­Js08\_txt.js

Slide8

let pokerGame = {

   currentBank: null,

   currentBet: null

};

Js08\_txt.js

Slide10

window.addEventListener("load", playDrawPoker);

function playDrawPoker() {

   // Reference buttons and images within the Poker Game page

   let dealButton = document.getElementById("dealB");

   let drawButton = document.getElementById("drawB");

   let standButton = document.getElementById("standB");

   let resetButton = document.getElementById("resetB");

   let statusBox = document.getElementById("status");

   let betSelection = document.getElementById("bet");

   let bankBox = document.getElementById("bank");

   let cardImages = document.querySelectorAll("img.cardImg");

   //Set the initial bank and bet values

   pokerGame.currentBank = 500;

   pokerGame.currentBet = 25;

   // Display the current bank value

   bankBox.value = pokerGame.currentBank;

   // Change the bet when the selection changes

   betSelection.onchange = function(){

      pokerGame.currentBet = parseInt(this.value);

   }

      dealButton.addEventListener("click", function() {

      if (pokerGame.currentBank >= pokerGame.currentBet) {

         // Enable the Draw and Stand buttons after the initial deal

         dealButton.disabled = true;        // Turn off the Deal button

         betSelection.disabled = true;      // Turn off the Bet Selection list

         drawButton.disabled = false;       // Turn on the Draw button

         standButton.disabled = false;      // Turn on the Stand Button

         statusBox.textContent = "";        // Erase any status messages

   });

   drawButton.addEventListener("click", function() {

      // Enable the Deal and Bet options when the player chooses to draw new cards

      dealButton.disabled = false;        // Turn on the Deal button

      betSelection.disabled = false;      // Turn on the Bet Selection list

      drawButton.disabled = true;         // Turn off the Draw button

      standButton.disabled = true;        // Turn off the Stand Button

   });

   standButton.addEventListener("click", function() {

      // Enable the Deal and Bet options when the player chooses to stand with their hand

      dealButton.disabled = false;        // Turn on the Deal button

      betSelection.disabled = false;      // Turn on the Bet Selection list

      drawButton.disabled = true;         // Turn off the Draw button

      standButton.disabled = true;        // Turn off the Stand Button

   });

   // Reload the current page when the Reset button is clicked

   resetButton.addEventListener("click", function() {

      location.reload();

   });

}

Js08\_txt.js

Slide12

// object defining the poker game

let pokerGame = {

   currentBank: null,

   currentBet: null,

   placeBet: function(){

      this.currentBank -= this.currentBet;

      return this.currentBank;

   }

};

**OR**

"use strict";

/\*    JavaScript 7th Edition

      Chapter 8

      Chapter case

      Draw Poker Game using Object Oriented Programming

      Author:

      Date:

      Filename:       js08.js

 \*/

      let pokerGame = {

        currentBank: null,

         currentBet: null,

         placeBet: function(){

      this.currentBank -= this.currentBet;

      return this.currentBank;

   }

      };

window.addEventListener("load", playDrawPoker);

function playDrawPoker() {

   // Reference buttons and images within the Poker Game page

   let dealButton = document.getElementById("dealB");

   let drawButton = document.getElementById("drawB");

   let standButton = document.getElementById("standB");

   let resetButton = document.getElementById("resetB");

   let statusBox = document.getElementById("status");

   let betSelection = document.getElementById("bet");

   let bankBox = document.getElementById("bank");

   let cardImages = document.querySelectorAll("img.cardImg");

   //Set the initial bank and bet values

   pokerGame.currentBank = 500;

   pokerGame.currentBet = 25;

   // Display the current bank value

   bankBox.value = pokerGame.currentBank;

   // Change the bet when the selection changes

   betSelection.onchange = function(){

      pokerGame.currentBet = parseInt(this.value);

   }

      dealButton.addEventListener("click", function() {

      if (pokerGame.currentBank >= pokerGame.currentBet) {

         // Enable the Draw and Stand buttons after the initial deal

         dealButton.disabled = true;        // Turn off the Deal button

         betSelection.disabled = true;      // Turn off the Bet Selection list

         drawButton.disabled = false;       // Turn on the Draw button

         standButton.disabled = false;      // Turn on the Stand Button

         statusBox.textContent = "";        // Erase any status messages

   });

   drawButton.addEventListener("click", function() {

      // Enable the Deal and Bet options when the player chooses to draw new cards

      dealButton.disabled = false;        // Turn on the Deal button

      betSelection.disabled = false;      // Turn on the Bet Selection list

      drawButton.disabled = true;         // Turn off the Draw button

      standButton.disabled = true;        // Turn off the Stand Button

   });

   standButton.addEventListener("click", function() {

      // Enable the Deal and Bet options when the player chooses to stand with their hand

      dealButton.disabled = false;        // Turn on the Deal button

      betSelection.disabled = false;      // Turn on the Bet Selection list

      drawButton.disabled = true;         // Turn off the Draw button

      standButton.disabled = true;        // Turn off the Stand Button

   });

   // Reload the current page when the Reset button is clicked

   resetButton.addEventListener("click", function() {

      location.reload();

   });

}

Js08\_txt.js

Slide13

window.addEventListener("load", playDrawPoker);

function playDrawPoker() {

   // Reference buttons and images within the Poker Game page

   let dealButton = document.getElementById("dealB");

   let drawButton = document.getElementById("drawB");

   let standButton = document.getElementById("standB");

   let resetButton = document.getElementById("resetB");

   let statusBox = document.getElementById("status");

   let betSelection = document.getElementById("bet");

   let bankBox = document.getElementById("bank");

   let cardImages = document.querySelectorAll("img.cardImg");

   //Set the initial bank and bet values

   pokerGame.currentBank = 500;

   pokerGame.currentBet = 25;

   // Display the current bank value

   bankBox.value = pokerGame.currentBank;

   // Change the bet when the selection changes

   betSelection.onchange = function(){

      pokerGame.currentBet = parseInt(this.value);

   }

   dealButton.addEventListener("click", function(){

      if (pokerGame.currentBank >= pokerGame.currentBet){

         //Enable the Draw and Stand buttons after the initial deal

         dealButton.disabled = true;

         betSelection.disabled = true;

         drawButton.disabled = false;

         standButton.disabled = false;

         statusBox.texContent = "";

         //Reduce the bank by the size of the bet

         bankBox.value = pokerGame.placeBet();

      } else {

         statusBox.textContent = "Insufficient Funds";

      }

   });

   drawButton.addEventListener("click", function() {

      // Enable the Deal and Bet options when the player chooses to draw new cards

      dealButton.disabled = false;        // Turn on the Deal button

      betSelection.disabled = false;      // Turn on the Bet Selection list

      drawButton.disabled = true;         // Turn off the Draw button

      standButton.disabled = true;        // Turn off the Stand Button

   });

   standButton.addEventListener("click", function() {

      // Enable the Deal and Bet options when the player chooses to stand with their hand

      dealButton.disabled = false;        // Turn on the Deal button

      betSelection.disabled = false;      // Turn on the Bet Selection list

      drawButton.disabled = true;         // Turn off the Draw button

      standButton.disabled = true;        // Turn off the Stand Button

   });

   // Reload the current page when the Reset button is clicked

   resetButton.addEventListener("click", function() {

      location.reload();

   });

}

Js08\_txt.js

Slide 16

// object defining the poker game

let pokerGame = {

   currentBank: null,

   currentBet: null,

   placeBet: function(){

      this.currentBank -= this.currentBet;

      return this.currentBank;

   }

};

/\* Constructor function for poker cards \*/

function pokerCard(cardSuit, cardRank){

   this.suit = cardSuit;

   this.rank = cardRank;

}

**OR**

"use strict";

/\*    JavaScript 7th Edition

      Chapter 8

      Chapter case

      Draw Poker Game using Object Oriented Programming

      Author:

      Date:

      Filename:       js08.js

 \*/

      let pokerGame = {

        currentBank: null,

         currentBet: null,

         placeBet: function(){

      this.currentBank -= this.currentBet;

      return this.currentBank;

   }

      };

/\* Constructor function for poker cards \*/

function pokerCard(cardSuit, cardRank){

   this.suit = cardSuit;

   this.rank = cardRank;

}

window.addEventListener("load", playDrawPoker);

function playDrawPoker() {

   // Reference buttons and images within the Poker Game page

   let dealButton = document.getElementById("dealB");

   let drawButton = document.getElementById("drawB");

   let standButton = document.getElementById("standB");

   let resetButton = document.getElementById("resetB");

   let statusBox = document.getElementById("status");

   let betSelection = document.getElementById("bet");

   let bankBox = document.getElementById("bank");

   let cardImages = document.querySelectorAll("img.cardImg");

   //Set the initial bank and bet values

   pokerGame.currentBank = 500;

   pokerGame.currentBet = 25;

   // Display the current bank value

   bankBox.value = pokerGame.currentBank;

   // Change the bet when the selection changes

   betSelection.onchange = function(){

      pokerGame.currentBet = parseInt(this.value);

   }

      dealButton.addEventListener("click", function() {

      if (pokerGame.currentBank >= pokerGame.currentBet) {

        // Enable the Draw and Stand buttons after the initial deal

        dealButton.disabled = true; // Turn off the Deal button

        betSelection.disabled = true; // Turn off the Bet Selection list

        drawButton.disabled = false; // Turn on the Draw button

        standButton.disabled = false; // Turn on the Stand Button

        statusBox.textContent = ""; // Erase any status messages

        //Reduce the bank by the size of the bet

        bankBox.value = pokerGame.placeBet();

      } else {

        statusBox.textContent = "Insufficient Funds";

      }

   });

   drawButton.addEventListener("click", function() {

      // Enable the Deal and Bet options when the player chooses to draw new cards

      dealButton.disabled = false;        // Turn on the Deal button

      betSelection.disabled = false;      // Turn on the Bet Selection list

      drawButton.disabled = true;         // Turn off the Draw button

      standButton.disabled = true;        // Turn off the Stand Button

   });

   standButton.addEventListener("click", function() {

      // Enable the Deal and Bet options when the player chooses to stand with their hand

      dealButton.disabled = false;        // Turn on the Deal button

      betSelection.disabled = false;      // Turn on the Bet Selection list

      drawButton.disabled = true;         // Turn off the Draw button

      standButton.disabled = true;        // Turn off the Stand Button

   });

   // Reload the current page when the Reset button is clicked

   resetButton.addEventListener("click", function() {

      location.reload();

   });

}

js08\_txt.js - Slide 18

let pokerGame = {

   currentBank: null,

   currentBet: null,

   placeBet: function(){

      this.currentBank -= this.currentBet;

      return this.currentBank;

   }

};

/\* Constructor function for poker cards \*/

function pokerCard(cardSuit, cardRank){

   this.suit = cardSuit;

   this.rank = cardRank;

}

/\* Constructor function for poker decks\*/

function pokerDeck(){

   //List the suits and ranks

   let suits = ["clubs", "diamonds", "hearts", "spades"];

let ranks = ["2", "3", "4", "5", "6", "7", "8", "9", "10", "jack", "queen", "king", "ace"];

   this.cards = [];

   //Add a card for each combination of suit and rank

   for (let i = 0; i < 4; i++){

      for (let j = 0; j < 13; j++){

         //Add a pokerCard object

         this.cards.push(new pokerCard(suits[i], ranks[j]));

      }

   }

};

js08\_txt.js

Slide 19

let pokerGame = {

   currentBank: null,

   currentBet: null,

   placeBet: function(){

      this.currentBank -= this.currentBet;

      return this.currentBank;

   }

};

/\* Constructor function for poker cards \*/

function pokerCard(cardSuit, cardRank){

   this.suit = cardSuit;

   this.rank = cardRank;

}

/\* Constructor function for poker decks\*/

function pokerDeck(){

   //List the suits and ranks

  let suits = ["clubs", "diamonds", "hearts", "spades"];

 let ranks = ["2", "3", "4", "5", "6", "7", "8", "9", "10", "jack", "queen", "king", "ace"];

   this.cards = [];

   //Add a card for each combination of suit and rank

   for (let i = 0; i < 4; i++){

      for (let j = 0; j < 13; j++){

         //Add a pokerCard object

         this.cards.push(new pokerCard(suits[i], ranks[j]));

      }

   }

   // Method to randomly sort the cards in the deck

   this.shuffle = function(){

      this.cards.sort(function(){

         return 0.5 - Math.random();

      });

   };

};

js08\_txt.js

Slide 20

window.addEventListener("load", playDrawPoker);

function playDrawPoker() {

   // Reference buttons and images within the Poker Game page

   let dealButton = document.getElementById("dealB");

   let drawButton = document.getElementById("drawB");

   let standButton = document.getElementById("standB");

   let resetButton = document.getElementById("resetB");

   let statusBox = document.getElementById("status");

   let betSelection = document.getElementById("bet");

   let bankBox = document.getElementById("bank");

   let cardImages = document.querySelectorAll("img.cardImg");

   //Set the initial bank and bet values

   pokerGame.currentBank = 500;

   pokerGame.currentBet = 25;

   // Create a deck of shuffled cards

   let myDeck = new pokerDeck();

   myDeck.shuffle();

   // Display the current bank value

   bankBox.value = pokerGame.currentBank;

   // Change the bet when the selection changes

   betSelection.onchange = function(){

      pokerGame.currentBet = parseInt(this.value);

   }

   dealButton.addEventListener("click", function(){

      if (pokerGame.currentBank >= pokerGame.currentBet){

         //Enable the Draw and Stand buttons after the initial deal

         dealButton.disabled = true;

         betSelection.disabled = true;

         drawButton.disabled = false;

         standButton.disabled = false;

         statusBox.texContent = "";

         //Reduce the bank by the size of the bet

         bankBox.value = pokerGame.placeBet();

      } else {

         statusBox.textContent = "Insufficient Funds";

      }

   });

      dealButton.addEventListener("click", function() {

      if (pokerGame.currentBank >= pokerGame.currentBet) {

         // Enable the Draw and Stand buttons after the initial deal

         dealButton.disabled = true;        // Turn off the Deal button

         betSelection.disabled = true;      // Turn off the Bet Selection list

         drawButton.disabled = false;       // Turn on the Draw button

         standButton.disabled = false;      // Turn on the Stand Button

         statusBox.textContent = "";        // Erase any status messages

   });

   drawButton.addEventListener("click", function() {

      // Enable the Deal and Bet options when the player chooses to draw new cards

      dealButton.disabled = false;        // Turn on the Deal button

      betSelection.disabled = false;      // Turn on the Bet Selection list

      drawButton.disabled = true;         // Turn off the Draw button

      standButton.disabled = true;        // Turn off the Stand Button

   });

   standButton.addEventListener("click", function() {

      // Enable the Deal and Bet options when the player chooses to stand with their hand

      dealButton.disabled = false;        // Turn on the Deal button

      betSelection.disabled = false;      // Turn on the Bet Selection list

      drawButton.disabled = true;         // Turn off the Draw button

      standButton.disabled = true;        // Turn off the Stand Button

   });

   // Reload the current page when the Reset button is clicked

   resetButton.addEventListener("click", function() {

      location.reload();

   });

}

Js08\_txt.js

Slide 21

let pokerGame = {

   currentBank: null,

   currentBet: null,

   placeBet: function(){

      this.currentBank -= this.currentBet;

      return this.currentBank;

   }

};

/\* Constructor function for poker cards \*/

function pokerCard(cardSuit, cardRank){

   this.suit = cardSuit;

   this.rank = cardRank;

}

/\* Constructor function for poker decks\*/

function pokerDeck(){

   //List the suits and ranks

  let suits = ["clubs", "diamonds", "hearts", "spades"];

 let ranks=["2", "3", "4", "5", "6", "7", "8", "9", "10", "jack", "queen", "king", "ace"];

   this.cards = [];

   //Add a card for each combination of suit and rank

   for (let i = 0; i < 4; i++){

      for (let j = 0; j < 13; j++){

         //Add a pokerCard object

         this.cards.push(new pokerCard(suits[i], ranks[j]));

      }

   }

   // Method to randomly sort the cards in the deck

   this.shuffle = function(){

      this.cards.sort(function(){

         return 0.5 - Math.random();

      });

   };

   /\* Constructor function for poker hands \*/

   function pokerHand(handLength){

      this.cards = new Array(handLength);

   }

//};

js08\_txt.js

Slide 22

window.addEventListener("load", playDrawPoker);

function playDrawPoker() {

   // Reference buttons and images within the Poker Game page

   let dealButton = document.getElementById("dealB");

   let drawButton = document.getElementById("drawB");

   let standButton = document.getElementById("standB");

   let resetButton = document.getElementById("resetB");

   let statusBox = document.getElementById("status");

   let betSelection = document.getElementById("bet");

   let bankBox = document.getElementById("bank");

   let cardImages = document.querySelectorAll("img.cardImg");

   //Set the initial bank and bet values

   pokerGame.currentBank = 500;

   pokerGame.currentBet = 25;

   // Create a deck of shuffled cards

   let myDeck = new pokerDeck();

   myDeck.shuffle();

   // Create an empty poker hand object

   let myHand = new pokerHand(5);

   // Display the current bank value

   bankBox.value = pokerGame.currentBank;

   // Change the bet when the selection changes

   betSelection.onchange = function(){

      pokerGame.currentBet = parseInt(this.value);

   }

   dealButton.addEventListener("click", function(){

      if (pokerGame.currentBank >= pokerGame.currentBet){

         //Enable the Draw and Stand buttons after the initial deal

         dealButton.disabled = true;

         betSelection.disabled = true;

         drawButton.disabled = false;

         standButton.disabled = false;

         statusBox.texContent = "";

         //Reduce the bank by the size of the bet

         bankBox.value = pokerGame.placeBet();

      } else {

         statusBox.textContent = "Insufficient Funds";

      }

   });

      dealButton.addEventListener("click", function() {

      if (pokerGame.currentBank >= pokerGame.currentBet) {

         // Enable the Draw and Stand buttons after the initial deal

         dealButton.disabled = true;        // Turn off the Deal button

         betSelection.disabled = true;      // Turn off the Bet Selection list

         drawButton.disabled = false;       // Turn on the Draw button

         standButton.disabled = false;      // Turn on the Stand Button

         statusBox.textContent = "";        // Erase any status messages

   });

   drawButton.addEventListener("click", function() {

      // Enable the Deal and Bet options when the player chooses to draw new cards

      dealButton.disabled = false;        // Turn on the Deal button

      betSelection.disabled = false;      // Turn on the Bet Selection list

      drawButton.disabled = true;         // Turn off the Draw button

      standButton.disabled = true;        // Turn off the Stand Button

   });

   standButton.addEventListener("click", function() {

      // Enable the Deal and Bet options when the player chooses to stand with their hand

      dealButton.disabled = false;        // Turn on the Deal button

      betSelection.disabled = false;      // Turn on the Bet Selection list

      drawButton.disabled = true;         // Turn off the Draw button

      standButton.disabled = true;        // Turn off the Stand Button

   });

   // Reload the current page when the Reset button is clicked

   resetButton.addEventListener("click", function() {

      location.reload();

   });

}

js08\_txt.js

Slide 23

let pokerGame = {

   currentBank: null,

   currentBet: null,

   placeBet: function(){

      this.currentBank -= this.currentBet;

      return this.currentBank;

   }

};

/\* Constructor function for poker cards \*/

function pokerCard(cardSuit, cardRank){

   this.suit = cardSuit;

   this.rank = cardRank;

}

/\* Constructor function for poker decks\*/

function pokerDeck(){

   //List the suits and ranks

  let suits = ["clubs", "diamonds", "hearts", "spades"];

 let ranks=["2", "3", "4", "5", "6", "7", "8", "9", "10", "jack", "queen", "king", "ace"];

   this.cards = [];

   //Add a card for each combination of suit and rank

   for (let i = 0; i < 4; i++){

      for (let j = 0; j < 13; j++){

         //Add a pokerCard object

         this.cards.push(new pokerCard(suits[i], ranks[j]));

      }

   }

   // Method to randomly sort the cards in the deck

   this.shuffle = function(){

      this.cards.sort(function(){

         return 0.5 - Math.random();

      });

   };

   // Method to deal cards from the deck into a hand

   this.dealTo = function(pokerHand){

      let cardsDealt = pokerHand.cards.length;

      pokerHand.cards = this.cards.splice(0, cardsDealt);

   };

};

   /\* Constructor function for poker hands \*/

   function pokerHand(handLength){

      this.cards = new Array(handLength);

   }

//

Js08\_txt.js

Slide 24

window.addEventListener("load", playDrawPoker);

function playDrawPoker() {

   // Reference buttons and images within the Poker Game page

   let dealButton = document.getElementById("dealB");

   let drawButton = document.getElementById("drawB");

   let standButton = document.getElementById("standB");

   let resetButton = document.getElementById("resetB");

   let statusBox = document.getElementById("status");

   let betSelection = document.getElementById("bet");

   let bankBox = document.getElementById("bank");

   let cardImages = document.querySelectorAll("img.cardImg");

   //Set the initial bank and bet values

   pokerGame.currentBank = 500;

   pokerGame.currentBet = 25;

   // Create a deck of shuffled cards

   let myDeck = new pokerDeck();

   myDeck.shuffle();

   // Create an empty poker hand object

   let myHand = new pokerHand(5);

   // Display the current bank value

   bankBox.value = pokerGame.currentBank;

   // Change the bet when the selection changes

   betSelection.onchange = function(){

      pokerGame.currentBet = parseInt(this.value);

   }

   /\*dealButton.addEventListener("click", function(){

      if (pokerGame.currentBank >= pokerGame.currentBet){

         //Enable the Draw and Stand buttons after the initial deal

         dealButton.disabled = true;

         betSelection.disabled = true;

         drawButton.disabled = false;

         standButton.disabled = false;

         statusBox.texContent = "";

         //Reduce the bank by the size of the bet

         bankBox.value = pokerGame.placeBet();

      } else {

         statusBox.textContent = "Insufficient Funds";

      }

   });

    \*/

      dealButton.addEventListener("click", function() {

      if (pokerGame.currentBank >= pokerGame.currentBet) {

         // Enable the Draw and Stand buttons after the initial deal

         dealButton.disabled = true;        // Turn off the Deal button

         betSelection.disabled = true;      // Turn off the Bet Selection list

         drawButton.disabled = false;       // Turn on the Draw button

         standButton.disabled = false;      // Turn on the Stand Button

         statusBox.textContent = "";        // Erase any status messages

         // Reduce the bank by the size of the bet

        bankBox.value = pokerGame.placeBet();

        // Get a new deck is there are less than 10 cards left

        if (myDeck.cards.length < 10){

         myDeck = new pokerDeck();

         myDeck.shuffle();

        }

        // Deal 5 cards from the deck to the hand

        myDeck.dealTo(myHand);

        console.log(myDeck, myHand);

}

else {

      statusBox.textContent = "Insufficient Funds";

    }

   });

   drawButton.addEventListener("click", function() {

      // Enable the Deal and Bet options when the player chooses to draw new cards

      dealButton.disabled = false;        // Turn on the Deal button

      betSelection.disabled = false;      // Turn on the Bet Selection list

      drawButton.disabled = true;         // Turn off the Draw button

      standButton.disabled = true;        // Turn off the Stand Button

   });

   standButton.addEventListener("click", function() {

      // Enable the Deal and Bet options when the player chooses to stand with their hand

      dealButton.disabled = false;        // Turn on the Deal button

      betSelection.disabled = false;      // Turn on the Bet Selection list

      drawButton.disabled = true;         // Turn off the Draw button

      standButton.disabled = true;        // Turn off the Stand Button

   });

   // Reload the current page when the Reset button is clicked

   resetButton.addEventListener("click", function() {

      location.reload();

   });

}

Js08\_txt.js

Slide 26

let pokerGame = {

   currentBank: null,

   currentBet: null,

   placeBet: function(){

      this.currentBank -= this.currentBet;

      return this.currentBank;

   }

};

/\* Constructor function for poker cards \*/

function pokerCard(cardSuit, cardRank) {

   this.suit = cardSuit;

   this.rank = cardRank;

}

/\* Method to reference the image source file for a card \*/

pokerCard.prototype.cardImage = function() {

  return this.rank + "\_" + this.suit + ".png";

};

/\* Constructor function for poker decks\*/

function pokerDeck(){

   //List the suits and ranks

  let suits = ["clubs", "diamonds", "hearts", "spades"];

   let ranks=["2", "3", "4", "5", "6", "7", "8", "9", "10", "jack", "queen", "king", "ace"];

   this.cards = [];

   //Add a card for each combination of suit and rank

   for (let i = 0; i < 4; i++){

      for (let j = 0; j < 13; j++){

         //Add a pokerCard object

         this.cards.push(new pokerCard(suits[i], ranks[j]));

      }

   }

   // Method to randomly sort the cards in the deck

   this.shuffle = function(){

      this.cards.sort(function(){

         return 0.5 - Math.random();

      });

   };

   // Method to deal cards from the deck into a hand

   this.dealTo = function(pokerHand){

      let cardsDealt = pokerHand.cards.length;

      pokerHand.cards = this.cards.splice(0, cardsDealt);

   };

};

   /\* Constructor function for poker hands \*/

   function pokerHand(handLength){

      this.cards = new Array(handLength);

   }

//

Js08\_txt.js – slide 27

window.addEventListener("load", playDrawPoker);

function playDrawPoker() {

   // Reference buttons and images within the Poker Game page

   let dealButton = document.getElementById("dealB");

   let drawButton = document.getElementById("drawB");

   let standButton = document.getElementById("standB");

   let resetButton = document.getElementById("resetB");

   let statusBox = document.getElementById("status");

   let betSelection = document.getElementById("bet");

   let bankBox = document.getElementById("bank");

   let cardImages = document.querySelectorAll("img.cardImg");

   //Set the initial bank and bet values

   pokerGame.currentBank = 500;

   pokerGame.currentBet = 25;

   // Create a deck of shuffled cards

   let myDeck = new pokerDeck();

   myDeck.shuffle();

   // Create an empty poker hand object

   let myHand = new pokerHand(5);

   // Display the current bank value

   bankBox.value = pokerGame.currentBank;

   // Change the bet when the selection changes

   betSelection.onchange = function(){

      pokerGame.currentBet = parseInt(this.value);

   }

   dealButton.addEventListener("click", function(){

      if (pokerGame.currentBank >= pokerGame.currentBet){

         //Enable the Draw and Stand buttons after the initial deal

         dealButton.disabled = true;

         betSelection.disabled = true;

         drawButton.disabled = false;

         standButton.disabled = false;

         statusBox.texContent = "";

         //Reduce the bank by the size of the bet

         bankBox.value = pokerGame.placeBet();

         // Get a new deck is there are less than 10 cards left

        if (myDeck.cards.length < 10){

         myDeck = new pokerDeck();

         myDeck.shuffle();

        }

        // Deal 5 cards from the deck to the hand

        myDeck.dealTo(myHand);

         // Display the card images on the table

         for (let i = 0; i < cardImages.length; i++) {

            cardImages[i].src = myHand.cards[i].cardImage();

        }

        //console.log(myDeck, myHand);

      } else {

         statusBox.textContent = "Insufficient Funds";

      }

   });

   drawButton.addEventListener("click", function() {

      // Enable the Deal and Bet options when the player chooses to draw new cards

      dealButton.disabled = false;        // Turn on the Deal button

      betSelection.disabled = false;      // Turn on the Bet Selection list

      drawButton.disabled = true;         // Turn off the Draw button

      standButton.disabled = true;        // Turn off the Stand Button

   });

   standButton.addEventListener("click", function() {

      // Enable the Deal and Bet options when the player chooses to stand with their hand

      dealButton.disabled = false;        // Turn on the Deal button

      betSelection.disabled = false;      // Turn on the Bet Selection list

      drawButton.disabled = true;         // Turn off the Draw button

      standButton.disabled = true;        // Turn off the Stand Button

   });

   // Reload the current page when the Reset button is clicked

   resetButton.addEventListener("click", function() {

      location.reload();

   });

}

Js08\_txt.js – slide 32

let pokerGame = {

   currentBank: null,

   currentBet: null,

   placeBet: function(){

      this.currentBank -= this.currentBet;

      return this.currentBank;

   }

};

/\* Constructor function for poker cards \*/

function pokerCard(cardSuit, cardRank){

   this.suit = cardSuit;

   this.rank = cardRank;

}

/\* Method to reference the image source file for a card \*/

pokerCard.prototype.cardImage = function() {

   return this.rank + "\_" + this.suit + ".png";

 };

/\* Constructor function for poker decks\*/

function pokerDeck(){

   //List the suits and ranks

let suits = ["clubs", "diamonds", "hearts", "spades"];

   let ranks=["2", "3", "4", "5", "6", "7", "8", "9", "10", "jack", "queen", "king", "ace"];

   this.cards = [];

   //Add a card for each combination of suit and rank

   for (let i = 0; i < 4; i++){

      for (let j = 0; j < 13; j++){

         //Add a pokerCard object

         this.cards.push(new pokerCard(suits[i], ranks[j]));

      }

   }

   // Method to randomly sort the cards in the deck

   this.shuffle = function(){

      this.cards.sort(function(){

         return 0.5 - Math.random();

      });

   };

   // Method to deal cards from the deck into a hand

   this.dealTo = function(pokerHand){

      let cardsDealt = pokerHand.cards.length;

      pokerHand.cards = this.cards.splice(0, cardsDealt);

   };

};

   /\* Constructor function for poker hands \*/

   function pokerHand(handLength){

      this.cards = new Array(handLength);

   }

   // Method to replace a card in a hand with a card from a deck

   pokerHand.prototype.replaceCard = function(index, pokerDeck){

      this.cards[index] = pokerDeck.cards.shift();

   };

Js08\_txt.js – slide 36

window.addEventListener("load", playDrawPoker);

function playDrawPoker() {

   // Reference buttons and images within the Poker Game page

   let dealButton = document.getElementById("dealB");

   let drawButton = document.getElementById("drawB");

   let standButton = document.getElementById("standB");

   let resetButton = document.getElementById("resetB");

   let statusBox = document.getElementById("status");

   let betSelection = document.getElementById("bet");

   let bankBox = document.getElementById("bank");

   let cardImages = document.querySelectorAll("img.cardImg");

   //Set the initial bank and bet values

   pokerGame.currentBank = 500;

   pokerGame.currentBet = 25;

   // Create a deck of shuffled cards

   let myDeck = new pokerDeck();

   myDeck.shuffle();

   // Create an empty poker hand object

   let myHand = new pokerHand(5);

   // Display the current bank value

   bankBox.value = pokerGame.currentBank;

   // Change the bet when the selection changes

   betSelection.onchange = function(){

      pokerGame.currentBet = parseInt(this.value);

   }

   dealButton.addEventListener("click", function(){

      if (pokerGame.currentBank >= pokerGame.currentBet){

         //Enable the Draw and Stand buttons after the initial deal

         dealButton.disabled = true;

         betSelection.disabled = true;

         drawButton.disabled = false;

         standButton.disabled = false;

         statusBox.texContent = "";

         //Reduce the bank by the size of the bet

         bankBox.value = pokerGame.placeBet();

         // Get a new deck is there are less than 10 cards left

        if (myDeck.cards.length < 10){

         myDeck = new pokerDeck();

         myDeck.shuffle();

        }

        // Deal 5 cards from the deck to the hand

        myDeck.dealTo(myHand);

         // Display the card images on the table

         for (let i = 0; i < cardImages.length; i++) {

            cardImages[i].src = myHand.cards[i].cardImage();

        // Flip the card images when clicked

        cardImages[i].onclick = function() {

         if (this.src.includes("cardback.png")){

            this.src = myHand.cards[i].cardImage();

         } else {

            // show the back of the card

            this.src = "cardback.png";

         }

        }

        //console.log(myDeck, myHand);

         }

      }

      else {

         statusBox.textContent = "Insufficient Funds";

      }

   });

      dealButton.addEventListener("click", function() {

      if (pokerGame.currentBank >= pokerGame.currentBet) {

         // Enable the Draw and Stand buttons after the initial deal

         dealButton.disabled = true;        // Turn off the Deal button

         betSelection.disabled = true;      // Turn off the Bet Selection list

         drawButton.disabled = false;       // Turn on the Draw button

         standButton.disabled = false;      // Turn on the Stand Button

         statusBox.textContent = "";        // Erase any status messages

      }

   });

   drawButton.addEventListener("click", function() {

      // Enable the Deal and Bet options when the player chooses to draw new cards

      dealButton.disabled = false;        // Turn on the Deal button

      betSelection.disabled = false;      // Turn on the Bet Selection list

      drawButton.disabled = true;         // Turn off the Draw button

      standButton.disabled = true;        // Turn off the Stand Button

   });

   standButton.addEventListener("click", function() {

      // Enable the Deal and Bet options when the player chooses to stand with their hand

      dealButton.disabled = false;        // Turn on the Deal button

      betSelection.disabled = false;      // Turn on the Bet Selection list

      drawButton.disabled = true;         // Turn off the Draw button

      standButton.disabled = true;        // Turn off the Stand Button

   });

   // Reload the current page when the Reset button is clicked

   resetButton.addEventListener("click", function() {

      location.reload();

   });

}

Js08\_txt.js – slide 37

"use strict";

/\*    JavaScript 7th Edition

      Chapter 8

      Chapter case

      Draw Poker Game using Object Oriented Programming

      Author:

      Date:

      Filename:       js08.js

 \*/

window.addEventListener("load", playDrawPoker);

function playDrawPoker() {

   // Reference buttons and images within the Poker Game page

   let dealButton = document.getElementById("dealB");

   let drawButton = document.getElementById("drawB");

   let standButton = document.getElementById("standB");

   let resetButton = document.getElementById("resetB");

   let statusBox = document.getElementById("status");

   let betSelection = document.getElementById("bet");

   let bankBox = document.getElementById("bank");

   let cardImages = document.querySelectorAll("img.cardImg");

   //Set the initial bank and bet values

   pokerGame.currentBank = 500;

   pokerGame.currentBet = 25;

   // Create a deck of shuffled cards

   let myDeck = new pokerDeck();

   myDeck.shuffle();

   // Create an empty poker hand object

   let myHand = new pokerHand(5);

   // Display the current bank value

   bankBox.value = pokerGame.currentBank;

   // Change the bet when the selection changes

   betSelection.onchange = function(){

      pokerGame.currentBet = parseInt(this.value);

   }

   dealButton.addEventListener("click", function(){

      if (pokerGame.currentBank >= pokerGame.currentBet){

         //Enable the Draw and Stand buttons after the initial deal

         dealButton.disabled = true;

         betSelection.disabled = true;

         drawButton.disabled = false;

         standButton.disabled = false;

         statusBox.texContent = "";

         //Reduce the bank by the size of the bet

         bankBox.value = pokerGame.placeBet();

         // Get a new deck is there are less than 10 cards left

        if (myDeck.cards.length < 10){

         myDeck = new pokerDeck();

         myDeck.shuffle();

        }

        // Deal 5 cards from the deck to the hand

        myDeck.dealTo(myHand);

         // Display the card images on the table

         for (let i = 0; i < cardImages.length; i++) {

            cardImages[i].src = myHand.cards[i].cardImage();

        // Flip the card images when clicked

        cardImages[i].onclick = function() {

         if (this.src.includes("cardback.png")){

            this.src = myHand.cards[i].cardImage();

         } else {

            // show the back of the card

            this.src = "cardback.png";

         }

        }

        //console.log(myDeck, myHand);

         }

      }

      else {

         statusBox.textContent = "Insufficient Funds";

      }

   });

      dealButton.addEventListener("click", function() {

      if (pokerGame.currentBank >= pokerGame.currentBet) {

         // Enable the Draw and Stand buttons after the initial deal

         dealButton.disabled = true;        // Turn off the Deal button

         betSelection.disabled = true;      // Turn off the Bet Selection list

         drawButton.disabled = false;       // Turn on the Draw button

         standButton.disabled = false;      // Turn on the Stand Button

         statusBox.textContent = "";        // Erase any status messages

      }

   });

   drawButton.addEventListener("click", function() {

      // Enable the Deal and Bet options when the player chooses to draw new cards

      dealButton.disabled = false;        // Turn on the Deal button

      betSelection.disabled = false;      // Turn on the Bet Selection list

      drawButton.disabled = true;         // Turn off the Draw button

      standButton.disabled = true;        // Turn off the Stand Button

 // Replace cards marked to be discarded

 for (let i = 0; i < cardImages.length; i++) {

   if (cardImages[i].src.includes("cardback.png")) {

      // Replace the card and its image on the table

      myHand.replaceCard(i, myDeck);

      cardImages[i].src = myHand.cards[i].cardImage();

   }

}

   });

   standButton.addEventListener("click", function() {

      // Enable the Deal and Bet options when the player chooses to stand with their hand

      dealButton.disabled = false;        // Turn on the Deal button

      betSelection.disabled = false;      // Turn on the Bet Selection list

      drawButton.disabled = true;         // Turn off the Draw button

      standButton.disabled = true;        // Turn off the Stand Button

   });

   // Reload the current page when the Reset button is clicked

   resetButton.addEventListener("click", function() {

      location.reload();

   });

}

Js08\_txt.js-slide 39

"use strict";

/\*    JavaScript 7th Edition

      Chapter 8

      Chapter case

      Custom Objects Used in Poker Games

      Author:

      Date:

      Filename:       objects.js

 \*/

      // object defining the poker game

let pokerGame = {

   currentBank: null,

   currentBet: null,

   placeBet: function(){

      this.currentBank -= this.currentBet;

      return this.currentBank;

   }

};

/\* Constructor function for poker cards \*/

function pokerCard(cardSuit, cardRank){

   this.suit = cardSuit;

   this.rank = cardRank;

}

/\* Method to reference the image source file for a card \*/

pokerCard.prototype.cardImage = function() {

   return this.rank + "\_" + this.suit + ".png";

 };

/\* Constructor function for poker decks\*/

function pokerDeck(){

   //List the suits and ranks

let suits = ["clubs", "diamonds", "hearts", "spades"];

   let ranks=["2", "3", "4", "5", "6", "7", "8", "9", "10", "jack", "queen", "king", "ace"];

   this.cards = [];

   //Add a card for each combination of suit and rank

   for (let i = 0; i < 4; i++){

      for (let j = 0; j < 13; j++){

         //Add a pokerCard object

         this.cards.push(new pokerCard(suits[i], ranks[j]));

      }

   }

   // Method to randomly sort the cards in the deck

   this.shuffle = function(){

      this.cards.sort(function(){

         return 0.5 - Math.random();

      });

   };

   // Method to deal cards from the deck into a hand

   this.dealTo = function(pokerHand){

      let cardsDealt = pokerHand.cards.length;

      pokerHand.cards = this.cards.splice(0, cardsDealt);

   };

};

   /\* Constructor function for poker hands \*/

   function pokerHand(handLength){

      this.cards = new Array(handLength);

   }

   // Method to replace a card in a hand with a card from a deck

   pokerHand.prototype.replaceCard = function(index, pokerDeck){

      this.cards[index] = pokerDeck.cards.shift();

   };

// Method to determine the value of the pokerHand

pokerHand.prototype.getHandValue = function() {

   return handType(this);

   /\* ------------------------------------------------+

   | The handType() function returns a text string of |

   | the type of hand held by 5-card poker hand.      |

   +-------------------------------------------------\*/

   function handType(pokerHand) {

      /\* Determine the rank value of each card in the hand

         by creating a property named rankValue         \*/

      for (let i = 0; i < pokerHand.cards.length; i++) {

         if (pokerHand.cards[i].rank === "ace") {

            pokerHand.cards[i].rankValue = 14;

         } else if (pokerHand.cards[i].rank === "king") {

            pokerHand.cards[i].rankValue = 13;

         } else if (pokerHand.cards[i].rank === "queen") {

            pokerHand.cards[i].rankValue = 12;

         } else if (pokerHand.cards[i].rank === "jack") {

            pokerHand.cards[i].rankValue = 11;

         } else {

            pokerHand.cards[i].rankValue = parseInt(pokerHand.cards[i].rank);

         }

      }

      /\* Function to return the highest ranked value in a five-card hand \*/

      function highCard() {

         return Math.max(pokerHand.cards[0].rankValue, pokerHand.cards[1].rankValue,

                         pokerHand.cards[2].rankValue, pokerHand.cards[3].rankValue,

                         pokerHand.cards[4].rankValue);

      }

      /\* Function to test for the presence of a flush in which all

         five cards have the same suit \*/

      function hasFlush() {

         let firstSuit = pokerHand.cards[0].suit;

         return pokerHand.cards.every(function(card) {

            return card.suit === firstSuit;

         });

      };

      /\* Function to test for the presence of a straight in which the

         rank value of the cards can be placed in sequential order \*/

      function hasStraight() {

         pokerHand.cards.sort(function(a, b) {

            return a.rankValue - b.rankValue;

         });

         return pokerHand.cards.every(function(card, i, cards) {

            if (i > 0) {

               return (cards[i].rankValue - cards[i-1].rankValue === 1);

            } else {

               return true;

            }

         });

      };

      /\* Function to test for the presence of a straight flush \*/

      function hasStraightFlush() {

         return hasFlush() && hasStraight();

      };

      /\* Function to test for the presence of a royal flush

         which consists of 10-J-Q-K-A of the same suit \*/

      function hasRoyalFlush() {

         return hasStraightFlush() && highCard() === 14;

      };

      /\* Function to test for the presence of: pairs, two pairs,

         three of a kind, four of a kind, and full houses  \*/

      function hasSets() {

         // handSets creates an associative array of the duplicates in the hand

         let handSets = {};

         pokerHand.cards.forEach(function(card) {

           if (handSets.hasOwnProperty(card.rankValue)) {

             handSets[card.rankValue]++;

           } else {

             handSets[card.rankValue] = 1;

           }

         });

         let sets = "none";

         let pairRank;

         for (let cardRank in handSets){

            if (handSets[cardRank] === 4) {sets = "Four of a Kind";}

            if (handSets[cardRank] === 3) {

               if (sets === "Pair") {sets = "Full House";}

               else {sets = "Three of a Kind";}

            }

            if (handSets[cardRank] === 2) {

               if (sets === "Three of a Kind") {sets = "Full House";}

               else if (sets === "Pair") {sets = "Two Pair";}

               else {sets = "Pair"; pairRank = cardRank;}

            }

         }

         if (sets === "Pair" && pairRank >= 11) {

            sets = "Jacks or Better";

         }

         return sets;

      }

      // Return a text string describing the hand for draw poker //

      if (hasRoyalFlush()) {return "Royal Flush";}

      else if (hasStraightFlush()) {return "Straight Flush";}

      else if (hasFlush()) {return "Flush";}

      else if (hasStraight()) {return "Straight";}

      else {

         let sets = hasSets();

         if (sets === "Pair" || sets === "none") {sets = "No Winner";}

         return sets;

      }

   }

   /\* ------------------------------------------------+

   |             End of the  handType() function      |

   +-------------------------------------------------\*/

}

Js08\_txt.js – slide 40

"use strict";

/\*    JavaScript 7th Edition

      Chapter 8

      Chapter case

      Draw Poker Game using Object Oriented Programming

      Author:

      Date:

      Filename:       js08.js

 \*/

window.addEventListener("load", playDrawPoker);

function playDrawPoker() {

   // Reference buttons and images within the Poker Game page

   let dealButton = document.getElementById("dealB");

   let drawButton = document.getElementById("drawB");

   let standButton = document.getElementById("standB");

   let resetButton = document.getElementById("resetB");

   let statusBox = document.getElementById("status");

   let betSelection = document.getElementById("bet");

   let bankBox = document.getElementById("bank");

   let cardImages = document.querySelectorAll("img.cardImg");

   //Set the initial bank and bet values

   pokerGame.currentBank = 500;

   pokerGame.currentBet = 25;

   // Create a deck of shuffled cards

   let myDeck = new pokerDeck();

   myDeck.shuffle();

   // Create an empty poker hand object

   let myHand = new pokerHand(5);

   // Display the current bank value

   bankBox.value = pokerGame.currentBank;

   // Change the bet when the selection changes

   betSelection.onchange = function(){

      pokerGame.currentBet = parseInt(this.value);

   }

   dealButton.addEventListener("click", function(){

      if (pokerGame.currentBank >= pokerGame.currentBet){

         //Enable the Draw and Stand buttons after the initial deal

         dealButton.disabled = true;

         betSelection.disabled = true;

         drawButton.disabled = false;

         standButton.disabled = false;

         statusBox.texContent = "";

         //Reduce the bank by the size of the bet

         bankBox.value = pokerGame.placeBet();

         // Get a new deck is there are less than 10 cards left

        if (myDeck.cards.length < 10){

         myDeck = new pokerDeck();

         myDeck.shuffle();

        }

        // Deal 5 cards from the deck to the hand

        myDeck.dealTo(myHand);

         // Display the card images on the table

         for (let i = 0; i < cardImages.length; i++) {

            cardImages[i].src = myHand.cards[i].cardImage();

        // Flip the card images when clicked

        cardImages[i].onclick = function() {

         if (this.src.includes("cardback.png")){

            this.src = myHand.cards[i].cardImage();

         } else {

            // show the back of the card

            this.src = "cardback.png";

         }

        }

        //console.log(myDeck, myHand);

         }

      }

      else {

         statusBox.textContent = "Insufficient Funds";

      }

   });

      dealButton.addEventListener("click", function() {

      if (pokerGame.currentBank >= pokerGame.currentBet) {

         // Enable the Draw and Stand buttons after the initial deal

         dealButton.disabled = true;        // Turn off the Deal button

         betSelection.disabled = true;      // Turn off the Bet Selection list

         drawButton.disabled = false;       // Turn on the Draw button

         standButton.disabled = false;      // Turn on the Stand Button

         statusBox.textContent = "";        // Erase any status messages

      }

   });

   drawButton.addEventListener("click", function() {

      // Enable the Deal and Bet options when the player chooses to draw new cards

      dealButton.disabled = false;        // Turn on the Deal button

      betSelection.disabled = false;      // Turn on the Bet Selection list

      drawButton.disabled = true;         // Turn off the Draw button

      standButton.disabled = true;        // Turn off the Stand Button

 // Replace cards marked to be discarded

 for (let i = 0; i < cardImages.length; i++) {

   if (cardImages[i].src.includes("cardback.png")) {

      // Replace the card and its image on the table

      myHand.replaceCard(i, myDeck);

      cardImages[i].src = myHand.cards[i].cardImage();

   }

}

 // Evaluate the hand drawn by user

 statusBox.textContent = myHand.getHandValue();

   });

   standButton.addEventListener("click", function() {

      // Enable the Deal and Bet options when the player chooses to stand with their hand

      dealButton.disabled = false;        // Turn on the Deal button

      betSelection.disabled = false;      // Turn on the Bet Selection list

      drawButton.disabled = true;         // Turn off the Draw button

      standButton.disabled = true;        // Turn off the Stand Button

      // Evaluate the hand drawn by user

      statusBox.textContent = myHand.getHandValue();

   });

   // Reload the current page when the Reset button is clicked

   resetButton.addEventListener("click", function() {

      location.reload();

   });

}

objects\_txt.js – slide 41

"use strict";

/\*    JavaScript 7th Edition

      Chapter 8

      Chapter case

      Custom Objects Used in Poker Games

      Author:

      Date:

      Filename:       objects.js

 \*/

      // object defining the poker game

let pokerGame = {

   currentBank: null,

   currentBet: null,

   placeBet: function(){

      this.currentBank -= this.currentBet;

      return this.currentBank;

   },

   payBet: function(type) {

      let pay = 0;

      switch (type) {

         case "Royal Flush": pay = 250; break;

         case "Straight Flush": pay = 50; break;

         case "Four of a Kind": pay = 25; break;

         case "Full House": pay = 9; break;

         case "Flush": pay = 6; break;

         case "Straight": pay = 4; break;

         case "Three of a Kind": pay = 3; break;

         case "Two Pair": pay = 2; break;

         case "Jacks or Better": pay = 1; break;

      }

      this.currentBank += pay\*this.currentBet;

      return this.currentBank;

   }

};

/\* Constructor function for poker cards \*/

function pokerCard(cardSuit, cardRank){

   this.suit = cardSuit;

   this.rank = cardRank;

}

/\* Method to reference the image source file for a card \*/

pokerCard.prototype.cardImage = function() {

   return this.rank + "\_" + this.suit + ".png";

 };

/\* Constructor function for poker decks\*/

function pokerDeck(){

   //List the suits and ranks

   let suits = ["clubs", "diamonds", "hearts", "spades"];

   let ranks =["2", "3", "4", "5", "6", "7", "8", "9", "10", "jack", "queen", "king", "ace"];

   this.cards = [];

   //Add a card for each combination of suit and rank

   for (let i = 0; i < 4; i++){

      for (let j = 0; j < 13; j++){

         //Add a pokerCard object

         this.cards.push(new pokerCard(suits[i], ranks[j]));

      }

   }

   // Method to randomly sort the cards in the deck

   this.shuffle = function(){

      this.cards.sort(function(){

         return 0.5 - Math.random();

      });

   };

   // Method to deal cards from the deck into a hand

   this.dealTo = function(pokerHand){

      let cardsDealt = pokerHand.cards.length;

      pokerHand.cards = this.cards.splice(0, cardsDealt);

   };

};

   /\* Constructor function for poker hands \*/

   function pokerHand(handLength){

      this.cards = new Array(handLength);

   }

   // Method to replace a card in a hand with a card from a deck

   pokerHand.prototype.replaceCard = function(index, pokerDeck){

      this.cards[index] = pokerDeck.cards.shift();

   };

// Method to determine the value of the pokerHand

pokerHand.prototype.getHandValue = function() {

   return handType(this);

   /\* ------------------------------------------------+

   | The handType() function returns a text string of |

   | the type of hand held by 5-card poker hand.      |

   +-------------------------------------------------\*/

   function handType(pokerHand) {

      /\* Determine the rank value of each card in the hand

         by creating a property named rankValue         \*/

      for (let i = 0; i < pokerHand.cards.length; i++) {

         if (pokerHand.cards[i].rank === "ace") {

            pokerHand.cards[i].rankValue = 14;

         } else if (pokerHand.cards[i].rank === "king") {

            pokerHand.cards[i].rankValue = 13;

         } else if (pokerHand.cards[i].rank === "queen") {

            pokerHand.cards[i].rankValue = 12;

         } else if (pokerHand.cards[i].rank === "jack") {

            pokerHand.cards[i].rankValue = 11;

         } else {

            pokerHand.cards[i].rankValue = parseInt(pokerHand.cards[i].rank);

         }

      }

      /\* Function to return the highest ranked value in a five-card hand \*/

      function highCard() {

         return Math.max(pokerHand.cards[0].rankValue, pokerHand.cards[1].rankValue,

                         pokerHand.cards[2].rankValue, pokerHand.cards[3].rankValue,

                         pokerHand.cards[4].rankValue);

      }

      /\* Function to test for the presence of a flush in which all

         five cards have the same suit \*/

      function hasFlush() {

         let firstSuit = pokerHand.cards[0].suit;

         return pokerHand.cards.every(function(card) {

            return card.suit === firstSuit;

         });

      };

      /\* Function to test for the presence of a straight in which the

         rank value of the cards can be placed in sequential order \*/

      function hasStraight() {

         pokerHand.cards.sort(function(a, b) {

            return a.rankValue - b.rankValue;

         });

         return pokerHand.cards.every(function(card, i, cards) {

            if (i > 0) {

               return (cards[i].rankValue - cards[i-1].rankValue === 1);

            } else {

               return true;

            }

         });

      };

      /\* Function to test for the presence of a straight flush \*/

      function hasStraightFlush() {

         return hasFlush() && hasStraight();

      };

      /\* Function to test for the presence of a royal flush

         which consists of 10-J-Q-K-A of the same suit \*/

      function hasRoyalFlush() {

         return hasStraightFlush() && highCard() === 14;

      };

      /\* Function to test for the presence of: pairs, two pairs,

         three of a kind, four of a kind, and full houses  \*/

      function hasSets() {

         // handSets creates an associative array of the duplicates in the hand

         let handSets = {};

         pokerHand.cards.forEach(function(card) {

           if (handSets.hasOwnProperty(card.rankValue)) {

             handSets[card.rankValue]++;

           } else {

             handSets[card.rankValue] = 1;

           }

         });

         let sets = "none";

         let pairRank;

         for (let cardRank in handSets){

            if (handSets[cardRank] === 4) {sets = "Four of a Kind";}

            if (handSets[cardRank] === 3) {

               if (sets === "Pair") {sets = "Full House";}

               else {sets = "Three of a Kind";}

            }

            if (handSets[cardRank] === 2) {

               if (sets === "Three of a Kind") {sets = "Full House";}

               else if (sets === "Pair") {sets = "Two Pair";}

               else {sets = "Pair"; pairRank = cardRank;}

            }

         }

         if (sets === "Pair" && pairRank >= 11) {

            sets = "Jacks or Better";

         }

         return sets;

      }

      // Return a text string describing the hand for draw poker //

      if (hasRoyalFlush()) {return "Royal Flush";}

      else if (hasStraightFlush()) {return "Straight Flush";}

      else if (hasFlush()) {return "Flush";}

      else if (hasStraight()) {return "Straight";}

      else {

         let sets = hasSets();

         if (sets === "Pair" || sets === "none") {sets = "No Winner";}

         return sets;

      }

   }

   /\* ------------------------------------------------+

   |             End of the  handType() function      |

   +-------------------------------------------------\*/

}

Js08\_txt.js – slide 42

"use strict";

/\*    JavaScript 7th Edition

      Chapter 8

      Chapter case

      Draw Poker Game using Object Oriented Programming

      Author:

      Date:

      Filename:       js08.js

 \*/

window.addEventListener("load", playDrawPoker);

function playDrawPoker() {

   // Reference buttons and images within the Poker Game page

   let dealButton = document.getElementById("dealB");

   let drawButton = document.getElementById("drawB");

   let standButton = document.getElementById("standB");

   let resetButton = document.getElementById("resetB");

   let statusBox = document.getElementById("status");

   let betSelection = document.getElementById("bet");

   let bankBox = document.getElementById("bank");

   let cardImages = document.querySelectorAll("img.cardImg");

   //Set the initial bank and bet values

   pokerGame.currentBank = 500;

   pokerGame.currentBet = 25;

   // Create a deck of shuffled cards

   let myDeck = new pokerDeck();

   myDeck.shuffle();

   // Create an empty poker hand object

   let myHand = new pokerHand(5);

   // Display the current bank value

   bankBox.value = pokerGame.currentBank;

   // Change the bet when the selection changes

   betSelection.onchange = function(){

      pokerGame.currentBet = parseInt(this.value);

   }

   dealButton.addEventListener("click", function(){

      if (pokerGame.currentBank >= pokerGame.currentBet){

         //Enable the Draw and Stand buttons after the initial deal

         dealButton.disabled = true;

         betSelection.disabled = true;

         drawButton.disabled = false;

         standButton.disabled = false;

         statusBox.texContent = "";

         //Reduce the bank by the size of the bet

         bankBox.value = pokerGame.placeBet();

         // Get a new deck is there are less than 10 cards left

        if (myDeck.cards.length < 10){

         myDeck = new pokerDeck();

         myDeck.shuffle();

        }

        // Deal 5 cards from the deck to the hand

        myDeck.dealTo(myHand);

         // Display the card images on the table

         for (let i = 0; i < cardImages.length; i++) {

            cardImages[i].src = myHand.cards[i].cardImage();

        // Flip the card images when clicked

        cardImages[i].onclick = function() {

         if (this.src.includes("cardback.png")){

            this.src = myHand.cards[i].cardImage();

         } else {

            // show the back of the card

            this.src = "cardback.png";

         }

        }

        //console.log(myDeck, myHand);

         }

      }

      else {

         statusBox.textContent = "Insufficient Funds";

      }

   });

      dealButton.addEventListener("click", function() {

      if (pokerGame.currentBank >= pokerGame.currentBet) {

         // Enable the Draw and Stand buttons after the initial deal

         dealButton.disabled = true;        // Turn off the Deal button

         betSelection.disabled = true;      // Turn off the Bet Selection list

         drawButton.disabled = false;       // Turn on the Draw button

         standButton.disabled = false;      // Turn on the Stand Button

         statusBox.textContent = "";        // Erase any status messages

      }

   });

   drawButton.addEventListener("click", function() {

      // Enable the Deal and Bet options when the player chooses to draw new cards

      dealButton.disabled = false;        // Turn on the Deal button

      betSelection.disabled = false;      // Turn on the Bet Selection list

      drawButton.disabled = true;         // Turn off the Draw button

      standButton.disabled = true;        // Turn off the Stand Button

 // Replace cards marked to be discarded

 for (let i = 0; i < cardImages.length; i++) {

   if (cardImages[i].src.includes("cardback.png")) {

      // Replace the card and its image on the table

      myHand.replaceCard(i, myDeck);

      cardImages[i].src = myHand.cards[i].cardImage();

   }

}

 // Evaluate the hand drawn by user

 statusBox.textContent = myHand.getHandValue();

  // Update the bank value

  bankBox.value = pokerGame.payBet(statusBox.textContent);

   });

   standButton.addEventListener("click", function() {

      // Enable the Deal and Bet options when the player chooses to stand with their hand

      dealButton.disabled = false;        // Turn on the Deal button

      betSelection.disabled = false;      // Turn on the Bet Selection list

      drawButton.disabled = true;         // Turn off the Draw button

      standButton.disabled = true;        // Turn off the Stand Button

      // Evaluate the hand drawn by user

      statusBox.textContent = myHand.getHandValue();

      // Update the bank value

      bankBox.value = pokerGame.payBet(statusBox.textContent);

   });

   // Reload the current page when the Reset button is clicked

   resetButton.addEventListener("click", function() {

      location.reload();

   });

}

**Updated code from slide 1 to slide 27**

"use strict";

/\*    JavaScript 7th Edition

      Chapter 8

      Chapter case

      Draw Poker Game using Object Oriented Programming

      Author:

      Date:

      Filename:       js08.js

 \*/

      let pokerGame = {

   currentBank: null,

         currentBet: null,

      placeBet: function(){

      this.currentBank -= this.currentBet;

      return this.currentBank;

   }

};

/\* Constructor function for poker cards \*/

function pokerCard(cardSuit, cardRank) {

   this.suit = cardSuit;

   this.rank = cardRank;

}

/\* Method to reference the image source file for a card \*/

pokerCard.prototype.cardImage = function() {

  return this.rank + "\_" + this.suit + ".png";

};

/\* Constructor function for poker cards \*/

function pokerCard(cardSuit, cardRank){

   this.suit = cardSuit;

   this.rank = cardRank;

}

/\* Constructor function for poker decks\*/

function pokerDeck(){

  //List the suits and ranks

  let suits = ["clubs", "diamonds", "hearts", "spades"];

  let ranks = [

    "2",

    "3",

    "4",

    "5",

    "6",

    "7",

    "8",

    "9",

    "10",

    "jack",

    "queen",

    "king",

    "ace",

  ];

  this.cards = [];

  //Add a card for each combination of suit and rank

  for (let i = 0; i < 4; i++) {

    for (let j = 0; j < 13; j++) {

      //Add a pokerCard object

      this.cards.push(new pokerCard(suits[i], ranks[j]));

    }

  }

  // Method to randomly sort the cards in the deck

  this.shuffle = function () {

    this.cards.sort(function () {

      return 0.5 - Math.random();

    });

   };

   // Method to deal cards from the deck into a hand

   this.dealTo = function(pokerHand){

      let cardsDealt = pokerHand.cards.length;

      pokerHand.cards = this.cards.splice(0, cardsDealt);

   };

};

  /\* Constructor function for poker hands \*/

  function pokerHand(handLength) {

    this.cards = new Array(handLength);

  }

//};

window.addEventListener("load", playDrawPoker);

function playDrawPoker() {

  // Reference buttons and images within the Poker Game page

  let dealButton = document.getElementById("dealB");

  let drawButton = document.getElementById("drawB");

  let standButton = document.getElementById("standB");

  let resetButton = document.getElementById("resetB");

  let statusBox = document.getElementById("status");

  let betSelection = document.getElementById("bet");

  let bankBox = document.getElementById("bank");

  let cardImages = document.querySelectorAll("img.cardImg");

  //Set the initial bank and bet values

  pokerGame.currentBank = 500;

  pokerGame.currentBet = 25;

  // Create a deck of shuffled cards

  let myDeck = new pokerDeck();

  myDeck.shuffle();

  // Create an empty poker hand object

  let myHand = new pokerHand(5);

  // Display the current bank value

  bankBox.value = pokerGame.currentBank;

  // Change the bet when the selection changes

  betSelection.onchange = function () {

    pokerGame.currentBet = parseInt(this.value);

  };

  dealButton.addEventListener("click", function () {

    if (pokerGame.currentBank >= pokerGame.currentBet) {

      // Enable the Draw and Stand buttons after the initial deal

      dealButton.disabled = true; // Turn off the Deal button

      betSelection.disabled = true; // Turn off the Bet Selection list

      drawButton.disabled = false; // Turn on the Draw button

      standButton.disabled = false; // Turn on the Stand Button

      statusBox.textContent = ""; // Erase any status messages

      //Reduce the bank by the size of the bet

      bankBox.value = pokerGame.placeBet();

      // Get a new deck is there are less than 10 cards left

      if (myDeck.cards.length < 10) {

        myDeck = new pokerDeck();

        myDeck.shuffle();

      }

      // Deal 5 cards from the deck to the hand

      myDeck.dealTo(myHand);

      // Display the card images on the table

      for (let i = 0; i < cardImages.length; i++) {

        cardImages[i].src = myHand.cards[i].cardImage();

      }

      //console.log(myDeck, myHand);

    } else {

      statusBox.textContent = "Insufficient Funds";

    }

  });

  drawButton.addEventListener("click", function () {

    // Enable the Deal and Bet options when the player chooses to draw new cards

    dealButton.disabled = false; // Turn on the Deal button

    betSelection.disabled = false; // Turn on the Bet Selection list

    drawButton.disabled = true; // Turn off the Draw button

    standButton.disabled = true; // Turn off the Stand Button

  });

  standButton.addEventListener("click", function () {

    // Enable the Deal and Bet options when the player chooses to stand with their hand

    dealButton.disabled = false; // Turn on the Deal button

    betSelection.disabled = false; // Turn on the Bet Selection list

    drawButton.disabled = true; // Turn off the Draw button

    standButton.disabled = true; // Turn off the Stand Button

  });

  // Reload the current page when the Reset button is clicked

  resetButton.addEventListener("click", function () {

    location.reload();

  });

}

**Final code - Js08\_txt.js**

"use strict";

/\*    JavaScript 7th Edition

      Chapter 8

      Chapter case

      Draw Poker Game using Object Oriented Programming

      Author:

      Date:

      Filename:       js08.js

 \*/

      let pokerGame = {

        currentBank: null,

        currentBet: null,

        placeBet: function () {

          this.currentBank -= this.currentBet;

          return this.currentBank;

        },

        payBet: function (type) {

          let pay = 0;

          switch (type) {

            case "Royal Flush":

              pay = 250;

              break;

            case "Straight Flush":

              pay = 50;

              break;

            case "Four of a Kind":

              pay = 25;

              break;

            case "Full House":

              pay = 9;

              break;

            case "Flush":

              pay = 6;

              break;

            case "Straight":

              pay = 4;

              break;

            case "Three of a Kind":

              pay = 3;

              break;

            case "Two Pair":

              pay = 2;

              break;

            case "Jacks or Better":

              pay = 1;

              break;

          }

          this.currentBank += pay \* this.currentBet;

          return this.currentBank;

        },

      };

/\* Constructor function for poker cards \*/

function pokerCard(cardSuit, cardRank) {

   this.suit = cardSuit;

   this.rank = cardRank;

}

/\* Method to reference the image source file for a card \*/

pokerCard.prototype.cardImage = function() {

  return this.rank + "\_" + this.suit + ".png";

};

/\* Constructor function for poker cards \*/

function pokerCard(cardSuit, cardRank){

   this.suit = cardSuit;

   this.rank = cardRank;

}

/\* Constructor function for poker decks\*/

function pokerDeck(){

  //List the suits and ranks

  let suits = ["clubs", "diamonds", "hearts", "spades"];

  let ranks = [

    "2",

    "3",

    "4",

    "5",

    "6",

    "7",

    "8",

    "9",

    "10",

    "jack",

    "queen",

    "king",

    "ace",

  ];

  this.cards = [];

  //Add a card for each combination of suit and rank

  for (let i = 0; i < 4; i++) {

    for (let j = 0; j < 13; j++) {

      //Add a pokerCard object

      this.cards.push(new pokerCard(suits[i], ranks[j]));

    }

  }

  // Method to randomly sort the cards in the deck

  this.shuffle = function () {

    this.cards.sort(function () {

      return 0.5 - Math.random();

    });

   };

   // Method to deal cards from the deck into a hand

   this.dealTo = function(pokerHand){

      let cardsDealt = pokerHand.cards.length;

      pokerHand.cards = this.cards.splice(0, cardsDealt);

   };

};

  /\* Constructor function for poker hands \*/

  function pokerHand(handLength) {

    this.cards = new Array(handLength);

}

// Method to replace a card in a hand with a card from a deck

   pokerHand.prototype.replaceCard = function(index, pokerDeck){

      this.cards[index] = pokerDeck.cards.shift();

   };

   // Method to determine the value of the pokerHand

pokerHand.prototype.getHandValue = function() {

  return handType(this);

  /\* ------------------------------------------------+

   | The handType() function returns a text string of |

   | the type of hand held by 5-card poker hand.      |

   +-------------------------------------------------\*/

  function handType(pokerHand) {

    /\* Determine the rank value of each card in the hand

         by creating a property named rankValue         \*/

    for (let i = 0; i < pokerHand.cards.length; i++) {

      if (pokerHand.cards[i].rank === "ace") {

        pokerHand.cards[i].rankValue = 14;

      } else if (pokerHand.cards[i].rank === "king") {

        pokerHand.cards[i].rankValue = 13;

      } else if (pokerHand.cards[i].rank === "queen") {

        pokerHand.cards[i].rankValue = 12;

      } else if (pokerHand.cards[i].rank === "jack") {

        pokerHand.cards[i].rankValue = 11;

      } else {

        pokerHand.cards[i].rankValue = parseInt(pokerHand.cards[i].rank);

      }

    }

    /\* Function to return the highest ranked value in a five-card hand \*/

    function highCard() {

      return Math.max(

        pokerHand.cards[0].rankValue,

        pokerHand.cards[1].rankValue,

        pokerHand.cards[2].rankValue,

        pokerHand.cards[3].rankValue,

        pokerHand.cards[4].rankValue

      );

    }

    /\* Function to test for the presence of a flush in which all

         five cards have the same suit \*/

    function hasFlush() {

      let firstSuit = pokerHand.cards[0].suit;

      return pokerHand.cards.every(function (card) {

        return card.suit === firstSuit;

      });

    }

    /\* Function to test for the presence of a straight in which the

         rank value of the cards can be placed in sequential order \*/

    function hasStraight() {

      pokerHand.cards.sort(function (a, b) {

        return a.rankValue - b.rankValue;

      });

      return pokerHand.cards.every(function (card, i, cards) {

        if (i > 0) {

          return cards[i].rankValue - cards[i - 1].rankValue === 1;

        } else {

          return true;

        }

      });

    }

    /\* Function to test for the presence of a straight flush \*/

    function hasStraightFlush() {

      return hasFlush() && hasStraight();

    }

    /\* Function to test for the presence of a royal flush

         which consists of 10-J-Q-K-A of the same suit \*/

    function hasRoyalFlush() {

      return hasStraightFlush() && highCard() === 14;

    }

    /\* Function to test for the presence of: pairs, two pairs,

         three of a kind, four of a kind, and full houses  \*/

    function hasSets() {

      // handSets creates an associative array of the duplicates in the hand

      let handSets = {};

      pokerHand.cards.forEach(function (card) {

        if (handSets.hasOwnProperty(card.rankValue)) {

          handSets[card.rankValue]++;

        } else {

          handSets[card.rankValue] = 1;

        }

      });

      let sets = "none";

      let pairRank;

      for (let cardRank in handSets) {

        if (handSets[cardRank] === 4) {

          sets = "Four of a Kind";

        }

        if (handSets[cardRank] === 3) {

          if (sets === "Pair") {

            sets = "Full House";

          } else {

            sets = "Three of a Kind";

          }

        }

        if (handSets[cardRank] === 2) {

          if (sets === "Three of a Kind") {

            sets = "Full House";

          } else if (sets === "Pair") {

            sets = "Two Pair";

          } else {

            sets = "Pair";

            pairRank = cardRank;

          }

        }

      }

      if (sets === "Pair" && pairRank >= 11) {

        sets = "Jacks or Better";

      }

      return sets;

    }

    // Return a text string describing the hand for draw poker //

    if (hasRoyalFlush()) {

      return "Royal Flush";

    } else if (hasStraightFlush()) {

      return "Straight Flush";

    } else if (hasFlush()) {

      return "Flush";

    } else if (hasStraight()) {

      return "Straight";

    } else {

      let sets = hasSets();

      if (sets === "Pair" || sets === "none") {

        sets = "No Winner";

      }

      return sets;

    }

  }

  /\* ------------------------------------------------+

   |             End of the  handType() function      |

   +-------------------------------------------------\*/

}

//};

window.addEventListener("load", playDrawPoker);

function playDrawPoker() {

  // Reference buttons and images within the Poker Game page

  let dealButton = document.getElementById("dealB");

  let drawButton = document.getElementById("drawB");

  let standButton = document.getElementById("standB");

  let resetButton = document.getElementById("resetB");

  let statusBox = document.getElementById("status");

  let betSelection = document.getElementById("bet");

  let bankBox = document.getElementById("bank");

  let cardImages = document.querySelectorAll("img.cardImg");

  //Set the initial bank and bet values

  pokerGame.currentBank = 500;

  pokerGame.currentBet = 25;

  // Create a deck of shuffled cards

  let myDeck = new pokerDeck();

  myDeck.shuffle();

  // Create an empty poker hand object

  let myHand = new pokerHand(5);

  // Display the current bank value

  bankBox.value = pokerGame.currentBank;

  // Change the bet when the selection changes

  betSelection.onchange = function () {

    pokerGame.currentBet = parseInt(this.value);

  };

  dealButton.addEventListener("click", function () {

    if (pokerGame.currentBank >= pokerGame.currentBet) {

      // Enable the Draw and Stand buttons after the initial deal

      dealButton.disabled = true; // Turn off the Deal button

      betSelection.disabled = true; // Turn off the Bet Selection list

      drawButton.disabled = false; // Turn on the Draw button

      standButton.disabled = false; // Turn on the Stand Button

      statusBox.textContent = ""; // Erase any status messages

      //Reduce the bank by the size of the bet

      bankBox.value = pokerGame.placeBet();

      // Get a new deck is there are less than 10 cards left

      if (myDeck.cards.length < 10) {

        myDeck = new pokerDeck();

        myDeck.shuffle();

      }

      // Deal 5 cards from the deck to the hand

      myDeck.dealTo(myHand);

      // Display the card images on the table

      for (let i = 0; i < cardImages.length; i++) {

        cardImages[i].src = myHand.cards[i].cardImage();

        // Flip the card images when clicked

        cardImages[i].onclick = function () {

          if (this.src.includes("cardback.png")) {

            this.src = myHand.cards[i].cardImage();

          } else {

            // show the back of the card

            this.src = "cardback.png";

          }

        };

        //console.log(myDeck, myHand);

      }

    }

 else {

      statusBox.textContent = "Insufficient Funds";

    }

  });

  drawButton.addEventListener("click", function () {

    // Enable the Deal and Bet options when the player chooses to draw new cards

    dealButton.disabled = false; // Turn on the Deal button

    betSelection.disabled = false; // Turn on the Bet Selection list

    drawButton.disabled = true; // Turn off the Draw button

    standButton.disabled = true; // Turn off the Stand Button

    // Replace cards marked to be discarded

    for (let i = 0; i < cardImages.length; i++) {

      if (cardImages[i].src.includes("cardback.png")) {

        // Replace the card and its image on the table

        myHand.replaceCard(i, myDeck);

        cardImages[i].src = myHand.cards[i].cardImage();

      }

    }

    // Evaluate the hand drawn by user

    statusBox.textContent = myHand.getHandValue();

    // Update the bank value

    bankBox.value = pokerGame.payBet(statusBox.textContent);

  });

  standButton.addEventListener("click", function () {

    // Enable the Deal and Bet options when the player chooses to stand with their hand

    dealButton.disabled = false; // Turn on the Deal button

    betSelection.disabled = false; // Turn on the Bet Selection list

    drawButton.disabled = true; // Turn off the Draw button

    standButton.disabled = true; // Turn off the Stand Button

    // Evaluate the hand drawn by user

    statusBox.textContent = myHand.getHandValue();

  });

  // Reload the current page when the Reset button is clicked

  resetButton.addEventListener("click", function () {

    location.reload();

  });

}