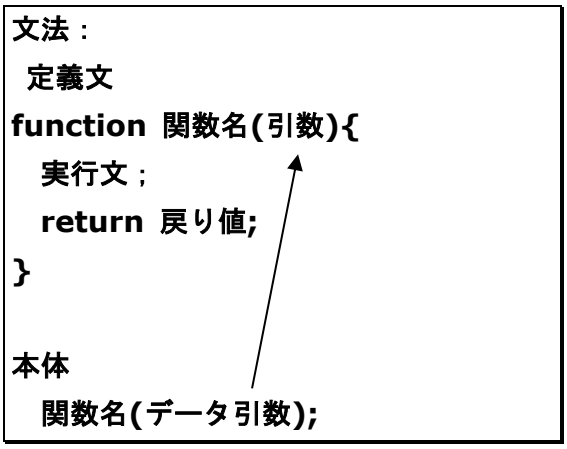
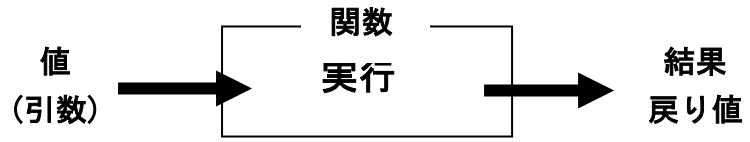
***function* 関数**

プログラミングでは「ある処理をするひとかたまりのもの」に名前を付けたものを「関数」と呼ぶ。本来は、function を訳した「機能」が正しいのだろうが、なぜか、誤訳のまま通ってしまっている。

JavaScript では、関数を自分で定義して使うことができる。関数の中には、数学的なものもあるが、文字列を処理することもできる。関数の定義は、<HEAD>の部分で定義され、＜BODY＞の中で実行する。



関数の中には、引き渡されたデータ（引数（ひきすう））に対して結果を返すものもある。これが return である。この際に、返される値を「戻り値」と呼ぶ。何も返さない return もある。【PR17】



JavaScript の関数は、同じ名前で定義してもエラーにならない。あとから定義された名前の方が、有効な関数名になる。

プログラム１：ドルを円に換算する関数。

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT LANGUAGE="JavaScript">
 function exchange(doll, rate){   // ドルを円に換算する関数の定義
  return doll*rate;
 }
</SCRIPT>
</HEAD>
<BODY>
```

```
<SCRIPT LANGUAGE="JavaScript">
var jRate=112;  //  レートはここに入れる
var oneDoll=280;  // ドルの値をここに入れる
 document.write(oneDoll,"ドルは、",exchange(oneDoll,jRate),"円です。<BR>");
</SCRIPT>
</BODY>
</HTML>
```

(解説)
定義した関数「exchange」は、「doll」と「rate」の 2 つの値を本体から渡され、この 2 つを掛けたもの「doll*rate」を本体へ返している。
本体では、exchange(oneDoll,jRate)で、2 つの値を渡し、結果を同じ場所に受け取る。

コンピュータの世界では、掛け算を「*（アスタリスク)」としてあらわす。

var は、「変数」を定義するときに使われる用語です。【PR37】
JavaScript の変数の定義は、比較的緩やからルールがとられています。そのため、var は、必ずしも必要ということはありません。

プログラム 2 :「文字列 ABCDEFG をスペースでずらしながら表示する。」
これまでの応用として、次のサンプルを示します。
f unction では、スペースを作る関数「spc()」を定義しています。

```
<HTML>
<HEAD>
<TITLE>use function</TITLE>
<SCRIPT LANGUAGE="JavaScript">
  function spc(n){  // スペースを入れる関数の定義
     s="";
     for (i=0; i<n; i++)
        s=s+"  ";  //  ‘’ と ‘’ の間がスペース
     return s;   //  戻り値「s」の実体は、スペース
  }
</SCRIPT>
</HEAD>

<BODY>
<!- -このタグは、改行やスペース、空白などがそのまま表示される。-->
```

```
<PRE>
<SCRIPT LANGUAGE="JavaScript">
   for (i=0; i<=8; i++){
      document.write(spc(i)+"ABCDEFG<BR>");
   }
   for (i=8; i>=0; i--){
      document.write(spc(i)+"ABCDEFG<BR>");
   }
</SCRIPT>
</PRE>
</BODY>
</HTML>
```

(解説)

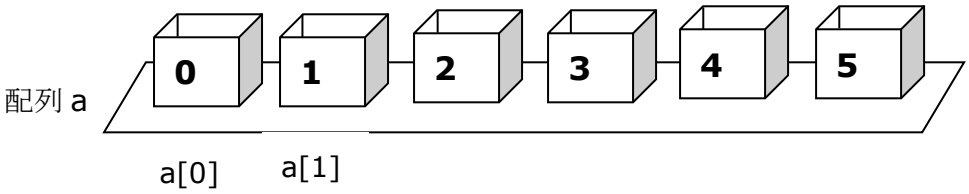
スペースを入れる関数「spc()」をヘッダーの部分で宣言する。
タグ＜PRE＞を使わないと、そのままの形では、表示されない。

5.Array 配列　【PR86】

配列は、変数をいくつも使ったもの。
連続して用意された「番号が書かれた箱」のようなイメージ。その箱の、番号を指定することで、中に入っている要素を取り出したり、しまったりできる。

文法：
配列名[添字]

例：
a[i]



サンプル：0 から 4 までの５つの箱を用意し、それぞれの箱の中に、同じく 0 から４までの数字が入れられる。

```
for(i=0; i<5; i++)
  a[i]=i;
```

この場合の配列の名前は「a」。
ふつう、配列の先頭は、0 から始まる。

他の言語と比較すれば、JavaScript の「配列」の書法は、非常にゆるやかである。

5-1.配列の宣言

配列を使うためには、配列を「宣言」しなくてはならない。
配列の宣言は、new でおこなう。*

文法：
配列名＝new Array(配列数)；

たとえば、

```
var abc=new Array(5);
```

と、すれば、

配列名： abc
配列数： ６ の配列が作られる。

* new については、日付けオブジェクトで詳しく扱う。

5-2. 配列への直接代入

配列に、要素を直接代入する方法。

alph＝new Array（”A”、”B”、”C”）；

プログラム１：「文字の色を配列に貯えて、表示する」

```
<HTML>
<HEAD>
<TITLE>Array</TITLE>
</HEAD>
<BODY>
<SCRIPT LANGUAGE="JavaScript">
   var col=new Array(6);   // 配列 col を作る
   col[1]="0000FF";col[2]="00FF00";col[3]="00FFFF";
                                  //配列 col に、要素を代入する。
   col[4]="FF0000";col[5]="FF00FF";col[6]="FFFF00";
   for (i=1; i<=6; i++){
      document.write("<FONT COLOR=#" +col[i]+">");
      document.write("ABCDEFG<BR>");
      document.write("</FONT>");
   }
</SCRIPT>
</BODY>
</HTML>
```

プログラム２：配列を使った「住所録」

```
<HTML>
<HEAD>
<TITLE>配列を使った住所</TITLE>
<SCRIPT LANGUAGE="JavaScript">
   function Array(n){
      this.length=n;
   }
</SCRIPT>
</HEAD>
<BODY>
<SCRIPT LANGUAGE="JavaScript">
var cell=new Array(6);
```

```
cell[1]="名前";cell[2]="電話";cell[3]="住所";
cell[4]="田中";cell[5]="564-5678";cell[6]="名古屋";
n=1;
document.write("<TABLE BORDER>");
for (i=0; i<2; i++){
   document.write("<TR>");
   for(j=0; j<3; j++){
      document.write("<TD>"+cell[n]+"</TD>");[1]
      n++;
   }
   document.write("</TR>");
}
document.write("</TABLE>");
</SCRIPT>
</BODY>
</HTML>
```

[1] JavaScript では、2 次元配列は扱えないので、n を添字とした 1 次元配列として考える。