3章　UI要素の操作

代表的なUI要素として<input>、<textarea>、<select>があります。

<input>要素は何種類かの入力要素を指定するもので、type属性に"text"、"button"、"radio"、"checkbox"を指定することができます。それぞれテキストボックス、ボタン、ラジオボタン、チェックボックスを意味します。

<textarea>は複数行のテキストエリア、<select>は選択ボックスを作ります。

この章ではUI要素の操作方法を説明します。

「補足」<input>、<textarea>、<select>などのUI要素は以下のように<form>要素の中に記述し、submitボタンでフォームの内容を一括サーバーに送信するという場合に良く使います。このため<input>、<textarea>、<select>などのUI要素はフォーム要素（部品）と呼ばれることがあります。

<form method="post" action="cgi-bin/form.cgi">

<input type="text"・・・>

<input type="text"・・・>

<input type="submit" value "送信"・・・>

</form>

ただし、このようにフォームの内容を一括して扱うような場合でなければ、<input>、<textarea>、<select>などのUI要素を<form>要素の中に置かなければいけないということではありません。本書ではこのような使用をしないので、<form>要素の中に置いていません。

3-1　テキストボックスとテキストエリア

　1行のテキストを扱うテキストボックスと複数行のテキストを扱うテキストエリアの操作方法を説明します。

1.　テキストボックスとボタン

　<input>タグは1行で構成される入力要素です。type属性にtext(テキストボックス)、button（ボタン）、radio（ラジオボタン）、checkbox（チェックボックス）、submit（サブミットボタン）などを指定します。sizeに文字数、valueに表示する文字列を指定します。

　「登録」と表示されたボタンを配置し、クリックしたときのイベント処理関数addを指定するには次のようにします。

<input type="button" value="登録" onClick="add()">

　4文字サイズのテキストボックスを配置するには以下のようにします。

<input id="t1" type="text" size="4">

このテキストボックスの内容を取得/設定するにはvalue属性を用いて以下のようにします。

var t1=document.getElementById("t1");

t1.value="メッセージ";

2.　テキストエリア

　<textarea>タグでテキストエリアを定義します。rows属性に行数、cols属性に1行の文字数を指定します。

<textarea id="ta1" rows="10" cols="20"></textarea>

このテキストエリアの内容を取得/設定するにはvalue属性を用いて以下のようにします。テキストエリア内での改行はエスケープシーケンスコードの"\n"を使います。

var ta1=document.getElementById("ta1");

ta1.value="メッセージ1\nメッセージ2";

「例題3-1」登録ボタンのクリックでテキストボックスから入力された名前をテキストエリアに追加表示します。

<!DOCTYPE html>

<head>

<script type="text/javascript">

function add()

{

var t1=document.getElementById("t1");

var ta1=document.getElementById("ta1");

ta1.value+=t1.value+"\n";

}

</script>

</head>

<body>

<input id="t1" type="text" size="12" />

<input type="button" value="登録" onClick="add()" /><br />

<textarea id="ta1" rows="10" cols="20"></textarea>

</body>

</html>



「補足」「例題3-1」をテキストエリアでなく、<div>領域に表示するにはinnerHTMLプロパティを使います。innerHTMLプロパティを使った表示における改行はHTMLタグの"<br />"を使います。

<!DOCTYPE html>

<head>

<script type="text/javascript">

function add()

{

var t1=document.getElementById("t1");

var div1=document.getElementById("div1");

div1.innerHTML+=t1.value+"<br />";

}

</script>

</head>

<body>

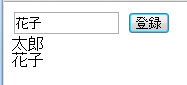
<input id="t1" type="text" size="12" />

<input type="button" value="登録" onClick="add()" /><br />

<div id="div1"></div>

</body>

</html>



「練習問題3-1」国旗表示ボタンのクリックでテキストボックスに入力されたイメージファイル名のイメージを<img>タグに表示しなさい。

<!DOCTYPE html>

<head>

<script type="text/javascript">

function disp()

{

var t1=document.getElementById("t1");

var img1=document.getElementById( ① );

② ;

}

</script>

</head>

<body>

<input id="t1" type="text" size="12" />

<input type="button" value="国旗表示" onClick="disp()" /><br />

<img id="img1" src="canada.gif" />

</body>

</html>



3-2　テキストデータの数値化

　テキストボックスの内容をvalueプロパティで取得した場合の値は文字列として扱われます。たとえば、2つのテキストボックスの内容が"10"と"20"であったときに、「t1.value+t2.value」を行えば、結果は"1020"のように文字列の連結となります。

　内容を数値として加算したい場合はevalメソッドで数値文字列を数値化してから加算します。

eval(t1.value)+eval(t2.value)

evalメソッドは「数値文字列→数値」のような単純な変換だけでなく、eval("10+20\*3")のような式を計算して70という値を返してくれます。

「例題3-2」ボタンのクリックで2つの入力ボックスの内容を加算した値を3番目の入力ボックスに表示します。

<!DOCTYPE html>

<head>

<script type="text/javascript">

function calc()

{

var t1=document.getElementById("t1");

var t2=document.getElementById("t2");

var t3=document.getElementById("t3");

t3.value=eval(t1.value)+eval(t2.value);

}

</script>

</head>

<body>

<input id="t1" type="text" size="8" />＋

<input id="t2" type="text" size="8" />

<input type="button" value="＝" onClick="calc()" />

<input id="t3" type="text" size="8" />

</body>

</html>



「練習問題3-2」2つの入力ボックスの内容を「+」、「-」、「\*」、「/」ボタンのクリックで加算、減算、乗算、除算を行った結果を表示しなさい。

<!DOCTYPE html>

<head>

<script type="text/javascript">

function calc(ope)

{

var t1=document.getElementById("t1");

var t2=document.getElementById("t2");

var t3=document.getElementById("t3");

switch ( ① ){

case "+":t3.value=eval(t1.value)+eval(t2.value);break;

case "-":t3.value=eval(t1.value)-eval(t2.value);break;

case "\*":t3.value=eval(t1.value)\*eval(t2.value);break;

case "/":t3.value= ② ;break;

}

}

</script>

</head>

<body>

<input id="t1" type="text" size="8" />

<input type="button" value="+" onClick="calc('+')" />

<input type="button" value="-" onClick="calc('-')" />

<input type="button" value="\*" onClick="calc('\*')" />

<input type="button" value="/" onClick="calc('/')" />

<input id="t2" type="text" size="8" />＝

<input id="t3" type="text" size="8" />

</body>

</html>



3-3　ラジオボタン

ラジオボタンは<input>タグのtype属性に"radio"を指定します。name属性に同じ名前を指定したラジオボタンが一つのグループを構成し、その中の一つがチェック状態になります。

1.　チェックされている内容の取得

クリックしたラジオボタンの値は、呼び出し元からのthis引数を、仮引数objに受け取り、obj.valueで取得します。

function check(obj)

{

if (obj.value=="男")

}

<input name="r1" type="radio" value="男" onClick="check(this)">

2.　this

thisはカレントオブジェクトへの参照を表す予約語です。あるイベントが発生したときにthisをイベント処理関数の引数として渡すことにより関数側で、イベントが発生したオブジェクトを参照することができます。

3.　グローバル変数

ラジオボタンのクリックで何かの処理をするのではなく、別のボタンなどのクリックでラジオボタンのチェックされている値を取得するには、グローバル変数を用いて、ラジオボタンのクリックのたびにラジオボタングループのチェックされているラジオボタンの値を代入しておきます。

var rv1;

function check(obj)

{

rv1=obj.value;

}

<input name="r1" type="radio" value="男" onClick="check(this)">

<input name="r1" type="radio" value="女" onClick="check(this)">

「例題3-3」クリックしたラジオボタンの色でクライアント画面のバックグランドカラーを設定します。また設定した色をテキストボックスに表示します。

<!DOCTYPE html>

<html>

<head>

<script type="text/javascript">

function check(obj)

{

document.getElementById("color").value=obj.value;

if (obj.value=="青色")

document.bgColor="blue";

else if (obj.value=="黄色")

document.bgColor="yellow";

else

document.bgColor="green";

}

</script>

</head>

<body>

背景を<input id="color" type="text" size="4" />に設定します<br>

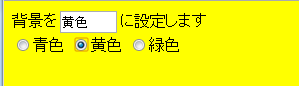
<input name="r1" type="radio" value="青色" onClick="check(this)" />青色

<input name="r1" type="radio" value="黄色" onClick="check(this)" />黄色

<input name="r1" type="radio" value="緑色" onClick="check(this)" />緑色

</body>

</html>



「練習問題3-3」男、女のラジオボタングループr1と学生、会社員、主婦のラジオボタングループr2を作り、「確認」ボタンのクリックでチェックされている値をテキストボックスに表示しなさい。

<!DOCTYPE html>

<html>

<head>

<script type="text/javascript">

var rv1,rv2;

function radio1(obj)

{

① ;

}

function radio2(obj)

{

② ;

}

function disp()

{

var result=document.getElementById("result");

result.value=rv1+":"+rv2;

}

</script>

</head>

<body>

性別<br />

<input name="r1" type="radio" value="男" onClick="radio1(this)" />男<br />

<input name="r1" type="radio" value="女" onClick="radio1(this)" />女<br />

職業<br />

<input name="r2" type="radio" value="学生" onClick="radio2(this)" />学生<br />

<input name="r2" type="radio" value="会社員" onClick="radio2(this)" />会社員<br />

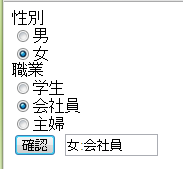
<input name="r2" type="radio" value="主婦" onClick="radio2(this)" />主婦<br />

<input type="button" value="確認" onClick="disp()" />

<input id="result" type="text" size="10" />

</body>

</html>



3-4　チェックボックス

　チェックボックスは<input>タグのtype属性に" checkbox "を指定します。name属性に同じ名前を指定したチェックボックスが一つのグループを構成します。

1.　getElementsByNameメソッド

同じ名前のチェックボックスオブジェクトを配列に取得するにはgetElementsByNameメソッドを使い、その結果を用いて、fruits[i]でi番目のチェックボックスを取得できます。

var fruits=document.getElementsByName("fruit");

<input name="fruit" type="checkbox" value="りんご">りんご<br>

<input name="fruit" type="checkbox" value="みかん">みかん<br>

<input name="fruit" type="checkbox" value="スイカ">スイカ<br>

2.　checkedプロパティ

チェックボックスがチェックされているか否かはcheckedプロパティで調べることができます。i番目のチェックボックスがチェックされているかは以下で判定します。

if (fruits[i].checked)

「例題3-4」りんご～いちごのチェックボックスでチェックされている項目を「確認」ボタンのクリックでテキストエリアに表示します。

<!DOCTYPE html>

<html>

<head>

<script type="text/javascript">

function check()

{

var result=document.getElementById("result");

result.value="";

var fruits=document.getElementsByName("fruit");

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

if (fruits[i].checked)

result.value+=fruits[i].value+"\n";

}

}

</script>

</head>

<body>

好きなくだものを選んでください<br />

<input name="fruit" type="checkbox" value="りんご" />りんご<br />

<input name="fruit" type="checkbox" value="みかん" />みかん<br />

<input name="fruit" type="checkbox" value="スイカ" />スイカ<br />

<input name="fruit" type="checkbox" value="メロン" />メロン<br />

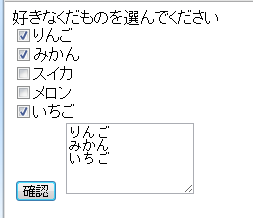
<input name="fruit" type="checkbox" value="いちご" />いちご<br />

<input type="button" value="確認" onClick="check()" />

<textarea id="result" rows="5" cols="15"></textarea>

</body>

</html>



「練習問題3-4」fruitとsweetという2つのチェックボックスグループを作り、それぞれのグループでチェックされている項目をテキストエリアに表示しなさい。

<!DOCTYPE html>

<html>

<head>

<script type="text/javascript">

function check()

{

var result=document.getElementById("result");

result.value="くだもの:\n";

var fruits=document.getElementsByName("fruit");

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

if (fruits[i].checked)

result.value+=fruits[i].value+"\n";

}

var sweets=document.getElementsByName("sweet");

result.value+="スイーツ:\n";

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

if ( ① )

result.value+= ② +"\n";

}

}

</script>

</head>

<body>

好きなくだものを選んでください<br />

<input name="fruit" type="checkbox" value="りんご" />りんご<br />

<input name="fruit" type="checkbox" value="みかん" />みかん<br />

<input name="fruit" type="checkbox" value="いちご" />いちご<br />

好きなスイーツを選んでください<br />

<input name="sweet" type="checkbox" value="パンケーキ" />パンケーキ<br />

<input name="sweet" type="checkbox" value="グラノーラ" />グラノーラ<br />

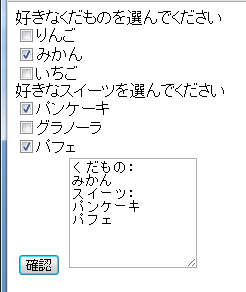
<input name="sweet" type="checkbox" value="パフェ" />パフェ<br />

<input type="button" value="確認" onClick="check()" />

<textarea id="result" rows="8" cols="15"></textarea>

</body>

</html>



3-5　選択ボックス

1.　<select>タグ

　選択ボックスは全体を<select>・・</select>で囲み、その中に各項目を<option>で指定します。<option>タグのvalue属性に項目の文字列を指定します。選択ボタン（▼）のクリックで選択項目が前の状態から変化するとonChangeイベントが発生します。

<select onChange="check(this)">

<option value="img1.gif">イメージ１

</select>

2.　options[ ]コレクション

選択ボックスの各項目はoptions[ ]コレクションで取得できます。同種のオブジェクトを集めたものをコレクションと呼びます。実体は配列です。選択ボックスの選択されている項目の番号（0スタート）はselectedIndexプロパティで取得できます。選択ボックスの項目の選択で、選択項目を取得するには次のようにします。

function check(box)

{　　　　　↓選択ボックスの各項目を示すコレクション

box.options[box.selectedIndex].value

}　　　　　　　　　　　↑　　　　↑値を示すプロパティ

　　　　　　　　　　　　選択されている項目の番号

「例題3-5」選択ボックスで選択した職業をテキストボックスに表示します。

<!DOCTYPE html>

<html>

<head>

<script type="text/javascript">

function check(box)

{

var text=document.getElementById("text");

text.value=box.options[box.selectedIndex].value;

}

</script>

</head>

<body>

<input type="text" id="text" size="10" value="学生" />

<select onChange="check(this)">

<option value="学生">学生</option>

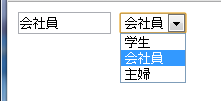
<option value="会社員">会社員</option>

<option value="主婦">主婦</option>

</select>

</body>

</html>



「練習問題3-5」選択ボックスで選択した国の国旗のイメージを表示しなさい。

<!DOCTYPE html>

<html>

<head>

<script type="text/javascript">

function check(box)

{

var img=document.getElementById( ① );

② =box.options[box.selectedIndex].value;

}

</script>

</head>

<body>

<img id="img" src="canada.gif" />

<select onChange="check(this)">

<option value="canada.gif">カナダ</option>

<option value="korea.gif">韓国</option>

<option value="australia.gif">オーストラリア</option>

</select>

</body>

</html>



「補足」リストボックス

　<select>要素にsize属性を指定して項目の個数を指定すればリストボックスになります。size属性に3以下の値を指定しても「4」とみなされます。

<!DOCTYPE html>

<html>

<head>

<script type="text/javascript">

function check(box)

{

var text=document.getElementById("text");

text.value=box.options[box.selectedIndex].value;

}

</script>

</head>

<body>

<input type="text" id="text" size="10" value="学生" />

<select size="4" onChange="check(this)">

<option value="学生">学生</option>

<option value="会社員">会社員</option>

<option value="主婦">主婦</option>

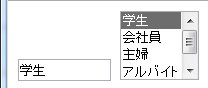
<option value="アルバイト">アルバイト</option>

<option value="無職">無職</option>

</select>

</body>

</html>



3-6　createElementメソッドによるHTML要素の生成

JavaScriptコードでHTML要素をオブジェクトとして生成するにはcreateElementメソッドにタグ名を指定します。たとえば、ボタンオブジェクトを生成するにはcreateElementメソッドの引数に"input"を指定してオブジェクトを生成し、typeプロパティに"button"を指定します。

var btn=document.createElement("input");

btn.type="button";

btn.value="a";

btn.addEventListener("click",disp);

document.body.appendChild(btn);

イメージオブジェクトはImageコンストラクタを使って「var img=new Image();」のように生成することは「2-4　オブジェクトの生成」で説明しましたが、createElementメソッドを使って以下のように生成することもできます。

var img=document.createElement("img");

「例題3-6」「A」～「M」と「N」～「Z」のボタンを2段に配置し、クリックしたボタンのアルファベットをテキストボックスに連結します。「クリア」ボタンでテキストボックスの内容を消去します。ボタンに表示するアルファベットは配列alpha[ ]に格納しておきます。ボタンMの直後に改行（<br>）を入れます。

<!DOCTYPE html>

<html>

<head>

<script type="text/javascript">

function disp(event)

{

var obj=document.getElementById("result");

var btn=event.target;

obj.value+=btn.value;

}

function cls()

{

var obj=document.getElementById("result");

obj.value="";

}

</script>

</head>

<body>

<input id="result" type="text" size="40" />

<input type="button" value="クリア" onClick="cls()" /><br />

<script type="text/javascript">

var alpha=new Array("A","B","C","D","E","F","G","H","I","J","K","L","M",

"N","O","P","Q","R","S","T","U","V","W","X","Y","Z");

var i;

for (i=0;i<26;i++){

var btn=document.createElement("input");

btn.type="button";

btn.value=alpha[i];

btn.addEventListener("click",disp);

document.body.appendChild(btn);

if (i==12){

document.body.appendChild(document.createElement("br"));

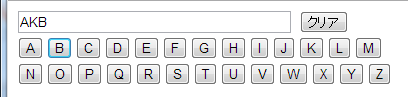
}

}

</script>

</body>

</html>



「練習問題3-6」キーボードの並びをQWERTY配列にしなさい。ボタンPの直後に改行と１つの全角空白を、ボタンLの直後に改行と2つの全角空白を入れます。文字列要素はdocument.createTextNode("文字列")で作成できます。

<!DOCTYPE html>

<html>

<head>

<script type="text/javascript">

function disp(event)

{

var obj=document.getElementById("result");

var btn=event.target;

obj.value+=btn.value;

}

function cls()

{

var obj=document.getElementById("result");

obj.value="";

}

</script>

</head>

<body>

<input id="result" type="text" size="36" />

<input type="button" value="クリア" onClick="cls()" /><br />

<script type="text/javascript">

var alpha=new Array("Q","W","E","R","T","Y","U","I","O","P",

"A","S","D","F","G","H","J","K","L",

"Z","X","C","V","B","N","M");

var i;

for (i=0;i<26;i++){

var btn=document.createElement("input");

btn.type="button";

btn.value=alpha[i];

btn.addEventListener("click",disp);

document.body.appendChild(btn);

if ( ① ){

document.body.appendChild(document.createElement("br"));

document.body.appendChild(document.createTextNode("　"));

}

if ( ② ){

document.body.appendChild(document.createElement("br"));

document.body.appendChild(document.createTextNode("　　"));

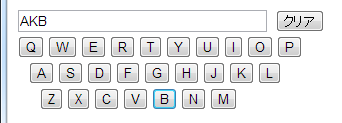
}

}

</script>

</body>

</html>



☆応用サンプル3-1　電卓

「0」～「9」、「+」、「-」、「\*」、「/」のボタンで数値および演算子を示す文字を入力し、「12+5\*6/3-2」のような式文字列を作り「Enter」ボタンで式文字列を評価する電卓を作ります。「Enter」キーの入力直後の「0」～「9」の入力時にはテキストボックスをクリアするものとします。この判定に変数EnterFlagを使用します。「Enter」キー入力直後のEnterFlagをtrueにし、「Enter」キーの入力直後の「0」～「9」の入力でfalseにします。

evalメソッドはeval("10+20\*3")のような式を計算して70という値を返してくれます。

<!DOCTYPE html>

<html>

<head>

<script type="text/javascript">

var EnterFlag=false;

function disp(n)

{

var obj=document.getElementById("result");

if (EnterFlag){ // Enterキー直後

obj.value="";

EnterFlag=false;

}

obj.value+=n;

}

function enter()

{

var obj=document.getElementById("result");

obj.value=eval(obj.value); // 式の計算

EnterFlag=true;

}

function cls()

{

var obj=document.getElementById("result");

obj.value="";

}

</script>

</head>

<body>

<input id="result" type="text" size="20" /><br />

<input type="button" value=" 7 " onClick="disp(7)" />

<input type="button" value=" 8 " onClick="disp(8)" />

<input type="button" value=" 9 " onClick="disp(9)" />

<input type="button" value="Cls" onClick="cls()" /><br />

<input type="button" value=" 4 " onClick="disp(4)" />

<input type="button" value=" 5 " onClick="disp(5)" />

<input type="button" value=" 6 " onClick="disp(6)" />

<input type="button" value="\*" onClick="disp('\*')" />

<input type="button" value="/" onClick="disp('/')" /><br />

<input type="button" value=" 1 " onClick="disp(1)" />

<input type="button" value=" 2 " onClick="disp(2)" />

<input type="button" value=" 3 " onClick="disp(3)" />

<input type="button" value="+" onClick="disp('+')" />

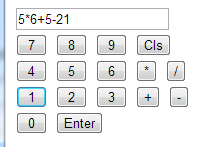
<input type="button" value="-" onClick="disp('-')" /><br />

<input type="button" value=" 0 " onClick="disp(0)" />

<input type="button" value="Enter" onClick="enter()" />

</body>

</html>



☆応用サンプル3-2　3択クイズ

問に対し3択の中から解答を選択する形式のクイズを作ります。設問を配列t[ ],3択の解答候補を配列x[ ]、y[ ]、z[ ]、３択の正解番号を配列a[ ]に格納しておきます。「次の問」ボタンのクリックでテキストボックスtoiに設問を、テキストボックスm1、m2、m3に3択の解答候補を表示します。「解答」ボタンのクリックでラジオボタンのチェック番号と正解番号a[ ]を比較し、正解か誤答かを判定します。

「注」ランダムな順列

　同じ値を使わないという規則を入れた乱数列をランダムな順列と呼びます。たとえば、0～5のランダムな順列とは、「3,2,5,1,0,4」のようなものです。ランダムな順列は問題番号をランダムに選ぶ時などに使います。乱数については6章の「6-1　Mathオブジェクト」を参照してください。

0~N-1のランダムな順列を配列に設定するには以下のようにします。

・0~N-1の数値を順序よく配列r[ ]に入れておく。

・乱数を使って配列の添字を作り、その要素と配列の右端（やるごとに左へ１つ移る）を交換することを繰り返す。乱数はやるだびに範囲を一つ減らしていく。

<!DOCTYPE html>

<html>

<head>

<title>クイズ</title>

<script type="text/javascript">

var N=5;

var r=new Array(N);

var tmp,count=-1;

for (var i=0;i<N;i++) // ランダムな順列

r[i]=i;

for (var i=N-1;i>0;i--){

j=parseInt ((N-1)\*Math.random()); // N-1をiに変更すべき

tmp=r[i];r[i]=r[j];r[j]=tmp;

}

var t=new Array("フランスの首都は",

"重力を発見した人は",

"７６年周期で現れるすい星は",

"ＺＡＲＤのボーカルは",

"我思う故に我有りと言ったのは");

var x=new Array("パリ",

"パスカル",

"ハレーすい星",

"酒井法子",

"カント");

var y=new Array("ロンドン",

"ダーウイン",

"バルタンすい星",

"坂井泉水",

"ヘーゲル");

var z=new Array("オタワ",

"ニュートン",

"エンケすい星",

"坂井真紀",

"デカルト");

var a=new Array(1,3,1,2,3);

function answer()

{

var toi=document.getElementById("toi");

var radio=document.getElementsByName("radio");

for (var i=0;i<3;i++){

if (radio[i].checked){

if (i+1==a[r[count]])

toi.value="正解";

else

toi.value="誤り";

}

}

}

function next()

{

if (count<N-1){

count++;

document.getElementById("toi").value="問"+(count+1)+":"+t[r[count]];

document.getElementById("m1").value=x[r[count]];

document.getElementById("m2").value=y[r[count]];

document.getElementById("m3").value=z[r[count]];

}

else

document.getElementById("toi").value="終わり";

}

</script>

</head>

<body onLoad="next()">

<input type="text" id="toi" size="40" /><br />

<input type="radio" name="radio" />

<input type="text" id="m1" size="20" /><br />

<input type="radio" name="radio" />

<input type="text" id="m2" size="20"><br />

<input type="radio" name="radio" />

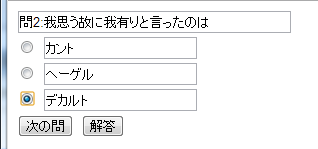
<input type="text" id="m3" size="20" /><br />

<input type="button" value="次の問" onClick="next()" />

<input type="button" value="解答" onClick="answer()" />

</body>

</html>



☆応用サンプル3-3　万年歴

西暦と月をテキストボックスから入力し、その年月のカレンダーを表示します。

グレゴリオ暦では以下の規則で閏年を設けます。

①西暦の年数が4で割り切れる年を閏年とする。

②西暦の年数が100で割り切れる年は閏年からはずす。

③西暦の年数が400で割り切れる年は閏年に戻す。

プログラムの主なポイントは以下です

・ｙ年が閏年かの判定で2月の日数を設定する

　先に示した①~③の規則から西暦ｙ年が閏年である条件は、「yが4で割り切れかつ100でわりきれない」または「ｙが400で割り切れる」となります。

・ｙ年m月１日の曜日を調べる

グレゴリオ暦は1582年に導入されたので、それ以前の年に対しては意味をもちませんが、計算の都合上、西暦１年１月１日を月曜日とします。

365日を１週間の7で割れば１余るので、閏年がなければ、翌年の１月１日の曜日は今年の１月１日の曜日の次になります。従って、もし閏年がないとしたら、西暦ｙ年ｍ月１日の曜日は日曜日からｙ日分ズレることになります。しかし実際には①~③の規則に従って閏年が入っているので、実際のズレbiasは以下のように表せます。閏年の補正はy-1年分が対象となります。

bias=parsInt(y)+parsInt((y-1)/4)-parsInt((y-1)/100)+parsInt((y-1)/400)

また、ｙ年の１月～(m-1)月までの合計日数は

sum=mt[1]+mt[2]+・・・+mt[m-1]

となりますから、結局ｙ年ｍ月１日の曜日は

w=(bias+sum)%7

であらわせ、wが0なら日曜日、・・・6なら土曜日と判定できます。

・カレンダの表示

　日を表示するテキストボックス配列をDay[0]～Day[41]としたとき、その月のｉ日を表示するテキストボックス配列の番号はw+iとなります。

<!DOCTYPE html>

<html>

<head>

<title>万年暦</title>

<script type="text/javascript">

function mannen()

{

var mt=new Array(0,31,28,31,30,31,30,31,31,30,31,30,31);

var y,m,sum,w,bias;

y=document.getElementById("Year").value;

m=document.getElementById("Month").value;

if ((y % 4)==0 && (y % 100)!=0 || (y % 400)==0)

mt[2]=29;

else

mt[2]=28;

bias=parseInt(y)+parseInt((y-1)/4)-parseInt((y-1)/100)+parseInt((y-1)/400);

sum=0;

for (var i=1;i<m;i++)

sum+=mt[i];

w=(bias+sum) % 7;

var Day=document.getElementsByName("Day");

for (var i=0;i<42;i++)

Day[i].value="";

for (var i=0;i<mt[m];i++)

Day[w+i].value=i+1;

}

</script>

</head>

<body>

<input type="text" id="Year" size="5">年<input type="text" id="Month" size="3" />月

<input type="button" name="calc" value="万年暦" onClick="mannen()" /><br /><br />

<script type="text/javascript">

var x,y;

var week=new Array("日","月","火","水","木","金","土");

for (var i=0;i<7;i++){

var box=document.createElement("input");

box.type="text";

box.value=week[i];

box.size="2";

document.body.appendChild(box);

}

document.body.appendChild(document.createElement("br"));

for (var i=0;i<6;i++){

for (var j=0;j<7;j++){

var box=document.createElement("input");

box.type="text";

box.name="Day";

box.size="2";

document.body.appendChild(box);

}

document.body.appendChild(document.createElement("br"));

}

</script>

</body>

</html>



5章　一定時間ごとの処理

イメージやテキストを一定時間ごとに表示したり移動したりしたい場合があります。一定時間ごとの処理を行う方法としてsetIntervalメソッドを用いる方法と、setTimeoutメソッドを用いる方法があります。この章ではイメージを一定時間ごとに移動したり交互に表示したりする基礎的な方法を説明します。一定時間ごとの処理の応用として「9章　ラケットゲーム」でさらに詳しく説明します。

5-1　setIntervalメソッド

一定時間ごとに「ある処理」を行うにはsetIntervalメソッドを以下のように使います。これで「指定ミリ秒」ごとに「ある処理」が繰り返されます。

setInterval(function(){

// ある処理

},指定ミリ秒);

「例題5-1」"おはよう"、"こんにちは"、"おやすみ"という3つのテキストを1000ミリ秒ごとに順次テキストボックスに表示します。

<!DOCTYPE html>

<html>

<body>

<input id="result" type="text" size="20">

<script type="text/javascript">

var msg=new Array("おはよう","こんにちは","おやすみ");

var result=document.getElementById("result");

var n=0;

setInterval(function() {

result.value=msg[n];

n++;

if (n==msg.length) n=0;

},1000);

</script>

</body>

</html>



「練習問題5-1」boy1.gifとboy2.gifという2つのイメージを1000ミリ秒ごとに交互に表示しなさい。

<!DOCTYPE html>

<html>

<body>

<img id="boy" src="boy1.gif" />

<script type="text/javascript">

var boy=document.getElementById("boy");

var flag=false;

setInterval(function() {

if ( ① )

boy.src="boy1.gif";

else

boy.src="boy2.gif";

② ;

},1000);

</script>

</body>

</html>



5-2　タイマーの開始と停止

5-1で説明したsetIntervalメソッドは引数の中に直接「ある処理」を行う関数本体を記述しましたが、以下のように関数本体を別の場所に置くこともできます。

setInterval(関数名,指定ミリ秒);

function 関数名()

{

// ある処理

}

設定したタイマーをクリアするには以下のようにsetInterval実行時の戻り値をtimeID変数に格納しておき、clearInterval(timeID)でクリアします。

timeID=setInterval(関数名,指定ミリ秒);

・

・

clearInterval(timeID);

「例題5-2」開始ボタンのクリックで2つのイメージを1000ミリ秒単位で交互に表示します。

<!DOCTYPE html>

<html>

<body>

<input type="button" value="開始" onClick="setInterval(start,1000)" /><br />

<img id="boy" src="boy1.gif" />

<script type="text/javascript">

var boy=document.getElementById("boy");

var flag=false;

function start()

{

if (flag)

boy.src="boy1.gif";

else

boy.src="boy2.gif";

flag=!flag;

}

</script>

</body>

</html>



「練習問題5-2」停止ボタンでタイマーをクリアしなさい。timeID変数は<head>部のスクリプトブロックでグローバル変数として宣言しておきます。

<!DOCTYPE html>

<html>

<head>

<script type="text/javascript">

var timeID;

</script>

</head>

<body>

<input type="button" value="開始" onClick=" ① " />

<input type="button" value="停止" onClick=" ② " /><br />

<img id="boy" src="boy1.gif" />

<script type="text/javascript">

var boy=document.getElementById("boy");

var flag=false;

function start()

{

if (flag)

boy.src="boy1.gif";

else

boy.src="boy2.gif";

flag=!flag;

}

</script>

</body>

</html>



5-3　イメージの移動

　オブジェクトの位置を制御するにはstyle属性を使います。たとえば、イメージを(50,50)ピクセル位置に配置するには以下のようにします。style属性の各パラメータは「;」で区切り、パラメータと値は「:」で区切ります。pxはピクセル単位を意味します。

<img id="boy" src="boy1.gif" style="position:absolute;left:50px;top:50px" />

|  |  |
| --- | --- |
| style属性のパラメータ | 意味 |
| position | 座標の取り方。absoluteはクライアント画面の左上隅を(0,0)とする絶対座標。他にrelative(相対座標)、fixed（絶対座標で固定）があります。 |
| left | オブジェクトの左上隅のx座標。 |
| top | オブジェクトの左上隅のy座標。 |

このオブジェクトをJavaScript側で制御するには次のようにオブジェクトを取得し、boy.style.leftでx座標、boy.style.topでy座標を取得、あるいは設定することができます。

var boy=document.getElementById("boy");

boy.style.left="100px";

boy.style.top="150px";

ここで注意しなければいけないことは、取得された値に"px"が付いていること、また逆に値を設定する場合は"px"を付加したものを指定しなければならないということです。従って変数xに数値データが入っているときにこの値でleft位置を設定するには以下のように"px"を付加しなければなりません。古い仕様では"px"がなくても自動的に"px"と判定していましたが、新しい仕様では"px"が必ず必要です。

boy.style.left=x+"px";

「例題5-3」開始ボタンのクリックで500ミリ秒ごとにイメージを10ピクセル右に移動します。

<!DOCTYPE html>

<html>

<body>

<input type="button" value="開始" onClick="setInterval(start,500)" /><br />

<img id="boy" src="boy1.gif" style="position:absolute;left:50px;top:50px" />

<script type="text/javascript">

var boy=document.getElementById("boy");

var x=50;

function start()

{

x+=10;

boy.style.left=x+"px";

}

</script>

</body>

</html>



「練習問題5-3」2つのイメージ（boy1.gifとboy2.gif）を交互に切り替えながら移動しなさい。変数flagでどちらのイメージを表示するか管理します。

<!DOCTYPE html>

<html>

<body>

<input type="button" value="開始" onClick="setInterval(start,500)" /><br />

<img id="boy" src="boy1.gif" style="position:absolute;left:50px;top:50px" />

<script type="text/javascript">

var boy=document.getElementById("boy");

var x=50;

var flag=false;

function start()

{

x+=10;

boy.style.left=x+"px";

if (flag)

① ;

else

② ;

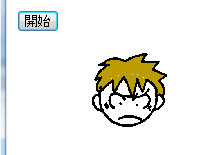
flag=!flag;

}

</script>

</body>

</html>



5-4　setTimeoutメソッド

　一定時間後に「ある処理」をしたい場合はsetTimeoutメソッドを使います。これで「指定ミリ秒」後に「ある処理」が実行されます。

setTimeout(関数名,指定ミリ秒);

function 関数名

{

ある処理

}

setIntervalメソッドのように一定時間ごとに「ある処理」を行うには関数の中で再度setTimeoutメソッドを実行します。途中で一定時間の値を変更したい場合はsetIntervalでなくsetTimeoutを用います。具体例は9章の「応用サンプル9-1」を参照してください。

function 関数名

{

setTimeout(関数名,指定ミリ秒);

ある処理

}

呼び出す関数に引数を渡す場合は以下のように関数名と引数を文字列として連結したものをsetTimeoutの引数に指定します。

setTimeout("関数名("+第1引数+","+第2引数+")",指定ミリ秒);

「例題5-4」マウスの移動に追従してイメージを移動します。このときイメージの移動を100ミリ秒後に遅らせます。

「注」ブラウザのクライアント画面全体でマウスイベントを取得するにはstyle属性を使ってbodyの幅と高さを画面一杯に設定しておかなければなりません。

<body style="position:absolute;width:100%;height:100%" ・・・>

古い仕様ではこのようにしなくてもbodyは画面一杯に設定されていました。9章9-1の「「注」画面サイズ」を参照。

<!DOCTYPE html>

<html>

<head>

<script type="text/javascript">

function move(event){

setTimeout("go("+event.clientX+","+event.clientY+")",100);

}

function go(x,y)

{

var obj=document.getElementById("chara1");

obj.style.left=x+"px";

obj.style.top=y+"px";

}

</script>

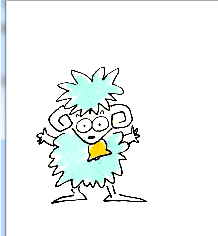
</head>

<body style="position:absolute;width:100%;height:100%" onMouseMove="move(event)">

<img id="chara1" src="chara.gif" style="position:absolute;left:200px;top:100px" />

</body>

</html>



「練習問題5-4」2つのめのイメージを追加しなさい。2つめのイメージは200ミリ秒後に、１つめのイメージの100ピクセル右横に移動しなさい。

<!DOCTYPE html>

<html>

<head>

<script type="text/javascript">

function move(event){

setTimeout("go1("+event.clientX+","+event.clientY+")",100);

setTimeout("go2("+event.clientX+","+event.clientY+")",200);

}

function go1(x,y)

{

var obj=document.getElementById("chara1");

obj.style.left=x+"px";

obj.style.top=y+"px";

}

①

{

var obj=document.getElementById( ② );

obj.style.left=x+100+"px";

obj.style.top=y+"px";

}

</script>

</head>

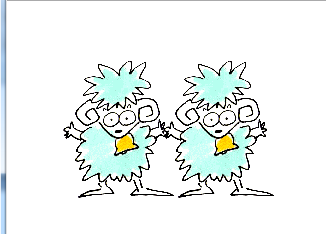
<body style="position:absolute;width:100%;height:100%" onMouseMove="move(event)">

<img id="chara1" src="chara.gif" style="position:absolute;left:200px;top:100px" />

<img id="chara2" src="chara.gif" style="position:absolute;left:300px;top:100px" />

</body>

</html>



6章　JavaScriptの標準オブジェクト

すでにdocumentやeventなどの予め定められたオブジェクトやdocument.getElementByIdによりHTML要素をオブジェクトとして取得する方法を説明しました。

document.getElementByIdにより取得したHTML要素のオブジェクトは任意のオブジェクトですが、documentやeventはJavaScriptが予め定めているオブジェクトです。

JavaScriptにはこの他にMath、Date、Stringなどの標準オブジェクトがあります。この章ではこれらの標準オブジェクトの使い方を説明します。またダイアログ関連のオブジェクトについても説明します。

6-1　Mathオブジェクト

Mathオブジェクトはsin、cos、sqrtなどの数値計算を行なうためのオブジェクトです。

1.　Mathオブジェクトのメソッド

Mathオブジェクトのメソッドは静的メソッドとして定義されていますので、ユーザはMathオブジェクトを生成せずに予約オブジェクト名のMathを用いて次のようにメソッドを使用します。これで、√10の結果が得られます。

Math.sqrt(10);

Mathオブジェクトの主なプロパティとメソッドとして以下があります。abs～minはメソッド、PIはプロパティです。

|  |  |
| --- | --- |
| メソッド/プロパティ | 機能 |
| abs(x) | ｘの絶対値。 |
| sin(x),cos(x),tan(x) | ｘのサイン、コサイン、タンジェント。ｘの単位はラジアン。 |
| acos(x),asin(x)  atan(x),atan2(y,x) | ｘのアークサイン、アークコサイン、アークタンジェント。atan2はy/xのアークタンジェント。戻り値の単位はラジアン。 |
| exp(x) | ex。ｘの指数。 |
| log(x) | logex。ｘの自然対数。 |
| pow(x,y) | xy。ｘのｙ乗。 |
| sqrt(x) | √x。ｘの平方根。 |
| ceil(x) | ｘを下回らない最小整数。 |
| floor(x) | ｘを上回らない最大整数。 |
| round(x) | ｘを直近の整数に丸める。 |
| random() | 0～1未満の乱数。 |
| max(x1,x2,・・・) | 引数の中の最大。 |
| min(x1,x2,・・・) | 引数の中の最小。 |
| PI | 円周率。3.141592653589793。 |

2.　三角関数

sin、cosなどの三角関数の引数の単位は度でなくラジアンです。ラジアンは１単位円の角度に対する円弧の長さです。１単位円の半周の円弧の長さはπでこれが180度に相当しますので、ｘ度をラジアンに変換するには以下のようにします。PIはMathオブジェクトのプロパティでπの値を示します。

x\*Math.PI/180

「例題6-1-1」中心(200,200)、半径150の円周上の点を20°ごとに中心と直線で結びます。

<!DOCTYPE html>

<html>

<body>

<canvas id="canvas" width="400" height="400"></canvas>

<script type="text/javascript">

var canvas = document.getElementById("canvas");

if(canvas.getContext){

var context = canvas.getContext("2d");

context.strokeStyle = "blue"; // 青

for (var i=0;i<360;i+=20){

var x=200+150\*Math.cos(i\*Math.PI/180);

var y=200+150\*Math.sin(i\*Math.PI/180);

context.beginPath();

context.moveTo(200,200);

context.lineTo(x,y);

context.stroke();

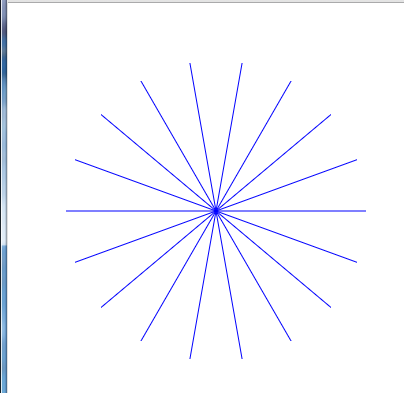
}

}

</script>

</body>

</html>



「練習問題6-1-1」中心(200,200)、半径150の円周上の120°離れた2点を5°置きに結びなさい。

<!DOCTYPE html>

<html>

<body>

<canvas id="canvas" width="400" height="400"></canvas>

<script type="text/javascript">

var canvas = document.getElementById("canvas");

if(canvas.getContext){

var context = canvas.getContext("2d");

context.strokeStyle = "blue"; // 青

for (var i=0;i<360;i+=5){

var x1=200+150\*Math.cos(i\*Math.PI/180);

var y1=200+150\*Math.sin(i\*Math.PI/180);

var x2=200+150\* ① ;

var y2=200+150\* ② ;

context.beginPath();

context.moveTo(x1,y1);

context.lineTo(x2,y2);

context.stroke();

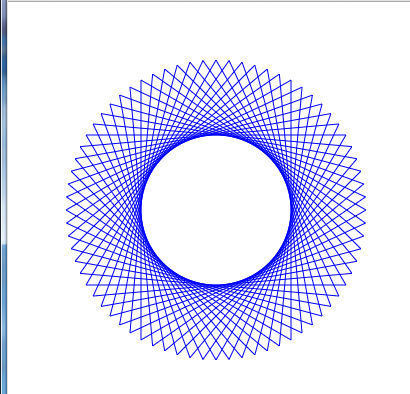
}

}

</script>

</body>

</html>



「練習問題6-1-2」中心(200,200)、半径150の円周上をn分割した各点を総当たりで結びなさい。描かれる図形をダイアモンドリングと呼びます。

<!DOCTYPE html>

<html>

<body>

<canvas id="canvas" width="400" height="400"></canvas>

<script type="text/javascript">

var canvas = document.getElementById("canvas");

if(canvas.getContext){

var context = canvas.getContext("2d");

context.strokeStyle = "blue"; // 青

var n=16; // 分割数

for (var i=0;i<n-1;i++){

var x1=200+150\*Math.cos(i\*2\*Math.PI/n);

var y1=200-150\*Math.sin(i\*2\*Math.PI/n);

for (var j=i+1;j<n;j++){

var x2=200+150\* ① ;

var y2=200-150\* ② ;

context.beginPath();

context.moveTo(x1,y1);

context.lineTo(x2,y2);

context.stroke();

}

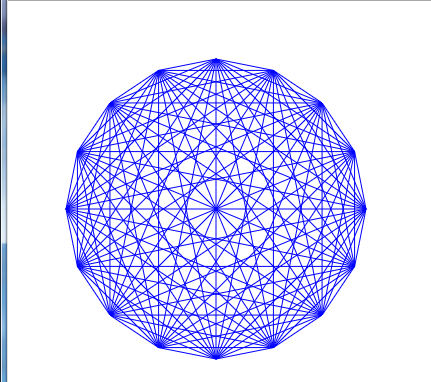
}

}

</script>

</body>

</html>



3.　乱数

　randomメソッドは0.0～1.0未満の実数乱数を1つ取得します。randomメソッドを使ってm～nの整数乱数を作る関数rndは以下のようになります。

function rnd(m,n) // m～nの整数乱数

{

return parseInt(Math.random()\*(n-m+1))+m;

}

mに1、nに3に与えた場合を考えてみます。この場合にはMath.random()\*(n-m+1)はMath.random()\*(3-1+1)、つまりMath.random()\*3となり、0.0～3.0未満の実数値が得られます。これをparseIntで小数点部を切り捨てると0～2の整数値となり、これにm（この場合1）を足すと、1～3の整数乱数が得られることになります。parseIntメソッドの代わりにMath.floorメソッドを使っても同じです。

「例題6-1-2」rnd関数を使って1～3の乱数を発生し、その値で大吉、末吉、凶の占いを行います。

<!DOCTYPE html>

<html>

<head>

<script type="text/javascript">

function rnd(m,n) // m～nの整数乱数

{

return parseInt(Math.random()\*(n-m+1))+m;

}

function uranai()

{

var n=rnd(1,3);

var obj=document.getElementById("result");

switch (n){

case 1:obj.value="大吉";break;

case 2:obj.value="末吉";break;

case 3:obj.value="凶";break;

}

}

</script>

</head>

<body>

<input id="result" type="text" size="8" />

<input type="button" value="占い" onClick="uranai()" />

</body>

</html>



「練習問題6-1-3」rnd関数を使って1～6の乱数を発生し、これをサイコロの目として2個使い、2つの目の合計値が奇数なら「半」、偶数なら「丁」と判定して表示しなさい。

<!DOCTYPE html>

<html>

<head>

<script type="text/javascript">

function rnd(m,n) // m～nの整数乱数

{

return parseInt(Math.random()\*(n-m+1))+m;

}

function uranai()

{

var n1=rnd(1,6);

var n2=rnd(1,6);

var obj=document.getElementById("result");

if ( ① )

obj.value=n1+"と"+n2+"の丁";

else

obj.value=n1+"と"+n2+"の半";

}

</script>

</head>

<body>

<input id="result" type="text" size="8">

<input type="button" value="サイコロ" onClick="uranai()" />

</body>

</html>



6-2　Dateオブジェクト

Dateオブジェクトは日時を管理するオブジェクトで、Dateコンストラクタを使ってインスタンスを生成してから使用します。

var t=new Date();

これでtにシステム時間が取得されます。このtに対しgetFullYearメソッドを適用すると、yに4桁の西暦年が得られます。

var y=t.getFullYear();

　Dateオブジェクトの日時を取得する主なメソッドとして以下があります。

|  |  |
| --- | --- |
| メソッド | 機能 |
| getFullYear() | 4桁の西暦年の取得。 |
| getMonth() | 月の取得。0：1月～11：12月。 |
| getDate() | 日の取得。 |
| getDay() | 曜日の取得。0：日～6：土。 |
| getHours() | 時（0～23）の取得。 |
| getMinutes() | 分の取得。 |
| getSeconds() | 秒の取得。 |
| getMilliseconds() | ミリ秒の取得。 |

「例題6-2」システム時間を「時分秒」形式で1秒ごとにテキストボックスに表示します。

<!DOCTYPE html>

<html>

<body>

<input type "text" id="result" size="16" />

<script type="text/javascript">

var result=document.getElementById("result");

setInterval(function() {

var t=new Date();

var h=t.getHours();

var m=t.getMinutes();

var s=t.getSeconds();

result.value=h+"時"+m+"分"+s+"秒";

},1000);

</script>

</body>

</html>



「練習問題6-2」システム時間を「時分秒」形式で1秒ごとにキャンバスにテキストとして描画しなさい。

<!DOCTYPE html>

<html>

<body>

<canvas id="canvas" width="400" height="200"></canvas>

<script type="text/javascript">

var canvas = document.getElementById("canvas");

if(canvas.getContext){

var context = canvas.getContext("2d");

context.strokeStyle = "blue"; // 輪郭の色

context.font = "36px 'ＭＳ Ｐゴシック'";

}

setInterval(function() {

var t=new Date();

var h=t.getHours();

var m=t.getMinutes();

var s=t.getSeconds();

var msg=h+"時"+m+"分"+s+"秒";

context.clearRect(0,0,400,200); // 領域のクリア

① ;

},1000);

</script>

</body>

</html>



6-3　Stringオブジェクト

1.　文字列の生成

　Stringオブジェクトは、文字列の変数への代入あるいはStringコンストラクタによる明示的な宣言のどちらかで生成できます。

var s="hello";

または

var s=new String("hello");

この文字列オブジェクトに対し、toUpperCaseメソッドを使って

p=s.toUpperCase();

とすると、pには"HELLO"が得られます。またlengthプロパティを使えば文字列の長さを調べることができます。以下はnには5が得られます。英数文字も日本語も１文字として扱います。

n=s.length;

2.　文字列の連結

文字列を含む式の中の＋演算子は文字列連結演算子として機能します。以下は"HelloJavaScript"となります。

"Hello"+"JavaScript"

文字列と数値を＋演算子でつないだ場合、数値は文字列に変換されて連結されます。以下は数値の5が文字列の"5"に変換されて連結されるので、"55"となります。

"5"+5

ただし、5+5+"5"のような場合、5+5が先に10と計算され、それが文字列に変換されて連結されるので、"105"となります。数値演算を明示するには(5+5)+"5"のように、かっこで囲みます。

　変数名と変数の内容を表す文字列を作るには次のようにします。"sum="という文字列に変数sumの内容の100が文字列に変換されて連結されるので、msgは"sum=100"という文字列になります。

var sum=100;

var msg="sum="+sum;

3.　文字列の比較

文字列の比較は「==」演算子を用いて行うことができます。

var girl="Riho";

if (girl=="Riho")

alert(girl);

　文字列中に特定の文字が含まれるかはmatchメソッドを使います。以下はfname中に"green.png"が含まれるか調べます。

var fname="img/src/green.png";

if (fname.match("green.png"))

alert(fname);

4.　部分文字列の取り出し

文字列中から指定した文字列を取り出すメソッドとしてcharAt、charCodeAt、substring、sliceがあります。substringとsliceの違いは引数に負数を与えた場合の解釈だけです。

　var s="JavaScript";

s.charAt(1);・・・"a"。１番目の文字

s.charCodeAt(1);・・・97。１番目の文字の文字コード（ASCIIコード）

s.substring(4,6);・・・"Sc"。4番目～5番目の文字列

s.substring(4);・・・"Script"。4番目～後端までの文字列

s.slice(4,6);・・・"Sc"。4番目～5番目の文字列

s.slice(1,-1);・・・"avaScrip"。１番目の文字～後ろから１番目の文字列

「例題6-3」msgの文字列を左スクロールして表示します。msgの1文字目とそれ以後を入れ替えて連結する処理を500ミリ秒間隔で行います。

<!DOCTYPE html>

<html>

<body>

<input type "text" id="result" size="30" />

<script type="text/javascript">

var result=document.getElementById("result");

var msg="ようこそ私のホームページへ．．．";

setInterval(function() {

result.value=msg;

msg=msg.substring(1,msg.length)+msg.substring(0,1);

},500);

</script>

</body>

</html>



「練習問題6-3」msgの文字列を右スクロールして表示しなさい。msgの最後の文字とそれ以前を入れ替えて連結する処理を500ミリ秒間隔で行います。

<!DOCTYPE html>

<html>

<body>

<input type "text" id="result" size="30" />

<script type="text/javascript">

var result=document.getElementById("result");

var msg="ようこそ私のホームページへ．．．";

setInterval(function() {

result.value=msg;

var n= ① ;

msg=msg.substring(n-1)+ ② ;

},500);

</script>

</body>

</html>



6-4　ダイアログ

　ダイアログ（dialog、dialogue：応答を行うもの）を開くメソッドとしてalert、confirmがあります。ダイアログは元のウインドウとは別に開くユーザとの応答を行うウインドウです。

1.　alertメソッド

alertメソッドはメッセージボックスを開き「メッセージ」を表示します。メッセージを途中で改行するには、エスケープシーケンスの「\n」を入れます。

alert("Hello\nJavaScript");



2.　confirmメソッド

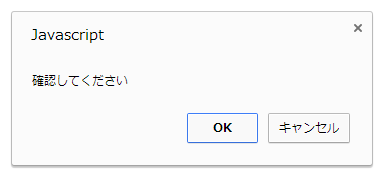
　confirmメソッドは確認ダイアログボックスを開き「メッセージ」を表示し、「OK」ボタンをクリックするとtrue、「キャンセル」ボタンをクリックするとfalseを返します。

if (confirm("確認してください"))

alert("OKボタンがクリックされました");

else

alert("キャンセルボタンがクリックされました");



「注」windowオブジェクト

windowオブジェクトはブラウザのウインドウを管理するオブジェクトで、Browserオブジェクトの一番上位階層に位置します。このため、windowオブジェクトのプロパティやメソッドは「window.」を付けずに使用できます。alert、confirmはwindowオブジェクトのメソッドです。windowの下位オブジェクトとしてdocumentオブジェクト、historyオブジェクト、locationオブジェクト、navigatorオブジェクト、screenオブジェクトがあります。

「例題6-4」配列Ques[ ]に5つの問題、Ans[ ]のその解答をtrue（Yes）、false（No）の形式で格納してあるものとします。confirmメソッドで問題を表示し、YesならOKボタン、Noならキャンセルボタンをクリックすることで解答します。

<!DOCTYPE html>

<html>

<head>

<script type="text/javascript">

var Ques=new Array("重力の発見者はニュートンである",

"我思う故に我ありと言ったのはヘーゲルである",

"オーストリアの首都はシドニーである",

"７６年周期で現れる彗星はバルタン彗星である",

"AKB48の32ndシングルは「恋するフォーチュンクッキー」である");

var Ans=new Array(true,false,false,false,true);

function toi()

{

var i,n=0;

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

if (confirm(Ques[i])==Ans[i])

n++;

}

alert(n+"問正解");

}

</script>

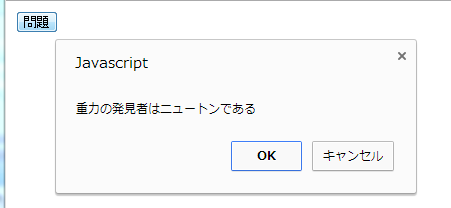
</head>

<body>

<input type="button" value="問題" onClick="toi()" />

</body>

</html>



「練習問題6-4」以下のような3つの質問にYes/Noで答え、判定された文字列を連結して表示しなさい。

・「日本人ですか？」という質問にYesと答えれば「日本人」、Noと答えれば「外国人」と判定。

・「女性ですか？」という質問にYesと答えれば「女性」、Noと答えれば「男性」と判定。

・「未成年ですか？」という質問にYesと答えれば「未成年」、Noと答えれば「成年」と判定。

<!DOCTYPE html>

<html>

<head>

<script type="text/javascript">

var Ques=new Array("日本人ですか？",

"女性ですか？",

"未成年ですか？");

var AnsYes=new Array("日本人","女性","未成年");

var AnsNo=new Array("外国人","男性","成年");

function toi()

{

var i,msg="";

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

if (confirm(Ques[i]))

msg+= ① +"で";

else

msg+= ② +"で";

}

alert("あなたは"+msg+"すね");

}

</script>

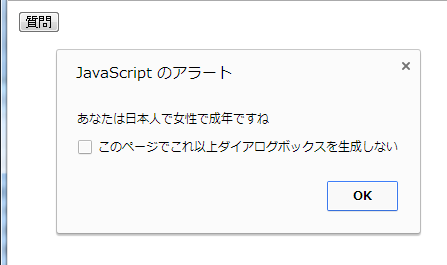
</head>

<body>

<input type="button" value="質問" onClick="toi()" />

</body>

</html>



☆応用サンプル6-1　相性占い

　2人の名前から相性度を求めます。相性度は名前をcharCodeAtメソッドを使って１文字ずつ文字コードとして取り出し、すべてを加算し、その結果を101で割った余りとします。

<!DOCTYPE html>

<html>

<head>

<script type="text/javascript">

function uranai()

{

var i,couple,s=0;

var boy=document.getElementById("boy");

var girl=document.getElementById("girl");

couple=boy.value+girl.value;

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

s+=couple.charCodeAt(i);

}

var result=document.getElementById("result");

result.value=s%101+"%";

}

</script>

</head>

<body>

<input id="boy" type="text" size="12" />と

<input id="girl" type="text" size="12" />の

<input type="button" value="相性度" onClick="uranai()" />は

<input id="result" type="text" size="6" />

</body>

</html>



10章　データ構造とアルゴリズム

　プログラムを使って問題を解くための論理または手順をアルゴリズム（algorithms：算法）といいます。プログラム処理では多量のデータを扱うことが多く、この場合、取り扱うデータをどのようなデータ構造（data structure）にするかで、問題解決のアルゴリズムが異なってきます。データ構造とアルゴリズムは密接な関係にあり、良いデータ構造を選ぶことが良いプログラムを作ることにつながります。

　プログラムの世界に特有なアルゴリズムとして再帰という考え方があります。再帰の代表的な例としてハノイの塔を紹介します。さらに長さと角度を与えて直線を引くタートル・グラフィックスや再帰を用いたリカーシブ・グラフィックス（再帰図形）を紹介します。

　データ構造としては、決定木を用いた食文化判定や性格判定プログラムを紹介します。

10-1　再帰を用いたハノイの塔の解法

　再帰という考え方は人間の一般的な感覚からは縁遠いものですがプログラムの世界では重要な考え方です。再帰の例として階乗、ハノイの塔について説明します。

1.　再帰とは

　再帰的（リカーシブ）な構造とは、自分自身（n次）を定義するのに、自分自身より1次低い部分集合（n-1次）を用い、さらにその部分集合は、より低次の部分集合を用いて定義するということを繰り返す構造です。このような構造を一般に再帰と呼んでいます。

　再帰を用いると、複雑なアルゴリズムを明解に記述することができます。

　プログラムにおける再帰は次のような構造をしています。

function rfunc(n,…) // 再帰的関数の定義

{

if (脱出条件) // 再帰からの脱出口

return;

・

・

rfunc(n-1,…); // 再帰呼び出し

}

・

・

rfunc(n,…); // 再帰関数の最初の呼び出し

　ある関数の内部で、再び自分自身を呼び出すような構造の関数を再帰的関数と呼び、関数内部で再び1次低い自分自身を呼び出すことを再帰呼び出し（リカーシブ・コール）と呼びます。

　再帰では一般に再帰からの脱出口を置かなければいけません。これがないと、再帰呼び出しが永遠に続いてしまうことになります。

2.　階乗

n!（nの階乗）は次のように定義できます。

　n!=n・(n-1)! n>0

0!=1

これは次のような意味に解釈できます。

・n!を求めるには１次低い(n-1)!を求めてそれにnを掛ける

・(n-1)!を求めるには１次低い(n-2)!を求めてそれにn-1を掛ける

　　・

　　・

・1!を求めるには0!を求め、それに1を掛ける

・0!は1である

「例題10-1-1」階乗を求める再帰関数kaijoを作り、0!～12!までを求めて表示します。

<!DOCTYPE html>

<html>

<head>

<script type="text/javascript">

function kaijo(n)

{

if (n==0)

return 1;

else

return n\*kaijo(n-1);

}

</script>

</head>

<body>

<textarea id="result" rows="20" cols="20"></textarea>

<script type="text/javascript">

var result=document.getElementById("result");

for (var n=0;n<13;n++){

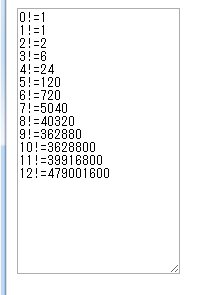
result.value+=n+"!="+kaijo(n)+"\n";

}

</script>

</body>

</html>



3.　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。」

　n枚の円盤をa⇒bに移す作業は、次のような作業に分解できます。①と③の作業が再帰的な作業となります。

①aのn-1枚の円盤をa⇒ｃに移す。

②n枚目の円盤をa⇒bに移す。

③cのn-1枚の円盤をc⇒bに移す。

「図10-1　ハノイの塔」

「例題10-1-2」n枚の円盤をa⇒bに移動する手順をalertで表示する関数hanoiを作ります。プログラムにおける①~③の意味は以下です。

①n-1枚の円盤を作業用棒bを使って棒aから棒cに移動する再帰呼び出し。

②n枚目の円盤を引数aが示す棒から引数bが示す棒に移す実際の処理。

③n-1枚の円盤を作業用棒aを使って棒cから棒bに移動する再帰呼び出し。

<!DOCTYPE html>

<html>

<head>

<script type="text/javascript">

function hanoi(n,a,b,c)

{

if (n>0) {

hanoi(n-1,a,c,b); // ①

alert(n+"番の板を"+a+"->"+b); // ②

hanoi(n-1,c,b,a); // ③

}

}

</script>

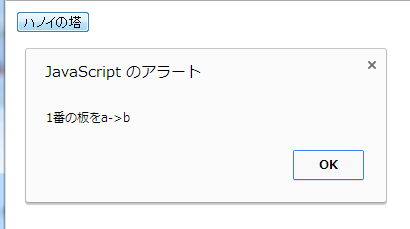
</head>

<body>

<input type="button" value="ハノイの塔" onClick="hanoi(3,'a','b','c')">

</body>

</html>



「練習問題10-1」テキストボックスから円盤の枚数を入力できるようにしなさい。表示結果をテキストエリアに表示しなさい。

<!DOCTYPE html>

<html>

<head>

<script type="text/javascript">

var result;

function hanoi(n,a,b,c)

{

if (n>0) {

hanoi(n-1,a,c,b);

result.value+=n+"番の板を"+a+"->"+b+"\n";

hanoi(n-1,c,b,a);

}

}

function init()

{

var n= ① ;

result=document.getElementById("result");

result.value="";

② ;

}

</script>

</head>

<body>

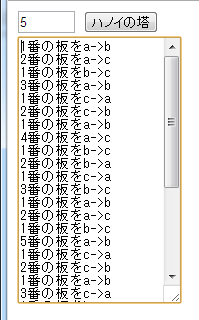
<input id="number" type="text" size="4" />

<input type="button" value="ハノイの塔" onClick="init()" /><br />

<textarea id="result" rows="20" cols="20"></textarea>

</body>

</html>



「補足」スマートフォンでは<textarea>にスクロールバーは付きません。「style="overflow:scroll"」のような属性指定をしても付きません。Mobile Chromeでは<textarea>領域でタッチムーブすれば、領域内はスクロールできますが、Mobile Safariではスクロールできません。このため、<textarea>でなく、<div>のinnerHTMLに書きだします。

<!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">

var result;

function hanoi(n,a,b,c)

{

if (n>0) {

hanoi(n-1,a,c,b);

result.innerHTML+=n+"番の板を"+a+"->"+b+"<br />";

hanoi(n-1,c,b,a);

}

}

function init()

{

var n=document.getElementById("number").value;

result=document.getElementById("result");

result.innerHTML="";

hanoi(n,"a","b","c");

}

</script>

</head>

<body>

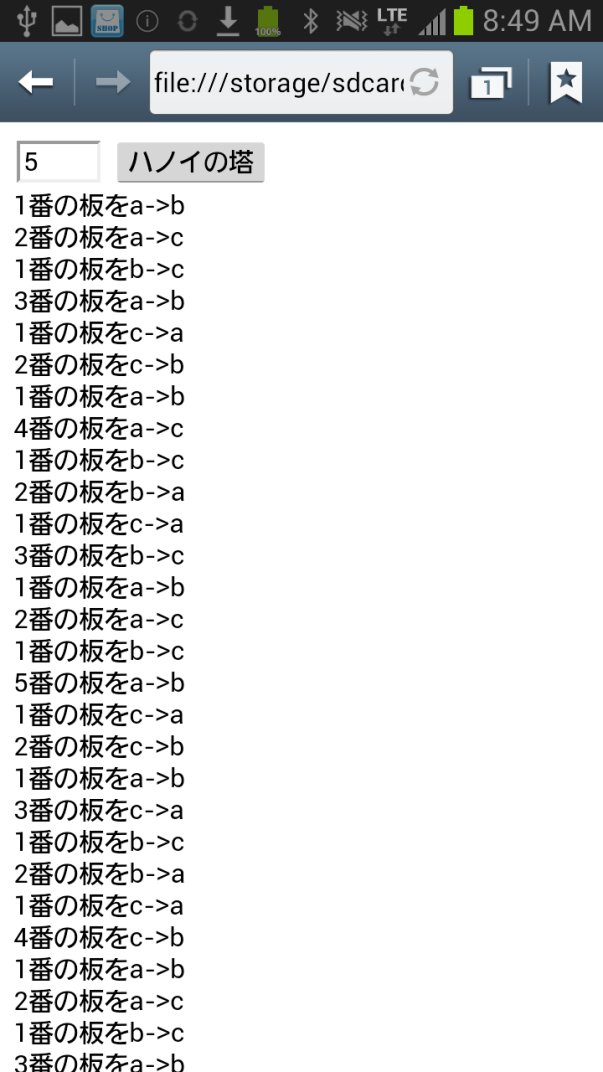
<input id="number" type="text" size="4" />

<input type="button" value="ハノイの塔" onClick="init()" /><br />

<div id="result"></div>

</body>

</html>



10-2　タートル・グラフィックス

長さと角度を与えて直線を引く関数moveと現在角を回転する関数turnを作ります。このようなmoveやturnを使って描く図形をタートル・グラフィックスと呼びます。

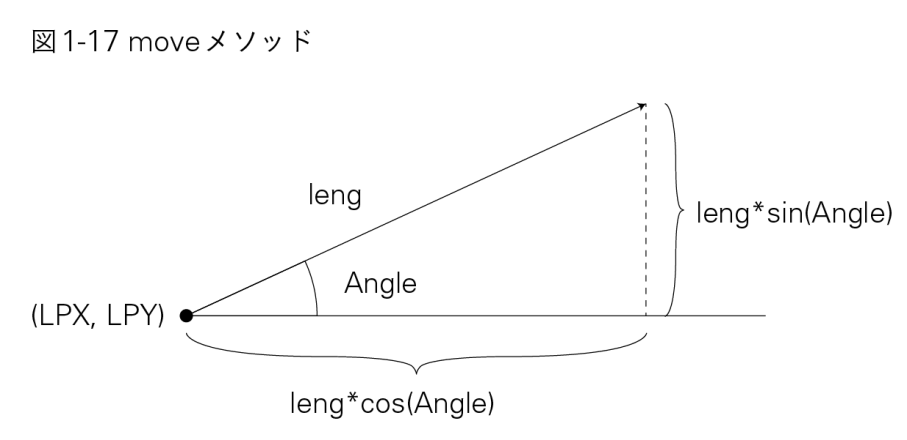
1.　move関数

　描画の現在位置を（LPX,LPY）、現在角をAngleとすると、そこから長さlengの直線を引く場合の終点座標(x,y)は以下の計算式から得られます。Angle度をラジアンに変換するために「Math.PI/180」を乗じます。

x=leng\*Math.cos(Angle\*Math.PI/180);

y=leng\*Math.sin(Angle\*Math.PI/180);

「図10-2　moveメソッド」



2.　turn関数

　現在角Angleをa°回転します。Angleの値が0~360°に入るように「Angle=Angle%360;」という式で補正します。

「例題10-2」lengに初期の長さ、angleに回転角、stepに減少する長さを与え、lengが10以上の間、move(leng)とturn(angle)を繰り返すことで渦巻き模様を描きます。angle、stepの値を変えることで異なる渦巻き模様ができます。

<!DOCTYPE html>

<html>

<head>

<script type="text/javascript">

var LPX,LPY,Angle;

function move(leng) // 指定長の直線

{

var x,y;

x=leng\*Math.cos(Angle\*Math.PI/180);

y=leng\*Math.sin(Angle\*Math.PI/180);

context.lineTo(LPX+x,LPY-y);

context.stroke();

LPX=LPX+x;LPY=LPY-y;

}

function turn(angle) // 回転

{

Angle+=angle;

Angle=Angle%360;

}

function setangle(angle) // 角度の設定

{

Angle=angle;

}

function setpoint(x,y) // 位置の設定

{

LPX=x;LPY=y;

context.moveTo(LPX,LPY);

}

</script>

</head>

<body>

<canvas id="canvas" width="600" height="600"></canvas>

<script type="text/javascript">

var canvas = document.getElementById("canvas");

if(canvas.getContext){

var context = canvas.getContext("2d");

context.strokeStyle ="blue";

}

var leng=200,angle=73,step=1;

setpoint(80,300);

setangle(0);

while (leng>10){ // 渦巻き模様

move(leng);

turn(angle);

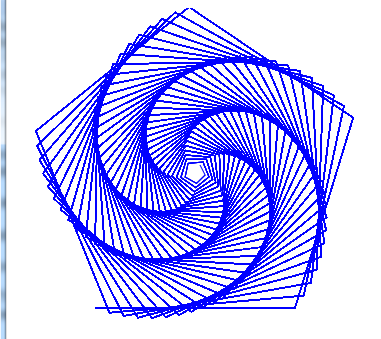
leng-=step;

}

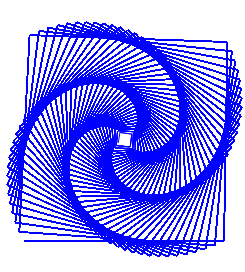
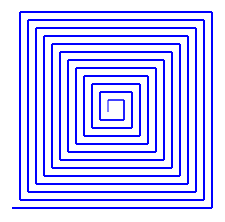
</script>

</body>

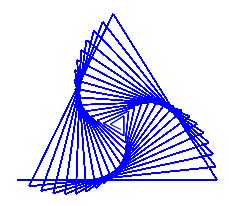
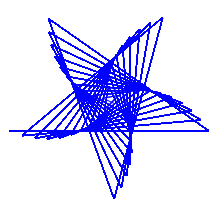
</html>



angle=73、step=1

angle=89、step=1 angle=90、step=4

angle=122、step=4 angle=145、step=4

「練習問題10-2」三角形～九角形までの正多角形を描きなさい。正n角形はn個の直線を360/n°回転しながら結んで行きます。例題9-2の<body>部の<script>ブロックを以下のように変更します。

<script type="text/javascript">

var canvas = document.getElementById("canvas");

if(canvas.getContext){

var context = canvas.getContext("2d");

context.strokeStyle ="blue";

}

for ( ① ){

setpoint(150,300);

setangle(0);

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

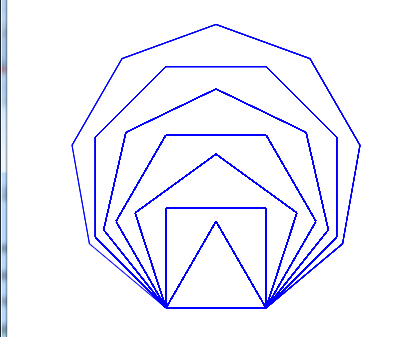
move(100);

② ;

}

}

</script>



10-3　リカーシブ・グラフィックス（再帰図形）

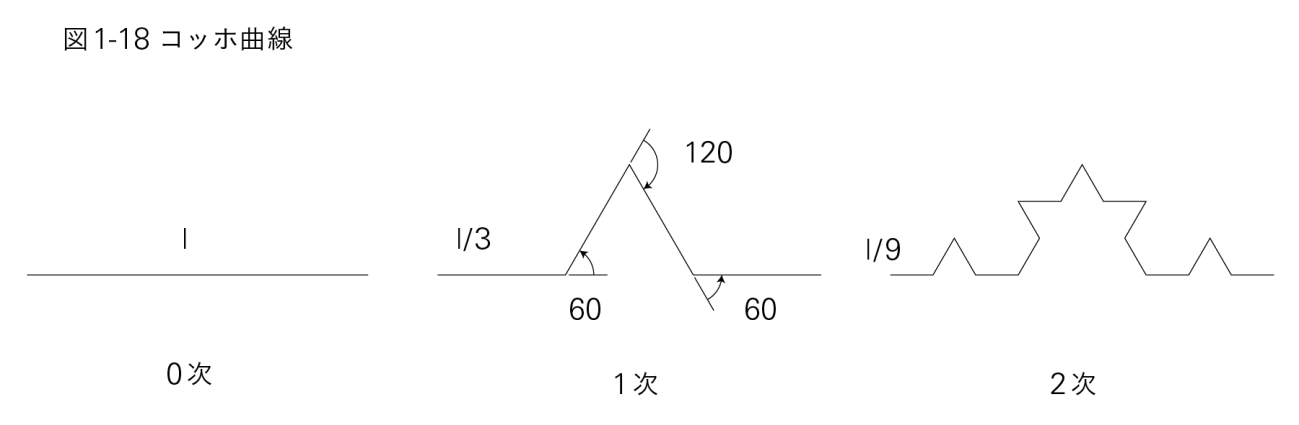
　リカーシブ・グラフィックスはグラフィックスの世界を解析的に表現せずに、再帰的に表現しようとするものです。再帰を使うと自然に近い図形（入り組んだ海岸線や樹木）がいとも簡単に表現できます。

1.　コッホ曲線

　コッホ曲線は、数学者のコッホにより発見されたものです。コッホ曲線は以下のように定義されています。

0次のコッホ曲線は長さlの直線である。1次のコッホ曲線は、1辺の長さがl/3の大きさの正三角形状のでっぱりを出す。2次のコッホ曲線は、1次のコッホ曲線の各辺（4つ）に対し、1辺の長さがl/9の大きさの正三角形状のでっぱりを出す。

「図10-3　コッホ曲線」



n次のコッホ曲線を描く手順は以下です。

①n-1次のコッホ曲線を1つ描く。

②向きを60°変えてn-1次のコッホ曲線を1つ描く。

③向きを-120°変えてn-1次のコッホ曲線を1つ描く。

④向きを60°変えてn-1次のコッホ曲線を1つ描く。

「例題10-3」長さ5の4次のコッホ曲線を描きます。0次のコッホ曲線（長さlの直線）を描くのに「10-2　タートルグラフィックス」で示したmoveとturnを使用します。

<!DOCTYPE html>

<html>

<head>

<script type="text/javascript">

var LPX,LPY,Angle;

function koch(n,leng) // n次のコッホ曲線

{

if (n==0){

move(leng);

}

else {

koch(n-1,leng);turn(60);

koch(n-1,leng);turn(-120);

koch(n-1,leng);turn(60);

koch(n-1,leng);

}

}

function move(leng)

{

var x,y;

x=leng\*Math.cos(Angle\*Math.PI/180);

y=leng\*Math.sin(Angle\*Math.PI/180);

context.lineTo(LPX+x,LPY-y);

context.stroke();

LPX=LPX+x;LPY=LPY-y;

}

function turn(angle)

{

Angle+=angle;

Angle=Angle%360;

}

function setangle(angle)

{

Angle=angle;

}

function setpoint(x,y)

{

LPX=x;LPY=y;

context.moveTo(LPX,LPY);

}

</script>

</head>

<body>

<canvas id="canvas" width="640" height="640"></canvas>

<script type="text/javascript">

var canvas = document.getElementById("canvas");

if(canvas.getContext){

var context = canvas.getContext("2d");

context.strokeStyle = "blue";

}

setpoint(50,150);

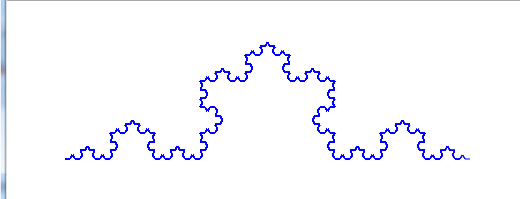
setangle(0);

koch(4,5); // 4次のコッホ曲線

</script>

</body>

</html>



「練習問題10-3-1」4次のコッホ曲線を「-120°」回転して、3個描きなさい。描かれた図形をコッホ島と呼びます。例題10-3の<body>部の<script>ブロックを以下のように変更します。

<script type="text/javascript">

var canvas = document.getElementById("canvas");

if(canvas.getContext){

var context = canvas.getContext("2d");

context.strokeStyle = "blue";

}

setpoint(50,150);

setangle(0);

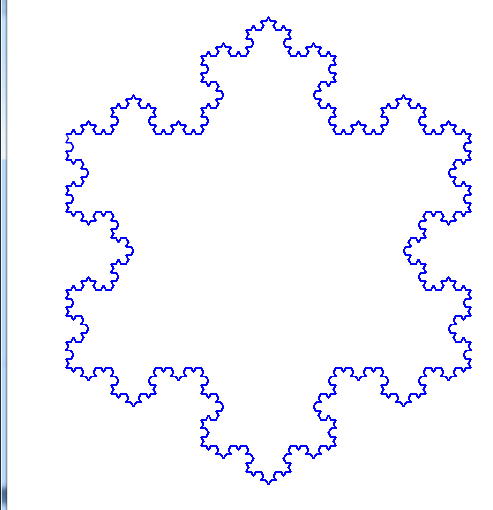
for ( ① ){ // コッホ曲線を3つ描く

koch(4,5); // 4次のコッホ曲線

② ;

}

</script>



2.　クロスステッチ

コッホ島と同じ要領でクロスステッチを描きます。コッホ曲線が正三角形を基本にしているのに対し、クロスステッチは正方形を基本にしていますが原理は同じです。n次のクロスステッチを描くにはn-1次のクロスステッチを5本描けばよく、描く方向は+90°、-90°、-90°、+90°の順に変わります。

「図10-4　クロスステッチ」

「練習問題10-3-2」4次のクロスステッチを「90°」回転して4個描きなさい。

<!DOCTYPE html>

<html>

<head>

<script type="text/javascript">

var LPX,LPY,Angle;

function stech(n,leng) // n次のステッチ

{

if (n==0){

move(leng);

}

else {

stech(n-1,leng); ① ;

stech(n-1,leng); ② ;

stech(n-1,leng);turn(90);

stech(n-1,leng);turn(-90);

stech(n-1,leng);

}

}

function move(leng)

{

var x,y;

x=leng\*Math.cos(Angle\*Math.PI/180);

y=leng\*Math.sin(Angle\*Math.PI/180);

context.lineTo(LPX+x,LPY-y);

context.stroke();

LPX=LPX+x;LPY=LPY-y;

}

function turn(angle)

{

Angle+=angle;

Angle=Angle%360;

}

function setangle(angle)

{

Angle=angle;

}

function setpoint(x,y)

{

LPX=x;LPY=y;

context.moveTo(LPX,LPY);

}

</script>

</head>

<body>

<canvas id="canvas" width="640" height="640"></canvas>

<script type="text/javascript">

var canvas = document.getElementById("canvas");

if(canvas.getContext){

var context = canvas.getContext("2d");

context.strokeStyle = "blue";

}

setpoint(200,400);

setangle(0);

for (var i=1;i<=4;i++){ // クロスステッチステッチを4つ描く

stech(4,3); // 4次のステッチ

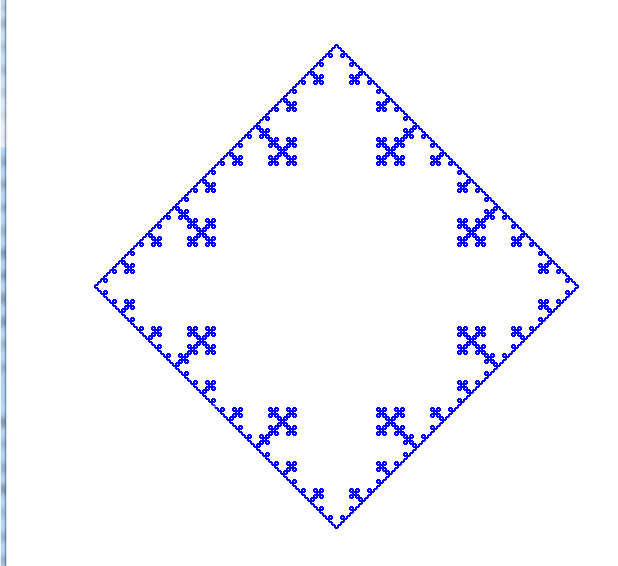
turn(90);

}

</script>

</body>

</html>



10-4　決定木

木はいくつかのノードとそれを結ぶ枝から構成されます。ノードはデータに対応し、枝はデータとデータを結ぶ親子関係に対応します。あるノードから下方に分岐する枝の先にあるノードを子と呼び、分岐元のノードを親と呼びます。

木の一番始めのノードをとくに根（ルート）と呼び、子を持たないノードを葉と呼びます。木の中のあるノードを相対的な根と考え、そこから枝分かれしている枝とノードの集合を部分木と呼びます。

　木のうちで、各ノードから出る枝が2本以下のものを特に2分木と呼び、データ構造の木の中心はこの2分木です。2分木の中で、ノードの内容の意味の持たせ方で、決定木、2分探索木などに分かれます。質問項目にyes、noで枝分かれするような木を決定木と呼びます。

「図10-5　食文化判定の決定木」

left[ ]が左のノードへのポインタ、right[ ]が右のノードへのポインタ、node[ ]が質問内容になります。ポインタは連結する配列要素の添字となります。子を持たない葉ノードのポインタ部には「-1」を置くことにします。

|  |  |  |  |
| --- | --- | --- | --- |
| 添字 | 左へのポインタ  left[ ] | 質問内容  node[ ] | 右へのポインタ  right[ ] |
| 0 | 1 | お米が好き | 2 |
| 1 | 3 | 味噌汁が好き | 4 |
| 2 | 5 | パスタが好き | 6 |
| 3 | -1 | 和食派 | -1 |
| 4 | -1 | 中華派 | -1 |
| 5 | -1 | イタリアン | -1 |
| 6 | -1 | フレンチ | -1 |

「例題10-4」図10-5の決定木をconfirmで表示されるダイアログの「OK」を「Yes」、「キャンセル」を「No」に対応させて進みます。プログラムの①~③の意味は以下です。

①葉ノードになるまで繰り返す

②質問に対しyesなら左の子へ、noなら右の子へ進む

③診断結果の表示

<!DOCTYPE html>

<html>

<head>

<script type="text/javascript">

var node=new Array(

"お米が好き",

"味噌汁が好き",

"パスタが好き",

"和食派",

"中華派",

"イタリアン",

"フレンチ");

var left=new Array(1,3,5,-1,-1,-1,-1);

var right=new Array(2,4,6,-1,-1,-1,-1);

function check()

{

var p=0;

while (left[p]!=-1){ // ①

if (confirm(node[p])) // ②

p=left[p];

else

p=right[p];

}

alert(node[p]); // ③

}

</script>

</head>

<body">

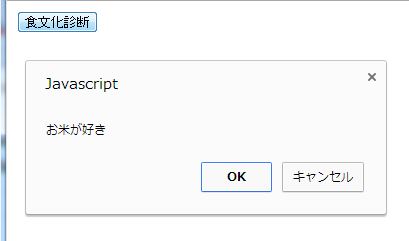
<form>

<input type="button" value="食文化診断" onClick="check()">

</form>

</body>

</html>



「練習問題10-4」図10-5の決定木は3階層でしたが、次のような性格を判定する4階層の決定木を作りなさい。check関数は同じで、node[ ]、left[ ]、right[ ]のデータが異なるだけです。ボタンのテキストを「性格診断」に変更します。

「図10-6　性格診断の決定木」

var node=new Array(

"どちらかといえば自己中心",

"人の上に立ちたい",

"皆でいろいろすることが好き",

"物事がきちんとしていないと気が済まない",

"理屈が好きである",

"リーダーシップがある",

"くよくよ考える方である",

"すべてがきちんとしていないと気が済まない完璧主義",

"おおざっぱな性格で権力志向が強い",

"物事を客観的に見る理論派",

"マイペースで自己主張できる",

"世の中のためになることは積極的に取り組む行動派",

"他人のためになることを好む世話好き",

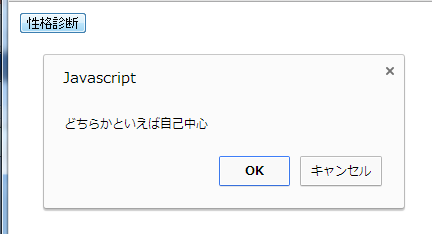
"自分に自身がなく他人に影響されやすい",

"他人まかせののんき派");

var left=new Array( ① );

var right=new Array( ② );

<input type="button" value="性格診断" onClick="check()">



11章　リバーシーゲーム

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。

1.　盤面を作る

2.　黒石を置く

3.　盤面の情報を配列に置く

4.　黒番白番で交互に置く

5.　石を置ける位置かどうかチェック

6.　自動的に反転する

7.　コンピュータが手を打つ

8.　コンピュータに戦略を持たせる

9.　完成版

　reversi1.htmlを元に例題が進むごとに前の例題に対し、追加または変更された箇所を赤色で区別します。

11-1　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.jpg 緑（石を置いていない状態）のイメージファイル。

white.jpg 白を置いた状態のイメージファイル。

black.jpg 黒を置いた状態のイメージファイル。

図11-1盤面

　　　　→x

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 9 |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  | 28 | 29 |  |  |  |
|  |  |  | 36 | 37 |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | 64 |

↓

y

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">

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();

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>

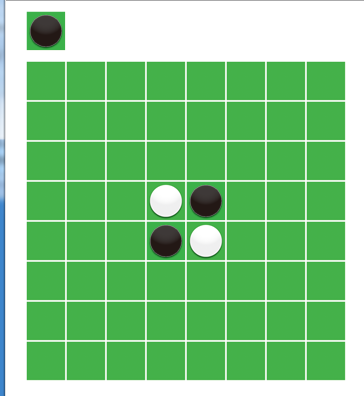
<script type="text/javascript">

init();

</script>

</body>

</html>



11-2　黒石を置く

クリックした位置に黒石を置くyx関数を作ります。各イメージには、イメージをクリックしたときに処理されるイベント処理関数を「img.addEventListener("click",yx);」のように指定します。クリックしたイメージオブジェクトは「obj=event.target;」で取得できます。

・reversi2.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">

function yx(event) // クリックイベント処理関数

{

var obj;

obj=event.target; // またはevent.srcElement

obj.src="black.png";

}

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";

img.addEventListener("click",yx);

document.body.appendChild(img);

n++;

}

}

img=new Image();

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>

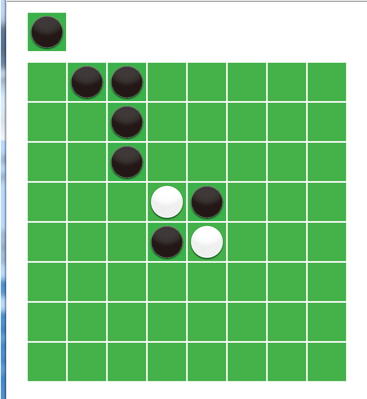
<script type="text/javascript">

init();

</script>

</body>

</html>



11-3　盤面の情報を配列に置く

2次元配列M[y][x]に盤面のy、x位置のマスの情報（0：緑、1:白、2:黒）を与えます。2次元配列M[ ][ ]を初期化する関数をarrayinitとします。この配列は各関数から共通に使用されるため、関数の外で宣言します。これによりグローバルスコープ（どこからでも使用できる）となります。盤面の初期状態は4、4と5、5位置が白、4、5と5、6位置が黒、その他は緑とします。

配列要素は8×8で良いのですが、後で作るok関数、reverse関数でマスの要素を超えた参照を行うのでそれに備えて10×10の要素とします。盤面に相当する要素は1～8で行います。盤面の外枠に相当する要素0と要素9の内容は0とします。

　クリックしたイメージのid番号（1スタート）から対応する2次元配列要素の行要素番号(y)と列要素番号(x)を求めるには以下のようにします。parseIntメソッドの代わりにMath.floorメソッドを使っても同じです。

obj=event.target;

n=obj.id;

n=n.substring(3); // 先頭のimgを削除

y= parseInt((n-1)/8)+1;

x=(n-1)%8+1;

・reversi3.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">

var M=new Array(10); // 盤面情報配列

function arrayinit() // 配列初期化関数

{

var i,j;

for (i=0;i<10;i++){

M[i]=new Array(10); // 2次元配列にする

}

for (i=0;i<10;i++){

for (j=0;j<10;j++){

M[i][j]=0; // 石無し

}

}

M[4][4]=1;M[5][5]=1; // 白石

M[4][5]=2;M[5][4]=2; // 黒石

}

function yx(event) // クリックイベント処理関数

{

var y,x,n,obj;

obj=event.target; // またはevent.srcElement

n=obj.id;

n=n.substring(3); // 先頭のimgを削除

y=parseInt((n-1)/8)+1;

x=(n-1)%8+1;

if (M[y][x]==0) {

obj.src="black.png";

M[y][x]=2;

}

else if (M[y][x]==1){

obj.src="black.png";

M[y][x]=2;

}

else if (M[y][x]==2){

obj.src="white.png";

M[y][x]=1;

}

}

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";

img.addEventListener("click",yx);

document.body.appendChild(img);

n++;

}

}

img=new Image();

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();

arrayinit();

</script>

</body>

</html>

11-4　黒番白番で交互に置く

黒番と白番を管理するために変数Flagを使いfalseなら白番、trueなら黒番とします。次の番手を画面左上の石（idがimg65）で表示します。これで黒、白対戦を手動で行えるリバーシゲームができます。

・reversi4.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">

var Flag=true; // 白黒手番フラグ

var M=new Array(10); // 盤面情報配列

function arrayinit() // 配列初期化関数

{

var i,j;

for (i=0;i<10;i++){

M[i]=new Array(10); // 2次元配列にする

}

for (i=0;i<10;i++){

for (j=0;j<10;j++){

M[i][j]=0; // 石無し

}

}

M[4][4]=1;M[5][5]=1; // 白石

M[4][5]=2;M[5][4]=2; // 黒石

}

function yx(event) // クリックイベント処理関数

{

var y,x,n,obj,ban;

obj=event.target; // またはevent.srcElement

n=obj.id;

n=n.substring(3); // 先頭のimgを削除

y=parseInt((n-1)/8)+1;

x=(n-1)%8+1;

ban=document.getElementById("img65");

if (M[y][x]==0) {

if (Flag) {

obj.src="black.png";

M[y][x]=2;

ban.src="white.png";

}

else {

obj.src="white.png";

M[y][x]=1;

ban.src="black.png";

}

Flag=!Flag;

}

else if (M[y][x]==1){

obj.src="black.png";

M[y][x]=2;

}

else if (M[y][x]==2){

obj.src="white.png";

M[y][x]=1;

}

}

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";

img.addEventListener("click",yx);

document.body.appendChild(img);

n++;

}

}

img=new Image();

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();

arrayinit();

</script>

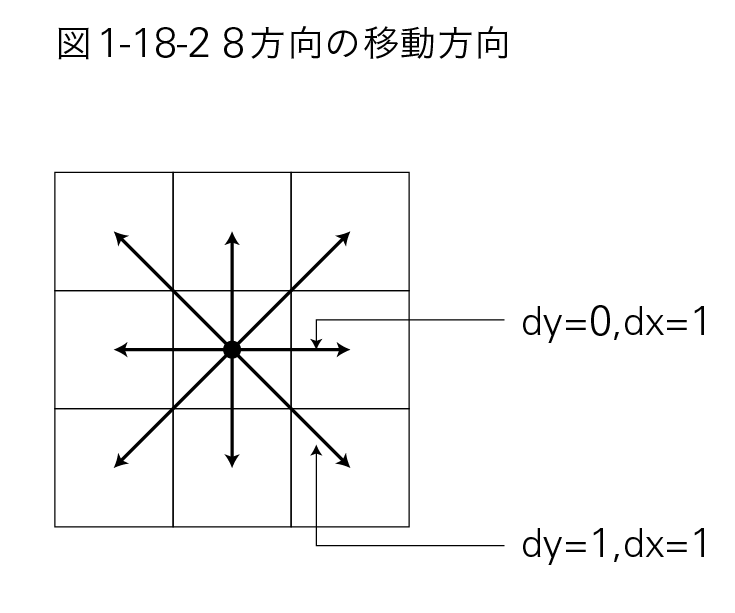
</body>

</html>

11-5　石を置ける位置かどうかチェックする

　クリックした位置に石が置ければtrue、置けなければfalseを返すok関数を作ります。クリックした位置をy、xとすると。そこを中心に8方向でチェックを行います。y、x位置からdy、dxで示す値を加えることでy、x位置を進めます。たとえばdy=0、dx=1ならマスの右方向への移動となり、dy=1、dx=1ならマスの右斜め下方向への移動となります。8方向のdy、dxの値はforの二重ループで-1、0、1と変化させます。組み合わせは9通りできますががdy=0、dx=0はどこにも進めないので、除外すると8通りとなります。

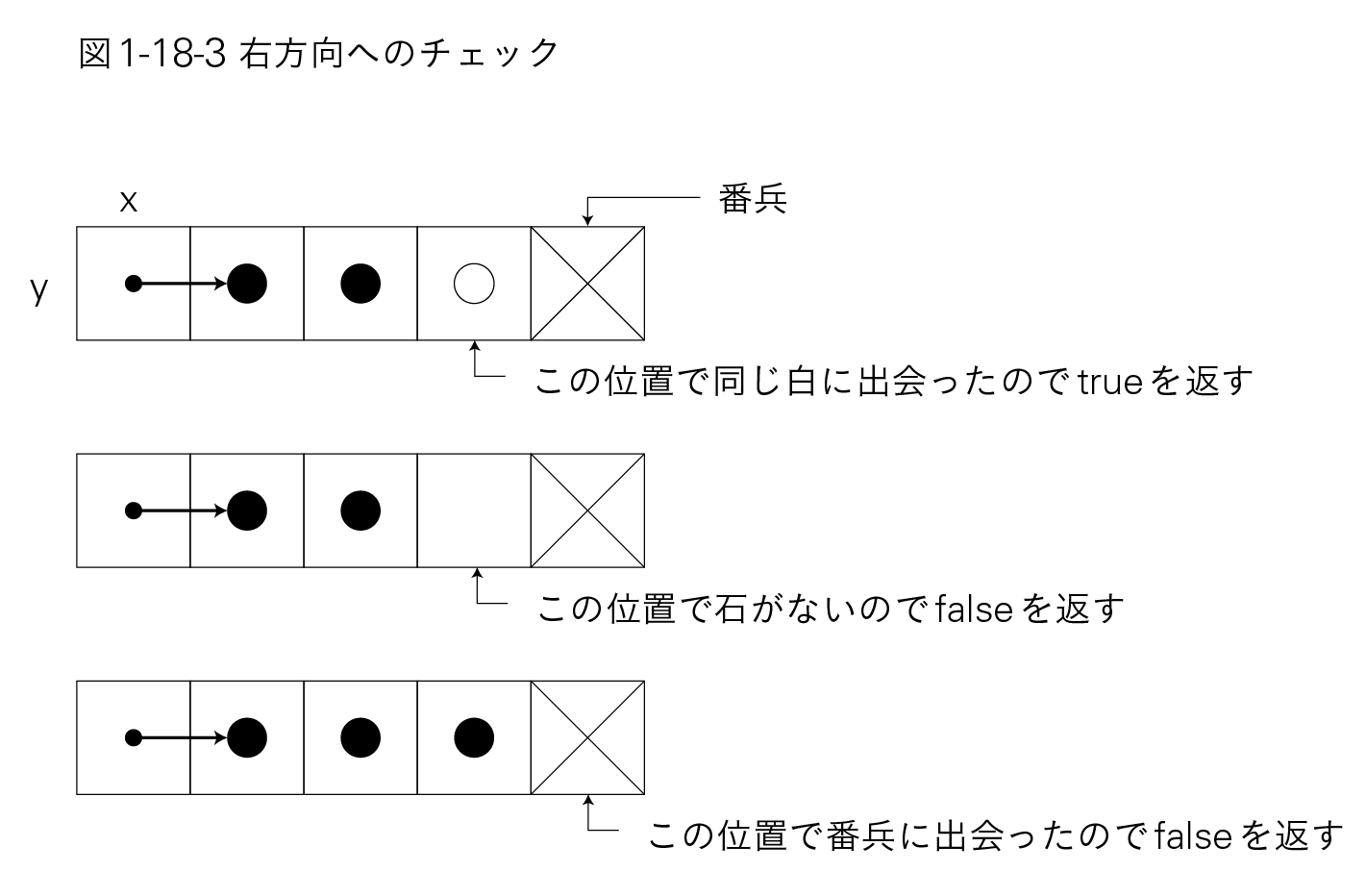
「図11-2　8方向の移動」



　y、x位置に対応する配列にその手番の石を置いたという情報を格納します。forの二重ループ内で、位置を進める操作を行うのはdy=0、dx=0でない場合でかつ進む方向の隣が同じ色でない場合です。

　位置を進めるのは端になるまでか緑になるまでです。マスを進めるときの変数としてry、rxを使います。y、x位置とry、rx位置のマスの石が同じ色ならtrueを返します。trueになる条件がなくforの2重ループを完了した場合はy、x位置に置けないということなのでfalseを返し、y、x位置に仮に置いたものを取り去ります。図11-3にy、x位置に白が置けるか右方向にチェックする場合を示しました。

「図11-3　右方向へのチェック」



・reversi5.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">

var Flag=true; // 白黒手番フラグ

var M=new Array(10); // 盤面情報配列

function ok(y,x) // 置けるか調べる関数

{

var ry,rx,dy,dx;

if (Flag)

M[y][x]=2; // 黒を置く

else

M[y][x]=1; // 白を置く

for (dx=-1;dx<=1;dx++){

for (dy=-1;dy<=1;dy++){

if (!(dx==0 && dy==0) && M[y][x]!=M[y+dy][x+dx]){

rx=x;ry=y;

while (1<=rx && rx<=8 && 1<=ry && ry<=8 && M[ry][rx]!=0 ){

rx+=dx;ry+=dy;

if (M[y][x]==M[ry][rx]){ // 同じ色の石に出会った時

return true;

}

}

}

}

}

M[y][x]=0; // 石の無い状態に戻す

return false;

}

function arrayinit() // 配列初期化関数

{

var i,j;

for (i=0;i<10;i++){

M[i]=new Array(10); // 2次元配列にする

}

for (i=0;i<10;i++){

for (j=0;j<10;j++){

M[i][j]=0; // 石無し

}

}

M[4][4]=1;M[5][5]=1; // 白石

M[4][5]=2;M[5][4]=2; // 黒石

}

function yx(event) // クリックイベント処理関数

{

var y,x,n,obj,ban;

obj=event.target; // またはevent.srcElement

n=obj.id;

n=n.substring(3); // 先頭のimgを削除

y=parseInt((n-1)/8)+1;

x=(n-1)%8+1;

ban=document.getElementById("img65");

if (M[y][x]==0 && ok(y,x)) {

if (Flag) {

obj.src="black.png";

M[y][x]=2;

ban.src="white.png";

}

else {

obj.src="white.png";

M[y][x]=1;

ban.src="black.png";

}

Flag=!Flag;

}

else if (M[y][x]==1){

obj.src="black.png";

M[y][x]=2;

}

else if (M[y][x]==2){

obj.src="white.png";

M[y][x]=1;

}

}

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";

img.addEventListener("click",yx);

document.body.appendChild(img);

n++;

}

}

img=new Image();

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();

arrayinit();

</script>

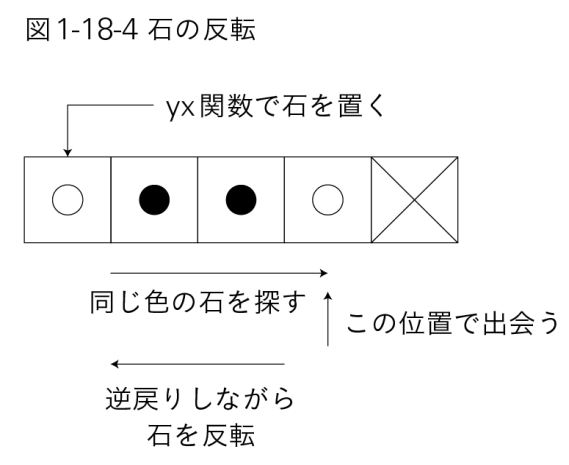
</body>

</html>

11-6　自動的に反転する

　はさんだ石を自動的に反転するreverse関数を作ります。先に作成したok関数と同様な方法でy、x位置を中心に8方向を検査します。y、x位置に置いた石と同じ色の石が見つかったら、そこから逆戻りしながら石を反転する処理を追加します。

「図11-4　石の反転」



reverse関数はyx関数から呼び出します。自動反転するので、すでに石が置いてある位置をクリックしたときに逆の色にするという処理(yx関数のelse if部)は削除します。

・reversi6.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">

var Flag=true; // 白黒手番フラグ

var M=new Array(10); // 盤面情報配列

function reverse(y,x) // 自動反転関数

{

var ry,rx,dy,dx,n,obj;

for (dx=-1;dx<=1;dx++){

for (dy=-1;dy<=1;dy++){

if (!(dx==0 && dy==0) && M[y][x]!=M[y+dy][x+dx]){

ry=y;rx=x;

while (1<=rx && rx<=8 && 1<=ry && ry<=8 && M[ry][rx]!=0){

ry+=dy;rx+=dx;

if (M[y][x]==M[ry][rx]){ // 同じ色の石

ry-=dy;rx-=dx; // 1つ戻す

while (!(x==rx && y==ry)){

n=(ry-1)\*8+(rx-1)+1;

obj=document.getElementById("img"+n);

if (M[ry][rx]==1){ // 白なら黒に

obj.src="black.png";

M[ry][rx]=2;

}

else { // 黒なら白に

obj.src="white.png";

M[ry][rx]=1;

}

ry-=dy;rx-=dx; // 逆戻り

}

break;

}

}

}

}

}

}

function ok(y,x) // 置けるか調べる関数

{

var ry,rx,dy,dx;

if (Flag)

M[y][x]=2; // 黒を置く

else

M[y][x]=1; // 白を置く

for (dx=-1;dx<=1;dx++){

for (dy=-1;dy<=1;dy++){

if (!(dx==0 && dy==0) && M[y][x]!=M[y+dy][x+dx]){

rx=x;ry=y;

while (1<=rx && rx<=8 && 1<=ry && ry<=8 && M[ry][rx]!=0 ){

rx+=dx;ry+=dy;

if (M[y][x]==M[ry][rx]){ // 同じ色の石に出会った時

return true;

}

}

}

}

}

M[y][x]=0; // 石の無い状態に戻す

return false;

}

function arrayinit() // 配列初期化関数

{

var i,j;

for (i=0;i<10;i++){

M[i]=new Array(10); // 2次元配列にする

}

for (i=0;i<10;i++){

for (j=0;j<10;j++){

M[i][j]=0; // 石無し

}

}

M[4][4]=1;M[5][5]=1; // 白石

M[4][5]=2;M[5][4]=2; // 黒石

}

function yx(event) // クリックイベント処理関数

{

var y,x,n,obj,ban;

obj=event.target; // またはevent.srcElement

n=obj.id;

n=n.substring(3); // 先頭のimgを削除

y=parseInt((n-1)/8)+1;

x=(n-1)%8+1;

ban=document.getElementById("img65");

if (M[y][x]==0 && ok(y,x)) {

if (Flag) {

obj.src="black.png";

M[y][x]=2;

ban.src="white.png";

}

else {

obj.src="white.png";

M[y][x]=1;

ban.src="black.png";

}

reverse(y,x);

Flag=!Flag;

}

}

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";

img.addEventListener("click",yx);

document.body.appendChild(img);

n++;

}

}

img=new Image();

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();

arrayinit();

</script>

</body>

</html>

11-7　コンピュータが手を打つ

　ここではコンピュータは何も考えずにランダムに白石を置くconputer関数を作ります。

　コンピュータは1～8の乱数を2組発生し、その値をy、x位置とし、ok(y,x)でそこに石を置けるか判定します。500回試して、手がみつからなければ、手を相手に渡します。

コンピュータの手を3秒後に打つにはsetTimeoutメソッドを使って以下のようにします。

setTimeout("computer()",3000);

・reversi7.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">

var Flag=true; // 白黒手番フラグ

var M=new Array(10); // 盤面情報配列

function computer() // コンピュータの手関数

{

var i,y,x,n,obj,ban;

ban=document.getElementById("img65");

ban.src="black.png";

for (i=0;i<500;i++){

y=parseInt(Math.random()\*8)+1;

x=parseInt(Math.random()\*8)+1;

n=(y-1)\*8+(x-1)+1;

obj=document.getElementById("img"+n);

if (M[y][x]==0 && ok(y,x)){

obj.src="white.png";

M[y][x]=1;

reverse(y,x);

Flag=!Flag;

return;

}

}

Flag=!Flag;

}

function reverse(y,x) // 自動反転関数

{

var ry,rx,dy,dx,n,obj;

for (dx=-1;dx<=1;dx++){

for (dy=-1;dy<=1;dy++){

if (!(dx==0 && dy==0) && M[y][x]!=M[y+dy][x+dx]){

ry=y;rx=x;

while (1<=rx && rx<=8 && 1<=ry && ry<=8 && M[ry][rx]!=0){

ry+=dy;rx+=dx;

if (M[y][x]==M[ry][rx]){ // 同じ色の石

ry-=dy;rx-=dx; // 1つ戻す

while (!(x==rx && y==ry)){

n=(ry-1)\*8+(rx-1)+1;

obj=document.getElementById("img"+n);

if (M[ry][rx]==1){ // 白なら黒に

obj.src="black.png";

M[ry][rx]=2;

}

else { // 黒なら白に

obj.src="white.png";

M[ry][rx]=1;

}

ry-=dy;rx-=dx; // 逆戻り

}

break;

}

}

}

}

}

}

function ok(y,x) // 置けるか調べる関数

{

var ry,rx,dy,dx;

if (Flag)

M[y][x]=2; // 黒を置く

else

M[y][x]=1; // 白を置く

for (dx=-1;dx<=1;dx++){

for (dy=-1;dy<=1;dy++){

if (!(dx==0 && dy==0) && M[y][x]!=M[y+dy][x+dx]){

rx=x;ry=y;

while (1<=rx && rx<=8 && 1<=ry && ry<=8 && M[ry][rx]!=0 ){

rx+=dx;ry+=dy;

if (M[y][x]==M[ry][rx]){ // 同じ色の石に出会った時

return true;

}

}

}

}

}

M[y][x]=0; // 石の無い状態に戻す

return false;

}

function arrayinit() // 配列初期化関数

{

var i,j;

for (i=0;i<10;i++){

M[i]=new Array(10); // 2次元配列にする

}

for (i=0;i<10;i++){

for (j=0;j<10;j++){

M[i][j]=0; // 石無し

}

}

M[4][4]=1;M[5][5]=1; // 白石

M[4][5]=2;M[5][4]=2; // 黒石

}

function yx(event) // クリックイベント処理関数

{

var y,x,n,obj,ban;

obj=event.target; // またはevent.srcElement

n=obj.id;

n=n.substring(3); // 先頭のimgを削除

y=parseInt((n-1)/8)+1;

x=(n-1)%8+1;

ban=document.getElementById("img65");

if (M[y][x]==0 && ok(y,x)) {

obj.src="black.png";

M[y][x]=2;

reverse(y,x);

Flag=!Flag;

ban.src="white.png";

setTimeout("computer()",3000); // コンピュータの手

}

}

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";

img.addEventListener("click",yx);

document.body.appendChild(img);

n++;

}

}

img=new Image();

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();

arrayinit();

</script>

</body>

</html>

11-8　コンピュータに戦略を持たせる

左上隅から始めそこに白石（コンピュータの手）が置けたら、黒石をひっくり返せる枚数を求めるcheck関数を作ります。reverse関数を参考にします。一番ひっくり返す枚数が多い位置をコンピュータの手にします。

・reversi8.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">

var Flag=true; // 白黒手番フラグ

var M=new Array(10); // 盤面情報配列

function check(y,x) // y,x位置に置いた時にひっくり返せる数を求める関数

{

var dy,dx,ry,rx,n=0;

for (dx=-1;dx<=1;dx++){

for (dy=-1;dy<=1;dy++){

if (!(dx==0 && dy==0) && M[y][x]!=M[y+dy][x+dx]){

ry=y;rx=x;

while (1<=rx && rx<=8 && 1<=ry && ry<=8 && M[ry][rx]!=0){

ry+=dy;rx+=dx;

if (M[y][x]==M[ry][rx]){ // 同じ色の石

ry-=dy;rx-=dx; // 1つ戻す

while (!(x==rx && y==ry)){

n++;

ry-=dy;rx-=dx; // 逆戻り

}

break;

}

}

}

}

}

return n;

}

function computer() // コンピュータの手関数

{

var i,y,x,yy,xx,n,obj,ban,max=0;

for (y=1;y<=8;y++){

for (x=1;x<=8;x++){

if (M[y][x]==0 && ok(y,x)){

M[y][x]=1; // 仮に置く

n=check(y,x);

if (n>max){

max=n;

yy=y;xx=x;

}

M[y][x]=0; // 戻す

}

}

}

if (max!=0){

n=(yy-1)\*8+(xx-1)+1;

obj=document.getElementById("img"+n);

obj.src="white.png";

M[yy][xx]=1;

reverse(yy,xx);

}

ban=document.getElementById("img65");

ban.src="black.png";

Flag=!Flag;

}

function reverse(y,x) // 自動反転関数

{

var ry,rx,dy,dx,n,obj;

for (dx=-1;dx<=1;dx++){

for (dy=-1;dy<=1;dy++){

if (!(dx==0 && dy==0) && M[y][x]!=M[y+dy][x+dx]){

ry=y;rx=x;

while (1<=rx && rx<=8 && 1<=ry && ry<=8 && M[ry][rx]!=0){

ry+=dy;rx+=dx;

if (M[y][x]==M[ry][rx]){ // 同じ色の石

ry-=dy;rx-=dx; // 1つ戻す

while (!(x==rx && y==ry)){

n=(ry-1)\*8+(rx-1)+1;

obj=document.getElementById("img"+n);

if (M[ry][rx]==1){ // 白なら黒に

obj.src="black.png";

M[ry][rx]=2;

}

else { // 黒なら白に

obj.src="white.png";

M[ry][rx]=1;

}

ry-=dy;rx-=dx; // 逆戻り

}

break;

}

}

}

}

}

}

function ok(y,x) // 置けるか調べる関数

{

var ry,rx,dy,dx;

if (Flag)

M[y][x]=2; // 黒を置く

else

M[y][x]=1; // 白を置く

for (dx=-1;dx<=1;dx++){

for (dy=-1;dy<=1;dy++){

if (!(dx==0 && dy==0) && M[y][x]!=M[y+dy][x+dx]){

rx=x;ry=y;

while (1<=rx && rx<=8 && 1<=ry && ry<=8 && M[ry][rx]!=0 ){

rx+=dx;ry+=dy;

if (M[y][x]==M[ry][rx]){ // 同じ色の石に出会った時

return true;

}

}

}

}

}

M[y][x]=0; // 石の無い状態に戻す

return false;

}

function arrayinit() // 配列初期化関数

{

var i,j;

for (i=0;i<10;i++){

M[i]=new Array(10); // 2次元配列にする

}

for (i=0;i<10;i++){

for (j=0;j<10;j++){

M[i][j]=0; // 石無し

}

}

M[4][4]=1;M[5][5]=1; // 白石

M[4][5]=2;M[5][4]=2; // 黒石

}

function yx(event) // クリックイベント処理関数

{

var y,x,n,obj,ban;

obj=event.target; // またはevent.srcElement

n=obj.id;

n=n.substring(3); // 先頭のimgを削除

y=parseInt((n-1)/8)+1;

x=(n-1)%8+1;

ban=document.getElementById("img65");

if (M[y][x]==0 && ok(y,x)) {

obj.src="black.png";

M[y][x]=2;

reverse(y,x);

Flag=!Flag;

ban.src="white.png";

setTimeout("computer()",3000); // コンピュータの手

}

}

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";

img.addEventListener("click",yx);

document.body.appendChild(img);

n++;

}

}

img=new Image();

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();

arrayinit();

</script>

</body>

</html>

「参考」４隅におければそれを優先するようにcomputer関数を作り変えなさい。

・reversi9.html

function computer() // コンピュータの手関数

{

var i,y,x,yy,xx,n,obj,ban,max=0;

for (y=1;y<=8;y++){

for (x=1;x<=8;x++){

if (M[y][x]==0 && ok(y,x)){

M[y][x]=1; // 仮に置く

n=check(y,x);

if (n>max){

max=n;

yy=y;xx=x;

}

M[y][x]=0; // 戻す

}

}

}

if (M[1][1]==0 && ok(1,1)){

yy=1;xx=1;

}

if (M[8][1]==0 && ok(8,1)){

yy=8;xx=1;

}

if (M[1][8]==0 && ok(1,8)){

yy=1;xx=8;

}

if (M[8][8]==0 && ok(8,8)){

yy=8;xx=8;

}

if (max!=0){

n=(yy-1)\*8+(xx-1)+1;

obj=document.getElementById("img"+n);

obj.src="white.png";

M[yy][xx]=1;

reverse(yy,xx);

}

ban=document.getElementById("img65");

ban.src="black.png";

Flag=!Flag;

}

☆応用サンプル11-1　リバーシーゲームの完成版

クライアントが黒、コンピュータが白です。黒番先手で開始します。黒石を置く位置をクリックします。3秒後にコンピュータが手を考えて白石を置きます。挟まれた石は自動反転されます。画面左上隅に次の番手の色の石が表示されています。黒番で打つ場所がないときは「黒番パス」ボタンをクリックして白番に手を渡します。白番で打つ場所がないときはアラートで表示し黒番に手を渡します。現在の白石と黒石の数をテキストボックスに表示します。「再戦」ボタンでもう一度最初から開始できます。

・reversi10.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">

var result;

var timeId;

var Flag=true; // 白黒手番フラグ

var M=new Array(10); // 盤面情報配列

function restart()

{

clearTimeout(timeId);

arrayinit();

for (var i=1;i<=8;i++){

for (var j=1;j<=8;j++){

var n=(i-1)\*8+(j-1)+1;

var obj=document.getElementById("img"+n);

if (M[i][j]==0)

obj.src="green.png";

else if(M[i][j]==1)

obj.src="white.png";

else if(M[i][j]==2)

obj.src="black.png";

}

}

Flag=true;

var ban=document.getElementById("img65");

ban.src="black.png";

score();

}

function pass()

{

Flag=!Flag;

var ban=document.getElementById("img65");

ban.src="white.png";

timeId=setTimeout("computer()",3000); // コンピュータの手

}

function score()

{

var a=0,b=0;

for (var i=1;i<=8;i++){

for (var j=1;j<=8;j++){

if (M[i][j]==1)

a++;

else if(M[i][j]==2)

b++;

}

}

result.value="　白："+a+"、黒："+b;

}

function check(y,x) // y,x位置に置いた時にひっくり返せる数を求める関数

{

var dy,dx,ry,rx,n=0;

for (dx=-1;dx<=1;dx++){

for (dy=-1;dy<=1;dy++){

if (!(dx==0 && dy==0) && M[y][x]!=M[y+dy][x+dx]){

ry=y;rx=x;

while (1<=rx && rx<=8 && 1<=ry && ry<=8 && M[ry][rx]!=0){

ry+=dy;rx+=dx;

if (M[y][x]==M[ry][rx]){ // 同じ色の石

ry-=dy;rx-=dx; // 1つ戻す

while (!(x==rx && y==ry)){

n++;

ry-=dy;rx-=dx; // 逆戻り

}

break;

}

}

}

}

}

return n;

}

function computer() // コンピュータの手関数

{

var i,y,x,yy,xx,n,obj,ban,max=0;

for (y=1;y<=8;y++){

for (x=1;x<=8;x++){

if (M[y][x]==0 && ok(y,x)){

M[y][x]=1; // 仮に置く

n=check(y,x);

if (n>max){

max=n;

yy=y;xx=x;

}

M[y][x]=0; // 戻す

}

}

}

if (M[1][1]==0 && ok(1,1)){

yy=1;xx=1;

}

if (M[8][1]==0 && ok(8,1)){

yy=8;xx=1;

}

if (M[1][8]==0 && ok(1,8)){

yy=1;xx=8;

}

if (M[8][8]==0 && ok(8,8)){

yy=8;xx=8;

}

if (max!=0){

n=(yy-1)\*8+(xx-1)+1;

obj=document.getElementById("img"+n);

obj.src="white.png";

M[yy][xx]=1;

reverse(yy,xx);

}

else

alert("白番パス");

ban=document.getElementById("img65");

ban.src="black.png";

Flag=!Flag;

score();

}

function reverse(y,x) // 自動反転関数

{

var ry,rx,dy,dx,n,obj;

for (dx=-1;dx<=1;dx++){

for (dy=-1;dy<=1;dy++){

if (!(dx==0 && dy==0) && M[y][x]!=M[y+dy][x+dx]){

ry=y;rx=x;

while (1<=rx && rx<=8 && 1<=ry && ry<=8 && M[ry][rx]!=0){

ry+=dy;rx+=dx;

if (M[y][x]==M[ry][rx]){ // 同じ色の石

ry-=dy;rx-=dx; // 1つ戻す

while (!(x==rx && y==ry)){

n=(ry-1)\*8+(rx-1)+1;

obj=document.getElementById("img"+n);

if (M[ry][rx]==1){ // 白なら黒に

obj.src="black.png";

M[ry][rx]=2;

}

else { // 黒なら白に

obj.src="white.png";

M[ry][rx]=1;

}

ry-=dy;rx-=dx; // 逆戻り

}

break;

}

}

}

}

}

}

function ok(y,x) // 置けるか調べる関数

{

var ry,rx,dy,dx;

if (Flag)

M[y][x]=2; // 黒を置く

else

M[y][x]=1; // 白を置く

for (dx=-1;dx<=1;dx++){

for (dy=-1;dy<=1;dy++){

if (!(dx==0 && dy==0) && M[y][x]!=M[y+dy][x+dx]){

rx=x;ry=y;

while (1<=rx && rx<=8 && 1<=ry && ry<=8 && M[ry][rx]!=0 ){

rx+=dx;ry+=dy;

if (M[y][x]==M[ry][rx]){ // 同じ色の石に出会った時

return true;

}

}

}

}

}

M[y][x]=0; // 石の無い状態に戻す

return false;

}

function arrayinit() // 配列初期化関数

{

var i,j;

for (i=0;i<10;i++){

M[i]=new Array(10); // 2次元配列にする

}

for (i=0;i<10;i++){

for (j=0;j<10;j++){

M[i][j]=0; // 石無し

}

}

M[4][4]=1;M[5][5]=1; // 白石

M[4][5]=2;M[5][4]=2; // 黒石

}

function yx(event) // クリックイベント処理関数

{

var y,x,n,obj,ban;

obj=event.target; // またはevent.srcElement

n=obj.id;

n=n.substring(3); // 先頭のimgを削除

y=parseInt((n-1)/8)+1;

x=(n-1)%8+1;

ban=document.getElementById("img65");

if (M[y][x]==0 && ok(y,x)) {

obj.src="black.png";

M[y][x]=2;

reverse(y,x);

Flag=!Flag;

ban.src="white.png";

score();

timeId=setTimeout("computer()",3000); // コンピュータの手

}

}

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";

img.addEventListener("click",yx);

document.body.appendChild(img);

n++;

}

}

img=new Image();

img.src="black.png";

img.id="img65";

img.style.position="absolute";

img.style.left="20px";

img.style.top="10px";

document.body.appendChild(img);

var btn1=document.createElement("input");

btn1.type="button";

btn1.value="黒番パス";

btn1.style.position="absolute";

btn1.style.left="60px";

btn1.style.top="20px";

btn1.addEventListener("click",pass);

document.body.appendChild(btn1);

var btn2=document.createElement("input");

btn2.type="button";

btn2.value="再戦";

btn2.style.position="absolute";

btn2.style.left="130px";

btn2.style.top="20px";

btn2.addEventListener("click",restart);

document.body.appendChild(btn2);

result=document.createElement("input");

result.type="text";

result.size=16;

result.style.position="absolute";

result.style.left="180px";

result.style.top="20px";

document.body.appendChild(result);

}

</script>

</head>

<body>

<script type="text/javascript">

init();

arrayinit();

score();

</script>

</body>

</html>

