!doctype html>

<html>

<head>

<meta charset="utf-8">

<title>Poker</title>

<style type="text/css">

.cardBlock {display:inline-block; padding: 6px; border: 1px solid #333; margin-right: 5px; width: 40px; text-align:center; height: 50px; line-height: 50px;}

#cards {margin-bottom:20px;}

.highlight {background-color:#99E3E7}

#message {height: 70px; font-size: 20px;}

</style>

</head>

<body>

<h1>Poker Slots!</h1>

<div id='cards'></div>

<div id="message">&nbsp;</div>

<button id="btnAction">Draw</button>

<script>

const instructions = "Click a card to hold for the next draw.<br>Click again to remove the hold.";

const winMessage = "Your best poker hand is: ";

payouts = {

"high card" : 0,

"one pair" : 1,

"two pair" : 2,

"three of a kind" : 3,

"straight" : 4,

"flush" : 5,

"full house" : 7,

"four of a kind" : 20,

"straight flush" : 50,

"royal flush" : 100

}

/\*\*\*\*\*\*\*\*\*\*\*\* Deck \*\*\*\*\*\*\*\*\*\*\*\*\*\*/

function Deck()

{

this.suits=['S', 'C', 'D', 'H'];

this.face=['J','Q','K'];

this.used = [];

this.dealCard = dealCard;

}

function dealCard()

{

if (this.used.length < 10) //"clear" the deck if insufficient cards for this hand

this.used = [];

do {

r = Math.round(Math.random()\*51 + 1);

} while (this.used.includes(r));

this.used.push(r);

suit = this.suits[r%4];

rank = r % 13 + 1;

value = Math.min(rank, 10);

rank = rank > 10 ? this.face[rank-11] : rank;

rank = rank==1 ? rank='A' : rank;

return new Card(suit, rank, value);

}

/\*\*\*\*\*\*\*\*\*\*\*\* Card \*\*\*\*\*\*\*\*\*\*\*\*\*\*/

function Card(suit='', rank=0, value=0)

{

this.suit = suit;

this.rank=rank;

this.value = value;

this.held = false;

this.unhold= () => this.held = false;

this.displayCardBlock = (i) => `<div class='cardBlock' name="${i}">${this.rank}${this.suit}</div>`;

this.updateCard = updateCard;

}

//gets a new card

function updateCard()

{

if (!this.held)

{

c = theDeck.dealCard();

this.suit = c.suit;

this.value = c.value;

this.rank = c.rank;

}

this.held = false;

return this;

}

/\*\*\*\*\*\*\*\*\*\*\*\* Player \*\*\*\*\*\*\*\*\*\*\*\*\*\*/

function Player(moneyAmount=100, bet=5)

{

this.money=moneyAmount;

this.bet = bet;

this.cards = [new Card(),new Card(),new Card(),new Card(),new Card()];

this.dealHand= dealHand;

this.handString = handString;

this.holdCard = (i) => this.cards[i].held = true;

this.unholdCard = (i) => this.cards[i].held = false;

}

function dealHand()

{

this.cards.forEach((card, i) => card.updateCard().unhold());

document.getElementById("cards").innerHTML =

this.cards.reduce((s, card, i) =>s + card.displayCardBlock(i), "");

//alternatives

//this.cards.forEach((card,i)=> s+= card.displayCardBlock(i));

//document.getElementById("cards").innerHTML =

//this.cards.map((card, i) =>card.displayCardBlock(i)).join("");

}

function handString()

{

return this.cards.map((card) => (card.rank==10 ? 'T' : card.rank) + card.suit.toLowerCase()).join(",");

}

/\*\*\*\*\*\*\*\*\*\*\*\* Game \*\*\*\*\*\*\*\*\*\*\*\*\*\*/

function Game(player)

{

this.player = player;

this.play=play;

this.unholdCard = i => this.player.unholdCard(i);

this.holdCard = i => this.player.holdCard(i);

}

function play()

{

document.getElementById("message").innerHTML = instructions;

this.player.dealHand();

addCardEvents();

}

/\*\*\*\*\*\*\*\*\*\*\*\* Utility funcitons \*\*\*\*\*\*\*\*\*\*\*/

function addCardEvents()

{

cardBlocks = document.getElementsByClassName("cardBlock");

for (i=0; i<cardBlocks.length; i++)

cardBlocks[i].addEventListener("click", cardBlockToggle);

}

function removeCardEvents()

{

cardBlocks = document.getElementsByClassName("cardBlock");

for (i=0; i<cardBlocks.length; i++)

{

cardBlocks[i].removeEventListener("click", cardBlockToggle);

}

}

function cardBlockToggle()

{

if (this.classList.contains("highlight"))

{

this.classList.remove("highlight");

poker1.unholdCard(this.getAttribute("name"));

}

else

{

this.classList.add("highlight");

poker1.holdCard(this.getAttribute("name"));

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* window.onload \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

window.onload = function() {

document.getElementById("btnAction").addEventListener("click",function(){

this.innerText == "Draw" ? drawEvent(this) : playAgainEvent(this);

}) // end btnAction add event handler

} // end window.onload event handler

async function drawEvent(btn)

{

poker1.player.dealHand();

document.getElementById("message").innerHTML = "Calculating ...";

btn.style.display= "none";

await getPokerHandRank();

btn.style.display= "block";

btn.innerText = "Play Again";

}

function playAgainEvent(btn)

{

btn.innerText = "Draw";

document.getElementById("message").innerHTML = instructions;

removeCardEvents();

poker1.play();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\* Get Poker Hand Rank \*\*\*\*\*\*/

async function getPokerHandRank()

{

strHand = poker1.player.handString();

console.log(strHand)

const options = {

method: 'GET',

headers: {

'X-RapidAPI-Host': 'sf-api-on-demand-poker-odds-v1.p.rapidapi.com',

'X-RapidAPI-Key': 'api key needed'

}

};

await fetch('https://sf-api-on-demand-poker-odds-v1.p.rapidapi.com/describe?hand='+strHand, options)

.then(response => response.json())

.then(response => {console.log(response);

document.getElementById("message").innerHTML = getWinMessage(response.data.hand\_name);

})

.catch(err => console.error(err));

}

function getWinMessage(hand)

{

return winMessage + hand + "<br>Your payout is " + payouts[hand.toLowerCase()] \* poker1.player.bet

}

poker1 = new Game(new Player());

theDeck = new Deck();

poker1.play();

</script>

</body>

</html>