第 4 章 電子情報ネットワークへの情報発信

4.1 WWW プレゼンテーションの計画

4.1.1 WWW での情報発信への道のり

これまでに、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を見てきました。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決める
- 公開する情報をどのような形に構成するかを決める
- 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。

4.1.2 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。

例題:目的
「なぜコンピュータについて学ぶのか?」という疑問について、私たちが考えたことを多くの人に知ってもらう。

例題:範囲

- **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**

4.1.3 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うかも知れません。

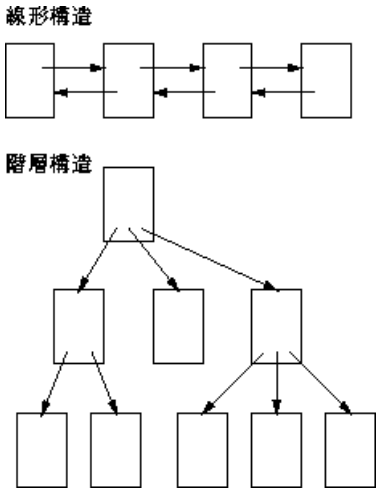

図 4.1 ハイパーテキストの基本的な構造

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 4.1)。

- ***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。**

メモ

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。

注意

図 4.1 の構造は、あくまでページ間の論理的な(互いに他のページを指しているリンクの結びつき方に基づく)構造です。各ページを構成するファイルをこの図のようなディレクトリ構造に配置する必要はありません。ファイルはすべて 1 つのディレクトリに入れておいてもよいのです。

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 4.2)。

例題:構造

- **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。**

メモ

将来、議論などの内容が増えたり、各教科内容の詳しい説明が必要になったりしたら、これらも別のページに分けて行くことができます。このように、階層構造は柔軟に構成を手直しできるという特徴を持っています。

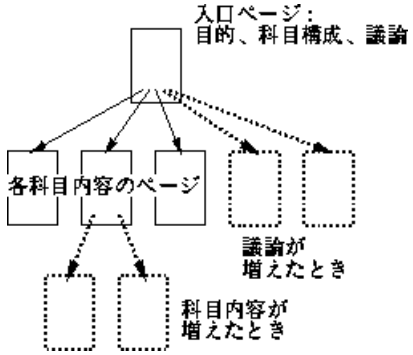

図 4.2 例題プレゼンテーションの構造

4.1.4 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とおり決めること。

4.1.5 よい WWW ページの設計とは

回線の種類、機械の種類、OS の種類、ソフトウェアの種類などが異なる、世界中のいろいろな環境にいる人が Web ページを見ています。どんな人でも等しく情報提供が受けられるような Web ページを目指すには、読み手の環境の違いを意識した HTML を書く必要があります。相手の画面の大きさや、相手の画面で使用可能な色数、他に必要なソフトウェアなどについても、Web ページを見るために必要な条件はなるべく少なくしておくべきでしょう。以下にいくつかのヒントを挙げておきます。

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

入口ページの問題

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、入口ページの役割です。入口ページは、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になります。従って、入口ページは「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。

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。

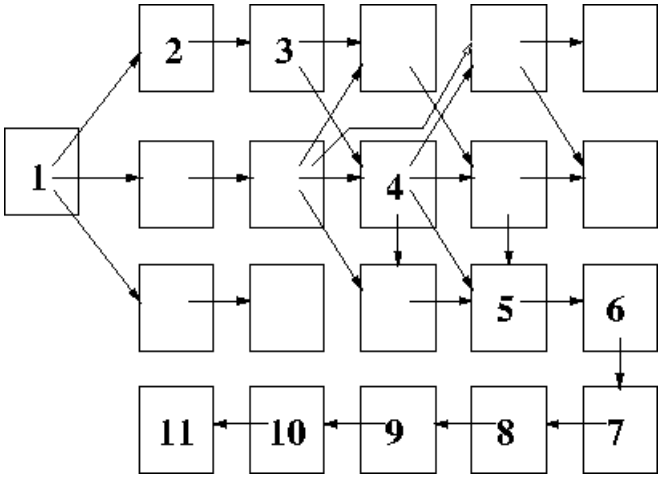

図 4.3 深すぎるリンク

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。

演習

好きなテーマを 1 つ選び、検索サービスを利用してそのテーマを取り上げている WWW プレゼンテーションを複数探して、それらを比較してみよ。それらはどのように違っているか? それぞれのよい点、悪い点は何か? 表の形にまとめてみよ。

4.2 HTML による WWW ページの記述

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、

ディ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。人間どうしで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、C や 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得られるのです。

4.2.2 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。

> **メモ**
> 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」でははく「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。

4.2.3 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 行目は ***DOCTYPE 宣言***と呼ばれ、続く記述が HTML 4.0 という規格に従っていることを示します。ほかに HTML 2.0、HTML 3.2 などもありますが、HTML 4.0 が HTML 規格の最新版です。

```
<!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。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。

4.2.4 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。

例題: 最初の HTML **コード**

```
<!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。

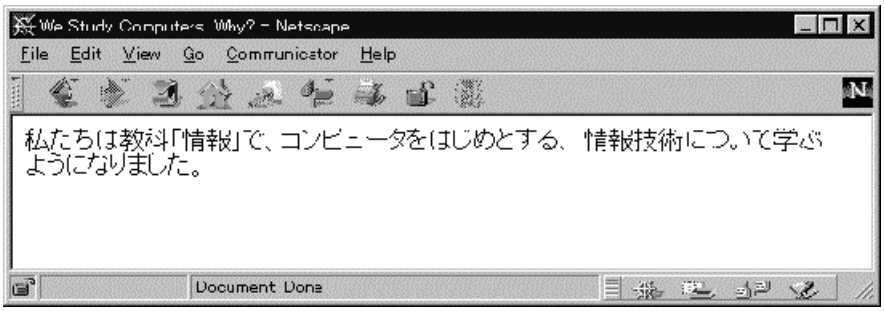

図 4.4 表題と段落

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 ファイルと違っている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。

メモ
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。

4.2.5 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、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<HR>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。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。

例題：見出し

```
<!DOCTYPE HTML PUBILC "-//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>
```

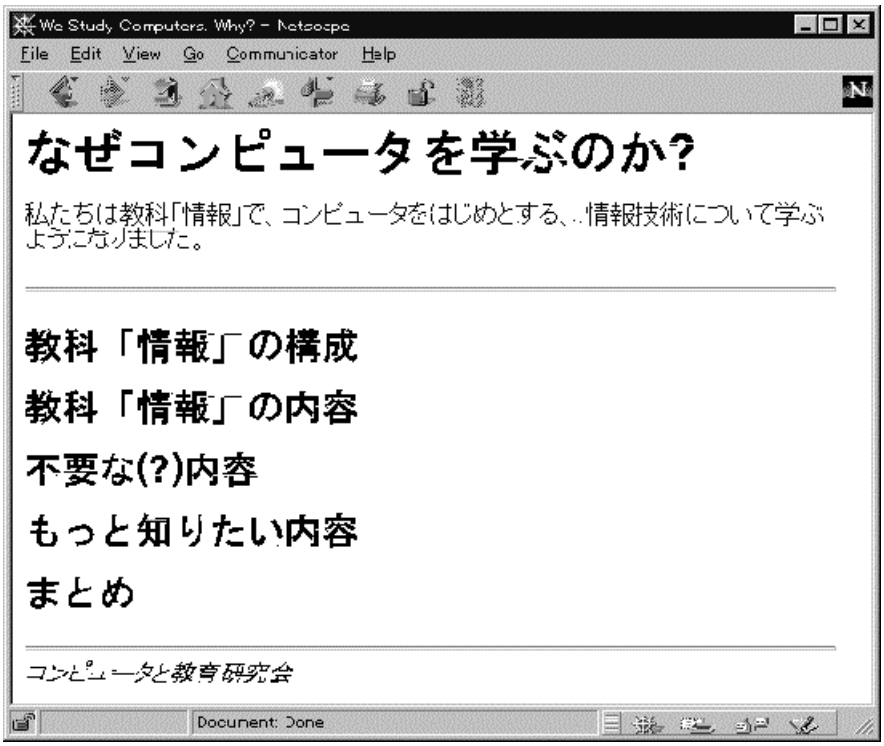

図 4.5 見出しと横線

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。

4.2.6 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きながら、HTML の機能を学んで行きます。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>
```

LI 要素(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。 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)。

例題: 箇条書き

```
<P>コンピュータを学ぶ目的について納得しておくと、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、私たちなりに調べたことや、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。

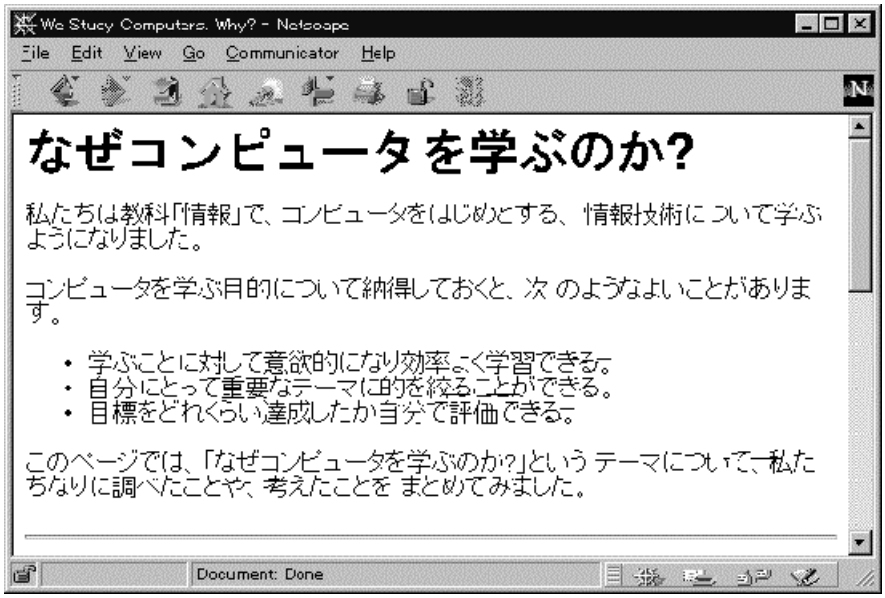

図 4.6 番号なしリスト

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、項目に 1 から順に番号が振られます(図 4.7)。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。

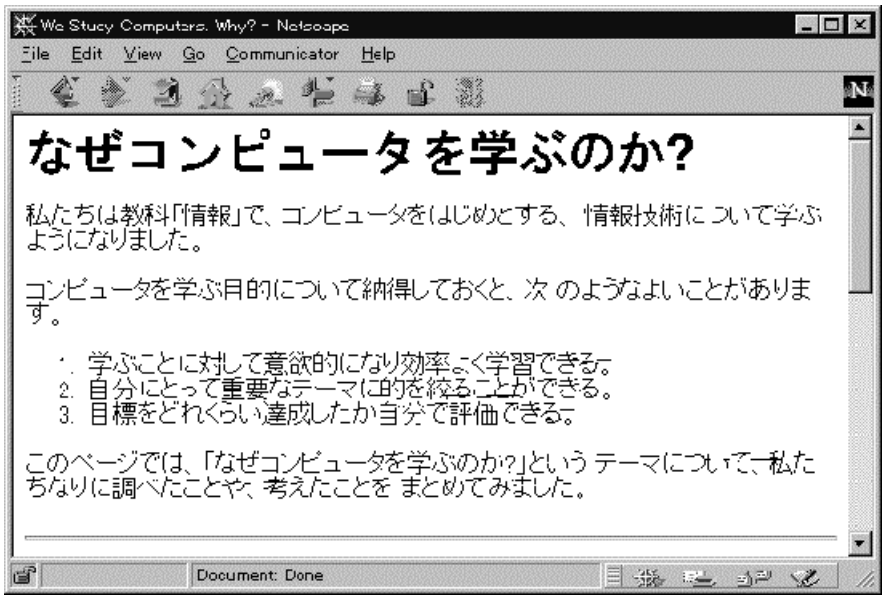

図 4.7 番号つきリスト

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。

```
<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**)。

例題: 箇条書き(2)

```
<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>コンピュータやネットワークを活用して、情報を選択・処理・発信
 できる力を身につける。
<DT>情報 B
 <DD>コンピュータの機能や仕組み、コンピュータ活用の方法について
 科学的に理解する。
<DT>情報 C
 <DD>ネットワークやコンピュータが社会の中で果たしている役割や影響
 を理解し、情報社会への参加のしかたを考える。
</DL>
```

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。

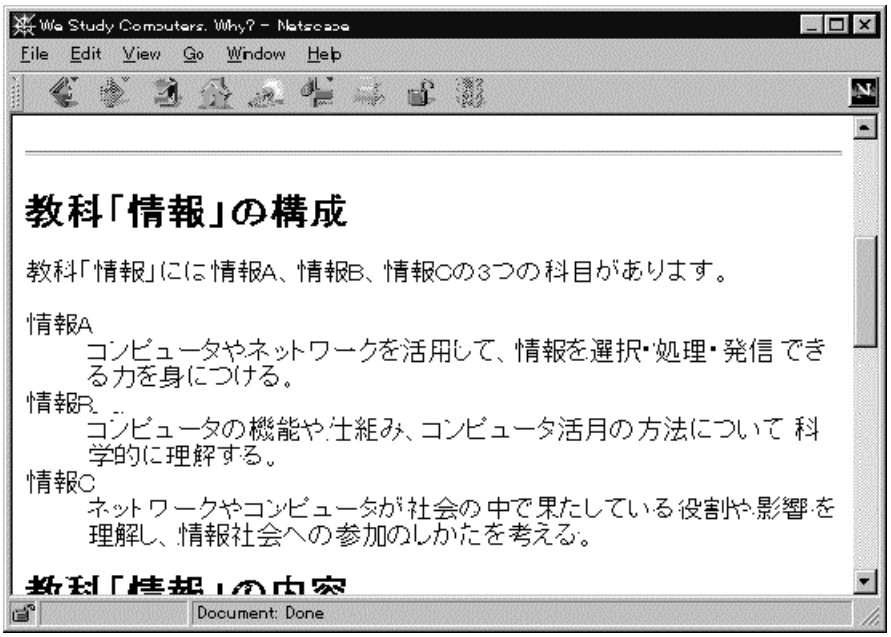

図 4.8 定義リスト

4.2.7 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。また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)。

例題：表

```
<TABLE BORDER=1>
<TR>
 <TH>項目
 <TH>概要
<TR>
 <TH>情報の表現
 <TD>ディジタル情報、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>
```

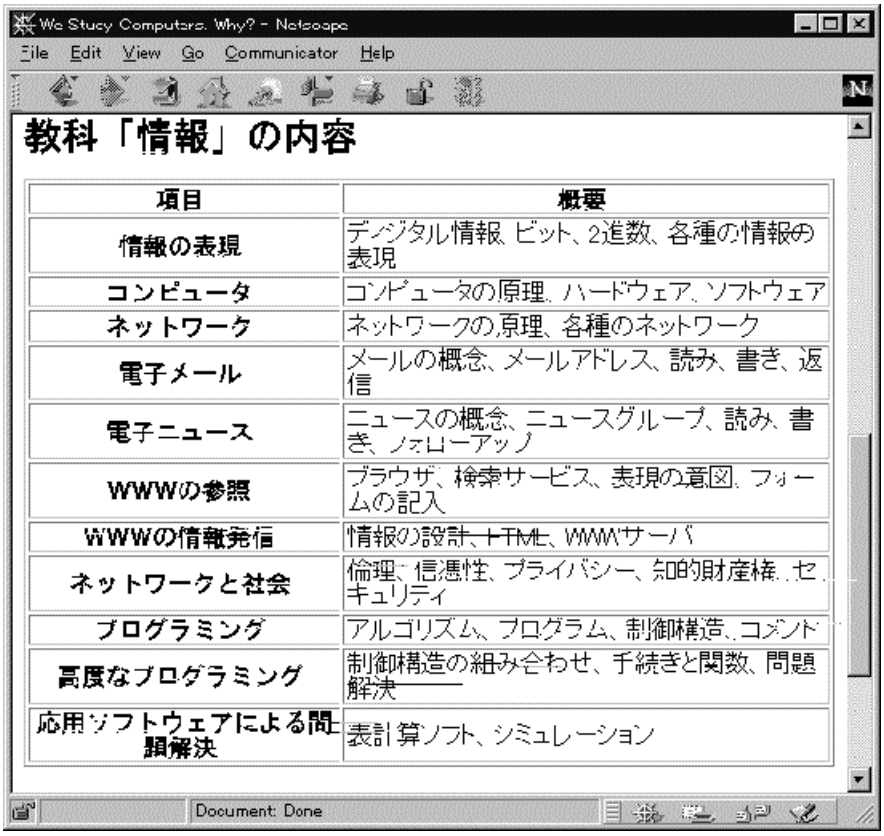

図 4.9 表

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。

4.2.8 インラインイメージ

ここまでは、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 のよいところは、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。

<IMG SRC="**ファイル名**" ALT="**説明**">

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。**

例題: インラインイメージ

```
<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。

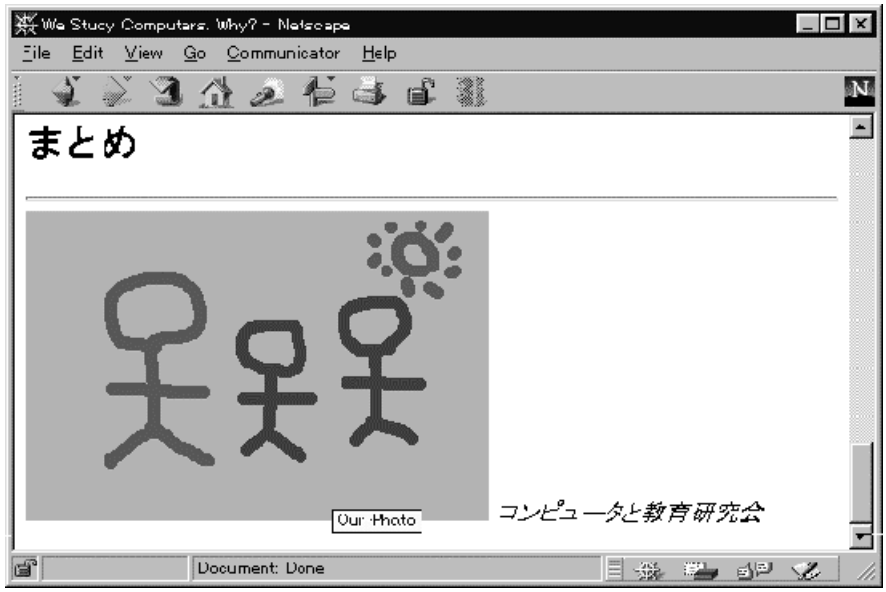

図 4.10 インラインイメージ

メモ
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。

4.2.9 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、これまでに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。**

例題: リンクテキスト

```
<TABLE BORDER=1>
<TR>
 <TH>項目
 <TH>概要
<TR>
 <TH><A HREF="info-repl.html">情報の表現</A>
 <TD>ディジタル情報、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。**

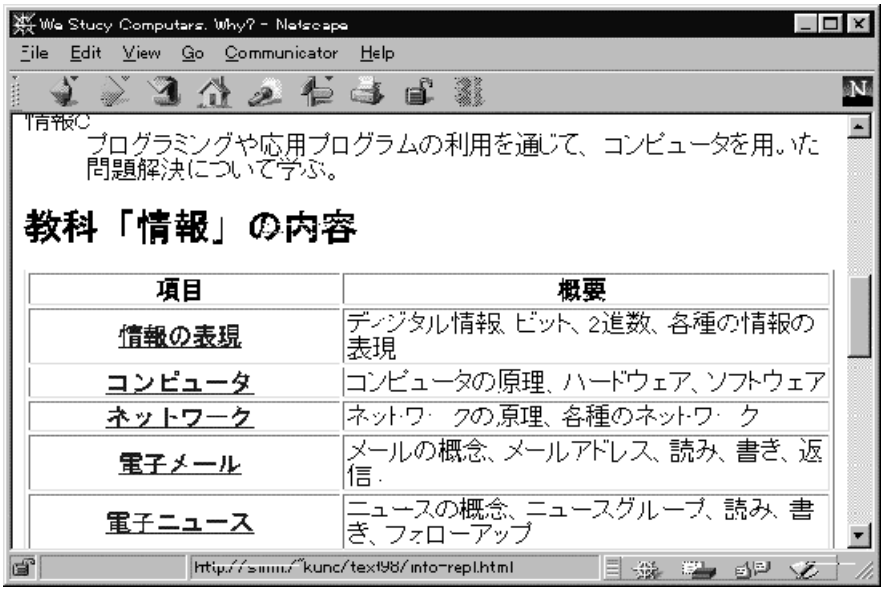

図 4.11 リンクテキスト

4.2.10 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できますし、(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。

例題: URL を指定したリンク

```
<!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。

4.2.11 整形ずみテキストと文字エントリ

ここまでに見て来たように、HTML ファイルの上での行かえは原則として意味を持たず、表示するときブラウザの画面幅に合わせて詰め合わせと行かえが行われ、その際に空白も取り除かれます。しかしこれでは、プログラムや詩のように勝手に空白や行かえを直して欲しくないものをページに入れるのにとても不便です。このため、<PRE>...</PRE> というタグに囲まれた範囲については空白や行かえをそのまま残すようになっています。たとえば次のようなぐあいです。

```
<PRE>
    昔むかし　浦島は
        助けた亀に　連れられて、
    龍宮城に　来てみれば
        絵にも描けない　美しさ。
</PRE>
```

これを表示したようすを図 4.12 に示します。ただし、 <PRE>...</PRE>の中は「何でもそのまま」というわけではなく、各種のタグは普通に使えます(リンクやインラインイメージなども入れられます)。

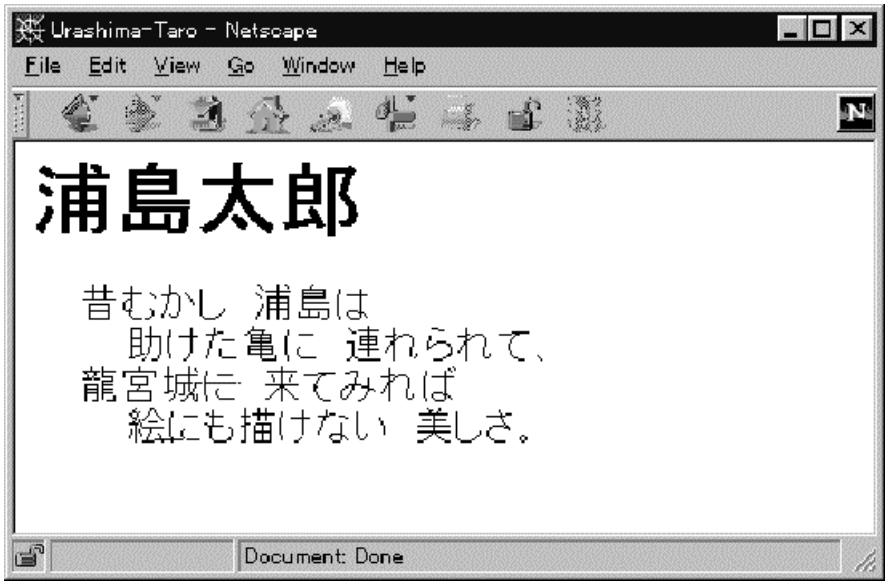

図 4.12 整形ずみテキスト

ただ自分の好きなところで行を変えたいときには、全部を <PRE>...</PRE>に入れる必要はなく、
というタグを使います。自動的な詰め合わせをしていても、このタグが書かれたところでは必ず行かえが起こります。

メモ
ところで、それではタグや属性の指定に使う文字「<」、「>」はどうやって入れればよいのでしょうか。HTML では、これらの文字はぞれぞれ「<」、「>」と書くことになっています(この書き方を HTML では***文字エントリ***と呼んでいます)。また、「&」も(文字エントリのための特別な文字として扱われるため)、WWW ページに入れるためには「&」と書く必要があります。

メモ
ブラウザによっては、文字エントリを使わずに「<」や「>」などを書いてもうまく表示されることがありますが、それはたまたまであって、別のブラウザで見ている人には見えないかも知れません。ですから面倒でも、上の 3 つの文字をページに入れる時は文字エントリを使って表してください。

演習
自分のページに整形ずみテキストと <、>、&を入れて見よ。

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。もっとも、普段から WWW サーバを実行するのと同じコンピュータで作業をしているとか、そのコンピュータのファイルをネットワーク経由で読み書きしている場合などは、特別なことをしなくても、ある決まった (公開用の) ディレクトリに HTML ファイルなどを置くだけでページが公開できることもあります。具体的な転送方法やファイルの置き場所はそれぞれのコンピュータ環境で違っています。

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ら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試すのがよい。

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

ここまでに学んだ 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(HTML 4.0)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。

メモ
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 HTML 4.0 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めがよいか、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">
```

やや繁雑だが、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 CSS 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 **4.13** に示します。

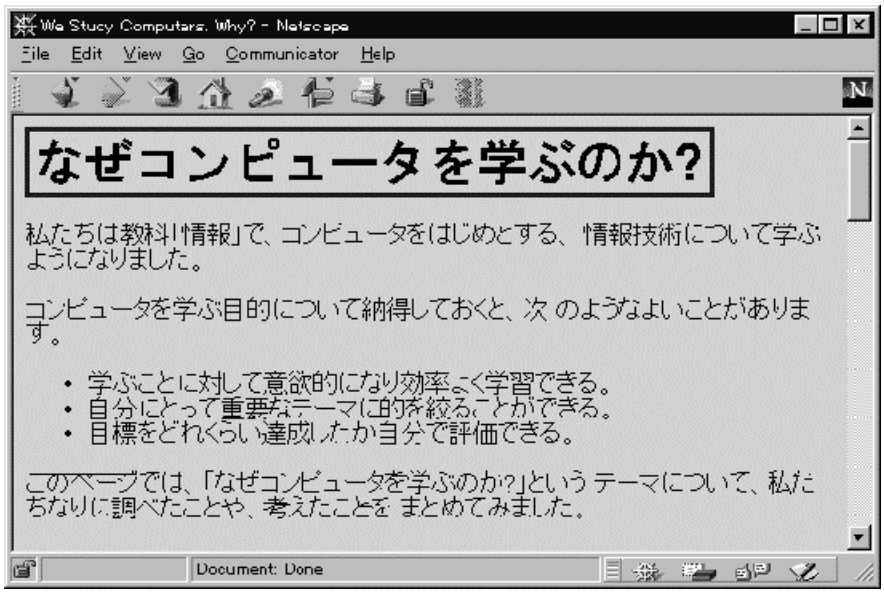

図 4.13 背景と枠のスタイル

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**{¥tt <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 --- ふちの幅。**
- 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 4.14 に示します。

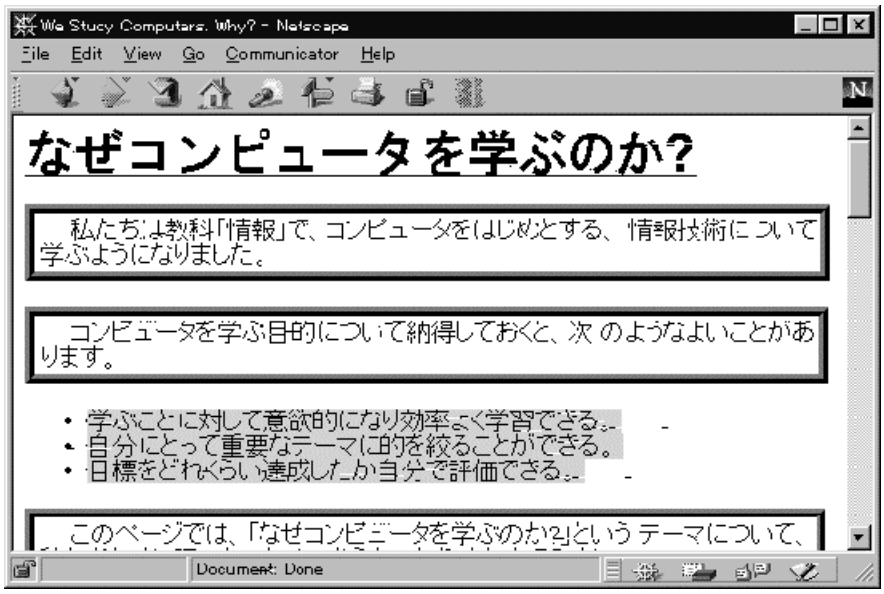

図 4.14 別のスタイル指定を適用

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る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るべきなのです。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。

- 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を拾うのにじゃま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。

4.4.6 個別的なスタイル指定

ここまでは、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。しかし場合によっては、「この H1 の見出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、次のようにするわけです。

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

SPAN はこのよう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。

4.4.7 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、hegith: 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。

たとえばスタイルとして

```
.overlay { position: absolute; left: 100px; top: 100px }
.overlay { 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 4.15 に示します。

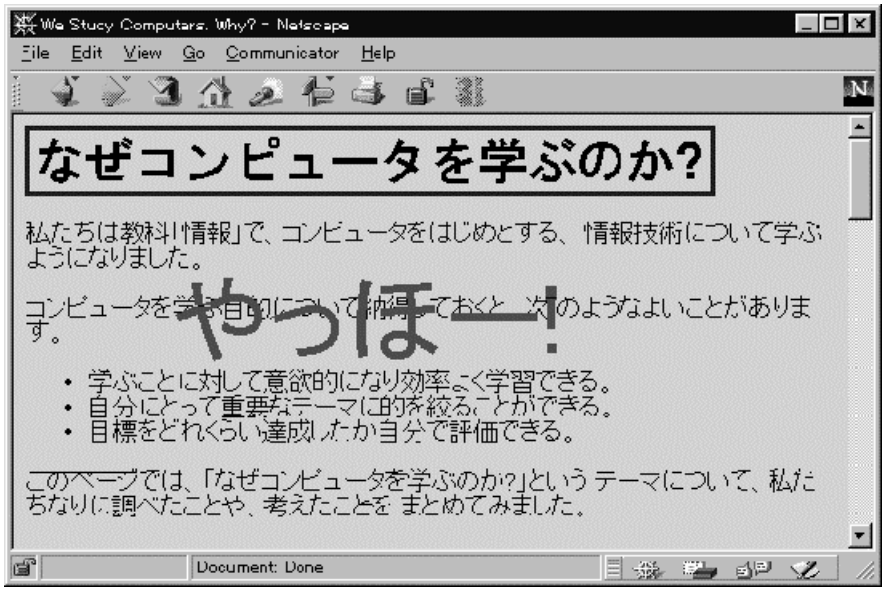

図 4.15 要素を独立配置する

4.5 スクリプト

4.5.1 スクリプトとは

コンピュータの動作がすべてプログラムによって定められることはすでに学びました。プログラムも人間が書いて計算機に読み込ませるので、ちょうど HTML や CSS のように人工言語を使います。これをプログラム言語と言いますが、ひとくちにプログラム言語といっても用途に応じてさまざまなものが作られ使われています。ここでは HTML ファイルの中に、画面上のボタンを押すと簡単な計算を実行する、小さなプログラムを埋め込んで動かしてみます。このように簡単な動作をすぐ書けることをめざして作られたプログラム言語のことを***スクリプト言語***、そのような言語で書かれた簡単なプログラムのことを***スクリプト***と呼びます。ここでは ***JavaScript*** という名前のスクリプト言語を使っています。

4.5.2 例題 1: ボタンを押した回数を数える

最初の例題として、ボタンを押した回数を数えるスクリプトを見てみましょう。この例題では図 **4.16** のような WWW ページを作り、「押して!」と書かれたボタンを押すたびに、回数欄の数が 1 ずつ増えて行きます。

図 4.16 ボタンを押した回数を数える

まず、HTML の部分から見て行きましょう。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
<HEAD>
<TITLE>Script Sample #1</TITLE>
<SCRIPT LANGUAGE="JavaScript">
```

```
※スクリプト
</SCRIPT>
</HEAD>
<BODY>
<H3>例題 1: ボタンを押した回数</H3>
<FORM>
<P>押した回数: <INPUT TYPE=TEXT VALUE=1 SIZE=5><BR>
    <INPUT TYPE=BUTTON VALUE="  押して!  " ONCLICK="keisan()"></P>
</FORM>
</BODY>
</HTML>
```

これまでに学んでいないタグがいくつか出てきています。

- <SCRIPT>...</SCRIPT>は、この中にスクリプトのプログラムを書くことを表す。LANGUAGE 属性でスクリプトを書くのに使うプログラム言語を指定する(上の例では JavaScript を指定している)。このタグは<HEAD>...</HEAD>の内側に書く。
- <FORM>...</FORM>は、この間が 1 つのフォーム(入力欄やボタンなどの集まり)であることを表す。ここでは回数表示のために入力欄、回数を増やす動作のためにボタンを使うので、これらを含んだ範囲を 1 つのフォームにしている。
- <INPUT TYPE=TEXT VALUE=1 SIZE=5>は、幅 5 文字ぶんの入力欄を作り、初期値として 1 を入れることを意味している。
- <INPUT TYPE=BUTTON VALUE="..." ONCLICK="...">は、ボタンを作ることを意味している。ボタンの中には VALUE 属性で指定したものがラベルとして表示され、ボタンが押されると{¥tt ONCLICK}属性で指定された動作が実行される。

なお、フォームの各要素を配置するには
などこれまでに学んだタグを使い、説明などを表示するのもこれまでと同じようにテキストを書けば済むことに注意してください。 さて、上の例では ONCLICK="keisan()"と指定していますが、こうするとボタンが押されたときにスクリプトの keisan という名前がついた命令列(関数)が実行されます。これは{¥tt <SCRIPT>...</SCRIPT>}の内側で定義するので(上で※となっている箇所)、見てみましょう。

```
function keisan() {
  field = document.forms[0].elements[0];
  field.value = parseInt(field.value) + 1;
}
```

まず、document というのはこのスクリプトのあるページ全体を表す名前です。そして、「.」は「〜に属している、」という意味を持ち、document.forms[0]は「このページにあるフォームの最初のもの」を意味します(コンピュータの上では、番号を 1 から順ではなく 0 から順につけることがあり、ここでもそうなっているのです)。このページにはフォームは 1 つしか入れませんでしたが、2 つ、3 つと入れた場合は document.forms[1]、document.forms[2]

などと書くことでそれらを指定できます。さらに、document.forms[0].elements[0]だと「このページにあるフォームの最初のものの、最初の部品」という意味になりますから、これは入力欄を意味します。そして、field = ...というのはその入力欄を以後 field という名前で扱えるようにすることを意味しているのです。次の行で、さっそくその入力欄を扱います。field.value というのは、入力欄に入っている内容(文字のならび)を表します。そして、parseInt(...)というのは、...が表している文字を整数の値に変換するという意味です。「+ 1」は想像がつくように、その値に 1 を足し算します。そして最後に、field.value = ...で計算した値を入力欄の内容として入れ直します。つまり、これによって今までの入力欄の値はなくなり、代わりに計算した結果が入力欄の新しい内容になるのです。

演習
この例題を自分の WWW ページとして打ち込んで動かせ。

4.5.3 例題 2: 華氏を摂氏に変換する

では次に、もう少し役に立つ例題として、華氏の温度を摂氏の温度に換算するスクリプトを見てみましょう。このスクリプトを使うページのようすを図 **4.17** に示します。1 番目の欄に華氏の温度を入力して、変換ボタンを押すと、2 番目の欄に摂氏の温度が表示されます。

図 4.17 摂氏華氏変換

今度はもう分かるでしょうから、スクリプトも一緒に示します。HTML の部分は、入力欄が増えただけで前と同様です。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
<HEAD>
<TITLE>Script Sample #1</TITLE>
<SCRIPT LANGUAGE="JavaScript">
function keisan() {
   kashi = document.forms[0].elements[0];
   sesshi = document.forms[0].elements[1];
   sesshi.value = (5.0/9.0) * (parseFloat(kashi.value) - 32.0);
}
</SCRIPT>
</HEAD>
<BODY>
<H3>例題 2: 華氏摂氏変換</H3>
<FORM>
<P>華氏の温度: <INPUT TYPE=TEXT VALUE=0 SIZE=8><BR>
    摂氏の温度: <INPUT TYPE=TEXT VALUE=0 SIZE=8><BR>
    <INPUT TYPE=BUTTON VALUE="華氏→摂氏変換" ONCLICK="keisan()"></P>
</FORM>
</BODY>
</HTML>
```

スクリプトですが、さっきは整数の計算でしたが今度は少数点つきで計算したいので実数に変換する parseFloat(...)を使います。今度は欄が 2 つあるので、それぞれの欄を kashi、sesshi という名前で扱えるようにします。そして、3 行目で華氏の温度を実数にして、 32 を引いて 5/9 倍すると、それで摂氏の温度になります。なお、JavaScript をはじめ多くのプログラミング言語では割り算は「÷」ではなく「/」、掛け算は「×」ではなく「*」で表します。そして最後に、計算した結果を、欄 sesshi の値として設定すればよいのです。

演習
この例題を自分の WWW ページとして打ち込んで動かせ。

演習
もう 1 つボタンを追加して、摂氏を華氏に変換できるようにしてみよ (ヒント: keisan のほかにもう 1 つ関数を追加し、新しいボタンからはこの関数を呼ぶようにするとよい)。

4.5.4 例題 3: 最大公約数

3 番目の例題として、もっと込み入った計算を繰り返すものを見てみましょう。この例題では、図 **4.18** のように、2 つの数値を入れてその数値の最大公約数を求めます。

図 4.18 最大公約数

HTML **の部分は、またまた入力欄が増えただけで前と同様です。**

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
<HEAD>
<TITLE>Script Sample #3</TITLE>
<SCRIPT LANGUAGE="JavaScript">
function keisan() {
   fieldx = document.forms[0].elements[0];
   fieldy = document.forms[0].elements[1];
   fieldz = document.forms[0].elements[2];
   x = parseInt(fieldx.value);
   y = parseInt(fieldy.value);
   while(x != y) {
      if(x > y) {
         x = x - y;
      } else {
         y = y - x;
      }
```

```
    }
    fieldz.value = x;
  }
  </SCRIPT>
  </HEAD>
  <BODY>
  <H3>例題 3: 最大公約数</H3>
  <FORM>
  <P>X: <INPUT TYPE=TEXT VALUE=0 SIZE=8><BR>
     Y: <INPUT TYPE=TEXT VALUE=0 SIZE=8><BR>
     X と Y の最大公約数: <INPUT TYPE=TEXT VALUE=0 SIZE=8><BR>
     <INPUT TYPE=BUTTON VALUE="最大公約数を求める"
ONCLICK="keisan()"></P>
  </FORM>
  </BODY>
  </HTML>
```

スクリプトですが、今度は入力欄だけでなく、2 つの数値も x と y という名前で使えるようにしています。そして次の

```
while(条件) {
  ...
}
```

というのは、「指定した条件が成り立っている間、...の部分を繰り返し実行する」という動作になります。ここでは条件は「 x != y」ですが、これは数値 x と数値 y が等しくない間繰り返す、ということになるのです。 さて、繰り返す中身を見て見ると、

```
if(条件) {
  ... (a) ...
} else {
  ... (b) ...
}
```

となっています。これは、「指定した条件が成り立っていれば(a)の部分、成り立っていなければ(b)の部分を実行する」という動作になります。ここで「x > y」というのは見ての通り数値 x が{¥tt y}より大きい、ということですから、x が y より大きいときは x から y を引いたものを改めて x という名前で表せるようにします。逆に y が x より大きいときは y から x を引きます。 すると、この繰り返し部分は全体としては、「x と y が等しくない間、大きい方から小さい方を引くことを繰り返す」動作になります。ところで、次のことに注意しましょう。

・x=y なら、x と y の最大公約数は x である。
・x>y なら、x と y の最大公約数は x-y と y の最大公約数と同じである。
・x<y なら、x と y の最大公約数は x と y-x の最大公約数と同じである。

つまり、この繰り返しを行っている間、x、y の最大公約数は最初の x、y の最大公約数と変わっていません。ということは、最後に x と y が等しくなって繰り返しが終わったときも同じことが成り立っていますから、そのときの x の値がもともとの 2 つの数の最大公約数であり、これを表示欄に入れればよいわけです。

演習
この例題を自分の WWW ページとして打ち込んで動かせ。

演習
何でも好きな JavaScript プログラムを作成してみよ。