            playerScore += card.valu;
            playerScoreDiv.innerHTML = "Player Score: " + playerScore;
```

10. We need a nested if here. Check to see if the **pAceScore** is 0, which tells us if this is the first Ace the player has been dealt:

```
            if(card.kind == "Ace") {
                if(pAceScore == 0) {
                }
            }
```

11. If **pAceScore** is 0, the player does *not* already have an Ace, so set **pAceScore** to 11:

```
            if(card.kind == "Ace") {
                if(pAceScore == 0) {
                    pAceScore = 11;
```

```
            }
        }
```

12. If the **pAceScore** is *not* 0, the player already *has* an Ace. A hand cannot have two "Ace 11's", so the second Ace counts 1. Right before "old step 27" where the player's score is updated, increase **pAceScore** to 12 *and* set the value of the card to 1. Log **pAceScore** to see if the logic is working:

```
        if(card.kind == "Ace") {
            if(pAceScore == 0) {
                pAceScore = 11;
            } else {
                pAceScore = 12;
                card.valu = 1;
            }
        }
        console.log('pAceScore:', pAceScore);

        // 27. Increment the player's score
        playerScore += card.valu;
        playerScoreDiv.innerHTML = "Player Score: " + playerScore;
```

13. The **else** handles dealer Aces in the same way. Copy the if-else from the previous step and paste it between "old steps" 31 and 32. Change **pAceScore** to **dAceScore**. Log **dAceScore** to see if it worked:

```
       // 28. Add the else part to handle dealers dealt to the dealer
       } else {
            // 29. Make the dealer cards a bit smaller, to make them
appear farther away
            pic.style.width = "105px";
            pic.style.height = "auto";
            // 30. Output the card to the dealer's div
            dealerCardsDiv.appendChild(pic);
            // 31. Push the card into the dealer's hand
            dealerHand.push(card);

            if(card.kind == "Ace") {
                if(dAceScore == 0) {
                    dAceScore = 11;
                } else {
                    dAceScore = 12;
                    card.valu = 1;
                }
            }
            console.log('dAceScore:', dAceScore);

            // 32. Update the dealer's score
            dealerScore += card.valu;
        }
```

With each clicking of the DEAL button, all global variables need to be reset.

14. Inside the deal function, as part of "old step 16", reset **pAceScore** and **pAceScore** to 0:

```
    // 15. Define the deal function:
    function deal() {
        // 16. Since this is a new hand, reset the scores and "clear the
table"
        pAceScore = 0; // reset player Ace score
        dAceScore = 0; // reset dealer Ace score
```

15. Run the application and deal a few hands. Eventually you should see an Ace. A check of the console will let you know if **pAceScore** and/or **dAceScore** equal 11, as expected.

Conducting a "Two Aces Test"
If we deal enough hands, we'll see an Ace sooner or later. But to get dealt two Aces to see if the score is 12, we will likely have to play dozens of hands. The way around this is to hard-code an Ace object as the dealt card.

16. In the deal function, in "old step 21", comment out the **card = shoe.pop()** line, where a card is dealt from the shoe, and hard-code an Ace:

```
    // 21. Pop a card object off the shoe array and save it as a new card
        // let card = shoe.pop();
        let card = {
            file: "Ace-of-Hearts.png",
            kind: "Ace",
            name: "Ace of Hearts",
            suit: "Hearts",
            valu: 11
        };
```

17. Refresh the browser and deal a hand. Both player and dealer should receive two Ace of Hearts.

18. Check the console. There should be output showing both **pAceScore** and **dAceScore**, which start at 11 and then increase to 12.

19. Now that the **"Two Aces Test"** is done, comment out the hard-coded Ace object, and restore the **let card = shoe.pop()** line, so that the game reverts to normal dealing behavior.

testing Blackjack scenarios
In the previous lesson, we wrote code for announcing Blackjack in the event that either the player or dealer (or both) are dealt 21. As a refresher, have a look at that code now, which comprises "old step 39":

```
        // 39. Check to see if either the player or dealer have Blackjack
        // Announce Blackjack on 1 second delay; if no one has Blackjack,
```

```
        // prompt player to HIT or STAND:
            setTimeout(() => {
                if(playerScore == 21 && dealerScore == 21) {
                     outputH2.innerHTML = 'Dealer and Player both have
Blackjack!<br>It's a Push!';
                } else if(playerScore == 21) {
                    outputH2.innerHTML = 'BLACKJACK! You win!';
                } else if(dealerScore == 21) {
                    outputH2.innerHTML = 'Dealer has BLACKJACK! You
lose!';
                } else { // no one has Blackjack
                    outputH2.innerHTML = "HIT or STAND?";
                }
            }, 1000);
         }
```

To test all these Blackjack scenarios, we need to hard-code Blackjack for the player, the dealer and both.

testing: player and dealer both have Blackjack
First we will test the rarest outcome of all: player and dealer both have Blackjack. We do this by hard-coding Blackjack hands for both.

20. In the **deal()** function, once again comment out **let card = shoe.pop()** line, and replace it with this if-else block that deals a Blackjack hand of Ace-Queen to both the player and the dealer.

- the first two cards are Aces: one each for player and dealer
- the last two cards are Queen: one each for player and dealer

```
        // 21. Pop a card off the shoe array and save it as a new card
        // let card = shoe.pop();
        let card;
        // the first two cards are Aces (one each for player and dealer)
        if(dealCounter < 3) {
            card = {
                file: "Ace-of-Hearts.png",
                kind: "Ace",
                name: "Ace of Hearts",
                suit: "Hearts",
                valu: 11
            };
        // the last two cards are Queens (one each for player and dealer)
        } else {
            card = {
                file: "Queen-of-Diamonds.png",
                kind: "Queen",
                name: "Queen of Diamonds",
                suit: "Diamonds",
                valu: 10
            };
        }
```

21. Test the game by dealing a hand. The player and dealer should both get Blackjack, with the output announcing this.

Dealer Blackjack has to be verified by revealing the hole card. To be able to do that, we need to have saved the hole card Image object to begin with. Then, if we need to prove that the dealer has Blackjack, we can change the hole card source from the back of the card to the front of the card.

22. Declare a variable called **holeCardPic**, but don't give it a value. This will be set equal to an Image object later. Do this right above the deal function:

```
    let holeCardPic;

    // 15. Define the deal function:
    function deal() {
```

23. Inside the setTimeout if-else block that assess blackjack, inside the two scenarios where the dealer has blackjack, call a function by the name of **showHoleCard()**, which we will write next:

```
    setTimeout(() => {
        if(playerScore == 21 && dealerScore == 21) {
            outputH2.innerHTML = "Dealer and Player both have Blackjack!
It's a Push!";
            showHoleCard();
        } else if(playerScore == 21) {
            outputH2.innerHTML = 'BLACKJACK! You win!';
        } else if(dealerScore == 21) {
            outputH2.innerHTML = 'Dealer has BLACKJACK! You lose!';
            showHoleCard();
        } else { // no one has Blackjack
            outputH2.innerHTML = "HIT or STAND?";
        }
    }, 1000);
```

24. Write the **showHoleCard()** function, which reveals the hole card by setting the **holeCardPic** source equal to the **file** property of the first card in the **dealerHand** array. The reveal occurs after a 1 second delay:

```
    function showHoleCard() {
        setTimeout(() => {
            holeCardPic.src = `images/cards350px/${dealerHand[0].file}`;
        }, 1000);
    }
```

25. Run the app again. Both the player and dealer are still dealt blackjack, but this time the hole card is revealed after a 1 second delay.

26. Now to test dealer-only Blackjack, which still must reveal the hole card, but which results in a different message being outputted.

- hard-code the first dealer card object (hole card) to be an Ace
- hard-code the second dealer card object (visible card) to be a Queen
- make the player's cards whatever rando cards get popped off the shoe

```
    // 21. Pop a card object off the shoe array and save it as a new card
        // let card = shoe.pop();
        let card;
        /* Testing dealer-only blackjack */
        // make dealer's hole card an Ace
        if(dealCounter == 2) {
            card = {
                file: "Ace-of-Hearts.png",
                kind: "Ace",
                name: "Ace of Hearts",
                suit: "Hearts",
                valu: 11
            };
        // make dealer's second card, the visible card, a Queen
        } else if(dealCounter == 4) {
           card = {
                file: "Queen-of-Diamonds.png",
                kind: "Queen",
                name: "Queen of Diamonds",
                suit: "Diamonds",
                valu: 10
            };
        // make player's cards whatever gets popped off the shoe
        } else {
            card = shoe.pop();
        }
```

27. Run the app again. The dealer should get Blackjack, with the correct message outputted and the hole-card revealed to prove that the dealer does indeed have Blackjack.

28. Now to test player Blackjack, which does not necessitate revealing the hole card.

- hard-code the first player card to be an Ace
- hard-code the second player card to be a Queen
- make the dealer's cards whatever gets popped off the shoe To do all this, simply change the the if-statement counter values from 2 and 4 to 1 and 3:

```
        if(dealCounter == 1) {
            -- ETC. --
        if(dealCounter == 3) {
            -- ETC. --
```

29. Test the app to make sure the player is getting blackjack and that the Ace scores are logging as expected.

30. Comment out all Blackjack testing code, and restore the **let card = shoe.pop()** line, so that the dealing goes back to normal.

END: Lesson 08.04
NEXT: Lesson 08.05