**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**

|  |  |  |
| --- | --- | --- |
| Студент гр. 0303 |  | Калмак Д.А. |
| Преподаватель |  | Беляев С.А. |

Санкт-Петербург

2022

## **Цель работы.**

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

## **Задание.**

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

* сервер – nginx, протокол взаимодействия – HTTPS версии не ниже 2.0;
* отображается страница для ввода имени пользователя с использованием HTML-элементов <input>;
* статическая страница отображает «стакан» для тетриса с использованием HTML-элемента, элемент 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»;
* пользователь имеет возможность двигать фигуру влево и вправо, повернуть на 90° 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;

## **Выполнение работы.**

Были сгенерированы открытый и закрытый ключи для использования шифрования и протокола взаимодействия https. (см. рис. 1)

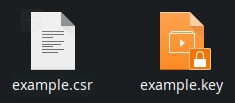


Рисунок 1 – Открытый и закрытый ключи

Сервер nginx был настроен на два режима работы: http (80) и https (443).

Создана форма входа. С помощью <form action="main.html" method="get"> после ввода имени пользователя в <input placeholder="Имя пользователя"> и нажатия <input type="submit"> происходит переход на основную форму, при этом сохраняется имя пользователя. С помощью функции store() имя помещается в локальное хранилище. (см. рис. 2)



Рисунок 2 – Форма входа

Создана основная форма. С помощью <canvas> созданы игровой «стакан» и окно с информацией об имени пользователя, текущем уровне и очках, с помощью <div> создан класс div\_table для показа следующей фигуры, который содержит класс строк div\_row и класс клеток div\_cell. Элементы <button> позволили создать кнопки начала игры и показа рекордов. (см. рис. 3)

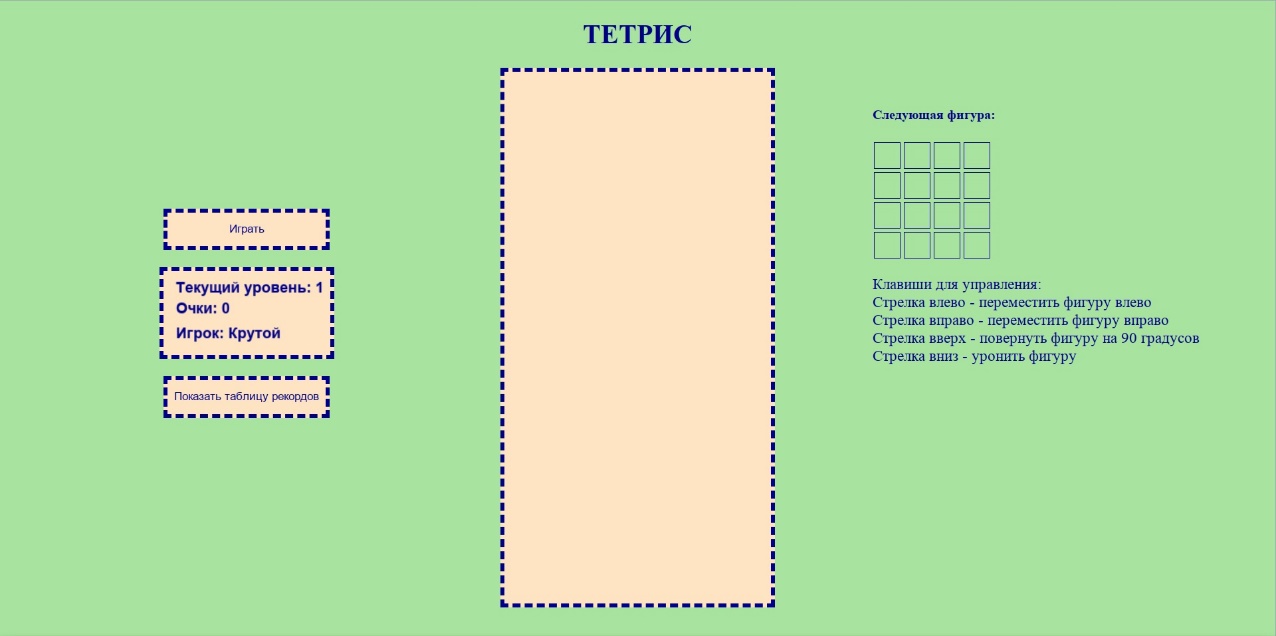


Рисунок 3 – Основная форма

Создан массив figures\_mas, который содержит названия фигур тетрамино. Создан словарь figures для соответствия фигуры и ее матрицы. Словарь colors для соответствия фигуры и ее цвета. num\_score – очки, level – уровень, name – имя пользователя, size\_square – размер квадрата фигуры, indent – расстояние между квадратами, start\_speed – начальная скорость падения фигуры, cadr – параметр падения. Функция show\_info() отрисовывает текущий уровень, очки, имя игрока. Создан массив поля tetris\_field, его высота height и его ширина width. Функция clear\_field() заполняет массив нулями. Функция choose\_figure() выбирает фигуру случайным образом с помощью Math.random(). Функция rotate\_figure() осуществляет поворот матрицы фигуры на 90 градусов. Массив queue\_figures содержит очередь фигур. Если он пустой, то с помощью цикла for и функции choose\_figure он заполняется. Массив copy\_queue является копией массива очереди фигур, который необходим для получения информации о следующей фигуре. field\_doc – HTMLCollection для управления элементами показа следующей фигуры. Функция show\_next\_figure() получает информацию о следующей фигуре и отрисовывает ее, заменяя backgroundColor клеток класса div\_cell на цвет следующей фигуры. Функция get\_figure() возвращает новую фигуру new\_figure и информацию о ней. Методом pop() из массива очереди фигур достается фигура, если после этого массив очереди фигур пустой, он заполняется новыми фигурами. Если фигура не является первой в этой игре, то вызывается функция show\_next\_figure(). Функция get\_figure() возвращает new\_figure – новую фигуру, new\_matr – матрицу новой фигуры (из соответствия словаря figures), new\_color – цвет новой фигуры (из соответствия словаря colors), new\_x – координату x новой фигуры, new\_y – координату y новой фигуры. first\_flag – флаг первой фигуры в игре, сначала true. figure\_on\_the\_field – текущая управляемая фигура, которая получает значение выполнения функции get\_figure(). Функция clear\_window() очищает <canvas> «стакана». Функция show\_records() получает из локального хранилища информацию о рекордах, сортирует игроков по очкам по убыванию и выводит информацию таблицы рекордов. Функция save\_record() сохраняет информацию о рекордах игроков. Если игрок сыграл первый раз, то о нем записывается информация в локальное хранилище: имя игрока и его очки, - иначе записывается информация лучшей попытки игрока. Функция check() проверяет фигуру на размещение на поле. Если клетка фигуры выходит за границы по горизонтали, или по вертикали снизу, или клетка на поле занята уже другой фигурой, то возвращается false, иначе true. Функция game\_over() обрабатывает конец игры. Выводится сообщение ‘Game over!’ в консоль. Очищается массив поля, отменяется анимация, game\_process принимает значение null, обрабатывается сохранение результата игры игрока, обнуляются очки, возвращается первый уровень, очищается <canvas> «стакана», показывается таблица рекордов, очищается показ следующей фигуры. check\_end – флаг конца игры. Функция isThere() для проверки ряда в функции processing(). Функция processing() для обработки поля и текущей фигуры. Проверяется текущая фигура на выход за границу поля сверху. Если да, то флаг конца игры check\_end становится true и вызывается функция конца игры game\_over(), иначе текущая фигура записывается в массив поля. Происходит проверка заполненного ряда снизу вверх с помощью метода every(isThere). Если ряд заполнен, то меняется число очков игрока, ряды выше перемещаются вниз, иначе производится проверка ряда выше. Вызывается новая фигура с помощью функции get\_figure() и значение получает figure\_on\_the\_field. С помощью *document*.addEventListener('keydown', function(event) обрабатывается управление фигурой на поле. Если игра закончена, то обработка не производится. key получает код нажатой клавиши. Если стрелки влево или вправо, то обрабатывается движение по горизонтали. Если была нажата клавиша стрелки влево, то x\_control получает значение x текущей фигуры figure\_on\_the\_field.new\_x – 1, иначе x\_control получает значение x текущей фигуры figure\_on\_the\_field.new\_x + 1. Производится проверка перемещения с помощью функции check(), и если перемещение возможно, то координата x фигуры figure\_on\_the\_field.new\_x меняется на x\_control. Если была нажата клавиша стрелки вверх, то обрабатывается поворот фигуры на 90 градусов, new\_matr получает матрицу текущей фигуры, повернутой на 90 градусов с помощью функции rotate\_figure(). Производится проверка поворота фигуры с помощью функции check(), и если поворот фигуры возможен, то матрица фигуры figure\_on\_the\_field.new\_matr меняется на new\_matr. Если была нажата клавиша стрелки вниз, то обрабатывается падение фигуры. Флаг flag устанавливается в true, и циклом while, пока flag true, производится поиск координаты y фигуры, соответствующей самой нижней из возможных на поле, с помощью функции check(). После нахождения такой координаты flag переводится в false, происходит обработка поля с помощью функции processing(). Функция game() обрабатывается игровой процесс. Начинается анимация, очищается <canvas> «стакана» с помощью функции clear\_window(), отрисовывается поле, отрисовывается информация о текущем уровне, очках, имени пользователя, обрабатывается свободное падение фигуры со скоростью, зависимой от уровня, отрисовывается текущая фигура. Функция start() управляет началом игры. Флаг первой фигуры first\_flag переводится в false, показывается следующая фигура, запускается анимация игры, check\_end переводится в false.

## **Выводы.**

Таким образом, была изучена работы web-сервера nginx со статическими файлами, сгенерированы открытый и закрытый ключи, настроен сервер nginx с работой на протоколе http и https, изучено создание клиентских JavaScript web-приложений, создан тетрис со страницей для ввода имени пользователя, страницей, отображающей «стакан» для тетриса с использованием HTML-элемента <canvas>, элемент <div> используется для отображения следующей фигуры, отображается уровень, очки, имя пользователя. Фигура тетрамино случайно генерируется и начинает падать в «стакан», пользователь может ей управлять. Целые строки исчезают, при наборе заданного числа очков увеличивается уровень, что заключается в увеличении скорости игры. Когда стакан заполняется, пользователь проигрывает, после чего ему отображается локальная таблица рекордов.