JavaScript+ HTML5で ブラウザゲームに挑戦

河西朝雄著

KASAI.SOFTWARELAB

定価1,500円＋税

JavaScript+HTML5 で

ブラウザゲームに挑戦

河西　朝雄著

はじめに

「全てのゲームは Web でやれ！」――そんなキャッチコピーを掲げるブラウザゲームの新プラットフォーム「Yahoo!ゲーム　ゲームプラス」を、ヤフーが 2017 年 7 月 18 日に公開しました。ブラウザゲームはパソコン、スマートフォン、タブレットなどの機種に依存しないで動作します。

本書は JavaScript+HTML5 を使ってブラウザゲームを作る基礎を学びます。ゲームは次の 10 本を用意し、Step を踏んで、完成させます。完成版のプログラムは多少長くなりますが、Step ごとに追加していくので、スムーズに学習できます。プログラムリストにおいて、Step ごとに追加された部分は赤色になっています。

「文法のまとめ」で JavaScript の文法のまとめを必要により解説しています。

・もぐらたたき
・移動板パズル
・ラケットゲーム
・シューティング
・戦略を持つじゃんけん
・相性占い
・ハノイの塔
・迷路
・マインスイーパー
・リバーシー

プログラム開発はパソコンで行い、スマートフォン（Android、iPhone）でも動作確認しました。本書の実行結果画面はスマートフォンのものです。

2018 年 1 月 20 日

河西　朝雄

目次

第１回　もぐらたたき

４×３のマスに配置した巣穴からもぐらを出してたたきます（クリックまたはタップ）。消えないうちにたたいたら得点を加算します。

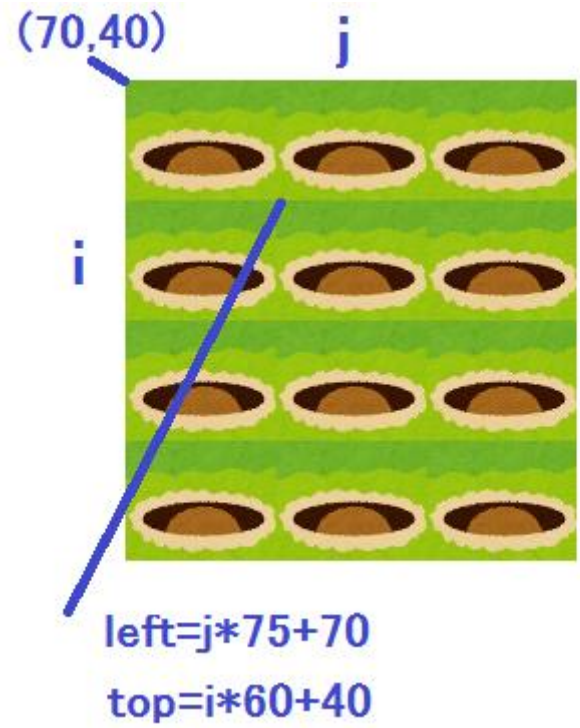
Step1　巣穴を作る
Step2　もぐらを出す
Step3　もぐらをたたく
Step4　もぐらを２匹出す

Step1　巣穴を作る

・４×３のマス目にもぐらの巣穴を配置します。巣穴のイメージは mogu0.png です。

・横幅 75 ピクセル、縦幅 60 ピクセルのサイズですので、i 行 j 列の位置のイメージ位置は以下の式で得られます。

```
img.style.left=j*75+70+"px";
img.style.top=i*60+40+"px";
```

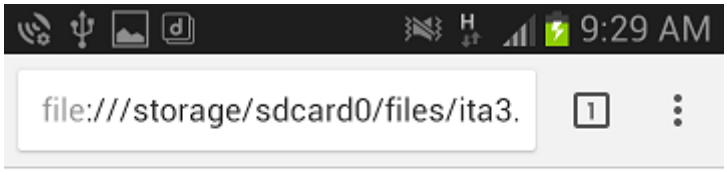


・mogu1.html

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0,
  maximum-scale=1.0, user-scalable=no" />
</head>
<body>
<script type="text/javascript">

// もぐらたたき（巣穴を作る）

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

```
            for (var j=0;j<3;j++){
                var img=new Image();
                img.src="mogu0.png";
                img.id="img"+(i*3+j+1);
                img.style.position="absolute";
                img.style.left=j*75+70+"px";
                img.style.top=i*60+40+"px";
                document.body.appendChild(img);
            }
        }
</script>
</body>
</html>
```

第２回　移動板パズル



1~8 の板を 3×3 のマスにバラバラに配置します。右下のマスを初期状態で空きとします。クリックした板の上下左右に空きがあれば、クリックした板を空き位置に移動します。これをくり返して 1~8 の順番に板を並べ替えます。

Step1　１~８の板を並べる
Step2　クリック（タッチ）した板の上下左右に空きがあれば移動
Step3　ドラッグドロップで板を移動

Step1　１～８の板を並べる

■ランダムな順列

1~8 のランダムな順列とは「2,5,3,1,7,4,8,6」のようなものです。ランダムな順列は以下の手順で作ります。

・配列 num[0]~num[7]に 1~8 の数字をこの順序で格納します

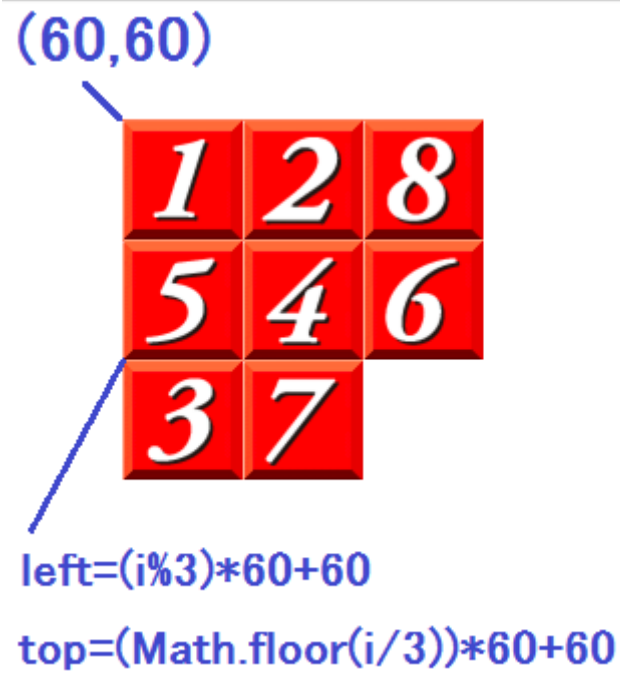
・i を 7~1 まで繰り返す間に j に 0~7 の乱数を求め、num[i]と num[j]の内容を交換します

```
for (var i=7;i>0;i--){
    j=rnd(0,7);
    t=num[i];num[i]=num[j];num[j]=t;
}
```

■板の配置位置

i 番目の板を 3 行 3 列のマスに配置する場合の位置は以下のように計算します。

```
img.style.left=(i%3)*60+60+"px";
img.style.top=(Math.floor(i/3))*60+60+"px";
```

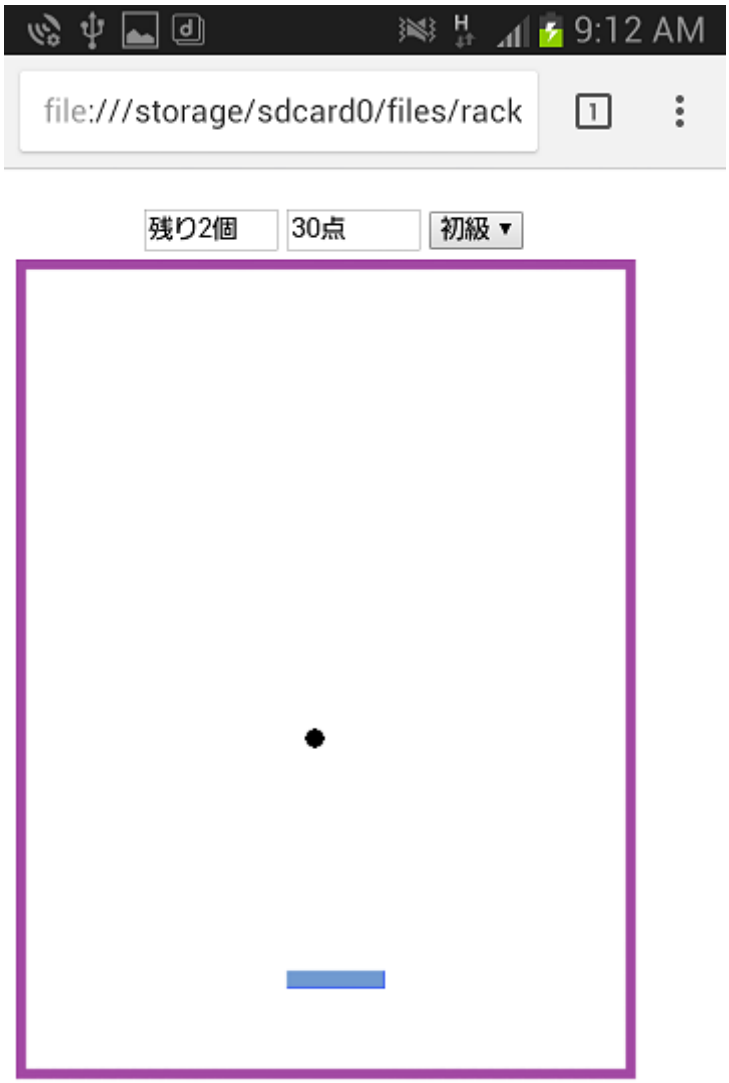


・ita1.html

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0,
  maximum-scale=1.0, user-scalable=no" />
<script type="text/javascript">

// 移動板パズル（１～８の板を並べる）
```

```
	var num=[1,2,3,4,5,6,7,8];
	function rnd(m,n)    // m～n の整数乱数
	{
		return Math.floor(Math.random()*(n-m+1))+m;
	}
</script>
</head>
<body>
<script type="text/javascript">
	var j,t;
	for (var i=7;i>0;i--){    // ランダムな順列
		j=rnd(0,7);
		t=num[i];num[i]=num[j];num[j]=t;
	}
	for (var i=0;i<8;i++){    // 板の配置
		var img=new Image();
		img.src="n"+num[i]+".gif";
		img.id="n"+num[i];
		img.style.position="absolute";
		img.style.left=(i%3)*60+60+"px";
		img.style.top=(Math.floor(i/3))*60+60+"px";
		document.body.appendChild(img);
	}
</script>
</body>
</html>
```

第３回　ラケットゲーム



ボールを斜め 45 度の方向で移動し、壁に当たったら跳ね返します。ラケットを指の動き（マウス）に追従して移動し、ボールを打ち返します。ラケットで当てるたびに１０点得点を加算します。ボールの数を３個とします。

Step1　ボールを飛び回らせる
Step2　ラケットの移動を追加
Step3　ラケットで跳ね返す
Step4　ボールを３回まで使用し、得点を加算
Step5　ボールの速さのレベルを設定

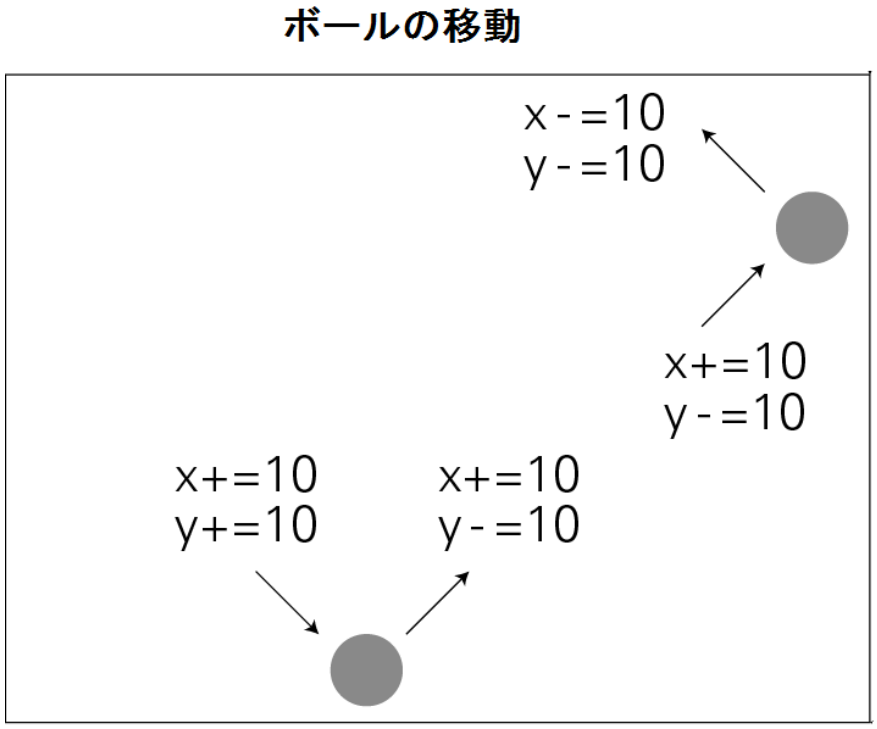
Step1　ボールを飛び回らす

・ボールを斜め 45 度の方向で移動し、壁に当たったら跳ね返します

■ボールの移動

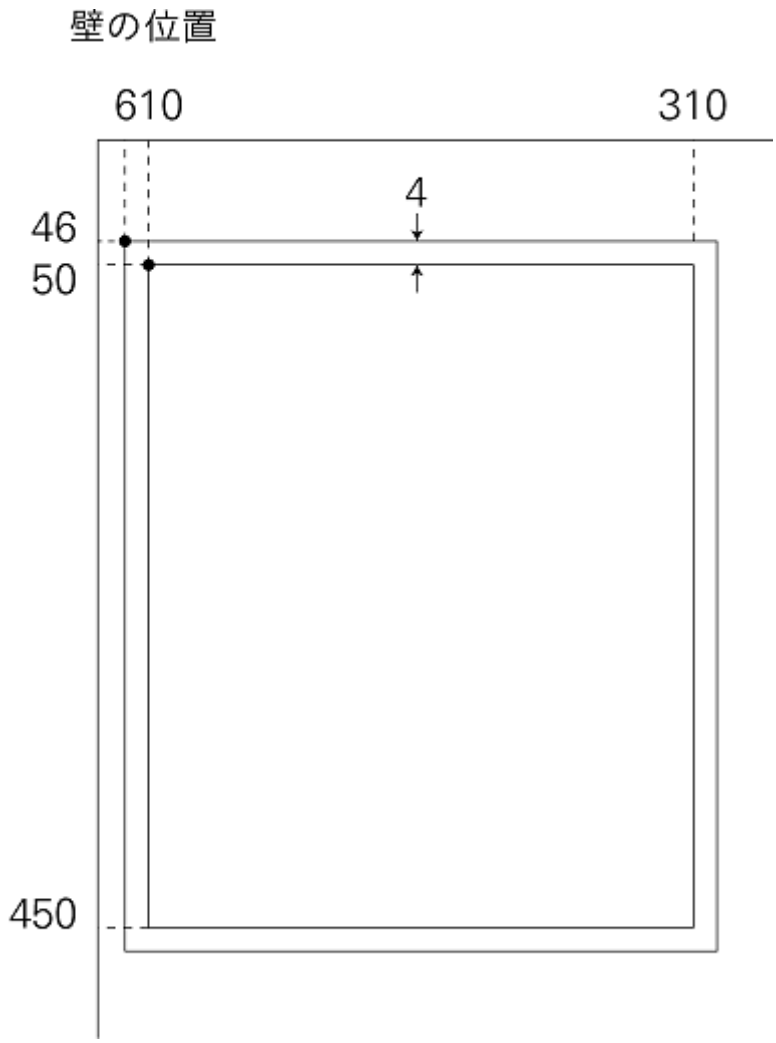
ボールを移動させる関数 move を作ります。ボールのイメージファイルは ball.png、サイズは 10×10 ピクセル、一回にボールが進む距離は 10 ピクセルとします。

ボールを左右に移動するには x+=dx、ボールを上下に移動するには y+=dy とします。斜めに移動するには、x+=dx と y+=dy の両方を行います。イメージの左上隅の x,y 座標は ball.style.left, ball.style.top で取得、設定できます。



■壁の設計

壁で囲まれた固定枠の中をボールが移動するようにします。壁のイメージを wall.png とします。壁の厚さを 4 ピクセル、内径を 300×400 ピクセルとします。内径はボールのサイズと移動距離が 10 なので、10 の倍数となる値に設定しています。壁の左上隅の内径がクライアント画面の（10,50）位置にくるように、壁イメージの左隅上は（6,46）位置にします。従って壁の内径位置は左が 10、上が 50、右が 310、下が 450 となります。



■壁での反射

ボールを左右壁で跳ね返すには以下のようにします。右壁の判定で「x+10」としているのはボールの幅を考慮しているためです。

```
if (x<=10 || x+10>=310)  // 左右壁での反射
    dx=-dx;
```

ボールを上下壁で跳ね返すには以下のようにします。下壁の判定で「y+10」としているのはボールの幅を考慮しているためです。

```
if (y<=50 || y+10>=450)  // 上下壁での反射
    dy=-dy;
```

・racket1.html

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0,
maximum-scale=1.0, user-scalable=no" />
<script type="text/javascript">

// ラケットゲーム（ボールを飛び回らす）

	var x=200,y=100; //  ボールの初期位置
	var dx=10,dy=10; //  ボールの移動距離
	function move()
	{
		var ball=document.getElementById("ball");
		x+=dx;
		y+=dy;
		ball.style.left=x+"px";
		ball.style.top=y+"px";
		if (x<=10 || x+10>=310)   // 左右壁での反射
			dx=-dx;
		if (y<=50 || y+10>=450)   // 上下壁での反射
			dy=-dy;
	}
</script>
</head>
<body onLoad="setInterval(move,100)">
<img src="wall.png" style="position:absolute;left:6px;top:46px" />
<img id="ball" src="ball.png" style="position:absolute;left:200px;top:100px" />
</body>
</html>
```

文法のまとめ

■複合代入演算子

「○＝○＋□」のように左辺と右辺に同じ変数○がある場合は変数の内容に□を足した値が新しい変数の値になります。このように変数の内容に値を足すなどして新しい値にすることを、変数の更新といいます。

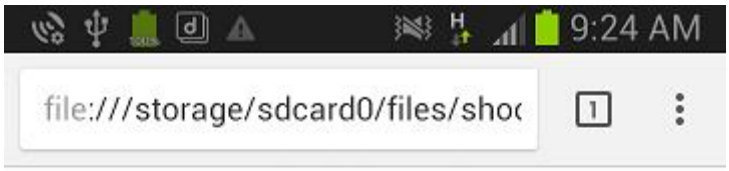
```
sum=sum+10;
```

は

```
sum+=10;
```

と書くことができ、+=を複合代入演算子といいます。+=の他に-=、*=、/=などが使えます。

第４回　シューティング



画面右端から左端に移動するモンスターを画面下部に配置した砲台から玉を打って、当てます。

Step1　砲台から玉を打つ
Step2　モンスターを移動
Step3　玉がモンスターに当たったか判定
Step4　画面のサイズで砲台の位置を設定
Step5　モンスターを４つ打ち損ねたら GameOver で得点を表示
Step6　乱数で左右どちらかから出るか決める

Step1　砲台から玉を打つ

・砲台のイメージは hodai.png、玉のイメージは bakudan.png です。

・玉を打っている最中に別の玉を打てないように変数 flag で管理します。flag が true なら玉を打てる状態です。

・shooting1.html

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0,
 maximum-scale=1.0, user-scalable=no" />
<script type="text/javascript">

// シューティング（砲台から玉を打つ）

    var bakudan;      //　玉
    var y=400;        //　玉の位置
    var TimeId;
    var flag=true;  //　玉が飛んでいない状態

    function move1()　// 玉の移動
    {
        bakudan.style.top=y+"px";
        if (y<-40){                          // 画面上部に達したら
            clearInterval(TimeId);
            flag=true;
            bakudan.style.visibility="hidden";
        }
        y-=40;
    }
    function shoot()
    {
        if (flag) {　//　玉が飛んでいない状態なら
            bakudan.style.visibility="visible";
            y=400;
            flag=false;
            TimeId=setInterval(move1,100);
        }
    }
</script>
</head>
<body>
<img id="hodai" src="hodai.png"
```

```
  style="position:absolute;left:143px;top:430px" onClick="shoot()" />
<img id="bakudan" src="bakudan.png"
  style="position:absolute;left:150px;top:400px;visibility:hidden" />
<script type="text/javascript">
      bakudan=document.getElementById("bakudan");
</script>
</body>
</html>
```

第５回　戦略を持つじゃんけん

クリックした手が「グー」、「チョキ」、「パー」のどれかを判定し、コンピュータも「グー」、「チョキ」、「パー」の手を出し、勝敗を判定します。対戦をするたびに相手のくせを読み、強くなるようにコンピュータに戦略を持たせます。

Step1　戦略なし
Step2　コンピュータ必勝（後だしじゃんけん）
Step3　コンピュータに戦略を持たせる（戦略１）
Step4　コンピュータに戦略を持たせる（戦略２）

Step1　戦略なし

・コンピュータの手は乱数で決めます。つまり戦略なしです。

・じゃんけんの手のイメージは、白い手と色付きの手があります。色付きの手はその手がクリックされたときに 0.5 秒表示されます。


■勝敗の判定

・グー、チョキ、パーをそれぞれ 0、1、2 で表し、コンピュータ（computer）とあなた（man）の手の対戦表をつくると次のようになります

・〇は man の勝ち、×は man　の負けです

man computer	グー 0	チョキ 1	パー 2
グー 0	―	×	◯
チョキ 1	◯	―	×
パー 2	×	◯	―

・computer と man の勝敗を判定するには次の式を使うとうまくいきます

```
jg=(computer-man+3) % 3;
```

・jg の値により次のように判定できます

0・・・引き分け

1・・・あなたの勝ち

2・・・あなたの負け

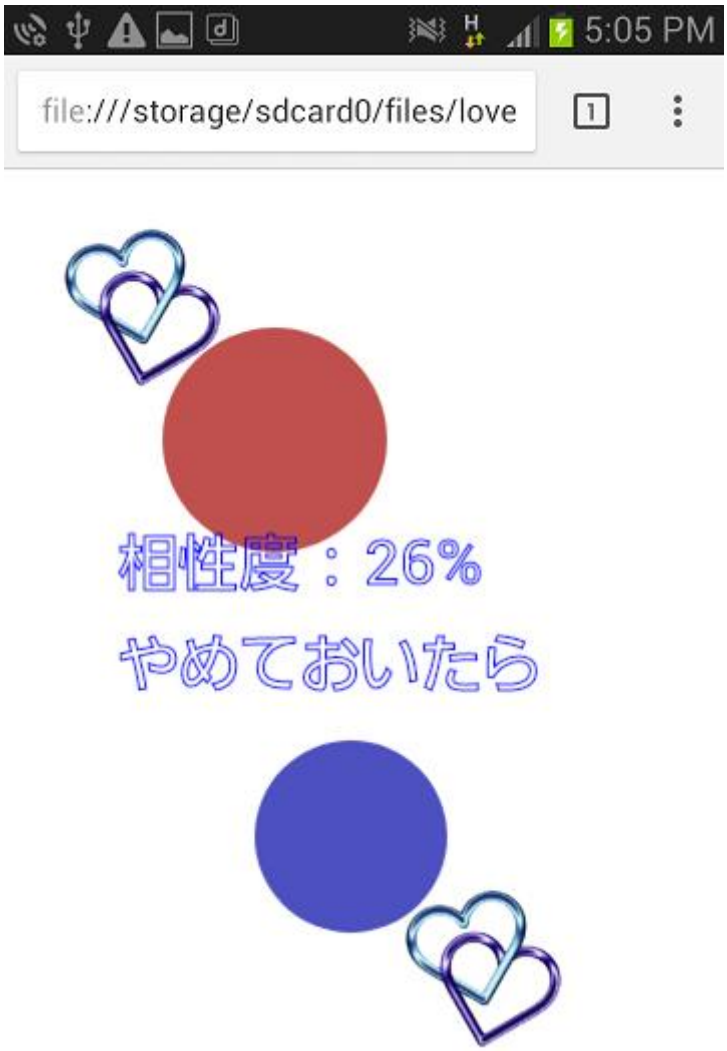
・janken1.html

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0,
```

```
  maximum-scale=1.0, user-scalable=no" />
<script type="text/javascript">

// 戦略を持つじゃんけん（戦略なし）

      var hand=["グー","チョキ","パー"];
      var msg=["ひきわけ","あなたの勝ち","コンピュータの勝ち"];
      var hist=[0,0,0]; // 勝敗の度数
      var img1=["gu1.png","tyoki1.png","pa1.png"];   // 白い手
      var img2=["gu2.png","tyoki2.png","pa2.png"];   // 色付きの手
      var obj;
      var context;

      function rnd(m,n)   // m～n の整数乱数
      {
            return Math.floor(Math.random()*(n-m+1))+m;
      }
      function handClick(n)   // 手がクリックされたときの処理
      {
            obj=document.getElementById("hand"+n);
            obj.src=img2[n];   // 色付きの手
            context.clearRect(0,0,640,640);
            setTimeout("check("+n+")",500);
      }
      function check(man)   // 勝敗の判定
      {
            var computer,jg;
            computer=rnd(0,2);
            jg=(computer-man+3) % 3;
            hist[jg]++;
            context.strokeStyle = "blue";      // グラフィック表示
            context.font = "24px 'ＭＳ　Ｐゴシック'";
            context.strokeText("あなたの手："+hand[man], 0, 50);
            context.strokeText("コンピュータの手："+hand[computer], 0, 80);
            context.strokeText("判定："+msg[jg], 0, 110);
            context.strokeText("あなたの通算勝敗：  "+hist[1]+" 勝 "+hist[2]+" 敗
"+hist[0]+"分", 0, 140);
            obj.src=img1[man];   // 白い手
      }
</script>
</head>
<body>
<img id="hand0" src="gu1.png" onClick="handClick(0)" />
<img id="hand1" src="tyoki1.png" onClick="handClick(1)" />
<img id="hand2" src="pa1.png" onClick="handClick(2)" /><br />
<canvas id="canvas" width="640" height="640"></canvas>
<script type="text/javascript">
      var canvas = document.getElementById("canvas");
```

```
        if(canvas.getContext){
            context = canvas.getContext("2d");
        }
</script>
</body>
</html>
```

第６回　相性占い



画面上の 2 つの heart.png イメージをタッチすると青い円と赤い円とが徐々に大きくなります。あるタイミングで指を離す（一人が離した時点）と二人の相性度を判定して表示します。

Step1　結果を alert で表示
Step2　結果をグラフィックで表示

Step1　結果を alert で表示

・タッチするハートリングのイメージは heart.png です。

・一方の円の半径を r1、他方の円の半径を r2 とします。setTimeout を使ってで 1000 ミリ秒ごとに半径を増やします。

・flag はゲームの一番最初の開始のとき（まだタッチされていない時）にいきなり相性度が表示されるのを防ぐものです。

・相性度は２つの半径 r1,r2 を元に「(r1+r2)%101」で計算し、結果を alert で表示します。

「注」iPhone、iPad ではタッチした 2 本の指を同時に離したときに clearTimeout や画面クリアが実行されない場合があります。

・love1.html

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0,
maximum-scale=1.0, user-scalable=no" />
<script type="text/javascript">

// 相性度（alert で表示）

	var context;
	var flag=false;
	var r1,r2;
	var timerId;

	document.addEventListener("touchstart", function(event){  // タッチ開始
		event.preventDefault();
		var id1=event.touches[0].target.id;
		var id2=event.touches[1].target.id;
		r1=r2=40;
		if (event.touches.length==2 && (id1=="img1" || id2=="img1") &&
(id1=="img2" || id2=="img2")){
			flag=true;
			disp();
		}
	});
	document.addEventListener("touchend", function(event){  // タッチ終了
		if (flag){
			clearTimeout(timerId);
			alert("相性度："+(r1+r2)%101+"%");  // 相性度
			context.clearRect(0,0,640,640);
			flag=false;
		}
	});
	function disp()
	{
		timerId=setTimeout("disp()",1000);
		if (flag){
			context.beginPath();
			context.fillStyle = "rgb(192, 80, 77)"; // 赤
			context.arc(100+r1/2,100+r1/2, r1, 0, Math.PI*2, false);
			context.fill();
			r1+=parseInt(Math.random()*20);

			context.beginPath();
			context.fillStyle = "rgb(77, 80, 192)"; // 青
			context.arc(190-r2/2,350-r2/2, r2, 0, Math.PI*2, false);
			context.fill();
```

```
                r2+=parseInt(Math.random()*20);
            }
        }
</script>
</head>
<body>
<img id="img1" src="heart.png" style="position:absolute;left:30px;top:30px" />
<img id="img2" src="heart.png" style="position:absolute;left:200px;top:360px" />
<canvas id="canvas" width="640" height="640"></canvas>
<script type="text/javascript">
        var canvas = document.getElementById("canvas");
        if(canvas.getContext){
            context = canvas.getContext("2d");
        }
</script>
</body>
</html>
```

文法のまとめ

■タッチイベント

・onClick イベントはタッチ動作でもマウス操作でも動作します。ところが、onMouseMove や onDragStar などのマウス専用イベントはタッチ操作では捕捉できませんのでスマートフォンやタブレットでは動作しません。逆にタッチイベントはマウス操作を行うパソコンでは動作しません。

・タッチ操作を捕捉するには touchstart、touchend、touchmove などのタッチ専用のイベントを使用します。

■タッチイベントの種類とプロパティ

・タッチイベントとして以下があります。

タッチイベントの種類	機能
touchstart	タッチ開始時に発生。
touchend	タッチ終了時に発生。
touchmove	タッチ移動時に発生。
touchcancel	システム割り込み（着信、アラームなど）によるキャンセルで発生。

・touches 配列にタッチしている指のオブジェクトが格納されています。touches[0]で 1 本目の指、touches[1]で 2 本目の指のオブジェクトが取得できます。touches オブジェクトには以下のプロパティがあります。

touches のプロパティ	機能
identifier	タッチポイントの ID。
clientX/clientY	クライアント領域（viewport）に対する座標。
pageX/pageY	ページ全体（html 要素）に対する座標。
screenX/screenY	画面の表示領域に対する座標。
target	イベントの発生元。

■イベントリスナーの組み込み

・タッチイベントは個々の要素の onXXX 属性に指定することもできますが、addEventListener で組み込むこともできます。

・touchstrat イベントを組み込むには以下のようにします。デフォルトのタッチ動作を無効にするには「event.preventDefault()」を実行します。

```
document.addEventListener("touchstart", function(event){
    event.preventDefault();
    // event.touches[0].clientX と event.touches[0].clientY でタッチ位置の座標
});
```

■タッチターゲット

タッチした要素は target プロパティで取得できますので、「event.touches[0].target.id」とすればタッチした要素の ID が取得できます。

第７回　ハノイの塔

ハノイの塔とは次のようなパズルゲームです。

「３本の棒 a、b、c がある。棒 a に、中央に穴の空いた n 枚の円盤が大きい順に積まれている。これを１枚ずつ移動させて棒 b に移す。ただし、移動の途中で円盤の大小が逆に積まれてはならない。また、棒 c は作業用に使用するものとする。」

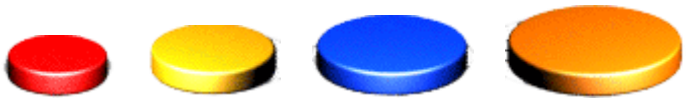
Step1　台に円盤を積む
Step2　ドラッグドロップで円盤を移動
Step3　円盤を移動できるか判定
Step4　再帰解（アラート版）
Step5　再帰解（手順を記録）

Step1　台に円盤を積む

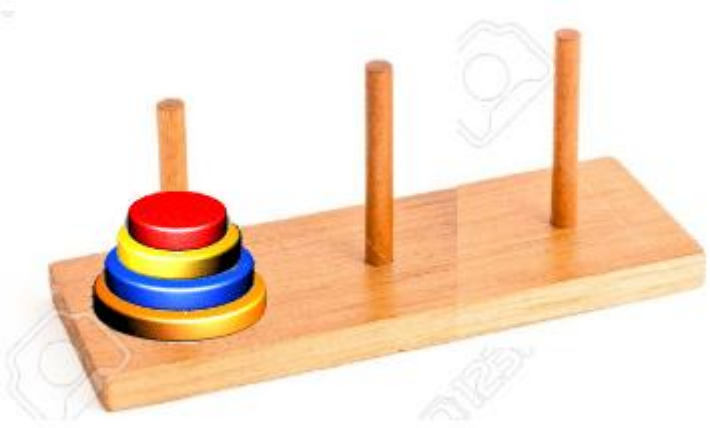
台に円盤を積みます。円盤の枚数はテキストボックスから入力します（最大４枚まで)。

・台のイメージは dai.png です。

・円盤のイメージは ita1.png～ita4.png の４枚です。小さい円盤が影に隠れないように、イメージの zIndex は小さい円盤が高くなるように設定します。



・初期状態は棒 a（1 番）に全ての円盤を置きます。sp[1],sp[2],sp[3]で各棒に入っている円盤の枚数を表します。

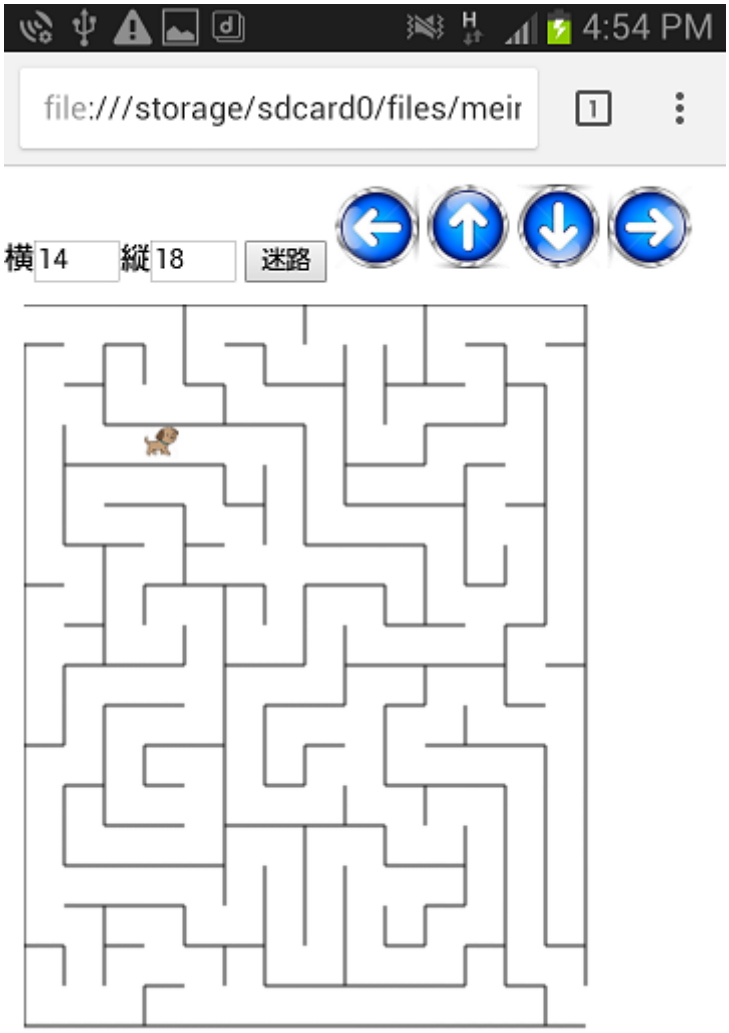


・hanoi1.html

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0,
  maximum-scale=1.0, user-scalable=no" />
<script type="text/javascript">
```

```
// ハノイの塔(台に円盤を積む)

    var sp=new Array(4);   // 各棒の円盤の枚数
    function disp()
    {
        var n=document.getElementById("t1").value;   // 円盤の枚数
        sp[1]=n;sp[2]=0;sp[3]=0;
        for (var i=1;i<=4;i++){
            obj=document.getElementById("ita"+i);
            if (i<=n){
                obj.style.visibility="visible";
                obj.style.left=180+(4-i)*5+"px";
                obj.style.top=(170+(i-n)*10)+"px";
            }
            else
                obj.style.visibility="hidden";
        }
    }
</script>
</head>
<body>
<script type="text/javascript">
    for (var i=1;i<=4;i++){
        var img=new Image();
        img.src="ita"+i+".png";
        img.id="ita"+i;
        img.style.position="absolute";
        img.style.zIndex=5-i;                    // zIndex の設定
        img.style.visibility="hidden";
        document.body.appendChild(img);
    }
</script>
<input id="t1" type="text" size="4" />
<input type="button" value="ハノイ" onClick="disp()" /><br />
<img src="dai.png" style="position:absolute;left:135px;top:50px;z-index:0" />
</body>
</html>
```

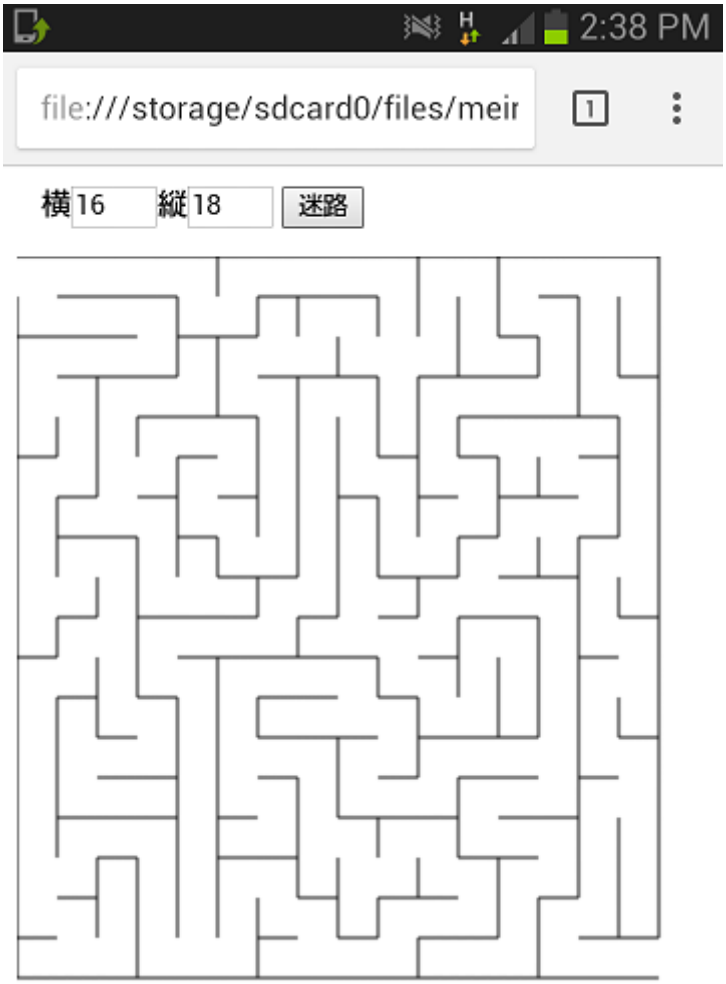
第８回　迷路



迷路ゲームは「迷路の作成」と「迷路の探索」の 2 つの部分から構成されます。迷路の探索を再帰を用いて解法する方法と矢印ボタンでゲーマーが移動する方法を示します。

Step1　迷路の作成
Step2　迷路の探索と経路の描画
Step3　迷路を矢印ボタンで上下左右に移動

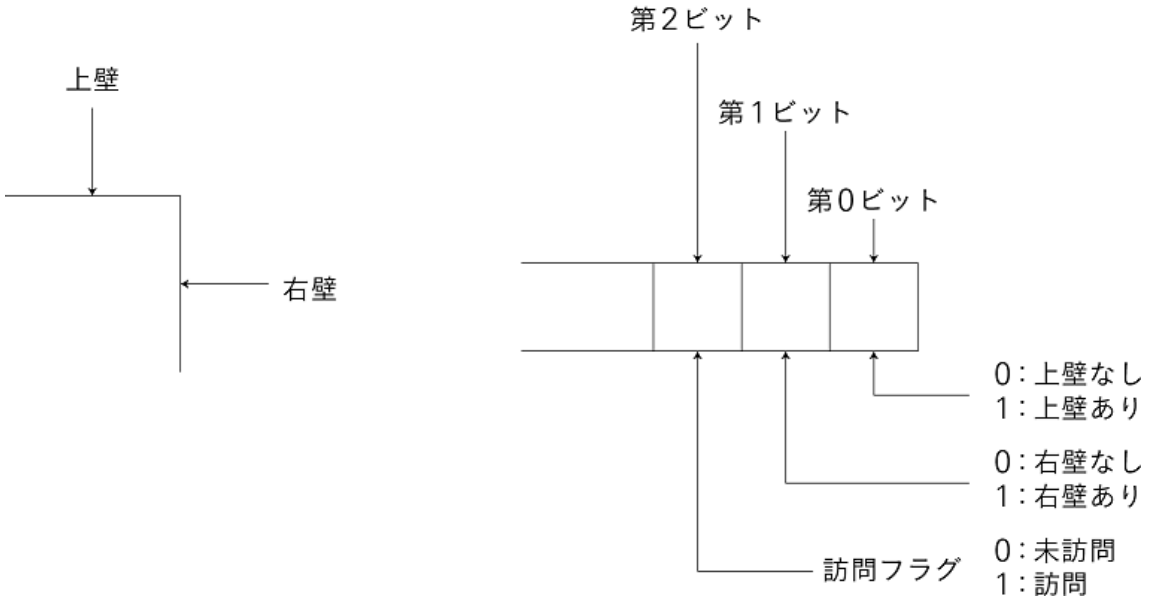
Step1　迷路の作成



・迷路配列の作成を genmaze 関数で行い、その迷路配列を元に迷路を描くのは meiro 関数で行います。

・縦 NY、横 NX の各マスに迷路配列 M[ ][ ]を対応させ、その各要素に下図に示すような上壁の有無、右壁の有無、訪問の有無の 3 つの情報を持たせます。

壁情報のビット表現



・(i,j) 位置から進む方向を乱数で 1：右、2：下、3：左、4：上の中から選択し、1 マス進むことにします。その方向に進めるかは、そのマスがまだ未訪問である場合とします。これを進む方向がなくなるまで再帰的に繰り返します。全てのマスに上壁、右壁をつけて置き、進む方向の壁を取り去るという方法で迷路を作ります。通過に伴う壁の取り崩しは、次の要領で行います。

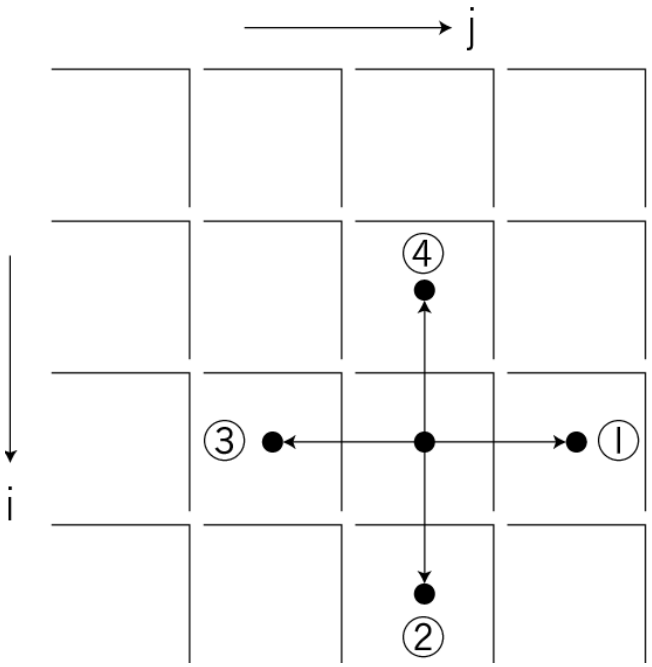
①右へ進む場合は、今いる位置の右壁をとる
②下へ進む場合は、進む位置の上壁をとる
③左へ進む場合は、進む位置の右壁をとる
④上へ進む場合は、今いる位置の上壁をとる

・具体的には、たとえば M[i][j]位置の右壁をとるには次のようにします。

```
M[i][j]&=0xd;
```

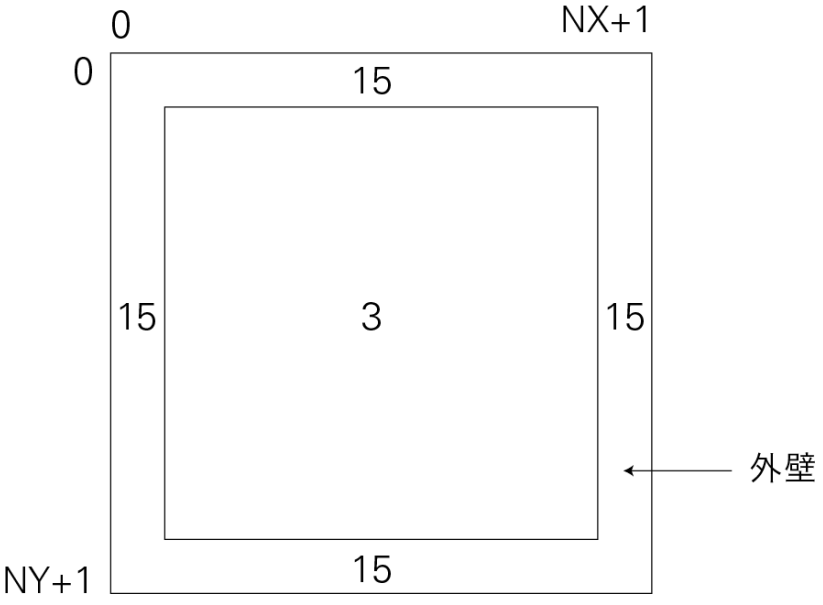
0x0d のビットパターンは「1101」で、これとの AND をとることにより、第 1 ビットだけを 0 にすることができます。

進む方向



・迷路配列の初期値は、外壁に番兵の 15（再トライ禁止、訪問済み、右壁あり、上壁あり）を置き、各マスには 3（未訪問、右壁あり、上壁あり）を置きます。

迷路配列の初期値



・壁の長さを W、迷路の左上隅位置を(0,0)、迷路の入り口と出口の配列要素を（Si,Sj)と(Ei,Ej)とします。配列の i、j 位置に対応するマスの y、x 位置は以下の式で得られます。

```
y=(i-1)*W;
x=(j-1)*W;
```

この y、x 位置に「M[i][j] & 1」が真なら上壁を

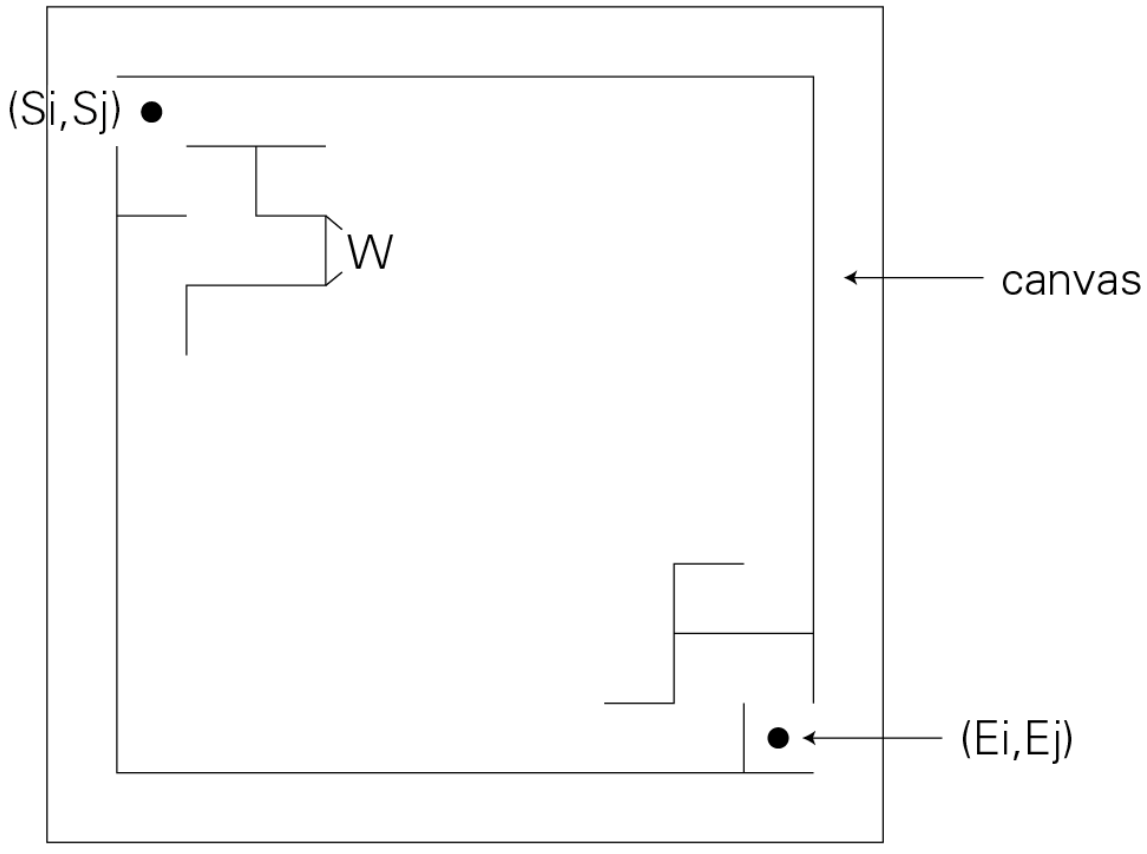
```
context.moveTo(x,y);
context.lineTo(x+W,y);
```

で描き、「M[i][j] & 2」が真なら右壁を

```
context.moveTo(x+W,y);
context.lineTo(x+W,y+W);
```

で描きます。

壁の描画



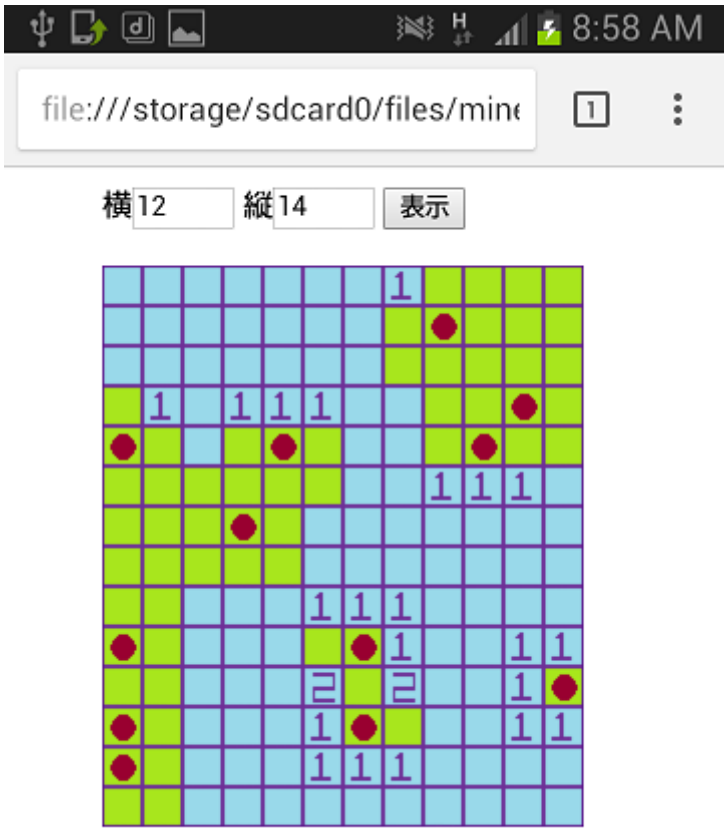
・meiro1.html

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0,
  maximum-scale=1.0, user-scalable=no" />
<script type="text/javascript">

// 迷路の作成

      var context;
      var NY,NX,Si,Sj,Ei,Ej;   // マスの縦と横、入口位置、出口位置
      var W=20;                        // 壁の長さ
      var M;              // 迷路配列
      var sp=0,RI,RJ;

      function genmaze(i,j)   // 迷路配列の作成
      {
            var n;
            M[i][j]|=4;
            while (M[i][j+1]==3 || M[i+1][j]==3 || M[i][j-1]==3 || M[i-1][j]==3){
                  n=parseInt(Math.random()*4)+1;
                  if (n==1 && M[i][j+1]==3){
                        M[i][j]&=0xd;          // 右へ進む場合は、今いる位置の右壁をとる
                        genmaze(i,j+1);
                  }
                  else if(n==2 && M[i-1][j]==3){
                        M[i][j]&=0xe;          // 下へ進む場合は、進む位置の上壁をとる
                        genmaze(i-1,j);
                  }
                  else if(n==3 && M[i][j-1]==3){
                        M[i][j-1]&=0xd;       // 左へ進む場合は、進む位置の右壁をとる
                        genmaze(i,j-1);
                  }
                  else if(n==4 && M[i+1][j]==3){
                        M[i+1][j]&=0xe;       // 上へ進む場合は、今いる位置の上壁をとる
                        genmaze(i+1,j);
                  }
            }
      }
      function meiro()
      {
            var i,j;
            NY=parseInt(document.getElementById("NY").value);
            NX=parseInt(document.getElementById("NX").value);
            Si=1;Sj=1;Ei=NY;Ej=NX;
            oldY=10;oldX=10;H=0;V=0;
            M=new Array(NY+2);
```

```
        RI=new Array(300);
        RJ=new Array(300);
        for (i=0;i<NY+2;i++){
            M[i]=new Array(NX+2);
        }
        for (i=0;i<NY+2;i++){   // 迷路配列の初期化
            for (j=0;j<NX+2;j++){
                if (i==0 || j==0 || i==NY+1 || j==NX+1)
                    M[i][j]=15;
                else
                    M[i][j]=3;
            }
        }

        genmaze(Ei,Ej);
        M[Ei][Ej]&=0xd;

        context.clearRect(0,0,640,640);   // 迷路の描画
        context.strokeStyle="black";
        context.beginPath();
        context.moveTo(0,W);
        context.lineTo(0,W*NY);
        context.lineTo(W*NX,W*NY);
        for (i=1;i<=NY;i++){
            for (j=1;j<=NX;j++){
                x=(j-1)*W;
                y=(i-1)*W;
                if (M[i][j] & 1){   // 上壁の描画
                    context.moveTo(x,y);
                    context.lineTo(x+W,y);
                }
                if (M[i][j] & 2){   // 右壁の描画
                    context.moveTo(x+W,y);
                    context.lineTo(x+W,y+W);
                }
            }
        }
        context.stroke();
    }
</script>
</head>
<body>
<div style="position:absolute;left:20px;top:10px">
横<input id="NX" type="text" size="3">縦<input id="NY" type="text" size="3" />
<input type="button" value="迷路" onClick="meiro()" />
</div>
<br /><br />
<canvas id="canvas" width="640" height="640"></canvas>
<script type="text/JavaScript">
```

```
        var canvas = document.getElementById("canvas");
        if(canvas.getContext){
            context = canvas.getContext("2d");
        }
</script>
</body>
</html>
```

第９回　マインスイーパー



マスの中に複数の機雷を置きます。開けたマスが機雷でなければ、そのマスに隣接する 8 マスに隠されている機雷の個数が表示されますので、それをたよりにマスを空けていきます。機雷の置かれているマスを開けると負けです。minesweeper は（機雷除去の）掃海艇、または地雷除去装置の意味です。

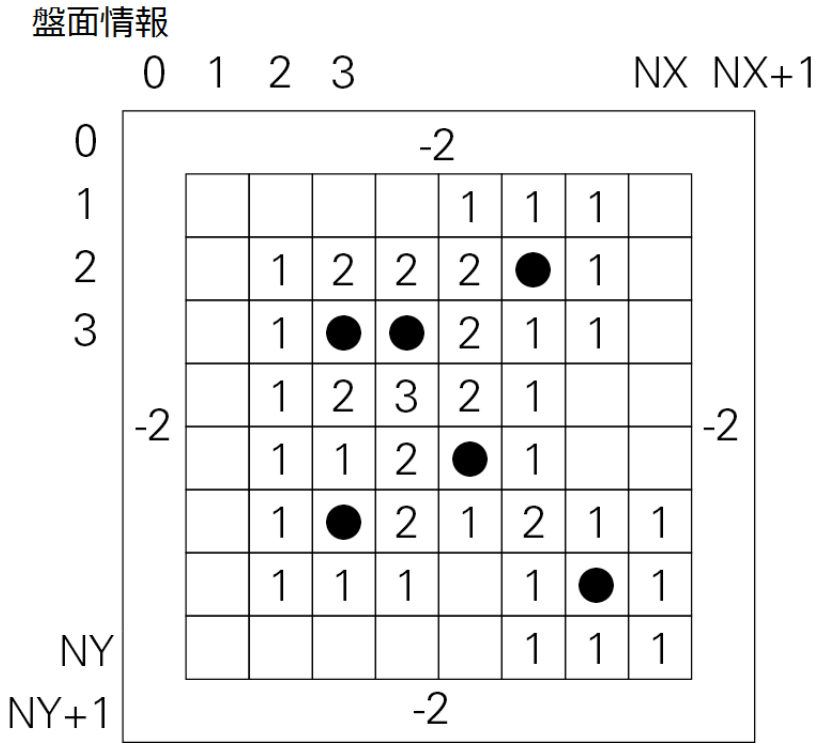
Step1　盤面を作る
Step2　マスを開く
Step3　接近度数 0 の領域を開く

Step1　盤面を作る

マインスイーパーの盤面を作ります。各マスに複数の機雷を配置し、各マスごとに機雷の接近度数を調べます。

■機雷の接近度数

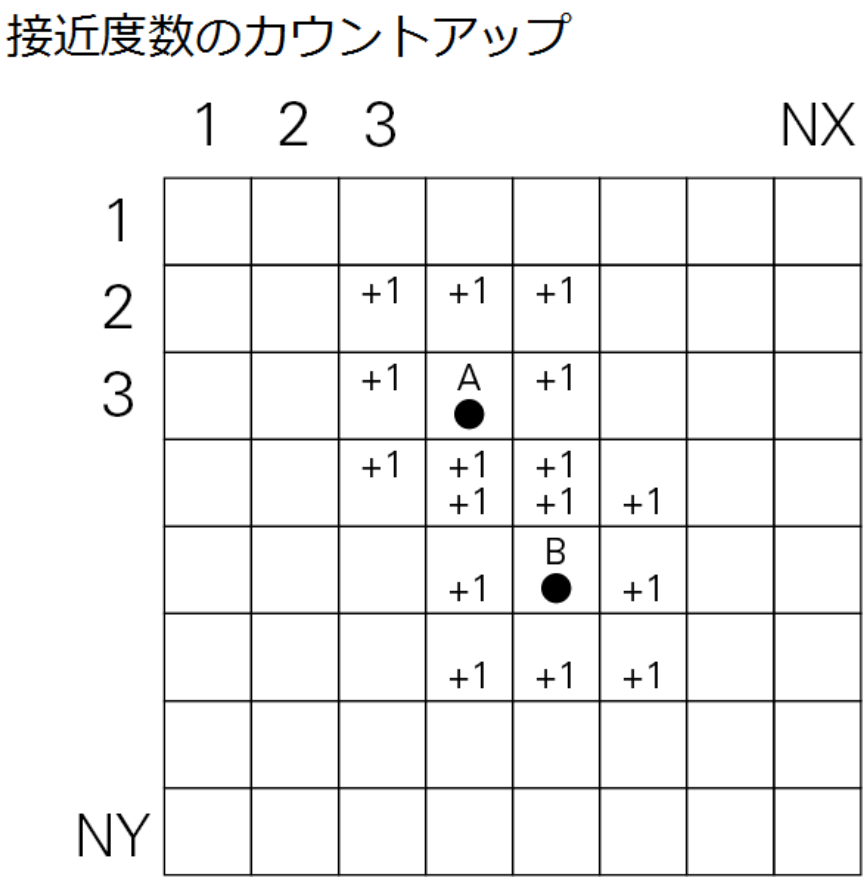
あるマスを中心に周辺 8 箇所のマスに機雷のある数を機雷の接近度数と呼びます。マインスイーパーの盤面情報を配列 M[][]に格納します。配列の各要素には機雷があれば-1、外枠要素は-2、それ以外の要素はその要素の隣接するところに機雷がある個数（接近度数）を格納します。例えば下図で M[2][3]の要素は機雷が 2 個隣接しているので 2 となります。隣接する機雷がないところは 0 となります。(1,1)－(NY,NX)を実際のマスと対応させます。配列の外枠要素の「-2」は Step3 で作る visit 関数で再帰呼び出しを行う際に、配列の外に進まないようにするためのものです。このように配列の範囲を越えて探索を行なわないように見張りをしているものを番兵と呼びます。



マスを開いたときに表示するイメージは接近度数 0,1,2・・に対し num0.png,num1.png,num2.png・・を使います。num1.png 以後は数字入りですが num0.png は数字は入っていません。

■機雷の接近度数をカウントアップする関数 count

機雷の接近度数を示す数値は、機雷を中心とする周辺 8 箇所を+1 することでカウントアップして求めます。このときその周辺 8 箇所の中で機雷のある要素と外枠の要素はカウントアップしません。下図に A 位置と B 位置でのカウントアップの様子を示してあります。



■オブジェクトの作成

表示ボタンで再度オブジェクトを作るような場合は appendChild でなく innerHTML を使います。appendChild ではオブジェクトを削除しなければなりません。

・mine1.html

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0,
  maximum-scale=1.0, user-scalable=no" />
<script type="text/javascript">

// マインスイーパー（盤面を作る）
```

```
var N;        // マスの数
var Bomb;   // 機雷の数
var W=20;   // マスのサイズ
var M;        // 盤面情報

function count()   // 接近度数カウント関数
{
    var i,j;
    for (i=1;i<=N;i++){
        for (j=1;j<=N;j++){
            if (M[i][j]==-1){
                for (dx=-1;dx<=1;dx++){
                    for (dy=-1;dy<=1;dy++){
                        if  ((dx!=0  ||  dy!=0)  &&  (M[i+dy][j+dx]!=-1  &&
M[i+dy][j+dx]!=-2))
                            M[i+dy][j+dx]++;
                    }
                }
            }
        }
    }
}
function init()
{
    var i,j,x,y,n=0,tag="";
    N=parseInt(document.getElementById("level").value);
    Bomb=N;
    for (y=50;y<50+W*N;y+=W){   // マスの配置
        for (x=50;x<50+W*N;x+=W){
            tag+="<img                src='sea.png'                id='img"+n+"'
style='position:absolute;left:"+x+"px;top:"+y+"px' onClick='yx(event)' />";
            n++;
        }
    }
    document.getElementById("canvas").innerHTML=tag; // キャンバスに盤面
を描く

    M=new Array(N+2);
    for (i=0;i<N+2;i++){ // 盤面情報の初期化
        M[i]=new Array(N+2);
        for (j=0;j<=N+1;j++){
            if (i==0 || i==N+1 || j==0 || j==N+1)
                M[i][j]=-2;  // 外枠要素。番兵
            else
                M[i][j]=0;
        }
    }
```

```
        for (i=0;i<Bomb;i++){   // y,x 位置のマスに機雷を最大で Bomb 個置く
            x=Math.floor(Math.random()*N+1);
            y=Math.floor(Math.random()*N+1);
            M[y][x]=-1;
        }
        count();
    }
</script>
</head>
<body>
<div style="position:absolute;left:50px;top:10px" />
マスの数<input id="level" type="text" size="4" />
<input type="button" value="表示" onClick="init()" />
</div>
<div id="canvas">
</div>
</body>
</html>
```

文法のまとめ

■innerHTML プロパティ

・innerHTML プロパティを使えば、親タグで囲まれた内部の HTML 要素を書きかえることができます。

```
<div id="msg">
ここの内容が変わります
</div>
```

という HTML 要素に対し、

```
var msg=document.getElementById("msg");
msg.innerHTML="<h1>ようこそ</h1>";
```

とすると ・・・ タグで囲まれた範囲の HTML 文が書き換えられます。 が親タグでその内部（inner）の HTML という意味です。

Step2　マスを開く

クリックしたマスを開きます。開けたマスの盤面情報 M[y][x]を元に、表示するイメージのファイル名を"num"+M[y][x]+".png"で作成します。開けたマスが機雷の場合は全ての機雷を表示します。

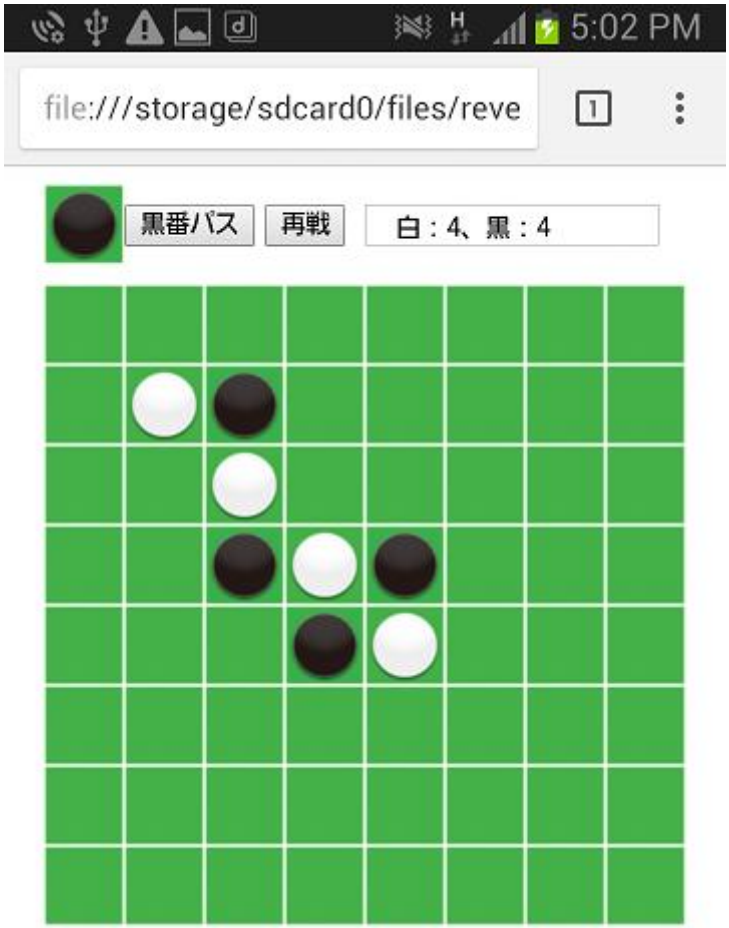
・mine2.html

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0,
  maximum-scale=1.0, user-scalable=no" />
<script type="text/javascript">

// マインスイーパー（マスを開く）

        var NX,NY; // マスの数
        var Bomb;   // 機雷の数
        var W=20;   // マスのサイズ
        var M;      // 盤面情報

        function yx(event)   // クリックイベント処理関数
        {
            var y,x,n,obj;
            y=Math.floor((event.clientY-50)/W+1);
            x=Math.floor((event.clientX-50)/W+1);
            n=(y-1)*NX+(x-1);
            obj=document.getElementById("img"+n);
            if (M[y][x]==-1){   // 機雷のときは全ての機雷を表示
                for (i=1;i<=NY;i++){
                    for (j=1;j<=NX;j++){
                        if (M[i][j]==-1){
                            n=(i-1)*NX+(j-1);
                            obj=document.getElementById("img"+n);
                            obj.src="bomb.png";
                        }
                    }
                }
            }
            else
                obj.src="num"+M[y][x]+".png";       // 機雷でないとき
        }
        function count()   // 接近度数カウント関数
        {
            var i,j;
            for (i=1;i<=NY;i++){
```

```
                for (j=1;j<=NX;j++){
                    if (M[i][j]==-1){
                        for (dx=-1;dx<=1;dx++){
                            for (dy=-1;dy<=1;dy++){
                                if ((dx!=0 || dy!=0) && (M[i+dy][j+dx]!=-1 &&
M[i+dy][j+dx]!=-2))
                                    M[i+dy][j+dx]++;
                            }
                        }
                    }
                }
            }
        }
        function init()
        {
            var i,j,x,y,n=0,tag="";
            NX=parseInt(document.getElementById("NX").value);
            NY=parseInt(document.getElementById("NY").value);
            Bomb=NX;
            for (y=50;y<50+W*NY;y+=W){ // マスの配置
                for (x=50;x<50+W*NX;x+=W){
                    tag+="<img src='sea.png' id='img"+n+"'
style='position:absolute;left:"+x+"px;top:"+y+"px' onClick='yx(event)' />";
                    n++;
                }
            }
            document.getElementById("canvas").innerHTML=tag; // キャンバスに盤面
を描く

            M=new Array(NY+2);
            for (i=0;i<NY+2;i++){ // 盤面情報の初期化
                M[i]=new Array(NX+2);
                for (j=0;j<=NX+1;j++){
                    if (i==0 || i==NY+1 || j==0 || j==NX+1)
                        M[i][j]=-2; // 外枠要素。番兵
                    else
                        M[i][j]=0;
                }
            }
            for (i=0;i<Bomb;i++){   // y,x 位置のマスに機雷を最大で Bomb 個置く
                x=Math.floor(Math.random()*NX+1);
                y=Math.floor(Math.random()*NY+1);
                M[y][x]=-1;
            }
            count();
        }
</script>
</head>
<body>
```

```
<div style="position:absolute;left:50px;top:10px" />
横<input id="NX" type="text" size="4" />
縦<input id="NY" type="text" size="4" />
<input type="button" value="表示" onClick="init()" />
</div>
<div id="canvas">
</div>
</body>
</html>
```

第１０回　リバーシー



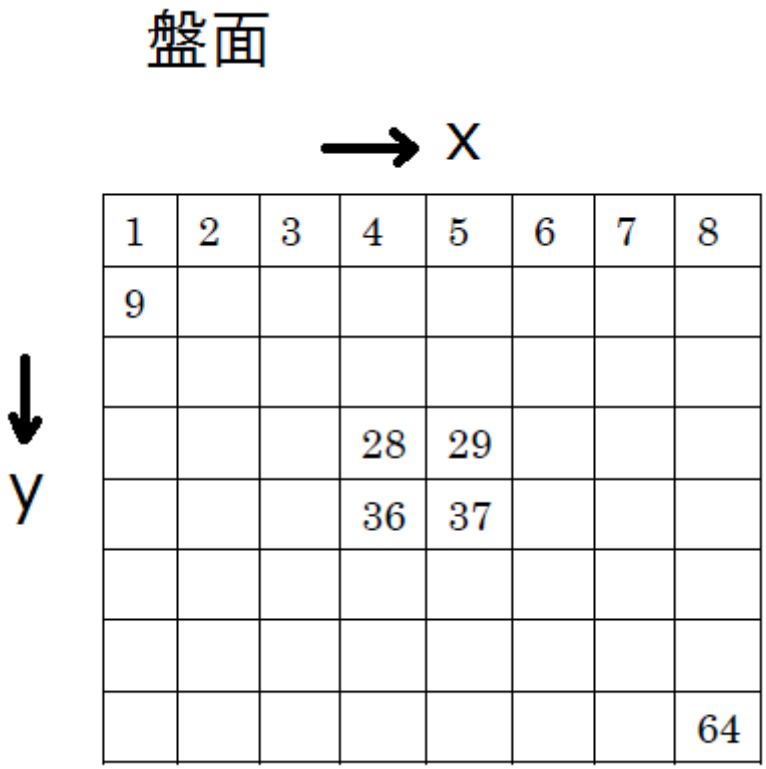
リバーシーゲームを以下のような段階を追って作ります。

Step1　盤面を作る
Step2　黒石を置く
Step3　盤面の情報を配列に置く
Step4　黒番白番で交互に置く
Step5　石を置ける位置かどうかチェック
Step6　自動的に反転する
Step7　コンピュータが手を打つ
Step8　コンピュータに戦略を持たせる
Step9　４隅を取る戦略
Step10　完成版

Step1　盤面を作る

・盤面を作る init 関数を作ります。8×8 のマスに 3 種類のイメージを配置します。配置するイメージには img1～img64 の通し番号を割り振ります。イメージのサイズは 40×40 ピクセルとします。

green.png　　緑（石を置いていない状態）のイメージファイル。
white. png　　白を置いた状態のイメージファイル。
black. png　　黒を置いた状態のイメージファイル。



盤面の横を x、縦を y で管理し 64 のマスに各イメージを以下の方法で配置します。y を外、x を内とする for の二重ループを作ります。変化するものは以下のものです。

・イメージファイルの種類

盤面の初期状態は 28 番、37 番が白、29 番、36 番が黒、その他は緑（石を置いていない状態）とします。img.src に該当するイメージファイル名を設定します。

・イメージの通し番号

変数 n にイメージの id に使う通し番号（1 スタート）をカウントアップしていきます。

・イメージの左隅上の y、x 座標

1 番目のマス位置を（60,20）とします。イメージの表示位置は y を 60 から、x を 20 から始めて 40 刻みで変化させます。

・reversi1.html

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0,
maximum-scale=1.0, user-scalable=no" />
<script type="text/javascript">

// リバーシー（盤面を作る）

    function init()   // 盤面初期化関数
    {
        var y,x,img,n=1;
        for (y=60;y<60+40*8;y+=40){
            for (x=20;x<20+40*8;x+=40){
                img=new Image();
                if (n==28 || n==37){          // 白石
                    img.src="white.png";
                }
                else if (n==29 || n==36){   // 黒石
                    img.src="black.png";
                }
                else {                              // 石無し
                    img.src="green.png";
                }
                img.id="img"+n;
                img.style.position="absolute";
                img.style.left=x+"px";
                img.style.top=y+"px";
                document.body.appendChild(img);
                n++;
            }
        }
        img=new Image();   // 手番を示す石
        img.src="black.png";
        img.id="img65";
        img.style.position="absolute";
        img.style.left="20px";
        img.style.top="10px";
        document.body.appendChild(img);
    }
</script>
</head>
<body>
<script type="text/javascript">
    init();
</script>
</body>
```

</html>

著者略歴

河西 朝雄（かさいあさお）

山梨大学工学部電子工学科卒（1974 年）。長野県岡谷工業高等学校情報技術科教諭、長野県松本工業高等学校電子工業科教諭を経て、現在は「カサイ．ソフトウエアラボ」代表。

「主な著書」

「入門ソフトウエアシリーズ C 言語」、「同シリーズ Java 言語」、「同シリーズ C++」、「入門新世代言語シリーズ VisualBasic4.0」、「同シリーズ Delphi2.0」、「やさしいホームページの作り方シリーズ HTML」、「同シリーズ JavaScript」、「同シリーズ HTML 機能引きテクニック編」、「同シリーズホームページのすべてが分かる事典」、「同シリーズ i モード対応 HTML と CGI」、「同シリーズ i モード対応 Java で作る i アプリ」、「同シリーズ VRML2.0」、「チュートリアル式言語入門 VisualBasic.NET」、「はじめての VisualC#. NET」、「C 言語用語辞典」ほか（以上ナツメ社）

「構造化 BASIC」、「Microsoft Language シリーズ Microsoft VISUAL C++初級プログラミング入門上、下」、「同シリーズ VisualBasic 初級プログラミング入門上、下」、「C 言語によるはじめてのアルゴリズム入門」、「Java によるはじめてのアルゴリズム入門」、「VisualBasic によるはじめてのアルゴリズム入門」、「VisualBasic6.0 入門編、中級テクニック編、上級編」、「Internet Language 改訂新版シリーズ ホームページの制作」、「同シリーズ JavaScript 入門」、「同シリーズ Java 入門」、「New Language シリーズ標準 VisualC++プログラミングブック」、「同シリーズ標準 Java プログラミングブック」、「VB.NET 基礎学習 Bible」、「原理がわかるプログラムの法則」、「プログラムの最初の壁」、「河西メソッド：C 言語プログラム学習の方程式」、「基礎から学べる VisualBasic2005 標準コースウエア」、「基礎から学べる JavaScript 標準コースウエア」、「基礎から学べる C 言語標準コースウエア」、「基礎から学べる PHP 標準コースウエア」、「なぞりがき C 言語学習ドリル」、「C 言語標準ライブラリ関数ポケットリファレンス[ANSI C,ISO C99 対応]」、「 C 言語　標準文法ポケットリファレンス[ANSI C,ISOC99 対応]」、「[標準] C 言語重要用語解説 ANSI C / ISO C99 対応」ほか（以上技術評論社）

「電子書籍：カサイ．ソフトウエアラボ」

「Android プログラミング Bible 初級 基礎編」、「Android プログラミング Bible 中級 Android 的プログラミング法」、「Android プログラミング Bible 上級　各種処理」、「Android プログラミング完全入門」、「iPhone&iPad プログラミング Bible[上] 」、「iPhone&iPad プログラミング Bible[下] 」、「JavaScript によるはじめてのアルゴリズム入門 」、「Web アプリ入門（HTML5+JavaScript）」、「HTML5 を使った JavaScript 完全入門 」、「Scratch プログラミング入門」、「小・中学生のための Scratch プログラミング入門」、「ideon で学ぶ小・中学生のためのプログラミング入門 C 言語編」、「同 Java 言語編」

JavaScript+HTML5 でブラウザゲームに挑戦

2018 年 1 月 20 日　初版　第 1 刷

著者＝河西 朝雄
発行者＝河西 朝雄
発行所＝カサイ．ソフトウエアラボ
長野県茅野市ちの 813　TEL.0266-72-4778

表紙デザイン＝河西 朝樹

定価＝1,500 円＋税