# 展開演習 A テキスト

梅山女学園大学現代マネジメント学部　三木　邦弘

2020 年 9 月 5 日版

# 1. はじめに

初めての授業をまさかオンラインでやることになるとは思いませんでした。3 月の後半はこのテキスト作成をやってました。授業の進行に合わせてテキストを作っていくつもりで、とりあえず最初の一月分ぐらいできていれば、大丈夫という感じでした。ですから今回送ったこのテキストはまだ一部です。まだまだ先があります。展開演習は、お二人とも 2 年生の「Web デザイン」や「プログラミング基礎」を受けていないので、そこをどのくらい早く通過できるかが鍵で、どこまで進めるかはやってみないとわからないなぁ、と考えていました。現在は、できなかった全部後期に回そう、と考えています。オンデマンド型の授業も、追いつけなくなったら遠慮なく再生を止めれば良いのだから、むしろ良いのではないか。周りに相談できないし、研究室まで泣きつきに行けないので、一層力が付くのではないか。バイトに走りたくてもバイト先が休業で時間もあるだろう。と明るく前向きに考えています。

「基礎演習」では様々な情報技術に関するものを、体験することを目的としました。よって難しいところは避けました。「展開演習」では、情報技術の中でも現在広く使われ、これからも使われるものについて、実際に使って理解することを目指します。2 年生の「Web デザイン」や「プログラミング基礎」で学んだ Web 関連技術をさらに学び、新たにデータベースや IoT (Internet of Things) も取り上げます。

これからの情報システムの中心となるものは、**データベース**と **AI** でしょう。インターネットなどを利用して様々なところから情報が得られます。IoT も情報源の一つです。IoT をうまく使うことにより、今までネットワークでは得られなかった情報を手に入れることができるようになります。得られた情報を蓄積するのがデータベースです。従来のシステムやこれからも使われ続ける多くのシステムがこのデータベースに蓄積した情報を元に、簡単な計算処理を行っています。銀行の基幹システムではお金のやり取りを扱いますが、結局のところある口座の金額の数値を減らして、同じ数だけ別の口座の金額の数値を増やすと言うのが基本です。文字情報になると、そもそも計算できないので何もしません。Amazon で買い物をしたとします。初めてでなければ、送り先である自宅の住所は既に Amazon のデータベースに入っているでしょう。データベースから取り出された自宅の住所は、配送に使う箱の表に印字されるので、商品が自宅に届きます。このように文字情報は処理らしい処理もされないことが多いのが現状です。

AI が使えるようになって、データベースに蓄えた情報を元に**判断**ができるようになりました。簡単な判断であっても、大量にしなければならない場合は大変役に立ちます。どのような判断まで可能かは現在様々な領域で検討中です。今後新しい技法が考えられて判断可能な領域はどんどん広がるでしょう。その反面、これまでその判断を行っていた人は不要となります。銀行業務でも専門性の高いものとして、融資可能かどうかの判断と言うものがあります。個人や会社に対して、銀行がお金を貸しても良いかどうかの判断です。これまでは様々な観点から情報を収集し、過去の経験なども利用して判断していました。お金を返せないところにうっかり大事なお金を貸してしまうと、銀行は損をするからです。このような分野にも AI による判断が導入されています。中国では既に大規模に使われていますし、国内の銀行でもぼちぼち利用が始まっています。事務処理の軽減のための RPA (Robotic Process Automation) の導入のため大規模なリストラが行われていますが、銀行の専門職の人々も安泰ではありません。

# 2. HTML

Web サーバーはクライアント (利用者) からの要求に従ってデータを送ります。クライアントはもらったデータを表示するのですが、そのデータは HTML(HyperText Markup Language) と言う言語で記述されています。ここでは「Web デザイン」の授業で学んだ HTML の復習についてここでは取り扱います。私の「Web デザイン」の授業での HTML はかなり昔の素朴な頃の HTML を元にしています。ここでは最新の HTML ver.5 の内容の一部にも触れます。

参考文献の代わりに、参考になる Web ページを紹介します。「とほほの WWW 入門」(http://www.tohoho-web.com/www.htm) です。このテキストで扱う HTML、スタイルシート、JavaScript、PHP などが、このテキストよりずっと詳しく例付きで説明されています。

## 2.1 mars における Web ページの作り方

mars にリモートデスクトップで接続すると、「WWW」という名前のフォルダがあります。ここに例えば「test.htm」と言う名前のファイルを入れると、インターネット経由で以下の URL でこのファイルにアクセスできるようになっています。

```
http://mars.mgt.sugiyama-u.ac.jp/ユーザ ID/test.htm
```

よって何か Web ページを作成する場合は、mars にリモートデスクトップで接続して、メニューの「アクセサリ」にある「Kate」などを使用してファイルを作成すると言うのが一つの方法です。また Windows 上でファイルを作成して、コピペで「WWW」の中にファイルを移すという方法もありますが、修正をするたびにコピペが必要になります。ここでは「プログラミング基礎」で利用して、割と好評だった peditor の mars 用改訂版を利用することにします。これによって mars にリモートデスクトップで接続せずに、パソコンのブラウザでファイルの作成から動作確認までができるようになります。mars 用の peditor を起動するには次の URL をブラウザに入力してください。

```
https://mars.mgt.sugiyama-u.ac.jp/PE/
```

すると図 1 のような画面になりますので、mars を利用する際の ID とパスワードを入力して「Send」をクリックします。

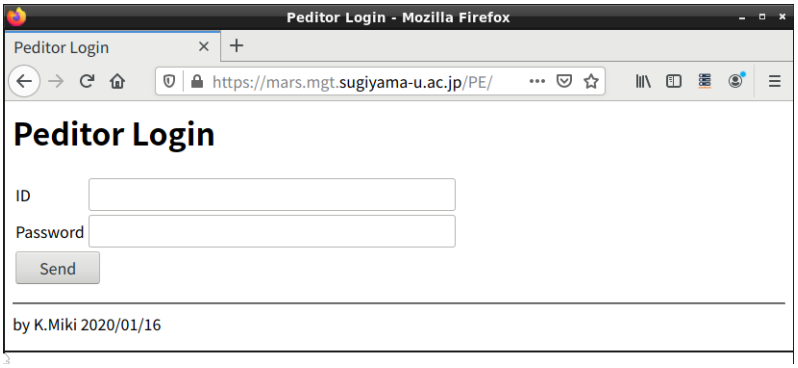



図 1 peditor: ログイン画面

すると図 2 のようなファイルの一覧の画面になります。最初の「./」は「WWW」ディレクトリ自身を示しています。次の「peditor.php」は peditor の本体なので消したり変更しないでください。「WWW」の下にサ

ブディレクトリを作成し、それを開いた場合は、「..」と言うものも出てきます。これは親ディレクトリを示しているので、これを「開く」ことにより一つ戻ることができます。以下ボタンについて上から順に説明します。

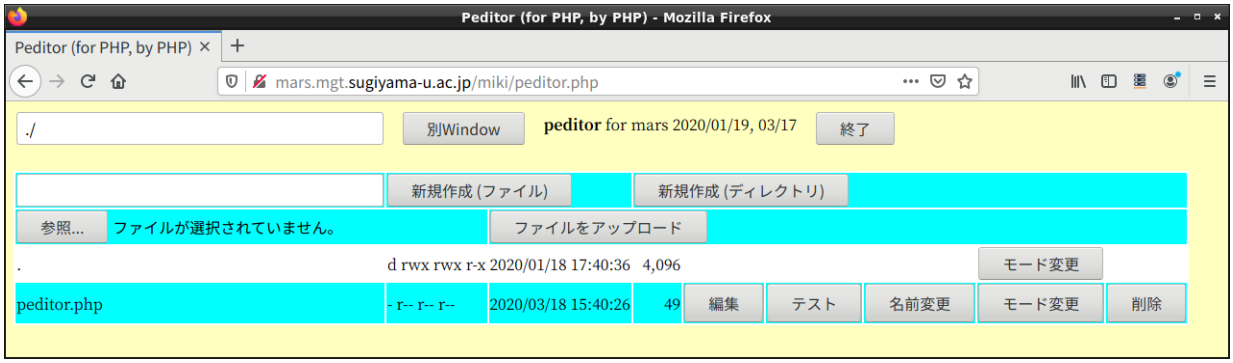



図 2 peditor: ファイルの一覧

- 別 Window: peditor をさらにブラウザの別のタブで開くことができます。
- 終了: peditor を終了します。
- 新規作成 (ファイル): 左側の入力欄にファイル名を入れてからこのボタンで新しいファイルを作成することができます。
- 新規作成 (ディレクトリ): 左側の入力欄にディレクトリ名を入れてからこのボタンで新しいディレクトリを作成することができます。
- 参照: パソコンにあるファイルを mars に送りたい場合、これで送りたいファイルを指定します。
- ファイルをアップロード: 「参照」のボタンで指定したファイルを mars に送ります。
- モード変更: ファイルの読み書きの設定をします。
- 編集: ファイルの内容を入力したり修正することができます。
- テスト: 別のタブでファイルを表示します。
- 名前変更: ファイルの名前を変更します。
- 削除: ファイルを消去します。

なお、「編集」ボタンの場合は図 3 のように画面が変わります。


図 3 peditor: ファイルの編集

- ファイル: ファイルの一覧にもどります。このとき編集中のファイルは保存しておかないと、せっかくの修正が消えてしまいます。

- 保存: 入力や修正した結果をファイルに保存します。「テスト」をする前も必ず先に保存をしてください。なお、修正をして保存をしていない場合は、ファイル名のところに「未保存」と表示されます。
- GO: PHP などのエラーメッセージで間違いのある行がわかった場合、すぐ左の入力欄に数字を入れてこのボタンをクリックすると、その行の先頭にカーソルが移動します。

## 2.2 HTML の簡単な例

例えば次のような内容を、peditor を使用して aaa.htm と言うファイルに入力してみましょう。

```
<HTML lang="ja">
<Head>
<Meta charset="UTF-8">
<Title>簡単な例</Title>
</Head>
<Body>
<H1>これはレベル 1 の見出し</H1>
<P>HTML の世界へようこそ。<Br>
これは 1 番目の段落です。</P>
<P>そしてこれは 2 番目の段落です。</P>
</Body>
</HTML>
```

HTML で記述した内容を入力してファイルを作成するのは、Windows に付属している「メモ帳」でも可能です。HTML で記述した内容は**「.htm」**または**「.html」**という拡張子の付いたファイル名にする必要があります。peditor で「保存」して「テスト」でこの内容をブラウザで見ると、見出しの所の字が大きくなっていたり、段落ごとに改行していたりします。

HTML は文章中に様々なマークアップタグ (markup tags) を挿入して様々な指示を行います。この例では、< >で囲まれた部分がそうです。<の次にはタグ名が続きます。タグ名には大文字と小文字の区別は無いので、<TITLE>の代わりに<title>と書いても構いません。タグ名が／で始まっているのはタグの有効範囲の終わりを示します。</XXXX>は<XXXX>の終わりを示しています。通常のタグは全て終わりのタグと対になって使われますが、例外も幾つかあります。

ブラウザで「ソース表示」をメニューで選ぶと、この HTML の記述をそのまま見ることができます。どうも思ったような表示が得られないときに確認するのに利用できます。

## 2.3 基本タグ

ここでは、先程の例にも出てきた基本的なタグについて説明します。

- 全体：ここの記述は HTML によるものだと言うことを示すものです。ファイルの最初と終わりに必ず入れます。

  ```
  <HTML lang="ja">
  ...   HTML での記述   ...
  </HTML>
  ```

  「lang="ja"」は内容が日本語であることも指定します。Firefox ではこの指定の有無でフォントが変わります。
- 設定：ページの設定のような事を記述している部分がここにあることを示します。

```
<Head>
...  表題などの記述  ...
</Head>
```

- メタ指定：一応タグの形をしていますが、内容は HTML より上位の設定です。様々な上位の設定がありますが、以下の例では文字コードの設定をしています。日本語の場合いくつかの文字コードが使われており、ブラウザは自動的に対応するようになっています。しかしたまに文字コードを誤って認識するために文字化けを生じます。mars では UTF-8 を使用していますので、文字化け防止のためにこれを最初に設定しておいてください。

```
<Meta charset="UTF-8">
```

- 表題：ページの表題を示すものです。通常ブラウザのタイトルバーの部分に、つまり本文とは別の場所に表示されます。また Web 検索システムなどはここに使われた単語を重視しますので、文章の内容を的確に示すものが望まれます。タグの形式は次のようなものです。

```
<Title>表題の文</Title>
```

- 本体：実際に表示されるページに関する記述はここの中にします。

```
<Body>
...  HTML でのページの記述  ...
</Body>
```

- 見出し：HTML は、1 から 6 までの 6 つのレベルの見出しが可能で、レベル 1 が一番大きな見出しになります。見出しとして指定された文は独立した左詰めの行として表示されます。タグの形式は次のようなものです。ただし、y の所は実際は 1〜6 の数字になります。

```
<Hy>見出しの文</Hy>
```

- 段落：何もタグの付いていない文章は、クライアント側の都合 (通常画面の幅) に合わせて詰め込まれます。段落として独立させたい場合には、段落に次のようなタグを付ける必要があります。

```
<P>文文文... 文</P>
```

- 強制改行：段落を示すタグ<P>を使用すると段落の間に空行が入ります。それを避け、単に改行をしたい場合には、次のようなタグを使います。

  **文文文**... **文**<Br>

- 番号なしリスト：この説明文のような • が先頭に付いた箇条書きをするためには次のようなタグを使います。なお、正確には「<Li>文章 1</Li>」のように閉じるタグも必要です。

```
<UL>
<Li>文章 1
<Li>文章 2
</UL>
```

→

- **文章** 1
- **文章** 2

<Li>が • になる感じです。<Li>の部分は幾つでも構いません。また 3 重までの入れ子にすることも可能です。

- 番号付きリスト：先頭に 1、2、3 と数字が順番に付いた箇条書きをするためには次のようなタグを使います。

```
<OL>
<Li>文章 1
<Li>文章 2
</OL>
```

→

1. **文章** 1
2. **文章** 2

今度は<Li>が数字になる感じです。<Li>の部分は幾つでも構いません。また 3 重までの入れ子にすることも可能です。

- 定義型リスト：言葉ではちょっと説明しがたいものですが、次のような形にしたいときにこれを用います。

**電子計算機**
　　**コンピュータのこと。**
**コンピュータ**
　　**かつて電子計算機と呼ばれたもの。パソコンの項を参照のこと。**

これは、次のような 3 種類のタグを使って記述します。

```
<DL>
<DT>電子計算機
<DD>コンピュータのこと。
<DT>コンピュータ
<DD>かつて電子計算機と呼ばれたもの。パソコンの項を参照のこと。
</DL>
```

- 引用文：引用などで通常の文章よりも行頭が右に凹んだ文章を記述したいときには、次のタグを使います。

```
<BlockQuote>
文文文... 文
</BlockQuote>
```

- 整形済み文章：既に整形が終わっていると言う事で、次のタグで囲まれた文章は入力したままの形で表示されます。つまり空白や改行が無視されませんし、勝手に改行されたりもしません。

```
<Pre>
              +- 炭水化物
三大栄養素----+- たんぱく質
              +- 脂肪
</Pre>
```

この場合画面に表示されるのは、<Pre>のタグが無いだけで後は全く同じものです。

- 水平線：画面一杯の水平線を引くタグは次のようなものです。

```
<Hr>
```

- コメント：文章の説明的なもので、表示されては困るものは次のようなタグを付けておきます。

```
<!--  文文... 文  -->
```

## 2.4 文字の修飾

通常の文字はそのままですが、< > & "の 4 文字は特殊な意味を持つためにそのまま使えません。それぞれ次のような形で記述します。

< → < > → > & → & " → "

表 1 のようなタグを付けることにより論理的な意味付けを文字に与えることが可能です。異なる意味付けのものはクライアントで色や書体の違いとして表示されます。

表 1 論理的な意味付け

| 使用例 | 説明 |
|---|---|
| <Dfn>定義された語</Dfn> | 通常イタリックで表示される |
| <EM>強調された語</EM> | 通常イタリックで表示される |
| <Cite>本等の表題</Cite> | 通常イタリックで表示される |
| <Code>プログラムなど</Code> | 通常等幅文字で表示される |
| <Kbd>キーボードのキー</Kbd> | 通常等幅の太字で表示される |
| <Samp>コンピュータの状態</Samp> | 通常等幅文字で表示される |
| <Strong>強調された語</Strong> | 通常太字で表示される |
| <Var>変数など</Var> | 通常イタリックで表示される |

また直接表 2 のように字体を指定することも可能です。残念ながら大抵の場合、漢字はイタリックにはなりませんし、等幅の効果もわかりません。

表 2 字体の指定

| | | |
|---|---|---|
| <B>太字 Bold</B> | → | **太字 Bold** |
| <I>イタリック Italic</I> | → | イタリック *Italic* |
| <TT>等幅文字 ijlmw</TT> | → | **等幅文字** `ijlmw` |

## 2.5 リンクの付け方

リンクの設定もやはりタグを利用して行います。また行き先はファイルだけでなく、指定した付近へという細かい指定も可能です。ただしその場合、行き先にタグで印をつけておく必要があります。

- ファイルへのリンク：これは次のような形式のタグを用います。

```
<A href="URL">クリックされる文</A>
```

URL の部分には実際にリンクするファイルの URL が入ります。「クリックされる文」の所はブラウザでは下線が付いてちょっと色が異なる表示がなされます。ここはリンク先が判るような文にします。実際は例えば次の様な形になります。

```
<A href="http://www.sugiyama-u.ac.jp/">椙山女学園大学のトップページ</A>
<A href="betu.htm">同じディレクトリにある betu.htm と言うファイル</A>
```

- ファイル内へのリンク：予め次の様なタグ (アンカー) を入れておくと、そこへ行くリンクを張ることが可能です。

```
文... 文<A name="namae">文</A>文... 文
```

namae は適当な語を使います。同じファイル中で同じ語は使えません。そしてリンクを張るときには次の様にします。

```
<A href="#namae">クリックされる文</A>
```

要するに先程指定した語の前に#を付けます。長めの文章で先頭の所に目次や索引を設けて、そこから後に続く文章の該当するところへリンクを張ると言う形でよく使われます。

この両者を同時に使う事も可能です。つまりファイルの中で予めアンカーを指定しておけば、リンクを張る側は、URL の後に#とアンカーで指定した語を書けば良いようになっています。

```
<A href="http://cc01.center.sugiyama-u.ac.jp/~mailbase/teacher/#mg">現マネ教員</A>
```

## 2.6 画像の指定の仕方

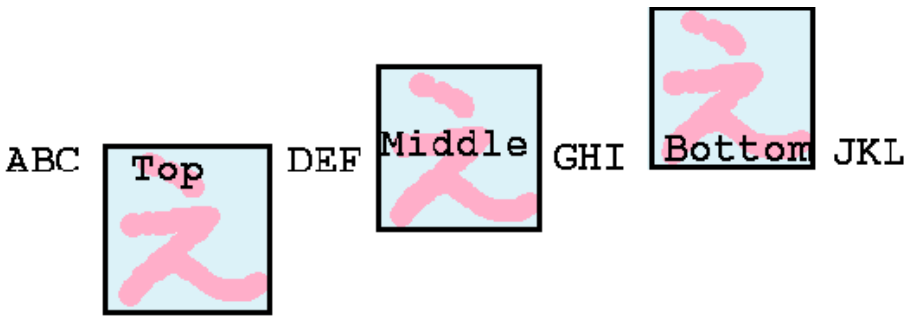
画像を取り込みたい場所に次のようなタグを挿入することにより、画像を表示することができます。

```
<Img src="画像ファイル名">
```

すると一つの画像が一つの文字と同様に扱われます。ところが通常画像は文字よりも大きいので前後の文字と画像のどこを合わせるかでかなり違ったものになります。そこで、

```
<Img src="画像ファイル名" align="Top">
```

とすると前後の文字に画像の上部が並ぶようになります。Top の所を Middle にすれば画像の中央が、Bottom にすれば下部が揃うようになります。




画像を見る事ができない場合や画像が表示されるまでに画像の代わりの文字列を出す事ができます。

```
<Img src="画像ファイル名" alt="画像の表題">
```

のように指定します。できるだけ画像にはこの指定を付けるようにして下さい。

画像ファイル名の所には URL を入れることも可能です。これによって他のページで使用されている画像を借りてくることができます。ただ大抵のページの作者は画像だけ使われるのは嫌うので注意して下さい。

同じページの中に表示しなくても良い場合には、

```
<A href="画像ファイル名">文</A>
```

も可能です。こうすると「文」をクリックすると画像が表示されるようになります。ページ内には小さく縮小したものを入れて、元の大きさを見たい人だけ選択すれば見えるようにすると良いでしょう。

また通常のリンクとの組み合わせも可能です。

```
<A href="URL"><Img src="画像ファイル名"></A>
```

とすると、表示された画像をクリックすると URL で指定したページが表示されます。

## 2.7 画面の背景の設定

画面の背景に色を付けたり、画像を敷き詰めたりすることが可能です。まず背景に色を付ける場合は、

```
<Body BGColor="色の指定">
```

のようにします。色の指定の仕方には次のようなものがあります。

- RGB 値を#FF0000 のように指定する方法があります。光の三原色である赤 (R)、緑 (G)、青 (B) の成分の強さを 00〜FF までの 2 桁の 16 進数でこの順番に記述します。00 が最小値で FF が最大値となり、00、01、02 と大きくなりますが、下の桁の数値を変えてもあまり違いはわかりません。一般に、値が大きいと明るい色、小さいと暗い色、RGB の値の差が大きいと派手な色、差が小さいと淡い色になります。
- 次の 16 色については名前で指定可能です。 Black、Gray、Silver、White、Red、Yellow、Lime(黄緑)、Aqua(水色)、Blue、Fuchsia(薄紫)、Maroon(えび茶)、Olive、Green、Teal(暗緑青色)、Navy、Purple

画面の背景の色を変更した場合、通常の文字の色では見にくくなることがあります。その場合には次のように文字やリンク場所の色を指定することも可能です。

```
<Body BGColor="black" Text="white" Link="red" VLink="yellow">
```

このようにすると画面の背景は黒、通常のテキストは白、リンクの部分は赤、そして一度選択されたことがあるリンクは黄色になります。

また画面の背景に画像を敷き詰める場合には次のような指定をします。

```
<Body Background="画像の URL">
```

指定された画像が画面より小さい場合は繰り返し敷き詰められる形になります。画像の内容に合わせて文字の色の設定も前述同様に行うことが可能です。ただし画像の内容がほとんど黒で文字の色を白にしたような場合、この背景用の画像が正しく転送されなかった場合には、まったく読めない画面になってしまう恐れがあるので、ほぼ同等の色を BGColor で同時に指定しておくと良いようです。

## 2.8 文字の修飾

ワープロなどと同様に文字の大きさを変更したり、色を付けたりすることができます。しかしあまりこのような修飾を乱用するとかえって見にくいものになってしまいます。本当に必要な所に適切なものを使ってください。

- **文字の大きさの設定**：n の所には 1 から 7 の数字が入ります。数字が大きいほど大きな文字になります。

```
<Font size="n">文文文</Font>
```

- **文字の色の設定**：「色」の所には先ほどの背景の色と同じ形式の指定 (#で始まる 16 進数によるものか色の名前にもの) が可能です。また上記の大きさの指定である size=も同時に指定することもできます。

```
<Font color="色">文文文</Font>
```

- **肩付き文字の指定**：$y = x^2$ の 2 のように文字の高さの半分上に字を出したいときに使います。

```
<Sup>字字字</Sup>
```

- **下付き文字の指定**：$H_2O$ の 2 のように文字の高さの半分下に字を出したいときに使います。

```
<Sub>字字字</Sub>
```

- **中央揃えの指定**：文字を行の中央に表示させたい場合に使います。

```
<Center>文文文</Center>
```

## 2.9 表の指定

表の指定は、通常の表を示すためと、単に大きさの違うものを綺麗に並べて表示するためによく用いられます。次の画面分割以上にタグがごちゃごちゃと大量に出てきますので混乱しないようにしてください。

1. 表の指定全体を<Table>と</Table>タグで囲みます。なお後に出てくる表の中身として表を用いることも可能ですが、その場合は<Table>が入れ子になることになります。
   表の罫線が必要な場合は、<Table Border>とします。Border を省略すると枠線がない表になりますが、単に整列させたい場合によく用いられます。さらに「Border="**数値**"」とすると枠線の太さを変えることができます。大きな数値にすると太い枠線になります。
2. 表に表題を付ける場合は、<Table>タグのすぐ後で次のような指定をします。

   ```
   <Caption>表の表題</Caption>
   ```

   ここで指定した表題は表の上に中央寄せされて表示されます。表の下に出したい場合には、
   <Caption align="Bottom">とします。
3. 各行の内容はそれぞれ、<Tr>と</Tr>で囲う必要があります。
4. 行に含まれる各セルはそれぞれ、<Td>と</Td>で囲う必要があります。

- セルの内容が複数行にわたる場合は、改行すべき所に<Br>を入れます。
- 横隣のセルと合体した横長いセルを作成したい場合には、<Td ColSpan="2">のようにします。(3にすれば 3 つ連結した形になります。)
- 下のセルと合体した縦長のセルを作成したい場合には、<Td RowSpan="2">のようにします。(3 にすれば 3 つ連結した形になります。)

以下に簡単な表の例を示します。

```
<Table Border>
  <Caption>表のサンプル</Caption>
    <Tr>
      <Td>aaaaa</Td><Td>bbbbb</Td><Td>ccccc</Td>
    </Tr>
    <Tr>
      <Td>ddddd</Td><Td>eeeee</Td><Td>fffff</Td>
    </Tr>
    <Tr>
      <Td>ggggg</Td><Td>hhhhh</Td><Td>iiiii</Td>
    </Tr>
</Table>
```

→

表のサンプル

| | | |
|---|---|---|
| aaaaa | bbbbb | ccccc |
| ddddd | eeeee | fffff |
| ggggg | hhhhh | iiiii |

### 表の指定の追加

表に関してはさらに次のような指定が可能です。

- <Td>の代わりに<Th>を使用することにより、このセルは表の見出しであることを示せます。この時このセルの内容が太字でかつセンタリングされて表示されます。
- <Td nowrap>と指定するとセルの内容がブラウザの表示幅に合わせて改行されないようになります。ただこれを乱用すると表が画面からはみ出して見にくくなります。
- <Td align="right">と指定するとセルの内容が右詰になります。同様に<Td align="center">と指定するとセルの内容が中央に揃います。
- <Td valign="top">と指定するとセルの内容が上に寄せられます。同様に<Td valign="bottom">と指定するとセルの内容が下に寄せられます。
- nowrap、align、valign などの指定は自由に組み合わせて使用できます。

## 2.10 画面の分割 (フレーム)

画面を分割することによって画面を目次とその内容のように分けて、いちいち利用者が目次のあるページの戻らなくても良いようにできます。ただこれを利用する場合は、分割の仕方を指定するファイルとその分割された内容のファイルの両方が必要になります。後者は 2 分割ならば 2 つ、3 分割ならば 3 つ必要になりますが、これらはこれまで説明してきた普通の HTML で記述された内容のものです。

1. 画面分割の指定のファイルは<HTML>で始まり</HTML>で終わるところと、その次に<Head>の部分がある所はこれまでのものと同じです。
2. 通常ならば<Body>と</Body>が来るところに、<Frameset>と</Frameset>が来ます。<Frameset>の中で画面の分割の指定をします。分割の方向としては上下と左右が可能です。画面を上下に分割する場合は、後の例のように Frameset の中で rows=を指定します。「20%,*」とすれば画面が上下に 20% と残りに分けられます。「33%,33%,*」とすればほぼ同じ高さに上中下と分割されます。左右に分割した

い場合は cols=を指定します。

3. <Frameset>のタグの間には、次に 3 種類のものが入ります。
   - <Frame>を用いて分割された画面に表示する内容が入ったものを指定することができます。
     <Frame src="URL" name="**フレーム名**">
     URL の部分に表示する内容の入ったファイル名を、「フレーム名」の所には分割された画面を識別するための適当な名前を入れます。
   - <Frameset>を入れるとさらに画面を細かく分割することができます。
   - <NoFrames>から</NoFrames>の間に、このような画面分割に対応していないブラウザの利用者に対するメッセージを記述するのに使用します。よくあるのが、「このページは Netscape Navigetor ver.2 以降か Internet Explorer ver.3 以降でご覧ください。」と言う感じのものです。また最近では i–mode 端末向けのメッセージをここに入れてある例もあります。内容の前後は<Body>と</Body>タグで囲う必要があります。

以下に実際に Frameset を使用した例を示します。

```
<HTML>
<Head>
  表題など
</Head>
<Frameset rows="20%,*">
  <Frame src="frame1.htm" name="FRAME1">
  <Frame src="frame2.htm" name="FRAME2">
  <NoFrames>
    <Body>
      フレーム未対応ブラウザ用メッセージ
    </Body>
  </NoFrames>
</Frameset>
</HTML>
```

→


これを画面分割に対応したブラウザで見ると、右上の図のように、画面が上下に分割されます。上の部分が 20%、残りが下の部分となり、それぞれ frame1.htm の内容と frame2.htm の内容が表示されます。

さて一旦このように画面が分割されてしまうと以後それぞれ分割された画面の中で変化するようになります。つまり別のファイルにリンクを張った場合にそれを選択すると、今選択を行った画面の中にそのリンク先が表示されます。大抵はこれで構わないのですが、左の画面で目次を示して、選択された内容は右の画面に出ると言うような場合は困ります。

## 2.11 表示先の指定

通常の A タグによるリンク先の指定では、クリックするとこのタグが表示されていた画面に表示されます。ここで A タグに target=と言う指定を追加することにより、表 2.11 別の場所に表示することが可能になります。

例えば椙山のトップページを新しいウィンドウに表示するリンクは次のように記述します。

```
<A href="http://www.sugiyama-u.ac.jp/" target="_blank">椙山のトップページ</A>
```

表 3 target の設定

| | |
|---|---|
| target=”フレーム名” | 指定されたフレームに表示 |
| target=”_blank” | 新しいウィンドウを作成してそこに表示 |
| target=”_top” | 画面分割を全て解除して表示 |
| target=”_parent” | 画面分割を一つ解除して表示 |

## 2.12 Web ページの埋め込み

Frame タグを使用して画面を分割して複数のページを表示するほかに、iFrame タグを使用して、他の Web ページを埋め込む事も可能です。例えば埋め込みたい部分に次のような iFrame タグを入力します。

```
<iFrame src="http://www.sugiyama-u.ac.jp/" height="300" width="400">
   iFrame に対応していないブラウザ用メッセージ
</iFrame>
```

すると高さ 300、幅 400 の長方形の領域に椙山女学園大学のトップページが表示されます。長方形の領域よりもトップページの方が大きいので、スクロールバーが長方形の内側に表示されます。長方形の内側にあるリンクをクリックすると、フレームで分割された場合と同様に、長方形の内側だけ置き換わるようになります。

iFrame タグでは、表示する内容を示す src、領域の高さを示す height、領域の幅を示す width は必須ですが、それ以外にも次のような設定が可能です。

- scrolling: スクロールバーの表示の設定。”auto”を指定すると内容の大きさに応じてスクロールバーを表示します。scrolling の指定を行なわない場合はこの設定になっています。”yes”を指定すれば常にスクロールバーを表示し、”no”を指定すればスクロールバーの表示を禁止することができます。
- name: フレームの名前の設定。適当な名前を設定する事により、A タグの target を使用して、フレームの外にあるリンクで、フレーム内に表示する内容を指定することができます。

## 2.13 動画や音声の再生

WWW が数あるインターネットを利用した技術の中でも注目を浴びたのは、当初から画像への対応が容易であったことが理由として考えられます。しかしかつてのインターネットは回線の容量が十分ではなく、大きな画像を含むページはなかなか表示されない、などのトラブルもありました。やがてインターネットの回線容量が大きくなり、動画の伝送も可能になり、web ページにも動画を表示したいと言う要望も強くなり、Object タグや Embed タグなどが導入されました。

Embed タグはブラウザのプラグインソフトを呼び出すためのものです。プラグインソフトを呼び出すことにより、音声や動画の再生が可能になります。プラグインソフトはブラウザの機能を拡張するためのプログラムで、これが入っていないパソコンもあります。またブラウザの設定で呼び出されるブラグインソフトが変わります。動画の再生ならば、QuickTime、Flash Player、Windows Media Player などのプラグインが呼び出されることが多いようです。

例えば、YouTube にある動画を埋め込んで再生できるようにする場合は、次のような感じになります。

```
<Embed src="http://www.youtube.com/v/7sFN6i3zhyA&hl=ja_JP&fs=1&rel=0"
   type="application/x-shockwave-flash"
   allowscriptaccess="always" allowfullscreen="true"
   width="480" height="385">
</Embed>
```

最初の行の http から&までが YouTube にある動画への URL で、&以下は動画再生時の設定です。3 行目にある allowscriptaccess などは、動画を再生する Flash Player の設定です。2 つの設定は関連があります。Flash Player のプラグインが設定されていない場合への対処などは HTML の範囲を超え、かつかなり複雑なものになるので省略しています。

最新の HTML5 (HTML ver. 5) では、動画や音声再生用のタグが導入されたので、より簡単に動画や音声を再生することができます。ただブラウザによってはまだ HTML5 に十分対応していないとか、動画のあらゆる形式に対応している訳でもない、と言うような問題があります。スマートフォンは全て HTML5 に対応しているので、今後は HTML5 のタグを使用したものになっていくでしょう。

動画を再生する video タグの最小限の設定例は次のようになります。

```
<Video src="http://www.mgt.sugiyama-u.ac.jp/miki/rekishi.mp4" controls></Video>
```

これで再生の指示をするためのボタンなどが付いた四角い画像が表示され、再生の指示をすると src で指定した動画が再生されます。動画はこの例のようにどこかのサーバーにあるものでも良いし、同じディレクトリにあればファイル名の指定のみで十分です。contols を省略すると動画の再生を始めることができません。

src や controls 以外には、表 4 のような設定をすることができます。これらは src と同様に video タグの中で指定します。

表 4　video タグの設定

| | |
|---|---|
| autoplay | 自動的に再生を開始しますが、大抵ブラウザが開始させません。 |
| loop | ループ再生します。 |
| height | 表示する高さ |
| width | 表示する横幅 |

同様に音声を再生する audio タグがあり、最小限の設定例は次のようになります。

```
<Audio src="http://www.mgt.sugiyama-u.ac.jp/miki/asagohan.mp3" controls></Audio>
```

音声は目に見えないので、このタグを書いたところに図 4 のようなものが表示されます。

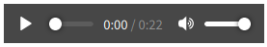



図 4　audio タグの表示

# 3. Style Sheet

ここでは Web ページの体裁を整えるために 1996 年末に導入された CSS (Cascading Style Sheet) について簡単に説明します。当初研究成果の自由な交換を考えて作られた WWW でしたがインターネットで広く使われるに従い、「より美しく」、「どのような環境でも作成者の意図したとおり」に表示させたいと言う要求は強くなりました。そのために当初は HTML のタグの拡張と言う形で行われてきましたが、情報内容の構造的なものを示すタグで行うのは適切でないと言うことになり、見た目 (Style) は Style Sheet で設定することになりました。

cascading は元々滝の流れが段々と落ちていく様と言うような意味ですが、ここでは上流で設定した style が下流にも伝わっていくと言う捉え方で良いでしょう。何が上流で何が下流かと言えば、HTML のタグは入れ子になっています。つまり<HTML>〜</HTML>の中に<Body>〜</Body>があり、その中にまた<H1>〜</H1>があったりします。外側が上流で内にあるものが下流です。よって<HTML>に指定した style が、<Body>や<H1>に引き継がれていくようになっています。

## 3.1 どこに入れるのか

後述の Style Sheet の様々な指定は 3 種類の入れ方があります。

- **別ファイルに入れる**：指定だけを別ファイルに入れて、それを読み込んで利用することができます。同じように形式を整えたい Web ページが複数ある場合にはこれが一番良いでしょう。なぜならば指定の入ったファイルを一つ直すだけで、それを取り込んでいる全てのファイルに形式の変更が及ぶからです。

  各 Web ページの Head のタグの間に次のような内容を入れます。

  ```
  <Link rel="stylesheet" type="text/css" href="ファイル名.css">
  ```

  そして指定は「ファイル名.css」と言う名前のファイルに入れます。
- **ファイル全体に対して指定する**：Head タグの間に次のような形で指定を入れます。その指定は Body タグの間全体に有効となります。

  ```
  <Style type="text/css">
  <!--
  スタイルの指定
  -->
  </Style>
  ```

- **個々のタグに対して指定する**：HTML のタグに対して指定することができます。この場合指定が及ぶ範囲は、そのタグの範囲に限られます。これは次のような形になります。

  ```
  <タグ Style="スタイルの指定">
  ```

これらの指定は任意に組み合わせて使用することができます。後のものほど有効範囲が狭くかつ優先されるので、最初のもので各 Web ページ共通部分を指定し、最後のもので個別の違いに対応と言った感じの使い分けをします。

## 3.2 基本的な形式

スタイル指定の基本的な形は次のようになっています。

```
セレクタ{属性:値}
セレクタ{属性:値; 属性:値; ・・・}
```

**セレクタ**は、スタイルを設定する対象 (各種タグ名、後述のクラスなど) です。**属性**は、対象のどの部分 (大きさ、色など) の設定をしたいのかを示し、**値**はその部分をどうするのか (100px、red など) を示します。属性と値の間は「:」なので「=」にしないように注意します。また複数の属性について指定する場合は、「;」で区切ります。具体例として、

```
H1{color: green}
```

によって H1 のタグで囲まれた文字の色を緑にすることができます。簡単な例を以下に示します。

```
<HTML lang="ja">
<Head>
<Meta charset="UTF-8">
<Title>Style のテスト</Title>
<Style type="text/css">
   H1{color: green}
</Style>
</Head>

<Body>
<H1>これは緑になる</H1>
<H1 Style="color: blue">これは青</H1>
<H1>またまた緑になる</H1>
</Body>
</HTML>
```

color 以外の属性としては、例えば表 5 のようなものがあります。○には数値が入ります。

表 5 style の属性の例

| 属性 | 説明 | 値 |
|---|---|---|
| background-color | 背景色の指定 | #RRGGBB または色の名前 |
| background-image | 背景画像の指定 | 画像ファイル名 |
| cursor | マウスカーソルの形状の指定 | crosshair、default、hand、move、text、wait など |
| font-style | フォントの書体の指定 | italic など |
| font-weight | フォントの強調の指定 | 100〜900(100 step)、normal、bold、bolder、lighter |
| font-size | フォント大きさの指定 | ○ px、○ pt、○ %、○ cm、○ mm、○ |
| text-decoration | テキストの装飾の指定 | none、underline、line-through など |
| text-align | 文字の位置の指定 | left、right、center |
| text-indent | テキストの字下げの指定 | ○ px、○ pt、○ %、○ em、○ cm、○ mm、○ |
| line-height | 行間の幅を指定 | ○ px、○ pt、○ cm、○ mm、○ |
| writing-mode | 縦書きか横書きの指定 | lr-tb (通常の横書き)、tb-rl (縦書き) |

なお、長さの指定の所で使用できる単位の意味は、表 6 のようになっています。

表 6 style の長さの例

| 単位 | 説明 |
|---|---|
| em | フォントサイズを 1 とした場合の相対指定 |
| % | フォントサイズや画面サイズを 100% とした場合の相対指定 |
| mm | ミリメートルで指定 |
| cm | センチメートルで指定 |
| in | インチで指定 (1in = 2.54cm) |
| px | ピクセル (画像の点) で指定 |
| pt | ポイントで指定 (1pt = 1/72in) |
| pc | パイカで指定 (1pc = 1/6in) |

ただ、いくらブラウザやパソコンが 1cm にしようとしてもディスプレイの大きさが違ってしまうとなんともなりません。よって em、%、px などが良く使われます。

## 3.3 クラス

HTML のタグにそれぞれ Style の指定が可能ですが、もう少し細かく指定することができます。つまり同じタグをクラス分けしてそれぞれについて異なる指定が可能です。例えば A タグで指定するリンクについては、表 7 のような 4 個のクラスがあらかじめ決められています。

表 7 A タグに設定されているクラス

| 名前 | 意味 |
|---|---|
| A:link | 未訪問のリンク |
| A:visited | 訪問済みのリンク |
| A:active | 選択中のリンク |
| A:hover | マウスが上にある時のリンク |

これによって A:hover に別のスタイル指定をすることにより、マウスがリンクの上に来ると形が変るようなものが実現できます。また通常のタグに対しても、次のようにしてクラス分けをすることができます。

```
セレクタ . クラス名{属性:値}
. クラス名{属性:値}
```

クラス名としては任意の英数字が使用できますが先頭は英字です。なぜかあらかじめシステムで決められたクラス名の前は「:」なのに、自前のクラス名の前は「.」になります。「.」の前にセレクタ名を指定しない場合は、どのようなタグでも使えるクラスになります。一方これを利用するタグの方では、

```
<タグ class="クラス名">
```

のように適用したいクラスの名前を指定します。このようなクラス分けのメリットとしては、例外的な指定もクラスとして指定することにより、統一化できると言うものがあります。以下の例では smallred というクラスを P や A で利用しています。

```
<HTML lang="ja">
<Head>
<Meta charset="UTF-8">
<Title>クラスのテスト</Title>
<Style type="text/css">
   P{color: green}
   P.special{color: blue}
   .smallred{color: red; font-size: 8pt}
</Style>
</Head>

<Body>
<P>これは緑色になる</P>
<P class="special">これは青</P>
<P>またまた緑色になる</P>
<P class="smallred">えらい小さい P</P>
<H1 class="smallred">えらい小さい H1</H1>
</Body>
</HTML>
```

## 3.4 位置の指定

より自由なデザインをする際には、文字や画像の表示する位置を自由に指定する事が必要になります。そのためには表 8 のようなスタイルが利用されます。

表 8 位置を設定するスタイル

| 属性 | 説明 | 値 |
|---|---|---|
| position | 位置の指定方法 | absolute (絶対位置指定)、relative (相対位置指定) |
| top | 画面の上端からの長さ | ○ px、○ pt、○ cm、○ mm、○ |
| left | 画面の左端からの長さ | ○ px、○ pt、○ cm、○ mm、○ |

position で absolute を指定した際には、top や left で指定した長さは次の図 5 のような意味になります。

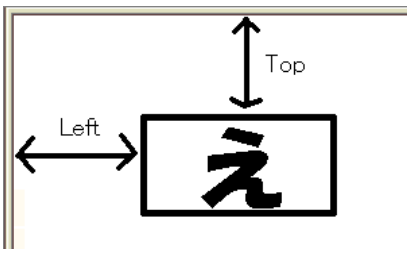



図 5 絶対的な位置の指定

実際に画像を上端から 100px、左端から 50px の位置に表示するには次のように指定します。

```
<Img src="http://www.ss.sugiyama-u.ac.jp/favicon.ico"
     style="position: absolute; top: 100px; left: 50px">
```

通常の文章を任意の位置に表示しようとする場合、文章の一文字ずつに位置の指定をする訳には行かないので、何かまとめるタグが必要となります。既に出てきた、P やセルが一つだけの枠が透明な Table でも良いのですが、通常は Div というタグでまとめて、このタグに位置を設定します。

```
<Div style="position: absolute; top: 400px; left: 200px">
    普通の文章。。。。。
</Div>
```

## 3.5 隙間の設定

HTML のタグは入れ子になっています。そうなると外側のタグと内側のタグとの間に隙間がないと見にくい場合があります。通常はブラウザが適当な大きさの隙間の設定を行っています。例えば Body のタグの中に Img タグによる画像だけを入れた場合、画像は左上にぴったりくっつくのではなく、少し隙間を空けて表示されます。隙間を空けたくない場合やもっと隙間を広くしたい場合は margin または padding の設定を行います。margin は指定したタグの周りに空ける隙間の設定です。padding は指定したタグの内側に空ける隙間の設定です。よって Body のタグの中にある Img タグによる画像の場合の画像の周りの隙間の設定は、Body の padding か Img の margin で行います。

margin と padding は、値として長さを 1 個〜4 個指定します。またある方向の隙間だけ指定したい場合は、少し名前の違う属性を使います。詳しくは表 9 をご覧ください。

表 9 隙間を設定するスタイル

| 使い方 | 説明 |
|---|---|
| margin: ○; | 上下左右に○だけ外側に隙間を空ける |
| margin: ○ △; | 上下は○だけ、左右は△だけ外側に隙間を空ける |
| margin: ○ △ ●; | 上は○だけ、下は●だけ、左右は△だけ外側に隙間を空ける |
| margin: ○ △ ● ▲; | 上は○だけ、下は●だけ、左は△だけ、右は▲だけ外側に隙間を空ける |
| margin-top: ○; | 上に○だけ外側に隙間を空け、他の外側の隙間については設定しない |
| margin-right: ○; | 右に○だけ外側に隙間を空け、他の外側の隙間については設定しない |
| margin-bottom: ○; | 下に○だけ外側に隙間を空け、他の外側の隙間については設定しない |
| margin-left: ○; | 左に○だけ外側に隙間を空け、他の外側の隙間については設定しない |
| padding: ○; | 上下左右に○だけ内側に隙間を空ける |
| padding: ○ △; | 上下は○だけ、左右は△だけ内側に隙間を空ける |
| padding: ○ △ ●; | 上は○だけ、下は●だけ、左右は△だけ内側に隙間を空ける |
| padding: ○ △ ● ▲; | 上は○だけ、下は●だけ、左は△だけ、右は▲だけ内側に隙間を空ける |
| padding-top: ○; | 上に○だけ内側に隙間を空け、他の内側の隙間については設定しない |
| padding-right: ○; | 右に○だけ内側に隙間を空け、他の内側の隙間については設定しない |
| padding-bottom: ○; | 下に○だけ内側に隙間を空け、他の内側の隙間については設定しない |
| padding-left: ○; | 左に○だけ内側に隙間を空け、他の内側の隙間については設定しない |

## 3.6 枠線のスタイル

枠線と言えば表ですが、表以外の要素の周囲にも枠線を描くことができます。例えば A タグに枠線を描けばボタンと同じような姿にすることができ、これをクリックすると通常の A タグと同様に指定した URL のページへ行くことができます。枠線のスタイル設定は、枠線の太さ、枠線の形状 (実線とか点線など)、枠線の色の 3 つに分かれていて、まとめて設定もできますし、個々に設定も可能です。またここでは触れませんが、角の丸い枠線、画像による枠線、影付きの枠線用のスタイルもあります。

枠線のスタイルである border は例えば次のように使います。

```
border: 1px solid gray;
```

これを設定したタグの周囲に、太さ 1px の gray 色の実線の枠線が表示されます。つまり太さ、形状、色が一度に設定されます。太さに関しては表 6 のものが使えます。形状に関しては solid (実線) 以外に none (線なし)、dotted (点線)、dashed (破線)、double (二重線) などがあります。A タグによるリンクを四角いボタン形にするならば、次のようにします。

```
A { border: 3px solid gray; padding: 5px; text-decoration: none; }
```

細かく上側の枠線だけ設定したい場合は border-top を、右側の枠線だけならば border-right、下側の枠線だけならば border-bottom、左側の枠線だけならば border-left を border の代わりに使用します。

枠線の太さだけを設定したい場合は、border-width を使用して太さだけ指定します。更に細かく上側の枠線の太さだけならば border-top-width、右側の太さならば border-right-width、下側の太さならば border-bottom-width、左側の太さだけならば border-left-width を使用します。

枠線の形状だけを設定したい場合は、border-style を使用して形状だけ指定します。更に細かく上側の枠線の形状だけならば border-top-style、右側の形状だけならば border-right-style、下側の形状だけならば border-bottom-style、左側の形状だけならば border-left-style を使用します。

枠線の色だけを設定したい場合は、border-color を使用して色だけ指定します。更に細かく上側の枠線の色だけならば border-top-color、右側の色だけならば border-right-color、下側の色だけならば border-bottom-color、左側の色だけならば border-left-color を使用します。

## 3.7 スマートフォンへの対応

現在の Web ページはパソコンからアクセスされるより、スマートフォン (以下スマホ) からのアクセスの方が多いところも少なくないでしょう。パソコンで使えるブラウザ (Web ページを見るためのソフト) は色々ありますが、どれを使用しても大体同じ上に見えます。一方スマホで同じページを見ると、何も対策をしていない場合、小さなスマホの画面にパソコンと同じものが縮小されて表示されます。最近のスマホは解像度が高いのでそれでも字が読めることもありますが、リンクやボタンなどはマウスでなく指先でタップしなければならないので、ちょっと困ります。スマホでは簡単に拡大表示できますので、指先に見合う大きさにまで拡大してからタップすれば良いのですが面倒です。

パソコンとスマホと両方に対応するためにはいくつかの方法があります。

- パソコン用とスマホ用を別々のファイルで作成し、何らかの手段で切り替える。
- 一つのファイルの中で、条件付きのスタイル設定を用いてパソコンとスマホに対応する。
- スマホ用に設定をいくつか追加して、スマホでもそれほど変な表示にならないようにする。

別々のファイルで対応する方が分かりやすいのですが、切り替える設定が必要ですし、内容の修正の場合に両方のファイルを修正しなければなりません。一つのファイルで対応する場合は、スタイルの設定が倍になるので分かりにくい内容になります。設定をいくつか追加する方式は一番お手軽ですが、スマホ用として十分見やすいものにならないこともあります。個人の Web ページでは、よほどデザインにこだわりがない限り最後のやり方が良いでしょう。会社でも、同じデザインを極端に追及するといくらでも費用がかかるので、ほどほどのところで諦めるのが一番だと思います。

以下 2 つのスマホ用追加設定と、さらにスマホだけでなく大画面ディスプレイでも同じ形になるように、全て相対的にサイズを設定する方法を紹介します。

### viewport の設定

viewport の設定は次のように Meta タグを用いて行います。

```
<META name="viewport" content="width=device-width, initial-scale=1">
```

Head タグの間にこれを入れます。パソコンのブラウザではこの設定は無視されます。スマホでは、画面の横幅が 320px となります。実際のスマホの解像度は 1,000px ぐらいありますが、一律に 320px のものとして表示されます。パソコンの表示と比べるとかなり大きく拡大されて表示されますので、文字は読みやすく、リンクやボタンなどがタッチしやすくなります。

### 画像の幅の設定

viewport の設定でかなりの部分は救われるのですが、逆に大きくなりすぎて困るのものも出てきます。例えば画像が大きくなりすぎてスマホの画面に入らなくなることがあります。スタイルとして次のような設定を行うと、画像の横幅が最大でも画面と同じ幅となるので、横にスクロールしないと何の画像か分からないと言うような事態を避けることができます。

```
Img { max-width: 100%; height: auto }
```

### 相対的なサイズの設定

表 6 にはありませんが、vw と vh と言う長さの単位があります。それぞれブラウザの表示幅の 1/100 と表示の高さの 1/100 を示します。これを用いて、

```
<HTML style="font-size: 3vw">
```

のように文字の基本の大きさを表示幅に応じて設定すると、大きな画面では大きな字、小さな画面では小さな字で表示されるようになります。画像などの大きさの設定にも em を使用して、全て基本の文字の大きさを基準にすれば、全体的な見え方を図 6 のように揃える事ができます。

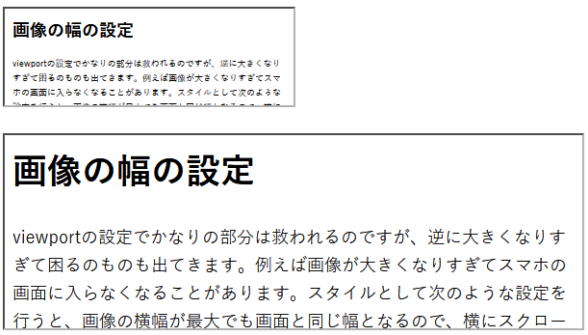



図 6 幅の違うブラウザでの見え方

## 3.8 クールな Web ページの作り方

Style Sheet を利用して Web ページのデザインを自由にできますが、なかなか見た目の良いクールな Web ページはできません。既にあるクールな Web ページを部分的に真似ても、かえって自分の手を入れたところ

が悪く目立ちうまく行きません。と言って完全に真似ると著作権や意匠権の問題を生じるでしょう。幸いなことに CSS フレームワークと呼ばれるスタイルのセットがいくつか公開されており、それを取り入れる事により簡単に Web ページが見た目の良いものに変わります。ここでは、Milligram[*1]と言う大変小さなフレームワークを紹介しますが、これ以外には同じく小さなフレームワークとして Pure-css[*2]、Ulkit[*3]や JavaScript なども含んでいる比較的大きな Bootstrap[*4]などがあります。

Milligram を使いたい場合、最小限の設定で済ますには、Head タグの中に次の行を追加します。これだけで図 7 のように変わります。

```
<link rel="stylesheet"
  href="https://cdn.rawgit.com/milligram/milligram/master/dist/milligram.min.css">
```

これによって常に最新のものが読み込まれますが、最新のものが常に良いとは限りません。仕様変更などのためにある日突然見た目が変わってしまうかもしれません。よって href に書かれているサイトから milligram.min.css をダウンロードして、次のようにそのファイルを取り込むようにした方が良いでしょう。

```
<link rel="stylesheet" href="milligram.min.css">
```

特に自分なりに Milligram が設定したスタイルを書き換えるような場合は元が変わると困ります。なお自分なりに Milligram のスタイルを書き換える場合、Milligram ではどのような設定をしているのか確認する必要が生じます。そのような場合は、同じサイトから milligram.css をダウンロードしてそちらの中身を見ましょう。通常名前に min の入ったファイルは、ファイルサイズ削減のため余分な空白や改行が全て削除されているので、人にとっては大変読みにくいものになっているからです。そして改変する場合は、この link のタグの後に改変後の内容のファイルを取り込むか、Style タグで記述します。同じタグや同じクラスに関するスタイルは、後から設定した方に置き換わります。元の milligram.css のファイルには手を入れない方が、将来改訂された milligram.css に乗り換える場合問題が起きにくく、どこを改変したか確認するのも容易です。

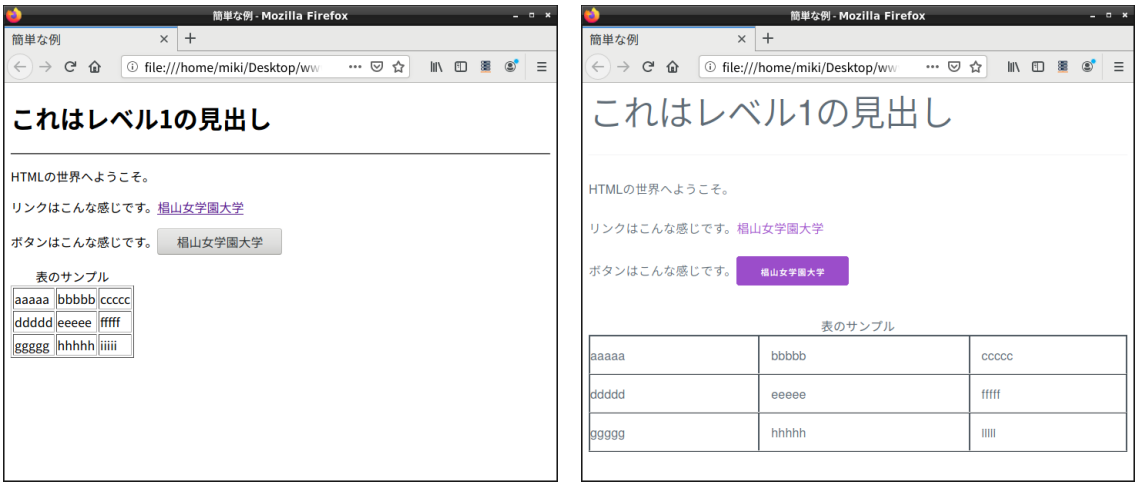



図 7　Milligram 使用前 (左) と使用後 (右)

*1 https://milligram.io/
*2 https://purecss.io/
*3 https://getuikit.com/
*4 https://getbootstrap.com/

# 4. JavaScript

HTML だけでは利用者ができることは自分の好きなリンクをクリックすることだけです。それ以外の利用者の入力に応じて変化する Web ページを作成するには、

- サーバー側で処理を行う。(CGI: Common Gateway Interface)
- ブラウザ側で処理を行う。(JavaScript、Java)

などの方法があります。CGI を行うにはサーバー側でプログラムを作成する必要があります。またよほど単純な事でない限りブラウザ側だけで片付くことはまず無いので、通常は CGI と JavaScript などが組み合わせて用いられます。Java は本格的なプログラミング言語なので、ここでは JavaScript を取り上げます。ブラウザ側で簡単な処理を行う例によってプログラミングの初歩の勉強も兼ねたいと思います。JavaScript は、C 言語に似ており、Object 指向の部分もかなりあります。

## 4.1 どこに入れるのか

JavaScript による記述はこれまでの HTML の入っていたファイルの中に含めます。Head のタグや Body のタグに囲まれた中に入れます。ファイルは最初から順番に読み込まれますので、関数の定義 (後述) 以外の部分は読み込まれた順に実行されます。関数は利用の前に定義されている必要があるので、関数の定義はよく Head タグに囲まれた中に入れられます。また表示されたときにすぐ実行される部分は Body のタグに囲まれた中に入れられます。

HTML の記述と JavaScript の記述が混ざらないように、JavaScript の記述は Script というタグに囲まれた中に入れます。さらに JavaScript に対応していない、つまり Script というタグを無視するブラウザの誤動作を防ぐために HTML のコメントのタグも入れるのが普通です。よって通常以下のような形で挿入します。

```
<Script type="text/javascript">
<!--
   JavaScript の記述
//-->
</Script>
```

1 行目と 5 行目は HTML のタグです。このタグの間は type で指定した形式で記述されている事を示します。ただこの Script タグを無視するブラウザのために、<!--と-->と言う HTML のコメントのタグが入れてあります。JavaScript は<!--は無視することになっています。-->は無視しないのでその前に//というJavaScript のコメントの指示が付いています。JavaScript では//以降行の終わりまではコメントと見なして無視するようになっています。最近ではまず JavaScript に対応していないブラウザには出会わないので、これ以降の例では 2 行目と 4 行目は省略しています。

複数の HTML のファイルの中で同じ JavaScript の記述を利用したい場合もあります。そのような場合は、スタイルシートと同様に別ファイル (例えば xxx.js) に JavaScript の記述を入れて、それを次のような記述で取り込むことができます。

```
<Script type="text/javascript" src="xxx.js"></Script>
```

## 4.2 画面への出力

JavaScript でブラウザの画面になにか出力したい場合は次のような文を利用します。

```
document.write(出力内容);
```

出力内容としては、”...”、’...’ や 3+5 のような式が指定できます。つまり出したい内容を「”」または「’」で囲います。JavaScript ではこのような「”」や「’」で囲まれたものを文字列と呼びます。また複数出したい場合はこれらを「,」で区切ります。出したい内容の中に HTML のタグを含めることもできます。

document はブラウザの中の一つの object(物) です。ちょうど今表示されているもの全体として捉えたら良いでしょう。この document に.write() をくっつける事により document に write をしてくれと依頼することになります。write() のように () が後に付くものはなんらかの動作を行う関数であることを示します。なお最後の「;」は文の区切りです。忘れないようにしてください。

例えば次のような記述を入れると、入れた場所に「3+5=8」と表示されます。

```
<Script type="text/javascript">
   document.write("3+5=",3+5);
</Script>
```

## 4.3 計算式

JavaScript では通常の四則演算に加えて剰余の計算が可能になっています。加減乗除に対してそれぞれ「+」、「-」、「*」、「/」が Excel などと同じように対応しており、剰余に対しては「%」を使用します。また () も使用可能なので、計算の順番を指定する際に使います。

ちょっと問題があるのは「+」です。通常の数値の加算以外に文字列の連結という意味があります。つまり 2+3 ならば 5 になるのですが、”2”+”3”とすると”23”になってしまいます。他の演算では文字列の形の数を計算しようとすると自動的に数値に変換されて、結果も数値で得られます。

## 4.4 JavaScript のエラーの探し方

以前のブラウザでは JavaScript にエラーがある場合、何行目に問題があると指摘してくれました。何が間違っているのかについては分かりにくい表示でしたが、何行目かは示されるのでその辺りを見直すことで間違いを発見することができました。最近のブラウザでは誤り発見のためのより高度な機能を提供してくれるようになりましたが、一方で通常の使用では何も表示されなくなりました。ここでは JavaScript の等のエラーを示してくれる Firefox の機能の一部を紹介します。

### ウェブコンソールの起動

Firefox には Web ページ作成のために様々な支援機能があります。その中の一つがウェブコンソールで、ここに現在表示されている Web ページの様々な問題点が表示されます。通常ウェブコンソールは表示されていないので、まずこれを起動する必要があります。問題のある Web ページを表示させたところで、図 8 のように「Alt」キーを押します。すると Firefox のメニューが表示されるので、「ツール」をクリックし、「ウェブ開発」の中の「ウェブコンソール」をクリックします。

するとウィンドウの下部に図 9。不要になったら右肩の×をクリックしてください。

### ウェブコンソールの見方

エラーがあるとその理由とその場所が表示されます。上記の例では「aaaa.htm:14:10」となっているので、aaaa.htm と言うファイルの 14 行目の 10 文字付近に一つ目の問題があることがわかります。二つ目のエラー

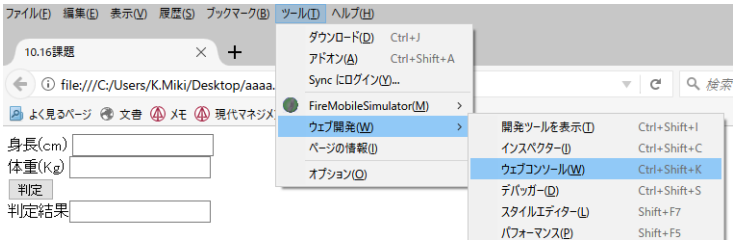



図 8　ウェブコンソールの起動方法

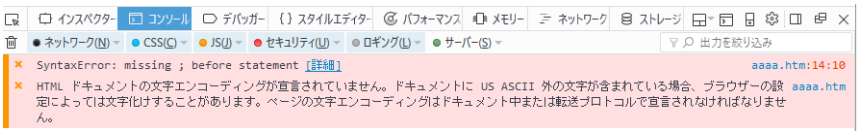



図 9　ウェブコンソールの画面

ではファイル名しかありませんが、全体の問題と言うことです。問題によってはかなり先になって初めて検出されるものもあるので、表示された場所以外に問題があることもあります。

- 「HTML ドキュメントの文字エンコーディングが宣言されていません。ドキュメントに US ASCII 外の文字が含まれている場合、ブラウザーの設定によっては文字化けすることがあります。ページの文字エンコーディングはドキュメント中または転送プロトコルで宣言されなければなりません。」

  このエラーは Meta タグで charset を設定すると出なくなります。charset の設定は忘れないようにしましょう。
- 「SyntaxError: missing ; before statement」

  メッセージの後にある「詳細」のリンクをクリックすると、このエラーの説明のページが表示されます。それを見て分かれば良いのですが。「;」を忘れていますと言う指摘ですが、本当に忘れている場合は単に「;」追加します。もっと別の原因であることも多く、その場合に安易に「;」を追加すると状況をさらに悪化させることになります。このテキストを作成する際に利用した例も、「function」が「funcution」になっていただけで「;」には全く関係のないエラーでした。
- 「TypeError: window.confilm is not a function」

  そのような関数はないと言うメッセージで、この場合「window.confirm」が正しい。

## 4.5 フォーム

フォームは利用者がデータを入れるための HTML のタグです。本来はここへ入力したものが Web サーバーに送られて CGI で処理されますが、Web サーバーに送る前に JavaScript でデータを処理することも可能です。ここで説明するタグは HTML のものなので Script タグの間やコメントのタグの間に入れないようにしてください。また JavaScript で利用する際には必ずフォームのタグで入れ物を出してから JavaScript が実行されるようにしてください。

まずフォームの記述全体を次のようなタグで囲います。

```
<Form name="名前">
   フォームの内容
</Form>
```

多くの人が from と間違えるので注意してください。「名前」の部分には適当な英数字によるものを入れます。違うフォームには違う名前を付けてください。フォームの内容としては色々なものがあるのですが、とりあえ

ず入力するためのものは次のような形になります。

```
<Input name="名前">
```

複数の入力欄を設ける場合にはそれぞれ異なる「名前」を付けてください。これによってブラウザによって適当な大きさの入力欄だけが表示されます。次のようにして大きさの指定も可能です。

```
<Input name="名前" size="文字数">
```

なおここで指定した「文字数」は表示際の入力欄の大きさであり、入力欄には通常これ以上の文字数を入れることが可能です。さらにあらかじめ入力欄に何か内容を入れておくことができます。

```
<Input name="名前" value="内容">
```

この場合「内容」に記述したものが、最初から入った形になります。value の指定が無い場合は空欄になります。

## 4.6 入力欄への入出力

入力欄へはクリックしてカーソルを出してからキーボードで入力することができます。ここでは JavaScript によって入力欄に何か出力したり、逆に入力欄に入っている内容を取り出したりする方法を説明します。

まず出力するには、

```
document.フォームの名前.入力欄の名前.value=出力内容;
```

とします。大抵のブラウザはこれで動きます。ただし「フォームの名前」や「入力欄の名前」がシステムの持っている別のものの名前と偶然一致したような場合に誤動作する恐れがあります。心配な人は、

```
document.forms['フォームの名前'].elements['入力欄の名前'].value=出力内容;
```

のようにしてください。また内容を取り出す場合には=の左側と同じものを使います。ただし、入力欄の内容が「123」と言うような数字の場合でも文字列として扱われますので注意します。数値に直したい場合は、eval() という関数を利用するのが正式なやり方ですが、数値を掛けることによっても直ります。以下は「aaa」と言う名前のフォームの中にある「x」と言う名前の入力欄に入れられた数字を数値として取り出す時の書き方です。

```
eval(document.forms['aaa'].elements['x'].value)
document.forms['aaa'].elements['x'].value*1
```

## 4.7 関数の定義 (1)

他のプログラミング言語などと比べて JavaScript では関数が使えることはかなり重要な意味を持っています。つまり関数がないと利用者からの指示に応じて何かすることができません。例えばどこかのボタンをクリックすると計算をしてくれるというような場合、その計算の内容を関数として定義して、ボタンにはクリックされたらその関数を呼び出せという指示を付けます。とりあえずここでは関数の定義方法を説明し、次章で

ボタンの方の説明をします。なお JavaScript 自体でもかなりの数の関数を既に持っています。それらで済む場合は自分で関数を定義する必要はありません。

関数を定義するには次のように書きます。

```
function 関数名 (){
    関数の中身
}
```

関数名は英数字です。関数の中身としては JavaScript の文を書きます。この記述は必ず関数の呼び出しに先行する必要があるので、よく<Head>のタグの間に書かれます。これまでの JavaScript の記述と異なりこれを書いただけでは関数の中身は実行されません。

関数の中身を実行したいところに、

```
関数名 ();
```

を書きます。するとここに関数の中身を書き写したのと同じように実行されます。

## 4.8 ボタン

Web ページに見られるボタンは HTML のタグで作られます。また本来 Form タグの中だけで有効なものなので記述する際は、必ず<Form>と</Form>の間に入れます。ボタンを置きたいところに、

```
<Input type="button" value="ボタンの文字" onClick="関数名 ()">
```

を記述すると、「ボタンの文字」で指定した文字の付いたボタンができ、これをクリックすると「関数名 ()」で指定した関数が実行されます。ここで指定する関数は通常自分で定義した関数ですが、JavaScript が持っている関数でもかまいません。

## 4.9 条件判断

コンピュータは昔は「電子計算機」と呼ばれていました。でもコンピュータと電卓は違います。共に計算ができますが、コンピュータはさらに条件判断が可能です。と言っても曖昧な判断は無理で yes か no かの論理的な判断しかできません。

判断の基本は 2 つのものの比較です。次のような比較演算子が利用可能です。

| | | | |
|---|---|---|---|
| < | 未満 | > | 超過 |
| <= | 以下 ($\leq$) | >= | 以上 ($\geq$) |
| == | 等しい | != | 等しくない ($\neq$) |

これらを利用することにより 2 つのものの比較ができますが、さらに多くのものを比較するために、これらの比較を&&(かつ) や||(または) でつなぐことができます。例えば次のような感じになります。

- x>3 && y<5
  これは x が 3 より大きくかつ、y が 5 より小さいという条件を示します。
- x==y && y==z
  これは x と y と z が等しいことを示します。比較は 2 つのものしかできないので、x==y==z のようには書けません。(書けば別の意味になる。)

このような条件判定を次のような if 文で使用します。

```
if（条件）{
   条件が成立したときに実行する内容
}
```

もし条件が成立しなかった場合は何も実行されません。さらに条件が不成立の場合に別の事をしたい場合は次のように記述します。

```
if（条件）{
   条件が成立したときに実行する内容
} else {
   条件が不成立のときに実行する内容
}
```

さらに if 文の中に if 文を入れることも可能です。次は if を利用した例の一部です。どのような場合にどのようなメッセージが表示されるか考えてください。

```
if (document.aaa.sex.value=="男") {
   if (documnet.aaa.looks.value=="Good") {
      document.aaa.message.value="携帯の番号教えてくれませんか？";
   } else {
      document.aaa.message.value="。。。";
   }
} else {
   document.aaa.message.value="こんにちは";
}
```

## 4.10 警告・確認画面

一旦ページが表示された後に、ページの内容で変更することができるところは限られます。そのうちの一つは Form のタグの中の入力欄でした。ただこれはあらかじめ入力欄を出しておかなければならないのと、あまり目立たないので中身を変更しても気がついてもらえない可能性があるなどの問題点があります。ここで説明する警告・確認画面は、必要なときに画面の中央に指定した内容を表示してくれるのでそのような問題点がありません。

### 警告画面

次のような記述が実行されると画面に別ウインドウが開きます。

```
window.alert(表示内容);
```

「表示内容」の部分には「”」や「’」で囲った文字列や式などが「,」で区切れば複数指定することができます。図 10 は、表示内容として”本当に好きですか？”を指定した場合です。これを見た利用者が「OK」のボタンをクリックするまで JavaScript の実行も停止します。

### 確認画面

間違ってボタンをクリックしたために全てのデータを失うことがあります。そのような不幸なことにならないように、後戻りが簡単にできないような処理を始める前に確認画面を出すようにしましょう。利用者が表示


図 10 警告画面の例

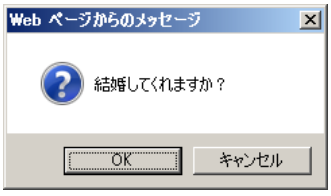



図 11 確認画面の例

された内容を見て「OK」または「キャンセル」のボタンをクリックするので、if を併用してその後に行う内容を切り替えます。

```
if (window.confirm(質問内容)==true){
    「OK」をクリックした場合にする内容
} else {
    「キャンセル」をクリックした場合にする内容
}
```

「質問内容」の部分が表示されます。「==true」は省略可能です。指定できるものは警告画面のものと同じです。「キャンセル」をクリックした場合何もしないのであれば else 以下を省略することができます。図 11 は質問内容として”結婚してくれますか？”を指定した場合です。

## 4.11 ラジオボタンとチェックボックス

入力される内容があらかじめ数種類に限定される場合には、ラジオボタン、チェックボックスや次に説明する選択メニューが用いられます。入力欄を用いた入力では想定外の入力が避けられませんが、ラジオボタンなどではこちらが用意したものしか選べないのでそのようなことが無く、また入力者にとってもマウスで選択するだけなので楽です。ただ県名のように多数あるものとなると、マウスで選択するのも大変になるので注意が必要です。

ラジオボタンとチェックボックスの違いは、複数選択ができるか、できないかにあります。それぞれ用途によって使い分けることになります。例えば、性別を問う場合は、男かつ女は通常ありえないので、複数選択できないラジオボタンを使います。また紅茶に追加するものを選択する場合は、ミルクや砂糖などを好みに合わせて選択することになるので、複数選択できるチェックボックスを使用するのが普通です。

まずラジオボタン自体は HTML のタグです。Form タグの有効範囲の中で使用できます。例えば、

```
<Input type="radio" name="sex" value="男"> 男<Br>
<Input type="radio" name="sex" value="女" checked> 女<Br>
<Input type="radio" name="sex" value="宇宙人"> その他<Br>
```

のように記述すると、図 12 のようになります。


図 12 ラジオボタンの例

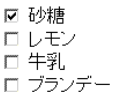



図 13 チェックボックスの例

name=のところは、同じグループに属するものは全て同じ名前にします。value=のところは、JavaScript で選択したものを調べた時に渡したい値を入れます。JavaScript では何番目の選択肢を選択したかどうかわかるので、必要としない場合もありますが、授業の後半で扱う PHP によるサーバーでの処理の際には、個々で

指定した値しかサーバーに渡らないので必須の項目となります。checked は同じグループの中で一つだけに付けます。これがある項目が最初に選択された状態になります。これを忘れると、どれも選択されていないままになることがあるので、トラブルの元になります。

そして Input タグの後に何か書かなければならない事に注意してください。Input のタグだけでは○しか出てこないので、何の選択肢なのか利用者にはわかりません。

JavaScript 側でどのラジオボタンが選択されたか調べるには次のようにします。

```
if (document.フォーム名.ラジオボタン名[0].checked){
   1番目のラジオボタンが選択されていた場合の処理内容
}
```

[ ] の中がゼロの場合、1 番目のラジオボタンを調べることになります。2 番目のラジオボタンならば、[ ] の中は 1 になります。要するにゼロから数えるようになっています。また value=で指定した内容を取り出すには、「document. フォーム名. ラジオボタン名 [0].value」のように記述します。

繰り返しを用いて多数のラジオボタンを一気に処理しようとする場合、ラジオボタンがいくつあるかを知る必要があります。そのような場合は、「document. フォーム名. ラジオボタン名.length」でわかります。

チェックボックスもラジオボタンとほぼ同様になります。

```
<Input type="checkbox" name="sugar" checked> 砂糖<Br>
<Input type="checkbox" name="lemon"> レモン<Br>
<Input type="checkbox" name="milk"> 牛乳<Br>
<Input type="checkbox" name="brandy"> ブランデー<Br>
```

のように記述すると、図 13 のようになります。

グループを構成する必要はないので、name=では異なる名前を指定します。複数選択可能なので、checked は複数付けても良いし、全く付けなくても問題ありません。JavaScript でチェックが付いたかどうか調べる時にはラジオボタンの時と同様に、

```
if (document.フォーム名.チェックボックス名.checked){
   チェックボックスにチェックが付いている場合の処理内容
}
```

のようにします。

## 4.12 選択メニュー

ラジオボタンは画面上に予め全ての選択肢が表示されなければならないので、画面上の場所をとると言う問題点があります。ここで説明する選択メニューならば、選択の際にだけ選択肢が表示されるので便利です。ラジオボタンと同様に選択メニュー自体は HTML のタグです。Form タグの有効範囲の中で使用できます。例えば、

```
<Select name="selone">
   <Option>寝る
   <Option selected>食べる
   <Option value="run">走る
</Select>
```

のように記述すると、図 14 のように表示されます。


図 14 選択メニューの例

この例では、「寝る」、「食べる」、「走る」の 3 つの中から一つ選ぶ事ができます。<Option>の所はいくつでも可能です。selected は 1 つだけ指定可能で、この項目が最初から表示されます。また value の指定をすると、あとで JavaScript で値を取り出すことができます。(この例では「走る」の項目のみ「run」と言う値を持つ。)

なお、複数選択可能な項目とするには、最初の行を次のようにします。

```
<Select name="selmul" multiple>
```

2 つめ以降の項目を選択する際には Ctrl を押しながらクリックします。この場合には selected を複数指定しても構いません。また JavaScript の方で複数選択した場合に対応できないといけません。

また、画面上で幾つかの項目が最初から表示されるようにしたい場合には、最初の行を次のようにします。

```
<Select name="selone" size="3">
```

とすれば、最初から項目が 3 つ表示されるようになります。項目が 3 つ以上ある場合には、残りの項目を表示させるためのスクロールバーが自動的に表示されます。

さて、このようにして作成した選択メニューでどのような選択が行われれたかは、次のようにして JavaScript で調べることができます。

1. 選択肢の数は、「document. フォーム名. セレクト名.options.length」[*5]でわかります。なお、「セレクト名」は Select タグで指定した名前のことです。
2. 選択されたかどうかは、「document. フォーム名. セレクト名.options[数字].selected」[*6]が true であるかどうかでわかります。「数字」の部分には、ゼロから選択肢の数-1 が入ります。複数選択されている場合は複数の true が、全く選択されていない場合は、true が一つも無いこともあるので注意します。通常は次のようにこれを if 文の条件のところに入れて、選択された項目のみ処理するようにします。

```
if (document.フォーム名.セレクト名.options[2].selected){
   3番目の項目が選択されている場合の処理内容
}
```

3. 選択肢に設定されている値は、「document. フォーム名. セレクト名.options[数字].value」[*7]で知ることができます。「数字」の意味は selected と同じです。
4. 選択を変更したら処理をしたい場合には、Select のタグの中に onChange という指定を入れます。

```
<Select name="selone" onChange="関数名()">
```

[*5] きちんと書くならば、document.forms[フォーム名].elements[セレクト名].options.length になります。
[*6] きちんと書くならば、document.forms[フォーム名].elements[セレクト名].options[数字].selected になります。
[*7] きちんと書くならば、document.forms[フォーム名].elements[セレクト名].options[数字].value になります。

## 4.13 変数

プログラミング言語には必ず「変数」と言うものが出てきます。ただの数ならばいつも同じ値なので問題ありませんが「変数」は字のごとく値が変化します。さらに表計算のセルの値のように直接値を見ることができません。と言うことで「変数」はプログラミングを勉強する人にとっての一つの越えなければならない壁であります。このテキストでは壁はできるだけ後回しと言うことでやってきましたがそろそろ限界なのでやります。

「変数」は名前の付いたメモのようなものです。メモには数値または文字や文字列を一つ書くことができます。書いた内容は読むことができます。一つしか書けないので新しい内容を書き込むと以前の内容は失われます。

「変数」を利用するためには通常「宣言」が必要となります。「宣言」によってどのような名前で、どのような形式 (数値や文字) の内容が書き込めるかと言うことを示します。プログラミング言語によっては「宣言」を必要としないもの、形式の指定が不要なものもあります。JavaScript は名前の宣言は必要ですが形式の指定が不要なものに属します。

宣言を行うとある範囲でその変数が使用できるようになります。関数の中で宣言を行うと関数の中でのみ使用可能になります。関数の外で宣言をするとその行以降で使用可能になります。ただどこで宣言しても通常のプログラミング言語では、変数の有効なのはそのプログラムの実行中に限定されますので注意してください。JavaScript の場合は、その記述があるページが表示されている間になります。

JavaScript での変数の宣言の仕方は次のようになります。

```
var 変数名，変数名，... ;
```

「変数名」として使用可能なものは英数字と「_」です。このような形で複数の変数を一度に使えるようにできます。またこのように宣言した変数には数値や文字など一つの変数に一つに限られますがなんでも入れることができます。また次のように宣言と同時に変数に値を入れておくことができます。

```
var x=100;
```

通常宣言をしただけの変数はどのような値を持つのかわからないのでこのように最初から入れておくと安心です。

変数の使い方で判りにくいのは、=の両側に同じ名前の変数が出てきた場合です。

```
x=x+3;
```

数学的に考えると x と x+3 が等しくなるわけ無いので矛盾していますが、プログラムの世界では、=の左側では x と言う名前の**入れ物**を示し、=の右側では x と言う名前の**入れ物の中身**のことなので矛盾は生じません。ただ両側に同じものが出てくるのは入れるのが面倒？と言った理由から、

```
x+=3;
```

と言った表現が可能になっています。「+=」は右側の値を左側の変数に加える、と言うような働きをします。

## 4.14 タイマー割り込み

コンピュータの分野で「割り込み」と言うのは結構重要な機能です。コンピュータは誕生以来かなり初期の頃から回りの人や機械と比べてかなり高速でした。その結果、人からの入力、テープ装置からの読み込み、プ

リンターへの印字などの際には速度を合わせるための待ち時間が実行時間全体の 99% 以上を占めると言うことも珍しくありませんでした。そこでその待ち時間を有効に利用するために他の仕事も並行して行おうとしましたが、他の仕事の方に夢中になってしまうと困ります。「割り込み」機能はちょうど我々が用いる電話のように、本来の仕事をする必要が生じたときに、割り込むことができるようなものです。これによって他の仕事に夢中になっていても構わなくなります。

JavaScript の onClick や onChange も広い意味では割り込みです。他の仕事は特にしていませんが、これらの指示をしておくとマウスの操作で関数の実行を割り込ませることができました。ここで説明するタイマー割り込みも似たようなものですが、関数実行の機会は指定した時間が経過するとやってきます。この機能を利用して実務的には時間切れの処理、趣味的には画面をどんどん変化させるのに用いられます。

タイマー割り込みの設定には次のように setTimeout 関数を用います。

```
x=setTimeout("関数名()",時間);
```

「関数名」のところには時間が経過したときに呼び出す関数を”owari()”のような感じで記述します。「時間」のところには、関数を呼び出すまで待つ時間を 1/1000 秒単位で記述します。たとえば 3 秒後に呼び出すならば 3000 と書きます。ただしこの時間に関しては様々な要因が絡むのでそれほど正確ではありません。「x」は任意の変数で構いませんが、これにタイマーの識別子 (ID) が入ります。これを使用して次のようにすると設定したタイマー割り込みを解除することができます。

```
clearTimeout(x);
```

注意しなければならないのは、setTimeout で設定した関数が一度呼び出されると、このタイマー割り込みの設定が消えることです。定期的に呼び出す場合は、次の setInterval 関数を使い、繰り返し呼び出す場合は、呼び出された関数の中で再び setTimeout を行う必要があります。

```
x=setInterval("関数名()",時間);
```

setInterval 関数の解除には clearInterval 関数を使用します。使い方は clearTimeout 関数と同じです。

## 4.15 画像のプロパティ

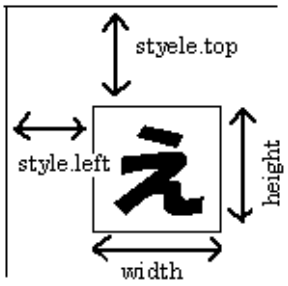
JavaScript では様々なものをオブジェクト (object: 物) として扱います。window や document もオブジェクトです。オブジェクトはプロパティ (property:特質) を持ちます。どのようなプロパティを持つかはオブジェクト次第です。ここでは画像のプロパティを例にあげ、それを JavaScript で扱う方法を説明します。

まず画像のプロパティとして代表的なものに図 15 のようなものがあります。

これらのプロパティは HTML のタグの一部として設定することができます。また style.top や style.left は Style Sheet で設定することができます。

```
<Img src="test.gif" height="100" width="100" name="test">
```

また JavaScipt でプロパティを変更することにより画像を動かしたり、変化させることができます。「position: absolute」と言う指定が出てきますが、これは位置を Window 内の座標で指定すると言う意味です。




| | |
|---|---|
| height | 画像の縦幅を画素数で表したもの |
| width | 画像の横幅を画素数で表したもの |
| style.top | 表示している画像の左上角の Y 座標 |
| style.left | 表示している画像の左上角の X 座標 |
| name | 画像の名前 |
| nameProp | 表示している画像のファイル名 |
| src | 表示している画像の URL |
| onclick | 画像をクリックしたときに起動される関数 |

図 15 画像のプロパティ

```
<HTML lang="ja">
<Head>
<Meta charset="UTF-8">
<Title>動く・変わる画像</Title>
</Head>

<Body>
<Img src="http://www.ss.sugiyama-u.ac.jp/web/gif/kousha.gif" name="kousha"
     style="position: absolute; top: 100px;">
<Script language="JavaScript">
window.alert("表示されている画像の縦幅は、"+document.kousha.height+"です。");
window.alert("表示されている画像を動かします。");
document.kousha.style.top="50px";
</Script>
</Body>
</HTML>
```

画像に名前が付いていない場合や他の名前と混同の恐れがある場合は次のように指定します。

```
document.images['kousha'].src="test.gif";  // 混同を避ける
document.images[0].src="test.gif";        // ページに最初に出てきた画像
```

## 4.16 Window の操作

アクセスすると本体以外に別の Window が開くページがあります。大抵は CM のページなので邪魔なだけですが、場合によってはそれを閉じるとさらに別の CM のページが出てくるというものもあります。ここでは Window に関する様々な操作法について説明します。

まず別 Window の作成の仕方です。次のようにして新しい Window を開くことができます。

```
var w;
w=window.open(URL,windowID,option);
```

URL には通常、”http://...”とか”ファイル名”が入ります。windowID は window に付ける名前で英数字で適当な名前を付けます。複数の Window を開く場合には違う名前にします。またこれを A タグの中の target で指定することにより、クリックすると別 Window に表示する様なこともできます。option の部分には、様々な指示を””で囲って、「,」で区切って複数指定することができます。(例えば、”width=200,height=100”のような感じ。) 表 10 に option に使用できるものを示します。

表 10 window.open の option 設定

| 指示名 | 内容 | 例 |
|---|---|---|
| height | Window の高さ | height=300 |
| width | Window の幅 | width=300 |
| directories | リンクのところを付けるかどうか | directories=yes |
| location | URL の表示を付けるかどうか | location=yes |
| menubar | メニューバーを付けるかどうか | menubar=yes |
| resizable | 大きさを変更できるようにするかどうか | resizable=yes |
| scrollbars | スクロールバーを付けるかどうか | scrollbars=yes |
| status | window の下に状況表示を出すかどうか | status=yes |
| toolbar | ツールバーを付けるかどうか | toolbar=yes |

何か option の指定をした場合、yes にしなかったものは表示されません。ただブラウザによっては出てくることもあるので no と指定した方が良いかもしれません。なお、URL、windowID、option の指定は省略可能です。

上記の例で w という変数には生成された window が設定されます。これを利用して新しくできた window を操作することができます。例えば、

```
w.close();
```

を実行するとその window を閉じる (消す) ことができます。window は手動で消すこともできるので、操作しようと思っても、その window が既にないこともあります。そのような場合に対処するには、

```
if (w.closed) {
   その window が閉じられていた場合の処理
}
```

のように記述します。

URL を指定する場合は新しい window の中味は別ファイルに入れることになりますが、URL を指定しない場合 (URL の場所に””を指定) は、open したあとに続けて document.write でその内容を送ることもできます。

```
var w;

w=window.open("","","width=300,height=200");
w.document.write("<HTML>");
w.document.write("<Head><Title>タイトル</Title></Head>");
w.document.write("<Body><H1>タイトルだぜ</H1></Body>");
w.document.write("</HTML>");
```

この例からもわかるように、先頭に「w.」を付けることにより、これまでやったことが大抵そのまま新しく作った Window に対して行うことができます[*8]。逆に新しく作られた Window から逆に元の Window に対して指示を出す事ができます。その場合には「window.opener.」を付けることになります。例えば子の Winodow の方で次のような文を実行すると、元の Window に椙山のトップページを出す事ができます。

[*8] document.write で指定する文字列の中に</Script>がある場合は、<¥/Script>のように記述してください。そうしないとここで JavaScript の記述が終了したと見なされて変なエラーになります。

```
window.opener.location.href="http://www.sugiyama-u.ac.jp/";
```

そして JavaScript が記述されている Window に対して操作したい場合は、「window.」を付けるか何も付けないようにします。