Training Day 8

30th June 2025

TOPICS COVERED

- **Map, Filter, Reduce (Higher-Order Functions)**

Higher-order functions are functions that either take other functions as arguments, return a function, or both. In JavaScript, map(), filter(), and reduce() are powerful tools for handling arrays more efficiently.

1. map() – The map() method creates a new array by transforming each element of the original array using a callback function. It does not mutate the original array.

Example:

```
let numbers = [1, 2, 3];

let doubled = numbers.map(num => num * 2);

console.log(doubled);  // [2, 4, 6]
```

2. filter() – The filter() method creates a new array with only those elements that pass a condition provided in a callback function.

Example:

```
let nums = [5, 10, 15, 20];

let filtered = nums.filter(num => num > 10);

console.log(filtered);  // [15, 20]
```

3. reduce() – The reduce() method applies a function against an accumulator and each element in the array to reduce it to a single value.

Example:

```
let values = [1, 2, 3, 4];

let sum = values.reduce((acc, curr) => acc + curr, 0);

console.log(sum);  // 10
```

- **DOM Manipulation – (Document Object Model)**

The DOM is a programming interface for web documents. It represents the page so that programs can change the document structure, style, and content dynamically using JavaScript.

Accessing Elements in the DOM

Method	Description
getElementById()	Returns the element with the specified ID
getElementsByClassName().	Returns an HTMLCollection of all elements with the given class
getElementsByTagName()	Returns all elements with the given tag
querySelector()	Returns the first element that matches a CSS selector
querySelectorAll()	Returns all elements that match a CSS selector (NodeList)

Example:

```
let title = document.getElementById("mainTitle");
let items = document.querySelectorAll(".list-item");
```

Modifying Elements

You can update content, styles, attributes, and more.

Example:

```
let el = document.getElementById("demo");
el.textContent = "Updated Text";
el.style.color = "blue";
el.setAttribute("class", "highlight");
```

TOOLS USED

Visual Studio Code (VS Code)

Chrome Browser (JavaScript Console)

TASK

TIC TAC TOE GAME

#html code

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Tic Tac Toe</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div id="gameContainer">
        <h1>Tic Tac Toe</h1>
        <div id="cellContainer">
            <div cellIndex="0" class="cell"></div>
            <div cellIndex="1" class="cell"></div>
            <div cellIndex="2" class="cell"></div>
            <div cellIndex="3" class="cell"></div>
            <div cellIndex="4" class="cell"></div>
            <div cellIndex="5" class="cell"></div>
            <div cellIndex="6" class="cell"></div>
            <div cellIndex="7" class="cell"></div>
            <div cellIndex="8" class="cell"></div>
        </div>
        <h2 id="statusText"></h2>
```

```
        <button id="restartBtn">Restart</button>
    </div>
    <script src="index.js"></script>
</body>
</html>
```

#css code

```
.cell{
    width: 75px;
    height: 75px;
    border: 2px solid;
    box-shadow: 0 0 0 2px;
    line-height: 75px;
    font-size: 50px;
    cursor: pointer;
    background-color: #f3dfdf;
    transition: transform 0.2s ease-in-out;}
.cell:hover {
  transform: scale(1.05);}
h1{
    font-size: 50px;
    color: rgb(93, 3, 3);
}
#gameContainer{
    font-family:Verdana, Geneva, Tahoma, sans-serif;
    text-align: center;}
#cellContainer{
    display: grid;
    grid-template-columns: repeat(3, auto);
```

```
    width: 225px;
    margin: auto;
}
*{background-color: #f3ebeb;}
```

#js code

```
const cells = document.querySelectorAll(".cell");
const statusText = document.querySelector("#statusText");
const restartBtn = document.querySelector("#restartBtn");
const winConditions = [
    [0, 1, 2],
    [3, 4, 5],
    [6, 7, 8],
    [0, 3, 6],
    [1, 4, 7],
    [2, 5, 8],
    [0, 4, 8],
    [2, 4, 6]
];
let options = ["", "", "", "", "", "", "", "", ""];
let currentPlayer = "X";
let running = false;
initializeGame();
function initializeGame(){
    cells.forEach(cell => cell.addEventListener("click", cellClicked));
    restartBtn.addEventListener("click", restartGame);
    statusText.textContent = `${currentPlayer}'s turn`;
    running = true;}
function cellClicked(){
```

```
    const cellIndex = this.getAttribute("cellIndex");
    if(options[cellIndex] != "" || !running){
        return;
    }
    updateCell(this, cellIndex);
    checkWinner();
}
function updateCell(cell, index){
    options[index] = currentPlayer;
    cell.textContent = currentPlayer;
}
function changePlayer(){
    currentPlayer = (currentPlayer == "X") ? "O" : "X";
    statusText.textContent = `${currentPlayer}'s turn`;
}
function checkWinner(){
    let roundWon = false;

    for(let i = 0; i < winConditions.length; i++){
        const condition = winConditions[i];
        const cellA = options[condition[0]];
        const cellB = options[condition[1]];
        const cellC = options[condition[2]];

        if(cellA == "" || cellB == "" || cellC == ""){
            continue;
        }
        if(cellA == cellB && cellB == cellC){
            roundWon = true;
```

```
            break; }
        }
        if(roundWon){
            statusText.textContent = `${currentPlayer} wins!`;
            running = false;
        }
        else if(!options.includes("")){
            statusText.textContent = `Draw!`;
            running = false;
        }
        else{
            changePlayer();
        }
    }
    function restartGame(){
        currentPlayer = "X";
        options = ["", "", "", "", "", "", "", "", ""];
        statusText.textContent = `${currentPlayer}'s turn`;
        cells.forEach(cell => cell.textContent = "");
        running = true;
    }
```