

ステップ30

# JavaScript

## ワークブック

Step 01	JavaScriptの概要と記述のルール	6
1.1	JavaScriptの概要	
1.2	JavaScriptを記述する場所	
1.3	JavaScriptを記述する際のルール	
Step 02	イベントハンドラの利用	12
2.1	イベントハンドラとは...?	
2.2	クリックでJavaScriptを実行 <code>onClick</code>	
2.3	マウスの移動でJavaScriptを実行 <code>onMouseOver</code> 、 <code>onMouseOut</code>	
2.4	ページが表示された直後にJavaScriptを実行する <code>onLoad</code>	
2.5	その他のイベントハンドラ	
Step 03	関数の基本	16
3.1	関数とは...?	
3.2	関数の作成	
3.3	関数名に指定できない文字	
3.4	イベントハンドラから関数を呼び出す	
Step 04	変数の宣言	20
4.1	変数の宣言	
4.2	変数に数値を代入する	
4.3	変数に文字を代入する	
4.4	変数の表示	
Step 05	変数の演算	25
5.1	演算子の記述	
5.2	プログラムならではの記述方法	
5.3	インクリメントとデクリメント	
5.4	文字の足し算	
5.5	変数に文字を追加して表示する	
Step 06	配列	29
6.1	配列とは...?	
6.2	配列の宣言	
6.3	配列に値を代入する	
Step 07	関数の引数	32
7.1	関数の引数とは...?	
7.2	引数の指定方法	
7.3	引数を活用して関数を汎用化する	
7.4	複数の引数を指定する場合	
Step 08	繰り返し処理	37
8.1	繰り返し処理とは...?	

8.2	繰り返し処理( <code>for</code> )の記述
8.3	比較演算子
8.4	繰り返し処理の活用例

Step 09	繰り返し処理と文字の表示	43
9.1	2重ループの繰り返し処理	
9.2	ホームページに文字を表示する <code>document.write()</code>	
9.3	<code>document.write()</code> でHTMLタグを出力する	
9.4	JavaScriptで九九の表を作成する	
Step 10	条件分岐 - 1	49
10.1	条件分岐とは...?	
10.2	if文の記述方法	
10.3	if ~ else で処理を2つに分岐させる	
10.4	if ~ else のサンプルプログラム	
Step 11	条件分岐 - 2	54
11.1	else if で処理を3つ以上に分岐させる	
11.2	論理演算子	
11.3	switch文で処理を分岐させる	
Step 12	breakとcontinue	60
12.1	breakの利用方法	
12.2	continueの利用方法	
12.3	while文による繰り返し	
Step 13	関数の戻り値	65
13.1	戻り値とは...?	
13.2	関数から戻り値を受け取るには...?	
13.3	returnで戻り値を返す	
13.4	戻り値を利用したJavaScriptの例	
Step 14	文字入力とエラー処理	71
14.1	プロンプトの利用方法	
14.2	プロンプトを活用したJavaScript	
14.3	エラー対策用の処理	
Step 15	オブジェクト	76
15.1	オブジェクトとは...?	
15.2	オブジェクトモデル	
15.3	オブジェクトと配列	
15.4	オブジェクトとname属性	
15.5	メソッドとは...?	
15.6	プロパティについて	

Step 16	ウィンドウの操作	81
	16.1 新しいウィンドウを開く <code>window.open()</code>	
	16.2 ウィンドウを閉じる <code>window.close()</code>	
	16.3 オブジェクト変数の作成	
	16.4 ウィンドウサイズの変更 <code>window.resizeTo()</code> 、 <code>window.resizeBy()</code>	
Step 17	スクロールの操作	87
	17.1 ページをスクロールさせる <code>window.scrollTo()</code>	
	17.2 現在位置を基準にページをスクロールさせる <code>window.scrollBy()</code>	
	17.3 スクロールの活用例	
Step 18	文字色、背景色の操作	92
	18.1 文字色、背景色の変更	
	18.2 ドキュメント情報を表示する	
	18.3 文字が浮かび上がってくるホームページ	
Step 19	画像の操作 - 1	97
	19.1 Imageオブジェクトのプロパティ	
	19.2 <code>this</code> の利用	
	19.3 <code>onMouseOver</code> イベントで画像を変更する	
	19.4 <code>onMouseOut</code> イベントで画像を元に戻す	
Step 20	画像の操作 - 2	101
	20.1 Imageオブジェクトの指定	
	20.2 サムネイル画像のマウスオーバーで拡大画像を変更する	
	20.3 ボタンのクリックで画像の表示サイズを変更する	
Step 21	日付、時刻の操作 - 1	106
	21.1 Dateオブジェクトの作成	
	21.2 Dateオブジェクトから年月日、時分秒を取り出す	
	21.3 現在の日時を表示するJavaScript	
	21.4 更新日時を表示する	
Step 22	日付、時刻の操作 - 2	111
	22.1 日時を指定するメソッド	
	22.2 指定日までの残り日数を計算するJavaScript	
Step 23	フォームの操作 - 1	115
	23.1 Formオブジェクトの指定	
	23.2 Formオブジェクトの下位オブジェクトを指定する	
	23.3 テキストボックスとテキストエリア	
	23.4 チェックボックス	
	23.5 テキストボックスとチェックボックスの活用例	

Step 24	フォームの操作 - 2	121
	24.1 ラジオボタン	
	24.2 ラジオボタンの活用例	
	24.3 セレクトボックス	
	24.4 セレクトボックスの活用例	
Step 25	URLと履歴の操作	127
	25.1 locationオブジェクト	
	25.2 locationオブジェクトの活用例	
	25.3 historyオブジェクト	
Step 26	Mathオブジェクト	131
	26.1 Mathオブジェクトとは...?	
	26.2 Mathオブジェクトのプロパティ	
	26.3 Mathオブジェクトのメソッド	
	26.4 乱数を利用したJavaScript	
Step 27	stringオブジェクト	137
	27.1 stringオブジェクトのプロパティ	
	27.2 メソッドで文字の書式を指定する	
	27.3 メソッドで文字にさまざまな処理を行う	
Step 28	一定間隔で処理を繰り返し実行する	141
	28.1 <code>setTimeout()</code> の活用	
	28.2 <code>setInterval()</code> の活用	
	28.3 <code>setInterval()</code> の中断	
	28.4 スライドショーの作成	
Step 29	クッキーの操作 - 1	147
	29.1 クッキー(Cookie)とは...?	
	29.2 クッキーの利用方法	
	29.3 クッキーの有効期限を指定する	
	29.4 クッキーの活用例	
Step 30	クッキーの操作 - 2	153
	30.1 複数の値をクッキーに保存する	
	30.2 クッキーへの入出力を汎用化する	
	30.3 クッキーへ値を書き込む関数「 <code>setCookie()</code> 」	
	30.4 クッキーから値を読み出す関数「 <code>getCookie()</code> 」	
	30.5 クッキーに訪問回数と名前を保存する	

さくいん	160
演習問題の解答	161

# JavaScriptの概要と記述のルール

JavaScriptは、一般的なプログラムのような動作をホームページ上で実現できるスクリプト言語です。JavaScript学習の第一歩となるステップ01では、JavaScriptの概要、およびJavaScriptの記述方法について学習します。

## 1.1 JavaScriptの概要

ホームページを作成するには、テキストエディタなどでHTMLを記述し、HTMLファイルを作成するのが一般的です。また、HTMLファイル内にCSSを記述し、ホームページ内の各要素について書式を指定することも可能です。これらをまとめると、HTMLファイルは以下の2つの言語から構成されると考えられます。

- ・HTML ..... ホームページに掲載する文章や画像を指定したり、ホームページ全体のレイアウトを指定したりする
- ・CSS ..... ホームページ内の各要素について、色やサイズ、枠線などの書式を指定する

もちろん、これらの2言語だけでもホームページは十分に作成できます。ただし、このようにして作成されたホームページは“動きのないホームページ”にしかありません。一方、JavaScriptを利用してホームページを作成すると、閲覧者の操作に応じて“動きのあるホームページ”を作成できるようになります。

たとえば、『ボタンをクリックすると画像表示を変更する』などの演出をJavaScriptで実現することも可能です。

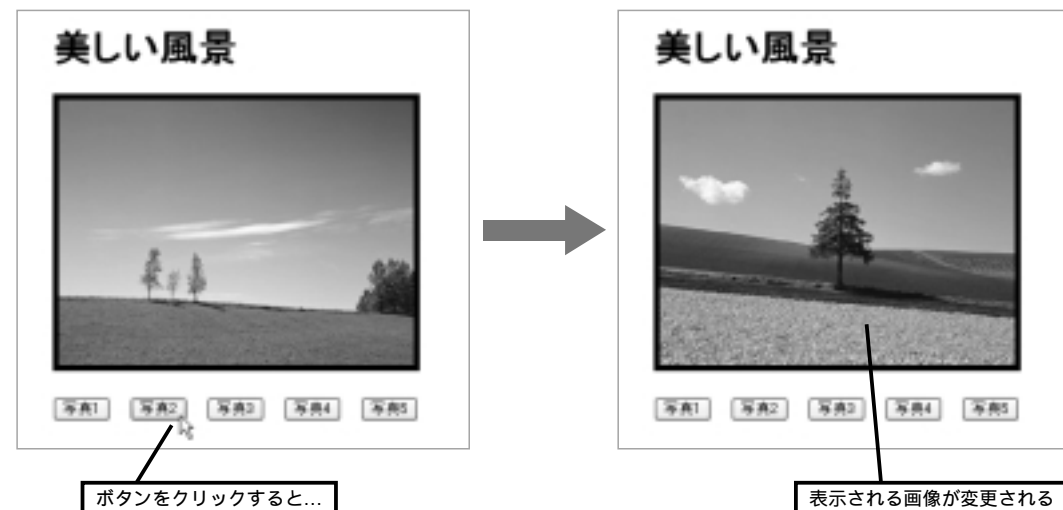


図1-1 JavaScriptを利用したホームページの演出

また、『選択された項目について合計金額を算出する』など、実用的な仕組みをJavaScriptで実現することも可能です。

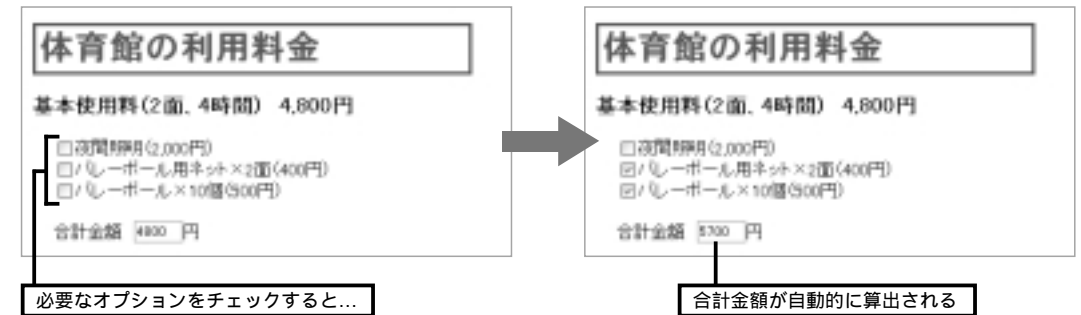


図1-2 JavaScriptを利用した実用的なホームページ

このように、閲覧者の操作に応じて、ホームページにさまざまな処理をプログラミングできるのがJavaScriptです。

## 1.2 JavaScriptを記述する場所

続いては、JavaScriptの記述方法について解説します。通常、JavaScriptを利用したホームページは、HTMLファイル内にJavaScriptを記述するのが一般的です。たとえば、図1-2の場合、HTMLファイルの内容は以下ようになります。

現時点では、JavaScriptの記述内容を理解できなくても構いません。JavaScriptの記述内容については、後のステップで詳しく解説していきます。

Sample1\_1.html

```
01 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
02 "http://www.w3.org/TR/html4/loose.dtd">
03
04 <HTML>
05
06 <HEAD>
07 <TITLE>体育館の利用料金</TITLE>
08 <META http-equiv="Content-Type" content="text/html; charset=Shift_JIS">
09
10 <STYLE type="text/css">
11 <!--
12 H1{
13   border:solid 3px #993300;
14   padding:3px;
15   color:#993300;
16 }
```

CSS

(次ページへ続く)

```

16 width:400px;
17 }
18 DIV.st1{
19   font-size:14pt;
20   font-weight:bold;
21 }
22 FORM{margin-left:20px;}
23 -->
24 </STYLE>
25
26 <SCRIPT language="JavaScript">
27 <!--
28 var total = 4800;
29 var a = new Array(3);
30 a[0] = 2000;
31 a[1] = 400;
32 a[2] = 500;
33 function goukei(i){
34   if(document.f1.elements[i].checked == true){
35     total = total + a[i];
36   } else {
37     total = total - a[i];
38   }
39   document.f1.elements[3].value = total;
40 }
41 //-->
42 </SCRIPT>
43
44 </HEAD>
45
46 <BODY>
47 <H1>体育館の利用料金</H1>
48 <DIV class="st1">基本使用料(2面、4時間) 4,800円</DIV>
49 <FORM name="f1">
50 <INPUT type="checkbox" onClick="goukei(0)">夜間照明(2,000円) <BR>
51 <INPUT type="checkbox" onClick="goukei(1)">バレーボール用ネット×2面(400円)
52 <BR>
53 <INPUT type="checkbox" onClick="goukei(2)">バレーボール×10個(500円) <BR>
54 合計金額 <INPUT type="text" size="6" value="4800">円
55 </FORM>
56 </BODY>
57
58 </HTML>

```

このように、HTMLファイルには以下の3つの言語を含むことが可能です。

(HTMLファイルに必須となる言語)

- ・HTML ..... ホームページに掲載する文章や画像を指定したり、ホームページ全体のレイアウトを指定したりする

(必要に応じてHTMLに追記する言語)

- ・CSS ..... ホームページ内の各要素について書式を指定する
- ・JavaScript ... 閲覧者の操作に応じて、ホームページの処理をプログラミングする

### 1.3 JavaScriptを記述する際のルール

これで、JavaScriptを記述する場所は理解できたと思います。続いては、JavaScriptを記述する際のルールについて解説します。命令文や各種指定などを半角英数字で記述する点はHTMLやCSSと同じですが、JavaScriptでは大文字/小文字が区別されることに注意してください。そのほか、JavaScriptを記述する際のルールは以下のとおりです。

- ・ **<SCRIPT>** ~ **</SCRIPT>** の間にJavaScriptを記述する  
HTMLファイル内にJavaScriptを記述する場合は、**<SCRIPT>** ~ **</SCRIPT>** 内にJavaScriptを記述しなければいけません。また、**language**属性に**"JavaScript"**を指定し、タグ内の記述がJavaScriptであることを示しておく必要があります。
- ・ コメントを示す**<!-- ~ //-->**を記述する  
SCRIPTタグに対応していないWebブラウザでは、JavaScriptの記述がそのまま画面に表示されてしまう場合があります。これを防ぐには、JavaScriptの記述を**<!--と//-->**で挟み、JavaScriptの記述をコメント文として処理する必要があります。

```

<SCRIPT language="JavaScript">
<!--
:
:
(ここにJavaScriptを記述)
:
:
//-->
</SCRIPT>

```

- ・ JavaScriptは半角文字で記述する  
JavaScriptの命令文や各種指定は半角文字で記述します。これらの記述に全角文字を使用すると、正しくJavaScriptが動作しません。

・大文字と小文字は区別される

JavaScriptでは、大文字と小文字が別の文字として認識されます。たとえば「ABC」「abc」「Abc」は、いずれも別の文字となります。HTMLやCSSのように、大文字/小文字を無視して記述できない点に十分注意してください。

・命令文の最後にセミコロン

JavaScriptでは、各文の最後に「;」(セミコロン)を記述し、文の区切りを明確に示す必要があります。CSSの場合と同様、改行は無視される点に注意してください。

正しい記述

```
var a = 1;
var b = 2;
var c;
c= a + b;
```

×間違った記述

```
var a = 1
var b = 2
var c
c= a + b
```

・半角スペース、タブ文字の扱い

JavaScriptでは、連続した半角スペースおよびタブ文字は無視されます。このため、行頭に半角スペースやタブ文字でインデント(余白)を設け、JavaScriptを見やすく記述することも可能です。

```
function abc(){
  c= a + b;
  d= c * 1.05;
  alert(a);
}
```

ワンポイント

JavaScriptに対応していないWebブラウザへの対応

JavaScriptに対応していないWebブラウザに対して、ページにJavaScriptが含まれていることを示す場合は、<NOSCRIPT> ~ </NOSCRIPT>を利用します。このタグ内に記述された内容は、JavaScriptに対応していないWebブラウザにだけ表示されます。

```
<SCRIPT language="JavaScript">
<!--
:
(JavaScriptの記述)
:
/-->
</SCRIPT>
<NOSCRIPT>
このページにはJavaScriptが使用されています。<BR>
JavaScriptを有効にしてから閲覧してください。
</NOSCRIPT>
```

演習

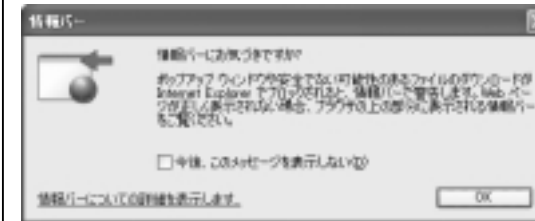
(1) P7の「Sample1\_1.html」に示したHTMLファイルを作成してみましょう。

(2) 演習(1)で作成したHTMLファイルをWebブラウザに表示し、チェックボックスの操作に応じて合計金額が変化することを確認してみましょう。

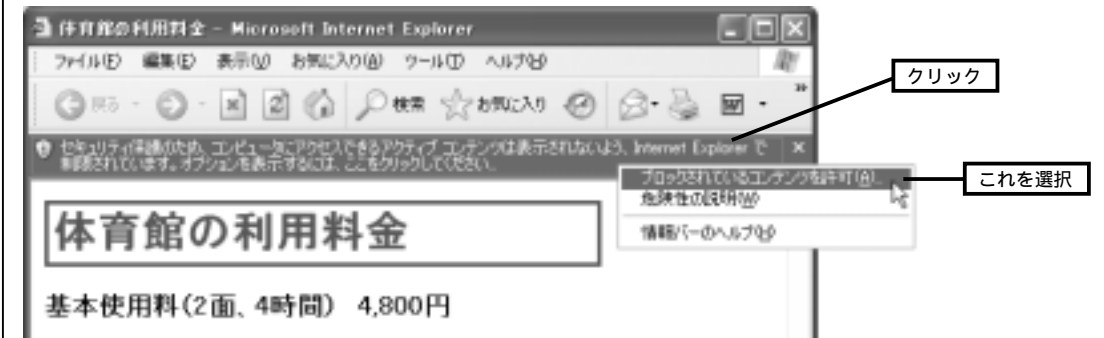
ワンポイント

Internet ExplorerでJavaScriptの実行を許可する

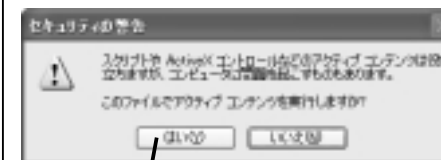
ハードディスクに保存されているHTMLファイルをInternet Explorerで閲覧した場合、以下のようなメッセージが表示される場合もあります。



このメッセージが表示された場合は、Internet Explorerのセキュリティ保護によりJavaScriptが無効になっています。よって、記述したJavaScriptも実行されません。JavaScriptを実行させるには、以下のように操作してセキュリティ保護を解除する必要があります。



情報バーをクリックし、「ブロックされているコンテンツを許可」を選択します。



このようなウィンドウが表示されるので、「はい」ボタンをクリックします。すると、セキュリティ保護が解除され、JavaScriptが実行可能となります。

## イベントハンドラの利用

マウス操作などに応じてJavaScriptを実行するには、「どのタイミングでJavaScriptを実行するか？」をHTMLファイル内で指定しておく必要があります。このタイミングを指定するのがイベントハンドラです。ステップ02では、イベントハンドラの指定方法について学習します。

### 2.1 イベントハンドラとは...?

HTMLファイル内に記述したJavaScriptを正しく動作させるには、JavaScriptを実行するタイミングを指定しておく必要があります。このタイミング指定に利用するのがイベントハンドラです。イベントハンドラには、「マウスでクリックしたとき」「マウスポインタを移動したとき」など、操作内容ごとに個別のイベントハンドラ名が決まっています。そして、HTMLのタグ内に以下のような形式でイベントハンドラを記述することにより、JavaScriptを実行するタイミング、および実行するJavaScriptを指定します。

イベントハンドラ名="実行するJavaScript"

### 2.2 クリックでJavaScriptを実行 onClick

ホームページにある要素をクリックした際にJavaScriptを実行させるには、**onClick**イベントを利用します。たとえば、画像をクリックした際にJavaScriptを実行させる場合、そのIMGタグにonClickイベントを追加し、その値にJavaScriptを記述します。

```
<IMG src="photo01.jpg" onClick="alert('北海道の風景写真です!')">
```

上の例では、「onClick」の部分イベントハンドラ名、「alert('北海道の風景写真です!')」の部分JavaScriptとなります。なお、**alert()**はメッセージウィンドウを表示するJavaScript命令(メソッド)であり、メッセージとして表示する文字をカッコ内に「!」(シングルクォーテーション)で挟んで記述します。

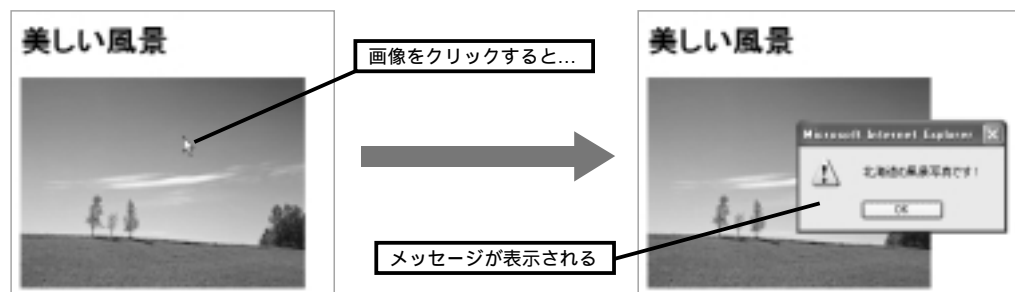


図2-1 onClickイベント

### 2.3 マウスの移動でJavaScriptを実行 onMouseOver、onMouseOut

要素の上にマウスポインタが来た際にJavaScriptを実行させるには、**onMouseOver**イベントを利用します。以下の例の場合、マウスポインタを画像の上に移動させると、「北海道の風景写真です!」というメッセージウィンドウが表示されます。

```
<IMG src="photo01.jpg" onMouseOver="alert('北海道の風景写真です!')">
```

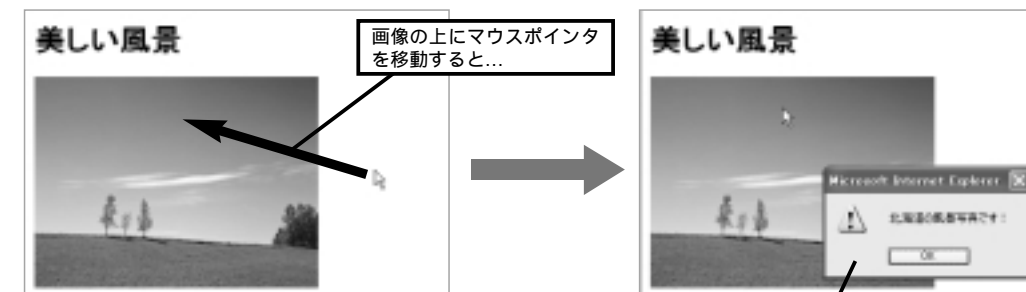


図2-2 onMouseOverイベント

これとは逆に、要素の外へマウスポインタを移動した際にJavaScriptを実行させるには、**onMouseOut**イベントを使用します。

```
<IMG src="photo01.jpg" onMouseOut="alert('北海道の風景写真です!')">
```

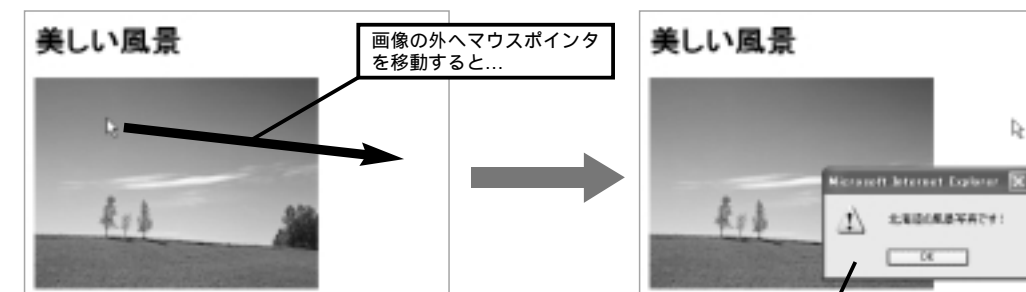


図2-3 onMouseOutイベント

### 2.4 ページが表示された直後にJavaScriptを実行する onLoad

ページが表示された直後にJavaScriptを実行する場合は、**onLoad**イベントを使用します。onLoadイベントはBODYタグに記述するのが普通であり、この場合、特に操作を行わなくても自動的にJavaScriptが実行されます。たとえば、次ページの例では、ホームページ全体が表示された直後に「ようこそホームページへ」というメッセージウィンドウが表示されます。

```
<BODY onLoad="alert('ようこそホームページへ')">
```

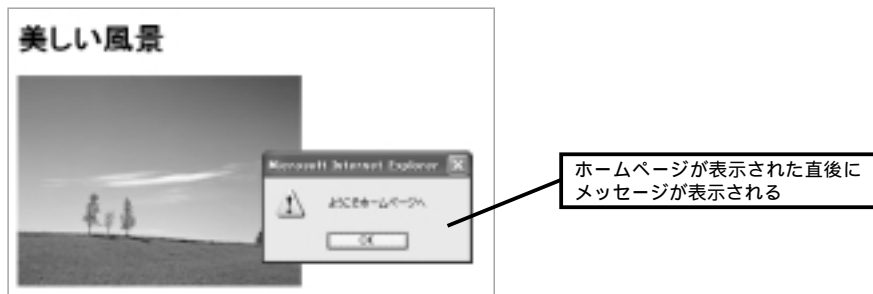


図2-4 onLoadイベント

## 2.5 その他のイベントハンドラ

これらのほかにも、JavaScriptを実行するタイミングに応じて以下のようなイベントハンドラが用意されています。

- ・ **onAbort**  
Webブラウザの[ 停止 ]ボタンをクリックした場合などで、画像の読み込みが中断された場合にJavaScriptを実行します。
- ・ **onBlur**  
その要素からフォーカスが離れた際にJavaScriptを実行します。  
フォーカスは、[ Tab ]キーやマウスのクリックで移動できます。
- ・ **onChange**  
フォームの内容を変更した際にJavaScriptを実行します。
- ・ **onError**  
ページや画像の読み込み時にエラーが発生した場合にJavaScriptを実行します。
- ・ **onFocus**  
その要素にフォーカスが移動した際にJavaScriptを実行します。
- ・ **onReset**  
フォームがリセットされた場合にJavaScriptを実行します。
- ・ **onSelect**  
テキストボックスやテキストエリア内をクリックし、フォームへの入力が可能になった際にJavaScriptを実行します。

- ・ **onSubmit**  
フォームのsubmitボタンをクリックした際にJavaScriptを実行します。
- ・ **onUnload**  
別のページへ移動する際にJavaScriptを実行します。

### 演習

(1) 以下のようなHTMLファイルを作成し、画像をクリックすると、メッセージウィンドウが表示されることを確認してみましょう。

```
01 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
02 "http://www.w3.org/TR/html4/loose.dtd">
03
04 <HTML>
05
06 <HEAD>
07 <TITLE>イベントハンドラの練習</TITLE>
08 <META http-equiv="Content-Type" content="text/html; charset=Shift_JIS">
09 </HEAD>
10
11 <BODY>
12 <IMG src="tahiti01.jpg" onClick="alert('タヒチの写真です!')">
13 </BODY>
14
15 </HTML>
```

この演習で使用する画像「tahiti01.jpg」は、以下のURLからダウンロードできます。  
[http://www.cutt.co.jp/book/html\\_818.html](http://www.cutt.co.jp/book/html_818.html)

- (2) onClickイベントをonMouseOverイベントに変更し、メッセージウィンドウが表示されるタイミングがどのように変化するかを確認してみましょう。
- (3) onMouseOverイベントをonMouseOutイベントに変更し、メッセージウィンドウが表示されるタイミングがどのように変化するかを確認してみましょう。
- (4) ページが表示された直後にメッセージウィンドウが表示されるようにイベントハンドラを書き換えてみましょう。