Webプログラミング

幸谷智紀
tkouya@cs.sist.ac.jp
http://member.nifty.ne.jp/tkouya/

Version 1.0: March 1, 2002

初めに

> インターネットのおかげで情報の発信が簡単にできるようになり，それまで情報に縁のなかった数千万という世界中の人々が情報を発信する喜び，その中にデタラメを混ぜる喜びを見出して，勝手に情報を発信し始めている。
>
> 土屋 賢二 (「哲学者かく笑えり」より)

このテキストは，2002 年度の講義「Web プログラミング」用に書き下ろしたもので，前年 (2001 年度) の Version 0.4 をベースに全面改稿したものです。

「ホームページ」「HP[1]」「WWW」「Web サイト」「Web ページ」… 等等，様々な言葉が実は Internet で提供されているある特定のサービスを指し示すために用いられますが，ここでは「Netscape や Internet Explorer といったブラウザで表示されるもの」を「Web ページ」という言葉で代表させることにします。

このテキストを使用する予定の講義の目的は次の 2 点です。

- 「Web ページ」とは何か？を理解する。
- Web ページを作成する練習を実際に行う。

Web は Internet の普及に多大な影響を及ぼしてきました。IT 革命と呼ばれる一大変革を巻き起こす原動力といっても過言ではありません。それまではコンピュータ好きの一部の研究者のものだった Internet を一気に親しみやすいものへと変貌させ，次々と一般のユーザを取り込んでいったのです。それも 1990 年代の初期に始まり，数年のうちに全世界へと普及するという驚異的なスピードで。

今や，Web ページは HTML という言語で記述されていることぐらい，大抵の人なら知っています。HTML はごく単純な原理で出来上がっており，Windows[2]に付属のメモ帳でも，デザインの才能さえあればカラフルな Web ページを作成することが可能です。

しかし Web ページ利用者が増加すると共に，様々な機能拡張が頻繁に行われるようになりました。HTML 自身も 1.0 から 4.01，そして XHTML へと変化していますし，HTML で記述された Web ページ内にプログラムを埋め込んだり (Javascript, VBscript, PHP 等)，Java Applet を使用することも増えてきました。Plug-In と呼ばれるブラウザの機能を拡張するソフトウェアも多様なものがあります。サーバ側で動作するプログラムを利用する (SSI, CGI 等) ことは初期の頃から行われ，今ではかなり大規模なデータベースと連動していたりすることも珍しくなくなりました。現代の Web ページにはこのような様々な技術が関連してきますから，全てに精通した人でないと規模の大きな Web サイトの設計は難しいのです。

[1] Home Page を省略した呼称。
[2] 以下，Microsoft 社製造の OS(Windows 95/98/2000/XP) をこのように記述する。

もっともこの講義では，そのようなご大層なことは殆ど触れません。ごく初歩的な事柄だけを舐める程度に済ませています。内容としては

- テキストエディタによる Web ページの作成練習 (第 3 章, 第 4 章)
- Netscape Composer を使用しての Web ページ作成練習 (第 7 章〜第 9 章)

の 2 点を中心に据え，お互いの長所短所を利用しつつ，以下の順で徐々に複雑な Web ページを作成していきます。

- フレームを使った Web ページの作成 (第 11 章)
- スタイルファイルによる Web ページの装飾 (第 12 章)
- フォームを使った Web ページの作成 (第 14 章)
- JavaScript を使った Web ページの作成 (第 15 章)
- Perl を使った CGI プログラミング例についての解説 (第 17 章, 第 18 章)

日ごろから Web サーフィンを楽しんでいる人であれば，この程度の内容をやさしく解説した Web ページがゴマンとあることは良くご存知でしょう。このテキストの内容以上のものも幾つかありますし，ちょっと大きな書店ではこの手の本が山積みです。好奇心と時間に余裕のある人は，それらを使って比較対照しつつ，視野を広げてみて下さい。

また，そのような余裕が無く，このテキストの記述が難しい，あるいは分かりづらいと思われた人は，遠慮なく著者にご意見を下さい。

最後に本書を作成するに当たって使用したソフトウェアを列挙します。これらのソフトウェアが無ければ，特に TEX や Visio，そして Internet と Web が存在しなければ，本書は存在しませんでした。開発に携わった全ての方々に感謝致します。

- Windows 2000 Professional
- 日本語 pLATEX
- dviout
- Netscape Navigator/Composer 4.7x, 6.x
- Internet Explorer
- Microsoft Visio
- MIcrosoft Photo Editor
- 秀丸エディタ

2002 年 2 月 26 日 (火)
遠州・茶畑のど真ん中にて
World Cup を待ちながら
幸谷 智紀

目 次

第1章　“The Internet”と World Wide Web

ARPA ネットが作られたときに用途として想定されていたのは，コンピュータという貴重な資源を共有するということだった。ところが，実際にネットワークが活用され始めると，利用者たちを強くひきつけたのは人につながるためのアプリケーションだった。

古瀬 幸弘・廣瀬克哉 (「インターネットが変える世界」より)

本章では The Internet(インターネット) について，最低限知っておいて欲しいことを列挙しています。後で必要になりますので，一応目を通しておいて下さい。

1.1　Internet 前史 (簡易バージョン)

Internet の歴史については大方，次のような記述がなされています。

> インターネットの発祥地はアメリカです。
>
> (中略)
>
> そもそもの始まりは軍事研究でした。いまから 20 年ほど前の 1970 年代は，ソ連との冷戦下で，ソ連からの核攻撃にどう対処するかが真剣に論じられていました。そして，ソ連の攻撃によって軍事拠点がいくつか破壊されても通信網を維持し，全体として命令系統が維持できるネットワーク技術の開発が必要だということになりました。
>
> この研究のために ARPANET(アーパネット) と呼ばれる実験ネットワークが作られました。これにはアメリカの主要大学や研究機関が参加して，研究開発と実験を進めました。
>
> 中村正三郎編「インターネットを使いこなそう」岩波ジュニア新書, P.40–41

その後，次第に ARPAnet が姿を変え，今の Internet になった，という感じでいいでしょう。細かいことについては参考文献にあれこれと書いてありますので，そちらを参照して下さい。

ただ，押さえておいて欲しいことがあります。まず，Internet が一般に普及するようになったのは World Wide Web，いわゆる Web が開発されてからのことです。1980 年代までは完全に研究者のネットワークとして発展してきました。徐々に普及してきたとはいえ，広く大衆化するには敷居 (料金，AUP，UNIX の操作性) を低くする必要がありました。

ただ，Internet には今も続く良い伝統がありました。ARPANET の時代から，ネットワークを利用したサービスはそれを欲した開発者が勝手に作り上げてきたという点です。開発されたコード (プログラム) は原則として大部分は公開され，誰もが好きに書き換えていましたし，近年までは特

許を取るということも稀でした。Internet を構成する主要なプロトコル (通信手順) として，TCP/IP と呼ばれるものがありますが，これも基礎の部分は誰もが自由に使用できることになっています。

Web もその伝統に習って，自由に使える技術として生まれてきたのです。

1.2 CERN のごちゃごちゃから生まれたクモの巣

WWW(World Wide Web) は，人の移動が激しく情報の共有化が難しかった CERN(ヨーロッパ原子核研究機構) に在籍していた，T.Berners-Lee と R.Cailliau が開発したものです。

1989 年，T.Berners-Lee は “Information Management: A Proposal”という文書を CERN に提案し，この中で初めて”Web”という言葉を使用しました。

そして，1993 年にイリノイ大学の NCSA で M.Andresen が X Window 用の GUI ベースのブラウザ “NCSA Mosaic”を開発しました。続いて Mac 用，Windows 用の Mosaic を開発し配布すると瞬く間に WWW は普及していきました。

WWW サーバもクライアントであるブラウザの広がりと共に増殖し,「Internet=WWW」という認識を大衆に与えるに至りました。これは今でも続いているようですが。

1994 年には Mosaic Communication 社 (後の Netscape Communication 社。後に America Online 社に買収された) が Netscape の Version 1 を開発，翌年には Windows95 の発売と共に Microsoft 社が Internet Explorer を開発し，ここにブラウザ戦争と呼ばれる競争が始まりました。現在では後者のシェアが前者に勝っているようです。

こうして，Internet がビジネスになることを企業は認識し，Internet は大衆化の道を猛進していくことになったのです。

1.3 Internet についての基礎教養

このテキストは TCP/IP 一般を講義するものではありませんが，最低限必要な事項のみ簡単に触れていきます。

1.3.1 Internet の構成と IP アドレス

Internet のサービス (Web ページを閲覧したり，メールをやり取りしたりする等) を，特定の契約者の人々に提供する会社・団体等を ISP(Internet Service Provider) と呼びます。これら ISP が全世界規模で多数集まって任意の相手と通信できるコンピュータネットワーク網が形成されており，これを総称して Internet(The Internet) と呼んでいます。

この Internet に接続されているコンピュータをホスト (host)[1] と呼びます。ホストとネットワークとの接続口にはそれぞれ固有の番号が振られています。これを IP アドレス (IP Address) といい，次のように表現します。

IPv4 202.242.34.10

[1] 本書では，ホストがパソコンをである場合は PC(Personal Computer) と呼ぶことにする。

IPv6 f0a3:0f3b:310c:1307:24a7:0030:ab98:7be4

現在 (2002 年 2 月末日) 多く使用されているのは IPv4 の方です。以下，IP アドレスはこの形式のもののみ扱うことにします。

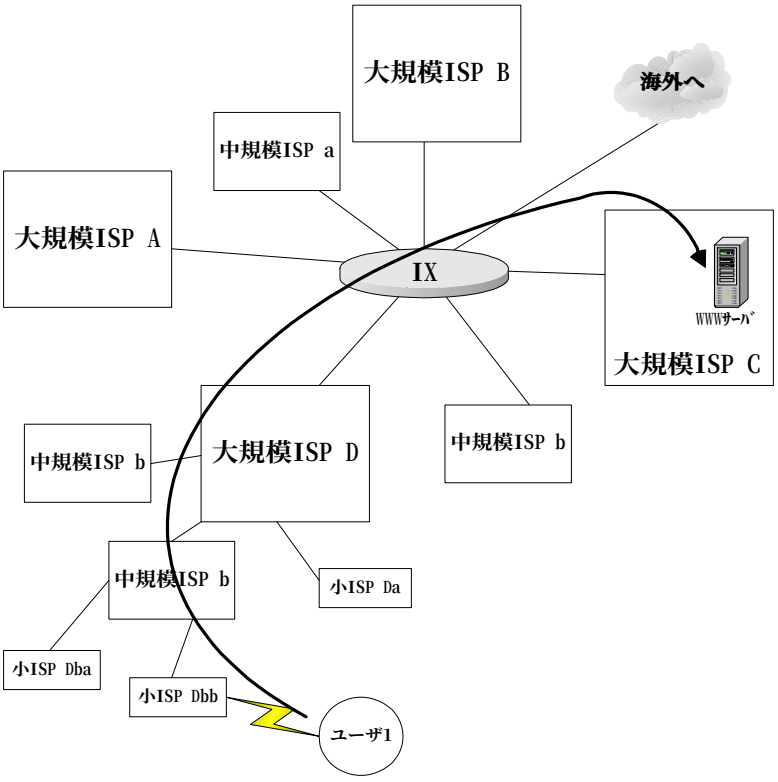


図 1.1: Internet の構成

この IP アドレスは ISP 毎にある決まった範囲内のものが割り当てられています。Internet に接続したいユーザは自分のホストのネットワークとの接続口に，その範囲内の IP アドレスを譲り受け使用することになります。従って，一部の特殊な範囲のものを除き，IP アドレスは重複することはありません。もし重複していればお互い，あるい片方が通信不能になってしまいます。

今，Internet が図 1.1 のような ISP の接続によって成立しているとしましょう。この中の比較的小さな ISP Dbb に所属しているユーザが，大規模 ISP C に所属しているホストと通信する場合を考えてみましょう。

ユーザから発せられた要求は，パケットと呼ばれる信号の塊に分割されて，中規模 ISP b →大規模 ISP D → IX (Internet Exchange) →大規模 ISP D という経路を通ってたどり着くことになります。パケットには発信元と着信先の IP アドレスがそれぞれ書き込まれています。経路上の交換機の役割を果たしているルータ (router) と呼ばれるホストは，このパケット内の IP アドレスを参照しつつ，次々に他のルータへとパケットをたらい回しにしていきます。

一般にパケットがどのような経路を辿るかは，その時のネットワーク接続の状態によって刻々と

変化していきます。更に，それぞれの ISP 間を繋ぐ回線は，当然他のユーザからのパケットをやり取りするのにも使用されていますから，日本全国で多数のユーザが同時に使用するような場合はパケットの伝達に時間が掛かるようになります。例えば平日の昼休み，夕方から真夜中にかけての時間帯がそれにあたります。

1.3.2 ドメイン名とホスト名

ホスト同士がパケットをやり取りする分には，互いの IP アドレスさえ判明していれば済むことです。しかしそのホストを動かしているのは人間です。数字の羅列にしか過ぎない IP アドレスを多数記憶することは困難ですし不便です。

そこである程度規則性のある半角アルファベット (ASCII 文字と呼ばれるもの) でホストに名前を付けることが考えられました。各 ISP 毎にもドメイン名 (Domain Name) という，全世界で一意な文字列が与えられています。各 ISP は自らに所属するホストにホスト名を自分の権限で割り振ることが出来ます。

例えば，先ほどの図 1.1 の大規模 ISP C には “hogehoge.jp”というドメイン名が与えられているとします。ここで先ほどのホストには “www”というホスト名が割り当てられていれば，ドメイン名とホスト名をピリオドを介して繋げた “www.hogehoge.jp”という名前は全 Internet で一意の文字列になり，重複しません。これを FQDN(Fully Qualified Domain Name) と呼びます。

前述しましたが，IP アドレスもドメイン名と同様，各 ISP 毎に範囲指定されて割り当てられていますので，このホスト名と IP アドレスの対応表は ISP 単位で管理することが出来ます。その管理しているホストに FQDN で問い合わせがなされた時に，IP アドレスを教えてくれるようなサービスを DNS(Domain Name Service) と呼びます。これによって，FQDN で Internet 上のホストにパケットを送ることが出来るようになるわけです。勿論，最初から相手側の IP アドレスがわかっているのであれば，それを直接使用することも可能です。

1.3.3 URI

本書では URI(Uniform Resource Identifier) という言葉を使用します。一般には URL(Uniform Resource Locator) という言葉が多用されていますが，URI の I は Identifier を意味していることから分かる通り，tkouya@cs.sist.ac.jp のようなメールアドレスも URI の一種です。従って，URI という言葉は，URL の意味も包含していることになります。

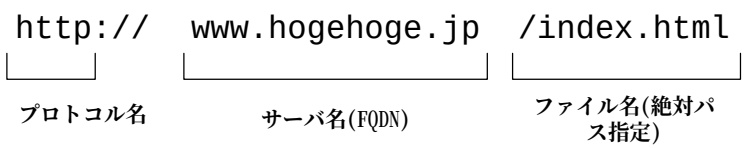


図 1.2: URI の例

ホームページを閲覧するときに用いられる URI は，もっぱら図 1.2 のような形式になります。文

字列のそれぞれには図に示したような意味があります。

最初のプロトコル名とは，ホスト同士がパケットをやり取りする規則を表わす文字列です。ホームページの場合は http(HyperText Transport Protocol)，ファイルをやり取りする際に使われるのは ftp (File Transfer Protocol) です。他にも mailto(メールを送信する), news(ニュースを見る), telnet(ホストを遠隔操作する), file(自分のホストにあるファイルを指定する) 等が使用できます。

サーバ名を指定するときには大抵 FQDN が使用されますが，前述のように IP アドレスを直接指定することも可能です。

最後のファイル名については次章で詳しく解説します。

— メモ —

第2章　Webの仕組み

前章でも触れましたが，いわゆる「ホームページ」を閲覧するための機構全体は，本来WWW(World Wide Web) と呼ぶべき物です。しかし，“WWW”と発音するのは難しいことと，World Wide であることは自明なこととして，“Web”と簡略化した呼称も多く用いられるようになってきました。以下，本書でもこの “Web”という呼び方を使うことにします。

この章では，Internet で提供されているサービスの一つであるこの Web の全体像を紹介していきます。

2.1　Webの仕組み

Internet 上で提供されるサービスの大部分は，24 時間，Internet のどこからでも使用可能です。これはサービスを提供する側が，故障でもしない限り動き続けるコンピュータだからです。このように，サービスを提供する側のコンピュータをサーバ (Server) と呼びます。反対に，サービスを要求し受ける側をクライアント (Client) と呼びます。クライアントもコンピュータですが，これはユーザが使いたいときにだけ動作させるようになっているのが普通です。使わないときには電気代節約のために電源さえ切られてしまいます。

それならば，サーバはクライアントの要求に応じて起動すればよいようですが，Internet は ISP と契約した不特定多数のユーザが世界中で使用していることを忘れてはいけません。少なくとも全世界のユーザに対して使用可能にしようとすれば，サーバは 24 時間動作し続けている方が合理的なのです。

これは Web でも全く同様です。Web の場合，サービスを提供するサーバ側を Web サーバ (Web Server)，クライアント側で Web サービスを要求するプログラムのことを (Web) ブラウザ (Browser) と呼びます。図 2.1 は，ブラウザが Web サーバ (www.hogehoge.jp) に対し,「ルートディレクトリにある，あらかじめ指定されたファイル (この場合は “index.html”という名のファイル) をよこせ」という要求をした時の動作を示しています。

ブラウザは Web サーバから，閲覧したい Web ページの元になるファイルを Internet を介して取得し，それをユーザが使用しているクライアント側のホストの画面に表示する役割を果たしています。これに対し，Web サーバは要求に応じて自分のディスクに保存されているファイルを送り出しているだけです[1]。但し，Web サーバはこの作業を複数のユーザからの要求に同時に答える必要があり，負担が少ない程良い訳です。

[1] それだけではないケースもある (第 17，第 18 を参照)。

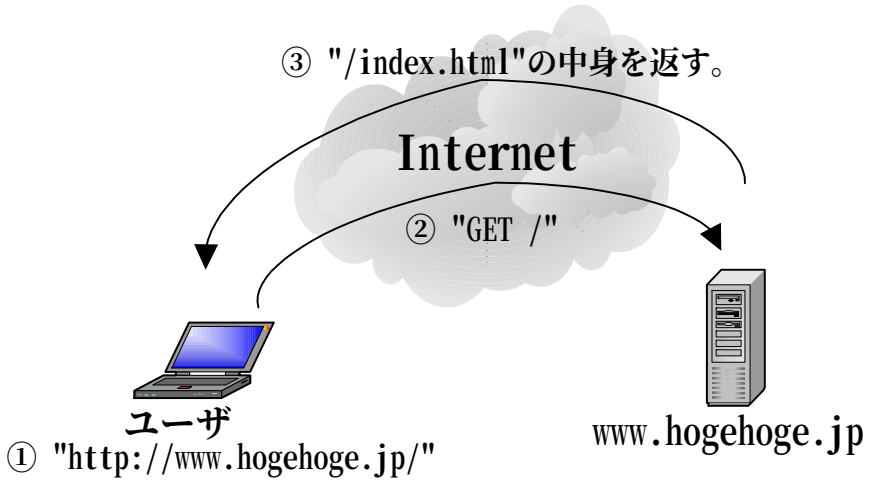


図 2.1: Web の仕組み (概念図)

2.2　絶対パス指定と相対パス指定

Web とは直接関係があるように感じられないかも知れませんが，Web サーバへ Web ページを upload する際に必要となる知識として (第 6 章を参照)，ここではディレクトリ構造について解説します。

ブラウザで Web ページを表示するためのデータが記入されたファイルは Web サーバのディスク (Disk, 大容量記憶装置) に保管されています。ファイルにはそれぞれ名前があり，これをファイル名と言います。データの量が多い時は，複数のファイルに分割しておいた方が扱いやすくなるものです。従って，データの量が増えるほど，ファイルの数も増えて行きます。

何百，何千のファイルが一箇所に固まっていると見通しが悪くなりますし，必要なファイルがどこにあるのか探すことも面倒になります。そこで，ファイルを入れる入れ物を作ることにしました。これをディレクトリ (Directory) と呼びます。Windows で言う所のフォルダ (Folder) と同じ物と思って下さい[2]。ディレクトリ (フォルダ) もファイルと同様名前が付きます。これをディレクトリ名 (フォルダ名) と呼びます。

ディレクトリにはファイルとディレクトリも入れることが出来ます。従って，ディレクトリの中のディレクトリの中のディレクトリの中の … という具合に，ディレクトリの階層構造を成したものが出来てきます。図 2.2 にこの例を示します。

一番左にある “/”をルートディレクトリ (Root Directory) と呼び，これが全てのディレクトリの元になります。ファイルは全てこのルートディレクトリ以下のディレクトリ置かれることになります。この場合は，ルートディレクトリの下に “home”というディレクトリがあり，“home”の下に “public_html”と “tmp”という 2 つのディレクトリがあります。そして，“public_html”の下には “cgi-bin”と “bitmap”というディレクトリが存在しています。

今，“public_html”ディレクトリの下に (中に)“index.html”というファイルがあるとしましょう。こ

[2]本書では区別のため，Windows の場合はフォルダ，Web サーバの場合はディレクトリという呼び方を使います。

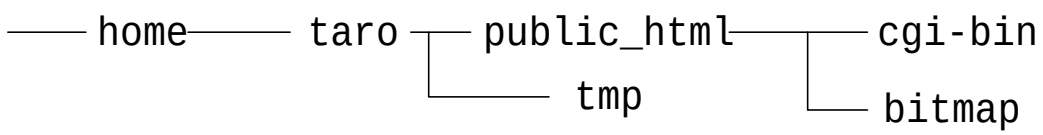


図 2.2: ディレクトリツリーの例

のファイルの位置を示す方法として，絶対パス指定と相対パス指定の 2 つがあります。前者はルートディレクトリを基準とした位置指定方法，後者は現在のディレクトリ (カレントディレクトリ) からの相対的な位置を示すものです。

従って，この “index.html”を絶対パス指定で表示すると

```
/home/taro/public_html/index.html
```

となります。ディレクトリの区切りをスラッシュ“/”で表わしています[3]。

これに対し，相対パス指定の場合，例えばカレントディレクトリが同じ “public_html”であれば

```
./index.html
```

又は

```
index.html
```

となります。もしカレントディレクトリが “taro”であれば

```
./public_html/index.html
```

又は

```
public_html/index.html
```

となります。そしてカレントディレクトリが “bitmap”であれば

```
../index.html
```

となります。

Web サーバにファイルを要求する際，ファイル位置があらかじめ分かっている時は直接絶対パス指定を使うことが出来ます。しかし，Web サーバに置いておくファイルの中で，同じルートディレクトリ以下の別のファイルの位置を指定する時には，相対パス指定の方が後々都合が良いことがあります。

どちらも場合に応じて使いこなすことが出来るようになりましょう。

[3]Windows では ¥マークで区切るのが普通。欧米ではスラッシュが逆に傾いた「バックスラッシュ」になる。

2.3　ブラウザと HyperText Markup Language(HTML)

ブラウザで Web サーバから取得してきたファイルに書き込まれているデータには，ブラウザ画面に表示させたい形式 (フォーマット) が記述されています。この時，表示されたものを Web ページ (Web Page)，その形式を記述する言語を HTML(HyperText Markup Language) と呼んでいます。一昔前でしたら，「ブラウザはパソコン上で動作するプログラム」と言うことも可能でしたが，今では携帯電話でもブラウザが動いていますね。しかし，今の所，携帯電話上のブラウザは PC のそれに比べてハード面での制約が多いので，本書では PC 上のブラウザのみを対象とします。

前述した通り，Web の登場後すぐに Microsoft 社と Netscape 社との間でブラウザ戦争と呼ばれるシェア争いが始まりました。前者のブラウザは Internet Explorer，後者のブラウザは Netscape Navigator です (図 2.3)。

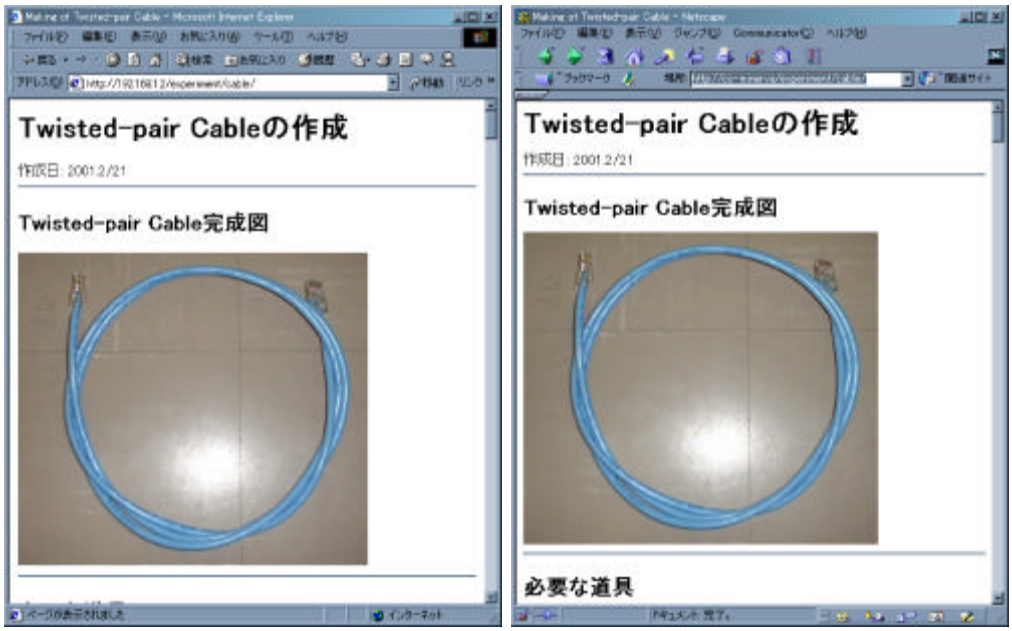


図 2.3: Internet Explorer(左) と Netscape Navigator(右)

現在では前者が圧倒的多数を占めているようですが，後者の愛好者も根強く[4]，今の所は双方で支障なく閲覧できる Web ページを作る必要があるでしょう。本書でもそのような方針で Web ページを作っていく予定です[5]。

2.4　HTML の例

Web ページの例を図 2.4 に，この Web ページを記述した HTML を図 2.5 に示します。

同じ Web ページを表示させてみると，Internet Explorer でも Netscape Navigator でも殆ど同じよ

[4]筆者もその一人。
[5]一部例外がある (第 16 章参照)。

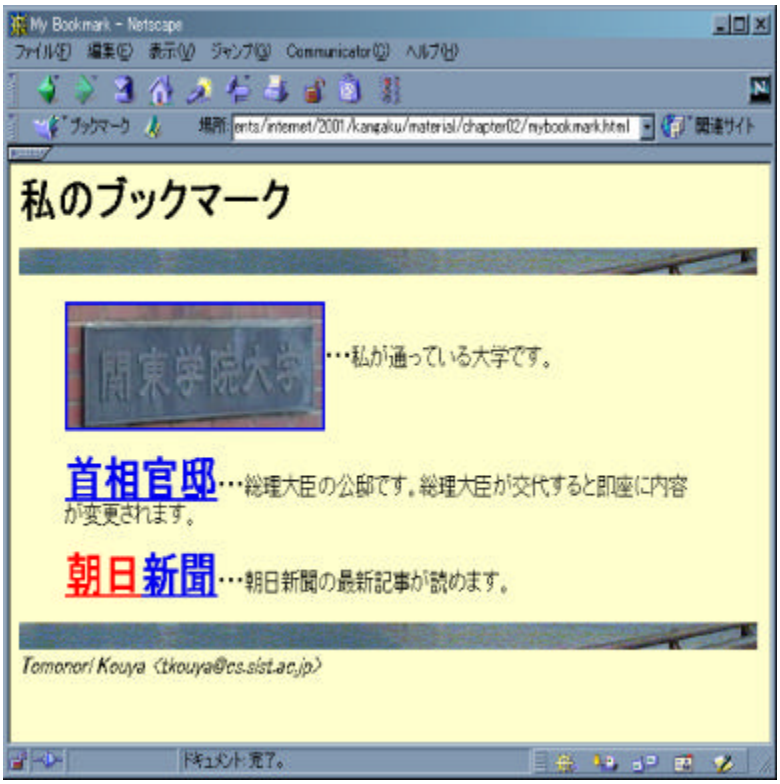


図 2.4: Web ページの例: Netscape Navigator による表示

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=Shift_JIS">
   <meta name="Author" content="Tomonori Kouya">
   <meta name="GENERATOR" content="Mozilla/4.75 [ja] (Windows NT 5.0; U) [
Netscape]">
   <title>My Bookmark</title>
</head>
<body text="#000000" bgcolor="#FFFFCC" link="#0000EE" vlink="#551A8B" alin
k="#FF0000">

<h1>
私のブックマーク</h1>
<img SRC="line.gif" height=18 width=640>
<blockquote><a href="http://www.kanto-gakuin.ac.jp/"><img SRC="kangaku.jpg
" ALT="関東学院大学" height=82 width=221 align=CENTER></a> ··· 私が通って
いる大学です。
<p><b><font size=+3><a href="http://www.kantei.go.jp/">首相官邸</a></font>
</b> ··· 総理大臣の公邸です。総理大臣が交代すると即座に内容が変更されます。
<p><b><font size=+3><a href="http://www.asahi.com/"><font color="#FF0000">
朝日</font>新聞</a></font></b> ··· 朝日新聞の最新記事が読めます。</blockq
uote>
<img SRC="line.gif" height=18 width=640>
<address>
Tomonori Kouya &lt;tkouya@cs.sist.ac.jp></address>

<br> 
</body>
</html>
```

図 2.5: HTML による Web ページの記述

```
<HTML>
<HEAD>
 ···
<TITLE>Web ページのタイトル</TITLE>
 ···
</HEAD>
<BODY>
 ···
Web ページ本文
 ···
</BODY>
</HTML>
```

図 2.6: HTML ファイルの構造

うな結果を得るものの微妙な違いがあることも分かります。ブラウザは同じものでも，PC の画面の解像度が異なっていたり，表示可能な色数が少なかったり，ブラウザの Version が古かったりするとこの違いはさらに顕著になります。環境によって Web ページの表示のされ方が違うことを認識しておいて下さい。

図 2.5 が HTML の一例です。はじめて見る人には，何やら難しげな文字が並んでいますが，よく見ると規則性が分かってきます。

まず，< 単語 > ··· < / 単語 > という部分が目に付きます。このような < 単語 > をタグ (tag) と呼びます。タグは文書の構造を表わすもので，例えば 12 行目から 13 行目の「<h1> 私のブックマーク < /h1>」は “私のブックマーク”という文字列が大見出しであることを指定しています。タグの種類は他にも多数あり，後の章でも触れますが,「タグを覚えること＝良い Web ページを作ることが出来る」ではありませんので，本書では必要に応じて，使用しなければならないものだけを pickup していくことにします。

図 2.5 を単純化して HTML の構造の大枠を抜き出すと，図 2.6 のようになります。詳細については次章以降で触れていくことにしましょう。

練習問題

1. 図 2.2 の構成で，“index.html”が “public_html”にあるとする。カレントディレクトリが以下の場合，この “index.html”が存在するディレクトリを示す相対パス指定はどのようになるかを考えよ。
   - /(ルートディレクトリ)
   - /home
   - /home/taro/public_html/cgi-bin

2. 『「タグを覚えること＝良い Web ページを作ることが出来る」ではありません』という主張の根拠は何か？ この主張に対する反論があれば述べよ。

コラム★Webページを作る目的と作成手順

ぼくが多くの人を引きつける魅力あるホームページづくりができないとは思わなかった。それは,「できるまでやめなければ，できる」と考えられた。いままでの仕事だって，いつもそうだったのだ。できないというところでやめなければ，できるのだ。

糸井 重里 (「ほぼ日刊イトイ新聞の本」より)

Webページの目的

後の章で説明している通り，Webページはワープロ文書と同様に，Webページ作成のためのソフトウェア (第5章を参照) で簡単に作成することが出来るようになりました。ワープロソフトの中には，ワープロ文書をWebページに変換する機能を持つものも現れています。そのため，ワープロ文書を作るのと同じ感覚でWebページを作ってしまう人が多いようです。

それがいけないというのではありませんが，紙に印刷されることを前提としているワープロ文書と，ディスプレイで表示され，他のWebページへ自由に移動することの出来る機能を備えたWebページとは，区別して考えるべきものです。特にWebページは，広く考えれば全世界の人が理解できることが望ましいのです。少なくとも自分のWebページの読者対象としている人々が理解できる言葉 (日本語，英語，フランス語 …) で，わかりやすく記述されていなければなりません。

そうなると，このテキストで縷縷説明している「Webページ作成のテクニック」は大して重要なことではありません。それよりも「分かりやすい文章を書く能力」や「読者に理解しやすいWebページの構成能力」，そして「余すことなく情報を詰め込み，なおかつ魅力的なWebページをデザインする能力」を持つことの方がずっと大切なことであり，そして難しいことなのです。

単なるテクニックよりも，構成能力。構成する以上に，魅力的なコンテンツを発見し，人に説明する能力を磨くこと。本書で述べている初歩のテクニックはそのための最初の踏み台に過ぎないことを忘れないようにして下さい。

Webページ作成の手順

万人に読んでもらうことをWebページの一番の目的とするならば，Webページは書店に並ぶ本と同じ工程で作成されなければなりません。違いがあるとすれば，本は印刷・製本され，流通ルートに乗って初めて出版されたことになりますが，Webページは作成者がWebサーバに転送しURIを公表すれば誰でも読むことができるようになるという点だけです。自分以外の第三者に読んで貰わなければ意味がありませんから，公開するまでには何重にもチェック (推敲) する必要がありま

す。本だって，最低，著者と編集者のチェックを経る訳ですし，慎重な著者は周囲の人にも読んで貰い，その反応を確認しているのです。

Web ページは Web サーバに転送されて，他の人が読むことができるようになるものですから，自分の手元の PC でチェックを行うのは勿論，Web サーバに転送したものに対しても同様のチェックを行うことが最低でも必要です。この過程を図 2.7 に示します。

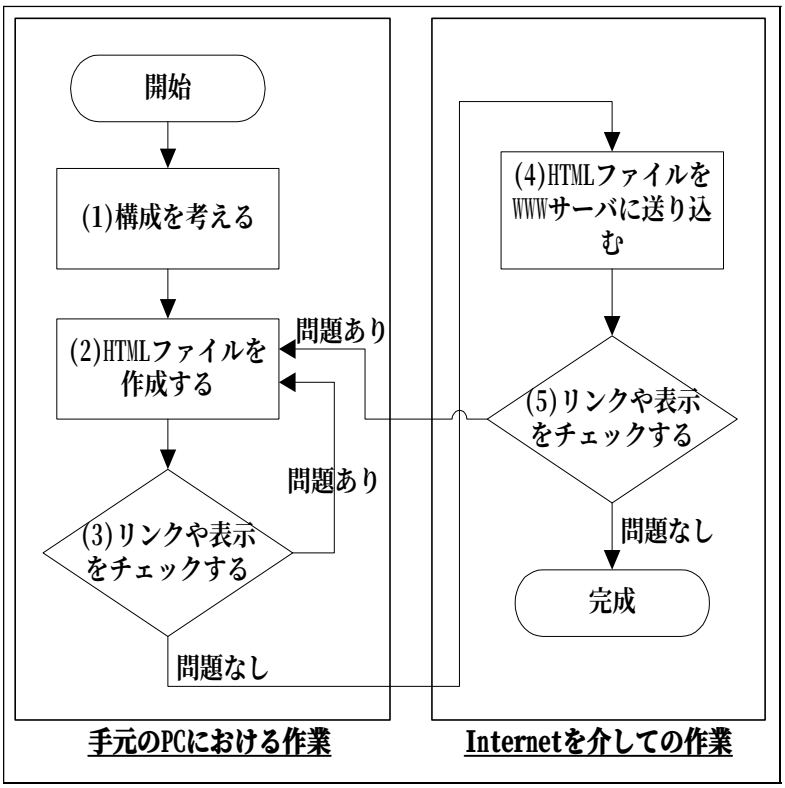


図 2.7: Web ページ作成手順のフローチャート

このチェックの際には，本と同様，自分以外の人に Web ページを見て貰うようにしましょう。自己満足が第一の趣味の Web ページなら兎も角，自分の会社や仕事に関連する情報を提供する Web ページが独り善がりでは人に読んで貰えないばかりか，業績に響く恐れが出てきます。

第3章　テキストエディタでWebページを作る (1/2)

> 最後に，とっておきの，ホームページ作成術を教えておこう。それは，インターネット上で素晴らしいサイトを見つけたら，そっくりそのまま，もしくは，気に入った部分のその構成をコピーして，取り入れることだ。… (略) … コンピュータの先生は眉をひそめるかもしれないが，これがロックンロールのやり方だ。
>
> ロックンロールの世界は，昔から，そして今も，すごいことをチープにやっているぜ。
>
> 鮎川 誠 (「DOS/V ブルース」より)

本章と次章では，HTML に慣れるため，テキストエディタと呼ばれる極めてシンプルなソフトウェアだけを使って Web ページを作成していきます。キーボードから打ち込む量が多くなりますが，練習だと思って我慢して下さい。

本書では，Web ページを作るための PC は Windows 2000 を搭載しているものを想定していますが，テキストエディタの基本機能はどの OS であれエディタであれ大差ないので，ここで述べていることは他の OS やエディタにもそのまま応用できます。

以下，自己紹介の例では,「氏名」「メールアドレス」「学籍番号」等は自分のものに置き換えて作成して下さい。

3.1　テキストエディタとは？

テキストエディタとは,「テキストファイル」(Plain Text File) を編集するためのソフトウェアの総称です。「テキストファイル」と対立する単語として「バイナリファイル」という呼び方があります。

現在の PC をはじめとしたコンピュータは，2 を基数とする整数 (自然数) のみをデータとして扱います。通常，2 進数では，同じ数を表現するにも我々が慣れ親しんでいる 10 進数より余計に桁数 (ビット (Bit) 数) が必要になりますので，2 進 4 桁 (0〜15) までをひとまとまりにして扱います。こうすると 2 進数が 16 進数になります。16 進数は表 3.1 のように，0〜FF とアルファベットも使って表現されます。前章で IPv6 での IP アドレスの形式で出てきた A〜F は 10〜15 の意味です。

そして，データとしての区切りは 2 進 8 桁ごとにとり，一般的にはこれを 1 バイト (Byte) と呼んでいます。これがコンピュータが扱うデータの最小単位です。

この 1 バイトは 0〜FF(10 進数では 0〜255) までの範囲の数を扱うことが出来ます。このうち特定の範囲の数は半角アルファベットやその他の記号と表わすものとして使用されます。つまり，こ

表 3.1: 2 進数と 10 進数の対応

10 進数	0	1	2	3	4	5	6	7	8
16 進数	0	1	2	3	4	5	6	7	8
2 進数	0	1	10	11	100	101	110	111	1000

10 進数	9	10	11	12	13	14	15
16 進数	9	A	B	C	D	E	F
2 進数	1001	1010	1011	1100	1101	1110	1111

の範囲のデータだけで構成されたファイルは，それぞれの数に対応付けられた文字だけで成立しているとも解釈できることになります。このような文字との対応が付けられた範囲の数のみをデータとして持つファイルをテキストファイルと呼び，この文字と数との対応付けをと呼んでいます。文字コード外の範囲の数もデータとして含んでいる可能性があるファイルをバイナリファイルと言うわけです。つまり，テキストファイル以外は全てバイナリファイルなのです。

但し，日本語については文字数が多いため，1 byte ではなく 2byte で平仮名・片仮名・漢字との対応を取っています。これもある一定の範囲内の数と日本語文字との対応が付けられており，通常このような範囲の 2 バイトデータも含むファイルもテキストファイルと呼びます。

ただ，日本語文字と数字との対応付けは，過去の歴史も引きずっていて，機種ごと，OS 毎に異なったものが並存してきました。今も用いられている文字コードは次の 4 つがあります。

JIS (ISO 2022-JP) ･･･ 日本工業規格 (JIS) で定められたもの。

Shift JIS ･･･ JIS の対応付けをずらし (shift) たもの。

EUC-JP ･･･ JIS と似ているところが多い。UNIX 環境で盛んに使われてきた。

UTF-8,16 (Unicode) ･･･ 世界各国の文字コードを 4byte(2 進 32 桁) に全て押し込んだもの。

これら 4 つの文字コードは，Internet での情報のやり取りでも使用されています。従って，互いに異なる文字コードを使用したデータが交換されてしまうと，テキストファイルであるはずの内容が，全然別の訳のわからない内容に見えてしまいます。このような状態を文字化けと呼んでいます。本書では特に断らない限り，文字コードとしては Shift JIS を使用します。

テキストエディタとは，このような日本語も含むテキストファイルのみを編集するためのソフトウェアなのです。

図 3.1 にあるのは代表的な 2 つのテキストエディタの画面です。メモ帳は Windows が導入された PC であれば必ずインストールされている，最もシンプルなテキストエディタです。それに対して，秀丸エディタ (`http://www.maruo.co.jp/`) は有料のソフトウェアで，Internet を介して自由に取得することが出来ますが，使用し続けるには料金を払わなければなりません。シェアウェアと呼ばれているソフトウェアの一種です。メモ帳と比べ，この秀丸エディタは様々な機能を備えており，「マクロ」と呼ばれる一種のプログラム言語を使うことで，独自の機能を追加することが可能です。

この章と次章ではこのテキストエディタを利用して Web ページを作っていきます。秀丸エディタが使える環境であればそれを使って下さい。なければメモ帳でも構いません。テキストエディタの使い方はワープロソフトの基本編集機能と大差ありませんので，本書では特に使い方の解説はしません。

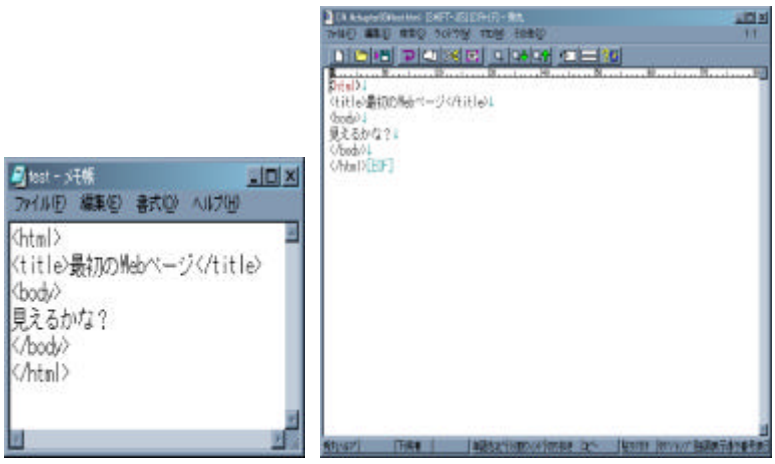


図 3.1: メモ帳 (左) と秀丸エディタ (右)

3.2 拡張子は.html か.htm

まず，ごく簡単な Web ページを HTML で作ってみましょう。出来上がりと HTML の内容を図 3.2 に示します。

次のような手順で作っていきます。以下，この手順で Web ページを作っていきます。

1. テキストエディタを開き，図 3.2(右) の HTML を記述します。
2. 適当なフォルダにファイルを保存します。ここではファイル名を “first.html”としましょう。
3. Netscape Navigator を起動し，[ファイル] → [ページを開く] → [ファイルを選択] とメニューとボタンを選んでいき，先ほど保存したファイルを指定します。Netscape Navigator を起動した後，表示画面にドラッグ & ドロップしても構いません[1]。
4. Netscape Navigator に図 3.2 左の画面が現れれば成功です。そうでないときはもう一度先ほど保存したファイルの内容を確認して下さい。

保存した”first.html”ですが，ファイル名は何でも良いのですが，拡張子は必ず “.html”か “.htm” にして下さい。そうでないとこれを Web ページとはみなして貰えません。このように HTML で記述されたファイルを HTML ファイルと呼びます。

この節は肩慣らしです。出来上がったら次の節に進んで下さい。

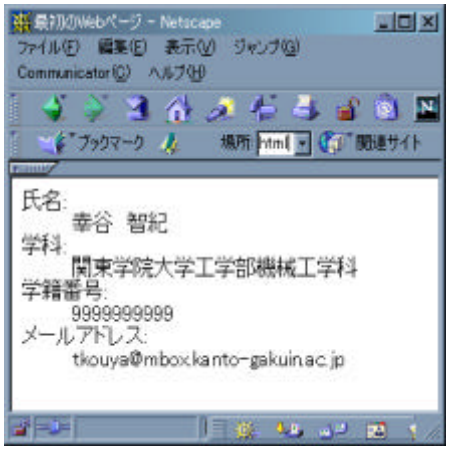
3.3 まずは自己紹介

今度は，次のような Web ページを作ってみましょう。右側の HTML を見ながら，先ほどと同じ手順でファイルを作成し，“intro.html”という名前でフォルダに保存し，Netscape Navigator で出来上がりをチェックして下さい。

[1] Internet Explorer でも同様の操作が可能。


```
<html>
<title>最初の Web ページ</title>
<body>
見えるかな？
</body>
</html>
```

図 3.2: 最初の Web ページ: (左) 表示画面　(右) その HTML




```
<html>
<title>最初の Web ページ</title>
<body>
<dl>
<dt>氏名:<dd>幸谷　智紀
<dt>学科:<dd>関東学院大学工学部機械工学科
<dt>学籍番号:<dd>9999999999
<dt>メールアドレス:<dd>tkouya@mbox.kanto-gak
uin.ac.jp
</dl>
</body>
</html>
```

図 3.3: 自己紹介の Web ページ

見てお分かりのように，これは「氏名」「学科」「学籍番号」「メールアドレス」を項目とした箇条書きの形式になっています。項目部分が <DT> 以下に，それに続く内容が <DD> 以下に続いています。

3.4 フォントサイズを変えよう

先ほどの “intro.html”をちょっと修正して，図 3.4 のように微修正してみます。名前の文字を大きくしています。


```
<html>
<title>最初の Web ページ</title>
<body>
<dl>
<dt>氏名:<dd><FONT SIZE="6">幸谷　智紀</FONT>
<dt>学科:<dd>関東学院大学工学部機械工学科
<dt>学籍番号:<dd>9999999999
<dt>メールアドレス:<dd>tkouya@mbox.kanto-gakuin.
ac.jp
</dl>
</body>
</html>
```

図 3.4: フォントサイズを変える

うまくいきましたか？　では次へどうぞ。

3.5 文字に色をつけよう

先ほど変更した “intro.html”に更に変更を加えます。どこを変えたか分かりますか？　名前の文字の色を青に変えていますね。HTML のどこを変更したらこのように出来るのか，分かったらそのように修正して下さい。

うまくいったら次へどうぞ。

3.6 線を引こう

先ほどの “intro.html”に，3 箇所の変更を加えたものが，図 3.6 です。変更点は

- Web ページの先頭に「自己紹介」という大見出しの追加
- そのすぐ下に水平線を引く


```
<html>
<title>最初の Web ページ</title>
<body>
<dl>
<dt>氏名:<dd><FONT size="6" color="blue">幸谷
智紀</FONT>
<dt>学科:<dd>関東学院大学工学部機械工学科
<dt>学籍番号:<dd>9999999999
<dt>メールアドレス:<dd>tkouya@mbox.kanto-gakuin.
ac.jp
</dl>
</body>
</html>
```

図 3.5: 文字に色をつける

- Web ページの一番下にも水平線を引く

というものです。

うまく修正できましたか？では最後の仕上げを行いましょう。

3.7　Web ページ作成者の証を残そう

自己紹介の Web ページ，“intro.html”を完成させましょう。通常，Web ページにはそれを作成した人の証を残しておきます。これを Web ページの一番下に引いた水平線の更に下に追加します。それが図 3.7 です。

完成したら，フォルダにきちんと保存しておきましょう。

練習問題

1. 自己紹介の「メールアドレス」の項目の下に「自己アピール」の項目を新たに追加し，自己アピール文を書け。

2. この章で作った自己紹介の Web ページは図 3.8 のような書式であった。このような書式にふさわしい内容を考え，新たな Web ページを作れ。


```
<html>
<title>最初の Web ページ</title>
<body>
<H1>自己紹介</H1>
<HR>
<dl>
<dt>氏名:<dd><FONT size="6" color="blue">幸谷
智紀</FONT>
<dt>学科:<dd>関東学院大学工学部機械工学科
<dt>学籍番号:<dd>9999999999
<dt>メールアドレス:<dd>tkouya@mbox.kanto-gakuin.
ac.jp
</dl>
<HR>
</body>
</html>
```

図 3.6: 水平線を引く


```
<html>
<title>最初の Web ページ</title>
<body>
<H1>自己紹介</H1>
<HR>
<dl>
<dt>氏名:<dd><FONT size="6" color="blue">幸谷
智紀</FONT>
<dt>学科:<dd>関東学院大学工学部機械工学科
<dt>学籍番号:<dd>9999999999
<dt>メールアドレス:<dd>tkouya@mbox.kanto-gakuin.
ac.jp
</dl>
<HR>
<ADDRESS>Tomonori Kouya : &lt;tkouya@mbox.kanto
-gakuin.ac.jp&gt;</ADDRESS>
</body>
</html>
```

図 3.7: Web ページ作成者の証を付ける

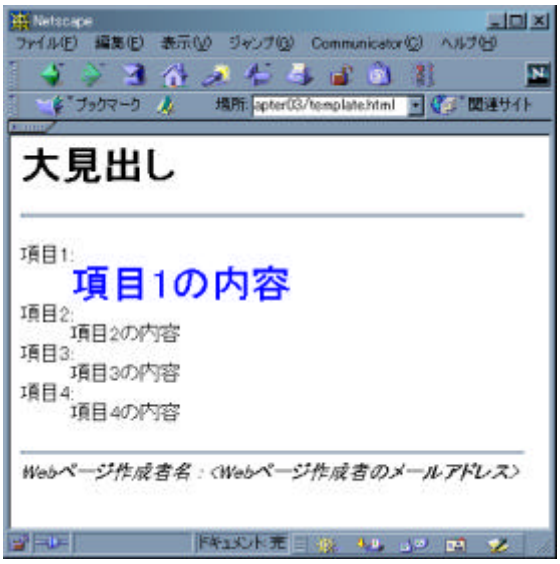


図 3.8: 自己紹介で使った書式

第4章　テキストエディタでWebページを作る (2/2)

前章に引き続き，テキストエディタでHTMLを書き，Webページを作っていきます。Webページが通常の文書と異なる点は，クリック一つで他のWebページへ異動することが出来るリンク (Hyper link) が使用できることにあります。本性では実際にリンクを張ったWebページを作成した後，前章で作成した自己紹介をテーブル機能を使って装飾してみます。

4.1　リンクがHypertextの命

リンク (Hyper link) はHTMLの要の部分です。では，

1. `http://www.kanto-gakuin.ac.jp/` … 関東学院大学
2. `http://www.kantei.go.jp/` … 首相官邸
3. `http://www.asahi.com/` … 朝日新聞

へのリンクを張ったHTMLを書いてみましょう (図4.1)。ここでは数字付き箇条書きを使っています。

リンクは現Webページから他方への一方向への移動を可能にするものです[1]。また12行目を図4.2のように修正すると，このリンクをクリックすることで宛先が自動的に記入されてメールソフト[2]が起動されます。

4.2　テーブルを作ろう

前章で作った自己紹介のページを，HTMLのテーブル (Table) 機能を使って装飾してみます (図4.4)。

テーブルタグは

[1] 戻る時にはブラウザの「戻る」ボタンを使用する。
[2] メーラ (mailer)，もしくはMUA(Mail User Agent) とも呼ばれる。


```
<HTML>
<HEAD><TITLE>Link</TITLE></HEAD>
<BODY>
<H1>リンクの練習</H1>
<HR>
<OL>
<LI><A HREF="http://www.kanto-gakuikn.ac.jp/">
関東学院大学</A>
<LI><A HREF="http://www.kantei.go.jp/">首 相 官
邸</A>
<LI><A HREF="http://www.asahi.com/">朝　日　新
聞</A>
</OL>
<HR>
<ADDRESS>Tomonori Kouya : &lt;tkouya@cs.sist.a
c.jp&gt;</ADDRESS>
</BODY>
</HTML>
```

図 4.1: リンクの練習

```
<ADDRESS>Tomonori Kouya : &lt;<A HREF="tkouya@cs.sist.ac.jp">tkouya@cs.sis
t.ac.jp</A>&gt;</ADDRESS>
```

図 4.2: リンクの練習

```
<table>
    <tr>
        ...
        <td>セルの内容</td>
        ...
    </tr>
    <tr>
        ...
    </tr>
    ...
</table>
```

という構造になります。<tr> タグで囲まれた部分は行 (row, 横方向) の内容を，<td> タグで囲まれた部分は行を列 (column, 縦方向) に細分化したセルの内容を記述します。行，列，セルの関係を図示したのが図 4.3 です。

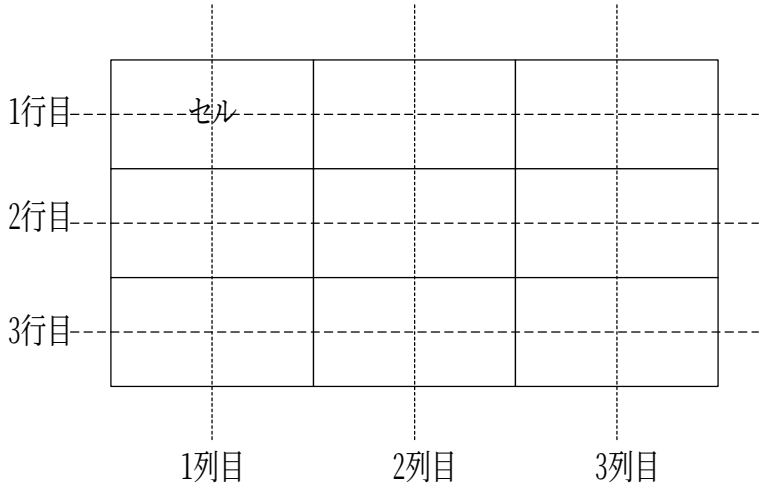


図 4.3: テーブルの行，列，セル

4.3 テーブルの中にテーブルを作ろう

更に先ほどの例の最後に「その他」の列を付加し，その内容の部分にテーブルを入れ子にしてみましょう。

練習問題

1. 図 4.5 の趣味・資格の次に「自己アピール」の項目を作り，自己アピールの文章を挿入せよ。


```
<html>
<title>最初の Web ページ</title>
<body>
<h1>自己紹介</h1>
<hr>
<table border>
<tr>
<th>氏名</th>
<td></font><dd><font size=5 color="blue"><s
trong>幸谷　智紀</strong></font></td>
<tr>
<tr>
<th>学科</th>
<td><a href="http://www.kanto-gakuin.ac.jp">
関東学院大学</a>工学部機械工学科</td>
</tr>
<tr>
<th>学籍番号</th>
<td>9999999999</td>
</tr>
<tr>
<th>メールアドレス</th>
<td>tkouya@mbox.kanto-gakuin.ac.jp</td>
</tr>
</table>
</body>
</html>
```

図 4.4: テーブルを使った自己紹介のページ


```
<html>
<title>最初の Web ページ</title>
<body>
<h1>自己紹介</h1>
<hr>
<table border>
<tr>
<th bgcolor=green>氏名</th>
<td></font><dd><font size=5 color="blue"><s
trong>幸谷　智紀</strong></font></td>
<tr>
<tr>
<th bgcolor=red>学科</th>
<td><a href="http://www.kanto-gakuin.ac.jp">
関東学院大学</a>工学部機械工学科</td>
</tr>
<tr>
<th bgcolor=yellow>学籍番号</th>
<td>9999999999</td>
</tr>
<tr>
<th bgcolor=black>メールアドレス</th>
<td>tkouya@mbox.kanto-gakuin.ac.jp</td>
</tr>
<tr>
<th>その他</th>
<td>
<table>
<tr>
<th>趣味</th>
<td>読書</td>
</tr>
<tr>
<th>資格</th>
<td>普通免許</td>
</tr>
</table>
</td>
</tr>
</table>
</body>
</html>
```

図 4.5: テーブルの中のテーブル

2. 枠無テーブルを利用して，図 4.6 のような自己アピールのページを作れ。

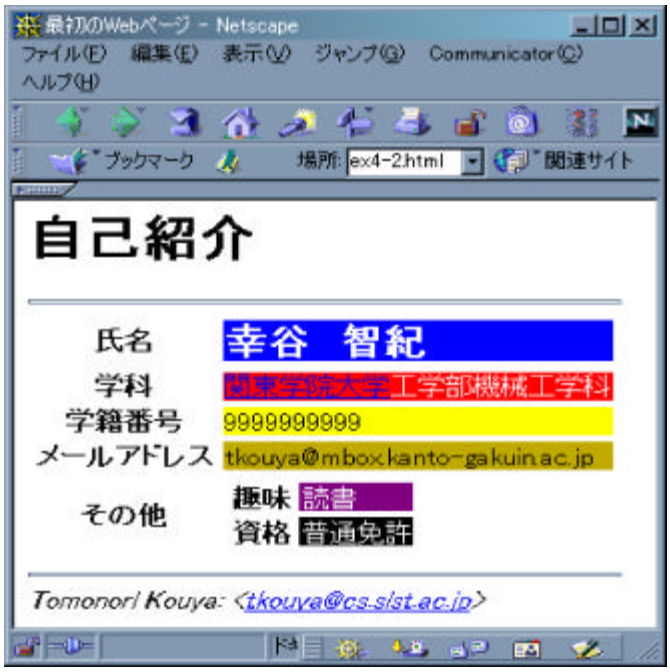


図 4.6: 枠無テーブル版の自己アピール

3. (自由課題) 日本政府の省庁の親子関係が一目でわかるような Web ページを，テーブルを利用して作れ。セルに背景色をつけるなどして工夫すると読みやすいものを作ることが出来る。

コラム★Webサーバとブラウザと通信の実際

> WWW のしくみは拍子抜けするほど簡単だ。単純に TCP/IP のソケットを WWW サーバに対して開いて “GET パス名 HTTP/1.0<CR><CR>” という文字列を送れば，WWW サーバの方は何も考えずに，たとえ相手が親の仇であろうが企業スパイであろうが，そのパス名のファイルを MIME 形式にして返してくる。
>
> 志村 拓・榊 隆 (「インターネットを 256 倍使うための本 Vol.1」より)

殆どの人は，Web ページを見る時，サーバとブラウザ（クライアント）でどんなやり取りが行われているのか，ご存知ないでしょう。また，そのやり取りを盗聴できるなんて考えたこともないでしょう。

別段，そんなことを知らなくても Web ページを見ることも作ることも出来ます。しかし，車のエンジンの構造を知っておけば，アクセルを踏み込むとエンジン音が大きくなる理屈がわかります。エンジンが掛からない時は，セルモータがいかれているか，それを動かすバッテリが上がっている可能性を考え付くことも出来ます。こと，IT 関係の技術は原理原則を知っておくと，困った時に大変役に立ちます。

ここでは実際に Web サーバとブラウザが通信している内容をご覧に入れます。

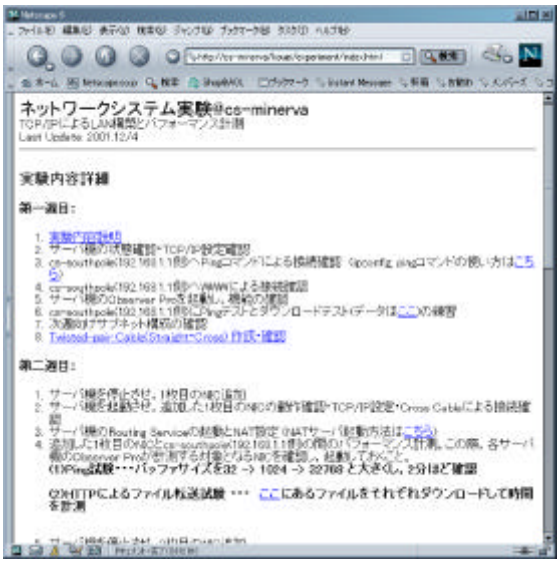


図 4.7: サンプルに使った Web ページ画面

データは小包 (パケット) になってやってくる

第 1 章でも述べていますが，Internet のアイディアはパケット通信 (packet communication) から発生しました。これは全てのデータをパケットという単位に分割し，受信先で再構成するというものです。現在の Internet でもこの精神は全くそのまま生かされており，大きいデータは複数のパケットに分割されて届けられます。Web でもそれは全く変わりません。大き目の HTML ファイルやそこに張り込まれている画像はパケット単位でバラバラにされて送られているわけです。

ここで取り上げる例 (図 4.7) は，画像の含まれていない HTML ファイルですが，１パケットに収まるサイズではないので，最終的には 4 つのパケットに分割されて送信されました。まずこのことを頭に入れておいて下さい。

通信の例

この例では図 4.8 にある通り，クライアント側 (IP アドレス: 192.168.1.11) で Netscape 6 を起動し，Web サーバ (133.133.133.133) にある”/kougi/experiment/”をアクセスしている例です。この通信で飛び交ったパケットは全部で 14 個 (図 4.8 内の (1)〜(14)) あるのですが，そのうち実際に Web ページ転送に必要なデータを運んでいるのは 5 個のパケットです。それ以外のパケットは，送られてきたパケットに対する確認だったり ((5), (7), (10))，Web ページ転送のための準備 ((1)〜(3))・終了 ((11)〜(14)) のために使われています。

ではやり取りされている内容を具体的に見ていくことにしましょう。(1)〜(3) までのパケットのやり取りが行われて[3]，ようやくブラウザからアクセスしたいページの位置を知らせるパケット (4) が Web サーバへ届けられます。その内容は以下のようになっています (一部省略してあります)。

```
GET http://cs-minerva/kougi/experiment/ HTTP/1.0
Accept: */*
Accept-Language: ja
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)
 ...
```

第 1 行目で，アクセスしたいページの位置とそのアクセス手段 (GET)，Web ページ転送の手順方式 (HTTP 1.0) を知らせていることが分かります。以後のデータにも全て意味があり，例えば 5 行目はブラウザ自身の情報 (User-Agent) です。“Mozilla”とは Netscape Navigator のことを示しています。

こうしたブラウザからの要求を受けて，Web サーバはそのページを検索し，該当の HTML ファイルを探してブラウザに送り返します。以下は (6) で運ばれているデータの一部です。

[3]TCP 接続を行うための 3-way handshake(握手) と呼ばれている処理。

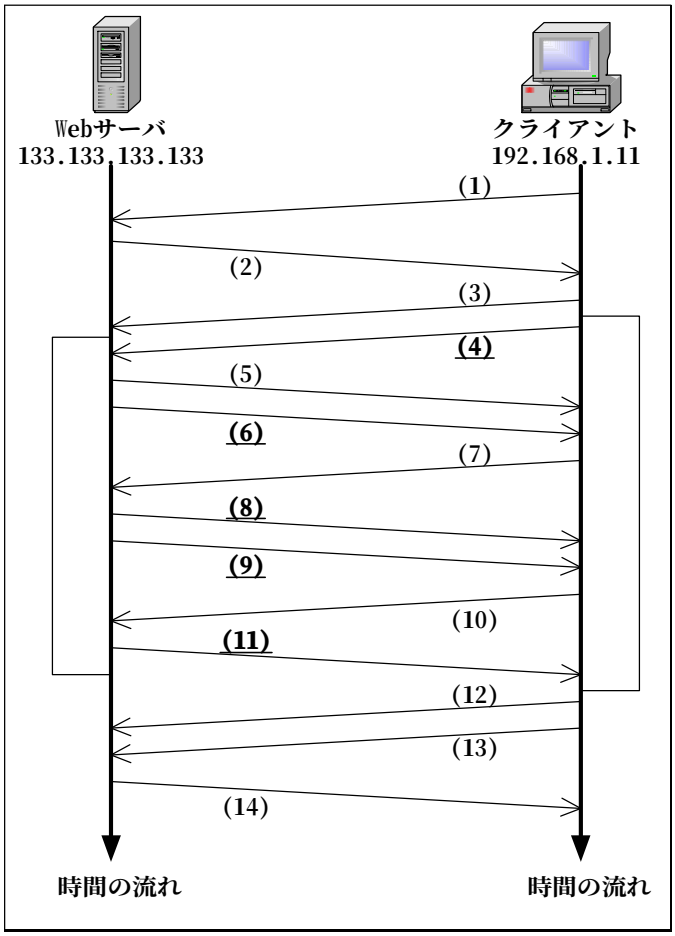


図 4.8: パケットのやり取り

```
HTTP/1.1 200 OK
Date: Wed, 16 Jan 2002 08:03:44 GMT
Server: Apache/1.3.9 (Unix)
Connection: close
Content-Type: text/html

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
 ....
```

その後，この Web ページの内容が分割されてブラウザに送信される訳です ((6), (8), (9), (11))。全てのデータを送り終わった時点で，Web サーバが通信の終了を宣言し ((11))，それを受けてブラウザも通信を切断します ((12)〜(14))。

リンクをクリックするごとに，このような通信が Web サーバとブラウザとで繰り返されているわけです。

おまけ：パケットキャプチャについて

以上の内容は，パケットキャプチャと呼ばれるソフトウェアを使って，通信内容を解読して作成しました。一般に，企業や大学構内で使用されている Ethernet と呼ばれる LAN(Local Area Network) でやり取りされる通信内容は全くガードされておらず，ほぼ丸見えといっていい状態です。最近ではスイッチングハブが使用されることが多いので，自分の通信内容が全て他人に覗かれる危険は少なくなりましたが，パケットそのものにはセキュリティ(安全性) という概念がないということは覚えておくべきでしょう。

以下に，パケットキャプチャの実際の内容をご覧に入れます。何が書いてあるのか，普通の人には全く皆目見当もつかないものですが，コンピュータ通信を勉強するとだんだん理解できるようになります。近くに知識のある人がいるようでしたら，この内容を解説して貰ってみて下さい。これは (8) のパケットの内容の一部です。

```
Packet 8:  00:50:DA:93:13:3C -> 00:50:DA:93:C3:07
    Network:  Ethernet
    Frame type:  802.3,  Frame size:  1518
    Time:  17h:22m 19.703 510s,  Diff. time: 0.000973
    Date:  Wed Jan 16 2002
IP,    133.133.133.133 -> 192.168.1.11
    Source IP:  133.133.133.133,  Destination IP:  192.168.1.11
    Version: 04,    IP header length: 05 (32 bit words)
    Service type:  0:  Precedence: 0, Delay: Norm, Throug: Norm, Reliab: Norm
    Total IP length: 1500
    ID: 3675h
```

```
    Fragment flags: [10] - don't fragment - last fragment
    Fragment offset: 0
    Time to live: 254
    PROTOCOL: [6] TCP
    Header checksum: 8199 (GOOD)
TCP  ACK,   [85] -> [1035]
    Source port: [85]    Destination port: [1035]
    Sequence number: 3877611471,   Acknowledgement: 4093031165
    TCP header length: 05 (32 bit words),   Window: 8760
    TCP data length: 1460,   Checksum: BC0Bh (GOOD)
    Sequence number + TCP data length: 3877612931
Data
    0000  CC 4E 49 43 82 CC 93 AE  8D EC 8A 6D 94 46 81 45     NIC の動作確認・
    0010  54 43 50 2F 49 50 90 DD  92 E8 81 45 43 72 6F 73    TCP/IP 設定・Cros
    0020  73 20 43 61 62 6C 65 82  C9 82 E6 82 E9 90 DA 91    s Cable による接
    0030  B1 8A 6D 94 46 3C 2F 6C  69 3E 0D 0A 0D 0A 3C 6C     確認</li>....<l
    0040  69 3E 0D 0A 83 54 81 5B  83 6F 8B 40 82 CC 52 6F    i>.. サーバ機の Ro
```

— メモ —

第5章　“Authoring Tool”とは？

「現在，最も人気のあるブラウザは，単なるディスプレイエンジンです。みないっせいにファイル変換ソフトに飛びつきました。Word で入力してボタンを押して変換する。これはあるべき姿ではないのです。変換を勧めている限り，オーサリングは絶対に学べません。」(R.Cailliau インタビュー)

N.Randall(村井 純 監訳「インターネットヒストリー」より)

HTML 4.01 Transitional で定義されているタグの一覧については付録を参照して下さい。種類も多いですが，それに伴うプロパティも沢山ありますから，全部正確に覚えようとすると結構大変です。使うタグだけ覚えるようにすれば何とかなるでしょうが，そもそもタグは自分が望むようにWeb ページを構成するための道具に過ぎません。タグを覚えずに Web ページが作れるのであれば，その方が能率が上がる場合もあるでしょう。

ワープロと同じ WYSWYG(What You See is What You Get=見たままのイメージが手に入る）感覚で Web ページを作成するためのソフトウェアをオーサリングツール (Authoring Tool) といいます。タグの種類は PC の機能が向上とユーザの要求と共に増え，複雑化していきました[1]。Web ページを作成する専門家になるのであれば兎も角，簡単に自分の望む Web ページを作りたいのであれば，オーサリングツールを使ってみるのは良いことでしょうし，そのようなソフトウェアの手助けを受けて Web ページを作ることがこれからの主流になっていくことでしょう。本章では Netscape Communicator 4.7x に付属している Composer を中心に，オーサリングツールについて簡単に解説します。

5.1　代表的なオーサリングツール

現在，日本国内で流通しているオーサリングツールは，無料ソフトウェア・有料ソフトウェア共に様々なものがあります。ワープロソフトにも，文書を HTML ファイルとして保存できる機能を持っているものがあり，これも含めると更に数が増えます。私自身，それ程多くのソフトウェアを使いこなしているわけではありませんので，ここでは良く知っている 2 つの有料・無料ソフトウェアを紹介します。

図 5.1 に，有料オーサリングツールの「ホームページビルダー」と無料ソフトウェアの「Netscape Composer(以下，Composer と略記)」の画面を載せておきました。

一見して，前者の方がツールバーの量が多く，機能が多いことが分かります。しかし，このようなソフトウェアは機能が多ければいいというものではありません。自分の用途に合った使い方が出

[1]将来，HTML は XML というタグの種類を限定しない，よりメタな言語に置き換えられていくとも言われているが，筆者はそのようには考えていない。HTML は今後も長期間に渡って使用されると考えている。

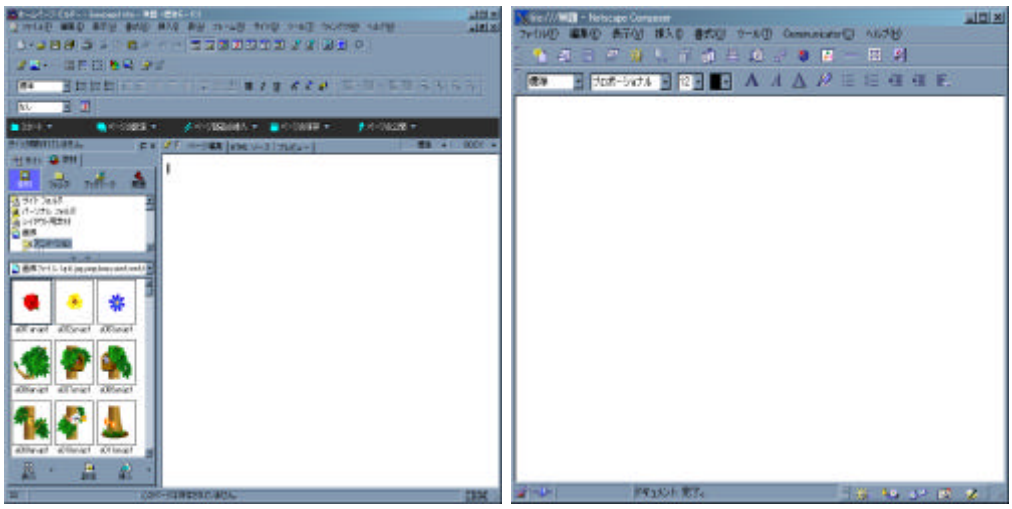


図 5.1: (左)IBM 製のホームページビルダー V.6 (右)Netscape Composer

来ればいい訳ですから，人によっては機能が多い方が便利でしょうし，最低限の機能だけ持っている方が見通しがよいという人もいるでしょう。

ホームページビルダーは，現在の 2 大ブラウザで使用可能な機能をかなりの部分フォローしており，後述する CSS(Cascading Style Sheet) もサポートしていますから，使いこなせばかなり便利なオーサリングソフトウェアです。値段は少々張りますが，それに見合った見返りを得ることが出来るでしょう。詳細を知りたければ，IBM の製品ページ [9] を参照して下さい。以降の章では，Composer を中心に説明していきます。

5.2 Netscape Composer の機能

5.3 メニューバーの全機能

ファイル (F)

- 新規作成
  - Navigator ウィンドウ (N)
  - メッセージ
  - 空白ページ
  - テンプレートからページを作成
  - ウィザードからページを作成
- ページを開く
- 上書き保存
- 名前を付けて保存

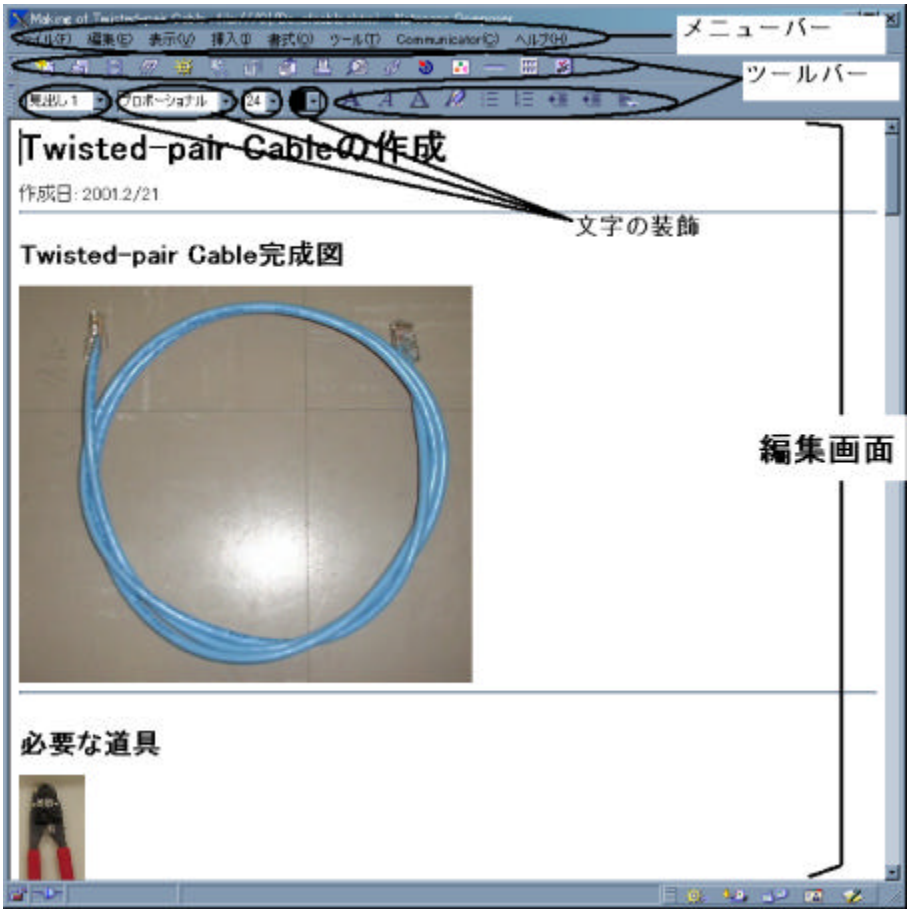


図 5.2: Netscape Composer

図 5.3: Netscape Composer のメニューバー

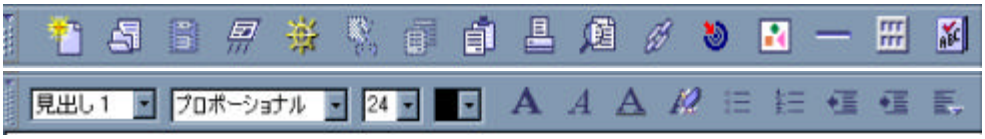


図 5.4: Netscape Composer のツールバー

図 5.5: ツールバーのボタンにフォーカスが移ると ···

- 出版
- ページを送信
- ページをブラウザで表示
- ページ設定
- 印刷プレビュー
- 印刷
- 閉じる
- 終了

編集 (E)

- 元に戻す
- 切り取り
- コピー
- 貼り付け
- 削除
- 表を削除
  - 表
  - 行
  - 列
  - セル
- リンクを削除
- すべて選択

- 表を選択
- ページ内を検索
- 次を検索
- HTML ソース
- 設定

表示 (V)

- 表示
  - 構成ツールバー
  - 書式ツールバー
  - コンポーネントバー
  - 段落マーク
- 再読み込み
- 画像を表示
- 再描画
- 読み込みを停止
- ページのソース
- ページ情報
- 文字コードセット

  … 略 …
  - 日本語 (自動選択)
  - 日本語 (Shift_JIS)
  - 日本語 (EUC_JP)

  … 略 …

挿入 (I)

- リンク
- ターゲット
- 画像
- 横罫線
- 表
  - 表
  - 行
  - 列
  - セル

- HTML タグ
- 改行
- 画像の下側に改行

書式 (O)

- フォント
  - プロポーショナル
  - 固定ピッチ

    ··· 略 ···
- サイズ
  - 8〜36
- スタイル
  - 太字
  - 斜体
  - 下線
  - 取り消し線
  - 上付き
  - 下付き
  - 点滅
  - 改行なし
- 色
- 全てのスタイルを削除
- 見出し
  - 1〜6
- 段落
  - 標準
  - アドレス
  - 整形済み
  - 引用文
  - 説明書きの表題
  - 説明書きのテキスト
- 箇条書き
  - なし
  - 行頭文字付き

  - 番号付き
  - 説明書き
  - ディレクトリ
  - メニュー
- 配置
  - 左
  - 中央
  - 右
- インデント
- インデント解除
- 文字のプロパティ
- 表のプロパティ
- ページの配色とプロパティ
- 暗号化して保存

ツール (T)

- スペルチェック

Communicator(C)

… 略 …

ヘルプ (H)

… 略 …

練習問題

1. ツールバーで出来ることを全て説明せよ。また，メニューバーのどの機能がそれに対応しているかも調べよ。
2. (自由課題) Composer 以外のオーサリングツール (例えばここで紹介したホームページビルダー等) の機能について，お互いに比較検討せよ。また，その使い勝手についても議論せよ。

— メモ —

第6章　Webページをアップロードする

自分のパソコンで作ったWebページは，誰もがアクセスできるWebサーバにあって初めて本来の役割を果たします。ここではHTMLファイルをWebサーバに移す作業をやってみましょう。本章では主にWS_FTP(`http://www.wsftp.com/`)をFTPクライアントソフトウェアの例として使用します。が，特定のアプリケーションにしか存在しない機能は使用していません。他のFTPクライアントソフトウェアでも同様のことは十分可能ですから，もしWS_FTP以外のものを使いたい時には，ここで取り上げた操作をどのように行うのか，チェックしてみて下さい。。

6.1　FTPクライアントソフトウェアとモード

FTP(File Transfer Protocol)とは，ファイルを送受信するための手順(プロトコル)です。本来はそうなのですが，FTPを使って実際に作業を行うためのソフトウェアも“何々FTP”とか“FTP何々”あるいは単に“ftp”という名前であったりして，少しややこしいことになっています。ここではそのようなプログラムをFTPクライアントソフトウェアと呼ぶことにします。

FTPという手順は，正確に言うと2つのTCP[1]接続を使用してファイルのやり取りを行います。片方は，制御コマンドを送受信するために，もう片方はファイルのデータそのものを送受信するために利用します。最初は当然，FTPクライアントソフトウェアから，FTPサーバへ接続要求を行います。この時点では前者のTCP接続だけで良いのですが，それが完了すると，今度は後者の接続をFTPサーバ側から行ってきます。このような動作を行うモードを，Active(能動的)モードと呼びます。

最近まではこのような手続きで問題なかったのですが，近年はセキュリティの問題があって，外部から内部への接続は極端に制限されているのが普通です。そのような環境では，FTPサーバからの接続が拒否されてしまい，ファイルデータの送受信を行うTCP接続が出来ません。そこで，その接続もFTPクライアントの方から行うという方法が考え出されました。このモードをPassive(受動的)モードと呼びます。

通常，FTPクライアントソフトウェアはActiveモードに設定されていると思われますので，もしFirewallやProxyサーバを介して外部と接続する環境で，ファイルがうまく送受信できないのであれば，Passiveモードに切り替えて再度試してみて下さい(図6.1)。

[1]Transmision Control Protocolの略。接続を維持しつつデータのやり取りを行う必要がある，高信頼性を要求する手順(プロトコル)でよく利用される。HTTPもこのTCP接続を利用している。

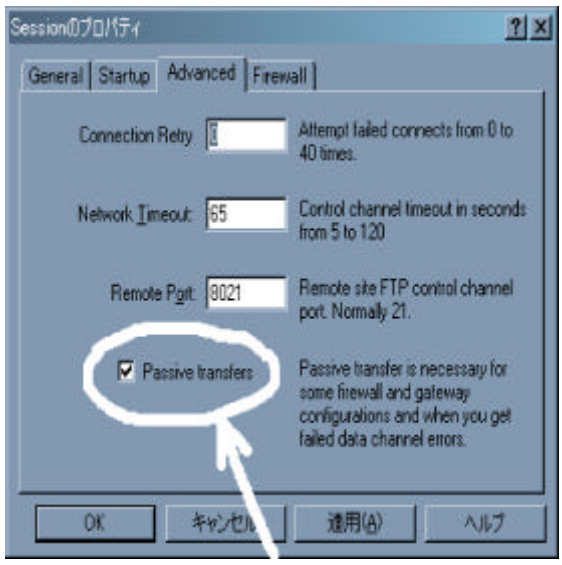


図 6.1: Passive モードの設定 (WS_FTP の場合)

6.2　準備

自分の PC で作成した HTML ファイルや，それと一緒に使用する画像ファイルは，Web サーバに転送する必要があります。ブラウザと Web サーバ間のやり取りは HTTP(HyperText Tranfer Protocol) という手順で行われますが，Web サーバへのファイルを送り込む (これを「アップロード (Upload) する」と呼びます) には通常，FTP が利用されます[2]。

FTP を利用するには

- (1) ユーザ ID と (2) パスワード
- (3)FTP サーバの FQDN，もしくは IP アドレス
- Web ページを送り込むディレクトリ名[3]
- FTP クライアントソフトウェア

が最低でも必要になります。ユーザ ID が “anonymous”，パスワードなしで利用できる FTP サーバもありますが[4]，ファイルをサーバに送り込むにはパスワードを要するユーザ ID で接続しないと許可されていないのが普通です。今回は HTML ファイルを送り込む作業なので，Web サーバ (FTP サーバも兼用していることが多い) の管理人が指定したユーザ IP とパスワードが必要になります。

どこの ISP でも，Web ページを開設するサービスを提供している所では，入会申込書に希望するユーザ ID とパスワードを書き込む場所があると思います。その時には自分しか知らない，他人

[2]HTTP でもファイルのアップロードは可能だが，許可していないことが多いという意味。
[3]決められた名前のディレクトリ以下のファイルが Web ページとして使用できるようになっているのが普通。Web サーバごとに異なるが，“public_html”や “homepage”等のディレクトリ名を指定されることが多い。
[4]Anonymous(匿名) FTP サーバと呼ばれます。日本では Ring サーバ (ftp://ftp.ring.gr.jp/) が有名。

に類推されづらく，自分には覚え易いパスワードを指定して下さい。万が一パスワードを忘れてしまったら管理者に泣きついて再発行してもらうしか手がありませんので，くれぐれも忘れないようにして下さい。

6.3 アップロードする

では，実際に HTML ファイルをアップロードしてみましょう。WS_FTP の初期画面 (図 6.2) の指定された場所に，(1) ユーザ ID，(2) パスワード (空白にしておくと接続後にパスワード入力画面が表示されます)，(3)FTP サーバの FQDN もしくは IP アドレスを入力します。何度も繰り返し利用するのであれば，(4)“Profile Name”に分かり易い名前を付けておくと便利でしょう。

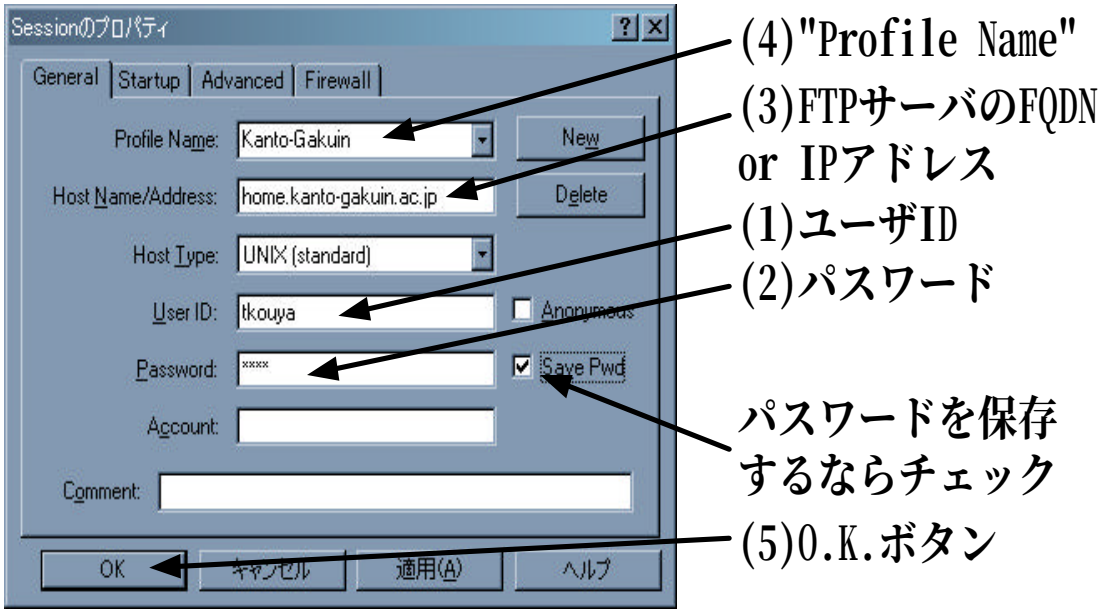


図 6.2: WS_FTP によるファイルの転送: 1. 初期設定

設定が正しければ，下の (5)「OK」ボタンをクリックすれば勝手に接続してくれます。

接続終了後は図**??**のような画面になります。向かって左側に “Local System”(自分の PC のフォルダ (ディレクトリ)) が，右側に接続先の “Remote Site”(FTP サーバのディレクトリ) のファイル一覧が表示されているはずです。フォルダ (ディレクトリ) の移動，ファイルの消去等は Windows 標準の方法で可能です。

自分の PC からファイルを転送する時には，転送したいファイルをダブルクリックするか，ファイルを選択した後，中央の “→”をクリックします。うまく転送できていれば，右側 “Remote Site” のファイル一覧が更新され，転送したファイル名が表示されるはずです。

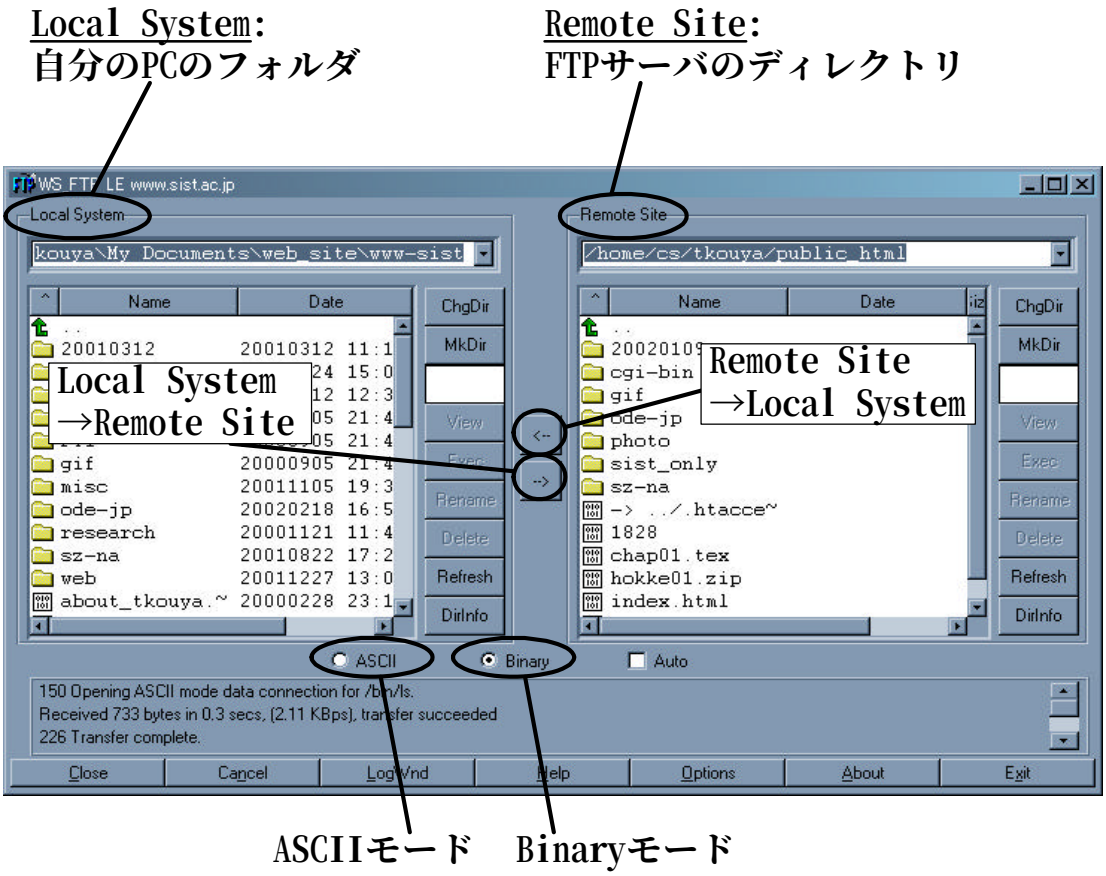


図 6.3: WS_FTP によるファイルの転送: 2. ファイル転送

6.4 確認作業

Upload が終了したら，ブラウザをリロード (Reload, 再読み込み) して確かに更新されたかどうかを確認します。もし，更新個所に誤りがあれば，もう一度修正を行った上で，再度アップロードします。

Upload したにも関わらず，内容が変化しないようであれば次の点を疑ってみて下さい。

1. HTML ファイルを転送したディレクトリ (フォルダ) の位置が誤っている。Web サーバごとに，HTML ファイルを置くディレクトリの名前や場所は異なります。
2. キャッシュが更新されていない。(「コラム★キャッシュとは？」を参照)
3. 内容が更新されていない HTML ファイルを転送している。ちゃんと更新した HTML ファイルを PC に保存しましたか？　画面では更新されているように見えても，それがファイルに保存されていないと転送しても無意味です。

練習問題

1. 本章で行った作業を，別の FTP クライアントソフトウェアで行う場合，どのような手続きをする必要があるのか？　例えば FFTP(`http://www2.biglobe.ne.jp/~sota/`) ではどのような設定が必要が，ファイルの転送はどのように行えばよいか調べよ。
2. 商用オーサリングツールの多くは FTP クライアントソフトウェアの機能も搭載している。従って，Web ページを作成し，それを Web サーバへ転送する作業が一つのオーサリングツールの中で完結することになる。では，そのようなオーサリングツールを使っている場合，WS_FTP や FFFTP のような，単独の FTP クライアントソフトウェアは全く必要ないのか？　必要が出てくるとすれば，それはどんな場面か？

― メモ ―

コラム★キャッシュ(cache) される Web ページ

6.5　キャッシュされる Web ページ

cash というと「現金」のことですが，ここでは cache(大事なものの「隠し場所」の意) のことを「キャッシュ」（共に発音は同じ）と呼びます。

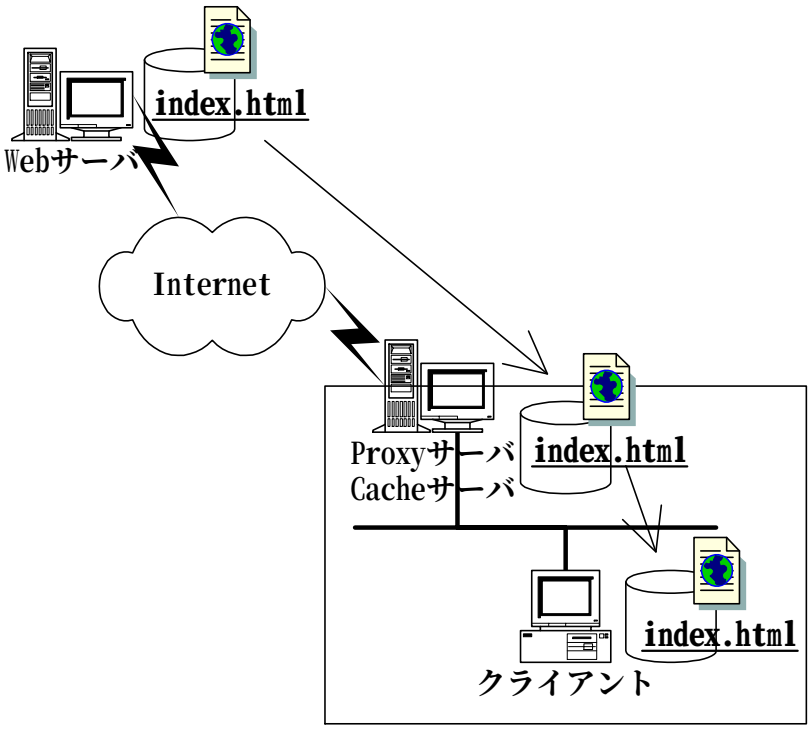


図 6.4: キャッシュされる Web ページ

一般に，Internet と直結している回線は組織内のネットワークよりも速度が遅く，同じ大きさのデータの送受信を行うにもより多くの時間を要します。昨今では ADSL(Asymmetric Digital Subscriber Line) や FTTH(Fiber To The Home) 等が普及し，ブロードバンド (Broadband)，即ち高速な回線がもてはやされていますが，人間の欲求には限りが無く，回線速度が上がればそれに合わせて送受信するデータの量は増えて行きます。組織内のネットワークも，10 倍づつ高速になっています[5]から，

[5]Ethernet は 10BASE(10Mbps)，100BASE(100Mbps)，1000BASE(1Gbps) ･･･ と 10 倍ずつ高速な規格が登場している。10Gbps の登場も間近と言われている (2002 年 2 月末日現在)。もっともこれは最高速度であって，常時この速度が出るわけではない。

相対的に Internet と接続する回線速度が遅いという事情はあまり変化が無いといってよいでしょう。

そうすると，通常，ブラウザは Internet を介して Web サーバに接続し，HTML ファイルをダウンロードして，ユーザの PC 画面に Web ページを表示しますから，同じデータ量でも，組織内のネットワークよりも受信に要する時間が長くなってしまいます。その時間のロスを少しでも減らすため，一度受信した Web ページの内容をブラウザが終了しても保存しておき，それが更新されていないようであれば，保存してあるものを使用して画面に表示する仕組みが考えられました。これをローカルキャッシュ(Local Cache) と呼びます。現在の主流ブラウザである Internet Explorer や Netscape Navigator は，ブラウザが動作している PC のハードディスクとメモリにキャッシュを作成します。消去しない限り，画像や HTML ファイルは指定されたフォルダ (ディレクトリ) に一定期間，一定量に達するまでゴッソリ保存されているのです。

少し大きな会社や大学のような組織では，沢山の人が同じ Web ページにアクセスするのを避けるため，組織内でキャッシュを共有する仕組みを導入するのが普通です。一度アクセスされたページは Cache サーバと呼ばれるコンピュータに保存しておき，別のユーザがアクセスした時に，元の Web ページが更新されていないようであれば，自分が保存しているものをそのユーザへ送ります (図 6.4)。

6.6　Web ページを更新した時には …

従って，自分の Web ページを更新し，Web サーバにアップロードした後に，それを確認する際には，ブラウザを動作させている自分の PC のローカルキャッシュ内に保存されている HTML ファイルを更新し，組織内で Cache サーバ兼用の Proxy サーバ[6]を介してアクセスしている時には，それも更新する必要があります。

通常は，ブラウザの「再読み込み (リロード)」「最新の情報に更新」ボタン，あるいはメニューを選択することでどちらも更新してくれるのですが (図 6.5, 6.6)，自分の PC や Proxy 兼 Cache サーバ，相手の Web サーバの内部時計が狂っていたりすると，それぞれで保存している HTML ファイルの日付もおかしくなってしまい，更新がうまくいかないケースがままあります。

そんな時には，強制的に自分の PC のローカルキャッシュを消去する必要があります。その上で再読み込みすれば，ブラウザは Web サーバからファイルを改めて取得せざるを得ないわけです。Proxy サーバを介している場合，それでもまれに更新されないこともありますが，その時にはサーバの管理者に連絡を取ってみて下さい。Cache サーバは多数のユーザが取得したファイルで溢れ返っており，ディスクが一杯になっていたり，アクセスが集中してファイル取得に失敗してしまうこともあるのです。

ブラウザのキャッシュを消去する時には，Netscape(4.x/6.x) の場合，メニューから「編集」→「設定」→「詳細」へ進み,「キャッシュ」を選択して下さい。確実を期すために,「メモリキャッシュ」「ディスクキャッシュ」の両方を削除して下さい (図 6.7)。Internet Explorer(5.x/6.x) の場合は,「ツール」→「インターネットオプション」→「全般」から,「インターネット一時ファイル」を消去し,「履歴」もクリアしておいて下さい (図 6.8)。

[6]Proxy(代理人) の意。Internet への接続を，内部のマシンに代わって行うサーバ。

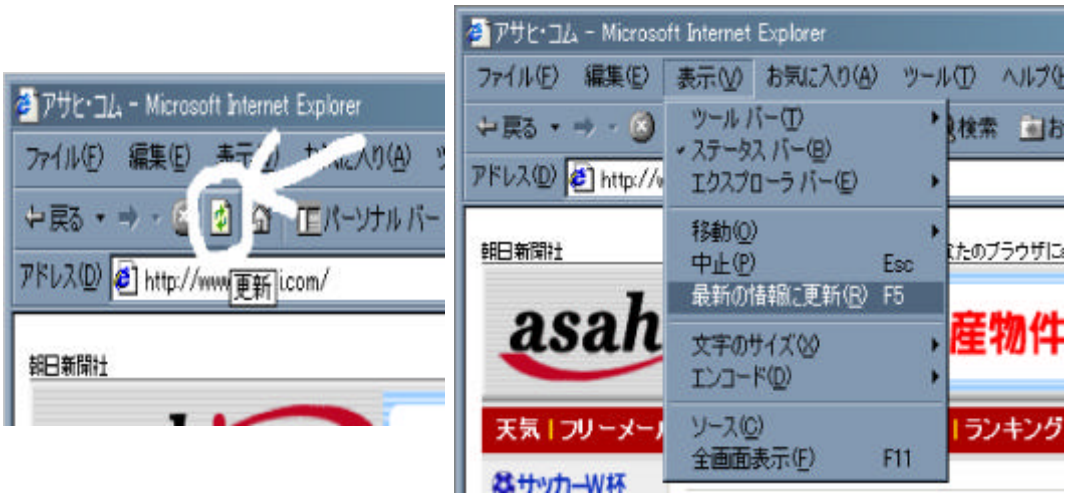


図 6.5: Internet Explorer の再読み込み・更新ボタンとメニュー

図 6.6: Netscape Navigator(左:4.7, 右:6.2) の再読み込み・更新ボタンとメニュー

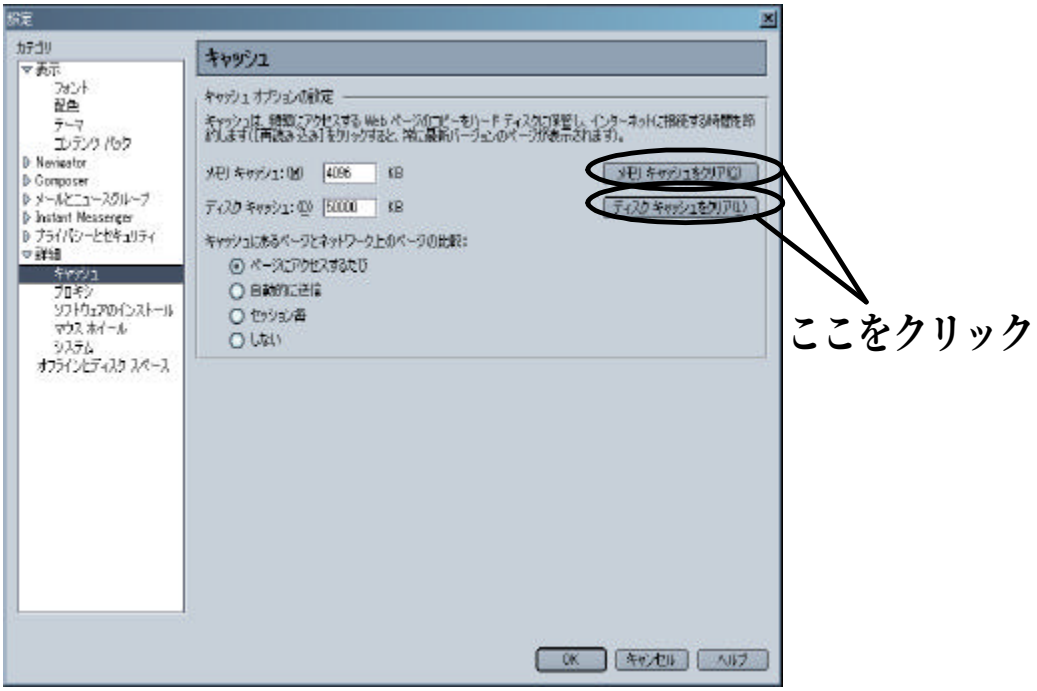


図 6.7: キャッシュの消去 (Netscape 6.2)

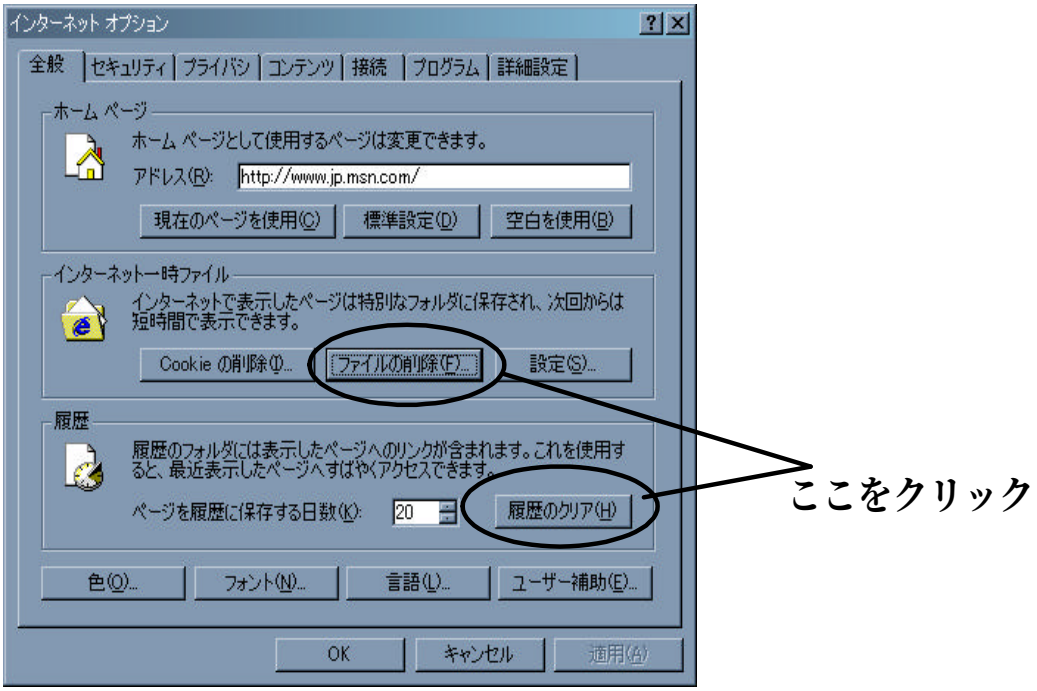


図 6.8: キャッシュの消去 (Internet Explorer)

第7章　Netscape Composerを使ってみる(1/2)

第 5 章で紹介した，オーサリングツールの一つである Composer を使って，今までに作ってきた Web ページを改めて作ってみましょう。

7.1　自己紹介を書いてみよう

前に作った自己紹介の Web ページを Composer の機能を利用して作成してみましょう。完成すると図 7.1 のようになります。

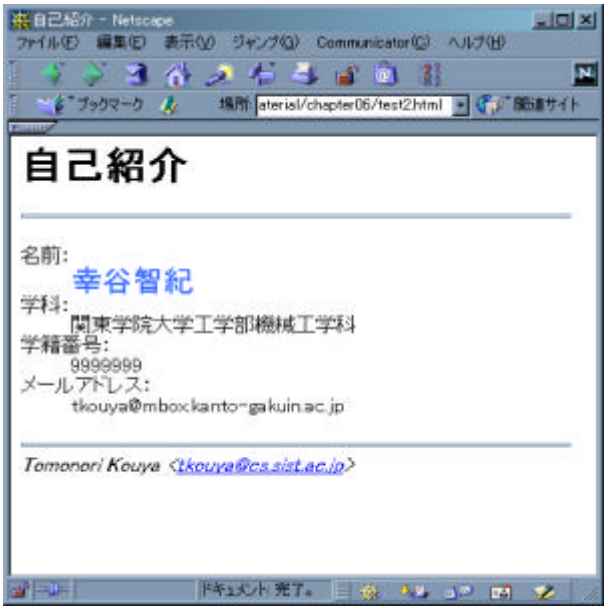


図 7.1: Composer で作った自己紹介

最初ですから，Composer の操作方法を丁寧に見ていくことにします。

1. Composer を起動し，図 7.2 のように文字や水平線を入力します。「名前:」の上と「tkouya@mbox.kanto-gakuin.ac.jp」の下に改行を入れておいて下さい。

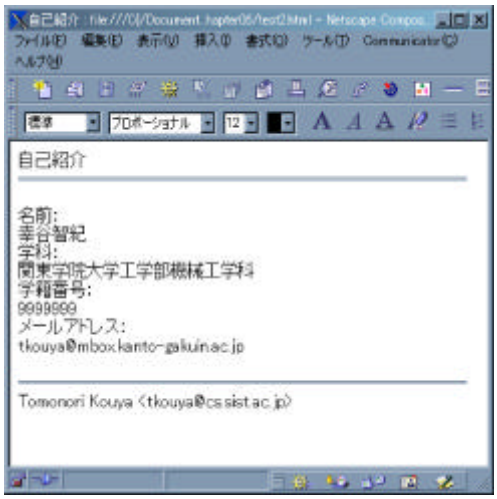


図 7.2:

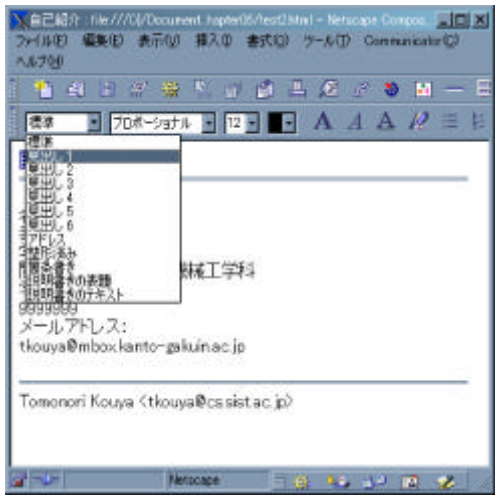


図 7.3:

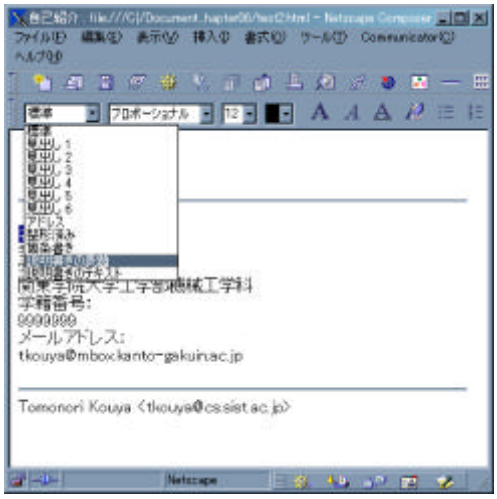


図 7.4:

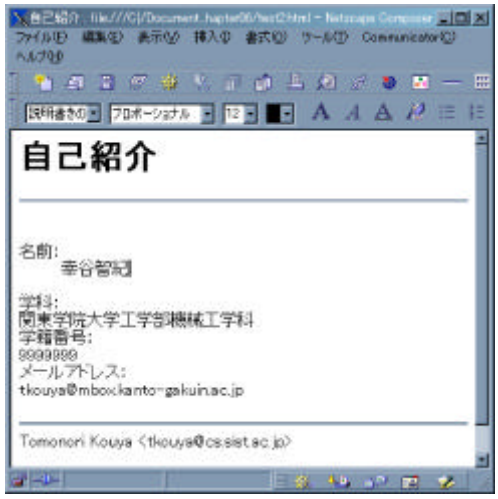


図 7.5:

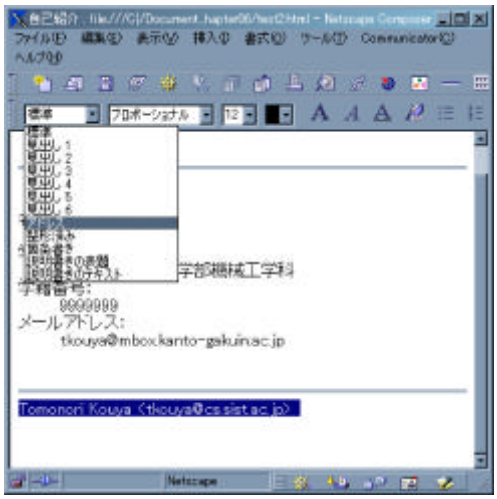


図 7.6:

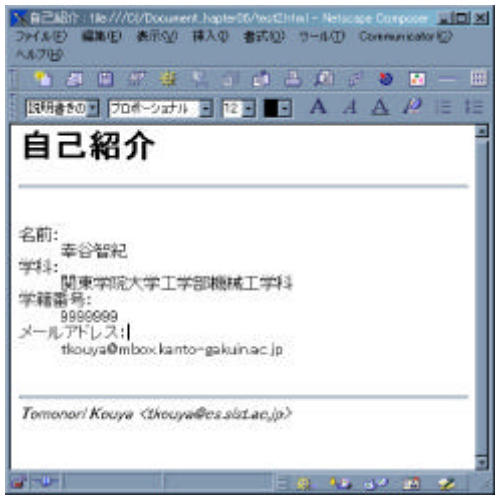


図 7.7:

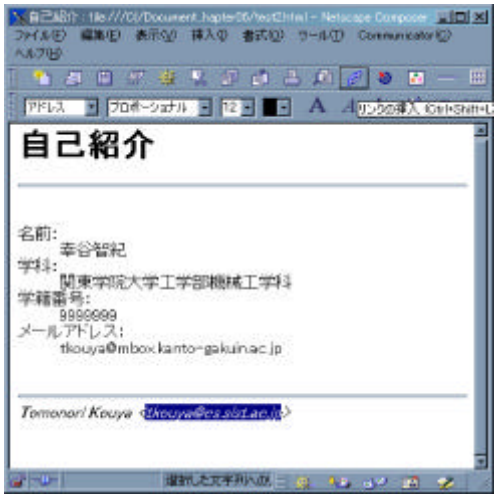


図 7.8:

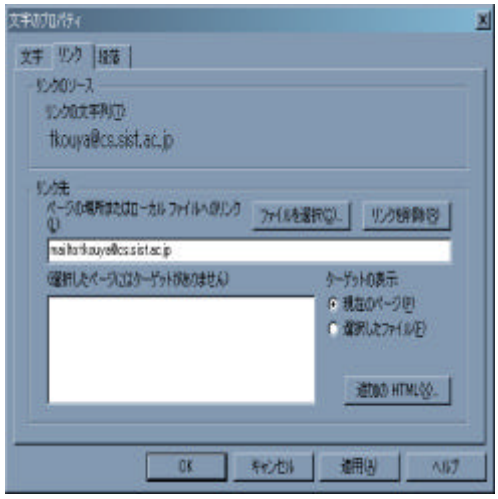


図 7.9:

図 7.10:

2. 一番上の「自己紹介」の部分を選択し，文字プロパティの中から「見出し 1」を選んで大見出しにします (図 7.3)。

3. 同様に,「名前：」を選択して文字プロパティから「説明書きの表題」を選択します。「幸谷智紀」の部分は「説明書きのテキスト」にします (図 7.4)。そうすると図 7.5 のようになります。

4. 以下,「学科：」「学籍番号：」「メールアドレス：」を「説明書きの表題」に,「関東学院大学工学部機械工学科」「9999999」「tkouya@mbox.kanto-gakuin.ac.jp」を「説明書きのテキスト」に変更していきます。

5. 一番下の行の「Tomonori Kouya <tkouya@cs.sist.ac.jp>」の部分を「アドレス」に変更します (図 7.6)。その結果，図 7.7 のようになります。

6. 「tkouya@cs.sist.ac.jp」の部分に “mailto:tkouya@cs.sist.ac.jp”へのリンクを挿入します。まずこの部分を選択しておいて，ツールボックスの「リンクの挿入」ボタンを押します (図 7.8)。図 7.9 のようなウィンドウが出ますので，リンク先の所に URI を書きます。

7. 最後に「幸谷智紀」の所を，フォントサイズを 18 に，色を青にして完成です。適当なファイル名で保存しておいて下さい (図 7.10)。

うまく出来ましたか？　Composer に限らず，オーサリングツールはそれぞれ独特の癖がありますので，試行錯誤しつつその特性を掴むようにして下さい。

7.2　テーブルを使う

次は，テーブルを使った自己紹介の Web ページを Composer で作成してみます。

図 7.11: テーブルを使った自己紹介

前の節の説明と重複する所は省きつつ，作り方を以下に示します。

1. Composer を起動し，図 7.12 のように入力し，適宜文字プロパティを変更し，リンクを挿入しておきます。
2. カーソルを 2 本の水平線の間に移動して，ツールバーの「表を挿入」ボタンを押します (図 7.13)。
3. これから挿入するテーブルのプロパティを図 7.14 のウィンドウで決めます。作りたいテーブルは 5 行 2 列ですからそのように変更し，OK ボタンを押します。すると図 7.15 のようにセルが空白の 5 行 2 列のテーブルが 2 本の水平線の間に挿入されます。
4. セルに文字を入力していきます。セル間を横に移動するときには [tab] キーを使います (図 7.16)。
5. 「その他」の横の空白のセルに移動し，先程と同じ手順で 2 行 2 列のテーブルを挿入します。この時,「枠線の幅」のチェックを外すと，枠無のテーブルを作ることが出来ます (図 7.18)。
6. 挿入した表中の表に文字を入力し，表題の文字プロパティを変更していきます。

図 7.12:

図 7.13:

図 7.14:

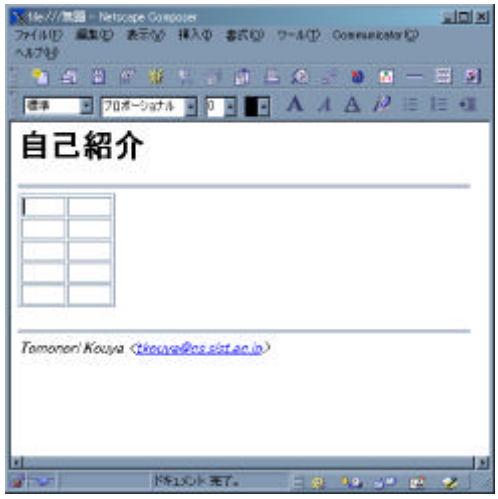


図 7.15:

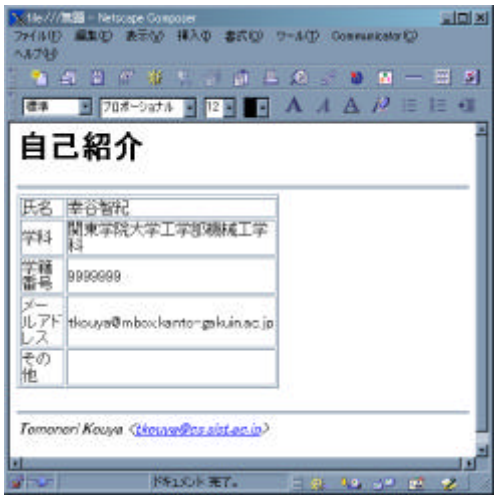


図 7.16:

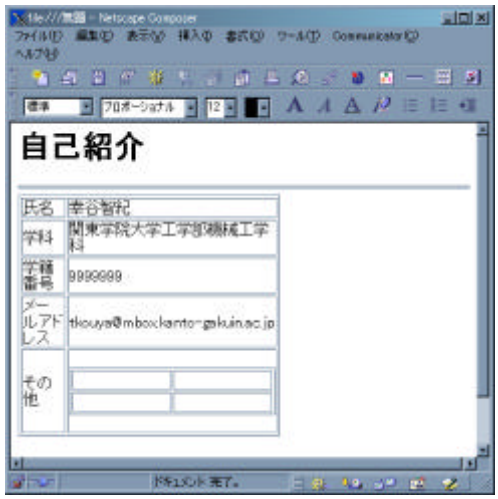


図 7.17:

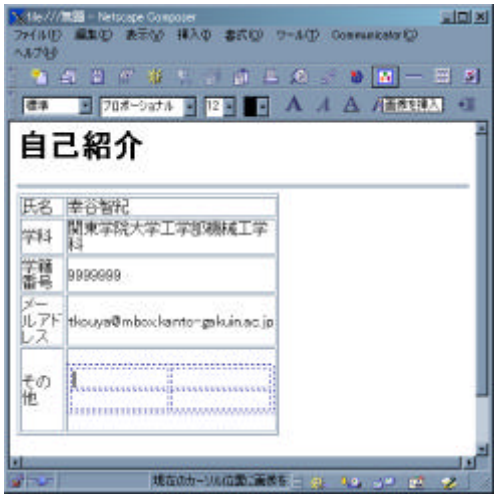


図 7.18:

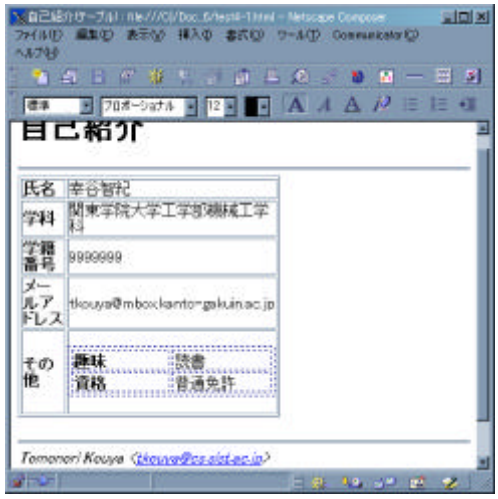


図 7.19:

図 7.20:

7. 各セルの背景色は，カーソルをそのセルへ移動し，マウスの右ボタンを押すと,「テーブルのプロパティ」Window が表示されますので,「セル」のプロパティへ移動し，そこで背景色を指定して下さい。

8. 表の幅やセルの幅については適当に変更して完成です (図 7.20)。

どうでしょうか？　テキストエディタで全てのタグを打ち込んでいくよりも，仕上がり状態を見ながら編集が出来る点は便利ですが，細かい修正がしづらいという点もあり，かなりイライラしたのではないでしょうか。しかし，そのイライラ感は Composer を使いこなしつつ，適切な作業手順を自分なりに見つけることでしか解決できません。頑張って下さい。

練習問題

1. 自分の講義時間割を Composer で作れ。例えば図 7.21 のようなもので良い。

2001年度前期(4月～7月)週間スケジュール(予定)

	月	火	水	木	金
9:00 - 10:30				微分積分	不在
10:40 - 12:10	ネットワーク管理				不在
13:00 - 14:30	不在			ネットワークシステム実験	不在
14:40 - 16:10	不在	会議(第2,3火曜日)		ネットワークシステム実験	不在
16:20 - 17:50	不在	会議		ネットワークシステム実験	不在

2001年度後期(9月～2002年1月)週間スケジュール(予定)

	月	火	水	木	金
9:00 - 10:30					セミナー
10:40 - 12:10			インターネット論		不在
13:00 - 14:30			論理数学及び演習	コンピュータネットワーク	不在
14:40 - 16:10		会議(第2,3火曜日)	論理数学及び演習		不在
16:20 - 17:50		会議			不在

図 7.21: 時間割の例

— メモ —

総合課題1

100 のリンクから成る Web ページを作れ。その際には次の条件を満足すること。

1. URI が恒久的に利用できそうな，なるべくトップに近いものを選択すること。例えば新聞社の Web ページへリンクする場合，時間の経過と共に消去されてしまう記事のページではなく，記事全体へのリンクを持っているトップの Web ページへのリンクを張るようにする。
2. リンクに際しては，なるべくその Web ページ内にリンクの許諾条件を確認すること。「許可なくリンクすること禁止」等の断り書きがある場合は別の Web ページを選択せよ。
3. リンク先の Web ページの画像や素材は絶対に使用しないこと。あくまでリンクのみに留めるようにする。
4. 集めたリンクは自分なりにまとめたカテゴリに分割し，1 ページに収まるよう，見やすく配置せよ。あまり一つのカテゴリに属するリンクが他よりも多くなりすぎることがないよう，バランスを考えてリンクを収集し，配置すること。

例を図 7.22 に掲載しておいたので適宜参照せよ。

図 7.22: 100 のリンク集の例

第8章　画像の利用

> しかし正直言って僕は，コンピュータ作成のグラフで記憶に残るようなものにはいまだお目にかかっていない。僕の経験からすると，本当に感銘を受けるようなプレゼンテーションは，発表者が黒板やオーバーヘッドプロジェクタを使って，話の要所要所でグラフを即興で描いてみせたり，要点を手早くまとめて書いたりするようなものだ。… (略) … つまりプレゼンテーションで重要なのは，自分の理解している内容をどれだけ効果的に伝えられるかであって，どれほど凝ったグラフを見せられるかではない。
>
> C. Stoll(「インターネットはからっぽの洞窟」より)

リンクと並んで重要な Web の利点は画像を多用できることです。百聞は一見にしかず，とは限りませんが，写真や絵などと多用すると一見華やかな文書を作ることが出来ます。ここでは簡単にその使い方を見て行く事にしましょう。

8.1　ファイルの種類 — BMP/GIF/JPEG/PNG

画像をコンピュータ上で扱うためには，前述した通り 16 進数のデータとして表現しなければなりません。画像の表現方式は大きく分けて次の 2 つがあります。

ビットマップ (bitmap) ... 絵を縦横に分割し，その一区画を一点 (ドット, dot) として表現します。色数は一点を何ビットで表現するかによって決まってきます。従って扱うことの出来る色数を増やしたり，ドット数が多くなるとデータは大きくなってしまいます。

ドロー (draw) ... 絵を曲線・直線の集合として捉え，線分であれば短点のデータとそれを結ぶ線の種類・太さ・色だけをデータとして保存します。ビットマップに比較して，建築設計図のようなものはデータ量が少なくて済みますが，写真のように一点一点細かく変化するような絵を表現するには向いていません。

どちらにしても，それぞれデータの格納の仕方，圧縮の有無などで様々なバリエーションが存在します。

Web の場合，大抵はビットマップで表現された絵を扱います。代表的なフォーマットには次のものがあります。

GIF ... 元々は CompuServe というコンピュータ通信サービスでよく使われていた画像フォーマット形式です。アニメーションも出来 (図 8.1)，Web の初期にはブラウザはこの形式しかサポー

トしていませんでした。その後，このフォーマット形式で使用されている圧縮アルゴリズムに UNISYS の特許がかかっていることが問題視され，GIF を捨てようという運動も起こりました。

BMP ... Windows では古くから標準の画像フォーマットとして用いられてきましたが，Web では標準ではありません。使わないに越したことはありません。

JPEG ... 圧縮機能が細かく設定できる画像フォーマット形式です。Web では GIF と並んでよく使用されています。

PNG ... GIF の代わりに使用できる特許料の掛からない画像フォーマット形式として，近年注目されるようになりました。古いブラウザではサポートされていない場合があります。

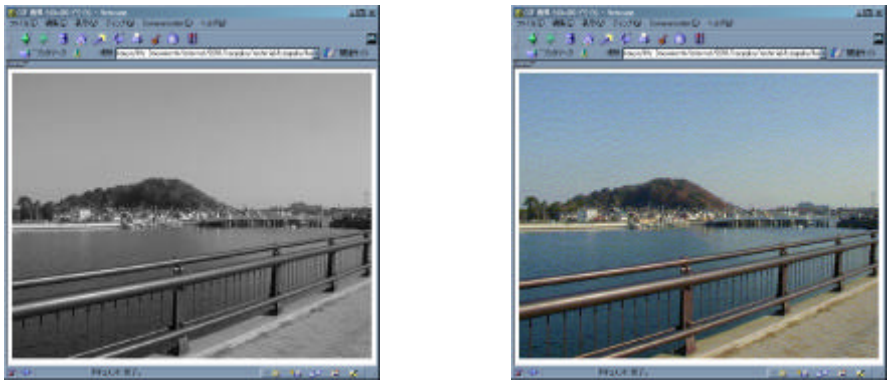


図 8.1: アニメーション GIF の例 (左右のカラー・白黒写真が交互に表示される)

これらの画像フォーマットはソフトウェアでお互いの形式に変換することが可能です。

8.2　絵を入れてみよう

画像がを張り込むには次のようなタグを使います。

```
<img SRC="hakkeijima_90.png" height="10%" width="10%">
```

height と width で画像の大きさを変化させることが出来ます。但し，画像データそのものは変化しませんので，ユーザの負担を減らす目的で画像を小さくしたいというのであれば，あらかじめ画像編集ソフトを使い，画像のサイズ (ドット数) を減らしたり，色数を減らすなどしてデータ量を減らして下さい。

Composer の場合は直接ファイルをドラッグ & ドロップで張り込んでもよいし,「画像を挿入」メニューから選んでも Web ページに貼り付けることが出来ます。画像を貼った後，それを右クリックするとその表示形式のプロパティを調整するためのウィンドウが開きます (図 8.2)。

同じ画像の大きさを変化させて貼り付けてみました (図 8.3)。

図 8.2: Composer の画像プロパティ編集画面

練習問題

1. (自由課題) 装飾用の画像や背景の素材を自由に使えるよう提供している Web ページは数多く存在する。適宜自分の好みのものを見つけ，今まで作ってきた自分の Web ページを装飾せよ。

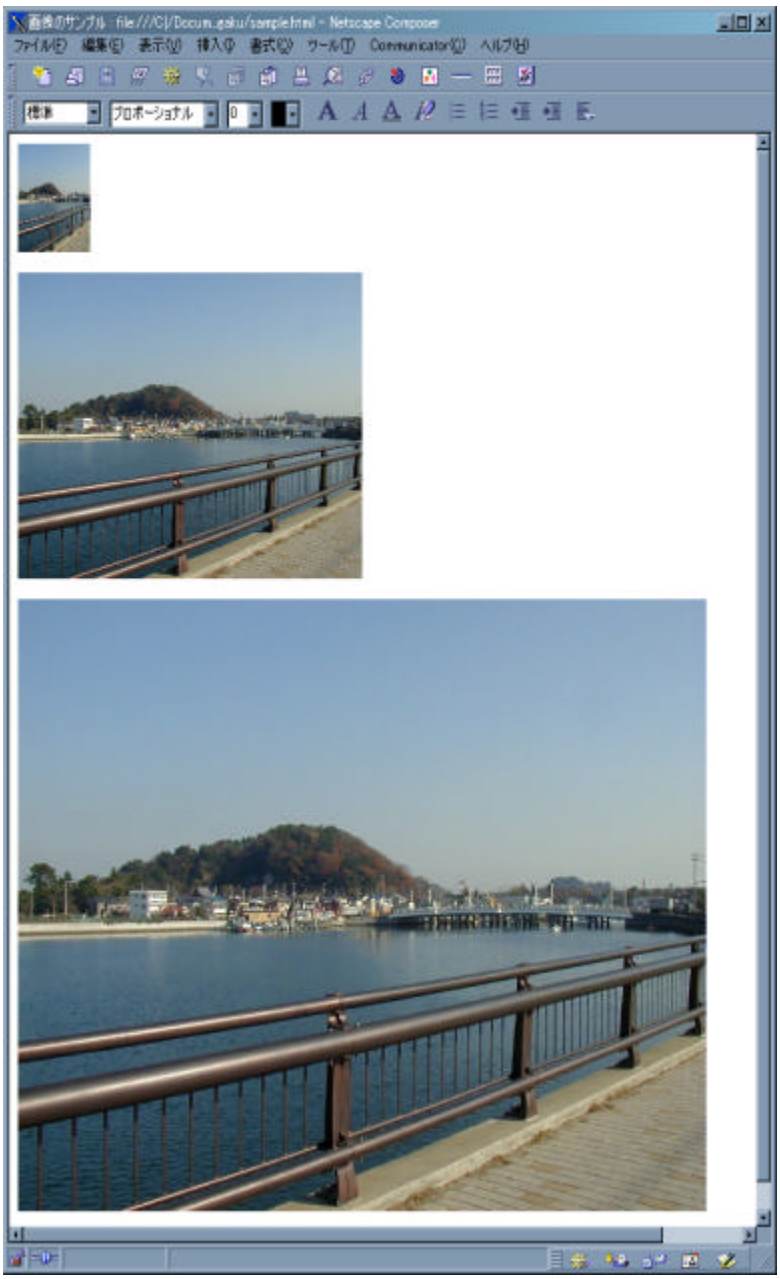


図 8.3: 画像の大きさ変更

第9章　Netscape Composerを使ってみる(2/2)

ここでは今まで使ってきたComposerの機能を使って，ちょっと大きなWebページを作ってみましょう。LAN[1]でよく利用されるTwisted-pair cableを作る手順を紹介するWebページを作ります。

9.1　Twisted-pair cableの作成

目指すWebページの構成は次のようにします。

- Twisted-pair cable完成写真を最初に載せる
- 作成に必要な工具・材料一覧を示す
- ストレートケーブル作成方法を縷縷，画像で紹介する
- クロスケーブルの結線写真を示す
- 結線のテスト結果を画像で示す

素材はデジタルカメラで集めましたが，最近は安くなりましたので揃えておくと便利かもしれません。スキャナもあると便利ですが，他人の著作物を取り込んで勝手に自分のWebページに使用してはいけません。

9.2　材料を取り揃える

素材となる画像をお見せします(図9.1〜9.7)。全てJPEGファイルになっています。

ケーブル完成図 — `cccable.jpg`(図9.1),

必要な道具 — `caulker.jpg`, `stripper.jpg`, `nippers.jpg`(図9.2)

材料 –`rj-45.jpg`, `cable.jpg`(図9.3)

ストレートケーブルの作り方 — `making1.jpg`〜`making6.jpg`(図9.4), `making7.jpg`〜`making12.jpg`(図9.5)

クロスケーブルの作成 — `cross.jpg`(図9.6)

結線のテスト — `straight_test.jpg`, `cross_test.jpg`(図9.7)

[1]Local Area Network の略称。

図 9.1: ケーブル完成図

図 9.2: 工具

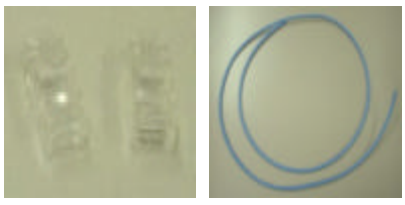


図 9.3: 材料

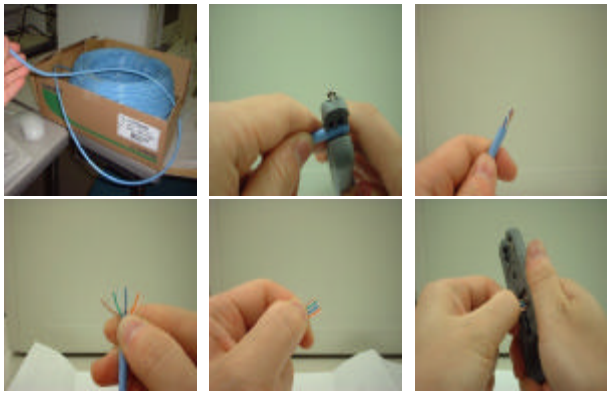


図 9.4: ストレートケーブルの作り方 (1/2)

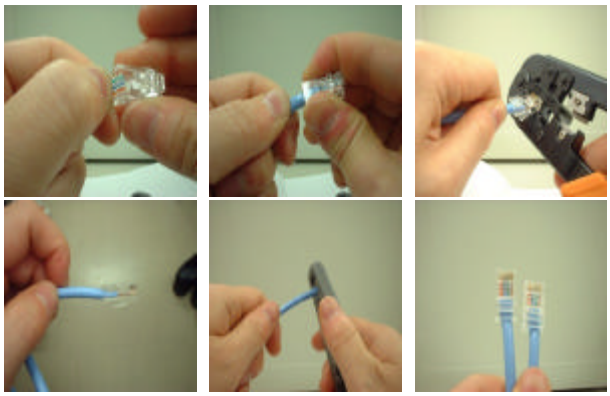


図 9.5: ストレートケーブルの作り方 (2/2)

図 9.6: クロスケーブルの結線

図 9.7: ケーブルのテスト

9.3　Web ページを作成する

適宜説明を加えながら素材の写真を並べていきます (図 9.8, 9.9)。その前に，画像は作成する HTML ファイルと同じフォルダ (ディレクトリ)，もしくはそれより下位のフォルダ (ディレクトリ) にまとめて置いてあることを確認して下さい。そうでないと，Composer はうまく相対パスに変換してくれず,「絶対パスになってしまうぞ！」という旨の警告を発します。

練習問題

1. 図 9.8，9.9 の Web ページには改善が必要であると思われる。自分なりに改善点を洗い出し改良せよ。一例は次章に提示したようにフレームの機能を使うことである。
2. (自由課題) 第 2 章，第 3 章で行った，テキストエディタによる HTML ファイル作成方法を解説する Web ページを作れ。

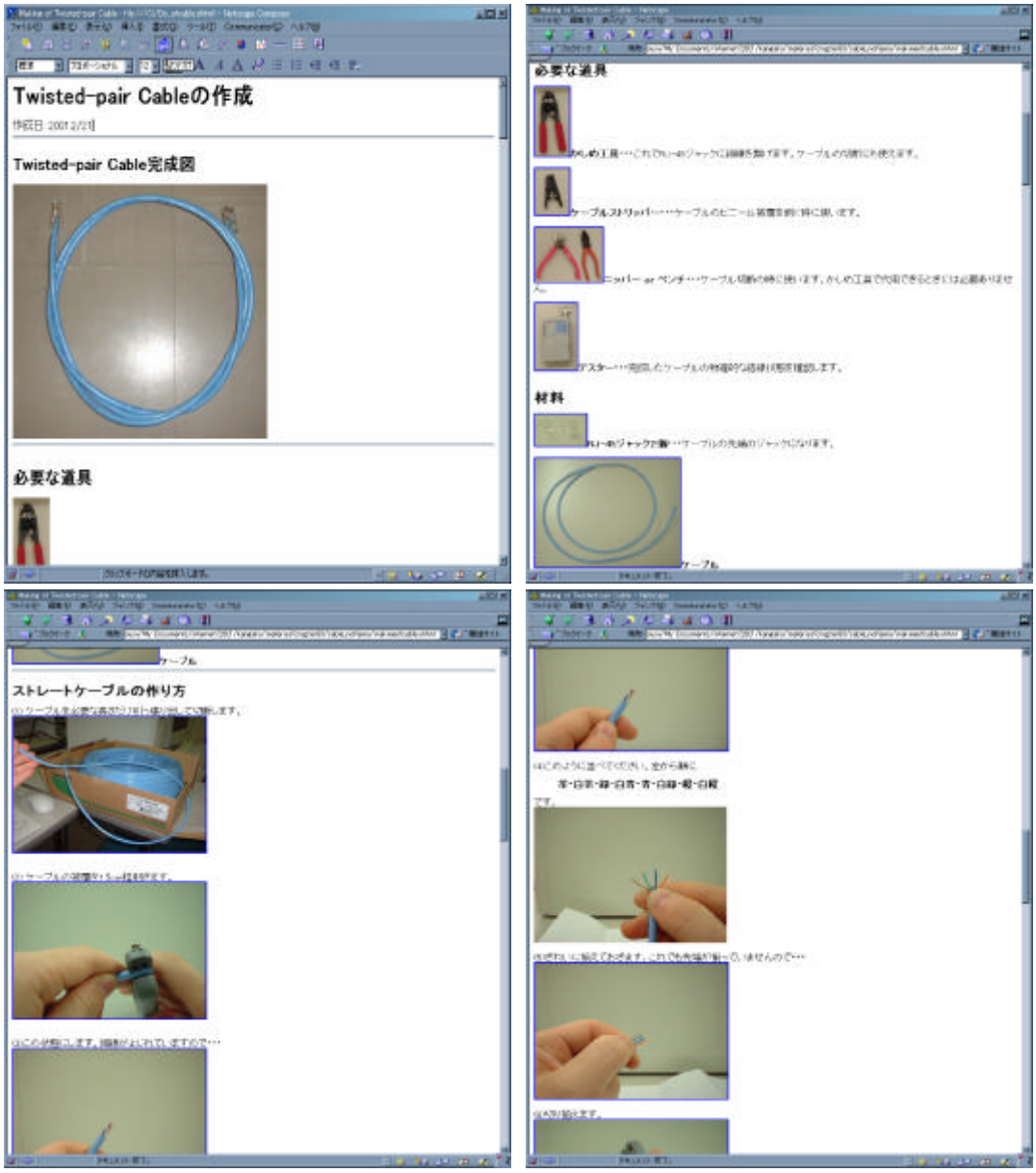


図 9.8: 完成した Web ページ (1/2)

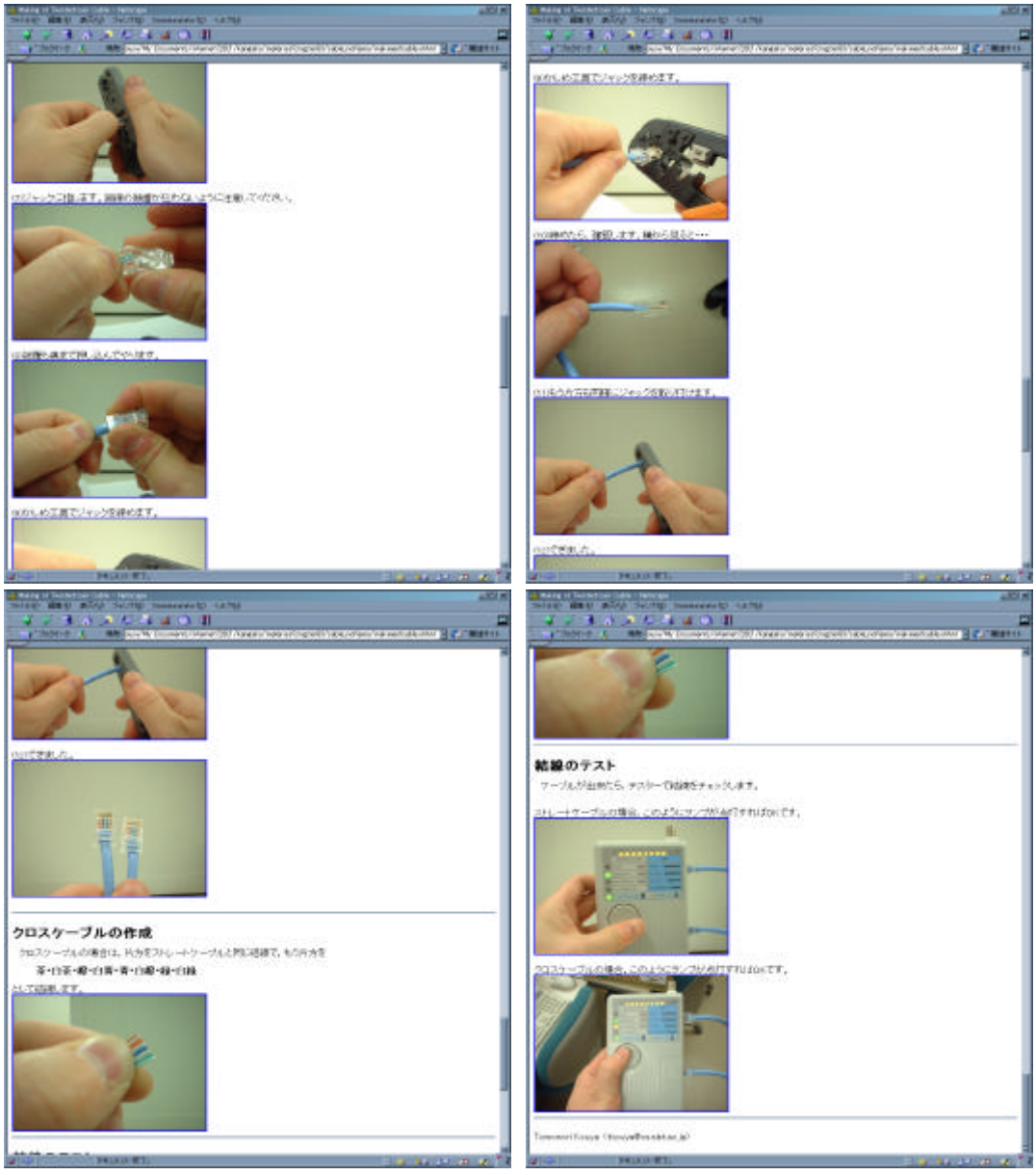


図 9.9: 完成した Web ページ (2/2)

コラム★Accessibility を考える

最初のネックは液晶画面だった。
「あくまで電話」にこだわった私たちは，液晶画面もそれまでより少し大きいくらいのものをイメージしていた。あまり液晶画面が大きいと，ユーザーが手にしたとき電話というには違和感を感じさせるからだ。
そうは言いつつも，私はいままでの携帯が表示していた以上の文字数を要求した。
「横八文字，縦六文字はほしいですね」

松永真里 (「i モード事件」より)

Accessibility = アクセスのし易さ

コンピュータの操作は，昔は文字 (コマンド) を打ち込んで行っていました。それが初心者にはわかりづらいということで，現在ではその殆どがグラフィックベースの，所謂 GUI(Graphical User Interface) に移行しています。その為，視覚障碍がある人にはかえって利用しづらくなった，という話を聞いたことがあります。

Web の世界ではよく Accessibility(アクセシビリティ) という言葉が使われます。直訳すれば,「アクセスのし易さ」，ということになるのでしょうが，狭義には五感に障碍がある人でにも Web ページの内容を伝えることが出来るかどうか，といった意味で使用されます。よく，視覚的な効果を狙った Plug-In(Shockware Flash 等) で構成された Web ページが非難の対象になりますが，それはこの Accessibility が劣っているからに他なりません。最低でも文字読み上げソフトが対応できるよう，代替のテキストを用意しておく必要があります。

「障碍者はマイノリティ(少数派) であるから，マジョリティ(多数派) である健常者だけに対応できていればよい」という考え方もあるでしょう。しかし，これは自分の Web ページのお客さんを自ら排除していることに他なりません。

マジョリティを相手にすることと，マイノリティにも一定の配慮をすることは矛盾しません。また，マイノリティへ配慮することはマジョリティを増やす効果もあります。それに，健常者が全て同じ条件にあると思うこと自体もおかしな話です。

Web の Accessibility とは？

五体満足な人でも，Web にアクセスする環境はまちまちです。今では携帯電話が飽和状態といってもいいぐらい，この日本では普及していますから，マジョリティを考えるならば，自宅や職場で

PC からアクセスしている人よりも携帯ユーザを考えるべきでしょう。都市部ではブロードバンド化が進む一方で，過疎地では未だ ISDN すら満足に使えない地域も残っています。古い Macintosh を使っている人もいるかと思えば，最新版の Pentium IV を搭載したゴツイマシンをぶん回している人もいます。そう考えていくと，果たして多数派とは何なのか，だんだん分からなくなってきます。

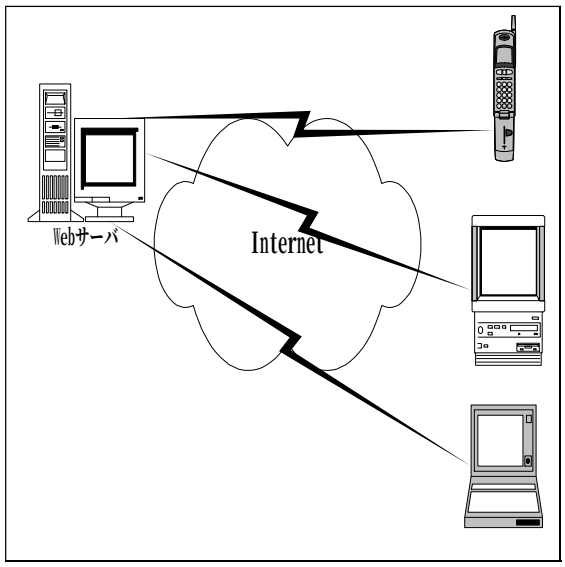


図 9.10: Web ページを見る人は千差万別

そこで一つの原則を提案しましょう。

自分の想像しうる一番貧弱な環境でも余計なストレスなしに内容が判明する Web ページを作るべし！

貧弱な環境とは例えば次の様なものです。

貧弱なディスプレイ PC の最低レベルは 640 × 480，16 色程度。携帯になると画面はさらに小さく，モノクロであることも考えるべきです。

貧弱なブラウザ 今でこそ機能豊富な Netscape Navigator や Internet Explorer ですが，初期の頃は <TABLE> タグも認識できず，画像は GIF 形式でしか利用できない時代がありました。現在はそこまで極端に考えなくとも良いとは思いますが，特殊なプラグインを必要とするものは見てもらえない可能性が高いと考えるべきでしょう。

貧弱な帯域幅 携帯ですと 9.6kbsp 程度のスピードしか得られないと考えるべきです。また，いくらブロードバンドが流行しているといえど，コンテンツを掲載する Web サーバの性能やそこに直結する回線容量が貧弱であれば，末端ユーザ側の回線に余裕があっても何にもなりません。

貧弱な環境を考慮しておくと，それ以上の環境を持つ人も取り込むことが可能になります。また，最低レベルを上げて行く事で，切り捨てられてしまうユーザ環境の境界線も見えてきます。「俺のページは携帯ユーザは考慮しない！」「基本的には自分の PC 環境程度のユーザを主体にするが，携帯ユーザにも最低限の情報は判別できるよう，テキストを主体に作る」という割り切りをする意味でも，他人の貧弱な環境に思いを馳せるようにしましょう。

誰のための Web ページですか？

Web ページは自分以外の第 3 者に読んでもらうための表現手段の一つです。Web ページをサーバにアップロードする前に，もう一度考えてみましょう。

- 貴方が作っているそのページは，一体誰に向かって発信するものなのでしょうか？
- リッチな環境に頼りすぎて，貧弱な環境のお客さんを排除していませんか？
- 貴方の Web ページは，他人に理解できる内容になっていますか？

Accessibility を向上する究極の目的は，他人を思いやることで自分の為にする，ということなのです。

— メモ —

第10章　あなたのWebページは何点？

HTMLにはきちんとした規格が存在しますが，ブラウザがきちんと規格を満足していなかったり，使われないタグや機能も沢山あります。HTMLはイイカゲンに書いてもエラーをはいたりしないぞんざいなところがありますので，普段はあまり厳密に書いたりしませんし，その必要性も感じないでしょう。しかし，文法に則った，きれいなHTMLを作成しておくと，後で他の形式に変換するのが楽になります。ここで紹介するのは厳密にHTMLをチェックするサービスです。一度は自分の作成したWebページがどのぐらい文法に忠実であるか，チェックしておくと良いでしょう。

10.1　HTML Validation

Webに関する様々な規格を定めたり，提示してくれたりしている重要なサイトとしてW3C[4]という組織があります。そこで提供されているHTMLのチェッカーです(図10.1)。

規格に則ったHTMLをこのチェッカに掛け，首尾よく合格すると「このbitmapを貼りなさい」という表示がされます。それを貼ったのが図10.2です。この場合はHTML 4.01という規格に則ったWebページである，という証になっています。

10.2　Another HTML-lint

先のサービスは全て英語で結果が表示されますが，文法チェックの結果が日本語でも表示できるものがあります(図10.3)。これは後で述べるPerlスクリプトによるCGIとして動作するようになっており，CGIが動作する環境であれば，自分の所にダウンロードして使用することも可能です。

10.3　100点でも-50点でも気にしない？

実は，Composerを用いて作ったHTMLファイルもこれらのチェッカに掛けるとエラーが出てきます[1]。試しにめぼしいURIを指定してこれらのチェッカに掛けてみると分かると思いますが，多くのサイトはそのデザインに関係なく，これらのテストには合格できないことが分かります。即ち，HTMLとしては「規格に則って記述されていない」ということになります。しかし，ブラウザの柔軟性に助けられ，殆どのサイトでは「ちゃんとしていないHTML」でも問題なく表示できるわけです。

一度，自分の作ったWebページをこれらのチェッカに掛けてみると面白いと思います。

[1] IBMのホームページビルダー Ver.6が生成したWebページは一部の例外を除き，HTML4.01形式として及第点のWebページが生成できる。

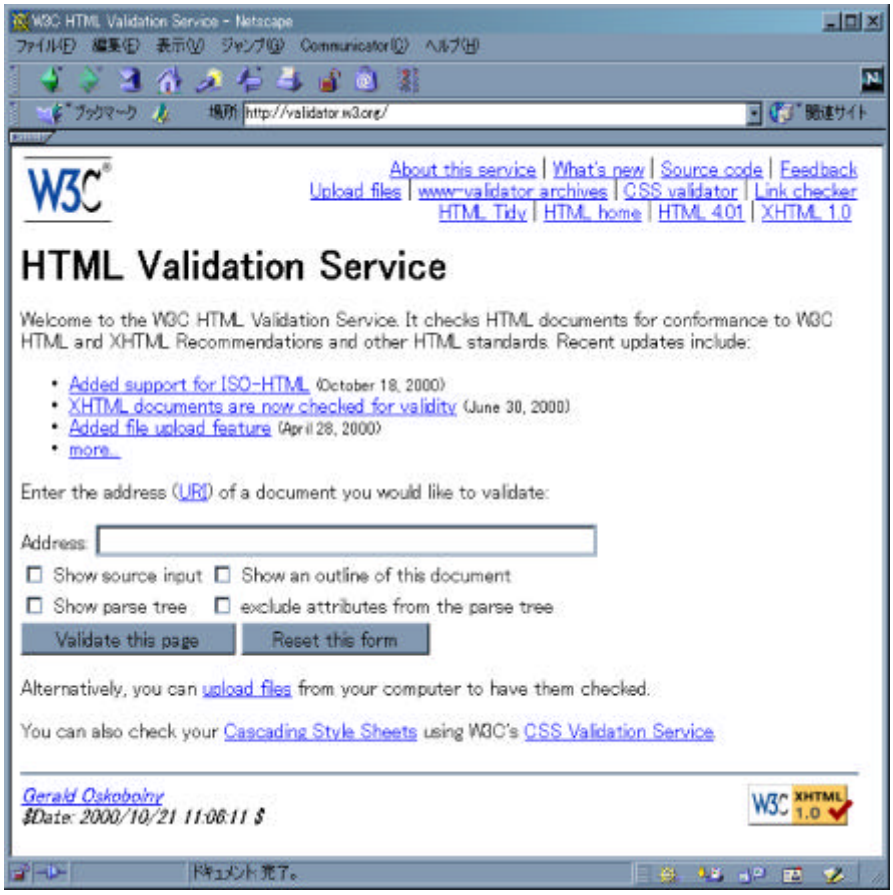


図 10.1: W3C HTML Validation Service

図 10.2: 合格のしるし

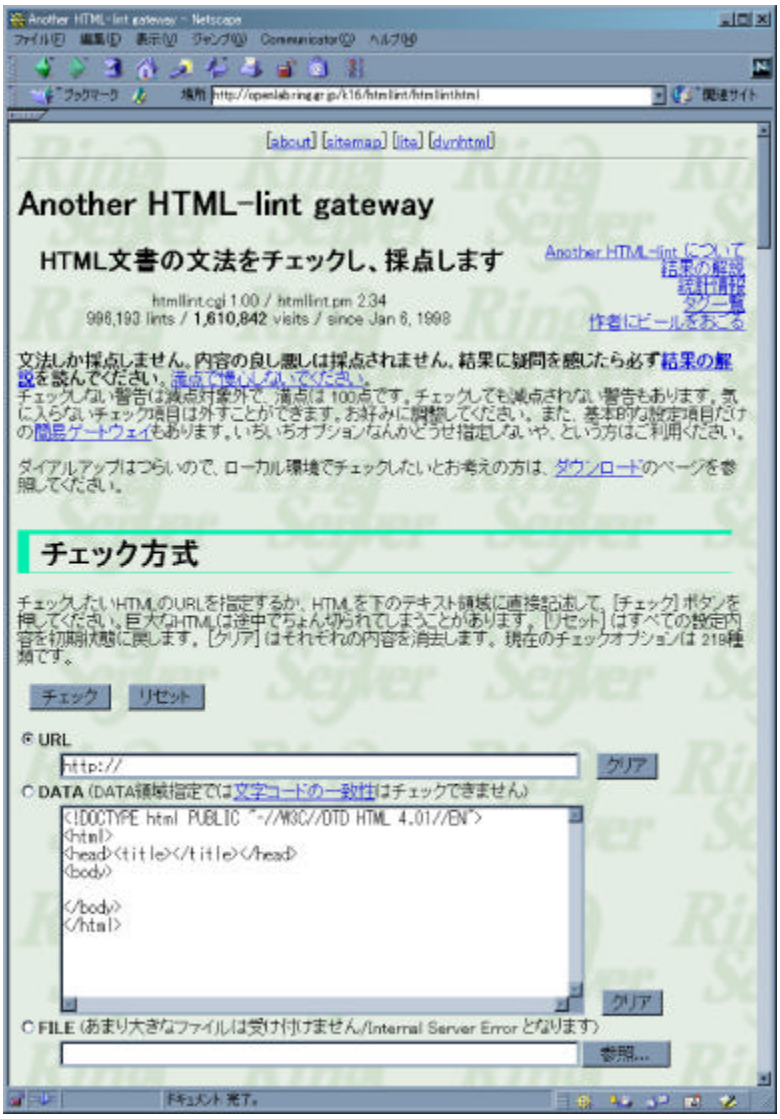


図 10.3: Another HTML-lint

練習問題

1. このようなサービスが存在する理由を考えよ。また，これらの文法チェッカに合格するような HTML ファイルを生成する意義はどこにあるのか？　これらのサイトのコメントを参照し，どのように主張しているかを調べよ。

2. 今まで自分が作ってきた Web ページを，HTML チェッカに掛けてみよ。エラーがあれば完璧になるまで修正してみよ。なお，この修正作業はテキストエディタで行う必要がある。

第11章　フレームを使って格好良くしよう

ここでは前の章で作ったような，長い Web ページを分割して読みやすくする手法としてフレームを紹介します。

11.1　フレームで画面分割

フレームとは図 11.1 のように，ブラウザの画面を縦方向 (column，列) と横方向 (row，行) に分割したものをいいます。

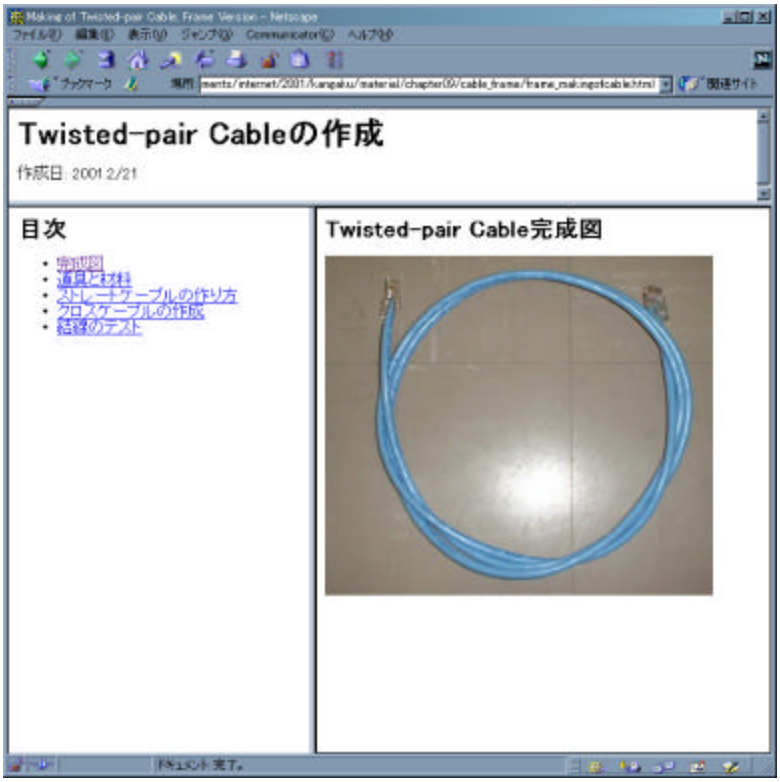


図 11.1: フレームを使った Web ページ

Composer にはフレームを作る機能はありません。従って，どのように分割するかはテキストエディタを使って HTML を記述して指定しなければなりません。例えば図 11.1 のように分割するに

は図 11.2 のようにします。

```
<HTML>
<TITLE>Making of Twisted-pair Cable; Frame Version</TITLE>
<FRAMESET rows="15%,*">
<FRAME name="title" src="title.html">
<FRAMESET cols="40%,*">
<FRAME name="menu" src="menu.html">
<FRAME name="main" src="kanseizu.html">
</FRAMESET>
<NOFRAMES>
<BODY>
<FONT SIZE="3" COLOR="RED">このページはフレーム対応のブラウザで閲覧して下さい。
</FONT>
</BODY>
</NOFRAMES>
</FRAMESET>
</HTML>
```

図 11.2: フレーム本体

フレームは，フレームを表示する機能に対応しているブラウザでなければ見ることは出来ません。対応していない場合は <noframes> タグで囲まれた所が表示されます。

フレーム対応のブラウザでは分割されたフレームそれぞれに “src=”で指定された HTML ファイルが表示されます。もし指定されたフレームの名前 “name=”をリンクのターゲット (Target)“target=” として指定すると，リンク先の HTML ファイルはその名前のフレームに表示されるようになります。これがターゲットの役割です。ターゲット名は <A> タグの中に追加して指定します。

なお，フレーム内に表示される Web ページは自作のものに限るようにしましょう。勝手に他のサイトを表示するのは相手側の著作物を勝手に利用していることになります。

ではフレーム表示のための部品を作ることにします。まず，前章で作った Web ページを図 11.3 のように

1. title.html
2. kanseizu.html
3. tool.html
4. straight.html
5. cross.html
6. test.html

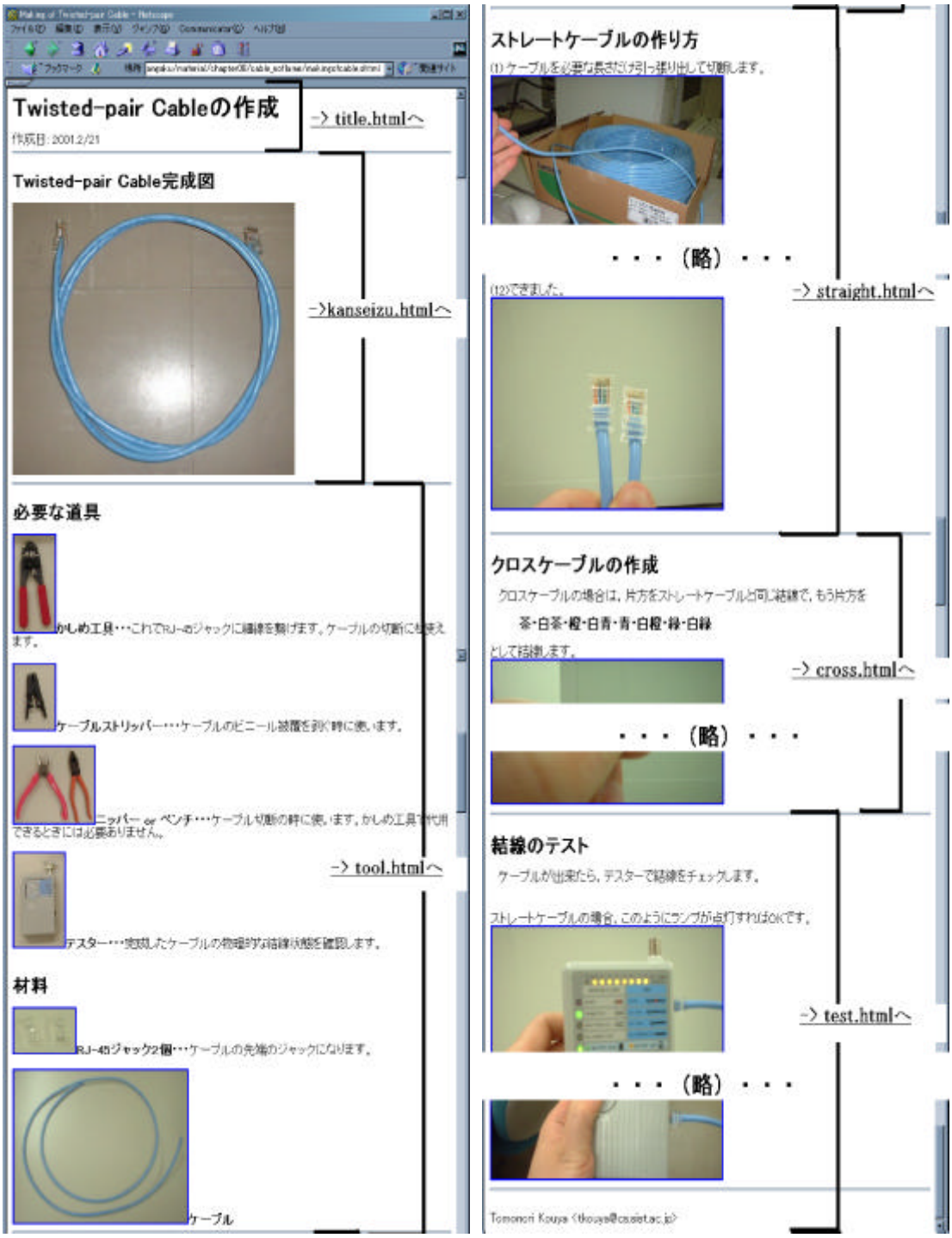


図 11.3: Web ページを分割する

図 11.4: メニュー用の Web ページ

と，6 つの HTML ファイルに分割して下さい。

次に，メニュー用の “menu.html”を Composer で新規に作成します (図 11.4)。

リンクを張るときには図 11.5 に示す手順で，どのフレームにリンク先の Web ページを表示させるかを指定します。

完成したら，フレームを表示し，メニュー内のリンクが正しく動作するか確認して下さい。

11.2　別 Window への表示

フレームに分割されていない Web ページでも，リンクをクリックすると別ウィンドウを表示するようなものがあります。これはフレームで使用したようにターゲットを指定することで可能になりますが，この場合は存在しないターゲット名を指定します。

存在しない名前のターゲットを指定したリンクをクリックすると，ウィンドウも別に開かれます。同じ名前のターゲットを持つ別のリンクがあれば，既に開かれたウィンドウはそのターゲット名を持ちますから，そのウィンドウにどちらのリンク先も表示されるようになります (図 11.6)。

練習問題

1. (自由課題) 前章までに作成してきた Web ページを全て統合して表示する Web ページを，フレームを使って構成せよ。

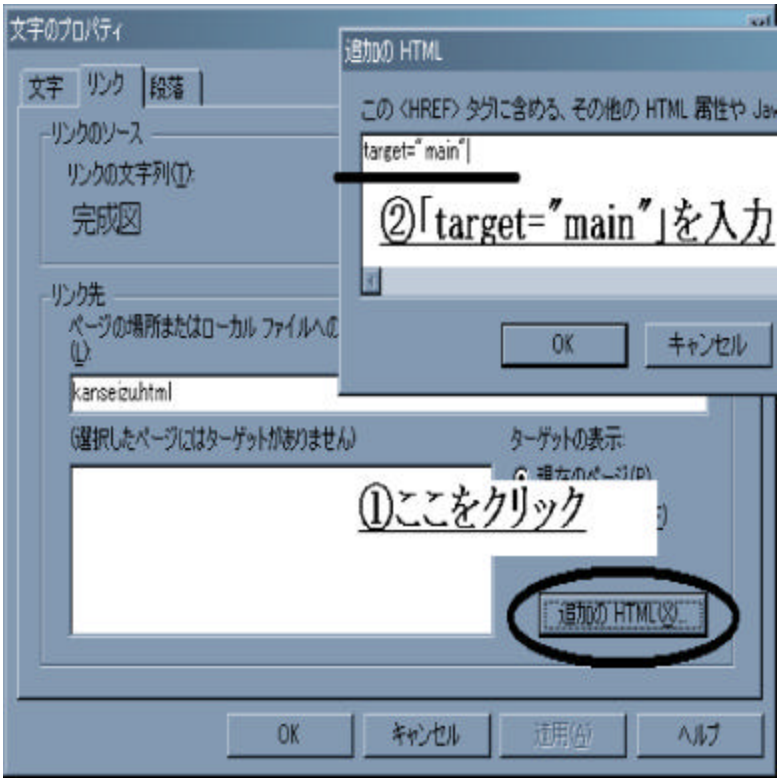


図 11.5: ターゲットの指定方法 (Composer)

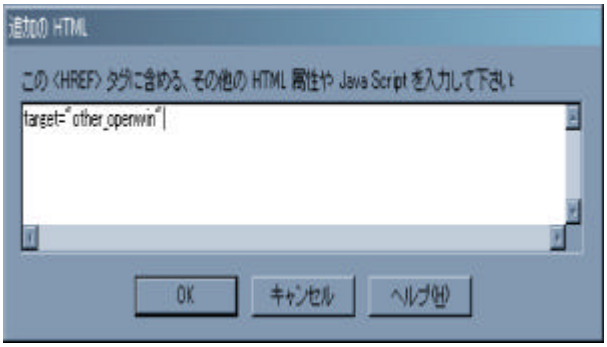


図 11.6: フレーム外の名前を指定する

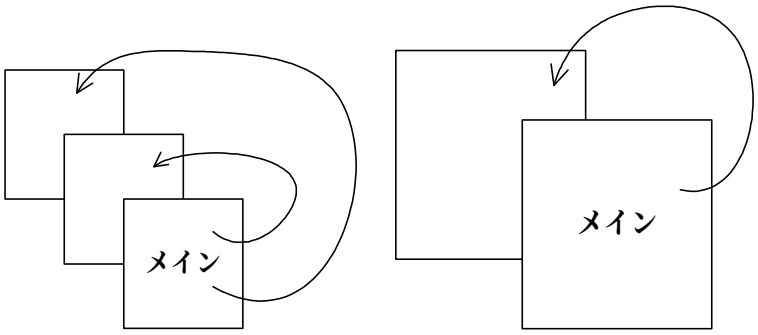


図 11.7: 別 Window への表示方法 (左... 別名のターゲットへ, 右... 同じターゲットへ)

第12章　Cascading Style Sheet(CSS)の活用

HTMLはWebページを装飾する役割も担ってきました。各種タグのプロパティが追加されてきたのもそのためです。

しかし，元々HTMLはWebページの構成を示すものであって，デザインのために発明されたものではありませんでした。

とはいえ，デザインも重要です。そこで装飾のためにCSS(Cascading Style Sheet)と呼ばれる機構が取り入れられました。これを使うと従来のタグに頼った手法では不可能な細かい装飾が可能になります。ここではその機能の一端に触れてみることにします。

12.1　CSSとは?

もともとHTMLは文書の構造を記述するものとして発明されました。その後，特にセルごとに色づけは背景画像を指定できるテーブルの機能をデザインを表現する手段として多用されるようになり，タグのプロパティの種類もデザインのための手段を提供するために増えてきました。しかし，その反面，タグの原点である「文書構造」を表わす役割との切り分けが難しくなってきました。

そこで，もっと細かいレベルの調整ができ，なおかつタグのプロパティには頼らない仕組みが導入されました。それがスタイルシート(Style Sheet)です。タグや文書全体の表示形式を細かく指定でき，複数のスタイルシートを入れ子(Cascading)にして使用できるため，CSS(Cascading Style Sheet)と呼ばれます。

スタイルシートの記述は，HTMLのタグと同様，テキストの形式で指定します。その方法は3つあります。

- HTMLファイルに記述されたタグそれぞれに指定するスタイルを埋め込む
- HTMLファイルのヘッダに，`<style type="text/css">...</style>`で囲んだ部分へ埋め込む
- スタイルシートを記述したCSSファイルを作成し，ヘッダで`<link>`タグを用いてそのCSSファイルを読み込む

最初に挙げた，タグごとにスタイルを埋め込む形式は，よほどのことが無い限り使用しない方が良いでしょう。後々再利用することを考えるのであれば，ヘッダに記述する形式が望ましく，できれば別ファイルにしておくことが最良ではないでしょうか。

2番目のヘッダにスタイルを埋め込む形式を次の節で，その後で別のCSSファイルを利用する例を見ていくことにしましょう。

12.2　長文を読みやすくする

ここでは次の文章を読みやすく見せるための Web ページを作ってみましょう。

> 日本国民は、正当に選挙された国会における代表者を通じて行動し、われらとわれらの子孫のために、諸国民との協和による成果と、わが国全土にわたつて自由のもたらす恵沢を確保し、政府の行為によつて再び戦争の惨禍が起ることのないやうにすることを決意し、ここに主権が国民に存することを宣言し、この憲法を確定する。そもそも国政は、国民の厳粛な信託によるものであつて、その権威は国民に由来し、その権力は国民の代表者がこれを行使し、その福利は国民がこれを享受する。これは人類普遍の原理であり、この憲法は、かかる原理に基くものである。われらは、これに反する一切の憲法、法令及び詔勅を排除する。
>
> 日本国民は、恒久の平和を念願し、人間相互の関係を支配する崇高な理想を深く自覚するのであつて、平和を愛する諸国民の公正と信義に信頼して、われらの安全と生存を保持しようと決意した。われらは、平和を維持し、専制と隷従、圧迫と偏狭を地上から永遠に除去しようと努めてゐる国際社会において、名誉ある地位を占めたいと思ふ。われらは、全世界の国民が、ひとしく恐怖と欠乏から免かれ、平和のうちに生存する権利を有することを確認する。
>
> われらは、いづれの国家も、自国のことのみに専念して他国を無視してはならないのであつて、政治道徳の法則は、普遍的なものであり、この法則に従ふことは、自国の主権を維持し、他国と対等関係に立たうとする各国の責務であると信ずる。
>
> 日本国民は、国家の名誉にかけ、全力をあげてこの崇高な理想と目的を達成することを誓ふ。
>
> (日本国憲法前文 [13])

これを普通の Web に流し込んでみると図 12.1 のようになります。長文だとぎっちり文章が詰まってしまい，かなり読みづらくなります。

これを修正するために CSS の機能を使ってみます。次のタグを HTML ファイルのヘッダ部分に挿入してみましょう。

```
<style type="text/css">
        H1 {
                color: blue;
                background-color: red;
        }
        DIV.kenpo {
                margin-left: 10%;
                margin-right: 10%;
                line-height: 1.5em;
        }
</style>
```

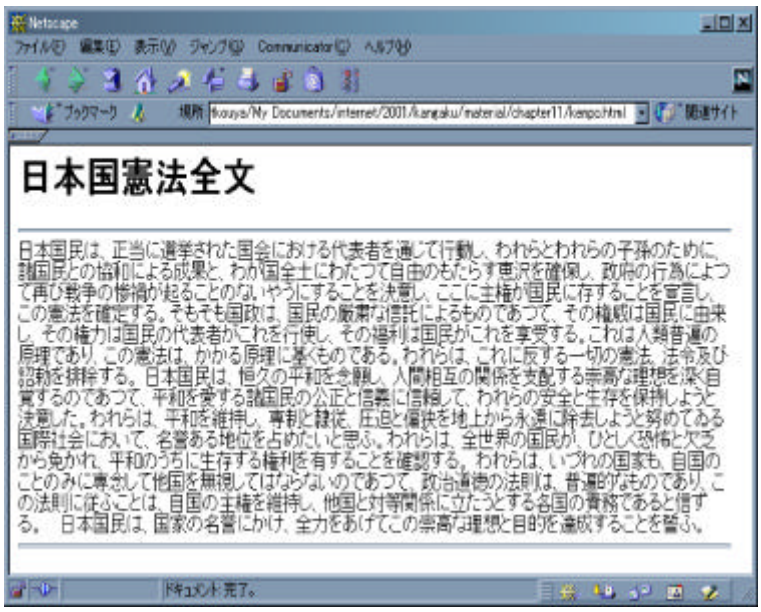


図 12.1: 日本国憲法全文 (CSS 指定なし)

ご覧のようにそれぞれのタグの装飾を決めています。 タグの指定は のようにクラスを使っていますので，この様式を使いたいときにはこのように書かなければなりません。

以上の変更を加え，長文を でくくると図 12.2 のように，左右マージンが入り，行間が広く開き，読みやすくなります。

このように，タグだけでは実現できない細かいデザインの指定が出来るようになっています。

12.3 CSS ファイルを作成する

もう少し大きなスタイル指定を行うには，別ファイルにして読み込むようにします。例えば図 12.3 のようなスタイルシートをテキストファイルで作成し，“book.css”という名前で保存しておくとします。スタイルシートは必ず “css”という拡張子で作成して下さい。この例では，H1〜H5 タグ，PRE，BLOCKQUOTE，A タグのスタイルを規定しています。英語が判別できればどのような指定をしているか，ある程度類推出来ますよね？

このスタイルファイルを使うには，HTML ファイルのヘッダ部分に

```
<head>
   ...
   <link rel="stylesheet" type="text/css" href="book.css">
   ...
</head>
```

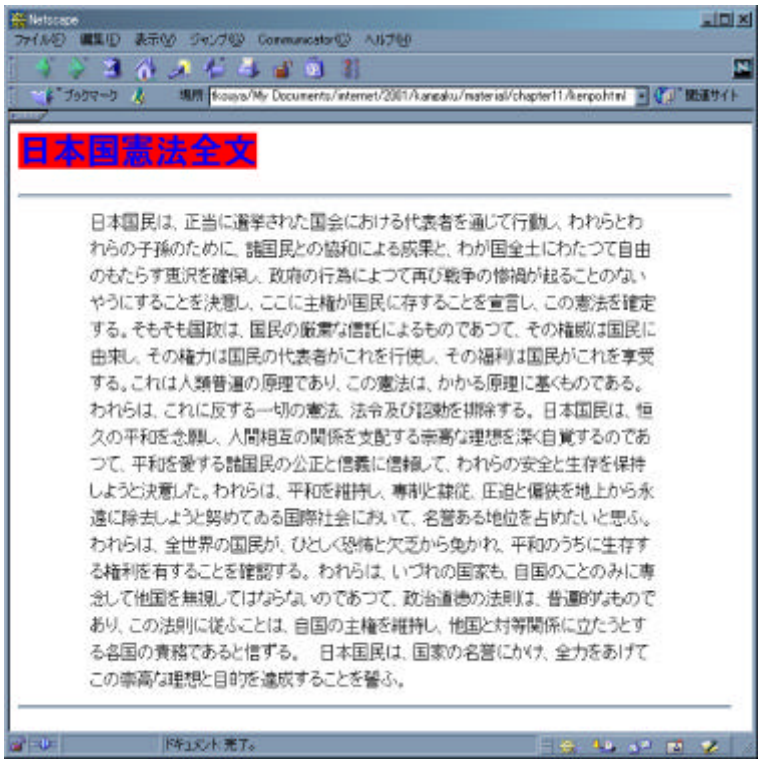


図 12.2: 日本国憲法 (CSS で装飾指定あり)

```
/* スタイルシートの例 */
/* 99999999 幸谷智紀  */

/* H1 タグのスタイル */
H1 {
color: blue;
/* background-color: red; */
border: 3px dashed green;
width: 70%;
}

/* H2〜H5 タグのスタイル */
H2, H3, H4, H5 {
border: double;
border-left: 10pt solid red;
width: 50%;
}

/* PRE, BLOCKQUOTE タグのスタイル */
PRE, BLOCKQUOTE {
font-size: 12pt;
margin-left: 10%;
margin-right: 10%;
line-height: 1.5em;
}

/* リンクのスタイル */
A:link{
background: #00ff0f;
}

/* 一度表示したリンクのスタイル */
A:visited{
background: #ffff0f;
}
```

図 12.3: book.css

と記述します。この場合は，この HTML ファイルと book.css ファイルが同じディレクトリ (フォルダ) に存在していると仮定しています。

このスタイルシートを使うと，図 12.4 のような表示が可能です。

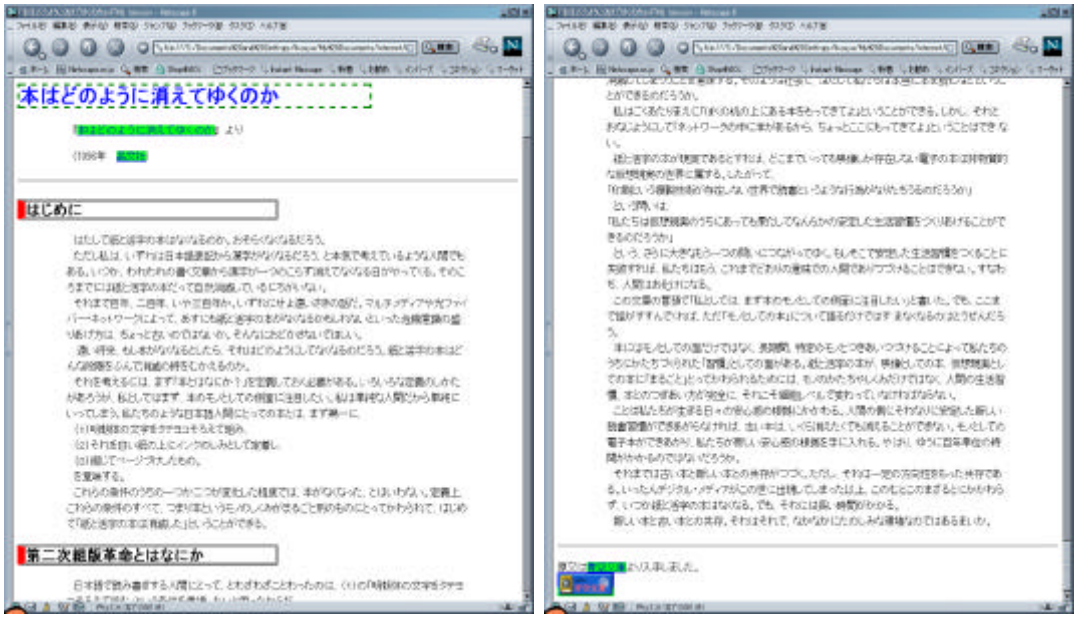


図 12.4: book.css を使用した Web ページの例

タグのプロパティだけでは制御できない細かい表示設定が，スタイルシートを使うことで可能になります。是非，使いこなせるようになって下さい。

練習問題

1. 図 12.3 に示したスタイルシートは，ブラウザのバージョンや種類によって表示のされ方が異なる。Internet Explorer ではどのように表示されるか，また Netscape Navigator でも Ver. 4.x 系と 6.x 系ではどのように異なるかを調べよ。

総合課題2

青空文庫 (http://www.aozora.gr.jp/，図 12.5) から，自分の好きな文学作品を選び，読みやすく加工して (但し，文章そのものに変更を加えてはならない)，今まで使用してきたテクニック全てを駆使して Web ページ上に提示せよ。あまり長いものを選ばず，短編 (例えばポーの「黒猫」など) を使うと良い。

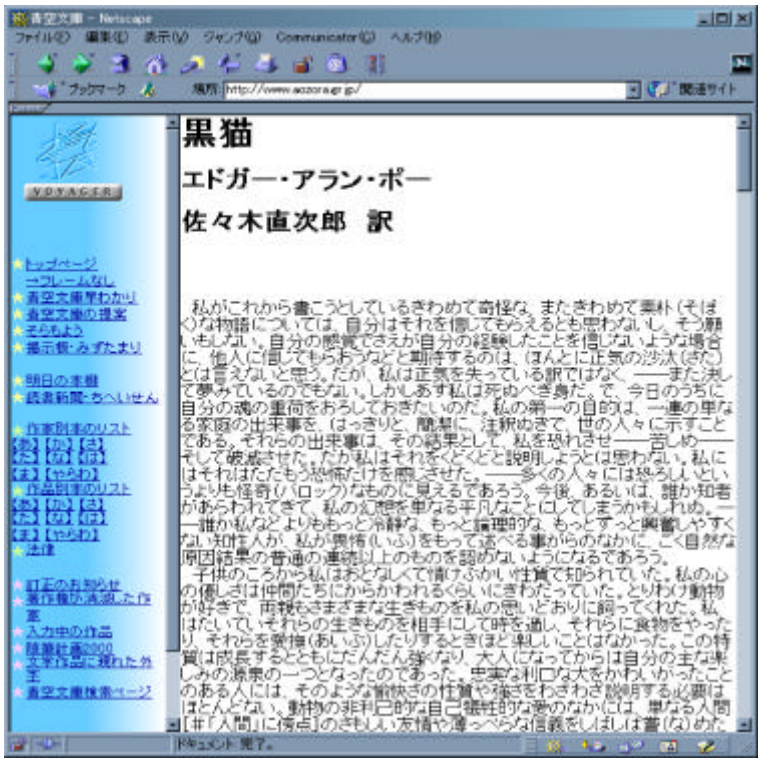


図 12.5: 青空文庫の「黒猫」

— メモ —

第13章　静的なWebページと動的なWebページ

今までの章で作ってきたのは，HTML で記述された Web ページでした。これは編集されない限り，そのままの状態で存在しつづけます。Web ページとしても表示された状態そのままですので，アニメーション GIF でも貼り付けない限り，変化することはありません。このような Web ページを静的な Web ページ (Static Web page) と呼びます。

それに対し，ユーザがその Web ページを閲覧した時刻を掲示したり (第 15 章)，Web ページ内に雪を降らせたり (第 16 章)，その Web ページの参照数を表示したり (第 17 章)，ユーザの入力を受け付けるアンケートを実行したり (第 18 章) と，場合に応じて変化するものを動的な Web ページ (Dynamic Web page) と呼びます。

本章では動的な Web ページについて，ごく大雑把な説明を試みます。

13.1　Client Side と Server Side

状況に応じて変化する Web ページは，何らかの形で Web サーバかクライアント (ブラウザ) がその仕事を引き受けることになります。サーバ側でその仕事がなされる場合を Server Side の処理，ブラウザ側で行われる場合を Client Side の処理と呼びます。

Server Side の概念図を図 13.1 に示します。

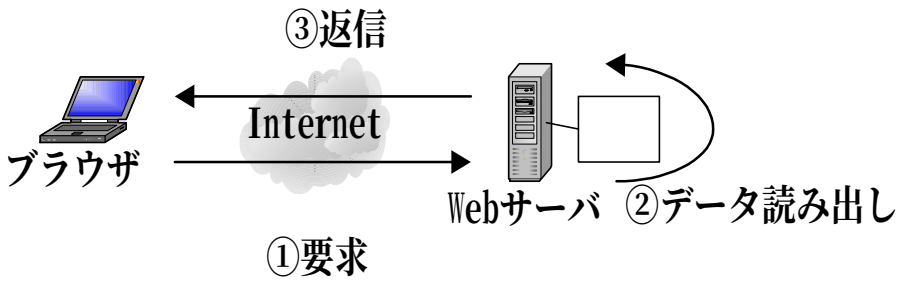


図 13.1: Server Side の概念図

例として，来訪者カウンタの場合を考えてみましょう (図 13.2)。これは，表示されている Web ページが今までに何回呼び出されたかを表示するものです。

前述のように，Web サーバは Internet を通じて不特定多数のユーザからの要求に 24 時間答え続

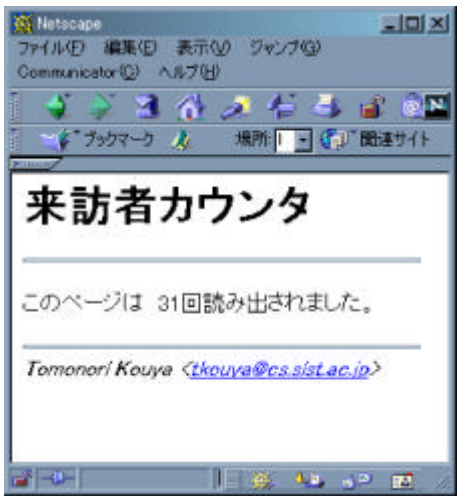


図 13.2: 来訪者カウンタの例

けています。そのため，Web ページの呼び出し回数はサーバ側で記録しておく必要があります。来訪者カウンタは，この Web ページが呼び出される度に呼び出し回数を一つ増やし，再び Web サーバのどこかのファイルに回数を記録する，という仕事を繰り返します。

では Cliend Side の場合 (図 13.3) はどうなるでしょうか。今まで見てきたように，Web ページの元となる HTML ファイルを綺麗に整形するという仕事や，アニメーション GIF の処理も Client Side で行われます。

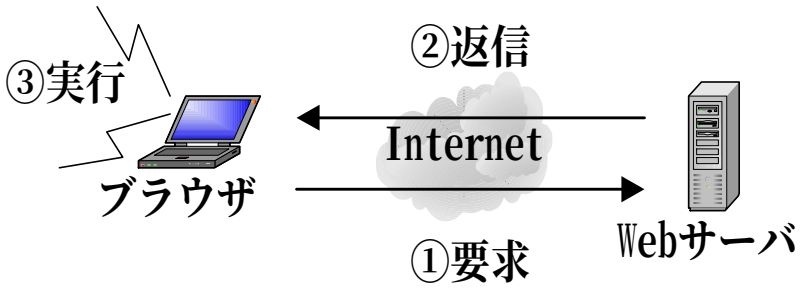


図 13.3: Client Side の概念図

それ以外にも，次のようなものが Client Side で行われます。

- (Client Side の) JavaScript の実行
- (Client Side の) VB Script の実行
- 各種フォームの処理
- 動画データや音声データの再生 (Plug-In 等)

これらは全て，Web サーバから送られてくる HTML ファイルのデータに埋め込まれています。しかし PHP のように，HTML ファイルに埋め込まれているスクリプト (プログラムの一種) であっても，実際の処理は Server Side で行われるものも存在します。

Server Side と Client Side の長所と短所をまとめて置きましょう。

長所

Server Side の処理 … データの集中管理が可能で，アンケートやオンラインショッピングなど，不特定多数のユーザに対するサービスの集計結果が簡単に手に入る。

Client Side の処理 … 動的な処理はユーザの手元にあるブラウザが実行するため，サーバの負担が少なくて済む。

短所

Server Side の処理 … 只でさえ負荷の掛かりやすいサーバに更なる負担を強いる。また，動的な処理を行うためのプログラムはサーバ本体で動作することから，不用意に設定するとセキュリティホールを招きやすい。

Client Side の処理 … クライアントに負担を掛け，各種 Script はセキュリティホールになることがある。

練習問題

1. (自由課題) 動的な Web ページを探し，そこで行われている処理が Server Side のものか，Client Side のものかを判別せよ。また，その判断の根拠を説明せよ。

— メモ —

第14章　フォームで遊ぼう

フォームとは，Web ページ上で，ユーザからの入力を受け付けることあが出来る部品を提供するためのものです。部品は全て HTML のタグで表現されます。ここで入力されたデータを後で述べるように Web サーバ側で受け取れるような仕組みを組み合わせることで，例えば自動的に入力データを収集するオンラインショッピングの Web ページや，アンケート調査のための Web ページが動作するようになります。

14.1　アンケートのページ

まず，簡単なアンケートページを書いてみましょう (図 14.1)。

図 14.1: アンケートのページ

このフォームには次の要素を使っています。

ラジオボタン (性別) ... どちらか一方だけ選択するために利用します。

選択 (生年月) ... あらかじめ用意しておいた選択肢から選びます。

テキスト (学籍番号・氏名) ... 一行で済むような短いテキストを入力します。

チェックボックス (OS) ... 複数のチェックをすることが出来ます。

HTML は以下のように記述されています。このうち <FORM> タグで囲まれた部分がフォームになります。この例ではフォーム内に箇条書きの <OL> タグを使用しています。このように，フォーム中に，直接フォームの要素とは関係の無いタグを挿入することも可能です。

```
<HTML>
<HEAD><TITLE>Enquate</TITLE></HEAD>
<BODY>
<H1>アンケート</H1>
<HR>
<FORM action="/cgi-bin/get-answer.cgi" method="post">
<OL>
<LI> 貴方の性別は？<BR>
男<INPUT name="sex" type="radio"> 女<INPUT name="sex" type="radio">

<LI> 貴方の生年月は？<BR>
<SELECT name="birth_year">
<OPTION selected label="" value="none"></OPTION>
<OPTION>1974</OPTION>
<OPTION>1975</OPTION>
<OPTION>1976</OPTION>
<OPTION>1977</OPTION>
<OPTION>1978</OPTION>
<OPTION>1979</OPTION>
<OPTION>1980</OPTION>
<OPTION>1981</OPTION>
<OPTION>1982</OPTION>
<OPTION>1983</OPTION>
<OPTION>1984</OPTION>
<OPTION>1985</OPTION>
</SELECT>年
<SELECT name="birth_month">
<OPTION selected label="" value="none"></OPTION>
<OPTION>1</OPTION>
<OPTION>2</OPTION>
<OPTION>3</OPTION>
```

```
<OPTION>4</OPTION>
<OPTION>5</OPTION>
<OPTION>6</OPTION>
<OPTION>7</OPTION>
<OPTION>8</OPTION>
<OPTION>9</OPTION>
<OPTION>10</OPTION>
<OPTION>11</OPTION>
<OPTION>12</OPTION>
</SELECT>月

<LI>貴方の学籍番号は？<BR>
<INPUT name="number" type="text" width="20">
<LI>貴方の氏名をローマ字で記入して下さい。<BR>
氏：<INPUT name="first_name" type="text" width="30">
名：<INPUT name="last_name" type="text" width="30">

<LI>貴方が今までに使ったことのあるパソコンの OS を教えて下さい。<BR>
<INPUT name="os" type="checkbox" checked>Windows 2000
<INPUT name="os" type="checkbox">Windows 95/98/ME
<INPUT name="os" type="checkbox">Linux

</OL>
<CENTER><INPUT type="submit" value="登録"> <INPUT type="reset" value="消去"></CENTER>
</FORM>
<HR>
<ADDRESS>Tomonori Kouya: &lt;<A HREF="mailto:tkouya@cs.sist.ac.jp">tkou
ya@cs.sist.ac.jp</A>&gt;</ADDRESS>
</BODY>
</HTML>
```

14.2 Yahoo! Japan を貼り付ける

ここでは実際にフォームと Web サーバで動作しているプログラムとの連携している例を見ることにしましょう。例えば Yahoo! Japan のようなサーチエンジンの多くは宣伝のためということもあるのでしょうが，利用者の Web ページに検索フォームを貼り付けることを許可しています (図 16.3)。

この Web ページのフォームの部分を抜き出すと次のようになります。

```
<FORM METHOD="GET" ACTION="http://search.yahoo.co.jp/bin/search">
```

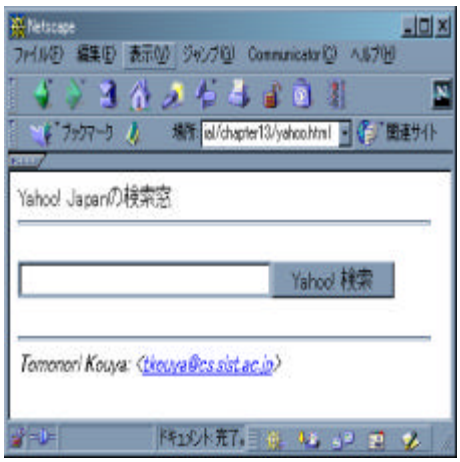


図 14.2: Yahoo! Japan の検索窓

```
<INPUT SIZE="30" NAME="p">  <INPUT TYPE=submit VALUE="Yahoo! 検索">
</FORM>
```

実際にフォームから入力された検索キーワードは“GET”という手段で”http://search.yahoo.co.jp/bin/search”で指定されている Web サーバで動作しているプログラムに渡されています。このフォームで例えば「関東学院大学」というキーワードを与えて「Yahoo!検索」ボタンを押すと，Yahoo! Japan の Web ページに飛び，検索結果が表示されます。そのときの URI は

```
http://search.yahoo.co.jp/bin/search?p=%8A%D6%93%8C%8Aw%89%40%91%E5%8Aw
```

となっています。”p=”の後はこの「関東学院大学」という文字列そのものを表わしています[1]。従って，この URI を再度指定すると同じ様に「関東学院大学」の検索結果を表示します。このように GET メソッドでは URI に入力データが反映されます。

GET メソッドとは別にもう一つ POST メソッドというものもあります。これは URI そのものに入力データは反映されません。が，Web サーバに入力データを与えていることは GET メソッドと同じです。

練習問題

1. アンケートのページをテーブルで囲み，デザインせよ。
2. 日本語で検索できるサーチエンジンの検索窓を集めた Web ページを作れ。検索窓の作り方は各サーチエンジンの Web ページからリンクの可否も含めて記述してあるのでそれを参照すること。

[1] ブラウザが使用している文字コードが異なるとこの値も異なる。

図 14.3: 主なサーチエンジンの検索窓を集めたページ

― メモ ―

コラム★ログって何？

ログ (log) という言葉を聞いたことはないでしょうか？ Internet の世界では，サーバと呼ばれるマシン (Web サーバ，FTP サーバ, SMTP(メール) サーバ ···) では必ず保存されている，一種のアクセス記録表を意味します。一度はどこかで耳にされたことがあるかもしれませんが，ログの実物を見たことはあまりないでしょう。ここではその一部をご覧に入れます。

アクセスの記録を保持する

以下のログは”cs-xxxx”というホストで，どういうプログラムが起動したかという記録です。

```
Feb  7 13:09:00 cs-xxxx in.pop3d[7297]: Servicing fml @ 192.168.1.11
Feb  7 13:15:24 cs-xxxx in.telnetd[7318]: connect from 192.168.1.11
Feb  7 13:15:35 cs-xxxx xntpd[183]: time reset (step) 0.227422 s
```

1 行ごとに「起動日時 (月/日/時刻)」「起動したホスト (この場合は全て cs-xxxx)」,「起動したプログラム名」,「コメント」という順で並んでいます。詳しく見ていくと次のようになります。

1 行目 2 月 7 日 13:09:00 に，192.168.8.11 からの要求を受けて POP3 デーモン (メール受信) が起動

2 行目 2 月 7 日 13:15:24 に，192.168.1.11 からの要求を受けて telnet デーモン (telnet) が起動

3 行目 2 月 7 日 13:15:35 に，xntpd(時刻同期用デーモン) が起動

このような記録を取ることは，サーバ管理者の義務です。勿論，管理業務以外の用途でこれらのログを他人に見せることは，アクセスした人のプライバシーを暴露することにもなりかねず，絶対にあってはならないことです。従って，自分のアクセス記録が誰かに漏れる危険は少ないといえます。しかし，利用者といえども自分のアクセス記録が常に取られているということは知っておく必要があるでしょう。一般の商用 ISP でも必ずこのようなログを記録し，一定期間保存している筈です。もし取っていなければ問題になります。不正なアクセスがあっても，その証拠を残すことが出来ず，取り締まることも不可能になるからです。

14.3 Web サーバのログ

当然のことながら，Web サーバでもログは常に記録されています。例えば，以下のようなものになります。

```
xxx.xx.xxx.xx - - [02/Mar/2001:17:04:08 +0900] "GET /seminar/index.html HTTP/1.1" 200 1035
xxx.xx.xxx.xx - - [02/Mar/2001:17:04:08 +0900] "GET /gif/smallbrick.gif HTTP/1.1" 200 1012
xxx.xx.xxx.xx - - [02/Mar/2001:17:04:08 +0900] "GET /gif/smallbrickr.gif HTTP/1.1" 200 1012
xxx.xx.xxx.xx - - [02/Mar/2001:17:09:53 +0900] "GET /mupad/index.html HTTP/1.1" 200 1907
xxx.xx.xxx.xx - - [02/Mar/2001:17:09:53 +0900] "GET /mupad/smallbrick.gif HTTP/1.1" 200 1004
xxx.xx.xxx.xx - - [02/Mar/2001:17:09:53 +0900] "GET /mupad/smallbrickr.gif HTTP/1.1" 200 1012
xxx.xx.xxx.xx - - [02/Mar/2001:17:10:00 +0900] "GET /about_tkouya.html HTTP/1.1" 200 1832
xxx.xx.xxx.xx - - [02/Mar/2001:17:10:38 +0900] "GET /index.html HTTP/1.1" 200 3070
```

この場合は，xxx.xx.xxx.xx(IP アドレス) からこのサーバにある/seminar/index.html 等のファイルへアクセスが正常に行われています。

しかし，通常のアクセスとは全く異なるものが記録されていることもあります。

```
xxx.xx.xxx.x - - [07/Feb/2002:13:15:20 +0900] "GET /scripts/..%252f../winnt/syst
em32/cmd.exe?/c+dir HTTP/1.0" 404 308
```

この例では，Web ページからは何のリンクも張っていない，存在していないパスへのアクセスが記録されていました。2001 年は，かなり悪質なウィルスが蔓延り，これは特定 OS および Web サーバへも感染するというものでした。Web サーバに感染したウィルスは次の感染先を求めて，大量のアクセスを別の Web サーバへ行います。ここで記録されていたアクセスはその一つでした。幸いこの Web サーバは感染の可能性のある OS や Web サーバプログラムを使用しておらず，ウィルスに侵されることはありませんでした。

もしウィルスに犯されても，管理人が怠慢こいていると，感染に気付かずそのまま放置され，さらに被害者を増やす結果に繋がります。常にこのようなログチェックを怠らず，マメにセキュリティホールを塞ぐ等の対策を取っていれば，たとえ感染してもすぐに気が付いて適切な対応を取ることが出来ます。

このように，ログの管理はセキュリティの面からも必須のものなのです。自分の記録が取られているという知識を持つと共に，その有効性も認識しておいて下さい。

第15章　JavaScriptを活用しよう

本章ではJavaScriptの簡単な例を試してみます。JavaScriptそのものはかなり複雑なオブジェクト指向の言語です。数多くのオブジェクト[1]とそれに付随したメソッドを持っており，役に立ちそうな所だけ抜粋してもかなりの分量になります。ここでは小さな例題を示し，その雰囲気を掴み取って貰えば十分です。

15.1　one linerなJavaScript

最近良く見かけるのは，Webページ内のボタンをクリックすると，ツールバーもステータスバーもないのっぺら坊のウィンドウが開き，そこに別のWebページが表示されるというものです。

この機能を抜粋した例を図15.1に示します。このHTMLファイルを作成してブラウザで表示すると，フォーム内に「ウィンドウを開く」というボタンが出現します。これをクリックすると図15.3のopen_window2.htmlが別ウィンドウに表示されます(図15.3)。

```
<html>
<title>別 Window を開く</title>
<body>
<form>
<input type="button" value="ウィンドウを開く" onclick="window.open('open_win
dow2.html', 'win2', 'scrollbars=no,toolbar=no,width=200,height=50');">
</form>
</body>
</html>
```

図15.1: 別ウィンドウを表示する

このopen_window.htmlは図15.4のように書くこともできます。このように

```
<script language="javascript">
    ...
</script>
```

に囲まれた部分や，”onclick=”以降に記述された部分でJavaScriptを記述しています。

[1]変数に関数(メソッド)が融合したようなもの。

```
<html>
<title>別 Window</title>
<body>
別 Window 画面です。
</body>
</html>
```

図 15.2: open_window2.html

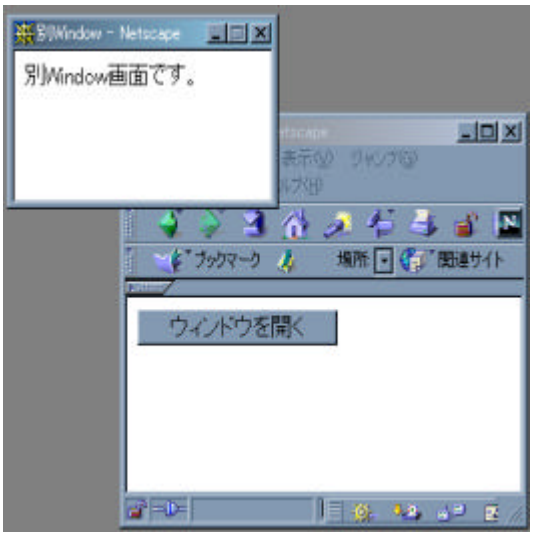


図 15.3: 別 Window を表示

```
<html>
<title>別 Window を開く</title>
<script language="javascript">
function openwin(uri)
{
window.open(uri, "win2", "scrollbars=no,toolbar=no,width=200,height=50");
}
</script>
<body>
<form>
<input type="button" value="ウィンドウを開く" onclick="openwin('open_window2.h
tml')">
</form>
</body>
</html>
```

図 15.4: 別 Window を表示する (簡易版)

15.2 JavaScript を書いてみよう

もうすこし，長いプログラム (スクリプト) らしいものを書いてみましょう。まずは時刻を表示する JavaScript です (図 15.5)。

これは HTML ファイルがブラウザに読み込まれる時に一度だけ”document.write(gettime())”(現在時刻を取得し，Web ページ内に表示する) を実行してして終了する JavaScript です。時刻を変化させるためには再読み込みを行う必要があります (図 15.6)。

そこで，自動的に時刻を更新させるためにタイマ (Timer) を使います。

図 15.5 中の

```
document.write(gettime());
```

という部分を

```
window.setInterval("window.status = gettime()", 1000);
```

と書き換えて下さい。こうすると，この場合は 1 秒ごとに指定の関数 (gettime()) を繰り返し実行します。すると，図 15.7 のように，ステータスバー (ウィンドウ下部) に時刻が表示され，刻々と変化していくのが確認できます。

今度は現在時刻を Weg ページ内に表示させてみましょう。そのためにフォームを利用します (図 15.8)。どこを変更したか分かりますか？

実行すると図 15.9 のように，テキストボックスに時刻が表示されます。

```
<html>
<title>時計</title>
<script language="javascript">
function gettime()
{
        time = new Date();
        display_time = "";
        display_time += String(time.getFullYear()) + "年";
        display_time += time.getHours() + "時";
        display_time += time.getMinutes() + "分";
        display_time += time.getSeconds() + "秒";
        return display_time;
}

document.write(gettime());

</script>
<body>
</body>
</html>
```

図 15.5: 日付を表示する

図 15.6: リロードすると時刻が変わる

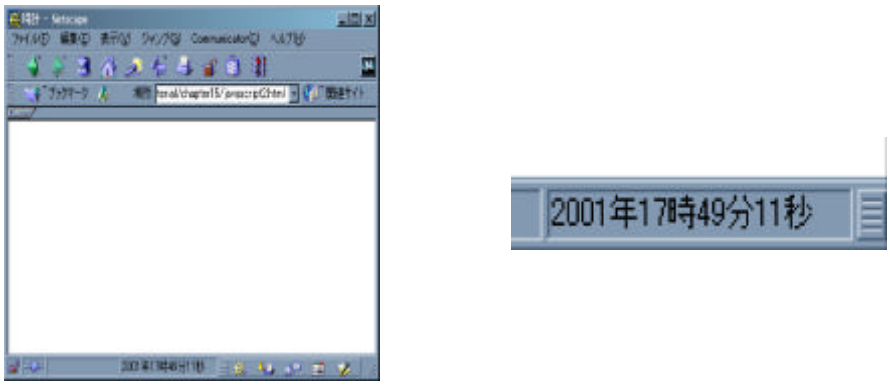


図 15.7: ステータスバーに時計を表示

```
<html>
<title>時計</title>
<script language="javascript">
function gettime()
{
time = new Date();
display_time = "";
display_time += String(time.getFullYear()) + "年";
display_time += time.getHours() + "時";
display_time += time.getMinutes() + "分";
display_time += time.getSeconds() + "秒";
return display_time;
}

window.setInterval("document.forms[0].time.value = gettime()", 1000);

</script>
<body>
<form>
<input type="text" name="time" maxlength=18>
</form>
</body>
</html>
```

図 15.8: 日付を表示する

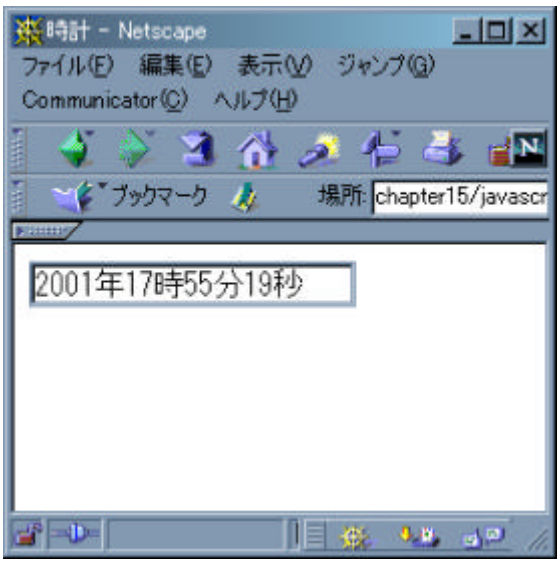


図 15.9: フォームに時計を表示

15.3　フォーカスを使ってボタンを変える

最近ではマウスカーソルをアンカー (リンクが張られた場所) に移動すると，その画像や文字列が変化するような Web ページがあります。そこでも JavaScript が活躍しています。

その原理を図 15.10 で確認してみて下さい。これは図 15.11 のように動作します。

<A> タグ内の”onmouseover=...”では，マウスカーソルがその位置に移動してきたときに実行される関数を指定しています。ここでその時々に合わせて表示される図を変化させれば良いわけです。このように，何か起こったときにその事 (イベントと呼びます) の対応した動作を行うことをイベントドリブン (event driven) と呼び，Web に限らず，GUI の基本となっています。

図 15.10 の例では，最初は “green.gif”(緑の正方形の画像) を表示し，マウスカーソルが来ると，“red.gif”(赤の正方形の画像) を表示します。マウスカーソルが外れると元の “green.gif”を表示しています。

練習問題

1. ボタンを押すと別 Window に時計を表示させる JavaScript を書け。実行画面は図 15.12 のようになる。

```
<html>
<script language="javascript">
var col = 1; // 1..green, 0..red
function change_color(imange_name)
{
        if(col == 1)
        {
                col = 0;
                document.image1.src = "red.gif";
        }
        else
        {
                col = 1;
                document.image1.src= "green.gif";
        }
}
</script>
<body>
<a href="http://www.kanto-gakuin.ac.jp/" onmouseover="change_color()" onmou
seout="change_color()">
<img name="image1" src="green.gif"></a>
</body>
</html>
```

図 15.10: フォーカスに応じて変化する画像

図 15.11: (左) フォーカスがない時　(右) フォーカスが移ってきた時

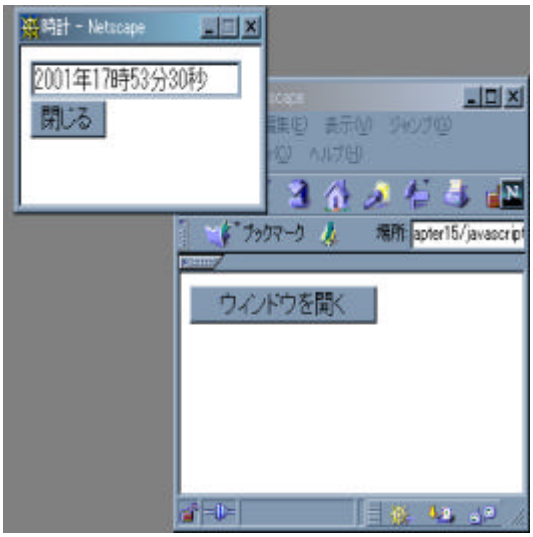


図 15.12: 別 Window に時計を表示

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。

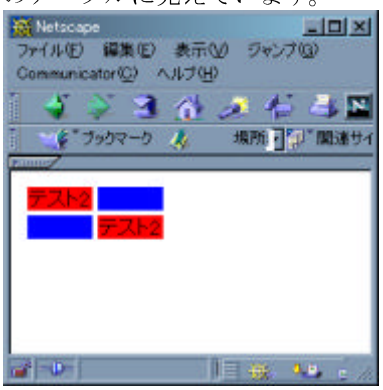
16.1　レイヤーという考え方

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。

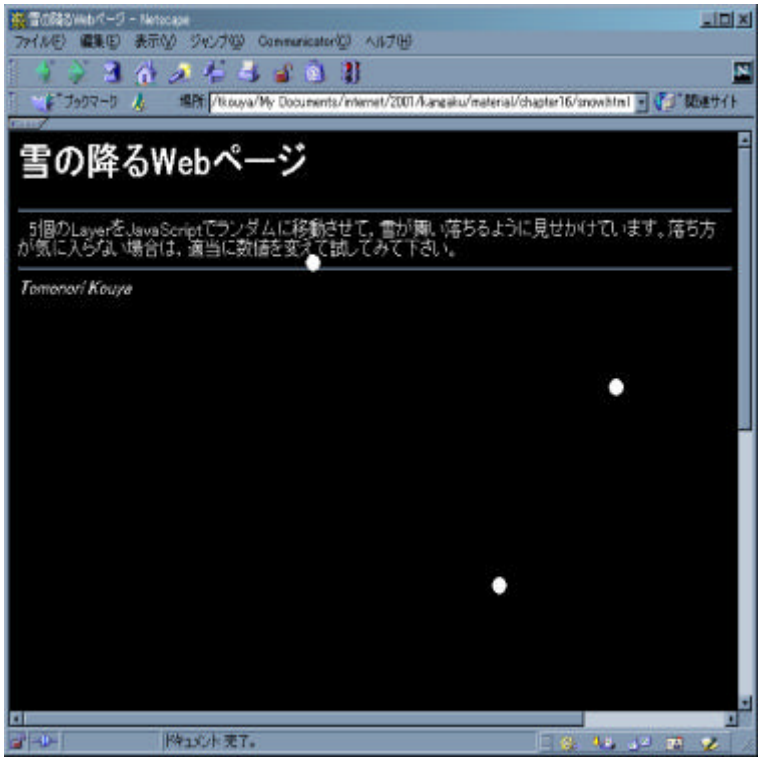


図 16.1: 雪の降る Web ページ

```
<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)。

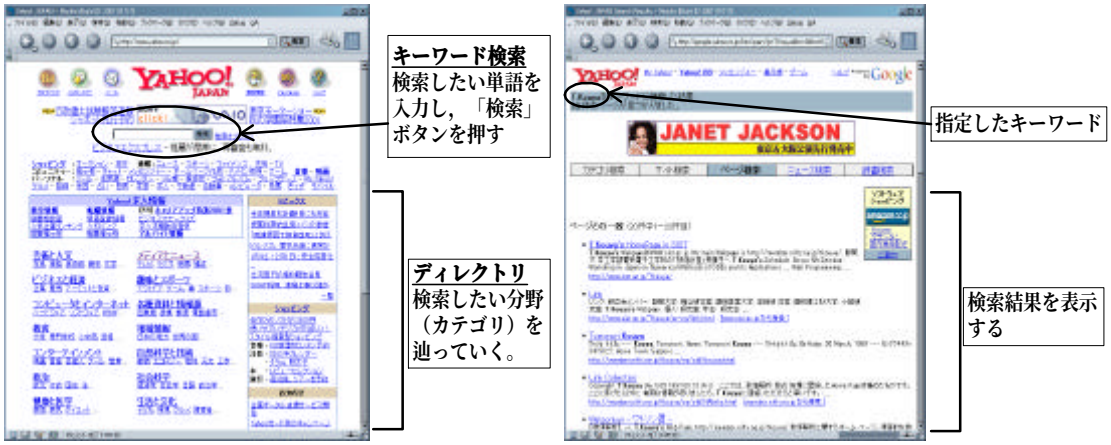


図 16.3: Yahoo! Japan の検索例 (左:検索前，右:検索後)

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。

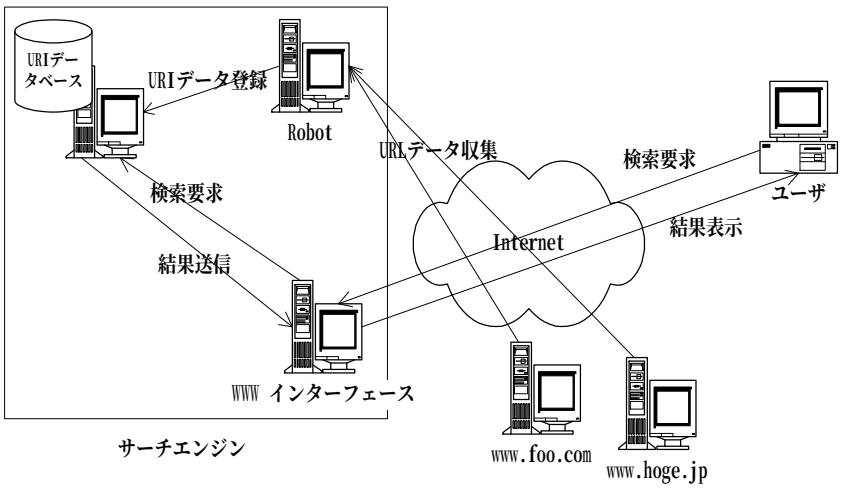


図 16.4: ロボットを利用したサーチエンジン

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。

― メモ ―

第17章　Perlを使ってみよう(1/2) — 初めてのSSIとCGI

ここではPerlスクリプトによる簡単なCGIやSSIを動作させてみます。前の章で説明したように，SSIもCGIもServer Sideで実行されるプログラムです。従って，Webサーバに実行の負担が掛かるため，全面的に禁止していたり，制限をかけている場合が多く見られます。本書で解説するSSIやCGIの例を実行させることが出来るかどうか，また実行する際にはどのような手続きが必要か，ということについてはWebサーバの管理者に必ず確認して下さい。

17.1　初めてのSSI

SSIはServer Side Includeの略称で，HTMLファイルに埋め込まれた命令をWebサーバ側で解釈し，実行するものです。Webサーバの設定によっては，拡張子が".shtml"というファイルにのみ，SSIの解釈が行われることがあります。以降は".html"ファイルも".shtml"もHTMLファイルと呼ぶことにします。

例えばHTMLファイルの適当な部分に

```
<!--#config timefmt="%Y年%m月%d日 %H時%M分" -->
この文書の更新日時: <!--#echo var="LAST_MODIFIED" -->
```

と書けば，このタグ内がWebサーバ側で解釈され，Webページ内に

この文書の更新日時: 2001年07月03日 19時24分

と表示されます。

また，アクセスしてきたユーザが使用しているマシンのIPアドレスやブラウザが知りたければ，HTMLファイルに

```
あなたが使っているPCのIPアドレスは<!--#echo var="REMOTE_ADDR" -->です。<br>
あなたが使っているブラウザは<!--#echo var="HTTP_USER_AGENT" -->です。
```

と書いておきます。Webサーバからこのssiが埋め込まれた文書を呼び出すと，前の例と同様にWebサーバ側でこのタグが解釈されて

あなたが使っているPCのIPアドレスは133.88.120.87です。
あなた使っているっているブラウザはMozilla/4.75 [ja] (Windows NT 5.0; U)です。

と表示されます。この例では，133.88.120.87 という IP アドレスを持つ PC 上で，Windows 2000(Windows NT 5.0) から Netscape 4.75(Mozilla 4.75) を使用してこの Web サーバにアクセスしていることが分かります。

SSI や CGI は，Web サーバ側で実行されるため，Web サーバでしか取得できない情報をユーザ側にフィードバックすることが可能になります。本章の最後に SSI と CGI を使用した簡単なアクセスカウンタの例を示しますが，これも Server Side で実行しなければ実現できないものです。

SSI は他にも様々な環境変数や書式・命令が使用可能です。詳細については付録を参照して下さい。

17.2 初めての CGI

SSI が HTML ファイルに埋め込まれて使用されるのに対し，CGI はもっと複雑な処理を行うプログラムを実行する仕組みを提供します。

まず最初に，以下の HTML を吐き出す CGI を書いてみましょう。

```
<html>
<title>First CGI</title>
<body>
<h1>初めての CGI</h1>
</body>
</html>
```

17.2.1 Perl スクリプトを書く

HTML を吐き出す CGI を Perl スクリプト[1]で記述すると次のようなものになります。

```
#!/usr/bin/perl

print "Content-type: text/html\n\n";
print "<html><title>First CGI</title>\n";
print "<body>\n";
print "<h1>初めての CGI</h1>\n";
print "</body>\n";
print "</html>\n";
```

C 言語等のプログラム言語を少し囓ったことにある人なら大体想像が付くと思いますが，これは標準出力に

```
Content-type: text/html


<html>
```

[1]Perl というインタプリタが実行するプログラムの一種。

```
<title>First CGI</title>
<body>
<h1>初めての CGI</h1>
</body>
</html>
```

という文字列を出力しているだけのプログラムです。

17.2.2 CGI として Web サーバに転送する

次の手順を踏んで，CGI として動作させてみます。最初に Perl スクリプトを自分の PC で作成するわけですが，その時にはなるべく秀丸エディタのような高機能のテキストエディタを使い，改行コードを UNIX 形式である “LF”だけにして保存するようにして下さい。それができない時，例えばメモ帳でこのスクリプトを作成せざるを得ないような時には，FTP クライアントソフトウェアでこのスクリプトを転送する時には，必ず ASCII モードで転送するようにして下さい。そうすることで，Web サーバに転送する時に，改行コードを自動的に UNIX 形式に変換してくれます。もし改行コードが UNIX 形式になっていなければ，恐らく Web サーバ上でこのスクリプトはエラーを吐いてしまうでしょう。

1. Perl スクリプトを “hellow.cgi”という名前で作成します。
2. WS_FTP で “public_html/cgi-bin”ディレクトリを Web サーバ側に作成し，そこに “hellow.cgi” を転送します。
3. 転送したら，転送先の “hellow.cgi”を右クリックし，出てきたメニューから “chmod(UNIX)” を選択します。
4. “Owner”, “Group”, “Other”のそれぞれの “Execute”にチェックを入れ，“OK”ボタンを押して下さい。これで”hellow.cgi”は単独で動作するプログラムとして使用可能になりました。
5. ブラウザから “http://Web サーバ名 (or IP アドレス)/~自分のユーザ ID/cgi-bin/hellow.cgi”と URI 指定して，“hellow.cgi”を動作させてみて下さい。画面に「初めての CGI」と表示されていれば OK です。

17.2.3 エラーのときは …

“Internal Server Error...”等が表示され，うまく実行できないときは，まず画面に表示されている情報をきちんと読みとって下さい。大体次のケースが考えられます。

”Permission Denied ...”と表示される CGI が実行できるディレクトリかどうかチェックして下さい[2]。CGI の実行を許可するディレクトリには特別に”.htaccess”というファイル[3]を作り

[2]Web サーバによっては CGI の実行が全面的に許可されていないこともある。Web サーバの管理者に確認すること。
[3]近年は Apache という Web サーバプログラムが高いシェアを誇っている。この Apache が動作している環境ではこのように書けば良いが，それ以外の Web サーバを使っていることもあり，管理者に確認すること。

```
Options ExecCGI
```

などと書いておく必要がある場合もあります。

”Not Found　...”と表示される ちゃんと指定されたディレクトリに”hellow.cgi”を転送してありますか？

”Internal Server Error ...”と表示される Perl スクリプトが正しく動作できない状態にあります。その際には以下の項目を再度チェックして下さい。

- CGI が動作可能なディレクトリか？
- “hellow.cgi”のモードが正しく “Execute”になっているか？
- 第一行目の “#!/usr/bin/perl” の絶対パス指定が正しいか？
- 改行コードが UNIX 形式になっているか？
- そもそも Perl スクリプトを動作させるためのインタプリタが無かった，など他にも原因は考えられる。どうしてもわからないようであれば Web サーバ管理者に確認せよ

エラーの一度や二度出したからといってめげないで下さい。また，エラーが出たからといって，単に「うまく動きません」と質問されても訳が分かりません。最低限「このようなエラーが出ていますが，どうすればいいでしょうか?」と聞いて下さい。慣れてくれば自分でエラー内容が把握でき，そこから解決策を自分で導くことができるようになります。

17.3　SSI の応用 – 来訪者カウンタの仕組み

来訪者カウンタの仕組みはごく簡単です。ブラウザから Web ページに対してアクセスがあると

1. SSI が埋め込まれた Web ページが Web サーバによって呼び出される。
2. SSI が実行される。この際
   (a) 呼び出す前に，”counter.dat”をロック
   (b) 現在までの呼び出し回数を”counter.dat”から入力する。
   (c) ロック解除。
   (d) 書き出す前に再度ロック。
   (e) 呼出し後，回数を 1 増やし”counter.dat”に出力。
   (f) ロック解除。

   という手順を行う。
3. SSI が埋め込まれた個所に数字が表示される。
4. 数字が埋め込まれた HTML ファイルとしてブラウザに表示される。

「ロック」とは，他のプロセスから同じファイルに読み書きできないようにすることです。特に Web サーバは複数の接続に対応できるよう，複数のプロセスで同時に走っている事があり，このアクセスカウンタが働く Web ページが並行して読み込まれることも考慮する必要があります。このようにロックをかけておくことで，一つのアクセスが終了するまでは counter.dat が他のプロセスによって書き変えられることが防止できます。

17.3.1 作ってみよう

この手順を Perl スクリプトにしたのが図 17.1 です。

```
#!/usr/bin/perl

#
# 来訪者カウンタ
#

# 来訪者数を記録するファイル
$counter = "/home/tkouya/public_html/cgi-bin/counter.dat";

# データファイルを開いて読み出す。
open (FILE, $counter) || die "データファイルが読めません\n";
flock (FILE, 2);
$visitor = <FILE>;
flock (FILE, 8);
close (FILE);

# 来訪者数を標準出力に書き出す。
print "Content-type: text/plain\n\n";
print ++$visitor;

#データファイルを開いて書き出す。
open (FILE, ">" . $counter) || die "データファイルに書き込めません\n";
flock (FILE, 2);
print FILE $visitor;
flock (FILE, 8);
close (FILE);
```

図 17.1: 来訪者カウンタの Perl スクリプト

SSI を埋め込む HTML ファイルは図 17.2 のように作成し，タグを図 17.3 のように埋め込んでお

きます。

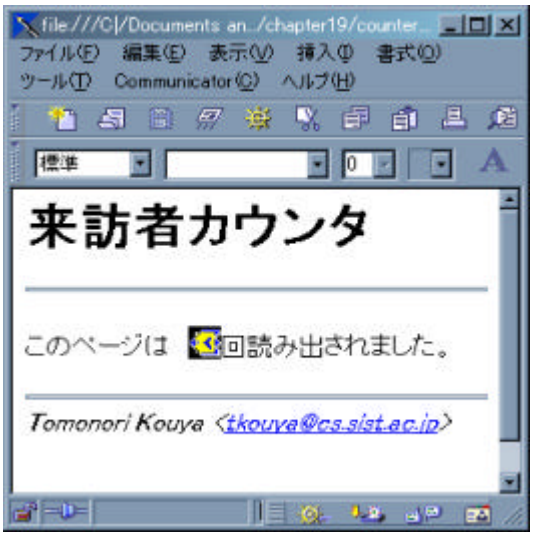


図 17.2: 来訪者カウンタ用 Web ページを作成

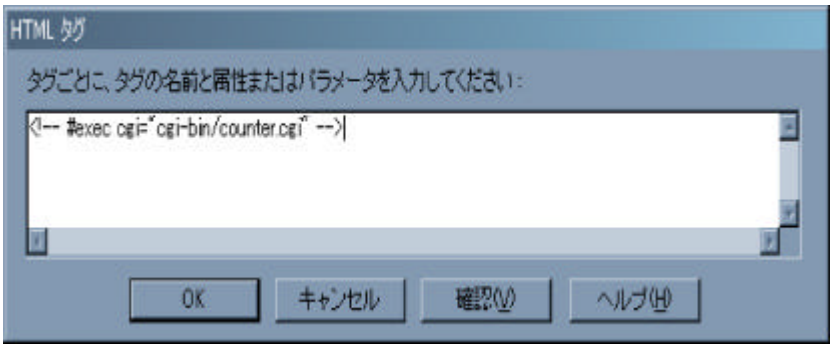


図 17.3: SSI のためのタグを挿入する

Web サーバの指定のディレクトリにこれらのファイルを転送し，ブラウザから動作確認を行ってください。うまく動作すると図 17.4 のように表示されます。

17.3.2 エラーのときは …

もしエラーになった時には，`counter.cgi` については先の「エラーの時は …」の項目をチェックし，これを呼び出している HTML ファイルについては次の項目をチェックしてみて下さい。

HTML ファイルについて

図 17.4: 来訪者カウンタ実行画面

SSI の記述に間違いはないか？ ... 単に実行するべき “counter.cgi”を正しく指定していないのか，“# exe ...”のタグに問題がないのかチェックして下さい。

HTML ファイルの拡張子は指定されたものを使っているか？ ... SSI を動作させるときには HTML ファイルの拡張子を “.html”ではなく，“.shtml”とする場合があります。Web サーバ管理者に確認しておいて下さい。また SSI を許可する時には CGI と同様，HTML ファイルと同じディレクトリに “.htaccess”というファイルを作り

```
Options Includes
```

と書いておく必要があるかもしれません。

— メモ —

第18章　Perlを使ってみよう(2/2) — アンケート集計

本章では今まで培ってきたHTMLの知識と，前章で見てきたCGIの機能を使って，アンケート集計用のフォームとCGIを連携させてみます。

18.1　アンケート集計の仕組み

アンケート用のフォームに入力されたデータは，Webサーバ側にある”cgi-bin/get-answer.cgi”という名前のCGIスクリプトに渡されます。この時，GETメソッドでは環境変数”QUERY_STRING”に，POSTメソッドでは標準入力からデータが渡されます。このフォームでは後者が使用されていますが，渡されるデータはどちらのメソッドにしろ

```
sex=male&birth_year=1974& … (中略) … &last_name=%92q%8BI&os1=windows2000
```

という長ったらしいものでしかありません。日本語に至っては16進数で表示されており，このままでは文字列としては使い物になりません。更に，前述の通り，頻繁に使用される日本語の文字コードは4種類 (JIS, Shift JIS, EUC, UNICODE) あり，入力された文字コードを判別し，出力する際にはどれか一種類に統一しなければなりません。

これらの作業を行ってくれるPerlスクリプトとして古くはcgi-lib.pl，jcode.plがあります。ここではそれに備わっている関数を利用して，CGIスクリプトを組むことにします。

18.2　作って動かしてみよう

前の章で作成したフォーム入りのHTMLファイルを以下のように少しだけ改良します。これを“enquate.html”という名前で保存して，Webサーバの“public_html”ディレクトリに転送します。

```
<HTML>
<HEAD><TITLE>Enquate</TITLE></HEAD>
<BODY>
<H1>アンケート</H1>
<HR>
<FORM action="cgi-bin/get-answer.cgi" method="post">
<OL>
<LI> 貴方の性別は？<BR>
```

```
男<INPUT name="sex" value="male" type="radio">
  女<INPUT name="sex" value="female" type="radio">
<LI> 貴方の生年月は？<BR>
<SELECT name="birth_year">
<OPTION selected label="" value="none"></OPTION>
<OPTION>1974</OPTION>
<OPTION>1975</OPTION>
<OPTION>1976</OPTION>
<OPTION>1977</OPTION>
<OPTION>1978</OPTION>
<OPTION>1979</OPTION>
<OPTION>1980</OPTION>
<OPTION>1981</OPTION>
<OPTION>1982</OPTION>
<OPTION>1983</OPTION>
<OPTION>1984</OPTION>
<OPTION>1985</OPTION>
</SELECT>年
<SELECT name="birth_month">
<OPTION selected label="" value="none"></OPTION>
<OPTION>1</OPTION>
<OPTION>2</OPTION>
<OPTION>3</OPTION>
<OPTION>4</OPTION>
<OPTION>5</OPTION>
<OPTION>6</OPTION>
<OPTION>7</OPTION>
<OPTION>8</OPTION>
<OPTION>9</OPTION>
<OPTION>10</OPTION>
<OPTION>11</OPTION>
<OPTION>12</OPTION>
</SELECT>月

<LI>貴方の学籍番号は？<BR>
<INPUT name="number" type="text" width="20">
<LI>貴方の氏名をローマ字で記入して下さい。<BR>
氏：<INPUT name="first_name" type="text" width="30">
名：<INPUT name="last_name" type="text" width="30">
```

```
<LI>貴方が今までに使ったことのあるパソコンの OS を教えて下さい。<BR>
<INPUT name="os1" value="windows2000" type="checkbox" checked>Windows 2000
<INPUT name="os2" value="windows95" type="checkbox">Windows 95/98/ME
<INPUT name="os3" value="linux" type="checkbox">Linux

</OL>
<CENTER><INPUT type="submit" value="登録"> <INPUT type="reset" value="消去"></CENTER>
</FORM>
<HR>
<ADDRESS>Tomonori Kouya: &lt;<A HREF="mailto:tkouya@cs.sist.ac.jp">
tkouya@cs.sist.ac.jp</A>&gt;</ADDRESS>
</BODY>
</HTML>
```

このアンケートフォームに入力されたデータを表示する CGI を “get-answer.cgi”という名前で作成し，“public_html/cgi-bin”ディレクトリに転送した上で，前章と同じ要領でこの CGI スクリプトのモードを変更します。同じディレクトリには “cgi-lib.pl”(入力データを加工しやすい形に変換する関数群) と “jcode.pl”(文字コードを変換する関数群) を置いておく必要があります。

```
#!/usr/bin/perl

require "./cgi-lib.pl";
require "./jcode.pl";

# POST メソッドで入力されたデータを取得し，Input ハッシュに代入する
&ReadParse(*Input);

# データを Shift JIS に変換
&jcode'convert(*Input, "sjis");

# 以下，*Input ハッシュのデータを HTML として出力
print "Content-type: text/html\n\n";

&HtmlTop("アンケート解答");
print "<HR>\n";
print "<DL>\n";
print "<DT>性別:<DD>";
print $Input{'sex'}, "\n";
print "<DT>生年月日:<DD>";
print $Input{'birth_year'}, "年";
print $Input{'birth_month'}, "月\n";
```

```
print "<DT>学籍番号:<DD>";
print $Input{'number'}, "\n";
print "<DT>氏名:<DD>";
print $Input{'first_name'}; print $Input{'last_name'}, "\n";
print "<DT>使用 OS:<DD>";
print $Input{'os1'}, "\n";
print $Input{'os2'}, "\n";
print $Input{'os3'}, "\n";
print "</DL>\n";
print "<HR>\n";
&HtmlBot;
```

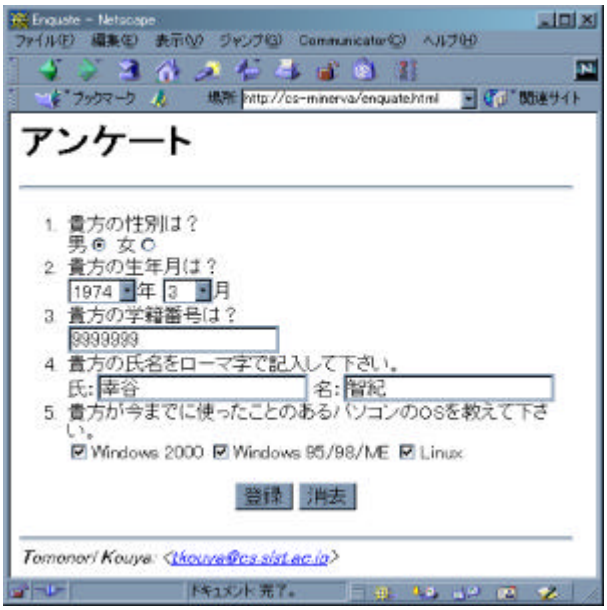


図 18.1: アンケートフォームにデータを入力

うまく動作すれば，図 18.1 のように入力したデータが，図 18.2 のように表示されるはずです。エラーが発生した時には，前の章のチェック項目を確認して下さい。

練習問題

1. (自由課題) このアンケート集計スクリプトを改変し，データをファイルに保存できるようにせよ。また，データをメールで転送することは可能か，検討せよ。

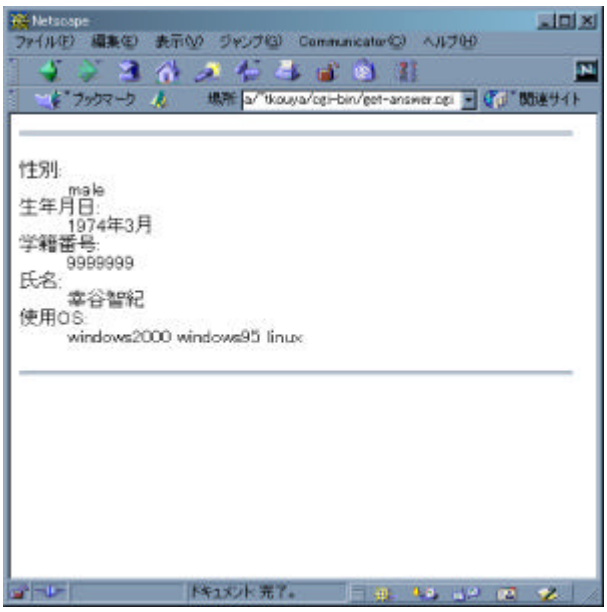


図 18.2: 入力データを HTML として表示

— メモ —

コラム★Webにおけるセキュリティ

「コンピュータの発達というのは，一つの合理的な中で発展を遂げるほど単純なものじゃないんですね。……要するに矛盾を包含しながら，それをさらに止揚して進歩していくものなんです。」(池田敏男語録)

田原 総一郎 (「日本コンピュータの黎明」より)

セキュリティ＝安心感を得ること

Internet が当たり前のものになってしまった昨今では,「セキュリティ」という言葉が重要視されています。ピーナッツ (スヌーピーが登場するアメリカの漫画) に登場するライナス君がいつも抱えている毛布は「セキュリティ・ブランケット」（日本語訳 :「安心毛布」）と呼ばれています。持っていないと不安になる，持っていると安心感を与えてくれる，という役目を持った毛布らしいのですが，これは「セキュリティ」という言葉そのものの意味をうまく表現しています。

普段の生活では特段必要になるものではない，けれども備えておけば憂いがなくなる，それがセキュリティなのです。自分の家に鍵をかけておくのはどんな時でしょう？　家を留守にする時だけですか？　それとも人が出入りする時以外は必ずかけておきますか？　チェーンは頑丈な鋼鉄製のものになっていますか？　窓ガラスをぶち破って侵入してくる泥棒にはどう対処しますか？　考え始めると，銀行の貸し金庫に入っても心配になることでしょう。しかし普通は，安全対策に要する費用と時間を，守りたいものの重要性とを秤にかけて判断することになります。

Internet におけるセキュリティ対策も同じことです。ただ，世界中のあらゆる人がどこからでもアクセス可能であるということ，アクセスは必ずしも人間だけではなく，プログラムが自動的に行っている可能性があるということが，日常生活のそれと一番異なる点です。

Webにおけるセキュリティ

「UNIX & インターネットセキュリティ」[10] には，Web で留意すべきセキュリティの問題点を次のように指摘しています。

1. 不正な攻撃者が，Web サーバまたは CGI スクリプトのバグを利用してシステム内のファイルにアクセスしようとすることがある。それだけでなく，コンピュータ全体を勝手に制御しようとすることさえあり得る。

2. Web サーバ上の極秘情報が閲覧権限を持たない人にも配布されてしまうことがある。

3. Web サーバからブラウザに転送される極秘情報が途中で盗まれることがある。

4. Web ブラウザのバグ (または気付かない機能) が原因で，Web クライアントの極秘情報に悪意ある Web サーバからアクセスされてしまうことがある。

5. WWW に関連するソフトウェア (サーバ，ブラウザ，Plug-in など) では特許保護されているものがあり，多くの企業や組織は個々に承諾を受けてソフトウェアを購入しなければならないケースが多い。そして，このような許諾を受けたソフトウェアが固有の欠点を生み出してしまう。

現在多く指摘されている問題は 1, 2, 4 です。3, 5 についても皆無とは言えませんが，前者に比べればまだ数は少ないと思われます。

1 については，今は一部の OS に組み込まれている商用 Web サーバが標的になることが多いようです。きちんとこまめに対策をしておけば問題は発生しないのですが，管理者がいい加減だったり，うっかりしていたりするとやられてしまうことがあります。また，現在では安全とされている Web サーバでも,「セキュリティホール」（脆弱な部分）を悪意ある人に突き止められて，大々的に攻め込まれる可能性は十分にあり得ます。まあ，Web サーバの管理人にさせられてしまった時にはこの点を十分に知っておく必要があるでしょう。

2 については，Web サーバの管理人もさることながら，ユーザとしても留意すべき点でしょう。よく，顧客データを盗まれたというニュースが流れますが，一番単純なケースで言うと，誰でもアクセス可能なディレクトリに，どこからもリンクの張られていないファイルを置いておいたというものです。私もよくやるので人のことは言えませんが，そのファイルが例えば「test.doc」とか「user.txt」というすぐに思いつきそうなありふれた名前だったとしたら，リンクのあるなしに関わらず，いずれは読み取られることになるでしょう。重要なファイルの取り扱いには十分留意すべきです。

3 は，これも特定 OS に組み込まれているブラウザがよく問題になっています。しかしこれも対策は取られており，修正プログラムをマメに当てていれば済む話です。また，そのような知識に欠けるとしても，ウィルスチェッカーや，それに含まれている「パーソナルファイアーウォール」という機能を利用することで，かなりの効果を上げることが出来ます。このような日々更新する必要のあるセキュリティソフトウェアを継続使用するには，定期的に料金を支払う必要があり，それは最低限の出費として覚悟しなければなりません。また，その覚悟がないなら，最低限，怪しげなメールを開いたり，Web サイトを覗いたりしてはいけません。

もっと知りたい人には …

古くからパソコン関係の雑誌に記事を書いてきた中村正三郎という人がいます。大変分かり易く，噛み砕いてセキュリティについて解説した本を昨年 (2001 年) に出版されました [11]。Internet やコンピュータにあまり詳しくない人にも分かるように，そういう人でも知っておくべきセキュリティについて蘊蓄を語った本ですので，一度ご覧頂くとよいでしょう。

「ウィルス，伝染るんです」，中村正三郎，廣済堂出版社，2001 年発行，1500 円 (税別)

関連図書

CGI 関係の資料は以下のものを使用しました。

[1] プログラミング Perl 第 3 版, L.Wall・近藤嘉雪 訳, O'Reilly Japan, 1997.
Perl の発明者である Larry Wall 自身が執筆した，バイブル的な書籍の翻訳です。

[2] CGI プログラミング第 2 版, S. Guelish 他・田辺茂也 監訳, O'Reilly Japan, 2001.
Perl で記述された CGI スクリプトの解説を行っています。CGI で動的にグラフィックスを生成する方法についての解説もあります。

[3] Borland C++Builder ライブラリリファレンス, ボーランド株式会社, 1997.
本書ではプログラミング言語としての C や C++は一切出て来ませんが，SSI で使用した日付の書式指定は C の関数を直接呼び出して実行させているだけなので，それと同じ書式指定が利用できます。付録に挙げてあるのはその抜粋です。

HTML 関係の資料は以下のものを使用しました。

[4] W3C, http://www.w3.org/
Web や XML に関する技術的な規格はここで発行されているものが圧倒的に多いです。殆どは英語ですが，マメにチェックすべきサイトです。

[5] HTML 4 仕様書邦訳計画, http://www.asahi-net.or.jp/~sd5a-ucd/rec-html40j/
[4] にある HTML 4.0 の仕様書の日本語訳が読めます。XML 関係の文書も含め，かなりの数の日本語訳が存在するので，英語が苦手な人はそちらを参照すると良いでしょう。

[6] 情報アーキテクチャ入門, L.Rosenfeld, P.Morvill, 篠原稔和 監訳, O'reilly Japan, 1998.
Web サイトを実際に運用する際に必要となるノウハウをわかりやすく提示しています。

Dynamic HTML と JavaScript については以下を参照しました。

[7] Dynamic HTML, Netscape Communication Co., 1997.

[8] Client-Side JavaScript Guide Version 1.3, Netscape Communication Co., 1999.

オーサリングツールについて参照した資料は以下の通りです。

[9] IBM ホームページビルダー, http://www.jp.ibm.com/software/internet/hpb/

セキュリティ関係の資料には次の書籍を利用しました。

[10] UNIX & インターネットセキュリティ, S.Garfinkel, G.Spafford, 山口 英 監訳, O'Reilly Japan, 1998.

[11] ウィルス，伝染るんです, 中村正三郎，廣済堂出版社，2001.

Internet 関係の資料としては以下の Web サイトを適宜参照しました。

[12] Hobbes' Internet Timeline, http://www.zakon.org/robert/internet/timeline/

[13] 青空文庫, http://www.aozora.gr.jp/

[14] .JPSearch.NET, http://www.jpsearch.net/

[15] 首相官邸, http://www.kantei.go.jp/

[16] 関東学院大学, http://www.kanto-gakuin.ac.jp/

[17] Asahi.com, http://www.asahi.com/

最後に。

Web を学ぶ上で，一番手本となるのが,「自分が面白いと感じ，惹き付けられた Web ページ」です。人の好みは千差万別ですが，だからこそ，世界に一人しかいない「自分」が興味を持ったものはそれだけで貴重な資料です。もし将来，自分独自の趣味を生かしたものであれ，会社から業務命令で渋々作らされるものであれ，Web ページを作る機会があれば，もう一度「自分が面白いと感じ，惹き付けられた Web ページ」群を思い浮かべてみて下さい。そしてどうして興味が湧いたのか，そのことをじっくり考えてみて下さい。それを踏まえた上で初めて,「自分以外の第三者が面白いと感じ，惹き付けられる Web ページ」が出来上がってくるのです。どうせ作るのならば,「自分が面白い」ものである上で「他人にも面白がってもらえる」ものにしたいじゃありませんか。本書はそのような，コンテンツ作りのハウツーを述べたものではありませんが，そこから先は「自分が面白いと感じ，惹き付けられる」ものを目指して頑張って下さい。最後に，私（幸谷）が面白いと思っている Web ページの URI を以下に示します。

[18] ほぼ日刊イトイ新聞, http://www.1101.com/

[19] 心理学総合案内「こころの散歩道」, http://www.n-seiryo.ac.jp/~usui/

[20] Web やぎの目, http://www.kt.rim.or.jp/~yhayashi/

付 録A　HTML 4.01 タグ・プロパティ一覧

A.1　タグ一覧

a : アンカー

```
<a
    accesskey=...
    charaset=...
    class=...
    coords=...
    dir=...
    href=
    hreflang=...
    id=...
    lang=...
    name=...
    rel=...
    rev=...
    shape=...
    style=...
    tabindex=...
    target=...
    title=...
    type=...

    onblur=...
    onclick=...
    ondblclick=...
    onfocus=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> ... </a>
```

abbr : 省略形であることを示す

```
<abbr
    class=...
    dir=...
    id=...
    lang=...
    style=...
    title=...

    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> ... </abbr>
```

acronym : 頭文字であることを示す

```
<acronym
    class=...
    dir=...
```

```
    id=...
    lang=...
    style=...
    title=...

    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> ... </acronym>
```

address : 作成者情報

```
<address
    class=...
    dir=...
    id=...
    lang=...
    style=...
    title=...

    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...

> ... </address>
```

applet : Java アプレットの埋め込み

```
<applet
    align=...
    alt=...
    archive=...
    class=...
    code=...
    codebase=...
    height=...
    hspace=...
    name=...
    object=...
    style=...
    title=...
    vspace=...
    width=...
> ... </applet>
```

area : client-side 画像マップ領域

```
<area
   alt=...
   class=...
   dir=...
   href=...
   id=...
   lang=...
   nohref=...
   shape=...
   style=...
   tabindex=...
   target=...
   title=...

   onblur=...
   onclick=...
   ondblclick=...
   onfocus=...
   onkeydown=...
```

```
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
>
```

b : フォントを太文字 (bold) にする

```
<b
    class=...
    dir=...
    id=...
    lang=...
    style=...
    title=...

    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> ... </b>
```

base : 文書の基盤 URI の指定

```
<base
    href=...
    target=...
>
```

basefont : デフォルトのフォントを指定

```
<basefont
    color=...
    face=...
    id=...
    size=...
>
```

bdo : 双方向アルゴリズムの上書き

```
<bdo
    class=...
    dir=...
    id=...
    lang=...
    style=...
    title=...

    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> ... </bdo>
```

big : 文字を拡大する

```
<big
    class=...
    dir=...
    id=...
    lang=...
    style=...
    title=...
```

```
    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> ... </big>
```

blockquote：長めの引用

```
<blockquote
    class=...
    cite=...
    dir=...
    id=...
    lang=...
    style=...
    title=...

   onclick=...
   ondblclick=...
   onkeydown=...
   onkeypress=...
   onkeyup=...
   onmousedown=...
   onmousemove=...
   onmouseout=...
   onmouseover=...
   onmouseup=...
> ... </blockquote>
```

body：文書本文の指定

```
(<body
   alink=...
   background=...
   bgcolor=...
   class=...
   dir=...
   lang=...
   link=...
   style=...
   text=...
   title=...
   vlink=...

   onclick=...
   ondblclick=...
   onkeydown=...
   onkeypress=...
   onkeyup=...
   onload=...
   onmousedown=...
   onmousemove=...
   onmouseout=...
   onmouseover=...
   onmouseup=...
   onunload=...
> ... </body>)
```

br：強制改行

```
<br
   class=...
   clear=...
   id=...
   style=...
   title=...

   onclick=...
   ondblclick=...
   onkeydown=...
   onkeypress=...
   onkeyup=...
   onmousedown=...
```

```
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
>
```

button : 押しボタン

```
<button
    class=...
    dir=...
    id=...
    lang=...
    style=...
    title=...

    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> ... </button>
```

caption : キャプション

```
<caption
    aling=...
    class=...
    dir=...
    id=...
    lang=...
    style=...
    title=...

    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> ... </caption>
```

center : 中央揃え

```
<center
    class=...
    dir=...
    id=...
    lang=...
    style=...
    title=...

    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> ... </center>
```

cite : 引用文献

```
<cite
    class=...
    dir=...
    id=...
    lang=...
```

```
    style=...
    title=...

    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> ... </cite>
```

code : 短いプログラムの表示

```
<code
    class=...
    dir=...
    id=...
    lang=...
    style=...
    title=...

    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> ... </code>
```

col : テーブルの列へ属性指定

```
<col
    align=...
    char=...
    charoff=...
    class=...
    dir=...
    id=...
    lang=...

    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
>
```

colgroup : テーブルの列グループ指定

```
<colgroup
    align=...
    char=...
    charoff=...
    class=...
    dir=...
    id=...
    lang=...
    span=...
    style=...
    title=...
    valign=...
    width=...

    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
```

```
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> ... </colgroup>
```

dd : 定義の記述

```
<dd
    class=...
    dir=...
    id=...
    lang=...
    style=...
    title=...

    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> ... </dd>
```

del : 削除されたテキストの表示

```
<del
    cite=...
    class=...
    dir=...
    id=...
    lang=...
    style=...
    title=...

    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> ... </del>
```

dfl : インスタンスの定義

```
<dfl
    class=...
    dir=...
    id=...
    lang=...
    style=...
    title=...

    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> ... </dfl>
```

dir : ディレクトリリスト

```
<dir
    class=...
    compact
```

```
    dir=...
    id=...
    lang=...
    style=...
    title=...

    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> ... </dir>
```

div：ブロックレベルの一般構造

```
<div
    class=...
    dir=...
    id=...
    lang=...
    style=...
    title=...

    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> ... </div>
```

dl：定義リスト

```
<dl
    class=...
    compact
    dir=...
    id=...
    lang=...
    style=...
    title=...

    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> ... </dl>
```

dt：定義される用語

```
<dt
    class=...
    dir=...
    id=...
    lang=...
    style=...
    title=...

   onclick=...
   ondblclick=...
   onkeydown=...
   onkeypress=...
   onkeyup=...
   onmousedown=...
   onmousemove=...
```

```
   onmouseout=...
   onmouseover=...
   onmouseup=...
(... </dt>)
```

em: 強調

```
<em
    class=...
    dir=...
    id=...
    lang=...
    style=...
    title=...

    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> ... </em>
```

fieldset : フォーム内のコントロールをグループ化する

```
<fieldset
    class=...
    dir=...
    id=...
    lang=...
    style=...
    title=...

    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> ... </fieldset>
```

font : フォント指定

```
<font
    class=...
    dir=...
    id=...
    lang=...
    style=...
    title=...

    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> ... </font>
```

form : 入力フォームを作る

```
<form
    accept-chaset=...
    accept=...
    action=...
    class=...
    dir=...
```

```
    enctype=...
    id=...
    lang=...
    method=...
    name=...
    style=...
    target=...
    title=...

    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
    onreset=...
    onsubmit=...
> ... </form>
```

frame：フレームウィンドウの指定

```
<frame
    frameborder=...
    longdesc=...
    marginheight=...
    marginwidth=...
    name=...
    noresize
    scrolling=...
    src=...
    style=...
    title=...
>
```

frameset：フレームを作る

```
<frameset
    cols=...
    rows=...

    onload=...
    onunload=...
> ... </frameset>
```

h1〜h6：見出し

```
<h1〜h6
    align=...
    class=...
    dir=...
    id=...
    lang=...
    style=...
    title=...

    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> ... </h1〜h6>
```

head：ヘッダ指定

```
(<HEAD>
... </HEAD>)
```

hr : 水平線

```
<hr
    align=...
    class=...
    dir=...
    id=...
    lang=...
    noshade=...
    size=...
    style=...
    title=...

    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
>
```

html : HTML 文書の指定

```
(<html
    version=...
> ... </html>)
```

i: イタリック

```
<i
    class=...
    dir=...
    id=...
    lang=...
    style=...
    title=...
    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> ... </i>
```

iframe : インラインのサブウィンドウ

```
<iframe
    aling=...
    class=...
    dir=...
    frameborder=...
    height=...
    id=...
    longdesc=...
    marginheight=...
    marginwidth=...
    name=...
    scrolling=...
    src=...
    style=...
    title=...
    width=...
> ... </iframe>
```

img : 画像の埋め込み

```
<img
    align=...
    alt=...
    border=...
    height=...
```

```
    hspace=...
    id=...
    ismap=...
    longdesc=...
    name=...
    src=...
    style=...
    title=...
    usemap=...
    vspace=...
    width=...

    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
>
```

input：フォームの部品

```
<input
    accept=...
    accesskey=...
    align=...
    alt=...
    checked=...
    class=...
    dir=...
    disabled
    id=...
    ismap=...
    lang=...
    maxlength=...
    name=...
    readonly
    size=...
    src=...
    style=...
    tabindex=...
    title=...
    type=...
    usemap=...
    value=...

    onblur=...
    onchange=...
    onclick=...
    ondblclick=...
    onfocus=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
    onselect=...
>
```

ins：挿入されたテキスト

```
<ins
    class=...
    datetime=...
    dir=...
    id=...
    lang=...
    style=...
    title=...

    onclick=...
    ondblclick=...
    onkeydown=...
```

```
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> ... </ins>
```

isindex : プロンプト表示

```
<isindex
    class=...
    dir=...
    id=...
    lang=...
    prompt=...
    style=...
    title=...
>
```

kdb : ユーザが入力するテキストの指定

```
<kdb
    class=...
    dir=...
    id=...
    lang=...
    style=...
    title=...

    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> ... </kdb>
```

label : フォーム内ラベル

```
<label
    accesskey=...
    class=...
    dir=...
    for=...
    id=...
    lang=...
    style=...
    title=...

    onblur=...
    onclick=...
    ondblclick=...
    onfocus=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> ... </label>
```

legend : グループ化されたフォーム内コントロールの説明文の指定

```
<legend
    accesskey=...
    align=...
    class=...
    dir=...
    id=...
```

```
    lang=...
    style=...
    title=...

    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> ... </legend>
```

li : リストの項目

```
<li
    class=...
    dir=...
    id=...
    lang=...
    style=...
    title=...
    type=...
    value=...

    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> (... </li>)
```

link : リンクの指定

```
<link
    charset=...
    class=...
    dir=...
    href=...
    hreflang=...
    id=...
    lang=...
    media=...
    rel=...
    rev=...
    style=...
    target=...
    title=...
    type=...

    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
>
```

map : cliend side の画像マップ

```
<map
    class=...
    dir=...
    id=...
    lang=...
    name=...
    style=...
    title=...
```

```
    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> ... </map>
```

menu : メニューリスト

```
<menu
    class=...
    compact
    dir=...
    id=...
    lang=...
    style=...
    title=...

    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> ... <menu>
```

meta : メタ情報の指定

```
<meta
    content=...
    dir=...
    http-equiv=...
    lang=...
    name=...
    scheme=...
>
```

noframes : フレームなしで描画する際の代替内容

```
<noframes
    class=...
    dir=...
    id=...
    lang=...
    style=...
    title=...

    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
>
... </noframes>
```

noscript : スクリプトをサポートしない場合の代替内容

```
<noscript
    class=...
    dir=...
    id=...
    lang=...
```

```
    style=...
    title=...

    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> ... </noscript>
```

object : 埋め込みオブジェクトの指定

```
<object
    align=...
    archive=...
    border=...
    class=...
    classid=...
    codebase=...
    codetype=...
    data=...
    dir=...
    height=...
    hspace=...
    id=...
    lang=...
    name=...
    standby=...
    style=...
    tabindex=...
    title=...
    type=...
    usemap=...
    vspace=...
    width=...

    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> ... </object>
```

ol : 番号付きリスト

```
<ol
    class=...
    compact
    dir=...
    id=...
    lang=...
    start=...
    style=...
    title=...
    type=...

    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> ... </ol>
```

optgroup : 選択項目のグループ化

```
<optgroup
    class=...
    dir=...
    disabled
    id=...
    label=...
    lang=...
    style=...
    title=...

    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> ... </optgroup>
```

option : 選択項目の指定

```
<option
    class=...
    dir=...
    disabled
    id=...
    label=...
    lang=...
    selected
    style=...
    title=...
    value=...

    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> (... </option>)
```

p : 段落

```
<p
    align=...
    class=...
    dir=...
    id=...
    lang=...
    style=...
    title=...

    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> (... </p>)
```

param : 名前付きプロパティの値を指定

```
<param
    name=...
    type=...
    value=...
    valuetype=...
```

```
>
```

pre : 整形済みテキストの表示

```
<pre
    class=...
    dir=...
    id=...
    lang=...
    style=...
    title=...
    width=...

    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> ... </pre>
```

q : 短い引用

```
<q
    cite=...
    class=...
    dir=...
    id=...
    lang=...
    style=...
    title=...

    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> ... </q>
```

s: 取り消し線

```
<s
    class=...
    dir=...
    id=...
    lang=...
    style=...
    title=...

    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> ... </s>
```

samp : サンプルプログラムの表示

```
<samp
    class=...
    dir=...
    id=...
    lang=...
    style=...
    title=...
```

```
    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> ... </samp>
```

script : スクリプトの記述

```
<script
    charset=...
    defer=...
    language=...
    src=...
    type=...
> ... </script>
```

select : オプションの選択肢

```
<select
    class=...
    dir=...
    disabled
    id=...
    lang=...
    multiple
    name=...
    size=...
    style=...
    tabindex=...
    title=...

    onblur=...
    onchange=...
    onclick=...
    ondblclick=...
    onfocus=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> ... </select>
```

small : 文字の縮小

```
<small
    class=...
    dir=...
    id=...
    lang=...
    style=...
    title=...

    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> ... </small>
```

span : インラインの一般構造

```
<span
    class=...
    dir=...
    id=...
```

```
    lang=...
    style=...
    title=...

    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> ... </span>
```

strike : 取り消し線

```
<strike
    class=...
    dir=...
    id=...
    lang=...
    style=...
    title=...

    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> ... </strike>
```

strong : 強調文字

```
<strong
    class=...
    dir=...
    id=...
    lang=...
    style=...
    title=...

    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> ... </strong>
```

style : スタイルの指定

```
<style
    dir=...
    lang=...
    media=...
    title=...
    type=...
> ... </style>
```

sub: 下付き添字

```
<sub
    class=...
    dir=...
    id=...
    lang=...
    style=...
```

```
    title=...

    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> ... </sub>
```

sup: 上付き添字

```
<sup
    class=...
    dir=...
    id=...
    lang=...
    style=...
    title=...

    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> ... </sup>
```

table : テーブルを作る

```
<table
    align=...
    bgcolor=...
    border=...
    cellpadding=...
    cellspacing=...
    class=...
    dir=...
    frame=...
    id=...
    lang=...
    rules=...
    style=...
    summary=...
    title=...
    width=...

    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> ... </table>
```

tbody : テーブル本体

```
(<tbody
    align=...
    char=...
    charoff=...
    class=...
    dir=...
    id=...
    lang=...
    style=...
    title=...
    valign=...
```

```
    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> ... </tbody>)
```

td : テーブルのセル

```
<td
    abbr=...
    align=...
    axis=...
    bgcolor=...
    char=...
    charoff=...
    class=...
    colspan=...
    dir=...
    headers=...
    height=...
    id=...
    lang=...
    nowrap=...
    rowspan=...
    scope=...
    style=...
    title=...
    valign=...
    width=...

    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> ... </td>
```

textarea : 複数行のテキスト入力フィールド

```
<textarea
    accesskey=...
    class=...
    cols=...
    dir=...
    disabled
    id=...
    lang=...
    name=...
    readonly
    rows=...
    style=...
    tabindex=...
    title=...

    onblur=...
    onchange=...
    onclick=...
    ondblclick=...
    onfocus=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
```

```
    onselect=...
>
```

tfoot : テーブルのフッタ

```
<tfoot
    align=...
    char=...
    charoff=...
    class=...
    dir=...
    id=...
    lang=...
    style=...
    title=...
    valign=...

    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> (... </tfoot>)
```

th : テーブルのヘッダのセル

```
<th
    abbr=...
    axis=...
    align=...
    bgcolor=...
    char=...
    charoff=...
    class=...
    colspan=...
    dir=...
    headers=...
    height=...
    id=...
    lang=...
    nowrap=...
    rowspan=...
    scope=...
    style=...
    title=...
    valign=...
    width=...

    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> (... </th>)
```

thead : テーブルのヘッダ

```
<thead
    align=...
    char=...
    charoff=...
    class=...
    dir=...
    id=...
    lang=...
    style=...
    title=...
    valign=...

    onclick=...
```

```
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> (... </thead>)
```

tr：テーブルの行

```
<tr
    align=...
    bgcolor=...
    char=...
    charoff=...
    class=...
    dir=...
    id=...
    lang=...
    style=...
    title=...
    valign=...

    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> (... </tr>)
```

tt：タイプライター書式指定

```
<tt
    class=...
    dir=...
    id=...
    lang=...
    style=...
    title=...

    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> ... </tt>
```

u：アンダーライン

```
<u
    class=...
    dir=...
    id=...
    lang=...
    style=...
    title=...

    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
```

```
    onmouseover=...
    onmouseup=...
> ... </u>
```

ul : 行頭文字付き箇条書き

```
<ul
    class=...
    dir=...
    id=...
    lang=...
    style=...
    title=...
    type=...

    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> ... </ul>
```

var : 変数やプログラムの引数指定

```
<var
    class=...
    dir=...
    id=...
    lang=...
    style=...
    title=...

    onclick=...
    ondblclick=...
    onkeydown=...
    onkeypress=...
    onkeyup=...
    onmousedown=...
    onmousemove=...
    onmouseout=...
    onmouseover=...
    onmouseup=...
> ... </var>
```

A.2 イベント一覧

onblur= ... タグがフォーカスを失った時

onchange= ... タグの値が変化した時

onclick= ... ポインタボタンがクリックされた時

ondblclick= ... ポインタボタンがダブルクリックされた時

onfocus= ... タグがフォーカスを取得した時

onkeydown= ... キーが押下された時

onkeypress= ... キーが押下されて離された時

onkeyup= ... キーが離された時

onload= ...

`<FRAMESET onload=...>`: 全てのフレームが読み込まれた時

`<BODY onload=...>`: ドキュメントが読み込まれた時

onmousedown= ... ポインタボタンが押下された時

onmousemove= ... ポインタが移動した時

onmouseout= ... ポインタが出て行った時

onmouseover= ... ポインタがタグの上に来た時

onmouseup= ... ポインタボタンが離された時

onreset= ... フォームがリセットされた時

onselect= ... ある選択肢が選ばれた時

onsubmit= ... フォームがサブミットされた時

onunload= ...

<FRAMESET onunload=...>: 全てのフレームが削除された時

<BODY onunload=...>: ドキュメントが削除された時

付 録B　SSI・CGI環境変数一覧

本章の内容は全て，Web サーバとして UNIX 上において Apache を使用している場合にのみ適用可能です。

B.1　SSI で使用可能な要素一覧

config … 表示形式の指定。

<!--#config errmsg="..." --> … エラー発生時にブラウザに送り返す文字列の指定。

<!--#config sizefmt="..." --> … ファイルサイズの単位指定。

<!--#config timefmt="..." --> … 時刻表示形式の指定 (後述)。

echo … SSI で使用可能な変数の値を表示します。

<!--#echo var="変数名" --> … 変数の値を表示する。

<!--#echo var="変数名" encoding="none" --> … 変数の値に含まれる特殊文字をエンコードしない。

<!--#echo var="変数名" encoding="url" --> … URL で使用可能なエンコード方法 (%付きで表示する)

<!--#echo var="変数名" encoding="entity" --> … これが含まれているブロック (段落など) に適合したエンコードを行う。

exec … 指定されたコマンドや CGI を動作させる。

<!--#exec cgi="CGI スクリプト" --> … 指定された CGI スクリプトを実行する。

<!--#exec cmd="コマンド" --> … 指定されたコマンドを/bin/sh シェルを使用して実行する。

fsize … 指定されたファイルのサイズを，sizefmt="..."で指定された形式で表示する。

<!--#fsize file="ファイル名" --> … ファイルのサイズを表示する。

<!--#fsize virtual="相対 URI or 相対パス" --> … 現在の文書からの相対 URI によるパスの指定。""から始まると現在の文書に対する相対パスの指定となる。

flastmod ··· 指定されたファイルの最終更新日時を，timefmt="..."で指定された形式で表示する。

<!--#flastmod file="ファイル名" --> ··· 指定されたファイルの更新日時を表示する。

include ··· 指定されたファイルの内容を挿入する。

<!--#include file="ファイル名" --> ··· 現在の文書に対する相対パスによるファイルの指定。

<!--#include url="相対 URI or 相対パス" --> ··· 現在の文書に対する相対 URI，あるいは相対パスによる指定。

set ··· 変数宣言とその値の指定。

<!--#set var="変数名" value="変数の値"> ··· 変数名の宣言とその値の指定。

B.2 timefmt の書式一覧

timefmt="..."による時刻の表示形式は次の SSI と共にに使用します。

```
<!--#config timefmt="%Y 年%M 月%d 日" -->

<!--#flastmod file="index.html" -->
<!--#echo var="LAST_MODIFIED" -->
```

この表示形式の指定は C の標準関数である strtime 関数のものと同じです。詳細は C 言語標準関数のリファレンスを参照して下さい。以下はその抜粋です [3]。

%% ··· %記号

%A, %a ··· 曜日とその省略形

%B, %b ··· 月とその省略形

%C ··· 世紀から 1 引いたもの

%c ··· 日付と時刻

%D ··· mm/dd/yy 形式の年月日

%d ··· 二桁で表わす日付 (01〜31)

%e ··· 一桁の場合はスペースが入る日付 (␣1〜31)

%H ··· 二桁で表わす 24 時間表示の時間 (00〜23)

%h ··· %b と同じ

%I … 二桁で表わす 12 時間表示の時間 (01〜12)

%j … 三桁で表わす年内の通算日数 (001〜366)

%M … 二桁の分 (00〜59)

%m … 二桁の月 (01〜12)

%n … 改行

%p … ”AM” or ”PM”

%r … ”AM”/”PM”文字列が入った時・分・秒

%S … 二桁の秒 (00〜59)

%T … hh:mm:ss 形式の 24 時間時刻

%t … タブ文字

%U … 日曜を週の一日目として数えた時の週の通算番号 (00〜53)

%u … 1 を月曜日とした曜日 (1〜7)

%w … 0 を日曜日とした曜日 (0〜6)

%W … 月曜を週の一日目として数えた時の週の通算番号 (00〜53)

%X, %x … 時刻と日付

%y … 末尾二桁の西暦

%Y … 4 桁の西暦

%Z … 時刻帯の名前 (日本だと”JST-9”等)

B.3 環境変数一覧

SSI・CGI で使用可能な環境変数一覧を抜粋したものを以下に示します。

DATE_LOCAL … ローカル時刻

DATE_GMT … 世界標準時刻

CONTENT_LENGTH … 渡された文字列の長さ

DOCUMENT_NAME … この文書のファイル名

DOCUMENT_URI … この文書ファイルの URI

GATEWAY_INTERFACE … CGI 形式のバージョン

HTTP_ACCEPT ··· サーバで受け入れ可能な MIME type

HTTP_ACCEPT_CHARSET ··· サーバで受け入れ可能な文字コード

HTTP_HOST ··· サーバ名

HTTP_USER_AGENT ··· ユーザが使用しているブラウザ名

HTTP_VIA ··· ユーザが経由してきた Proxy 名のリスト

LAST_MODIFIED ··· この文書ファイルの最終更新日時

PATH ··· パスの指定

QUERY_STRING ··· 入力データ文字列

REMOTE_ADDR ··· ユーザ側の IP アドレス

REMOTE_HOST ··· ユーザ側のホスト名 (FQDN)

REMOTE_PORT ··· ユーザ側のポート番号

REQUEST_METHOD ··· 要求メソッド名 (”GET”，”POST”等)

REQUEST_URI ··· 要求 URI

SCRIPT_NAME ··· 実行スクリプト名 (SSI の場合，文書ファイル名)

SERVER_ADDR ··· Web サーバの IP アドレス

SERVER_NAME ··· Web サーバ名 (FQDN)

SERVER_PORT ··· Web サーバのポート番号 (通常は 80)

SERVER_SOFTWARE ··· Web サーバのソフトウェア名

SERVER_PROTOCOL ··· Web サーバで使用しているプロトコル

USER_NAME ··· ユーザ名

索 引