КАФЕДРА

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ

должность, уч. степень, звание	подпись, дата	инициалы, фамилия

Отчет о лабораторной работе №3

Основы JavaScript

По дисциплине: Web-технологии

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № _____ _____ _____
подпись, дата инициалы, фамилия

Санкт-Петербург 2024

Содержание отчета:

Цель работы:
Знакомство с языком javascript.

Варианты заданий

Базовое задание:

Используя редактор javascript (см. листинг 1) написать программу, которая выполняет задание по варианту (базовая часть) и использует массив. Вывести на экран матрицу в форме прямоугольника. Предусмотреть возможность задавать произвольный размер матрицы через диалог с пользователем.

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

Расширенное задание:

Нарисовать заданную вариантом фигуру, используя объект canvas и образец скрипта из листинга 2. Повторяющие фрагменты формировать с помощью циклов. Предусмотреть возможность задавать произвольный размер фигуры через диалог с пользователем.

Вариант 12

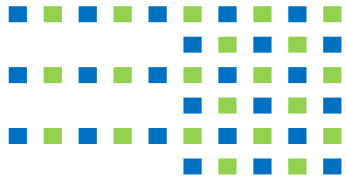

Таблица с описанием всех переменных программы

Базовое задание:

Имя переменной	**Тип**	**Назначение**
rows	**Number**	Хранит количество строк матрицы, вводимое пользователем.
cols	**Number**	Хранит количество столбцов матрицы, вводимое пользователем.
matrix	**Array**	Двумерный массив, представляющий матрицу, заполняется числами равными произведению индексов.
maxNumLengths	**Array**	Массив, хранящий максимальную длину числа в каждом столбце матрицы.
numLength	**Number**	Временная переменная для хранения длины текущего числа в матрице.
matrixString	**String**	Строка, представляющая матрицу для отображения в элементе HTML.
num	**Number**	Текущее число матрицы, используемое при формировании строки **matrixString**.
numString	**String**	Строковое представление текущего числа матрицы.
spacesToAdd	**Number**	Количество пробелов, необходимых для выравнивания текущего числа в столбце.
leftSpaces	**Number**	Количество пробелов, добавляемых слева от текущего числа.
rightSpaces	**Number**	Количество пробелов, добавляемых справа от текущего числа.

Расширенное задание:

Имя переменной	**Тип**	**Назначение**
canvas	Объект	Ссылка на элемент canvas в HTML, используется для рисования фигуры.
context	Объект	Контекст для рисования на холсте, предоставляет методы и свойства для работы с элементом canvas.
size	Number	Размер каждого квадрата, задается пользователем через диалоговое окно.
totalCols	Number	Общее количество столбцов квадратов, задается пользователем.
numRows	Number	Количество строк квадратов, задается пользователем.
padding	Number	Расстояние между квадратами на холсте.
topPadding	Number	Вертикальный отступ от верхнего края холста до начала рисования квадратов.
leftPadding	Number	Горизонтальный отступ от левого края холста до начала рисования квадратов.
i	Number	Переменная цикла, используется для итерации по строкам.
j	Number	Переменная цикла, используется для итерации по столбцам и определения необходимости пропуска столбцов.

Текст программы на javascript

Базовое задание:

Index1.html

```
// Запрашиваем у пользователя количество строк и столбцов матрицы
let rows = parseInt(prompt("Введите количество строк:", "3"));
let cols = parseInt(prompt("Введите количество столбцов:", "3"));

// Инициализируем матрицу - пустой массив
let matrix = [];

// Заполняем матрицу
for (let i = 0; i < rows; i++) {
    matrix[i] = [];
    for (let j = 0; j < cols; j++) {
        matrix[i][j] = (i + 1) * (j + 1);
    }
}

// Функция для нахождения максимальной длины числа в каждом столбце
function findMaxNumLengths(matrix) {
    let maxNumLengths = new Array(cols).fill(0);
    for (let i = 0; i < matrix.length; i++) {
        for (let j = 0; j < matrix[i].length; j++) {
            let numLength = String(matrix[i][j]).length;
            if (numLength > maxNumLengths[j]) {
                maxNumLengths[j] = numLength;
            }
        }
    }
    return maxNumLengths;
}

// Функция для преобразования матрицы в строку с учетом выравнивания
```

```
function matrixToString(matrix) {
    let matrixString = "";
    let maxNumLengths = findMaxNumLengths(matrix);

    // Выводим матрицу с выравниванием
    for (let i = 0; i < matrix.length; i++) {
        for (let j = 0; j < matrix[i].length; j++) {
            let num = matrix[i][j];
            let numString = String(num);
            let spacesToAdd = maxNumLengths[j] - numString.length;
            let leftSpaces = Math.floor(spacesToAdd / 2);
            let rightSpaces = spacesToAdd - leftSpaces;
            if (numString.length % 2 !== 0) {
                leftSpaces++;
            }
            matrixString += " ".repeat(leftSpaces) + numString + " ".repeat(rightSpaces + 1);
        }
        matrixString += "\n"; // Перенос строки для новой строки матрицы
    }
    return matrixString;
}

// Выводим матрицу в форме прямоугольника
document.getElementById('myresult').innerHTML = matrixToString(matrix);
```

Расширенное задание:

```
var canvas = document.getElementById("drawingCanvas");
var context = canvas.getContext("2d");
context.clearRect(0, 0, canvas.width, canvas.height); // Очистка холста

var size = parseInt(prompt("Введите размер квадратов:", "20")); // Размер квадрата
var totalCols = parseInt(prompt("Введите общее количество столбцов:", "10")); // Общее количество столбцов
var numRows = parseInt(prompt("Введите количество строк:", "6")); // Количество строк
var padding = 5; // Отступ между квадратами
var topPadding = 20; // Отступ сверху
var leftPadding = 20; // Отступ слева

for (var i = 0; i < numRows; i++) {
    for (var j = 0; j < totalCols; j++) {
        // Для четных строк добавляем пропуски после каждых 5 квадратов
        if (i % 2 !== 0 && (j % 10 < 5)) {
            continue; // Пропускаем первые 5 столбцов в четных строках
        }

        context.fillStyle = ((i + j) % 2 === 0) ? "blue" : "green"; // Чередование цветов
        // Рисуем квадрат с учетом отступов
        context.fillRect(
            leftPadding + j * (size + padding),
            topPadding + i * (size + padding),
            size,
            size
        );
    }
}
```

Скриншоты web-страницы с результатами работы программы при разных данных

Базовое задание:

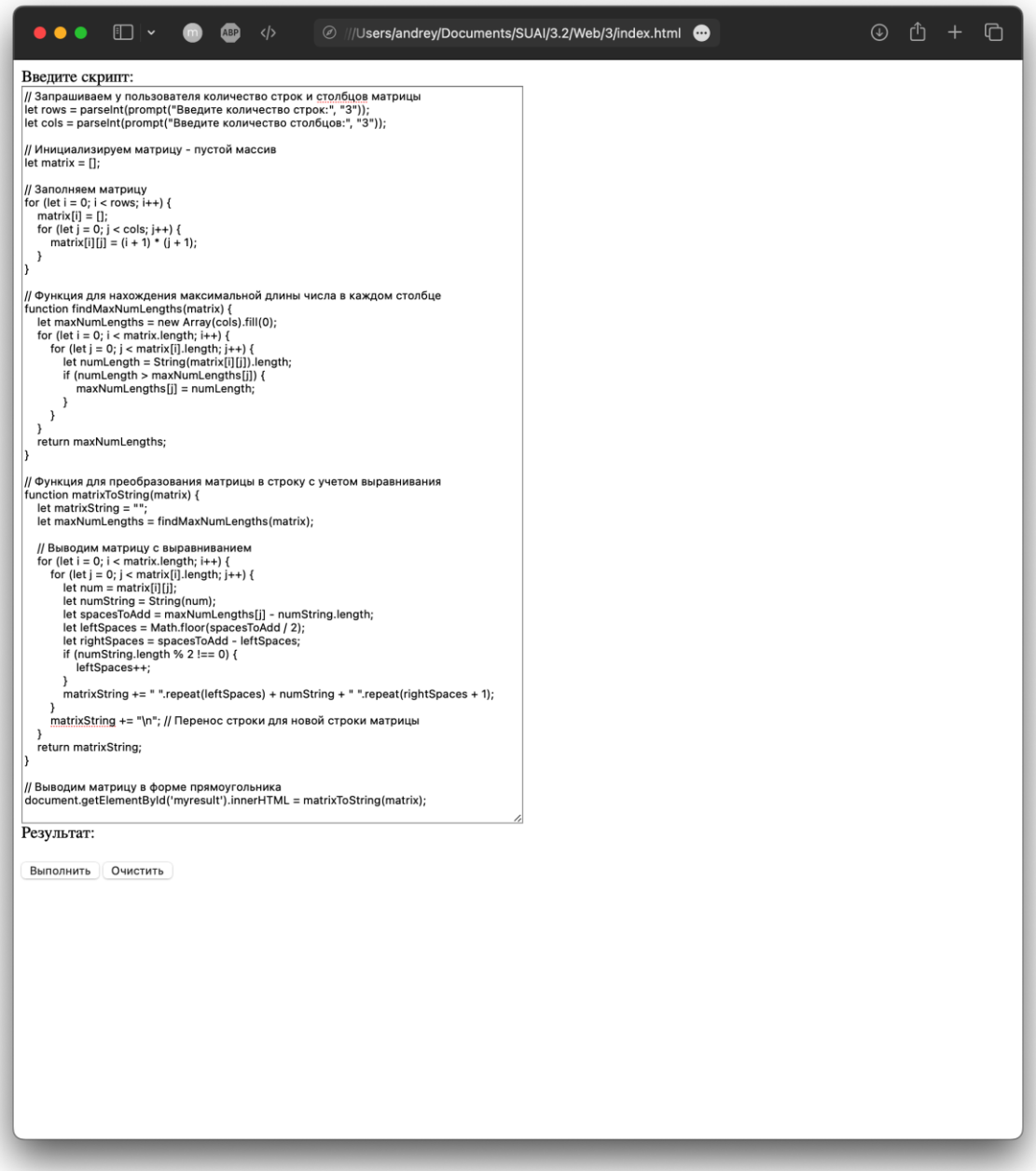

Рисунок 1 – общий вид страницы

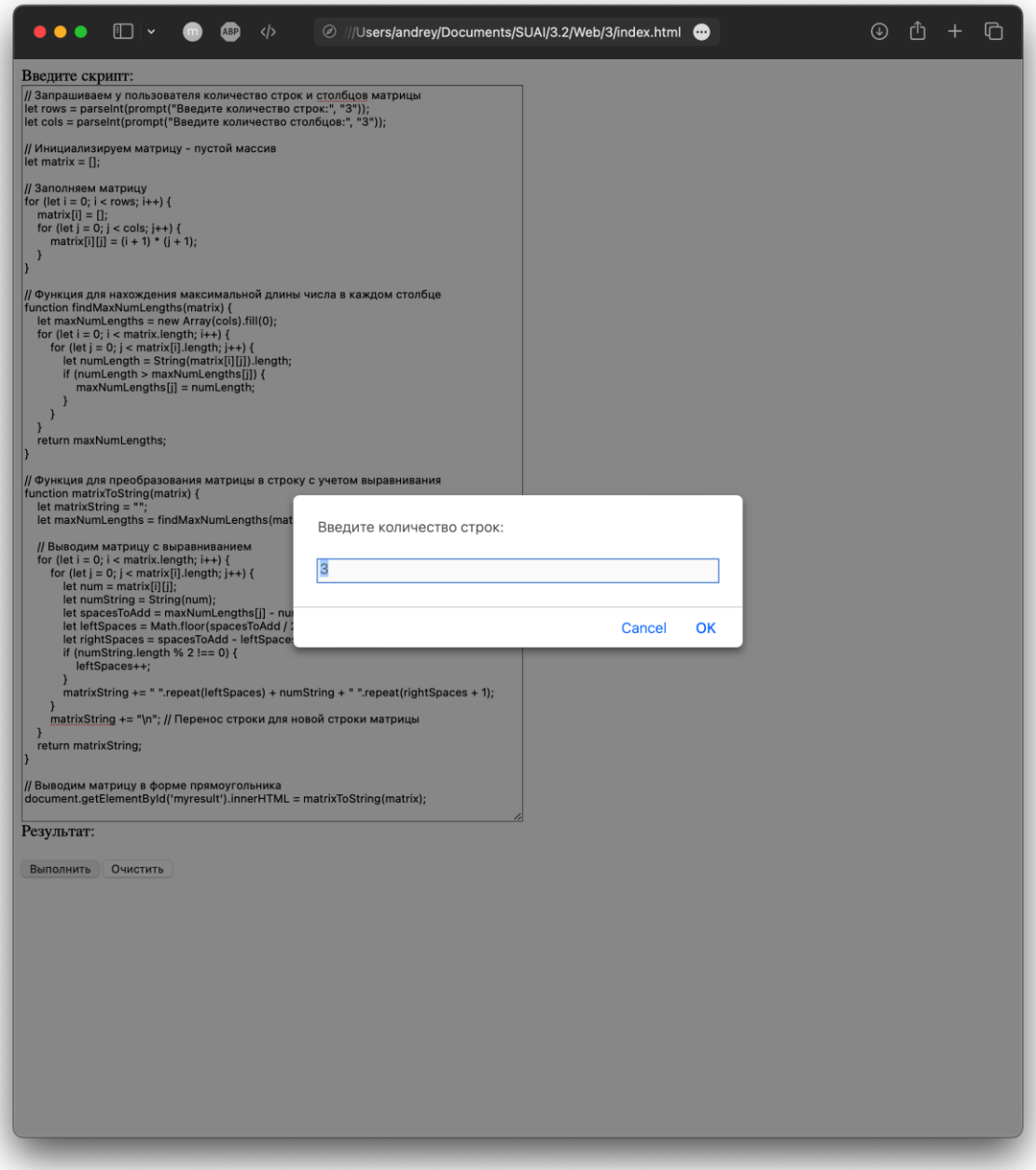

Рисунок 2 – запрос у пользователя количества строк матрицы

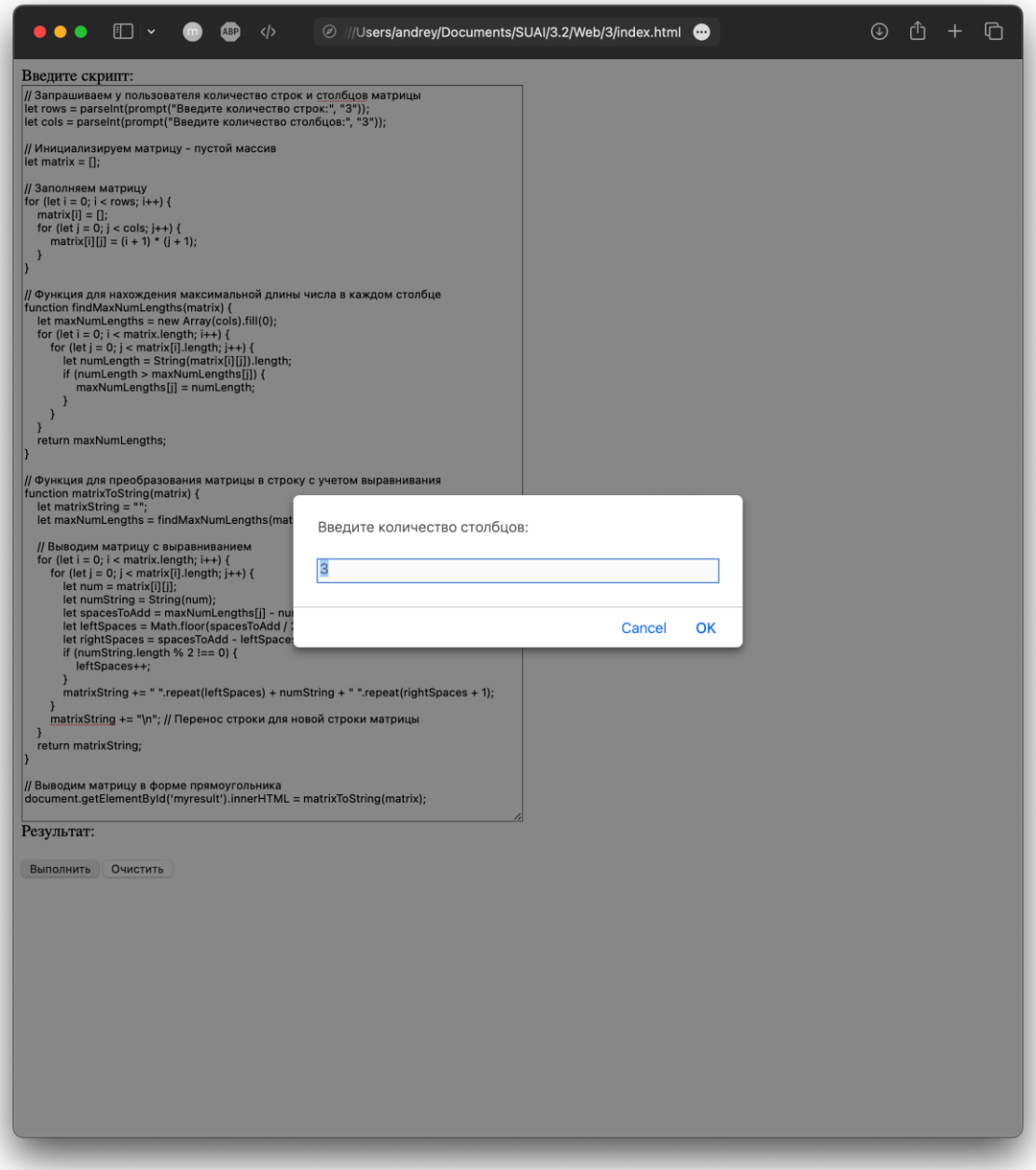

Рисунок 3 – запрос у пользователя количества столбцов матрицы

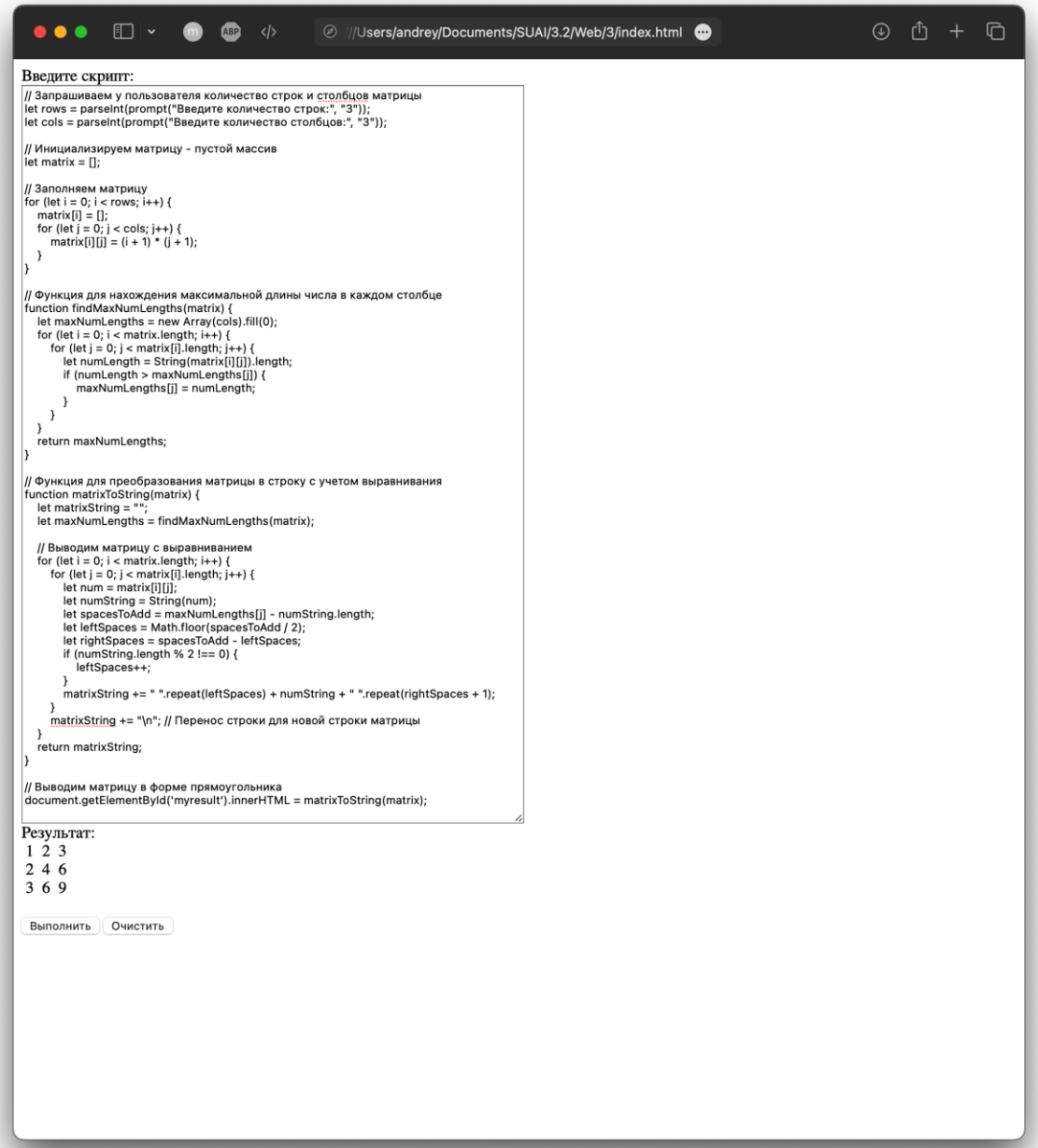

Рисунок 4 – пример матрицы 3x3, выводится ровный прямоугольник

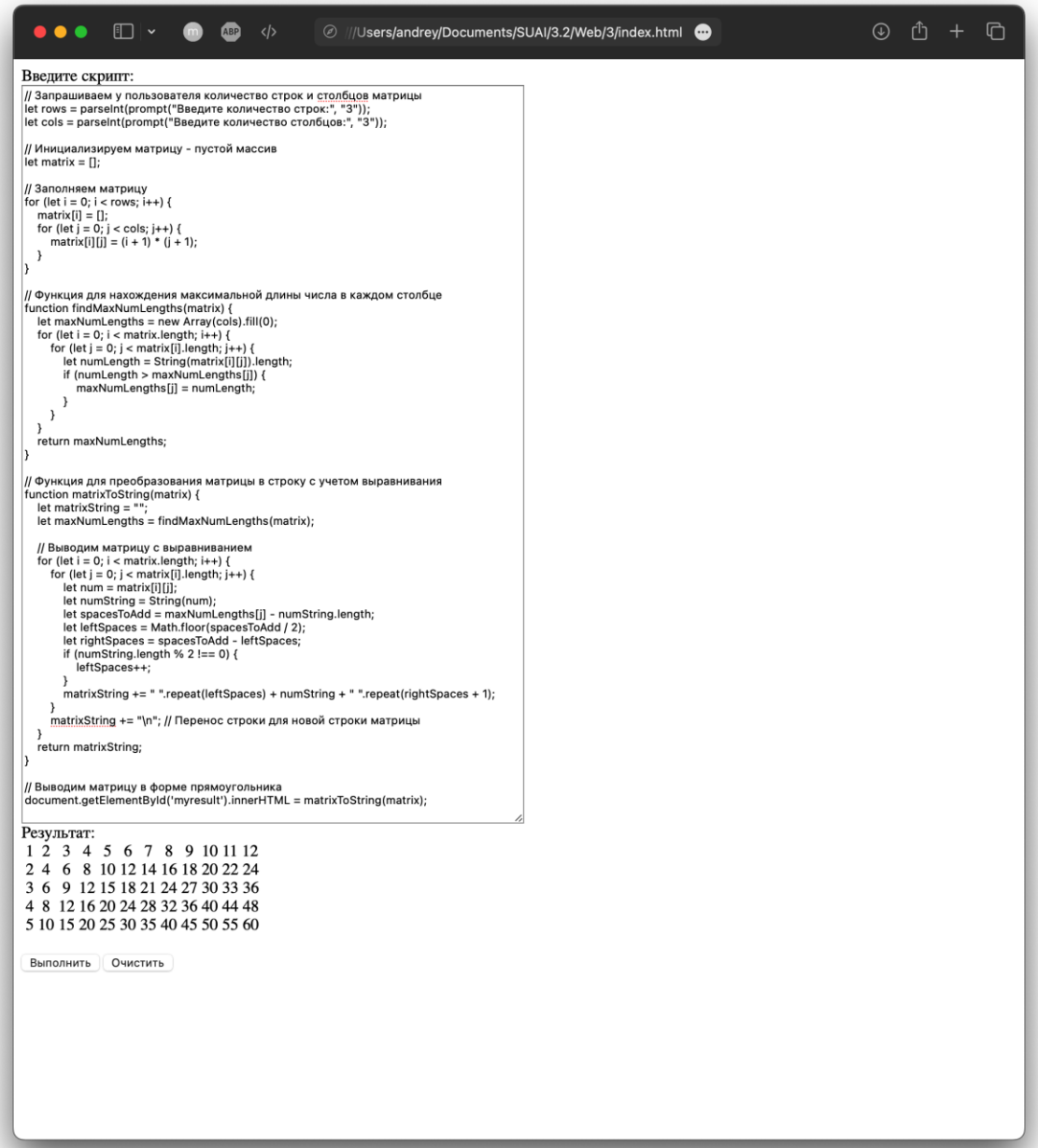

Рисунок 5 – пример матрицы 5x12, выводится ровный прямоугольник

Расширенное задание:

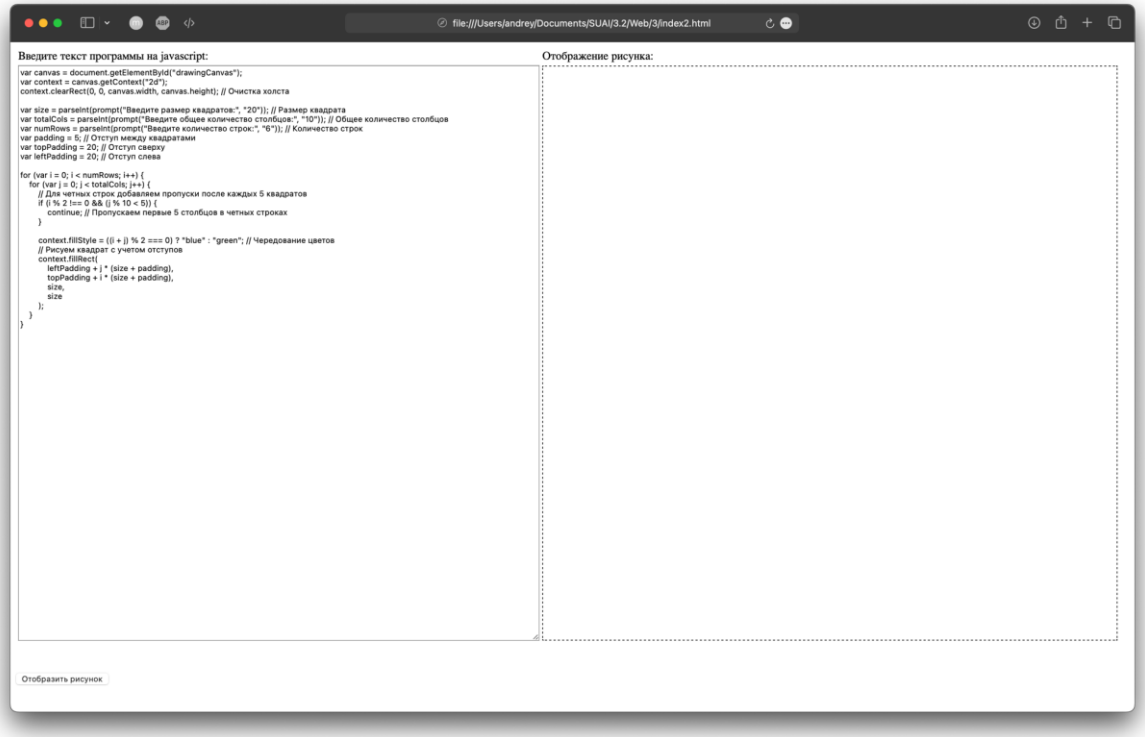

Рисунок 6 – общий вид страницы, увеличено окно для отображения фигуры

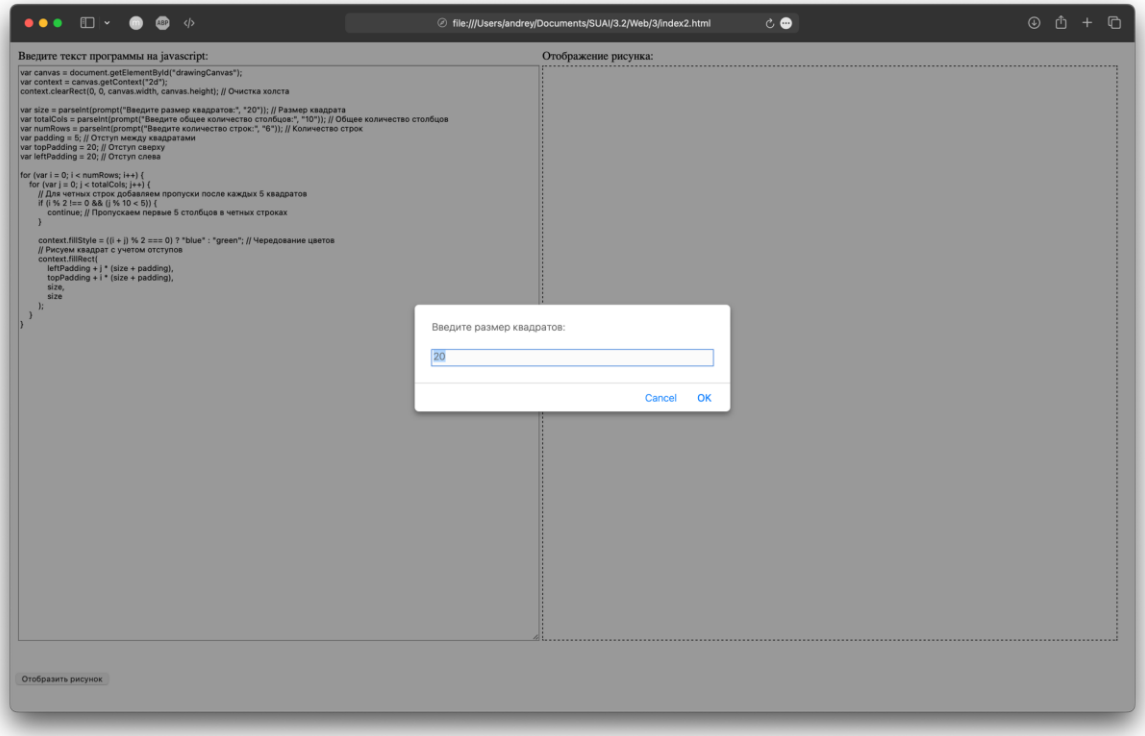

Рисунок 7 – запрос у пользователя размера квадрата

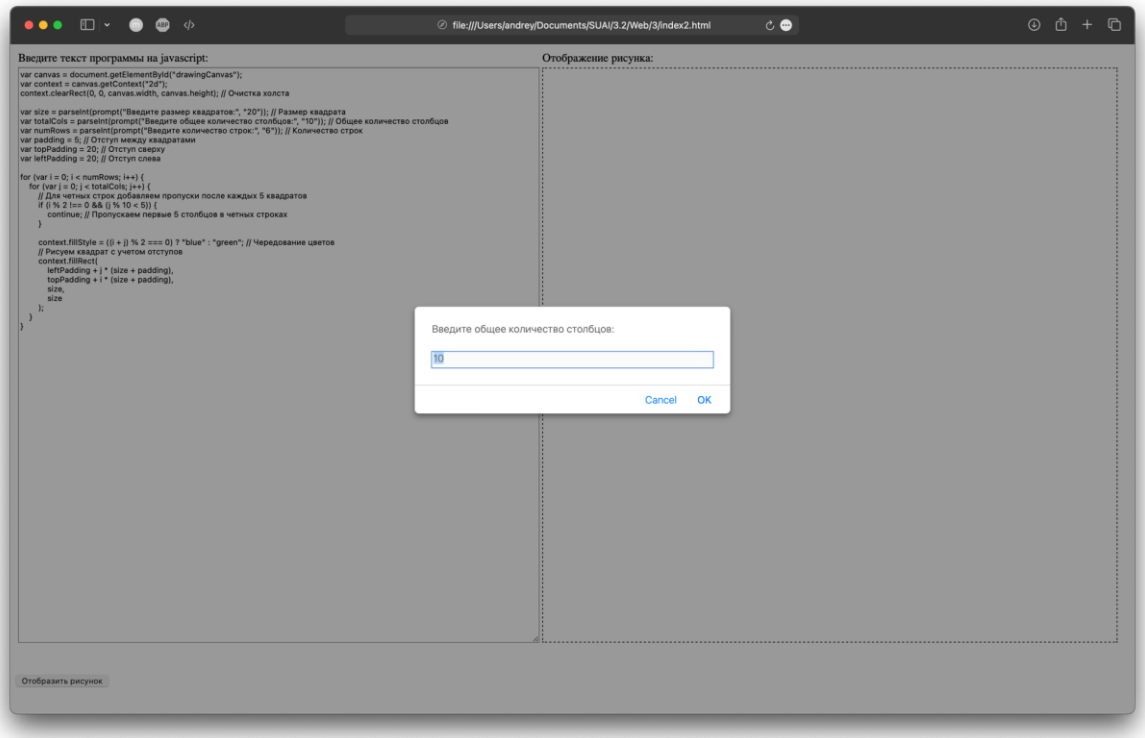

Рисунок 8 – запрос у пользователя количества столбцов

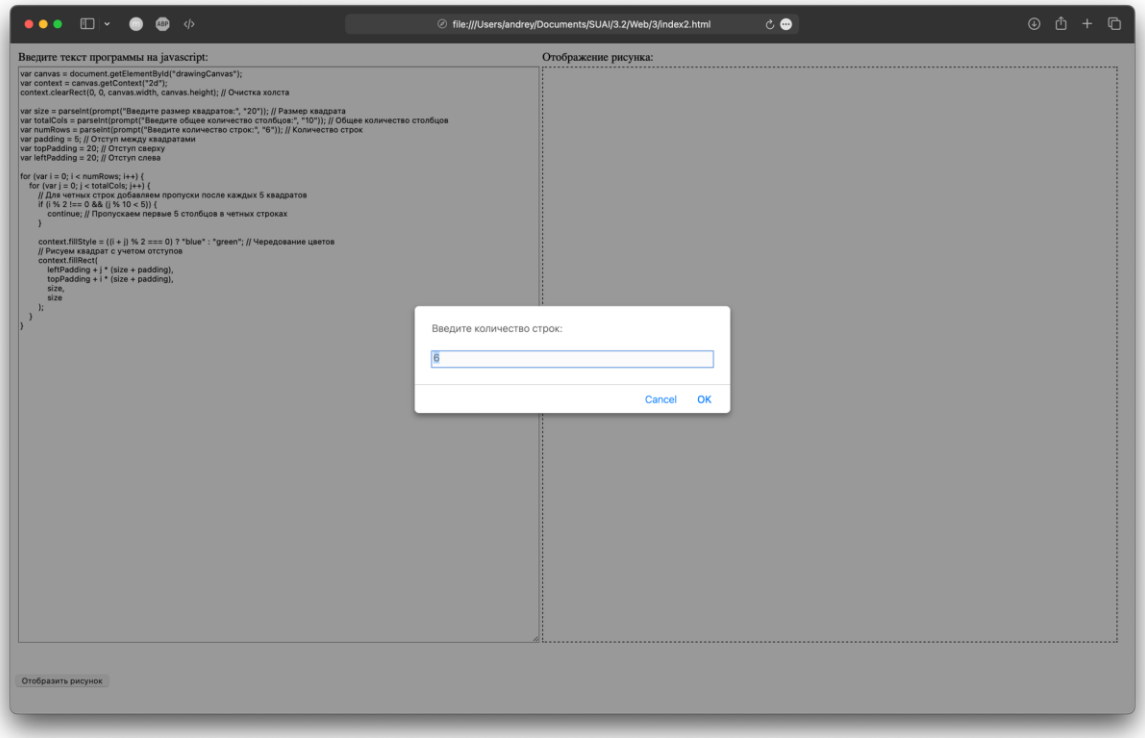

Рисунок 9 – запрос у пользователя количества строк

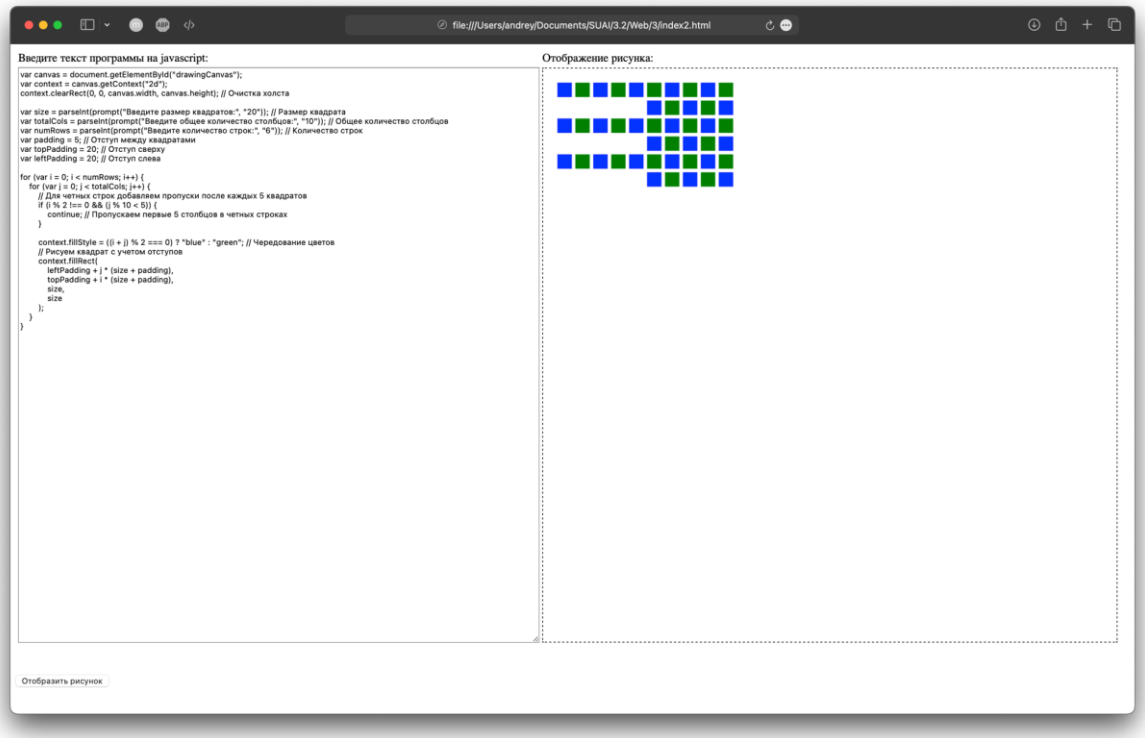

Рисунок 10 – по умолчанию получаем результат как во варианте

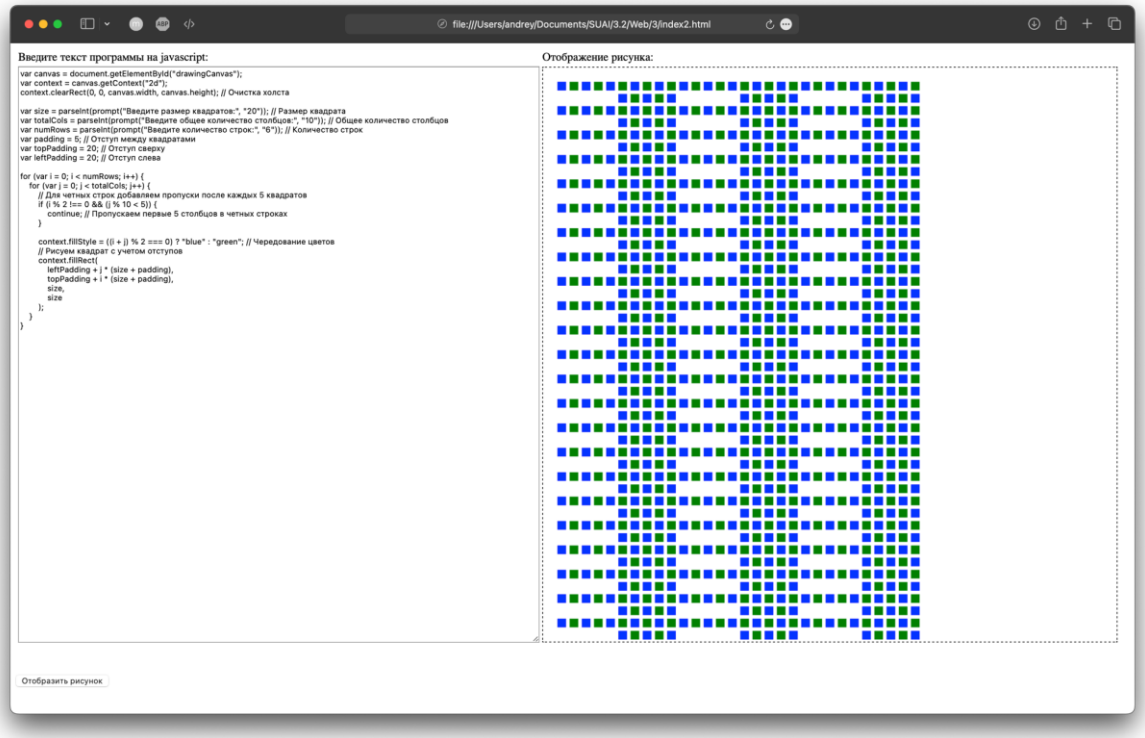

Рисунок 11 – введены значения 8x30x61

Анализ программы и результатов ее работы

Базовое задание:

Решение задачи:

1. **Инициализация переменных**: Созданы переменные для хранения количества строк (**rows**) и столбцов (**cols**) матрицы, введенных пользователем. Использован двумерный массив (**matrix**) для хранения значений матрицы.
2. **Заполнение матрицы**: Двумерный массив заполняется в двух вложенных циклах, где каждый элемент равен произведению индексов строки и столбца, увеличенных на единицу (так как индексация в JavaScript начинается с нуля).
3. **Выравнивание и вывод матрицы**: Реализована функция **findMaxNumLengths** для нахождения максимальной длины числа в каждом столбце, и функция **matrixToString** для преобразования матрицы в строку с учетом выравнивания. Матрица отображается в виде прямоугольника.

Результаты работы: Программа успешно генерирует матрицу с заданными параметрами и корректно выводит ее на экран, обеспечивая четкое выравнивание чисел в столбцах.

Расширенное задание:

Решение задачи:

1. **Работа с canvas**: Использование элемента **canvas** для рисования. Пользовательская настройка размера квадратов, общего количества столбцов и строк.
2. **Рисование квадратов**: Квадраты рисуются в двух вложенных циклах, где каждый квадрат имеет фиксированный размер и цвет, изменяющийся по заданному правилу. Чередование синего и

зеленого цветов с пропуском первых пяти квадратов в каждой второй строке.

3. **Учет отступов**: Программа учитывает отступы как между квадратами, так и отступы от краев холста.

Результаты работы: Программа успешно создает узор из квадратов на **canvas**, соответствуя заданным параметрам. Цвета квадратов чередуются в соответствии с заданным шаблоном, создавая ожидаемую визуализацию фигуры.