展開演習 A テキスト

梱山女学園大学現代マネジメント学部　三木　邦弘

令和 6 年 4 月 10 日版

1. はじめに

今回で 5 年目の展開演習になりますので、テキストもかなり完成しましたが、内容の充実、問題点の修正、記述をよりわかり易いものにしていくつもりです。「展開演習 A」ではまず「Web デザイン」や「プログラミング基礎」でやったことの復習に加えて、Web 関連技術をさらに学び、新たにデータベースを取り上げます。情報システムを作る基礎能力の習得が目標です。そして「展開演習 B」で様々な情報システムを作ります。

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 (Internet of Things) 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て、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数値情報や文字情報などを元に**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。例えば銀行業務の中で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。それは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ようとしています。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。メガバンクでは事務処理の軽減のための RPA (Robotic Process Automation) の導入の結果、余剰人員の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。

また生成系 AI は売り物である情報を作ることができるので、これらの応用は今後の大きな課題です。現在では様々な問題が指摘されていますが、生成された情報をインターネット経由で売ると言うビジネスは、これまでにない高い収益をもたらすでしょう。また普通の言葉で指示ができる ChatGPT は大きな話題になっていますが、返ってくるのも通常言葉なのでよくわからない点があります。

情報システムにリストラされる人でなく、情報システムを作る人になって欲しいと思います。単なるシステム作りでは理系の大学出身の人に勝てるようになるのは難しいと思います。ただ理系の大学出身の人は、情報システムで儲けると言う辺りが苦手です。この辺りが経営や会計も学べる本学部の学生の強みとなります。情報システムによる合理化を避けていては会社がだめになります。また情報システムができて人が余った時に、リストラしか思いつかないような経営者では合理化は実現できないでしょう。そのような先まで考えられる人になって欲しいと願っています。

課題は難しくなります。その日のうちにできるると思わないでください。レポートならば誤字脱字が少々あってもレポートになりますが、コンピュータの場合一文字のミスで大抵動かなくなります。エラーメッセージが出るのならば、それがヒントになりますが、エラーメッセージが出ないがまともに動かない事はよくあります。それでも課題はできて動かないと面白くありません。昨年度よりは人数が減るので、より個別に対応も可能になります。頑張ってください。

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 の復習と多少細かい追加の話をします。例えば私の「Web デザイン」の授業での HTML はかなり昔の素朴な頃の HTML を元にしていますが、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「FeatherPad」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」で利用した peditor の mars 用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 2.1 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。

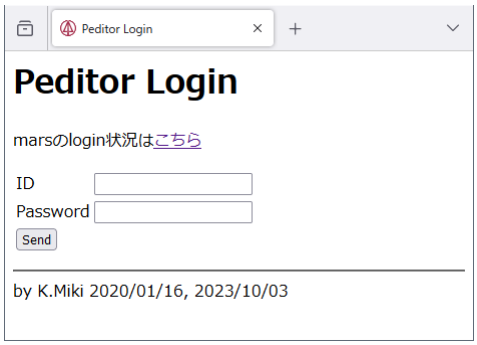


図 2.1 peditor: ログイン画面

すると図 2.2 のようなファイルの一覧の画面になります。「peditor.php」は peditor の本体なので消したり変更しないでください。以下ボタンについて説明します。

- 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。

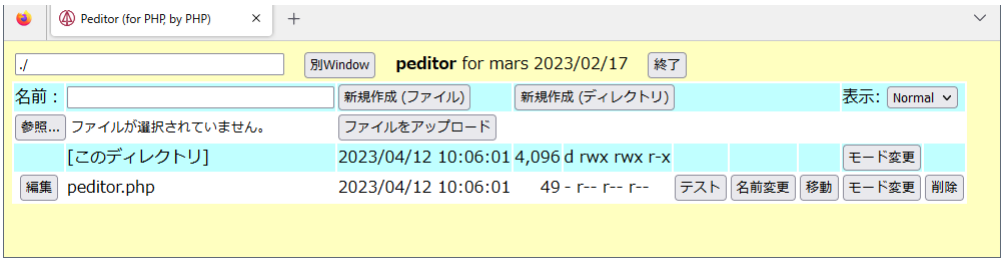


図 2.2　peditor: ファイルの一覧

- 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。Winodws ではフォルダと呼びますが、大抵の OS ではディレクトリと呼びます。新しいディレクトリを開くことにより、その中にファイルなどを入れることができます。
- **表示**　ここでファイルの内容を表示する際の色つけを選択することができます。Old、Lite、Normal、Heavy の 4 種類があり、色つけの全く無い Old〜一番カラフルな Heavy となっています。パソコンやブラウザにもよりますが、行数の多いプログラムでは Heavy は反応が悪くなることがあります。
- 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　指定したファイルを mars にアップロードします。
- 上のディレクトリに移る　新しいディレクトリを作成し、それを開いた場合にこのボタンが出ます。これで前のディレクトリに戻ることができます。
- 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。
- 表示、テスト　別のタブでファイルの内容を表示することができます。
- 名前変更　ファイルの名前を変更します。
- 移動　ファイルを別のディレクトリに移すことができます。
- モード変更　ファイルの読み書きの設定をします。
- 削除　ファイルを消去します。
- 開く　ディレクトリの場合、「編集」や「テスト」の代わりにこれができるようになります。これでディレクトリの中のファイルの一覧に切り替わります。

なお、編集ボタンの場合は図 2.3 のように画面が変わります。

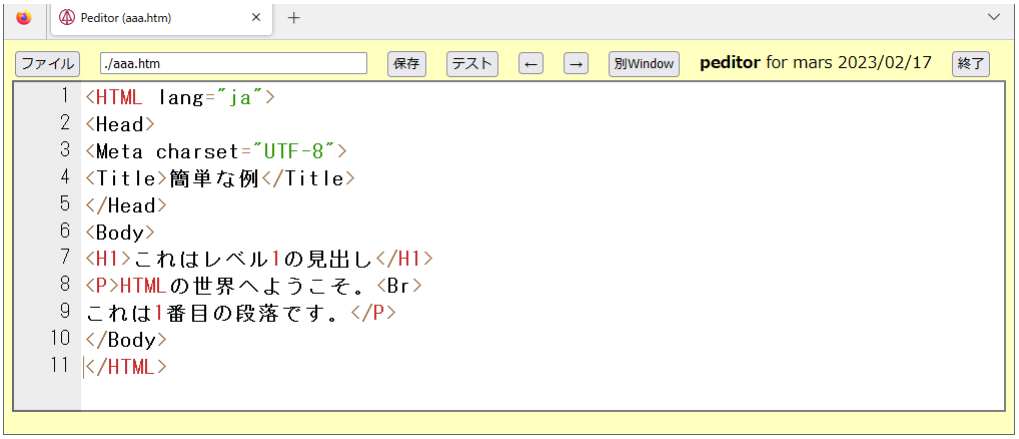


図 2.3　peditor: ファイルの編集

- 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。
- ←　キー入力などの変更を一つ戻すことができます。クリックする度に更に戻ります。
- →　戻したものを戻します。←キーで戻し過ぎた時に使います。
- 前の保存に戻す　保存する前の状態にもどすことができます。一つ前までしか残っていないので、何度も保存した場合に最初まで戻すことはできません。一度も保存したことがない場合は、このボタンは表示されません。もし保存したのにこのボタンが表示されない場合は、ブラウザの更新ボタンをクリックしてください。

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>
</Body>
</HTML>
```

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[*1]、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。

[*1] Firefox では CTRL+U で表示されます。

```
<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

- 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。

```
<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。

- **中央揃え**：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>
```

- 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 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="https://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>
```

- web ページ内へのリンク：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>
```

表 2.1 target の設定

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
target=”フレーム名”	指定されたフレームに表示 (フレームは後述)

表示先の指定

通常の A タグによるリンクの設定では、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。A タグに target=と言う指定を追加することにより、表 2.1 のように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="https://www.sugiyama-u.ac.jp/" target="_blank">椙山のトップページ</A>
```

2.5 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。図 2.4 のように 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。

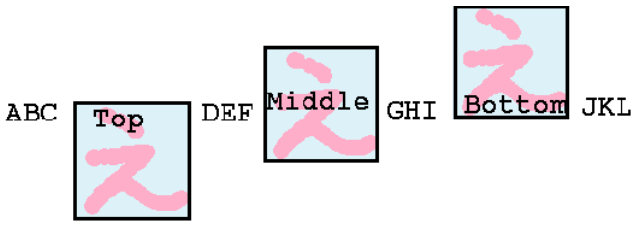


図 2.4 画像の表示位置

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。

2.6 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。このような見た目の設定は、次章の Style Sheet を使用するのが普通です。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。

2.7 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 2.2 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 2.2 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 2.3 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 2.3 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`

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。これらの設定も普通は次章の Style Sheet で行います。

- **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>
```

- **肩付き文字の設定**：$y = x^2$ 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>
```

- **下付き文字の設定**：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>
```

2.8 表の設定

表の設定は、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. 表の設定全体を<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. 表の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。
   - <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。
   - セルの内容が複数行にわたる場合は、改行すべき所に
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。)
   - <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。

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

2.9 画面の分割 (フレーム)

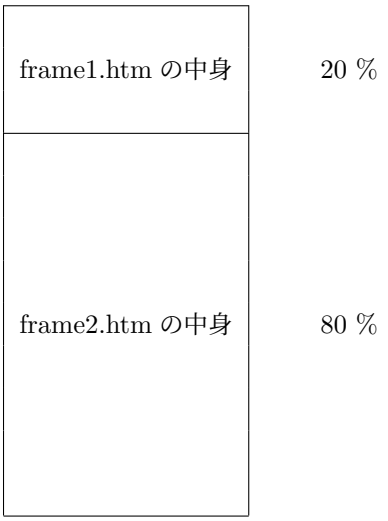
画面を分割することによって画面を目次とその内容のように分けて、いちいち利用者が目次のあるページの戻らなくても良いようにできます。ただこれを利用する場合は、分割の仕方を指定するファイルとその分割された内容のファイルの両方が必要になります。後者は 2 分割ならば 2 つ、3 分割ならば 3 つ必要になりますが、これらはこれまで説明してきた普通の HTML で記述された内容のものです。

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 lang="ja">
<Head>
  Meta、Title 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。

2.10 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。

2.11 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。さらに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。これらの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。呼び出されたプラグインソフトにより、音声や動画の再生が可能になりましたが、これが入っていないパソコンでは再生できないと言う問題があります。

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るでしょう。

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。この例では MP4 と呼ばれる形式の動画ファイルを指定していますが、一番普及しているこの形式でも内容が異なるものがあり、再生に失敗することがあります。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 2.4 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 2.4 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 2.5 のようなものが表示されます。

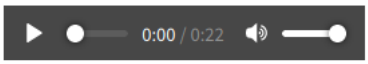


図 2.5 audio タグの表示

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) を取り上げます。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対応されましたが、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、ここでは上流で指定した style が下流にも伝わっていくと言う捉え方になります。HTML のタグは入れ子になっています。例えば`<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、その中にある`<Body>`や`<H1>`に引き継がれていくようになっています。共通する Style を上流で指定することにより、個々に設定する手間を減らすことができます。

3.1 どこに入れるのか

後述の Style の指定には 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。

  ```
  <タグ名 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。後のやり方ほど有効範囲が狭くかつ優先されるので、最初のやり方で各 Web ページ共通部分を指定し、最後のやり方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 3.1 のようなものがあります。○には数値が入ります。

表 3.1 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
height	高さの指定	○ px、○ %、○ em、○ rem、○ cm、○ mm
width	幅の指定	○ px、○ %、○ em、○ rem、○ cm、○ mm
cursor	マウスカーソルの形状の指定	crosshair、default、hand、move、text、wait など
font-style	文字の書体の指定	italic など
font-weight	文字の強調の指定	100〜900(100 step)、normal、bold、bolder、lighter
font-size	文字の大きさの指定	○ 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	行間の幅の指定	○ px、○ pt、○ cm、○ mm、○
writing-mode	縦書きか横書きの指定	horizontal-tb (通常の横書き)、vertical-rl (通常の縦書き)、vertical-lr (左から右へ改行する縦書き)

なお、長さの指定の所で使用できる単位の意味は、表 3.2 のようになっています。

表 3.2 style の長さの例

単位	説明
%	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
em	1 文字の大きさを 1 とした指定
rem	HTML タグで指定した 1 文字の大きさを 1 とした指定
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)
vw	ブラウザの表示幅の 1/100
vh	ブラウザの表示高さの 1/100

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。

通常の文章の一部に style を設定したいと言う場合は、Span や Div と言うような HTML のタグがよく用いられます。P やセルが一つだけの枠が透明な Table でも良いのですが、文中の一部に style を適用したい場合は、これらのタグは改行などを引き起こすため使えません。Span タグは何も働きが無いタグなので次のように使えます。Div タグも同様にほとんど働きがありませんが、大きさが固定されているので、後述のように表示する位置を設定するような場合は Div の方を使わないとうまく動きません。

```
途中で<Span style="color: red; font-size: 3em">重要</Span>なところを強調する。
```

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 3.3 のような 4 個の疑似クラスがあらかじめ決められています。

表 3.3 A タグに設定されている疑似クラス

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 (このクラスは他のタグでも使用可)

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。あらかじめシステムで決められた疑似クラス名の前は「:」で、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、

```
<タグ名 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: yellow}
   P.special {color: aqua}
   .smallred {color: red; font-size: 8pt}
</Style>
</Head>

<Body>
<P>これは黄色になる</P>
<P class="special">これは水色</P>
<P>またまた黄色になる</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 3.4 のようなスタイルが利用されます。

表 3.4 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 3.1 のような意味になります。

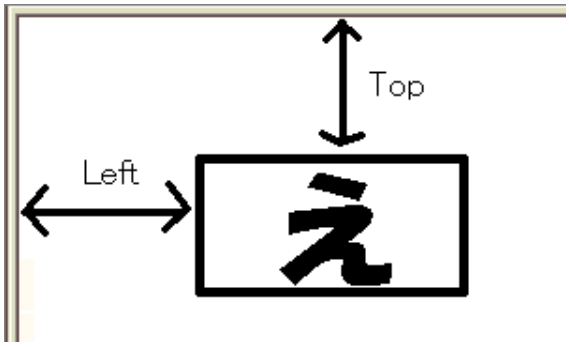


図 3.1 絶対的な位置の指定

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.mgt.sugiyama-u.ac.jp/favicon.ico"
     style="position: absolute; top: 100px; left: 50px">
```

表や画像の表示位置を設定したい場合は、このようにそれを構成するタグに style を設定すれば良いのですが、文章の一文字ずつに位置の指定をする訳にも行かないので、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 1〜4 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 3.5 をご覧ください。

表 3.5 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 枠線の設定

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 (solid) 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 3.2 の長さが使えます。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。

- 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。
- 内容をブロックに分割し、パソコンの横長画面ではブロックが横に並び、スマホの縦長画面ではブロックが縦に並ぶようにする。
- スマホ用にデザインし、パソコンで巨大な表示になっても気にしない。

ここでは最初と最後の方法について紹介します。最近は 2 つ目の方法を用いているサイトがよく見られます。次節で紹介する Milligram.css を利用すると比較的用意に実現することができますが、説明が長くなるので省略します。３つ目は、全て相対的にサイズを設定する方法を用いています。スマホでも大画面ディスプレイでも同じ形になるので、デザインは崩れませんが、パソコンでは見やすいとは思えません。

スマホ対応ページの確認方法

実際に作成した Web ページがスマホではどのように見えるかを確認するには、実際にスマホからアクセスすれば良いのですが、アクセスのためにスマホで URL を入力するのは面倒です。現在の多くのパソコンのブラウザ (Chrome、Edge、Firefox など) では簡単に表示をスマホの形に変更できます。Chrome や Edge では F12 で開発者モードにしてから、Firefox では直接 Ctrl+Shift+M で図 3.2 標準の表示とスマホ表示を切り替えることができます。縦向き表示や横向き表示もボタンで切り替えることができます。

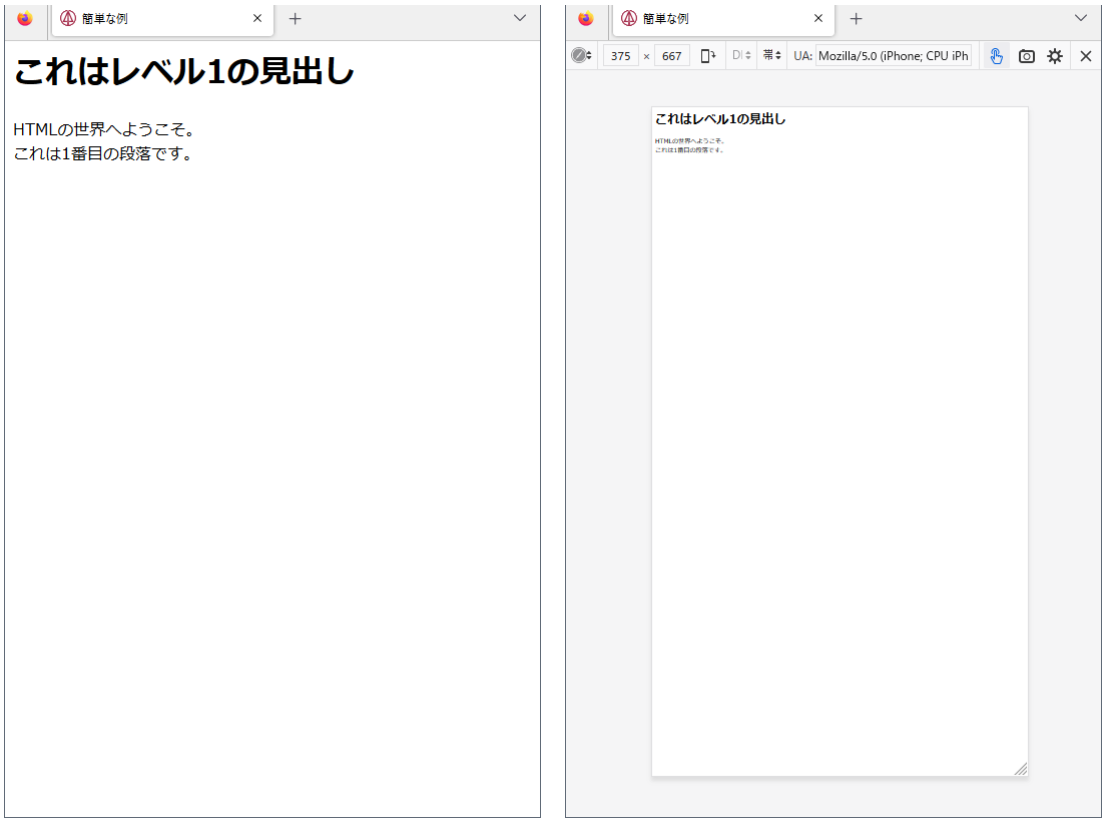


図 3.2　左が標準表示、右がスマホ表示

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 以上ありますが、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 }
```

この max-width は画像だけではなく P や Div などの他のタグにも利用できます。スマホと逆にパソコンでは物理的に横幅の大きな画面で見ることも多く、その際にブラウザの幅いっぱいに文字を表示すると大変読みにくいものになります。読みやすい幅に width で設定すると、逆にブラウザの幅が狭い時に、右側が切れてスクロールしないと読めなくなります。max-width で設定すると、ブラウザの幅が狭い時はいっぱいに、広すぎる時は適当なところまで表示されるようにできます。

相対的なサイズの設定

表 3.2 の長さの単位は大半が絶対的な大きさですが、vw と vh はそれぞれブラウザの表示幅の 1/100 と表示の高さの 1/100 と言う実際の表示に合わせた相対的な大きさになっています。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 rem (HTML タグで指定した 1 文字の大きさを 1 とした指定) を使用して、全て基本の文字の大きさを基準にすれば、同じ style の記述でも見え方を図 3.3 のように揃える事ができます。

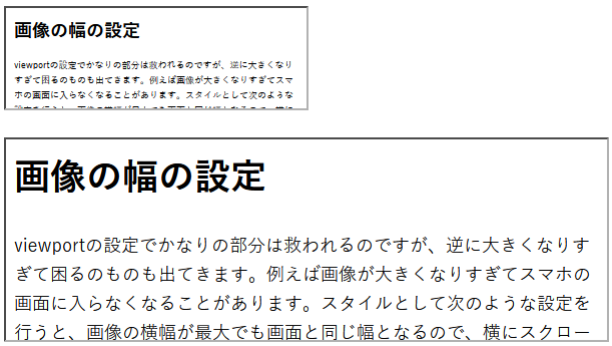


図 3.3 幅の違うブラウザでの見え方

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[*2]と言う大変小さなフレームワー

[*2] https://milligram.io/

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[*3]、Ulkit[*4]や JavaScript なども含んでいる比較的大きな Bootstrap[*5]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 3.4 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 3.4　Milligram 使用前 (左) と使用後 (右)

Milligram ではリンクに下線が付かないのでリンクらしくありません。A タグに class として「button」を指定するとボタンの形状になるのでわかりやすくなります。

[*3] https://purecss.io/
[*4] https://getuikit.com/
[*5] 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。CGI の具体的な例として次章の PHP がありまます。よほど単純な処理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、ここではより簡易的に扱える JavaScript を取り上げます。JavaScript は、C 言語に似ており、Object 指向の言語でもあります。

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 ブラウザへの出力

JavaScript でブラウザの画面になにか出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。つまり出したい文字や HTML のタグを「”」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。

document はブラウザの中の一つの object(物) で、今ブラウザで表示されているもの全体に相当します。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。

最後の「;」は文の区切りですので必ず付けましょう。JavaScript にはこれを自動的に挿入する仕組みがあるので、通常 1 行に複数の文がある場合以外は、省略してもエラーにはなりません。

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。

ちょっと問題になるのは「+」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、通常の使用ではエラーは表示されなくなりました。ここでは JavaScript 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、Cntl+Shift+I を押すと図 4.1 のように画面の下部に表示されます。もし「コンソール」以外になっている場合は「コンソール」をクリックしてください。Chrome ブラウザでも同様に起動できます。Edge はちょっと面倒な手続きが必要になります。

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。図 4.1 の例では「e0426.htm:27」となっているので、e0426.htm と言うファイルの 27 行目付近に問題があることがわかります。問題の内容は英語で書かれていま

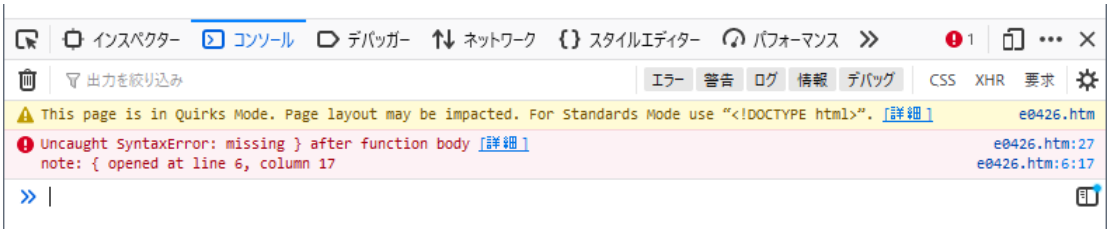


図 4.1 ウェブコンソールの画面

すが、「関数の本体の後の } がないよ」とのことです。その次に 6 行目の 17 字目に { があるよと注が付いていますが、実際に } が抜けていたのは 8 行目でした。このようにエラーはかなり先になって初めて検出されるものも多いので注意します。以下に他のエラーの例を示します。

- 「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、これは「;」を忘れていますと言う指摘です。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。「=」はその左右が等しいという数学的な意味ではなく、右側のものが左側に入るという意味です。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。

内容を取り出す場合には「document. フォームの名前. 入力欄の名前.value」を「=」の右側で使います。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 2 つの書き方です。

```
eval(document.aaa.x.value)
document.aaa.x.value*1
```

4.7 関数の定義

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。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
}
```

関数名は英数字と下線 (_) が使えます。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、通常<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 (≦)	>=	以上 (≧)
==	**等しい**	!=	等しくない (≠)

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) のような論理演算子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 より大きい、または 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。2 つのものの比較しかできないので、x==y==z のようには書けません。(書けば別の意味になる。)
- !(x==y)
  これは x と y が等しくないことを示します。「!」によって論理が反対になるからです。「!」は結びつく力が強いので対象を ( ) で囲う方が安全です。

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 = "メアド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[*6]。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 4.2 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。

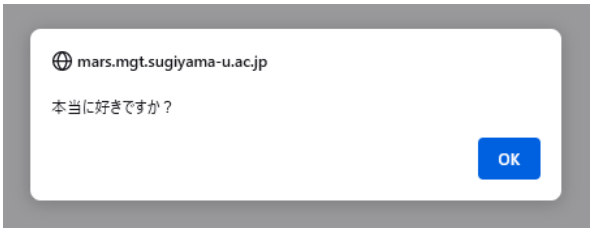


図 4.2　警告画面の例

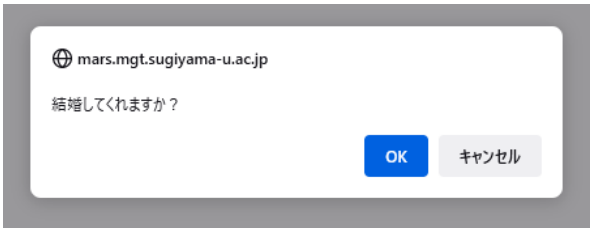


図 4.3　確認画面の例

[*6] JavaScript で style を変更したり、HTML のタグを追加や削除も可能ですので、上級者にとってはこのような制限はありません。

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。質問内容の部分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 4.3 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 4.4 のようになります。

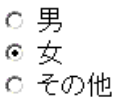


図 4.4 ラジオボタンの例

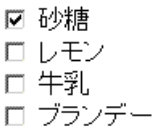


図 4.5 チェックボックスの例

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、次章で扱う PHP によるサーバーでの処理の際には、ここ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 4.5 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 4.6 のように表示されます。

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。選択項目を指定する<Option>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。)

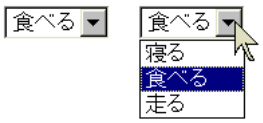


図 4.6 選択メニューの例

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」[*7]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。表示されている選択肢の数は、「document. フォーム名. セレクト名.size」[*8]で分かりますし、これを使って表示する選択肢の数を変更することもできます。
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」[*9]が true であるかどうかで分かります。「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「value="〜"」で設定されている値は、「document. フォーム名. セレクト名.options[数字].value」[*10]で文字列として取り出すことができます。「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()">
```

[*7] 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。
[*8] きちんと書くならば、document.forms[フォーム名].elements[セレクト名].size になります。
[*9] 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。
[*10] 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。

「変数」は名前の付いたメモのようなものです。メモには数値や文字などを一つだけ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」によって変数がどのような名前で、どのような形式 (数値や文字など) 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は内容に関する記述がないために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もできます。

```
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。加算だけでなく減算 (-=)、乗算 (*=)、除算 (/=) も可能です。

4.14 繰り返し (for)

選択メニューで多数の選択子がある場合、その数だけ if を並べるのも大変です。checkbox のように各々名前が異なるのであればまだしも、[ ] の中の数字が違うだけとなるともう少しなんとかならないのかと思うでしょう。そこで繰り返しをうまく使えば、少ない記述で大量の処理が行えるようになります。

たとえば JavaScript では次のような記述で「I love you.」を 100 回表示することができます。

```
var i;
for (i=0;i<100;i++){
   document.write(i," I love you.<Br>");
}
```

これは for 文と呼ばれるもので、類似のものが通常のプログラム言語では必ず存在します[*11]。JavaScript では for 文は、for(A;B;C){D} のようなちょっと複雑な形をしており、次のような意味になります。

1. Aを実行します。ここには通常代入文が入ります。
2. Bの条件を調べます。もし条件不成立の場合は for 文は終了し、次へ進みます。
3. Dを実行します。ここには任意の複数の文を書くことができます。
4. Cを実行します。ここには通常例のような式または代入文が入ります。
5. 2 番目に戻る。

Bが条件であること、CとDが書いてある順番と逆に実行されることに注意してください。上記の実例の場合次のような感じで実行されます。

1.「i=0」を実行するので変数 i の値がゼロになる。
2.「i<100」の条件を調べると、変数 i の値はゼロなので条件は成立する。
3.「document.write」を実行するので「0 I love you.」と表示される。
4.「i++」を実行する。これは「i=i+1」と同じ意味なので、変数 i の値は 1 になる。
5.「i<100」の条件に戻る。変数 i の値は **1** なので条件は成立する。
6.「document.write」を実行するので「1 I love you.」と表示される。
7.「i++」を実行する。変数 i の値は **2** になる。
8. 繰り返すたびに変数 i の値は増加していく。
9.「99 I love you.」と表示した後、変数 i の値は 100 になる。
10.「i<100」を満たさなくなるので for 文は終了する。

もし 1000 回「I love you」と出したい場合は、「i<1000」に変更します。i の値をどんどん減らしたい場合は、「i++」の代わりに「i−−」を使用します。これは「i=i-1」と同じ意味です。

4.15 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。

[*11] 次章の PHP にも for 文はあります。

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。

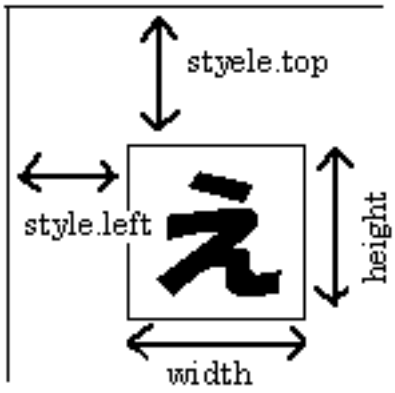
```
x=setInterval("関数名()",時間);
```

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。

4.16 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 4.7 のようなものがあります。



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 4.7 画像のプロパティ

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。

```
<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 type="text/javascript">
window.alert("表示されている画像の縦幅は、"+document.kousha.height+"です。");
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
```

4.17 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 4.1 に option に使用できるものを示します。

表 4.1 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[*12]。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。

```
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。

[*12] 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。

5. PHP

前章の JavaScript では、ブラウザで利用者の入力に応答する Web ページを作成することができました。しかしさらに利用者に対して高度な応答を行うためには、ブラウザ側だけでは無理があります。例えば Yahoo などのキーワード検索を行おうとしても、あらかじめ検索のデータベースの内容を全てブラウザに送ることは無理でしょう。また注文の受付のページでは入力された注文内容をサーバーの方で記録し、発注の処理に入らなければなりませんが、お客のブラウザからではそのようなことができません。

この章で扱う PHP は、1994 年に Rasmus Lerdorf 氏によって開発が始まったスクリプト言語[*13]です。もともとは「Personal Home Page」から名づけられたようですが、現在は個人の手を離れて「PHP: Hypertext Preprocessor」として多くの人々の手によって開発が続けられています。

PHP の特徴の一つは、サーバー側での処理を通常の HTML のファイルの中に混在させることができる点です。サーバーで処理した結果をブラウザで表示するためには、これまでどおり HTML による記述が必要です。そしてそのような部分はこれまでどおり記述し、サーバーからの応答の部分だけ PHP の記述するような形になります。

PHP の特徴として他に良く上げられるのは、様々なデータベースソフトと簡単に接続できる点です。さきほど例に挙げたキーワード検索や注文受付の処理では、通常データベースが利用されます。検索ならばあらかじめデータをデータベースに入れておき、その中から探します。また注文受付も、受け付けた注文の内容をデータベースに登録するのが普通です。このように大抵の処理ではデータベースの利用が必須となっていますが、PHP から簡単にデータベースを利用できるので、大抵の処理が簡単に記述できるということになります。データベースの操作法については次の章で説明します。

PHP についての参考サイトとしては本家のマニュアルのページが一番だと思います。ほとんどの部分が日本語になっていますし、使用例などもあります。

「PHP マニュアル」 https://www.php.net/manual/ja/index.php

5.1 どこに入れるのか

PHP 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。入れる場所は特に決まっていません。記述された内容は基本的にはファイルの最初の方に記述された方から実行されます。そして実行された結果がブラウザに送られます。ブラウザ側で「ソースの表示」を行っても PHP の記述は残っていません。

HTML の記述と PHP の記述が混ざらないように、PHP の記述は`<?php` と`?>`の間に入れます。つまり HTML 的には「?php」というタグの形をしています。終わりのタグではありませんから、`/?>`のように「/」を入れないでください。

```
<?php
   PHP の記述
?>
```

PHP を利用する際の重要な注意は、

- ファイルの拡張子は「.htm」ではなく「.php」にする。
- 動作を確認する際は必ずサーバーに送らなければならない。

と言う点です。拡張子が従来どおりの「.htm」では、PHP の記述は単なる変なタグと言うことでブラウザで無視されます。またサーバーが PHP の実行を行うので、パソコンに保存したファイルのアイコンをダブルク

[*13] 簡易プログラミング言語とも呼ばれるもの。通常実行のための変換を必要とせず、すぐにコンピュータで実行できる。

リックしたのでは PHP の記述の部分は動きません。peditor で作成した場合は、できたファイルは mars 上にありますので問題はありません。

5.2 表示の命令

まずは JavaScript の document.write に相当するものです。これを利用して後述の変数の内容を表示するだけでなく、通常のメッセージなどを HTML のタグを含めた形で表示することもできます。簡単な例を次に示します。

```
<HTML lang="ja">
<Head>
<Meta charset="UTF-8">
<Title>PHP の例</Title>
</Head>

<Body>
<?php
   echo "こんにちは<Br>";
   echo 'He said, "I love you.".<Br>';
   echo '<Font size="7">',3+4,"</Font>";
?>
</Body>
</HTML>
```

この例からわかるように何らかの表示を行いたいときは、PHP では「echo」と言う命令を用います。JavaScript の document.write は関数だったので () の間に表示したいものを入れましたが、こちらは命令なので () は不要です[*14]。また区切りの「;」は必ず必要なので忘れないようにしてください。

PHP で何か間違えるとエラーのメッセージが表示されます。例えば、先ほどの例で 9 行目にある二つ目の echo の行の最後の「;」を忘れると、「`Parse error:  parse error, unexpected T_ECHO, expecting ',' or ';' in ..../test.php on line 10`」のようなメッセージが表示されます。10 行目で「,」か「;」を期待していたのに、お呼びでない「echo」があったぜ、と言うような意味です。このように行の最後で間違えるとエラーの場所が次の行として表示されることもあります[*15]。

5.3 変数・計算式

PHP では変数を宣言せずに使用することができます。ただそれではどれが変数なのか見分けがつきにくくなるので、「$x」のように必ず先頭に「$」を付けます。また変数名には残念ながら漢字は使用できません。英数字[*16]にしてください。

変数が「=」の左側に出てきた場合は、変数に何か入れようとしていることを示しています。変数には数値や文字列などを入れることができます。その他の場合は変数に入っている内容を示しています。"123"と言うような文字列は計算式の中にあれば数値の 123 として扱われます。また必要があれば数値が文字列に自動的に変換されることもあります。このテキストの例の中で"〜"、'〜' や文字や数字で書かれている部分は、変数で置き換え可能です。逆に例の中で変数で書かれている部分は大抵変数でないと動きません。

[*14] PHP にも関数があります。PHP の関数を用いる場合は、JavaScript と同様に () の中に入れることになります。
[*15] 「}」を忘れると、はるかかなたでエラーになることもあります。
[*16] 英字または数字で、先頭は英字にしてください。

```
$x=999;
echo "x=",$x,"<Br>";  // x=999 と表示される。
$x="123"+456;
echo "x=$x<Br>";      // x=579 と表示される。$x の後が英数字の場合はうまく行かない。
echo "x={$x}<Br>";    // x=579 と表示される。$x の後に何が来てもこちらは大丈夫。
echo 'x=$x<Br>';      // x=$x  と表示される。
```

JavaScript では”〜”と’〜’ は全く同じ意味でしたが、PHP では少し違います。”〜”では変数の部分がその中身と置き換えられますが、’〜’ ではそのようなことがありません。なお、PHP では「//」以降はコメントとして行の終わりまで無視されます。

HTML の記述の中に PHP の出力を混ぜるというような場合は、「<?=」と「?>」の間に変数や計算式を入れることが可能です。これは「<?php echo 」と「?>」と同じ結果になります。以下のようにして変数$x や $h に入れた数値で幅や高さを設定することができます。

```
<Div style="width: <?= $w ?>; height: <?= $h ?>"> ... </Div>
```

計算式は JavaScript と同様に記述することができます。ただ「+」は本当に加算の意味でしかありません。文字列と文字列をくっつけたいときは「.」を使用します。

```
$yen=1234;
$kekka="お値段は".$yen."円です。<Br>";
echo $kekka;                    // お値段は 1234 円です。 と表示される。
```

5.4 条件判断

PHP の条件判断も JavaScript と全く同じ形をしています。() の中の条件の書き方も同様です。例えば「変数 a の内容が 10 に等しくない」は「$a!=10」のように書きます。「等しい」時には「$a==10」のように等号が 2 つになるのを忘れがちなので注意します。

```
if (条件) {
   条件が成立したときの内容
} else {
   条件が成立しなかったときの内容
}
```

条件が不成立の場合の内容がないときは、else 以下は省略できます。また else 以下に更に if が来る場合は次のように書くこともできます。

```
if (条件 1) {
   条件 1 が成立したときの内容
} else if (条件 2) {
   条件 1 は成立しないが、条件 2 が成立したときの内容
} else {
   条件 1 も条件 2 も成立しなかったときの内容
}
```

PHP が JavaScript と大きく異なるところは内容のところに HTML などの内容を直接記述する方法があることです。例えば次のような感じです。

```
if ($tokuten==100) {  // 得点の変数($tokuten)が100ならば画像を表示する。
?>
   <Img src="manten.gif"> <!-- ここはHTMLの世界 -->
<?php
}
```

つまり一度?>で PHP の記述を終わらせて通常の HTML のタグなどを記述することができます[*17]。ただ閉じる「}」などを忘れるとエラーになります。

5.5 繰り返し (1)

繰り返しと言うものを利用すれば、数行のプログラムで何億回も計算させると言うような事が可能です。たとえば PHP では for 文と呼ばれる次のような記述で「I love you.」を 100 回表示することができます。

```
for ($i=0;$i<100;$i++){
   echo $i," I love you.<Br>";
}
```

これはと類似のものが通常のプログラム言語では必ず存在します。PHP の for 文は、JavaScript の for 文と全く同じ形をしています。for(A;B;C){D} は、次のような意味になります。

1. Aを実行します。ここには通常代入文が入ります。
2. Bの条件を調べます。もし条件不成立の場合は for 文は終了します。
3. Dを実行します。ここには任意の複数の文を書くことができます。
4. Cを実行します。ここには例のような式または代入文が入ります。
5. 2 番目に戻る。

Bが条件であること、CとDが書いてある順番と逆に実行されることに注意してください。

もし 1000 回「I love you」と出したい場合は、「$i<1000」に変更します。i の値をどんどん減らしたい場合は、「$i++」の代わりに「$i−−」を使用します。これは「$i=$i-1」と同じ意味です。

繰り返している途中で、繰り返しを中断したい場合は、「break;」を入れます。単独でこれを用いると必ずそこで繰り返しが中断されますので、大抵 if と組み合わせます。また繰り返しの途中で、次の繰り返しへ進む場合は「continue;」を入れます。

```
for ($i=0;$i<100;$i++){
   if ($i==20) { break; }      // $iが20になったら繰り返しをやめて次へ進む
   if ($i==7) { continue; }    // $iが7の時は、次のechoに行かずに$i++へもどる
   echo $i," I love you.<Br>";
}
```

5.6 配列型変数

コンピュータで大量の情報を処理したい場合、それに応じて変数も大量に必要になります。そのような場合にそれぞれの変数に別の名前を与えるのは大変です。そこで配列型の変数と言うものがプログラミング言語では使えるようになっています。PHP では配列型の変数も変数ですので「$」から始まります。そして変数の名前の後に「[ ]」がありその中に添え字と呼ばれるものを指定します。添え字は数値に限定されるプログラミング言語もありますが、PHP では文字列も使用できます。

[*17] JavaScript ではこのような場合、document.write を使用する必要がありました。

```
$data[1]=123;$data[2]=456;$data[3]=789;
for ($i=1;$i<=3;$i++) {
    echo "$data[$i]<Br>";
}
$data['name']="三木";$data['address']="名古屋市";
echo $data['name'],"の住まいは",$data['address'],"です。";
echo "{$data['address']}の住まいは{$data['name']}です。";
```

普通の変数ではまずありえませんが、配列型変数では誤って存在しない変数を使ってしまうことがあります。例えば先程の例で$data[1]〜$data[3] については数値を入れてますが、$data[4] には何も入れていませんので、$data[4] は存在しません。うっかり for を回しすぎて、$data[4] を使ってしまうとエラーになります。次の節のフォームからの入力の際にも、選択されなかった場合、対応する変数が存在しないことがあります。変数が存在するかどうか確認する際は次のようにします。

```
if (isset($x)) {
   $x が存在し、NULL でないものが入っていた場合の処理
}
```

NULL は特殊な値で、通常これを代入して入れない限りまず変数には入りません。「isset($x,$y,$z)」のように複数の変数を指定すると、指定した全ての変数が存在したらと言う条件になります。

5.7 フォームからの入力

フォームの入力欄に利用者が入力した内容を PHP で処理するためには、一旦サーバーに入力した内容を送り返してもらわなければなりません。そのためにはクリックしたらフォームの内容を送信するボタンが必要になります。このボタンは HTML で次のように記述します。

```
<Input type="submit" value="送信">
```

value で指定した内容はボタンの上に書かれる文字ですので任意のものが可能ですが、利用者が最後にこれをクリックしなければならないことがわかるようなものにします。このボタンをクリックすると<Form>〜</Form>の間にあるものだけがサーバーに送られます。

またこれだけではサーバーに送った内容をどの PHP のファイルが処理をすれば良いのかわかりません。この処理する PHP のファイルの指定は、Form のタグのところで行います。

```
<Form method="POST" action="prog.php">
```

method については POST 以外に GET も指定可能です。GET の場合は URL の部分に入力された内容が追加されます。そのため GET の方はサーバーに送れるデータ量に制限がありますし、URL を見ると何を送ったのか判ってしまいます。action ではサーバーで処理する PHP のファイルを指定します。

指定された PHP のファイルの方では、特別な配列型変数を用いることにより、簡単にフォームの中で入力した内容を取り出すことができます。例えば「namae」と言う名前を付けた入力欄の値は、「$_POST['namae']」で取り出すことができます。もし GET で送った場合は、「$_GET['namae']」で取り出すことができます。

```
<HTML lang="ja">
<Head>
<Meta charset="UTF-8">
<Title>入力欄の例</Title>
</Head>

<Body>
<Form method="POST" action="prog.php">
名前：<Input name="namae"><Br>
<Input type="submit" value="送信">
</Form>
</Body>
</HTML>
```

この入力欄の例には PHP の記述は何も含まれて居ませんので、通常の拡張子が.htm のファイルに入れます。(例えば「input.htm」) そして次のは必ず「prog.php」と名前を付けて保存してください。(上記の例で action=でこの名前を指定しているので。)

```
<HTML lang="ja">
<Head>
<Meta charset="UTF-8">
<Title>入力欄処理の例</Title>
</Head>

<Body>
<?php
   echo "名前には、{$_POST['namae']}が入っていました。<Br>";
?>
</Body>
</HTML>
```

パスワードの入力のように、入力された文字が別の記号に置き換えられて表示される入力欄があります。これは次のような type の指定を追加すればできます。

```
<Input type="password" name="pass">
```

表示は記号に置き換えられますが、Web サーバーには入力されたものがそのまま送られますので、http では内容を盗まれる可能性があります。

入力欄には文字や数字などを自由に入力できますが、長さを指定しても 1 行と言う制限があります。より長い文章などを入力するために TextArea と言うタグがあります。このタグは次のように使います。

```
<TextArea name="namae" cols="40" rows="4">
あいうえお
</TextArea>
```

cols で 1 行の文字数、rows で行数を指定します。タグの間に入れたもの (上記の例では「あいうえお」) が長方形の入力欄の中に最初から入りますので、空っぽの入力欄でよい場合は何も入れないようにします。入力された内容を PHP で取り出す方法は通常の Input タグと同じです。

他にフォームで使えるものとしてラジオボタン、チェエクボックス、選択メニュー等があります。これらの選択状況は次のようにして知ることができます。

- ラジオボタンの場合：

```
<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>
```

このような場合、選択したものは$_POST['sex'] の内容でわかります。この場合変数の内容は、「男」、「女」、「宇宙人」のいずれかになります。

- チェックボックスの場合：

```
<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>
```

このような場合例えば$_POST['sugar'] があるかどうかで選択したかどうか分かります。通常次のように isset() を使います。

```
if (isset($_POST['sugar'])) {
   砂糖を選択していた場合の処理
}
```

- 選択メニューの場合：

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

このような場合、選択したものは$_POST['selone'] の内容でわかります。この場合変数の内容は、「sleep」、「eat」、「run」のいずれかになります。

- 選択メニュー (複数選択可) の場合：
次のように name で指定する名前の後に [ ] を追加します。

```
<Select name="selmul[]" multiple>
   <Option value="sleep">寝る
   <Option value="eat" selected>食べる
   <Option value="run">走る
</Select>
```

このような場合、最初に選択したものは$_POST['selmul'][0] の内容でわかります。そして 2 つ選択した場合は、2 つ目に選択したものが$_POST['selmul'][1] の内容でわかります。やっかいなのは 1 つしか選択しなかった場合は$_POST['selmul'][1] は存在しないので isset() で確認をしてから内容を見る必要があることです。

5.8 URL で入力内容を送る

URL に入力内容を付けて送ることができます。この方法を利用するとブラウザの URL を表示するアドレスのところに入力内容も表示されてしまうので、知られては困るような内容を送ることには使えません。また、

余り大きなデータは URL の長さの制限に引っかかる恐れがあるので使えません。それでもリンクをクリックするだけで送るようなことも可能なので、この方法を利用する場面も少なくありません。いくつかの送り方がありますが、それを受け取る PHP の方は同じ方法で受け取ります。

まず一つ目の方法ですが、Form タグの method を GET にする方法です。前節の例で

```
<Form method="POST" action="prog.php">
```

となっていたところを

```
<Form method="GET" action="prog.php">
```

にするだけです。

2 つめの方法として URL に直接記述する方法です。例えば次のようにします。

```
<A href="prog.php?namae=aiueo">名前は aiueo</A>
<A href="prog.php?namae=aiueo&age=20">名前は aiueo、年齢は 20 歳</A>
```

1 つ目の例では、クリックすると「namae」という入力欄に「aiueo」と入力して prog.php を呼び出すのと同じ事になります。2 つ目の例は、2 つの入力欄に対応した場合で、「&」を追加してより多くの入力欄にも対応できます。この方法で注意しなければならないのは、送る内容に記号や漢字が含まれる場合は、それらを文字コードに置き換える必要があることです。幸い PHP には urlencode と言う変換をする関数があるのでこれを利用します。

```
echo '<A href="prog.php?namae=',urlencode("椙山花子"),'">名前は椙山花子</A>';
```

PHP 側でこのような方式で送られてきたデータを取り込むには、$_POST['namae'] の代わりに$_GET['namae'] を使います。

5.9 ファイルへの入出力

利用者からの入力を保存するためにはファイルを作成して、そこに入力内容を入れなければなりません。また逆にファイルに入っている情報を取り出して表示するようなこともしばしば行われます。ここではそのようなファイルに関する操作をどのように PHP では記述するのかについて説明します。

ファイルを開く

ファイルを利用するためにはまず「ファイルを開く」という操作が必要になります。このとき、これから扱うファイルの名前とファイルに対してどのような操作を行いたいかを指定します。

```
$file=fopen("aaa.txt","w");
```

fopen が「ファイルを開く」関数です。fopen の最初に指定しているのがこれから操作するファイルの名前です。この例では「aaa.txt」という名前のファイルが対象となります。PHP のすごいところは、このファイルを指定しているところに URL を書けば、他のサーバーにあるファイルを読み込むこともできることです。さすがに書き込むことはできませんが。インターネットには Web ページの形で非常に多くの有益な情報があります。そのような Web ページを開き、取り込み、その中の本当に必要な部分だけ取り込むということが PHP では比較的容易に行うことができます。

次に”w”という指定がありますが、これはファイルに情報を書き込む (write) ことを示します。もし「aaa.txt」がなければ、ここで「aaa.txt」と言う空のファイルが作成されます。もし既に存在した場合は、これまで入っていた内容は消されてしまうので注意します。

これまでファイルに入っていた内容に追加 (append) をしたい場合は、”w”の代わりに”a”を指定します。ファイルがなかった場合は”w”を指定したのと同じ動作になります[18]。

ファイルの内容を読み出し (read) をしたい場合は、”r”を指定します。もし存在しないファイルに対してこの指定をすると fopen は「false」と言う特殊な値を返し、以下の読み出しはできませんのでご注意ください。

fopen の返す値をこの例では$file という変数に入れています。これは後で使用しますので、適当な名前の変数に必ず入れてください。なお、同時に複数のファイルを扱うことができますが、その場合は fopen の結果をそれぞれ異なる変数に入れてください。

ファイルへの書き込み

ファイルへ何かデータを書き込む場合は、次のように fwrite を使用します。

```
fwrite($file,"書き込むデータ¥n");
```

この例では先ほど fopen で開いたファイルに「書き込むデータ」と言う文字列と改行 (¥n) が書き込まれます。echo と同様に変数を指定すれば変数の内容を書き込むこともできます。fwrite の () の中には一つしか書き込みたい内容を指定できないので、複数ある場合は、文字列の結合を示す「.」でつなぎます。書き込みたい内容が多数ある場合は fwrite を必要な数だけ繰り返し記述します。複数のデータを書き込む場合は、データとデータの間に空白や改行が入るようにします。例えば「123」と「456」を続けてファイルに書き込むとファイル中には「123456」が残るので、後でこれを読み込んでもどこで区切れば良いのかわからなくなります。次のようにファイルに書き込むと、

```
$a=123;
fwrite($file,"$a¥n");
fwrite($file,"ありがとう¥n");
```

ファイルには次のような内容が入ります。

```
123
ありがとう
```

ファイルからの読み出し

ファイルからデータを読み出すには、次のように fgets を用います。

```
$data=fgets($file,256);
```

これで先ほどの fopen で開いたファイルより 1 行分読み出され、変数$data に入ります。256 はデータの最大長です。もしファイルにこれより長い行があった場合は、とりあえずここで指定した分だけ変数に入ります。この fgets を繰り返すことによりファイルの先頭から順番に 1 行ずつ読み出すことができます。

[18] PHP 本来の機能ではこれで説明終わりなのですが、実際に使用してみると「Permission denied」と言うエラーが生じることがあります。これは許可がないと言う意味で、Web サーバーのソフトが PHP を動かしてファイルを作成しようとしたところ、利用者の領域だったのでファイルを作成できなかったためです。そのような事が生じないように ruid2 と言う Web サーバーのモジュールでこの問題を回避するように設定してあります。

ファイルを閉じる

ファイルに関する操作が終わったら次のようにしてファイルを閉じます。ファイルの読み出しの際は大きな問題になることは少ないですが、書き込みの際に忘れると色々問題を生じることがあります。

```
fclose($file);
```

ファイルの内容の確認

peditor でファイルを編集することにより、ファイルに書き込んだ内容を確認することができます。また mars に接続しメニューの「アクセサリ」にある「Kate」でファイルの内容を見たり、編集したりすることができます。

ファイルと配列型変数のやりとり

通常はファイルから行単位で取り出したり入れたりするのですが、PHP にはファイルの内容を配列型変数に一気に取り込んだり、逆に配列型変数の内容を一気にファイルに入れることができます。

```
$lines=file("aaa.txt");
$lines2=file("aaa.txt",FILE_IGNORE_NEW_LINES);
```

これで「aaa.txt」という名前のファイルの内容が 1 行毎に$lines 配列型変数に入ります。つまり「aaa.txt」の 1 行目が$lines[0]、2 行目が$lines[1] に入ります。2 つ目の例では$line2 配列型変数に「aaa.txt」の内容から各行にあった行末の改行 (￥n) を除いたものが入ります。ファイル名の代わりに URL を指定すると、他のサーバーのファイルを取り込むこともできます。

逆に配列型変数または普通の変数$lines に入っている内容を「bbb.txt」と言うファイルに一気に書き込むのであれば、

```
file_put_contents("bbb.txt", $lines);
```

とします。このとき「bbb.txt」が存在しなければ新たにファイルが作られます。また既存の場合は、入っていた内容は消えて$lines の内容と置き換わります。

5.10 繰り返し (2)

for 文以外にも while 文と言われ繰り返し文があります。

```
while（条件）{
   条件が成立している間に繰り返し実行する内容
}
```

for 文でも時々終了しない繰り返しになって困りますが、while 文は繰り返される内容の中で条件が変化するようなことをしないと簡単に終わらないものになってしまうので注意します。また、do–while 文と呼ばれる繰り返し文もあり、次のような形をしています。

```
do {
   条件が成立している間に繰り返し実行する内容
} while（条件）;
```

while 文と異なるのは条件を調べるのが実行をしてからと言う点です。つまり繰り返し実行される内容は少なくとも一回は実行されます。

あらかじめ何行分のデータがファイルにあるとわかっている場合は、その数だけ fgets を書けば良いのですが、実際はどのくらい入っているかわからない例も多数あります。そういう場合はファイルから読み出しができなかった場合に fgets が「false」と言う値を返すのを利用して次のような while による繰り返しを使います。

```
while ($data=fgets($file,256)) {
   $data に読み出されたデータの処理
}
```

前節で file 関数を使用してファイルの内容を全て配列型の変数に取り込めると言う説明がありましたが、ファイルの大きさが不明の場合、配列変数の何番目まで入っているのか分かりません。そういう場合は次のような繰り返しを使用します。

```
foreach ($lines as $data) {
   $data に読み出されたデータの処理
}
```

foreach が配列型変数$lines から順番に一つずつ取り出して$data 変数に入れて、{ }の部分を実行してくれます。全て取り出しが終わったら繰り返しが終了します。

5.11 ファイルとディレクトリの操作

ファイルの中身とのやり取りの方法は既に述べましたが、ここではファイル自体の操作について説明します。また複数のファイルに対して何か操作を行いたい場合、それが入っているディレクトリは判っているが、どのような名前のファイルがいくつあるのか判らない場合もあります。そのような場合はディレクトリの情報を取り出して調べることになります。

ファイルの存在確認

指定した名前のファイルが存在するかどうかを調べることができます。ファイルからデータを読み込む場合に、ファイルが存在しない場合 fopen の際にエラーが発生します。エラーを避けるためには file_exists 関数で確認してから fopen するようにしましょう。

```
if (file_exists("aaa.txt")) {
   aaa.txt が存在した場合の処理
}
```

またファイルが存在しない場合に処理をしたい場合は、file_exists の前に「!] を付けると条件を反対にすることができます。

ファイル名の変更

ファイルの名前を変更したい場合は次のように rename を使います。この例の場合、「aaa.txt」の名前が「bbb.txt」になります。ただし元のファイル (aaa.txt) が無い場合はエラーになりますし、変更先のファイル (bbb.txt) が既に存在している場合もエラーになりますのでご注意ください。

```
rename("aaa.txt","bbb.txt");
```

ファイルの削除

不要になったファイルを消すことができます。次のように unlink を用いて指定したファイルを削除することができます。

```
unlink("aaa.txt");
```

ディレクトリの読み出し

ディレクトリを開いて、その中にあるファイルを調べることができます。以下は PHP のマニュアルに出ていた例を簡略化したものです。

```
$d=opendir(".");
while ($file=readdir($d)) {
   echo "ファイル名: ",$file,"<Br>\n";
}
closedir($d);
```

1. opendir 関数で指定したディレクトリ (".": カレントディレクトリ) を開きます。ファイルを開いた時と同様にその結果を変数$d に入れています。
2. readdir 関数で、ディレクトリ内のファイル名を一つ取り出します。readdir 関数を繰り返すことによりディレクトリ内の全てのファイルの名前を取り出すことができます。
3. opendir 関数で開いたディレクトリは、closedir 関数で閉じてください。

5.12 文字列関数

コンピュータは、数値の計算をするから電子計算機と呼ばれたのですが、現在の使われ方は計算以外の仕事の方が多いようにも思われます。数値以外のデータとして例えば文字列があります。ここでは PHP で文字列を扱う際に使われる関数のいくつかを紹介します。

- **strlen 関数**：を求める関数です。文字列の長さは、文字列を構成する文字の数です。

  ```
  echo strlen("abcdefg");    // 7 と表示される
  ```

- **strpos 関数**：文字列の中に、指定した文字列が含まれるかどうか調べる関数です。最初に指定した文字列の中に、次に指定した文字列が含まれれば、何文字目からかを答えます。もし含まれて居なければ false を返します。

  ```
  echo strpos("abcdefg","cde");    // 2 と表示される
  ```

  実は先頭をゼロと数えるので上記の例では 3 ではなく 2 が表示されます。PHP ではゼロと false は同等に扱われます。そのために含まれなければ何かすると言う場合は次のような書き方をします。

  ```
  if (strpos($x,"abc")===false) {
      $x に abc が含まれなかった場合の処理
  }
  ```

  等号が 2 つでも妙な感じでしょうが、等号が 3 つ必要となります。これの否定の条件 (含まれた場合) は「!==」になります。

- **substr 関数**：文字列の一部を切り出す関数です。何文字目から切り出すのかと、何文字分切り出すかを指定します。何文字分の指定が省略された場合は、指定された文字以降全てが切り出されます。

```
echo substr("abcdefg",2,3);   // cde が表示される
echo substr("abcdefg",4);     // efg が表示される
```

何文字目かの指定の際も先頭の文字がゼロなので注意します。また何文字目かのところを負の数にすると、文字列の後ろから指定することもできます。

```
echo substr("abcdefg",-3,2);   // ef が表示される
```

- **str_replace 関数**：文字列の置き換えをする関数です。探す文字列、それを置き換える文字列、探して置き換えられる文字列の 3 つを指定する必要があります。置き換える文字列が複数含まれていても、全て置き換えてくれます。

```
echo str_replace("ab","x","abcdabc");   // xcdxc が表示される
```

同時に複数のものを探して、各々別のものに置き換えることもできます。次の例では”ab”が”x”に、”c”が”y”に全て置き換えられます。

```
echo str_replace(["ab","c"],["x","y"],"abcdabc");
          // xydxy が表示される
```

また以下の操作は既にやりました。

- **文字列の連結**：文字列と文字列を連結したい時には「.」を使います。
- **文字列の比較**：二つの文字列が等しいかどうか調べる時は「==」、等しくないかどうかを調べる時は「!=」を使用します。アルファベット順で前にあるものが、より小さいと判断されるので、「<」や「>」も使用可能です。

なお、処理する対象の文字列が、半角の英数字だけでなく、日本語の漢字などを含む場合はそういう文字に対応できる関数を用います。例えば「strlen(”あいう”)」の結果は 9 になります[*19]。一方表 5.1 にある「mb_strlen(”あいう”)」ならば結果は 3 になります。文字コードが不明の場合は、「mb_strlen(”あいう”,”UTF-8”)」のように正しい文字コードの指定も追加しなければならないので、日本語限定でも結構面倒です。

表 5.1 日本語対応の文字列関数

	半角英数字用	日本語を入れると	日本語にも対応
長さ	strlen(”abc”) → 3	strlen(”あいう”) → 9	mb_strlen(”あいう”) → 3
位置	strpos(”abc”,”b”) → 1	strpos(”あいう”,”い”) → 3	mb_strpos(”あいう”,”い”) → 1
切り出し	substr(”abc”,1,1) → ”b”	substr(”あいう”,3,3) → ”い”	mb_substr(”あいう”,1,1) → ”い”

[*19] 文字コードが UTF-8 の場合。EUC-JP などの場合は 6 になります。

5.13 他のページへの移動

PHP で内部的な処理を行った後、何も表示しないで別のページへ移動したい場合は、次のような header() を使用します。

```
header("Location: https://www.sugiyama-u.ac.jp/");
```

この例では椙山のトップページに移動しますが、同じディレクトリにある別のファイルでも構いません。その場合はファイル名のみ指定します。使用上注意することは、これ以前に echo などでブラウザに他の内容を送ると無効になる点です。<HTML>などのタグを<?php の前にうっかり書いてしまう事が多いので注意しましょう。

条件によって移動したり、しなかったりする場合は、header() の次に exit; を入れておくと、ここで PHP の処理が終わって以降の部分は実行されなくなります。そのため大きな else が不要になって見やすくなります。

5.14 Web ページ間の情報のやり取り

Web ページの間で情報をやり取りする必要がある場合、表 5.2 のようにいくつかの方法があります。状況に合わせて方法を一つ選択したり、複数の方法を併用します。

表 5.2 Web ページの間で情報をやり取りする方法

状況	方法	受け取る変数
submit の際に追加情報を送る	hidden を使う	$_POS[...] または $_GET[...]
クリックしたリンク先に情報を伝える	URL に情報を付ける	$_GET[...]
以前来た事があることを伝える	cookie を使う	$_COOKIE[...]
同じサーバーの Web ページ間で情報の共有	session を使う	$_SESSION[...]
同じサーバーの Web ページ間で過去の情報の共有	ファイルやデータベースを使う	–

URL に情報を付ける方法は既に 5.8 で説明しました。ファイルの扱いについては既に 5.9 で説明しました。データベースについては次章に説明があります。

hidden を使う方法

<Form>〜</Form>の間に入力用のタグを入れることにより、利用者が入力した内容を action で指定した Web ページに送ることができます。この際に利用者が入力した以外の情報も送ることができます。例えば次のようなタグを<Form>〜</Form>の間に入れることにより、action で指定した Web ページでは$_POST['id'] または$_GET['id'] で「abc123」という情報を受け取ることができます。

```
<Input type="hidden" name="id" value="abc123">
```

このとき、この入力欄はブラウザの画面に表示されないため、Web ページのソースを確認しない限り利用者は気が付きません。逆に言えばソースの表示で送る内容がばれてしまうので、利用者に知られては困る内容を送る際には使えません。ブラウザの画面に表示されないという点が異なるだけで、他は type の指定のない<Input>と同じですので、JavaScript で値を読み取ったり、書き換えたりすることもできます。

cookie を使う方法

Web サーバーから Web ページの内容を送る際に、cookie と呼ばれる情報も送ることができます。cookie を受け取ったブラウザは、次回**同じディレクトリにある Web ページ**へアクセスする際にこれを送り返します。よって初回のアクセスの際に cookie を送っておけば、次にアクセスがあった場合に cookie の有無で二回目以降かどうかわかります。

特に期限を設定しない場合は、ブラウザを終了すると cookie は消えます。よって次の日以降の再訪問を調べたいのであれば期限を設定します。一方パスワード認証を通過した証拠として cookie を使用する場合は、期限を設定しないことにより、正しく終了処理 (cookie の消去) をしなかった際に、他の利用者が使ってしまうことをある程度防ぐことができます。

まず「namae」という名前の cookie に「12345」を設定したい場合は次のように setcookie() を使用します。ただし header() と同様に他の出力が実行される前に使用する必要があります。後になると cookie が設定されません。

```
setcookie("namae","12345");
```

名前が違えば setcookie() を複数用いて複数の cookie を設定することができます。この例では有効期限の設定がないので、ブラウザを閉じるまで有効になります。期限を設定する場合は、現在の時刻に期限までの秒数を加えたものを指定します。現在の時刻は time() で得ることができるので次のような形になります。

```
setcookie("namae","12345",time()+3600);
```

これでページを表示してから 1 時間後まで有効になります。期限が表示してからではなく、ある決めた日までと言うのであれば、指定した日時を秒単位に変換する mktime() を使用して次のようにします。

```
setcookie("namae","12345",mktime(13,25,45,6,30,2023));
```

mktime() のところに数字が並んでいますが、これで 2023 年 6 月 30 日 13 時 25 分 45 秒と言う期限が設定されます。順番がわかりにくいですが、時・分・秒・月・日・年の順番に指定します。一方有効期限としてゼロを指定すると cookie を消去することができます。このとき設定する値としては何を指定しても無視されますが省略はできません。

```
setcookie("namae","",0);
```

このように設定した cookie の値は、`$_COOKIE['namae']` のような形の配列型の変数で取り出すことができます。取り出しはできても違う値を設定することはできませんので注意します。設定する際は setcookie() を使います。また cookie が設定されていない場合にこの変数を使うとエラーになるので、必ず isset() で存在を確認してから使いましょう。

session を使う方法

これも内部的には cookie を使用していますが、記憶する内容は Web サーバー上にあるので、内容が利用者にばれる恐れが低くなります。使い方は最初に `session_start()` をすれば、`$_SESSION['namae']` のような配列型変数が使えるようになります。ただし setcookie() と同様に他の出力が実行される前に `session_start()` を使用する必要があります。この変数に入れたものは、別の web ページでも取り出すことができるようになります。

```
<?php
session_start();  // <HTML>より先にする
?>
<HTML lang="ja">
<Head>
... 中略 ...
<?php
$_SESSION['namae']=12345;          // 他のページで 12345 を取り出せるようになる
echo $_SESSION['bango'],"<Br>\n"; // 他のページで入れたものが読める
```

このように作成した変数を削除したい場合は次のように unset() を使用します。通常の PHP の変数はそのページを表示している間しか存在しません。よってわざわざ変数を削除する必要はまずありませんが、$_SESSION[ ] は他のページで取り出せるようにずっと残るので、削除した方がわかりやすいこともあります。存在しない変数を unset() してもエラーにならないので注意します。

```
unset($_SESSION['bango']);  // 「,」で区切って複数の変数を一度に削除可能
```

session の標準的な設定は Web サーバーで行われていて、状況によっては問題を引き起こすことがあります。例えば mars では session で使用する cookie の有効期限が 3 時間になっているので、それ以上アクセスの間隔が開くと session で設定した情報が失われてしまいます。また cookie の設定場所が web サーバーのトップになっているので、同じサーバ内で違うことに session を使おうとしても、同じ cookie を使うために設定内容が混ざってしまいます。session で設定した情報は web サーバーの中にファイルの形で保存されますが、その場所も共通になっています。このファイルはほっておくとどんどん貯まるので、設定次第では自動的に消すようにもできますが、共通の場所を利用している場合、誰かが自動的消去の設定をすると一緒に消されてしまう恐れがあります。これらの設定を自分の今使っているディレクトリのみ変更することができます。そのためには「.htaccess[*20]」と言う名前のファイルを作成して、次のような内容を入力します。

```
# Cookie を 100 日後まで有効に
php_value session.cookie_lifetime  8640000
# session ファイルを 100 日後まで有効に
php_value session.gc_maxlifetime  8640000
# このディレクトリ以下で session 情報を共有する
php_value session.cookie_path  "./"
# session ファイルは ses ディレクトリの中に入るので、ses と言うディレクトリを作っておく
php_value session.save_path  "./ses/"
# 期限を過ぎた session ファイルを削除するのは 100 回に 1 度
php_value session.gc_probability   1
php_value session.gc_divisor     100
```

「#」で始まる行はコメントなので入力しなくても構いませんが、入れておいた方が設定した内容がわかりやすくなります。この.htaccess をブラウザでアクセスされて見られると、大事な設定内容がばれてしまうので、WWW サーバーがアクセスを禁止しています。

この.htaccess の例では session_start() を 100 回使うと古い session ファイルを削除します。session.gc_probability をゼロにするとファイルがどんどんたまるようになるので、何らかの別の方法で消すようにしないと、忘れた頃にディスクが溢れて慌てることになります。

[*20] 利用者が設定変更するためのファイル名はシステムの設定によって変わります。解説の Web ページなどによく出てくる名前は php.ini や.user.ini になっています。その場合コメントの記号や書き方が少し変わります。システムの設定によって、利用者が設定変更できないようになっていることもあります。

5.15 送信前のチェック

人は誰でも間違いを犯します。Web ページの入力に誤りがあった場合、その誤りが機械的に検出可能であっても、PHP は送信ボタンをクリックして入力内容がサーバーに来ない限り対応できません。そして誤りがあったことをまた送り返さないと利用者には伝わりません。ネットワークの往復をする代わりに、送信ボタンをクリックすると、まず JavaScript でチェックを行い、問題が無ければ本当にサーバーに送信することが考えられます。ネットワークを使用しない分即座に応答を返すことができます。

そのためには、色々手直しする必要があります。元は次のようになっていたとします。

```
<Form method="POST" action="syori.php">
... 中略...
<Input type="submit" value="送信">
```

Form のタグには、「name=”bbb”」のように名前を付けます。これは後で JavaScript からこのフォームに対して送信の指示ができるようにするためです。また、「type=”submit”」のボタンを「type=”button”」に変更し、クリックしても即座に送信されないようにします。さらに onClick を利用して内容をチェックする関数を指定します。そうすると次のような感じになります。

```
<Form method="POST" action="syori.php" name="bbb">
... 中略...
<Input type="button" value="送信" onCLick="check()">
```

ここでは内容をチェックする関数として check() を指定していますが、もちろん check() の内容も定義しなければなりません。入力欄などの内容を調べて、何か問題などがあれば、そのことを警告画面などを使用して利用者に知らせてから return 文を実行するか、関数を終了します。入力の内容に問題がなければ、次のようにしてフォームの submit() を呼び出すことによって、入力された内容をサーバーへ送ることができます。

```
function check(){
   ... 中略...
   document.bbb.submit();   // bbb はフォームにつけた名前
   ... 後略...
}
```

6. SQL

SQL (Structured Query Language) は**関係データベース** (Relational Database) を操作するための言語です。データベースはどのようなデータをどのように記憶するかによって様々な方式が考えられており、関係データベースは IBM 社の E.F.Cood によって 1969 年に提案されました。関係データベースでは、データは二次元の表 (テーブル: table) の形で記憶されます。複数のテーブルの間に関係 (relation) を設定することにより、データ間の複雑な関係を扱うことができます。1 年生の「コンピュータと情報 II」に出てきた「Access」も関係データベースを基にしたデータベースソフトです。他にも関係データベースを基にしたデータベースソフトは多数ありますが、大抵 SQL が使えるようになっています。SQL を使用すると、データベースを作成し、データを入れ、データを修正し、データを検索するなどの操作を行うことができます。「Access」の使い方を学んでも「Access」しか使えませんが、SQL を学べば様々なデータベースソフトが使えるようになります。

関係データベースにおいてデータを入れる**テーブル**は、図 6.1 のように「コンピュータと情報 II」に出てきた Excel の「リスト」と同じ形をしています。1 番上の行が列の見出しで、その下に値が並んでいます。1 行が 1 件のデータでこれを**レコード**と呼びます。テーブルと「リスト」の大きな違いは主キーの有無です。テーブルには必ず**主キー** (primary key) と呼ばれる列があり、そこにある値は全て互いに異なる必要があります。この列の値を用いてデータベースは各レコードを識別します。

フィールド名 →	id	name	pass
レコード →	1	miki	12345
レコード →	2	maki	abcde
レコード →	3	mika	abc123

↑主キー

図 6.1 テーブルの例

SQL を用いてデータベースに対して様々な操作を行うことができます。表 6.1 はデータベースの操作を行う際によく使われる文の一覧です。

表 6.1 SQL の文の一覧

名前	操作内容
CREATE	テーブルの作成
SELECT	データの検索
INSERT	テーブルへ新しいレコードを追加する
DELETE	テーブルのレコードを削除する
UPDATE	テーブルのデータを更新する

6.1 DB Browser for SQLite

以下実際にデータベースを操作しながら、関係データベースや SQL を理解できるように、簡単に SQL を実行できるアプリケーションとして DB Browser for SQLite (以下 DB Browser) を使用します。これは SQLite と呼ばれる関係データベースソフトを視覚的に操作することができるものです。SQLite は小規模なデータベースを構築するためのデータベースシステムで、同時に複数の利用者で扱うことを想定していません。その代わりに一つのデータベースはテーブルがいくつあってもファイルが一つであり、利用者管理に関するところがなくて扱いやすくなっています。PHP は SQLite を内蔵しており、PHP が使えるならば必ず SQLite も使えます。また Android も SQLite を標準的に使用しています。

DB Browser は mars のデスクトップのメニューの中の「プログラミング」の中にあります。起動すると図 6.2 のような画面になります。

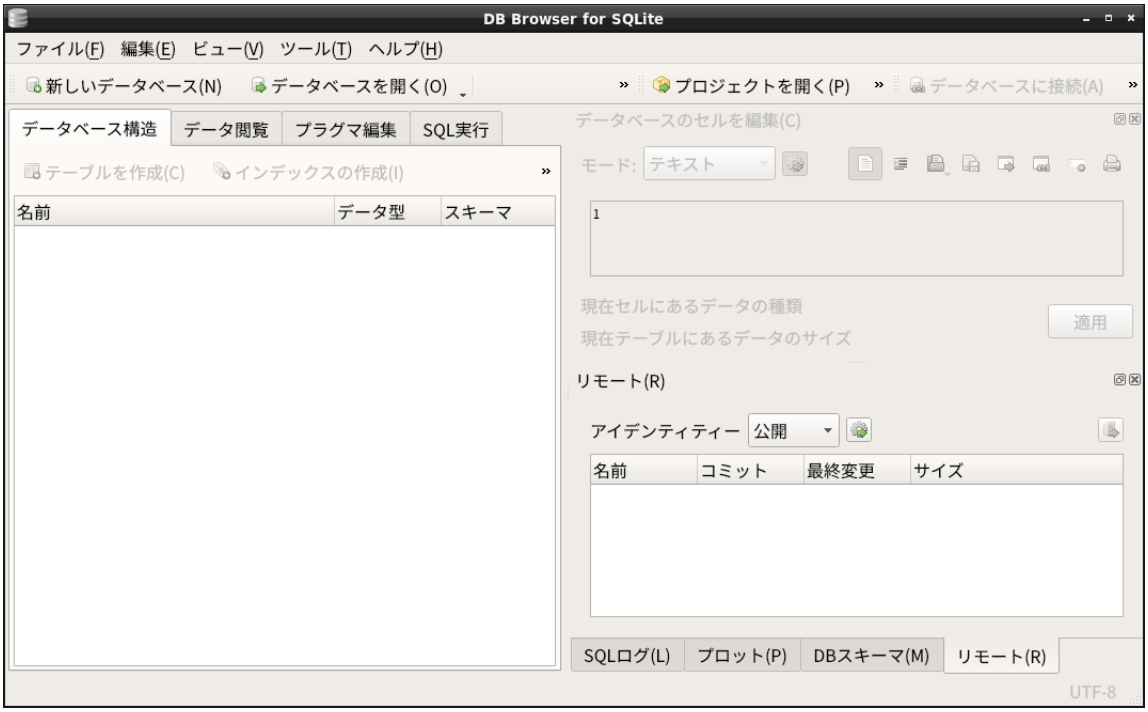


図 6.2 DB Browser for SQLite の最初の画面

まずデータベースファイルを指定します。既にデータベースがあるならば「データベースを開く」、新規のデータベースであれば「新しいデータベース」をクリックし、保存先 (例えば「デスクトップ」) を開き、ファイル名は「aaa.db」などにします。実は拡張子は特に決まっていません。データベースファイルを PHP で使用する場合は、PHP のファイルと同じディレクトリの方がわかりやすいと思いますが、直接データベースファイルをアクセスされると問題があるので注意します。

6.2 テーブルの追加

新しいデータベースファイルを指定すると、データを入れるテーブルを作成する図 6.3 のような画面が出ます。2 つ以上のテーブルを作成する場合は「テーブルを作成」をクリックすると同じ画面になります。ここでテーブルの名前とフィールドを設定します。

テーブルの名前は①の欄に英数字で入力します。次に②の「追加」ボタンをクリックするとフィールドの設定が一つ入れられるようになります。「名前」のところに英数字による列の名前を入力し、「データ型」は「Text」、「 Integer」、「Real」などから選択します。文字列ならば「Text」、小数点のない数値は「Integer」、小数点付きの数値は「Real」になります。SQLite はデータ型に関してはかなり適当で他のデータベースシステムではもっと細かい設定をすることになります。主キーの列ならば「PK」をチェックします。さらにデータを追加するたびに自動的に異なる数値を入れるならば、「AI」をチェックします。この autoincrement の機能により主キーに異なる値を簡単に入れることができます。この主キーの設定は一つのテーブルにつき一つないといけません。フィールドを全て追加したら「OK」をクリックしてテーブルの定義を保存します。表 6.2 のような設定の「student」と言う名前のテーブルを追加してみましょう。

作られたばかりのテーブルには何も入っていません。DB Browser でテーブルの中にデータを入れたり、中のデータを修正したり、削除したりすることができます。図 6.4 のように①の「データ閲覧」のタブをクリックして、②のところでデータを扱いたいテーブルを選択すると現在のテーブルの内容が表示されます。修正し

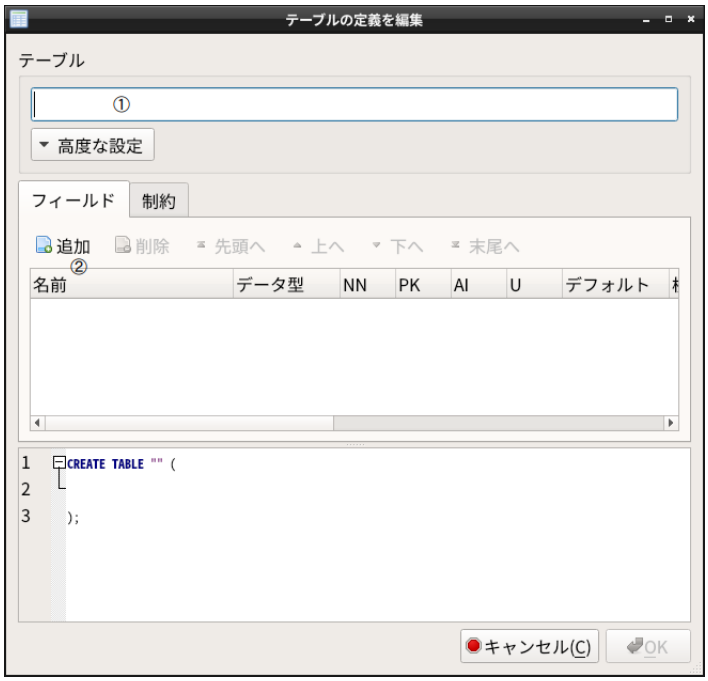


図 6.3　テーブルの設定画面

表 6.2　テーブルの例

名前	型	オプション
id	Integer	PK, AI
name	Text	
pass	Text	

たいところをクリックすれば内容を変更することができます。また③のボタン[*21]をクリックすると新しいレコードを追加することができます。

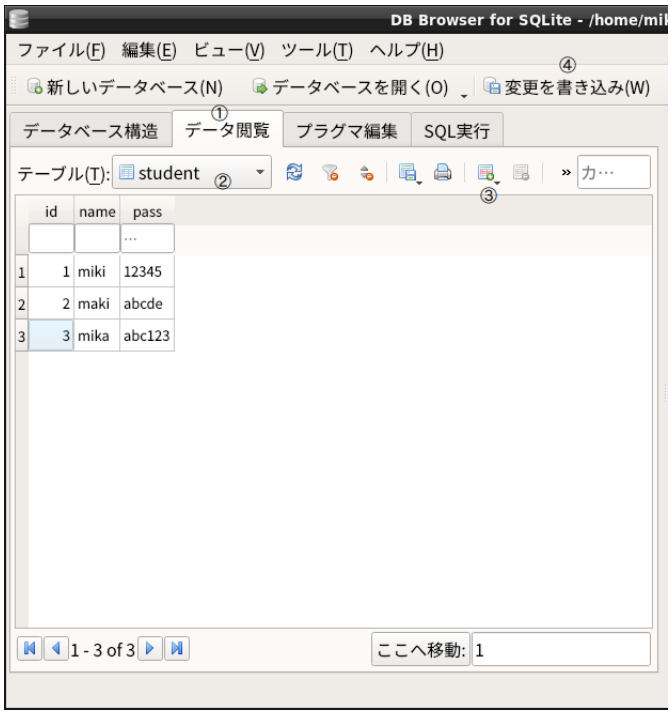


図 6.4　テーブルのデータ編集画面

autoimcrement の設定がある「id」には数字が自動的に入りますが、それ以外の項目には「NULL」という空を示すものが入ります。修正したいところをクリックしてデータを入れることができます。右隣りの項目に

[*21] このボタンが表示されていない場合は、DB Browser のウィンドウを横に伸ばすか、右の方にある「>>」をクリックすると更にメニュー項目が表示されるので、「新しいレコード」→「新しいレコード」と選択すると新しいレコードが追加されます

引き続き入力したい場合はTabを押します。図 6.1 と同じ内容を入れてみましょう。

データの入力や修正、レコードの追加や削除を行ってもすぐにはデータベースのデータは書き換わりません。図 6.4 の④の「変更を書き込み」が黒くなっているときはまだ書き換わっていないので、ここをクリックしましょう。クリックするまでは書き換わらないだけでなく、データベースファイルを他のプログラムから触れないようにロックもかけているので注意しましょう。

6.3 SELECT 文

SQL の中で最もよく使われるのが SELECT 文と言われるものです。これによってデータベースの検索が行えます。またここで扱う検索条件は、データの更新や削除の際にも利用します。つまりどのデータを更新したり削除するのかを指定する際に検索条件を設定しなければ、全てのデータが書き換わったり、削除されてしまいます。

まず一番基本的な形の SELECT 文の例は次のようになります。SQL のキーワードとなる語は大文字です。

```
SELECT * FROM student
```

SELECT と FROM の間には取り出す列の名前 (フィールド名) を並べます。複数の列を指定する場合は「,」で区切る必要があります。この例のように「*」を指定すると全ての列を指定したことになります。FROM の後には使用するテーブルの名前を並べます。この場合も複数のテーブルを指定する場合は「,」で区切る必要があります。

これを DB Browser で実行する際には、図 6.5 のように①の「SQL 実行」のタブをクリックして、②のところに SELECT 文を入力し、③のボタンをクリックします。この際 SQL のキーワードを小文字で入力しても問題はありません。

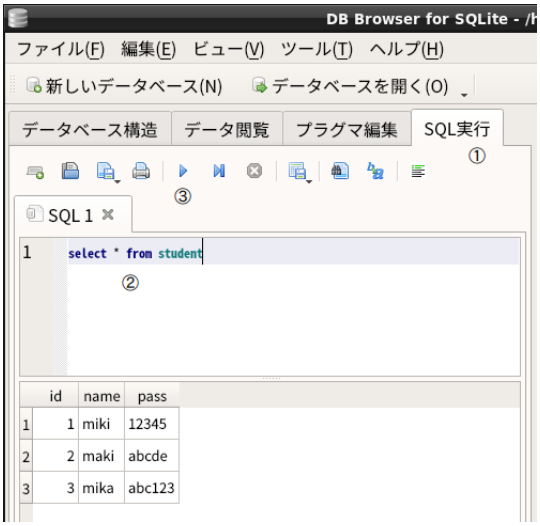


図 6.5 SELECT の実行

通常はテーブルの全ての列を取り出すような事はしません。pass の列だけ必要であれば次のようにします。

```
SELECT pass FROM student
```

さらに条件を設定することにより、取り出すレコードを限定することができます。

```
SELECT pass FROM student WHERE name='miki'
```

WHERE の後に取り出したいレコードの条件を指定します。この場合は「name='miki'」となっていますので、name の列に miki が入っているレコードのみが出てきます。JavaScript や PHP などと異なり**等しい場合は「=」**を使います。また**等しくない場合は「<>」**を使います。比較の対象が文字列の場合は「'」で囲います。条件によっては、複数のレコードが出てきたり、全く何も出てこないこともあります。

複数の比較は AND や OR でまとめます。AND は全ての比較が成立した場合、OR はどれか一つの比較が成立すれば良い場合に用います。例えば次のようにすると name が miki で pass が 12345 であるレコードの id のみが取り出せます。

```
SELECT id FROM student WHERE name='miki' AND pass='12345'
```

次のように不等号の記号を使用することもできます。不等号の条件の書き方は JavaScript や PHP などと同じです。

```
SELECT name FROM student WHERE id>=2 AND id<=3
```

これによって id の値が 2〜3 であるレコードの name が出てきますが、1 年生のテキストの Access で出てきた「Between And」を利用した次のような書き方も可能です。

```
SELECT name FROM student WHERE id BETWEEN 2 AND 3
```

SELECT 文でテーブルの内容を取り出す際に並び替えの指示を追加することができます。例えば name の昇順であれば次のようにします。

```
SELECT * FROM student ORDER BY name
```

もし WHERE があるならばその条件の後に「ORDER BY」の指示を付けます。降順にする場合は次のように DESC を追加します。

```
SELECT * FROM student ORDER BY name DESC
```

name に同じ値を持つレコードがあり、その場合は pass の値で並び替えたいような場合は次のように「,」で区切ってフィールド名を指定します。

```
SELECT * FROM student ORDER BY name DESC, pass
```

6.4 PHP による SQL の実行

PHP でデータベースに対して SQL の命令を送り、その結果を変数で受け取ることができます。データベースとしては SQLite は内蔵しているためにいつでも利用可能です。他には MySQL や PostgreSQL などの企業の基幹システムでも使われることのあるデータベースなどにも対応しています。対応方法も各々のデータベース専用の方法と共通の方法が用意されており、共通の方法を使って記述すると、後で容易に別のデータベースシステムに乗り換えることができるようになります。

データベースとの接続

データベースを利用する際には、まずデータベースとの接続を行う必要があります。MySQL や PostgreSQL などの利用者の管理もするデータベースシステムでは、この際に利用者の ID やパスワードが必要になりますが、SQLite の場合はデータベースのファイル名のみが必要です。リスト 1 はデータベースのファイル名が data.db の場合です。

リスト 1 データベースとの接続の部分

```
1  $dbf="data.db";
2  if (!file_exists($dbf)) { die("{$dbf}がありません。"); }
3  $db=new PDO("sqlite:$dbf");
4  if (!$db) { die('接続失敗です。'); }
5  $db->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_WARNING);
6  $db->setAttribute(PDO::ATTR_TIMEOUT,10);
7  if (!($db->beginTransaction())) { die("Transaction が開始できません"); }
```

1 行目でデータベースファイルの名前を変数に入れてます。2 行目でその名前のファイルが存在するかどうかを確認しています。ファイルが存在しない場合、die(”〜”) を実行して「〜」を表示して PHP の実行を停止します。3 行目でデータベースファイルを開き、開いたデータベースを変数$db に入れてます。ファイルの内容に誤りがあるような場合は、この変数の中身が NULL になるので 4 行目の if で調べています。「!」で条件を反転しているので、本来 false 扱いとなる NULL の場合に die() を実行します。

5 行目はエラーメッセージを出すレベルの設定を行っています。

6 行目で実行時間を 10 秒に制限しています。これ以降の SQL の実行に 10 秒以上かかるとエラーになります。

7 行目の「beginTransaction()」でトランザクション (transaction) の開始の指示をデータベースに送っています。これはこれ以降の処理が排他的な一連の処理であることを示しています。同じデータベースファイルをを他のプログラムが利用したり、同じプログラムが多重に起動されると、データベースの内容の整合性が取れなくなる可能性が生じます。これはそれを防ぐための処置で、これ以降他のプログラムは同じデータベースファイルには手が出せなくなります。ただ他のプログラムが既にデータベースファイルを排他的に使用中だった場合は、その終了まで待つことになります。永遠に待つことを避けるために、6 行目の実行時間制限をしています。

検索の実行

次のようにして「SELECT * FROM student」と言う SQL の SELECT 文を実行することができます。検索した結果は$result に入ります。SELECT 文に誤りがあった場合は、その内容を表示して停止します。

```
$sql="SELECT * FROM student";
$result=$db->query($sql);
if (!$result) { die($sql."の実行ができません。"); }
```

$result は二次元配列型の変数になります。つまり検索した結果は通常表のような二次元の形になるので、単純な配列では対応できないからです。foreach を利用すると検索結果を配列型の変数に 1 レコードずつ取り出すことができます。

```
foreach ($result as $data) {
   $data['id']、$data['name']、$data['pass'] にデータが入っている
}
```

検索結果を全て同じ形で扱う場合はこの方法が良いと思いますが、検索結果が一つしかない場合は次のようにします。

```
$data=$result->fetch(PDO::FETCH_ASSOC);
```

これで$data['id']、$data['name']、$data['pass'] にデータが入ります。ただし検索結果が存在しない場合もあるかもしれないのであれば、次のように if を併用した方が良いでしょう。

```
if ($data=$result->fetch(PDO::FETCH_ASSOC)) {
   $data['id']、$data['name']、$data['pass'] にデータが入っている
}
```

またこれを次のように while を併用すると foreach と同様に結果を順番に取り出すことができます。

```
while ($data=$result->fetch(PDO::FETCH_ASSOC)) {
   $data['id']、$data['name']、$data['pass'] にデータが入っている
}
```

データベース処理の反映・終了

データベースの検索はデータベースファイルを変更しませんが、後に出てくる UPDATE 文などはデータベースファイルの内容を変更します。しかし実際は UPDATE 文を実行してもまだデータベースファイル自体は変更されません。一連の処理が無事終わった場合は最後に、「commit()」でデータベースファイルを更新することができます。

```
$db->commit();
```

これを忘れると一見処理は正常に終了しているのに、データベースファイルの内容が変わらないと言うバグに頭を悩ませることになります。

もし他のプログラムが同じデータベースファイルを使おうと待っていた場合は、これで排他的使用が終わるので動きだします。一方処理中に問題が生じてデータベースファイルを更新する必要がなくなった場合は、次のようにして「rollback()」をします。データベースファイルはトランザクションを開始する前のままになります。

```
$db->rollback();
```

大抵の場合最後にこれらのどちらかを行いますが、データベースの処理の後に延々と別の処理を行う場合は、その前に COMMIT しましょう。

6.5 INSERT 文

INSERT 文を用いてデータベースのテーブルにレコードを追加します。例えば次のような形です。

```
INSERT INTO student (name, pass) VALUES ('mike','999abc')
```

INTO の次にテーブルを指定します。最初の ( ) の間に内容を入れるフィールド名を指定します。複数ある場合は「,」で区切ります。autoincrement の指定によって自動的に入るものや、入っていないことを示す

NULL が入ってよいフィールド名は省略して構いません。2 つ目の ( ) の間に入れる値を示します。数字や NULL 以外の値は「'」で囲います。順番は最初の ( ) のフィールド名の順番と同じにします。

PHP による INSERT 文の実行は次のようにします。

```
$sql="INSERT INTO student (name, pass) VALUES ('mike','999abc')";
if ($db->exec($sql)===false) { die("データの挿入失敗: ".$sql); }
```

`$db->exec($sql)` は$sql に入っている SQL 文を実行し、変更があったレコードの数または false を返します。本当に false だった場合は die() で実行を中止します。挿入する内容の指定の無い場合 id は、autoincrement 機能により、これまで使用した id の最大値 +1 が設定されます。この id の値が知りたいことがよくあります。そのような場合は lastInsertId() を次のように使用すると変数 ($id) に id の値が入ります。

```
$id=$db->lastInsertId();
```

これで student テーブルに、一番最後に実行した INSERT 文で挿入されたレコードの主キーに、autoincrement で入れた値がわかります。

6.6 DELETE 文

テーブルのレコードを DELETE 文を使用して削除することができます。削除したものを戻すことは SQL ではできませんし、条件を間違えると一気に多くのレコードが削除されるので注意します。心配な場合はデータベースのファイルのコピーを作っておくとよいでしょう。

例えば次のような形で DELETE 文を使用します。

```
DELETE FROM student WHERE id=3
```

FROM の次に削除するレコードのあるテーブルを指定します。WHERE 以下を省略すると全てのレコードが削除されます。WHERE の条件次第では複数のレコードを一度に削除することも可能です。PHP で次のように DELETE 文を実行した場合、変数`$result` には削除されたレコードの数が入ります。DELETE 文に問題があった場合はゼロではなく false が入るので if の条件では「===」を使用します。

```
$sql="DELETE FROM student WHERE id=3";
$result=$db->exec($sql);
if ($result===false) { die("データの削除失敗: ".$sql); }
```

削除した数が必要ない場合は次のように「`$db->exec($sql)`」を if の条件の中に入れても構いません。

```
$sql="DELETE FROM student WHERE id=3";
if ($db->exec($sql)===false) { die("データの削除失敗: ".$sql); }
```

6.7 UPDATE 文

既に入っているテーブルの中のデータを書き換えるのが UPDATE 文です。レコードの中の指定した項のみ書き換えることができますし、条件次第で複数のレコードの書き換えも一度に行えます。例えば次のような形で name が miki の pass を 5555 に書き換えることができます。

```
UPDATE student SET pass='5555' WHERE name='miki'
```

もし student テーブルの中に miki が複数あった場合は全て書き換えられてしまいますし、逆に miki がなければ何も書き換えられません。DELETE 文と同様に以下のように PHP で実行した場合は、$result 変数に書き換えられたレコードの数が入ります。

```
$sql="UPDATE student SET pass='5555' WHERE name='miki'";
$result=$db->exec($sql);
if ($result===false) { die("データの更新失敗: ".$sql); }
```

WHERE 以降を省略すると条件が無いので全てのレコードが書き換えの対象になります。また複数の項を書き換える場合は、次のように「,」で区切って指定します。

```
UPDATE student SET name='kiki', pass='5555' WHERE id=1
```

次のように既に入っているレコードの内容をもとに書き換えることもできます。

```
UPDATE student SET money=money+1000
```

これで全てのレコードの中の money の値を、1,000 増やすことができます。

6.8 ER 図

昔々、私は学生時代に初めて関係データベースの話を聴いた時に、二次元のテーブル (表) の形で様々なデータを扱うのは無理ではないか？と思いました。例えば学生の名前、身長、体重などのように 1 対 1 に対応しているものであればテーブルの形で済みます。これが 1 対多の関係になると難しくなります。例えば学生の家族を入れるとすると、最大で 3 人までと決まっていればテーブルの形になりますが、兄弟姉妹が複数いる学生も居るでしょう。とりあえず最大を 3 人にして、これを越える学生が出たら、最大を増やすと言うようなデータを入れる過程でどんどんテーブルの形が変わる、というのも色々困ります。

このような 1 対多の関係に対しては、学生と家族を 1 名で 1 レコード (行) と言うテーブルを用意して、家族が 5 人居る学生については、5 レコード使うというのが関係データベースのやり方です。このやり方であればどんなに多くの家族がいる学生が出現しても問題は生じません。また家族が居ない学生への対応も簡明です。行数が家族の数を示すので、家族がいない学生の行は存在しないだけです。

この方法で問題になるのは、学生の身長や体重のデータを、家族が 5 人いる学生については、5 回記述するのか？です。このような場合、関係データベースでは 1 対 1 の関係のものと、1 対多の関係のものは別のテーブルに入れて、体重を 5 回記述するような無駄を省きます。

学生と授業のように多対多の関係も考えられます。つまり一人の学生は複数の授業を取りますし、一つの授業には通常複数の学生が受講します。このような場合も複数のレコードで対応します。つまり一つのレコードには学生と授業を 1 つだけ記述して、もし 200 人の学生がそれぞれ 20 の授業を受ける場合は、200×20 個のレコードを作ります。

関係データベースの設計は、データベースに入れるデータの関係が、1 対 1 か 1 対多か多対多なのかを調べて、それを元にテーブルを分けます。そのためにデータの関係を表現するために図 6.6 のような ER 図 (Entity-Relationship Diagram) が使われます。

図 6.6 の四角い箱は実体 (entity) を示します。直線や矢印が実体の関係 (relationship) を示します。まず「体重」と「学生」は 1 対 1 の関係なので直線で結びます。「身長」と「学生」の関係も同様です。「学生」と「家族」は 1 対多の関係なので「学生」から「家族」への矢印になります。「学生」と「授業」の関係は多対多なので両方向の矢印になります。兄弟や姉妹の関係の学生がいると、同じ家族が生じるので、こちらも多対多ではないか？となりますが、とりあえずみんな違う家庭ということにしてください。

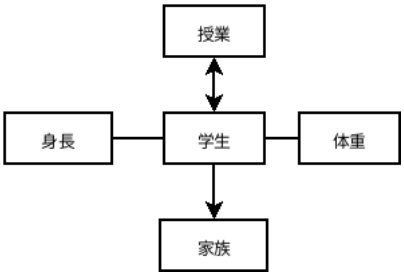


図 6.6 ER 図の例

学生テーブル

id	氏名	身長	体重	住所

家族テーブル

id	家族名	続柄	学生 id

授業テーブル

id	授業名	曜日	時限	担当

受講テーブル

id	授業 id	学生 id

図 6.7 ER 図に対応するテーブルの例

図 6.6 の実体と関係を関係データベースで扱うとなると、テーブルが 4 つになります。まず「学生テーブル」には「学生」と 1 対 1 の関係の「身長」や「体重」が入ります。他には住所などの個人情報も入るでしょう。「授業テーブル」には、授業名の他、何曜日の何限目にあるとか、担当教員などが入るでしょう。1 対多の関係の「学生」と「家族」の場合、「家族テーブル」には家族名や続柄などに加えて、「学生テーブル」とつなぐものが入ります。そして多対多の関係の「学生」と「授業」をつなぐための「受講テーブル」には、「学生テーブル」と「授業テーブル」をつなぐものが入ります。

「つなぐもの」は、ちゃんと対象となるテーブルの 1 行を指定できるものでないと困ります。関係データベースではテーブル同士をつなぐために、各テーブルにテーブルの 1 行を指定できる「主キー」と呼ばれる項目を設けることになっています。1 行だけを指定できるように、主キーの内容は各行全て異なる必要があります。ただ異なっていれば文字でも数値でも構いません。また「姓」だけでは区別できないので「名」の方と合わせて主キーとするようなことも可能です。各テーブルの主キーの名前を「id」とすると、各テーブルの項目は図 6.7 のようになります。

6.9 テーブルの結合

ここでは 2 つのテーブルからなるデータベースを題材に複数のテーブルの扱い方を説明します。この章の最初の例と同じ表 6.3 のような student テーブルに加えて、表 6.4 のような money テーブルを設けて、学生のお小遣いの管理をすることにします。一人の学生が複数回お小遣いをもらったり使ったりするために、student テーブルだけではお小遣いには対応できません。

表 6.3 student テーブルの例

id	name	pass
1	miki	12345
2	maki	abcde
3	mika	abc123

表 6.4 money テーブルの例

id	sid	date	amount
1	1	8/1	1000
2	1	8/5	-500
3	1	8/8	-350
4	2	8/15	800
5	2	8/17	-350
6	2	8/24	-400

student テーブルと money テーブルの id は主キーです。money テーブルの sid は sudent テーブルの id の値が入ります。最初の 3 行の sid は 1 なので、student テーブルの内容から、この 3 行は miki さんの記録ということになります。student テーブルの name の値である miki を使わず、student テーブルの id の値を使うのは、student テーブルの id は主キーなので複数のレコードが対応する恐れがないからです。amount の値が正の場合は、その分だけ持金が増えたことを意味し、負の場合はも持金が減ったことにします。最初の持金を

ゼロとすると、miki さんは +1000-500-350 で 150 残ったことになります。

student テーブルの name と money テーブルの date と amount を元に、だれがいつ、いくら使用したかの一覧を求める SQL は次のようになります。

```
SELECT name, date, amount FROM student, money WHERE student.id=money.sid
```

2 つのテーブルを使用するために FROM の後にテーブルの名前が 2 つ必要になります。そして WHERE のところに「student.id=money.sid」と指定することによって 2 つのテーブルが結合されます。この SELECT 文によって student テーブルの id と money テーブルの sid が同じレコードが結合して残ります。表 6.4 には sid の値が 3 のレコードがありません。そのため mika のレコードは出てきません。対応するレコードが無い場合も残したい場合は、外部結合という方法で対応することができますがここでは省略します。

student.id は student テーブルの主キーでしたが、money.sid は結合のために使われるものとして「外部キー」と呼ばれます。外部キーはテーブルの結合のために使われる、と言う条件だけなので主キーのように同じ値があってはいけないなどの制約はありません。

テーブルのどのフィールドが他のテーブルのどのフィールドに対応しているか、と言う条件を複数指定すれば 3 つ以上のテーブルを結合することもできます。ところが複数のテーブルに同じフィールド名が使われていると、曖昧な条件になってしまいます。それを避けるために「student.id」のようにテーブルの名前を付けます。先程の例では name、date、amount は片方のテーブルにしか出てこないためにテーブル名を省略していますが、両方のテーブルにある id を表示したい場合は、次のようにテーブル名も付ける必要があります。

```
SELECT student.id, money.id FROM student, money WHERE student.id=money.sid
```

これらを PHP で実行する際に一つ問題になるのは、SELECT 文の結果を取り出そうとする際に、student.id のようなテーブル名が付いたフィールドは取り出せない点です。そのために AS を使用して別名を付けます。次の例では money.id に mid と言う別名を付けて取り出しています。

```
$sql="SELECT money.id AS mid FROM student, money WHERE student.id=money.sid";
$result=$db->query($sql);
if (!$result) { die($sql."の実行ができません。"); }
foreach ($result as $data) {
   $data['mid'] に money.id の値が入っている
}
```

なおここで説明したテーブルの結合の書き方は古い書き方です。新しい書き方では INNER JOIN を使用して次のように書きます。テーブルの結合の条件が WHERE の検索条件と混ざらないため、わかりやすいと言われますがいかがでしょうか。3 つ以上のテーブルを結合する際には INNER JOIN 以降を繰り返すことになります。

```
SELECT name, date, amount FROM student INNER JOIN money ON student.id=money.sid
```

6.10 レコードのグループ化

テーブルのある項目について、同じ値が複数のレコードに入っている時に、同じ値ごとに集計したくなることがあります。例えば表 6.4 の場合、sid が同じ値のものについて amount の合計を出せば、各自が使用した金額が得られます。同じ値でまとめたい時は、次のように「GROUP BY」で sid を指定します。さらに SQL の SUM 関数を使ってグループごとの合計の計算をすることができます。なお WHERE による条件がある場合は、それらのあとに GROUP BY を付けます。

```
SELECT sid, SUM(amount) AS goukei FROM money GROUP BY sid
```

関数の結果を PHP で取り出すために AS を利用して goukei という別名を付けています。SUM 関数以外に表 6.5 のような関数があります。

表 6.5 SQL の関数

関数の形	関数の働き
AVG(フィールド名)	指定した列の値の平均値を求めます
COUNT(フィールド名)	指定した列の値が NULL 以外の行数を求めます
COUNT(*)	行数を求めます
MAX(フィールド名)	指定した列の値の最大値を求めます
MIN(フィールド名)	指定した列の値の最小値を求めます
SUM(フィールド名)	指定した列の値の合計を求めます

6.11 インデックスの設定

テーブルの列ごとにインデックス (索引) を設定することができます。検索対象として使われる列にインデックスを設定すると、検索の高速化ができます。これはインデックスが設定されてない場合、列に指定した値が入っているレコードを探すためには、テーブルの全てのレコードを順番に見る必要があるためです。テーブルとテーブルを結合する場合も、対応するレコードを全て探す必要があるため、インデックスの効果はレコード数が多い場合絶大的ものになります。ただテーブル内のデータの変更に合わせて、インデックスの内容も更新しなければならないので、データの変更が頻繁に大量に行われる場合は、システムへの負担が問題になるかもしれません。

インデックスの設定は SQL でも行うことができますが、ここでは DB Browser で設定する方法を紹介します。まず DB Browser でインデックスの設定をしたいデータベースファイルを開き、図 6.8 のようにまず①「データベース構造」のタブを選択し、②「インデックスの作成」をクリックします。すると図 6.9 のような設定のウィンドウが出てきますので、①適当な名前を入力し、②のところでインデックスを設定したいテーブルを選択し、③のところでインデックスを設定したいフィールド名をクリックして、④のボタンで右側にフィールド名を送ります。すると⑤のところの「OK」ボタンがクリックできるようになるので、クリックします。なお、各テーブルの主キーに対しては、最初からインデックスの設定がされているので、このような設定をする必要はありません。

6.12 SQL インジェクション

SQL が使えると、様々なシステムを構築する際にデータベースが利用できます。データベースを利用すると、必要な部分だけ必要に応じて取り出すことができるので、システムに修正が生じた場合に、その影響が他の部分に及びにくくなります。例えば money テーブルが追加されても、これまで他のテーブルを利用していた部分に影響はまずありません。一方様々なシステムが SQL を利用しているために、SQL を利用した攻撃が広く行われています。データベースには大量のデータが入っていることが多いために、攻撃が成功すると大きな問題につながります。例えば矢野経済研究所は 2022 年 6 月 24 日、同社の Web サイトが SQL インジェクション攻撃を受け、会員のメールアドレスと暗号化されたパスワード情報が、最大 10 万 1988 件流出した可能性があると発表しました[*22]。ここではこの SQL インジェクション (SQL injection) の仕組みと対策について

[*22] piyokango、「相次ぐ SQL インジェクション攻撃、矢野研は 10 万件超の個人情報が漏洩か」、日経 XTECH 2022 年 7 月 5 日 https://xtech.nikkei.com/atcl/nxt/column/18/00598/070100172/

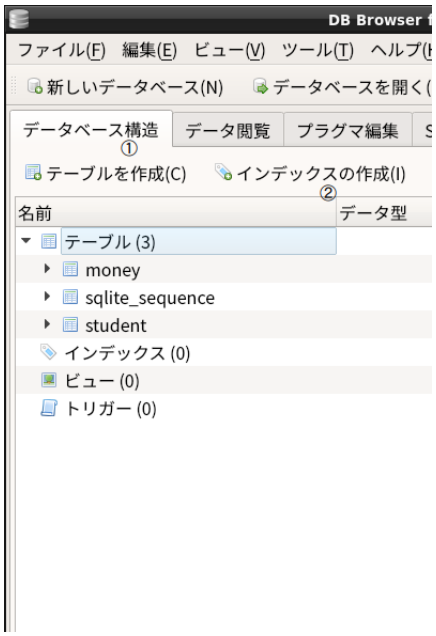


図 6.8 インデックスの設定の呼び出し

図 6.9 インデックスの設定画面

説明します。

SQL 文を区切る記号があります。PHP などと同じ「;」です。よってこれを利用して一度に複数の SQL 文を実行することができます。例えば、

```
SELECT id FROM student WHERE name='miki' AND pass='12345'; DELETE FROM money WHERE id<>''
```

を実行すると SELECT 文だけでなく、DELETE 文まで実行されます。この DELETE 文は id が空でない、money テーブルの恐らく全てのレコードを削除しますので大変危険なものです。ところが、PHP のプログラムに不備があると、このような SQL 文を攻撃側にしかけられてしまいます。

以下のような SQL 文を実際に実行する際は、「miki」や「12345」の部分は入力欄に入力された内容を使用します。

```
SELECT id FROM student WHERE name='miki' AND pass='12345'
```

もし攻撃者側が「12345」の代わりに「12345'; DELETE FROM money WHERE id<>'」と言う内容を入力すると、先程の例と同じ内容になるので DELETE 文も実行されてしまいます。同様にして INSERT 文を追加すれば勝手に利用者を追加することもできるでしょう。このように、攻撃側に都合の良い SQL 文を追加するので「SQL インジェクション攻撃」と呼ばれます。

対策は入力された内容を確認してから SQL 文に取り込むことです。大抵の入力では「;」や「'」などの記号は用いませんので、そのような記号があれば、記号を削除してから使うと言う対応が考えられます。パスワードには記号を含めることを推奨する場合もあります。そのような場合は記号が区切りなどとして働かないようにします。

入力された内容を書き換えるには、PHP の str_replace() などを利用すると、複数の書き換えを一気に行うことができます。また、次のように quote() を利用することも可能です。

```
$pass="12345'; DELETE FROM money WHERE id<>'";
$qpass=$db->quote($pass);
```

これで$qpass には「'12345''; DELETE FROM money WHERE id<>'''」が入ります。最初と最後に「'」が追加されるので、SQL 文を作る時に「'」で囲わなくて済む他、「'」が「''」に置き換わります。「''」は SQL では文字列を囲う記号ではなく、単独の「'」として扱われるため、文字列が「12345」で終わらなくなり、その後の「;」が文字列の中に取り込まれて、SQL 文の終わりとして働かなくなります。

例えばこれまで次のようにやっていたところは、

```
$sql="SELECT id FROM student WHERE name='{$_POST['namae']}' AND pass='{$_POST['pass']}'";
```

以下のようにします。

```
$namae=$db->quote($_POST['namae']);
$pass=$db->quote($_POST['pass']);
$sql="SELECT id FROM student WHERE name=$namae AND pass=$pass";
```

quote() を使うより、prepare() でプレースホルダの入った SQL 文を生成してやる方が良いと言われますが、ちょっと面倒なので省略します。

7.　演習課題

7.1　HTML の復習 4/10

図 7.1 のような Web ページを作成せよ。

図 7.1　作成例

- ファイル名は **k0410.htm** にする。
- 赤い星印の画像は「展開演習 A」のフォルダーにある star.png を mars に送って使う。
- 「椙山」のリンクをクリックすると別画面に「https://www.sugiyama-u.ac.jp/」が出るようにする。
- 音楽再生はテキスト p.14 の例をそのまま入れる。
- はめ込んだ web ページの URL は「椙山」のリンクと同じものである。大きさは 400 × 400 にする。

7.2　Style Sheet の復習 4/17

図 7.2 のような Web ページを作成せよ。

- ファイル名は **k0417.htm** にする。
- 表は Td のみに枠線を設定する。
- 赤い星印の画像は先回の star.png を使う。周囲に 1rem の隙間がある。
- 見出しの「4 月 17 日の課題」の上にマウスを持って行くと赤色になるようにする。
- 「重要」の大きさは 2rem にする。

図 7.2　作成例

- はめ込んだ web ページの大きさは縦横 20rem にする。
- ブラウザの幅を変更すると、見出しの「4 月 17 日の課題」、「重要」、はめ込んだ web ページの大きさが変わるようにする。

7.3　JavaScript **平均の計算・大小比較** 4/24

図 7.3 のような Web ページを作成せよ。

- ファイル名は **k0424.htm** にする。
- x と y の横の欄に数字を入力して「平均の計算」ボタンをクリックすると、「平均」の横の欄に平均の値が入るようにせよ。
- x と y の横に数字を入力して「大小比較」ボタンをクリックすると、警告画面で「x の方が大きい」、「y の方が大きい」、「x と y は等しい」のどれかが表示されるようにせよ。
- 「クリア」ボタンをクリックすると、3 つの入力欄が空になるようにせよ。

ヒント：ボタンが 3 つあるので関数を 3 つ定義します。関数の名前と入力欄の名前が同じにならないようにしてね。大小比較は結果が 3 つに分かれるので if が 2 個以上必要です。

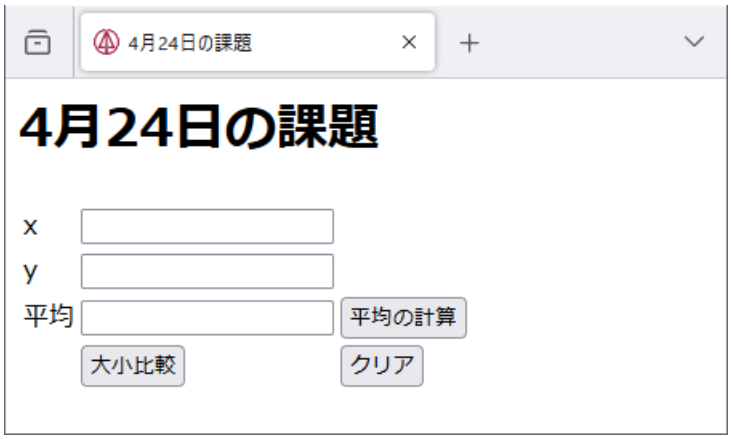


図 7.3 作成例

7.4 JavaScript 合計の計算 5/1

昨年度の「プログラミング基礎」の課題を参考にして、図 7.4 のような Web ページを作成せよ。

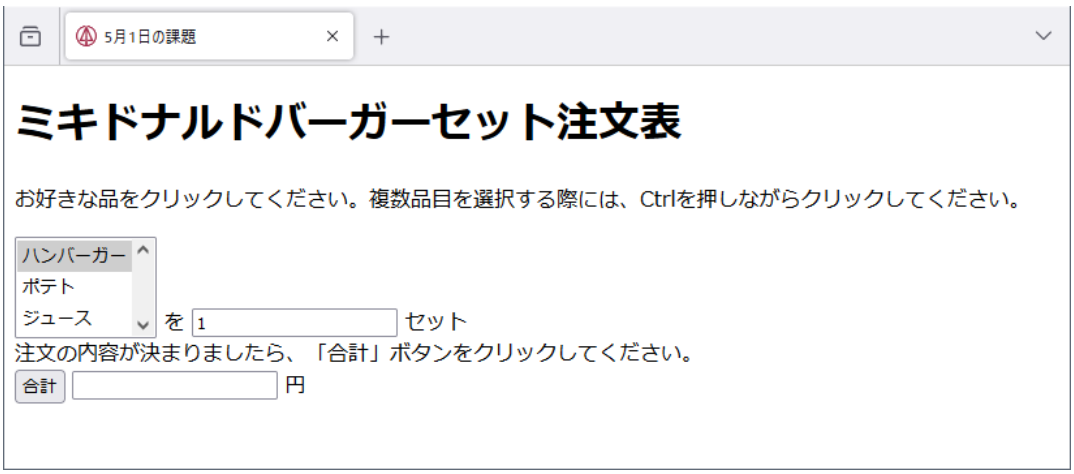


図 7.4 作成例 チーズバーガー追加前

- ファイル名は **k0501.htm** にする。
- if はひとつにする。
- 「合計」のボタンをクリックしなくても品の選択を変更した時も合計金額が出るようにせよ。
- これに 140 円のチーズバーガーを追加する場合、「`<Option value="140">チーズバーガー`」を入れるだけで、他は何も変更しなくても、図 7.5 のようになるようにせよ。

7.5 PHP ラジオボタン・チェックボックス 5/8

昨年度の「プログラミング基礎」の課題を参考にして、図 7.6 のような Web ページを作成せよ。

- 図 7.6 の左側のファイル名は **k0508a.htm** にする。
- 「診断」をクリックすると、**k0508b.php** が呼ばれて図 7.6 の右側のような診断結果が出るようにする。
- 「男」を選択すると「男の性格は分かりません。」だけが出る。
- 「その他」を選択すると「実は女心しか分かりません。」だけが出る。
- 「女」を選択し、砂糖以下で「砂糖」～「ブランデー」のどれも選択しなかった場合には、「すっきりし

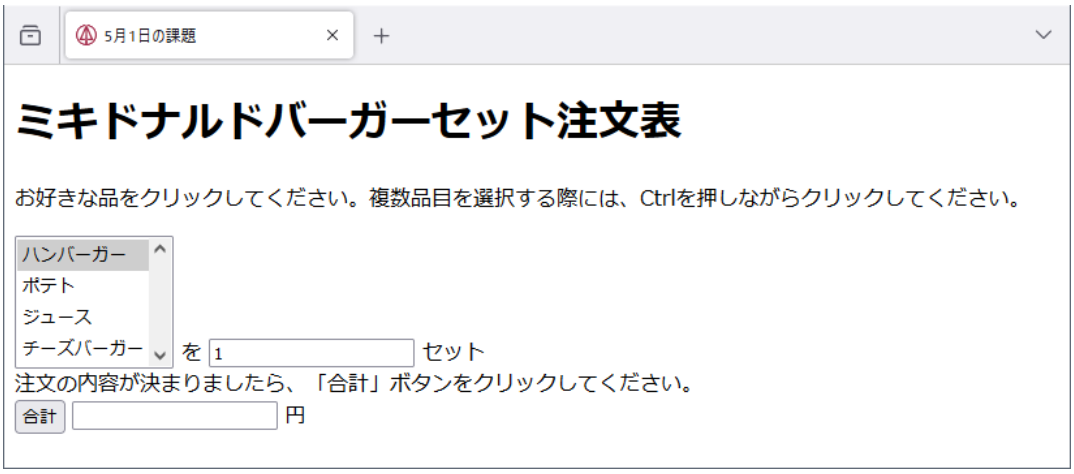


図 7.5　作成例 チーズバーガー追加後

図 7.6　作成例

た性格ですね」が出るようにする。

- 「女」を選択し、砂糖以下を一つだけ選択した場合は、一つだけメッセージを出す。例えば「砂糖」のみを選択すると「あま～～い性格ですね。」だけが出る。
- 「女」を選択し、砂糖以下を複数選択した場合は、各々のメッセージが出るようにする。ただし複数のメッセージを出す場合に重複しないようにする。例えば「砂糖」と「レモン」を選択した場合「あま～～い性格ですね。さっぱりとした性格ですね。」ではなく、「あま～～い・さっぱりとした性格ですね。」と出るようにする。

なお診断結果の文章は自分で適当に考えてよい。for を使わなくて if が 10 個ぐらい、結構長くなります。前回の課題の合計金額を変数を使って求めたように、例えば$msg を空にしておいて、チェックがあればこの変数にメッセージを追加して、最後に表示する方法がよいでしょう。

7.6　PHP TextArea・ファイル読み書き 5/15

図 7.7 のような「電子メモ」のページを作成せよ。

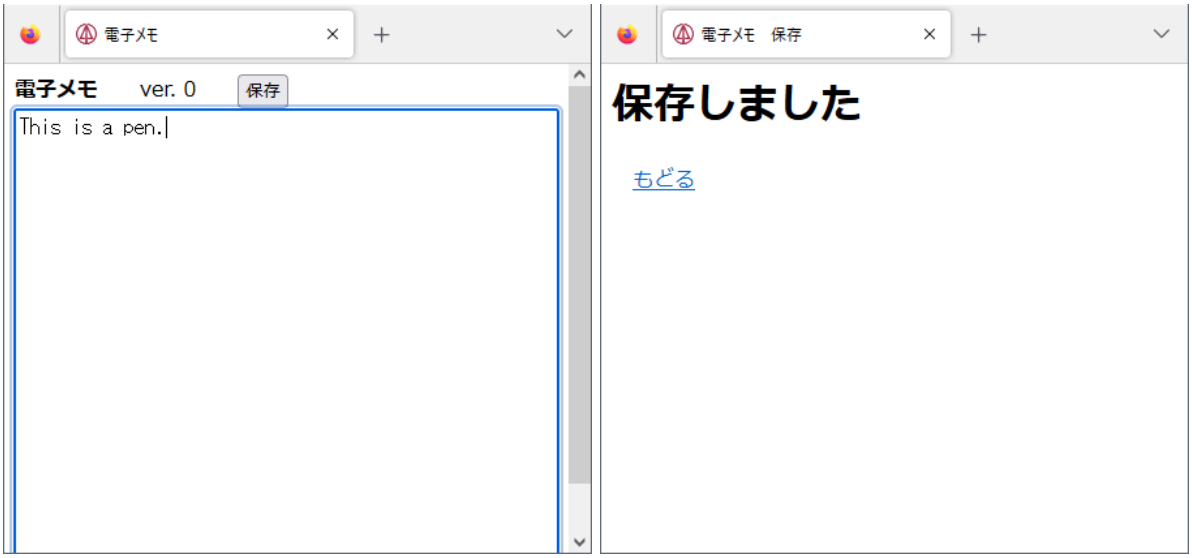


図 7.7　作成例

- 図 7.7 の左側のファイルの名前は **k0515a.php** にする。
- 「ver.」の次の数字は、0 からスタートして「保存」をする度に数字が増えるようにする。数値は **k0515c.txt** に保存しておく。なお一番最初は k0515c.txt がないのでエラーが出てしまわないようにする。同様のエラーは k0515b.php でも出ないようにする。
- 「保存」ボタンの下の大きな入力欄は Textarea である。ウィンドウ一杯に出すためには、cols や rows で大きさを指定するのではなく、style を使って指定する。横一杯は簡単だが、縦一杯は図 7.7 の左側のように少しはみ出す形で良い。縦方向きっちり指定は、テキストでやった範囲ではできないだろう。

  textarea の内容は **k0515d.txt** の内容が出るようにする。そのために file() と foreach() を使うこと。
- 「保存」ボタンをクリックすると図 7.7 の右側の **k0515b.php** が呼び出されるようにして、ここで Textarea の内容を k0515d.txt に保存し、k0515c.txt に入っている数値を一つ増やす。「もどる」をクリックすると k0515a.php に戻るようにする。

7.7　PHP ディレクトリ・ファイル削除 5/22

図 7.8 のようなファイルの一覧表示するものを作成せよ。

- 図 7.8 の左側のファイルの名前は **k0522a.php** にする。
- ファイルの一覧は「.」ディレクトリのものである。ファイルの順番は問わないが「.」や「..」は出ないようにする。
- **［削除］**のリンクをクリックすると **k0522b.php** へ行って該当するファイルを削除し、さらにリンクをクリックするなどの操作なしにまた k0522a.php へ戻るようにする。なお削除のテスト用には「〜.bak」と言う名前のバックアップファイルを使用すると良い。
- ファイル名をクリックすると **k0522c.php** が呼び出されれて、例えば「aaa.htm」をクリックした場合、図 7.8 の右側のように aaa.htm ファイルの内容を表示し、「もどる」をクリックすると k0522a.php

図 7.8 作成例

に戻るようにする。ブラウザで HTML のタグがそのまま表示されるようにするには、「<」を何とかしなければならない。

注意：漢字の名前のファイルを作成して、ちゃんと表示や削除ができるか確認すること。**ヒント**：k0522c.php では、`<UL>`、`<Hr>`、`<Pre>`、`&lt;` などを使います。

7.8 PHP Cookie 5/29

図 7.9 のような Name と Password を入れると、ファイルの一覧表示するものを成せよ。

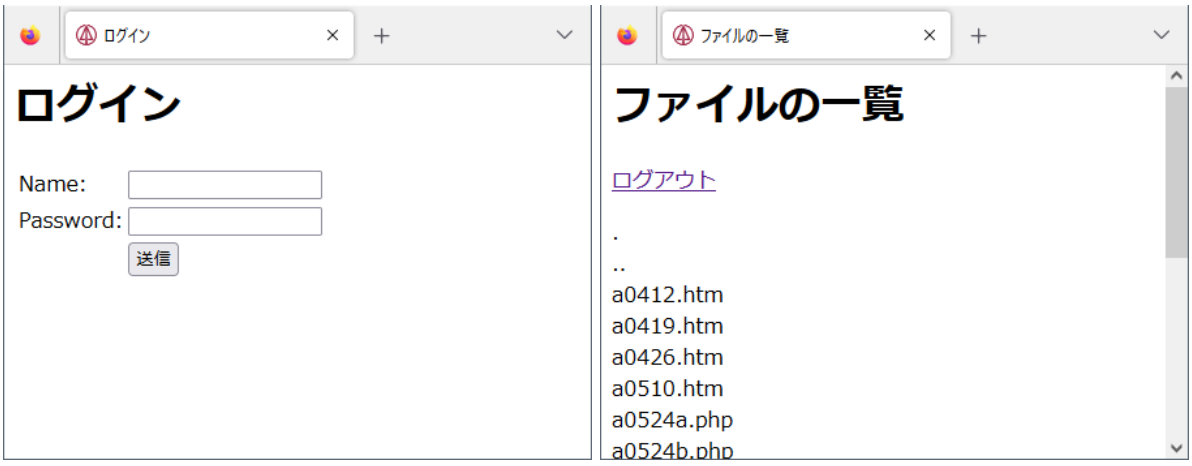


図 7.9 作成例

- 図 7.9 の左側のファイルの名前は **k0529a.htm** にする。
- パスワードを入れる入力欄は入力した文字が読めないようにする。
- 「送信」をクリックしたら **k0529b.php** に行くようにする。
- k0529b.php では、まず「ok」と言う名前のクッキー変数があるかどうか調べて、これがない場合は Name と Password の入力内容を調べて、「miki」と「maru」でなければ k0529a.htm に戻る。もし「miki」と「maru」であれば、「ok」と言う名前のクッキー変数を設定する。変数に入れる内容はなんで

もよい。有効期限は指定しない。

- 「ok」と言う名前のクッキー変数があったり、入力内容が「miki」と「maru」であれば、図 7.9 の右側のようにファイルの一覧を表示する。ディレクトリから取り出したファイル名を即座に表示するのではなく、まず$files[0]、$files[1]…のような配列変数に全て入れる。そして「sort($files);」で並び替えをしてから foreach を使って最初から順番に表示すれば良い。
- 一度ログインに成功すると、クッキー変数が残るので、直接 k0529b.php へ行ってもファイルの一覧が表示されるようになる。
- 「ログアウト」のリンクをクリックすると **k0529c.php** へ行くようにする。
- k0529c.php では、「ok」と言う名前のクッキー変数を削除し、k0529a.htm へ行くようにする。
- ログアウトするとクッキー変数が消えるので、直接 k0529b.php へ行ってもファイルの一覧が表示されないようになる。

注意：setcookie() と header() は<HTML>より前にないと働きません。

7.9 PHP Session 6/5

5 月 15 日や 5 月 29 日の課題を参考にして、今回は session 変数を用いて Name と Password で保護された電子メモを作成せよ。

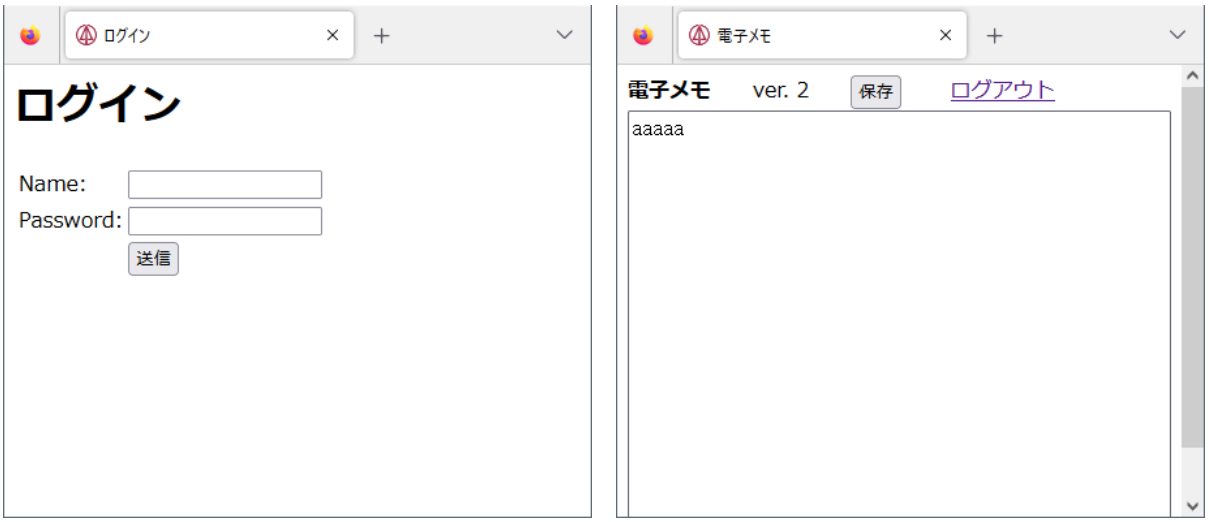


図 7.10 作成例

- 「k0605」と言う名前のディレクトリを作成し、以下のファイルは全てその中に入れること。
- **.htaccess** を作成し、内容はテキストの p.53 とほぼ同じ内容にする。変更するところは有効期限が 100 日は長いので 10 日にするのと、session ファイルを入れるディレクトリを「./ses/」ではなく「./」にする点です。
- 図 7.10 の左側のファイルの名前は **login.htm** にする。内容はほぼ k0529a.htm と同じです。「送信」をクリックしたら **memo.php** に行くようにします。
- memo.php では、まず「ok」と言う名前の session 変数があるかどうか調べて、これがない場合は Name と Password の入力内容を調べて、「miki」と「maru」でなければ login.htm に戻る。もし「miki」と「maru」であれば、「ok」と言う名前の session 変数を設定する。内容はなんでもよい。
- login.htm に戻らなければ、図 7.10 の右側のように電子メモの内容を表示する。これは「memo」と言う名前の session 変数に入っているものとする。「ver.」の次の数字は、0 からスタートして「保存」をす

る度に数字が増えるようにする。この数値も「ver」と言う名前の session 変数に入っているものとする。

- 「保存」ボタンをクリックすると **save.php** が呼び出されるようにする。save.php では Textarea の内容を「memo」と言う名前の session 変数に入れて、「ver」と言う名前の session 変数を一つ増やし、memo.php へ行くようにする。
- 「ログアウト」のリンクをクリックすると **logout.php** へ行くようにする。
- logout.php では、「ok」と言う名前の session 変数を削除し、login.htm へ行くようにする。

7.10 SQL SELECT 6/12

これまでは ID とパスワードの一人分にしか対応していなかったので、利用者が一人しか使えないシステムでした。データベースで ID とパスワードを管理する事により、利用者が複数にも対応できるシステムを作り上げていきます。

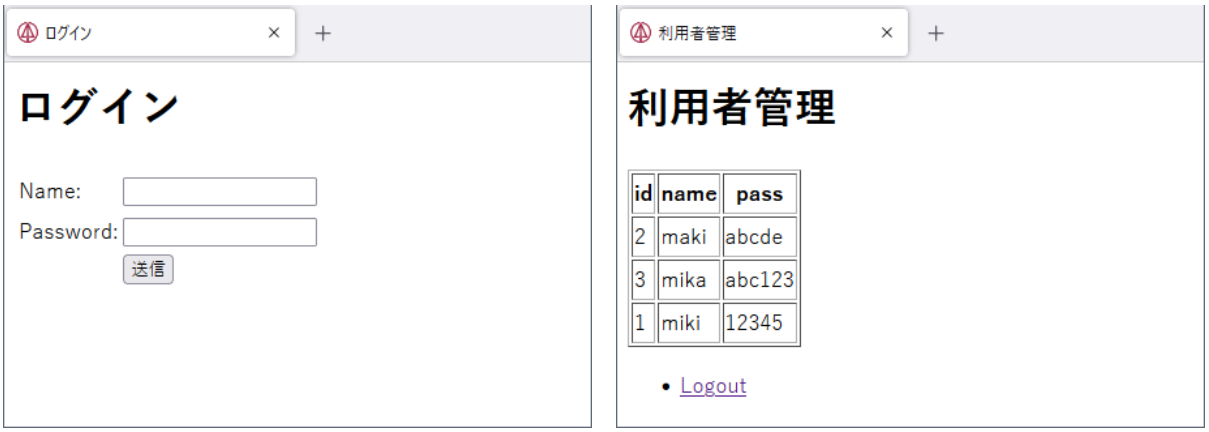


図 7.11 作成例

1. 「k0612」と言うディレクトリを作成して、以下のファイルは全てこの中に入れるようにします。
2. **data.db** と言うデータベースファイルの中に、「student」と言うテーブルを作成せよ。テーブルには「id」、「name」、「pass」と言う列を作成し、「id」を主キーにし自動的に数字が入るようにせよ。また「name」と「pass」については文字列が入れられるようにして、テキスト 55 ページの図 6.1 のテーブルの例と同じ内容を入力せよ。
3. 図 7.11 の左側の **login.htm** を作成せよ。中身はほとんど k0605/login.htm でよい。
4. 図 7.11 の右側の **main.php** を作成せよ。まず k0605/memo.php をコピーして、データベースの内容が表の形で出るようにする。表示の際には name で並び替えをする。最初の ID とパスワードのチェックの部分は変更しない。よって「miki」と「maru」で表が出ることになる。
5. 図 7.11 の右側の「Logout」で **logout.php** を呼び出すようにして、ここで session 変数を消去して login.htm に行くようにする。k0605/logout.php と同じ内容でよい。

7.11 送信前チェックとデータベースによるチェック 6/19

前回作成したシステムは、まだ一人分の Name と Password (miki と maru) にしか対応していなかったので、今回はデータベースに登録されている Name と Password の組み合わせに一致していれば入れるようにします。

1. 前回作成した「k0612」ディレクトリをデスクトップにコピーし、名前を「k0619」に変更した上で

図 7.12 作成例

「www」の中に移動し、この「k0619」の中に、以下のファイルは入れるようにします。

2. 前回作成した図 7.12 の左側の **login.htm** を、「送信」ボタンをクリックした際に Name や Password の欄が空欄であれば window.alert() でメッセージを出し pass.php に行かないようにします。どちらも空欄でなければ pass.php へ行きます。
3. 前回作成した main.php の最初の部分を参考にして **pass.php** を作成します。その内容は、
   (a) Name の入力欄の内容が入っている変数や Password の入力欄の内容が入っている変数が存在しない場合は、login.htm に戻します。
   (b) データベースの student テーブルにある「name」と「pass」と入力欄に入力された内容が一致しない場合は、login.htm に戻します。入力された内容をもとに id を検索して、結果の入っているはずの変数が存在しなければ一致しなかったことになります。
   (c) データベースの内容と一致した場合は、一致したレコードの id の値を「ok」と言う名前の session 変数に入れます。(前回の課題では何を入れても構わない事になっていました。) そして main.php へ行きます。
4. 図 7.12 の右側の **main.php** は最初の Name と Password のチェックの部分を削除して、「ok」と言う名前の session 変数が存在しない場合に login.htm へ戻るようにします。また、「ok」と言う名前の session 変数の内容をもとにデータベースを検索して、「利用者管理」の見出しの下に name を表示するようにします。

7.12 SQL INSERT, DELETE 6/26

今回は利用者の追加と削除ができるようにします。以下に出て来るファイルは全て「k0619」ディレクトリ内にあるものとします。

1. **main.php** を修正して図 7.13 の左のように「削除」のリンクと「利用者の追加」のリンクを追加する。「削除」のリンクの行先は delete.php とするが、delete.php がどの利用者を削除すれば良いのか分かるように、テキスト p.45 や 5 月 22 日の課題を参考にして id を送るようにする。また「利用者の追加」のリンクの行先は insert.htm にする。
2. 図 7.13 の右側のような、追加する利用者の情報を入力するための **insert.htm** を作成せよ。「追加」をクリックすると insert.php へ行くようにする。「追加せずに戻る」のリンクの行先は main.php にする。
3. 追加の実行を行う **insert.php** では、「ok」と言う名前の session 変数が存在する事を確認した上で、

図 7.13 作成例

データベースに入力した内容を追加し main.php へ行くようにする。もし session 変数が存在しなかった場合は、login.htm へ行くようにする。

4. 削除の実行を行う **delete.php** では、「ok」と言う名前の session 変数が存在する事を確認した上で、データベースから指定された利用者を削除し main.php へ行くようにする。もし session 変数が存在しなかった場合は、login.htm へ行くようにする。元のデータを残すため、削除のテストは追加したデータに対して行うようにする。

7.13 SQL UPDATE 7/3

今回は利用者情報の修正ができるようにします。今回作成するファイルも k0619 に入れてください。

図 7.14 作成例

1. main.php を修正して図 7.14 の左側のように「利用者情報の修正」のリンクを追加する。このリンクの行先は update.php にする。

2. 図 7.14 の右側のような、利用者情報の修正内容を入力するための update.php を作成せよ。「ok」と言う名前の session 変数が存在する事を確認した上で、入力欄にはログインした利用者の現在の内容が入るようにする。修正を行った上で「修正」をクリックすると update2.php へ行くようにする。「修正せずに戻る」のリンクの行先は main.php にする。もし session 変数が存在しなかった場合は、login.htm へ行くようにする。
3. 修正の実行を行う update2.php では、「ok」と言う名前の session 変数が存在する事を確認した上で、データベースの内容を修正し main.php へ行くようにする。もし session 変数が存在しなかった場合は、login.htm へ行くようにする。

7.14　SQL テーブルの結合 7/10

今回は利用者のお金の利用状況が出せるようにします。今日作成するファイルは k0619 へ入れてください。

図 7.15　作成例

1. main.php を修正して図 7.15 の左側のように「小遣いの一覧」のリンクを追加する。このリンクの行先は money.php にする。
2. data.db にテキスト p.64 の表 6.4 と同じ内容の money テーブルを追加する。なお、id、sid、amount は INTEGER 型、date は TEXT 型である。
3. 図 7.15 の右側のような、小遣いの状況を表示する money.php を作成せよ。「ok」と言う名前の session 変数が存在する事を確認した上で、残額の一覧と小遣いの詳細を表示する。小遣いの詳細の方は金額がマイナスのものは図のように黄色く表示する。もし session 変数が存在しなかった場合は、login.htm へ行くようにする。「もどる」をクリックしたら main.php へ行くようにする。

＊ money.php の 1 つ目の表は、名前順、残額は右詰めに、2 つ目の表の金額も右詰めになっています。

索引

R

S

T

U

V

W

あ

か

さ

た

な

は

ま

や

ら

わ