cookie を使って訪問回数をカウントする

ここから、JavaScript で cookie（クッキー）を扱う方法について見ていきます。 最初は、数字を扱ってみます。訪問回数をカウントして表示してみることにしましょう。

訪問回数をカウントする流れ

まずは、訪問回数を表示するまでの流れを見ておきましょう。 初めてサイトを訪れた場合は、以下のようになります。

カウントを記録している cookie を読み込む

初めてなので訪問回数データが存在しない→「1 回目の訪問」と表示する。

cookie に訪問数（1）を書き込みます。

2 回目以降の訪問の場合は、次のようになります。

カウントを記録している cookie を読み込む

読み込んだ数字を+1 して、訪問回数を表示します。

cookie に訪問数を書き込みます。

訪問回数を表示するスクリプト

では上記の流れでスクリプトを組んでみましょう。 HTML ファイルを作成し、以下のコードを BODY 内に記入してみてください。

```
<script>

var cook;            //cookie データを格納する変数
var cStart,cEnd;   //訪問回数部分を切取る為の位置を格納
var cnt;             //訪問回数を格納

//cookie が使えるか確認
if (navigator.cookieEnabled)
{
  cook=document.cookie + ";";   //変数 cook に cookie データを入れる

  //変数 cStart にカウントデータの最初の位置を入れる
  cStart = cook.indexOf("counts=",0);

  //データの有無で分岐
  if (cStart == -1)
  {
    //データの無い場合は最初の訪問ということ
```

```
      document.write("1 回目の訪問です！");

      //cookie に訪問回数=1 を書き込む
      document.cookie="counts=1;";
    }
    else
    {
      //カウントデータの最後の部分「;」の位置を取得
      cEnd=cook.indexOf(";",cStart);

      //数値の部分だけを切り取る
      cnt=cook.substring(cStart+7,cEnd);

      //数値に変換できない例外が出た時の処理
      try
      {
        //取得した回数に+1 して表示する
        cnt=parseInt(cnt)+1;
        document.write(cnt+"回目の訪問です！");

        //cookie に訪問回数を書き込む
        document.cookie="counts="+cnt+";";
      }
      catch(e)
      {
        document.write("訪問回数の取得に失敗しました。");
      }
    }
}
else
{
    //cookie が使用できない時の処理
    document.write("cookie が使用できません。");
}

</script>
```

スクリプトの解説
では上記スクリプトの解説をしておきます。
変数宣言
最初に、使う変数をまとめて宣言しておきました。その方が解説し易かったからです。 変数 cook は読み込んだクッキーデータを格納します。
変数 cStart と変数 cEnd は、 そのクッキーデータのうち訪問回数が格納されている部分を切り取るための位置情報を格納します。 実は cookie には自分で書き込んだもの以外にも沢山の文字が書き込まれています。 以下のボタンを押して、実際にクッキーに記入されているものを見てみてください。
フォームの始まり
フォームの終わり
変数 cnt は、切り取った訪問回数、及びそれに 1 プラスしたものを格納します。
cookie が使えるか確認
次に、cookie が使用可能か if 文を使って確認しています。 ブラウザの設定によっては、cookie が使えないこともあります。 cookie が使用可能かどうかは、navigator.cookieEnabled を使って確認できます。
navigator.cookieEnabled
cookie が使用可能なら true、不可なら false を返します。
cookie データを取得する
今度は、変数 cook に cookie データを読み込ませます。 cookie データを取得したり、書き込んだりするには、document.cookie を使用します。 document.cookie が右辺にくれば、データを取得することになり、 左辺に来るとデータ書き込みになります。
str = document.cookie
変数 str に cookie データを読み込ませます。
document.cookie = str
変数 str の情報を cookie に書き込みます。
最後に、取得した cookie データの末尾にセミコロン「;」を付けてやります。 理由は後で説明します。
カウントデータを探す
取得した cookie データの中に、カウント数が記録されているかどうか調べます。 普通 cookie にデータを書き込むには、 データ名、値、 終了記号の３つを組み合わせて書き込みます。 こうすると後からデータを取り出しやすくなります。 今回はデータ名を「counts」、終了記号を「;」にして、以下の形式で書き込みます。
counts=値;
ということで「counts=」が cookie データに含まれていれば、 2 回目以降の訪問というこ

とになります。
文字列を検索するのは indexOf()を使います。 変数 cStart に、「counts=」の位置を格納します。 もし初めての訪問であれば「counts=」は存在しないので、変数 cStart の値は-1 となります。 if 文で「counts=」が無い場合と、ある場合に分岐させます。
初めての訪問の時
初めての訪問の時（変数 cStart の値が-1 の時）は、 document.write()で「1 回目の訪問です！」と表示させます。
続いて、カウントデータを書き込みます。 document.cookie を左辺に持ってきて、「counts=1;」と書き込みます。
2 回目以降の訪問の時
2 回目以降の訪問の時は、「counts=」が存在しているので、 今度は終了記号の「;」を探します。indexOf()を使って探しますが、 その検索開始位置を変数 cStart にすれば、 「;」が複数存在していても間違った位置を取得することはありません。取得した数値を変数 cEnd に代入します。
さて、カウントデータの最初の位置と最後の位置が分かったら、 その中から必要な値の部分だけを切り出します。 文字の切り出しは substring()を使います。 終了位置は変数 cEnd でいいのですが、開始位置は変数 cStart のままではいけません。 欲しいのは値の部分だけなので、 「counts=」の 7 文字分を加えた所を開始場所にしています。 切り取った値を、変数 cnt に代入します。
一応 try～catch 文を使って、例外処理をしています。 変数 cnt の値を parseInt()で数値に変換してから+1 します。 そして document.write()で訪問回数を書き出し、document.cookie で訪問回数を cookie に記録します。 記録の形式は「counts=値;」でした。
最後にセミコロンを付けた理由
さて、先程クッキーデータを変数 cook に代入する際、最後にセミコロン「;」を付加しました。 実は counts=値;の形で cookie に書き込んでも、このデータが cookie の一番最後に記述されるとセミコロンが脱落してしまうのです。 すると終了記号がなくなるので、切り出しに失敗します。これを回避するには、 クッキーデータ取得後末尾にセミコロンを改めて付けてあげれば良いわけです。

さて、上記のままだと、実はブラウザを閉じた時に訪問回数がリセットされてしまいます。これは、cookie の期限を記載していないからです。 ということでこのままでは訪問カウンターとしては使えません。 では次のページで、cookie の期限を記載する方法について解説してみます。

cookie の期限を指定する
前のページで訪問カウントを Cookie に記録する方法について見ました。 しか有効期限を

指定していなかったので、一旦ブラウザを閉じてしまうとカウント数が消えてしまう問題がありました。 それで今回は Cookie の有効期限を指定して、その時までデータを保存しておく方法について見てみることにしましょう。

cookie の有効期限を指定する流れ
最初に、Cookie の有効期限を指定する流れを掲載しておきます。 期限を 30 日後に指定したい場合、次のような手順を踏みます。
現在の日付データを取得します。
そのデータをもとに、30 日後の日付データを作成します。
日付データを GMT（グリニッジ標準時）に変換します。
cookie に書き込むとき、有効期限も一緒に書き込みます。
では前回の訪問数カウントスクリプトに、期限を付けて記録してみることにしましょう。

期限付きで Cookie に記録するスクリプト
前回のスクリプトをそのまま利用したいと思います。 変更箇所以外は灰色表示にしています。また前回のコメントは取り除いています。

```
<script>

var cook;
var cStart,cEnd;
var cnt;

var date1,date2;  //日付データを格納する変数
var kigen = 30;    //cookie の期限（今回は 30 日）

//現在の日付データを取得
date1 = new Date();

//30 日後の日付データを作成
date1.setTime(date1.getTime() + kigen*24*60*60*1000);

//GMT 形式に変換して変数 date2 に格納する
date2 = date1.toGMTString();

if (navigator.cookieEnabled)
{
```

```
  cook=document.cookie + ";";
  cStart = cook.indexOf("counts=",0);

  if (cStart == -1)
  {
    document.write("1 回目の訪問です！");
    document.cookie="counts=1;expires=" + date2;
  }
  else
  {
    cEnd=cook.indexOf(";",cStart);
    cnt=cook.substring(cStart+7,cEnd);
    try
    {
      cnt=parseInt(cnt)+1;
      document.write(cnt+"回目の訪問です！");
      document.cookie="counts="+cnt+";expires="+ date2;
    }
    catch(e)
    {
      document.write("訪問回数の取得に失敗しました。");
    }
  }
}
else
{
  document.write("cookie が使用できません。");
}

</script>
```

ブラウザを閉じてもう一度アクセスしてみてください。 今度はカウント数がリセットされないと思います（30 日以内であれば)。

上記スクリプトの解説

ではスクリプトを詳しく見ていきましょう。前回解説した部分は前のページを参照して下

さい。 新しく書き足した部分を説明していきます。

変数の宣言

まずは日付データを入れる変数 date1,date2 と、 有効期限（日数）を入れる変数 kigen を宣言しています。 期限を変更したい場合は、変数 kigen の値を変更すればすぐに対応できます。

現在の日付データを取得する

現在の日付データを取得して、変数 date1 に代入しています。 現在の日付データは new Date()で取得できました。 これは簡単ですね。

30 日後の日付データを作成する

次に 30 日後の日付データを作成したいのですが、これが少し厄介です。 現在の日付データから年・月・日を取得して、月に 1 プラスして新たに日付データを作ってもいいですが、12 月の場合は年を加える必要がありますし、31 日や 28 日の月もあります。 それらを条件分岐していると結構煩雑なコードになります（うるう年なども考えるとえらく大変です）。こんな時に便利なのが、setTime()という命令文です。setTime()は「時間」を指定する命令文ですが、 この「時間」とは 3"時"とか 12"時"の時間ではありません。 1970 年 1 月 1 日午前 0 時から何ミリ秒後か…という意味の時間を設定する命令文なのです。

日付データ.setTime()

日付データの時間（1970/1/1 00:00:00 からのミリ秒）を設定します。

同じ意味で getTime()というのは、日付データが 1970 年 1 月 1 日午前 0 時から何ミリ秒後かを取得します。

日付データ.getTime()

日付データの時間（1970/1/1 00:00:00 からのミリ秒）を取得します。

変数 date1（現在の日時）のミリ秒を getTime()で取得し、 30 日＝30×24×60×60×1000 ミリ秒を加えた値を、setTime()で再び変数 date1 にセットしています （ただし今回は 30 ではなく、変数 kigen を使用しています）。

GMT（グリニッジ標準時）で表わす

cookie に期限を書き込む場合は、GMT で書き込みます。ということで、日付データを GMT に変換し、変数 date2 に格納しています。 グリニッジ標準時に変換するには、toGMTString()を使います。

日付データ.toGMTString()

日付データをグリニッジ標準時に変換します。

それぞれどのように表示されるか、下のボタンで確認してみてください。 日本の場合、9 時間の差が出ていると思います。

フォームの始まり

フォームの終わり

有効期限も一緒に cookie に書き込む

最後に、有効期限を cookie に書き込む方法について見ておきます。 有効期限付きにするには、expires=GMT という形式で書き込みます。
expires=GMT
cookie に有効期限（GMT）を付します。
これで無事に訪問回数をカウントすることができるようになりました。

ここまでは、数値を書き込む方法について見てきました。 数字の場合、そのまま cookie に書き込むことができるので簡単です。 しかし文字の場合は簡単ではありません。 では次のページで、cookie で文字を扱う方法について見てみることにしましょう。

cookie で文字を扱う
前回までで、cookie に数値を書き込む方法について考えてきました。 しかし、同じ方法で文字を扱おうとしてもうまくいきません。 このページでは cookie に文字を書き込み・読み出す方法について見ていきたいと思います。

cookie にそのまま文字を書き込むと…
ではまず、cookie にそのまま文字を書き込むとどうなるか試してみましょう。 以下のようなスクリプトを body 内に書き込んだサンプルを作っています。 本来なら変数 word1 と変数 word2 は同じはずですが、結果は如何に・・・

```
<script>

var word1="あいうえお";   //書き込む文字
var word2;                 //cookie から読み出した文字
var cook;                  //cookie のデータを格納
var cStart,cEnd;           //文字切り取りの位置を格納

if (navigator.cookieEnabled)
{
  //cookie に文字を書き込む
  document.cookie = "moji="+word1+";";
  document.write("書込文字＝"+word1+"<br>");

  //cookie を読み込む
  cook = document.cookie + ";";
  cStart= cook.indexOf("moji=",0);
  cEnd   = cook.indexOf(";",cStart);
```

```
   word2= cook.substring(cStart+5,cEnd);
   document.write("読込文字＝"+word2);
}

</script>
```

書き込んだ文字は「あいうえお」ですが、読み出した文字はどうなっているでしょうか？ 使用しているブラウザによっては一致しているものもありますし、全然違う文字に変わっているものもあります。 このような状況ゆえに、cookie で日本語などの文字を扱う場合は、一工夫する必要があります。 ではどうするか、次の部分で見てみましょう。

cookie ではエスケープ文字を使う

cookie で文字を扱うには、エスケープ文字に変換する必要があります。 読み出すときは、エスケープ文字を普通の文字に戻す作業が入ります。

では上のスクリプトを少し変更して、文字を扱えるようにしてみましょう。 変更箇所は赤文字で表しています。たった 2 箇所変更しているだけです。

```
<script>

var word1="あいうえお";   //書き込む文字
var word2;                 //cookie から読み出した文字
var cook;                  //cookie のデータを格納
var cStart,cEnd;           //文字切り取りの位置を格納

if (navigator.cookieEnabled)
{
   //cookie に文字を書き込む
   document.cookie = "moji="+escape(word1)+";";
   document.write("書込文字＝"+word1+"<br>");

   //cookie を読み込む
   cook = document.cookie + ";";
   cStart= cook.indexOf("moji=",0);
   cEnd   = cook.indexOf(";",cStart);
   word2= unescape(cook.substring(cStart+5,cEnd));
   document.write("読込文字＝"+word2);
}
```

</script>

サンプルをご覧になってみてください。 今度は書き込み文字と読み込み文字が一致していると思います。
このように、cookie で文字を扱う場合は、文字をエスケープ文字に変換する必要があります。 エスケープ文字に変換するのは escape()、 エスケープ文字を元に戻すには unescape()を使います。
escape()
カッコ内の文字列をエスケープ文字に変換します。
unescape()
カッコ内のエスケープ文字列を、通常の文字に戻します。
上記スクリプトでは、cookie に書き込む時に escape()を使って文字を変換しています。 また読み出した後に unescape()で元に戻してるのが分かると思います。

数値をエスケープ文字にしても問題ありません（というか、変化しません)。 と言うことで cookie にデータを書き込むときは、 エスケープ文字に変換するようにしておけば、 数値・文字どちらにも対応できます。

ここまでで､cookie にデータを書き込んだり読み出したりする方法について見てきました。次のページでは、cookie を削除する方法について見てみることにしましょう。

cookie データを削除する
前のページまでで､cookie に書き込んだり読み込んだりする方法について見てきました。このページでは、そうして書き込んだ cookie のデータを削除する方法について考えていきます。

cookie を削除するスクリプト
cookie には有効期限があることを以前に学びました。 ということで、 有効期限を過去に設定すれば cookie を削除できます。 ではそのようなスクリプトを組んで、確認してみましょう。
以前作った訪問数をカウントするスクリプトは、まだこのページでも有効なはずです。 試しに表示してみます。
もし「1 回目の訪問」になっているようなら、 ページを再読み込みしてカウント数を少し増やしておいてください。
では、ボタンを押すことで cookie データを削除するスクリプトを記述してみます。

```
<script>

function delCookie()
{
 //日付データを作成する
 var date1 = new Date();

 //1970 年 1 月 1 日 00:00:00 の日付データをセットする
 date1.setTime(0);

 //有効期限を過去にして書き込む
 document.cookie = "counts=;expires="+date1.toGMTString();

  //ページを再読み込みする
  location.reload();

}

</script>

<form>
<input type="button" value="クッキー削除" onclick="delCookie()">
</form>
```

ボタンを押すと、cookie データを削除してページを再読み込みします。 訪問回数がリセットされているか確認してください。

スクリプトの解説

では上記スクリプトを見ていきましょう。

関数 delCookie()

今回はボタンから JavaScript を呼び出すために、関数 delCookie()を作成しました。 cookie データを削除して、ページを再読み込みする処理を記述していきます。

過去データの作成

関数の中で最初に、過去データを作成します。 変数 date1 を宣言し、現在の日付データを入れます。 そして setTime()で過去のデータにセットし直します。 setTime()は「1970 年 1 月 1 日午前 0 時から経過したミリ秒」の意味でしたので、 括弧内に 0 を入れると 1970 年 1 月 1 日午前 0 時になります。

cookie の削除

過去データを作成できたら、cookie に書き込みます。 訪問回数をカウントする時のデータ名は「counts」にしていました。 その値を空欄にし、終了記号「;」も忘れず記入します。有効期限を先ほどの過去のデータに指定しますが、 GMT に変換することを忘れてはいけません。

cookie データを削除する場合も「データ名」を入れているので、 cookie 内の目的のデータだけ削除できることが分かります。

ページを再読み込み

最後に、ページを再読み込みさせます。location.reload()を使います。 ここまでが関数 delCookie()で実行される処理です。

フォームのボタンから関数を呼び出す

ボタンから関数を呼び出すときは、onclick イベントの中にそのまま関数名を記述すれば OK でした。

cookie を扱う方法のまとめ

ここまでで cookie を扱う方法について見ることができました。以下に簡単にまとめておきます。

書き込む時は「データ名=値;」の形式で書き込む
書き込む時はエスケープ文字に変換する
有効期限指定は「expires=」を使い、GMT で指定する
有効期限を過去にすると、データを削除できる
読み込みの際、データの末尾にセミコロン「;」を追加する
読み込んだらデータ名と終了記号の位置を取得して、値部分を切り取る
読み込んだあとエスケープ文字を通常文字に戻す

実際に cookie を初めて扱う時には、難しく感じるかもしれません。 私も最初は難しく感じました。 しかし上に挙げた要点さえ理解できれば、恐れる必要はないでしょう。 もう一度しっかり読み返して、十分理解できるよう頑張って下さい。

次のページでは、cookie に関連した書き込み関数、読み込み関数、 削除関数を作って cookie.js という外部ファイルにまとめてみたいと思います。 一度作ってしまえば、 必要な時にその外部ファイルを読み込むことで、cookie を楽に扱えるようになります。

cookie.js を作ってみる

前のページまでで、cookie ファイルの読み書きと削除方法について見てきました。 cookie 編の最後では、cookie 関連の関数を一つの外部ファイルにまとめて、 複数のページからで

も楽に使えるようにしてみたいと思います。

cookie 書き込み関数

最初に cookie 書き込み関数を作ってみたいと思います。以下のようにしてみました。

```
function ckSave(nam,val,term)
{
  if (navigator.cookieEnabled)
  {
    var date = new Date();

    try
    {
      term = parseInt(term);
    }
    catch(e)
    {
      term = 30;
    }

    date.setTime(date.getTime()+term*24*60*60*1000);

    document.cookie = nam+"="+escape(val)+";expires="+date.toGMTString();

    return true;
  }
  else
  {
    return false;
  }
}
```

関数名は「ckSave」、引数は順に「項目名」,「値」,「有効期限（日数）」です。

if 文で cookie が使用可能か調べて、 可能なら日付データ date を作成します。 有効期限は数値で指定しても、引数を介して値を受け取ると数値ではなくなることがあるので、parseInt()を使って数値に変換します。数値変換に失敗した場合は、期限を 30 日に設定し直します。 そして有効期限の日数をプラスした日付データに書き換えます。

そして cookie を書き込みますが、注意点が２つありました。 値は escape 文字に変換する

ことと、日付データは GMT に変換するという 2 点です。
書き込みが成功したか調べるために、return を使って true もしくは false を呼び出し元に返します。 ということで、cookie を書き込む際には if 文を使って書き込みに成功したか調べると良いでしょう。

```
if ( ckSave("count",10,30) )
{
   alert("データを保存しました！");
}
else
{
   alert("データ保存に失敗しました！");
}
```

cookie 読み込み関数
今度は cookie を読み込む関数を作ってみます。 こちらは return を使って「値」を呼び出し元に返すようにしてみました。

```
function ckLoad(nam)
{
   if (navigator.cookieEnabled)
   {
      var cook = document.cookie + ";";
      var cStart,cEnd;

      nam=nam+"=";

      cStart = cook.indexOf(nam,0);

      if (cStart == -1)
      {
         return "nodata";
      }
      else
      {
         cEnd = cook.indexOf(";",cStart);
         return unescape(cook.substring(cStart+nam.length,cEnd));
      }
   }
```

```
  else
  {
    return "nodata";
  }
}
```

関数名は「ckLoad」、引数は「項目名」の１つです。
最初に cookie が使用可能か調べます。 不可の場合は return で「nodata」という文字列を返すようにしています。
使用可能なら変数 cook に cookie データを読み込ませます。 そして引数「nam」に「=」を追加してから、indexOf()を使って検索しています。 検索結果が「-1」であれば、やはり return で「nodata」を返します。
項目名が cookie データ内にあれば、今度は終了記号「;」を検索し、 substring()で値部分を切り取ります。 この時切り取り開始位置は、変数 nam の文字数だけプラスしなければなりません。 length を使って文字数を取得しています。
最後に unescape()で escape 文字を普通の文字に戻しています。
cookie を読み出すには、以下のようにします。

```
var cnt = ckLoad("count");

if ( cnt != "nodata" )
{
  try
  {
    cnt=parseInt(cnt)+1;
    document.write(cnt+"回目の訪問です。");
  }
  catch(e)
  {
  }
}
else
{
  document.write("初めての訪問です。");
}
```

cookie 削除関数
最後は、cookie 削除関数を作ってみましょう。 やはり削除に成功したかどうか確認できるよう、return で true か false を返すようにしています。

```
function ckDelete(nam)
{
  if ( navigator.cookieEnabled )
  {
    var date = new Date();
    date.setTime(0);

    document.cookie=nam+"=;expires="+date.toGMTString();
    return true;
  }
  else
  {
    return false;
  }
}
```

関数名は「ckDelete」，引数は「削除したい項目名」を指定できるようにしています。
cookie が使用可能か調べて、可能なら過去の日付データを作成し、 それを有効期限に指定して書き込みを行います。 書き込みができたら true を返し、できない時は false を返すようにしています。

cookie.js の完成

では、上記の関数 ckSave()，関数 ckLoad()，関数 ckDelete()を一つのファイルにコピペして、 「cookie.js」というファイル名で保存しましょう。 これで cookie 関数をまとめて扱うことができます。

この cookie.js を任意のページで読み込めば、簡単に cookie を読み書きできます。 簡単なサンプルを作っていますのでご確認ください。