□ Webページとスクリプト

ユーザーの入力に応答するWebページの原理

今日のブラウザ上では非常に高度なアプリケーションが動作可能になっています。これは、ブラウザ上で動くスクリプト（プログラム）とサーバー上で動くプログラムが緊密に連携することで初めて可能になっています。

サーバー側スクリプト

最初にWWWができたときには、Webページは単にサーバーから送らてきたコンテンツを表示する機能しか持っていませんでした。しかしまもなく、ユーザーに調べたいと思う語を入力させてその応答を返す機能が追加され、HTML 2.0では一般的な入力**フォーム**をページ内に含められるようになりました。

入力フォームの機能は、ユーザーがフォーム中の入力部品に値を設定した後に送信用のボタンを押すと、入力された値がWebサーバーに送られ、Webサーバー上のプログラム（**CGIプログラム**と呼ばれます）で処理され、結果のページが返される、という形が基本です（図1）。

これを**サーバー側スクリプト**（server-side scripting）と呼びます。フォームとサーバー側スクリプトを組み合わせることで、ブラウザさえあればネットのどこからでも使える便利なアプリケーションが作れます。

ただし、サーバー側スクリプトだけを用いる方式だと、「送信→処理→返送→表示」に1秒程度はかかるため、あまり高度なインターフェースは作れませんでした。

クライアント側スクリプト

その後、HTMLにJavaScript言語プログラムを含めてサーバーからブラウザに送らせ、ブラウザ上でプログラムが動作していろいろな処理を行う方式が考案されました。これをWebクライアント（ブラウザ）上でプログラムが動くことから**クライアント側スクリプト**（client-side scripting）と呼びます。クライアント側スクリプトによって、ブラウザ上で単独で計算して結果を表示するようなプログラムが作れるようになりました。

また、JavaScriptを使ってブラウザ上のHTMLをその場で書き換え、見た目を柔軟に変化させるページも作れるようになりました。これをDHTML（**ダイナミックHTML**）などと呼ぶこともあります。

サーバー側とクライアント側の連携

やがてクライアント側スクリプトは、ブラウザ上で単独で動くだけでなく、Webサーバー側とも通信してデータを受け渡しながら、使いやすいインターフェースを提供する形でも使われるようになりました（図2）。

一方、サーバー側スクリプトは、データベースとも連携して必要な処理を行い、結果をクライアント側に返します。このように両方のスクリプトが連携することで、高度な処理が行えるようになっているのです。

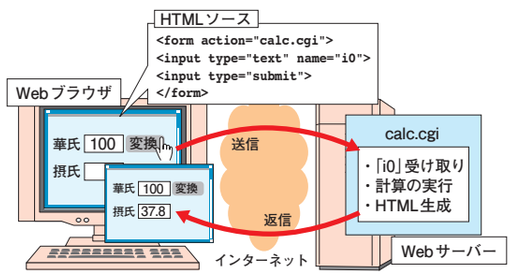

図1　入力フォームの仕組み

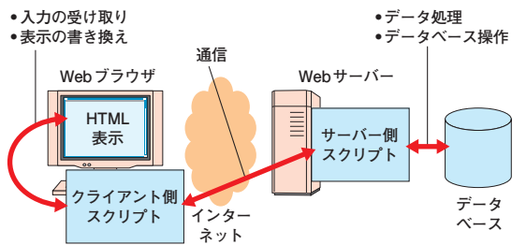

図2　サーバー側スクリプトとクライアント側スクリプトが連携

スクリプトでHTMLを操作する

JavaScriptではフォーム部品ではない一般のHTML要素に変更を加えることも可能です。スクリプトの起動は、表1のような属性により指定できます。

スクリプト側からHTML文書の内容をオブジェクトとしてアクセスする枠組みを**DOM**（Document Object Model）と呼びます。それぞれのHTML要素に対応するオブジェクトの取得は、id属性を指定して関数document.getElementByIdを呼ぶことで行えます。

HTML要素に対する操作で一番簡単なのは、**innerHTML**という属性に値を入れてその要素の内側を書き換えるというものです。名前のとおり、任意の文字列はHTMLとして解釈されるので様々な要素を作り出すこともできます。また、要素の親を取得したり、子供の（内側に入っている）要素からどれかを削除したりもできます。

図3では、ページ本体のdiv要素中に3つのspan要素があり、その3つ目にonmouseover、onmouseout、onclickが指定されています。最初の2つは関数msgを呼び出し、その中でspanオブジェクトのinnerHTMLを書き換えることで表示を変更します。クリック時にはspan要素の親オブジェクト（この場合はdiv）を取得し、その先頭要素を取り除いて末尾に入れ直すことで「A」「B」と当該spanの順番を変化させています。

また、いちいちid属性を指定しなくても、特定の要素の中に入っているすべての要素を取り出してきて内容に応じて加工を施したり、ユーザー操作に応答する仕組みを組み込んだりする処理を、JavaScriptプログラムを通じて行うこともできるのです。

```
<!DOCTYPE html>
<html><head>
<meta charset="utf-8">
<title>Dom Demo</title>
<script type="text/javascript">
function msg(s) {
 var s0 =
  document.getElementById('s0');
 s0.innerHTML = s;
}
function move() {
 var s0 =
  document.getElementById('s0');
 var p = s0.parentNode;
 var e = p.firstChild;
 p.removeChild(e);
 p.appendChild(e);
}
</script></head><body>
<div><span>A</span>
<span>B</span>
<span id="s0"
 onmouseover="msg('はい')"
 onmouseout="msg('いいえ')"
 onclick="move()">Click Me
</span></div></body></html>
```

図3　DOMを用いてHTMLを変化させる

表1　スクリプトの動作を起動するHTML属性

属性	意味
onclick	クリックしたとき動作
onmouseover	マウスカーソルが上に乗ったら動作
onmouseout	マウスカーソルが上から外れたら動作

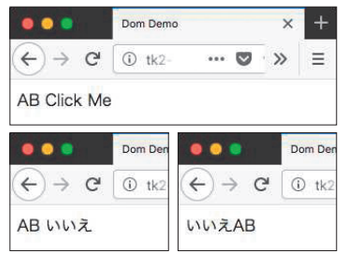

図4　図3を動かして画面が変化する様子

スクリプトでスタイルを操作する

スクリプト側からは、HTMLだけでなくCSSのプロパティを変化させることもできます。各HTML要素オブジェクトxについて、「x.style.プロパティ名」というフィールドに書き込むことで値を設定できます（ただしJavaScriptでは名前の中に「-」を含められないので、「-」は削除して代わりに次の文字を大文字にします。たとえばbackground-colorなら「backgroundColor」とします）。

図5では、bodyの開始タグでstartを呼び出しています。startではh1要素のオブジェクトを変数h1に入れ、位置指定を絶対位置指定に変更し、30ミリ秒間隔でstepを呼ぶように設定します。stepでは変数cnt（初期値は0）を1増やし、255を超えたら0に戻します。そして、h1要素の横位置をcntに設定し、また文字の色を「rgb(100,cnt,200)」に設定します。これで図6のように徐々に色が変化しながら横に動く見出しになります。

```
<!DOCTYPE html>
<html><head>
<meta charset="utf-8">
<title>CSS Dom Demo</title>
<script type="text/javascript">
var h1, cnt = 0;
function start() {
 h1 =
   document.getElementById('h1');
 h1.style.position = 'absolute';
 setInterval(step, 30);
}
function step() {
 if(++cnt > 255) { cnt = 0; }
 h1.style.left = cnt + 'px';
 h1.style.color =
   'rgb(100,' + cnt + ',200)';
}
</script></head>
<body onload="start()">
<h1 id="h1">Look at Me</h1>
</body></html>
```

図5　DOMを用いてCSSを操作する（見出しを変化させる）

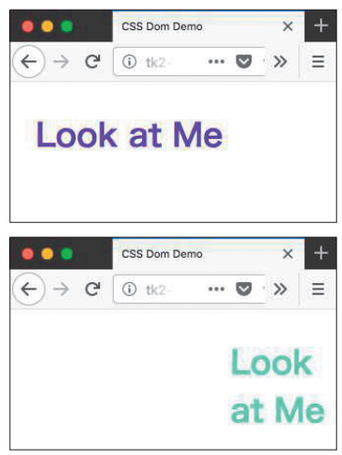

図6　図5を動かして画面が変化する様子