ГУАП КАФЕДРА № 43

ОТЧЕТ

ЗАЩИЩЕН С ОЦЕНКОЙ ПРЕПОДАВАТЕЛЬ

|  |  |  |
| --- | --- | --- |
| ассистент |  | М.А. Мурашова |
| должность, уч. степень, звание | подпись, дата | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №3 |
| ОСНОВЫ JAVASCRIPT |
| по дисциплине: WEB-ТЕХНОЛОГИИ |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |
| --- | --- | --- |
| СТУДЕНТ ГР. | 4232 | А.В. Матвеев |
|  | подпись, дата | инициалы, фамилия |

Санкт-Петербург

2025

**Содержание**

[1. Цель работы 3](#_Toc192092890)

[2. Задание 3](#_Toc192092891)

[3. Вариант задания 3](#_Toc192092892)

[4. Названия средств, использованных при выполнении работы 3](#_Toc192092893)

[5. Выполнение задания 4](#_Toc192092894)

[6. Листинг программного кода 8](#_Toc192092895)

[7. Выводы 9](#_Toc192092896)

# 1. Цель работы

Знакомство с языком JavaScript.

# 2. Задание

**Базовое задание**

Используя редактор javascript (см. листинг 1) написать программу, которая выполняет задание по варианту (базовая часть) для матрицы произвольного размера (n\*m) и использует массив. Обязательно использовать методы массива. Вывести на страницу матрицу в форме прямоугольника. Размеры матрицы получать через диалог с пользователем.

**Расширенное задание**

Нарисовать заданную вариантом фигуру, используя объект canvas и образец скрипта из листинга 2. Повторяющие фрагменты формировать с помощью циклов.

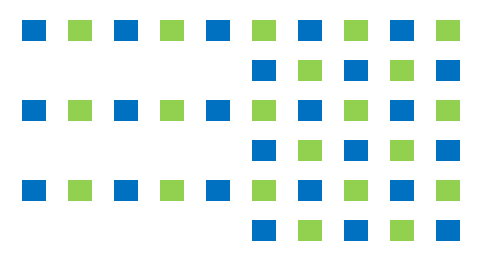
Размер фигуры должен задаваться пользователем через диалог. Под размером понимается количество элементов по вертикали и горизонтали. На пример, для варианта 1 задание можно сформулировать так: построить фигуру, состоящую из четного количества (n) вертикальных синий линий, которые пересекаются с нечетным количеством (n-1) горизонтальных линий, при чем меньшая половина горизонтальных линий синего цвета, а большая половина красного цвета. Пересечения линий должны образовывать квадраты.

При желании до рисования фигуры можно сформировать матрицу, отражающую заполнение прямоугольника, в который вписана фигура, элементами, а потом визуализировать эту матрицу.

# 3. Вариант задания

Базовое задание: 12) Сформировать матрицу, каждый элемент которой равен произведению номера строки на номер столбца.

Расширенное задание: Вариант 12



# 4. Названия средств, использованных при выполнении работы

1. Редактор кода Visual Studio Code.

2. Браузер Mozilla Firefox.

# 5. Выполнение задания

Ниже представлены таблицы (1-4) описывающие все переменные, методы и функции.

Таблица 1 – описание всех переменных программ

|  |  |  |
| --- | --- | --- |
| **Имя переменной** | **Тип** | **Назначение** |
| rows | Number | Хранит количество строк матрицы, введенное пользователем. |
| cols | Number | Хранит количество столбцов матрицы, введенное пользователем. |
| matrix | Array | Хранит сгенерированную матрицу, состоящую из значений на основе индексов. |
| myrezult | String | Хранит строковое представление матрицы с разделением табуляцией и строк. |
| canvas | HTMLElement | Ссылка на элемент канваса (HTML-элемент), на котором рисуется паттерн. |
| ctx | CanvasRenderingContext2D | Контекст рисования канваса, используемый для рисования на нем. |
| size | Number | Размер клетки (сторона квадрата), используемого для рисования на канвасе. |
| blue | String | Цвет (строка в формате HEX) для рисования синего квадрата. |
| green | String | Цвет (строка в формате HEX) для рисования зеленого квадрата. |
| pattern | Array (двумерный) | Массив, который хранит паттерн (шаблон) для рисования на канвасе. |
| i | Number | Индекс строки при обходе двумерного массива pattern. |
| j | Number | Индекс столбца при обходе двумерного массива pattern. |

Таблица 2 – описание использованных методов массива

|  |  |  |
| --- | --- | --- |
| **Название метода** | **Параметры** | **Назначение** |
| Array.from() | arrayLike, mapFn (опционально) | Создает новый массив из объекта-подобного массива или итерируемого объекта. В данном случае используется для создания массива заданной длины и заполнения его значениями, вычисляемыми через функцию. |
| map() | callback(item, index, array) | Создает новый массив, с каждым элементом которого выполняется функция callback. В данном случае используется для трансформации элементов (например, заполнение строк матрицы). |
| join() | separator (по умолчанию ',') | Объединяет все элементы массива в строку, используя указанный разделитель. Здесь используется для соединения элементов каждой строки матрицы с табуляцией. |
| push() | value | Добавляет в конец массива указанное значение |

Таблица 3 – описание использованных методов других стандартных объектов

|  |  |  |
| --- | --- | --- |
| **Название метода** | **Параметры** | **Назначение** |
| parseInt() | string, radix (optional) | Преобразует строку в целое число. Здесь используется для преобразования введенных данных (количества строк и столбцов) в числа. |
| isNaN() | value | Проверяет, является ли переданное значение NaN. В данном коде проверяется корректность введенных данных для строк и столбцов матрицы. |
| alert() | message | Выводит модальное окно с сообщением для пользователя. В данном случае, используется для уведомления о некорректных данных. |
| getElementById() | id | Возвращает элемент с указанным id. Здесь используется для получения элемента <canvas> по его id. |
| getContext() | contextType | Возвращает объект контекста рисования для указанного канваса. Здесь используется для получения контекста рисования в 2D. |
| fillRect() | x, y, width, height | Рисует прямоугольник на канвасе, начиная с координат (x, y) и с размерами width и height. Используется для рисования клеток на канвасе. |
| clearRect() | x, y, width, height | Очищает прямоугольную область канваса. В данном коде используется для очистки канваса перед началом рисования нового паттерна. |

Таблица 4 – описание использованных стрелочных функций

|  |  |  |
| --- | --- | --- |
| **Стрелочная функция** | **Параметры** | **Назначение** |
| (\_, i) => Array.from() | (\_, i) | Эта стрелочная функция используется в Array.from() для генерации строк матрицы. Параметры \_ (не используется) и i — индекс текущей строки. |
| (\_, j) => (i + 1) \* (j + 1) | (\_, j) | Эта стрелочная функция используется в Array.from() для вычисления значений в каждой ячейке матрицы. Параметры \_ (не используется) и j — индекс текущего столбца. |
| row => row.join('\t') | row | Эта стрелочная функция используется для объединения элементов каждой строки матрицы с разделителем \t (табуляция). |
| () => {} | Нет параметров | Стрелочные функции без параметров используются в качестве обработчиков (например, для создания массива). |

На рисунках 1-4 представлена работа программы базового задания, вычисляющая матрицу, содержащую каждый элемент, представленный произведением номера строки и столбца. У пользователя запрашивается количество строк и столбцов, если же введены не числа, то выводится ошибка, если же всё в порядке – выводится матрица.

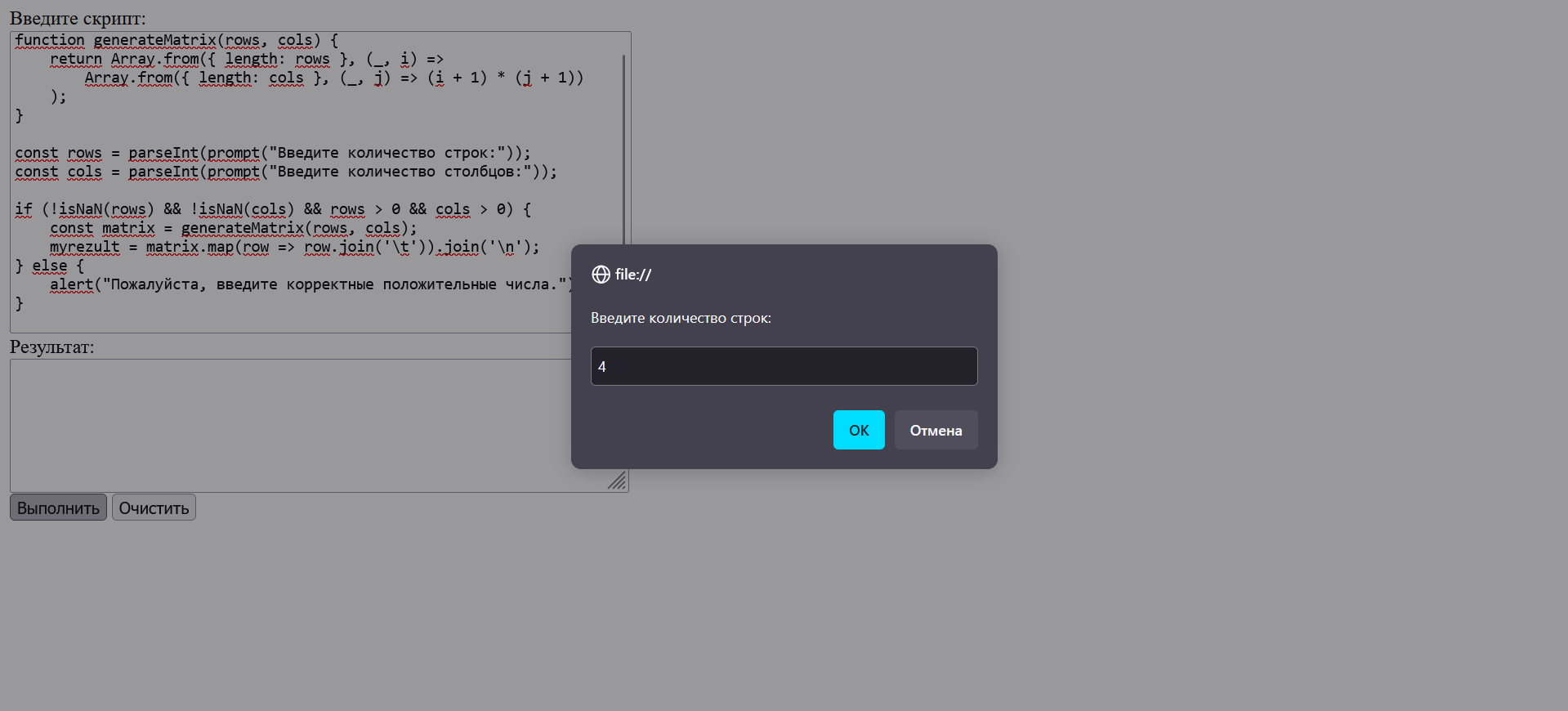


Рисунок 1 – запрос у пользователя количества строк

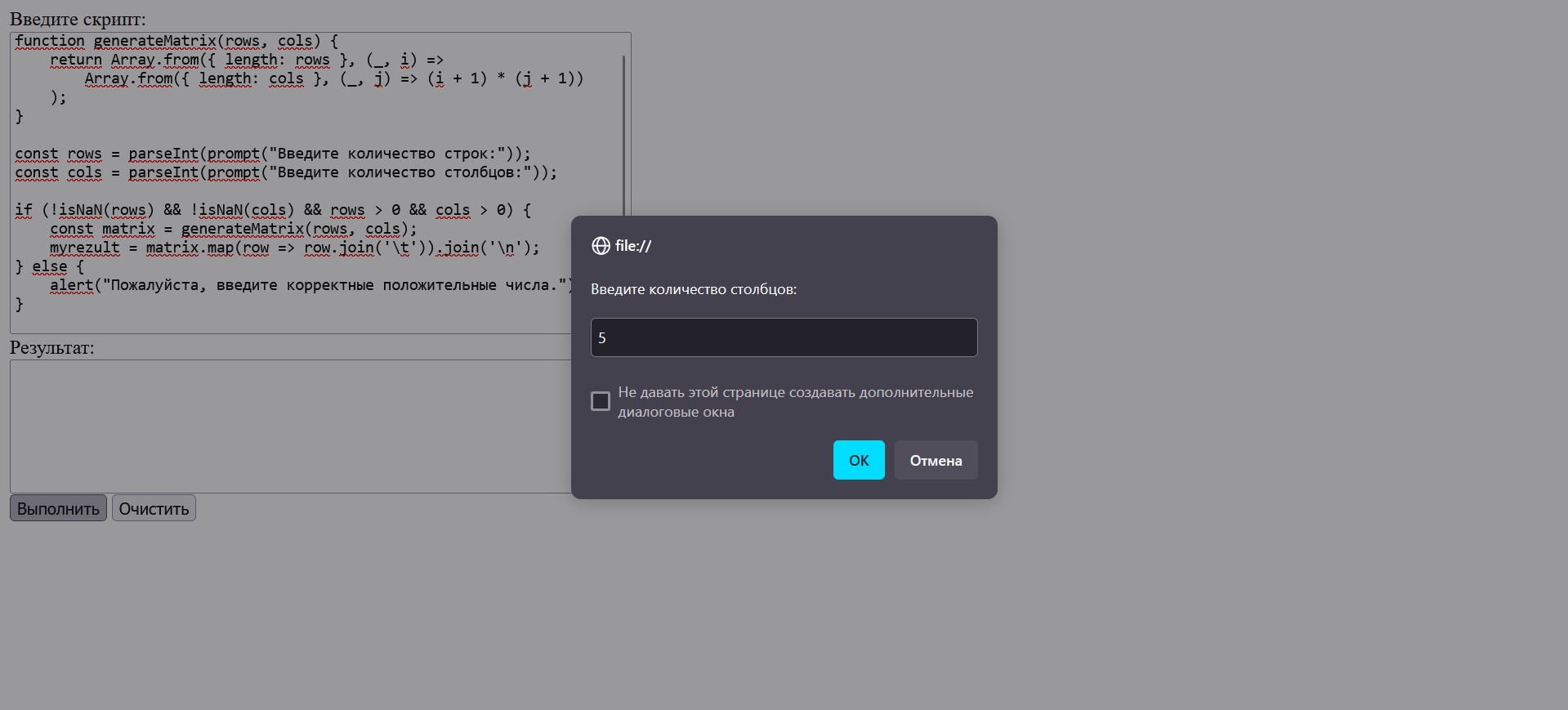


Рисунок 2 – запрос у пользователя количества столбцов

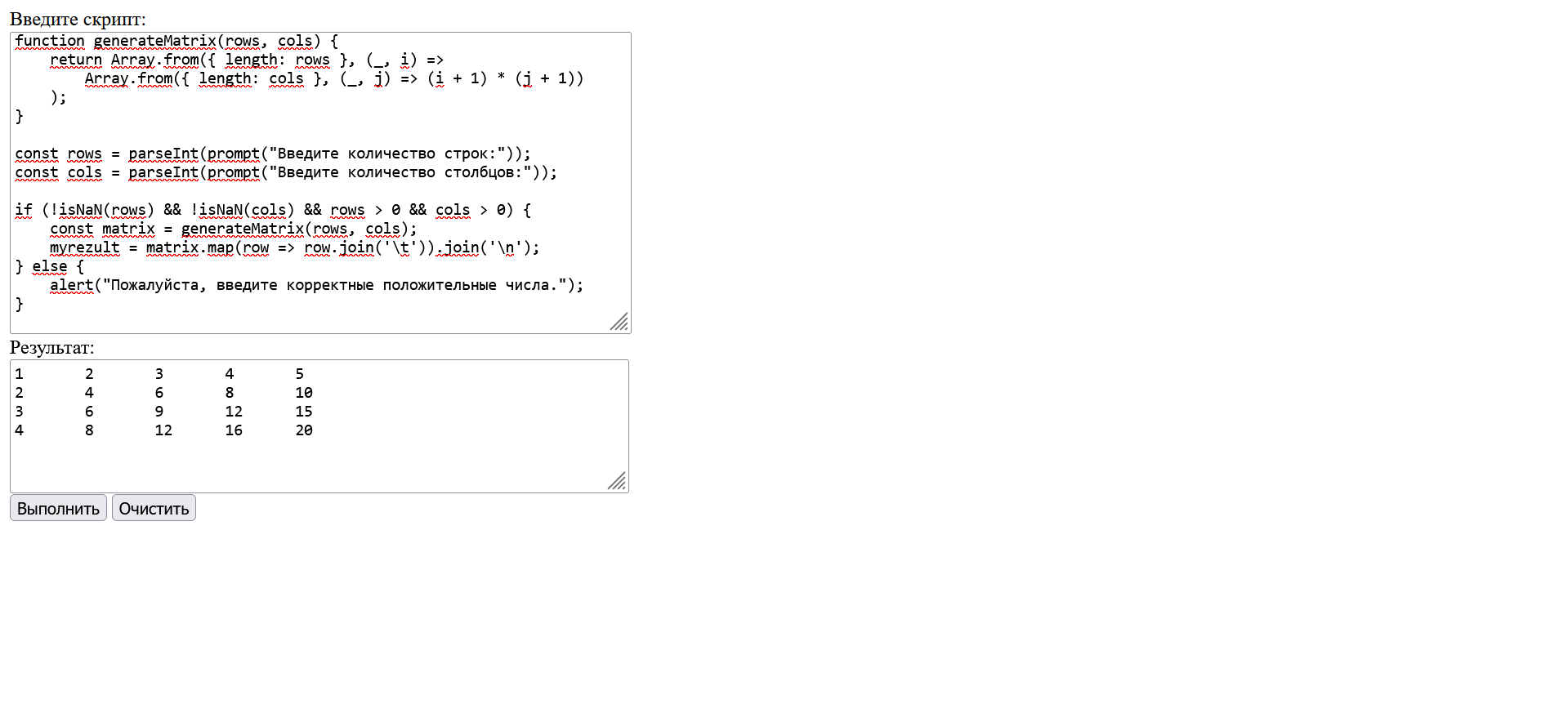


Рисунок 3 – вывод при успешном вводе размера

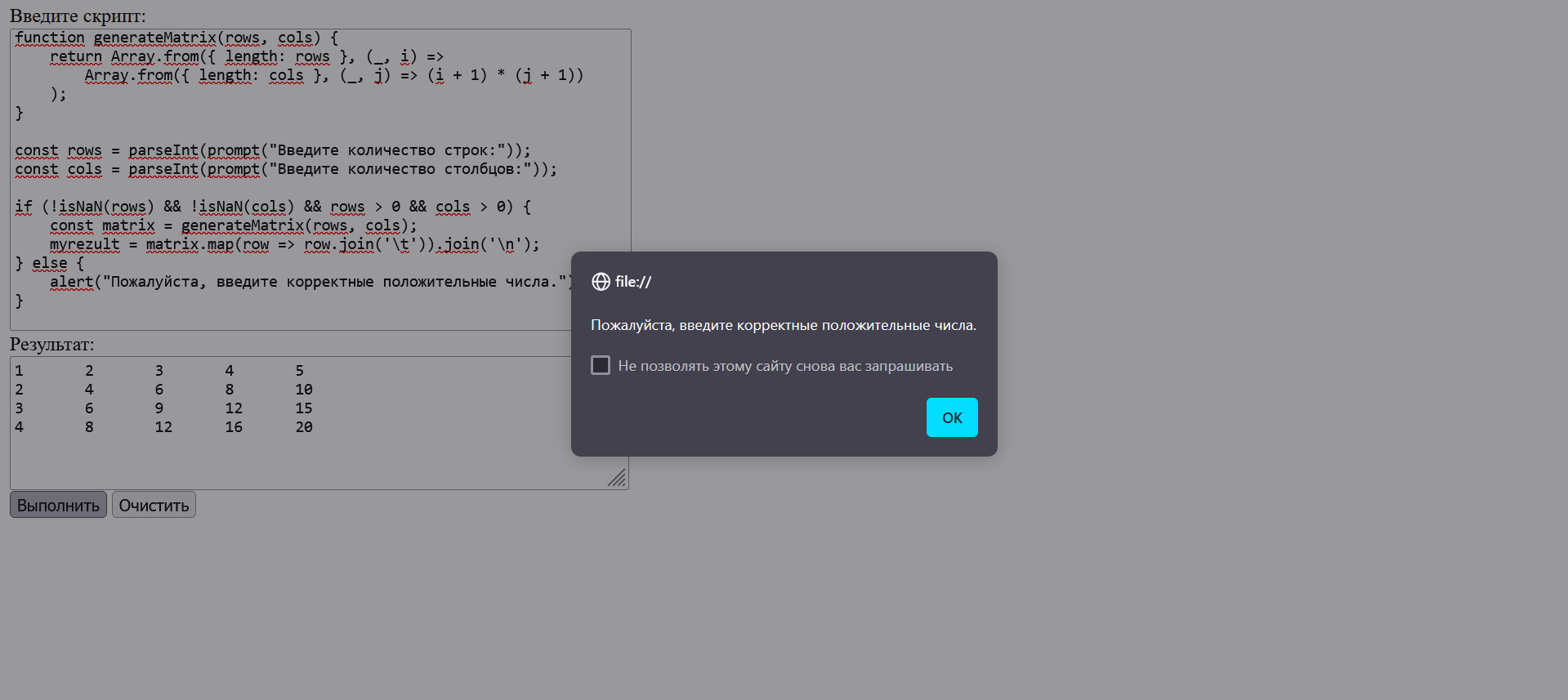


Рисунок 4 – вывод при не успешном вводе

На рисунке 5 представлен результат работы программы генерирующий рисунок с заданным в варианте паттерном.

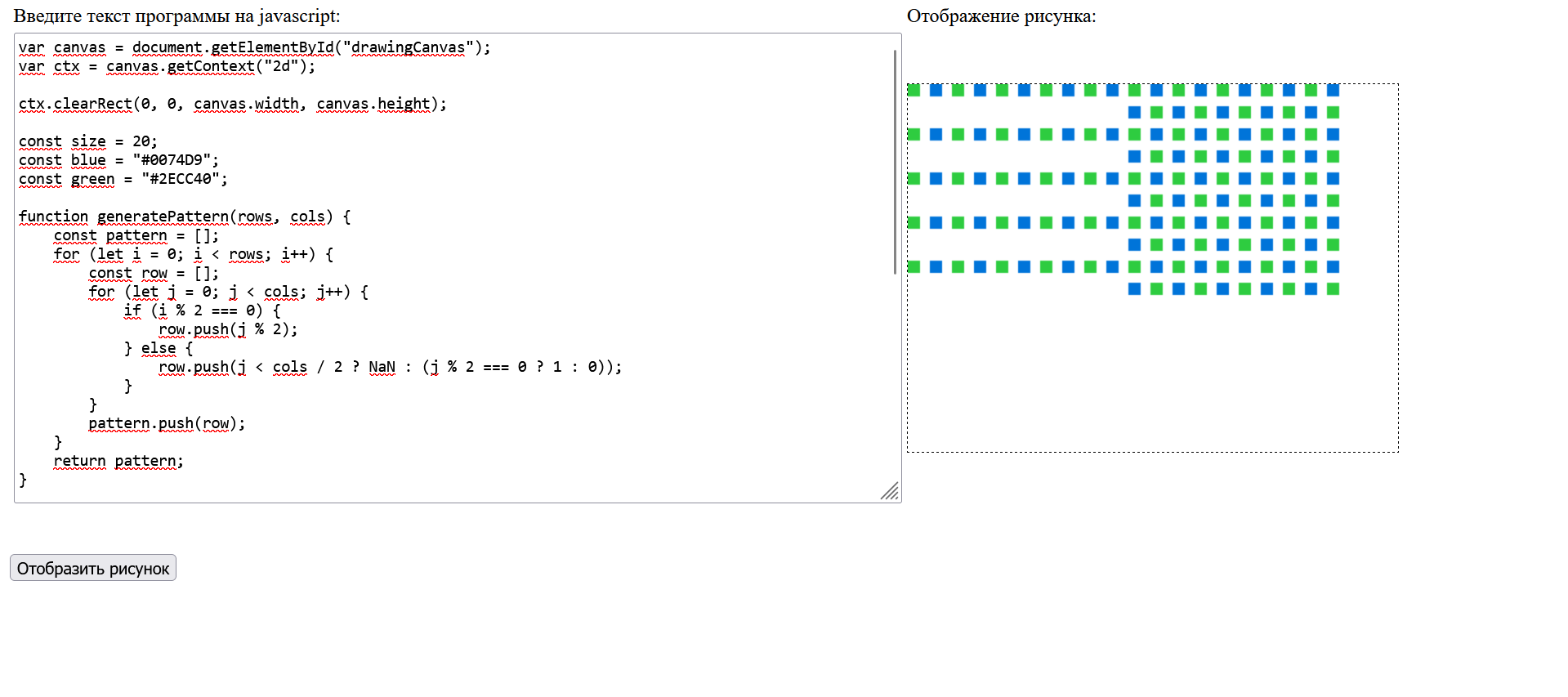


Рисунок 5 – вывод работы расширенного задания на страницу

На рисунках 6-7 показан запрос размеров матрицы для расширенного задания.

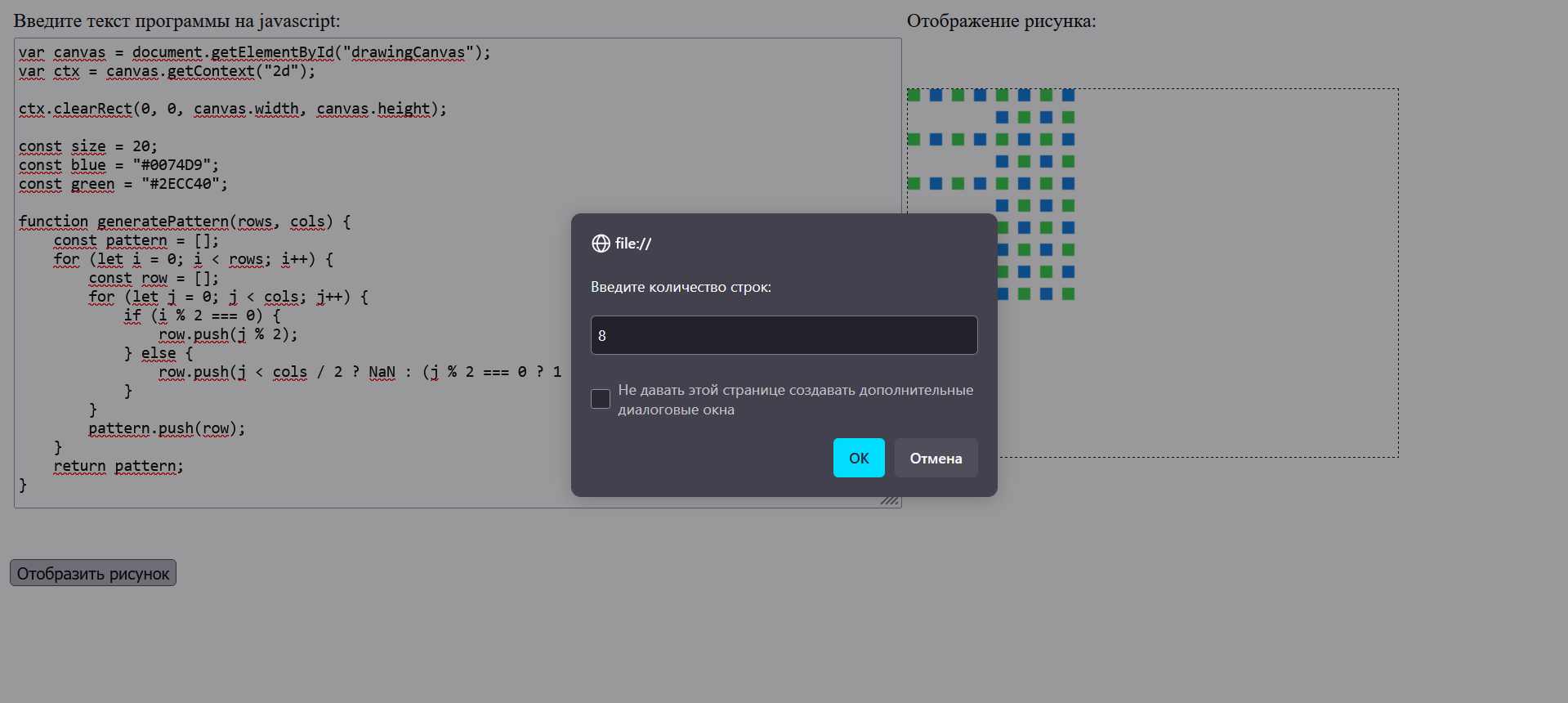


Рисунок 6 – запрос количества строк

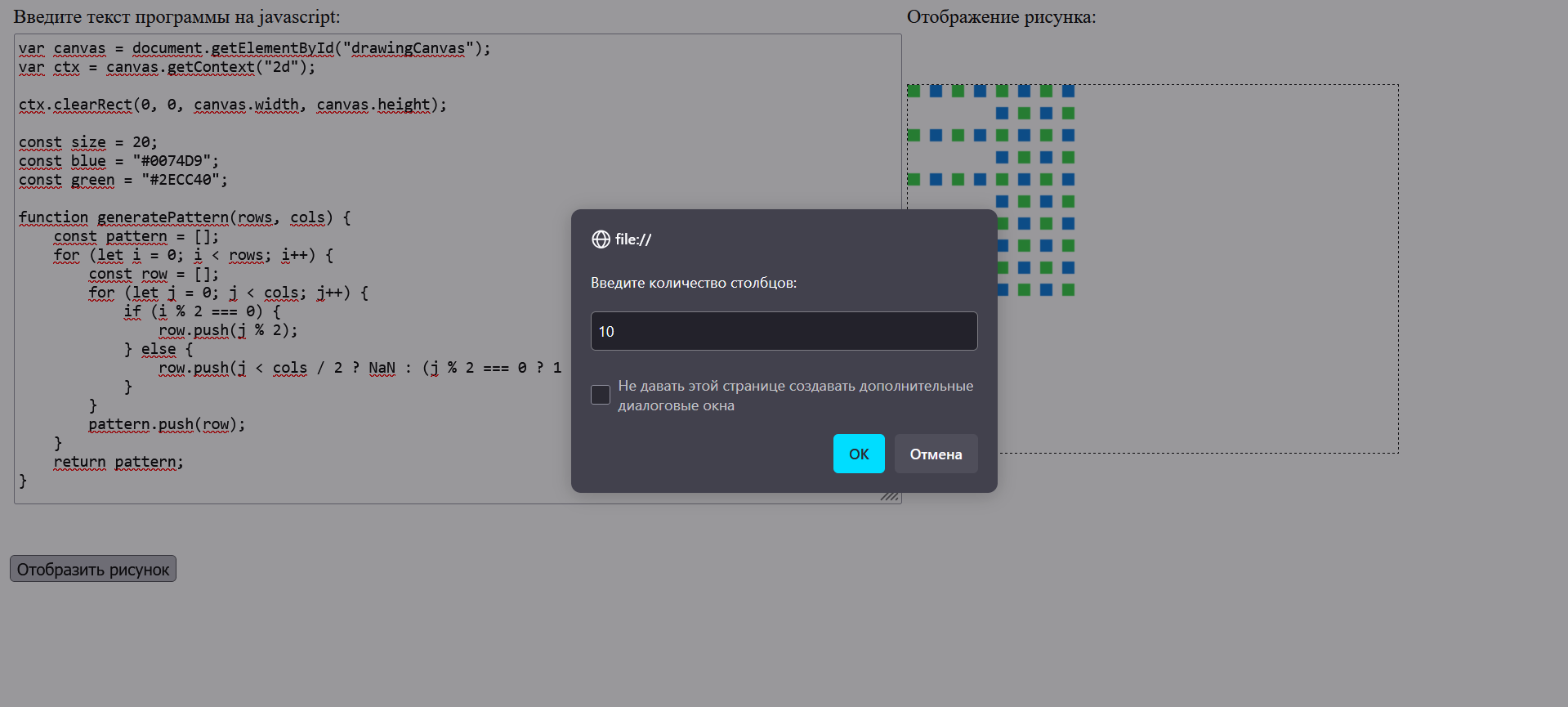


Рисунок 7 – запрос количества столбцов

# 6. Листинг программного кода

**Базовое задание:**

**function** generateMatrix(rows, cols) {

**return** Array.from({ length: rows }, (\_, i) =>

Array.from({ length: cols }, (\_, j) => (i + 1) \* (j + 1))

);

}

**const** rows = parseInt(prompt(*"Введите количество строк:"*));

**const** cols = parseInt(prompt(*"Введите количество столбцов:"*));

**if** (!isNaN(rows) && !isNaN(cols) && rows > 0 && cols > 0) {

**const** matrix = generateMatrix(rows, cols);

myrezult = matrix.map(row => row.join(*'\t'*)).join(*'\n'*);

} **else** {

alert(*"Пожалуйста, введите корректные положительные числа."*);

}

**Расширенное задание:**

**var** canvas = document.getElementById(*"drawingCanvas"*);

**var** ctx = canvas.getContext(*"2d"*);

ctx.clearRect(0, 0, canvas.width, canvas.height);

**const** size = 20;

**const** blue = *"#0074D9"*;

**const** green = *"#2ECC40"*;

**function** generatePattern(rows, cols) {

**const** pattern = [];

**for** (**let** i = 0; i < rows; i++) {

**const** row = [];

**for** (**let** j = 0; j < cols; j++) {

**if** (i % 2 === 0) {

row.push(j % 2);

} **else** {

row.push(j < cols / 2 ? **NaN** : (j % 2 === 0 ? 1 : 0));

}

}

pattern.push(row);

}

**return** pattern;

}

**function** drawPattern(rows, cols) {

ctx.clearRect(0, 0, canvas.width, canvas.height);

**const** pattern = generatePattern(rows, cols);

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

**for** (**let** j = 0; j < pattern[i].length; j++) {

**if** (!isNaN(pattern[i][j])) {

ctx.fillStyle = pattern[i][j] === 1 ? blue : green;

ctx.fillRect(j \* size \* 0.9, i \* size \* 0.9, size / 2, size / 2);

}

}

}

}

**const** rows = parseInt(prompt(*"Введите количество строк:"*));

**const** cols = parseInt(prompt(*"Введите количество столбцов:"*));

drawPattern(rows, cols);

# 7. Выводы

В ходе выполнения лабораторной работы была изучена работа с языком JavaScript, включая создание и обработку массивов, а также использование метода canvas для визуализации данных. Программа успешно генерирует матрицу, элементы которой равны произведению номера строки на номера столбца, и выводит ее на страницу. Для выполнения расширенного задания был использован объект canvas для рисования заданной фигуры, что позволило сформировать паттерн на основе данных от пользователя. В процессе работы были применены методы массива, такие как Array.from(), map(), и join(), а также стандартные методы для обработки введенных данных и работы с DOM.