A screenshot of a cell phone

Description automatically generated

A picture containing clipart

Description automatically generated

WEB Technology 512

Assignment 2



Anthony Gordon Kruger

9208175020080

Student number 139399

061 034 0820

[antman.kruger@gmail.com](mailto:antman.kruger@gmail.com)

Table of conte­­­­­­­­­­­nts:

[Question 1: 4](#_Toc20050339)

[Question 2: 13](#_Toc20050340)

[Question 3: 19](#_Toc20050341)

[Bibliography 22](#_Toc20050342)

Question 1: (70)

***(Tic-Tac-Toe)*** Create a class Tic-Tac-Toe that will enable you to write a program to play Tic-Tac-Toe. The class contains a private 3-by-3 two-dimensional array. Use an enumeration to represent the value in each cell of the array. The enumeration’s constants should be named X, O and EMPTY (for a position that does not contain an X or an O). The constructor should initialize the board elements to EMPTY. Allow two human players. Wherever the first player moves, place an X in the specified square, and place an O wherever the second player moves. Each move must be to an empty square. After each move, determine whether the game has been won and whether it’s a draw. Also, allow the player to specify whether he or she wants to go first or second.

Screen shot of the game initializing and requesting player what symbol they would like to be:

A screenshot of a cell phone

Description automatically generated

Screenshot of game initialized, the coloured square is an on mouse over event for the div and message showing it is X’s turn:

A screenshot of a cell phone

Description automatically generated

Screenshot showing that it is now O’s turn and X has been displayed in the game:

A screenshot of a cell phone

Description automatically generated

Screenshot showing that O has played and that it is now X’s turn:

A screenshot of a cell phone

Description automatically generated

Screenshot of the game ending in a tie:

A screenshot of a cell phone

Description automatically generated

Screenshot of X winning the game:

A screenshot of a cell phone

Description automatically generated

Screenshot of O winning the game:

A screenshot of a cell phone

Description automatically generated

Screenshot of an alert when a block is clicked twice:

A screenshot of a cell phone

Description automatically generated

HTML source code:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <meta http-equiv="X-UA-Compatible" content="ie=edge">

    <link rel="stylesheet" href="stylesheet.css">

    <script defer src="script.js"></script>

    <title>Tic Tac Toe</title>

</head>

<body>

    <h1>Tic Tac Toe</h1>

    <h2></h2>

    <br>

    <div class="flex-container flex-column">

        <div class="flex-container flex-wrap" id="board">

            <div class="square" id="cell0"></div>

            <div class="square" id="cell1"></div>

            <div class="square" id="cell2"></div>

            <div class="square" id="cell3"></div>

            <div class="square" id="cell4"></div>

            <div class="square" id="cell5"></div>

            <div class="square" id="cell6"></div>

            <div class="square" id="cell7"></div>

            <div class="square" id="cell8"></div>

        </div>

        <br>

        <br>

        <br>

        <br>

        <button id="reset-button"><strong>New Game</strong></button>

    </div>

</body>

</html>

CSS source code:

body {

    text-align: center;

    font-family: Arial, Helvetica, sans-serif;

}

h2 {

    margin: 0 auto;

    font-size: 35px;

    margin: 10px;

}

.flex-container {

    display: flex;

    justify-content: center;

    align-items: center;

}

h1 {

    font-size: 100px;

}

.flex-column {

    height: 100%;

    width: 100%;

    flex-direction: column;

}

.flex-wrap {

    flex-wrap: wrap;

    height: 432px;

    width: 432px;

}

.square {

    border: 3px solid rgba(0, 0, 0, .75);

    height: 140px;

    width: 140px;

    font-size: 130px;

    text-align: center;

    justify-content: center;

    cursor: pointer;

    border-collapse: collapse;

}

body {

    font-family: "Courier New", Courier, monospace;

    letter-spacing: 2px;

    word-spacing: 2px;

    color: #201E59;

    font-weight: bold;

    text-decoration: none;

    font-style: normal;

    font-variant: normal;

    text-transform: uppercase;

}

.square:hover {

    background-color: rgba(224, 223, 209, 88);

}

#reset-button {

    text-align: center;

    font-size: 40px;

    border: 0px;

    height: 80px;

    width: 300px;

    margin: 10px;

    background-color: rgba(224, 223, 209, 88);

    color: #201E59;

    border-radius: 12px;

    font-family: "Courier New", Courier, monospace;

}

body {

    background-color: rgb(243, 245, 250);

}

#cell0,

#cell1,

#cell2 {

    border-top: 0px;

    border-bottom: 0px;

}

#cell0,

#cell3,

#cell6 {

    border-left: 0px;

}

#cell2,

#cell5,

#cell8 {

    border-right: 0px;

}

#cell6,

#cell7,

#cell8 {

    border-bottom: 0px;

    border-top: 0px;

}

Javascript source code:

*//this is an array that is made from all the div elements within the div container*

const squares = Array.from(document.querySelectorAll("#board div"));

*// This is an array of winning combinations used to check for whether the game has been won or not*

const winningCombos = [

    [0, 1, 2],

    [3, 4, 5],

    [6, 7, 8],

    [0, 3, 6],

    [1, 4, 7],

    [2, 5, 8],

    [0, 4, 8],

    [2, 4, 6]

];

*//Variables needed to initialize the game before init function*

let turn;

let board;

const messages = document.querySelector("h2");

let win;

*//event listeners needed for click events occuring on the page*

document.getElementById("board").addEventListener("click", handleTurn); *//listener for click  events on divs*

document.getElementById("reset-button").addEventListener("click", init) *//listener for click on new game button*

*// getWinner function: used in the handleTurn function to determaine if there is a winner,*

*//  if so then the event listener is remove so that no more blocks can be selected*

*//  and the message to the players is adusted accordingly*

function getWinner() {

    let winner = null;

    winningCombos.forEach(function(combo, index) {

        if (board[combo[0]] && board[combo[0]] === board[combo[1]] && board[combo[0]] === board[combo[2]]) {

            winner = board[combo[0]];

        };

    });

    return winner ? winner : board.includes('') ? null : 'T';

};

*// handleTurn function: used by the event listener to handle the click event parsed onto the  document,*

*// iterates through the squares array and changes the relevent index in the array into the  symbol for the player*

*// and also changes the turn variable to be the next players turn and makes sure that the same  block isnt played twice*

*// it also makes a call to render in order to display the changes made to the squares*

function handleTurn() {

    let idx = squares.findIndex(function(square) {

        return square === event.target;

    });

    if (board[idx] === "X" || board[idx] === "O") {

        alert("block already taken!!");

    } else {

        board[idx] = turn

        turn = turn === "X" ? "O" : "X";

        win = getWinner();

        render();

    };

};

*// render function: this function is used in the init and the handleTurn functions. it is used  to update the display and according to who is currently*

*// meant to be playing and it maps out the markings of board array onto the square divs*

function render() {

    board.forEach(function(mark, index) {

        squares[index].textContent = mark;

    });

    messages.textContent = win === 'T' ? `That's a TIE, try again!` : win ? `${win} wins the game!` : `It's ${turn}'s turn!`;

    messages.textContent === `${win} wins the game!` ? document.getElementById("board").removeEventListener("click", handleTurn) : document.getElementById("board").addEventListener("click", handleTurn);;

};

*//init function: this functions is called at the bottom of the script and is used to initialize the game.*

*//  it is also called by a click event for the reset/new game button.*

function init() {

    win = null;

    document.getElementById("board").addEventListener("click", handleTurn);

    messages.textContent = "Waiting to initialize the game, press new game to start.";

    turn = prompt("Player 1, would you like to be an X or an O?, click ok to default to X", "X");

    turn = turn.toUpperCase()

    messages.textContent = win === 'T' ? `That's a TIE, try again!` : win ? `${win} wins the game!` : `It's ${turn}'s turn!`;

    board = [

        "", "", "",

        "", "", "",

        "", "", ""

    ];

    render();

};

*// initializing function that starts the game*

init();

Question 2: (20)

Write a javascript program that asks the user to enter up to 10 golf scores, which are to be stored in an array. You should provide a means for the user to terminate input prior to entering 10 scores. The program should display all the scores on one line and report the average score. Handle input, display, and the average calculation with three separate array processing functions.

Screenshot of scoresheet when the page loads:

A screenshot of a cell phone

Description automatically generated

Screenshot of prompt that appears when enter scores button is pressed. Every time a score is entered a new prompt appears asking for another score. The user can also cancel at this point.

A screenshot of a cell phone

Description automatically generated

Screenshot of scoresheet with total scores and score average.

A screenshot of a social media post

Description automatically generated

Screenshot of total score and average score should the user press cancel instead of entering a new score.

A screenshot of a social media post

Description automatically generated

HTML source code:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <meta http-equiv="X-UA-Compatible" content="ie=edge">

    <script defer src="script.js"></script>

    <link rel="stylesheet" href="stylesheet.css">

    <title>question 2 golf scores</title>

</head>

<br>

<br>

<br>

<br>

<br>

<br>

<br>

<center>

    <table id="scorechart">

        <tr>

            <th colspan="13">

                scores

            </th>

        </tr>

        <tr>

            <td>

            </td>

            <td>

                hole 1

            </td>

            <td>

                hole 2

            </td>

            <td>

                hole 3

            </td>

            <td>

                hole 4

            </td>

            <td>

                hole 5

            </td>

            <td>

                hole 6

            </td>

            <td>

                hole 7

            </td>

            <td>

                hole 8

            </td>

            <td>

                hole 9

            </td>

            <td>

                hole 10

            </td>

            <td>

                TOTAL:

            </td>

            <td>

                AVERAGE SCORE:

            </td>

        </tr>

        <tr>

            <td>

                player 1:

            </td>

            <td id="0">

            </td>

            <td id="1">

            </td>

            <td id="2">

            </td>

            <td id="3">

            </td>

            <td id="4">

            </td>

            <td id="5">

            </td>

            <td id="6">

            </td>

            <td id="7">

            </td>

            <td id="8">

            </td>

            <td id="9">

            </td>

            <td id="total">

            </td>

            <td id="average">

            </td>

        </tr>

    </table>

    <br>

    <button id="getScores"> enter scores</button>

    <br>

    <br>

    <br>

    <br>

    <br>

    <br>

    <h2 id="message"></h2>

</center>

<body>

</body>

</html>

CSS source code:

table,

th,

td {

    border: 2px solid black;

    border-collapse: collapse;

    margin: 10px;

    padding: 10px;

}

button {

    width: 200px;

    height: 50px;

    font-size: 20px;

}

Javascript source code:

*//event listeners*

document.getElementById("scorechart").addEventListener("click", init);

document.getElementById("getScores").addEventListener("click", init);

*//onload variables*

let total;

let score;

let scoresArray = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0];

let holesPlayed = 0;

*//functions*

*//initializes game, called by enter scores button click event*

function init() {

    getScores();

};

*//uses a for loop to request scores through prompts for a user, breaks if cancel is pressed*

function getScores() {

    clearScores();

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

        score = prompt(`What is your score for course ${i + 1} ?\r\nIf you click cancel your total score will be tallied up to where you have stopped playing and your average score will be calculated.`, "0");

        checkIfStillPlaying();

        addToTotal();

        render();

        if (score === 0) {

            break;

        };

    }

    if (holesPlayed === 0) {

        document.getElementById("message").innerHTML = `Your total score was ${total} and your average score was ${total}`;

    } else {

        document.getElementById("message").innerHTML = `Your total score was ${total} and your average score was ${total / holesPlayed}`;

    }

};

*//function to clear scores when enter scores button is pressed after a game has been played*

function clearScores() {

    holesPlayed = 0;

    total = 0;

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

        scoresArray[i] = 0;

        document.getElementById("" + i).innerHTML = 0;

    };

};

*//function to check whether cancel button was pressed on alert box*

function checkIfStillPlaying() {

    if (score != null) {

        holesPlayed += 1;

    };

    score === null ? score = 0 : scoresArray[i] = parseFloat(score);

};

*//function to add score entered to total*

function addToTotal() {

    total += scoresArray[i];

};

*//function to add score to scoresheet*

function render() {

    document.getElementById("" + i).innerHTML = score;

    document.getElementById("total").innerHTML = total;

    if (score == 0 && holesPlayed == 0) {

        document.getElementById("average").innerHTML = total;

    } else {

        document.getElementById("average").innerHTML = total / holesPlayed;

    }

};

Question 3: (10)

Write a javascript program that asks the user to enter his or her first name and then last name, and then constructs, stores, and displays a third string consisting of the user’s last name followed by a comma, a space, and first name. Use string objects and methods from the string header file. A sample run could look like this:

Enter your first name: **Paul**

Enter your last name: **Smith**

Here’s the information in a single string: Smith, Paul

Screenshot of a prompt asking for your first name when document loads, default name is John if no name is entered.

A screenshot of a computer

Description automatically generated

Screenshot of a prompt asking for your surname when document loads, default surname is Doe if no surname is entered.

A screenshot of a computer

Description automatically generated

Screenshot of requested result

A screenshot of a computer

Description automatically generated

HTML source code:

<!DOCTYPE html>

<html lang="en">

<script defer src="script.js"></script>

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <meta http-equiv="X-UA-Compatible" content="ie=edge">

    <title>Question 3</title>

</head>

<body>

    <p id="details"></p>

</body>

</html>

No CSS used

Javascript source code:

firstName = prompt("User, what is your first name?", "John");

lastName = prompt("User, what is your last name?", "Doe")

document.getElementById("details").innerHTML = `Enter your firstname:

    ${firstName} </br> Enter your last name: ${lastName}

    <br>Here's is the information in a single string:

${lastName}, ${firstName}`;

# Bibliography

freeCodeCamp.org, 2017. *JavaScript Tic Tac Toe Project Tutorial - Unbeatable AI w/ Minimax Algorithm.* [Online]   
Available at: https://www.youtube.com/watch?v=P2TcQ3h0ipQ  
[Accessed 05 09 2019].

w3schools.com, n.d. *Window prompt() method.* [Online]   
Available at: https://www.w3schools.com/jsref/met\_win\_prompt.asp  
[Accessed 05 09 2019].

w3schools.com, n.d.. *HTML DOM querySelectorAll() Method.* [Online]   
Available at: https://www.w3schools.com/jsref/met\_document\_queryselectorall.asp  
[Accessed 06 09 2019].

portEXE, 2019. *How To Code Tic-Tac-Toe In Plain JavaScript - Functional JavaScript Tutorial.* [Online]   
Available at: https://www.youtube.com/watch?v=yaPUl31nypk  
[Accessed 06 09 2019].

train, T. C., 2019. *Coding Challenge #149: Tic Tac Toe.* [Online]   
Available at: https://www.youtube.com/watch?v=GTWrWM1UsnA   
[Accessed 06 09 2109].

Blog, 1., n.d. *How To Make A TIC TAC TOE Game In Javascript.* [Online]   
Available at: https://1bestcsharp.blogspot.com/2017/11/javascript-tic-tac-toe-game.html  
[Accessed 06 09 2019].

w3schools.com, n.d.. *JavaScript Array forEach() Method ❮ JavaScript Array Reference.* [Online]   
Available at: https://www.w3schools.com/jsref/jsref\_foreach.asp   
[Accessed 10 09 2019].

Anon., n.d.. *Conditional (ternary) operator.* [Online]   
Available at: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Conditional\_Operator   
[Accessed 10 09 2019].