今日の日付と現在時刻

ここから JavaScript で日付と時間を扱う方法について解説していきます。 まず最初は、今日の日付と現在時刻を取得する方法について説明します。

今日の日付を取得する

まずは HTML の BODY 内に以下のコードを記入してみてください。

```
<script>

//今日の日付データを変数 hiduke に格納
var hiduke=new Date();

//年・月・日・曜日を取得する
var year = hiduke.getFullYear();
var month = hiduke.getMonth()+1;
var week = hiduke.getDay();
var day = hiduke.getDate();

var yobi= new Array("日","月","火","水","木","金","土");

document.write("西暦"+year+"年"+month+"月"+day+"日 "+yobi[week]+"曜日");

</script>
```

スクリプトの結果＝ 西暦 2014 年 5 月 12 日 月曜日

今日の日付データを取得する

上記スクリプトについて見ていきましょう。 最初に変数 hiduke に今日の日付データ一式を取得して格納します。 取得方法は new Date()を用います。

new Date()

現在の日付&時刻を取得します。

年・月・日・曜日を求める

次に、日付データの中から年・月・日・曜日をそれぞれ抜き出します。 月と曜日はそれぞれちょっと加工が必要な形で抜き出されます。

日付.getFullYear()

4 桁の西暦年を取得します。

日付.getMonth()

月を取得します。値は実際より 1 つ少ない数になります。

日付.getDay()

曜日を取得します。日曜日が 0 で、月曜が 1，…,土曜が 6 となります。

日付.getDate()
日にちを取得します。
月は 1 つ少なく取得されるので、+1 とします。 曜日は数字の形で表されるので、 文字の形で順に配列 yobi に代入しておきます。 最後に document.write()で全ての要素を書き出しています。
（この new Date()で作成された日付データ一式を正式には日付オブジェクトと呼びます)。

現在時刻を取得する
HTML の BODY 内に以下のスクリプトを記入してみましょう。

```
<script>

//時刻データを取得して変数 jikan に格納する
var jikan= new Date();

//時・分・秒を取得する
var hour = jikan.getHours();
var minute = jikan.getMinutes();
var second = jikan.getSeconds();

document.write(hour+"時",+minute+"分"+second+"秒");

</script>
```

サンプルの結果＝ 11 時 18 分 7 秒
時刻データを取得する
今度は現在時刻を求めていますが、実は new Date()で時刻も取得できます。 時刻データ一式を変数 jikan に格納しています。
時・分・秒を取得する
今度は時刻データからそれぞれ時・分・秒を抜き出します。
時刻.getHours()
時刻データの中から時間を取得します。
時刻.getMinutes()
時刻データの中から分を取得します。
時刻.getSeconds()
時刻データの中から秒を取得します。
最後に document.write()で書き出しています。

今回取得された日付や時刻は、ＰＣで設定されているものです。 ですからパソコンの日付設定などが狂っていると、 正しいデータが取得できないということを覚えておいて下さい。

期間を計算する

ここでは、任意の時まであと何日か計算する方法を見てみたいと思います。 イベントページなどでよく「開催日まであと○○日」といったカウントダウンの表示を見かけますが、JavaScript を使うと自動で計算して表示してくれます。

来年まであと何日か計算する

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>

//今日の日付データを作成する
var today = new Date();
var year1 = today.getFullYear();

//来年 1 月 1 日の日付データを作成する
var year2 = year1+1;
var newYear= new Date(year2+"/1/1");

//来年 1 月 1 日－今日のミリ秒を計算し、日にちに換算する
var day = Math.ceil((newYear-today)/(60*60*24*1000));

//表示
document.write("来年まであと"+day+"日です。");

</script>
```

サンプルの表示結果＝ 来年まであと 234 日です。

２つの日付データを作成する

では、上記のスクリプトを見ていきましょう。 最初に現在の日付データを new Date()で作成します。そして、今年は何年か getFullYear()を使って取得し、変数 year1 に代入します。次に来年 1 月 1 日の日付データを作成します。 まず来年が何年になるか、year1+1 で求めます （今 2013 年なら、変数 year2 には 2014 が代入されます)。

続いて、来年の日付データを作成します。 new Date()の括弧内に任意の日時を記入すると、その日の日付データが作成できます。 記入方法は以下の通りです。

new Date("年/月/日 時:分:秒")

任意の日付データを作成します。(時間指定も可能です)。
期間を計算する
２つの日付データが出来た所で、その両者の差を求めます。 日付データ同士を計算すると、結果はミリ秒で出ます。 ということで、それを日にちに換算します。そして出た値を切上げして表示します （残り 10 分になったとしてもまだ新年になった訳では無いので、 1 日と表示する為に切り上げとなります)。
最後に、document.write()で書き出します。
日付データの正体
さて、日付データ同士の差を求めると、なぜミリ秒単位で表示されるのでしょうか？ 実を言うと、日付データというのは 1970 年 1 月 1 日 0 時 0 分 0 秒から何ミリ秒経過しているかを表す数値なのです。 数値ですから、色々と計算するのに便利です。
1970 年 1 月 1 日より前でも計算可能です。 ご自分の生まれた日から何日経過しているか計算してみたりされるのも楽しいかも知れません。 是非日付データを駆使して、面白いスクリプトを組んでみてください。

世界時計を作る
このページでは世界各地の時刻を表示できる世界時計を作ってみたいと思います。 本当は世界地図を用意してクリックした地点の時刻を表示できればいいですが、 大変大掛かりになるのでやめます。今回は 3 箇所だけ表示できるようにしてみましょう。
（このページで解説しているのは、日本にいる場合のみ使える世界時計です）
（実際にはサマータイム等が存在するので、日付を取得して振り分けるなど複雑な処理をしなければ正確な世界時計は作れません）

世界時計のスクリプト
世界時計を作るには、日本時間を求めて、時差分だけ時刻を加減すればよさそうですが、そう簡単にはいきません。例えば日本で午前 7 時だと、イギリスやアメリカでは前日の日付になります。 ということで、以下の方法を取ります。
日本の現在の時刻を数値として取得します。
時差分を加減します。
時差補正した数値を日付データにします。
日付と時刻を取得して表示します。
まずは HTML の BODY 内に以下のフォームを記入してください。BODY タグには onload イベントを記入し、 時計を表示する関数を呼び出せるようにしておきます。 oputionタグの value 値には日本との時差（単位は時間）を設定します。
<body onload="sekaiTokei()">

```
<form name="wrdClock">
<select name="city">
<option value="0">東京・ソウル</option>
<option value="1">グラム・フィジー・シドニー</option>
<option value="-19";>ハワイ</option>
</select>
<br>
<input type="text" name="wrdTime" value="" size="30">
</form>
```

次に JavaScript 部分です。HTML のヘッダーに記入してみましょう。

```
<script>

// 1. 1 秒毎に時刻を表示する
function sekaiTokei()
{
  setInterval("timeSet()",1000);
}

function timeSet()
{

  // 2.option タグの value 値に設定した時差を取得する
  var num = document.wrdClock.city.selectedIndex;
  var jisa = parseInt(document.wrdClock.city.options[num].value);

  // 3.日本の現在の日付データを求め数値にする
  var japan= (new Date()).getTime();

  // 4.時差を加減して、日付データに戻す
  var jikan = new Date( japan + jisa*60*60*1000 );

  //日付と時刻を求める
  var year = jikan.getFullYear();
  var month= jikan.getMonth()+1;
  var day  = jikan.getDate();
  var hour = jikan.getHours();
```

```
  var minu = jikan.getMinutes();
  var sec  = jikan.getSeconds();

  // 5.時間表示の修正（10 未満は前に 0 を付ける）
  if (hour < 10) hour="0"+hour;
  if (minu < 10) minu="0"+minu;
  if (sec  < 10) sec ="0"+sec;

  //日付と時刻を表示する
  var wDate = year+"年"+month+"月"+day+"日";
  var wTime = hour+":"+minu+":"+sec;
  document.wrdClock.wrdTime.value = wDate+" "+wTime;

}
</script>
```

スクリプトの説明
では、上記スクリプトについて詳しく見ていきましょう。
1. 1 秒毎に時刻を表示する
Body タグ内の onload イベントで呼び出される関数 sekaiTokei()ですが、 setInterval()を使って時刻を表示する関数 timeSet()を 1000 ミリ秒毎に呼び出しています。
2.option タグの value 値を取得する
続いて関数 timeSet()を見ていきます。最初に option タグの value 値に記入した、日本との時差（単位は時間）を取得します。 現在選択されている option タグの番号を selectedIndex で取得し、 変数 num に代入します。 そして選択された都市の value 値を求めます。 これは文字列なので、parseInt()を使って数値に変換し、変数 jisa に代入します。
3.日本の現在の時刻を数値として求める
次に、日本の現在の時刻を求めます。 new Date()で日付データが得られますが、それを数値として取得するため、 getTime()を用います。得られる数値の単位はミリ秒になります。
日付データ.getTime()
1970 年 1 月 1 日 0 時からのミリ秒を取得する
4.時差を加減して、日付データに戻す
得られた現在時刻を表す数値に、時差を加えます。 この時差の単位は時間（hour）でしたから、ミリ秒に換算しています。
続いて、得られた数値をダイレクトに new Date()の括弧内に記入しました。 1970 年 1 月

1 日 0 時から経過したミリ秒を、括弧内に記入することができます。 これで時差の分だけずれた日付データになります。それを変数 jikan に代入しました。
あとは日付データ jikan から日付と時刻を取得します。
5.時間表示を修正して表示
時間、分、秒が 1 桁の場合に前に「0」を付けて表示するようにしています。 if 文で各要素が 10 未満か調べて、当てはまる時に、前に「0」を付け足します。
日付や月の前にも「0」を付けたいならば、同様にして付けることが可能です。
最後にフォームのテキストボックスに、日付と時刻を書き出しています。

さて、今回は日本にいる人限定の世界時計を作ってみました。 しかし別の時間帯の地域に住んでおられる方や、 現在の時刻を日本以外にしている人にはこの方法では駄目です。 それで次のページでは、標準時との時差を基準にして世界時計を作る方法について考えてみます。

標準時との時差を求める
前のページでは日本時間と他の地域の時差から世界時計を作りましたが、 日本以外の国からアクセスした人は正しく時間を表示できませんでした。 それで今度はアクセスしてきたポイントと標準時との時差を測定して、 そこから世界時計を作ってみます。

標準時との時差を求める方法
グリニッジ標準時との時差を求めるには、getTimezoneOffset()を使います。
日付データ.getTimezoneOffset()
グリニッジ標準時との時差を求める。単位は分（minute）。
日付データの部分は、現在の日付時刻を new Date()で求めれば OK です。 日本であれば 9 時間（540 分）の時差があるので、540 となりそうですが、 実際は-540 となります。 「標準時は日本時間より 540 分前になる」ということです。
具体的には下のように使います。

```
<script>

var jisa = (new Date()).getTimezoneOffset();

document.write("標準時との時差は、"+jisa/60+"時間です！");

</script>
```

上記サンプルの結果 ＝ 標準時との時差は、-9 時間です！

標準時の時差から世界時計を作る

では標準時との時差を求めて補正をする方式で、世界時計を作ってみたいと思います。 基本的には前のページのコードを利用します。 HTML の Body 内に以下のフォームを記入してください。 前のページと違うのは、 option タグの value 値に指定する数値が標準時との時差に変わっている点だけです。 onload イベントも忘れずに書いておきましょう。

```
<body onload="sekaiTokei()">

<form name="wrdClock">
<select name="city">
<option value="9">東京・ソウル</option>
<option value="10">グラム・フィジー・シドニー</option>
<option value="-10";>ハワイ</option>
</select>
<br>
<input type="text" name="wrdTime" value="" size="30">
</form>
```

次に JavaScript を記入します。以下のスクリプトを記入してみてください。 HTML のヘッダーが良いでしょう。

```
<script>

function sekaiTokei()
{
  setInterval("timeSet()",1000);
}

function timeSet()
{

  //option タグの value 値に設定した時差を取得する
  var num = document.wrdClock.city.selectedIndex;
  var jisa = parseInt(document.wrdClock.city.options[num].value);

  //アクセスした PC の日付データを求め数値にする
  var here= (new Date()).getTime();

  //グリニッジ標準時を求める
```

```
  var gmt = here + (new Date()).getTimezoneOffset()*60*1000;

  //時差を加減して、日付データに戻す
  var jikan=new Date(gmt+jisa*60*60*1000);

  //日付と時刻を求める
  var year = jikan.getFullYear();
  var month= jikan.getMonth()+1;
  var day  = jikan.getDate();
  var hour = jikan.getHours();
  var minu = jikan.getMinutes();
  var sec  = jikan.getSeconds();

  //時間表示の修正（10 未満は前に 0 を付ける）
  if (hour < 10) hour="0"+hour;
  if (minu < 10) minu="0"+minu;
  if (sec  < 10) sec ="0"+sec;

  //日付と時刻を表示する
  var wDate = year+"年"+month+"月"+day+"日";
  var wTime = hour+":"+minu+":"+sec;
  document.wrdClock.wrdTime.value = wDate+" "+wTime;

}

</script>
```

前のページと大きく変わったのは、赤文字で示した部分です。 その一つ前の行で、アクセスしている PC の現在時刻を求めて数値として変数 here に代入しています。
次に、グリニッジ標準時の数値データを計算します。数値データを代入するために変数 gmt を宣言し、 getTimezoneOffset()で求めた標準時との時差（分をミリ秒に換算）を、 現在時刻の変数 here に加えています。
あとは、変数 gmt と option タグの value 値で設定した標準時との時差を加えた値から、日付データ jikan を作成しています。

協定世界時

このページでは協定世界時を求める方法について解説します。 協定世界時（UTC）はグリ

ニッジ標準時（GMT）に変わる世界共通の標準時です （詳しくは Wikipedia をご覧下さい)。
前のページではグリニッジ標準時との時差を用いて標準時を導き出しましたが、UTC はダイレクトに求めることができます。 これを用いて世界時計を作ることができますが、しつこくなるので今回は求め方だけ解説します。

協定世界時の日付を求める
では HTML の BODY 内に以下のスクリプトを記入してみてください。

```
<script>

var hiduke = new Date();

var year = hiduke.getUTCFullYear();
var month= hiduke.getUTCMonth() +1;
var yobi = hiduke.getUTCDay();
var day   = hiduke.getUTCDate();

var week = new Array("日","月","火","水","木","金","土");

var str = "西暦"+ year + "年" + month + "月"+ day
          + "日(" + week[yobi]+ ")";
document.write(str);

</script>
```

上記スクリプトの結果 ＝ 西暦 2014 年 5 月 12 日(月)
協定世界時の日にちを求める命令文
今日の日付を求める方法と非常に似ていることに気づかれるでしょう。 一応 UTC を求める命令の前には日付データが必要なので、 new Date()で現在の日付データを作成しています。
日付データ.getUTCFullYear()
協定世界時の 4 桁の西暦年を取得します。
日付データ.getUTCMonth()
協定世界時の月を取得します。値は実際の月より 1 少ない数になります。
日付データ.getUTCDay()
協定世界時の曜日を取得します。日曜が 0，月曜が 1，…土曜が 6 になります。
日付データ.getUTCDate()

協定世界時の日にちを取得します。
曜日が数値として取得される点と、月数が 1 少なくなる点を気をつけましょう。なお「Day」が曜日で「Date」が日にちという点も間違いやすいので気をつけて下さい。
協定世界時の時刻を求める
こちらも現在時刻を求める方法とほとんど同じですが、一応見ておきましょう。 HTML の BODY 内に、以下のスクリプトを記入してみてください。

```
<script>

var jikoku = new Date();

var hour  = jikoku.getUTCHours();
var minute= jikoku.getUTCMinutes();
var second= jikoku.getUTCSeconds();
var millsc= jikoku.getUTCMilliseconds();

document.write(hour+"時"+minute+"分"+second+"秒"+millsc);

</script>
```

上記スクリプトの結果 ＝ 2 時 20 分 0 秒 36
協定世界時の時刻を求める命令文
日付データ.getUTCHours()
協定世界時の時（hour）を求めます。
日付データ.getUTCMinutes()
協定世界時の分を求めます。
日付データ.getUTCSeconds()
協定世界時の秒を求めます。
日付データ.getUTCMilliseconds()
協定世界時のミリ秒を求めます。