**New Super Blackjack**

Leah Judd, Connor Perry, Andrew Stevens

**Abstract**

This paper goes over the functionality, development process, and inner workings of our group’s Mario-inspired blackjack game. We took various assets and ideas and slowly turned them into something that we felt would be a proper representation of the knowledge we gained from this semester.

**Introduction**

For our project, we decided to create a small game based on blackjack. We recreated different assets from an existing casino-based game in New Super Mario Bros. for the Nintendo DS and repurposed them to use for blackjack. Once we had the assets prepared, we began work on the basic code for the website. The CSS utilized these assets so that the game space looks charming and user-friendly. The JavaScript was first made as a standalone file which ran on its own, but was then later modified to work with the HTML file and grant functionality to the web page.

**The Game**

When the website is loaded, the player is met with a blank screen with all of the controls and information that they will need to play the game. They start with 50 coins, as shown in the top left corner.



Upon clicking the triangular “Bet!” button in the top left corner, the player spends 5 coins on the initial bet. Both the player and the dealer then receive two cards. The goal from this point will be to try to reach a value higher than the dealer’s, but not over 21. Face cards are worth 10 points, but the ace can be either 1 or 11 points depending on which is better for the holder. If someone gets a value of 21 from their first two cards, which is only possible with an ace and a card worth 10 points, it is considered a blackjack, which gives an immediate win and a higher payout.



From here, the player has three options indicated by the three buttons at the bottom of the screen. Hitting will cause the player to receive another card from the deck. Standing simply finalizes the player’s cards, which will allow the dealer to draw theirs. Doubling down is similar to hitting, although it allows the player to double their bet in exchange for not being able to hit again.

In this example, the player’s cards make a total of 11. Since the most likely value one can receive from a hit is 10, doubling down will likely result in a value at or near 21.



The player drew a face card, which is worth 10 points. That brings the player’s hand to the perfect value of 21. After the player is done, the dealer must keep hitting until they reach a minimum of 17. In this case, they went over 21, meaning the dealer has busted. The player wins in this scenario, and makes a net profit of 10 coins instead of the 5 that they would’ve gotten if they hit instead of doubling down.

If the player wants to play again, they can hit the “Bet!” button again, which will clear the game space and start a new game. If the player reaches a value of 0, they will not be able to double down or place another bet. Their only option would be to use the reset button to reset their coins to 50.

**The HTML File**

The HTML file simply sets up the website using values from the CSS and JavaScript files, both of which are linked in the head element. The body sets up several different buttons and dividers, each modified by an ID from the CSS file. The buttons related to the gameplay are set to trigger their respective functions in the event that they are clicked. The dividers are left blank by the HTML document, but will come back during the JavaScript portion of the project.

**The CSS File**

The CSS file houses several IDs to be used by the elements of the HTML file. This file links to and uses many of the assets in the asset folder, as well as the themed font that we chose to display the number of coins in. It also uses relative positioning via percentages to make the website usable at different resolutions.

**The JavaScript File**

The JavaScript file is the most intricate of the files, as the logic of the game and its relationships with the other files require many operations and variables that relate to each other. It begins by defining a few global variables that will be largely used to manage how the game functions.

The currency is managed by two variables. Firstly, there’s of course the number of coins, which is set to 50. Then there is the number of coins being betted, which defaults to 0 and will change depending on the game. There are two booleans: one managing whether the player can hit, and one managing whether the player can place a new bet. They are set to false and true respectively, as a player can’t hit until they place a bet first. Two empty arrays are used to represent the hands of the player and the dealer. Similarly, two integers defaulting to 0 are used to represent the value of said hands. Finally, an empty deck is created which will soon hold the cards for the game.

When the website loads, a function is triggered which updates the HTML ID for the coin variable. This function will be used repeatedly to update the value being displayed in the top left corner of the screen. Another function is defined which will clear a given divider of any children, which will be used for when the game needs to be reset.

Next in the code is the constructor function for the cards. Each card has its own rank, suit, and string ID, the latter of which is created using the first 2 in order to create unique names for each card.

Two functions are used to create the deck used during the game. One creates the deck, while the other shuffles it. The former of the two creates two arrays, one filled with every rank, and one filled with every suit. A nested for loop is used to create a new object for each combination of ranks and suits and to push them into the deck array. The shuffle function uses another for loop to swap each index with a random index.

Another function is used to calculate the value of a hand by adding up numeric ranks and interpreting the other cards. Ace is a special case, since it can be either 1 or 11. The function will first assume that an ace has a value of 11, but if the hand exceeds 21 and has an ace, the ace will instead be counted as 1.

The next several functions give functionality to the buttons on the HTML page. Whether or not the buttons perform their functions depends on the conditions under which one would normally be able to use them. For example, the hit button can’t be used unless a bet is already started. The most important part to note is that anything that draws a new card does so by using the string ID as an ID for an HTML element. The program will find the filename using the string ID and append it to the appropriate divider.

The bet function essentially resets the board from any previous rounds, makes and shuffles a deck, sets up each pair of first cards, and reverses the booleans from earlier so that a new bet can’t be made and the player can hit.

Once a game-ending condition is detected, it will go to the function that ends that game by letting the dealer hit as much as needed, calculating the winner, and displaying the appropriate image depending on the outcome.

The final function causes the reset button to reload the page, resetting everything.

**Conclusion**

This project resulted in something far better than we initially aimed for. Our plans were originally aimed toward a calculator that tells the user the best move to make in an inputted scenario, but it became so much more as we figured out how to implement all sorts of different things that we learned from the semester into the website and slowly discovered how we wanted to handle the project as a whole.