**МИНОБРНАУКИ РОССИИ**

**Федеральное государственное бюджетное**

**образовательное учреждение высшего образования**

**«Тульский государственный университет»**

**Кафедра информационной безопасности**

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовой работе по дисциплине

**«Методы и технологии программирования»**

на тему

Web-приложение «Игра Судоку»

Автор работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ студент гр. 221111 Петухов К. А.

(дата, подпись)

Руководитель работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ доцент каф. ИБ Басалова Г. В.

(дата, подпись)

Работа защищена \_\_\_\_\_\_\_\_\_\_\_\_\_ с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата)

Члены комиссии \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата, подпись) (должность) (фамилия и инициалы)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата, подпись) (должность) (фамилия и инициалы)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата, подпись) (должность) (фамилия и инициалы)

Тула 2023

Оглавление

[Введение 5](#_Toc155887030)

[1. Постановка задачи 6](#_Toc155887031)

[2. Анализ и сравнение аналогов 7](#_Toc155887032)

[2.1 Sudoku.com 7](#_Toc155887033)

[2.2 Sudoku-online.com 8](#_Toc155887034)

[2.3 Сравнение 9](#_Toc155887035)

[3. Обзор и выбор технологий и средств разработки 10](#_Toc155887036)

[4. Разработка программного комплекса 11](#_Toc155887037)

[4.1 Описание структуры программы 11](#_Toc155887038)

[4.2 Разработка базы данных 13](#_Toc155887039)

[4.3 Проектирование логики обработки данных 14](#_Toc155887040)

[4.4 Разработка класса 16](#_Toc155887041)

[4.5 Разработка основных алгоритмов класса 17](#_Toc155887042)

[4.5.1 Заполнение сетки Судоку 17](#_Toc155887043)

[4.6 Разработка основных алгоритмов игрового поля 20](#_Toc155887044)

[4.6.1 События нажатий по полю 20](#_Toc155887045)

[4.7 Программная реализация взаимодействия с клиентом 23](#_Toc155887046)

[5. Инструкция по эксплуатации 27](#_Toc155887047)

[Заключение 28](#_Toc155887048)

[Список литературы 29](#_Toc155887049)

[ПРИЛОЖЕНИЕ 1. html-разметка 30](#_Toc155887050)

[ПРИЛОЖЕНИЕ 2. SCSS 35](#_Toc155887051)

[ПРИЛОЖЕНИЕ 3. класс Sudoku 39](#_Toc155887052)

[ПРИЛОЖЕНИЕ 4. OnStart.js 41](#_Toc155887053)

[ПРИЛОЖЕНИЕ 5. java.js 44](#_Toc155887054)

# **Введение**

В современном мире игры на устройствах стали неотъемлемой частью повседневной жизни. Однако они не просто развлечение; игры также служат мощным инструментом для развития умственной деятельности и логического мышления.

В частности, головоломки, такие как Судоку, предоставляют уникальную практику, способствуя развитию критического мышления и снижая риск некоторых заболеваний, таких как болезнь Альцгеймера.

В данной работе рассмотрено веб-приложение, предлагающее интересную и увлекательную версию игры Судоку, сочетая в себе развлечение и пользу для умственного здоровья.

# **1. Постановка задачи**

Необходимо разработать web-приложение «Игра Судоку».

Функции приложения:

Рейтинговая система – в web-приложении будет доступен просмотр рейтинга других пользователей и сохранение собственного результата.

Информационная справка – каждому пользователю будет доступна базовая информация по правилам игры.

Подсказки игрокам – пользователи будут иметь возможность лёгкого просмотра выбранных чисел и подсветку некорректных введённых данных в сетку Судоку.

Доступный ввод – приложение будет иметь возможность вводить значения в ячейки при помощи клавиатуры и интерактивного меню ввода сайта.

Уровни сложности – приложение будет иметь разные уровни сложности для опытных игроков и новичков.

# **2. Анализ и сравнение аналогов**

## **2.1 Sudoku.com**

Фрагмент сайта на рисунке 2.1

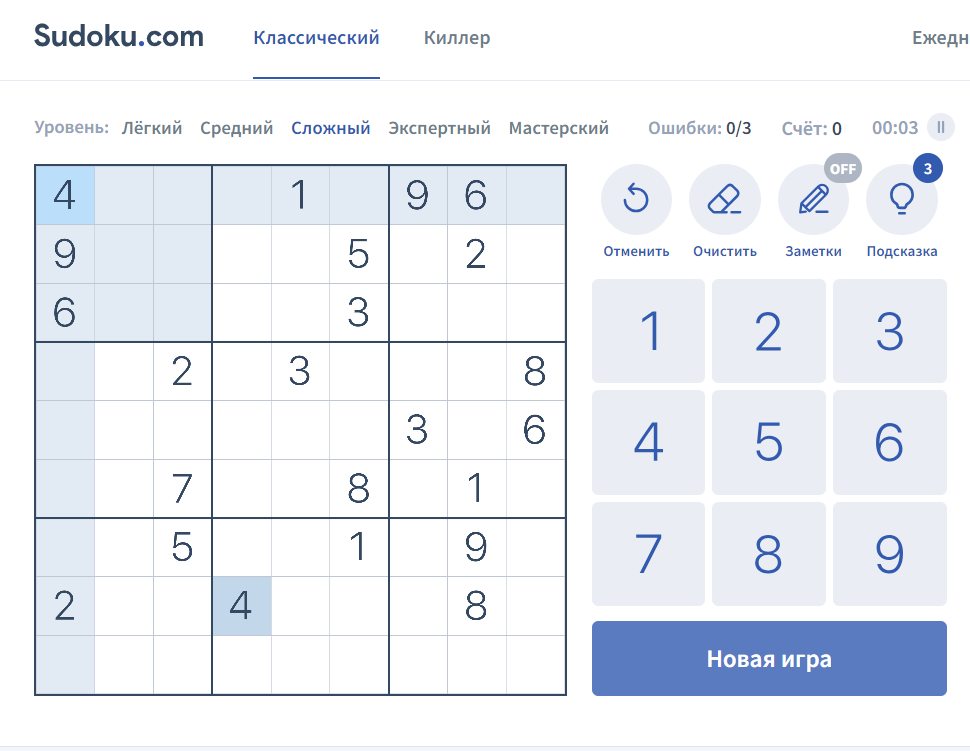


Рисунок 2.1 Web-сайт Sudoku.com

Приложение имеет:

Хорошее графическое исполнение

Счёт пользователя

Выбор сложности

Отмена предыдущего действия

Подсказка и подсветка выделенного числа

## **2.2 Sudoku-online.com**

Фрагмент сайта на рисунке 2.2

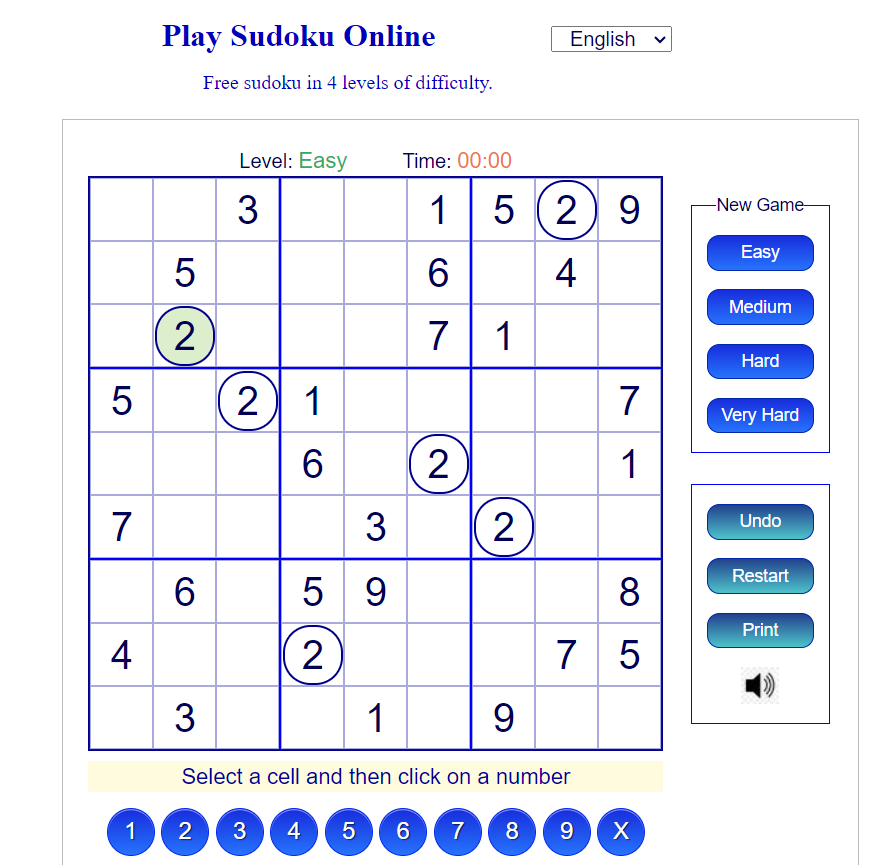


Рисунок 2.2 Web-сайт Sudoku-online.com

Приложение имеет:

Плохой графический визуал

Нет подсветки строки и столбца

Нет рейтинга пользователя

Не работает звуковое сопровождение

Есть возможность распечатать таблицу Судоку

Есть возможность возвратить предыдущее действие

Есть выбор сложности

## **2.3 Сравнение**

Сравнение приложений представлено в таблице 2.1.

Таблица 2.1 Сравнение приложений

|  |  |  |  |
| --- | --- | --- | --- |
|  | Мое | Sudoku.com | Sudoku-online.com |
| Статистика | + | + | - |
| Подсветка чисел | + | + | - |
| Отображение рейтинга | + | - | - |
| Выбор сложности | + | + | + |
| Реклама | - | + | - |
| Распечатывание таблицы | - | - | + |
| Ежедневные задания | - | + | - |

Исходя из таблицы, можно понять, что каждое приложение имеет свои особенности и недостатки, и выбор пользователя может зависеть от разных потребностей.

Приложение, получившее финансовую поддержку с рекламы или иных спонсоров будет иметь более сильную конкурентоспособность по сравнению с безвозмездными аналогами.

Однако разрабатываемое в этом проекте приложение уже имеет неплохой потенциал при доработке и финансировании вытеснить некоторые аналоги и подходит для пользователей, которым не нужны излишне перегруженные сайты.

# **3. Обзор и выбор технологий и средств разработки**

При выборе технологий для разработки web-приложения следует опираться на их доступность, цели проекта, опыт использования данных средств, популярность языка программирования, безопасность, совместимость и эффективность.

Приведу наиболее популярные языки программирования для web-приложений: JavaScript, PHP, C#, Python, TypeScript.

Фреймворки для разработки: React (иммеет мощный пользовательских интерфейсов для разработки веб-приложений, что делает ее одним из лучших языков интерфейса, популярность которого в наши дни стремительно растет), Angular( Поддерживаемый огромным сообществом разработчиков по всему миру и Google, AngularJS является частью экосистемы Javascript. Как фреймворк с открытым исходным кодом, это один из интерфейсных языков для веб-разработки, который стоит использовать, если разработчик заинтересован в аспекте дизайна веб-приложения), Vue (Vue входит в топ-10 языков веб-дизайна и подходит для веб-интерфейсов и одностраничных приложений.), Bootstrap (один из наиболее часто используемых интерфейсных языков веб-разработки, представляет собой фреймворк HTML, CSS и JS для разработки адаптивных веб-приложений).

Технологии, выбранные для разработки данного проекта – JavaScript, PHP и и препроцессор SCSS, который предоставляет дополнительные возможности css.

JavaScript – Выбор обоснован тем, что данный язык наиболее популярен и имеет множество методов для взаимодействия с HTML. Реализует класс и его взаимодействие с html-разметкой.

PHP – Выбран для обработки запросов клиента и сервера. Реализует взаимодействие web-приложения и базы данных.

SCSS – упрощает работу с декорированием документа.

В качестве среды разработки был выбран Visual Studio Code, так как имеет следующие преимущества: кроссплатформенность, встроенный IntelliSense, большое количество бесплатных расширений, полностью бесплатен.

# **4. Разработка программного комплекса**

## **4.1 Описание структуры программы**

Web-приложение состоит из html-файла, в котором находится структура сайта. Html-код представлен в приложении 1. В виде указателей для скриптов (JavaScript, PHP) выступают определённые элементы, имеющие идентификаторы, как представлено на иллюстрации ниже (рис. 4.1).

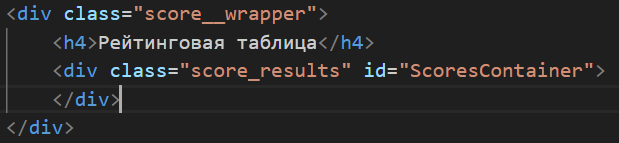


Рисунок 4.1 Идентификатор

Для декорирования внешнего вида приложения используется препроцессор SCSS (см. приложение 2).

Для обработки событий web-приложения используется файл java.js, который реализует события кнопок, выбор сложности, подсвечивание игрового поля, запросы к серверным скриптам req\_connection.php и display\_score.php. Также подключается класс Sudoku из файла sudoku.js который необходим для формирования и заполнения сетки судоку (см. приложение 3).

Для взаимодействия с сервером, а именно получение и отправление информации в базу данных используются скрипты display\_score.php и req\_connection.php соответственно.

Все результаты игры хранятся в базе данных, которая использует популярную реляционную систему управления MySQL.

Файлы проекта и структура проекта представлены на рисунке 4.2 и рисунке 4.3 соответственно.

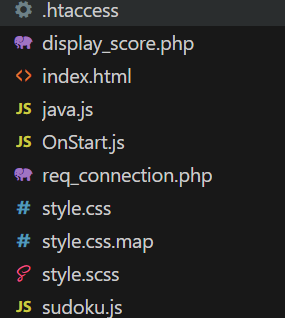


Рисунок 4.2 Файлы проекта

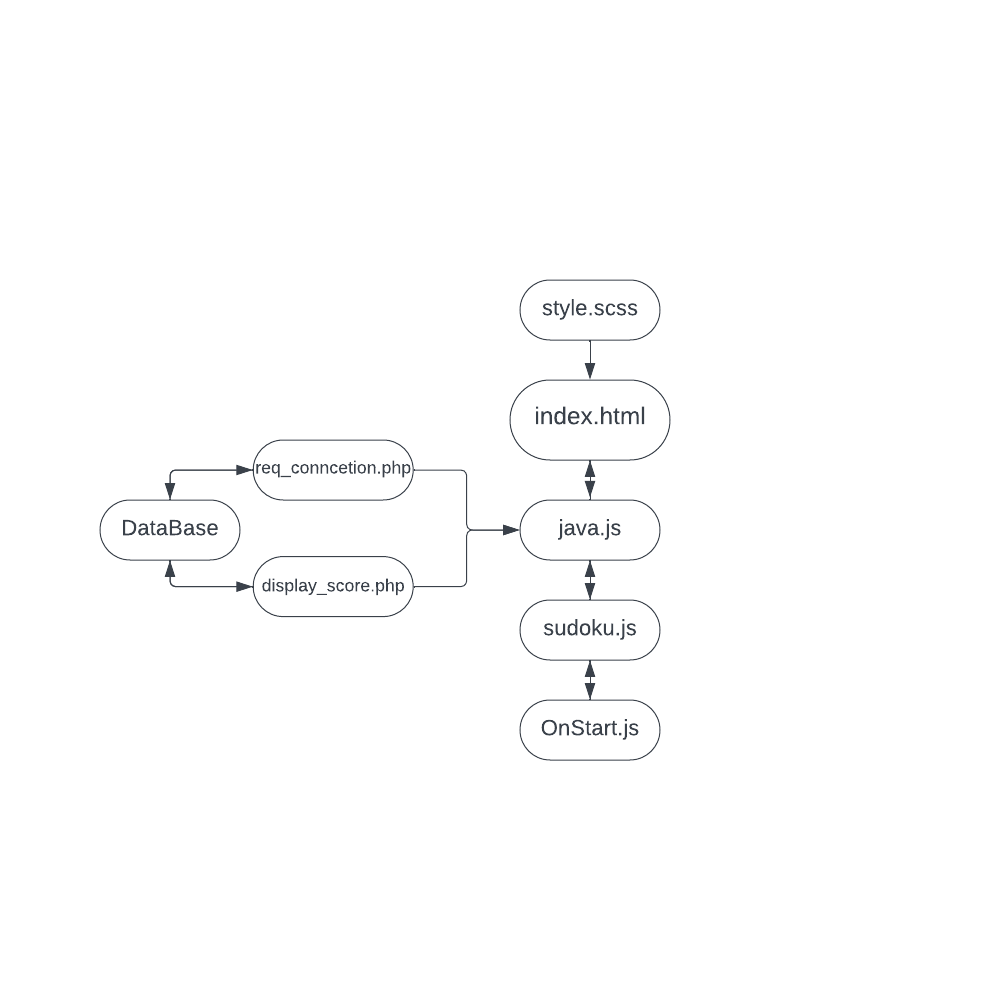


Рисунок 4.3 Схема проекта

## **4.2 Разработка базы данных**

База данных под системой управления MySQL содержит одну таблицу для сохранения результатов пользователей. Таблица содержит три поля: id – первичный ключ, name – имя игрока и score – счёт игрока. Name и id являются уникальными полями для предотвращения одинаковых записей в рейтинговой таблице.

Код создания базы данных:

CREATE TABLE `Players\_score` (

`id` int UNSIGNED NOT NULL,

`name` varchar(100) NOT NULL,

`score` int NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

INSERT INTO `Players\_score` (`id`, `name`, `score`) VALUES

(3, 'Verolet', 10),

(4, 'Petr 1', -5),

(6, 'BillGates', 10),

(12, 'Peremoga', 10),

(15, 'PLALALALA', -5);

ALTER TABLE `Players\_score`

ADD UNIQUE KEY `id` (`id`),

ADD UNIQUE KEY `name` (`name`);

ALTER TABLE `Players\_score`

MODIFY `id` int UNSIGNED NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=16;

COMMIT;

## **4.3 Проектирование логики обработки данных**

В начале игры пользователь вводит имя в поле с идентификатором input-name, выбирает сложность и начинает игру. В процессе игры программа считает рейтинг пользователя. При завершении игры исполняется скрипт в файле req\_connection.php и отправляет результат на сервер.

Код req\_connection.php:

<?php

$servername = "localhost";

$username = "root";

$password = "";

$dbname = "Score\_Sudoku";

$conn = @new mysqli($servername, $username, $password, $dbname);

if ($conn->connect\_errno) {

exit('Ошибка подключения');

}

echo "Connected successfully to MySQL server. Server version: " . $conn->server\_info;

$playername = trim($\_POST['PlayerName']);

$playerscore = trim($\_POST['PlayerScore']);

$conn->query("INSERT INTO `Players\_score` (`name`, `score`)

VALUES ('$playername', '$playerscore')");

$conn->close();

?>

При загрузке страницы используются полученные значения из базы данных (display\_score.php):

<?php

$servername = "localhost";

$username = "root";

$password = "";

$dbname = "Score\_Sudoku";

$conn = @new mysqli($servername, $username, $password, $dbname);

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

$result = $conn->query("SELECT name, score FROM Players\_score");

if ($result->num\_rows > 0) {

$data = [];

while ($row = $result->fetch\_assoc()) {

$data[] = ["name" => $row["name"], "score" => $row["score"]];

}

echo json\_encode($data);

} else {

echo "0 results";

}

$conn->close();

?>

## **4.4 Разработка класса**

Класс Sudoku – предоставляет возможность создания сетки игры Судоку в виде массива, состоящего из двумерного массива, в каждом массиве 9 элементов, что даёт поле из 81 ячейки.

В таблице 4.1 приведено описание класса:

Таблица 4.1 Описание класса Sudoku

|  |  |
| --- | --- |
| Методы | |
| Имя | Описание |
| constructor(int difficulty) | Конструктор класса. В него передаётся выбранная сложность. |
| public DuplicatePositions(row, column, value) | Возвращает все повторные значения числа в строке, столбце, квадрате 3x3. |
| public find\_dup\_column(row, column, value) | Возвращает все повторные значения в колонке. |
| public find\_dup\_row(row, column, value) | Возвращает все повторные значения в строке. |
| public find\_dup\_square(row, column, value) | Возвращает все повторные значения в квадрате 3x3. |
| public EmptyCells() | Возвращает true или false в зависимости от оставшихся ячеек. |

## **4.5 Разработка основных алгоритмов класса**

### **4.5.1 Заполнение сетки Судоку**

Алгоритм создаёт пустой массив (рис. 4.3), далее заполняет созданный массив с помощью функции FillGrid (рис. 4.4), с помощью функции Checked проверяем подходит ли текущее число в цикле для заполнения (рис. 4.5), затем с помощью функции Clear удаляем элементы из сетки для создания сложности (рис 4.6).

Полный код представлен в приложении 4.

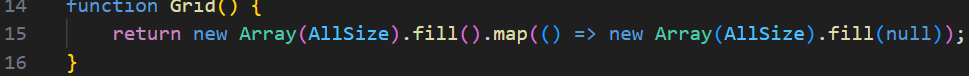


Рисунок 4.4 Создание массива

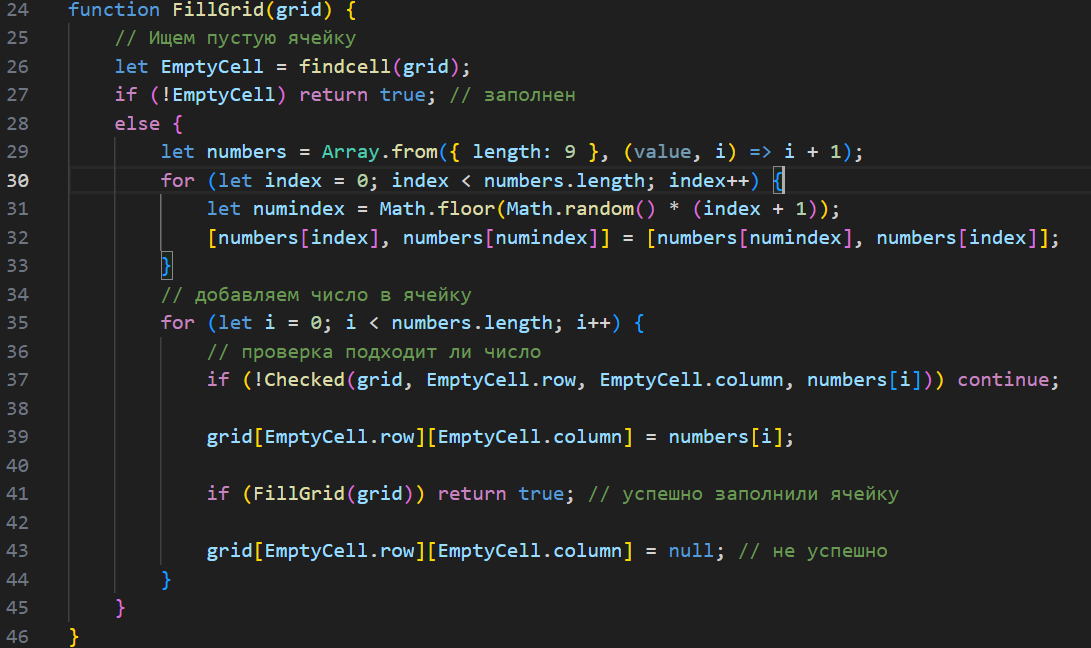


Рисунок 4.5 Заполнение массива

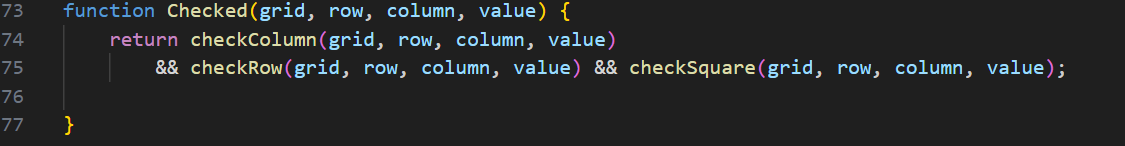


Рисунок 4.6 Проверка значения

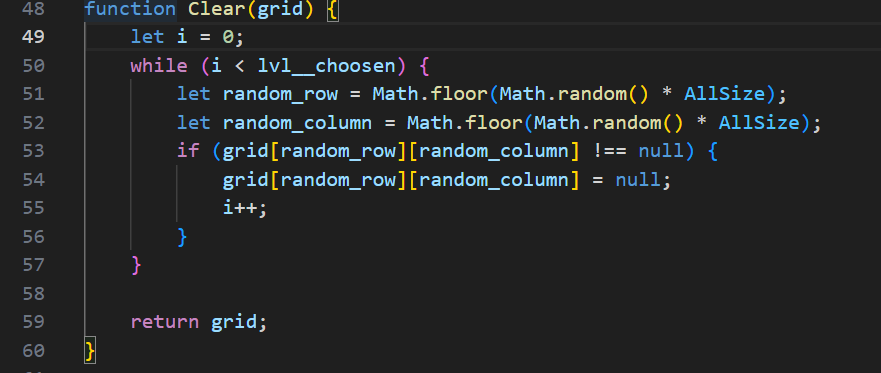


Рисунок 4.7 Очистка игрового поля

Входные данные и выходные данные: значения сложности и частично заполненная сетка Судоку.

Формальное значение (см. рис. 4.7): x – передача уровня сложности (чем больше значение, тем выше сложность).

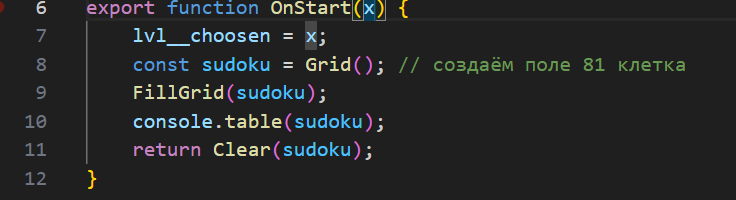


Рисунок 4.8 Формальное значение

Глобальные перeменные: AllSize = 9, SmallSize = 3.

Пример вызова из файла java.js (рис. 4.8)



Рисунок 4.9 Вызов функции

Полный код алгоритма в приложении 3.

Используемые переменные приведены в таблице 4.4

Таблица 4.4 Переменные алгоритма

|  |  |
| --- | --- |
| Переменные | |
| Название | Описание |
| const AllSize const SmallSize | Константные переменные, задают размер двумерных массивов для сетки и размер подполя. |
| lvl\_\_choosen | Хранит уровень сложности. |
| EmptyCell | Хранит null если пустых ячеек на игровом поле нет, иначе хранит индекс пустой ячейки. |
| const sudoku | Содержит игровое поле в виде матрицы. |
| i | Счётчик, существует в поле видимости функции Clear. |
| beginRow beginColumn | Хранят координаты левой верхней ячейки квадрата 3x3. |

Схема алгоритма представлена на рисунке ниже (рис. 4.10)

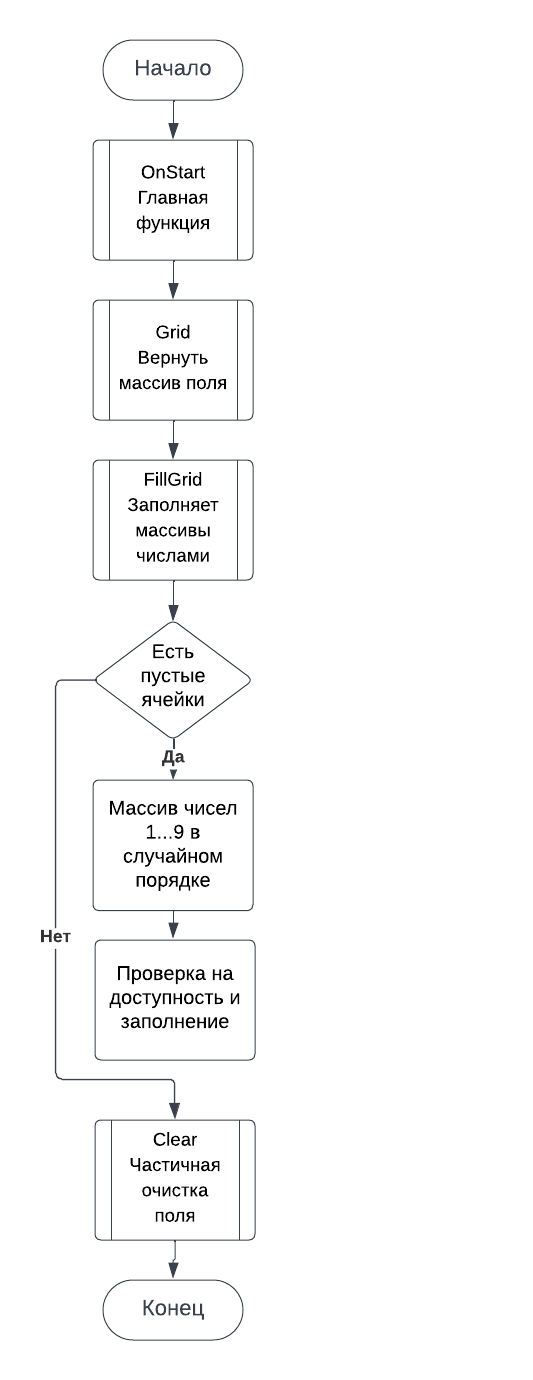


Рисунок 4.10 Схема работы алгоритма

## **4.6 Разработка основных алгоритмов игрового поля**

### **4.6.1 События нажатий по полю**

Игровое поле состоит из 81 ячейки, в начале игры некоторая часть ячеек заполнена изначально и имеет стиль «filled». Каждая ячейка обладает событием click, при срабатывании которого происходит вызов функции CellClick. При взаимодействии с ячейками, имеющими стиль «filled», происходит подсветка только тех ячеек, значения которых совпадают со значением нажатой ячейки (см. рис. 4.11).

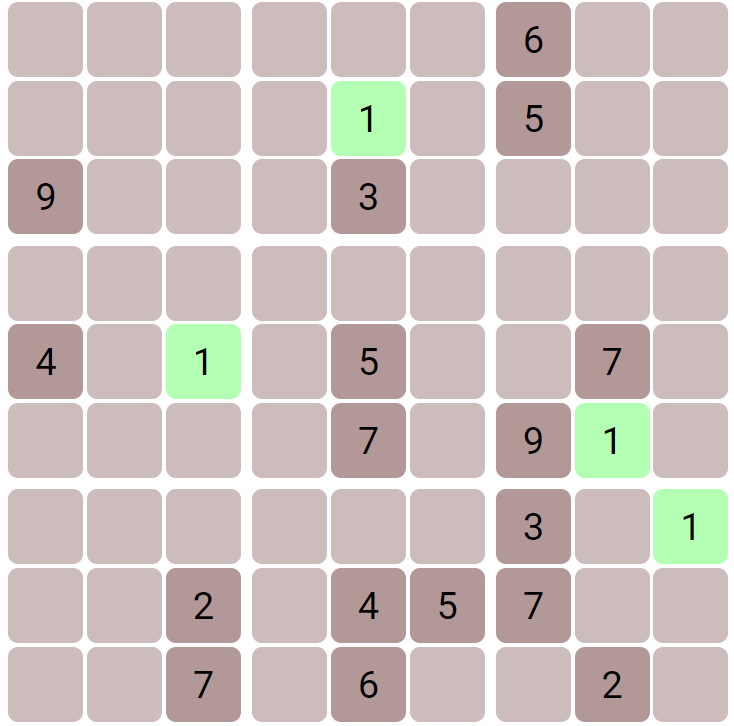


Рисунок 4.11 Нажатие по заполненной ячейке

Если изначально ячейка была пуста, добавляется стиль «clicked», который подсвечивает красным цветом выбранную ячейку и происходит вызов функции highlight для графического выделения строки, столбца и квадрата 3x3, содержащих данную ячейку (см. рис. 4.12).

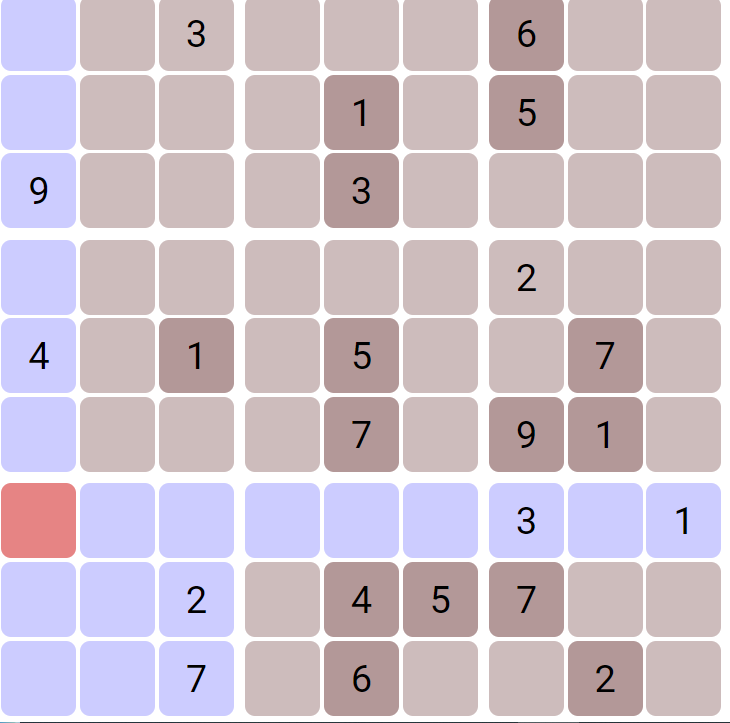


Рисунок 4.12 Графическое выделение

Входные данные: ссылка на html-элемент с которым проходит взаимодействие и его индекс в массиве.

Формальные значения: cell и index – ссылка на элемент взаимодействия и его индекс (см. рис. 4.13).



Рисунок 4.13 Формальные значения

На рисунке 4.14 показано добавление события нажатия по ячейке при котором будет вызываться данная функция:



Рисунок 4.14 Добавление события

Переменная cells содержит статический список всех ячеек.

Схема алгоритма приведена на рисунке 4.15 ниже:

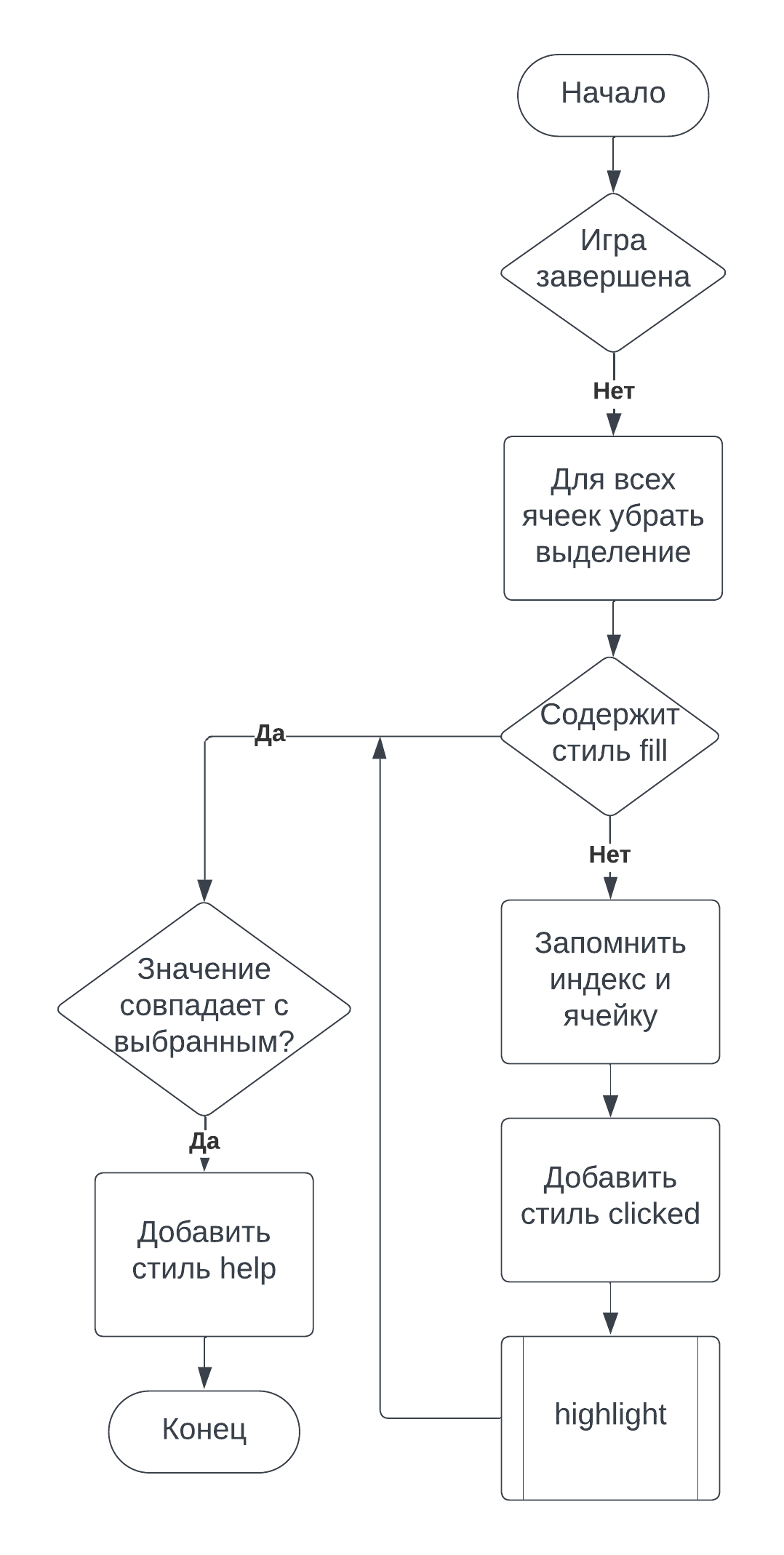


Рисунок 4.15 Схема алгоритма

Код функции highlight:

function highlight(index) {

// выделить

// колонку

let columnIndex = index % 9;

for (let I = 0; I < AllSize; i++) {

cells[columnIndex + I \* 9].classList.add(“highlighted”);

}

// строку

let rowIndex = Math.floor(index / AllSize);

for (let I = 0; I < AllSize; i++) {

cells[rowIndex \* AllSize + i].classList.add(“highlighted”);

}

// квадрат

let beginRow = rowIndex – rowIndex % SmallSize;

let beginColumn = columnIndex – columnIndex % SmallSize;

for (let I = beginRow; I < beginRow + SmallSize; i++) {

for (let j = beginColumn; j < beginColumn + SmallSize; j++) {

cells[I \* AllSize + j].classList.add(“highlighted”);

}

}

}

## **4.7 Программная реализация взаимодействия с клиентом**

Для начала игры пользователю требуется ввести своё имя, длина которого не меньше 4 символов, и, выбрав уровень сложности, нажать кнопку «Играть!».

Для работоспособности кнопок с помощью подключенного файла java.js добавляю обработчик событий “click” для кнопок (см. рис. 4.16)

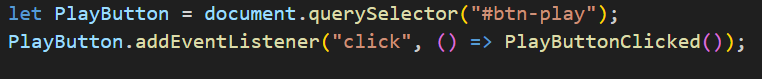


Рисунок 4.16 Обработчик события

Для ввода чисел в сетку с клавиатуры используется функция KeyInput, которая добавляет обработчик событий нажатия кнопки backspace или цифр от 1 до 9. Для ввода с помощью встроенного меню используется функция ButtonsInIt, которая добавляет событие нажатия для кнопок ввода на сайте (рис. 4.17, рис. 4.18 и рис. 4.19)

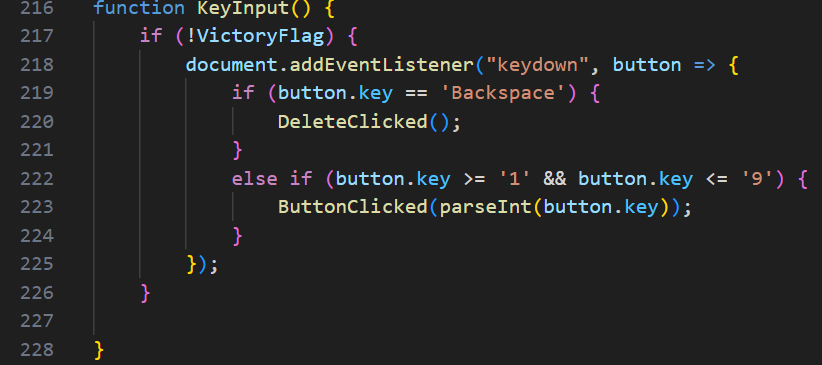


Рисунок 4.17 Функция KeyInput

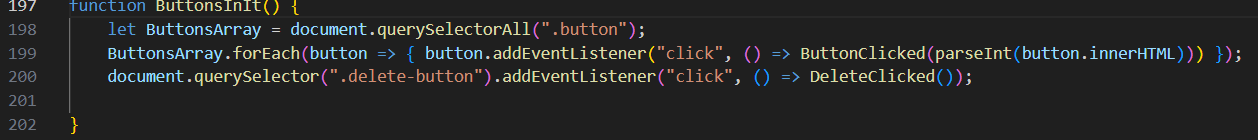


Рисунок 4.18 Функция ButtonsInIt



Рисунок 4.19 Кнопки ввода

Функция cleargrid вызывается при нажатии кнопки «Играть!» и очищает сетку Судоку для дальнейшей игры (рис. 4.20). При вызове очищаются стили, добавленные в процессе игры.

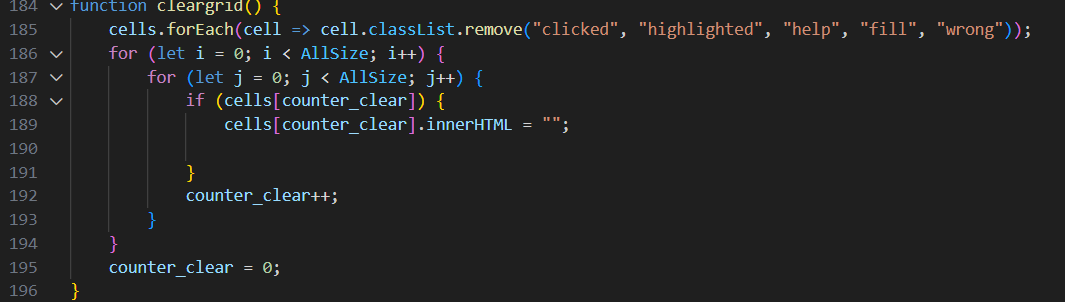


Рисунок 4.20 Функция clearGrid

Для отображения рейтинговой таблицы используется функция InsertScore, которая вставляет полученные значения из базы данных в специальный html контейнер с идентификатором «ScoreContainer» (рис. 4.21). Функция отправляет запрос на сервер с помощью функции fetch и обрабатывает полученные данные.

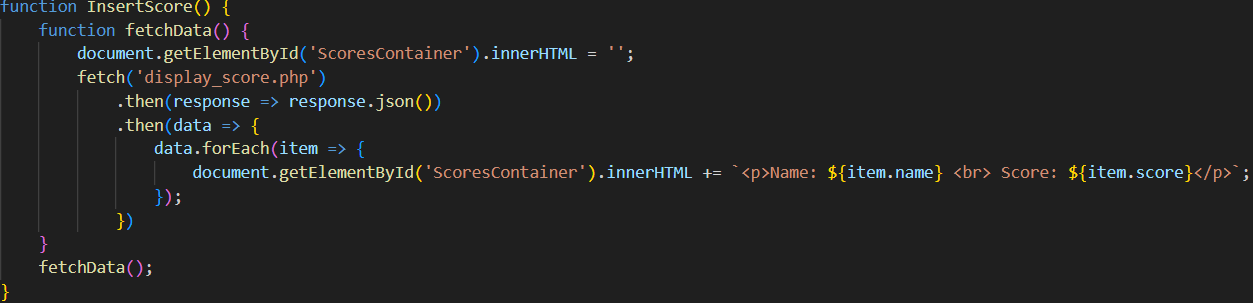


Рисунок 4.21 Функция InsertScore

Для инициализации игрового поля используется функция main, которая заполняет сетку значениями из массива (рис. 4.22). Удаление стиля «start» необходимо для появления поля на экрана, так как он скрыт до начала игры.

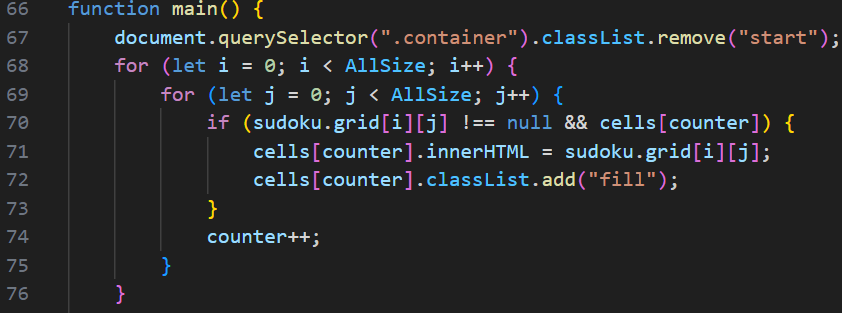


Рисунок 4.22 Функция main

Для выделений, вставки значений в ячейки используется функция ButtonClicked (в следующем пункте разбирается алгоритм выделения).

Удаление введённого значения а также подсветки ячеек происходит с помощью функции DeleteClicked (рис. 4.23).

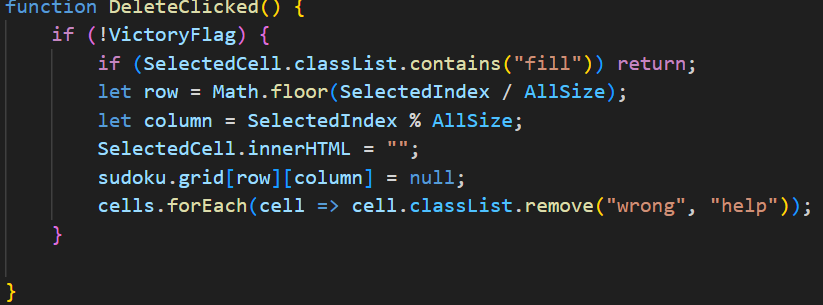


Рисунок 4.23 функция DeleteClicked

Функции CellClick и highlight используются для подсвечивания ячеек (см. рис. 4.24, рис. 4.25)

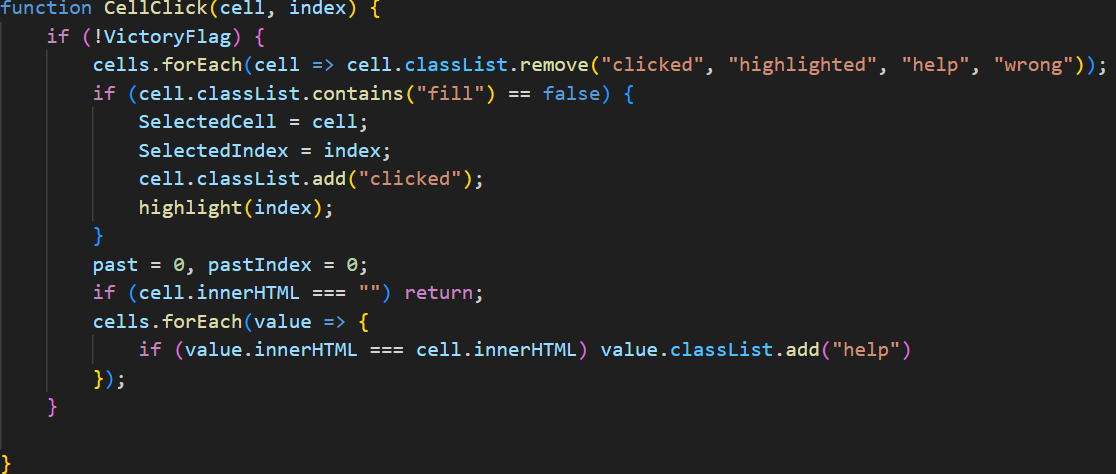
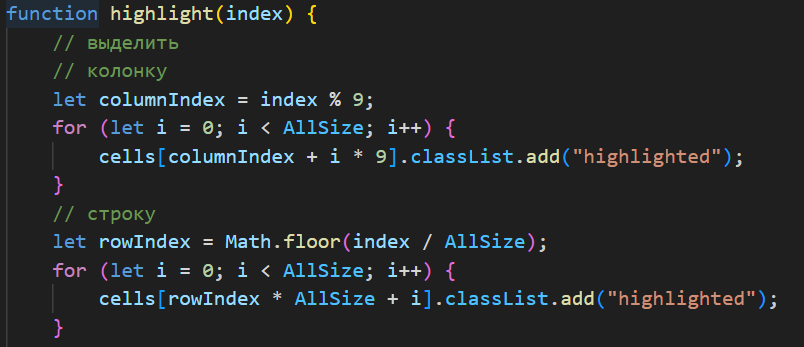


Рисунок 4.24 Функция CellClick



Риснуок 4.25 Фрагмент функции highlight

Функция PlayButtonClicked срабатывает при нажатии кнопки «Играть!» и вызывает выполнение приведённых выше функций (рис. 4. 26).

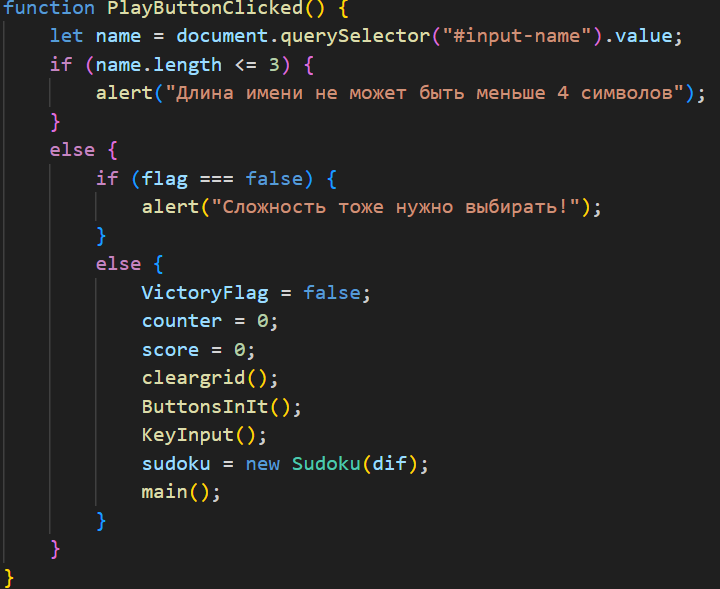


Рисунок 4.26 Функция PlayButtonClicked

Входные данные: имя пользователя и уровень сложности.

Выходные данные: сохранённый результат в базе данных (имя и счёт игрока).

Глобальные перeменные: AllSize = 9, SmallSize = 3.

Полный код в приложении 5.

# **5. Инструкция по эксплуатации**

Данное приложение отличается простотой эксплуатации, пользователю требуется иметь доступ в интернет.

При открытии web-приложения требуется ввести имя пользователя в поле input-name и выбрать уровень сложности, нажав на одну из кнопок difficulty-buttons. Далее остаётся только нажать кнопку «Играть» и заполнить сетку судоку.

По завершению игры счёт пользователя внесётся в базу данных и будет виден для отображения в рейтинговой таблице score\_results.

Ниже представлены иллюстрации по этапам (cм. рис. 5.1 – 5.2):

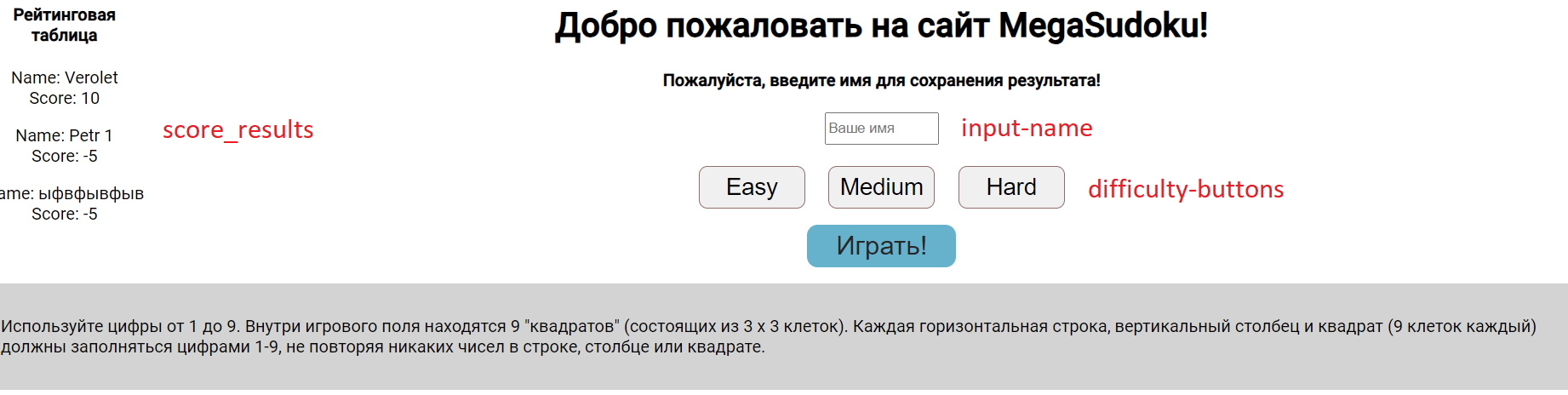


Рисунок 5.1 Ввод имени

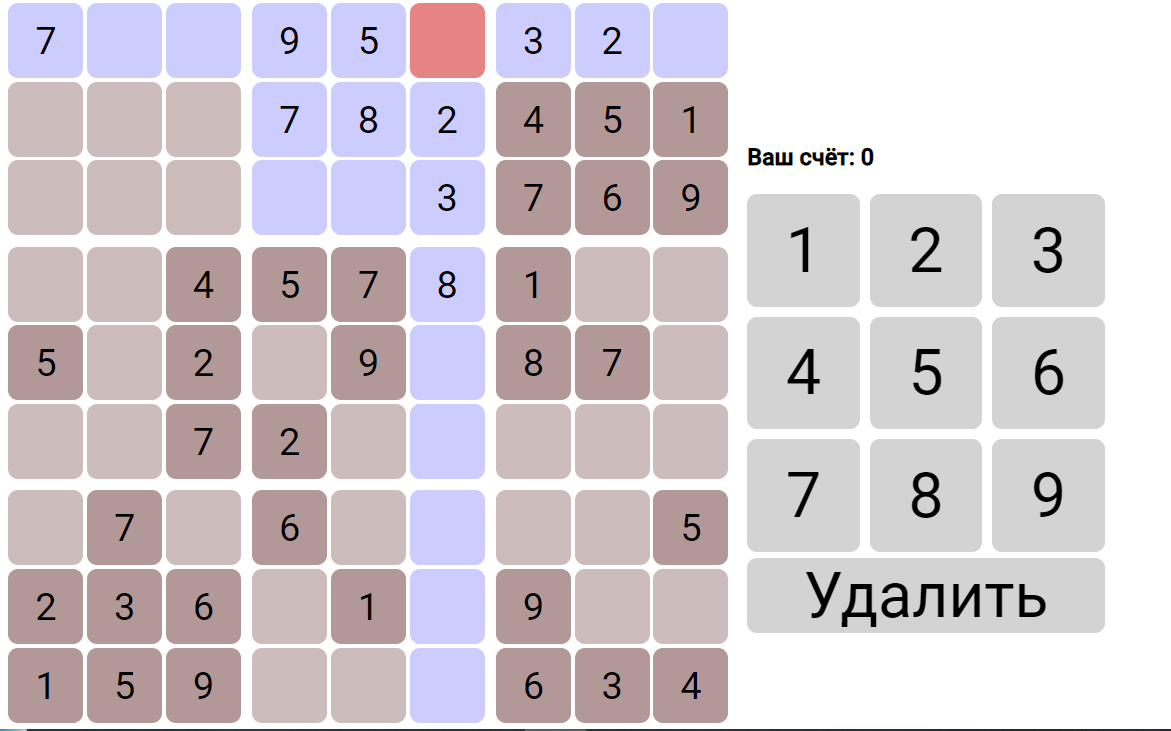


Рисунок 5.2 Заполнение сетки

# **Заключение**

В результате выполнения курсовой работы было разработано web-приложение «Игра Судоку», которое может понравиться многим пользователям из-за своего оформления и рейтинговой системы.

Была разработана база данных, в которой хранится рейтинг игроков. Приложение имеет лёгкую возможность доработки, добавления новых функций и переноса на любые хостинговые сервисы.

При разработке данного web-приложения были изучены технологии взаимодействия сервера с клиентом, серверные скрипты и обработка событий в JavaScript, что позволило улучшить свои профессиональные навыки в сфере IT.

# **Список литературы**

1. JavaScript // <https://learn.javascript.ru/> (дата обращения 11.01.2023)

2. SCSS // <https://sass-lang.com/documentation> (дата обращения 05.01.2023)

3. HTML // <https://hcdev-ru.pages.dev/html/> (дата обращения 02.01.2023)

4. Статьи по судоку // <https://nuancesprog.ru/p/17048/> <https://habr.com/ru/articles/113837/> (дата обращения 20.12.2023)

# ПРИЛОЖЕНИЕ 1. html-разметка

<!DOCTYPE html>

<html lang=”en”>

<head>

<meta charset=”UTF-8”>

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

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

<script src=”./java.js” type=”module”></script>

<title>Судоку для всех!</title>

</head>

<body>

<header>

<div class=”header\_\_wrapper”>

<div class=”score\_\_wrapper”>

<h4>Рейтинговая таблица</h4>

<div class=”score\_results” id=”ScoresContainer”>

</div>

</div>

<div class=”header\_\_wrapper\_\_menu”>

<h1>Добро пожаловать на сайт MegaSudoku!</h1>

<h4>Пожалуйста, введите имя для сохранения результата!</h4>

<div class=”button”><input type=”text” placeholder=”Ваше имя” maxlength=”13” class=”input-name”

id=”input-name”>

</div>

<div id=”difficulty-buttons” class=”difficulty-buttons”>

<button onclick=”setDifficulty(1)”>Easy</button>

<button onclick=”setDifficulty(40)”>Medium</button>

<button onclick=”setDifficulty(60)”>Hard</button>

</div>

<div class=”wrapper-button”>

<button class=”btn” id=”btn-play” type=”button”>Играть!</button>

</div>

</div>

</div>

</header>

<div class=”rules”>

<p>Используйте цифры от 1 до 9.

Внутри игрового поля находятся 9 «квадратов» (состоящих из 3 x 3 клеток). Каждая горизонтальная строка,

вертикальный столбец и квадрат (9 клеток каждый) должны заполняться цифрами 1-9, не повторяя никаких чисел в

строке, столбце или квадрате.</p>

</div>

<div class=”container start”>

<div class=”container\_\_grid”>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

<div class=”container\_\_grid\_\_cell”></div>

</div>

<div class=”wrapper”>

<div class=”score”>

<h3>Ваш счёт: <span id=”ScoreBD”>0</span></h3>

</div>

<div class=”container\_\_buttons”>

<div class=”button”>1</div>

<div class=”button”>2</div>

<div class=”button”>3</div>

<div class=”button”>4</div>

<div class=”button”>5</div>

<div class=”button”>6</div>

<div class=”button”>7</div>

<div class=”button”>8</div>

<div class=”button”>9</div>

</div>

<div class=”delete-button”>Удалить</div>

</div>

</div>

</body>

</html>

# ПРИЛОЖЕНИЕ 2. SCSS

@import url(“https://fonts.googleapis.com/css2?family=Roboto:wght@300;400;500&display=swap”);

:root {

--bg-main: #2a2a38;

--color-txt: hsl(0, 0%, 15%);

--border-radius: 10px;

--font-size-lg: 25px;

--blue: hsl(196, 50%, 60%);

--cells\_color: hsl(0, 15%, 77%);

--cells\_fill: hsl(0, 15%, 65%);

--cell\_clicked: hsl(0, 66%, 71%);

--cell\_high: hsl(240, 100%, 90%);

--color\_help: hsl(120, 100%, 85%);

--bg\_color: hsl(0, 100%, 60%);

}

body,

html {

margin: 0;

font-family: “Roboto”, sans-serif;

}

header {

display: grid;

& > \* {

justify-content: center;

text-align: center;

}

& h1 {

margin-bottom: 2px;

}

& h4 {

text-align: center;

}

& .entry {

text-align: center;

}

}

.wrapper {

display: grid;

margin-top: 15px;

margin-left: 15px;

}

.rules {

padding: 15px;

background-color: lightgray;

margin-top: 15px;

}

.header\_\_wrapper {

display: grid;

grid-template-columns: 150px 1fr;

& .score\_results {

height: 100%;

}

&\_\_menu {

display: flex;

flex-direction: column;

place-content: center;

}

}

.start {

display: none !important;

}

.input-name {

width: 100px;

height: 25px;

}

.btn {

height: 40px;

width: 140px;

background-color: var(--blue);

color: var(--color-txt);

border-radius: var(--border-radius);

display: grid;

place-items: center;

transition: width 0.3s ease-in-out;

overflow: hidden;

font-size: var(--font-size-lg);

cursor: pointer;

border: none;

&:hover {

background-color: hsl(196, 50%, 75%);

}

}

.difficulty-buttons {

margin-top: 20px;

display: flex;

justify-content: center;

gap: 22px;

& button {

width: 100px;

height: 40px;

cursor: pointer;

font-size: 22px;

border-radius: 8px;

border: 1px solid hsl(0, 15%, 50%);

&:hover {

background-color: hsl(0, 0%, 80%);

}

}

}

.container {

margin-top: 15px;

display: flex;

flex-direction: row;

justify-content: center;

align-items: center;

&\_\_grid {

display: grid;

grid-template-columns: repeat(9, auto);

grid-template-rows: repeat(9, auto);

gap: 3px;

place-items: center;

&\_\_cell {

display: flex;

font-size: 30px;

height: 60px;

width: 60px;

background-color: var(--cells\_color);

cursor: pointer;

justify-content: center;

align-items: center;

border-radius: 8px;

}

}

&\_\_buttons {

font-size: 50px;

display: grid;

grid-template-columns: repeat(3, 1fr);

grid-template-rows: repeat(3, 1fr);

place-items: center;

gap: 8px;

& .button {

justify-content: center;

align-items: center;

display: flex;

background: lightgray;

height: 90px;

width: 90px;

border-radius: 8px;

cursor: pointer;

}

& .del-but {

width: 100%;

grid-column: 2;

}

}

}

.delete-button {

margin-top: 5px;

font-size: 50px;

background-color: lightgray;

cursor: pointer;

display: flex;

border-radius: 8px;

justify-content: center;

width: 100%;

}

.wrapper-button {

margin-top: 15px;

display: flex;

justify-content: center;

}

.container\_\_grid\_\_cell {

&:nth-child(9n + 3),

&:nth-child(9n + 6) {

margin-right: 6px;

}

@for $i from 19 to 28 {

&:nth-child(27n + #{$i}) {

margin-bottom: 6px;

}

}

}

.fill {

background-color: var(--cells\_fill);

}

.highlighted {

background-color: var(--cell\_high);

}

.clicked {

background-color: var(--cell\_clicked);

}

.help {

background-color: var(--color\_help);

}

.wrong {

background-color: var(--bg\_color);

}

# ПРИЛОЖЕНИЕ 3. класс Sudoku

import { OnStart, findcell } from “./OnStart.js”;

const AllSize = 9;

const SmallSize = 3;

export class Sudoku {

constructor(difficulty = 0) {

this.grid = OnStart(difficulty);

}

DuplicatePositions(row, column, value) {

const dup\_Column = this.find\_dup\_column(row, column, value)

const dupl\_Row = this.find\_dup\_row(row, column, value);

const dup\_square = this.find\_dup\_square(row, column, value);

const duplicates = [...dup\_Column, ...dupl\_Row, ...dup\_square];

return duplicates;

}

find\_dup\_column(row, column, value) {

const duplicates = [];

for (let I = 0; I < AllSize; i++) {

if (this.grid[i][column] === value && I !== row) {

duplicates.push({ row: I, column: column });

}

}

return duplicates;

}

find\_dup\_row(row, column, value) {

const duplicates = [];

for (let I = 0; I < AllSize; i++) {

if (this.grid[row][i] === value && I !== column) {

duplicates.push({ row: row, column: I });

}

}

return duplicates;

}

find\_dup\_square(row, column, value) {

const duplicates = [];

const beginRow = row – row % SmallSize;

const beginColumn = column – column % SmallSize;

for (let I = beginRow; I < beginRow + SmallSize; i++) {

for (let j = beginColumn; j < beginColumn + SmallSize; j++) {

if (this.grid[i][j] === value && I !== row && j !== column) {

duplicates.push({ row: I, column: j });

}

}

}

return duplicates;

}

EmptyCells() {

return Boolean(findcell(this.grid));

}

}

# ПРИЛОЖЕНИЕ 4. OnStart.js

const AllSize = 9;

const SmallSize = 3;

var lvl\_\_choosen = 0;

// выбранный уровень будет браться из уровня сложности

export function OnStart(x) {

lvl\_\_choosen = x;

const sudoku = Grid(); // создаём поле 81 клетка

FillGrid(sudoku);

console.table(sudoku);

return Clear(sudoku);

}

function Grid() {

return new Array(AllSize).fill().map(() => new Array(AllSize).fill(null)); // массив из 9 массивов заполненые 0

}

// заполняем массив числами

// Алгоритм: фун-ия находит пустую ячейку и пытается добавить значение

// проверяя, что значение не повторяется в строке, столбце и маленьком квадрате (3x3)

// после добавления числа ф-ия вызывает сама себя, ищет пустую ячейку и т.д. пока сетка не заполнится

// если сетка заполнена некорректно, удалит последнее значение, возвращается на шаг вверх и пробует другое значение

function FillGrid(grid) {

// Ищем пустую ячейку

let EmptyCell = findcell(grid);

if (!EmptyCell) return true; // заполнен

else {

let numbers = Array.from({ length: 9 }, (value, i) => i + 1);

for (let index = 0; index < numbers.length; index++) {

let numindex = Math.floor(Math.random() \* (index + 1));

[numbers[index], numbers[numindex]] = [numbers[numindex], numbers[index]];

}

// добавляем число в ячейку

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

// проверка подходит ли число

if (!Checked(grid, EmptyCell.row, EmptyCell.column, numbers[i])) continue;

grid[EmptyCell.row][EmptyCell.column] = numbers[i];

if (FillGrid(grid)) return true; // успешно заполнили ячейку

grid[EmptyCell.row][EmptyCell.column] = null; // не успешно

}

}

}

function Clear(grid) {

let i = 0;

while (i < lvl\_\_choosen) {

let random\_row = Math.floor(Math.random() \* AllSize);

let random\_column = Math.floor(Math.random() \* AllSize);

if (grid[random\_row][random\_column] !== null) {

grid[random\_row][random\_column] = null;

i++;

}

}

return grid;

}

export function findcell(a) {

for (let row = 0; row < AllSize; row++) {

for (let column = 0; column < AllSize; column++) {

if (a[row][column] === null) return { row, column };

}

}

return null; // если нет пустых ячеек

}

function Checked(grid, row, column, value) {

return checkColumn(grid, row, column, value)

&& checkRow(grid, row, column, value) && checkSquare(grid, row, column, value);

}

function checkColumn(grid, row, column, value) {

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

if (grid[j][column] === value && j !== row) {

return false;

}

}

return true;

}

function checkRow(grid, row, column, value) {

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

if (grid[row][j] === value && j !== column) {

return false;

}

}

return true;

}

function checkSquare(grid, row, column, value) {

let beginRow = row - row % SmallSize;

let beginColumn = column - column % SmallSize;

for (let i = beginRow; i < beginRow + SmallSize; i++) {

for (let j = beginColumn; j < beginColumn + SmallSize; j++) {

if (grid[i][j] === value && i !== row && j !== column) return false;

}

}

return true;

}

# ПРИЛОЖЕНИЕ 5. java.js

import { Sudoku } from "./sudoku.js";

const AllSize = 9;

const SmallSize = 3;

let cells = document.querySelectorAll(".container\_\_grid\_\_cell");

let counter = 0;

let dif = 0;

let flag = false;

let sudoku;

let counter\_clear = 0;

let SelectedCell;

let VictoryFlag = false;

let SelectedIndex = 0;

let score = 0;

let past = 0, pastIndex = 0;

InsertScore();

window.setDifficulty = setDifficulty;

export function setDifficulty(x) {

dif = x;

flag = true;

}

function InsertScore() {

function fetchData() {

document.getElementById('ScoresContainer').innerHTML = '';

fetch('display\_score.php')

.then(response => response.json())

.then(data => {

data.forEach(item => {

document.getElementById('ScoresContainer').innerHTML += `<p>Name: ${item.name} <br> Score: ${item.score}</p>`;

});

})

}

fetchData();

}

let PlayButton = document.querySelector("#btn-play");

PlayButton.addEventListener("click", () => PlayButtonClicked());

function PlayButtonClicked() {

let name = document.querySelector("#input-name").value;

if (name.length <= 3) {

alert("Длина имени не может быть меньше 4 символов");

}

else {

if (flag === false) {

alert("Сложность тоже нужно выбирать!");

}

else {

VictoryFlag = false;

counter = 0;

score = 0;

cleargrid();

ButtonsInIt();

KeyInput();

sudoku = new Sudoku(dif);

main();

}

}

}

function main() {

document.querySelector(".container").classList.remove("start");

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

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

if (sudoku.grid[i][j] !== null && cells[counter]) {

cells[counter].innerHTML = sudoku.grid[i][j];

cells[counter].classList.add("fill");

}

counter++;

}

}

cells.forEach((cell, index) => { cell.addEventListener("click", () => CellClick(cell, index)) });

}

function ButtonClicked(x) {

if (!VictoryFlag) {

if (SelectedCell === null || SelectedCell.classList.contains(".fill")) {

return;

}

cells.forEach(cell => cell.classList.remove("wrong", "help"));

SelectedCell.classList.add("clicked");

let row = Math.floor(SelectedIndex / AllSize);

let column = SelectedIndex % AllSize;

let WrongPositions = sudoku.DuplicatePositions(row, column, x);

if (WrongPositions.length > 0) {

WrongPositions.forEach(position => {

let index = position.row \* AllSize + position.column;

cells[index].classList.add("wrong");

});

if (x != past && SelectedIndex != pastIndex) {

score -= 5;

document.querySelector(".score").querySelector("span").innerHTML = `${score}`;

}

past = x;

pastIndex = SelectedIndex;

return;

}

if (x != past && SelectedIndex != pastIndex) {

score += 10;

document.querySelector(".score").querySelector("span").innerHTML = `${score}`;

}

past = x;

pastIndex = SelectedIndex;

sudoku.grid[row][column] = x;

SelectedCell.innerHTML = x;

const PlayerName = document.getElementById('input-name').value;

const PlayerScore = document.getElementById('ScoreBD').textContent;

if (sudoku.EmptyCells() == false) {

VictoryFlag = true;

fetch('req\_connection.php', {

method: 'POST',

headers: {

'Content-Type': 'application/x-www-form-urlencoded',

},

body: 'PlayerName=' + encodeURIComponent(PlayerName) +

'&PlayerScore=' + encodeURIComponent(PlayerScore),

})

.then(response => response.text())

.then(data => {

console.log(data);

})

.catch(error => {

console.error('Error:', error);

});

alert("Вы победили!");

}

}

}

function CellClick(cell, index) {

if (!VictoryFlag) {

cells.forEach(cell => cell.classList.remove("clicked", "highlighted", "help", "wrong"));

if (cell.classList.contains("fill") == false) {

SelectedCell = cell;

SelectedIndex = index;

cell.classList.add("clicked");

highlight(index);

}

past = 0, pastIndex = 0;

if (cell.innerHTML === "") return;

cells.forEach(value => {

if (value.innerHTML === cell.innerHTML) value.classList.add("help")

});

}

}

function highlight(index) {

// выделить

// колонку

let columnIndex = index % 9;

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

cells[columnIndex + i \* 9].classList.add("highlighted");

}

// строку

let rowIndex = Math.floor(index / AllSize);

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

cells[rowIndex \* AllSize + i].classList.add("highlighted");

}

// квадрат

let beginRow = rowIndex - rowIndex % SmallSize;

let beginColumn = columnIndex - columnIndex % SmallSize;

for (let i = beginRow; i < beginRow + SmallSize; i++) {

for (let j = beginColumn; j < beginColumn + SmallSize; j++) {

cells[i \* AllSize + j].classList.add("highlighted");

}

}

}

function cleargrid() {

cells.forEach(cell => cell.classList.remove("clicked", "highlighted", "help", "fill", "wrong"));

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

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

if (cells[counter\_clear]) {

cells[counter\_clear].innerHTML = "";

}

counter\_clear++;

}

}

counter\_clear = 0;

}

function ButtonsInIt() {

let ButtonsArray = document.querySelectorAll(".button");

ButtonsArray.forEach(button => { button.addEventListener("click", () => ButtonClicked(parseInt(button.innerHTML))) });

document.querySelector(".delete-button").addEventListener("click", () => DeleteClicked());

}

function DeleteClicked() {

if (!VictoryFlag) {

if (SelectedCell.classList.contains("fill")) return;

let row = Math.floor(SelectedIndex / AllSize);

let column = SelectedIndex % AllSize;

SelectedCell.innerHTML = "";

sudoku.grid[row][column] = null;

cells.forEach(cell => cell.classList.remove("wrong", "help"));

}

}

function KeyInput() {

if (!VictoryFlag) {

document.addEventListener("keydown", button => {

if (button.key == 'Backspace') {

DeleteClicked();

}

else if (button.key >= '1' && button.key <= '9') {

ButtonClicked(parseInt(button.key));

}

});

}

}