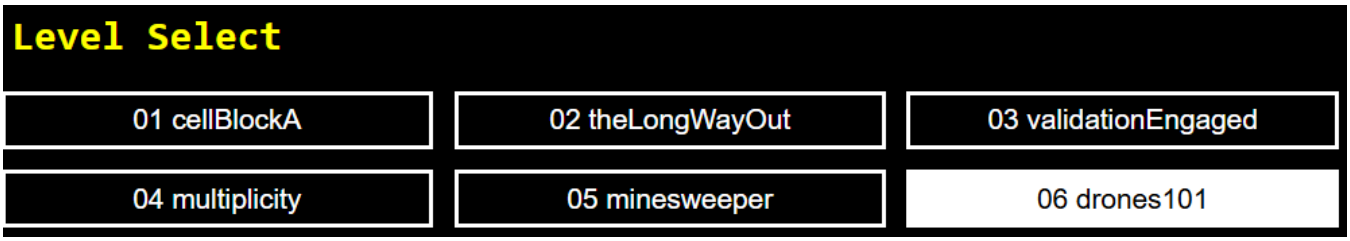
UNTRUSTED GAME



1.
```
function startLevel(map) {
    map.displayChapter('Chapter 1\nBreakout');

    map.placePlayer(7, 5);

    for (var y = 3; y <= map.getHeight() - 10; y++) {
        map.placeObject(5, y, 'block');
        map.placeObject(map.getWidth() - 5, y, 'block');
    }

for (var y = 3; y <= map.getHeight() - 10; y++) {
    if (y !== 5) {
        map.placeObject(5, y, 'block');
    }
    map.placeObject(map.getWidth() - 5, y, 'block');
}

    map.placeObject(15, 12, 'computer');

    map.placeObject(map.getWidth()-7, map.getHeight()-5, 'exit');
}

function onExit(map) {
    if (!map.getPlayer().hasItem('computer')) {
        map.writeStatus("Don't forget to pick up the computer!");
        return false;
    } else {
        return true;
    }
}
```

2.

```
function startLevel(map) {
    map.placePlayer(7, 5);

    var maze = new ROT.Map.DividedMaze(map.getWidth(), map.getHeight());
/*
    maze.create( function (x, y, mapValue) {

        // don't write maze over player
        if (map.getPlayer().atLocation(x, y)) {
            return 0;
        }

        else if (mapValue === 1) { //0 is empty space 1 is wall
            map.placeObject(x, y, 'block');
        }
        else {
            map.placeObject(x, y, 'empty');
        }
    });

    map.placeObject(map.getWidth()-4, map.getHeight()-4, 'block');
    map.placeObject(map.getWidth()-6, map.getHeight()-4, 'block');
    map.placeObject(map.getWidth()-5, map.getHeight()-5, 'block');
    map.placeObject(map.getWidth()-5, map.getHeight()-3, 'block');
*/
    map.placeObject(map.getWidth()-5, map.getHeight()-4, 'exit');
}
```

3.

```
function startLevel(map) {
    map.placePlayer(map.getWidth()-7, map.getHeight()-5);

    for (var y = 10; y <= map.getHeight() - 3; y++) {
        map.placeObject(6, y, 'block');
        map.placeObject(map.getWidth() - 5, y, 'block');
    }

    for (var x = 5; x <= map.getWidth() - 5; x++) {
        map.placeObject(x, 8, 'block');
        map.placeObject(x, map.getHeight() - 3, 'block');
    }

    map.placeObject(7, 5, 'exit');
}

function validateLevel(map) {
    var numBlocks = 2 * (map.getHeight()-13) + 2 * (map.getWidth()-10);

    map.validateAtLeastXObjects(numBlocks, 'block');
    map.validateExactlyXManyObjects(1, 'exit');
}
```

4.

```
function startLevel(map) {

    map.placePlayer(map.getWidth()-5, map.getHeight()-4);

    for (var y = 7; y <= map.getHeight() - 3; y++) {
        map.placeObject(7, y, 'block');
        map.placeObject(map.getWidth() - 3, y, 'block');
    }
    map.placeObject(map.getWidth() - 5, 10, 'exit');
    for (var x = 7; x <= map.getWidth() - 3; x++) {
        map.placeObject(x, 7, 'block');
        map.placeObject(x, map.getHeight() - 3, 'block');
    }

    map.placeObject(map.getWidth() - 5, 5, 'exit');
}
```

5.

```
function getRandomInt(min, max) {
    return Math.floor(Math.random() * (max - min + 1)) + min;
}

function startLevel(map) {
    for (var x = 0; x < map.getWidth(); x++) {
        for (var y = 0; y < map.getHeight(); y++) {
            map.setSquareColor(x, y, '#f00');
        }
    }

    map.placePlayer(map.getWidth() - 5, 5);

    for (var i = 0; i < 75; i++) {
        var x = getRandomInt(0, map.getWidth() - 1);
        var y = getRandomInt(0, map.getHeight() - 1);
        if ((x != 2 || y != map.getHeight() - 1)
            && (x != map.getWidth() - 5 || y != 5)) {
            // don't place mine over exit or player!
            map.placeObject(x, y, 'mine');
            map.setSquareColor(x,y, '#f111111');
        }
    }

    map.placeObject(2, map.getHeight() - 1, 'exit');
}

function validateLevel(map) {
    map.validateAtLeastXObjects(40, 'mine');
    map.validateExactlyXManyObjects(1, 'exit');
}
```

6.

```
    map.placePlayer(1, 1);
    map.placeObject(map.getWidth()-2, 12, 'attackDrone');
    map.placeObject(map.getWidth()-1, 12, 'exit');

    map.placeObject(map.getWidth()-1, 11, 'block');
    map.placeObject(map.getWidth()-2, 11, 'block');
    map.placeObject(map.getWidth()-1, 13, 'block');
    map.placeObject(map.getWidth()-2, 13, 'block');
    map.defineObject('defenseDrone', {
            'type': 'dynamic',
            'symbol': 'D',
            'color': 'blue',
            'behavior': function (me) {
                moveToward(me, 'attackDrone');
            }
        });
    map.placeObject(2, 2, 'defenseDrone');
}

function validateLevel(map) {
    map.validateExactlyXManyObjects(1, 'exit');
}
```