**blackjack**

**logic code**

// ======= ======= ======= saveNewPlayer ======= ======= =======

function saveNewPlayer() {

console.log("-- saveNewPlayer");

// ======= validate name entered

var playerName = $("#playerName\_field").val();

if ((playerName == '') || (playerName == null) || (playerName == undefined)) {

gameObject.infoDisplay.text('Need some players before starting!');

// ======= find empty slot in playersArray

} else {

playerCount = gameObject.playersArray.length - 1;

console.log(" gameObject.playersArrayP1: " + gameObject.playersArray.length);

console.log(" playerCount: " + playerCount);

// ======= validate 3 players max

if (playerCount >= 3) {

gameObject.infoDisplay.text('uh oh... already 3 players!');

$("#playerName\_field").val('');

} else {

newPlayerIndex = playerCount + 1;

newPlayer = new Player (newPlayerIndex, playerName, [],0, 193,23, 3,2,1,20...

updateDisplay('newPlayer', newPlayerIndex);

}

}

}

// ======= ======= ======= dealCards ======= ======= =======

function dealCards() {

console.log("dealCards");

// ======= initialize deck

// var suitArray = ['&clubs; ','&diams; ','&hearts; ','&spades; '];

var suitArray = ['C','D','H','S'];

var valueArray = ['A','2','3','4','5','6','7','8','9','10','J','Q','K'];

var pointsArray = [11, 2, 3, 4, 5, 6, 7, 8, 9, 10, 10, 10, 10];

var nextValue, nextPoints, cardPoints, nextCard, nextPlayer;

for (var i = 0; i < suitArray.length; i++) {

nextSuit = suitArray[i];

for (var j = 0; j < valueArray.length; j++) {

nextValue = valueArray[j];

nextPoints = pointsArray[j];

deckArray.push(nextValue + nextSuit);

deckPointsArray.push(nextPoints);

}

}

// ======= clear previous player hand

for (var i = 0; i < (gameObject.playersArray.length); i++) {

nextPlayer = gameObject.playersArray[i];

nextPlayer.currentHand = [];

}

// ======= deal cards to each player and dealer

var winnersArray = [];

for (var i = 0; i < (gameObject.playersArray.length); i++) {

nextPlayer = gameObject.playersArray[i];

// ======= getNextCard

for (var j = 0; j < 2; j++) {

cardPoints = getNextCard(); // get card from deck; shrink deck

nextCard = cardPoints[0];

nextPoints = cardPoints[1];

nextPlayer.currentHand.push(nextCard);

nextPlayer.currentScore = nextPlayer.currentScore + nextPoints;

console.log(' nextCard: ' + nextCard);

}

// ======= if Ace card and > 21 (2 aces)

if (nextPlayer.currentScore > 21) {

for (var k = 0; k < nextPlayer.currentHand.length; k++) {

nextCard = nextPlayer.currentHand[k];

// ======= change Ace value to 1

if (nextCard.indexOf("A") > 0) {

nextPlayer.currentScore = nextPlayer.currentScore - 10;

break;

}

}

}

// ======= instant winner

if (nextPlayer.currentScore == 21) {

winnersArray.push(nextPlayer);

}

}

// ======= set default player (unless winner)

if (winnersArray.length > 0) {

calculateWinner();

} else {

gameObject.activePlayer = 1;

updateDisplay('deal');

}

}

// ======= ======= ======= hitMe ======= ======= =======

function hitMe() {

console.log("-- hitMe --");

var nextPlayer = gameObject.playersArray[gameObject.activePlayer];

var cardPoints = getNextCard(); // get card from deck; shrink deck

var nextCard = cardPoints[0];

var nextPoints = cardPoints[1];

console.log(' nextCard: ' + nextCard);

nextPlayer.currentHand.push(nextCard);

nextPlayer.currentScore = nextPlayer.currentScore + nextPoints;

// ======= check for Aces and adjust score

if (nextPlayer.currentScore > 21) {

for (var i = 0; i < nextPlayer.currentHand.length; i++) {

nextCard = nextPlayer.currentHand[i];

// ======= change A value to 1 if > 21 score

if (nextCard.indexOf("A") > 0) {

nextPlayer.currentScore = nextPlayer.currentScore - 10;

break;

}

}

// ======= score still high after adjustment

if (nextPlayer.currentScore > 21) {

gameObject.infoDisplay.text("Bummer... you're over 21!");

nextPlayer.totalBal = nextPlayer.totalBal - nextPlayer.totalBet;

updateDisplay('hitMe', nextPlayer.playerIndex);

updateDisplay('nextTurn', nextPlayer.playerIndex);

} else {

updateDisplay('hitMe', nextPlayer.playerIndex);

}

} else {

updateDisplay('hitMe', nextPlayer.playerIndex);

}

}

// ======= ======= ======= passNhold ======= ======= =======

function passNhold() {

console.log("passNhold");

updateDisplay('nextTurn');

}

// ======= ======= ======= dealerHits ======= ======= =======

function dealerHits() {

console.log("-- dealerHits --");

var dealer = gameObject.playersArray[0];

var whileCounter = 0;

var hitDelay;

while (dealer.currentScore < 18) {

console.log(' whileCounter: ' + whileCounter++);

cardPoints = getNextCard(); // get card from deck; shrink deck

nextCard = cardPoints[0];

nextPoints = cardPoints[1];

console.log(' nextCard: ' + nextCard);

dealer.currentHand.push(nextCard);

dealer.currentScore = dealer.currentScore + nextPoints;

hitDelay = setTimeout(function(){

updateDisplay('dealerHits');

clearTimeout(hitDelay);

}, 600);

}

clearTimeout(hitDelay);

calculateWinner();

}

// ======= ======= ======= calculateWinner ======= ======= =======

function calculateWinner() {

console.log("calculateWinner");

var nextPlayer, nextName, winLossLabel;

var dealerScore = gameObject.playersArray[0].currentScore;

var playerWinLoss = 0;

var playerWinLossString = 'RESULTS!!\nDealer score: ' + dealerScore + '\n';

// =======

for (var i = 0; i < (gameObject.playersArray.length); i++) {

nextPlayer = gameObject.playersArray[i];

nextName = nextPlayer.playerName;

console.log(" nextName: " + nextName);

if (nextName != 'dealer') {

// ======= calculate win/loss results

playerWinLoss = (nextPlayer.chipsBet\_1) + (nextPlayer.chipsBet\_5 \* 5) + (nextPlayer.chipsBet\_10 \* 10);

playerWinLossString = playerWinLossString + nextName + ' score: ' + nextPlayer.currentScore + winLossLabel + playerWinLoss + '\n';

console.log(" playerWinLoss: " + playerWinLoss);

// ======= calculate wins/losses for players

if ((nextPlayer.currentScore > dealerScore) && (nextPlayer.currentScore < 22)) {

winLossLabel = ' and won $';

nextPlayer.totalBal = nextPlayer.totalBal + playerWinLoss;

nextPlayer.chipsBal\_1 = nextPlayer.chipsBal\_1 + nextPlayer.chipsBet\_1;

nextPlayer.chipsBal\_5 = nextPlayer.chipsBal\_5 + nextPlayer.chipsBet\_5;

nextPlayer.chipsBal\_10 = nextPlayer.chipsBal\_10 + nextPlayer.chipsBet\_10;

} else {

winLossLabel = ' and lost $';

nextPlayer.totalBal = nextPlayer.totalBal - playerWinLoss;

nextPlayer.chipsBal\_1 = nextPlayer.chipsBal\_1 - nextPlayer.chipsBet\_1;

nextPlayer.chipsBal\_5 = nextPlayer.chipsBal\_5 - nextPlayer.chipsBet\_5;

nextPlayer.chipsBal\_10 = nextPlayer.chipsBal\_10 - nextPlayer.chipsBet\_10;

}

// ======= set bet values on player object

nextPlayer.chipsBet\_1 = 0;

nextPlayer.chipsBet\_5 = 0;

nextPlayer.chipsBet\_10 = 0;

}

// ======= initialize all players/dealer for next hand

nextPlayer.currentHand = [];

nextPlayer.currentScore = 0;

nextPlayer.totalBet = 0;

}

updateDisplay('calculateWinner');

alert(playerWinLossString);

dealCards();

}