

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. 9383

Лапина А.А.

Преподаватель

Беляев С.А.

Санкт-Петербург

2021

Цель работы.

Целью работы является изучение работы web-сервера nginx со статическими файлами и создание клиентских JavaScript web-приложений.

Задачи.

- генерация открытого и закрытого ключей для использования шифрования (<https://www.openssl.org/>);
- настройка сервера nginx для работы по протоколу HTTPS;
- разработка интерфейса web-приложения;
- обеспечение ввода имени пользователя;
- обеспечение создания новой фигуры для тетриса по таймеру и ее движение;
- обеспечение управления пользователем падающей фигурой;
- обеспечение исчезновения ряда, если он заполнен;
- по окончании игры – отображение таблицы рекордов, которая хранится в браузере пользователя.

Необязательно: оформление с использованием CSS.

Выполнение работы.

Были использованы следующие переменные:

var **canvas**, **figure** — для связи HTML-кода и основного игрового поля, поля, отражающего следующую фигуру соответственно;

var **ctx**, **ctx_figure** - для отрисовки основного поля и поля, где изображена следующая фигура;

var **xm** = 10 — количество клеток на игровом поле по горизонтали;

var **ym** = 15 - количество клеток на игровом поле по вертикали;

var **N** = 30 — ширина одной клетки в пикселях;

var **Time** — время падения фигуры, которое будет уменьшаться при увеличении уровня;

var **matrix** = матрица, хранящая информацию о заполнении поля (0 — если

клетка свободна и цифры от 1 до 10 в зависимости от цвета, если клетка несвободна;

var **active_figure** - массив, хранящий координаты по x и y каждой клетки активной фигуры, а также тип фигуры;

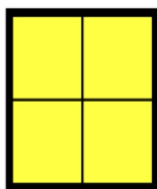
Номер элемента	0	1	2	3	4	5	6	7	8
Что хранит	x1	y1	x2	y2	x3	y3	x4	y4	Тип фигуры

x1..x4 — значения по x для 1..4 клеток соответственно

y1..y4 — значения по y для 1..4 клеток соответственно

Было разработано 7 типов фигур, используемых в тетрисе:

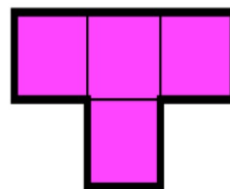
1)



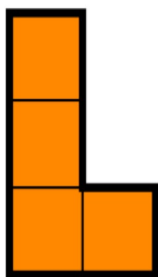
2)



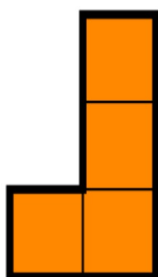
3)



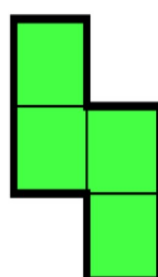
4)



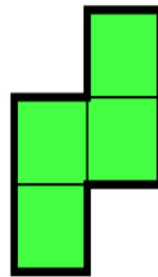
5)



6)



7)



var **rgb_active_figure** - массив, хранящий значения r, g, b и номер цвета активной фигуры;

Номер элемента	0	1	2	3
Что хранит	r	g	b	Номер цвета

var **next_figure** - хранит описание следующей фигуры, аналогично, **active_figure**;

var **rgb_next_figure** - хранит описание цветов следующей фигуры, аналогично,

rgb_active_figure ;

var ***matrix_figure*** — матрица для изображения следующей фигуры;

var ***r, g, b*** — значения цвета red, green, blue;

var ***flag*** — флаг для определения конца игры (false, если стакан заполнен);

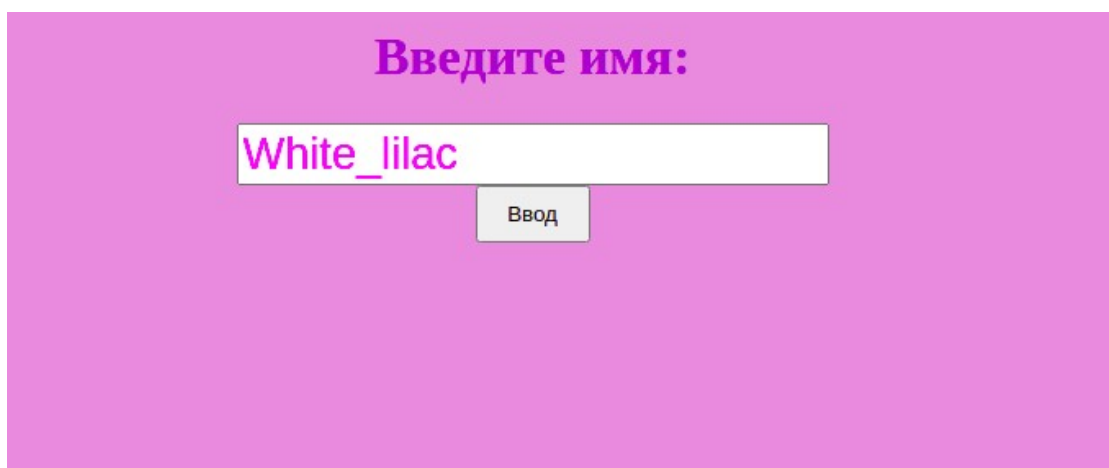
var ***level*** - переменная, отвечающая за подсчет уровня;

Были реализованы следующие функции:

- 1) color(col) — задает значения r, g, b для переданного цвета;
- 2) figure1, ..., figure7 - для описания фигур;
- 3) changeMatrix() - меняет матрицу, при появлении новой фигуры;
- 4) turnF2(), ..., turnF7() - обеспечивают повороты фигур, меняют координаты матрицы;
- 5) getRandomIntInclusive(min, max) — возвращает рандомное значение от min до max включительно;
- 6) game() - запускает игру, заполняет матрицу для хранения информации о состоянии игрового поля и матрицу о следующей фигуре нулями;
- 7) startGame() - обеспечивает задание основных переменных, вызывает функции для отрисовки поля, взаимодействия пользователя с игрой, движение фигур;
- 8) liveFigure() - функция, реализующая «жизнь» фигуры, в ней проверяется условие окончания игры, вывод таблицы рекордов;
- 9) init() - инициализирует имя пользователя и уровень, связывает javascript-код и HTML-код;
- 10) drawNet(ctx1, xmax, ymax) — рисует сетку на полях;
- 11) drawNextFigure() - отрисовывает следующую фигуру;
- 12) square1(col, x, y) — отрисовывает квадрат на поле следующей фигуры;
- 13) new_figure() - рандомно выбирает следующую фигуру;
- 14) checkStr() - проверяет можно ли убрать строку и уменьшает время, если можно;
- 15) canCleanStr(n) — условие для проверки заполненности строки;

- 16) `cleanStr(n)` — очистка строки;
- 17) `canGoDown()` - проверка условия, что можно двигать фигуру вниз;
- 18) `square(col, x, y)` - отрисовывает квадрат на игровом поле;
- 19) `go()` - движение фигуры вниз на одну клетку;
- 20) `CanGoRight()` - условие для проверки возможности двигать фигуру вправо;
- 21) `CanGoLeft()` - условие для проверки возможности двигать фигуру влево;
- 22) `moving_figure()` - взаимодействие пользователя с игрой, обеспечение перемещения фигуры вправо, влево, поворота и падения, при нажатии клавиш.
- 23) `store()` - функция, достающая введенное пользователем имя в окне ввода;
- 24) `read()` - возвращает имя пользователя;
- 25) `setUsername()` - функция для сохранения имени игрока и отображения его в поле ввода при повторном входе;
- 26) `saveRecord(level)` — сохраняет рекорды игроков;
- 27) `getRecords()` - возвращает записи о рекордах.

В файле `entry.html` — описана начальная страница для входа пользователя



Введите имя:

White_lilac

Ввод

Рисунок 1 — Демонстрация начальной страницы

В файле `main.html` — описана основная страница с игрой

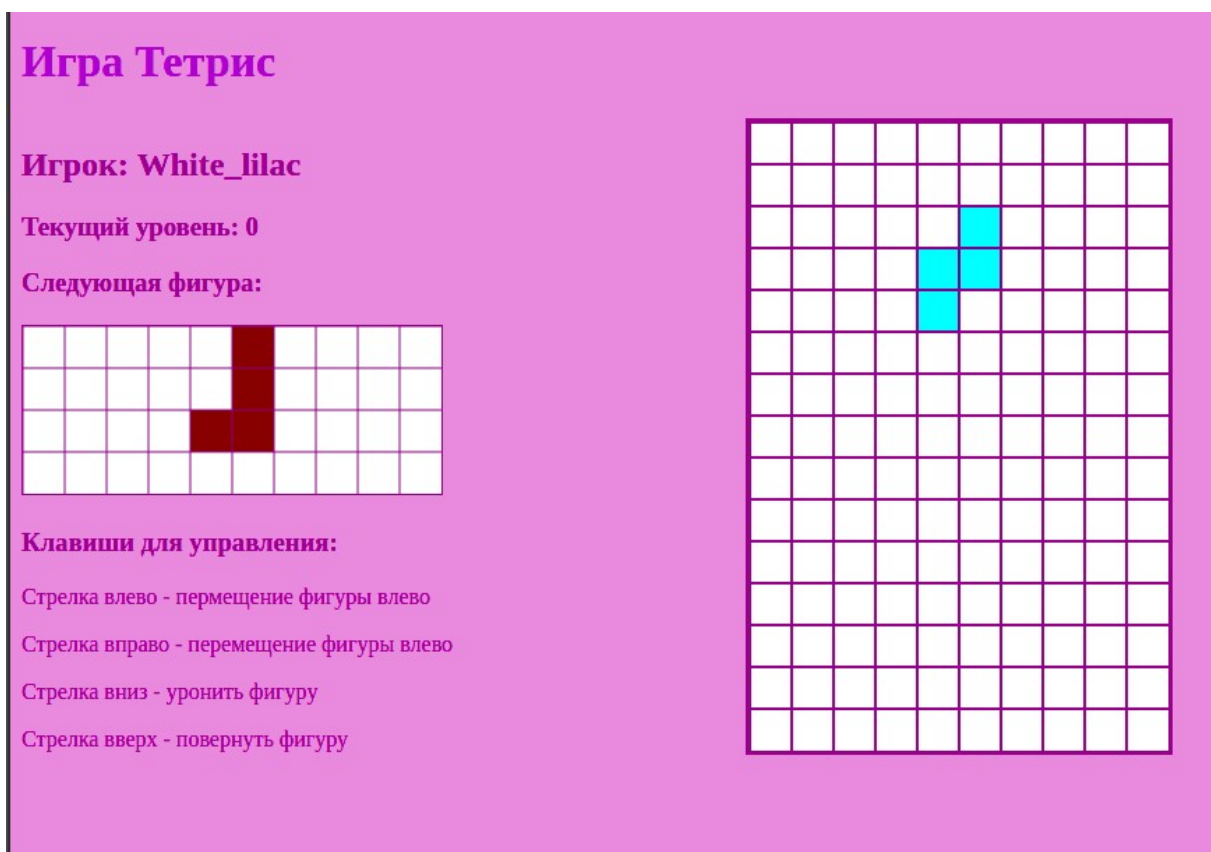


Рисунок 2 — Основная страница

В файле style.css разработан дизайн основной страницы и всплывающего модального окна с результатами.

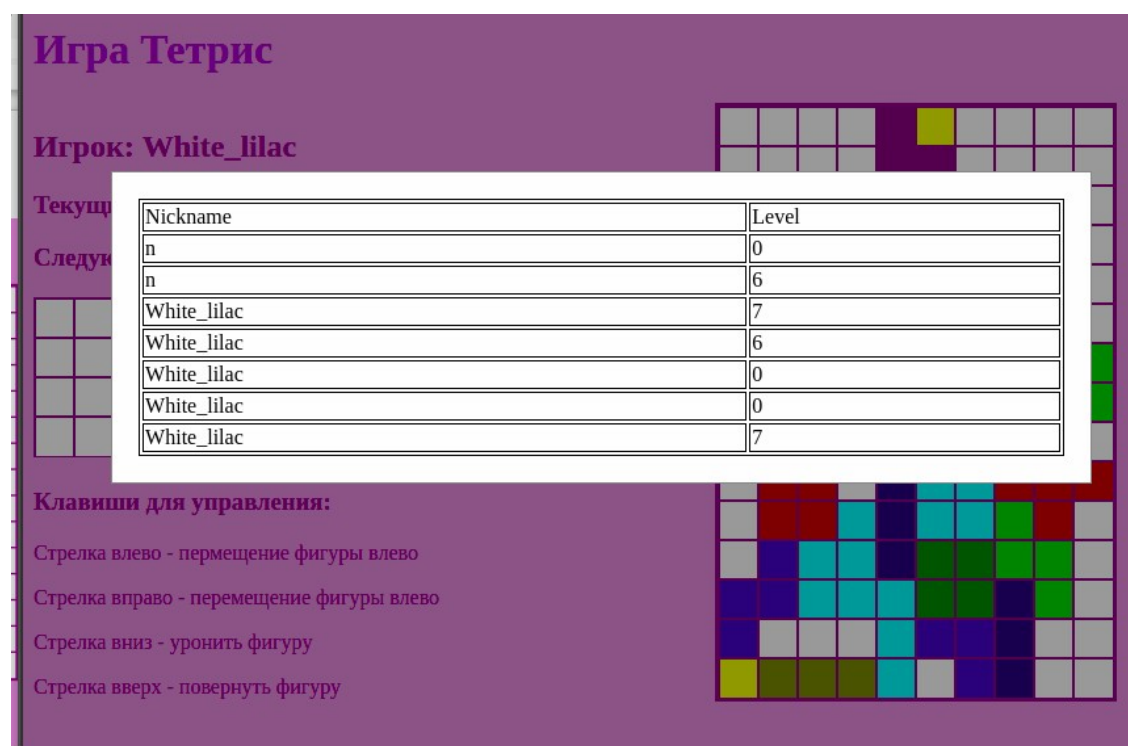


Рисунок 3 — Всплывающее окно — таблица результатов

Разработанный программный код см. в приложении А.

Выводы.

Была изучена работа web-сервера nginx со статическими файлами и создано клиентское web-приложение на JavaScript .

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: Figures.js

```
function color(col){
  switch (col){
    case 1:
      r = 200;
      g = 0;
      b = 0;
      break;

    case 2: //lime
      r = 0;
      g = 200;
      b = 0;
      break;

    case 3:
      r = 0;
      g = 0;
      b = 200;
      break;

    case 4: //yellow
      r = 255;
      g = 255;
      b = 0;
      break;

    case 5: //aqua
      r = 0;
      g = 255;
      b = 255;
      break;

    case 6: //purple
      r = 128;
      g = 0;
      b = 128;
      break;

    case 7: //green
      r = 0;
      g = 128;
      b = 0;
      break;

    case 8: //olive
      r = 128;
      g = 128;
      b = 0;
      break;

    case 9: //maroon (red)
      r = 128;
```



```

        g = 0;
        b = 0;
        break;

    case 10: //navy (blue)
        r = 0;
        g = 0;
        b = 128;
        break;

    default:
        r = 255;
        g = 255;
        b = 255;
        break;
    }
}

function figure1(col){
    next_figure.push(Math.round(xm/2));
    next_figure.push(0);
    next_figure.push(Math.round(xm/2) + 1);
    next_figure.push(0);
    next_figure.push(Math.round(xm/2));
    next_figure.push(1);
    next_figure.push(Math.round(xm/2) + 1);
    next_figure.push(1);
    next_figure.push(1);
    color(col);
    rgb_next_figure.push(r);
    rgb_next_figure.push(g);
    rgb_next_figure.push(b);
    rgb_next_figure.push(col);
}

function figure2(col){
    next_figure.push(Math.round(xm/2) - 2);
    next_figure.push(0);
    next_figure.push(Math.round(xm/2) - 1);
    next_figure.push(0);
    next_figure.push(Math.round(xm/2));
    next_figure.push(0);
    next_figure.push(Math.round(xm/2) + 1);
    next_figure.push(0);
    next_figure.push(2);
    color(col);
    rgb_next_figure.push(r);
    rgb_next_figure.push(g);
    rgb_next_figure.push(b);
    rgb_next_figure.push(col);
}

function figure3(col){
    next_figure.push(Math.round(xm/2) - 1);
    next_figure.push(0);
    next_figure.push(Math.round(xm/2));
    next_figure.push(0);
    next_figure.push(Math.round(xm/2) + 1);

```

```

    next_figure.push(0);
    next_figure.push(Math.round(xm/2));
    next_figure.push(1);
    next_figure.push(3); //figure number
    color(col);
    rgb_next_figure.push(r);
    rgb_next_figure.push(g);
    rgb_next_figure.push(b);
    rgb_next_figure.push(col);
}

function figure4(col) {
    next_figure.push(Math.round(xm / 2) - 1);
    next_figure.push(0);
    next_figure.push(Math.round(xm / 2) - 1);
    next_figure.push(1);
    next_figure.push(Math.round(xm / 2) - 1);
    next_figure.push(2);
    next_figure.push(Math.round(xm / 2));
    next_figure.push(2);
    next_figure.push(4);
    color(col);
    rgb_next_figure.push(r);
    rgb_next_figure.push(g);
    rgb_next_figure.push(b);
    rgb_next_figure.push(col);
}

function figure5(col){
    next_figure.push(Math.round(xm/2));
    next_figure.push(0);
    next_figure.push(Math.round(xm/2));
    next_figure.push(1);
    next_figure.push(Math.round(xm/2));
    next_figure.push(2);
    next_figure.push(Math.round(xm/2) - 1);
    next_figure.push(2);
    next_figure.push(5);
    color(col);
    rgb_next_figure.push(r);
    rgb_next_figure.push(g);
    rgb_next_figure.push(b);
    rgb_next_figure.push(col);
}

function figure6(col){
    next_figure.push(Math.round(xm/2)-1);
    next_figure.push(0);
    next_figure.push(Math.round(xm/2)-1);
    next_figure.push(1);
    next_figure.push(Math.round(xm/2));
    next_figure.push(1);
    next_figure.push(Math.round(xm/2));
    next_figure.push(2);
    next_figure.push(6);
    color(col);
    rgb_next_figure.push(r);
    rgb_next_figure.push(g);

```

```

        rgb_next_figure.push(b);
        rgb_next_figure.push(col);
    }

function figure7(col){
    next_figure.push(Math.round(xm/2));
    next_figure.push(0);
    next_figure.push(Math.round(xm/2));
    next_figure.push(1);
    next_figure.push(Math.round(xm/2)-1);
    next_figure.push(1);
    next_figure.push(Math.round(xm/2)-1);
    next_figure.push(2);
    next_figure.push(7);
    color(col);
    rgb_next_figure.push(r);
    rgb_next_figure.push(g);
    rgb_next_figure.push(b);
    rgb_next_figure.push(col);
}

function changeMatrix(){
    matrix[active_figure[0]][active_figure[1]] = rgb_active_figure[3];
    matrix[active_figure[2]][active_figure[3]] = rgb_active_figure[3];
    matrix[active_figure[4]][active_figure[5]] = rgb_active_figure[3];
    matrix[active_figure[6]][active_figure[7]] = rgb_active_figure[3];
}

```

Название файла: Caretaker.h

```

#ifndef CARETAKER_H
#define CARETAKER_H

#include <stack>
#include <sstream>
#include "../GameManager/GameManager.h"
#include "../Memento/Memento.h"
#include "../SaveFile/SaveFile.h"

class Caretaker {
private:
    SaveFile* saveFile;
    GameManager* gameManager;

public:
    Caretaker(GameManager* newGameManager);
    ~Caretaker();

    void save();
    void undo();
};

#endif

```

Название файла: turn_figures.js

```

function turnF2(){
  for (let i = 0; i < 4; i++) {
    matrix[active_figure[i * 2]][active_figure[i * 2 + 1]] = 0;
  }

  if (active_figure[0] === active_figure[2]) {
    if(active_figure[0]<xm-1 && active_figure[2]<xm-1 &&
active_figure[4]>xm<1 && active_figure[6]<xm-1
    && active_figure[0]>1 && active_figure[2]>1 &&
active_figure[4]>1 && active_figure[6]>1) {
      for (let i = 0; i < 4; i++) {
        active_figure[i * 2 + 1] = active_figure[3]
      }
      active_figure[0] = active_figure[0] - 2;
      active_figure[2]--;
      active_figure[6]++;
    }
  } else {
    for (let i = 0; i < 4; i++) {
      active_figure[i * 2] = active_figure[4];
    }
    active_figure[1]--;
    active_figure[5]++;
    active_figure[7] = active_figure[7] + 2;
  }
  for (let i = 0; i < 4; i++) {
    matrix[active_figure[i * 2]][active_figure[i * 2 + 1]] =
rgb_active_figure[3];
  }
}

```

```

function turnF3(){
  for (let i = 0; i < 4; i++) {
    matrix[active_figure[i * 2]][active_figure[i * 2 + 1]] = 0;
  }
  if(active_figure[0] === active_figure[2]){
    if(active_figure[2]<active_figure[6]) {
      active_figure[0]--;
      active_figure[1]++;
      active_figure[4]++;
      active_figure[5]--;
      active_figure[6]--;
      active_figure[7]--;
    }
    else{
      active_figure[0]--;
      active_figure[1]++;
      active_figure[4]++;
      active_figure[5]--;
      active_figure[6]++;
      active_figure[7]++;
    }
  }
  else{
    active_figure[0]++;
    active_figure[1]--;
  }
}

```

```

    active_figure[4]--;
    active_figure[5]++;

    if(active_figure[3]<active_figure[7]) {
        active_figure[6]++;
        active_figure[7]--;
    }
    else{
        active_figure[6]--;
        active_figure[7]++;
    }
}
for (let i = 0; i < 4; i++) {
    matrix[active_figure[i * 2]][active_figure[i * 2 + 1]] =
rgb_active_figure[3];
}
}

function turnF4(){
    for (let i = 0; i < 4; i++) {
        matrix[active_figure[i * 2]][active_figure[i * 2 + 1]] = 0;
    }
    if(active_figure[0] === active_figure[2]){
        if(active_figure[4]<active_figure[6]) {
            if(active_figure[6]<xm-1) {
                active_figure[1]++;
                active_figure[2]++;
                active_figure[4] = active_figure[4] + 2;
                active_figure[5]--;
                active_figure[6]++;
                active_figure[7] = active_figure[7] - 2;
            }
        }
        else{
            if(active_figure[0]<xm-1) {
                active_figure[0]++;
                active_figure[1]--;
                active_figure[4]--;
                active_figure[5]++;
                active_figure[7] = active_figure[7] + 2;
            }
        }
    }
}
else{
    if(active_figure[5]>active_figure[7]) {
        active_figure[0]++;
        active_figure[1]++;
        active_figure[4]--;
        active_figure[5]--;
        active_figure[6] = active_figure[6] - 2;
    }
    else{
        active_figure[0]--;
        active_figure[1]--;
        active_figure[4]++;
    }
}
}

```

```

        active_figure[5]++;
        active_figure[6] = active_figure[6] + 2;
    }
}
for (let i = 0; i < 4; i++) {
    matrix[active_figure[i * 2]][active_figure[i * 2 + 1]] =
rgb_active_figure[3];
}
}

function turnF5(){
    for (let i = 0; i < 4; i++) {
        matrix[active_figure[i * 2]][active_figure[i * 2 + 1]] = 0;
    }
    if(active_figure[0] === active_figure[2]){

        if(active_figure[4]>active_figure[6]) {
            if(active_figure[0]<xm-1) {
                active_figure[0]--;
                active_figure[1]++;
                active_figure[4]++;
                active_figure[5]--;
                active_figure[6] = active_figure[6] + 2;
            }
        }
        else{
            if(active_figure[0]>0) {
                active_figure[0]++;
                active_figure[1]--;
                active_figure[4]--;
                active_figure[5]++;
                active_figure[6] = active_figure[6] - 2;
            }
        }
    }
    else{

        if(active_figure[5]<active_figure[7]) {
            active_figure[0]++;
            active_figure[1]++;
            active_figure[4]--;
            active_figure[5]--;
            active_figure[7] = active_figure[7] - 2;
        }
        else{
            active_figure[0]--;
            active_figure[1]--;
            active_figure[4]++;
            active_figure[5]++;
            active_figure[7] = active_figure[7] + 2;
        }
    }
    for (let i = 0; i < 4; i++) {
        matrix[active_figure[i * 2]][active_figure[i * 2 + 1]] =
rgb_active_figure[3];
    }
}

```

```

function turnF6(){
  for (let i = 0; i < 4; i++) {
    matrix[active_figure[i * 2]][active_figure[i * 2 + 1]] = 0;
  }
  if(active_figure[0] === active_figure[2]){
    if(active_figure[4]<xm-1){
      active_figure[1]++;
      active_figure[2]++;
      active_figure[5]--;
      active_figure[6]++;
      active_figure[7] = active_figure[7] - 2;
    }
  }
  else{
    active_figure[1]--;
    active_figure[2]--;
    active_figure[5]++;
    active_figure[6]--;
    active_figure[7] = active_figure[7] + 2;
  }
  for (let i = 0; i < 4; i++) {
    matrix[active_figure[i * 2]][active_figure[i * 2 + 1]] =
rgb_active_figure[3];
  }
}

```

```

function turnF7(){
  for (let i = 0; i < 4; i++) {
    matrix[active_figure[i * 2]][active_figure[i * 2 + 1]] = 0;
  }
  if(active_figure[0] === active_figure[2]){
    if(active_figure[0]<xm-1){
      active_figure[0]--;
      active_figure[3]--;
      active_figure[4]++;
      active_figure[6] = active_figure[6] + 2;
      active_figure[7]--;
    }
  }
  else{
    active_figure[0]++;
    active_figure[1]--;
    active_figure[4]--;
    active_figure[5]--;
    active_figure[6] = active_figure[6] - 2;
  }
  for (let i = 0; i < 4; i++) {
    matrix[active_figure[i * 2]][active_figure[i * 2 + 1]] =
rgb_active_figure[3];
  }
}

```

Название файла: safe_user_name.js

```

function store() {
  localStorage["tetris.username"] = document.getElementById("input-
field").value;
}
function read() {

```

```

    let username = localStorage["tetris.username"];
    if (username === undefined) {
        return "";
    }
    return username;
}
function setUsername() {
    let inputField = document.getElementById("input-field");
    console.log(read());
    inputField.value = read();
}

function saveRecord(level){
    if(localStorage["records"] === undefined || localStorage["records"]
=== null){
        localStorage["records"] = JSON.stringify([]);
    }

    const nickname = read();

    const records = JSON.parse(localStorage["records"]);

    records.push({nickname: nickname, level: level});

    localStorage["records"] = JSON.stringify(records);
}

function getRecords() {
    if(localStorage["records"] === undefined || localStorage["records"]
=== null){
        return [];
    }

    return JSON.parse(localStorage["records"]);
}

```

Название файла: tetris.js

```

var canvas, figure;
var ctx, ctx_figure;
var xm = 10;
var ym = 15;
var N = 30;

var Time;

var matrix = new Array(10);
for (let i = 0; i < matrix.length; i++) {
    matrix[i] = new Array(15);
}
var active_figure = [];
var rgb_active_figure = [];
var next_figure = [];
var rgb_next_figure = [];

var matrix_figure = new Array(10);
for (let i = 0; i < matrix_figure.length; i++) {

```



```

    matrix_figure[i] = new Array(4);
}
var r, g, b;
var flag = true;
var level = 0;

function getRandomIntInclusive(min, max) {
    min = Math.ceil(min);
    max = Math.floor(max);
    return Math.floor(Math.random() * (max - min + 1)) + min; //Максимум
и МИНИМУМ ВКЛЮЧАЮТСЯ
}

function game(){
    flag = true;
    level = 0;
    init();
    for (let i = 0; i<xm; i++){
        for (let j = 0; j<ym; j++){
            matrix[i][j] = 0;
        }
    }

    for (let i = 0; i<xm; i++){
        for (let j = 0; j<4; j++){
            matrix_figure[i][j] = 0;
        }
    }
    console.log(matrix);
    canvas = document.getElementById('canvas_game');
    ctx = canvas.getContext('2d');
    ctx.clearRect(0, 0, canvas.width, canvas.height);
    startGame();
}

function init() {
    const playerName = document.getElementById("player-name");
    playerName.innerText = read();
    let levelPlayer = document.getElementById("level");
    levelPlayer.innerText = level;
}

function drawNet(ctx1, xmax, ymax){

    for (let x = 0; x < xmax*N; x += N) {
        ctx1.moveTo(x, 0);
        ctx1.lineTo(x, ymax*N);
    }

    for (let y = 0; y < ymax*N; y += N) {
        ctx1.moveTo(0, y);
        ctx1.lineTo(xmax*N, y);
    }

    ctx1.strokeStyle = "#800080";
    ctx1.stroke();
}

```

```

function drawNextFigure(){
    for (let i = 0; i<xm; i++){
        for (let j = 0; j<4; j++){
            matrix_figure[i][j] = 0;
        }
    }
    figure = document.getElementById('canvas_figure');
    ctx_figure = figure.getContext('2d');
    ctx_figure.clearRect(0, 0, figure.width, figure.height);
    for (let i = 0; i < xm; i++) {
        for (let j = 0; j < 4; j++) {
            if ((i === next_figure[0] && j === next_figure[1]) || (i ===
next_figure[2] && j === next_figure[3])
                || (i === next_figure[4] && j === next_figure[5]) || (i
=== next_figure[6] && j === next_figure[7])){
                matrix_figure[i][j] = next_figure[8];
                square1(matrix_figure[i][j], i * N, j * N);
            }
        }
    }
    drawNet(ctx_figure, xm, 4);
}

function square1(col, x, y) {
    ctx_figure.fillStyle = `rgba(${rgb_next_figure[0]}, $
{rgb_next_figure[1]}, ${rgb_next_figure[2]}, 1)`;
    ctx_figure.fillRect(x, y, N, N);
}

function new_figure(){
    switch (getRandomIntInclusive(1, 7)) {
        case 1:
            figure1(getRandomIntInclusive(1, 10));
            break;
        case 2:
            figure2(getRandomIntInclusive(1, 10));
            break;
        case 3:
            figure3(getRandomIntInclusive(1, 10));
            break;
        case 4:
            figure4(getRandomIntInclusive(1, 10));
            break;
        case 5:
            figure5(getRandomIntInclusive(1, 10));
            break;
        case 6:
            figure6(getRandomIntInclusive(1, 10));
            break;
        default:
            figure7(getRandomIntInclusive(1, 10));
            break;
    }
}

function checkStr(){
    for(let n = 0; n<ym; n++) {
        if (canCleanStr(n)) {

```

```

        if (Time>200){
            Time = Time - 100;
        }
        else{
            if(Time>50) {
                Time = Time - 10;
            }
        }
        cleanStr(n);
        for (let i = 0; i < xm; i++) {
            for (let j = 0; j < ym; j++) {
                square(matrix[i][j], i * N, j * N);
            }
        }
        level++;
        console.log("level = ", level);
        init();
    }
}

function canCleanStr(n){
    let count = 0;
    for (let i = 0; i<xm; i++){
        if(matrix[i][n] != 0)
            count++;
    }
    if(count===xm)
        return true;
    else
        return false;
}

function cleanStr(n){
    for(let j = 0; j < n; j++){
        for(let i = 0; i < xm; i++){
            matrix[i][n - j] = matrix[i][n - 1 - j]
        }
    }

    for(let i = 0; i < xm - 1; i++){
        matrix[i][0] = 0;
    }
}

function startGame(){
    Time = 1000;
    new_figure();
    active_figure = next_figure.slice();
    rgb_active_figure = rgb_next_figure.slice();
    changeMatrix();
    next_figure.splice(0, next_figure.length);
    rgb_next_figure.splice(0, rgb_next_figure.length);
    new_figure();
    console.log(active_figure);
    console.log(next_figure);
    drawNextFigure();
    for (let i = 0; i < xm; i++) {

```

```

        for (let j = 0; j < ym; j++) {
            square(matrix[i][j], i * N, j * N);
        }
    }
    drawNet(ctx, xm, ym);
    moving_figure();
    console.log(matrix);
    console.log("rgb = ", rgb_active_figure);
    flag = true;
    let timerId = setTimeout(function tick() {
        liveFigure();
        timerId = setTimeout(tick, Time);
    }, 1000);
}

function liveFigure(){
    if(flag) {
        if (canGoDown()) {
            go();
        } else {
            checkStr();
            console.log("Time = ", Time);
            if (active_figure[1] === 0 || active_figure[3] === 0 ||
active_figure[5] === 0 || active_figure[7] === 0) {
                flag = false;
                console.log("Вы проиграли!");
                alert("Вы проиграли!");
                saveRecord(level);
                let array = getRecords();
                console.log(array);
                const modalTable = document.getElementById("modal-
table");
                modalTable.style.display = "block";
                const recordsTable = document.getElementById("records-
table");
                recordsTable.innerHTML =
"<tr><td>Nickname</td><td>Level</td></tr>";
                let resultsStr = "";
                for (let el of array) {
                    resultsStr += `<tr><td>${el.nickname}</td><td>${
{el.level}</td></tr>`;
                }
                recordsTable.innerHTML += resultsStr;
            }
            active_figure = next_figure.slice();
            rgb_active_figure = rgb_next_figure.slice();
            changeMatrix();
            next_figure.splice(0, next_figure.length);
            rgb_next_figure.splice(0, rgb_next_figure.length);
            new_figure();
            drawNextFigure();
            for (let i = 0; i < xm; i++) {
                for (let j = 0; j < ym; j++) {
                    square(matrix[i][j], i * N, j * N);
                }
            }
            drawNet(ctx, xm, ym);
        }
    }
}

```

```

    }
}

function canGoDown() {
    if ((active_figure[1] >= ym - 1) || (active_figure[3] >= ym - 1) ||
        (active_figure[5] >= ym - 1) || (active_figure[7] >= ym - 1))
        return false; //дошли до конца
    if (((matrix[active_figure[0]][active_figure[1] + 1] === 0) ||
        ((active_figure[0] === active_figure[2]) && ((active_figure[1] + 1) ===
        active_figure[3])) ||
        ((active_figure[0] === active_figure[4]) &&
        ((active_figure[1] + 1) === active_figure[5])) || ((active_figure[0] ===
        active_figure[6]) && ((active_figure[1] + 1) === active_figure[7]))))

        && ((matrix[active_figure[2]][active_figure[3] + 1] === 0) ||
        ((active_figure[2] === active_figure[0]) && ((active_figure[3] + 1) ===
        active_figure[1])) ||
        ((active_figure[2] === active_figure[4]) &&
        ((active_figure[3] + 1) === active_figure[5])) || ((active_figure[2] ===
        active_figure[6]) && ((active_figure[3] + 1) === active_figure[7]))))

        && ((matrix[active_figure[4]][active_figure[5] + 1] === 0) ||
        (active_figure[4] === active_figure[0] && active_figure[5] + 1 ===
        active_figure[1])) ||
        (active_figure[4] === active_figure[2] && active_figure[5] + 1
        === active_figure[3])) || (active_figure[4] === active_figure[6] &&
        active_figure[5] + 1 === active_figure[7]))

        && ((matrix[active_figure[6]][active_figure[7] + 1] === 0) ||
        ((active_figure[6] === active_figure[0]) && ((active_figure[7] + 1) ===
        active_figure[1])) ||
        ((active_figure[6] === active_figure[2]) && ((active_figure[7] +
        1) === active_figure[3])) || ((active_figure[6] === active_figure[4]) &&
        ((active_figure[7] + 1) === active_figure[5]))))

        return true;
    else
        return false;
}

function square(col, x, y){
    ctx.clearRect((active_figure[x])*N, (active_figure[y])*N, N, N);
    color(col);
    ctx.fillStyle = `rgba(${r}, ${g}, ${b}, 1)`;
    ctx.fillRect(x, y, N, N);
}

function go(){
    for(let i = 0; i<4; i++){
        matrix[active_figure[i*2]][active_figure[i*2+1]] = 0;
        active_figure[i*2+1]++;
    }
    for (let i = 0; i<4; i++) {
        matrix[active_figure[i*2]][active_figure[i*2+1]] =
rgb_active_figure[3];
    }
    for(let i = 0; i<xm; i++){

```

```

        for (let j = 0; j<ym; j++){
            square(matrix[i][j], i*N, j*N);
        }
    }
    drawNet(ctx, xm, ym);
}

function CanGoRight(){
    if(((matrix[active_figure[0] + 1][active_figure[1]] === 0) ||
    (((active_figure[0]+1) === active_figure[2]) && (active_figure[1] ===
    active_figure[3]))
        ||(((active_figure[0]+1) === active_figure[4]) &&
    (active_figure[1] === active_figure[5]))
        ||(((active_figure[0]+1) === active_figure[6]) &&
    (active_figure[1] === active_figure[7]))))
        && ((matrix[active_figure[2] + 1][active_figure[3]] === 0) ||
    (((active_figure[2]+1) === active_figure[0]) && (active_figure[3] ===
    active_figure[1]))
        ||(((active_figure[2]+1) === active_figure[4]) &&
    (active_figure[3] === active_figure[5]))
        ||(((active_figure[2]+1) === active_figure[6]) &&
    (active_figure[3] === active_figure[7]))))
        && ((matrix[active_figure[4] + 1][active_figure[5]] === 0) ||
    (((active_figure[4]+1) === active_figure[2]) && (active_figure[5] ===
    active_figure[3]))
        ||(((active_figure[4]+1) === active_figure[0]) &&
    (active_figure[5] === active_figure[1]))
        ||(((active_figure[4]+1) === active_figure[6]) &&
    (active_figure[5] === active_figure[7]))))
        && ((matrix[active_figure[6] + 1][active_figure[7]] === 0) ||
    (((active_figure[6]+1) === active_figure[2]) && (active_figure[7] ===
    active_figure[3]))
        ||(((active_figure[6]+1) === active_figure[4]) &&
    (active_figure[7] === active_figure[5]))
        ||(((active_figure[6]+1) === active_figure[0]) &&
    (active_figure[1] === active_figure[7]))))
        return true;
    else
        return false;
}

function CanGoLeft(){
    if(((matrix[active_figure[0] - 1][active_figure[1]] === 0) ||
    (((active_figure[0]-1) === active_figure[2]) && (active_figure[1] ===
    active_figure[3]))
        ||(((active_figure[0]-1) === active_figure[4]) &&
    (active_figure[1] === active_figure[5]))
        ||(((active_figure[0]-1) === active_figure[6]) &&
    (active_figure[1] === active_figure[7]))))
        && ((matrix[active_figure[2] - 1][active_figure[3]] === 0) ||
    (((active_figure[2]-1) === active_figure[0]) && (active_figure[3] ===
    active_figure[1]))
        ||(((active_figure[2]-1) === active_figure[4]) &&
    (active_figure[3] === active_figure[5]))

```

```

        ||(((active_figure[2]-1) === active_figure[6]) &&
(active_figure[3] === active_figure[7])))

        && ((matrix[active_figure[4] - 1][active_figure[5]] === 0) ||
(((active_figure[4]-1) === active_figure[2]) && (active_figure[5] ===
active_figure[3])))

        ||(((active_figure[4]-1) === active_figure[0]) &&
(active_figure[5] === active_figure[1])))
        ||(((active_figure[4]-1) === active_figure[6]) &&
(active_figure[5] === active_figure[7])))

        && ((matrix[active_figure[6] - 1][active_figure[7]] === 0) ||
(((active_figure[6]-1) === active_figure[2]) && (active_figure[7] ===
active_figure[3])))

        ||(((active_figure[6]-1) === active_figure[4]) &&
(active_figure[7] === active_figure[5])))
        ||(((active_figure[6]-1) === active_figure[0]) &&
(active_figure[1] === active_figure[7]))))
    return true;
    else
        return false;
}

function moving_figure(){
    document.addEventListener( 'keydown', (event) => {
        const keyName = event.key;
        console.log( ' Событие keydown: ' + keyName) ;
        if(keyName === "ArrowRight" && (active_figure[0]<xm-1 &&
active_figure[2]<xm-1 && active_figure[4]<xm-1 && active_figure[6]<xm-1)
&&(active_figure[1]<ym-1 && active_figure[3]<ym-1 &&
active_figure[5]<ym-1 && active_figure[7]<ym-1) && CanGoRight){

            for (let i = 0; i<4; i++) {
                active_figure[i*2]++;
            }

            for(let i = 0; i<4; i++){
                matrix[active_figure[i*2] - 1][active_figure[i*2+1]] =
0;
            }
            for(let i = 0; i<4; i++){
                matrix[active_figure[i*2]][active_figure[i*2+1]] =
rgb_active_figure[3];
            }
            console.log(active_figure);
            for(let i = 0; i<xm; i++){
                for (let j = 0; j<ym; j++){
                    square(matrix[i][j], i*N, j*N);
                }
            }
        }

        if(keyName === "ArrowLeft" && (active_figure[0]!==0 &&
active_figure[2]!==0 && active_figure[4]!==0 && active_figure[6]!==0)
&& (active_figure[1]!==ym-1 && active_figure[3]!==ym-1 &&
active_figure[5]!==ym-1 && active_figure[7]!==ym-1) && CanGoLeft()){

            for(let i = 0; i<4; i++){

```

```

        active_figure[i*2]--;
    }

    for(let i = 0; i<4; i++){
        matrix[active_figure[i*2] + 1][active_figure[i*2+1]] =
0;
    }

    for(let i = 0; i<4; i++){
        matrix[active_figure[i*2]][active_figure[i*2+1]] =
rgb_active_figure[3];
    }

    console.log(active_figure);
    for(let i = 0; i<xm; i++){
        for (let j = 0; j<ym; j++){
            square(matrix[i][j], i*N, j*N);
        }
    }
}

if(keyName === "ArrowUp" && active_figure[8]>1){
    if((active_figure[1]<ym-2 && active_figure[3]<ym-2 &&
active_figure[5]<ym-2 && active_figure[7]<ym-2)
        && (active_figure[1]>0 && active_figure[3]>0 &&
active_figure[5]>0 && active_figure[7]>0)){
        switch (active_figure[8]){
            case 2:
                turnF2();
                break;
            case 3:
                turnF3();
                break;
            case 4:
                turnF4();
                break;
            case 5:
                turnF5();
                break;
            case 6:
                turnF6();
                break;
            case 7:
                turnF7();
                break;
        }
        console.log(matrix);
        for (let i = 0; i < xm; i++) {
            for (let j = 0; j < ym; j++) {
                square(matrix[i][j], i * N, j * N);
            }
        }
    }
}

if(keyName === "ArrowDown" && canGoDown()){

```



```

        while ((active_figure[1]!=ym-1 || active_figure[3]!=ym-1 ||
active_figure[5]!=ym-1|| active_figure[6]!=ym-1) && canGoDown()){
            go();
        }
    }
    drawNet(ctx, xm, ym);
}
}
}

```

Название файла: main.html

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Игра Тетрис</title>
    <link href="style.css" rel="stylesheet" type="text/css">
</head>
<body onload="game()">
<div id="modal-table" class="modal">
    <div class="modal-content">
        <table id="records-table" border="1">
        </table>
    </div>
</div>

<h1> Игра Тетрис </h1>
<div class="wrap">
    <div class = "left">
        <div id="main_div">
            <h2> Игрок: <div id = "player-name"> </div> </h2>
            <h3><p class="info" id="tetris_level"> Текущий уровень: <div id = "level"> </div>
</p></h3>
            <label>
                <h3><p>Следующая фигура:</p></h3>
                <div id = "canvas_div2">
                    <canvas id="canvas_figure" width="300px" height="120px"> </canvas>
                </div>
            </label>

            <h3>Клавиши для управления:</h3>
            <p>Стрелка влево - перемещение фигуры влево</p>
            <p>Стрелка вправо - перемещение фигуры влево</p>
            <p>Стрелка вниз - уронить фигуру</p>
            <p>Стрелка вверх - повернуть фигуру</p>
        </div>
    </div>
    <div class="right">
        <div id = "canvas_div">
            <canvas id="canvas_game" width="300" height="450"></canvas>
        </div>
    </div>
</div>

```

```

<script src="tetris.js"></script>
<script src="safe_user_name.js"></script>
<script src="figures.js"></script>
<script src="turn_figures.js"></script>
</body>
</html>

```

Название файла: entry.html

```

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>Вход</title>
    <link href="style.css" rel="stylesheet" type="text/css">
  </head>
  <body onload="setUsername()">
    <div class = "entry">
      <form action="main.html" method="get">
        <label>
          <h1>Введите имя: <br> </h1>
          <input placeholder="Имя пользователя" id="input-field"><br>
        </label>
          <input type="submit" style = "width: 70px; height: 35px"
value="Ввод" onclick="store()">
        </form>
      </div>
      <script src="safe_user_name.js"></script>
    </body>
  </html>

```

Название файла: style.css

```

#canvas_game {
  border: solid;
}

#canvas_figure {
  border: 1px solid;
  background-color: white;
}

html {
  background-color: plum;
}

.wrap {
  width: 100%;
  color: darkmagenta;
}

#player-name, #level, #tetris_level {

```

```
    display: inline;
}
```

```
.left {
    float:left;
    width: 60%;
}
```

```
.right {
    float: right;
    width: 40%;
}
```

```
h1{
    color: darkorchid;
}
```

```
.modal {
    display: none;
    position: fixed;
    z-index: 1;
    left: 0;
    top: 0;
    width: 100%;
    height: 100%;
    overflow: auto;
    background-color: rgb(0,0,0);
    background-color: rgba(0,0,0,0.4);
}
```

```
.modal-content {
    background-color: #fefefe;
    margin: 15% auto;
    padding: 20px;
    border: 1px solid #888;
    width: 80%;
}
```

```
#records-table {
    border: 1px solid black;
    width: 100%;
}
```

```
#input-field{
    color: magenta;
```

```
    font-size: 1.7em;
}
```

```
.entry{
    text-align: center;
}
```