文字列を数値に変換する

ここから JavaScript で数字と文字を扱うスクリプトについて見ていきます。 その最初に文字列を数字に変換する方法について考えます。 これは例えばフォームに記入された数字を使って計算する場合などに必要となります。 なぜならフォームに記入されたものは文字列なので、そのままでは計算できないからです。

文字列を数字に変換する

以下のスクリプトを HTML の BODY 内に記入してみましょう。 なお以下のフォームの意味が分からない方は、 当コンテンツのフォーム関係をご覧になってみて下さい。

```
<form name="keisan">
<input type="text" value="3">＋
<input type="text" value="5"><br>
<input type="button" value="計算" onclick="tasu()">
</form>

<script>

function tasu()
{
 //テキストボックスの数値を変数に格納する
 var num1=document.keisan.elements[0].value;
 var num2=document.keisan.elements[1].value;

 //計算結果をアラートで表示
 alert(num1+num2);
}
</script>
```

サンプルのボタンを押すと、計算結果の「8」ではなく、 「35」と表示されると思います。 なぜならテキストボックスに記入されている数字は実際には文字列なので、 「3」という文字と「5」という文字が連結されて表示される訳です。

では数値として認識してもらうために、 変数に格納する部分を以下のように書き足してみてください。

```
 var num1=parseInt(document.keisan.elements[0].value);
 var num2=parseInt(document.keisan.elements[1].value);
```

今度はきちんと計算結果「8」が表示されると思います。 テキストボックスの数字を色々と変えて試してみて下さい。

上記スクリプトにあるように、 parseInt()が文字列を整数に変換する JavaScript の命令文

です。parseFloat()を使えば文字列を少数に変換します。

parseInt(文字列)

文字列を整数に変換する

parseFloat(文字列)

文字列を少数に変換する

エラーが出たときの処理

エラーが出るとスクリプトがそこで止まってしまいます。 それを回避するための方法を以下に記しておきます。

```
try
{
  エラーが出そうな処理を記述
}
catch(e)
{
  エラーが出た時の処理を記述
}
```

try の中括弧内に書かれたスクリプトにエラーが出た場合、 catch の中括弧内の処理を行います。 catch にはエラーの内容を格納する引数が必要ですが、 通常はあまり意識する必要はありません。上記のように「e」を入れておきます。

try{ 処理 }catch(e){ エラーが出た時の処理 }

try のカッコ内の処理でエラーが出た場合、catch()の中カッコ内処理を行う

べき乗・平方根・円周率

このページでは JavaScript でべき乗や平方根、円周率を求める方法について解説します。

べき乗を求める

以下は、ある数字のべき乗を求めるスクリプトです。HTML の BODY 内に記入してみてください。

```
<form name="bekijo">
<input type="text" value="5">基数<br>
<input type="text" value="2">指数<br>
<input type="button" value="計算" onclick="beki()">
</form>

<script>

function beki(){
```

```
  //基数と指数をそれぞれ数字に変換して変数に格納
  var num1=parseInt(document.bekijo.elements[0].value);
  var num2=parseInt(document.bekijo.elements[1].value);

  //べき乗を求めてアラートで表示
  alert( Math.pow(num1,num2) );

}

</script>
```

前のページで見たようにフォームに入力された数値は実際には文字列なので、 parseInt()を使って数字に変換し、変数 num1,num2 に格納しています。
続いて、べき乗を計算して alert()で表示します。Math.pow()がべき乗を求める命令文です。「M」が大文字なので注意してください。 カッコ内の最初が基数、コンマで区切って 2 番目に指数を指定します。

Math.pow(基数,指数)
べき乗を求める

平方根を求める

続いて平方根を求める方法についても見てみましょう。 HTML の BODY 内に以下のフォームとスクリプトを記入してみてください。

```
<form name="ROOT">
<input type="text" value="2">
<input type="button" value="平方根" onclick="root()">
</form>

<script>

function root(){

 //数字に変換して変数に格納
 var num1=parseInt(document.ROOT.elements[0].value);

 //平方根を求めてアラート表示
 alert( Math.sqrt(num1) );
}
```

```
</script>
```

フォームのデータを数値変換してから変数に格納するのは今までと同じです。 そしてMath.sqrt()を使って平方根を求め、 alert()で表示しています。

Math.sqrt(数値)

平方根を求める

円周率を求める

このページ最後は円周率を求める方法です。 以下のフォームと JavaScript を HTML の BODY 内に記入してみましょう。

```
<form>
<input type="button" value="円周率" onclick="pai()">
</form>

<script>

function pai(){

 //円周率を表示
 alert( Math.PI );

}

</script>
```

今回はフォームデータを利用しないので、ボタン１つだけ設置して計算結果を表示させます。 円周率を求めるのは、Math.PI です。 それを alert()で表示します。

Math.PI

円周率を求める

これで円柱や円錐の体積などを求めるスクリプトも組めると思います。 余力のある方は是非チャレンジしてみてください。

四捨五入・切り上げ・切り捨て

このページでは、数字を四捨五入や切り上げ、切り捨てをして整数にまとめる方法について解説します。 JavaScript ではいろいろな場面で大変よく使います。記述自体は簡単なので、しっかり覚えて下さい。

さらにこのページでは、整数以外に丸める方法についても考えます。 小数二桁などに丸めたいこともあるでしょう。 金額などを扱う場合は千円単位、万円単位で丸めたいこともあると思います。

実際丸める桁数を変えることはできないのですが、ほんのちょっとした工夫で可能になり

ます。 その方法についても見ることにしましょう。
四捨五入・切り上げ・切り捨て
では以下のスクリプトを HTML の BODY 内に記入してみてください。

```
<form name="marume">
<input type="text" value="3.14159">：丸める数字<br>
<input type="text" value="">：四捨五入<br>
<input type="text" value="">：切り上げ<br>
<input type="text" value="">：切り捨て<br>
<input type="button" value="計算" onclick="keisan()">
</form>

<script>

function keisan(){

  //最初のテキストボックスの値を数字に変換
  var num = parseFloat(document.marume.elements[0].value);

  //順に、四捨五入・切り上げ・切り捨てを求める
  document.marume.elements[1].value = Math.round(num);
  document.marume.elements[2].value = Math.ceil(num);
  document.marume.elements[3].value = Math.floor(num);
}
</script>
```

関数 keisan()の最初に、 「丸める数字」テキストボックスの文字列を parseFloat()を使って数字に変換し、変数 num に代入します。
続いてテキストボックスの 2 番目以降に、 それぞれ四捨五入・切り上げ・切り捨てた値を計算して表示しています。
Math.round(数字)
数字を四捨五入して整数に丸めます
Math.ceil(数字)
数字を切り上げして整数にします
Math.floor(数字)
小数以下を切り捨てして整数にします
整数以外に丸めたい場合
上記の場合は整数に丸めましたが、小数〇〇桁に丸めたいと言うこともあると思います。

また金額を計算するときは千円とか 1 万円の単位で丸めたいこともあります。 そんな時は小数点の位置を移動させて数字を丸め、その後元に戻します。
以下のスクリプトを HTML の BODY 内に記入してみてください。

```
<form name="marume2">
<input type="text" value="12345.6789">：丸める数字<br>
<input type="text" value="">：小数 2 桁に丸める<br>
<input type="text" value="">：1000 単位に丸める<br>
<input type="button" value="計算" onclick="keisan2()">
</form>


<script>

function keisan2(){

  //最初のテキストボックスの値を数字に変換
  var num = parseFloat(document.marume2.elements[0].value);

  //小数点の位置を 2 桁右に移動する（1234567.89 にする）
  var num1 = num * 100;

  //四捨五入したあと、小数点の位置を元に戻す
  num1 = Math.round(num1) / 100;

  //2 番目のテキストボックスに表示する
  document.marume2.elements[1].value = num1;


  //小数点の位置を左に 3 桁移動して計算し、元に戻す
  var num2 = num / 1000;
  num2 = Math.round(num2) * 1000;
  document.marume2.elements[2].value = num2;
}

</script>
```

四捨五入や切り上げ、切り捨ては JavaScript においても大変よく使います。 しっかりマ

スターしてください。

乱数を扱う

このページでは、JavaScript で乱数を発生させる方法、またその乱数を扱う方法についてみていきます。 乱数はゲーム等を作る時に必要となる場合がよくあります。

乱数を発生させる

JavaScript の乱数は、0 以上 1 未満の範囲で取得できます。 例えば以下のスクリプトを実行したサンプルページをご覧下さい。

```
<script>

for (i=0 ; i < 20 ; i++)
{
   document.write( Math.random() + "<br>");
}

</script>
```

for 文を使って、乱数を 20 回発生させています。 そして document.write()で書き出しました。0～1 の範囲で乱数が発生しているのが分かります。 上記のように、Math.random()が乱数を発生させる命令文です。random の括弧内は何も記入しません。

Math.random()

0 以上 1 未満の範囲で乱数を発生させる

ただし、上記サンプルのような乱数では、扱いにくいと思います。 それで、具体例を用いて乱数の実際的な扱い方について見てみることにしましょう。

乱数を使ったサンプルスクリプト

では英単語の色名を当てる簡単なクイズを作ってみましょう。以下のスクリプトを HTML の BODY 内に記入して下さい。 また body タグの中には、下記のように onload イベントを記入してください。 onload イベントは、ページが読み込まれた時に実行されます。

```
<body onload="mondai()">
<form name="quiz">
<input type="text" value="">
<input type="button" value="赤" onclick="push(0)">
<input type="button" value="青" onclick="push(1)">
<input type="button" value="黄" onclick="push(2)">
<input type="button" value="緑" onclick="push(3)">
<input type="button" value="白" onclick="push(4)">
</form>
```

```
<script>

//色名の英単語を配列に入れる
var col = new Array("red","blue","yellow","green","white");

//乱数を入れる変数
var rnd;

//テキストボックスに問題文（色名）を表示する関数
function mondai(){

  //0～4 までの乱数を発生させる
  rnd = Math.floor( Math.random() * 5 );
  document.quiz.elements[0].value = col[rnd];
}

//正誤判定関数
function push(num){
  //引数を数字に変換
  var n = parseInt(num);

  //正解なら次の問題を表示、間違っていたらアラートを表示する
  if ( n == rnd ){
     mondai();
  }else{
     alert( "違います！" );
  }
}

</script>
```

ではスクリプトを見ていきましょう。フォームのボタンの onclick イベントでは、関数 push()を呼び出すようにしています。引数は、色名を入れた配列 col の対応する数字を記入しています。

そして、乱数を入れる変数 rnd を宣言しています。 上記のように関数の外で宣言された変数は、 グローバル変数と言って複数の関数で使うことができます。 （関数の中で宣言され

た変数はその関数の中でしか使えません)。
続いて、関数 mondai()を見ていきましょう。最初に乱数を 1 回発生させ、 0～4 の範囲の整数に整えて変数 rnd に代入します。 上で述べたように Math.random()は 0 以上 1 未満の乱数なので、 5 倍して切捨てを行うことにより 0～4 の範囲の整数が得られます。 そして得られた乱数を使って、配列 col の対応カラー名をテキストボックスに表示します。
今度は関数 push()です。 最初に引数を parseInt()を使って数字に変換しています。 元々引数を数字で指定しているのですが、残念ながらうまく動きません。 このようなこともあるので、とりあえず変換しています。
そして if 文を使って、 引数（を数字に変換した変数 n）と変数 rnd を比較します。 （ここでも変数 rnd を使っているので、グローバル変数にした訳です)。 正解なら次の問題を表示するため、関数 mondai()を呼び出します。 間違っていたら alert()を使って間違いを指摘します。
配列をシャッフルする
ここでは、乱数を使って配列の要素を入れ替える方法について解説します。 問題文をランダムに出題したい場合などに使うことができます。
前のページでも、問題文（カラー名の英単語）をランダムに表示する方法を見ましたが、 同じ問題文が何度も続けて表示されることもあったと思います。 それを避けるために、配列そのものをシャッフルして、 その後その配列を順に表示すればダブることがありません。
配列をシャッフルする
配列をシャッフルする方法ですが、一番分かり易いのは以下の方法でしょう。
まず乱数を使って、配列のどれか 1 つの要素を決めます
それを配列の 1 番最初の要素と入れ替えます
これを繰り返すことでシャッフルできます
では、以下のスクリプトを HTML の BODY 内に記入してみて下さい。

```
<form name="quiz">
<input type="text" value="">
<input type="button" value="出題" onclick="mondai()">
</form>

<script>

//カラー名を格納する配列 col の作成
var col = new Array("red","blue","yellow","green","white");

//配列の大きさに合わせて適当な回数繰り返す
for ( i = 0 ; i < 20 ; i++ )
```

```
{
  //0～4 までの乱数を作成し、変数 Rnd に格納する
  var rnd = Math.floor(Math.random()*5);

  var str1 = col[0];      //配列 col の最初の要素
  var str2 = col[rnd];   //配列 col の乱数で決定した要素

  //配列の各要素を入れ替える
  col[rnd] = str1;
  col[0]    = str2;
}

var cnt = 0; //何問出題したかカウントする変数

function mondai(){
  if( cnt < 5 )
  {
    document.quiz.elements[0].value = col[cnt];
    cnt++;
  }else{
    //全問（5 問）終了したら、そのことを表示してボタンを押せなくする
    document.quiz.elements[0].value = "全問終了";
    document.quiz.elements[1].disabled = true;
  }
}

</script>
```

シャッフルスクリプトの解説

では上記スクリプトを詳しく見ていきたいと思います。最初のカラー名を格納する配列 col は、前ページと同じにしました。

続いて for 文を使い、配列 col をシャッフルします。 配列の要素によって繰り返し回数を調整することができます。 今回は 20 回シャッフルしています。

for 文の中で最初に、0～4 までの整数の乱数を作成し、変数 rnd に代入します。

次に、配列の最初の要素と、乱数で決定した要素をそれぞれ変数 str1,str2 に代入します。そして、各要素を入れ替えます。下のリンクをクリックすると、 どのようにシャッフルされ

ているか確認できます。

（確認ページを見ると分かるように、変数 rnd が 0 の時はシャッフルされません。 従って乱数を 1～4 までの範囲にするか、0 の時は最後の要素と入れ替える等の処理をすると、 効率よくシャッフルできます。色々と工夫してみてください)。
今度は問題を出題する部分を見ていきます。 最初に出題した回数を格納するグローバル変数 cnt を宣言し、初期値 0 を代入しておきます。
次に関数 modai()の中で、最初に出題数を if 文でチェックします。 5 問未満であれば、問題を表示します。そして出題数の変数 cnt を 1 増加させます。5 問全部表示していたら、テキストボックスに「全問終了」と表示して、ボタンを禁止状態にします。
これでシャッフルの説明は終わりですが、 配列自体をシャッフルしているので前のページのような正誤判定はできません。 幾つか方法がありますが、 例えば以下のように正解の文字列を格納した配列をもう一つ作っておくと、 それを使って判断できます。

```
//正解の文字列を入れた配列をもう一つ作成
var seikai = new Array();
seikai = new Array("red","blue","yellow","green","white");

//配列 col だけをシャッフル（上述）

//正誤判定は、テキストボックスの文字列と seikai[]を比較
function push(num){
  num = parseInt(num);
  if( document.quiz.elements[0].value == seikai[num] )
  {
     //正解の処理
  }
}
```

2 進数・16 進数・10 進数
このページでは 10 進数の数字を 2 進数や 16 進数に変換する方法について考えます。 また変換した数値を 10 進数に戻す方法も扱います。
これは数学や科学の分野でしか使わないように感じるかもしれません。 しかしゲームのフラグを記録したり、暗号化するのに用いることもできます。 記述も簡単なので、いろいろと工夫して使ってみましょう。
n 進数への変換
では、以下のスクリプトを HTML の BODY 内に記入してみましょう。

```
<form name="math">
<input type="text" name="" value="255">：元の数字<br>
<input type="text" name="" value="">：2 進数<br>
<input type="text" name="" value="">：16 進数<br>
<input type="button" value="変換" onclick="sinsu()"><br>
</form>


<script>

function sinsu(){

  //「元の数字」欄の文字を数字に変換
  var suji = parseInt(document.math.elements[0].value);

  //2 進数に変換して 2 番目のテキストボックスに表示
  document.math.elements[1].value = suji.toString(2);

  //16 進数に変換して 3 番目のテキストボックスに表示
  document.math.elements[2].value = suji.toString(16);
}
</script>
```

上記スクリプトの解説です。 最初に、1 番目のテキストボックスに記入された数値（実際には文字列）を、parseInt()を使って数値に変換し、変数 suji に代入しています。
続いて 2 進数・16 進数に変換し、2 番目と 3 番目のテキストボックスに表示します。 上記スクリプトを見ると分かるように、toString()が n 進数に変換する命令文です。 括弧内に基数を記入します（基数は 2～36 まで使えるようです）。 toString()の前に、 数値もしくは数値が入った変数を記入し、ピリオドで繋げます。

数値.toString(基数)

数値を n 進数に変換する

10 進数に戻す

さて、2 進数や 16 進数の数値を 10 進数に戻すにはどうすればよいでしょうか？ 2 進数の数字を 10 進数に戻す方法を見てみましょう。 以下のスクリプトを HTML の BODY 内に記入してみてください。

```
<form name="math2">
```

```
<input type="text" value="11111111">：2 進数<br>
<input type="text" value="">：10 進数<br>
<input type="button" value="10 進数に変換" onclick="jusinsu()">
</form>

<script>

function jusinsu(){

  //最初のテキストボックスの文字列を変数 suji2 に格納
  var suji2 = document.math2.elements[0].value;

  //10 進数にして 2 番目のテキストボックスに表示
  document.math2.elements[1].value = parseInt(suji2,2);

}

</script>
```

10 進数に戻すには、parseInt()を使います。 括弧内の最初の要素に n 進数の文字列を、 2 番目に基数を記入します。

parseInt(n 進数文字列,基数)

n 進数の数値を 10 進数に変換する

たとえばアドベンチャーゲームなどでは、沢山のフラグを立てます。 フラグとは誰々に会ったか会っていないか、あるアイテムを持っているか持っていないか等々、 様々な要素のことで､これによって物語を分岐させます。このフラグを 0 と 1 の数値で記録していけば、保存する時に各フラグを文字として結び付けることで 2 進数の数値文字列ができます。 これを 10 進数、さらに n 進数等に変換しておけば、記録データを覗かれても意味が分かりにくくなるでしょう。

文字の装飾

ここから、文字列に関する操作を見ていきたいと思います。 最初は、簡単に文字列を装飾する方法です。 HTML タグを組み込んでも文字を装飾できますが、 JavaScript でも簡単な装飾ならできるので、その方法を見てみることにしましょう。

文字を装飾するスクリプト

では以下のスクリプトを、HTML の BODY 内に記入してみましょう。

```
<script>
```

```
document.write("文字を装飾します<br>");
document.write("大きくする".big()+"<br>");
document.write("小さくする".small()+"<br>");
document.write("太字にする".bold()+"<br>");
document.write("斜体にする".italics()+"<br>");
document.write("取り消し線を付ける".strike()+"<br>");
document.write("固定幅フォントにする".fixed()+"<br>");
document.write("下付き文字にする"+"2".sub()+"<br>");
document.write("上付き文字にする"+"2".sup()+"<br>");
document.write("文字色を指定する".fontcolor("red")+"<br>");

</script>
```

文字を装飾するには、上記のように文字列の後ろにピリオドを付け、 装飾に関する命令文を繋げます。 ただしこのページではフォントを「メイリオ」にしているので、 Windows ユーザーの方の多くは「斜体」の部分は傾いていないかも知れません（現時点でメイリオに斜体は無いので)。

文字列.big()
文字を大きくする
文字列.small()
文字を小さくする
文字列.bold()
太字にする
文字列.italics()
斜体にする
文字列.strike()
取り消し線にする
文字列.fixed()
固定幅フォントにする
文字列.sub()
下付き文字にする
文字列.sup()
上付き文字にする
文字列.fontcolor(カラー名)
文字色を指定する

なお、下線を引く命令文は調べてみたのですが、無さそうです。 下線を引くにはまた別の

方法を用います。 まあスタイルシートで指定すれば済むことですが（苦笑）
複数の装飾を指定する
文字列に複数の装飾を施すこともできます。 以下のスクリプトを HTML の BODY 内に記入してみてください。

```
<script>

var str = "複数の装飾";

document.write(str.fontcolor("green").big().bold());
document.write("を施した文字列");

</script>
```

上述のスクリプトのように、装飾命令を複数繋げて指定すれば OK です。 なお上記のように変数に格納された文字列も装飾可能です。
文字を抜き出す
このページでは、文字列の中から文字を抜き出す 3 種類の方法について考えます。 指定位置から 1 文字だけ抜き出す、文字数を指定して抜き出す、範囲指定して抜き出す方法です。

文字を抜き出すスクリプト
文字を切り出す簡単なスクリプトを見てみましょう。 以下のコードを HTML の BODY 内に記入してみてください。

```
<script>

//元の文字列
var str = "あいうえおかきくけこさしすせそ";

//1 文字切り出す
document.write(str.charAt(0) + "<br>");

//文字数を指定して抜き出す
document.write(str.substr(1,4) + "<br>");

//指定範囲の文字を切り出す
document.write(str.substring(5,10) + "<br>");
```

</script>
まずは変数 str に切り出す元となる文字列を代入しています。 文字の位置は先頭が 0 になります （変数 str の場合「あ」は 0,「か」は 5 になります)。
そして 3 種類の方法で切り出しています。一文字だけ切り出す場合は charAt()を使います。括弧内は切り出す位置です。 上の例では 0 が入っているので、「あ」の文字が表示されます。
指定位置から文字を切り出す場合は substr()を使います。括弧内は最初が切り出し位置、2 番目に切り出す文字数を指定します。 上記の例では「い」から 4 文字切り出します。 結果は「いうえお」となります。
最後は指定範囲の文字を切り出す方法で、substring()を使います。 括弧内は切り出す文字の開始位置、終了位置です。 上記の場合、開始位置は「か」、終了位置は「さ」の‘前’で、 結果は「かきくけこ」になります。
文字列.charAt(開始位置)
開始位置の文字を 1 文字抜き出す
文字列.substr(開始位置,文字数)
開始位置から指定文字数だけ抜き出します
文字列.substring(開始位置,終了位置)
開始位置～終了位置の間の文字を抜き出す
2 進数の数値文字列を配列に格納する
前の前のページで、ゲームのデータを 2 進数にして保存できることを記しました。 では実際に保存データをロードし、ゲームで使えるようにするにはどうしたら良いでしょうか？文字の切抜きを使い、配列に格納すればＯＫです。

```
<script>

//2 進数文字列
var data = "1010101";

//データを格納する配列を作成する
var flg = new Array();

//文字列変数 data の文字数をカウントし、変数 cnt に格納する
var cnt = data.length;

//for 文を使って配列に格納する
for ( var i = 0 ; i < cnt ; i++ ){
```

```
  flg[i] = data.charAt(i);

}

</script>
```

上の例では､｢1010101｣という 2 進数のデータを配列 flg に格納しています。for 文を使い、前から 1 文字ずつ抜き出して配列に格納していきます。 for 文の場合範囲を指定する必要があるので、文字数を取得しています。 文字数は length を使って取得できます。

文字列.length

文字列の文字数をカウントします。

length で取得した文字数は実際の文字の数ですが、 charAt()は最初の文字を 0 番としてカウントするので、 最後の文字は length で取得した数値より 1 つ少なくなります。 for 文の「i」の範囲を変数 cnt 未満にしているのはそういう理由です。

あとはゲームの各場面で、○○○に会ったなら flg[1]=1、 あるアイテムを拾ったなら flg[2]=1、 という感じでフラグを立てていけば、物語を分岐させることができます。

文字列を分割する

このページでは、文字列を任意の文字で分割する方法について解説します。 これは配列を文字列として保存し、また読み込んで配列に戻す時などに使います。

文字列分割のスクリプト

例えば、以下のような商品データがあるとします。

商品名	種類	価格
メロンパン	菓子パン	130 円
あんぱん	菓子パン	100 円
ウィンナーロール	惣菜パン	170 円
ハムエッグ	惣菜パン	140 円
ミックスサンド	サンドウィッチ	200 円

これを、パンの名称だけ表示しておいて、 お客さんが選択したら価格などを表示するようにしてみたいと思います。 以下のスクリプトを HTML の BODY 内に記入してみてください。

```
<form name="pan">
<select name="sentaku" onchange="choice()">
<option>商品を選択してください</option>
```

```
<option>メロンパン</option>
<option>あんぱん</option>
<option>ウィンナーロール</option>
<option>ハムエッグ</option>
<option>ミックスサンド</option>
</select>
<br>
商品：<input type="text" name="item" value=""><br>
種類：<input type="text" name="type" value=""><br>
価格：<input type="text" name="price" value=""><br>
</form>


<script>

//商品データを配列 bread に入れる
var bread=new Array(
"",
"メロンパン/菓子パン/130 円",
"あんぱん/菓子パン/100 円",
"ウィンナーロール/惣菜パン/170 円",
"ハムエッグ/惣菜パン/140 円",
"ミックスサンド/サンドウィッチ/200 円"
);

function choice()
{

  //データを分割して格納する配列 pain の作成
  var pain = new Array();

  //セレクトボックスの最初のオプションを選択したか確認
  if ( document.pan.sentaku.selectedIndex != 0 )
  {
    //選択された商品データを変数 str に入れる
    var str = bread[document.pan.sentaku.selectedIndex];
```

```
    //商品データの各要素を「/」で分離し、配列 pain に格納する
    pain = str.split("/");

    //テキストボックスの該当箇所にデータを表示する
    document.pan.Item.value=pain[0];
    document.pan.Type.value=pain[1];
    document.pan.Price.value=pain[2];
  }
  else
  {
    //「商品を選択してください」を選んだ時の処理
    for ( var i = 1; i <= 3 ; i++)
    {
      document.pan.elements[i].value = "";
    }
  }
}

</script>
```

文字分割スクリプトの説明

では上記スクリプトについて見ていきましょう。フォーム名は「pan」， セレクトボックス名は「sentaku」， データを表示するテキストボックスは順に「item」，「type」，「price」としました。

続いて JavaScript 部分を見てみます。最初に配列 bread を作成して、 商品データを格納します。商品名，種類，価格の間は「/」で区切っておきます。 セレクトボックスの各 option に対応させるため、最初は空データを入れておきました。

次に、セレクトボックスが変更された時に呼び出される関数 choice()を記入していきます。最初に、それぞれのデータから商品名・種類・価格を分割して格納する配列 pain を作成しました （pain はフランス語でパン）。

続いて if 文で、セレクトボックスの「商品を選択してください」が選択されているかどうかを調べます。 それ以外（つまり商品のどれか）が選択されていたら、データ表示処理にかかります。 まずは変数 str に、選択されたパンのデータを格納しています。

ここで、文字の分割を行います。 split(指定文字列）が文字を分割する命令文です。 今回は「/」でデータを区切っているので、split の括弧内に「/」を記入します。 分割された文

字は配列に格納されます。上記の場合、pain[0]は商品名， pain[1]は種類，pain[2]は価格が入ります。それを各テキストボックスに表示します。

配列 = 文字列.split(分割文字列)

任意の文字で、指定文字列を分割して配列に格納します

else 文の中では、各テキストボックスを空にする処理を記入しています。 データを表示するときはテキストボックス名を使いましたが､こちらでは elements[]を使ってみました。セレクトボックスが elements[0]なので、テキストボックスは 1～3 の範囲ということになります。 for 文を使って一気に空文字を代入します。

文字を検索する

このページでは、文字列の中から任意の文字を検索する方法について解説します。 と言っても既に何度か検索命令文 indexOf()は出てきたので、 これを使って文字列置換を行ってみたいと思います。

indexOf()による文字の検索と置換

JavaScript を使って長い文章を表示する場合、または複雑な HTML タグを用いる場合、タグを何かの文字に置き換えておいて、 表示する時にタグに戻すとスッキリします。 以下のスクリプトは強調タグを「##」「%%」に置き換え、document.write()で表示する前にタグに戻しています。

```
<script>

//長い文字列を配列に格納
var str = new Array(
"文字を検索するには、##indexOf()%%を使います。",
"括弧内には##検索文字列%%と、##検索開始位置%%を指定します。",
"検索文字が存在すると、##文字の位置%%を返します。",
"検索文字が存在しないと、##-1%%を返します。"
);

//配列の文字列を順番に表示
for ( var i = 0 ; i < 4 ; i++ )
{
  //while 文を使って無限ループにする
  while(true)
  {
    //「##」を検索し、文字位置を変数 num に代入
    var num = str[i].indexOf("##",0);
```

```
    //もし「##」が無ければ、無限ループから出る
    if (num == -1) break;

    var tag = "<strong>";

    //「##」の前まで切り取る
    var str1 = str[i].substring(0,num);

    //「##」の後ろを切り取る
    var str2 = str[i].substring(num+2,str[i].length);

    //「##」を変数 tag に代入した HTML タグに置き換える
    str[i] = str1 + tag + str2;
  }

  //終了タグに関しても同様の処理をする
  while(true)
  {
    var num = str[i].indexOf("%%",0);
    if (num == -1) break;
    var tag = "</strong>";
    var str1 = str[i].substring(0,num);
    var str2 = str[i].substring(num+2,str[i].length);
    str[i] = str1 + tag + str2;

  }
  //HTML に置き換えられた文字列を表示
  document.write(str[i]);
}

</script>
```

スクリプトの解説

では上記のソースについて見ていきましょう。 最初に配列 str に表示する文字列を記入しています。 開始タグの位置に「##」、終了タグの位置に「%%」を記入します。

続いて for 文を使って、配列の文字列を順に表示していきます。 しかし document.write()

で表示する前に、HTML タグを組み込む処理を行います。
while 文を使った文字の置換処理部分について見て下さい。 while 文を使うのは、str[i]の中に何回「##」が出てくるか分からないからです。 while()の括弧内は繰り返し条件を記入するのですが、今回は true としました。 つまり無限に繰り返すように指定した訳です。
while 文の中で、最初に「##」の位置を調べます。 ここで使うのが indexOf()ですが、おさらいしておきましょう。
文字列.indexOf(検索文字,検索開始位置)
文字列の中から特定の文字を検索し、存在すればその位置を取得します
「##」の位置を変数 num に格納します。 もし存在しなければ変数 num には-1 が入るので、 if 文で「##」が存在しているかどうか調べます。 もし無いのであれば、変換処理が終了しているということです。break を使って無限ループから脱出します。 この処理を忘れたらブラウザが暴走するので、絶対忘れないようにしましょう。
変数 tag に、置換する HTML タグを代入します。 変数 str1 には「##」の前までの文字を、変数 str2 には「##」の後ろの文字を抜き出します。 文字の抜き出しは substring()を使います。後ろ側を抜き出す時の開始位置ですが、 indexOf()で取得した「##」の位置に 2 文字（つまり##）を加えた所となります。 終了位置は length を使って文字列長を取得すればOK です。
最後に str[i]を、「##」を HTML タグに置き換えた文字列に書き換えています。 「##」が複数あれば再び While 文の中で置き換えが行われ、 「##」が存在しなくなるまで繰り返されます。
終了タグの置き換えについても同様の処理を行っています。

今回見た while 文を使った無限ループは良く使います。 勿論下のようにきちんと条件を指定することも可能です。

```
while( str[i].indexOf("##",0) != -1 )
{
  置換処理
}
```

個人的には無限ループを使う方が好きなのですが、 条件をきちんと記入した方が理解しやすいかも知れません。 どちらでも好きな方を使ってください。 ただし無限ループは抜け出す処理を書き忘れないように気をつけて下さい。
文字の置換
前のページでは indexOf()を使って文字を検索し、 置換を行いました。 このページではもう少し簡単に文字を置換する方法について解説します。
文字置換サンプル
では以下のスクリプトを HTML の BODY 内に記入してみてください。

```
<script>

var str
 = "文字を置換するには、##replace()%%を使います。"
 + "括弧内の最初に##検索文字列%%、2 番目に##置換文字列%%を記入します。"
 + "結果は、##最初に Hit した文字列%%が置換された状態で返されます。";

//文字を置換する
str = str.replace("##","<strong>");
str = str.replace("%%","</strong>");

document.write(str);

</script>
```

＜上記スクリプトの結果＞

文字を置換するには、replace()を使います。括弧内の最初に##検索文字列%%、2 番目に##置換文字列%%を記入します。結果は、##最初に Hit した文字列%%が置換された状態で返されます。

前回は配列を使って長文を作成しましたが、 今回は長い文字列をそのまま変数 str に代入しました。 ページで表示する関係上、3 行に分けて「+」で繋いでいます。

そして、文字を置換する命令文 replace()を使い、 「##」と「%%」を開始タグ，終了タグに置き換えます。replace の括弧内は、最初が検索文字列、2 番目が置換文字列となります。

文字列.replace(検索文字列 , 置換文字列)

文字列の置換を行います

しかし、上記のスクリプトの結果を見ると分かるように、 最初に Hit した箇所しか置換されていません。 これでは不便なので、続く部分で全ての箇所を書き換える方法について見てみましょう。

全ての箇所を一括で置換する

replace()は最初に Hit した文字列の置換しか行わないので、 文字列全体の中に複数置換したい場所がある場合は何らかの手を打たなければなりません。 方法は以下のようなものがあります。

while 文と indexOf()を使って置換前文字が無くなるまで続ける

正規表現を使う

1 番目は直感的で分かりやすいかも知れませが、少し記述が長くなります。 以下のような感じになります。

```
while ( str.indexOf("##",0) != -1 )
```

```
{
   str=str.replace("##","<strong>");
}

//終了タグの方も同様にする
```

2 番目の正規表現というのは、高度な検索を行うための記述法です。以下のようにすると、検索文字列にマッチする全ての部分を書き換えることができます。

```
<script>

var str
 = "文字を置換するには、##replace()%%を使います。"
 + "括弧内の最初に##検索文字列%%、2 番目に##置換文字列%%を記入します。"
 + "結果は、##最初に Hit した文字列%%が置換された状態で返されます。";

//文字を置換する
str=str.replace(/##/g,"<strong>");
str=str.replace(/%%/g,"</strong>");

document.write(str);

</script>
```

＜スクリプトの結果＞

文字を置換するには、replace()を使います。括弧内の最初に検索文字列、2 番目に置換文字列を記入します。結果は、最初に Hit した文字列が置換された状態で返されます。

今回は全ての部分で HTML タグに置換することができました。 正規表現を使ったほうが記述が簡単になります。

次のページでは、この正規表現を使った検索について簡単に解説します。

正規表現による検索

前のページで、正規表現を使った文字列置換の方法について簡単に触れましたが、 ここでは正規表現を使えばどんな検索が行えるか見てみたいと思います。

正規表現を使った検索のサンプル

では以下のスクリプトを HTML の BODY 内に記入してみてください。

```
<script>

var str = "JavaScript で正規表現を使えば、高度な検索を行えます。";
var rep; //検索した文字列を格納する変数
```

```
//任意の文字を検索
rep = str.match(/正規表現/);
document.write(rep+"<br>"); //結果「正規表現」

//指定文字前後も含めて検索
rep = str.match(/..表現/);
document.write(rep+"<br>"); //結果「正規表現」

rep = str.match(/を.../);
document.write(rep+"<br>"); //結果「を使えば」

//指定文字の何れかを検索
rep = str.match(/[姓製正]/);
document.write(rep+"<br>"); //結果「正」

//指定文字列の何れかを検索
rep = str.match(/姓規 | 製規 | 正規/);
document.write(rep+"<br>"); //結果「正規」

//先頭文字を検索
rep = str.match(/^正規表現/);
document.write(rep+"<br>"); //結果「null」

rep = str.match(/^JavaScript/);
document.write(rep+"<br>"); //結果「JavaScript」

rep = str.match(/^..../);
document.write(rep+"<br>"); //結果「Java」

//末尾文字を検索
rep = str.match(/正規表現$/);
document.write(rep+"<br>"); //結果「null」

rep=str.match(/...$/);
document.write(rep+"<br>"); //結果「ます。」
```

```
//大文字小文字を区別しない
rep = str.match(/javascript/i);
document.write(rep+"<br>"); //結果「JavaScript」

//普通の検索は大文字小文字を区別する
rep = str.match(/javascript/);
document.write(rep+"<br>"); //結果「null」

</script>
```

正規表現の記入

では上記のスクリプトについて見てみることにしましょう。 変数 str に検索元となる文字列を指定しました。 また変数 rep を宣言し、検索結果の文字（列）を格納することにします。

そして検索に移りますが、最初に match()というのが出てきます。 これは検索に Hit するとその文字列を取得する命令文です。 見つからなければ null を返します。

文字列.match(検索文字)

文字列から任意の文字を検索する。Hit すればその文字列を、しなければ null を返す。

最初は通常の文字検索です。スラッシュ「/」で囲まれた文字を検索します。 ピリオド「.」を使えば、指定文字の前後も含めて検索できます。 ピリオドの数に応じて前後の文字列を取得します。

大括弧「[ ]」を使うと、括弧内の文字の内の何れかが存在するか調べます。 文字列の場合は、 縦線「|」で区切って複数指定することで、 いずれかの文字列が存在するか調べることができます。

先頭の文字に特化した検索を行う場合は、「^」を使います。 文末に特化した検索は「$」を使います。

指定文字列をスラッシュで閉じた後ろに「i」を付けると、 大文字と小文字を区別せず検索します。通常の検索では区別します。

では正規表現の幾つかをまとめておきましょう。

/文字列/

任意の文字列を検索する

/..文字列../

任意の文字列の前後も含めて検索する

/[文字１文字２文字３]/

指定した文字の何れかを検索する

/文字列１|文字列２|文字列３/

指定した文字列の何れかを検索する

/^文字列/

先頭の文字列を検索する

/文字列$/

末尾の文字列を検索する

/文字列/i

大文字と小文字の区別をせずに検索する

/文字列/g

指定した文字列に Hit する全ての文字列を取得する