МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №1
по дисциплине «Web-технологии»
Тема: Тетрис на Javascript.

Студентка гр. 9304 _____ Рослова Л.С.

Преподаватель _____ Беляев С.А.

Санкт-Петербург
2021

Цель работы.

Изучение работы web-сервера nginx со статическими файлами и создание клиентских JavaScript web-приложений. Основные теоретические положения. Асимметричные ключи используются в асимметричных алгоритмах шифрования и являются ключевой парой. Закрытый ключ известен только владельцу. Открытый ключ может быть опубликован и используется для проверки подлинности подписанного документа (сообщения). Открытый ключ вычисляется, как значение некоторой функции от закрытого ключа, но знание открытого ключа не дает возможности определить закрытый ключ. По секретному ключу можно вычислить открытый ключ, но по открытому ключу практически невозможно вычислить закрытый ключ. nginx (https://nginx.ru/ru/) – веб-сервер, работающий на Unix-подобных операционных системах и в операционной системе Windows. JavaScript (https://learn.javascript.ru/) – язык программирования, он поддерживает объектно-ориентированный и функциональный стили программирования. Является реализацией языка ECMAScript.

Задание.

Необходимо создать web-приложение – игру в тетрис. Основные требования:

– сервер – nginx, протокол взаимодействия – HTTPS;

– отображается страница для ввода имени пользователя с использованием HTML-элементов ;

– статическая страница отображает «стакан» для тетриса с использованием HTML-элемента используется для отображения следующей фигуры, отображается имя пользователя;

– фигуры в игре – классические фигуры тетриса (7 шт. тетрамино);

– случайным образом генерируется фигура и начинает падать в «стакан» (описание правил см., например, https://ru.wikipedia.org/wiki/Тетрис);

– пользователь имеет возможность двигать фигуру влево и вправо, повернуть на 90 градусов и «уронить»;

– если собралась целая «строка», она должна исчезнуть;

– при наборе некоторого заданного числа очков увеличивается уровень, что заключается в увеличении скорости игры;

– пользователь проигрывает, когда стакан «заполняется», после чего ему отображается локальная таблица рекордов;

– вся логика приложения написана на JavaScript. Необязательно: оформление с использованием CSS .

Выполнение работы.

Сначала была написана стартовая страница index.html, реализует форму для входа в игру. Изображение страницы представлено на рисунке 1.

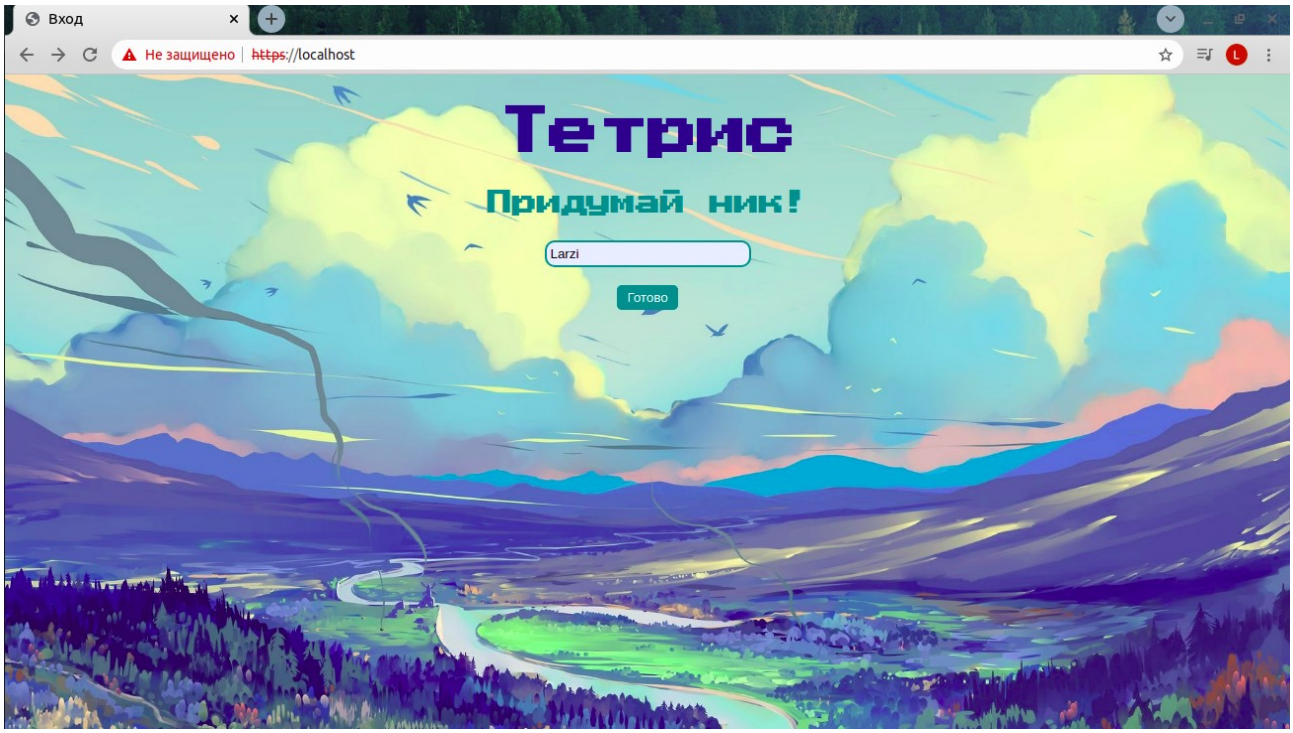

Рисунок 1 – Форма входа

Далее была написана основная страница main.html, содержащая стакан тетриса, данные о текущей игре и таблицу рекордов. Представление основной страницы игры изображено на рисунке 2.

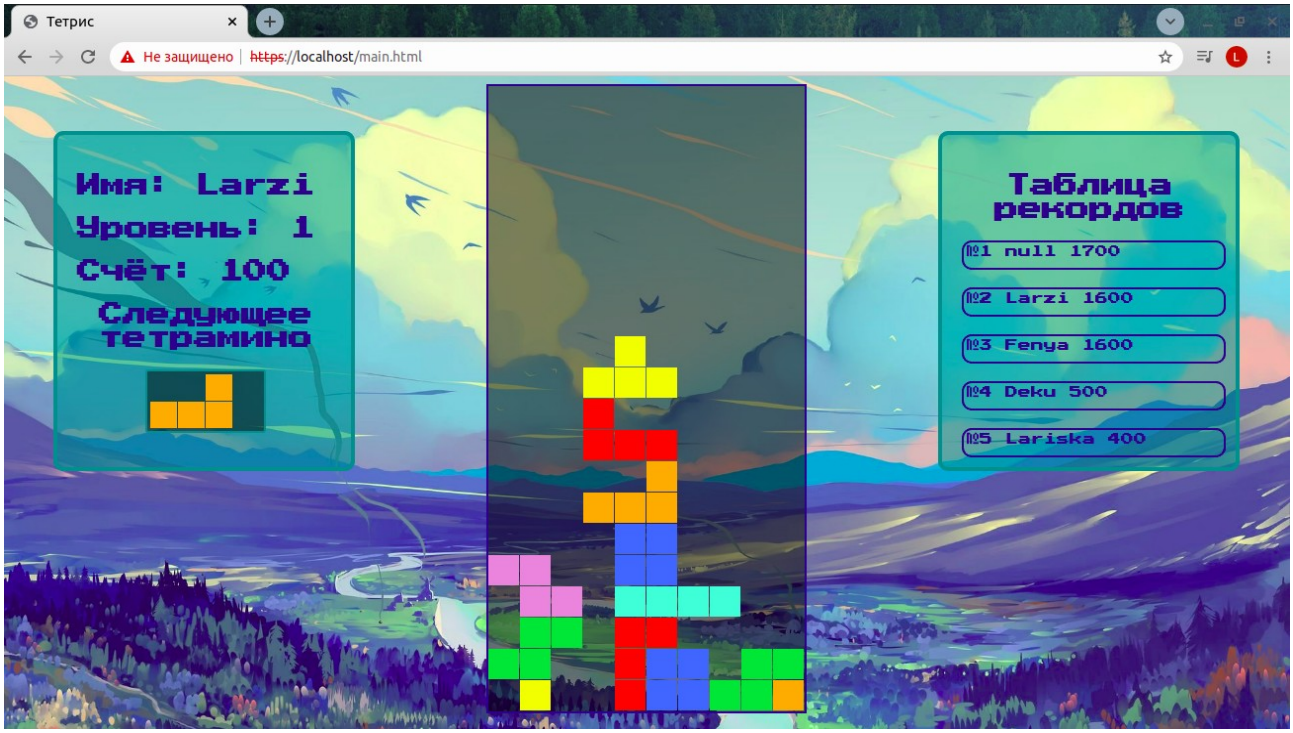

Рисунок 2 – Страница с игрой

Выводы.

Изучили работу web-сервера nginx со статическими файлами и создали в качестве клиентского JavaScript web-приложения игру Тетрис.

Приложение А

Исходный код программы

Файл entry.js

```
function check_name(el){
    var user = el.user_name.value;
    var fail = "";

    if(user =="")
        fail = "Заполните поле с именем!";
    else if(user.length <=1 || user.length > 50)
        fail = "Введите значение в пределах от 1 до 50 символов!";

    if(fail != ""){
    document.getElementById('error').innerHTML = fail;
    return false;
    } else{
        localStorage["tetris.username"] = user;
        window.location = 'https://localhost/main.html';
        return false;
    }
}
```

Файл game.js

```
let level_on_window = document.getElementById('level');
let level = 1;
level_on_window.innerHTML = level_on_window.innerHTML + level;

let score_on_window = document.getElementById('score');
let score = 0;
```

```
score_on_window.innerHTML = score_on_window.innerHTML + score;

let name_on_window = document.getElementById('name');
let user_name = localStorage.getItem("tetris.username");
name_on_window.innerHTML = name_on_window.innerHTML + user_name;

// получаем доступ к холсту
const canvas = document.getElementById('field');
const context = canvas.getContext('2d');

//Доступ к холсту следующей фигуры
const canvas_info = document.getElementById('next_tetramino');
const context_info = canvas_info.getContext('2d');

// размер квадратика
const grid = 32;
const grid_info = 28;

// массив с последовательностями фигур
let tetrominoSequence = [];

//двумерный массив игрового поля заполенный на старте пустыми ячейками
let playfield = [];
for (let row = -2; row < 20; row++) {
    playfield[row] = [];
    for (let col = 0; col < 10; col++) {
        playfield[row][col] = 0;
    }
}
```

```
// Список фигур
const tetrominos = {
    'I': [
        [0,0,0,0],
        [1,1,1,1],
        [0,0,0,0],
        [0,0,0,0]
    ],
    'J': [
        [1,0,0],
        [1,1,1],
        [0,0,0],
    ],
    'L': [
        [0,0,1],
        [1,1,1],
        [0,0,0],
    ],
    'O': [
        [1,1],
        [1,1],
    ],
    'S': [
        [0,1,1],
        [1,1,0],
        [0,0,0],
    ],
    'Z': [
        [1,1,0],
```

```
        [0,1,1],
        [0,0,0],
    ],
    'T': [
        [0,1,0],
        [1,1,1],
        [0,0,0],
    ]
};

// цвет каждой фигуры
const colors = {
    'I': 'Aquamarine',
    'O': 'DodgerBlue',
    'T': 'yellow',
    'S': 'LimeGreen',
    'Z': 'plum',
    'J': 'red',
    'L': 'orange',
};

// счётчик
let count = 0;
// текущая фигура в игре
let tetromino = getNextTetromino();
// следим за кадрами анимации, чтобы если что — остановить игру
let rAF = null;
// флаг конца игры, на старте — неактивный
let gameOver = false;
```

```
        // Функция возвращает случайное число в заданном диапазоне
        function getRandomInt(min, max) {
            min = Math.ceil(min);
            max = Math.floor(max);

            return Math.floor(Math.random() * (max - min + 1)) + min;
        }

        // создаём последовательность фигур, которая появится в игре
        function generateSequence() {
            const sequence = ['I', 'J', 'L', 'O', 'S', 'T', 'Z'];
            while (sequence.length) {
                // случайным образом находим любую из них
                const rand = getRandomInt(0, sequence.length - 1);
                const name = sequence.splice(rand, 1)[0];
                // помещаем выбранную фигуру в игровой массив с
последовательностями
                tetrominoSequence.push(name);
            }
        }

        // Отрисовка следующей тетраминошки в боковом окошке.
        function next_tetramino(){
            if(tetrominoSequence.length === 1){
                generateSequence();
            }
            let next_tetromino_name = tetrominoSequence[tetrominoSequence.length -
1];
            let next_tetromino_matrix = tetrominos[next_tetromino_name];
```

```
            context_info.clearRect(0, 0, canvas_info.width, canvas_info.height);
            context_info.fillStyle = colors[next_tetromino_name];

            for(let row = 0; row < next_tetromino_matrix.length; row++){
                for(let col = 0; col < next_tetromino_matrix[row].length; col++){
                    if(next_tetromino_matrix[row][col]) {

                        context_info.fillRect(col * grid_info, row * grid_info, grid_info-1,
grid_info-1);
                    }
                }
            }
        }
        // получаем следующую фигуру
        function getNextTetromino() {
            // если следующей нет — генерируем
            if (tetrominoSequence.length === 0) {
                generateSequence();
            }
            const name = tetrominoSequence.pop();
            const matrix = tetrominos[name];

            // I и O стартуют с середины, остальные — чуть левее
            const col = playfield[0].length / 2 - Math.ceil(matrix[0].length / 2);
            // I начинает с 21 строки (смещение -1), а все остальные — со строки 22
(смещение -2)
            const row = name === 'I' ? -1 : -2;
            next_tetramino();
            return {
```

```
            name: name,      // название фигуры (L, O, и т.д.)
            matrix: matrix,  // матрица с фигурой
            row: row,        // текущая строка (фигуры стартуют за видимой областью холста)
            col: col         // текущий столбец
        };
    }

    // поворачиваем матрицу на 90 градусов
    function rotate(matrix) {
        const N = matrix.length - 1;
        return matrix.map((row, i) =>
            row.map((val, j) => matrix[N - j][i])
        );
    }

    // проверяем после появления или вращения, может ли матрица (фигура) быть в этом месте поля или она вылезет за его границы
    function checkMove(matrix, cellRow, cellCol) {
        // проверяем все строки и столбцы
        for (let row = 0; row < matrix.length; row++) {
            for (let col = 0; col < matrix[row].length; col++) {
                if (matrix[row][col] && (
                    // если выходит за границы поля…
                    cellCol + col < 0 ||
                    cellCol + col >= playfield[0].length ||
                    cellRow + row >= playfield.length ||
                    // …или пересекается с другими фигурами
                    playfield[cellRow + row][cellCol + col])
                ) {
```

```
                return false;
            }
        }
    }
    return true;
}

// когда фигура окончательна встала на своё место
function placeFigure() {
    for (let row = 0; row < tetromino.matrix.length; row++) {
        for (let col = 0; col < tetromino.matrix[row].length; col++) {
            if (tetromino.matrix[row][col]) {
                // если край фигуры после установки вылезает за границы поля, то игра закончилась
                if (tetromino.row + row < 0) {
                    return GameOver();
                }
                // если всё в порядке, то записываем в массив игрового поля нашу фигуру
                playfield[tetromino.row + row][tetromino.col + col] = tetromino.name;
                //playfield[tetromino.row + row][tetromino.col + col] = 'S';
            }
        }
    }

    // проверяем, чтобы заполненные ряды очистились снизу вверх
    for (let row = playfield.length - 1; row >= 0; ) {
        // если ряд заполнен
        if (playfield[row].every(cell => !!cell)) {
```

```
            // очищаем его и опускаем всё вниз на одну клетку
            for (let r = row; r >= 0; r--) {
                for (let c = 0; c < playfield[r].length; c++) {
                    playfield[r][c] = playfield[r-1][c];
                }
            }
            score += 100;
            score_on_window.innerHTML = "Счёт: " + score;
            level = Math.floor(score/200)+1;
            level_on_window.innerHTML = "Уровень: " + level;
        }
        else {
            // переходим к следующему ряду
            row--;
        }
    }
    // получаем следующую фигуру
    tetromino = getNextTetromino();
}

// показываем надпись Game Over
function GameOver() {
    // прекращаем всю анимацию игры
    cancelAnimationFrame(rAF);
    gameOver = true;
    context.fillStyle = 'darkblue';
    context.globalAlpha = 0.6;
    context.fillRect(0, 0, canvas.width, canvas.height);

    context.globalAlpha = 1;
```

```
        context.fillStyle = 'white';
        context.font = '35px Arial';
        context.textAlign = 'center';
        context.textBaseline = 'middle';
        context.fillText('Конец игры!', canvas.width / 2, canvas.height / 2);
        addRecord();
        updateRecordTable();
    }

    // следим за нажатиями на клавиши
    document.addEventListener('keydown', function(e) {
        // если игра закончилась — сразу выходим
        if (gameOver) return;

        // стрелки влево и вправо
        if (e.which === 37 || e.which === 39) {
            const col = e.which === 37
                // если влево, то уменьшаем индекс в столбце, если вправо — увеличиваем
                ? tetromino.col - 1
                : tetromino.col + 1;
            // если так ходить можно, то запоминаем текущее положение
            if (checkMove(tetromino.matrix, tetromino.row, col)) {
                tetromino.col = col;
            }
        }

        // стрелка вверх — поворот
        if (e.which === 38) {
            // поворачиваем фигуру на 90 градусов
```

```
        const matrix = rotate(tetromino.matrix);
        // если так ходить можно — запоминаем
        if (checkMove(matrix, tetromino.row, tetromino.col)) {
            tetromino.matrix = matrix;
        }
    }

    // стрелка вниз — ускорить падение
    if(e.which === 40) {
        // смещаем фигуру на строку вниз
        const row = tetromino.row + 1;
        // если опускаться больше некуда — запоминаем новое положение
        if (!checkMove(tetromino.matrix, row, tetromino.col)) {
            tetromino.row = row - 1;
            // ставим на место и смотрим на заполненные ряды
            placeFigure();
            return;
        }
        // запоминаем строку, куда стала фигура
        tetromino.row = row;
    }
    //Пауза игры
    if (e.which === 27) {
        window.alert("ПАУЗА. Для продолжения нажмите OK.");
    }
});

// Добавляем игрока в таблицу рекордов.
function addRecord(){
    let tmp = localStorage.getItem(user_name);
```

```
    if(tmp){
        if(score > tmp){
            localStorage.removeItem(user_name);
            localStorage.setItem(user_name, score);
        }
    }else{
        localStorage.setItem(user_name, score);
    }
}


// Обновляем таблицу рекордов.
function updateRecordTable(){

    localStorage.removeItem("tetris.username");
    let record = [];
    for(let i = 0; i < localStorage.length; i++) {
        let key = localStorage.key(i);
        record.push([key, localStorage.getItem(key)]);
    }

    if(record.length >= 2){
        record.sort(function compareNumbers(a, b){
            return b[1] - a[1];
        });
    }

    for(let j = 0; j < 5; ++j){
        if(j > record.length - 1){
            break;
```

```
        }
        let position = document.getElementById((j + 1).toString());
        position.innerHTML = "№"+(j+1) + ' ' + record[j][0] + ' ' + record[j][1];
    }
}

// главный цикл игры
function loop() {
    // начинаем анимацию
    rAF = requestAnimationFrame(loop);
    // очищаем холст
    context.clearRect(0,0,canvas.width,canvas.height);

    for (let row = 0; row < 20; row++) {
        for (let col = 0; col < 10; col++) {
            if (playfield[row][col]) {
                const name = playfield[row][col];
                context.fillStyle = colors[name];
                context.fillRect(col * grid, row * grid, grid-1, grid-1);
            }
        }
    }

    // рисуем текущую фигуру
    if (tetromino) {

        if (++count > (35 - level*3)){
            tetromino.row++;
            count = 0;
```

```
            // если движение закончилось — рисуем фигуру в поле и
проверяем, можно ли удалить строки
            if (!checkMove(tetromino.matrix, tetromino.row, tetromino.col)) {
              tetromino.row--;
              placeFigure();
            }
          }
          context.fillStyle = colors[tetromino.name];
          // отрисовываем её
          for (let row = 0; row < tetromino.matrix.length; row++) {
            for (let col = 0; col < tetromino.matrix[row].length; col++) {
              if (tetromino.matrix[row][col]) {
                context.fillRect((tetromino.col + col) * grid, (tetromino.row +
row) * grid, grid-1, grid-1);
              }
            }
          }
        }

        updateRecordTable();
        rAF = requestAnimationFrame(loop);
```