Javascript & jQuery

2017 / 12

Javascript - 基礎編

■ はじめに

Webサイトを作成するには、テキストエディタなどでHTMLファイルを作成します。また、HTMLファイル内にはCSSを記述し、Webページ内の各要素について書式を指定することも可能です。
HTMLとCSS、この2つだけでもWebページは十分に作成できます。ただし、「動きのないページ」にしかなりません。
HTMLだけではユーザーやブラウザが提供する情報を使って表示する内容を変えることはできません。Javascriptは、「動的なWebページ」を作るための技術です。
Javascriptでは、Webページに動きを与えたりユーザーのマウス操作に反応してメッセージを表示するなど、ページ上でユーザーと対話ができるようになります。
「動的なWebページ」を可能にする手段としては、基本的に「サーバーサイドプログラム」と「クライアントサイドプログラム」に分かれます。
「サーバーサイドプログラム」にはPHPやCGIなどがあり、ユーザーが提供する情報をいったんWebサーバーに送信・アップロードをし、サーバーの中で情報を処理し、その処理結果をユーザーに返すという仕組みです。これに対して、ユーザー側が提供する情報を、そのままブラウザの中で処理する仕組みが「クライアントサイドプログラム」であり、JavascriptやVBscript、Javaアプレットなどがあります。
JavascriptはHTMLの中に記述したスクリプトをブラウザが直接解釈していきます。
JavaとJavascriptはどちらも"Java"という文字が付きますが、全く違うものです。
Javaによる方法はコンパイラを用いて、作ったアプレットをHTMLから呼び出して使います。
Javascript を用いることにより、ブラウザで表示するWebページ上で、リアルタイムで入力した値をチェックしエラー表示をしたり、時刻を表示したり、ページ上に何かしらの動きを与えたり、さらにはゲームまでできたりします。
文法的にはCに近い感じです。

■ Javascriptの記述方法

JavascriptはWebページに動きを与える事が出来るスクリプトですので、当然HTML文書と組み合わせて記述します。
組み合わせる方法は、html文書内に埋め込む方法と、別途用意したJavascriptファイル(jsファイル)をHTML文書から外部ファイルとして読み込む方法と、HTML/XHTML要素内に直接記述する方法があります。

<HTML文書に埋め込む方法>

JavascriptをHTML文書に埋め込むには、<script>タグを使用します。以下のように記述します。

```
<script type="text/javascript">
```

ここから、終了タグの</script>までの間はブラウザによってJavascriptだと解釈され、処理が実行されます。
また、<head> 部分に、

```
<meta http-equiv="Content-Script-Type" content="text/javascript" />
```

と書いておくのが望ましいとされています。
また、HTML5では、type属性が不要となり、<script></script>という形で記述し、metaタグの記述も省略できます。

○ 埋め込む位置

<script>タグで挟まれたJavascriptはHTML文書のどこに記述してもかまいませんが、<head>タグと</head>タグの間に記述する方法を多くとっています。ただし、文書中の特定の場所にJavascriptの処理結果を書き出す場合には、<body>タグと</body>タグの間の書き出したい場所に記述していきます。
JavascriptはHTML文書のなかに複数記述することもできます。

○ 外部Javascriptファイルを読み込む方法（この利用方法が一番望ましいです）

別ファイルにスクリプト部分だけを記述しておき（拡張子は .js）、HTML文書に読み込むことも可能です。
その場合は、スクリプトを埋め込みたい部分に

```
<script type="text/javascript" src="xxx.js"></script>
```

のように記述します。埋め込むスクリプトファイルはsrc属性で指定します。複数のページで同じスクリプトを使用したい場合に便利な方法です。

○ コメントの書き方

Javascriptでは、行の // より後ろに記述された部分はコメントとなります。
また、/* と */ で囲むことにより、複数行にわたってコメントを記述することができます。
コメントの部分は実行時に無視され、スクリプトの動作に影響を与えません。スクリプトに関する説明を記述したり、エラーの原因と思われる部分をコメントアウトして動作を確認したりするなどの用途で使用しましょう。

○ 記述の注意点

xhtmlでは大文字小文字の区別をしますが、Javascriptでも大文字小文字は区別されます。
大文字 / 小文字を無視して記述できませんので、十分注意してください。
また、HTMLタグと同じように、Javascriptでは必ず半角e英数文字で記述します。空白についても同様です。
空白文字は視覚的に判別しにくいので気をつけましょう（特に全角スペース）。
連続した半角空白やタブ文字は、htmlのソースコード同様無視されます。
このため、見やすく記述するためのインデントを、タブや半角スペースなどを利用し設定することが可能になります。
Javascriptでは、各文の最後に「;」（セミコロン）を記述し、文の区切りを明確に示す必要があります。
Javascriptで文字を扱うときは、「"」（ダブルコーテーション）または「'」（シングルコーテーション）で囲みます。
また、スクリプト内にタグを埋め込む場合、タグも文字列の一部として、「"」（ダブルコーテーション）または「'」（シングルコーテーション）で囲む必要があります。
<script>タグで囲まれた部分は、htmlのタグはそのまま使えませんので注意してください。

■ プログラム例

○ 例題1　sample1.html

```
<script type="text/javascript">
document.write( "<h2>Hello</h2>" ) ;
</script>
```

オブジェクト　メソッド　HTML文

body に書く

Javascriptの基本的な処理はdocument.write("HTML文");
でHTMLファイルに出力することです。

prompt() メソッド
入力ボックス
変数＝prompt（"文字列", "初期値"）

画面に出すMSG.

document オブジェクト
表示されているページに関する情報を持つ
オブジェクト

オブジェクト. プロパティ
オブジェクト. メソッド ( )

変数は宣言無しでも使えますが、宣言するには、
var 変数名; と記述します。
宣言なしだと、グローバル変数となるため、必ず変数宣言をするようにしましょう。
変数は、データ型はなく、数字や文字列でも格納することができます。
ユーザーがダイアログに入力した名前をresという変数に格納し、document.writeで書き出します。

複数の文字列を出力するとき
＋：連結してひとまとめにして出力
，：カンマで区切って別項目として出力

sample1.html

```
<body>
<h1>Javascript練習</h1>
<script type="text/javascript">
document.write( "<h2>Hello</h2>" ) ;
var res = prompt( "名前を入力してください" , "わせだ" ) ;
document.write( "こんにちは " , res , "さん" ) ;
</script>
</body>
```

変数
res ＝ 値
↖
代入（右辺の値を左辺の変数に代入する）

■ オブジェクト、プロパティ、メソッド、イベント

○ オブジェクト

Javascriptでは、ウィンドウ、ドキュメント、フォーム、文字列など、ページの表示に関わるさまざまなものを制御します。これらはすべて「オブジェクト」と呼ばれます。それぞれのオブジェクトには、状態を変更するための「プロパティ」や、命令を実行するための「メソッド」が多数用意されています。

○ プロパティ

「プロパティ」とはオブジェクトの属性です。たとえば、document（ドキュメント）オブジェクトにはドキュメントの背景色や文字色、ドキュメントのURL、タイトル、ドメインなどの属性があります。これらをスクリプトを使って参照または設定することによって、ページの背景色を変えたり別のページに移動させるなどの動作を実現できます。

```
オブジェクト. プロパティ＝値;
```

○ メソッド

「メソッド」とはあらかじめ用意されている命令です。Javascriptには、オブジェクトごとにさまざまなメソッドがあります。たとえば、String（文字列）オブジェクトには文字列の色やサイズを変更したり、文字列の検索や分割を行ったりするメソッドがあります。メソッドを使用する際には、必要に応じてカッコ内に引数を指定します。

```
オブジェクト. メソッド（値）;
```

↑ 引数

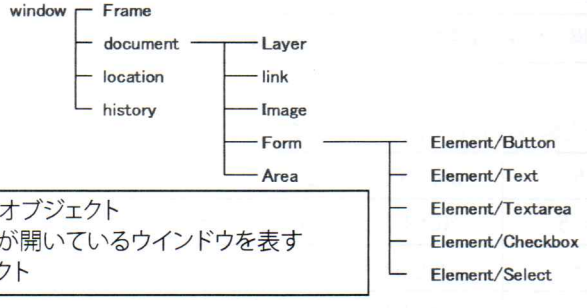
○ イベント

イベントは、マウスのボタンがクリックされた、ページの読み込みが完了した、フォームの選択メニューが変更されたなど、何かしらの動作が起こったときに発生します。

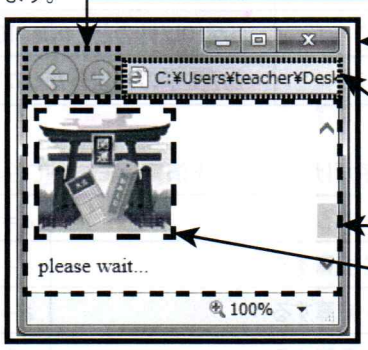
```
イベントハンドラ名＝"実行するJavascript命令";
```

○ オブジェクトモデル

Webブラウザ自体の情報や、ページの表示に関わる機能/部品をオブジェクト化したものをナビゲーターオブジェクトといいます。下図のように、Windowオブジェクトを最上位とする階層構造でオブジェクトを管理しています。
これらのオブジェクトは、そのひとつひとつがHTMLのタグと対応していると考えるとわかりやすいでしょう。
documentオブジェクトは、<body>、イメージオブジェクトは<img>のように対応しています。
また、Webブラウザが本来持っているオブジェクトのほかに、時間のオブジェクトや数値計算を行うオブジェクトなど、Javascript独自のオブジェクトもいくつか組み込まれています。これをビルトインオブジェクトといいます。
（Date、Array、String、Math など）



window オブジェクト
ブラウザが開いているウインドウを表す
オブジェクト



window オブジェクト
history オブジェクト
location オブジェクト
document オブジェクト
image オブジェクト

○ 例題2　sample2.html

```
<body onload="alert('ようこそ')">
<p><input type="button" value="もしもし" onclick="alert('はい')" /></p>
<p><img src="images/click.jpg" onmouseover="alert('マウスが画像にのった')" /></p>
</body>
```

← 読み込み完了イベント
← クリックイベント
← マウスオーバーイベント

■ 演算子

算術演算子、ビット演算子、論理演算子、比較演算子、代入演算子があります。

算術演算子

+	加算	2つの数値の和を求める	a + b	aにbを加える
-	減算	2つの数値の差を求める	a - b	aからbを引く
*	乗算	2つの数値の積を求める	a * b	aとbを掛ける
/	除算	2つの数値で除算を行います	a / b	aをbで割る
%	剰余	2つの数値で除算し、その余りを求める	a % b	aをbで割った余り
++	インクリメント	変数の値を1増やす	a++	aの値を1増やす(a = a + 1 と同じ)
--	デクリメント	変数の値を1減らす	a--	aの値を1減らす(a = a - 1 と同じ)

ビット演算子

&	ビットごとのAND	2つの値で各ビットの論理積を求める
|	ビットごとのOR	2つの値で各ビットの論理和を求める
^	ビットごとのXOR	2つの値で各ビットの排他的論理和を求める
~	ビットごとのNOR	各ビットを反転させた結果を求める
<<	左シフト	指定された分だけ各ビットを左にシフトする
>>	右シフト	指定された分だけ各ビットを右にシフトする
>>>	符号なし右シフト	符号を考慮せず、右シフトする

論理演算子

&&	論理積	2つの値の論理積を求める(両方がtrueの場合のみtrue)
||	論理和	2つの値の論理和を求める(一方または両方がtrueならtrue)
!	論理否定	指定された値の論理否定を求める
?:	条件	条件がtrueなら1つめの文、falseなら2つめの文を実行する
,	カンマ	2つの式を順に続けて実行する

比較演算子

<	より小さい	a < 10	aが10未満のときtrue、10以上のときfalse
<=	以下	a <= 10	aが10以下のときtrue、10より大きいときfalse
>	より大きい	a > 10	aが11以上のときtrue、10以下のときfalse
>=	以上	a >= 10	aが10以上のときtrue、10未満のときfalse
==	等しい	a == 10	aが10のときtrue、それ以外のときfalse
===	等しい	a === 10	aが10のとき型も一致する場合true、それ以外のときfalse
!=	等しくない	a != 10	aが10以外のときtrue、10のときfalse
!==	等しくない	a !== 10	aが文字の10以外だった場合true、数値の10だったときfalse

代入演算子(複合代入)*算術演算子だけでなくビット演算子についても=と組み合わせて同様に記述できます

=	代入	値を変数に代入する	a = 5	aに5を代入する
+=	加算	指定された値を加算する	a += 5	aに5を加える(a = a + 5 と同じ)
-=	減算	指定された値を減算する	a -=5	aから5を引く(a = a - 5 と同じ)
*=	乗算	指定された値を乗算する	a *= 5	aに5を掛ける(a = a * 5 と同じ)
/=	除算	指定された値で除算する	a /= 5	aを5で割る(a = a / 5 と同じ)
%=	剰余	指定された値で除算し、余りを求める	a %= 5	aを5で割った余り(a = a % 5 と同じ)

■ 主要な言語仕様

<for文 処理を指定回数繰り返す>

条件がtrueの間、変数を初期値から増減させながら処理を実行する

```
for( 変数名 = 初期値 ; 条件 ; 増分 ) {
        処理
}
```

例

```
for( i=0 ;  i<3 ;  i++ ){          ← iを0から処理するごとに1づつ増加しながらi<3の間実行する
    :                              ← { } 内にループさせたい処理を記述します
}
```

初期値　ループの条件式　増減式

○ 例題3 forの記述例

sample3.html

```
<body>
<h1>for文</h1>
<hr />
<script type="text/javascript">
for ( var i = 1 ; i <= 5 ; i++ ) {
        document.write( i , " : Javascript<br />" ) ;
}
</script>
<hr />
</body>
```

この様な実行結果になります

1:JavaScript
2:JavaScript
3:JavaScript
4:JavaScript
5:JavaScript

○ 例題4 for文を使って、指定した数字までの合計を順に表示させてみましょう

例題4:整数の合計

5までの合計を計算
0+1=1
1+2=3
3+3=6
6+4=10
10+5=15
答え:15

sample4.html

```
<body>
<script type="text/javascript">
var total = 0 ;
var num = prompt( "いくつまでの合計を表示しますか" , 5 ) ;
document.write( num , "までの合計を計算<br />" ) ;
for ( var i = 1 ; i <= num ; i++ ) {
        document.write( total , "+" , i ) ;
        total = total + i ;
        document.write( "=" , total , "<br />" ) ;
}
document.write( "答え:" ,  total ) ;
</script>
</body>
```

○ 例題4追加1 以下のようにテーブルのタグを使って、整列させて表示してみよう

回数	計算式	合計
1回	0+1	=1
2回	1+2	=3
3回	3+3	=6
4回	6+4	=10
5回	10+5	=15

sample4-1.html

```
<body>
<table border="1">
        <tr>
                <th>回数</th><th>計算式</th><th>合計</th>
        </tr>
<script type="text/javascript">
var total = 0 ;
var num = prompt("いくつまでの合計を表示しますか",5);
for(                              ) {
        document.write("<tr><td>" , i , "回</td>" ) ;
        document.write("<td>" ,        , "+",    ,"</td>" ) ;
        total =       +    ;
        document.write( "<td>=" , total , "</td></tr>" ) ;
}
</script>
</table>
</body>
```

○ 例題5 for文を使い、ユーザーエージェント情報を表示させてみましょう

sample5.html

```
<body>
<table border="1">
<script type="text/javascript">
for( i in navigator ) {
        var val = navigator[ i ] ;
        document.write( "<tr><td>" , i , "</td><td>" , val , "</td></tr>" ) ;
}
</script>
</table>
</body>
```

for in 構文

配列の中身を繰り返し処理で順番に要素を取り出す

<while文 繰り返し処理 条件が真の間、処理を繰り返す>

```
while (条件式){
        文1      ←条件を満たしたときに繰り返される文
}
```

```
do   {
        文1      ←条件を満たしたときに繰り返される文
} while (条件式)
```

{と}で囲まれた部分は複文になります。
~~単文の場合は{}はなくてもかまいません。~~ 原則つける!
Whileの場合、最初から条件がfalseの場合一度も処理は行われません。
doの場合、処理の実行後に条件判断が行われるため、最初から条件を満たしていない場合でも最低一度は処理が実行されます。

○ 例題6 while文を使って、例題4で作成した『指定した数字までの合計を順に表示』を作成してみましょう

例題4:整数の合計

5までの合計を計算
0+1=1
1+2=3
3+3=6
6+4=10
10+5=15
答え:15

sample6.html

```
<body>
<script type="text/javascript">
var total = 0 ;
var num = prompt( "いくつまでの合計を表示しますか" , 5 ) ;
document.write( num , "までの合計を計算<br />" ) ;
var i=1;
while ( i <= num ) {
        document.write( total , "+" , i ) ;
        total = total + i ;
        document.write( "=" , total , "<br />" ) ;
        i++;
}
document.write( "答え:" , total ) ;
</script>
</body>
```

<多重ループ>
ループの中にまたループが入っている構造を多重ループといいます。一般にループの中にループが入ることや条件判断の中に条件判断が入ることを「ネスト」(入れ子)と呼びます。
例

```
for( i = 1 ; i <= 3 ; i++ ) {
   :
      for( j = 1 ; j <= 2 ; j++ ) {
         :
      }
   :
}
```

内側のループ　外側のループ

sample7.html

```
<body>
<h1>多重ループ</h1>
<p> i : j </p>
<hr />
<script type="text/javascript">
for ( var i = 1 ; i <= 3 ; i++ ) {
        for (var  j = 1 ; j <=2 ; j++ ) {
                document.write( i , " : " , j , "<br />" ) ;
        }
        document.write( "<hr />" ) ;
}
</script>
</body>
```

```
i : j
1 : 1
1 : 2
2 : 1
2 : 2
3 : 1
3 : 2
```

○ 演習1 九九表を作成しましょう
多重ループを使用し、九九表を完成させましょう。

九九表

1×1=1	1×2=2	1×3=3	1×4=4	1×5=5	1×6=6	1×7=7	1×8=8	1×9=9
2×1=2	2×2=4	2×3=6	2×4=8	2×5=10	2×6=12	2×7=14	2×8=16	2×9=18
3×1=3	3×2=6	3×3=9	3×4=12	3×5=15	3×6=18	3×7=21	3×8=24	3×9=27
4×1=4	4×2=8	4×3=12	4×4=16	4×5=20	4×6=24	4×7=28	4×8=32	4×9=36
5×1=5	5×2=10	5×3=15	5×4=20	5×5=25	5×6=30	5×7=35	5×8=40	5×9=45
6×1=6	6×2=12	6×3=18	6×4=24	6×5=30	6×6=36	6×7=42	6×8=48	6×9=54
7×1=7	7×2=14	7×3=21	7×4=28	7×5=35	7×6=42	7×7=49	7×8=56	7×9=63
8×1=8	8×2=16	8×3=24	8×4=32	8×5=40	8×6=48	8×7=56	8×8=64	8×9=72
9×1=9	9×2=18	9×3=27	9×4=36	9×5=45	9×6=54	9×7=63	9×8=72	9×9=81

multi-table.html

```
<body>
<h1>九九表</h1>
<table border="1">
<script type="text/javascript">
for ( i : 1~9 ) {
        document.write( "<tr>" ) ;
        for ( j : 1~9 ) {
                var z = i x j
                document.write( "<td>" , i , "×" , j , "=" , z , "</td>" ) ;
        }
        document.write( "</tr>" ) ;
}
</script>
</table>
</body>
```

<if文 条件分岐（2分岐）条件が真か偽かで処理を分ける>

```
if(条件式){
        処理1      ←条件を満たしたときに実行される処理
} else {
        処理2      ←条件を満たさないときに実行される処理
}
```

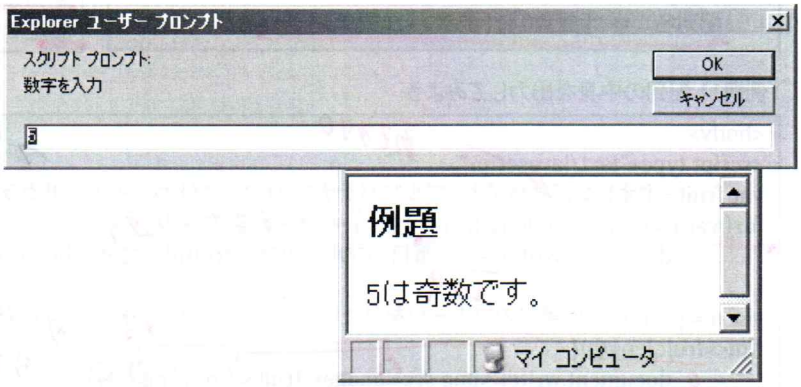
{と}で囲まれた部分は複文になります。
true(真)の場合のみ処理をさせたい場合は、else以下を記述しません。
また、else if・・・のようにelseの後にifを続けて、さらに条件を分岐させることもできます。

○ 例題8 入力した数字が奇数か偶数かを判断する

sample8.html

```
<body>
<script type="text/javascript">
var kazu = prompt( "数字を入力" , 5 ) ;
document.write( kazu , "は" ) ;
if( kazu %2 == 0 ) {
        document.write( "偶数です。" ) ;
} else {
        document.write( "奇数です。" ) ;
}
</script>
</body>
```

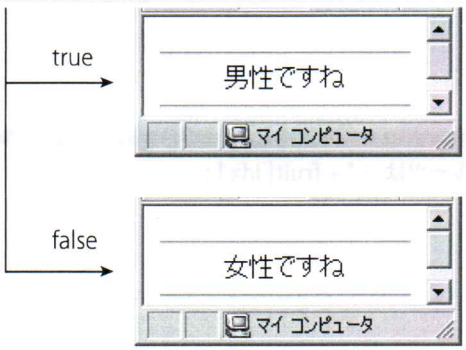
2^53 までの数値は計算可能.



○ 例題9 確認ダイアログを表示して、OK・キャンセルボタンクリックによって、出力メッセージを変えてみましょう


```
confirm()
確認ダイアログボックス
変数 = confirm ("文字列");
```

「OK」のとき true
「キャンセル」のとき false が返されます



sample9.html

```
<body>
<hr />
<script type="text/javascript">
var seibetu=confirm("男性?");
if (seibetu === true){
        document.write("男性ですね<br />");
} else {
        document.write("女性ですね<br />");
}
</script>
<hr />
</body>
```

○ 例題10 上記の例題9を以下のように追加・修正してみましょう　0～19/20～59/60～
年齢を入力させて、未成年なら「あとXX年で成年です」、成年で60歳未満なら「あとXX年で還暦です」、60歳以上は「100歳まであとXX年」などと、年齢によって違ったメッセージを表示させましょう。（可能であれば、100歳以上や、0歳未満の場合も想定して追加してみましょう。）

```
' 男性ですね
' あなたは63才ですね
' 100才まであと37年いきてね
```

```
' 男性ですね
' あなたは25才ですね
' 還暦まであと35年がんばって
```

```
' 女性ですね
' あなたは15才ですね
' 未成年ですね
( あと5年お酒は我慢です
```

<else if文 条件分岐（多分岐）条件によって複数の処理を振り分ける>

```
if( 条件式 ) {
        処理
} else if ( 条件式 ) {
        処理
} else if ( 条件式 ) {
        処理
} else {
        処理
}
```

記述した順に条件式をきき、条件を満たす時(真)には処理を、満たさなかった時(偽)は、次の条件式をききます
条件式のいずれも満たさなかったときは、else以降を実行します

<switch case文 条件分岐（多分岐）条件によって複数の処理を振り分ける>

```
switch( 条件式 ) {
        case  値 : 処理
                break ;
        case  値 : 処理
                break ;
        case  値 : 処理
                break ;
        default : 処理
}
```

条件が当てはまる値のときに処理を実行、どれにも当てはまらなければ、default以降の処理を実行します。
breakは処理を抜けるときに使います。命令が入れ子になっている場合は、一番内側の処理を抜けます。

注意! case と 値 の間には、必ず半角空白を入れること

<配列>

複数の値をひとまとめにして扱う仕組み（データ構造）のことで、同じ属性のものを同じ名前で繰り返して領域をとります。配列の要素を参照するときは、先頭からの番号（0始まり）で指定します。この番号を添字といいます。要素を参照するには、配列名[添字] と設定します。

```
配列の定義
    配列名 = new Array( 要素の数 ) ;
または
    配列名 = new Array( 要素[ 0 ] , 要素[ 1 ] , 要素[ 2 ]・・・・ ) ;
または
    配列名 = [ 要素0 ] , [ 要素1 ] , [ 要素2 ] , ・・・ ;
```

```
a=["あ","い","う","え","お"];
a=new Array("あ","い","う","え","お");
```

a[0]	a[1]	a[2]	a[3]	a[4]
あ	い	う	え	お

a.length プロパティ
配列 a の要素数を参照、設定できます

○ 例題11 配列の中身を出力してみよう

sample11.html

```
<body>
<script type="text/javascript">
var fruit = ["オレンジ","パイナップル","バナナ","メロン","ブドウ","モモ","サクランボ","イチゴ","ドリアン" ] ;
for( var i = 0 ; i < fruit.length ; i++ ) {
        document.write( i+1 , "番目のフルーツは" , fruit[i] , "です。<br />" ) ;
}
var n = prompt( "何番目のフルーツを見たいですか?" , 5 ) ;
if(n<=fruit.length){
        document.write("<img src='images/fruit_", n , ".jpg' />") ;
        document.write("<br />" , n , "番目のフルーツは・・・ " , fruit[n-1] , "です" ) ;
}else{
        document.write( n , "番目のフルーツはまだありません。" ) ;
}
</script>
</body>
```

for in 構文
配列の中身を繰り返し処理で順番に要素を取り出す

getElementById
引数に指定されたIDのタグのオブジェクトを取得する

selectedIndexプロパティ
選択項目の参照番号を取得（0～）
値ではなく、配列の添字であることに注意

innerHTML 属性
タグの内容として設定されるHTMLソースコードを示す属性

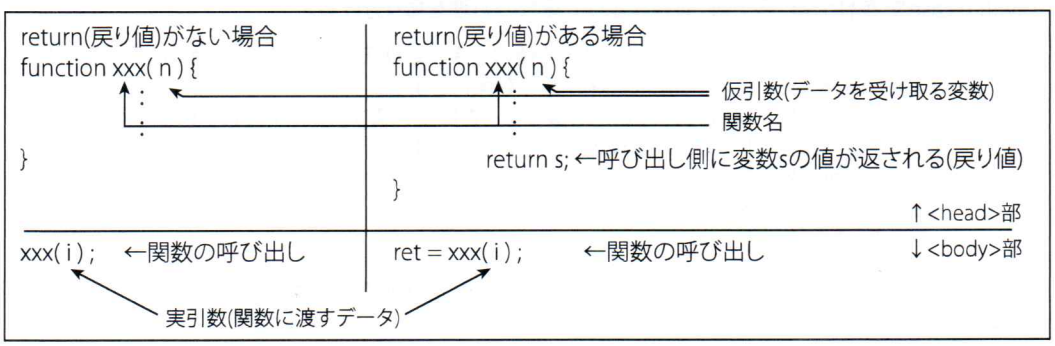
○ 例題12 例題11を以下の様に変更してみましょう（配列にスイカを加えてみましょう）

sample12.html

```
<head>
<script type="text/javascript">
function show( ) {
        var idx = document.getElementById( "basket" ).selectedIndex ;
        document.fImg.src = "images/fruit_"+ ( idx + 1 ) + ".jpg" ;
        document.getElementById( "msg" ).innerHTML = ( idx+1 ) + "番目のフルーツは・・・ " + fruit[ idx ] ;
}
</script>
</head>
<body>
<form name ="f1">
        <select id="basket" name="basket" onchange="show( )">
        <script type="text/javascript">
        var fruit = [ "オレンジ","パイナップル","バナナ","メロン","ブドウ","モモ","サクランボ","イチゴ","ドリアン","スイカ" ] ;
        for( i in fruit ) {
                document.write( "<option>" + fruit[ i ] + "</option>" ) ;
        }
        </script>
        </select>
</form>
<p id="msg">好きなものを選んでね</p>
<p><img name="fImg" src="images/fruit.jpg" /></p>
</body>
```

■ 関数

ある処理単位をひとかたまりにして、その名前で呼び出しが行えるようにしたプログラムを関数と呼びます。



※ 関数定義部
通常<head>部で行います

○ 例題13 returnのある関数の例
2つの数字の間を合計する処理。入力した2数を引数とし、その間の合計を返す関数 sum(a,b)

sample13.html

```
<head>
<script type="text/javascript">
function sum(x,y){
        var kei = 0 ;
        var x = parseInt( x ) ;
        var y = parseInt( y ) ;
        while(x <= y) {
                kei = kei + x ;
                document.write( x ,"  ") ;
                x++;
        }
        return kei ;
}
</script>
</head>
<body>
<h1>関数sum</h1>
<script type="text/javascript">
var a = prompt( "加算する最初の数字を入力" , 10 ) ;
var b = prompt( "加算する最後の数字を入力" , 20) ;
document.write( a , "から" , b , "までの合計 <br />" ) ;
var kotae = sum( a , b ) ;
document.write( "=" , kotae ) ;
</script>
</body>
```

変換しないと文字として扱われてしまう.

parseInt( )
文字列を整数に変換する関数

○ 例題14 ボタンのクリックでバックカラーを変更します。その時確認メッセージを表示します。

sample14.html

```
<head>
<script type="text/javascript">
function change( col ) {
        var ret = confirm( "背景色を変更していいですか?" ) ;
        if( ret === true ) {
                document.bgColor = col ;
        }
}
</script>
</head>
<body>
<h1>ボタンクリックによるバックカラーの変更</h1>
<p><input  type="button" value="水色" onclick="change('#00ffff')" />
<input  type="button" value="紫色" onclick="change('#ff00ff')" />
<input  type="button" value="黄色" onclick="change('#ffff00')" /></p>
</body>
```

■ 予約オブジェクト

あらかじめ定義された予約オブジェクトには、ECMAScriptの仕様で定められているオブジェクト（Dateなど）と、ブラウザやウェブページを操作するオブジェクト（windowなど）があります。

<documentオブジェクト>

documentオブジェクトは、それが記述されているHTMLファイルの示すオブジェクトです。
文字列を表示（書き出す）には、writeメソッド、文字列を書き出して改行するならwritelnメソッドを使います。
また、文字色を設定したいなら、document.fgColor（文字色）、document.linkColor（リンクの文字色）、document.alinkColor（リンク中の文字色）、document.vlinkColor（訪問済みリンクの文字色）で設定します。
また、HTMLファイルの最終更新日時をlastModifiedプロパティで参照することもできます。

<Mathオブジェクト>

Mathオブジェクトは、sin、cos、log（自然対数）、random（乱数）、ceil（小数点以下切り上げ）、floor（小数点以下切り下げ）、round（小数点以下四捨五入）などのメソッドを使って数値計算を行うことや、PI（円周率）、E（自然対数の底）プロパティなどを参照できるオブジェクトです。

○ 例題15 乱数を発生させて、おみくじを作成しましょう。
6つのメッセージと画像を配列に定義して、乱数の値によって対応するメッセージと画像を表示しましょう。
（画像は「images」フォルダ内に大吉:image1.jpg、中吉:image2.jpg・・という順で名前がついています）

sample15.html

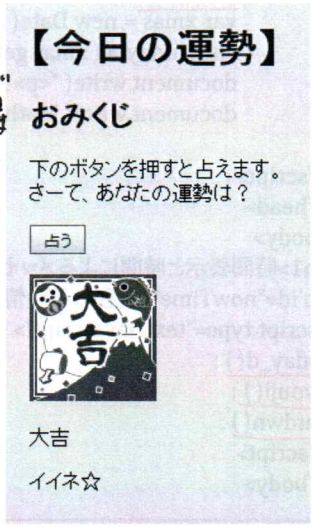
```
<head>
<title>おみくじ</title>
<script type="text/javascript">
function startomikuji( ) {
        var omikuji = [ "大吉","中吉","小吉","吉","末吉","凶" ] ;
        var message = [ "イイネ☆","まずまずか。","フーン。","平凡な日だね。","イマイチじゃんよー。","ツイてないね…。" ] ;
        var back = [ "#ffff00","#ff00ff","#808000","#808080","#00ffff","#f0f0f0" ] ;
        var i = Math.floor( Math.random( ) * omikuji.length ) ;
        document.getElementById( "msg1" ).innerHTML = omikuji[ i ] ;
        document.getElementById( "msg2" ).innerHTML = message[ i ] ;
        document.result.src = 'images/image' + ( i+1 ) + '.jpg' ;
        document.bgColor = back[ i ] ;
}
</script>
</head>
<body>
<h1>【今日の運勢】</h1>
<h2>おみくじ</h2>
<p>下のボタンを押すと占えます。<br />さーて、あなたの運勢は?</p>
<p><input type="button" value="占う" onclick="startomikuji()" /></p>
<p><img name="result" src="images/0.jpg" width="100" alt="占い結果" /></p>
<p id="msg1">please wait 1 ...</p>
<p id="msg2">please wait 2 ...</p>
</body>
```

msg1
msg2

6 → 6 x (0 ～ 0.99...)
= 0 ～ 5.999...
まで取得可能

必須
( ) がないと、文字列結合してしまう.

Math.random( )
乱数を発生（0～1未満）
Math.floor(値)
小数点以下切り下げ



<Dateオブジェクト>

Dateオブジェクトを使うと、取得日時で異なる処理をしたり、日数計算をすることができます。
日時を取得するには以下のように記述します。

```
var t = new Date( ) ;
```

t には、"Thu Jun 06 21:42:25 2015"のような現在日時が得られます。この t から「時」だけを取り出すには次のように記述します。

```
var h = t.getHours( ) ;
```

○ 例題16 時間表示と曜日と時間により、違うメッセージを表示させましょう。

sample16.html

```
<head>
<script type="text/javascript">
var now = new Date( ) ; //日付オブジェクト作成(現在時刻)
function today_d( ) {
        setTimeout( "today_d( )" , 1000 ) ; //1秒ごとに日付オブジェクトを作成・表示
        document.getElementById( "nowTime" ).innerHTML = new Date( );
}
function hyouji( ) {
        var yy = now.getFullYear( ) ; //西暦を返すメソッド
        var mm = now.getMonth( ) ; //月を返すメソッド(0～11)
        var dd = now.getDate( ) ; //日を返すメソッド
        var hh = now.getHours( ) ; //時刻を返すメソッド
        var mi = now.getMinutes( ) ; //分を返すメソッド
        var today = now.getDay( ) ; //曜日を返すメソッド(0～6)
        var tbl = [ "日" , "月" , "火" , "水" , "木" , "金" , "土" ] ;
        var youbi = tbl[ today ] ; //配列dayのyoubi番号の要素
        //時刻によりメッセージを変える② 1行目
        if ( 4 <= hh && hh < 10 ) {
                document.write( "<p>おはよう<br />" ) ;
        } else if ( 10 <= hh && hh <= 17 ) {
                document.write( "<p>こんにちは<br />" ) ;
        } else if ( hh < 4 || hh > 17 ) {
                document.write( "<p>こんばんは<br />" ) ;
        }
        //曜日によりメッセージを変える② 2行目以降
        switch( today ) {
                case 0 : // 値0 日曜日を返す
                        document.write( "日曜日です。どこかに行きますか?</p>" ) ;
                        break ;
                case 6 : // 値6 土曜日を返す
                        document.write( "明日は日曜日ですね。</p>" ) ;
                        break;
                default : // 値1～5 平日を返す
                        document.write( "頑張って勉強しましょう。<br />" ) ;
                        h=7-today;
                        document.write( "日曜まであと",h,"日です!</p>" ) ;
        }
        document.write( "<p>今日は", yy ,"年", mm + 1 ,"月", dd ,"日", youbi ,"曜日です<br />" ) ;
        document.write( hh , " 時" , mi , " 分です</p>" ) ;
}
function cntdwn( ) {
        var xmas = new Date( now.getFullYear( ),11,24 ) ; //今年の12月24日
        var cntday = ( xmas.getTime( ) - now.getTime( ) ) / ( 24 * 60 * 60 * 1000 ) ;
        document.write( "<p>クリスマスイヴまであと <strong>" ) ;
        document.write( Math.ceil( cntday ) , "</strong> 日です!!</p>" ) ;
}
</script>
</head>
<body>
<h1>時間表示と時間によるメッセージ</h1>
<p id="nowTime">現在の日時情報表示エリア</p>
<script type="text/javascript">
today_d( ) ;
hyouji( ) ;
cntdwn( ) ;
</script>
</body>
```

時間表示①

時刻により変わるメッセージ②

getHours( )	時
getMinutes( )	分
getSeconds( )	秒
getFullYear( )	西暦
getMonth( )	月(0～11)
getDate( )	日にち
getDay( )	曜日(0～6)

Youbi の値に対し、配列dayを以下のように対応させる

日 月 火 水 木 金 土
0 1 2 3 4 5 6

getTime( ) メソッド
1970年1月1日0時0分0秒以降の経過時間をミリ秒で取り出す

クリスマスまでのカウントダウン③

Math.ceil( )
小数点以下切り上げ

最初は、関数の呼び出し部分はコメントアウトし、上記の関数が出来たら、一つずつ関数の呼び出しが出来る様に、コメントアウトを解除をして動作確認をしましょう!

○ 演習2 Dateオブジェクトを使用して、javascript カレンダーを作成しましょう
当月の表示が出来るカレンダーを作成し、順番に機能の追加等を行っていきましょう

calendar.html

```
<body>
<h1>カレンダー</h1>
<script type="text/javascript">
var date = new Date();
var Monthdays = new Array( 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 ) ;
var Days = new Array( "Sun", "Mon", "Tue", "Wed", "Thr", "Fri", "Sat" ) ;
var year = date.getFullYear( ) ;
var today = date.getDate( ) ;
// 年、うるう年の設定
if( ( ( year%4 == 0 ) && ( year %100 != 0 ) ) || ( year % 400 == 0 ) ) {
        Monthdays[1] = 29;
}
var thisMonthDays = Monthdays[ date.getMonth( ) ] ;
date.setDate(1);
var Startday = date.getDay( ) ;
// 年・月表示
document.write( "<table border='1'>" ) ;
document.write( "<caption>" , year , "年" , date.getMonth( ) + 1 , "月</caption>" ) ;
// 曜日の表示
document.write( "<tr>" ) ;
for( var i = 0 ; i < 7 ; i++ ) {
        document.write( "<th>" , Days[ i ] , "</th>" ) ;
}
document.write( "</tr>" ) ;
// 日にちの表示
document.write( "<tr>" ) ;
var col = 0 ;
for( i = 0 ; i < Startday ; i++ ) {
        document.write( "<td> </td>" ) ;
        col++ ;
}
for( i = 1 ; i <= thisMonthDays ; i++ ) {
        if( i == today ) {
                document.write( "<td><strong>" , i , "</strong></td>" ) ;
        } else {
                document.write( "<td>" , i , "</td>" ) ;
        }
        col++ ;
        if( ( i != thisMonthDays ) && ( col == 7 ) ) {
                document.write( "</tr><tr>" ) ;
                col = 0 ;
        }
}
for( i = 1 ; i <= 7 - col ; i++ ) {
        document.write( "<td> </td>" ) ;
}
document.write( "</tr></table>" ) ;
</script>
</body>
```

date( )
日付および時刻を格納しておくオブジェクト
setDate( )
任意の日付（日）を設定します。
負数を指定した場合には、現在の日付からの差分が設定されます。

カレンダー表示用の初期設定・情報の準備（各月の日数や曜日の設定）

カレンダータイトルや曜日の表示設定

月初めの空白処理

カレンダー本体表示処理

終わりの空白処理

○ 追加課題
カレンダー完成後、以下の内容で順番にカスタマイズをし機能追加をして下さい。

1. function でカレンダーを関数化
2. 関数化後、<body>タグ内で呼び出して実行する
3. calendar.js という形で外部ファイルにする
4. calendar.js をhtmlファイル内に読み込む
5. 外部cssで外観の装飾設定
6. 曜日ごとの区別を出来る様にする
7. 曜日（土・日）に色を付ける
8. 引数を使用し、指定の月の表示が出来る様にする（前月/当月/次月 など）
9. ボタンを押して、前月・次月等の表示切替が出来るようにする

※必ず一つ一つ手順を踏んで機能追加を行ってください。細かい動作確認が機能追加時のエラーを減らすコツになります。頑張りましょう!

■ イベント処理

フォームのボタンをクリックしたり、ホットテキスト(リンクテキスト)の上をマウスが通過したなどの事象をイベントと呼びます。
イベントの発生によりイベント関数に処理を移すことをイベント処理と呼びます。
指定できる主なイベント処理は以下のものがあります。

onclick	button , checkbox , radio , reset , submit またはホットテキストをクリックしたとき
onchange	select , text , textarea の値が変化したとき
onfocus	select , text , textarea にフォーカスが移ったとき
onload	ページがロードされたとき
onmouseover	ホットテキストまたはアンカー上をマウスが通過したとき
onsubmit	submit ボタンがクリックされたときなど

<onchangeイベント処理>

onchangeイベントは、オブジェクトの変更時に発生し、組み合わせるタグによって選択メニューでの選択項目やテキスト入力欄の文字列が変更されたときに発生させることができます。

<onselectイベント処理>

onselectイベントは、選択時に発生します。

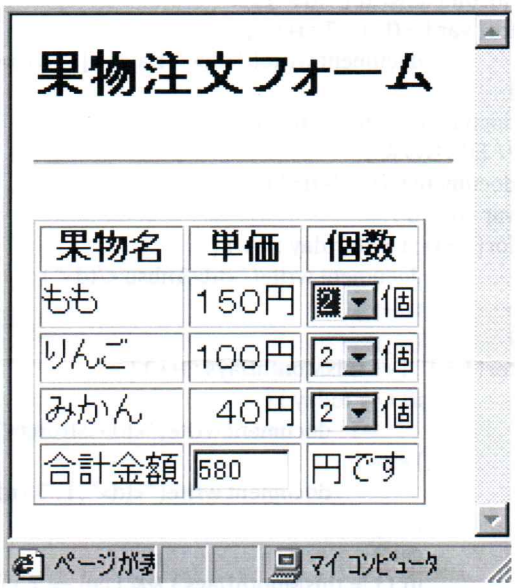
○ 例題17 フォーム操作時に計算させてみましょう。

フォーム選択メニューが変更されたときに関数calcを呼び、計算を行って結果をフォームに表示してみましょう。

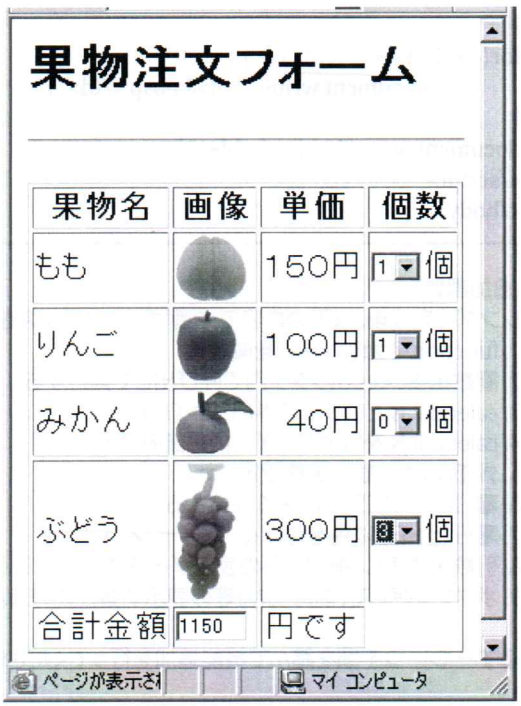
sample17.html

```
<head>
<title>onchange</title>
<script type="text/javascript">
function calc( ) {
        var mo = document.fruits.momo.selectedIndex * 150 ;
        var ri = document.fruits.ringo.selectedIndex * 100 ;
        var mi = document.fruits.mikan.selectedIndex * 40 ;
        document.fruits.output.value = mo + ri + mi ;
}
</script>
</head>
<body>
<h1>果物注文フォーム</h1>
<form name="fruits"><table border="1">
        <tr>
                <th>果物名</th><th>単価</th><th>個数</th>
        </tr><tr>
                <td>もも</td>
                <td>150円</td>
                <td><select name="momo" onchange="calc()">
                        <option>0</option>
                        <option>1</option>
                        <option>2</option>
                        <option>3</option></select>個
                </td>
        </tr><tr>
                <td>りんご</td>
                <td>100円</td>
                <td><select name="ringo" onchange="calc()">
                        <option>0</option>
                        <option>1</option>
                        <option>2</option>
                        <option>3</option></select>個
                </td>
        </tr><tr>
                <td>みかん</td>
                <td>40円</td>
                <td><select name="mikan" onchange="calc()">
                        <option>0</option>
                        <option>1</option>
                        <option>2</option>
                        <option>3</option></select>個
                </td>
        </tr><tr>
                <td>合計金額</td>
                <td><input type="text" name="output" size="8" /></td>
                <td>円です</td>
        </tr>
</table></form>
</body>
```

selectedIndexプロパティ
選択項目の参照番号を取得(0～)
値ではなく、配列の添字であることに注意



追加機能 (sample17-1.html)
各果物の画像とぶどうの項目を追加してみよう



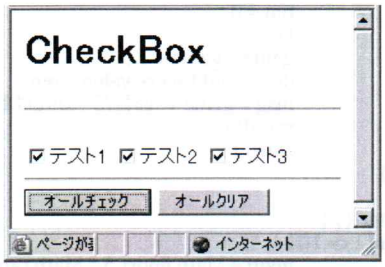
■ フォームの操作

formオブジェクトはHTMLの<form> ～ </form>に対応するオブジェクトで、配列で管理することができます。
また、name属性でformタグに名前をつけて指定することもできます。
formオブジェクトには、ボタンやラジオボタンなど、下位のオブジェクトが含まれています。
この下位のオブジェクトは、elements[添字]として指定ができます。
<form> ～ </form>内にinput、textarea、select、button などのタグが記述された順に0から順に添字が振られていきます。

<checkbox>

checkboxの各要素に対してチェックをコントロールするのは、checkedプロパティです。このプロパティにtrueを設定するとチェックされ、falseを設定するとチェックが外れます。

○ 例題18 チェックボックスを、すべてチェック、すべてクリアするボタンを作成しましょう。



sample18.html

```
<head><title>CheckBox</title>
<script type="text/javascript">
function ccheck( bvalue ) {
        var ccount = document.forms[ 0 ].elements.length ;
        for(var i = 0; i < ccount; i++){
                document.forms[ 0 ].elements[ i ].checked = bvalue ;
        }
}
</script></head>
<body>
<h1>checkbox</h1>
<form>
        <input type="checkbox" checked />テスト1   ← elements[0] として参照
        <input type="checkbox" />テスト2           ← elements[1] として参照
        <input type="checkbox" />テスト3           ← elements[2] として参照
</form><form>
        <input type="button" value="オールチェック" onclick="ccheck(true)" />
        <input type="button" value="オールクリア" onclick="ccheck(false)" />
</form></body>
```

○ 以下の様に追加・修正してみましょう

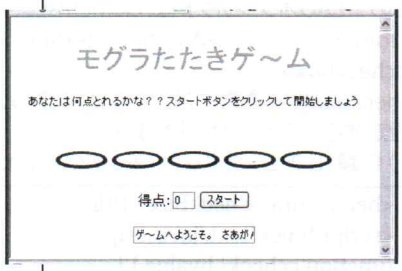
sample19.html

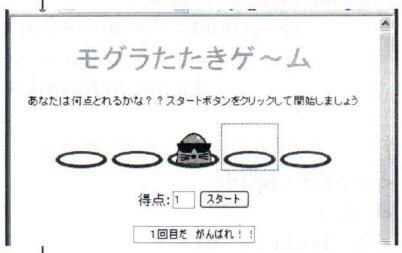
```
<head>
<title>体育館の利用料金</title>
<script type="text/javascript">
var total = 4800;
var kin = [2000,400,500];
function goukei( i ) {
        if( document.forms[ 0 ].elements[ i ].checked === true ) {
                total = total + kin[ i ] ;
        } else {
                total = total - kin[ i ] ;
        }
        document.forms[ 1 ].elements[ 0 ].value = total ;
}
function ccheck( bvalue ) {
        var ccount = document.forms[ 0 ].elements.length;
        for( var i = 0 ; i < ccount ; i++ ) {
                document.forms[ 0 ].elements[ i ].checked = bvalue ;
        }
        if( bvalue === true ) {
                total = 7700 ;
        } else {
                total = 4800 ;
        }
        document.forms[ 1 ].elements[ 0 ].value = total ;
}
</script></head><body>
<h1>体育館の利用料金</h1>
<p>基本使用料(2面、4時間) 4,800円</p>
<form>
        <p><input type="checkbox" onclick="goukei(0)" />夜間照明(2,000円)<br />
        <input type="checkbox" onclick="goukei(1)" />バレーボール用ネット×2面(400円)<br />
        <input type="checkbox" onclick="goukei(2)" />バレーボール×10個(500円)</p>
</form><form>
        <p>合計金額 <input type="text" size="6" value="4800" />円<br />
        <input type="button" value="オールチェック" onclick="ccheck(true)" />
        <input type="button" value="オールクリア" onclick="ccheck(false)" /></p>
</form></body>
```

○ 例題20 簡単なもぐらたたきゲームを作りましょう
追加　モグラがたたかれたときに、画像を変えてみましょう

sample20.html

```
<head><title>モグラたたきゲーム</title>
<script type="text/javascript">
var myimg1 = new Image( ) ;
myimg1.src="images/img1.gif" ;
var myimg2 = new Image( ) ;
myimg2.src = "images/img2.gif" ;
var myimg3 = new Image( ) ;
myimg3.src = "images/img3.gif" ;
var game = 0 ;
var i ;
var imgn ;
var ten ;
var msg ;
function syokika( ) {
        document.f.score.value = 0 ;
        msg = 'モグラたたきゲームへようこそ!!' ;
        scroll( ) ;
}
function play(){
        if( confirm( "スタートします" ) ) {
                ten = 0 ;
                i = 0 ;
                game = game + 1 ;
                document.f.score.value = ten ;
                msg = game + "回目だ　がんばれ!!" ;
                scroll( ) ;
                start( ) ;
        }
}
function start( ) {
        if( i <= 10 ) {
                imgn = Math.floor( 5 * Math.random( ) ) ;
                for( var a = 0 ; a <= 4 ; a++ ) {
                        document.imgmog[ a ].src = myimg3.src ;
                }
                document.imgmog[ imgn ].src = myimg2.src ;
                i++ ;
                var t = setTimeout( "start( )" , 750 ) ;
        } else {
                clearTimeout( t ) ;
                for( a = 0 ; a <= 4 ; a++ ) {
                        document.imgmog[ a ].src=myimg1.src ;
                }
                document.f.message.value = "あなたのスコアは" + ten + "点です。" ;
                if( ten < 3 ) {
                        alert( "なにやってんの。" ) ;
                } else if( ten < 6 ) {
                        alert( "まだまだだね。" ) ;
                } else if( ten < 10 ) {
                        alert( "そんなもんかな。" ) ;
                } else if( ten < 14 ) {
                        alert( "なかなかですね。" ) ;
                } else {
                        alert( "あんたはすごい!" ) ;
                }
        }
}
function count( n ) {
        if( imgn === n ) {
                ten++ ;
                document.f.score.value = ten ;
        }
}
function scroll( ) {
        if( i <= 10 ) {
                msg = msg.substring( 1 , msg.length ) + msg.substring( 0 , 1 ) ;
                document.f.message.value = msg ;
                setTimeout( "scroll( )" , 300 ) ;
        }
}
</script></head>
<body bgcolor="#ffffff0" onload="syokika( )">
<center>
<h1>モグラたたきゲーム</h1>
<p>あなたは何点とれるかな??スタートボタンをクリックして開始しましょう</p>
<script type="text/javascript">
for ( i = 0 ; i <= 4 ; i++ ) {
        document.write( "<img src='images/img3.gif' name='imgmog' border='0' onclick='count(" , i , ")'>") ;
}
</script>
<form name="f">
        <p>得点:<input name="score" size="5" />
        <input type="button" value="スタート" onclick="play( )" /></p>
        <p><input name="message" size="25" /></p>
</form>
</center>
</body>
```





jQuery Lesson 1

1.練習用のhtmlを<body>タグ内に記述します。

html

```
<h1>jQuery Lesson</h1>
<h2>アコーディオン</h2>
<dl>
	<dt>メニューA</dt>
	<dd>ダミーリスト<br />ダミーリスト<br />ダミーリスト<br />ダミーリスト<br />
	ダミーリスト<br />ダミーリスト<br />ダミーリスト<br />ダミーリスト</dd>
	<dt>メニューB</dt>
	<dd>ダミーテキスト<br />ダミーテキスト<br />ダミーテキスト<br />ダミーテキスト<br />
	ダミーテキスト<br />ダミーテキスト<br />ダミーテキスト<br />ダミーテキスト</dd>
	<dt>メニューC</dt>
	<dd>アコーディオンリスト<br />アコーディオンリスト<br />アコーディオンリスト<br />
	アコーディオンリスト<br />アコーディオンリスト<br />アコーディオンリスト<br />
	アコーディオンリスト<br />アコーディオンリスト<br />アコーディオンリスト</dd>
</dl>
```

アコーディオンによる開閉

javascript

```
$( function( ) {
	$( "dd" ).hide( ) ;
	$( "dt , h3" ).click( function( ) {
		$( this ).next( ).slideToggle( ) ;
	} ) ;
} ) ;
```

css

```
#slideShow {
	position: relative ;
	width: 350px ;
	height: 350px ;
	margin: 0 auto ;
	list-style: none ;
}
#slideShow li {
	position: absolute ;
	left: 0 ;
	top: 0 ;
}
```

2.jQueryを利用したスライドショー

html

```
<ul id="slideShow">
    <li><img src="images/fruit_1.jpg" width="350" height="350" alt="オレンジの画像" /></li>
    <li><img src="images/fruit_2.jpg" width="350" height="350" alt="パイナップルの画像" /></li>
    <li><img src="images/fruit_3.jpg" width="350" height="350" alt="バナナの画像" /></li>
    <li><img src="images/fruit_4.jpg" width="350" height="350" alt="メロンの画像" /></li>
    <li><img src="images/fruit_5.jpg" width="350" height="350" alt="ぶどうの画像" /></li>
    <li><img src="images/fruit_6.jpg" width="350" height="350" alt="ももの画像" /></li>
    <li><img src="images/fruit_7.jpg" width="350" height="350" alt="さくらんぼの画像" /></li>
    <li><img src="images/fruit_8.jpg" width="350" height="350" alt="苺の画像" /></li>
    <li><img src="images/fruit_9.jpg" width="350" height="350" alt="ドリアンの画像" /></li>
    <li><img src="images/fruit_10.jpg" width="350" height="350" alt="西瓜の画像" /></li>
</ul>
```

javascript

```
$( function( ) {
    $('#slideShow li:gt(0)').hide( );
    setInterval(function( ){
        $('#slideShow li:first').fadeOut(1500).next( 'li' ).fadeIn(1500).end( ).appendTo( '#slideShow' );
    },4000);
} ) ;
```

3.jQueryを利用したタブ切り替えメニュー

html

```
<h1>タブメニュー</h1>
<ul class="tab">
	<li class="select">タブ1</li>
	<li>タブ2</li>
	<li>タブ3</li>
	<li>タブ4</li>
</ul>
<div class="content">
	<div>
		<h2>タブコンテンツ1</h2>
		<p>タブ1の内容が入ります。タブ1の内容が入ります。<br />
		タブ1の内容が入ります。タブ1の内容が入ります。</p>
	</div>
	<div>
		<h2>タブコンテンツ2</h2>
		<p>タブ2の内容が入ります。タブ2の内容が入ります。<br />
		タブ2の内容が入ります。タブ2の内容が入ります。</p>
	</div>
	<div>
		<h2>タブコンテンツ3</h2>
		<p>タブ3の内容が入ります。タブ3の内容が入ります。<br />
		タブ3の内容が入ります。タブ3の内容が入ります。</p>
	</div>
	<div>
		<h2>タブコンテンツ4</h2>
		<p>タブ4の内容が入ります。タブ4の内容が入ります。<br />
		タブ4の内容が入ります。タブ4の内容が入ります。</p>
	</div>
</div>
```

css

```
li {
	list-style: none ;
}
.tab {
	padding: 0 ;
}
.tab li {
	background: #ccc ;
	padding: 5px 25px ;
	float: left ;
	margin-right: 1px ;
	cursor: pointer ;
}
.tab li:hover {
	background: #ddd ;
}
.tab li.select {
	background: #eee ;
}
.content {
	clear: both ;
}
.content div {
	background: #eee ;
	padding: 20px ;
}
```

javascript

```
$( function() {
	$( ".content div" ).hide() ;
	$( ".content div:first-child" ).show() ;

	$( ".tab li" ).click( function() {
		var num = $( ".tab li" ).index( this ) ;
		$( ".content div" ).hide() ;
		$( ".content div" ).eq( num ).show() ;
		$( ".tab li" ).removeClass( "select" ) ;
		$( this ).addClass( "select" ) ;
	}) ;
}) ;
```

クリック時の動作を function にまとめて指定
.index()を使いクリックされたタブが何番目かを取得後変数に代入
コンテンツ内容を一度すべて非表示
クリックされたタブと同じ順番のコンテンツを表示
一度タブについているクラスselectを消去
クリックされたタブのみにクラスselectを設定

※jQueryは、html と css と javascript を組み合わせて動作させる事が基本になります。
その為、それぞれを上手く連携させて利用していく必要があります。(基本的な html と css の理解が必要)
見た目や配置をcssで設定するのですが、その css の設定も css ファイルから行うのか、jQuery から側から行うのかで意味が少し変わってきます。基本的に javascript をオフにしても、見れなくなってしまうコンテンツ内容があってはいけません。
(javascript に依存した一部情報が、javascript オフ時には非表示になってしまう状態が不可という事になります。)
動作確認時、上記の内容に気を付けて javascript や jQuery の組み込みを行いましょう。
Firefox用のお勧めAdd-on [QuickJava] をインストールすると、css / flash / javascript の切替が簡単に出来る様になります。