

## 第4章 WWWによる情報発信

### 4.1 WWWプレゼンテーションの計画

#### 4.1.1 良いWeb ページを見つけよう

まず、他人の書いた Web ページを見てみましょう。見やすい Web ページ、わかりやすい Web ページがあったら、それが「良い Web ページ」なのです。逆に、見にくい Web ページ、わかりにくい Web ページは「悪い Web ページ」になります。

良い Web ページを書こうとするならば、まず最初に、「良い Web ページと思われるものを探すこと」が大切です。

#### 4.1.2 扉ページの問題

あるまとまった内容を WWW を使って情報発信する時、最初に訪れてもらうようにするのが、扉ページの問題です。扉ページは、ひと目見ただけでその中身のおおよその検討がつき、しかも、その中身を読む気にさせるものに工夫して書かれます。

しかし、グラフィックに凝ってしまった結果、そのページを閲覧するために時間がかかってしまうと、遅い回線を使ってる人にとっては事実上読むことができないこととなります。

従って、扉ページは「早く、分かりやすく」という一見すると矛盾する条件を満たすことが求められます。

#### 4.1.3 文書の分散

同じ量の文書を WWW を使って掲示する際に、それをいくつかの HTML 文書に分割するかによって、見え方が違ってきます。

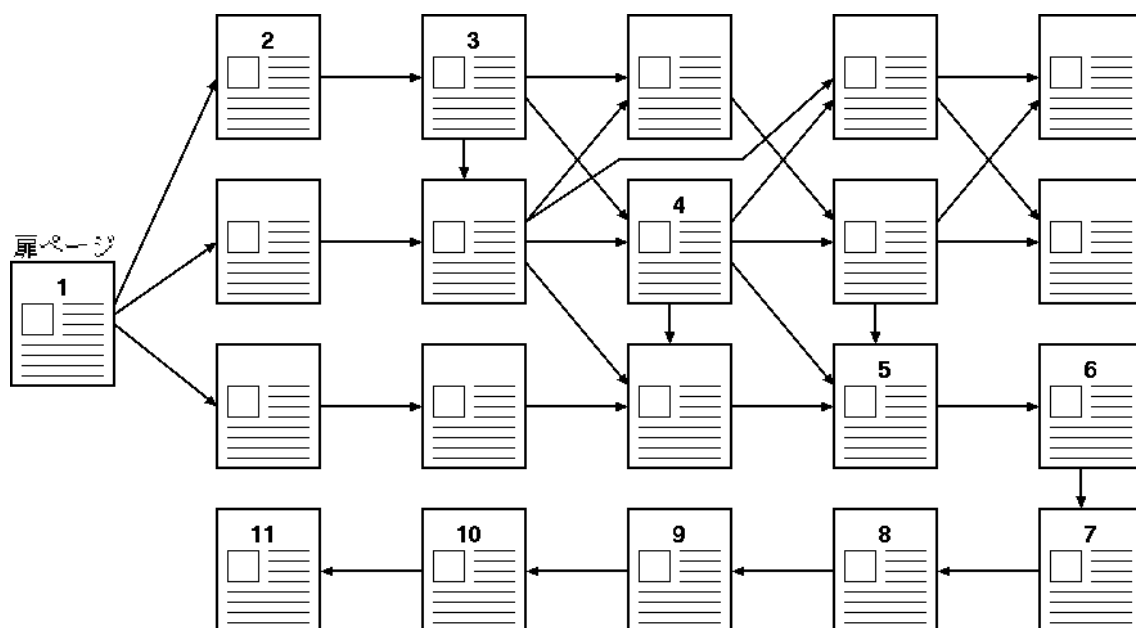


図 4.1: リンクが深いと読みにくくなる。

しかし、リンク機能を使いたいために、不必要に多くの文書に分割をしてしまい、本当に得たい文書にたどり着く迄に、いくつものリンクをたどる必要がある文書は、読むのが非常に面倒です。多くても 3 回程度で最終的な文書にたどり着けるようなリンク構成を考えるべきでしょう。

#### WWW での情報発信への道のり

前節までで、WWW 上にはさまざまな人がさまざまな目的で公開しているさまざまなページがあることが分かりました。この節では、自分で WWW のページを作り公開することについて学びます。

しかし、ひと口に「ページを作り公開する」といっても、そのためには次のような多くの作業が必要になります。

- ・ 公開したい情報や公開の目的が何であるかを定める
- ・ 公開する情報をどのような形に構成するかを定める
- ・ ページのスタイルや 1 つひとつのページの構成を決める
- ・ 実際にページ群のファイルを作成する
- ・ ブラウザでページ群を見て意図通りであることを確認する
- ・ WWW サーバにページ群を置いて公開する

コンピュータになじみのない人にとっては、一見すると「ページ群のファイルを作成する」ことや「サーバにページ群を置いて公開する」ことが一番難しくかつ重要であるように思えてしまいがちですが、実際にはその他の部分も等しく重要なことなのです。

以下ではこれらのことがらについて、順に学んでゆきます。この節では、上の課題のうち前半の 3 項目を取り上げます。残りの内容は次の節以降で取り上げます。

##### 4.1.4 公開する情報と目的を決める

どのようなプレゼンテーションを作るのであっても、まず次の 2 点は、はっきりさせておかなければなりません。

- ・ 目的 プレゼンテーションの目的は何か
- ・ 範囲 プレゼンテーションにはどのような情報が含まれるのか

これらがあいまいなままプレゼンテーションを作成しても、結果は「ただ何となく作っただけ」に終わってしまい、役に立たないものになりがちです。

目的は、明快にひとことで言いあらわせるようなものであるべきです。たとえば、「WWW のページを作成する練習をする」ということでもよいのです。しかし WWW は他の人に何かを伝えるための手段ですから、「これこれのことを伝える」という目的の方がよいプレゼンテーション（他の人から見て面白いプレゼンテーション）になりやすいでしょう。

プレゼンテーションに含まれる情報の範囲を決めることも重要です。そうしておかないと、プレゼンテーションが何パーセントぐらい完成したのか、まだ何が足りないのかといったことが分かりませんし、情報の構成を決めるのもうまく行かないでしょう。ここでは例題として、目的と範囲を次のように決めてプレゼンテーションを作成して行くことにします。

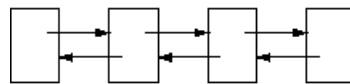
【演習】

私たちは教科「情報」でどんなことを学んでいるか  
なぜこれらのことを学ぶ必要があるのか  
知らなくてもよいこと、もっと知った方がよいことは何か

#### 4.1.5 情報の構成を決める

WWW のプレゼンテーションを作成するにあたって、どのようなページにどのようなつながりを持たせて構成するか決めることは、自分たちの表現したいことを相手に効率よく伝える上でとても重要です。構成がでたらめだと、読み手は必要な情報にたどりつく前にあきらめてよそへ行ってしまいかも知れません。

線形構造



階層構造

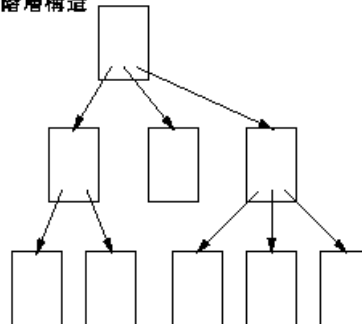


図 4.2: ハイパーテキストの基本的な構造

WWW のようなハイパーテキストのシステムでは、次の 2 とおりの構造が基本になると考えてよいでしょう (図 4.2)。

- ・線形構造 最初のページ、次のページ、その次のページ、というふうにページが直線的にリンクでつながって順番に並んだ構造。
- ・階層構造 全体のページに各章のページへのリンク、各章のページにはその章に含まれる各節へのリンクというふうに、全体から多数の細かい部分に向かって順におりていくような構造。

【ちょっと気をつけて!】

コンピュータの技術用語では、階層構造のことを木の幹からの枝分かれになぞらえて木構造と呼ぶこともあります (上下がさかさまですが)。その場合、全体に相当する部分を根、根の反対側の端を葉と呼びます。

実際には、線形構造でも途中で枝分かれがあったり、階層構造の下の方で一部に線形構造が使われたりして両者が組み合わされることが多いものです。しかしともかく、自分がプレゼンテーションを作る場合にはどちらの構造を基本にするかを決めてから始めるのが

よいでしょう。

ここの例題では、簡単な階層構造を採用することにして、次のような構成を採用します(図 4.3)。

【演習】

入口ページに、目的、科目構成、教科内容、議論などを入れる。

各教科内容ごとにページを用意し、入口ページからリンクする。

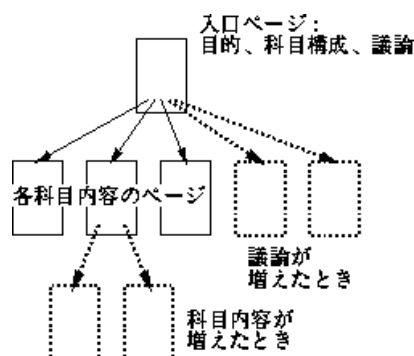


図 4.3: 例題プレゼンテーションの構造

#### 4.1.6 ページのスタイルや構成を決める

プレゼンテーション全体の構成が決まったら、次は個々のページの構成に進みますが、その前に 1 つ考えておくことがあります。それは、プレゼンテーションに含まれるページのスタイル（見え方）を決めておくことです。1 つのプレゼンテーション内で各ページが共通のスタイルを持つようにしておくと、読み手は「まだ同じプレゼンテーション内を見ている」と安心できますし、複数人で分担してページを作成してもばらばらな印象にならずに済みます。

ここの例題では簡単ながら、次のようなスタイルを取ることにします。

例題:ページのスタイル

各ページは次のような構成を取る。

- ・ 先頭の見出し そのページの内容全体をあらわす
- ・ 先頭の説明 そのページの目的や概要を述べる（省略可）
- ・ 区切り線
- ・ 中見出し+ 内容 そのページに含まれる項目（必要なだけ繰り返す）
- ・ 区切り線
- ・ 署名 ページの書き手に関する情報

ではいよいよ、ページの構成に進みます。きちんとやるなら、主要なページすべてについて構成を考えて検討してから作成に進むことになりますが、この例題プレゼンテーションはごく簡単なものですから、入口ページの構成だけ決めて、残りのページは作るときにそのつど構成を考えることにしました。

例題:入口ページの構成

タイトル:なぜコンピュータを学ぶのか?

- ・教科「情報」の構成 科目構成の説明
- ・教科「情報」の内容 学ぶ項目の一覧 (内容ごとのページにリンク)
- ・不要な (?) 内容
- ・もっと知りたい内容
- ・まとめ

【演習】

自分たちオリジナルの WWW プレゼンテーションを企画せよ。目的、範囲、構造、ページのスタイルと構成をひとつと決めよう。

Web ページに書いておきたいこと

多くの場合、WWW の Web ページは次の 4 つの部分からなります。

1. 表題
2. 表紙に相当する部分
3. 中身
4. 奥付

これらは内容による分類です。

4.1.7 表題

表題とは、そのページ全体の主題となっている言葉のことで、ブラウザで閲覧すると、多くの場合はブラウザのウィンドウの最上部に表示されます。ここには短く端的に内容を表わす言葉を入れるべきでしょう。

4.1.8 表紙に相当する部分

Web ページをデザインする場合、その冒頭部には表題で示された内容を視認性を高めて掲示するのが一般的です。また、この部分に画像を挿入することも多いようです。

中身の部分は、今までに述べてきたような普通の文書が表示されるように書けばよいでしょう。

4.1.9 奥付

本の場合、本の最初か最後のページ付近に、

- ・著者名、著者略歴等
- ・出版社名、印刷所名
- ・出版年月日、印刷年月日、版数、刷数
- ・著作権表示
- ・定価
- ・ISBN コード

などが表示されています。これは、その本についての「出版業的な情報」を書いたもので

「奥付」と呼ばれる部分です。書店、図書館などは奥付をみて本を整理し、読者は奥付を見てその本の著者を知ります。また、その本の内容を引用したり、紹介するときにも、奥付にかかれた内容が正式な情報になるのが通例です。即ち、奥付とは「本の権利を守る部分」です。そこで、WWWによる情報発信を考えてみます。WWWで発信された情報は、情報作成者・情報発信者がさまざまな権利を有しています。そこで、そのような権利が存在していることを明示し、権利保有者を明示すれば、これらの権利を見ず知らずのうちに侵害してしまうことを防ぐ効果が期待出来ます。そこで、Webページには必ず奥付をつけるようにします。本の奥付に倣ってWebページの奥付を考えると、以下の項目を表示するとよいでしょう。

- ・ 著者名
- ・ 制作者名
- ・ 制作年月日、改訂年月日
- ・ 著作権表示（著作権者の表示）
- ・ 正式な URL と、URL の公開の可否
- ・ 連絡先（メールアドレス）

## 4.2 HTML による WWW ページの記述

### 【ちょっと気をつけて!】

この節ではいよいよ、WWW ページを作成します。WWW ページを作成するには、HTML(HyperText Markup Language) と呼ばれる計算機言語を使用します。このため、まず計算機言語の話から説明します。

### 4.2.1 コンピュータと計算機言語

コンピュータは、さまざまな種類の情報をさまざまな形でビット列に符号化して表し、デジタル情報として取り扱います。ここで、情報の種類として「人間から計算機への指示や意思の伝達」について考えてみましょう。

人間どうして指示や意思を伝達するときには、日本語や英語などの言語（自然言語）を使います。コンピュータに対しても同じにできればよいのですが、それには次のような問題があります。

- ・ 自然言語は大変複雑であり、それによって指示や意思を表現したものをコンピュータで取り扱うことは、現在の技術をもってしても難しい。
- ・ 自然言語はあいまいさがあり、コンピュータのために厳密かつ簡潔に指示や意思を表現するのには向いていない。

そこで、コンピュータに指示や意思を伝達するために、自然言語の代わりにコンピュータで取り扱うのに適した人工言語を定めて使うことが行なわれています。これを計算機言語と呼びます。

計算機言語の代表は、BASIC や COBOL など、コンピュータの動作（プログラム）を記述するためのプログラミング言語です。しかしそれ以外にも、データベースのデータを取り扱うためのデータ操作言語や、コンピュータで扱う文書を表すためのマークアップ言語

など、さまざまな計算機言語が作られ使われています。

それぞれの計算機言語は、それ固有の書き方の規則があり、それに厳密に従っておかないとコンピュータでうまく扱ってもらえません。非常に杓子定規で不便な感じがしますが、その代わりとても短い記述でコンピュータにさまざまな指示を与えられるという利点があります。

## HTML 言語

HTML(HyperText Markup Language) は、WWW のページを記述するために作られたマークアップ言語です。HTML を使うことで、箇条書き、画像、表、リンクなど WWW ページに現われるすべてのものを作りだすことができます。

### 【ちょっと気をつけて!】

HTML の重要な考え方に、「見え方ではなく意味を指定する」というものがあります。たとえば見出しは通常大きい字で表示されますが、HTML では「ここからここまで大きい字で表示」ではなく「ここからここまで見出し」という形で指定するのです。そしてその具体的な見え方は環境ごとに変わってきます。たとえば、文字の大きさが 1 種類しか使えない環境用のブラウザは、見出しを下線つきで表示したり反転表示することで、見出しらしく見せてくれます。

## HTML の基本構造

HTML による記述は、基本的に次の形をしています。ここで 1 行目 (DOCTYPE 宣言) は、以下の記述が HTML4.0 と呼ばれる規格に従っていることを示しています (ほかに HTML2.0、HTML3.2 などもありますが、HTML4.0 が最新版です)。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
<HEAD> ... ヘッダ部分... </HEAD>
<BODY> ... 本体部分... </BODY>
</HTML>
```

2 行目からが HTML による記述ですが、「<なんとか...>」という形のものが多数ありますね。これを HTML ではタグと言います。タグはさらに、開始タグ (例: <HTML>) と終了タグ (例: </HTML>) に分けられ、一対の開始タグと終了タグではさまれた範囲を要素 (例: HTML 要素) と言います。HTML では要素の中に別の要素が含まれることができます。上に示したように、HTML 記述では HTML 要素の中に HEAD 要素と BODY 要素がこの順で含まれます。

HEAD 要素の内側をヘッダといい、ここにはこの WWW ページがどのようなページであるかについての情報を記述します。一方、BODY 要素の内側を本体といい、ここにページの内容 (ブラウザの窓の中に現われるもの) を記述します。表示させてみよう

いつまでもお話ばかりではつまりませんから、WWW ページを作ってブラウザで表示させてみましょう。次に示すのは、そのままブラウザで表示させられる、完結した HTML コードです。

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
<HEAD>
<TITLE>We Study Computers. Why?</TITLE>
</HEAD>
<BODY>
<P>

私たちは教科「情報」で、コンピュータをはじめとする、
情報技術について学ぶようになりました。

</P>
</BODY>
</HTML>

```

ここでは新しいタグが 2 つ、出て来ています。TITLE(表題) は、ヘッダに入る代表的な要素で、ページの表題を定めます。表題はブラウザの窓のタイトルバーに表示されたり、しおり (bookmark) の項目名として使われたりします。P(paragraph、段落) は、本体に入る代表的な要素で、普通の文章 (地の文) を書くのに使われます。

では実際にやって見ましょう。ファイル home.html に上の内容を打ち込んで、ブラウザの「ファイルを開く」機能で表示させたところを図 4.4 に示します。

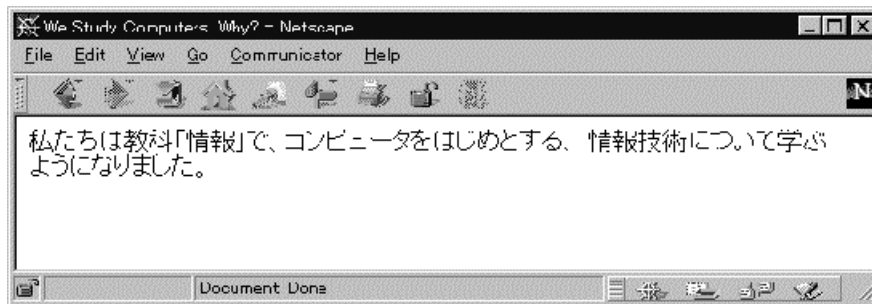


図 4.4: 表題と段落

これを見ると、確かに表題がブラウザのタイトルバーに、また段落がブラウザ窓の中に表示されています。ここで、段落の行の折り返しが元の HTML ファイルと違うことに注意してください。ブラウザは窓の幅に合わせて段落の中の折り返しを自動的に調整します。一般に HTML ファイルの中での行かえや空白の入れ方はブラウザによる表示には影響しません。

**【ちょっと気をつけて!】**

ただし、後で出てくる<PRE>...</PRE>の中だけは例外で、PRE 要素の内側では HTML ファイルの空白や行かえがそのまま表示されます。

**【演習】**

上の例の HTML ファイルと同じものを打ち込み、ブラウザで表示させてみよ。またブラウザの窓の幅を変更して、折り返しのようすを観察せよ。

#### 4.2.2 見出しと横線



ではいよいよページの内容を予定に沿って作っていくことにしましょう。その場合、上から順番にいきなり打ち込んでいくのではなく、まず見出しなど全体構造の分かるものを作り、そのあとでその項目ごとの内容を追加して行くのがよいでしょう。こうすると、ページ全体の内容のなかで現在作っている部分がどのような位置にあるかを常に見直しながらか、ページを作っていくことができます。このように、自由な順序で文書を作成していけることも、コンピュータで文書を作ることの利点の 1 つです。

HTML での見出しは、H1、H2、H3、H4、H5、H6 という 6 種類の要素で表します。H1 が一番上のレベルの大見出しで、H2、H3 と数字が大きくなるほど小見出しになっていくのです。

HTML では区切りの横線は、HR という要素で表します。ただしこれまでと違って、HR は

という単独のタグで、閉じタグはありません。横線を引くだけなので、それで十分なわけです。

ついでに、ページの最後の署名にはそれが署名であることを表す ADDRESS 要素を使うのが通例です。これらを使って入口ページの骨組みを完成させたところを次に示します。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HEAD>
<TITLE>We Study Computers. Why?</TITLE>
</HEAD>

<BODY>
<H1>なぜコンピュータを学ぶのか?</H1>
<P>
私たちは教科「情報」で、コンピュータをはじめとする、
情報技術について学ぶようになりました。
</P>

<HR>
<H2>教科「情報」の構成</H2>
<H2>教科「情報」の内容</H2>
<H2>不要な(?)内容</H2>

<H2>もっと知りたい内容</H2>
<H2>まとめ</H2>
<HR>
<ADDRESS>コンピュータと教育研究会</ADDRESS>
</BODY>
</HTML>
```

【演習】 先の HTML ファイルに見出しを追加してブラウザで表示させてみよ。

#### 4.2.3 箇条書き

ここからはいよいよ、それぞれの項目の内容を書いていながら、HTML の機能を学んでいきます。

情報をまとめる文章では、箇条書きを多く活用しますから、次に HTML の箇条書きについてまとめましょう。まず、いちばん多く使われる箇条書きである番号なしリスト (UL、unordered list) は次の形をしています。

```
<UL>
<LI> ... 項目 1 ...
<LI> ... 項目 2 ...
<LI>
</UL>
```

要素 LI (list item) が 1 つの項目に対応します。項目はいくつあっても構いません。<LI> の閉じタグは書いてもいいのですが、省略してもよいので、ここではそうしました。

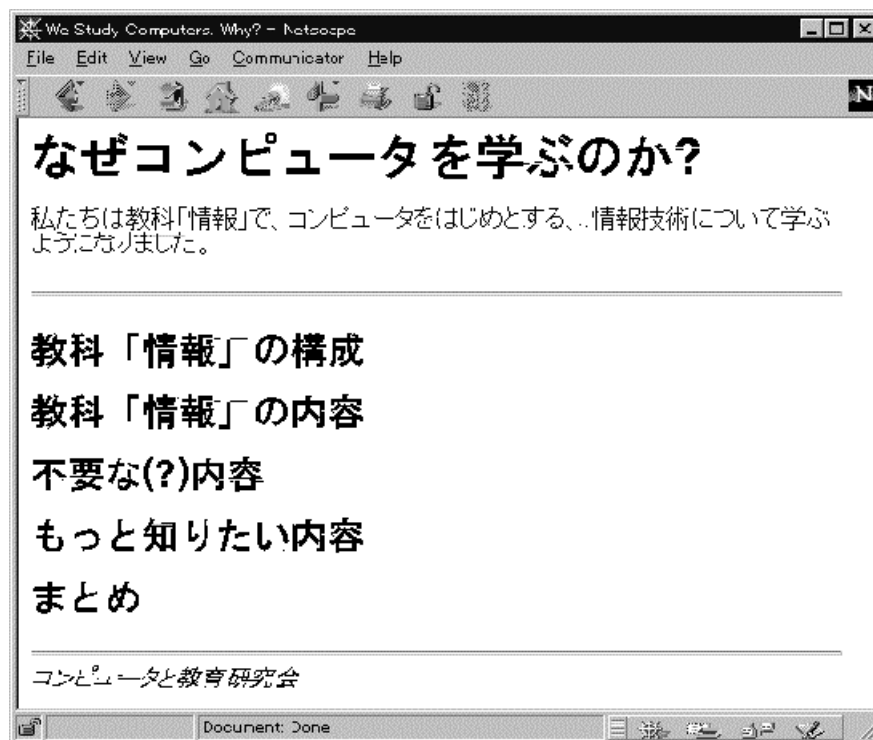


図 4.5: 見出しと横線

ではこれを利用して書いた内容の追加部分を見てみましょう (箇条書きでない部分はこれまで通り P 要素になります)。

```

<P>
コンピュータを学ぶ目的について納得しておく、
次のようなよいことがあります。
</P>
<UL>
  <LI>学ぶことに対して意欲的になり効率よく学習できる。
  <LI>自分にとって重要なテーマに的を絞ることができる。
  <LI>目標をどれくらい達成したか自分で評価できる。
</UL>
<P>

このページでは、「なぜコンピュータを学ぶのか?」というテーマについて、
私たちに調べたことや、考えたことをまとめてみました。
</P>

```

<LI>タグが他の行より 1 文字下げて書かれていますが、これは HTML のファイルを見たり編集するときに分かりやすくするためです (HTML ファイルでの行かえや字下げはブラウザでの表示には影響しません)。これを表示したようすを図 4.6 に示します。

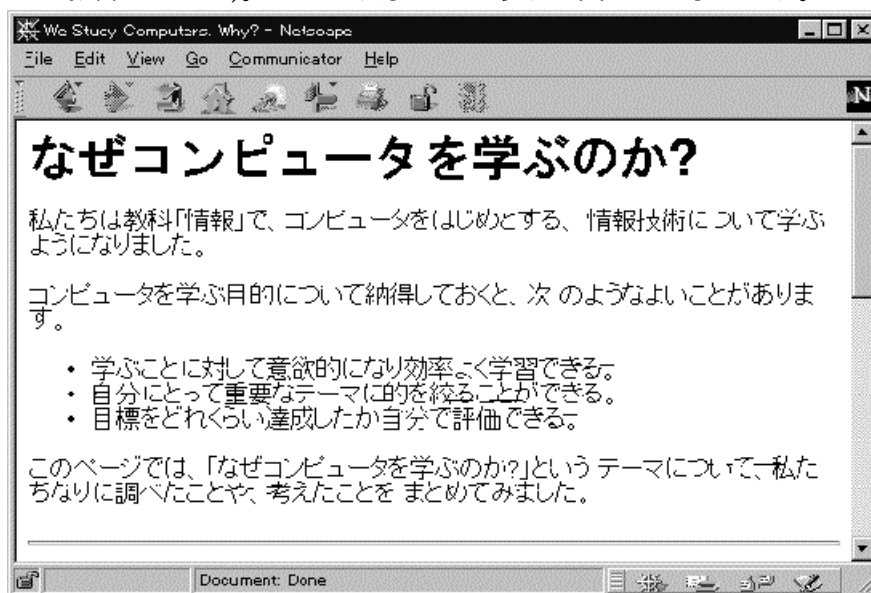


図 4.6: 番号なしリスト

ここでタグ <UL>...</UL>を <OL>...</OL>に取り換えると、番号つきリスト (OL、ordered list) になり、図 4.7 のように項目に 1 から順に番号が振られます。番号つきリストは、項目の順番に意味があるような箇条書きに使うとよいでしょう。

箇条書きの 3 番目の形として、項目ごとに短い語が見出しのようになった定義リスト (DL、definition list) があります。これは各項目が DT 要素と DD 要素の対になっていて、全体として次の形をしています。

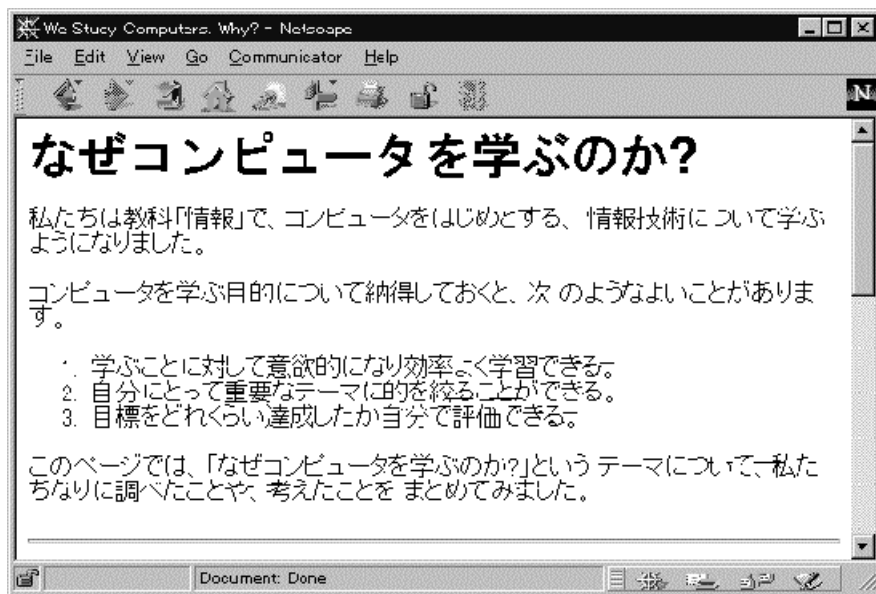


図 4.7: 番号つきリスト

```
<DL>
<DT>用語 1
<DD> ... 説明 1 ...
<DT>用語 2
<DD> ... 説明 2 ...
</DL>
```

DT 要素や DD 要素も LI 要素と同様、閉じタグは省略できます。  
ではこれを使った内容を見てみましょう (図 4.8)。

```
<HR>
<H2>教科「情報」の構成</H2>

<P>
教科「情報」には情報 A、情報 B、情報 C の 3 つの科目があります。
</P>
<DL>

<DT>情報 A
<DD>コンピュータネットワークの成り立ちについて学ぶ。
コンピュータを用いてネットワーク上の情報を利用したり、ネットワークに情報を発信する。

<DT>情報 B
<DD>コンピュータ上での情報の表現やプログラムについて学ぶ。
コンピュータネットワークの利用を通じて、情報技術と社会の関わりについて学ぶ。
```

<DT>情報 C

<DD>プログラミングや応用プログラムの利用を通じて、コンピュータを用いた問題解決について学ぶ。

</DL>

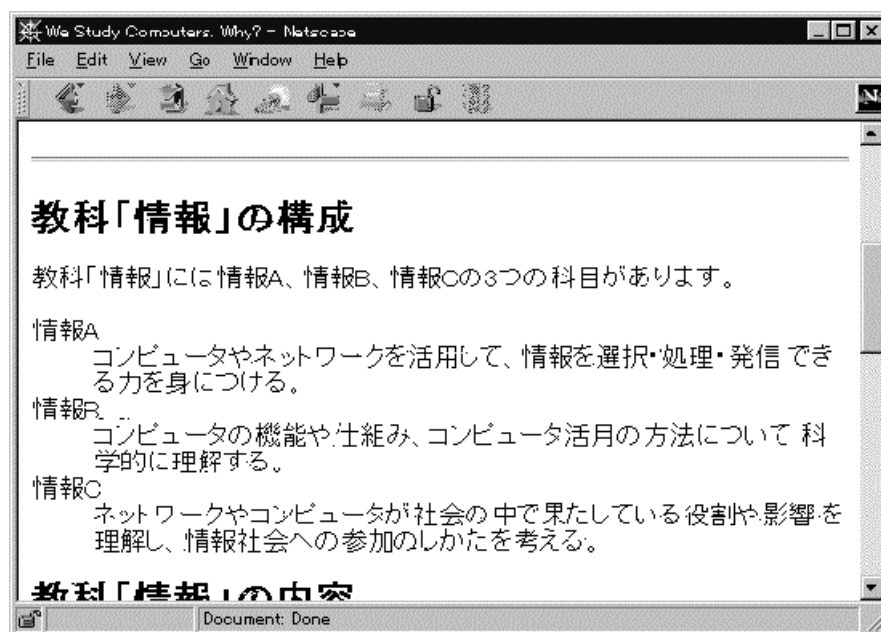


図 4.8: 定義リスト

【演習】先の HTML ファイルに箇条書きを追加してブラウザで表示させてみよ。

## 表

箇条書きと並んで情報を整理するのに有効な手段に、表があります。表もタグの構造は箇条書きに似ていますが、箇条書きが 1 次元なのに対して表は 2 次元だという違いがあります。具体的には次のようになります。

- TABLE 要素で、表全体を表す。
- TABLE 要素の中に入る TR (TABLE row) 要素で、表の 1 行を表す。
- TR 要素の中に入る TH (TABLE header) 要素または TD (TABLE data) 要素で、表の 1 つのます目を表す。

なお、TH と TD は、そのます目の内容が見出しであるか普通の情報であるかで使い分けます (見出しの場合はやや太字で中央揃えして表示されます)。そして、TR、TH、TD の閉じタグは省略できます。

これらをまとめると、具体的には表を HTML で書くと次のようになるわけです (すべて TD 要素にした場合)。

```
<TABLE BORDER=1>
<TR> <TD> 項目<TD> 項目...
<TR> <TD> 項目<TD> 項目.....
</TABLE>
```

表の行数や列数は中に入っている TR 要素や TH、TD 要素の数から自動的に決まります。

なお、TABLE の開始タグについている「BORDER=1」というのは、表の罫線の太さを指定するもので、0 を指定するか、またはこの指定をまったく書かないと罫線は描かれませんが、もっと大きい数字を指定すると太い罫線になります。このように、開始タグの内側に書いてその要素に関する特性を調整するための指定を HTML では属性 (attribute) と呼んでいます。

では、表を使った内容を見てみましょう (図 4.9)。

```
<TABLE BORDER=1>
<TR><TH>項目<TH>概要
<TR><TH>情報の表現
<TD>デジタル情報、ビット、2 進数、各種の情報の表現
<TR><TH>コンピュータ
<TD>コンピュータの原理、ハードウェア、ソフトウェア
<TR> ... (途中略)...
<TR><TH>応用ソフトウェアによる問題解決
<TD>表計算ソフト、
シミュレーション
</TABLE>
```

【演習】先の HTML ファイルに表を追加してブラウザで表示させてみよ。

#### 4.2.4 インラインイメージ

ここまででは、プレゼンテーションの内容はすべて 1 つの HTML ファイルで「閉じて」いました。しかし WWW の良いところは、あるファイルから別のファイルを参照できるところにあります。ここではそのような方法を 2 つ、取り上げます。

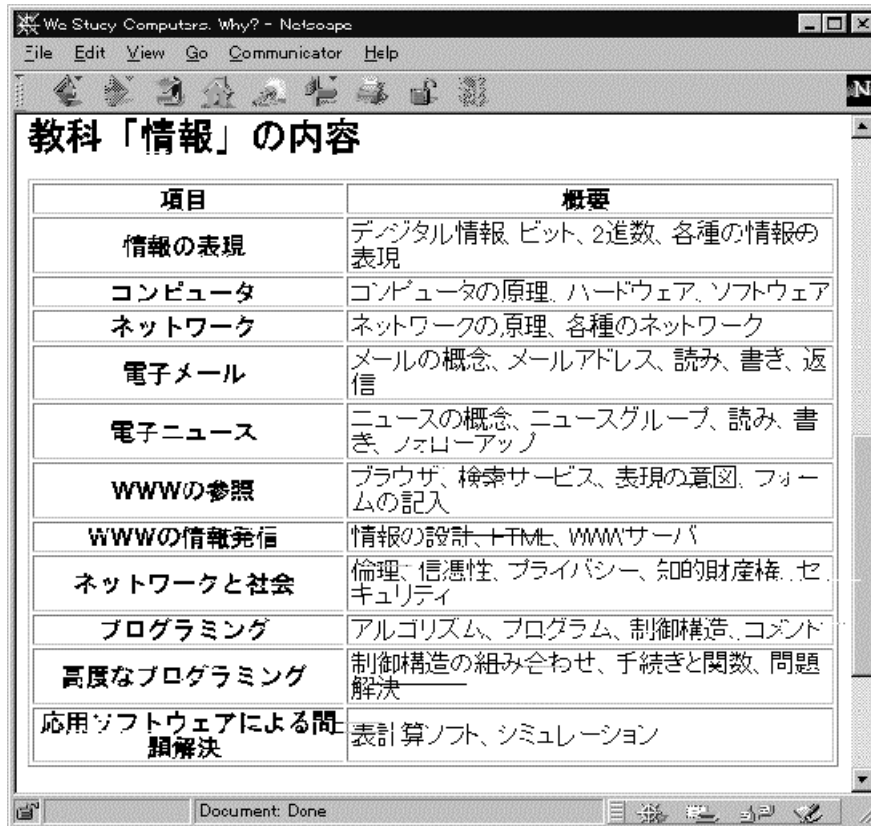
1 つ目は、あるページの中に画像 (イメージ) を埋め込むことです。これをインラインイメージ (inline image) と呼びます (inline というのは「その場所に」という意味です)。そのためには、次のような IMG タグを使います。

```

```

ここで SRC 属性はイメージ (GIF 形式または JPEG 形式) を格納したファイルを指定し、ALT 属性はそのイメージに関する説明を書くようになっています。たとえば末尾の署名のところに自分たちの写真 (?) を入れたとしましょう。そのファイル名を ourphoto.gif であ

るとして、最後の部分を次のように直しました。



項目	概要
情報の表現	デジタル情報、ビット、2進数、各種の情報の表現
コンピュータ	コンピュータの原理、ハードウェア、ソフトウェア
ネットワーク	ネットワークの原理、各種のネットワーク
電子メール	メールの概念、メールアドレス、読み、書き、返信
電子ニュース	ニュースの概念、ニュースグループ、読み、書き、フォローアップ
WWWの参照	ブラウザ、検索サービス、表現の意図、フォームの記入
WWWの情報発信	情報の設計、HTML、WWWサーバ
ネットワークと社会	倫理、信頼性、プライバシー、知的財産権、セキュリティ
プログラミング	アルゴリズム、プログラム、制御構造、コメント
高度なプログラミング	制御構造の組み合わせ、手続きと関数、問題解決
応用ソフトウェアによる問題解決	表計算ソフト、シミュレーション

図 4.9: 表

```
<HR>
<ADDRESS><IMG SRC="ourphoto.gif" ALT="Our Photo">
コンピュータと教育研究会</ADDRESS>
</BODY>
</HTML>
```

これを表示しているようすを図 4.10 に示します。

【ちょっと気をつけて!】

このブラウザではマウスカーソルが画像の上でしばらく止まると説明文が表示されるようになっています。このあたりの動きはブラウザによって違います。

【演習】 作成したページにインラインイメージを加えてみよ。

#### 4.2.5 リンク

他のファイルを参照するもう 1 つの方法であるリンクは、機能的にはページの中のある部分を選択すると表示が別のページに切り替わるというもので、これまでも多く利用してきましたが、これも HTML のコードにおいて別のファイルを参照しているわけです。HTML でリンクを作る場合は、A(anchor) タグと呼ばれるタグを使い、次のように記述し

ます。

```
<A HREF="ファイル名"> ... リンクテキスト...</A>
```

ここでリンクテキストと書いた部分は下線つきなど他とは区別できるように表示され、ここを選択する（多くのブラウザではマウスでクリックする）と、HREF 属性で指定したファイルに表示が切り替わり

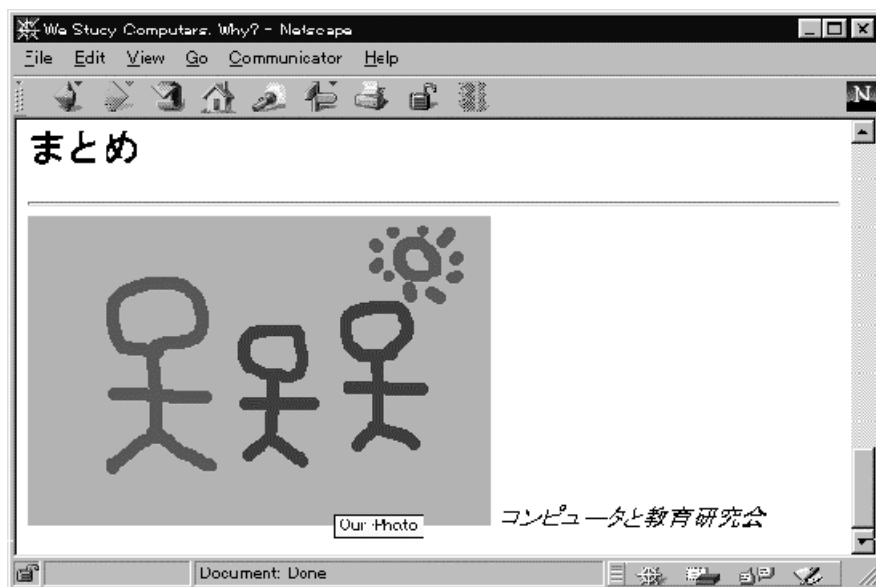


図 4.10: インラインイメージ

ます。なお、「リンクテキスト」の部分は実はテキストに限らず、インラインイメージなどを含んでいても構いません。

ここでは、先に出てきた教科の各内容に関する表を手直しして、各項目の部分をリンクテキストにしてみます。

```
<TABLE BORDER=1>
<TR>

<TH>項目<TH>概要<TR>
<TH><A HREF="info-repl.html">情報の表現</A>
<TD>デジタル情報、ビット、2 進数、各種の情報の表現
<TR>
<TH><A HREF="computers.html">コンピュータ</A>
<TD>コンピュータの原理、ハードウェア、ソフトウェア
<TR> ... (以下略)...
```

このように手直しした表の表示のようすを図 4.11 に示します。ここで「情報の表現」と記された部分を選択すると、info-repl.html というファイルの内容がブラウザに表示されるのです。



## URLの参照

上ではインラインイメージもリンクも、元のページのファイルと一緒に置いたファイルの名前を指定していました。しかし実際には、SRC 属性や HREF 属性では任意の URL を指定できます。これによって、リンクを選択すると他のサーバのページが表示されるようになりますし、(必要なら) 他のサーバにあるイメージをページに埋め込んだりできるのです。

たとえば、この章で最初に出てきた WWW ページの HTML コードは次のようになっています。

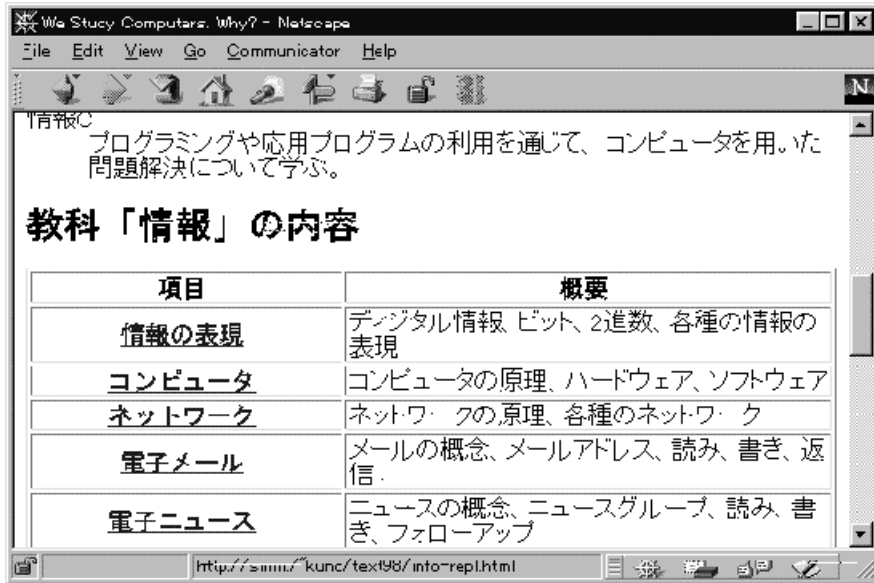


図 4.11: リンクテキスト

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
<HEAD>
<TITLE>Welcome Page of "Jouhou"</TITLE>
</HEAD>
<BODY>
<H1>科目「情報」の入口ページ</H1>
<UL>
<LI><A HREF="http://www.ntt.co.jp/japan/index-j.html">日本の情報(NTT)</A>
<HR>
<LI><A HREF="http://www.yahoo.co.jp/">ディレクトリサービス(Yahoo)</A>
<LI><A HREF="http://www.infoseek.co.jp/">検索サービス(Infoseek)</A>
</UL>
</BODY>
</HTML>
```

【演習】自分の作ったページに、他のサーバへのリンクをつけ加えてみよ。

## 4.3 プレゼンテーションの公開・検査・保守

#### 4.3.1 サーバへの転送

ここまでは、自分が作成したページやそれが参照する画像のファイルなどは、すべて手元のコンピュータに置かれていました。実際にこれらのページを WWW に公開するためには、これらのファイルを WWW サーバ（正確には WWW サーバプログラムを実行しているコンピュータ）に転送する必要があります。

サーバへのファイル転送にはさまざまな方法があり得ますが、ここでは ftp コマンドを使うものとして例を示します。ftp による転送では、次の手順を踏みます。

1. 転送先のサーバを指定して ftp コマンドを起動する。
2. サーバ上でのユーザ名とパスワードを入力する。
3. サーバ上のファイルを置くディレクトリを指定する。
4. バイナリ転送モードにする。
5. 各ファイルを転送する。
6. ftp コマンドを終了する。

では実際にこの手順にしたがって転送しているようすを見てみましょう。

##### 【ちょっと気をつけて!】

実際のサーバ名、ユーザ名、パスワード、およびディレクトリは、それぞれのクラスや個人ごとに違ってきます。正しいサーバ名とディレクトリ名を教えてもらってから実習してください。

```
% ftp www.kou1.ne.jp  ftp コマンドを起動
220 kou1 FTP server ready.  OK の表示
Name: masa  ユーザ名を入力
331 Password required ...  パスワード必要
Password: xxxxxxxx  パスワードを入力
(表示されない)
230 User masa logged in.  接続成功
ftp> cd /put/htdocs/masa  ディレクトリを指定
250 CWD command successful.  指定 OK
ftp> bin  バイナリモードに変更
200 Type set to I.  変更 OK
ftp> put home.html  ファイルを指定して転送
150 Binary data connection ...  ...  転送のメッセージ
ftp> put ourphoto.gif  別のファイルを指定して転送
150 Binary data connection ...  ...  転送のメッセージ
ftp> bye  これでおしまい
221 Goodbye.  終了 OK %
```

転送が完了したらただちに、作成したページは WWW 経由で世界中からアクセス可能になります。たとえば、上の例ではファイル home.html の URL は

<http://www.kou1.ne.jp/users/masa/home.html>

になります。

**【ちょっと気をつけて!】**

自分が作ったページの URL はそれぞれ教えてもらってください。

**【ちょっと気をつけて!】**

環境によっては、ある特定のディレクトリにファイルを置くとそれだけで自動的にサーバによって公開できるようになっている場合もあります。具体的には、自分が使っているホストで WWW サーバが動いている場合や、WWW サーバのディスクをネットワーク経由で自分の使っているマシンから読み書きできるように設定されている場合がこれに相当します。

**【演習】**

自分の作成したページを WWW サーバに転送してみよ。さらにその URL を指定してネットワーク経由で表示させてみよ。

#### 4.3.2 プレゼンテーションのチェック

ここまでで、コンテンツをサーバに転送してサーバから表示できるようになりました。しかし、「一応表示できる」ということと「プレゼンテーションとして完成した」ということはまったく別のことです。作成したプレゼンテーションは、最終的には世界中の人が（日本語で作ったのな日本語ができる人に限られはしますが）見ることになるのです。ですから、あなたがプレゼンテーションを良くするために 1 分間を掛けることが、多数の人の 1 分間を節約するのだと思ってください。

そのためにはまず、作成したプレゼンテーションに欠陥がないか確認します。具体的には、次の点をチェックしてください。

- ・ タグの使い方が HTML の規則に合っているかどうか。開始タグと終了タグが正しく対になっているか、タグ名や属性のスペルミスはないか、等。
- ・ リンクの指定は正しいか。たどれない（存在しないページやコンテンツを指している）リンクはないか。たどった先は意図した通りの内容か。

言うのは簡単ですが、これらをきちんとチェックするのは結構大変です。しかしもちろん、これらのことを自動的にチェックしてくれるプログラムがいくつも用意されています。そのようなプログラムの助けが借りられるのなら、ぜひ利用してください。

しかし実は、もっと大切なのは上に挙げたような「機械的にチェックできる」ことがらではなく、「読み手にとってわかりやすいプレゼンテーションであるかどうか」なのです。このために、次のようなチェックをしてください。

- ・ できるだけ多くのブラウザを用意して、作成したプレゼンテーションが問題なく見られるか、使いにくいところはないか検討する。
- ・ プレゼンテーションの設計に当たってたてた方針に照らして、各ページがプレゼンテーションの目的にかなっているか検討する。

- ・ 知合いに頼んでブラウザでプレゼンテーションを見てもらい、まずいいところがないかチェックする。何となく面白そうなところを見てもらうのと、特定の情報を探してもらうのと両方の使い方でするのがよい。

#### 【演習】

作成したプレゼンテーションをチェックして、問題点を思いつく限りたくさん書き出せ。それらを手直しする方法を考えて、できるものから実行してみよ（最初から完璧を目指さなくてもよい）。

### 4.3.3 プレゼンテーションの公開

これでようやく、自分のプレゼンテーションが公開できたことになります。しかしそれは「終わり」ではなく「始まり」にすぎません。プレゼンテーションを公開すれば、それを見た人からさまざまな反応が（電子メールなどで）あるかも知れません。

また、そのような反応はなくても、WWW サーバはいつどのページが取り寄せられたか、どのようなエラーが発生したかなどをログファイルと呼ばれるファイルに記録していますから、これを見ることでどのページがどれくらい参照されているか、どんなエラーが起きているかを知ることができます。

そして、WWW ページは生きものですから、これらの反応に対応して、絶えずよりよく改良して行くべきものです。また、公開した内容自体も、状況が変化したり、新しい情報が現われるのに対応して改訂して行くべきなのです（何か月もほったらかしの WWW ページは見るからにつまらなそうに感じられるものです）。ですから、ページの公開を続ける限り、改良と更新は続けて行ってください。

#### 【演習】

公開した WWW ページについて、反応を収集して整理してみよ。またその情報を参考に、どのような改訂がのぞましいかを計画せよ。

## 4.4 スタイルシート

### 4.4.1 スタイルシートとは

ここまでで学んだ HTML の機能はすべて、「ここは箇条書き」「ここは見出し」のようにテキストの論理的構造を規定するものでした。具体的に箇条書きがどのように整形され、見出しがどのような大きさの文字で表示されるかはブラウザに任されることになります。

実は、HTML には文字の大きさや色など、見ため（表現）を直接指定するような機能も用意されています。しかし、最新の HTML 規格 (HTML4.0) ではこのような機能を非推奨（なるべく使わない）と指定しています。なぜでしょう？ それは、HTML で論理的構造と表現を両方とも指定してしまうと、文章をさまざまな形で利用しようとしたとき、じゃまになったりするからです。

#### 【ちょっと気をつけて!】

HTML で表現まで指定してしまうと、見出しだけ取り出して目次を作るときに、`<H1>...</H1>`の中を取り出して整理しようとしたら、中に色指定のタグが混ざっていた、などの不都合が起きるかも知れません。

そこで HTML4.0 では、HTML そのものは文章の論理的構造を指定するのに用い、表現を指定するためにはスタイルシートと呼ばれる別の指定方法を併用するという方針になったのです。スタイルシートとは簡単に言えば、「`<H1>`の見出しはすべてゴシック体で」というふうに HTML の各要素単位でその表現を指定する機能です。

以下では現在もっとも普及しているスタイルシートの記法である CSS(直列スタイルシート) を取り上げ、その機能や使い方を見てみましょう。

#### 【演習】

先に作った HTML プレゼンテーションについて、表現（色、文字の大きさ、かこみ枠など）をどのようにつけたら見た目がよいか、計画を立ててみよ。

#### 4.4.2 スタイルシートの指定方法

HTML でスタイルシートを使う場合、次の 3 通りの指定方法があります。

(1) `<P STYLE="color: blue">...</P>`のように、表現を指定したい箇所のタグに`<STYLE>`属性を指定して、その値としてスタイル記述を行う。この方法は、全体に統一的なスタイルを指定するのには向いていません。

(2) HTML ファイルのヘッダ (`<HEAD>...</HEAD>`の中) にスタイル記述を書く。まとめて統一的にスタイルが設定できる。具体的には次のようにする。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN"> HTML4.0 だという指定
<HTML>

<HEAD>
<TITLE>... 表題...</TITLE>
<STYLE TYPE="text/css"> ... スタイル定義... </STYLE>
</HEAD>
<BODY> ... 本文... これまで通り</BODY>
</HTML>
```

(3) HTML ファイルのヘッダで、スタイルシートを格納したファイルの URL を指定する。この場合は、上記の`<STYLE>...</STYLE>`の代わりに次のように`<LINK>`タグを使う。

```
<LINK REL="stylesheet" HREF="スタイルシートファイル">
```

やや複雑だが、複数の HTML ファイルでスタイルシートを共通に利用できるという利点がある。以下では (2) の方法を使うことにします。

#### 4.4.3 C S S の指定方法

ではいよいよ、CSSの指定方法を説明しましょう。CSSではスタイル定義は

```
セレクタ, セレクタ, ... - 規則; 規則; ... ; 規則"
```

という形をしています。セレクタはとりあえずタグ名だと考えてください。そしてそれぞれの規則は

```
プロパティ: 値
```

という形になっています。たとえばページ全体の背景をうす青にするには、セレクタ BODY に対して background-color プロパティを指定します。

```
BODY - background-color: #CCCCFF "
```

また、H1 と H3 の見出しすべてに対して青い枠をつけたければ、border プロパティとして solid、border-color プロパティとして blue を指定します。

```
H1, H3 - border: solid; border-color: blue "
```

上の2つのスタイル規則を適用したようすを、図 4.12 に示します。

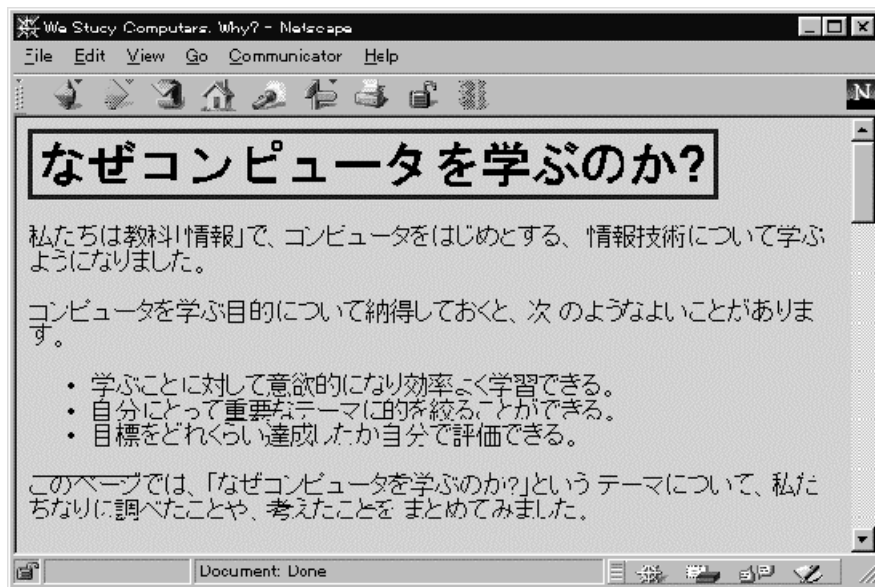


図 4.12: 背景と枠のスタイル

#### 【演習】

自分の作ったページについて、上で挙げたようなスタイル指定を入れてみよ。  
<HEAD>...</HEAD>の中に入れ、全体を<STYLE>...</STYLE>で囲むのを忘れないように。成功したら、タグ指定や色指定を変更してみよ。

#### 4.4.4 CSS のおもなプロパティ

上の例に出て来たものを含めて、代表的なプロパティについて紹介しておきましょう。

- ・ color: 色指定

文字などの色を指定

- background-color: 色指定

背景の色を指定

- border-style: スタイル

ふちをつける指定。ふちのスタイルとしては solid(実線のふち)、dashed(点線のふち)、double(2本線のふち)、groove(溝)、ridge(土手)が指定できる。

- border-width: 幅

ふちの幅。幅や長さは 10px(10ピクセル)、8pt(ポイント 1pt は 1/72 インチ)、12mm(ミリメートル) など単位をつけて指定できる。

- border-color: 色

ふちの色。

- font-size: 大きさ

文字の大きさ。10pt のように長さでも指定できるが、xx-small、x-small、small、normal、large、x-large、xx-large(それぞれ、極小、とても小さい、小さい、普通、大きい、とても大きい、極大) という指定もできる。

- line-height: パーセント

行間を指定。200%はダブルスペース。

- margin-left: 長さ、margin-right: 長さ

左右のマージン(余白)を指定。

- text-indent: 長さ

段落1行目の字下げを指定。長さのほかに、10%のように全体幅に対する百分率でも指定できる。

- text-align: 揃え

left(左寄せ)、right(右寄せ)、center(中央揃え)のいずれかを指定する。

- text-decoration: 飾り

underline(下線)、line-through(横線抹消)、blink(点滅)のいずれかを指定。

点滅は目障りだから最少限度にすること。

例として、先と同じ内容に

```
H1 - text-decoration: underline "  
P - text-indent: 20px; border: ridge "  
UL - background: #CCCCFF "
```

というスタイルを適用したものを図 4.13 に示す。

#### 【演習】

本節で説明したスタイル機能から面白そうなものを選び、前節のスタイルシート定義に追加してみよ。それぞれのプロパティはどのような目的に利用できると思うか考えよ。

#### 4.4.5 スタイルの設計

ここまででスタイルシートの機能について説明しましたが、どのようにスタイルを利用すべきかについては説明しませんでした。単に「目立てばよい」では、見ためだけ派手になってしまい、かえって伝えたい内容が読み手に伝わらなくなるため逆効果です。

ですから、ちょうど内容（コンテンツ）について行ったように、いきなり気ままにスタイルをつけるのではなく、どのようにスタイルをつけるか予め設計すべきなのです。スタイルの設計に際しては次のことがらを参考にしましょう。

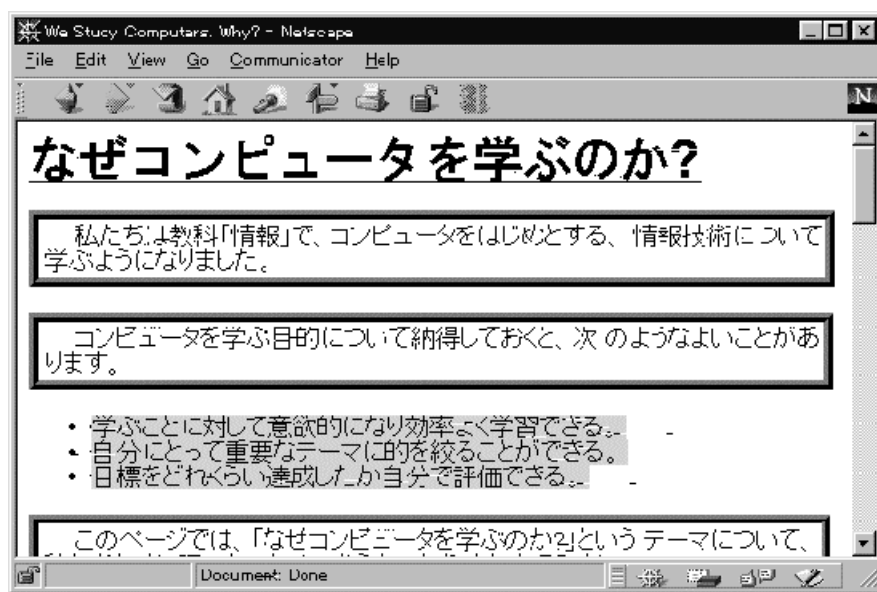


図 4.13: 別のスタイル指定を適用

- ・スタイルの目的は読み手に内容が伝わりやすくすることだと考えて設計する。
- ・そのため、ページの構造が分かるようにスタイルを設計する。たとえば大見出しはある程度目立つように枠などを使い、その中の小見出しは地の文よりは目立つが大見出しを拾うのに邪魔にならないようなスタイルを使う等。
- ・色は同色系統でまとめ、「ここだけは強調したい」という箇所に限って異色系統を使って目立たせる。
- ・スタイルは自分で考えるより読み手にとっては刺激的に感じられる。自分では控え目だと思うくらいでちょうどよい。極端に大きい文字や原色の色づかいは避ける。

#### 【演習】

1つのWWWページに対して、数人/数グループでそれぞれが別個にスタイルをつけてみてコンテストを行え。さらにその結果に基づいて、どのようなスタイルが好感を持たれるか分析してみよ。

#### おまけ1： 個別的なスタイル指定

ここまででは、スタイルを「すべての H1」とか「すべての P」のようにタグごとに指定していましたが、しかし場合によっては、「この H1 の見出しは重要なので赤にしたい」ということもあるでしょう。そのような場合はまず、セレクトタとしてタグの代わりに「クラス」



を指定します。セレクタにクラス名を指定するときは、先頭にピリオドをつけて

```
.important - color: red; font-size: x-large "
```

のように指定します。次に、タグの方は CLASS 属性でクラス名（こんどはピリオドなし）を指定します。

```
<H1 CLASS="important">.....</H1>
```

こうすると、その H1 の見出しには通常の H1 のスタイルに加えて important クラスとして指定されたスタイルが追加適用されます。

しかし、これでもまだスタイルの適用範囲はタグの範囲と一緒ですね。そこで HTML では、スタイル指定に便利な次の 2 つのタグを用意しています。

- ・ <DIV CLASS="...">...</DIV> 段落などを含むブロックの範囲を指定する。
- ・ <SPAN CLASS="...">...</SPAN> 段落の中などのテキストの範囲を指定する。

たとえば、段落のある 1 フレーズだけに important クラスを適用したければ、次のように

```
<P>この段落は<SPAN CLASS="important">きわめて重要</SPAN>なので...
```

するわけです。

SPAN はこのように段落の内側に使いますが、もっと広い範囲（複数の段落、箇条書き、見出しなどを含む範囲）を囲む場合には DIV を使ってください。

おまけ 2: 要素の独立配置 ここまでのところ、ある要素にスタイルをつけてもその要素が置かれる位置は他の要素との関係で決まってきました。しかし、スタイルシートで要素に position 指定を行うことで、その要素を本来の位置からずらしたり、他の要素と独立に任意の位置に置いたりできます。具体的には次のプロパティを使います。

- ・ position: 位置指定 位置指定をおこなう。relative(相対配置、top、left プロパティで本来の位置からどれだけずらすかを指定する)、absolute (独立配置: top、left プロパティで画面上の位置を指定する) が使える。
- ・ top: 位置、left: 位置 どれだけずらすか/どの位置に置くかを指定する。
- ・ width: 長さ、height: 長さ その要素を表示する領域の幅/高さを指定する。
- ・ overflow: あふれ指定 width、height で指定した大きさより要素の大きさが大きいときの処理を指定。clip(あふれた部分を切り取る)、scroll(スクロールバーをつける) が指定できる。

たとえばスタイルの部分で

```
.overlay {position: absolute; left: 100px; top: 100px;  
font-size: xx-large; color: red}
```

のように指定し、本文に

```
<DIV CLASS="overlay"><P>やっほー!</P></DIV>
```

を入れた様子を図 4.14 に示します。

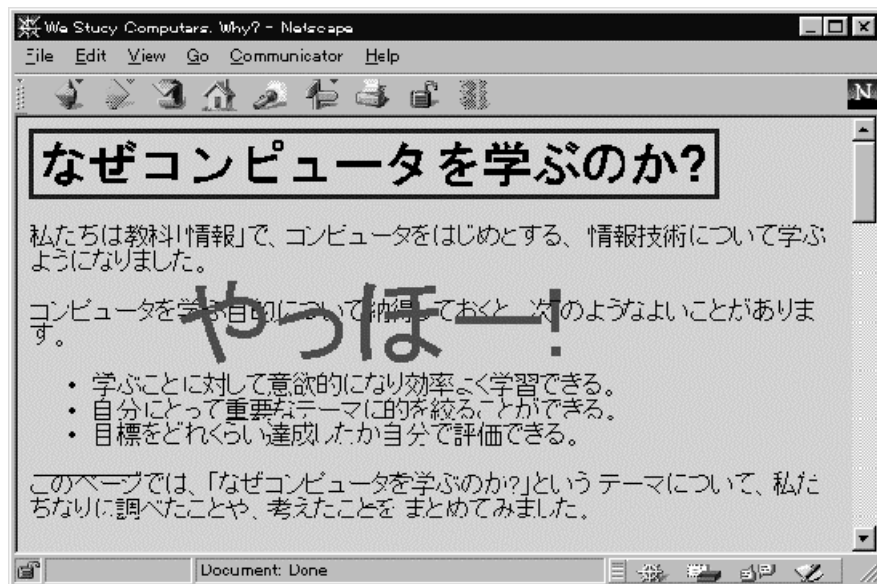


図 4.14: 要素を独立配置する