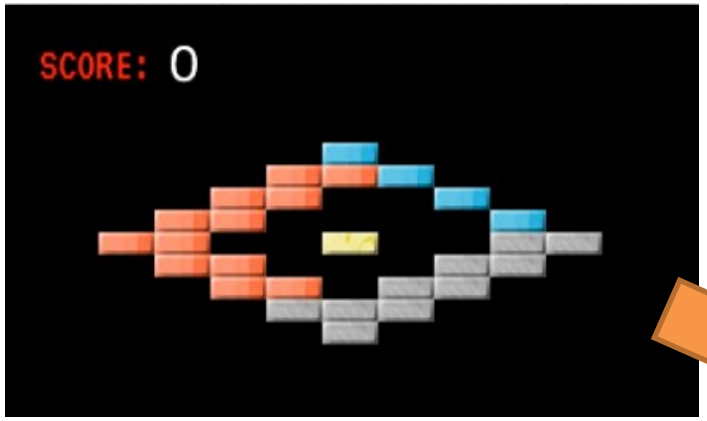
ブロック崩しのカスタマイズ例

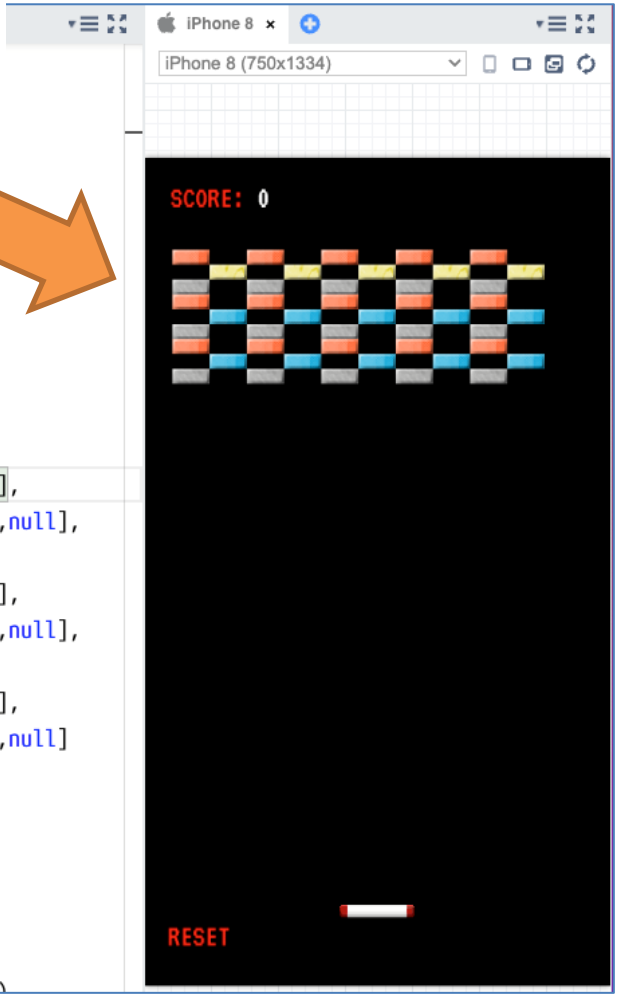
④ブロックの初期配置を変える

■ ブロックの初期配置を変える



index.html main.js ×

```
34          paddle: null,
35          balls: [],
36          blocks: [],
37          score: 0,
38          scoreLabel: r
39          accelLabel: r
40          isMouseDown:
41
42          // Create blocks map
43          setMap: function () {
44              var blockMap = [
45                  ['red',null,'red',null,'red',null,'red',null,'red',null],
46                  [null, 'gold',null, 'gold',null, 'gold',null, 'gold',null, 'gold'],
47                  ['silver',null,'silver',null,'silver',null,'silver',null,'silver',null],
48                  ['red',null,'red',null,'red',null,'red',null,'red',null],
49                  [null, 'blue',null, 'blue',null, 'blue',null, 'blue',null, 'blue'],
50                  ['silver',null,'silver',null,'silver',null,'silver',null,'silver',null],
51                  ['red',null,'red',null,'red',null,'red',null,'red',null],
52                  [null, 'blue',null, 'blue',null, 'blue',null, 'blue',null, 'blue'],
53                  ['silver',null,'silver',null,'silver',null,'silver',null,'silver',null]
54              ];
55
56              var position = parseInt(BB.renderer.width * 0.5) - (5 * 30)
57              for (j = 0; j < blockMap.length; j++) {
58                  for (i = 0; i < blockMap[j].length; i++) {
59                      if (blockMap[j][i] !== null) {
60                          var block = BB.addBlock(position + (30 * i), 80 + (12 * j)
```



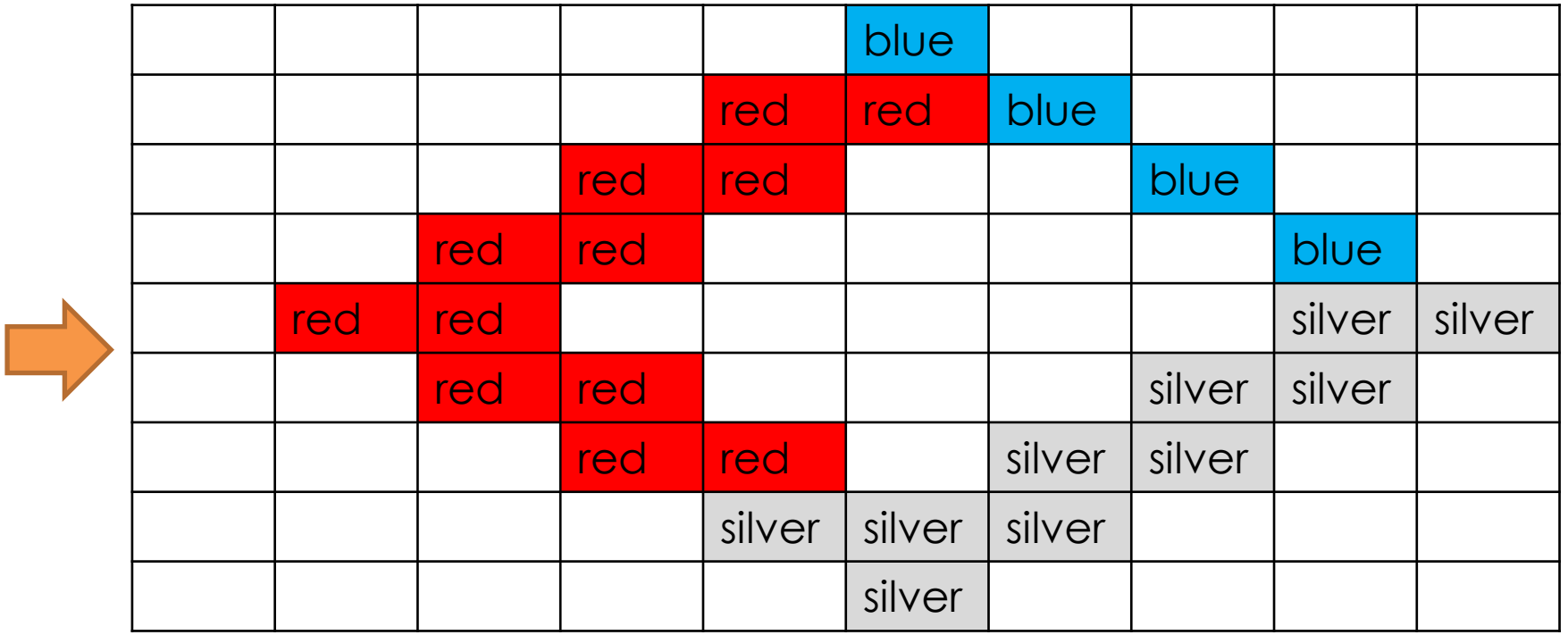
■ 修正箇所

- main.jsの44行目～ blockMap

```
42      // Create blocks map
43      setMap: function () {
44          var blockMap = [
45              ['red',null,'red',null,'red',null,'red',null,'red',null],
46              [null, 'gold',null, 'gold',null, 'gold',null, 'gold',null, 'gold'],
47              ['silver',null,'silver',null,'silver',null,'silver',null,'silver',null],
48              ['red',null,'red',null,'red',null,'red',null,'red',null],
49              [null, 'blue',null, 'blue',null, 'blue',null, 'blue',null, 'blue'],
50              ['silver',null,'silver',null,'silver',null,'silver',null,'silver',null],
51              ['red',null,'red',null,'red',null,'red',null,'red',null],
52              [null, 'blue',null, 'blue',null, 'blue',null, 'blue',null, 'blue'],
53              ['silver',null,'silver',null,'silver',null,'silver',null,'silver',null]
54          ];
```

- 1行に注目すると、[ ] の中にカンマ（,）で区切った値が10個ある
- その行が、9行ある（行と行の間も、カンマ（,）で区切っている）
- それをさらに[ ]で囲んでいる
- 1行が画面の横1行を表す。9行あるので、ブロックは9段になる

■ 概念図



[null, 'red', 'red', null, null, null, null, null, 'silver', 'silver'],

- ブロックを置かない場所は、null と書く
- 色の名前は’’（引用符）で囲む

■ 使用できる色

- 初期のプログラムでは、赤（red）、青（blue）、銀（silver）、金（gold）の４色を指定できる

- ４色だけになるのは、次の理由による
  - imgフォルダにある、ブロックを表す画像ファイルは５色（※main.jsの19行目から27行目も確認すること）
  - main.jsの72行目からはじまるaddBlock()関数のswitch文で制御している

- addBlock関数の中では、ブロックの色ごとの、得点も設定している

■ 補足事項・学習のヒント：二次元配列

- [値1，値2，…，値n]で、配列を作ることができる

値1	値2	値3	…	値n

- 配列[ ]を、 別の配列の要素にすることができる
- [

  [値11，値12，…，値1n]，

  [値21，値22，…，値2n]，

  [値31，値32，…，値3n]

  ]

値11	値12	値13	…	値1n
値21	値22	値23	…	値2n
値31	値32	値33	…	値3n

- **「ブロックの配置」のように、行・列の形式のデータを表現するのに便利**
- 二次元配列の全てのデータを操作するとき、二重ループが使われる