Предлог поена:

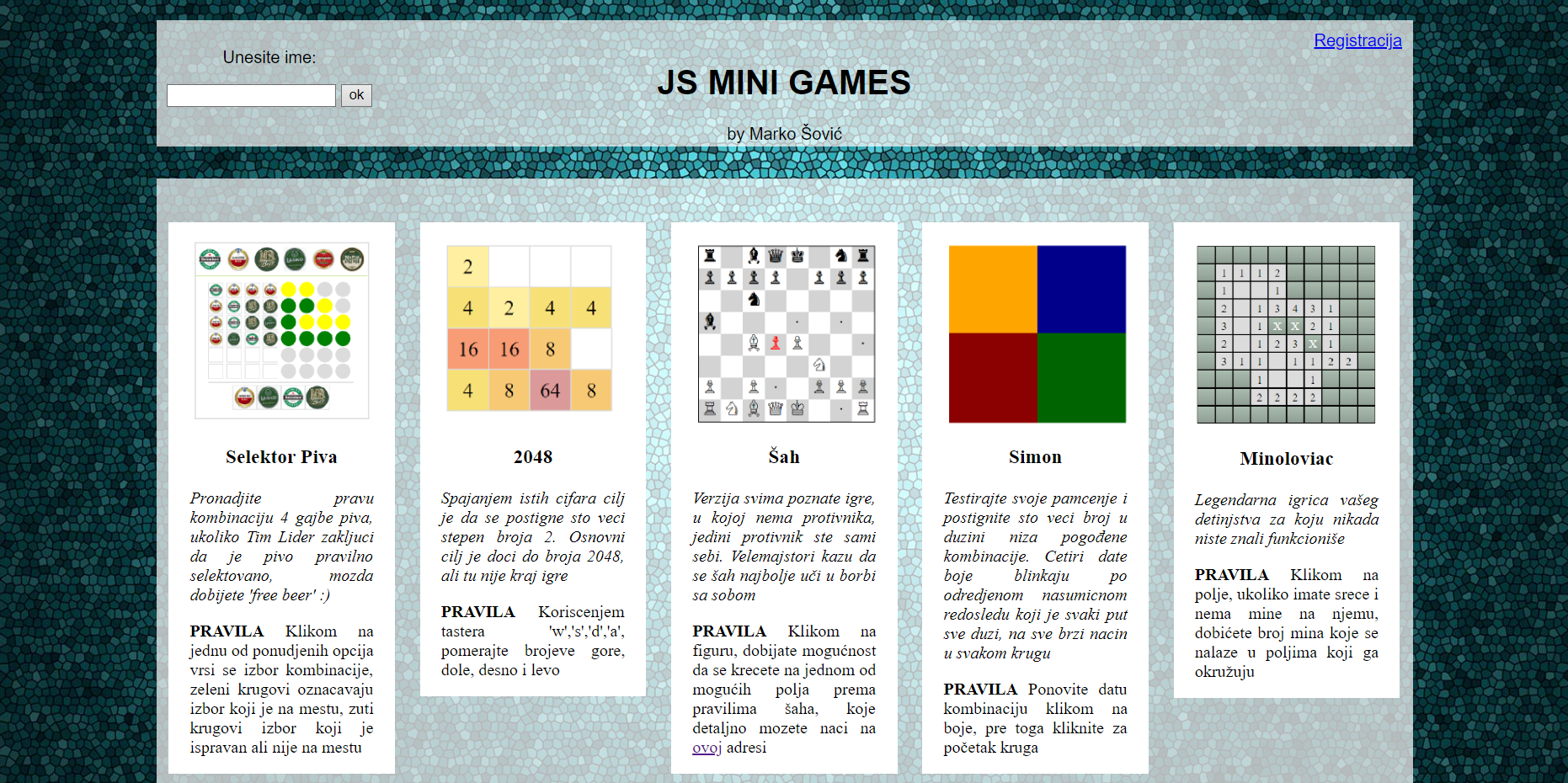
Коначно поена:

Интернет језици и алати 2: Испитни Пројекат

Марко Шовић ИТ 153/17

**Пројекат**

**Кратак опис: Пројекат представља апликацију која саджи 5 игара направљених у JavaScript-u, кликом на жељену игру добијате могућност да је играте.**



Слика1

**Код:**

**HTML**

**index.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<link rel="stylesheet" href="style.css">

<title>Document</title>

</head>

<body>

<header>

<div id="unos">

<p>Unesite ime:</p>

<input type="text" id="ime">

<input value="ok" type="button" onclick="setCookie('unos')">

</div>

<h1>JS mini games</h1>

<p>by Marko Šović</p>

<div id="registracija"><a href="registracija.html">Registracija</a></div>

</header>

<main>

<div id="app"></div>

</main>

<script src="index.js"></script>

<script src="cookie.js"></script>

</body>

</html>

**registracija.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<link rel="stylesheet" href="style.css">

<title>Document</title>

</head>

<body>

<header>

<h1>JS mini games</h1>

<p>by Marko Šović</p>

</header>

<main>

<form name="reg">

<p>Unesite ime:</p>

<input type="text" name="ime">

<p>Unesite email:</p>

<input type="text" name="email">

<p>Unesite lozinku:</p>

<input type="password" name="loz1">

<p>Potvrdite lozinku:</p>

<input type="password" name="loz2">

<br><br>

<input type="button" onclick="registracija()" value="Potvrdi">

<p id="greska"></p>

</form>

</main>

<script src="registracija.js"></script>

</body>

</html>

**BeerSelector/start.html**

<!DOCTYPE html>

<html lang="en" dir="ltr">

<head>

<meta charset="utf-8">

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

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

<title>Selektor piva</title>

</head>

<body>

<header>

<a href="../index.html"><h1>JS mini games</h1></a>

<p>by Marko Šović</p>

</header>

<main>

<div>

<div id="ponudjeno">

<img src="assets/img/heineken.jpg" onclick="potez(0,'h')">

<img src="assets/img/amstel.jpg" onclick="potez(1,'a')">

<img src="assets/img/mb.jpg" onclick="potez(2,'m')">

<img src="assets/img/lasko.jpg" onclick="potez(3,'l')">

<img src="assets/img/zajecarsko.jpg" onclick="potez(4,'z')">

<img src="assets/img/pilsplus.jpg" onclick="potez(5,'p')">

</div>

<div id="krugovi"></div>

<div id="restart">

<button onclick="restart()">Restart</button>

</div>

</div>

</main>

<script src="assets/main.js" charset="utf-8"></script>

</body>

</html>

**2048/index.html**

<!DOCTYPE html>

<html lang="en" dir="ltr">

<head>

<meta charset="utf-8">

<title>2048</title>

<link rel="stylesheet" href="style.css">

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

</head>

<body>

<header>

<a href="../index.html"><h1>JS mini games</h1></a>

<p>by Marko Šović</p>

</header>

<main>

<div>

<div id="root"></div>

<div id="unazad">

<button onclick="undo()">Korak Unazad</button>

</div>

</div>

</main>

<script src="script.js" charset="utf-8"></script>

</body>

</html>

**Chess/index.html**

<!DOCTYPE html>

<html lang="en" dir="ltr">

<head>

<meta charset="utf-8">

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

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

<title></title>

</head>

<body>

<header>

<a href="../index.html"><h1>JS mini games</h1></a>

<p>by Marko Šović</p>

</header>

<main>

<div>

<div id="container">

<div id="root"></div>

</div>

</div>

</main>

<script src="pieces.js" charset="utf-8"></script>

<script src="script.js" charset="utf-8"></script>

</body>

</html>

**Simon/index.html**

<!DOCTYPE html>

<html lang="en" dir="ltr">

<head>

<meta charset="utf-8">

<title>Simon</title>

<link rel="stylesheet" href="style.css">

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

</head>

<body>

<header>

<a href="../index.html"><h1>JS mini games</h1></a>

<p>by Marko Šović</p>

</header>

<main>

<div>

<div id="root">

<div class='color' id='yellow'></div>

<div class='color' id='blue'></div>

<div id='break'></div>

<div class='color' id='red'></div>

<div class='color' id='green'></div>

</div>

<div id="info">

<h3 id="msg"></h3>

<button onclick='start()'>Počni level 1</button>

<h3>Najveći dostignuti level: 1</h3>

</div>

</div>

</main>

<script src="script.js" charset="utf-8"></script>

</body>

</html>

**MinesWeeper/index.html**

<!DOCTYPE html>

<html lang="en" dir="ltr">

<head>

<meta charset="utf-8">

<title></title>

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

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

</head>

<body>

<header>

<a href="../index.html"><h1>JS mini games</h1></a>

<p>by Marko Šović</p>

</header>

<main>

<div>

<div id='menu'>

<button onclick="toggleOptions()">Opcije</button>

<button onclick='newGame()'>Nova Igra</button>

</div>

<div id="options"></div>

<div id="main">

<div id="table"></div>

<div id="game-stats"></div>

</div>

</div>

</main>

<script src="cell.js" charset="utf-8"></script>

<script src="stats.js" charset="utf-8"></script>

<script src="script.js" charset="utf-8"></script>

<script src="options.js" charset="utf-8"></script>

</body>

</html>

**CSS**

**style.css**

\* {

box-sizing: border-box;

}

body {

background: url("\_img/background.jpg");

background-attachment: fixed;

background-position: center;

background-repeat: no-repeat;

background-size: cover;

margin: 0px;

}

header {

width: 80%;

height: 120px;

margin: auto;

margin-top: 20px;

text-align: center;

font-family: sans-serif;

background-color: rgba(255, 255, 255, 0.7);

padding: 20px;

position: relative;

}

header h1 {

text-transform: uppercase;

}

main {

margin-top: 30px;

}

main > div {

width: 80%;

margin: auto;

background: rgba(255, 255, 255, 0.7);

padding: 30px 0px;

/\* height: 400px; \*/

}

.kartica {

width: 18%;

/\* height: 400px; \*/

margin: 1%;

float: left;

display: inline-block;

background-color: white;

}

.kartica p {

padding: 0px 20px;

text-align: justify;

}

.kartica h3 {

text-align: center;

margin-top: -5px;

}

.kartica img {

width: 100%;

}

#clear {

clear: both;

}

form {

width: 400px;

padding: 20px;

background-color: white;

margin: auto;

text-align: center;

}

#greska {

color: red;

}

#registracija {

position: absolute;

right: 10px;

top: 10px;

}

#unos {

position: absolute;

left: 10px;

top: 10px;

}

**BeerSelector/style.css**

#ponudjeno, #krugovi, #restart {

padding: 10px;

width: 570px;

background-color: white;

margin: auto;

}

#ponudjeno img {

width: 80px;

height: 80px;

margin-right: 10px;

transition: transform 0.5s;

}

#ponudjeno img:last-of-type {

margin-right: 0px;

}

#ponudjeno img:hover {

cursor: pointer;

transform: scale(1.2);

}

#ponudjeno {

border-bottom: 1px solid yellowgreen;

}

#krugovi {

padding: 20px 45px;

}

#krugovi img {

width: 100%;

height: 100%;

}

.polje {

width: 50px;

height: 50px;

border: 2px solid gainsboro;

display: inline-block;

margin-right: 10px;

}

.provera {

width: 50px;

height: 50px;

border-radius: 25px;

background: gainsboro;

display: inline-block;

margin-right: 10px;

}

.trazeno {

width: 80px;

height: 80px;

border: 2px solid gainsboro;

display: inline-block;

}

#prvi {

margin-left: 80px;

}

#restart button {

display: block;

margin: auto;

margin-bottom: 30px;

width: 100px;

height: 30px;

cursor: pointer;

}

**2048/style.css**

#root {

width: 428px;

padding: 50px;

background-color: white;

margin: auto;

}

.cell {

display: inline-block;

float: left;

border: 1px solid gainsboro;

text-align: center;

width: 80px;

height: 80px;

line-height: 80px;

font-size: 40px;

-webkit-transition: color 0.3s;

}

.break {

display: block;

clear: both;

}

#unazad {

width: 428px;

background-color: white;

margin: auto;

padding: 10px 0px;

}

#unazad button {

display: block;

margin: auto;

margin-bottom: 30px;

width: 200px;

height: 30px;

cursor: pointer;

}

**Chess/style.css**

\* {

-webkit-touch-callout: none;

-webkit-user-select: none;

-khtml-user-select: none;

-moz-user-select: none;

-ms-user-select: none;

user-select: none;

}

#container {

width: 500px;

padding: 50px;

margin: auto;

background-color: white;

}

#root {

box-sizing: content-box;

width: 400px;

height: 400px;

border: 2px solid black;

background-color: white;

}

.feild {

cursor: grab;

font-size: 42px;

line-height: 45px;

text-align: center;

width: 50px;

height: 50px;

display: inline-block;

float: left;

border: none;

}

.black {

background-color: lightgray;

}

**Simon/style.css**

.color {

width: 200px;

height: 200px;

display: inline-block;

margin-top: -5px;

float: left;

}

#break {

width: 0px;

height: 0px;

display: block;

clear: both;

}

button {

display: block;

margin: auto;

width: 150px;

height: 30px;

}

#root {

margin: auto;

height: 500px;

padding: 50px;

background-color: white;

width: 500px;

}

#info {

margin: auto;

box-sizing: content-box;

background-color: white;

width: 500px;

padding-bottom: 30px;

padding-top: 1px;

}

#info h3 {

text-align: center;

}

**MinesWeeper/style.css**

#menu {

margin-bottom: 20px;

width: 478px;

background-color: white;

margin: auto;

}

#options {

width: 478px;

background-color: white;

margin: auto;

}

#main {

height: 500px;

padding: 50px;

margin: auto;

}

div > .cell {

border: 1px solid black;

width: 25px;

height: 25px;

display: inline-block;

background: linear-gradient(#a9bcae, #8f9e93);

text-align: center;

line-height: 25px;

float: left;

cursor: pointer;

}

#options input {

width: 40px;

}

#options > \* {

display: inline-block;

margin-left: 10px;

}

#game-stats {

text-align: center;

background-color: white;

width: 478px;

margin: auto;

margin-top: -50px;

padding-bottom: 5px;

}

#game-stats #left {

margin: 0px;

}

#table {

margin: auto;

padding: 50px;

box-sizing: content-box;

background-color: white;

margin-top: -50px;

}

**JavaScript**

**index.js**

var json = [

{

"naziv": "Selektor Piva",

"opis" : "Pronadjite pravu kombinaciju 4 gajbe piva, ukoliko Tim Lider zakljuci da je pivo pravilno selektovano, mozda dobijete 'free beer' :) ",

"pravila" : "Klikom na jednu od ponudjenih opcija vrsi se izbor kombinacije, zeleni krugovi oznacavaju izbor koji je na mestu, zuti krugovi izbor koji je ispravan ali nije na mestu",

"adresa" : "BeerSelector/start.html",

"slika" : "\_img/selector.png"

},{

"naziv": "2048",

"opis" : "Spajanjem istih cifara cilj je da se postigne sto veci stepen broja 2. Osnovni cilj je doci do broja 2048, ali tu nije kraj igre",

"pravila" : "Koriscenjem tastera 'w','s','d','a', pomerajte brojeve gore, dole, desno i levo",

"adresa" : "2048/index.html",

"slika" : "\_img/2048.png"

},{

"naziv": "Šah",

"opis" : "Verzija svima poznate igre, u kojoj nema protivnika, jedini protivnik ste sami sebi. Velemajstori kazu da se šah najbolje uči u borbi sa sobom",

"pravila" : "Klikom na figuru, dobijate mogućnost da se krecete na jednom od mogućih polja prema pravilima šaha, koje detaljno mozete naci na <a href='https://sr.wikipedia.org/wiki/Правила\_шаха'>ovoj</a> adresi",

"adresa" : "Chess/index.html",

"slika" : "\_img/chess.png"

}, {

"naziv": "Simon",

"opis" : "Testirajte svoje pamcenje i postignite sto veci broj u duzini niza pogođene kombinacije. Cetiri date boje blinkaju po odredjenom nasumicnom redosledu koji je svaki put sve duzi, na sve brzi nacin u svakom krugu",

"pravila" : "Ponovite datu kombinaciju klikom na boje, pre toga kliknite za početak kruga",

"adresa" : "Simon/index.html",

"slika" : "\_img/simon.png"

}, {

"naziv" : "Minoloviac",

"opis" : "Legendarna igrica vašeg detinjstva za koju nikada niste znali funkcioniše",

"pravila" : "Klikom na polje, ukoliko imate srece i nema mine na njemu, dobićete broj mina koje se nalaze u poljima koji ga okružuju",

"adresa" : "MinesWeeper/index.html",

"slika" : "\_img/mines.png"

}

]

console.log(json);

var app = document.getElementById('app');

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

app.innerHTML += `<div class="kartica" ><a href="${json[i].adresa}">`+

`<img src="${json[i].slika}"/></a>`+

`<h3>${json[i].naziv}</h3>`+

`<p><i>${json[i].opis}</i></p>`+

`<p><b>PRAVILA</b> ${json[i].pravila}</p></div>`;

} app.innerHTML += "<div id='clear'></div><hr>";

**registracija.js**

function registracija() {

var forma = document.reg;

var ime = forma.ime;

var email = forma.email;

var pass1 = forma.loz1;

var pass2 = forma.loz2;

console.log(ime.value + " " + email.value + " " + pass1.value + " " + pass2.value);

try {

if (ime.value == "" ||

email.value == "" ||

pass1.value == "" ||

pass2.value == "") {

throw "Molimo popunite sva polja";

} else {

if (validacijaEmaila(email.value) == false) {

throw "Email nepravino unet";

} else if (pass1.value != pass2.value) {

throw "Lozinka se ne podudara"

}

}

window.location.href = "index.html";

} catch (error) {

document.getElementById("greska").innerHTML = error;

}

}

function validacijaEmaila(email) {

var regex = /^(([^<>()[\]\\.,;:\s@\"]+(\.[^<>()[\]\\.,;:\s@\"]+)\*)|(\".+\"))@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$/;

return regex.test(email);

}

**Cookie.js**

function setCookie(provera) {

console.log(document.cookie);

if (provera == "unos") {

if (document.cookie != "" ) {

document.getElementById("unos").innerHTML = "Dobrodosli " + document.cookie.substr(9);

}

else {

var username = document.getElementById("ime").value;

document.cookie = "username=" + username;

console.log(document.cookie);

}

} else {

if (document.cookie.substr(9) != "" ) {

document.getElementById("unos").innerHTML = "Dobrodosli " + document.cookie.substr(9);

}

}

}

setCookie();

**BeerSelector/main.js**

function izbor(naziv) {

this.naziv = naziv;

this.provereno = false;

}

var piva = [

'assets/img/heineken.jpg',

'assets/img/amstel.jpg',

'assets/img/mb.jpg',

'assets/img/lasko.jpg',

'assets/img/zajecarsko.jpg',

'assets/img/pilsplus.jpg'

];

var kombinacija = new Array(4);

var izabrana = [];

var brojac = 0;

var polje = 0;

var krug = 1;

function restart() {

kombinacija = new Array(4);

izabrana = [];

brojac = 0;

polje = 0;

krug = 1;

proveraPolje = 0;

for (i = 0; i < 24; i ++) {

document.getElementsByClassName('polje')[i].innerHTML = "";

document.getElementsByClassName('provera')[i].style.background = "gainsboro";

if( i < 4 ) {

document.getElementsByClassName('trazeno')[i].innerHTML = "";

}

}

novaKombinacija();

}

var krugovi = document.getElementById('krugovi');

function nacrt() {

for (var i = 0; i < 6; i++) {

for (j = 0; j < 4; j++) {

krugovi.innerHTML += "<div class='polje'></div>";

}

for (j = 0; j < 4; j++) {

krugovi.innerHTML += "<div class='provera'></div>";

}

krugovi.innerHTML += "</br>";

}

krugovi.innerHTML += "<hr>";

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

if (i == 0)

krugovi.innerHTML += "<div id='prvi' class='trazeno'></div>";

else

krugovi.innerHTML += "<div class='trazeno'></div>";

}

novaKombinacija();

}

nacrt();

function novaKombinacija() {

for (i = 0; i < kombinacija.length; i++)

{

var broj = Math.floor((Math.random() \* 60) + 1);

if (broj <= 10)

kombinacija[i] = new izbor('h');

else if (broj <= 20)

kombinacija[i] = new izbor('a');

else if (broj <= 30)

kombinacija[i] = new izbor('m');

else if (broj <= 40)

kombinacija[i] = new izbor('l');

else if (broj <= 50)

kombinacija[i] = new izbor('z');

else

kombinacija[i] = new izbor('p')

}

console.log("kombinacija je:");

console.log(kombinacija);

}

function potez(a,b) {

var node = document.getElementsByClassName('polje')[brojac];

node.innerHTML += "<img src='"+piva[a]+"'>";

izabrana[polje] = new izbor(b);

polje++;

brojac++;

if (brojac % 4 == 0) {

console.log(izabrana);

provera();

polje = 0;

}

}

function provera() {

var naMestu = 0;

var pogodjeni = 0;

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

{

if(kombinacija[i].naziv == izabrana[i].naziv){

izabrana[i].provereno = true;

kombinacija[i].provereno = true;

naMestu++;

}

else {

kombinacija[i].provereno = false;

}

}

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

if (kombinacija[i].provereno == false) {

for(j=0;j<4;j++){

if (kombinacija[i].naziv == izabrana[j].naziv && izabrana[j].provereno == false) {

izabrana[j].provereno = true;

pogodjeni++;

break;

}

}

}

}

console.log("na mestu: " + naMestu);

console.log("pogodjeni: " + pogodjeni);

console.log("kombinacija je:");

console.log(kombinacija);

ispis(naMestu, pogodjeni, krug);

krug++;

if (naMestu == 4 || krug == 7)

{

resenje();

}

}

var proveraPolje = 0;

function ispis(m,p,k) {

switch (krug) {

case 2:

proveraPolje = 4;

break;

case 3:

proveraPolje = 8;

break;

case 4:

proveraPolje = 12;

break;

case 5:

proveraPolje = 16;

break;

case 6:

proveraPolje = 20;

break;

default:

break;

}

for(i = 0; i < m; i++ )

{

node = document.getElementsByClassName('provera')[proveraPolje];

node.style.background = 'green';

proveraPolje++;

}

for(i = 0; i < p; i++ )

{

node = document.getElementsByClassName('provera')[proveraPolje];

node.style.background = 'yellow';

proveraPolje++;

}

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

kombinacija[i].provereno = false;

}

izabrana = [];

}

function resenje() {

var node = document.getElementsByClassName('trazeno');

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

{

switch (kombinacija[i].naziv) {

case 'h':

node[i].innerHTML = "<img src='"+piva[0]+"'>";

break;

case 'a':

node[i].innerHTML = "<img src='"+piva[1]+"'>";

break;

case 'm':

node[i].innerHTML = "<img src='"+piva[2]+"'>";

break;

case 'l':

node[i].innerHTML = "<img src='"+piva[3]+"'>";

break;

case 'z':

node[i].innerHTML = "<img src='"+piva[4]+"'>";

break;

case 'p':

node[i].innerHTML = "<img src='"+piva[5]+"'>";

break;

default:

break;

}

}

}

**2048/script.js**

var root = document.getElementById('root');

var grid = [

[0,0,0,0],

[0,0,0,0],

[0,0,0,0],

[0,0,0,0]

]

var previousGrid;

function rememberGrid() {

var temp = [];

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

temp.push([]);

for (j = 0; j<4;j++) {

temp[i].push(grid[i][j]);

}

}

return temp;

};

function draw() {

addNums(2);

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

for(j=0; j<4; j++){

root.innerHTML+= "<div class='cell' id='"+i+""+j+"'>"+ (

grid[i][j] != 0? grid[i][j]:"")+"</div>"

}

root.innerHTML+="<div class='break'>";

}

previousGrid = rememberGrid();

write();

}

function addNums(n) {

var x = 0;

while (x < n){

var randX = Math.floor(Math.random() \* 4);

var randY = Math.floor(Math.random() \* 4);

if (grid[randX][randY] == 0) {

var rand = Math.floor(Math.random()\*100);

grid[randX][randY] = rand>25? 2 : 4;

x++;

}

}

}

function write() {

for(var a = 0; a < 4; a++) {

for (var b = 0; b < 4; b++) {

var element = document.getElementById(a+""+b);

var value = grid[a][b];

if (value != 0) {

element.innerHTML = value;

element.style.color = 'black';

switch (value) {

case 2:

element.style.backgroundColor = '#ffefa0';

break;

case 4:

element.style.backgroundColor = '#f7e176';

break;

case 8:

element.style.backgroundColor = '#f7c975';

break;

case 16:

element.style.backgroundColor = '#f79c75';

break;

case 32:

element.style.backgroundColor = '#f78275';

break;

case 64:

element.style.backgroundColor = '#db9999';

break;

default:

}

}

else {

element.innerHTML = "";

element.style.color = 'white';

element.style.backgroundColor = 'white'

}

}

}

}

document.addEventListener("keypress", function(event) {

console.log(event.keyCode);

if (event.keyCode == 119) {

previousGrid = rememberGrid();

moveUp();

}

if (event.keyCode == 115) {

previousGrid = rememberGrid();

moveDown();

}

if (event.keyCode == 100) {

previousGrid = rememberGrid();

moveRight();

}

if (event.keyCode == 97) {

previousGrid = rememberGrid();

moveLeft();

}

})

function sortRight() {

var changed = false;

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

for (var k = 0; k < 3; k++) {

if (grid[i][k] != 0 && grid[i][k+1] == 0) {

grid[i][k+1] = grid[i][k];

grid[i][k] = 0;

write();

changed = true;

}

}

}

return changed;

}

function sortDown() {

var changed = false;

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

for (var k = 0; k < 3; k++) {

if (grid[k][i] != 0 && grid[k+1][i] == 0) {

grid[k+1][i] = grid[k][i];

grid[k][i] = 0;

write();

changed = true;

}

}

}

return changed;

}

function moveRight() {

var changed = false;

changed = sortRight();

sortRight();

sortRight();

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

for (var j = 3; j >= 1; j--) {

if (grid[i][j] != 0 && grid[i][j] == grid[i][j-1]) {

grid[i][j] += grid[i][j-1];

grid[i][j-1] = 0;

write();

changed = true;

j--;

}

}

}

sortRight();

sortRight();

if (changed)

{

addNums(1);

write();

}

}

function moveDown() {

var changed = false;

changed = sortDown();

sortDown();

sortDown();

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

for (var j = 3; j >= 1; j--) {

if (grid[j][i] != 0 && grid[j][i] == grid[j-1][i]) {

grid[j][i] += grid[j-1][i];

grid[j-1][i] = 0;

write();

changed = true;

j--;

}

}

}

sortDown();

sortDown();

if (changed)

{

addNums(1);

write();

}

}

function undo() {

grid = previousGrid;

write();

}

function sortLeft() {

var changed = false;

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

for (var k = 3; k > 0; k--) {

if (grid[i][k] != 0 && grid[i][k-1] == 0) {

grid[i][k-1] = grid[i][k];

grid[i][k] = 0;

write();

changed = true;

}

}

}

return changed;

}

function sortUp() {

var changed = false;

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

for (var k = 3; k > 0; k--) {

if (grid[k][i] != 0 && grid[k-1][i] == 0) {

grid[k-1][i] = grid[k][i];

grid[k][i] = 0;

write();

changed = true;

}

}

}

return changed;

}

function moveLeft() {

var changed = false;

changed = sortLeft();

sortLeft();

sortLeft();

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

for (var j = 0; j < 3; j++) {

if (grid[i][j] != 0 && grid[i][j] == grid[i][j+1]) {

grid[i][j] += grid[i][j+1];

grid[i][j+1] = 0;

write();

changed = true;

j++;

}

}

}

sortLeft();

sortLeft();

if (changed)

{

addNums(1);

write();

}

}

function moveUp(){

var changed = false;

changed = sortUp();

sortUp();

sortUp();

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

for (var j = 0; j < 3; j++) {

if (grid[j][i] != 0 && grid[j][i] == grid[j+1][i]) {

grid[j][i] += grid[j+1][i];

grid[j+1][i] = 0;

write();

changed = true;

j++;

}

}

}

sortUp();

sortUp();

if (changed)

{

addNums(1);

write();

}

}

draw();

**chess/script.js**

var root = document.getElementById('root');

const col = ['a','b','c','d','e','f','g','h'];

var selectedPiece = null;

function Place(id) {

this.id = id;

this.piece = null;

this.placePiece = function(name,symbol) {

this.piece = new Piece(name,symbol,this.id);

document.getElementById(id).innerHTML = symbol;

}

this.removePiece = function() {

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

this.piece = null;

}

}

function drawBoard() {

for (i = 8; i >= 1; i--) {

for (j = 0; j < 8; j++) {

var id = col[j] + i;

fields[id] = new Place(id);

if (j % 2 == 0) {

if (i % 2 == 0)

root.innerHTML += `<div onclick = fieldClick('${id}') id='${id}' class='feild white'></div>`;

else

root.innerHTML += `<div onclick = fieldClick('${id}') id='${id}' class='feild black'></div>`;

}

else {

if (i % 2 == 0)

root.innerHTML += `<div onclick = fieldClick('${id}') id='${id}' class='feild black'></div>`;

else

root.innerHTML += `<div onclick = fieldClick('${id}') id='${id}' class='feild white'></div>`;

}

}

}

}

drawBoard();

function placePeices() {

fields['e1'].placePiece('whiteKing', "&#9812;");

fields['d1'].placePiece('whiteQueen',"&#9813;");

fields['c1'].placePiece('whiteBishup', "&#9815;");

fields['f1'].placePiece('whiteBishup', "&#9815;");

fields['b1'].placePiece('whiteKnight',"&#9816;");

fields['g1'].placePiece('whiteKnight',"&#9816;");

fields['a1'].placePiece('whiteRook', "&#9814;");

fields['h1'].placePiece('whiteRook', "&#9814;");

for (i = 0; i<8;i++)

fields[col[i]+'2'].placePiece('whitePawn', "&#9817;");

fields['e8'].placePiece('blackKing', "&#9818;");

fields['d8'].placePiece('blackQueen', "&#9819;");

fields['c8'].placePiece('blackBishup', "&#9821;");

fields['f8'].placePiece('blackBishup', "&#9821;");

fields['b8'].placePiece('blackKnight',"&#9822;");

fields['g8'].placePiece('blackKnight',"&#9822;");

fields['a8'].placePiece('blackRook', "&#9820;");

fields['h8'].placePiece('blackRook', "&#9820;");

for (i = 0; i<8;i++)

fields[col[i]+'7'].placePiece('blackPawn', "&#9823;");

}

placePeices();

function fieldClick(id) {

console.log(fields[id]);

if (selectedPiece!= null) {

clearMoves(selectedPiece.movePlaces);

clearAttack(selectedPiece.attackPlaces);

if (selectedPiece.attackPlaces.indexOf(id) > -1) {

console.log(true);

fields[id].placePiece(selectedPiece.name, selectedPiece.symbol);

fields[selectedPiece.position].removePiece();

clearMoves(selectedPiece.movePlaces);

clearAttack(selectedPiece.attackPlaces);

}

}

if (fields[id].piece != null) {

selectedPiece = fields[id].piece;

selectedPiece.findMoves();

drawMoves(selectedPiece.movePlaces);

drawAttack(selectedPiece.attackPlaces);

console.log("move places");

console.log(selectedPiece.movePlaces);

console.log("attack places");

console.log(selectedPiece.attackPlaces);

// console.log(selectedPiece.name);

} else {

if (selectedPiece != null && selectedPiece.movePlaces.indexOf(id)>-1) {

fields[id].placePiece(selectedPiece.name, selectedPiece.symbol);

fields[selectedPiece.position].removePiece();

clearMoves(selectedPiece.movePlaces);

clearAttack(selectedPiece.attackPlaces)

}

selectedPiece = null;

}

}

function drawMoves(arr) {

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

if (fields[arr[i]] && fields[arr[i]].piece == null) {

document.getElementById(arr[i]).innerHTML = "·";

}

}

}

function drawAttack(arr) {

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

document.getElementById(arr[i]).style.color = "red";

}

}

function clearAttack(arr) {

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

document.getElementById(arr[i]).style.color = "black";

}

}

function clearMoves(arr) {

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

if (fields[arr[i]] && fields[arr[i]].piece == null) {

document.getElementById(arr[i]).innerHTML = "";

}

}

}

**Chess/pieces.js**

var fields = [];

function Piece(name,symbol,position) {

this.name = name;

this.symbol = symbol;

this.position = position;

this.col = col.indexOf(position.substring(0,1));

this.row = parseInt(position.substring(1));

this.attackPlaces = [];

this.movePlaces = [];

this.color = name.substring(0,5);

this.findMoves = function () {

this.attackPlaces = [];

this.movePlaces = [];

switch (this.name) {

case "whitePawn":

if (this.row === 2) {

this.movePlaces = [col[this.col] + (this.row + 1) ,

col[this.col] + (this.row + 2)]

} else

this.movePlaces = [col[this.col] + (this.row + 1)];

if ( fields[col[this.col+1] + (this.row +1)] != undefined

&& fields[col[this.col+1] + (this.row +1)].piece != null

&& this.color != fields[col[this.col+1] + (this.row +1)].piece.color) {

this.attackPlaces.push(col[this.col+1] + (this.row +1));

}

if ( fields[col[this.col-1] + (this.row +1)] != undefined

&& fields[col[this.col-1] + (this.row +1)].piece != null

&& this.color != fields[col[this.col-1] + (this.row +1)].piece.color) {

this.attackPlaces.push(col[this.col-1] + (this.row +1));

}

break;

case "blackPawn":

if (this.row === 7) {

this.movePlaces = [col[this.col] + (this.row - 1) ,

col[this.col] + (this.row - 2)]

} else

this.movePlaces = [col[this.col] + (this.row - 1)];

if ( fields[col[this.col+1] + (this.row - 1)] != undefined

&& fields[col[this.col+1] + (this.row - 1)].piece != null

&& this.color != fields[col[this.col +1] + (this.row -1)].piece.color) {

this.attackPlaces.push(col[this.col+1] + (this.row -1));

}

if ( fields[col[this.col-1] + (this.row - 1)] != undefined

&& fields[col[this.col-1] + (this.row - 1)].piece != null

&& this.color != fields[col[this.col-1] + (this.row -1)].piece.color) {

this.attackPlaces.push(col[this.col-1] + (this.row -1));

}

break;

case "whiteKing":

case "blackKing":

this.findKingMoves();

break;

case "whiteQueen":

case "blackQueen":

this.findRookMoves();

this.findBishupMoves();

break;

case "whiteRook":

case "blackRook":

this.findRookMoves();

break;

case "whiteBishup":

case "blackBishup":

this.findBishupMoves();

break;

case "whiteKnight":

case "blackKnight":

this.knightMoves();

break;

default:

}

}

}

Piece.prototype.findKingMoves = function () {

for (var i = -1; i <= 1; i++) {

for (var j = -1; j <= 1; j++) {

if (i == 0 && j == 0) {} else {

if (fields[col[this.col + i] + (this.row + j)] != undefined

&& fields[col[this.col + i] + (this.row + j)].piece != null) {

if ( this.color != fields[col[this.col + i] + (this.row + j)].piece.color) {

this.attackPlaces.push(col[this.col + i] + (this.row + j));

}

}

else if (fields[col[this.col + i] + (this.row + j)] != undefined){

this.movePlaces.push(col[this.col + i] + (this.row + j));

}

}

}

}

}

// Rook Moves

Piece.prototype.findRookMoves = function () {

var row = this.row;

var coll = this.col;

// moving up

while (true) {

if ( fields[col[this.col] + (row+1)] != undefined

&& fields[col[this.col] + (row+1)].piece == null){

this.movePlaces.push(col[this.col] + (row+1));

row++;

}

else {

if (fields[col[this.col] + (row+1)] != undefined

&& fields[col[this.col] + (row+1)].piece != null

&& this.color != fields[col[this.col] + (row+1)].piece.color) {

console.log(true);

this.attackPlaces.push(col[this.col] + (row+1));

}

row = this.row;

break;

};

}

// moving down

while (true) {

if ( fields[col[this.col] + (row-1)] != undefined

&& fields[col[this.col] + (row-1)].piece == null){

this.movePlaces.push(col[this.col] + (row-1));

row--;

}

else {

if (fields[col[this.col] + (row-1)] != undefined

&& fields[col[this.col] + (row-1)].piece != null

&& this.color != fields[col[this.col] + (row-1)].piece.color) {

console.log(true);

this.attackPlaces.push(col[this.col] + (row-1));

}

row = this.row;

break;

};

}

// moving right

while (true) {

if ( fields[col[coll+1] + row] != undefined

&& fields[col[coll+1] + row].piece == null){

this.movePlaces.push(col[coll+1] + row);

coll++;

}

else {

if (fields[col[coll+1] + row] != undefined

&& fields[col[coll+1] + row].piece != null

&& this.color != fields[col[coll+1] + row].piece.color) {

console.log(true);

this.attackPlaces.push(col[coll+1] + row);

}

coll = this.col;

break;

};

}

// moving left

while (true) {

if ( fields[col[coll-1] + row] != undefined

&& fields[col[coll-1] + row].piece == null){

this.movePlaces.push(col[coll-1] + row);

coll--;

}

else {

if (fields[col[coll-1] + row] != undefined

&& fields[col[coll-1] + row].piece != null

&& this.color != fields[col[coll-1] + row].piece.color) {

console.log(true);

this.attackPlaces.push(col[coll-1] + row);

}

coll = this.col;

break;

};

}

};

// Bishup Moves

Piece.prototype.findBishupMoves = function() {

var i = 1;

// up right

while (true) {

if (fields[col[this.col + i] + (this.row + i)] != undefined

&& fields[col[this.col + i] + (this.row + i)].piece == null) {

this.movePlaces.push(col[this.col + i] + (this.row + i));

i++;

} else {

if (fields[col[this.col + i] + (this.row + i)] != undefined

&& fields[col[this.col + i] + (this.row + i)].piece != null

&& this.color != fields[col[this.col + i] + (this.row + i)].piece.color) {

console.log(true);

this.attackPlaces.push(col[this.col + i] + (this.row + i));

}

i = 1;

break;

}

}

// downLeft

while (true) {

if (fields[col[this.col - i] + (this.row - i)] != undefined

&& fields[col[this.col - i] + (this.row - i)].piece == null) {

this.movePlaces.push(col[this.col - i] + (this.row - i));

i++;

} else {

if (fields[col[this.col - i] + (this.row - i)] != undefined

&& fields[col[this.col - i] + (this.row - i)].piece != null

&& this.color != fields[col[this.col - i] + (this.row - i)].piece.color) {

console.log(true);

this.attackPlaces.push(col[this.col - i] + (this.row - i));

}

i = 1;

break;

}

}

while (true) {

if (fields[col[this.col + i] + (this.row - i)] != undefined

&& fields[col[this.col + i] + (this.row - i)].piece == null) {

this.movePlaces.push(col[this.col + i] + (this.row - i));

i++;

} else {

if (fields[col[this.col + i] + (this.row - i)] != undefined

&& fields[col[this.col + i] + (this.row - i)].piece != null

&& this.color != fields[col[this.col + i] + (this.row - i)].piece.color) {

console.log(true);

this.attackPlaces.push(col[this.col + i] + (this.row - i));

}

i = 1;

break;

}

}

while (true) {

if (fields[col[this.col - i] + (this.row + i)] != undefined

&& fields[col[this.col - i] + (this.row + i)].piece == null) {

this.movePlaces.push(col[this.col - i] + (this.row + i));

i++;

} else {

if (fields[col[this.col - i] + (this.row + i)] != undefined

&& fields[col[this.col - i] + (this.row + i)].piece != null

&& this.color != fields[col[this.col - i] + (this.row + i)].piece.color) {

console.log(true);

this.attackPlaces.push(col[this.col - i] + (this.row + i));

}

i = 1;

break;

}

}

}

Piece.prototype.knightMoves = function () {

let rowMoves = [-1,1,-2,2,-2,2,-1,1], r = 0;

for (i = -2; i<=2; i++){

for (j=0;j<2;j++) {

if (i != 0) {

if (fields[col[this.col + i] + (this.row + rowMoves[r])] != undefined

&& fields[col[this.col + i] + (this.row + rowMoves[r])].piece == null)

this.movePlaces.push(col[this.col + i] + (this.row + rowMoves[r]));

else {

if ( fields[col[this.col + i] + (this.row + rowMoves[r])] != undefined

&& fields[col[this.col + i] + (this.row + rowMoves[r])].piece != null

&& this.color != fields[col[this.col + i] + (this.row + rowMoves[r])].piece.color)

this.attackPlaces.push(col[this.col + i] + (this.row + rowMoves[r]));

}

r++

}

}

}

// this.movePlaces.push(col[this.col - 2] + (this.row - 1)); 1

// this.movePlaces.push(col[this.col - 2] + (this.row + 1)); 3

// this.movePlaces.push(col[this.col - 1] + (this.row - 2)); -1

// this.movePlaces.push(col[this.col - 1] + (this.row + 2)); 3

// this.movePlaces.push(col[this.col + 1] + (this.row - 2)); -3

// this.movePlaces.push(col[this.col + 1] + (this.row + 2)); 1

// this.movePlaces.push(col[this.col + 2] + (this.row - 1)); -3

// this.movePlaces.push(col[this.col + 2] + (this.row + 1)); -1

}

**Simon/script.js**

var yellow = document.getElementById('yellow');

var blue = document.getElementById('blue');

var red = document.getElementById('red');

var green = document.getElementById('green');

var btn = document.getElementsByTagName('button')[0];

var msg = document.getElementById('msg');

var maxlvl = document.getElementsByTagName('h3')[1];

yellow.style.background = 'orange';

blue.style.background = 'darkblue';

red.style.background = 'darkred';

green.style.background = 'darkgreen';

var level = 1;

var levelSpeed = 1000;

var max = 1;

var combo = [];

var picked = [];

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

var div = document.getElementsByClassName('color')[i];

div.addEventListener('click', function (e) {

blink(e.target.id, 100);

picked.push(e.target.id);

console.log(picked);

if (picked.length == combo.length) {

if (checkSolution()) {

msg.innerHTML = "Uspesno zavrsen level: " + level;

level++;

if (level > max) {

max = level;

maxlvl.innerHTML = "Max level achived: " + max;

}

levelSpeed -= 100;

btn.innerHTML = "Start level " + level;

}

else {

msg.innerHTML = "Izabrana kombinacija je netacna";

level = 1;

levelSpeed = 1000;

btn.innerHTML = "Start level " + level;

}

}

})

}

function start() {

msg.innerHTML = "";

console.log(levelSpeed);

picked = [];

var n = level + 3;

combo = comboCreate(n);

console.log(combo);

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

(function (i) {

setTimeout(function () {

blink(combo[i],levelSpeed);

}, (2\*levelSpeed)\*i);

})(i);

}

}

function checkSolution() {

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

if (combo[i] != picked[i]) {

return false;

break;

}

}

return true;

}

function blink(element,speed) {

switch (element) {

case 'yellow':

yellow.style.background = 'yellow';

setTimeout(function() {

yellow.style.background = 'orange';

}, speed)

break;

case 'blue':

blue.style.background = 'lightblue';

setTimeout(function() {

blue.style.background = 'darkblue';

}, speed)

break;

case 'red':

red.style.background = 'red';

setTimeout(function() {

red.style.background = 'darkred';

}, speed)

break;

case 'green':

green.style.background = 'lightgreen';

setTimeout(function() {

green.style.background = 'darkgreen';

}, speed)

break;

default:

}

}

function comboCreate(n) {

var array = [];

for (i=0;i<n;i++){

var rand = Math.floor(Math.random() \* 100) + 1;

if (rand <= 25) {

array.push('yellow');

}

else if (rand <=50) {

array.push('blue');

}

else if (rand <= 75) {

array.push('red');

}

else {

array.push('green');

}

}

return array;

}

**MinesWeeper/script.js**

var size = 15;

var numOfMines = 40;

var cells = [];

var table = document.getElementById('table');

var restart = false;

function setUp() {

for (i = 0; i < size; i++) {

cells[i] = [];

for (j = 0; j< size;j++) {

cells[i][j] = new Cell((i<10?"0"+i:i)+""+(j<10?"0"+j:j));

}

}

var x = 0;

do {

var randX = Math.floor(Math.random() \* size);

var randY = Math.floor(Math.random() \* size);

if (!cells[randX][randY].mine) {

cells[randX][randY].mine = true;

x++;

}

} while(x<numOfMines);

}

function draw() {

table.style.width = "" + (size \* 25) + "px";

for (i = 0; i < size; i++) {

for (j = 0; j< size;j++) {

cells[i][j].countMines();

table.innerHTML += cells[i][j].draw;

}

table.innerHTML += "<div style='clear:both; width:0px; height:0px; border:none'>"

}

}

function reveal(a) {

console.log(a.id);

if (!timeStarted) {

timeStart();

}

var i = parseInt(a.id.substring(0,2));

var j = parseInt(a.id.substring(2,4));

if (cells[i][j].mine){

gameOver();

}

cells[i][j].reveal();

if (cells[i][j].mineCount == 0) {

emptyReveal(i,j);

}

}

function mark() {

for (i = 0; i< size\*size; i ++) {

document.getElementsByClassName('cell')[i].addEventListener('mousedown', function(e) {

var i = parseInt(e.target.id.substring(0,2));

var j = parseInt(e.target.id.substring(2,4));

if (e.button == 1 && !cells[i][j].revealed) {

if (e.target.innerHTML == 'X')

{

e.target.innerHTML = '';

e.target.style.color = 'black';

markRemoved();

}

else {

e.target.innerHTML = 'X';

e.target.style.color = 'white';

markUsed();

}

}

})

}

}

function emptyReveal(i,j) {

setTimeout(function () {

for(x = -1; x <= 1; x++) {

for (var y = -1; y <=1; y++) {

var a = i + x;

var b = j + y;

if (a > -1 && b > -1 && a < size && b < size) {

if(!cells[a][b].mine && !cells[a][b].revealed){

if(cells[a][b].mineCount==0){

cells[a][b].reveal();

emptyReveal(a,b);

}

else {

cells[a][b].reveal();

}

}

}

}

}

},50)

}

function gameOver() {

var over = true;

for (i = 0; i < size; i++) {

for (j = 0; j< size;j++) {

cells[i][j].reveal(over);

}

}

clearInterval(interval);

timeElapsed = 0;

timeStarted = false;

}

function newGame() {

table.innerHTML = "";

setUp();

draw();

mark();

writeStats(numOfMines);

}

newGame();

**MinesWeeper/cell.js**

function Cell(id) {

this.mine = false;

this.revealed = false;

this.mineCount = 0;

this.id = id;

this.draw = "<div id='"+this.id+"' class='cell' onclick='reveal(this)'></div>";

}

Cell.prototype.reveal = function(over) {

this.revealed = true;

var node = document.getElementById(this.id);

node.style.color = 'black';

if (this.mine) {

node.innerHTML = "";

node.style.background = 'radial-gradient(circle, red, yellow)';

}

else {

node.style.background = 'gainsboro';

if (this.mineCount > 0 && !over)

{

node.innerHTML = this.mineCount;

}

}

};

Cell.prototype.countMines = function() {

var i = parseInt(this.id.substring(0,2));

var j = parseInt(this.id.substring(2,4));

for(x = -1; x <= 1; x++) {

for (var y = -1; y <=1; y++) {

var a = i + x;

var b = j + y;

if (a > -1 && b > -1 && a < size && b < size) {

if(cells[a][b].mine){

this.mineCount++;

}

}

}

}

}

**MinesWeeper/options.js**

var options = document.getElementById('options');

options.innerHTML += "<p>Velicina: <input type='number' value='15' max='20' min='10'></p>";

options.innerHTML += "<p>Broj mina: <input type='number' value='40'></p>";

options.innerHTML += "<button onclick='start()'>Start</button>";

options.innerHTML += "<button onclick='cancel()'>Cancel</button>";

options.style.display = 'none';

var displayed = false;

function toggleOptions() {

if (!displayed) {

options.style.display = 'block';

displayed = true;

}

else {

options.style.display = 'none';

displayed = false;

}

}

function start() {

size = document.getElementsByTagName('input')[0].value;

numOfMines = document.getElementsByTagName('input')[1].value;

options.style.display = 'none';

newGame();

}

function cancel() {

options.style.display = 'none';

}

**MinesWeeper/stats.js**

var stats = document.getElementById('game-stats');

var leftToMark = 0;

var timeElapsed = 0;

var timeStarted = false;

var interval = setInterval(function(){}, 1000);;

function writeStats(numOfMines) {

leftToMark = numOfMines;

stats.innerHTML = `<p>Preostalo za označavanje: ` +

`<big><b><span id='left'>${leftToMark}</span></b></big></p>` +

`Proteklo vreme: <big><b><span id='time'>${timeElapsed}</span></b></big>`;

}

function markUsed() {

leftToMark--;

document.getElementById('left').innerHTML = leftToMark;

}

function markRemoved() {

leftToMark++;

document.getElementById('left').innerHTML = leftToMark;

}

function timeStart() {

timeStarted = true;

interval = setInterval(timer, 1000);

}

function timer() {

timeElapsed++;

var min;

var sec;

if (timeElapsed <= 59) {

document.getElementById('time').innerHTML = timeElapsed + "";

}

else if (timeElapsed > 59) {

sec = timeElapsed % 60;

min = Math.floor(timeElapsed / 60);

document.getElementById('time').innerHTML = min + " : " + (sec<10? "0" + sec: sec);

}

else if (restart) {

timeElapsed = 0;

document.getElementById('time').innerHTML = "0";

}

}