**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. 1303 |  | Чубан Д.В. |
| --- | --- | --- |
| Преподаватель |  | Беляев С. А. |

Санкт-Петербург

2023

## Цель работы.

Целью работы является изучение работы web-сервера nginx со статиче-

скими файлами и создание клиентских JavaScript web-приложений.

Для достижения поставленной цели требуется решить следующие 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 хранит-

ся в браузере пользователя.

## Выполнение работы.

В файле *init.js* описана реализация стартового окна с выбором имени игрока, проверки его длины, чтобы она не превышала 10 символов и сохранения имени в *localStorage.* В соответствующем файле *init.html* описан внешний вид окна с выбором имени, которое представляет собой текст “Введите имя”, поле для ввода имени и кнопку подтверждения ввода.

Реализация игры находится в файле *main.js*. Сначала задаются константные значения размера ячейки, матричного представления тетрамино, возможных цветов фигур, распределения перехода на новый уровень в зависимости от счета и зависимость скорости падения фигур в зависимости от уровня.

Далее инициализируются переменные счетчика одновременно “уничтоженных” линий, игрового поля, счета, уровня, скорости падения фигур и имени пользователя, которое получается через *localStorage*.

Функция *nextfig()* создает следующую фигуру, выбирая ее случайным образом из возможных вариантов тетрамино.

Функция *newfig()* размещает на поле созданную с помощью *nextfig()* фигуру.

Функция *rotate()* описывает принцип поворота падающей фигуры с помощью транспонирования матрицы.

Функция *canmove()* проверяет возможность смещения фигуры на поле.

Функция *finishgame()* предназначена для завершения игры. Она записывает итоговый счет и имя игрока в таблицу лидеров.

Функция *draw()* предназначена для отрисовки падающей фигуры.

Функция *setfig()* проверяет, может ли фигура передвинуться ниже. Если движение вниз больше невозможно, происходит проверка на заполненность линии. Если линия поля была полностью заполнена, то она удаляется, а игроку начисляются баллы. Также происходит проверка количества удаленных линий: если их количество превышает одну, то игрок также получает бонусные баллы. Если же фигура остановилась выше верхней границы поля, то игра завершится.

Функция *automove()* отвечает за автоматическое падение фигуры вниз.

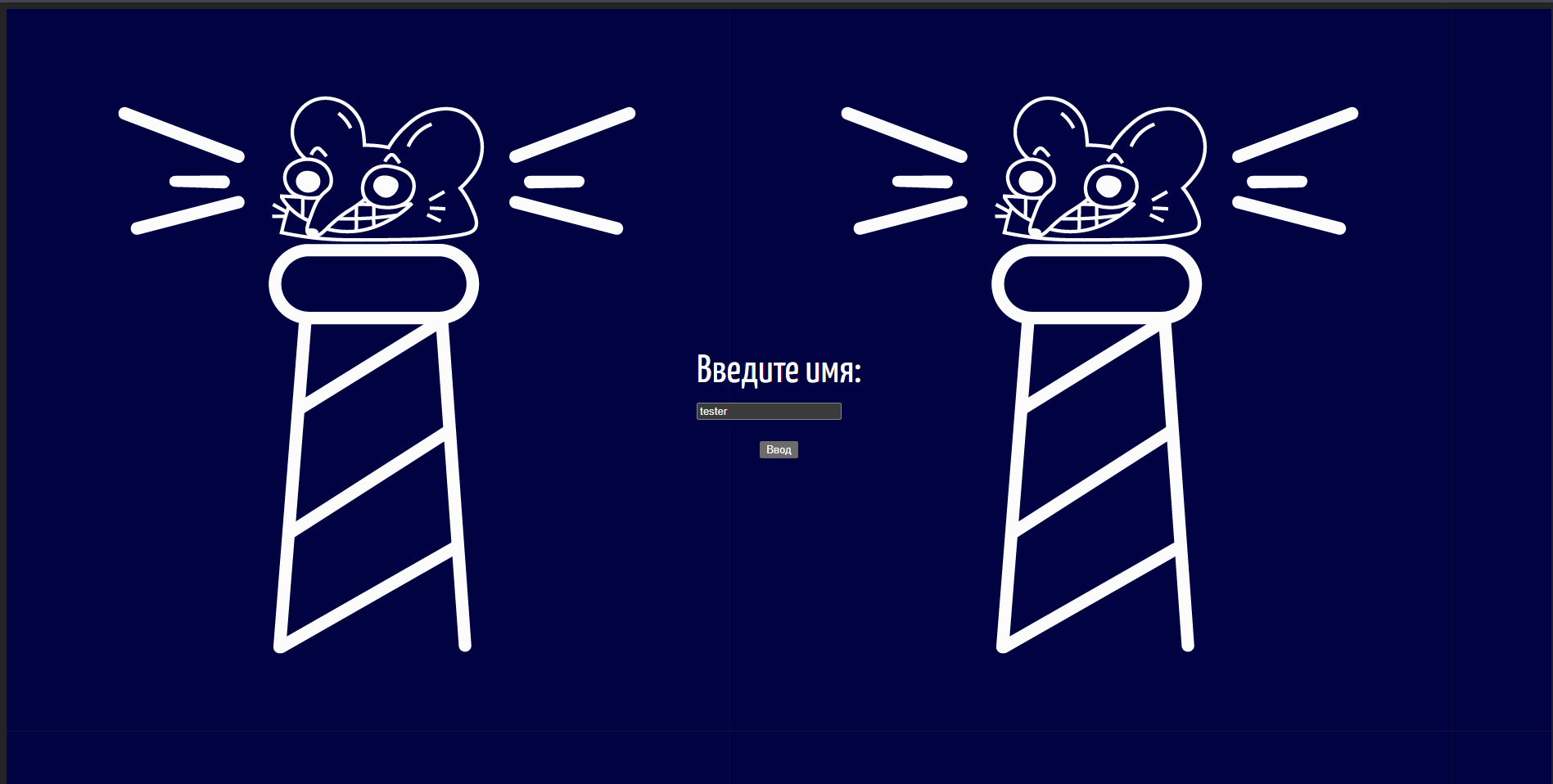
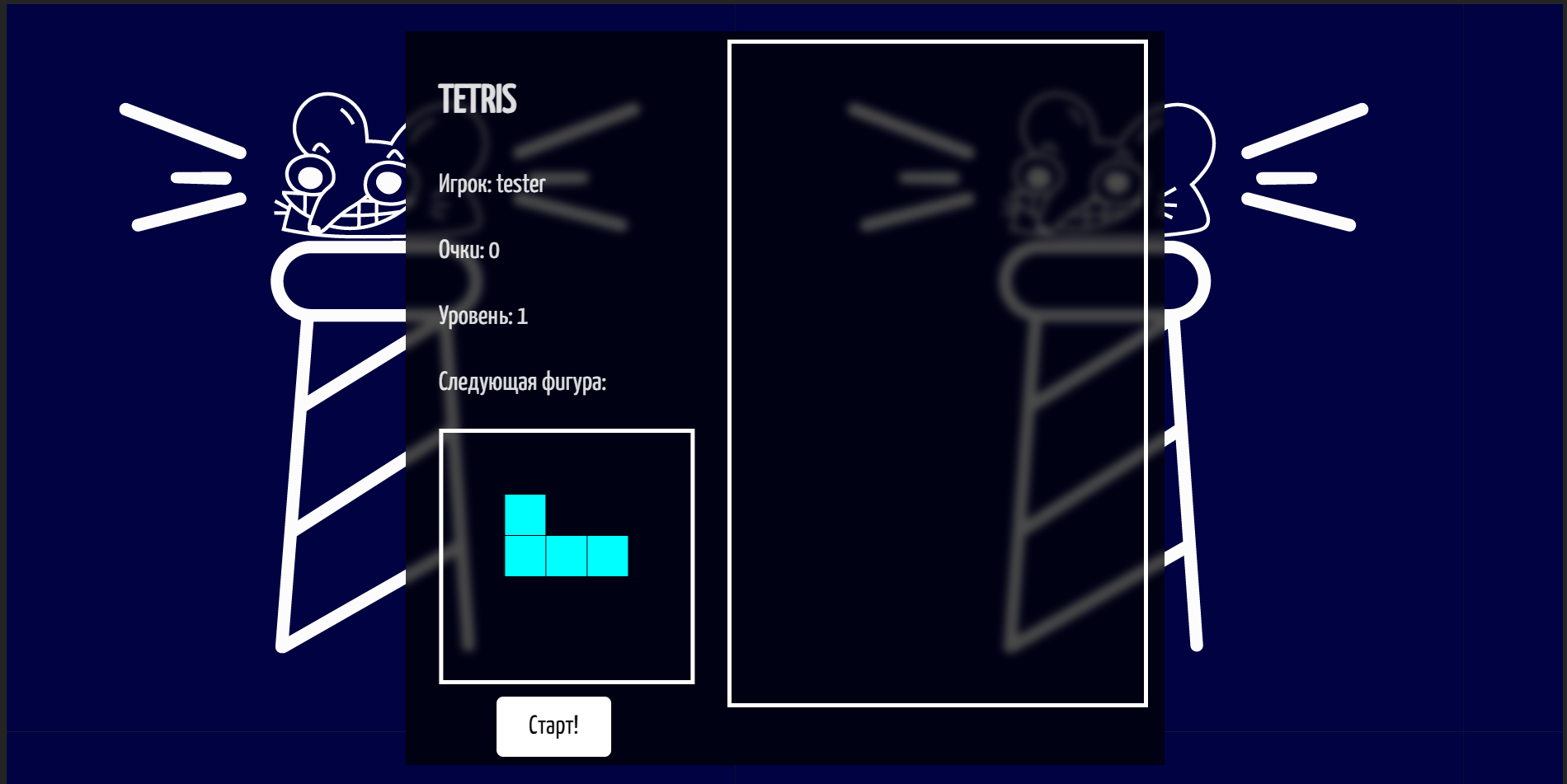
В *document.addEventListener()* задается управление на клавиши клавиатуры.

Рисунок 1 – Ввод имени игрока.

Рисунок 2 – Игровой интерфейс.

# 

Рисунок 3 – Окончание игры и таблица лидеров.

# Приложение А Исходный код

**main.js:**

const CELL = 50

const fig = {

'A': [

[0,0,0,0],

[1,1,1,1],

[0,0,0,0],

[0,0,0,0]

],

'B': [

[1,0,0],

[1,1,1],

[0,0,0],

],

'C': [

[0,0,1],

[1,1,1],

[0,0,0],

],

'D': [

[1,1],

[1,1],

],

'E': [

[0,1,1],

[1,1,0],

[0,0,0],

],

'F': [

[1,1,0],

[0,1,1],

[0,0,0],

],

'G': [

[0,1,0],

[1,1,1],

[0,0,0],

],

};

const color = {

'A': 'red',

'B': 'cyan',

'C': 'green',

'D': 'purple',

'E': 'blue',

'F': 'pink',

'G': 'orange'

};

const scores = {

1:100,

2:300,

3:500,

4:800,

5:1000,

6:1200,

7:1300,

8:1500

}

const levels = {

1: 600,

2: 570,

3: 500,

4: 470,

5: 400,

6: 370,

7: 320,

8: 175

};

let counter = 0;

let map = [];

let score = 0;

let level = 1;

let finish = false;

let interval;

let username = localStorage["gamer\_name"];

const field = document.getElementById('field');

const ctx = field.getContext('2d');

const lead\_table = document.getElementById("table");

lead\_table.close();

for (let y = -2; y < 16; y++) {

map[y] = [];

for (let x = 0; x < 10; x++) {

map[y][x] = 0;

}

}

function randInt(first, last){

return Math.floor(Math.random() \* (last-first+1)) + first;

}

function nextfig(){

const figures = ['A', 'B', 'C', 'E', 'F', 'D', 'G'];

const num = randInt(0, 6);

const currfig = figures[num]

const next = document.getElementById('next');

const ctxn = next.getContext('2d');

ctxn.clearRect(0,0,next.width,next.height);

for(let yn = 0; yn < fig[currfig].length; yn++){

for(xn = 0; xn < fig[currfig][0].length; xn++){

if(fig[currfig][yn][xn] != 0) {

ctxn.fillStyle = color[currfig]

ctxn.fillRect((300 - 50\*fig[currfig].length)/2 + xn \* CELL, (300 - 50\*fig[currfig][0].length)/2 + yn \* CELL, CELL - 1, CELL - 1)

}

}

}

return{

name: currfig,

matrix: fig[currfig]

};

}

let next\_fig = nextfig()

function newfig(){

const nm = next\_fig.name

const mtr = next\_fig.matrix

const column = map[0].length/2 - Math.ceil(mtr[0].length/2);

const row = nm === 'A' ? -1 : -2;

next\_fig = nextfig()

return {

name: nm,

matrix: mtr,

row: row,

col: column

};

}

let cur\_fig = newfig()

function rotate(matrix){

let result = [];

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

result[i] = [];

for (let j = 0; j < matrix[0].length; j++) {

result[i][j] = matrix[j][matrix.length - i - 1];

}

}

return result;

}

function canmove(matrix, x, y){

for(let ty = 0; ty < matrix.length; ty++){

for(let tx = 0; tx < matrix[0].length; tx++){

if(matrix[ty][tx] != 0 && (x+tx < 0 || x+tx >= map[0].length || y+ty >= map.length || map[ty+y][tx+x] != 0)){

return false;

}

}

}

return true

}

function finishgame() {

finish = true;

let lead;

if(localStorage.hasOwnProperty('leaders')){

lead = JSON.parse(localStorage.getItem("leaders"));

lead.push({user: username, score: score});

lead.sort(function (a,b) {

return b.score - a.score;

});

while(lead.length>10){

lead.pop();

}

localStorage.setItem('leaders',JSON.stringify(lead));

}else{

lead = [];

lead.push({user:username,score: score});

localStorage.setItem('leaders',JSON.stringify(lead));

}

let array = '<ol>';

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

array += '<li>';

array += lead[i].user + " - ";

array += lead[i].score;

array += '</li>';

}

array += '</ol>';

document.getElementById("table").innerHTML = "Таблица лидеров\n" + array;

lead\_table.showModal();

}

function draw(){

ctx.clearRect(0,0,field.width,field.height);

for(let y = 0; y < 16; y++){

for(let x = 0; x < 10; x++){

if(map[y][x] != 0){

ctx.fillStyle = color[map[y][x]];

ctx.fillRect(x\*CELL, y\*CELL, CELL-1, CELL-1);

}

}

}

ctx.fillStyle = color[cur\_fig.name];

for(let y = 0; y < cur\_fig.matrix.length; y++){

for(let x = 0; x < cur\_fig.matrix[0].length; x++){

if(cur\_fig.matrix[y][x] != 0){

ctx.fillRect((cur\_fig.col+x)\*CELL, (cur\_fig.row+y)\*CELL, CELL-1, CELL-1);

}

}

}

}

function setfig(){

for(let y = 0; y < cur\_fig.matrix.length; y++){

for(let x = 0; x < cur\_fig.matrix[0].length; x++){

if(cur\_fig.matrix[y][x] != 0){

if(cur\_fig.row+y < 0){

finish = true;

return;

}

map[cur\_fig.row+y][cur\_fig.col+x] = cur\_fig.name

}

}

}

let row = map.length - 1

let rowsCount = 0;

while(row >= 0){

if(map[row].every(cur => cur != 0)){

counter++;

if(counter >=2) score += 100;

for (let y = row; y >= 0; y--) {

for (let x = 0; x < map[y].length; x++) {

map[y][x] = map[y-1][x];

}

}

rowsCount++;

}

else{

counter = 0;

row--;

}

}

for (; rowsCount > 0; rowsCount--) {

score += 100 \* rowsCount;

rowsCount--;

}

document.getElementById("score").innerHTML = score;

if(score >= scores[level+1]){

level++;

if(level > 8){

level = 8;

}

document.getElementById("level").innerHTML = level;

}

cur\_fig = newfig();

clearInterval(interval);

interval = setInterval(automove, levels[level]);

}

function automove(){

let row = cur\_fig.row + 1

if(!canmove(cur\_fig.matrix, cur\_fig.col, row)){

setfig();

if(finish){

clearInterval(interval);

finishgame();

return;

}

}

else{

cur\_fig.row = row;

}

draw();

}

document.addEventListener('keydown', function(event) {

if (event.which === 37 || event.which === 39) {

let col = 0;

if(event.which === 37){

col = cur\_fig.col - 1;

}

else{

col = cur\_fig.col + 1;

}

if(canmove(cur\_fig.matrix, col, cur\_fig.row)) {

cur\_fig.col = col;

draw();

}

}

if (event.which === 38) {

const matrix = rotate(cur\_fig.matrix);

if (canmove(matrix, cur\_fig.col, cur\_fig.row)) {

cur\_fig.matrix = matrix;

draw();

}

}

if(event.which === 40) {

const row = cur\_fig.row + 1;

if(!canmove(cur\_fig.matrix, cur\_fig.col, row)){

setfig();

draw();

return;

}

else{

cur\_fig.row++;

draw();

}

}

});

function start(){

if(interval) {

clearInterval(interval)

}

interval = setInterval(automove, levels[level])

}

**main.html:**

<!doctype html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<link rel="icon" type="image/jpg" href="./public/icon.jpg" />

<link href="https://fonts.googleapis.com/css?family=Yanone+Kaffeesatz" rel="stylesheet"/>

<link rel="stylesheet" type="text/css" href="style.css">

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

<script type="text/javascript" src="init.js"></script>

<title>Tetris</title>

</head>

<body onload="read()">

<dialog id="table" open>

</dialog>

<div id="leaders"></div>

<div class="grid">

<div class="left-column">

<div>

<h2>TETRIS</h2>

<p>Игрок: <label id="username"></label></p>

<p>Очки: <span id="score">0</span></p>

<p>Уровень: <span id="level">1</span></p>

<p>Следующая фигура:</p>

<canvas id="next" width="300" height="300"></canvas>

</div>

<button onclick="start()" class="play-button">Старт!</button>

</div>

<canvas id="field" width="500" height="800"></canvas>

</div>

<script type="text/javascript" src="main.js"></script>

</body>

</html>

**init.js:**

function store() {

let input = document.getElementById("input");

let username = input.value;

if(username !== "" && username.length <= 10){

console.log(username.length);

localStorage.setItem("gamer\_name", username);

window.location.href = ("../main.html");

}else{

if(username.length > 10){

console.log("Ошибка");

window.alert("Имя не может быть длиннее 10 символов!");

}

else {

console.log("Ошибка");

window.alert("Введите имя");

}

}

}

function set\_name() {

if(localStorage.hasOwnProperty("gamer\_name")){

let name = localStorage.getItem("gamer\_name");

console.log(name)

let input = document.getElementById("input");

input.value = name;

}

}

function read(){

document.getElementById("username").innerHTML = localStorage["gamer\_name"];

}

**init.html:**

<!doctype html>

<html lang="en">

<head>

<meta charset="UTF-8"/>

<link rel="icon" type="image/jpg" href="./public/icon.jpg" />

<link href="https://fonts.googleapis.com/css?family=Yanone+Kaffeesatz" rel="stylesheet"/>

<link rel="stylesheet" type="text/css" href="style.css">

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

<script type="text/javascript" src="init.js"></script>

<title>Tetris</title>

</head>

<body onload="set\_name()">

<label>

<font color = "white", size = 10em>Введите имя:</font><br>

<input id="input" placeholder="Имя пользователя">

</label><br>

<input id="button" type="button" value="Ввод" onclick="store()">

</body>

</html>

**style.css:**

:root {

font-family: 'Yanone Kaffeesatz', cursive;

line-height: 1.5;

font-weight: 300;

color-scheme: light dark;

color: rgba(255, 255, 255, 0.87);

background-color: #242424;

}

html, body {

height: 100vh;

}

.left-column {

justify-content: center;

align-items: center;

font-size: 2em;

width: 80%;

transform: translate(10%);

}

.grid {

display: grid;

grid-template-columns: 380px 520px;

position: absolute;

left: 50%;

transform: translate(-50%);

backdrop-filter: blur(5px) brightness(0.3);

padding: 10px;

}

body {

margin: 5;

display: flex;

flex-direction: column;

justify-content: center;

align-items: center;

background: url("bgfkit.png");

}

canvas {

border: 5px solid white;

}

.right-column {

display: flex;

flex-direction: column;

justify-content: space-between;

}

h1 {

font-size: 3.2em;

line-height: 1.1;

}

button {

border-radius: 8px;

border: 1px solid transparent;

left: 30%;

transform: translate(50%);

padding: 0.6em 1.2em;

font-size: 1em;

font-weight: 500;

font-family: inherit;

background-color: white;

cursor: pointer;

color: black;

transition: border-color 0.25s;

}

button:hover {

border-color: #646cff;

}