一定時間で繰り返す（setInterval）

ここから、JavaScript のタイマーについて解説していきます。 タイマーは、一定時間毎に動作を繰り返す時などに用いることができます。 スライドショーを作ったり、文字を切り替えたり流したりする時などに使用します。

このページでは文字を一文字ずつ表示する方法について見てみましょう。

setInterval()を使ったタイマーのサンプルスクリプト

タイマーには 2 種類あって、ここでは setInterval()という命令文を使ってみます。 もう一つは次のページで解説します。 setInterval()は、一定時間毎に特定の関数を呼び出します。

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="timer">
<input type="button" value="スタート" onclick="startfnc()"><br>
<input type="text" name="moji" size="30" value="">
</form>

<script>

function hyoji()
{
  //表示する文字
  var str = "一文字ずつ表示します。";

  //テキストボックスの文字数
  var cnt = document.timer.moji.value.length;

  //文字が全部表示されているか確認
  if ( cnt < 11 )
  {
    //現在より 1 文字多く切り出して表示
    document.timer.moji.value = str.substr(0,cnt+1);
  }
  else
  {
    //全て表示されたら、空文字に戻す
    document.timer.moji.value = "";
  }
```

```
}


function startfnc()
{
  //関数 hyoji()を 1000 ミリ秒間隔で呼び出す
  setInterval("hyoji()",1000);
}

</script>
```

スクリプトの解説
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「timer」、 ボタンを一つ配置してタイマーを起動させる関数 startfnc()を呼び出します。 文字を表示するテキストボックスの名称は「moji」としました。
続いて、1 文字ずつ表示する関数 hyoji()の部分を見てみましょう。 変数 str に表示させる文字列を代入します。変数 cnt は､現在テキストボックスに表示されている文字数を length を使って取得し、 代入しています。全部表示されていたら、文字数は 11 になります。
そして、if～else 文を使って、 文字が全部表示されてる場合と、そうでない場合とで分岐します。 まだ全部表示されていない場合は、substr()を使って現在より 1 文字多く変数 str から切り出し、 テキストボックスに表示します。 全部表示されていたら、テキストボックスを空にします。
続いて、タイマーを起動する関数 startfnc()の説明です。 ここで setIntarval()が出てきます。 括弧内の 1 番目は関数名を、2 番目は呼び出し間隔を記入します。 呼び出し間隔はミリ秒（1000 分の 1 秒）単位になります。
setInterval(関数名,間隔)
特定の関数を指定した間隔で繰り返し呼び出します。単位はミリ秒。
上記のコードの場合、関数 hyoji()を 1000 ミリ秒（つまり 1 秒）間隔で呼び出すことになります。 呼び出し間隔を少しずつ変えて、確認してみてください。
setInterval()の括弧内で関数名を記入するとき、 クォーテーションが必要です。良く忘れるので、 うまく動かないときは確認してみましょう。

一定時間で繰り返す（setTimeout）
前のページでは setInterval()を使ったタイマー設定について見ました。 このページではもう一つのタイマー設定方法、setTimeout()について説明していきます。

setTimeout()を使ったサンプルスクリプト

setTimeout()は設定時間後に関数を呼び出す命令文で、 それ自体では繰り返しは起こりません。 例えば以下のスクリプトを HTML の BODY 内に書いてみてください。 3 秒後にメッセージが表示されて終わりになります。

```
<form>
<input type="button" value="Timer"
                        onclick="setTimeout('mes()',3000)">
</form>

<script>
function mes()
{
   alert("3 秒経ちました！");
}
</script>
```

setTimout()の括弧内は、以下のようになっています。 関数名はクォーテーションで括るのを忘れないようにしましょう。 上記サンプルではダブルクォーテーションで囲まれた中に記入しているので、 関数名はシングルクォーテーションで括っています。

setTimeout(関数名,時間)

設定した時間（単位はミリ秒）が経過すると、関数を呼び出します。

では、setTimeout()を使って動作を繰り返すにはどうしたらいいか、次に考えましょう。

setTimeout()を使って繰り返す

ここでは、前のページと同じように一文字ずつ表示するスクリプトを組んでみたいと思います。 setTimeout()は 1 度関数を呼び出すと、役目を終えてしまいます。 では、繰り返すためどうすればいいでしょうか？ setTimeout()を関数の中に記入して、自分自身を含む関数を呼び出すようにすれば良いのです。

```
<form name="timer">
<input type="button" value="スタート" onclick="hyoji()"><br>
<input type="text" name="moji" size="30" value="">
</form>

<script>

function hyoji()
{
```

```
  //表示する文字
  var str = "一文字ずつ表示します。";

  //テキストボックスの文字数
  var cnt = document.timer.moji.value.length;

  //文字が全部表示されているか確認
  if ( cnt < 11 )
  {
    //現在より 1 文字多く切り出して表示
    document.timer.moji.value = str.substr(0,cnt+1);
  }
  else
  {
    //全て表示されたら、空文字に戻す
    document.timer.moji.value = "";
  }

  //setTimeout()を含む関数を呼び出す
  setTimeout("hyoji()",1000);
}


</script>
```

スクリプトの解説は前のページを見てもらえたら良いと思います。 変更点は、ボタンで呼び出す関数が hyoji()になっています。 直接文字表示関数を呼び出しています。
そして関数 hyoji()の一番最後に、setTimeout()で再び関数 hyoji()を呼び出します。 これで繰り返しが発生する訳です。

注意点を一つ。setTimeout()による繰り返しは、setInterval()と比べてずれが生じます。 上記スクリプトでは 1000 ミリ秒毎に関数が呼び出される訳ですが、 11 文字表示するのに 11 秒以上掛かります。 1000 ミリ秒後に関数が呼び出され、関数内の処理が行われた後に、 再び setTimeout()が呼び出されるからです。 関数内の処理が 0.2 秒掛かるとすれば、上記の文字表示の間隔は 1.2 秒毎ということになります。
setInterval()は関数の呼び出し間隔を指定するので、 一定時間毎に処理を繰り返したいな

ら、こちらを使うと良いかもしれません。 しかし重い処理を繰り返す場合、処理し切らないうちに次を呼び出してトラブルになったりします。 目的・用途に合わせて使い分けていきましょう。

タイマーを停止する

前回までででタイマーの設定方法について見てきましたが、 一度タイマーを稼動させるとずっと動き続けていました。 それで今回は、タイマーを停止する方法について考えていきたいと思います。

タイマー停止スクリプト

タイマーを停止させるには、タイマーに名前を付ける必要があります （これをタイマーIDと言います)。 具体的には、タイマー自体を変数に格納します。 変数名がタイマーID ということになります。

では、一定時間でランダムに数字を表示していくタイマーを ON/OFF 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="tm">
<input type="text" value="">
<input type="button" value="ON" onclick="tmstr()">
<input type="button" value="OFF" onclick="tmrOff(0)"><br>
<input type="text" value="">
<input type="button" value="ON" onclick="tmrOn(3)">
<input type="button" value="OFF" onclick="tmrOff(3)"><br>
</form>


<script>

//タイマーを格納する変数の宣言
var timer1,timer2;

//setInterval()を使ったタイマーの起動関数
function tmstr()
{
   timer1 = setInterval("tmrOn(0)",1000);
}
```

```
function tmrOn(num)
{
   //引数を数字に変換
   var n = parseInt(num);

   //０～９までの乱数を取得して変数に代入
   var rnd = Math.floor(Math.random()*10);

   //テキストボックスに連続して表示
   var str = document.tm.elements[n].value;
   document.tm.elements[n].value = str+rnd;

   //引数が 3 の時は setTimeout()を使って繰り返す
   if (n==3)
   {
      timer2 = setTimeout("tmrOn(3)",1000);
   }
}


function tmrOff(num)
{
   //タイマーを停止する
   if (num==0)
   {
      clearInterval(timer1);
   }
   else
   {
      clearTimeout(timer2);
   }
}

</script>
```

上のボックスが setInterval()を使ったタイマーの ON/OFF、下が setTimeout()を使ったタ

イマーの ON/OFF です。
タイマー停止スクリプトの解説
では上記スクリプトを詳しく見ていきましょう。 フォーム名は「tm」にしました。 そして「テキストボックス＋ON/OFF 各ボタン」のセットを ２つ作っています。
上の列の ON ボタンでは、関数 tmstr()を呼び出していますが、 これは setInterval()を使ってタイマーを起動する関数です。下の列の ON ボタンは関数 tmrOn()を呼び出しますが、こちらは setTimeout()を使ったタイマーを起動します。 引数はテキストボックスの elements[]の番号です。
OFF ボタンはタイマー停止関数 tmrOff()を呼び出します。引数を設定しているのはどちらのタイマーを停止させるのか識別するためです。 別に数字は何でも構わないのですが、今回は 0 と 3 にしました。
続いて JavaScript の部分を見ていきたいと思います。最初に変数 timer1,timer2 を宣言しています。グローバル変数にしているのは、複数の関数で timer1,timer2 を使うからです。timer1 は setInterval()タイマーを格納します。timer2 が setTimeout()となります。
次に関数 tmstr()がありますが、 setInterval()を使って関数 tmrOn()を 1000 ミリ秒単位で呼び出します。 関数 tmrOn()の引数はテキストボックスの elements[]の番号です。 このタイマーを変数 timer1 に格納します。
次は関数 tmrOn()です。最初に引数を parseInt()で数字に変換します。 続いて 0～9 までの乱数を発生させて、変数 rnd に代入しています。 変数 str は現在表示されているテキストボックスの文字列です。 それに変数 rnd を加えてテキストボックスに表示しています。
関数 tmrOn()の最後で、if 文を使って引数が 3 か確認します。 3 の時は setTimeout()を使って処理を繰り返すようにします。 この時、変数 timer2 にタイマーを格納します。
最後はタイマー停止関数 tmOff()です。 引数が 0 の時は、setInterval()のタイマーを停止します。 clearInterval()がタイマーを停止する命令文です。 括弧内はタイマーID（タイマーを格納した変数）を指定します。引数が 0 以外の時は clearTimeout()を使って setTimeout()のタイマーを停止します。
clearInterval(タイマーID)
タイマーID で指定された setInterval タイマーを停止します。
clearTimeout(タイマーID)
タイマーID で指定された setTimeout タイマーを停止します。
スライドショー
ここからタイマーを使ってどんなことができるか、具体的に見ていくことにしましょう。
最初はスライドショーを作ってみたいと思います。 スライドショーは、一定時間で画像が順に変わっていく機能のことです。
因みにここのページで解説している内容は少し難しいと思います。 理解し難い場合は、DOM 編を少し見てからここに戻ってこられると、理解できるかも知れません。

スライドショーを作成する
では、某アニメ「ル○ン三世」のタイトルで使われていたようなスライドショーを作ってみることにしましょう。 以下の画像をまずダウンロードし、フォルダ名を「img」にしてその中に保存してください。 ファイル名は記入されているもので保存して下さい。

kuro.png 0.png 1.png

2.png　　3.png 4.png

5.png　　6.png 7.png

ではスライドショーのスクリプトを書いてみましょう。 HTML ファイルを img フォルダと「同じ場所」に作成します（下図）。

そしてヘッダーに以下のコードを記入してください。

```
<script>

//画像を格納する配列の作成
var image = new Array();

//配列の各要素を画像に特化して、そのパスを記入
image[0]=new Image();
image[0].src="img/0.png";
image[1]=new Image();
image[1].src="img/1.png";
image[2]=new Image();
image[2].src="img/2.png";
image[3]=new Image();
image[3].src="img/3.png";
image[4]=new Image();
image[4].src="img/4.png";
image[5]=new Image();
image[5].src="img/5.png";
image[6]=new Image();
image[6].src="img/6.png";
image[7]=new Image();
```

```
image[7].src="img/7.png";

var cnt=0;

function slidesw()
{

  //getElementById が使える場合のみ後の処理をする
  if(document.getElementById)
  {

    //スライド中はボタンを押せなくする
    document.slide.elements[0].disabled=true;

    //id 属性が「sd」の画像タグの画像パスを切り替える
    document.getElementById("sd").src = image[cnt].src;

    //一つ画像を表示したらカウント用変数 cnt の値を＋１にする
    cnt++;

    //画像が最後まで表示されたか確認
    if( cnt <= 7 )
    {
      //まだ表示されていなければ、setTimeout()で次の画像を表示する
      var timer1=setTimeout("slidesw()",300);
    }
    else
    {
      //全て表示されていたら、ボタンを押せるようにして、タイマーを停止する
      cnt=0;
      document.slide.elements[0].disabled=false;
      clearTimeout(timer1);
    }
  }
}
</script>
```

続いて HTML の BODY 内に画像タグとボタンを一つ設置します。

```
<img src="img/kuro.png" id="sd" width="320" height="240" alt="">
<br>

<form name="slide">
<input type="button" value="開始" onclick="slidesw()">
</form>
```

スライドショースクリプトの解説

上記のコードについて見ていきましょう。 まず先に HTML 部分を見てみたいと思いますが、 スライドさせる画像の最初の 1 枚を表示しておきます。 今回は別に「kuro.png」を作りましたが、0.png で代用しても構いません。 そして大事なのは、id 属性を付ける事です。 今回は「sd」という名前の id を付けました。

続いて JavaScript 部分です。スライドショーにする時には、 画像を先に読み込んでおくとスムーズに表示できます。 ということでヘッダーで画像をプレロード（先読み）する処置をとりました。

先読みの方法ですが、まず画像を格納する配列 image を作成しています。 この配列に今までは文字列や数字を入れましたが、今回は画像を格納します。 画像を格納する場合は、上記スクリプトのように 2 段階かけて行います。

配列[]=new Image()

配列の要素に画像を作成する（画像オブジェクトと言います）

配列[].src=画像パス

配列の画像のパスを指定します

次に、グローバル変数 cnt を宣言し、0 を代入しておきます。 この変数 cnt を 1 つずつ加算して、画像の入った配列を順に表示していきます。

関数 slidesw()について見てみましょう。まず最初に if 文で document.getElementById が使えるか調べています。 （この getElementById はまた別の部分で解説します。 今はスライドショーでのお決まりの方法だと思っておいてください）。 古いブラウザでは使えないことがあるので、最初に確認しておきます。

getElementById が使えるなら、if 文の中の処理を行います。 最初に、ボタンを禁止状態にしています。 スライド中はボタンを押せなくなります。

次に、画像を切り替える命令文です。 先ほどの「sd」という id 属性の付いた画像タグを getElementById で取得し、 その画像パスを切り替えます。切り替えたら変数 cnt の値を +1 しておきます。

document.getElementById("id 名").src=画像.src

指定した id 名のついた画像タグの画像を切り替えます

そして変数 cnt の値を再び if 文でチェックし、全ての画像が表示されたかどうか確認しま

す。 配列 image の最後の要素は 7 ですから、それ以下ならまだ画像が残っています。 画像が残っている場合は、setTimeout()を使って再び関数 slidesw()を呼び出します。
全ての画像を表示したら、cnt の値を元に戻し、再びボタンを押せるようにして、 タイマーを停止します。
今回は画像を全て表示するとスライドショーを停止しました。 もし画像をエンドレスに表示したい場合は、 cnt の値を 0 に戻して setTimeout()を使って再び関数 slidesw()を呼び出せば OK です。

スタッフロール
今回は映画やドラマのラストで、 キャストやスタッフの一覧が下から上に流れるスタッフロールを JavaScript で作ってみたいと思います。
スタッフロールのスクリプト
では最初に、640×480 程度のサブウィンドウを画面中央に表示させましょう。 HTML の BODY 内に、以下のように記入なさってください。

```
<script>

function winOpen()
{
   var w=(screen.width-640)/2;
   var h=(screen.height-480)/2;

   window.open("staffroll.html","sub","width=640,height=480,"+
      "left="+w+",top="+h+",scrollbars=no,menubar=no,toolbar=no");
}

</script>

<form>
<input type="button" value="staffroll" onclick="winOpen()">
</form>
```

次に staffroll.html を作ります。 body タグ以下に下の文字をコピー&ペーストしてみてください。body タグ内には onload イベントを記入し、画面をスクロールさせる関数 scroll()を呼び出します。

```
<body onload="scroll()">
<br><br><br><br><br><br><br><br><br><br>
STAFF<br>
```


企画
パズルネット智慧

構成
パズルネット智慧

プログラム
パズルネット智慧

製作環境
UTF-8Writer
http://pori2.net/

技術提供
JavaScript 入門
http://pori2.net/js/

Fin

</body>

さらに、ヘッダーでスタイルを指定します。 背景を黒色，文字サイズを 20pt で太字、センタリング，文字色を白色にします。 以下のスタイルシートを staffroll.html のヘッダー部分に記入してみてください。

```
body
{
   background-color:black;
   font-size:20pt;
   font-weight:bold;
   text-align:center;
   color:white;
}
```

前置きが長くなりましたが、いよいよスタッフロールのスクリプトです。 中身はわずか 2 行だったりします。 staffroll.html のヘッダーに、以下のスクリプトを記入なさってください。

```
<script>

function scroll()
{
   scrollBy(0,2);
   setTimeout("scroll()",100);
}

</script>
```

scrollBy()を使って、画面を 2px 下にずらします。 そして setTimeout()を用いて、100 ミリ秒毎に関数 scroll()を呼び出します。 これで画面が徐々に下に移動し、文字が上に流れているように見えるのです。 もっと滑らかに見せたければ、移動量を 1px にして呼び出し間

隔を半分にしてみてください。
スタッフロールを実際に作る場合は、最初と最後は改行タグではなく、 透明画像を用いて縦幅を調整するとベターです。画像だと height 属性を使ってきちんとした縦幅を指定できるからです。

```
<img src="blank.gif" height="480" width="1" alt=""><br>
STAFF<br>
…（以下略）
```

例えば上のように最初に縦幅 480px の透明画像を入れておけば、 画面を開いた時には何も表示されておらず、 すぐに下から「STAFF」の文字が現れることになります。
あとは音楽でも流せば本物のスタッフロールに見せることができます！

カウントダウンタイマー
JavaScript の timer を使って、カウントダウンタイマーを作る方法についてみていきましょう。 本当はスタートと現在の時刻を取得して、そこから差を求める方が正確なのですが、とりあえずここでは setInterval()を用いて作ってみたいと思います。

カウントダウンタイマーのスクリプト
では、HTML の BODY 内に、以下のフォームを記入して下さい。

```
<form name="timer">
<input type="text" value="">分
<input type="text" value="">秒<br>
<input type="button" value="スタート" onclick="cntStart()">
<input type="button" value="ストップ" onclick="cntStop()">
</form>
```

さらに HTML のヘッダーに、以下のスクリプトを記入します。

```
<script>

var timer1; //タイマーを格納する変数（タイマーID）の宣言


//カウントダウン関数を 1000 ミリ秒毎に呼び出す関数
function cntStart()
{
   document.timer.elements[2].disabled=true;
   timer1=setInterval("countDown()",1000);
}
```

```
//タイマー停止関数
function cntStop()
{
   document.timer.elements[2].disabled=false;
   clearInterval(timer1);
}

//カウントダウン関数
function countDown()
{
   var min=document.timer.elements[0].value;
   var sec=document.timer.elements[1].value;

   if( (min=="") && (sec=="") )
   {
      alert("時刻を設定してください！");
      reSet();
   }
   else
   {
      if (min=="") min=0;
      min=parseInt(min);

      if (sec=="") sec=0;
      sec=parseInt(sec);

      tmWrite(min*60+sec-1);
   }
}

//残り時間を書き出す関数
function tmWrite(int)
{
   int=parseInt(int);
```

```
  if (int<=0)
  {
    reSet();
    alert("時間です！");
  }
  else
  {
    //残り分数は int を 60 で割って切り捨てる
    document.timer.elements[0].value=Math.floor(int/60);
    //残り秒数は int を 60 で割った余り
    document.timer.elements[1].value=int % 60;
  }
}

//フォームを初期状態に戻す（リセット）関数
function reSet()
{
  document.timer.elements[0].value="0";
  document.timer.elements[1].value="0";
  document.timer.elements[2].disabled=false;
  clearInterval(timer1);
}

</script>
```

スクリプトの解説

フォーム部分

最初は HTML のフォーム部分です。フォーム名は「timer」にしました。 以下のパーツを設置しています。

elements[0]　テキストボックス　分数を入力

elements[1]　テキストボックス　秒数を入力

elements[2]　スタートボタン　関数 cntStart()を呼び出す

elements[3]　ストップボタン　関数 cntStop()を呼び出す

JavaScript の最初の部分

続いて JavaScript 部分を見ていきましょう。 最初に変数 timer1 を宣言しています。関数の外で宣言しているので、これはグローバル変数（複数の関数で使用可）です。この timer1

にタイマーを格納して、ON/OFF できるようにします。

関数 cntStart()

関数 cntStart()で最初にスタートボタンを禁止状態にしています。 その後、カウントダウン関数 countDown()を setInterval()で 1000 ミリ秒、つまり 1 秒毎に呼び出します。 このタイマーを変数 timer1 に格納します。

関数 cntStop()

タイマー停止関数 cntStop()では、スタートボタンの禁止状態を解除し、 clearInterval()でタイマーを解除します。 しかしフォームのテキストボックスは弄らないので、再びスタートボタンを押すと続きからカウントされます。 どちらかと言うと一時停止ボタンに近いです。

関数 countDown()

関数 countDown()の最初で変数 min,sec を宣言し、テキストボックスの分数と秒数の値を代入します。

そして if 文を用いて、テキストボックスに何も記入されていないか確認します。 &&というのは「and」の意味です（２つの条件のどちらにも該当する)。 テキストボックスが両方とも空の場合は、alert()を使って時刻を設定してもらうよう警告します。 そして関数 reSet()を呼び出し、スタートボタンを元に戻してタイマーを解除します。

テキストボックスのどちらかに記入されているなら、parseInt()を使って数値に変換していきます。 どちらかが空の場合、0 にしておきます。

そして残り時間を書き出す関数 tmWrite()を呼び出します。 その際引数に残り秒数－１を指定します（分数は×60 で秒数になります)。 ここで 1 秒減らすことによって、カウントダウンされるわけです。

関数 tmWrite()

カウントダウン関数が長くなるので、残り時間を書き出す部分を別の関数にしました。 最初に引数を parseInt()で数値に変換しています。この引数は「残り秒数－1」が指定されていました。

続いてこの引数が 0 以下か、if 文で調べます。 条件は「int==0」でも良いようですが、 最初から 0 秒やマイナスの数値を記入された場合のことも想定して、0 以下にします。

int が 0 以下であれば、関数 reSet()を呼び出してフォームを初期状態に戻し、 タイマーを停止します。そして alert()を用いて「時間です！」と表示します。

int がまだ 0 に達していない場合は、残り秒数をテキストボックスに表示します。 分数は残り秒数を 60 で割って切り捨てをした値、秒数は残り秒数を 60 で割った余りです。 余りを求めるのは「%」を使います。

関数 reSet()

フォームを初期状態に戻す処理は、このスクリプトで 2 回出てきます。 ですから一々その場に書くより、関数にして呼び出せばスッキリします。 フォーム初期化として、以下の処

理を行っています。
分数のテキストボックスに「0」を表示する。
秒数のテキストボックスに「0」を表示する。
スタートボタンの禁止状態を解除する。
タイマーを解除する