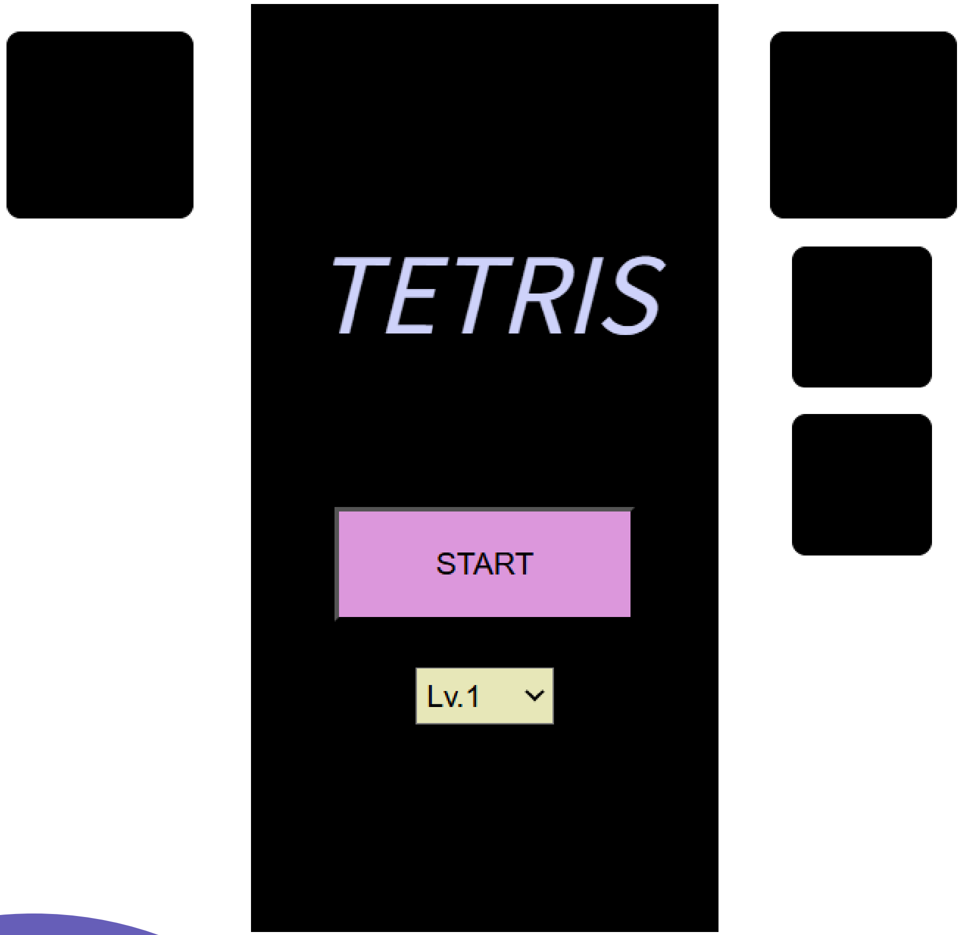
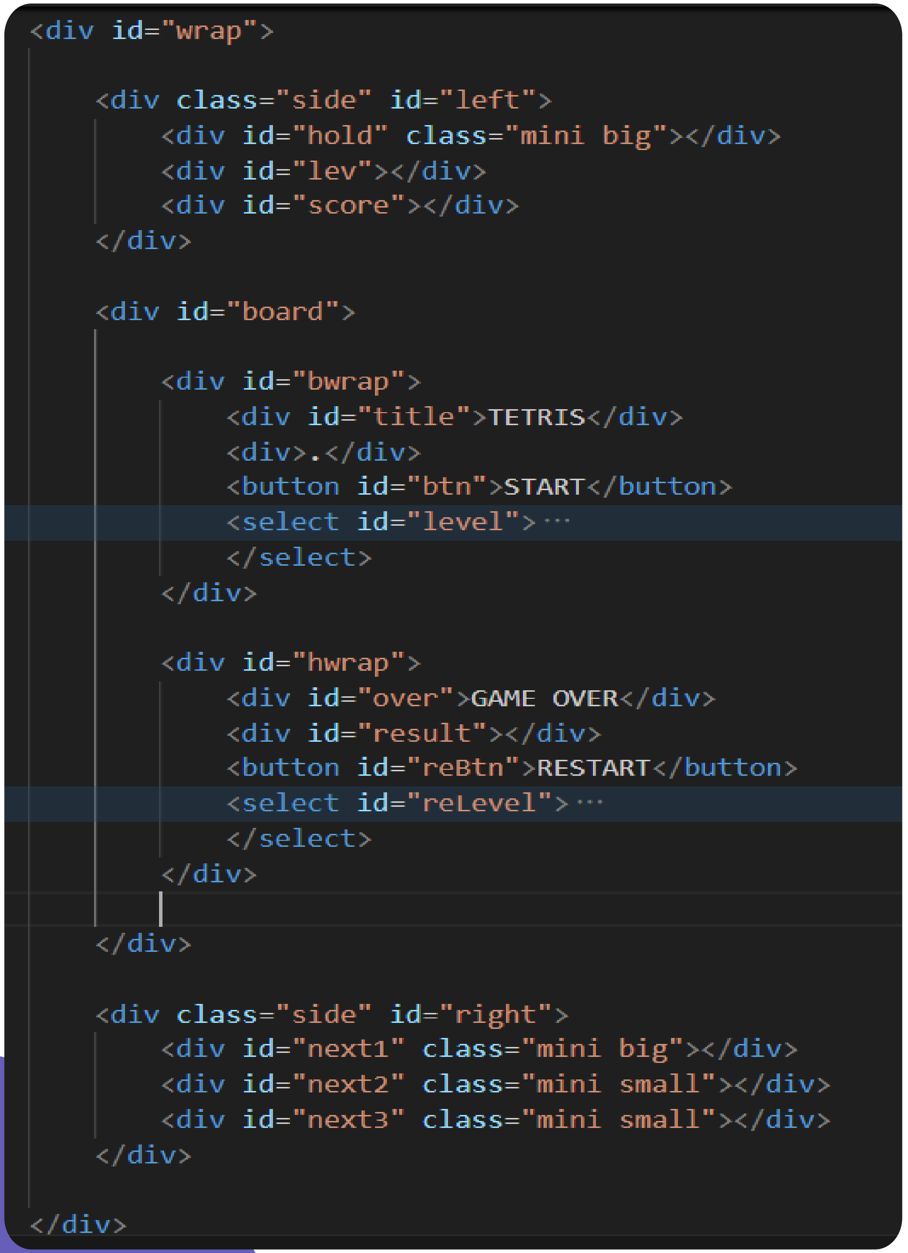
테트리스



- 블록을 쌓아서 한 줄을 다 채우면 그 줄은 사라지면서 점수를 획득하고, 블록이 맨 윗줄까지 다 쌓이면 패배하는 게임

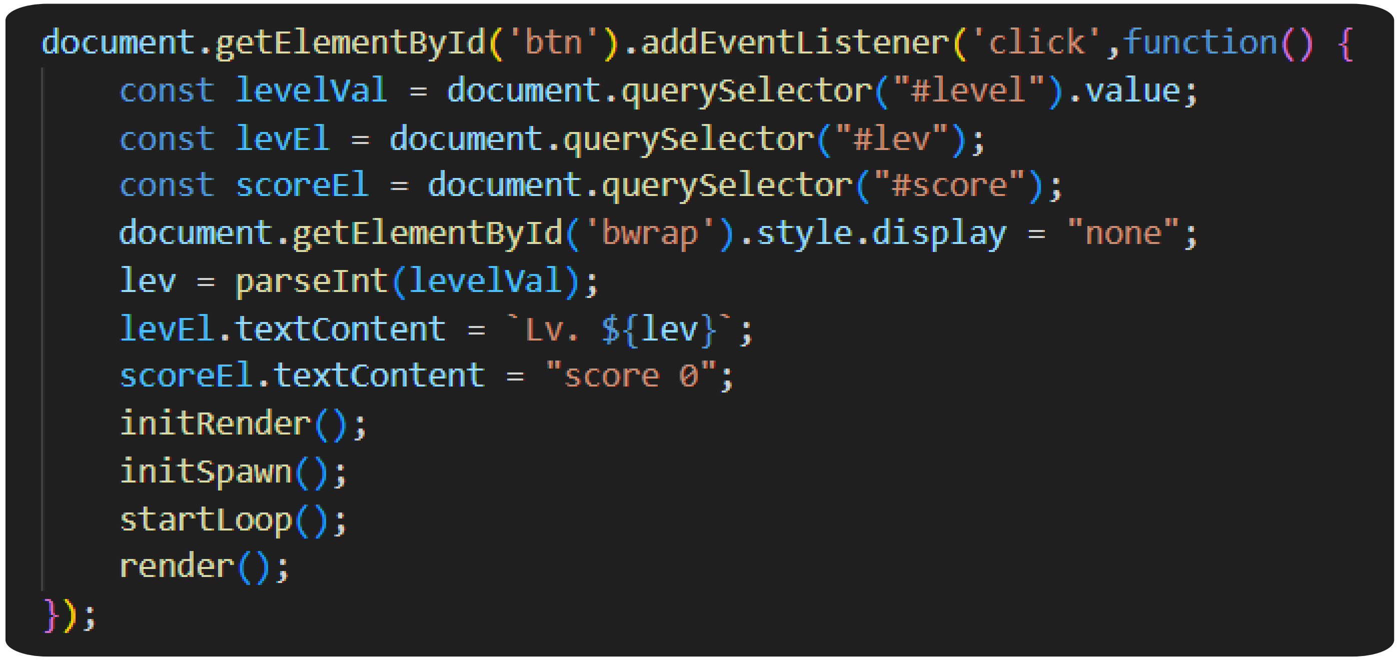
- START 버튼 클릭 시 게임 시작

- 방향키 : 이동
- SPACE : 블록 다운
- SHIFT-LEFT : 블록 홀딩



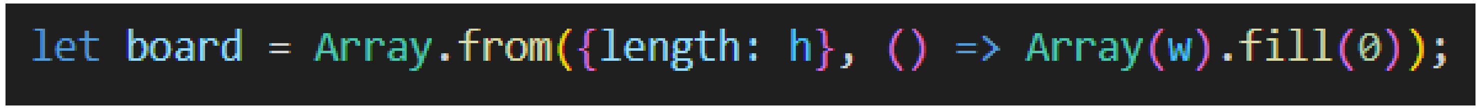
```html
<div id="wrap">

    <div class="side" id="left">
        <div id="hold" class="mini big"></div>
        <div id="lev"></div>
        <div id="score"></div>
    </div>

    <div id="board">

        <div id="bwrap">
            <div id="title">TETRIS</div>
            <div>.</div>
            <button id="btn">START</button>
            <select id="level">···
            </select>
        </div>

        <div id="hwrap">
            <div id="over">GAME OVER</div>
            <div id="result"></div>
            <button id="reBtn">RESTART</button>
            <select id="reLevel">···
            </select>
        </div>

    </div>

    <div class="side" id="right">
        <div id="next1" class="mini big"></div>
        <div id="next2" class="mini small"></div>
        <div id="next3" class="mini small"></div>
    </div>

</div>
```

- 블록 = 정사각형집합
- 보드판을 좌표평면화

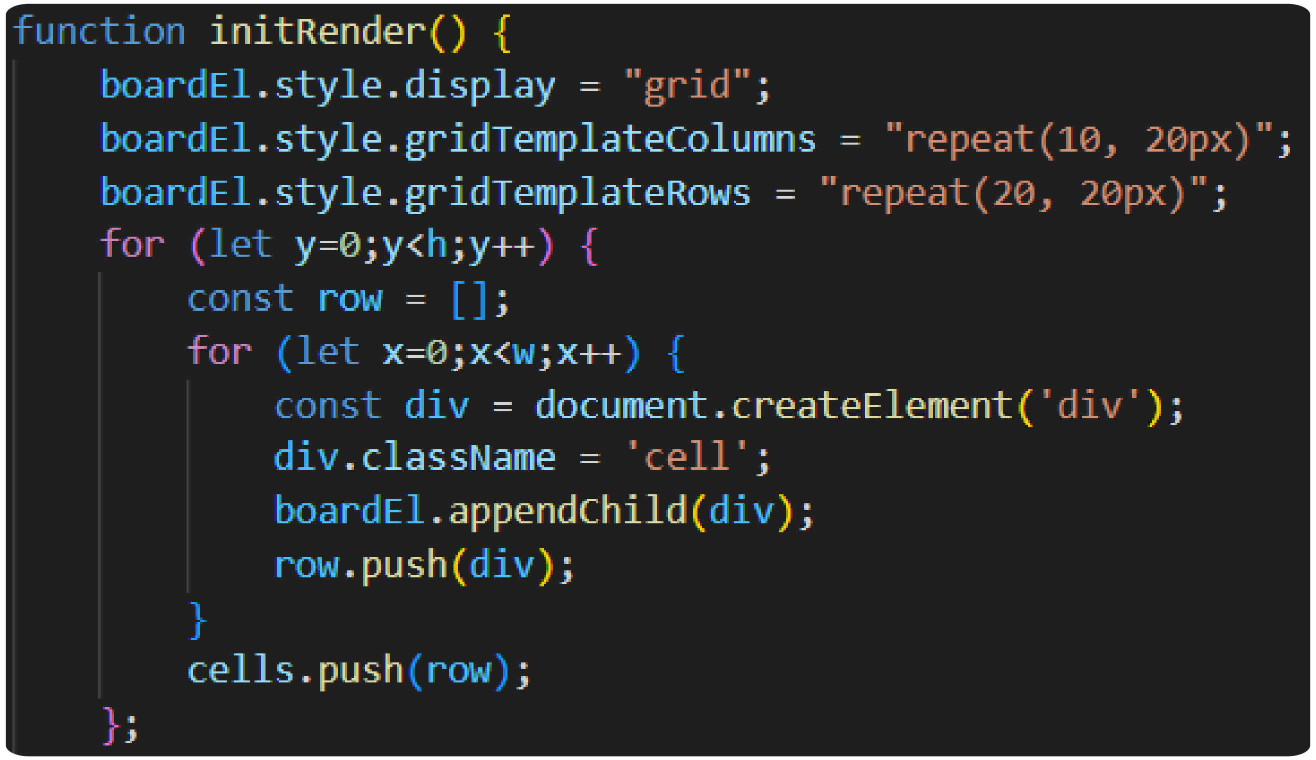


```
document.getElementById('btn').addEventListener('click',function() {
    const levelVal = document.querySelector("#level").value;
    const levEl = document.querySelector("#lev");
    const scoreEl = document.querySelector("#score");
    document.getElementById('bwrap').style.display = "none";
    lev = parseInt(levelVal);
    levEl.textContent = `Lv. ${lev}`;
    scoreEl.textContent = "score 0";
    initRender();
    initSpawn();
    startLoop();
    render();
});
```

- START 버튼

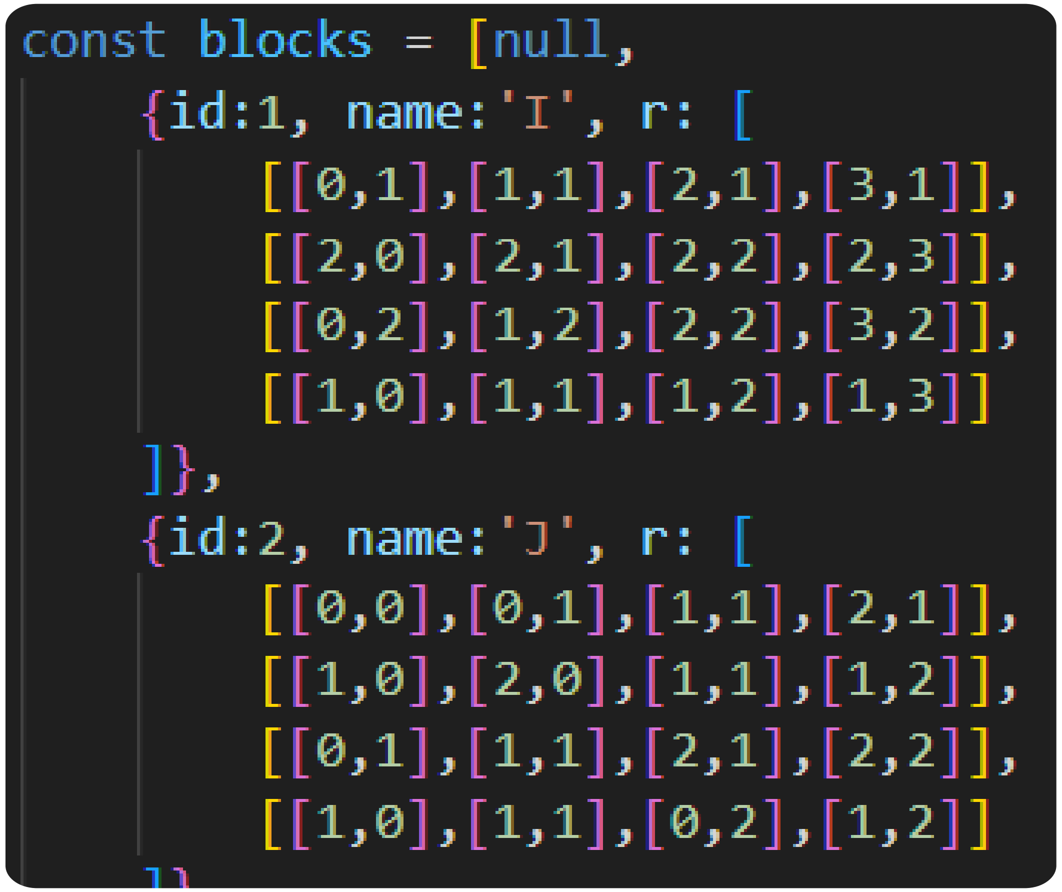


```
let board = Array.from({length: h}, () => Array(w).fill(0));
```



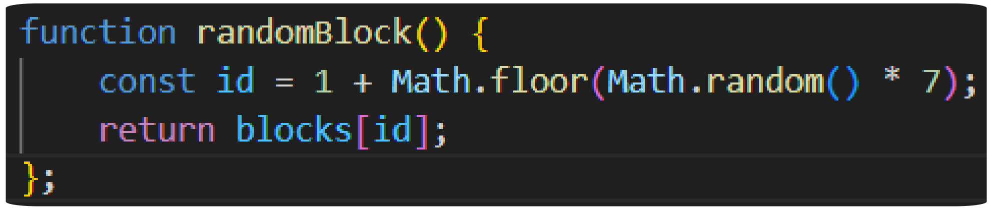
```
function initRender() {
    boardEl.style.display = "grid";
    boardEl.style.gridTemplateColumns = "repeat(10, 20px)";
    boardEl.style.gridTemplateRows = "repeat(20, 20px)";
    for (let y=0;y<h;y++) {
        const row = [];
        for (let x=0;x<w;x++) {
            const div = document.createElement('div');
            div.className = 'cell';
            boardEl.appendChild(div);
            row.push(div);
        }
        cells.push(row);
    };
```

- board = 좌표평면(10x20)
- initRender : cell 생성
- cells : cell 집합
- board와 cells의 index = 좌표

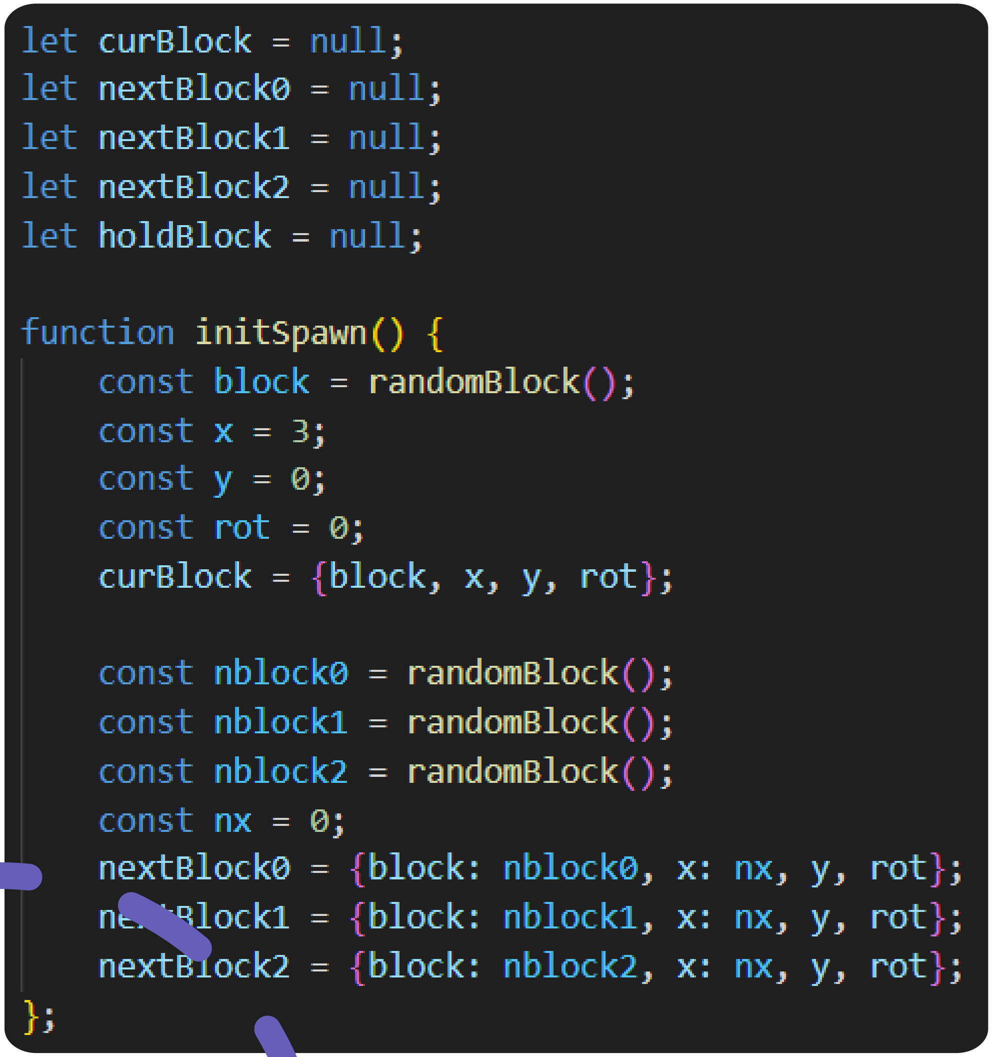


```
const blocks = [null,
    {id:1, name:'I', r: [
        [[0,1],[1,1],[2,1],[3,1]],
        [[2,0],[2,1],[2,2],[2,3]],
        [[0,2],[1,2],[2,2],[3,2]],
        [[1,0],[1,1],[1,2],[1,3]]
    ]},
    {id:2, name:'J', r: [
        [[0,0],[0,1],[1,1],[2,1]],
        [[1,0],[2,0],[1,1],[1,2]],
        [[0,1],[1,1],[2,1],[2,2]],
        [[1,0],[1,1],[0,2],[1,2]]
```

- blocks : block객체 배열
- r : block의 상대좌표
- id : board좌표에 입력될 값

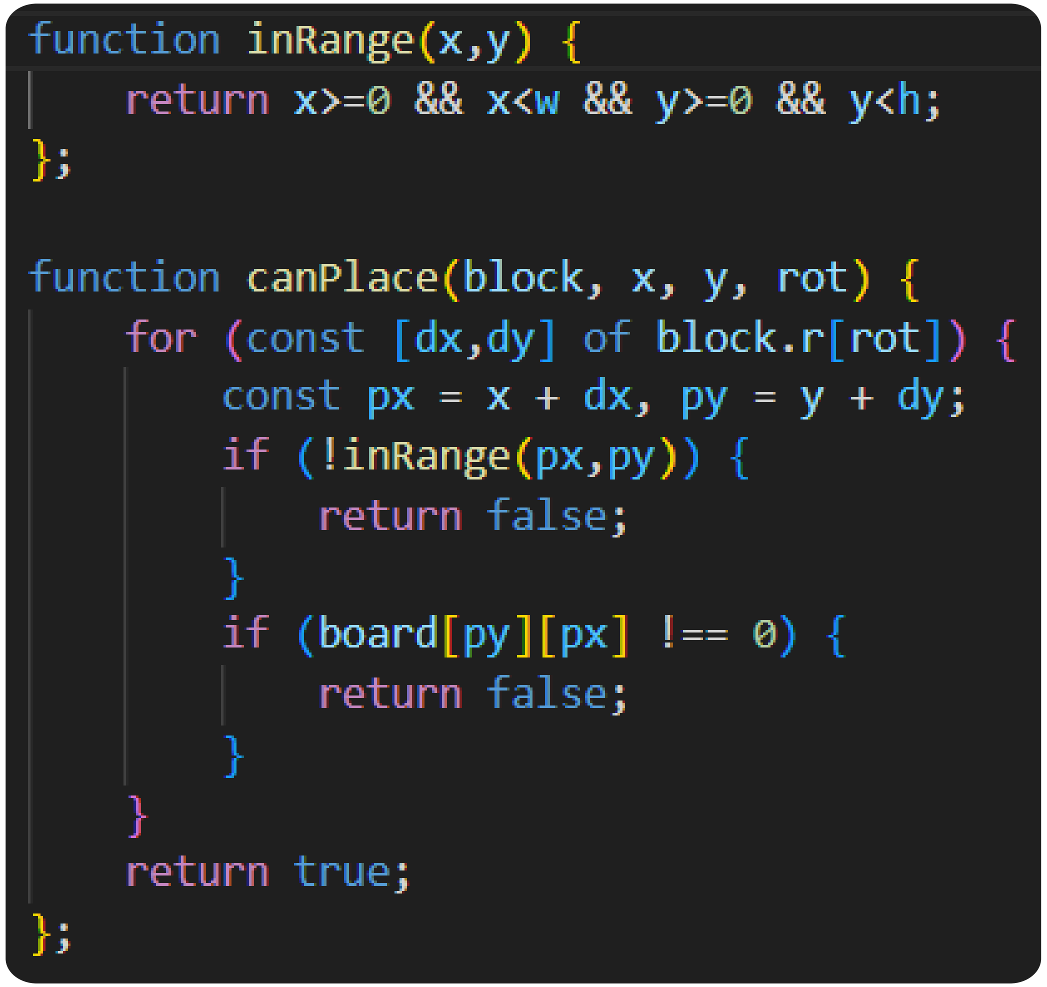


```javascript
function randomBlock() {
    const id = 1 + Math.floor(Math.random() * 7);
    return blocks[id];
};
```



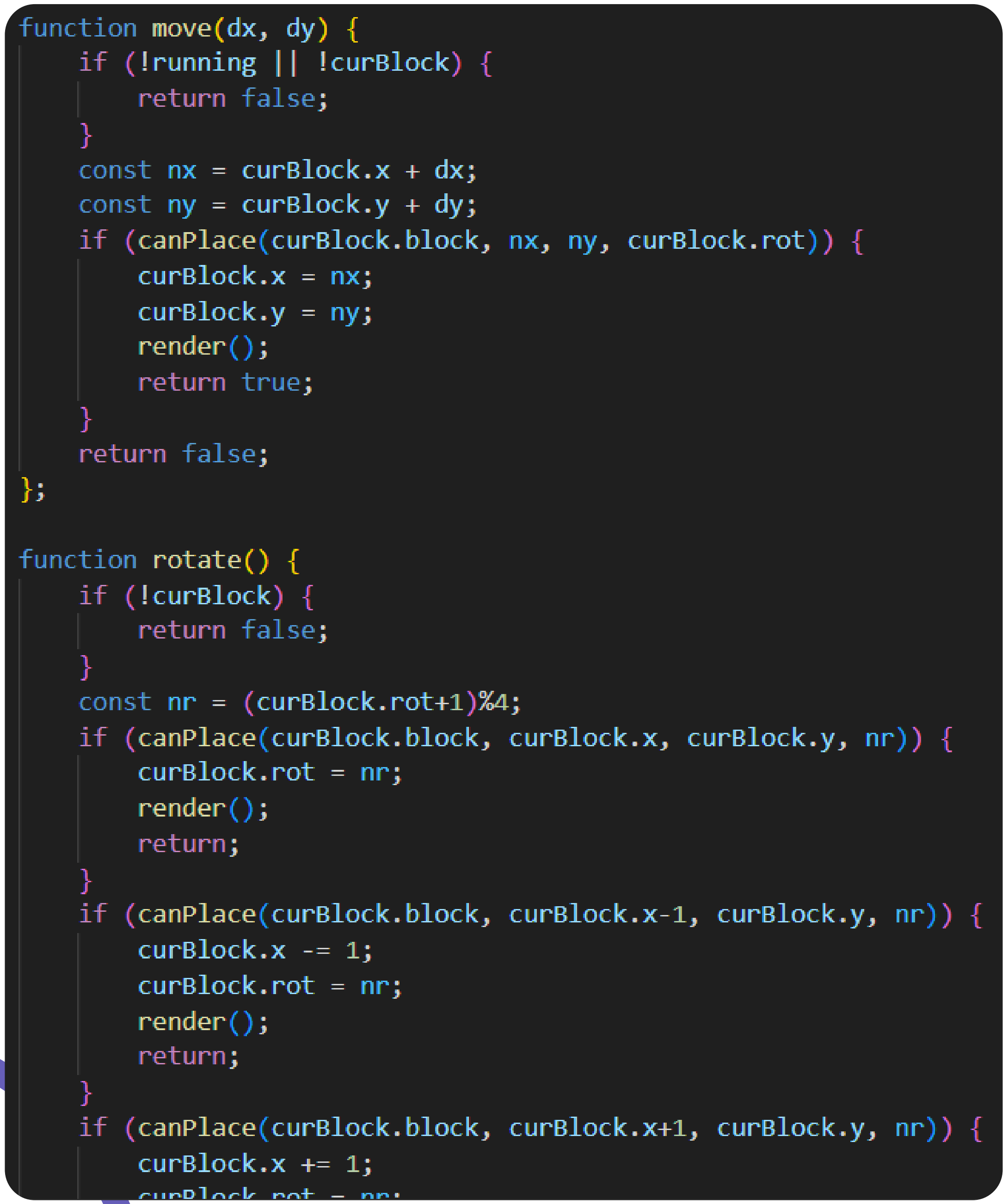
```javascript
let curBlock = null;
let nextBlock0 = null;
let nextBlock1 = null;
let nextBlock2 = null;
let holdBlock = null;

function initSpawn() {
    const block = randomBlock();
    const x = 3;
    const y = 0;
    const rot = 0;
    curBlock = {block, x, y, rot};

    const nblock0 = randomBlock();
    const nblock1 = randomBlock();
    const nblock2 = randomBlock();
    const nx = 0;
    nextBlock0 = {block: nblock0, x: nx, y, rot};
    nextBlock1 = {block: nblock1, x: nx, y, rot};
    nextBlock2 = {block: nblock2, x: nx, y, rot};
};
```

- XxxBlock : {block, x, y, rot} 필드를 가진 객체
- 필드명을 일치시켜서 다른 로직에서 활용

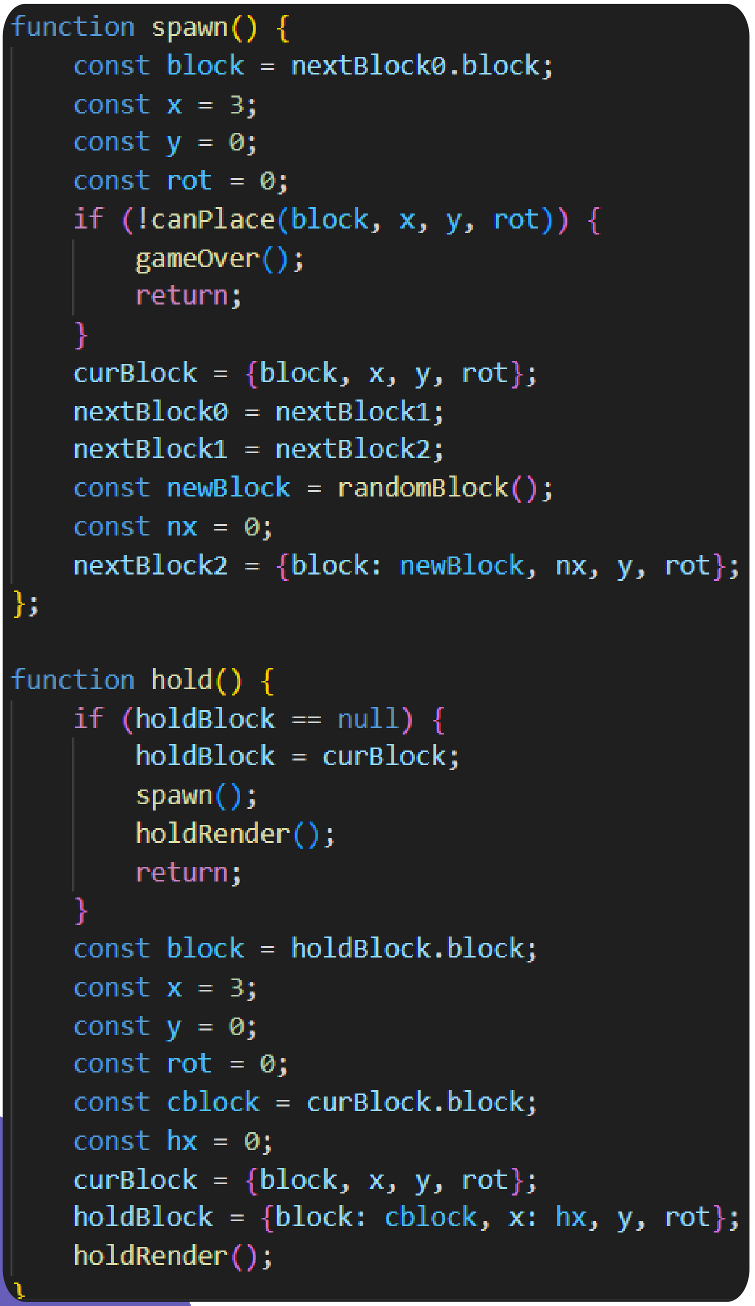


```
function inRange(x,y) {
    return x>=0 && x<w && y>=0 && y<h;
};

function canPlace(block, x, y, rot) {
    for (const [dx,dy] of block.r[rot]) {
        const px = x + dx, py = y + dy;
        if (!inRange(px,py)) {
            return false;
        }
        if (board[py][px] !== 0) {
            return false;
        }
    }
    return true;
};
```

- XxxBlock 객체이용
- block.r[rot] : 현재 블록 모양


```
function move(dx, dy) {
    if (!running || !curBlock) {
        return false;
    }
    const nx = curBlock.x + dx;
    const ny = curBlock.y + dy;
    if (canPlace(curBlock.block, nx, ny, curBlock.rot)) {
        curBlock.x = nx;
        curBlock.y = ny;
        render();
        return true;
    }
    return false;
};

function rotate() {
    if (!curBlock) {
        return false;
    }
    const nr = (curBlock.rot+1)%4;
    if (canPlace(curBlock.block, curBlock.x, curBlock.y, nr)) {
        curBlock.rot = nr;
        render();
        return;
    }
    if (canPlace(curBlock.block, curBlock.x-1, curBlock.y, nr)) {
        curBlock.x -= 1;
        curBlock.rot = nr;
        render();
        return;
    }
    if (canPlace(curBlock.block, curBlock.x+1, curBlock.y, nr)) {
        curBlock.x += 1;
```

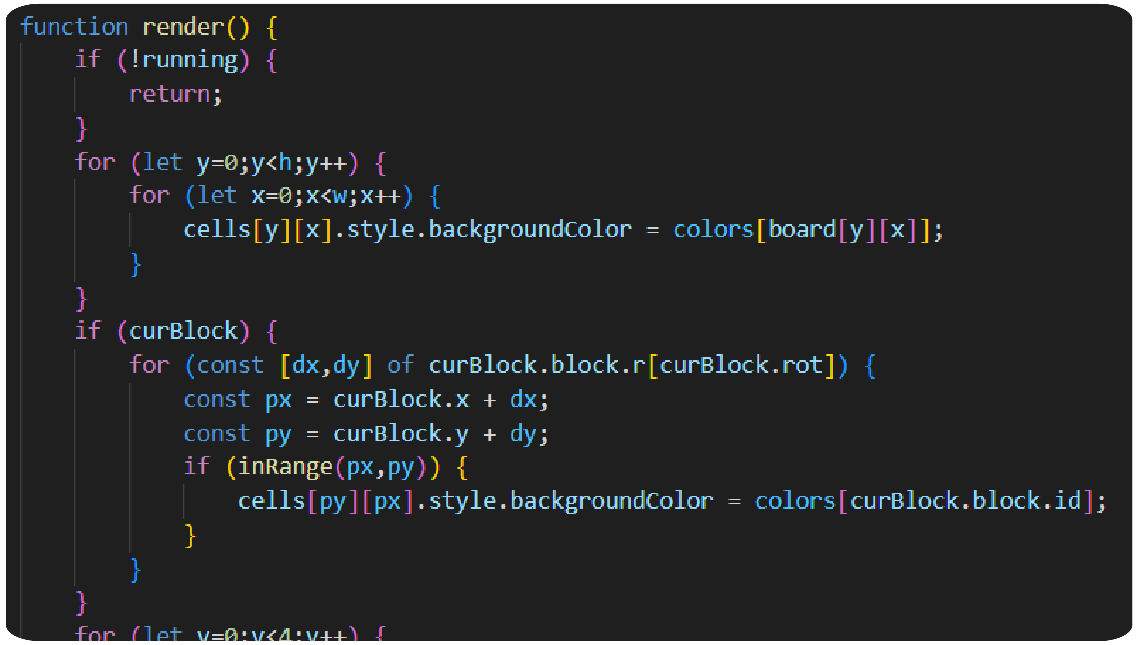
- move
- rotate



```
function spawn() {
    const block = nextBlock0.block;
    const x = 3;
    const y = 0;
    const rot = 0;
    if (!canPlace(block, x, y, rot)) {
        gameOver();
        return;
    }
    curBlock = {block, x, y, rot};
    nextBlock0 = nextBlock1;
    nextBlock1 = nextBlock2;
    const newBlock = randomBlock();
    const nx = 0;
    nextBlock2 = {block: newBlock, nx, y, rot};
};

function hold() {
    if (holdBlock == null) {
        holdBlock = curBlock;
        spawn();
        holdRender();
        return;
    }
    const block = holdBlock.block;
    const x = 3;
    const y = 0;
    const rot = 0;
    const cblock = curBlock.block;
    const hx = 0;
    curBlock = {block, x, y, rot};
    holdBlock = {block: cblock, x: hx, y, rot};
    holdRender();
}
```

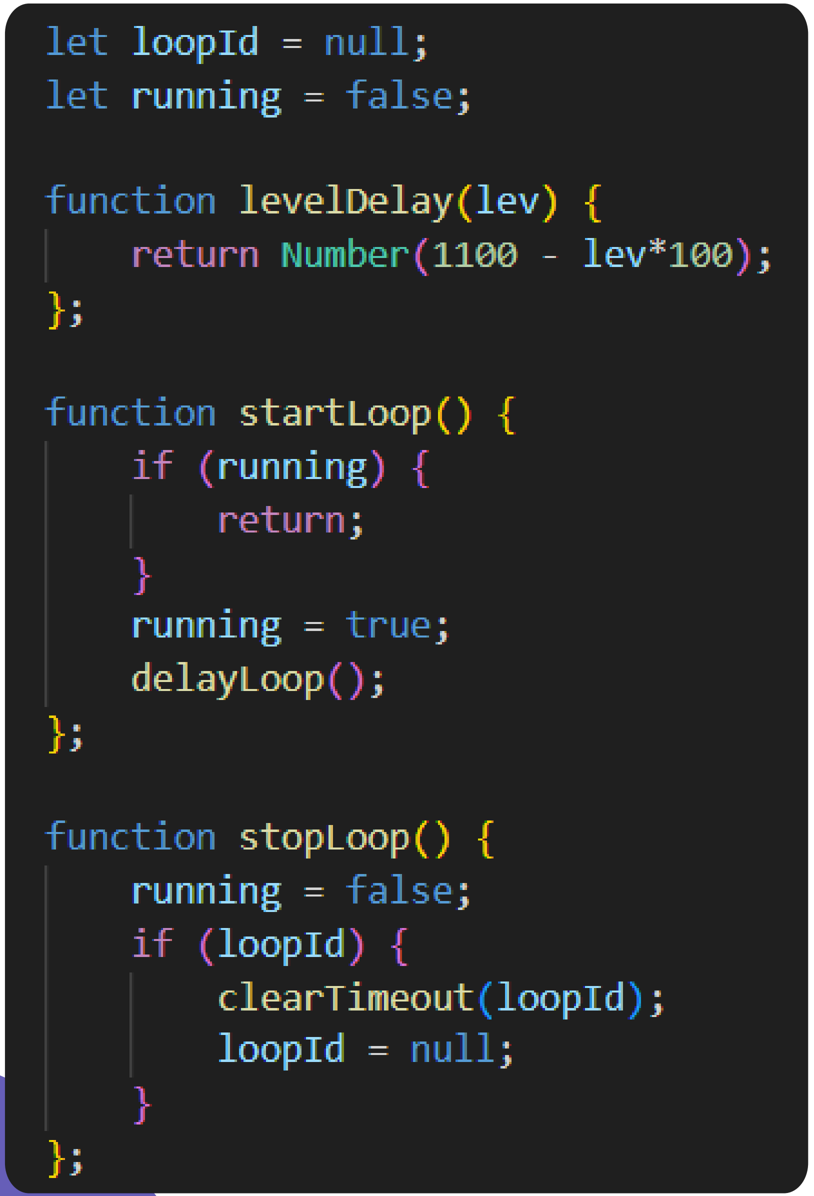
- spawn실패 시 gameover

- hold



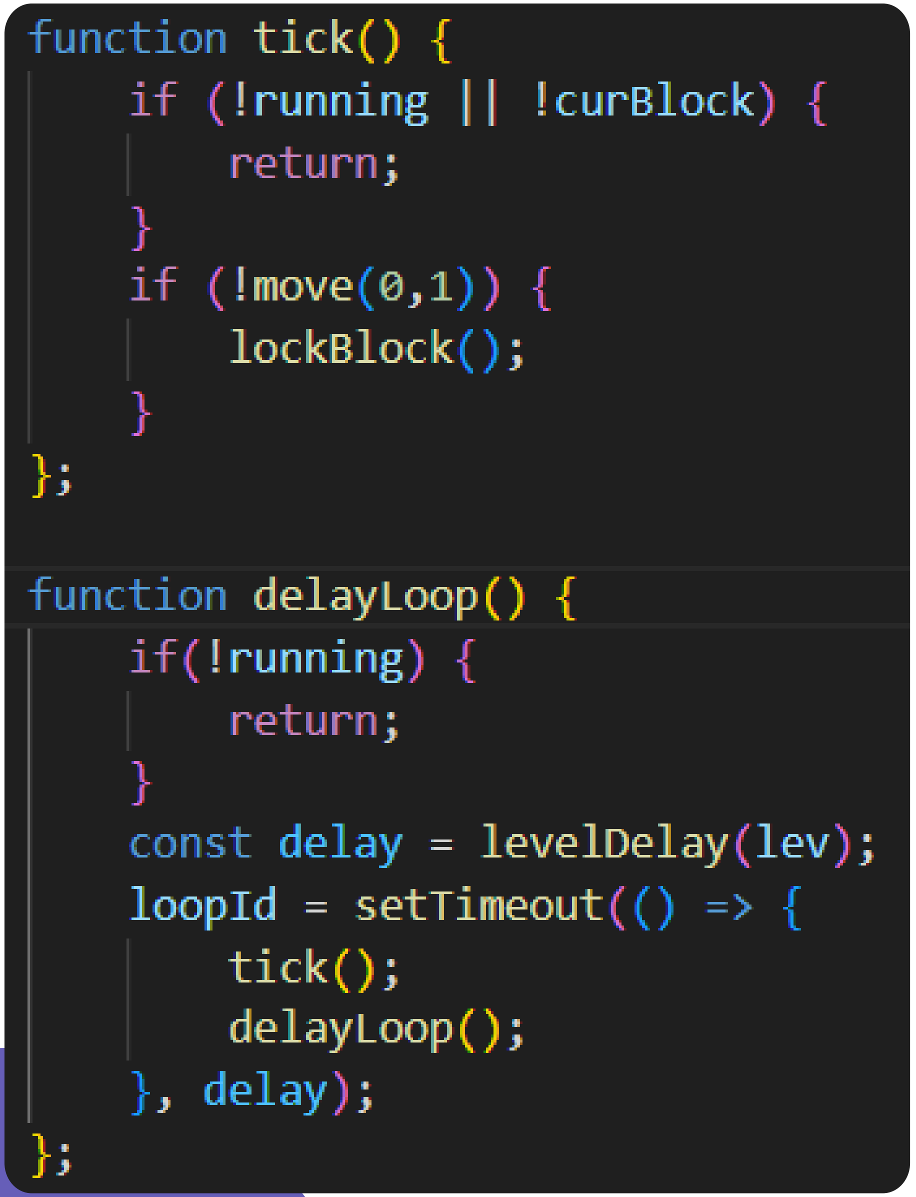
```
function render() {
    if (!running) {
        return;
    }
    for (let y=0;y<h;y++) {
        for (let x=0;x<w;x++) {
            cells[y][x].style.backgroundColor = colors[board[y][x]];
        }
    }
    if (curBlock) {
        for (const [dx,dy] of curBlock.block.r[curBlock.rot]) {
            const px = curBlock.x + dx;
            const py = curBlock.y + dy;
            if (inRange(px,py)) {
                cells[py][px].style.backgroundColor = colors[curBlock.block.id];
            }
        }
    }
    for (let y=0;y<4;y++) {
```

- board와 cells의 좌표를 이용해서 해당 좌표에 색상부여



```
let loopId = null;
let running = false;

function levelDelay(lev) {
    return Number(1100 - lev*100);
};

function startLoop() {
    if (running) {
        return;
    }
    running = true;
    delayLoop();
};

function stopLoop() {
    running = false;
    if (loopId) {
        clearTimeout(loopId);
        loopId = null;
    }
};
```

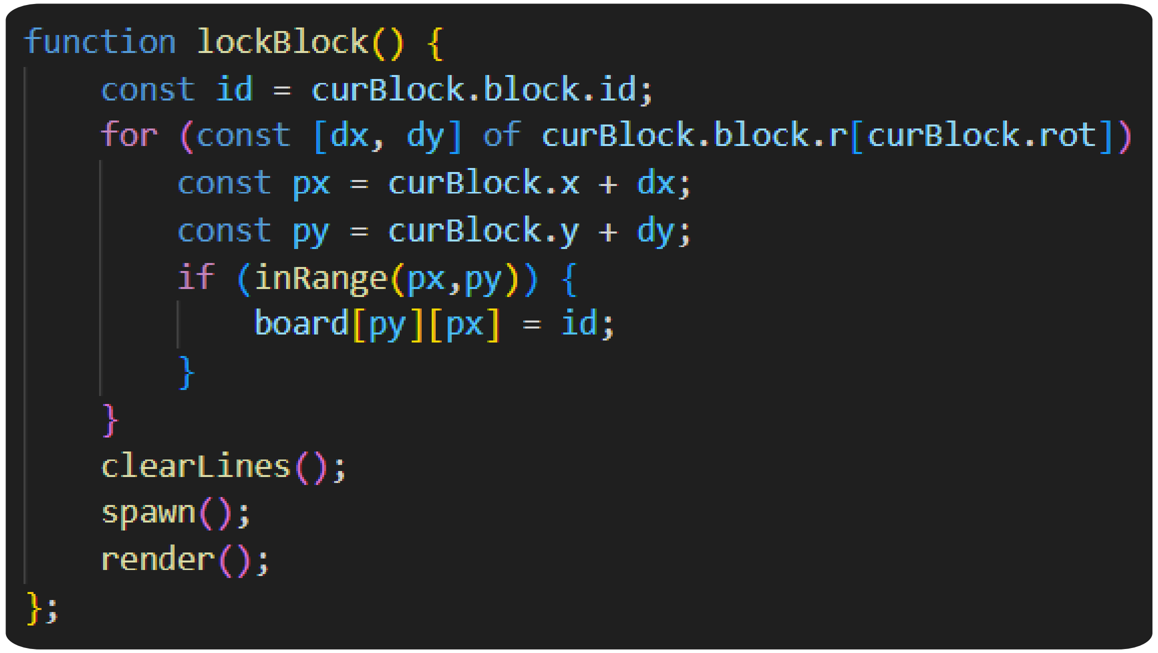
- 딜레이 설정



```
function tick() {
    if (!running || !curBlock) {
        return;
    }
    if (!move(0,1)) {
        lockBlock();
    }
};

function delayLoop() {
    if(!running) {
        return;
    }
    const delay = levelDelay(lev);
    loopId = setTimeout(() => {
        tick();
        delayLoop();
    }, delay);
};
```

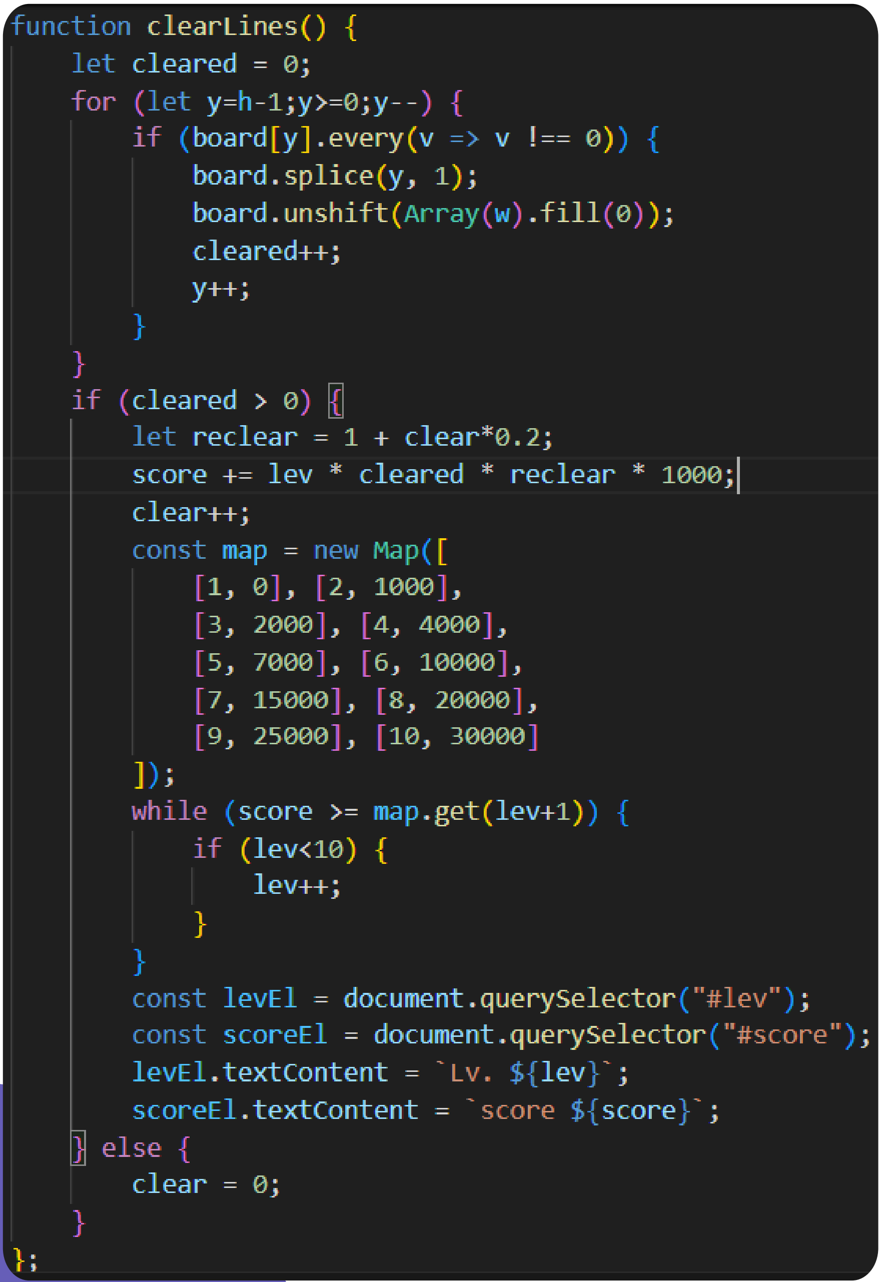
- startLoop -> delayLoop
  -> tick -> move(0,1) -> delayLoop
  -> tick -> move(0,1) -> delayLoop
  -> ...

- delayLoop 실행 시 setTimeout
  -> tick 마다 딜레이 조절



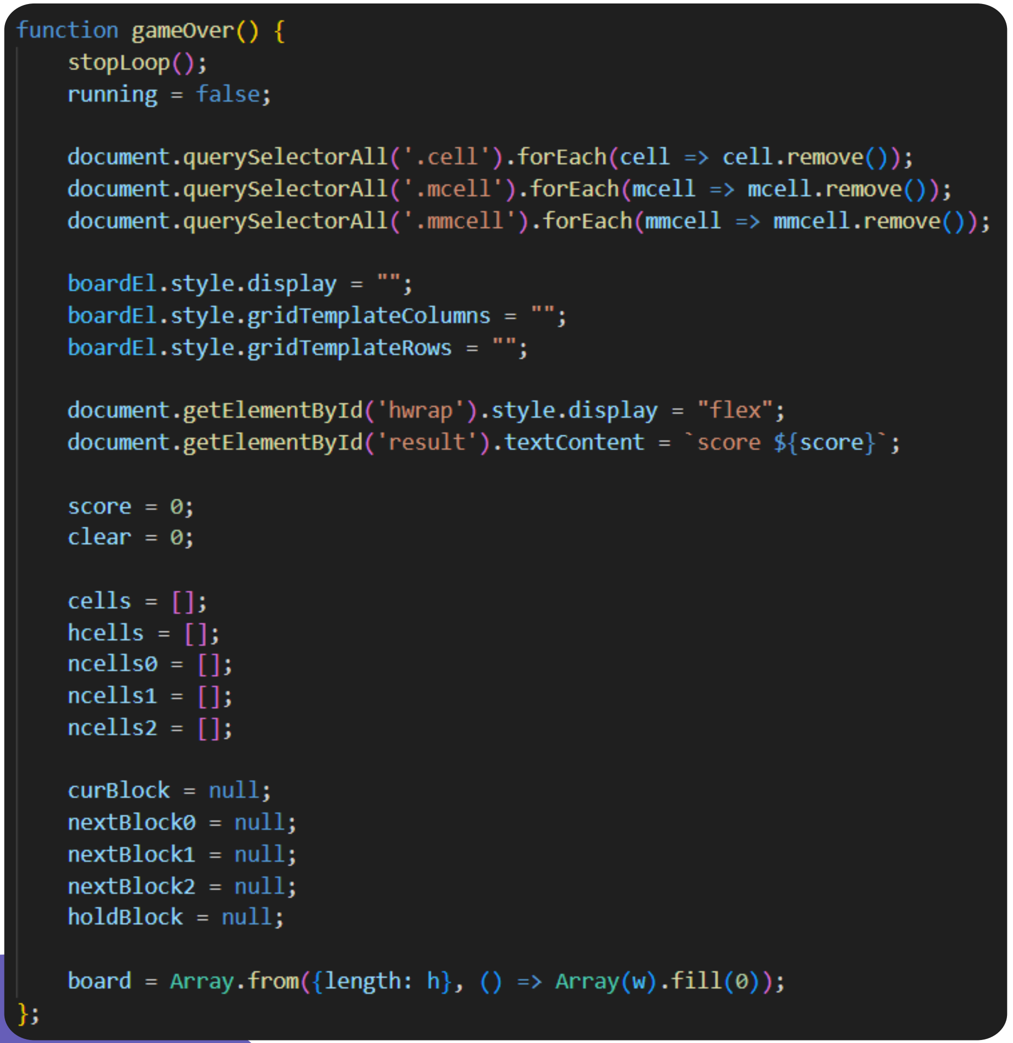
```
function lockBlock() {
    const id = curBlock.block.id;
    for (const [dx, dy] of curBlock.block.r[curBlock.rot])
        const px = curBlock.x + dx;
        const py = curBlock.y + dy;
        if (inRange(px,py)) {
            board[py][px] = id;
        }
    }
    clearLines();
    spawn();
    render();
};
```

- 블록 고정



```
function clearLines() {
    let cleared = 0;
    for (let y=h-1;y>=0;y--) {
        if (board[y].every(v => v !== 0)) {
            board.splice(y, 1);
            board.unshift(Array(w).fill(0));
            cleared++;
            y++;
        }
    }
    if (cleared > 0) {
        let reclear = 1 + clear*0.2;
        score += lev * cleared * reclear * 1000;
        clear++;
        const map = new Map([
            [1, 0], [2, 1000],
            [3, 2000], [4, 4000],
            [5, 7000], [6, 10000],
            [7, 15000], [8, 20000],
            [9, 25000], [10, 30000]
        ]);
        while (score >= map.get(lev+1)) {
            if (lev<10) {
                lev++;
            }
        }
        const levEl = document.querySelector("#lev");
        const scoreEl = document.querySelector("#score");
        levEl.textContent = `Lv. ${lev}`;
        scoreEl.textContent = `score ${score}`;
    } else {
        clear = 0;
    }
};
```

- board[y]순회하면서 줄지우기
- Board.unshift : index 0에 요소 추가
- y++
- 레벨과 레벨 목표 점수 map구조화



```
function gameOver() {
    stopLoop();
    running = false;

    document.querySelectorAll('.cell').forEach(cell => cell.remove());
    document.querySelectorAll('.mcell').forEach(mcell => mcell.remove());
    document.querySelectorAll('.mmcell').forEach(mmcell => mmcell.remove());

    boardEl.style.display = "";
    boardEl.style.gridTemplateColumns = "";
    boardEl.style.gridTemplateRows = "";

    document.getElementById('hwrap').style.display = "flex";
    document.getElementById('result').textContent = `score ${score}`;

    score = 0;
    clear = 0;

    cells = [];
    hcells = [];
    ncells0 = [];
    ncells1 = [];
    ncells2 = [];

    curBlock = null;
    nextBlock0 = null;
    nextBlock1 = null;
    nextBlock2 = null;
    holdBlock = null;

    board = Array.from({length: h}, () => Array(w).fill(0));
};
```

- gameover
- 재 시작을 위한 자원 초기화