第16章　Dynamic HTMLで雪が降る

レイヤーとJavaScriptとを併用すると面白いことが可能になります。ここでは一例として，雪が降るWebページを作ってみましょう。但し，Accessibilityという観点からすると，多用するのは控えるべきです。やろうと思えばここまで出来る，という程度に考えておいて下さい。

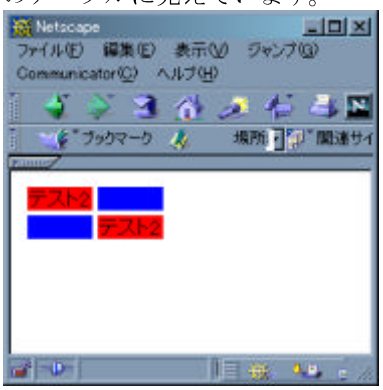
16.1　レイヤーという考え方

例えば，次のようなHTMLファイルを作ってみましょう。これには二つの2行2列のテーブルがあります。一つは“テスト1”，もう一つは“テスト2”という文字がセルに入っています。この二つのテーブルが，それぞれ<layer>タグで囲まれています。

```
<html>
<body>
<layer>
<table>
<tr>
        <td>        </td>
        <td>テスト1</td>
</tr>
<tr>
        <td>テスト1</td>
        <td>        </td>
</tr>
</table>
</layer>
<layer>
<table>
<tr>
        <td>テスト2</td>
        <td>        </td>
</tr>
<tr>
        <td>        </td>
        <td>テスト2</td>
```

```
</tr>
</table>
</layer>
</body>
</html>
```

これをブラウザで表示させると次のようになります。二つのテーブルが重なって，一つの 2 行 2 列のテーブルに見えています。



一つの透明なシートの上に一つの Web ページが描かれ，それを重ねたように用いることが出来る，それがレイヤーというものです。

16.2　雪の降る Web ページ

このレイヤーは透明シートで，Web ページ内を自由に移動させることが出来ます。上の例では特に指定はしませんでしたので同じ位置で重なっているだけでした。

JavaScript で雪 (全角の○) だけを描いたレイヤーをそれらしく動かしているのが図 16.2 です。何をやっているかは詳しく言いませんので，解読してみて下さい。ちなみに，この JavaScript は Netscape Navigator でしか正常に動作しません。

練習問題

1. 図 16.2 を改良し，本物の雪に近い動作をするように変更せよ。
2. (自由課題) 図 16.2 の JavaScript が Internet Explorer で動作しない原因を説明せよ。

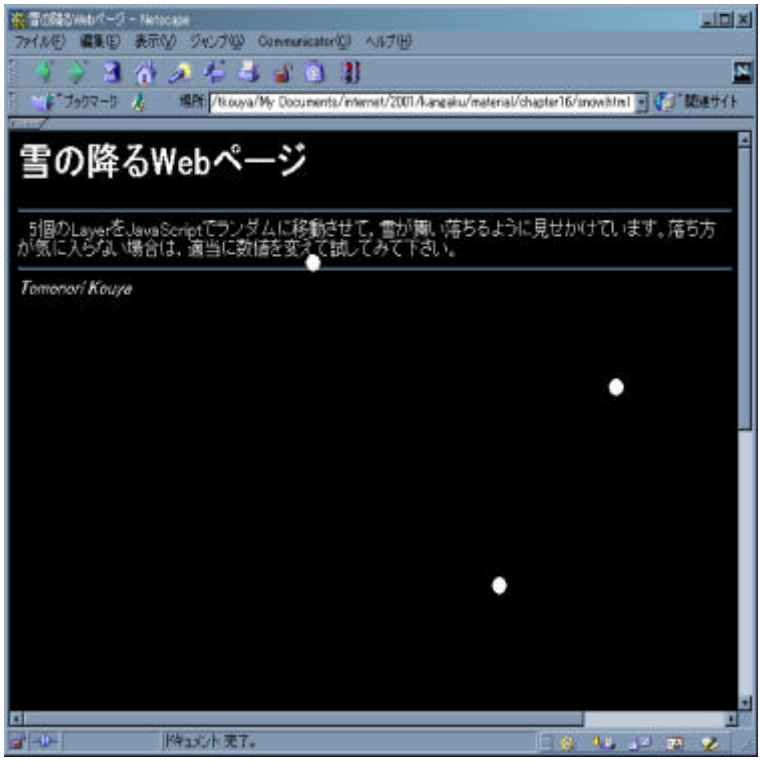


図 16.1: 雪の降る Web ページ

```
<html>
<title>雪の降る Web ページ</title>
<script language="javascript">
var snow_num = 5;
function init_snow()
{
        for(i = 0; i < snow_num; i++)
        {
                document.layers[i].left = Math.floor(Math.random() * 1000);
                document.layers[i].top = -Math.floor(Math.random() * 1000);
        }
}
function start_snow()
{
        for(i = 0; i < snow_num; i++)
        {
                sign = (Math.floor(Math.random() * 100) % 2 == 0) ? +1 : -1;
                dx = sign * Math.floor(Math.random() * 10);
                dy = Math.floor(Math.random() * 10);
                document.layers[i].visibility = "show";
                document.layers[i].moveBy(dx, dy);
                if((document.layers[i].left < -50) || (document.layers[i].left > 1
280) || (document.layers[i].top > 1024))
                        init_snow();
        }
}
window.setInterval("start_snow()", 100);
</script>
<body bgcolor=black text=white onload="init_snow()">
<!-- 雪の Layer -->
<layer>●</layer>
<layer>●</layer>
<layer>●</layer>
<layer>●</layer>
<layer>●</layer>
<!-- 本文 -->
<h1>雪の降る Web ページ</h1>
<hr>
  5 個の Layer を JavaScript でランダムに移動させて，雪が舞い落ちるように見せかけています。落ち
方が気に入らない場合は，適当に数値を変えて試してみて下さい。
<hr>
<address>Tomonori Kouya</address>
</body>
</html>
```

図 16.2: 雪を降らせる JavaScript

コラム★サーチエンジンと robots.txt

変なログが …

前のコラムで，サーバではログを採集しているという話をしました。そしてサーバの管理者であれば，それを直接，あるいは編集したものを間接的にチェックしており，それは管理業務を行う上で必要不可欠である，ということもご理解いただけたと思います。

私 (幸谷) 自身も，現在 2 つの Web サーバを持っていますから，ログのチェックは定期的に行っています。ある日，ログにこんな記録があるのを見つけました。

```
crawler0.archive.org - - [21/Feb/2002:05:38:49 +0900] "GET /robots.txt HTTP/1.0"
 404 275
crawler0.archive.org - - [21/Feb/2002:05:38:49 +0900] "GET /research/mupad/koen1
9991208/sld001.htm HTTP/1.0" 200 1424
```

1 行目，2 行目とも crawler0.archive.org というサイトからの接続記録です。2 行目のアクセス先は，私のトップページからのリンクを辿った末にたどり着いたものでしょうが，1 行目のアクセスは，どこからもリンクが張られていない，しかも存在してない robots.txt というファイルを取得しようとしてエラーになっていることを表わしています。

これは不正なアクセスなのでしょうか？　実はこの robots.txt というファイルは一種の紳士協定で定められているファイル名で，サーチエンジン等が利用する「ロボット」(AIBO や ASIMO とは違い，プログラムの一種です) が参照すべきファイルなのです。何故こんなファイルが必要なのでしょうか？　そしてそのファイルはどのような役割を果たしているのでしょうか？

サーチエンジンの機能— Yahoo! Japan を例に

まず最初に，サーチエンジンの機能を見ていくことにしましょう。例として，商用サーチエンジン最古参である Yahoo! Japan を取り上げます。

サーチエンジン（検索エンジン）とは，Internet 上で利用可能な Web ページ等への膨大なリンクを収拾した Web ページです。指定した単語 (キーワード) を含む Web ページの URI を探し出したい時には，キーワードを検索用のフォームに入力し，自動的に探し出してくれる機能を持っているのが普通です。

Yahoo! Japan も数あるサーチエンジンの一つで，ディレクトリ階層に Web ページを分類すると共に，他のもっと強力なサーチエンジンの助力も得て[1]，キーワードによる検索も可能になってい

[1]Google という全文検索サーチエンジンのデータも参照して検索できるようになっている (2002 年 2 月現在)。

ます (図 16.3)。

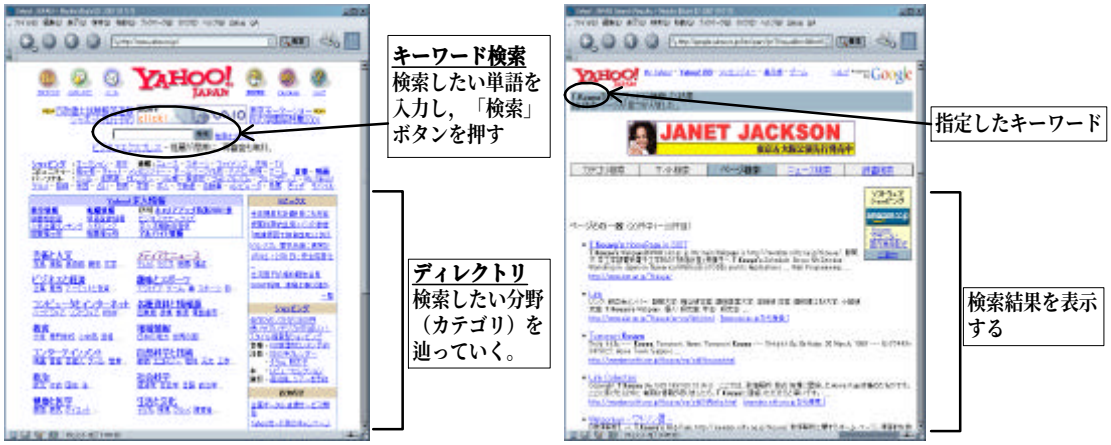


図 16.3: Yahoo! Japan の検索例 (左:検索前，右:検索後)

最初，Yahoo! Japan のデータは全て人手を介して収集され，ディレクトリ階層に分類されていたようです。その基本は現在でも変わっていないようですが，IT 革命と言われて久しい現在，Web サイトの数は日々増え続けており，とても全てを人力で集めきれるものではありません。そこで，人間に代わってプログラムが Web ページのリンクを辿りつつ，データを自動的に収集するというタイプのサーチエンジンが登場しました。有名な所では，Google(`http://www.google.co.jp/`)，AltaVista(`http://www.altavista.com/`) などがあります。研究用に作成されたサイトも多く，今や，Web ページにアクセスする数は，人手を介したものよりも，プログラムが自動的に行うものの方が圧倒的なのではないでしょうか。

サーチエンジンの仕組み

自動的にデータを収集するサーチエンジンは，概ね，図 16.4 のような構成になっています。その中で，リンクを辿って URI を直接集める役割を担うプログラムを「ロボット (robot)」と称しています。AIBO や ASIMO のような人間型の機械とは大分趣が異なりますが，一度起動されると，指定された手続き (アルゴリズム) に従って自律的にデータを集め続けるところは，周囲の状況に応じて反応したり，転倒しないように歩行したりする前者とよく似ています。

ロボットはサーチエンジンだけではなく，様々な用途で使われます。Web サーバに寄生して，次の宿主を探すウィルス[2]も，片っ端からアクセス可能な Web サーバを探すあたり，一種のロボットと言えるでしょう。嫌なロボットですが。

Web が発明されて数年間は，Web ページを所持する人や組織も少なく，他人の Web ページへのリンクを張る時には，メールなどで許可を求めるという風習がありました。今でもリンクの際には許可を取るように宣言している Web ページがありますが，ロボットでは許可の求めようもありません。否応無く，自分の Web ページへのリンクが取られてしまうことになります。

[2]自分の複製を作りつづけて増殖するタイプのプログラムをワーム (worm) と呼ぶ。

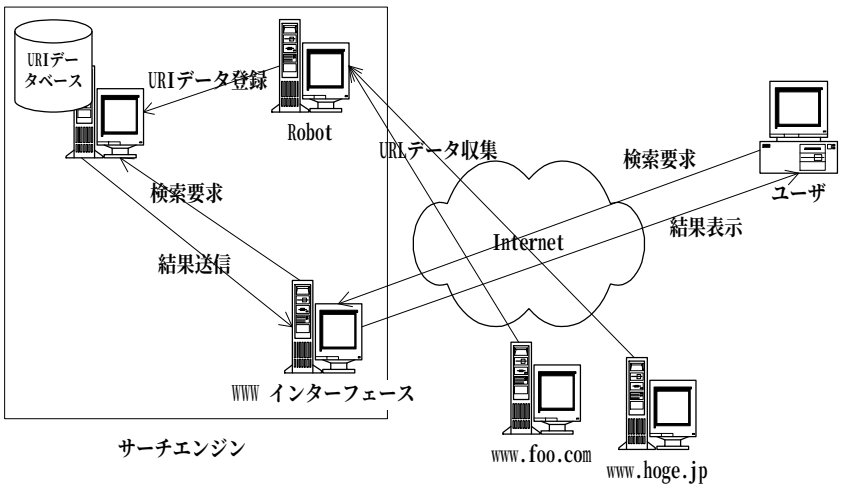


図 16.4: ロボットを利用したサーチエンジン

しかし，ウィルスだけではなく，ロボット一般に言える事ですが，あまり多数のロボットが同一の Web サーバの中身を漁るようになると余計な負荷を Web サーバに与えてしまい，好ましいことではありません。それにリンクはトップページだけ留めてもらい，それ以外のページへは極力控えてほしいという人もいるでしょう。そこで，ある程度はロボットの制御が出来るよう，ユーザが自衛策を取ることの出来る手段が用意されています。その一つが，一番最初に取り上げた `robots.txt` なのです。

ロボットに探られるのが嫌な人へ

以下の内容は，“The Web Robots Pages”(`http://www.robotstxt.org/wc/faq.html`) の内容のダイジェスト版になります。詳細を知りたければこの Web ページを参照して下さい。

手っ取り早い方法として，ロボットがその HTML ファイルからのリンクを辿らせるかどうかを，HTML ファイルの Meta タグに記述する方法があります[3]。

```
<META NAME="ROBOTS" CONTENT=制御方法>
```

というタグを，<HEAD>...</HEAD>の間に書いておけば，ロボットは制御方法に則って動作するようになります。例えば

```
<META NAME="ROBOTS" CONTENT="NOINDEX">
```

とすれば，この HTML ファイルをデータに加えることは排除しますが，そこからのリンクを辿ることは禁止しません。禁止するためには

```
<META NAME="ROBOTS" CONTENT="NOFOLLOW">
```

[3] http://http://www.w3.org/Search/9605-Indexing-Workshop/ReportOutcomes/Spidering.txt を参照

と書きます。全て不許可にしたい場合は

```
<META NAME="ROBOTS" CONTENT="NOINDEX, NOFOLLOW">
```

とするか，単に

```
<META NAME="ROBOTS" CONTENT="NO">
```

とします。

もし，この Meta タグを

```
<META NAME="ROBOTS" CONTENT="ALL">
```

とするか，全く記述しなければ，この HTML ファイル自身もそこからのリンクも全てデータに加えることを許可したことになります。

その Web サーバ全体へのロボットの動作を制御したい場合には，ルートディレクトリに `robots.txt` を置いておき，その中身を

```
User-agent: *
Disallow: /
```

のように記述します。この例では，全てのロボット (`User-agent: *`) に対して，ルートディレクトリ以降，そこからリンクが張られている，この Web サーバに存在する全ての Web ページへのデータ収集を禁止しています。

もし一部のパスのみを禁止対象にしたい時には

```
User-agent: *
Disallow: /tmp
Disallow: /logs
```

とします。ここで示された `/tmp` ディレクトリ及び `/logs` ディレクトリ以外のパスは収集対象となります。

このように，HTML ファイルへ Meta タグを記述するか，あるいはルートディレクトリに `robots.txt` を置いておくことで，ロボットの動きを制御することができます。但し，これはロボット自身がそれらの手段をちゃんと判断できる機能をプログラム内に持っていなければなりません。何も考慮が払われていないロボットにはこれらの手段は全く無意味です。その意味で，これらの手段が守られるかどうかは，ロボットを作成するプログラマの判断に委ねられているのです。

余談：archive.org の正体は?

ところで，最初に取り上げたログに記録されたロボットは何のためにこの Web ページを訪れていたのでしょうか？

その正体は “The Internet Archive”(`http://www.archive.org/`) という，ロボットが訪れた Web ページを全て保存し，変更がある度にそれを収集して回るという，一種の図書館のようなサイトでした。このロボットが足繁く訪問した結果，例えば私の個人ページは図 16.5 にあるように，変更

図 16.5: archive.org が提供する Web ページの履歴一覧

される度にそれが保存されており，1999 年に 1 回，2000 年に 2 回，2001 年に 4 回，2002 年に 1 回，その時々の Web ページが参照できるようになっていました。

本人の手元からも既に消えてしまった内容が，こうして閲覧できるというのもおかしな話です。私は別段構いませんが，このような履歴は見るのもイヤという人もいるでしょう。そもそもこのように他人が著作権を持っているものを保存しておいていいものだろうか？という疑問も湧いてきます。

それは兎も角，ログをチェックしていたおかげで面白い (?)Web ページを新たに自分のブックマークに登録できて，私は満足しています。