<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>chat</title>

<style>

/\* 테이블과 이미지의 간격을 딱맞추기위해 0으로 초기화\*/

table{

border:0px;

border-collapse:collapse;

border-spacing: 0;

}

img,tr,td{

width: 30px;

height: 30px;

margin: 0px;

padding: 0px;

line-height: 0px;

}

.fixed\_div {

position:fixed;

bottom:0;

width:100%;

height:50px;

background:#ccc;

display: inline-block;

}

</style>

</head>

<body>

<!--<embed id="player" src="ab.mp3" autostart=true hidden=true loop=false type="audio/mp3">-->

<form id = "btn">

<input type='button' value ="보드판 생성" id = "CBT" onclick = "CreateTable()">

<input type="text" value = "보드판 생성 버튼을 누르세요" id = "Info" size = "40">

</form>

<hr>

<table border="2" style="border-color: black">

<tbody>

<tr>

<td id = "tab\_par">

<table id="tab">

</table>

</td>

</tr>

</tbody>

</table>

<div style="font-size:2.0em; font-family: 궁서체" class="fixed\_div" id = "fix"></div>

<div id="test"></div>

<script src="/socket.io/socket.io.js"></script>

<script src="http://code.jquery.com/jquery-1.11.1.js"></script>

<script src=""></script>

<script>

var socket = io();

var Global\_values = {

map\_size:0,

ing: 0,

random\_ing: 0,

random\_check: 0,

eventSave: null,

isIntervalPlaying: 0,

head\_location: "0.000",

tail\_count: -1,

check: 0,

name\_count: 2,

move: 0,

SleepTIme: 100,

bomb\_count: 0,

fast\_count: 0,

slow\_count: 0,

long\_count: 0,

temp: null

};

function bar\_info() {

var bar = document.getElementById("fix");

bar.innerHTML = "<font color=#006400>몸 길이:"+(-Global\_values.tail\_count)+"</font>";

bar.innerHTML +="<font color=#000000> 폭탄:"+(Global\_values.bomb\_count)+"</font>";

bar.innerHTML +="<font color=#FF1493> 먹이:"+(Global\_values.long\_count)+"</font>";

bar.innerHTML +="<font color=#FF0000> Fast item:"+(Global\_values.fast\_count)+"</font>";

bar.innerHTML +="<font color=#0000FF> Slow item:"+(Global\_values.slow\_count)+"</font>";

bar.innerHTML +="<font color=#800080> Delay Speed:"+(Global\_values.SleepTIme)+"</font>";

bar.innerHTML +="<font color=#000000> // P 입력시 일시정지</font>";

}

function Random\_Create\_Item() {

var random\_time = Math.floor(Math.random() \* 10);

switch(random\_time){

case 1:

RandomItem("/long\_item");

Global\_values.long\_count++;

break;

case 2:

RandomItem("/fast");

Global\_values.fast\_count++;

break;

case 3:

RandomItem("/slow");

Global\_values.slow\_count++;

break;

case 4:

RandomItem("/boom");

Global\_values.bomb\_count++;

break;

case 5:

case 6:

case 6:

RandomItem("/long\_item");

Global\_values.long\_count++;

break;

}

}

function RandomItem(item) {

//x,y랜덤값 생성

while(true){

//생성된 위치에 아무거도 없을동안 반복 -> 겹치지 않게

var ran\_x = Math.floor(Math.random() \* Global\_values.map\_size);

var ran\_y = Math.floor(Math.random() \* Global\_values.map\_size);

//소숫점 만들기

if (ran\_y < 10)

ran\_y = ".00" + ran\_y;

else if (ran\_y < 100)

ran\_y = ".0" + ran\_y;

else

ran\_y = "." + ran\_y;

//합체

ran\_x += ran\_y;

var Long\_Item = document.getElementById(ran\_x + "");

//지금 위치에 아무거도 없을시 그자리에 그림 출력

if(Long\_Item.innerHTML == '<img src="/blank">')

break;

}

Long\_Item.innerHTML = '<img src='+item+'>';

/\*

//머리정보를 서버에 전송

socket.emit('move loc',Long\_Item+"",item);

//다른 클라이언트들의 머리정보를 서버로부터 가져옴

socket.on('moved loc', function(head\_loc, imge){

var head\_temp = document.getElementById(head\_loc);

head\_temp.innerHTML = '<img src='+item+'>';

});

\*/

}

//아이템을 먹었는지 검사

function eat\_item(head\_point) {

var tail\_point = null;

//꼬리 길어지는 아이템을 먹었을시

if(head\_point.innerHTML == '<img src="/long\_item">'){

Global\_values.tail\_count--;

Global\_values.long\_count--;

}

else {

//꼬리 길어지는 아이템을 안먹었을시 마지막꼬리 짜르기

tail\_point = document.getElementsByName((Global\_values.name\_count + Global\_values.tail\_count) + "");

tail\_point[0].innerHTML = "<img src='/blank'>";

//머리정보를 서버에 전송

socket.emit('move loc',(tail\_point[0].id)+"","<img src='/blank'>");

//다른 클라이언트들의 머리정보를 서버로부터 가져옴

socket.on('moved loc', function(head\_loc, imge){

var head\_temp = document.getElementById(head\_loc);

head\_temp.innerHTML = imge;

});

}

//빨라지는 아이템을 먹었을시 속도증가

if(head\_point.innerHTML == '<img src="/fast">'){

Global\_values.fast\_count--;

clearInterval(Global\_values.ing);

Global\_values.ing = setInterval("Start(Global\_values.eventSave)", Global\_values.SleepTIme-=10);

}

//느려지는 아이템을 먹었을시 속도 감소

if(head\_point.innerHTML == '<img src="/slow">'){

Global\_values.slow\_count--;

clearInterval(Global\_values.ing);

Global\_values.ing = setInterval("Start(Global\_values.eventSave)", Global\_values.SleepTIme+=5);

}

}

// 게임다시시작 버튼 눌렀을시

function Replay\_Game() {

Global\_values.ing = 0;

Global\_values.random\_ing = 0;

Global\_values.random\_check = 0;

Global\_values.eventSave = null;

Global\_values.isIntervalPlaying = 0;

Global\_values.head\_location = "0.000";

Global\_values.tail\_count = -1;

Global\_values.check = 1;

Global\_values.name\_count = 2;

Global\_values.move = 0;

Global\_values.SleepTIme = 100;

//생성 삭제

var Get\_Table = document.getElementById("tab");

while(Get\_Table.childElementCount>0)

Get\_Table.removeChild(Get\_Table.lastChild);

//삭제버튼 -> 시작버튼으로 변경

var Cre\_BT = document.getElementById("DBT");

Cre\_BT.setAttribute('value', '보드판 생성');

Cre\_BT.setAttribute('onclick', 'CreateTable()');

Cre\_BT.setAttribute('id', 'CBT');

var info = document.getElementById("Info");

info.setAttribute('value','보드판 생성 버튼을 누르세요');

}

function CreateTable() {

//테이블 동적생성

var Table\_par = document.getElementById("tab\_par");

var Get\_Table = document.getElementById("tab");

//보드판 생성시 한번 테이블을 지우고 새로 생성

if(Table\_par != null)

Table\_par.children[0].remove(this);

var Create\_TAB = document.createElement("table");

var Create\_TR = document.createElement("TR");

var Create\_TD = document.createElement("TD");

Create\_TAB.id="tab";

Global\_values.check = 0;

//삭제된 테이블을 추가하고 생성된 테이블의 id값을 추가

//입력 받은 사이즈만큼 A\*A 길이 생성

for(var i = 0; i<Global\_values.map\_size; i++) {

Create\_TR = document.createElement("TR");

for (var j = 0; j < Global\_values.map\_size; j++) {

Create\_TD = document.createElement("TD");

Create\_TD.innerHTML = '<img src="/blank">';

if(j<10)

Create\_TD.setAttribute('id',i+".00"+j);

else if(j<100)

Create\_TD.setAttribute('id',i+".0"+j);

else

Create\_TD.setAttribute('id',i+"."+j);

Create\_TR.appendChild(Create\_TD);

Create\_TAB.appendChild(Create\_TR);

}

//테이블에 적용

Table\_par.appendChild(Create\_TAB);

Global\_values.temp = Get\_Table;

}

//아이템 생성

for(var i = 0; i<Global\_values.map\_size/3; i++) {

RandomItem("/long\_item");

Global\_values.long\_count++;

//RandomItem("/boom");

Global\_values.bomb\_count++;

}

for(var i = 0; i<Global\_values.map\_size/5; i++){

RandomItem("/fast");

Global\_values.fast\_count++;

RandomItem("/slow");

Global\_values.slow\_count++;

}

//시작버튼 -> 삭제버튼으로 변경

var Cre\_BT = document.getElementById("CBT");

Cre\_BT.setAttribute('value', '보드판 감추기+다시하기');

Cre\_BT.setAttribute('onclick', 'Replay\_Game()');

Cre\_BT.setAttribute('id', 'DBT');

//다음 설명 적기

var info = document.getElementById("Info");

info.setAttribute('value','방향키를 누를시 게임시작/ P키 누를시 일시정지');

//테이블을 서버에게 넘겨주는 과정

var origin\_table = document.getElementById("tab");

socket.emit('crt table',toJSON(origin\_table), Global\_values.map\_size);

//서버로부터 테이블을 받는 과정

socket.on('created table', function (cr\_tab, input\_e) {

cr\_tab = toDOM(cr\_tab);

var Get\_Table = document.getElementById("tab\_par");

var child\_Table = document.getElementById("tab");

//맵크기를 다른 클라이언트랑 동기화

Global\_values.map\_size = input\_e;

if(child\_Table != null){

Get\_Table.removeChild(child\_Table);

Get\_Table.appendChild(cr\_tab);

}

//시작점 랜덤위치로 지정

var ran\_1 = parseInt(Math.floor(Math.random()\* Global\_values.map\_size))+"";

var ran\_2 = parseInt(Math.floor(Math.random()\* Global\_values.map\_size))+"";

if(Global\_values.map\_size<=10)

Global\_values.head\_location = ran\_1+".00"+ran\_2;

else if(Global\_values.map\_size<=100){

if(ran\_2 < 10)

Global\_values.head\_location = ran\_1+".00"+ran\_2;

else

Global\_values.head\_location = ran\_1+".0"+ran\_2;

}

else

Global\_values.head\_location = ran\_1+"."+ran\_2;

//머리부분 출력

var Create\_First = document.getElementById(Global\_values.head\_location+"");

Create\_First.innerHTML = "<img src='/head'>";

Create\_First.setAttribute('name', (++Global\_values.name\_count)+"");

//머리정보를 서버에 전송

socket.emit('move loc',Global\_values.head\_location,"<img src='/head'>");

//다른 클라이언트들의 머리정보를 서버로부터 가져옴

socket.on('moved loc', function(head\_loc, imge){

var head\_temp = document.getElementById(head\_loc+"");

head\_temp.innerHTML = imge;

});

//시작화면을 중앙으로 전환 - 현재 window의 가로 세로길이를 구하여 중앙으로

var offset = $(Create\_First).offset();

offset.left -= (window.innerWidth/2);

offset.top -= (window.innerHeight/2-90);

$('html, body').animate({scrollLeft : offset.left, scrollTop : offset.top},0);

});

//맨밑 바 정보 출력

bar\_info();

}

//맵 사이즈 정하기

while(true) {

Global\_values.map\_size = prompt("가로\*세로 길이를 입력하세요(2-150) @@ 권장:100 @@");

if(Global\_values.map\_size>=2 && Global\_values.map\_size<=150){

socket.emit('map size', Global\_values.map\_size);

CreateTable(Global\_values.map\_size);

break;

}

}

window.onkeydown = function (event) {

var testKeycode = document.getElementById("test");

testKeycode.innerHTML = event.keyCode; // 주석 제거시 현재 진행방향 표시

var check\_BT = document.getElementById("DBT");

if((event.keyCode=='37' || event.keyCode=='38'|| event.keyCode=='39'|| event.keyCode=='40' || event.keyCode == '80')

&& check\_BT != null) {

Global\_values.eventSave = event;

//event가 다른 함수로 넘어갈경우 값이 사라져서 변수선언해서 잡아 놓음

//또한, 다른 키 입력시 바뀌는값을 Interval함수에 넘겨줌

if(Global\_values.isIntervalPlaying === 0) {

//2번째인자값ms마다 Start 함수 실행, 최초 한번만 Interval호출(여러개호출 방지)

Global\_values.ing = setInterval("Start(Global\_values.eventSave)", Global\_values.SleepTIme);

Global\_values.isIntervalPlaying = 1;

}

}

}

//매ms마다 실행될 함수

function Start(event) {

var fir = parseInt(Global\_values.head\_location);

var sec = Math.round((parseFloat(Global\_values.head\_location)-parseInt(fir))\*1000);

var check\_BT = document.getElementById("DBT");

var head\_point = document.getElementById(Global\_values.head\_location+"");

//하단바 값 최신화

bar\_info();

if(check\_BT == null || event.keyCode == 80){

//게임도중 삭제버튼을 누를시 실행 종료

clearInterval(Global\_values.ing);

clearInterval(Global\_values.random\_ing);

Global\_values.isIntervalPlaying = 0;

Global\_values.random\_check = 0;

return;

}

//이전키가 P(Stop)키를 눌렀으면 랜덤생성쓰레드 run

if(Global\_values.move.keyCode == 80 || Global\_values.random\_check == 0){

Global\_values.random\_ing = setInterval("Random\_Create\_Item()", 1000);

Global\_values.random\_check = 1;

}

//check로 버튼눌렀을시 실행되는것을 제어

//현재키와 이전키가 정반대 키를 눌렀을시: 현재키 <= 이전키

if(Global\_values.check!=1) {

if(Global\_values.move != null) {

if (Global\_values.move.keyCode == 37) {

if (event.keyCode == 39) {

event = Global\_values.move;

}

}

else if (Global\_values.move.keyCode == 38) {

if (event.keyCode == 40) {

event = Global\_values.move;

}

} else if (Global\_values.move.keyCode == 39) {

if (event.keyCode == 37) {

event = Global\_values.move;

}

} else if (Global\_values.move.keyCode == 40) {

if (event.keyCode == 38) {

event = Global\_values.move;

}

}

}

//화면 자동 조절 - 현재 window의 가로 세로길이를 구하여 중앙으로

var offset = $(head\_point).offset();

offset.left -= (window.innerWidth/2);

offset.top -= (window.innerHeight/2-90);

//예외 발생시 GameOver

try {

head\_point.innerHTML = "<img src='/tail'>";

//꼬리정보를 서버에 전송

socket.emit('move loc',Global\_values.head\_location,"<img src='/tail'>");

//다른 클라이언트들의 머리정보를 서버로부터 가져옴

socket.on('moved loc', function(head\_loc, imge){

var head\_temp = document.getElementById(head\_loc+"");

head\_temp.innerHTML = imge;

});

switch (event.keyCode) {

//←

case 37:

Global\_values.head\_location = (parseFloat(Global\_values.head\_location) - 0.001).toFixed(3);

head\_point = document.getElementById(Global\_values.head\_location + "");

//각 case의 offset 설정은 도중 다른화살표 입력시 화면흔들림을 줄여준다.

offset.left -= 30;

break;

//↑

case 38:

Global\_values.head\_location = (parseFloat(Global\_values.head\_location) - 1.000).toFixed(3);

head\_point = document.getElementById(Global\_values.head\_location + "");

offset.top -= 30;

break;

//→

case 39:

Global\_values.head\_location = (parseFloat(Global\_values.head\_location) + 0.001).toFixed(3);

head\_point = document.getElementById(Global\_values.head\_location + "");

offset.left += 30;

break;

//↓

case 40:

Global\_values.head\_location = (parseFloat(Global\_values.head\_location) + 1.000).toFixed(3);

head\_point = document.getElementById(Global\_values.head\_location + "");

offset.top += 30;

break;

}

Global\_values.move = event;

//이번에 밟은 땅이 벽이나 몸통,폭탄이면 에러발생

if(head\_point.innerHTML == '<img src="/head">' ||

head\_point.innerHTML == '<img src="/tail">' ||

head\_point.innerHTML == '<img src="/boom">' ||

head\_point.innerHTML == null

){

throw "E";

}

//플레이어가 움직인 위치정보를 중복없이 각<td>의 name속성에 저장

head\_point.setAttribute('name', (++Global\_values.name\_count)+"");

//아이템을 먹었는지 검사

eat\_item(head\_point);

//이번에 밟은 땅이 벽이나 몸통이 아니면 머리 생성

head\_point.innerHTML = '<img src="/head">';

//현재 화면을 머리의 중앙으로 이동(이동시간 0초 지정)

$('html, body').animate({scrollLeft : offset.left, scrollTop : offset.top},0);

//머리정보를 서버에 전송

socket.emit('move loc',Global\_values.head\_location,"<img src='/head'>");

//다른 클라이언트들의 머리정보를 서버로부터 가져옴

socket.on('moved loc', function(head\_loc, imge){

var head\_temp = document.getElementById(head\_loc+"");

head\_temp.innerHTML = imge;

});

}catch(exception){

alert("Game Over");

head\_point.innerHTML = '<img src="/head">';

clearInterval(Global\_values.ing);

clearInterval(Global\_values.random\_ing);

//GameOver만족시 쓰레드 전부 종료

Global\_values.isIntervalPlaying = 1;

Global\_values.random\_check = 1;

return;

}

}

}

//--------------객체를 서버로 넘기기위한 JSON 사용 ----------------------

function toJSON(node) {

node = node || this;

var obj = {

nodeType: node.nodeType

};

if (node.tagName) {

obj.tagName = node.tagName.toLowerCase();

} else

if (node.nodeName) {

obj.nodeName = node.nodeName;

}

if (node.nodeValue) {

obj.nodeValue = node.nodeValue;

}

var attrs = node.attributes;

if (attrs) {

var length = attrs.length;

var arr = obj.attributes = new Array(length);

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

attr = attrs[i];

arr[i] = [attr.nodeName, attr.nodeValue];

}

}

var childNodes = node.childNodes;

if (childNodes) {

length = childNodes.length;

arr = obj.childNodes = new Array(length);

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

arr[i] = toJSON(childNodes[i]);

}

}

return obj;

}

function toDOM(obj) {

if (typeof obj == 'string') {

obj = JSON.parse(obj);

}

var node, nodeType = obj.nodeType;

switch (nodeType) {

case 1: //ELEMENT\_NODE

node = document.createElement(obj.tagName);

var attributes = obj.attributes || [];

for (var i = 0, len = attributes.length; i < len; i++) {

var attr = attributes[i];

node.setAttribute(attr[0], attr[1]);

}

break;

case 3: //TEXT\_NODE

node = document.createTextNode(obj.nodeValue);

break;

case 8: //COMMENT\_NODE

node = document.createComment(obj.nodeValue);

break;

case 9: //DOCUMENT\_NODE

node = document.implementation.createDocument();

break;

case 10: //DOCUMENT\_TYPE\_NODE

node = document.implementation.createDocumentType(obj.nodeName);

break;

case 11: //DOCUMENT\_FRAGMENT\_NODE

node = document.createDocumentFragment();

break;

default:

return node;

}

if (nodeType == 1 || nodeType == 11) {

var childNodes = obj.childNodes || [];

for (i = 0, len = childNodes.length; i < len; i++) {

node.appendChild(toDOM(childNodes[i]));

}

}

return node;

}

</script>

</body>

</html>