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

ОТЧЕТ

по лабораторной работе № 1

по дисциплине «WEB-технологии»

Тема: Тетрис на JavaScript

Студент гр. 6304 _____ Григорьев И.С.

Преподаватель _____ Беляев С.А.

Санкт-Петербург
2018

Цель работы

Изучение работы web-сервера nginx со статическими файлами и создание клиентских JavaScript web-приложений.

Постановка задачи

Для достижения поставленной цели требуется решить следующие задачи:

1) Генерация открытого и закрытого ключей для использования шифрования.

2) Настройка сервера nginx для работы по протоколу HTTPS.

3) Разработка интерфейса web-приложения – игра в тетрис.

4) Обеспечение ввода имени пользователя.

5) Обеспечение создания новой фигуры для тетриса по таймеру и её движение.

6) Обеспечение управления пользователем падающей фигурой.

7) Обеспечение исчезновения ряда, если он заполнен.

8) По окончании игры – отображение таблицы рекордов, которая хранится в браузере пользователя.

Основные теоретические сведения

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.

Ход работы

Подготовка среды выполнения

1. Среда разработки – WebStorm.
2. Среда выполнения – Node.JS.
3. ОС – Windows.

Генерация открытого и закрытого ключей для использования шифрования

1. Установлена криптографическая библиотека OpenSSL.
2. Созданы открытый и закрытый ключи с помощью команды:

```
openssl req -x509 -nodes -days 36500 -newkey rsa:4096 -keyout
C:\Users\grigo\Desktop\nginx-1.14.0\conf\nginx.key -out
C:\Users\grigo\Desktop\nginx-1.14.0\conf\nginx.crt
```

Настройка сервера nginx для работы по протоколу HTTPS

1. Для корректной работы сервера пути до созданных файлов с ключами должны быть указаны в конфигурациях сервера *ssl_certificate* и *ssl_certificate_key*.
2. Также необходимо указать и другие настройки:

```
server {
    listen       443 ssl default_server;
    server_name  localhost;

    ssl_certificate      nginx.crt;
    ssl_certificate_key  nginx.key;

    ssl_session_cache    shared:SSL:20m;
    ssl_session_timeout  20m;


    ssl_prefer_server_ciphers  on;
    ssl_ciphers ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSAAES128-GCM-SHA256


    location / {
        root   html;
        index  index.html index.htm;
    }
}
```

Рисунок 1 – Файл nginx.conf

3. Сервер запускается командой *start nginx*. Останавливается – командой *nginx –s stop*. Доступ к разработанному приложению будет осуществляться с локального компьютера по адресу *https://localhost/*.

Разработка интерфейса пользователя

1. Создана форма входа. Форма приведена на рис. 2.

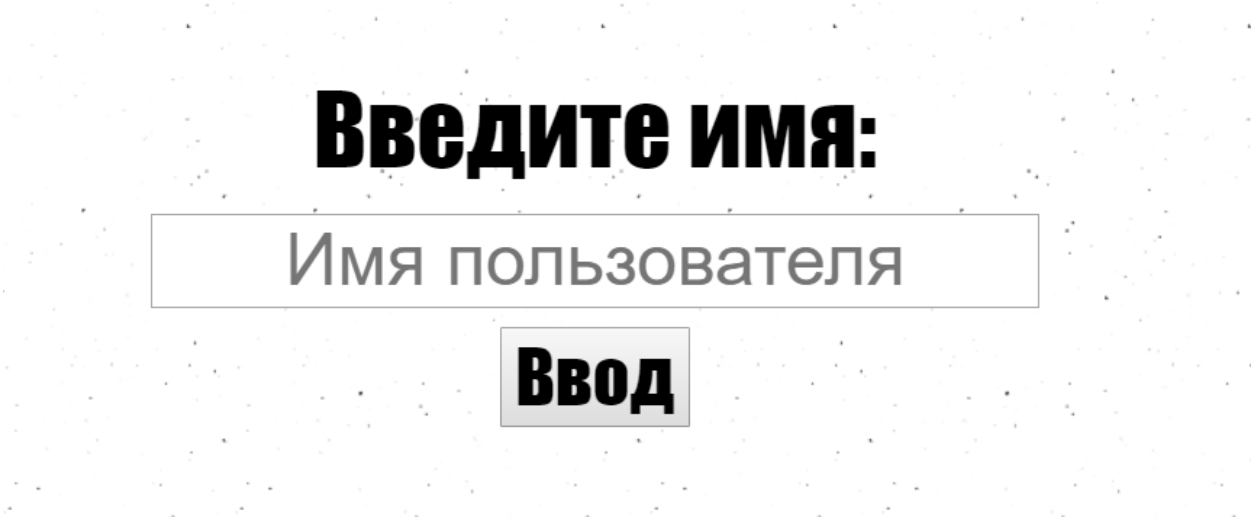

Рисунок 2 – Форма входа

2. Создана основная форма игры. Форма приведена на рис. 3.

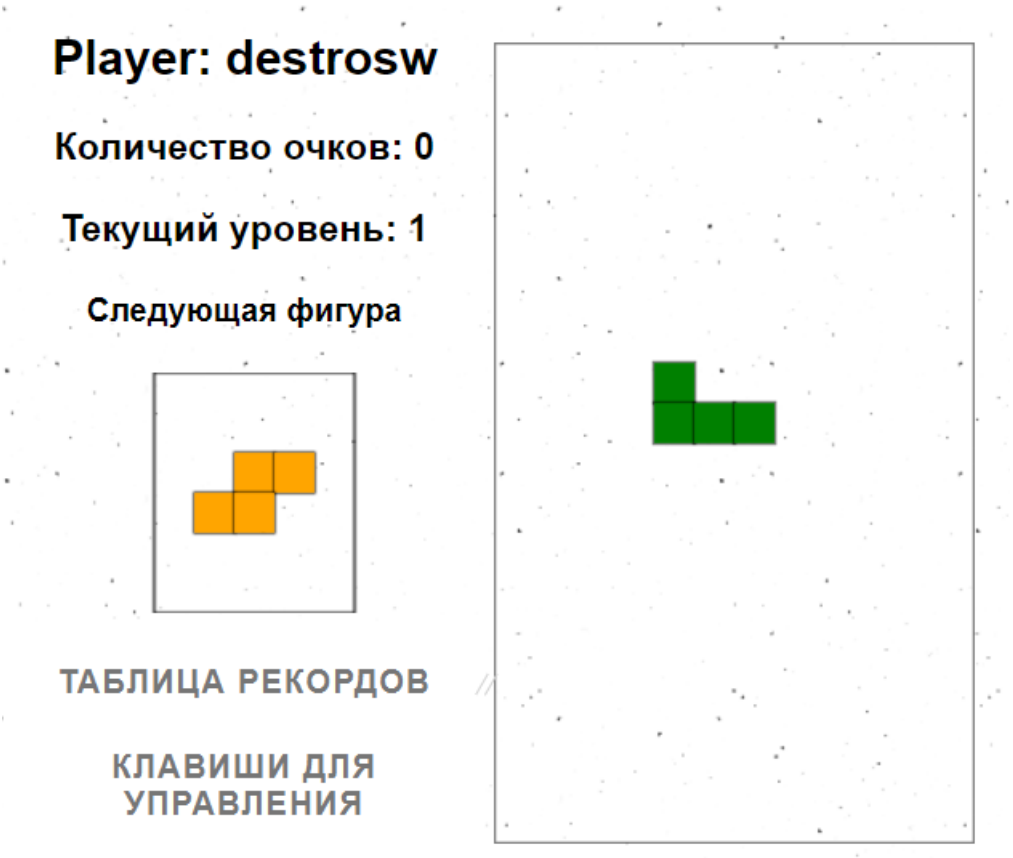

Рисунок 3 – Основная форма игры

3. При переходе на главную страницу имя пользователя с первой формы сохраняется в локальном хранилище *localStorage*. Также там сохраняются набранные рекорды пользователей для построения таблицы рекордов.
4. Создана необходимая структура проекта. Код .js .css .html представлен в приложении.

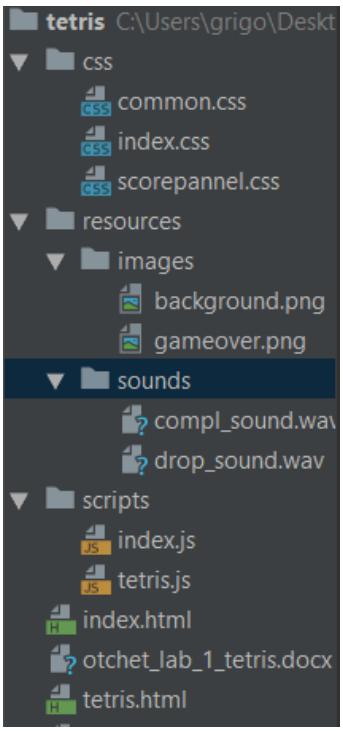

Рисунок 4 – Структура проекта

5. В скрипте tetris.js описана вся логика тетриса: создание чаши, создание фигур-тетрамино, их передвижение, повороты, накопление ряда, а также генерация следующей случайной фигуры. Также в этом файле содержится обработчик клавиш клавиатуры, отрисовка в canvas фигур, чаши и настроено перемещение фигуры по таймеру.
6. Добавлена рамка с изображением следующей тетрамины, система очков и уровней. Реализовано повышение сложности игры с каждым уровнем посредством увеличения скорости падения фигур. Добавлена таблица рекордов.

Тест приложения

1. Запущен сервер

```
C:\Users\grigo\Desktop\nginx-1.14.0>start nginx

C:\Users\grigo\Desktop\nginx-1.14.0>tasklist
```

Рисунок 4 – Запуск сервера

```
nginx.exe                    10052 Console                    5      7 888 КБ
nginx.exe                    16124 Console                    5      8 436 КБ
```

Рисунок 5 – Процессы nginx

2. Осуществлен переход по адресу https://localhost. Введено имя пользователя, в игре набрано 10 очков.

Рисунок 6 – Таблица рекордов

3. Выполнен выход обратно к стартовой форме, введено другое имя пользователя. Набрано 20 очков.

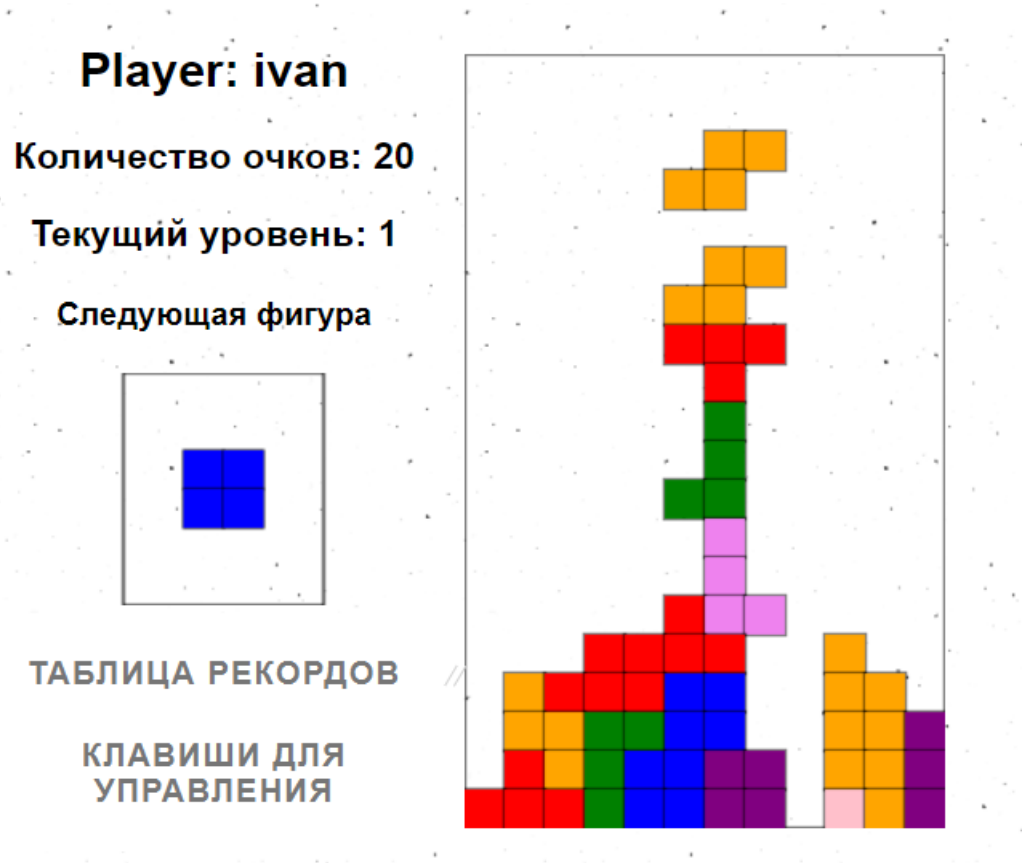

Рисунок 7 – Тетрис

Рисунок 8 – обновленная таблица рекордов

Вывод

В ходе лабораторной работы изучена работа web-сервера nginx со статическими файлами и создано клиентское JavaScript web-приложение. Сервер настраивался по протоколу HTTPS, для этого были сгенерированы открытый и закрытый ключи шифрования с помощью криптографической библиотеки OpenSSL. Таким образом, полученные теоретические знания закреплены на практике.

ПРИЛОЖЕНИЕ А
Файл index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Login</title>
</head>
<body>
    <script src="scripts/index.js"></script>
    <form action="tetris.html" method="get">
        <label>
            Введите имя:<br>
            <input id="username" placeholder="Имя пользователя"><br>
        </label>
        <input id="b1" onclick="store();" type="submit" value="Ввод">
    </form>
    <link rel="stylesheet" href="css/index.css">
</body>
</html>
```

ПРИЛОЖЕНИЕ Б
Файл tetris.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Tetris</title>
</head>
<body>
    <div id="wrap">
        <div class="left">
            <h2 align="center"><div id="player" hidden>Player:</div></h2>
            <h3 align="center"><div id="score" hidden></div></h3>
            <h3 align="center"><div id="level" hidden></div></h3>
            <h4 id="nextfsign" align="center" hidden>Следующая фигура</h4>
            <canvas id="nextf" width="200" height="120" hidden></canvas>
            <button id="startbutton" onclick="runTetris()">Старт</button>

            <nav>
                <ul class="topmenu">
                    <li><a href="">Таблица рекордов</a>
                        <ul class="submenu">
                            <div id="scoretable"></div>
                        </ul>
                    </li>
                    <li><a href="">Клавиши для управления</a>
                        <ul class="submenu">
                            <div id="keyboardkeys">
                                <h4 align="center">Клавиши для управления:</h4>
                                <ul type="square">
                                    <li><strong>'A'</strong> - движение влево</li>
                                    <li><strong>'D'</strong> - движение вправо</li>
                                    <li><strong>'S'</strong> - ускорение движения вниз</li>
                                    <li><strong>'Space'</strong> - "уронить" фигуру</li>
                                    <li><strong>'Q'</strong> - поворот фигуры влево</li>
                                    <li><strong>'E'</strong> - поворот фигуры вправо</li>
                                </ul>
                            </div>
                        </ul>
                    </li>
                </ul>
            </nav>
        </div>
        <div class="right">
            <img id="img1" width="240" height="350" src="resources/images/gameover.png"
hidden>
            <canvas id="tetris" width="240" height="400" hidden></canvas>
        </div>
    </div>
    <audio id="dropaudio" src="resources/sounds/drop_sound.wav" preload="auto"></audio>
    <audio id="complaudio" src="resources/sounds/compl_sound.wav" preload="auto"></audio>
    <link rel="stylesheet" href="css/common.css">
    <link rel="stylesheet" href="css/scorepannel.css">
    <script src="scripts/tetris.js"></script>
</body>
</html>
```

ПРИЛОЖЕНИЕ В
Файл index.js

```
function store() {
    localStorage["tetris.username"]
        = document.getElementById("username").value;
}
```

ПРИЛОЖЕНИЕ Г
Файл tetris.js

```
const canvas = document.getElementById('tetris');
var ctx = canvas.getContext('2d');

document.getElementById('player').innerText = 'Player: ' + localStorage["tetris.username"];


function arenaSweep() {
    var rowCount = 1;
    outer: for (var y = arena.length - 1; y > 0; y--) {
        for (var x = 0; x < arena[y].length; x++) {
            if (arena[y][x] === 0) {
                continue outer;
            }
        }
        playCompSound();
        const row = arena.splice(y, 1)[0].fill(0);
        arena.unshift(row);
        y++;

        player.score += rowCount * 10;
        rowCount *= 2;
    }
}

function createMatrix(w, h) {
    const matrix = [];
    while (h--) {
        matrix.push(new Array(w).fill(0));
    }
    return matrix;
}

function isBorder(arena, player) {
    const [m, o] = [player.matrix, player.pos];
    for (var y = 0; y < m.length; y++) {
        for (var x = 0; x < m[y].length; x++) {
            if (m[y][x] !== 0 &&
                (arena[y + o.y] &&
                arena[y + o.y][x + o.x]) !== 0) {
                return true;
            }
        }
    }
    return false;
}

const arena = createMatrix(12, 20);

const player = {
    pos: {x: 0, y: 0},
    matrix: null,
    score: 0,
    level: 1,
    deltasc: 50,
    dropInterval: 1000,
}

const colors = [
    'red',
    'blue',
    'violet',
    'green',
    'purple',
    'orange',
    'pink'
```

```
];


const SQ = 20;
function drawMatrix(matrix, offset, ctx) {
    matrix.forEach((row, y) => {
        row.forEach((value, x) => {
            if (value !== 0) {
                ctx.fillStyle = colors[value - 1];
                ctx.strokeStyle = 'black';
                ctx.strokeRect(SQ*(x + offset.x), SQ*(y + offset.y), SQ, SQ);
                ctx.fillRect(SQ*(x + offset.x), SQ*(y + offset.y), SQ, SQ);
            }
        });
    });
}

function merge(arena, player) {
    player.matrix.forEach((row, y) => {
        row.forEach((value, x) => {
            if (value !== 0) {
                arena[y + player.pos.y][x + player.pos.x] = value;
            }
        });
    });
}

function createPiece(type) {
    if (type === 'T') {
        return [
            [0, 0, 0],
            [1, 1, 1],
            [0, 1, 0]
        ];
    } else if (type === 'O') {
        return [
            [2, 2],
            [2, 2]
        ];
    } else if (type === 'L') {
        return [
            [0, 3, 0],
            [0, 3, 0],
            [0, 3, 3]
        ];
    } else if (type === 'J') {
        return [
            [0, 4, 0],
            [0, 4, 0],
            [4, 4, 0]
        ];
    } else if (type === 'I') {
        return [
            [0, 5, 0, 0],
            [0, 5, 0, 0],
            [0, 5, 0, 0],
            [0, 5, 0, 0]
        ];
    } else if (type === 'S') {
        return [
            [0, 6, 6],
            [6, 6, 0],
            [0, 0, 0]
        ];
    } else if (type === 'Z') {
        return [
            [7, 7, 0],
            [0, 7, 7],
            [0, 0, 0]
```

```
        ];
    }
}

function draw() {
    ctx.clearRect(0, 0, canvas.width, canvas.height);
    ctx.strokeStyle = 'black';
    ctx.strokeRect(0, 0, canvas.width, canvas.height);

    drawMatrix(arena, {x: 0, y: 0}, ctx);
    drawMatrix(player.matrix, player.pos, ctx);
}

const pieces = 'ILJOTSZ';
var tmpTime = player.dropInterval;
var next_matrix = createPiece(pieces[pieces.length * Math.random() | 0]);
function playerReset() {
    player.dropInterval = tmpTime;
    player.matrix = next_matrix;
    player.pos.y = 0;
    player.pos.x = (arena[0].length / 2 | 0) -
                   (player.matrix[0].length / 2 | 0);
    if (player.score > player.deltasc) {
        increaseLevel();
        updateLevel();
    }
    if (isBorder(arena, player)) {
        localStorage["tetris.username"] = document.getElementById('player')
                                          .innerHTML.split(' ')[1];
        isContinue = false;
        arena.forEach(row => row.fill(0));
        document.getElementById('tetris').hidden = true;
        document.getElementById('nextf').hidden = true;
        document.getElementById('nextfsign').hidden = true;
        document.getElementById('startbutton').hidden = false;
        document.getElementById('img1').hidden = false;


        var playerName = localStorage['tetris.username'];
        if (localStorage.getItem(playerName) === null) {
            storeScores();
        } else if (Number(localStorage[playerName]) < player.score) {
            storeScores();
        }
        createScoreTable();
    }

    var fig_ind = pieces[pieces.length * Math.random() | 0];
    next_matrix = createPiece(fig_ind);
    updateNextFigureArea(next_matrix, fig_ind);
}

function lvlReset() {
    player.score = 0;
    player.level = 1;
    player.deltasc = 50;
    player.dropInterval = 1000;
    tmpTime = player.dropInterval;
}

function increaseLevel() {
    player.level++;
    player.deltasc += 50;
    player.dropInterval -= 99;
}

function playerMove(direction) {
    player.pos.x += direction;
    if (isBorder(arena, player)) {
```

```
        player.pos.x -= direction;
    }
}

function playerRotate(direction) {
    rotateMatrix(player.matrix, direction);

    if (isBorder(arena, player)) {
        rotateMatrix(player.matrix, -direction);
    }

}

function playerDrop() {
    player.pos.y++;
    if (isBorder(arena, player)) {
        player.pos.y--; //player backup
        merge(arena, player);
        playerReset();
        arenaSweep();
        playDropSound();
        updateScore();
    }
    dropCounter = 0;
}

function rotateMatrix(matrix, direction) {
    for (var y = 0; y < matrix.length; y++) {
        for (var x = 0; x < y; x++) {
            [matrix[x][y], matrix[y][x]] =
                [matrix[y][x], matrix[x][y]];
        }
    }

    if (direction > 0) {
        matrix.forEach(row => row.reverse());
    } else {
        matrix.reverse();
    }
}

var dropCounter = 0;
var lastTime = 0;
var isContinue = true;
function update(time = 0) {
    if (isContinue) {
        const deltaTime = time - lastTime;
        lastTime = time;

        dropCounter += deltaTime;
        if (dropCounter > player.dropInterval) {
            playerDrop();
        }

        draw();
        requestAnimationFrame(update);
    }
}

function updateScore() {
    document.getElementById('score').innerText = 'Количество очков: ' + player.score;
}

function updateLevel() {
    document.getElementById('level').innerText = 'Текущий уровень: ' + player.level;
}

function updateNextFigureArea(figure, fig_ind) {
    const canvas = document.getElementById('nextf');
```

```
    var ctx = canvas.getContext('2d');
    ctx.clearRect(0, 0, canvas.width, canvas.height);
    if (fig_ind === 'O') {
        drawMatrix(figure, {x: 4.5, y: 2}, ctx);
    } else if (fig_ind === 'L' || fig_ind === 'J') {
        drawMatrix(figure, {x: 4, y: 1.5}, ctx);
    } else if (fig_ind === 'Z' || fig_ind === 'S') {
        drawMatrix(figure, {x: 4, y: 2}, ctx);
    } else {
        drawMatrix(figure, {x: 4, y: 1}, ctx);
    }
    ctx.strokeStyle = 'black';
    ctx.strokeRect(SQ*3, 0, SQ*5, SQ*6);
}

document.addEventListener('keydown', event => {
    if (event.keyCode === 65) {
        playerMove(-1);
    } else if (event.keyCode === 68) {
        playerMove(1);
    } else if (event.keyCode === 83) {
        playerDrop(); //w/o another drop down
    } else if (event.keyCode === 81) {
        playerRotate(-1);
    } else if (event.keyCode === 69) {
        playerRotate(1);
    } else if (event.keyCode === 32) {
        tmpTime = player.dropInterval;
        player.dropInterval = 10;
    }

})

function runTetris() {
    document.getElementById('score').hidden = false;
    document.getElementById('level').hidden = false;
    document.getElementById('nextf').hidden = false;
    document.getElementById('nextfsign').hidden = false;
    document.getElementById('tetris').hidden = false;
    document.getElementById('player').hidden = false;
    document.getElementById('startbutton').hidden = true;
    document.getElementById('img1').hidden = true;
    isContinue = true;
    lvlReset();
    playerReset();
    updateScore();
    updateLevel();
    update();
}

//saving players and their best scores in localStorage
function storeScores() {
    localStorage[localStorage['tetris.username']] = player.score;
}

function createScoreTable() {
    document.getElementById('scoretable').innerText = '';
    var playersAndScores = [];
    for (var i = 0; i < localStorage.length; i++) {
        if (localStorage.key(i) !== 'tetris.username') {
            var tmp = {};
            tmp['name'] = localStorage.key(i);
            tmp['score'] = Number(localStorage.getItem(localStorage.key(i)));
            playersAndScores.push(tmp)
        }
    }

    playersAndScores.sort(function(a, b) {
        return b.score - a.score;
```

```
    });
    playersAndScores.unshift(null);

    document.getElementById('table');
    var newTable = document.createElement('table');
    var newTitle = newTable.insertRow(0);
    newTitle.insertCell(0).innerHTML = 'Игрок';
    newTitle.insertCell(1).innerHTML = 'Количество очков';


    for (var y = 1; y < localStorage.length; y++) {
        var newRow = newTable.insertRow(y);
        for (var x = 0; x < 2 ; x++) {
            var newCell = newRow.insertCell(x);

            if (x === 0) {
                newCell.innerHTML = playersAndScores[y].name;
            } else {
                newCell.innerHTML = playersAndScores[y].score;
            }
        }
    }
    document.getElementById('scoretable').appendChild(newTable);
}

function playDropSound() {
    document.getElementById("dropaudio").play();
}

function playCompSound() {
    document.getElementById("dropaudio").pause();
    document.getElementById("complaudio").play();
}

createScoreTable();
```

ПРИЛОЖЕНИЕ Д
Файл common.css

```
* { box-sizing: border-box; }
body {
    margin: 0;
    background: url("../resources/images/background.png");
    font-family: sans-serif;
    position: absolute;
    left: 35%;
    top: 20%;
}

#wrap {
    width: 485px;
}

.left{
    width: 240px;
    height: 400px;
    float:left;
}
.right{
    width: 240px;
    height: 400px;
    float:right;
}

#tetris{
    margin-top: 25px;
}

table {
    font-family: "Lucida Sans Unicode", "Lucida Grande", Sans-Serif;
    text-align: center;
}

td {
    border: 1px solid black;
    padding: 3px;
    text-align: center;
}

#startbutton {
    text-align: center;
    height: 50px;
    width: 250px;
    border-radius: 30px;
    background: #FFFAFA;
    font-size: large;
}

#nextf {
    margin-left: 15px;
}

#img1 {
    margin-left: 15px;
    margin-bottom: 100px;
}
```

ПРИЛОЖЕНИЕ Е
Файл index.css

```
body {
    background: url("../resources/images/background.png");
    text-align: center;
    margin-top: 10%;
    font-family: Impact, Charcoal, sans-serif;
    font-size: 40pt;
}

#username {
    text-align: center;
    font-size: 30pt;
    font-family: sans-serif;
}

#b1 {
    font-size: 30pt;
    font-family: Impact, Charcoal, sans-serif;
}
```

ПРИЛОЖЕНИЕ Е
Файл scorepannel.css

```
nav { background: white; }
nav ul {
    list-style: none;
    margin: 0;
    padding: 0;
}
nav a {
    text-decoration: none;
    outline: none;
    display: block;
    transition: .4s ease-in-out;
}
.topmenu {
    text-align: center;
    padding: 10px 0;
}
.topmenu > li {
    display: inline-block;
    position: relative;
}
.topmenu > li:after {
    content: "";
    position: absolute;
    right: 0;
    width: 1px;
    height: 12px;
    background: #d2d2d2;
    top: 16px;
    box-shadow: 4px -2px 0 #d2d2d2;
    transform: rotate(30deg);
}
.topmenu > li:last-child:after {
    background: none;
    box-shadow: none;
}
.topmenu > li > a {
    padding: 12px 26px;
    color: #767676;
    text-transform: uppercase;
    font-weight: bold;
    letter-spacing: 1px;
    font-family: 'Exo 2', sans-serif;
}
.topmenu li a:hover { color: #c0a97a; }
.submenu {
    position: absolute;
    left: 50%;
    top: 100%;
    width: 210px;
    margin-left: -105px;
    background: #fafafa;
    border: 1px solid #ededed;
    z-index: 5;
    visibility: hidden;
    opacity: 0;
    transform: scale(.8);
    transition: .4s ease-in-out;
}
.submenu li a {
    padding: 10px 0;
    margin: 0 10px;
    border-bottom: 1px solid #efefef;
    font-size: 12px;
    color: #484848;
    font-family: 'Kurale', serif;
```

```
}
.topmenu > li:hover .submenu {
    visibility: visible;
    opacity: 1;
    transform: scale(1);
}
```