```
// Data set of cards created from scratch, with icons for every card front and back from
Wikimedia Commons:
//  https://commons.wikimedia.org/wiki/Category:SVG_English_pattern_playing_cards
//  https://commons.wikimedia.org/wiki/Category:SVG_playing_card_backs

// titleScreen image from Oneida Casino, found on Google Images:
//  https://oneidacasino.net/wp-content/uploads/2020/03/Blackjack.png

// playScreen background image from Adobe Stock:
//  https://stock.adobe.com/search/images?k=blackjack+table

// Some code brainstormed and worked on with assistance from collaborative partner

// Create lists for card names, values, and icons
var cardNamesList = getColumn("Cards List", "Name");
var cardValuesList = getColumn("Cards List", "Number");
var cardIconsList = getColumn("Cards List", "Icon");

// Create filtered lists for the names, values, and icons of all cards dealt to the player
var playerCardNamesList = [];
var playerCardValuesList = [];
var playerCardIconsList = [];

// Create filtered lists for the names, values, and icons of all cards dealt to the computer
var cpuCardNamesList = [];
var cpuCardValuesList = [];
var cpuCardIconsList = [];

// Create variables for hand totals and win totals for the player and CPU
var playerWins = 0;
var cpuWins = 0;
var playerHandValue = 0;
var cpuHandValue = 0;
var hitCounter = 0;

// When back button is clicked, reset filtered lists and go back to title screen
onEvent("playScreenBackButton", "click", function( ) {
  resetGame();
  setScreen("titleScreen");
});

// Back button for the How to Play screen, takes user back to title
onEvent("howToPlayScreenBackButton", "click", function( ) {
  setScreen("titleScreen");
```

```
});

// Start button on title screen takes players to the main game screen
onEvent("startButton", "click", function( ) {
  setScreen("playScreen");
});

// How to Play button on title screen takes players to the screen of the same name
onEvent("howToPlayButton", "click", function( ) {
  setScreen("howToPlayScreen");
});

// When dealButton is clicked, the game starts and the initial cards are dealt,
// followed by giving the player the option to hit or stand; if the player immediately
// gets blackjack then it skips to the CPU's turn and checks for who won or if there is a draw.
onEvent("dealButton", "click", function( ) {
  hideElement("dealButton");
  showElement("playerWinsLabel");
  showElement("cpuWinsLabel");
  getCard("player");
  setImageURL("playerFirstCard", playerCardIconsList[0]);
  showElement("playerFirstCard");
  showElement("cpuFirstCard");
  showElement("playerHandValueLabel");
  getCard("player");
  setImageURL("playerSecondCard", playerCardIconsList[1]);
  showElement("playerSecondCard");
  getCard("cpu");
  setImageURL("cpuSecondCard", cpuCardIconsList[0]);
  showElement("cpuSecondCard");
  showElement("cpuHandValueLabel");
  if (playerHandValue == 21) {
    cpuExtraTurns();
  } else if (playerHandValue < 21) {
    showElement("standButton");
    showElement("hitButton");
  }
});

// Gives the player another card then checks whether they bust or if they can choose again.
// When they can't choose anymore, it starts the CPU's turn and checks for the winner.
onEvent("hitButton", "click", function( ) {
  if (hitCounter == 1) {
    hideElement("standButton");
```

```
    hideElement("hitButton");
    getCard("player");
    setImageURL("playerFourthCard", playerCardIconsList[3]);
    showElement("playerFourthCard");
    if (playerHandValue > 21) {
      findWinner();
    } else if (playerHandValue <= 21) {
      cpuExtraTurns();
    }
  } else if (hitCounter == 0) {
    playerFirstHit();
  }
});

// The player ends their turn and the CPU's turn begins, then the game checks for a winner.
onEvent("standButton", "click", function( ) {
  hideElement("standButton");
  hideElement("hitButton");
  cpuExtraTurns();
});

// The reset button appears after a winner is decided, and clicking it resets the game so the next
can be played.
onEvent("playAgainButton", "click", function() {
  resetGame();
});

// Function to reset all the filtered card lists, the hand values, and the hit counter whenever the
game is reset.
function resetGameVariables() {
  playerCardNamesList = [];
  playerCardValuesList = [];
  playerCardIconsList = [];
  cpuCardNamesList = [];
  cpuCardValuesList = [];
  cpuCardIconsList = [];
  playerHandValue = 0;
  cpuHandValue = 0;
  hitCounter = 0;
}

// Function to reset the visual elements of the game except for the win counters whenever the
game is reset.
function resetGame() {
```

```
  showElement("dealButton");
  setImageURL("cpuFirstCard", "playing-card-back.png");
  hideElement("playerHandValueLabel");
  hideElement("cpuHandValueLabel");
  hideElement("cpuFirstCard");
  hideElement("cpuSecondCard");
  hideElement("cpuThirdCard");
  hideElement("cpuFourthCard");
  hideElement("playerFirstCard");
  hideElement("playerSecondCard");
  hideElement("playerThirdCard");
  hideElement("playerFourthCard");
  hideElement("winStatusLabel");
  hideElement("playAgainButton");
  resetGameVariables();
}

// Function to deal a new card for either the player or the CPU. It adds the new card
// to the filtered lists for that given side, and checks if the card and/or hand
// will be impacted by the value of the ace if there is one in either hand.
//  - playerOrCpu {string} - either "player" or "cpu"
function getCard(playerOrCpu) {
  var newCardNumber = randomNumber(0, cardValuesList.length-1);
  if (playerOrCpu == "player") {
  appendItem(playerCardNamesList, cardNamesList[newCardNumber]);
  appendItem(playerCardValuesList, cardValuesList[newCardNumber]);
  appendItem(playerCardIconsList, cardIconsList[newCardNumber]);
  checkAce(cardNamesList[newCardNumber], playerCardValuesList);
    } else if (playerOrCpu == "cpu") {
  appendItem(cpuCardNamesList, cardNamesList[newCardNumber]);
  appendItem(cpuCardValuesList, cardValuesList[newCardNumber]);
  appendItem(cpuCardIconsList, cardIconsList[newCardNumber]);
  checkAce(cardNamesList[newCardNumber], cpuCardValuesList);
    }
}

// Function traversing the player's hand and adding the value of every card
// to be displayed in the player's hand value label.
function updatePlayerValue() {
  playerHandValue = 0;
  for (var i = 0; i < playerCardValuesList.length; i++) {
    playerHandValue = playerHandValue + playerCardValuesList[i];
    setText("playerHandValueLabel", "Your Hand: " + playerHandValue);
  }
```

```
}

// Function traversing the CPU's hand and adding the value of every card
// to be displayed in the CPU's hand value label.
function updateCpuValue() {
  cpuHandValue = 0;
  for (var i = 0; i < cpuCardValuesList.length; i++) {
    cpuHandValue = cpuHandValue + cpuCardValuesList[i];
    setText("cpuHandValueLabel", "CPU's Hand: " + cpuHandValue);
  }
}

// Function for the CPU's next turn and potential later turns. Reveals the CPU's second card
// and plays out the hand as if it were a dealer, then checks to see who won or if there is a draw.
function cpuExtraTurns() {
  getCard("cpu");
  setImageURL("cpuFirstCard", cpuCardIconsList[1]);
  showElement("cpuFirstCard");
  if (cpuHandValue < 17) {
    getCard("cpu");
    setImageURL("cpuThirdCard", cpuCardIconsList[2]);
    showElement("cpuThirdCard");
    if (cpuHandValue < 17) {
      getCard("cpu");
      setImageURL("cpuFourthCard", cpuCardIconsList[3]);
      showElement("cpuFourthCard");
      findWinner();
    } else if (cpuHandValue >= 17) {
      findWinner();
    }
  } else if (cpuHandValue >= 17) {
    findWinner();
  }
}

// Function for the player's first hit if they choose to hit. Either the player gets another turn
// again, it goes straight to the CPU's turn, or straight to checking for winner depending on what
card comes out.
function playerFirstHit() {
  hideElement("standButton");
  hideElement("hitButton");
  hitCounter++;
  getCard("player");
  setImageURL("playerThirdCard", playerCardIconsList[2]);
```

```
  showElement("playerThirdCard");
  if (playerHandValue < 21) {
    showElement("hitButton");
    showElement("standButton");
  } else if (playerHandValue == 21) {
    cpuExtraTurns();
  } else if (playerHandValue > 21) {
    findWinner();
  }
}

// Function to check who wins the game by comparing the hand values,
// then shows a win/loss/draw message and reveals a button to play again.
function findWinner() {
  var winner;
  if (cpuHandValue > 21) {
    winner = "Player";
  } else if ((playerHandValue > 21)) {
    winner = "CPU";
  } else if (playerHandValue == cpuHandValue) {
    winner = "Draw";
  } else if (playerHandValue > cpuHandValue) {
    winner = "Player";
  } else if (cpuHandValue > playerHandValue) {
    winner = "CPU";
  }
  if (winner == "Draw") {
    setText("winStatusLabel", "Draw!");
  } else if (winner == "Player") {
    setText("winStatusLabel", "You win!");
    playerWins++;
  } else if (winner == "CPU") {
    setText("winStatusLabel", "CPU Wins!");
    cpuWins++;
  }
  setText("playerWinsLabel", "Your Wins: " + playerWins);
  setText("cpuWinsLabel", "CPU Wins: " + cpuWins);
  showElement("winStatusLabel");
  showElement("playAgainButton");
}

// Function to check whether there is an ace in play, and if so whether the value
// should be 1 or 11 based on how that value changes the total hand value for either the player
or CPU.
```

```
function checkAce (cardName, playerOrCpuCardValuesList){
  if (cardName == "Ace") {
    removeItem(playerOrCpuCardValuesList, playerOrCpuCardValuesList.length-1);
    appendItem(playerOrCpuCardValuesList, 11);
    updatePlayerValue();
    updateCpuValue();
    if (playerHandValue > 21) {
      removeItem(playerCardValuesList, playerCardValuesList.length-1);
      appendItem(playerCardValuesList, 1);
      updatePlayerValue();
    } else if ((cpuHandValue > 21)) {
      removeItem(cpuCardValuesList, cpuCardValuesList.length-1);
      appendItem(cpuCardValuesList, 1);
      updateCpuValue();
    }
  } else if (cardName != "Ace") {
    updatePlayerValue();
    updateCpuValue();
    if (playerHandValue > 21) {
      for (var i = 0; i < playerCardNamesList.length; i++) {
        if (playerCardNamesList[i] == "Ace") {
          removeItem(playerCardValuesList, i);
          appendItem(playerCardValuesList, 1);
          updatePlayerValue();
        } else if (playerCardNamesList[i] != "Ace") {
          updatePlayerValue();
        }
      }
    } else if ((cpuHandValue > 21)) {
      for (var ii = 0; ii < cpuCardNamesList.length; ii++) {
        if (cpuCardNamesList[ii] == "Ace") {
          removeItem(cpuCardValuesList, ii);
          appendItem(cpuCardValuesList, 1);
          updateCpuValue();
        } else if (cpuCardNamesList[ii] != "Ace") {
          updateCpuValue();
        }
      }
    }
  }
}
```