# 応用情報処理 No. 8

## JavaScriptのswitch文

一つの変数の値が、値1、値2、値3、・・・、値Nというように、複数の値のどれかに一致したときに、条件分岐する場合は、if～else文にif～else文を繋いで行って（elseの後の{}を省略して書くことができ、if～else if 文と呼ばれる）、条件1：変数==値1、条件２：変数==値２、・・・、条件N：変数==値Nとすれば良い。しかし、Nが大きくなると、見通しが悪くなる。この場合は、switch文で書き換えた方が見通しは良くなる。

変数==値1

JavaScript文1;

 ．．．．．．

JavaScript文n;

真

偽

JavaScript文n+1;

 ．．．．．．

JavaScript文n+m;

変数== 値2

変数==値3

JavaScript文n+m+1;

 ．．．．．．

JavaScript文n+m+k;

真

真

偽

偽

JavaScript文n+m+k+1;

 ．．．．．．

JavaScript文n+m+k+h;

switch( 変数 ){

case 値1 :

JavaScript文1;

JavaScript文2;

 …

JavaScript文n;

break;

case 値2 :

JavaScript文n+1;

JavaScript文n+2;

 …

JavaScript文n+m;

break;

case 値3 :

JavaScript文n+m+1;

JavaScript文n+m+2;

 …

JavaScript文n+m+k;

break;

default:

JavaScript文n+m+k+1;

JavaScript文n+m+k+2;

 …

JavaScript文n+m+k+h;

}

switch文についての注意は以下の通りである：

* switch文は丸括弧()で囲まれた変数を引数とする。
* switch文の引数の変数は、case句の値に一致するかどうかで、条件分岐となる。
* case句の書き方は、caseの後に値を書き、その後にコロン（:）を書く。
* case句に続く処理は、break;で終了する。break;を書き忘れると、そのcase句より後にかかれている全てのcase句に続く処理を実行することになる。
* default句の後には、その前に書かれているどのcase句の値にも一致しない場合の処理を書く。

## 1.1. switch文の例

[switch01.html]

<html>

<head>

<meta http-equiv="content-type" content="text/html;charset=Shift\_JIS">

<title>switch文</title>

</head>

<body>

<h2>何曜日？</h2>

<script type="text/javascript">

var year = 1940;

var month = 10;

var day = 9;

document.write('<p>' + year + '年' + month + '月' + day + '日は、');

// Zellerの公式

if( month == 1 || month == 2){

 year = year - 1;

 month = month + 12;

}

var wday = ( year + Math.floor(year / 4) - Math.floor( year / 100) + Math.floor(year / 400) + Math.floor((13 \* month + 8)/5) + day ) % 7;

var youbi;

switch( wday ){

 case 0:

 youbi = '日';

 break;

 case 1:

 youbi = '月';

 break;

 case 2:

 youbi = '火';

 break;

 case 3:

 youbi = '水';

 break;

 case 4:

 youbi = '木';

 break;

 case 5:

 youbi = '金';

 break;

 case 6:

 youbi = '土';

 break;

 default:

 youbi = '×';

}

document.write( youbi + '曜日です。</p>');

</script>

</body>

</html>

## 1.2．Do!　 問題

if.html（プリントNo. 7の課題07）の複製をswitch.htmlとし、それのyoubiの値を求める部分（if～else～文）を、（switch01.htmlのソースコードを参考にして、）switch文を使って書き直しなさい。

# 反復（／繰り返し／ループ処理）

JavaScriptでは、ループ処理には、for文、while文、do/while文がある

## 2-1. JavaScriptのfor文

ある変数（カウンタ変数）の値が、繰り返しの条件を満たす間は、繰り返すという制御である。

for( カウンタ変数の初期値; 繰り返しの条件; カウンタ変数の増加 ) {

 JavaScript文1;

JavaScript文1;

JavaScript文2;

・・・

JavaScript文n;

条件

真

偽

 JavaScript文2;

 ・・・

 JavaScript文n;

}

for文の丸括弧()の中の三つの項目はセミコロン（;）で区切ることに注意しよう。これらの3番目の項目の「カウンタ変数の増加」の部分は、何らかの処理で置き換えることもできる。例えば--演算子（デクリメント演算子）を使って、カウンタ変数の増加ではなく、減少する処理にすることも可能である。この場合、カウンタ変数は初期値から、繰り返しを実行するたびに小さくなっていく。

## Do! 以下のソースコードの内容のHTMLファイルをfor.htmlという名前で作成しなさい。

[for.html]

<html>

<head>

<meta http-equiv="content-type" content="text/html;charset=Shift\_JIS">

<title>for文</title>

</head>

<body>

<h2>7不思議</h2>

<script type="text/javascript">

var a;

alert('割り算を実行します');

for(var i = 1; i <10; i++){

 a = 1 / i;

 document.write('<h3> 1 / ' + i + ' = ' + a + '</h3>');

}

</script>

</body>

</html>

## 2-2. JavaScriptのwhile文

for文はカウンタ変数が必要であったが、カウンタ変数がないような繰り返し条件で、ループ処理をするときに使うのがwhile文である。もちろん、カウンタ変数を使ったループ処理を、while文で書くこともできる。

JavaScript文1;

JavaScript文2;

・・・

JavaScript文n;

条件

真

偽

while( 繰り返しの条件) {

 JavaScript文1;

 JavaScript文2;

 ・・・

 JavaScript文n;

}

## Do! for.htmlをコピーしてwhile.htmlという名前に変更し、for文の代わりにwhile文で同じ動作をするように、ソースコードを書き換えなさい。

Hint: 下の空欄　　　　にはどのようなJavaScript文を書けばよいか？この空欄の部分には、for文の丸括弧内の3番目の項目の処理が必要である。

[while.html]

<html>

<head>

<meta http-equiv="content-type" content="text/html;charset=Shift\_JIS">

<title>while文</title>

</head>

<body>

<h2>7不思議</h2>

<script type="text/javascript">

var a;

for文の丸括弧内の最初の項目の情報

alert('割り算を実行します');

for文の丸括弧内の2番目の項目の情報

var i = 1;

while( i < 10 ){

 a = 1 / i;

 document.write('<h3> 1 / ' + i + ' = ' + a + '</h3>');

 　　　　 ;

}

</script>

</body>

</html>

## 2-3. while文：カウンタ変数 i を使わない例

[omikuji.html]

<html>

<head>

<meta http-equiv="content-type" content="text/html;charset=Shift\_JIS">

<title>while文</title>

</head>

<body>

<h2>必勝おみくじ</h2>

<script type="text/javascript">

var kuji; // 0～5のどれか

alert('おみくじを引きます');

var answer = true;

while( answer ){

 kuji = Math.floor( Math.random() \* 6 );

 switch( kuji ){

 case 0:

 document.write('<h3>大吉</h3>');

 break;

 case 1:

 document.write('<h3>中吉</h3>');

 break;

 case 2:

 document.write('<h3>小吉</h3>');

 break;

 case 3:

 document.write('<h3>吉</h3>');

 break;

 case 4:

 document.write('<h3>末吉</h3>');

 break;

 case 5:

 document.write('<h3>凶</h3>');

 break;

 default:

 document.write('<h3 style="color: #F00;">エラーが発生しました！</h3>');

 }

 answer = confirm('もう一度、おみくじを引きますか？');

}

</script>

</body>

</html>

【解説】

 kuji = Math.floor( Math.random() \* 6 );

Math.random()は0以上1未満の小数の乱数を返す関数である。Math.floor()関数は、小数点以下を切り捨てて、実数を整数にする関数である[[1]](#footnote-1)。 Math.random() \* 6は、0以上6未満の実数の乱数なので、kujiには、0以上6未満の整数、すなわち、0, 1, 2, 3, 4, 5のどれかがランダムに格納される。0に大吉、1に中吉、2に小吉、3に吉、4に末吉、5に凶を対応させることにして、このkujiの値に対して、switch文で、大吉、中吉、・・・のうち対応するものをdocument.write()関数で表示するようにしてある。

 answer = confirm('もう一度、おみくじを引きますか？');

confirm()関数は、[OK]と[キャンセル]の二つのボタンをもったダイアログボックスを表示する。返り値は、[OK]をクリックした場合はtrue、[キャンセル]をクリックした場合はfalseである。[キャンセル]するまで、while文で何度でも繰り返される。このように、answerは回数を数えるカウンタ変数ではないことに注意すること。

## Do! 上のomikuji.htmlを作成し、必勝おみくじになっているか、動作を確認しなさい。

## 2-4. JavaScriptのdo/while文

while文では、繰り返し条件を判定してから処理を行うが、do/while文は処理をおこなってから繰り返し条件の判定を行う。

JavaScript文1;

JavaScript文2;

・・・

JavaScript文n;

条件

真

偽

do {

 JavaScript文1;

 JavaScript文2;

 ・・・

 JavaScript文n;

} while( 繰り返しの条件);

omikuji.htmlをdo/while文で書き換えると次のようになる。

[omikuji2.html]

<html>

<head>

<meta http-equiv="content-type" content="text/html;charset=Shift\_JIS">

<title>do/while文</title>

</head>

<body>

<h2>必勝おみくじ</h2>

<script type="text/javascript">

var kuji; // 0～5のどれか

alert('おみくじを引きます');

var answer;

do{

 kuji = Math.floor(Math.random() \* 6);

 switch( kuji){

 case 0:

 document.write('<h3>大吉</h3>');

 break;

 case 1:

 document.write('<h3>中吉</h3>');

 break;

 case 2:

 document.write('<h3>小吉</h3>');

 break;

 case 3:

 document.write('<h3>吉</h3>');

 break;

 case 4:

 document.write('<h3>末吉</h3>');

 break;

 case 5:

 document.write('<h3>凶</h3>');

 break;

 default:

 document.write('<h3 style="color: #F00;">エラーが発生しました！</h3>');

 }

 answer = confirm('もう一度、おみくじを引きますか？');

}while( answer );

</script>

</body>

</html>

## Do! omikuji.htmlの複製をomikuji2.htmlという名前で作成し、while文を上のようにdo/while文に書き換え、必勝おみくじになっているか、動作を確認しなさい。

# 3. 配列変数

複数の変数に添え字（具体的には、0から始まる数字、変数の背番号のようなもの）を付けてひとかたまりとして扱うことができる。そのような変数として配列変数(array)がある。

　具体例で説明しよう。配列変数omikujiは次のように定義できる。

var omikuji = ['大吉', '中吉', '小吉', '吉', '末吉', '凶'];

配列変数は、変数名のあとに[番号]をつけた背番号つきの変数の集まりと見なせる。この背番号のことをインデックス（添え字）という。インデックスでそのひとまとまりの変数[[2]](#footnote-2)のうちの一つの変数を参照できるので、繰り返し処理でよく使用される。この例では、

omikuji[0] = '大吉';

omikuji[1] = '中吉';

omikuji[2] = '小吉';

omikuji[3] = '吉';

omikuji[4] = '末吉';

omikuji[5] = '凶';

というように、六つの変数[[3]](#footnote-3)に背番号0～5が付けられ、各々に文字列データを格納して初期化されるようにするために、上のようにコンマ（,）で区切って[　]で囲った値のリストを代入した[[4]](#footnote-4)。

　この配列変数を使えば、必勝おみくじomikuji.htmlでswitch文を使わずに同じ結果になるようにプログラム（コーディング）できる。次のようにすると良い。行数の少ないシンプルなソースコードとなる。従って、おみくじの結果に、半吉、平などを追加するのも容易である。

[array.html]

<html>

<head>

<meta http-equiv="content-type" content="text/html;charset=Shift\_JIS">

<title>配列変数(array)</title>

</head>

<body>

<h2>必勝おみくじ</h2>

<script type="text/javascript">

var omikuji = ['大吉', '中吉', '小吉', '吉', '末吉', '凶'];

var kuji; // 0～5のどれか

alert('おみくじを引きます');

var answer = true;

while( answer ){

 kuji = Math.floor( Math.random() \* 6 );

 document.write('<h3>' + omikuji[kuji] + '</h3>');

 answer = confirm('もう一度、おみくじを引きますか？');

}

</script>

</body>

</html>

## Do! omikuji.htmlの複製をomikuji3.htmlという名前で作成し、配列変数omikujiを上のよう使ってソースコードを書き換え、必勝おみくじになっているか、動作を確認しなさい。

# 4. 課題08

while文と配列を使ったomikuji3.htmlを改造して、「大吉」の文字だけは、色を赤にして、見出し2の書式で表示されるようにしなさい。更に、大吉、中吉、小吉、吉、末吉、凶の6種類以外の結果が出るように拡張しなさい。

*Hint:* 文字の色を赤にするには、<span style="color:red;">大吉</span>と書き出すようにすればよい。また、見出し2の書式とは<h2>～</h2>のことである。今回の場合、h2要素の内容を「大吉」という文字だけにするならば、わざわざspan要素を追加しなくても、h2要素にインラインでCSSを適用して、「<h2 style="color: red;">大吉</h2>」としても良い。

*Hint：*「大吉」の場合とは、変数kujiの値が0の場合である。従って、kujiの値が0に等しいという条件でif～else文を使って、0の場合だけ、色と書式を変えて出力すれば良い。

或いは、配列変数omikujiの各要素omikuji[0]～omikuji[5]の値に最初から<span …>～</span>、或いは<h3 …>～</h3>を付け加えたものにするという方法でも可能である。やり方は一通りではないので、自分で解答(solution)を見つけて欲しい。

*Hint：*おみくじの結果を6種類よりも増やすには、配列変数omikujiの要素を増やし、kujiの値がomikujiのインデックスの範囲になるようにすれば良い。下のように記述してやれば、配列の要素がいっぱい増えても、読みやすさは損なわれない。また、各要素の後ろに//でコメントを付けることができて便利である。

var omikuji = [

'大吉',

'中吉',

'小吉',

'吉', // 大吉の次にラッキーとされている

'末吉',

'凶'

];

# 5. Check List

* switch文の書き方を理解したか？
* for文の書き方を理解したか？
* while文の書き方を理解したか？
* for文をwhile文に書き換えるには、どのように書けば良いか理解したか？従って、for文は必ずwhile文に書き換え可能であることを理解したか？
* do/while文の書き方を理解したか？
* 配列変数とは、どのようなものか理解したか？
* Math.random( )は、何をする関数か？
1. 四捨五入にはMath.round()関数、小数点以下切り上げにはMath.ceil()関数を使う。 [↑](#footnote-ref-1)
2. 背番号の部分を除いた変数名が共通である変数の集まりのこと。ここでの例では共通の変数名は“omokuji”である。 [↑](#footnote-ref-2)
3. 変数名は共通の“omikuji”であるが、インデックスで区別されている。 [↑](#footnote-ref-3)
4. JavaScriptのオブジェクト指向プログラミングでは、Arrayという配列のクラスが用意されているので、この例のように[　]で囲った値のリストを代入して初期化するのではなく、var omikuji = new Array('大吉', '中吉', '小吉', '吉', '末吉', '凶');として配列変数オブジェクトを定義するのが普通である。 [↑](#footnote-ref-4)