# 応用情報処理 No. 10

# **JavaScriptの応用**

## １．イベントハンドラ

マウスでボタンをポイントしたり、クリックしたりすることや、ウィンドウのサイズを変えたりなどの何らかの操作をイベント(event)という。あるイベントが発生したときに実行する関数などのJavaScriptのコードのことをイベントハンドラ(event handler)という。

例えば、ボタン<input type=”button”>をクリックされたというイベントに対して、onClickという属性の値としてイベントハンドラにstart\_uranai()関数を設定した[[1]](#footnote-1)ことは、すでに学習した通りである（No.9、6節）。

## ２．マウスの生成するイベント

画像にリンクを張ったとき、その画像をポイントすると、画像が変化して（ロールオーバーするとも言われる）、リンク先にジャンプ可能であることを知らせることは効果的であり、これを使用したwebページをよく見かける。これを実現する為には、マウスで画像をポイントした、またはポイントを解除したというイベントで画像を切り替える処理するようにすればよい。

マウスでポイントするというイベントを処理する簡単な例として、画像をポイントすると、別の画像を表示するJavaScriptのプログラムを紹介しよう。cat.gif、revcat.gif（200×200ピクセル）を用意して、以下のソースファイルを作成しよう。GIF画像を作成するときは、MS ペイントでGIF形式にするよりも、GIMPで書きだすと綺麗な画像にできる。

[onmouse.html]

<html>

<head>

<title>Roll over</title>

<script type="text/javascript">

<!--

function chg\_image(mySrc, ImageElement){

 ImageElement.src = mySrc;

}

//-->

</script>

</head>

<body>

<img src="cat.gif" border="0" name="cat"

onMouseOver="chg\_image('revcat.gif', cat)"

 onMouseOut="chg\_image('cat.gif', cat)" >

</body>

</html>

【解説】

(1) img要素のonMouseOver、 onMouseOut 属性

画像にはimg要素のname属性で「cat」という名前がつけられていることに注意する。

onmouse.htmlが格納されているフォルダには、cat.gifとrevcat.gifという2種類の画像が格納されている必要がある。

img要素には、マウスでポイントされたときのイベントハンドラはonMouseOver、マウスのポイントが解除されたときのイベントハンドラはonMouseOutの属性の値として設定するので、以下のように記述されている。

onMouseOver="chg\_image('revcat.gif', cat)"

onMouseOut="chg\_image('cat.gif', cat)”

ともに、chg\_image()というJavaScriptで定義した関数を呼び出している。

chg\_image( , )関数は第1引数に画像のファイル名、第2引数にimg要素の名前（name属性の値）を指定することで、name属性がcatのimg要素の画像を第1引数の画像に変更するという働きをする。その定義は、具体的には、

function chg\_image(mySrc, ImageElement){

 ImageElement.src = mySrc;

}

といった単純なものである。img要素はこの関数内では、ImageElementという名前で参照されており、ImageElementオブジェクトのsrcプロパティ（つまり、ImageElement.src）に第1引数の画像ファイル名mySrcを代入するという内容である。

Do! onmouse.htmlを作成し、ブラウザで動作を確かめなさい。

## ３．thisキーワードを使う方法

JavaScriptにはthisキーワードという便利な表記法があり、プログラム実行時の対象となっているオブジェクトをあらわすことができる。例えば、前節の例では、img要素である画像にname属性で“cat”という名前をつけたが、onMouseOverなどの値として設定されたイベントハンドラでは、catの代わりにthisと書き換えることができる。それと同時にimg要素にname属性で名前を付ける必要がなくなる。

[onmouse\_this.html] (※網掛けされていないところが変更点である。)

<html>

<head>

<title>Roll over</title>

<script type="text/javascript">

<!--

function chg\_image(mySrc, ImageElement){

 ImageElement.src = mySrc;

}

//-->

</script>

</head>

<body>

<img src="cat.gif" border="0"

onMouseOver="chg\_image('revcat.gif', this)"

 onMouseOut="chg\_image('cat.gif', this)" >

</body>

</html>

Do! onmouse\_this.htmlを作成し、ブラウザで動作を確かめなさい。

更に、chg\_imageのようにコードが1行しかない関数の場合は、わざわざfunctionでchg\_imageを定義しなくても、JavaScriptのコードをイベントハンドラとして＝の後に”（二重引用符）で囲って直接書き込むことができる。functionは削除して、次のようにできる。

[onmouse\_simple.html] (※網掛けされていないところが変更点である。)

<html>

<head>

<title>Roll over</title>

</head>

<body>

<img src="cat.gif" border="0"

onMouseOver="this.src='revcat.gif'"

 onMouseOut="this.src='cat.gif'" >

</body>

</html>

Do! onmouse\_simple.htmlを作成し、ブラウザで動作を確かめなさい。

## ４．応用例：画像リストの画像を拡大表示するページ

cat.gif、revcat.gifを32x32ピクセルに縮小して並べて表示し、マウスでポイントすると、下に300ピクセルの幅で拡大表示するように改造した。この場合、onMouseOutは不要であることに注意せよ。

[onmouse02.html] (※網掛けされていないところがonmouse.htmlからの変更点である。)

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=Shift\_JIS">

<title>Roll over</title>

<script type="text/javascript">

<!--

function chg\_image(mySrc, ImgElmnt){

 ImgElmnt.src = mySrc;

}

//-->

</script>

</head>

<body>

<h2>拡大したい画像をポイントしてください</h2>

<p>

<img src="cat.gif" width="32" height="32" border="1"

 onMouseOver="chg\_image('cat.gif', cat)" >

<img src="revcat.gif" width="32" height="32" border="1"

 onMouseOver="chg\_image('revcat.gif', cat)" >

</p>

<p align="center">

<img src="cat.gif" width="300" border="0" name="cat" >

</p>

</body>

</html>

Do! onmouse02.htmlを作成し、ブラウザで動作を確かめなさい。

Do! onmouse02.htmlをコピーしてonmouse03.htmlとし、cat.gif、revcat.gif以外に画像をもう一枚作成して追加しなさい。

# **簡単なゲーム**

JavaScriptでアニメーションを実現するには、一定時間ごとに表示を切り替えることが必要である。一定時間後に処理を実行することをタイマー処理という。

　一定時間ごとに繰り返し実行するタイマー処理には、setInterval関数を使うと良い[[2]](#footnote-2)。

 タイマー識別子 = setInterval(実行する関数, 実行までの時間)

という書き方をする。

　タイマーを停止するには、clearInterval関数を使う。

 clearInterval(タイマー識別子)

という書き方をする。

## ５．熊たたきゲーム ver. 1

熊をクリックして、動きを止めるゲーム

※あらかじめ、教材ドライヴ（T:）からコピーしてsbuc.gifというクマの画像ファイルを用意する。

[bear01.html]

<html>

<head>

<title>Hit the bear!</title>

</head>

<body>

<h2>熊たたきゲーム</h2>

<img src="sbuc.gif" id="bear"

onClick="clearInterval(MyTime); alert('Ouch!');">

<script type="text/javascript">

<!--

var t = 0;

var IMG = document.getElementById('bear');

MyTime = setInterval( 'Animate()', 50 );

function Animate(){

 IMGstyle = IMG.style;

 IMGstyle.position = 'absolute';

 IMGstyle.left = Math.sin( t \* 0.1 ) \* 200 + 350 + 'px';

 IMGstyle.top = Math.cos( t \* 0.1) \* 200 + 150 + 'px';

 t++;

}

//-->

</script>

</body>

</html>

【解説】

クマの画像ファイルsbuc.gifを表示するimg要素には、bearというidを付けておく。

IMG = document.getElementById('bear')を実行すると、IMGでこのimg要素を（DOMのオブジェクトとして）参照できる。

Animate()は、IMGの表示位置を変更する関数である。

## ６．熊たたきゲーム ver. 2

※あらかじめ、倒れたクマの画像SB\_down.gifを用意しておこう。

クマをクリックしたとき、クマの画像を別のものに変えるようしよう。

[bear02.html] (※網掛けされていないところがbear01.htmlからの変更点である。)

<html>

<head>

<title>Hit the bear!</title>

</head>

<body>

<h2>熊たたきゲーム</h2>

<img src="sbuc.gif" id="bear"

onClick="hit();">

<script type="text/javascript">

<!--

var t = 0;

var IMG = document.getElementById('bear');

MyTime = setInterval( 'Animate()', 50 );

function Animate(){

 IMGstyle = IMG.style;

 IMGstyle.position = 'absolute';

 IMGstyle.left = Math.sin( t \* 0.1 ) \* 200 + 350 + 'px';

 IMGstyle.top = Math.cos( t \* 0.1) \* 200 + 150 + 'px';

 t++;

}

function hit()

{

 IMG.src = 'SB\_down.gif';

 clearInterval(MyTime);

 alert('Ouch!');

}

//-->

</script>

</body>

</html>

## ７．問題　bear02.htmlで、Animate関数を以下のように変更すると、どうなるか答えなさい。

function Animate(){

 IMGstyle = IMG.style;

 IMG.style.position = ' absolute';

 IMG.style.left = Math.sin( t \* 0.1 ) \* 350 + 350 + "px";

 IMG.style.top = Math.cos( t \* 0.5 ) \* 20 + 150 + "px";

 t++;

}

## 課題１０　クマたたきゲーム（の画像や、その動き方など）を改造して、game.htmlを作成しなさい。

# ９. Check List

* イベントとは何か？
* イベントハンドラとは何か？
* onClick属性の値は、どのようなイベントのイベントハンドラであるか？
* onMouseOver属性の値は、どのようなイベントのイベントハンドラであるか？
* onMouseOut属性の値は、どのようなイベントのイベントハンドラであるか？
* onMouseDown属性の値は、どのようなイベントのイベントハンドラであるか？
* setInterval()関数はどのような関数か？
* clearInterval()関数はどのような関数か？
1. しばしばonClickのことをイベントハンドラと呼ぶことがあるので注意すること。本来は、onClick属性の値が、clickイベントのイベントハンドラであるというのが正しい。 [↑](#footnote-ref-1)
2. 一定時間ごとの繰り返し処理を行う方法として一般的なのはsetInterval関数を使う方法であるが、setTimeout関数を使うこともできる。setTimeout()関数では、前の処理が終了してから一定時間後に次の処理を実行することから遅延が生じることがあるが、今回のマウスでクリックするイベントを扱う場合には適しているかもしれない。 [↑](#footnote-ref-2)