**Лабораторна робота № 3**

*Робота з одновимірними та багатовимірними масивами.*

**Хід роботи:**

**Завдання 1:**

**1.1:**

Розмістіть на сторінці **div**-блок з ідентифікатором **“paragraphs”**. Розмістіть у ньому 5 абзаців з довільним текстом. Сформуйте масив, який міститиме 5 елементів - кількість символів, які входять до відповідного абзацу тексту.

Для пошуку абзаців використовуйте метод **querySelectorAll** та відповідний CSS-селектор. Для доступу до тексту, який входить в абзац використовуйте властивість **innerText** для DOM-елемента. Виведіть у консоль отриманий масив.

**Лістинг програми:**

HTML file:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Document</title>

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

</head>

<body>

    <div id="paragraphs">

        <p>Перший абзац з довільним текстом</p>

        <p>Другий абзац з довільним текстом</p>

        <p>Третій абзац з довільним текстом</p>

        <p>Четвертий абзац з довільним текстом</p>

        <p>П'ятий абзац з довільним текстом</p>

    </div>

    <script src="script.js"></script>

</body>

</html>

Java Script file:

let arr = [];

let paragraphs = document.querySelectorAll("p");

paragraphs.forEach(paragraph => {

  let text = paragraph.textContent;

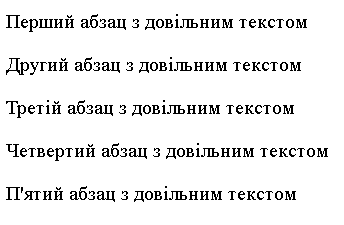
  let characterCount = text.length;

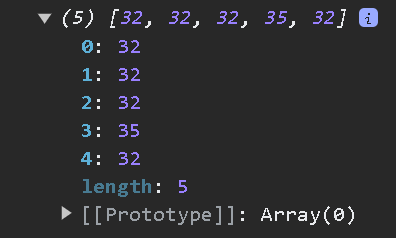
  arr.push(characterCount);

});

console.log(arr);

**Результати виконання:**

****

****

**1.2:**Дано масив цілих чисел:

let arr = [20, 17, 4, -4, 10, -9, 13, 4, 12, 22, 13, 19, 1, 3];

-  Для заданого масиву визначте та виведіть у консоль:

* найменше та найбільше значення;
* кількість парних та непарних чисел;
* кількість чисел, що містять дві цифри;
* кількість додатних та кількість від’ємних чисел.

- Відсортуйте масив **arr** за допомогою методу Sort у порядку зростання значень та виведіть у консоль результат.

-  Створіть масив з оберненим порядком елементів та виведіть його у консоль.

**Лістинг програми:**

HTML file:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Task (1) 1.2</title>

</head>

<body>

    <script src="script.js"></script>

</body>

</html>

Java Script file:

let symbols\_counter = [0, 0, 0, 0, 0]

let strings = document.querySelectorAll("p")

function fillingCounter() {

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

        symbols\_counter[i] = strings[i].innerText.length

        console.log(symbols\_counter[i])

    }

}

fillingCounter()

let arr = [20, 17, 4, -4, 10, -9, 13, 4, 12, 22, 13, 19, 1, 3]

console.log(Math.min(...arr))

console.log(Math.max(...arr))

function parseElements(arr) {

    let counter = 0

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

        if (arr[i] % 2 == 0) {

            counter++

        }

    }

    return counter

}

function nonParseElements(arr) {

    let counter = 0

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

        if (arr[i] % 2 != 0) {

            counter++

        }

    }

    return counter

}

console.log(parseElements(arr))

console.log(nonParseElements(arr))

function hasTwoDigits(arr) {

    let counter = 0

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

        if (parseInt(arr[i] / 10) != 0) {

            counter++

        }

    }

    return counter

}

console.log(hasTwoDigits(arr))

function positiveNumbers(arr) {

    let counter = 0

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

        if (arr[i] > 0) {

            counter++

        }

    }

    return counter

}

function negativeNumbers(arr) {

    let counter = 0

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

        if (arr[i] < 0) {

            counter++

        }

    }

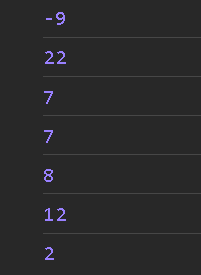
    return counter

}

console.log(positiveNumbers(arr))

console.log(negativeNumbers(arr))

**Результати виконання:**



**Завдання 2:**

Розмістіть на сторінці 3 div-блоки з класами “table”. У кожному з них розмістіть табличку. У першому блоці - табличку розміром 3х3, другому - 4х4, третьому - 5х5.

В CSS-файлі розмістіть стиль для класу “selected”, який передбачає жовтий фон.

За допомогою querySelectorAll знайдіть div-блоки з класом “table”. Для кожного знайденого блоку у циклі знайдіть всі вкладені у нього теги td. Для кожного другого тега td додайте клас “selected”.

**Лістинг програми:**

HTML file:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Task 2</title>

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

</head>

<body>

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

    <div class="table">

        <table>

            <tr>

                <td></td>

                <td></td>

                <td></td>

            </tr>

            <tr>

                <td></td>

                <td></td>

                <td></td>

            </tr>

            <tr>

                <td></td>

                <td></td>

                <td></td>

            </tr>

        </table>

    </div>

    <div class="table">

        <table>

            <tr>

                <td></td>

                <td></td>

                <td></td>

                <td></td>

            </tr>

            <tr>

                <td></td>

                <td></td>

                <td></td>

                <td></td>

            </tr>

            <tr>

                <td></td>

                <td></td>

                <td></td>

                <td></td>

            </tr>

            <tr>

                <td></td>

                <td></td>

                <td></td>

                <td></td>

            </tr>

        </table>

    </div>

    <div class="table">

        <table>

            <tr>

                <td></td>

                <td></td>

                <td></td>

                <td></td>

                <td></td>

            </tr>

            <tr>

                <td></td>

                <td></td>

                <td></td>

                <td></td>

                <td></td>

            </tr>

            <tr>

                <td></td>

                <td></td>

                <td></td>

                <td></td>

                <td></td>

            </tr>

            <tr>

                <td></td>

                <td></td>

                <td></td>

                <td></td>

                <td></td>

            </tr>

            <tr>

                <td></td>

                <td></td>

                <td></td>

                <td></td>

                <td></td>

            </tr>

        </table>

    </div>

</body>

</html>

Java Script file:

let table = document.querySelectorAll(".table")

table.forEach(function (tables) {

    tds = tables.querySelectorAll("td")

    tds.forEach(function (td, index) {

        if (index % 2 != 0) {

            td.classList.add("selected")

        }

    })

})

CSS file:

.selected {

    background: yellow;

}

td {

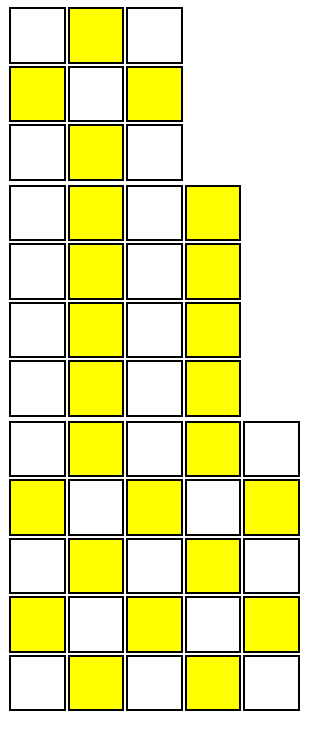
    width: 40px;

    height: 40px;

    border: 2px black solid;

}

**Результати виконання:**

****

**Завдання 3:**

Створіть матрицю розміру N на M, яка містить цілі числа (наприклад, 7х5 та числами на власний розсуд). Знайдіть та виведіть у консоль:

* кількість додатних елементів;
* кількість рядків, які не містять жодного нульового елемента;
* кількість стовпців, які містять хоча б один нульовий елемент;
* номер рядка, в якому знаходиться найдовша серія однакових елементів;
* добуток елементів в тих рядках, які не містять від’ємних елементів;
* суму елементів в тих стовпцях, які не містять від’ємних елементів;
* суму елементів в тих стовпцях, які  містять хоча б один від’ємний елемент;
* транспоновану матрицю;

**Лістинг програми:**

HTML file:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Document</title>

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

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

</head>

<body>

    <div>

        <label>Rows: <input id="row" type="number"></label>

        <label>Colls: <input id="coll" type="number"></label>

        <button onclick="submit()">Submit</button>

    </div>

</body>

</html>

CSS file:

\* {

    margin-top: 10px;

    margin-bottom: 10px;

}

div {

    display: flex;

    flex-direction: column;

}

button {

    width: 200px;

    height: 40px;

}

Java Script file:

let matrix = []

let matrixRows = 0

let matrixColls = 0

function createMatrix(arr) {

    arr = []

    matrixRows = document.getElementById("row").value

    matrixColls = document.getElementById("coll").value

    for (let i = 0; i < matrixRows; ++i) {

        let row = []

        for (let j = 0; j < matrixColls; ++j) {

            row.push(Math.floor(Math.random() \* (9 + 9 + 1)) - 9)

        }

        arr.push(row)

    }

    return arr

}

function positiveElements(matrix) {

    let counter = 0

    for (let i = 0; i < matrixRows; ++i) {

        for (let j = 0; j < matrixColls; ++j) {

            if (matrix[i][j] > 0) {

                counter++

            }

        }

    }

    return counter

}

function displayMatrix(matrix) {

    console.log(matrix)

}

function rowsWithoutNullElements(matrix) {

    let counter = 0

    let positive

    for (let i = 0; i < matrixRows; ++i) {

        positive = false

        for (let j = 0; j < matrixColls; ++j) {

            if (matrix[i][j] != 0) {

                positive = true

            }

        }

        if (positive) {

            counter++

        }

    }

    return counter

}

function collsWithNullElements(matrix) {

    let counter = 0

    let positive

    for (let i = 0; i < matrixColls; ++i) {

        positive = false

        for (let j = 0; j < matrixRows; ++j) {

            if (matrix[j][i] === 0) {

                positive = true

            }

        }

        if (positive) {

            counter++

        }

    }

    return counter

}

function sameElementsInRow(matrix) {

    let row = 0

    let mostSameElementsInRow = 0

    for (let i = 0; i < matrixRows; ++i) {

        let elementsInRow = []

        let seriesOfSameElements = 0

        for (let j = 0; j < matrixColls; ++j) {

            if (elementsInRow.indexOf(matrix[i][j]) === -1) {

                elementsInRow.push(matrix[i][j])

            }

            else {

                seriesOfSameElements++

            }

        }

        if (mostSameElementsInRow < seriesOfSameElements) {

            mostSameElementsInRow = seriesOfSameElements

            row = i

        }

    }

    return row

}

function mullRowsWithoutNulls(matrix) {

    for (let i = 0; i < matrixRows; ++i) {

        let nulls = false

        for (let j = 0; j < matrixColls; ++j) {

            if (matrix[i][j] === 0) {

                nulls = true

            }

        }

        if (!nulls) {

            let mullRow = 1

            for (let j = 0; j < matrixColls; ++j) {

                mullRow \*= matrix[i][j]

            }

            console.log(`Mulltiply in rows ${i} without nulls = ${mullRow}`)

        }

    }

}

function mullCollsWithoutNulls(matrix) {

    for (let i = 0; i < matrixColls; ++i) {

        let nulls = false

        for (let j = 0; j < matrixRows; ++j) {

            if (matrix[j][i] === 0) {

                nulls = true

            }

        }

        if (!nulls) {

            let mullRow = 1

            for (let j = 0; j < matrixRows; ++j) {

                mullRow \*= matrix[j][i]

            }

            console.log(`Mulltiply in colls ${i} without nulls = ${mullRow}`)

        }

    }

}

function summCollsWithoutNulls(matrix) {

    for (let i = 0; i < matrixColls; ++i) {

        let negatives = false

        for (let j = 0; j < matrixRows; ++j) {

            if (matrix[j][i] < 0) {

                negatives = true

            }

        }

        if (negatives) {

            let mullRow = 1

            for (let j = 0; j < matrixRows; ++j) {

                mullRow += matrix[j][i]

            }

            console.log(`Summ in colls ${i} where elements < 0 = ${mullRow}`)

        }

    }

}

function transposeMatrix(matrix) {

    console.log(`Transpose matrix: `)

    let transposedMatrix = []

    for (let i = 0; i < matrixColls; ++i) {

        let row = []

        for (let j = 0; j < matrixRows; ++j) {

            row.push(matrix[j][i])

        }

        transposedMatrix.push(row)

    }

    console.log(transposedMatrix)

}

function maxSumMainDiagonals(matrix) {

    let elemDiagonals = []

    let counter = 0

    for (let i = 0; i < matrixRows; ++i) {

        for (let j = i + 1, k = 0; j < matrixColls; ++j) {

            if (i === 0 && j !== matrixColls - 1) {

                elemDiagonals.push(matrix[i][j])

                elemDiagonals.push(matrix[j][i])

            }

            if (i !== 0) {

                elemDiagonals[k] += matrix[i][j]

                elemDiagonals[k + 1] += matrix[j][i]

                k += 2

            }

        }

    }

    console.log(`The maximum elements parallel to the main diagonal of the matrix ${Math.max(...elemDiagonals)}`)

}

function submit() {

    matrix = createMatrix(matrix)

    displayMatrix(matrix)

    console.log(positiveElements(matrix))

    console.log(rowsWithoutNullElements(matrix))

    console.log(collsWithNullElements(matrix))

    console.log(sameElementsInRow(matrix))

    mullRowsWithoutNulls(matrix)

    mullCollsWithoutNulls(matrix)

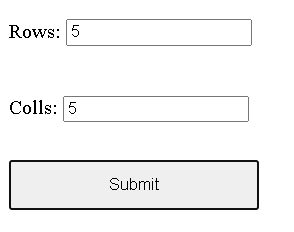
    summCollsWithoutNulls(matrix)

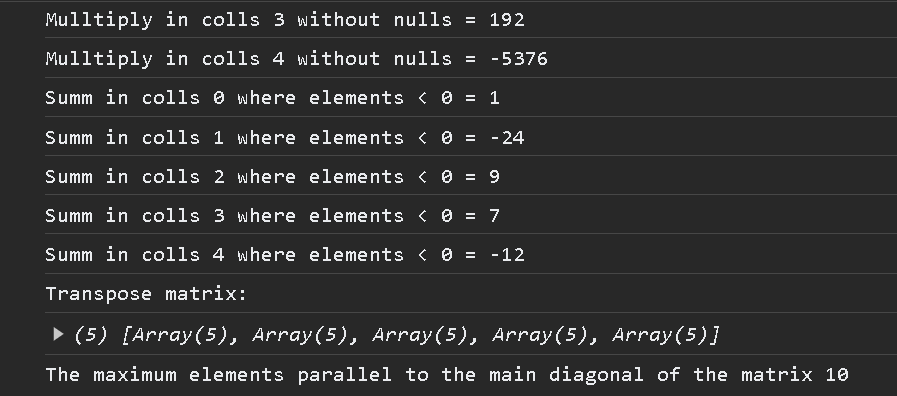
    transposeMatrix(matrix)

    maxSumMainDiagonals(matrix)

}

**Результати програми:**





GitHub:

https://github.com/djproblematic/frontend