

システムの画面モック

CHAPTER

14

ここまでHTMLとCSSを一通り学習してきました。ここでは、今まで学んでいたことを使って、システム画面を作成してみます。作成するのはあくまで画面のみですが、Webシステムの画面がどのようなものになるのか、その要点を学びましょう。

14-1 システム作成は画面から

本書ではChapter 1でWebの仕組み、Chapter 2で環境構築、Chapter 3～9でHTMLを、Chapter 10～13でCSSを学んできました。Webデザイナーは、これらの知識を基礎としてデザインスキルを駆使してWebサイトを作成していきます。一方、Webプログラマはこれらの知識をどのように活かしていくのでしょうか。その話から始めていきましょう。



ポイントはこれ！

- ✓ システムの利用者がシステムと接する部分をUIといい、大半のシステムは画面が該当する
- ✓ システムの利用者からすると、UIがシステムのすべて
- ✓ UIの良し悪しがシステムの良し悪し
- ✓ Webは画面部分をHTML+CSSで作るので、プログラムから切り離して作成可能
- ✓ プログラミングされていないHTMLだけのシステム画面をHTML画面モックという
- ✓ プログラミングに先立って、システムの打ち合わせ、設計段階でHTML画面モックを作ってしまう
- ✓ HTML画面モックを作成するのはwebプログラマの役割

【14-1-1 システム利用者からすると画面がすべて】

プログラマの役割は、プログラミングを行ってシステムを作成することです。そのシステムと、システムの利用者との関係というのを少し見てみます。

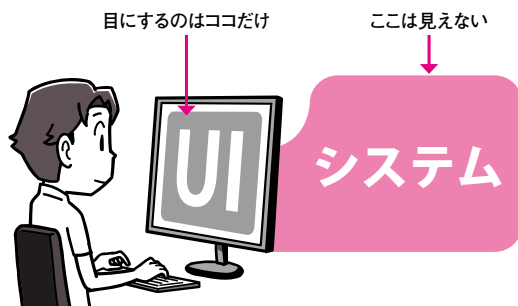
▶▶▶ UI

システムの利用者がシステムと接する部分は、**UI** (User Interface) と呼ばれます。例えば、静脈認証システムでは、センサーがシステムとその利用者と接する部分であり、UIといえます。あるいは、最近流行りのスマートスピーカーの場合は、音声そのものがUIとなっています。このようなUIがある一方で、大半のシステムのUIというのは画面です。パソコンのモニターやスマホなどの画面を通じて、利用者はシステムと接します。

▶▶▶ 利用者からするとプログラムは見えない

プログラマの役割というのはプログラミングをすることです。そのプログラマが作成するプログラムというのは、システムの中心的存在といえます。ところが、そのプログラム部分というのはシステムの利用者からは基本的に見えない部分です。システムの利用者が目にする部分というのは、画面、つまりUIのみです (図 14-1)。

図 14-1 : 利用者からするとUIがシステムのすべて



つまり、利用者目線では、UIがシステムのすべてなのです。ここにプログラマとシステム利用者との意識の乖離が生まれます。プログラマはUIの後ろに隠れたプログラムを作っているだけに、そこが中心のように思えてきます。もちろん、バグがないなどの良いプログラムでないと、システムはちゃんと動きません。そういった意味で、システム内のプログラムが良いプログラムであることは必要条件で

す。ただし、十分ではありません。そこに、使い勝手のいい、システム利用者が満足するUIがそろって、初めて良いシステムといえます。

▶▶▶ WebシステムのUIはHTML+CSS

本書中に何度か登場したように、このように重要なUIを、Webシステムの場合はHTML+CSSで作ります。つまり、利用者にとって良いシステムを作ろうとするならば、そのWebシステムを作る側、つまりWebプログラマにはHTML+CSSの知識は必須なのです。さらに、より良いUIを実現しようとする、当然Webデザイナーとの協業というのは必須といえます。

C O L U M N

JavaScriptは避けられない

最近のWebの世界は、HTML+CSSだけでは使い勝手が悪く、ブラウザ上で動きのあるUIを実現する必要があります。これには、JavaScriptによるプログラミングが必須です。このことは、Webシステムでも同様で、最近のWebシステムではHTML+CSS+JavaScriptの組み合わせでUIを作るのが普通です。本書はこのうちのHTML+CSSのみを取り扱っています。残されたJavaScriptについては、本書を終えられた後、拙著『これから学ぶ JavaScript』（インプレス）で学んでいただければ幸いです。



言われないと気づきませんが、確かに、システムの良し悪しってUIだけで判断していますね。特に、僕みたいにまだプログラミングを知らない人からするとUIの後ろ側なんて、全くわかりません。

でしょうね。キョウタクくんみたいにプログラマを志しているなら、UIの後ろにシステムがある、ぐらいいはわかるかもしれないけど、システム開発と全く無縁の利用者さんは後ろがあることすら気づかないと思うわ。





僕は利用者さんは、それでいいと思うよ。逆に、作る側は利用者がそういう存在であることをちゃんと意識しておく必要があると思う。そのために、画面を作る工程をもっと大切にしなければならないんだ。

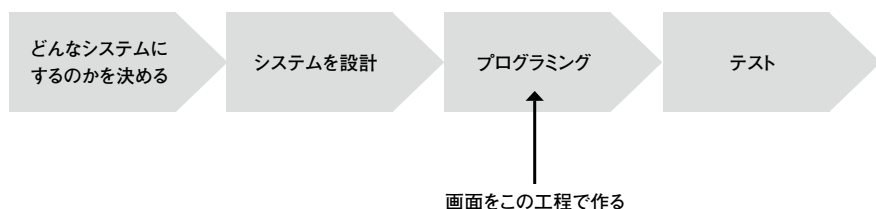
[14-1-2 プログラミングより前に画面を作ってしまう]

会話の中でワトソン先生が言っているように、システムを作成する全行程の中で、画面を作る工程をどのように位置づけるかで、UIの良し悪しが変わり、ひいてはシステム全体の良し悪しへとつながっていきます。

▶▶▶ Webシステムは画面のみを作成できる

システム開発というのは、単にプログラミングをするだけではありません。その前段階として、どのようなシステムにするかを取り決めたり、設計を行ったりします。また、プログラミング後もテストを行ったりします。では、画面作成はいつ行うのかというと、このプログラミングと同時に行うことが多々あります（図14-2）。

図 14-2：プログラミングと同時に画面を作る



システムを作成する言語によっては、そのような方法を取らざるを得ないこともあります。

一方、Webシステムは、UIをHTML+CSS (+JavaScript) で作成するため、

JavaやPHPなどのプログラミングを行うことなく画面を作成することができます。しかも、実際のシステムをプログラミングするよりはるかに少ない労力で画面を作成できます。

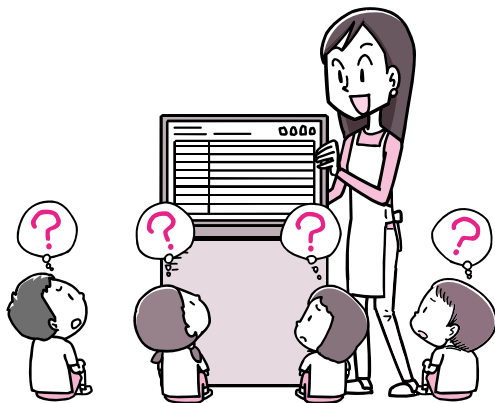
▶▶▶ 紙芝居では想像できない

ところで、通常、システム開発では、どのようなシステムにするのかといった内容や設計情報というのは、ドキュメント（書面）として残します。もし、システムの納入先が自社ではない場合などは、それらドキュメントをもとに発注者と打ち合わせを行ったりします。自社内の場合でも、それらのドキュメントをもとに社内で打ち合わせを行います。これらの打ち合わせは、当然プログラミングに先立って行い、ほとんどの内容、仕様が確定していなければ、プログラミングができません。

もしプログラミングと同時に画面を作成するとしたら、システム利用者にとって一番大切な画面の打ち合わせというのはどのようにするのでしょうか。それは、やはりドキュメントとして記述したものをもとに打ち合わせを行い、画面仕様を取り決めます。とはいえ、あくまで紙に書かれたものですので、例えていうならば紙芝居のような状態です。画面を紙芝居のように見せられながら、システムの動きを想像するしかないのです。

もちろん、実際にプログラミングする人間は紙芝居でも問題ありません。しかし、一般の利用者はそういう想像がなかなかしにくいのが現状です。

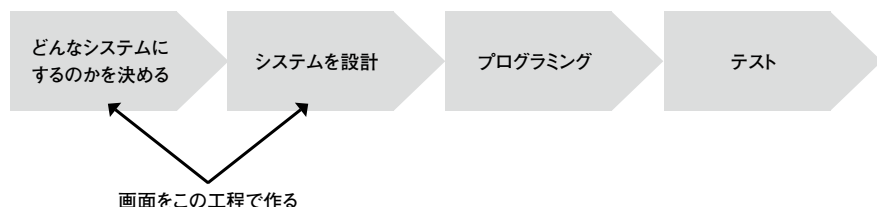
図 14-3：紙芝居ではシステムの動きが想像できない



▶▶▶ HTML 画面をドキュメントの代わりに作ってしまう

そこで、実際のプログラミングに先行して画面を作ることができるWebシステムの大きな利点を活かします。システムを設計する段階、あるいは、どのようなシステムにするかを打ち合わせしている段階で、このHTMLによる画面を作ってしまう。こうすることには、非常にメリットがあります（図14-4）。

図 14-4：画面をプログラミングより前に作ってしまう



というのは、紙に書いたドキュメントとは違い、HTMLで作成された画面は、リンクを正しく記述しておくだけで、ちゃんと画面遷移をします。つまり、システムを組み込まずにHTMLのリンクだけでシステムの画面遷移を模倣できるのです。画面上の表示データは変更されませんが、実際に画面が切り替わるため、システムの使い方がイメージしやすいです。このようにして作られた画面を**画面モック**、あるいは、HTMLで作られたことをより正確に表現するために**HTML画面モック**などといいます*。この画面モックを、システム利用者、あるいは、システム発注者などに実際に触ってもらい、それをもとに打ち合わせを行っていくことで、より有用なフィードバックを得ることができます。

さらに、HTMLはプログラムに比べて修正が簡単ですので、そのフィードバックをすぐに画面に反映させられます。それをもとにさらに打ち合わせを行っていくというサイクルを繰り返すことで、より良いUIのシステムを作ることができます。

*）モックというのは、モックアップ（mock-up）の略で、本来は実物大模型という意味です。



HTMLで実際に画面を作るというのは、かなりメリットがあるということですけど、プログラミングとHTMLの作成とはそんなに違うんですか？

全然違うよ。例えば、データの入力画面があって、登録ボタンをポチッと押して完了画面表示、みたいなのを考えても、その入力画面と完了画面の2枚のHTMLの間に、プログラムが挟まることになるけど、そのコードは画面の何倍もの量になるのよ。



そんなに違うんですね。

そうよ。で、プログラミングと同時にHTMLを作っていて、プログラム部分も出来上がってテスト段階で、例えば入力項目がひとつ足りない、なんて言われたら、もう、どれだけのコードを書き直さなければならないか。こういうのを手戻りって言うんだけどね。



それがHTMLだけならinputタグをひとつ書くだけでいいと。

そうそう。そうやって、画面の仕様が決まってからプログラミングすると、手戻りが少なくて済むのよ。



そして重要なのは、打ち合わせや設計段階でこのシステム画面を作るのは、システムのことがよくわかっているプログラマだけなんだ。だから、プログラマがHTMLやCSSを知っておく必要があるのには、こういうわけがあるんだよ。プログラマが仕様として確定したHTML画面を、今度はWebデザイナーにお願いしてより使い勝手のいいものにデザインしてもらうんだ。

だから、Webプログラマは、まずHTMLとCSSを学んでおく必要があるんですね！



そういうこと。

142: 会員管理システムのHTML画面モックを作る

概要はここまでにして、実際にHTML画面モックを作っていきます。といっても、14-1-1項のコラムにもあるように、最近のWebシステム画面はJavaScript抜きでは作れません。ですので、本来ならば、JavaScriptを含めた状態でモックを作成する必要があります。しかし、本書ではJavaScriptは未習であることを前提としますので、あくまでHTML+CSSで作成したものを紹介していきます。



ポイントはこれ！

- ✓ ダッシュボードは本来計器盤の意味で、そこから転じてログイン直後に表示される画面のことをいう
- ✓ 情報の一覧表示はテーブルが便利
- ✓ 縦並びの詳細表示用の表ではtableタグではなくdlタグを使おう
- ✓ データの入力欄はデータ項目ごとに用意する
- ✓ Webシステム内、Webサイト内で迷子にならないようにパンくずリストを表示しよう

[14-2-1 ダッシュボード画面の用意]

これから作成するHTML画面モックの題材として取り上げるのは、会員管理システムです。ここでは、まず実際の会員管理の画面作成に入る前段階までの画面を用意しましょう。それは、ダッシュボード画面です。

▶▶▶ ダッシュボード画面とは

実は、このダッシュボード画面の大枠はChapter 13で作成しています。Chapter 13で作成したナビゲーションの画面がそれにあたります。図13-1にもあ

るように、ナビゲーションの最初が「ダッシュボード」、その次が「会員管理」となっています。しかも、そのナビゲーションのあるヘッダボックスの次のボックスには「ダッシュボード」という表示があります。つまり、Chapter 13で作成してきた画面はこのダッシュボードを想定しています。

ではダッシュボード画面とはどのような画面をいうのでしょうか。**ダッシュボード**というのは本来は、計器盤のことです。まさに、自動車のダッシュボードが本来の意味です。ここから転じて、そのシステム内でのデータ解析の概要を一度に表示させる画面のこととして使われるようになりました。さらに、現在ではより広い意味として、そのシステムにログインした直後に表示される画面のこととして使われています。これは、ログイン直後に、そのシステムをこれから利用していく上で必要な情報、最新の情報、まず確認しておかなければならない情報を一挙に表示させる画面となっていることが多いからです。



Chapter 13でナビをいろいろ作成してきましたけど、headerタグの次のmainタグに書かれていた「ダッシュボード」って何だろうと思っていたのですが、こういう意味だったんですね。

そう。あの画面はログイン直後に表示される画面を想定していたんだよ。本来のダッシュボードは、それこそ計器盤のようにそこにさまざまな情報が表示されていないといけないけど、ここではあくまでサンプルなので、そうしたのはすべてカットして「ダッシュボード」の表記のみで済ませたんだ。で、今から作成する会員管理画面は、そのダッシュボードのナビの「会員管理」をクリックするところから始まるんだ。



じゃあ、Chapter 13のサンプルのどれかをコピーすればいいんですね。

そうだね。ここでは、13-2-2項で作成したflexNavi.htmlを使うことにしよう。メディアクエリなしにナビが柔軟に変わるからね。ただ、少し変更することにするね。



それでは早速ファイルを作成しましょう。Chapterが新しくなりましたので、websamplesフォルダ内にこのChapter用のフォルダとして「chap14」を作成してください。その中に、chap13内のflexNavi.htmlをdashboard.htmlとしてコピーし、リスト14-1の太字のように変更してください*。

*) これ以降のサンプルについては、IE11がmainタグを部分的にしかサポートしていないため、意図したように表示されない場合があります。IE11以外のブラウザで確認するようにしてください。

リスト14-1 : dashboard.html

```
<!DOCTYPE html>
<html lang="ja">
  <head>
    <meta charset="utf-8">
    <title>ダッシュボード</title>
    <meta name="viewport" content="width=device-width,⇒
initial-scale=1">
    <link href="main.css" rel="stylesheet" type="text/⇒
css">
  </head>
  <body>
    <header>
      <h1>これから学ぶシステム画面</h1>
      <nav>
        <ul>
          <li><a href="/websamples/chap14/⇒
dashboard.html">ダッシュボード</a></li>
          <li><a href="/websamples/chap14/⇒
memberList.html">会員管理</a></li>
          <li><a href="#">受注管理</a></li>
          <li><a href="#">商品管理</a></li>
          <li><a href="#">マスター管理</a></li>
          <li><a href="#">ログイン情報</a></li>
        </ul>
      </nav>
    </header>
    <main>
```

```

        <h2>ダッシュボード</h2>
    </main>
</body>
</html>

```

違いは、titleタグの内容、読み込み先cssファイル、headerタグ内のh1タグ、liタグ内のリンク先、mainタグ内です。headerタグ内のh1タグは増えていきます。また、mainタグ内のh2タグはもともとmainタグ内のsectionタグ内に書かれていましたが、mainタグ直下に移動しました。

次に、cssファイルを作成します。これは、chap13フォルダ内のchap13FlexNavi.cssをコピーしてmain.cssとし、リスト14-2の太字の部分を変更してください。なお、header nav ulセクタ、header nav liセクタ、header nav li aセクタには変更部分はありませんが、念のため記載しておきます。それ以外の太字の部分は、新しいことは何もありませんので、コメントを頼りに変更、追記してください。

なお、このmain.cssはこのChapter中の全htmlファイルから共通で読み込むようにします。

リスト14-2：main.css

```

/*headerボックスのスタイル設定*/
header {
    /*背景色を設定*/
    background-color: aliceblue;
}
/*header内のh1ボックスのスタイル設定*/
header h1 {
    /*内余白の上を5px、左右を15px、下を0pxに設定*/
    padding: 5px 15px 0px;
    /*外余白を0pxに設定*/
    margin: 0px;
    /*文字色を設定*/
    color: darkgrey;
}

```

```

/*navボックスのスタイル設定*/
header nav {
    /*内余白の上を5px、左右を15px、下を15pxに設定*/
    padding: 5px 15px 15px;
    /*外余白を0pxに設定*/
    margin: 0px;
}
header nav ul {
    margin: 0px auto;
    padding: 0px;
    box-sizing: border-box;
    width: 100%;
    display: flex;
    flex-direction: row;
    flex-wrap: wrap;
}
header nav li {
    list-style: none;
    background-color: turquoise;
    text-align: center;
    padding: 10px;
    box-sizing: border-box;
    margin: 2px;
    flex: 1 0 140px;
}
header nav li a {
    color: white;
    text-decoration-line: none;
}
/*mainボックスのスタイル設定*/
main {
    /*背景色を設定*/
    background-color: gainsboro;
    /*内余白を15pxに設定*/
    padding: 15px;
}
/*main内のh2ボックスのスタイル設定*/
main h2 {

```

```
/*外余白を0pxに設定*/  
margin: 0px;  
/*背景色を設定*/  
background: white;  
/*内余白を5pxに設定*/  
padding: 5px;  
/*フォントサイズを16ポイントに設定*/  
font-size: 16pt;  
/*文字色を設定*/  
color: darkgray;  
/*アンダーラインのような装飾としてボックス下境界線を設定*/  
border-bottom: solid 2px darkgray;  
}
```

作成が終了したら、dashboard.htmlを表示させてください。図 14-5 のように表示されます。

図 14-5：表示させたダッシュボード画面



あ!なんか、ちょっとかっこよくなった。Chapter 13 の図 13-1 との違いは、今太字で入力した CSS コードのおかげなんですね。

そうね。それにしても、先生、「これから学ぶシステム画面」って、なんてベタなシステム名なんですか! (笑)





(笑) まあ、そこはサンプルなんで許してよ。実際は、ここにちゃんとしたシステム名や、場合によってはロゴが入ることになるね。システム名はちゃんと打ち合わせの上決めることだし、ロゴはデザイナーさんにデザインしてもらわないといけない。そこは協業だね。もうひとつ、許してほしいのは、ここで適用しているスタイルは、あくまで最低限だということ。この程度のCSSをプログラマの方で作っておいて、さらに、Webデザイナーさんにもっと使い勝手がいいようにデザインしてもらうんだ。それこそ、ロゴを入れてもらったりね。

なるほど。そこも協業ですね。



さて、いよいよ会員管理に入っていくよ。ナビの「会員管理」にリンクを付けたから、そこをクリックしたら表示されるファイルを作っていくよ。

[14-2-2 会員管理のトップは会員リスト]

いよいよ会員管理の画面作成です。

▶▶ たたき台を作成

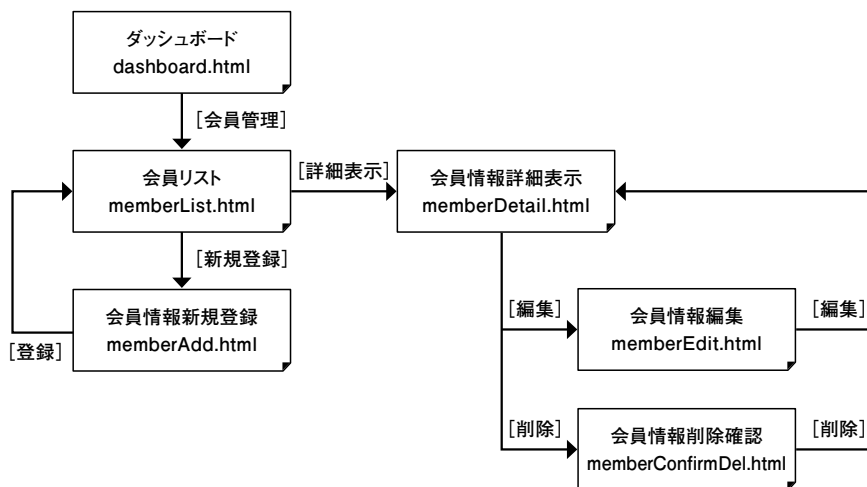
14-1-2項で説明したように、作成したHTML画面モックというのは、それをもとに打ち合わせを行ったり、そのフィードバックを盛り込んでHTMLを修正しながらひとつの設計ドキュメントの代わりとするものです。

そこで、ここではまず、その最初の段階の画面を作成します。今から作成する画面は完成形ではなく、これをもとにシステム発注者と打ち合わせを行っていくためのたたき台と位置づけ、作成していきます。

▶▶▶ 会員管理のトップ画面はリスト表示

ナビゲーションの「会員管理」をクリックして表示される画面として、ここでは会員情報が一覧表示される会員リスト画面を想定します。それ以降も含めて会員管理部分の画面遷移を図にすると図14-6となります。

図 14-6：会員管理部分の画面遷移



それぞれの矢印に付記している「[詳細表示]」などは、ボタンやリンクを表します。それをクリックすることで表示される画面名、およびファイル名を記載しています。

なお、このChapterで図14-6に記載のすべての画面を作成するには紙面が足りません。そこで、実際に作成するのは「会員リスト」、「会員情報詳細表示」、「会員情報新規登録」の3画面のみとします。

さて、その会員リスト画面を実際に作成しましょう。実は、リスト14-1でナビゲーションのリンク先として、あらかじめ図14-6の遷移図をもとにmemberList.htmlとリスト表示用のファイル名を記述しています。まず、このmemberList.htmlを、会員情報が表示される一步手前の状態まで作成します。ベース部分はdashboard.htmlですので、dashboard.htmlをコピーしてmemberList.htmlとして、リスト14-3の太字の部分を変更、追記してください。

リスト14-3: memberList.html

```
<!DOCTYPE html>
<html lang="ja">
  <head>
    <meta charset="utf-8">
    <title>会員リスト&nbsp;|&nbsp;会員管理</title>
    ~省略~
  </head>
  <body>
    <header>
      ~省略~
    </header>
    <main>
      <h2>会員管理</h2>
      <section>
        <h3>会員リスト</h3>
      </section>
    </main>
  </body>
</html>
```

違いは、titleタグの内容とmainタグ内です。そのmainタグ内に新たにsectionタグと、その中にh3タグが増えていますので、これに関するセレクトを追加します。main.cssにリスト14-4の内容を追記してください。

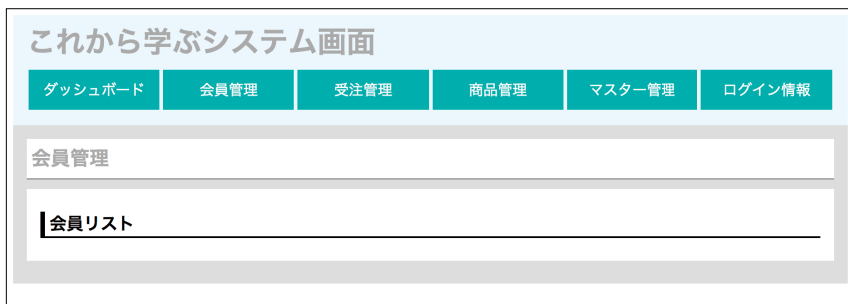
リスト14-4: main.css (追記分)

```
~省略~
/*main内のsectionボックスのスタイル設定*/
main section {
  /*背景色を設定*/
  background: white;
  /*内余白を15pxに設定*/
  padding: 15px;
  /*外余白を上下が10px、左右が0pxに設定*/
  margin: 10px 0px;
}
```

```
/*main内のsection内のh3ボックスのスタイル設定*/
main section h3 {
  /*外余白を上下が10px、左右が0pxに設定*/
  margin: 10px 0px;
  /*左内余白を5pxに設定*/
  padding-left: 5px;
  /*装飾としてボックスの左と下の境界線を設定*/
  border-left: solid 5px black;
  border-bottom: solid 2px black;
}
```

作成が終了したら、dashboard.htmlの「会員管理」リンクをクリックしてmemberList.htmlを表示させてください。図14-7のように表示されます。

図 14-7：表示させた会員リスト画面

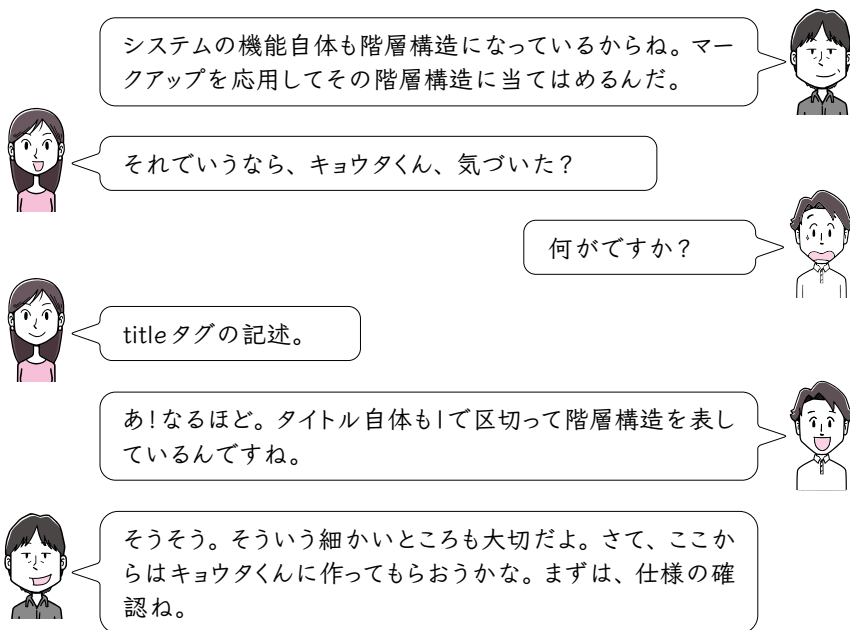


「ダッシュボード」として表示されていたmain内のh2を「会員管理」と変えて、その次にsectionタグを持ってきて、さらに、その中にh3。このh3が「会員リスト」という表示になっていて、この階層構造って、Chapter 5で学んだマークアップの応用なんですね！

そうそう。よく気づいてくれたね。



あのときは、文章のマークアップだったのでわかりやすかったですけど、それをシステム画面にどう持ち込むのだろうと思っていたら、こういう階層構造を取るんですね。



▶▶▶ 一覧表示では表示項目を絞る

これから扱う会員情報のデータ項目としては、以下のものがあります。

- 会員ID：システムが自動で付けた連番
- 会員姓：会員氏名の苗字部分
- 会員名：会員氏名の下の名前部分
- 会員姓ふりがな：会員姓のひらがな表記
- 会員名ふりがな：会員名のひらがな表記
- 電話番号：携帯電話番号を想定
- 性別：男女のどちらか
- 会員種類：通常、特別、優良のうちのどれか

一人の会員のデータとしては、上記の内容ですが、これらすべてを一覧表示として記載するには無理があります。そこで、以下のものに絞ります。

- 会員ID
- 会員氏名（姓と名を一続きに表示）
- 会員種類
- 電話番号



一覧表示項目をまずこのように絞るんだ。そして画面を作って、打ち合わせのときに、これでいいかどうかのフィードバックをもらうんだよ。そこで、足らなければ増やせばいいし、逆にいらないものがあったりもする。このあたりは、実際にそのシステムに触る人でないとわからないところだからね。

なるほど。



さて、とりあえず、上の絞った項目を一覧表示させるタグをリスト14-3のsectionタグ内、h3タグの次に記述してもらおうかな。キョウタくんなら、何を使う？

…。リスト画面だから最初はリストタグかなと思ったのですが、何か違うように思うのですよねえ。



リストタグでもできないことはないけど、もっとぴったりのがあるよ。

Chapter 7でやったよ。



あ！テーブル！

そう。まずは、一人分だけダミーデータを使って作成してみようか。



▶▶▶ 情報の一覧表示はテーブルが便利

7-1-1項で説明したように、システム画面での一覧表示というのはテーブルタグが見やすいです。ワトソン先生の指示通り、一人分を表示するようにキョウタくんが記述したコードはリスト14-5の太字の内容です。これを、memberList.htmlに追記してください。

リスト14-5: memberList.html (更新版)

～省略～

```
<section>
  <h3>会員リスト</h3>
  <table>
    <thead>
      <tr>
        <th>会員ID</th>
        <th>会員氏名</th>
        <th>会員種類</th>
        <th>電話番号</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td>4153587</td>
        <td>田中麻衣</td>
        <td>通常</td>
        <td>09012345678</td>
      </tr>
    </tbody>
  </table>
</section>
```

～省略～

さらに、テーブルタグを使うので、キョウタくんは11-3-2項で学んだテーブルタグ用のセレクトをmain.cssに追記しました。それがリスト14-6です。

リスト14-6: main.css (追記分)

～省略～

```
table {  
    border-collapse: collapse;  
}  
td, th {  
    border: 1px solid black;  
}
```

追記が終了したらmemberList.htmlを再表示させてください。図14-8のように表示されます。

図 14-8：表が追加された会員リスト画面



うん。基本はこれでいいよ。ちゃんと、テーブル用のセレクトも追記しているね。あとは、これをベースにtrタグをコピーしながら、ダミーデータを必要人数分だけ書いていけばいいね。ただ、図14-6の遷移図にある「詳細表示」のボタンが必要だよ。これ、どこに追加するかわかるかな？

…。



詳細表示は会員ひとりひとりの情報に対して行うよね。



ということは、この一人分の行中にボタンを追加する形ですか？



そうだね。こんな形になるよ。

▶▶ 詳細表示のボタンを追加する

memberList.htmlにリスト14-7の太字の部分を追記してください。

リスト14-7: memberList.html (更新版)

～省略～

```
<section>
  <h3>会員リスト</h3>
  <table>
    <thead>
      <tr>
        <th>会員ID</th>
        <th>会員氏名</th>
        <th>会員種類</th>
        <th>電話番号</th>
        <th>操作</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td>4153587</td>
        <td>田中麻衣</td>
        <td>通常</td>
        <td>09012345678</td>
        <td>
          <form action="/websamples/chap14/ =>
memberDetail.html" method="GET"> ←①
            <input type="hidden" name="memberId" =>
value="4153587"> ←②
```

```
<button type="submit">詳細表示</button>
</form>
</td>
</tr>
</table>
</section>
～省略～
```

追記が終了したらmemberList.htmlを再表示させてください。図14-9のように表示されます。

図 14-9：詳細表示ボタンが追加された会員リスト画面



図14-9からもわかるように、「操作」と表示された列を追加します。列を追加するので、それぞれの行にtdタグ（ヘッダ部分はthタグ）が追記されています。そのボタンが入ったtdタグ内に注目します（リスト14-7の❶～❸）。

この記述の中心は何といっても❸のbuttonタグです。しかし、これを使うためにはformタグで囲まないと機能しません。そのための記述が❶です。9-1-2項でも解説したように、このformタグのaction属性は通常サーバ側のプログラムを表すパスを記述しますが、ここでは、HTML画面モックですので、図14-6に記述された遷移先のhtmlファイルを指定しています。ここでのmethod属性としては、GETとしています。このmethod属性の値はプログラムと関わる場所なので、この段階では気にしなくてもいいです。ただ、詳細は割愛しますが、図14-6の遷移

図にある詳細画面から先、編集画面に移動し、その後、また詳細画面に戻ってくるときに、GETの方がプログラムが組みやすいのです。そのためにGETとしています。

では、②の記述はなんでしょう。これも、プログラムと関わる場所です。このボタンをクリックして、この会員の詳細情報を表示させる際、プログラム上はこの会員のid情報を入手しておかないと、表示するための会員情報が用意できません。そこで、それをinput type hiddenタグとして記述しておきます。そうすることで、ボタンをクリックした時にサーバにこの会員のid情報が送信されます。



先生が、システムの画面モックはプログラマが作成しないとダメ、と言っていた意味が少しわかりました。この、GETに設定したり、hiddenを記述したりなんてのは、今の僕にはまだわかりませんもん。多分、プログラムを知らない人が画面を作成したら、僕と同じように、こんな記述はできません。

うん。そうだね。プログラムを知っているからこそ、出てくる発想なんだ。逆に、キョウタくんは、今は思い浮かなくていいからね。むしろ、ここで、そういったことを何となく理解しておいて、この後、Webプログラミング、特にPHPやJavaなどのサーバ上で動くプログラミングを学んだときに、はっきり理解すればいいよ。



はい。

さあ、リスト14-7でとりあえず、一人分の表示ができたから、あとは、これをコピペして何人か表示して、表らしくしておこうか。



ここでは紙面の都合上、ソースコードの掲載は割愛して結果のみ表示しておきます。図14-10のようになります。皆さんもぜひコーディングしてください。

図 14-10：表示行数が増えた会員リスト画面



ここでは5人分になっています。もちろん、打ち合わせ上この人数では足りない場合は、さらにtrタグをコピー＆ペーストして行数を増やします。

[14-2-3 会員情報詳細表示画面]

一覧表示が一通り完成したところで、次に詳細表示に移りましょう。表中の「詳細表示」ボタンを押したときに表示される画面を作成します。



詳細表示では、14-2-2項に掲載しているデータ項目のすべてを表示させるとするね。キョウタくんだったらどのような形式で表示させる？

やっぱり表ですかね？



表示の向きは縦、横、どっち？

…。縦ですか？





正解。このあたりは7-2-2項で解説したから復習になるね。こういう詳細表示は図7-12のようにヘッダ部分を列にした表の方が見やすいね。

じゃあ、早速tableタグで書いてみます。



ちょっと待った。確かに、表だからtableタグで書きたくなると思うけど、ここではあえて違うタグを使ってみるよ。

どんなタグですか？



dlタグ。



え!リストですか？



考えてみて。縦の詳細表のヘッダ部分って意味的にdtでしょ。で、その内容がdd。実は、縦の詳細表ってdlタグで記述できるのよ。



なるほど!



じゃあ、早速やってみようか。



ベース部分はmemberList.htmlですので、memberList.htmlをコピーしてmemberDetail.htmlとして、mainタグ内のsectionタグ内をリスト14-8のように丸々変更してください。

リスト14-8: memberDetail.html (一部)

```
<section>
  <h3>会員情報詳細</h3>
  <dl class="detail"> ←①
    <dt>会員ID</dt> ←②
    <dd>4153587</dd> ←③
```

```
<dt>会員氏名</dt>
<dd>田中 麻衣</dd>
<dt>会員氏名ふりがな</dt>
<dd>たなか まい</dd>
<dt>電話番号</dt>
<dd>09012345678</dd>
<dt>性別</dt>
<dd>女</dd>
<dt>会員種類</dt>
<dd>通常</dd>
</dl>
<p>この会員情報を編集する場合は<a href="/websamples/chap14/ =>
memberEdit.html">こちら</a>から</p>
<p>この会員情報を削除する場合は<a href="/websamples/chap14/ =>
memberConfirmDel.html">こちら</a>から</p>
</section>
```

この段階で、一度会員リストの「詳細表示」ボタンをクリックしてmemberDetail.htmlを表示させてください。図14-11のように表示されます。

図 14-11：表示させた会員情報詳細画面



リスト14-8では、「会員ID」のような表示項目名（表ならばヘッダ列）に当たる内容をdtとして記述しています（例えばリスト14-8の❷）。それに対応する内容を、例えば❸のようにddタグで記述します。これらの繰り返し全体を❶のdlタグで囲んでいます。そのdlタグには、のちにCSSでレイアウトさせるためのクラス名としてdetailを記述しています。

なお、❹は会員情報編集画面、会員情報削除確認画面へ遷移するためのリンクです。図14-6の遷移図にもあるように、この詳細画面からは編集と削除へ遷移するようになっていきますので、そのリンクをあらかじめ記述しています。ただ、遷移先ファイルは存在しませんので、現状はリンクをクリックしても表示されません。



このdlで書くというのは思いつかなかったです。図14-11ではスタイルが設定されていませんので、表とは違いますが、それでも、確かに詳細表示っぽいのがよくわかります。

でしょ？じゃあ、これにスタイルを当てはめて表っぽくするね。



main.cssにリスト14-9の内容を追記してください。

リスト14-9：main.css（追記分）

～省略～

```
/*detailクラスのdl、dt、ddをボーダー基準に設定*/
.detail, dt, dd {
    box-sizing: border-box;
}
/*detailクラスボックスのスタイル設定*/
.detail {
    /*幅を100%に設定*/
    width: 100%;
    /*境界線を設定*/
    border: 1px solid black;
    /*フレックスボックスレイアウトに設定*/
    display: flex; ←❶
```

```

/*フレックスボックスを横方向に設定*/
flex-direction: row; ←②
/*フレックスボックスの折り返しを設定*/
flex-wrap: wrap; ←③
}
/*detailクラス内のdtとddボックスのスタイル設定*/
.detail dt, dd {
  /*外余白を0pxに設定*/
  margin: 0px;
  /*内余白の上下を5px、左右を15pxに設定*/
  padding: 5px 10px;
  /*境界線を設定*/
  border: 1px solid black;
}
/*detailクラス内のdtボックスのスタイル設定*/
.detail dt {
  /*太字に設定*/
  font-weight: bold;
  /*背景色を設定*/
  background: gainsboro;
  /*フレックスアイテムを幅30%で設定*/
  flex: 1 0 30%; ←④
}
/*detailクラス内のddボックスのスタイル設定*/
.detail dd {
  /*フレックスアイテムを幅70%で設定*/
  flex: 1 0 70%; ←⑤
}

```

追記が終了したらmemberDetail.htmlを再表示させてください。図14-12のように表示されます。

図 14-12：表形式のスタイルが適用された会員情報詳細画面

これから学ぶシステム画面

ダッシュボード

会員管理

受注管理

商品管理

マスター管理

ログイン情報

会員管理

会員情報詳細

会員ID	4153587
会員氏名	田中 麻衣
会員氏名ふりがな	たなか まい
電話番号	09012345678
性別	女
会員種類	通常

この会員情報を編集する場合は[こちら](#)から

この会員情報を削除する場合は[こちら](#)から



本当だ！縦表になった！

ポイントはどこかわかる？



やっぱりフレックスボックスレイアウトですか？

そうそう。



▶▶▶ dlにフレックスボックスレイアウトを適用

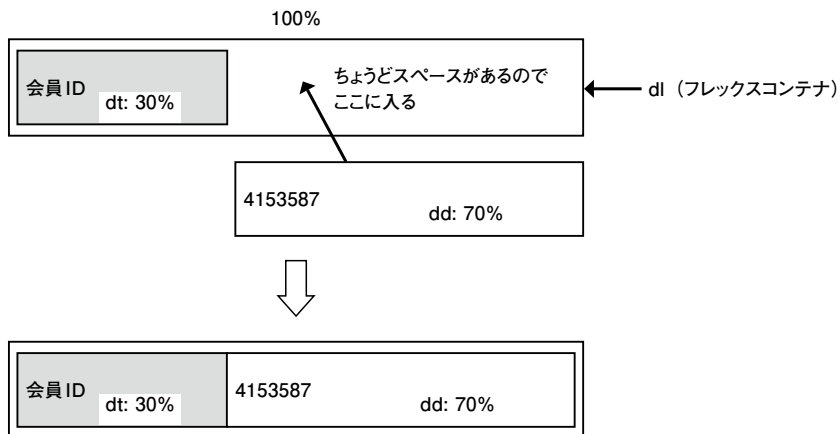
dlにフレックスボックスレイアウトを適用することで、表のように表示させることができます。それが、リスト14-9の①～③です。これは、13-2-2項の復習です。①によって、detailクラスで指定したdlタグがフレックスコンテナとなります。②で方向が横向きとなり、③で折り返しが可能となります。

次に、dtとddそれぞれをフレックスアイテムとして設定します。その際、flex-

basisの値としてそれぞれ30%（リスト14-9の❹）、70%（リスト14-9の❺）と指定しています。この指定が%指定であることと、両者を合わせて100%になることがポイントです。

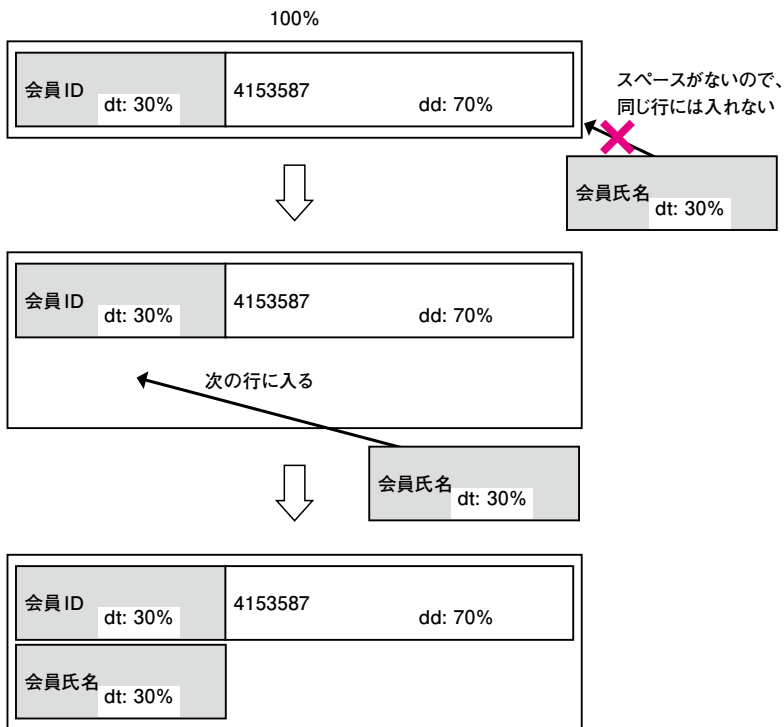
まず、フレックスコンテナに最初の30%幅のdtが格納されます。次に、70%幅のddがその右横に格納されます。これは、ちょうど収まるスペースがあるので1行に収まり、表示されます（図14-13）。

図 14-13：1行目のdtとddが格納される様子



問題はその次です。2個目の30%幅のdtを同じ行に格納しようとしませんが、スペースがありません。そこで、リスト14-9❸の折り返しの設定通りに次の行にこのdtが収まります（図14-14）。

図 14-14：2行目が作られる様子



4個目のddは2個目のddと同じ流れで3個目のdtの右横に収まります。以降、次々とこの処理を繰り返すことで、結果、dtとddのペアで1行が構成されながら表のような表示となります。

▶▶▶ 縦向きの詳細表示表ではdlタグの方が使い勝手がいい

このように、スタイルを設定すれば、縦向きの詳細表示表は、tableタグよりも、リスト14-9のようにdlタグを使った方が使い勝手がいいことの方が多いです。これは、主にレスポンシブレイアウトと関係があります。tableタグというのはなかなかレスポンシブレイアウトにはしにくいです。一方、dlタグとフレックスボックスを組み合わせた場合、レスポンシブレイアウトとなります。これについては、もう一度後で扱います。

[14-2-4 会員情報登録画面]

ここまでは表示画面ばかりでした。ここで、入力画面を扱っておきましょう。

▶▶▶ リスト画面に新規登録画面へのリンクを追加する

ナビゲーションの「会員管理」をクリックするなどして、会員リスト画面を表示させてください。図14-6の画面遷移図では、このリスト画面から新規登録画面に遷移するようになっています。まずは、そのリンクを追加しましょう。



現状、図 14-10 のようなリスト画面だけど、これに新規登録画面へのリンクを追加しようと思うんだけど、キョウタくんならどう追加する？

…。さっきの詳細画面のようにこの表の下にpタグを追加してそこに「こちらへ」みたいなリンクを追加するのがいいんでしょうか？うーん、なんか違うような気が…。



うん。確かに違うね。「一覧表示」というのと「新規登録」というのは、機能としては別だよな。だから、それぞれ別の section として表示させるんだ。

なるほど！じゃあ、やってみます。



キョウタくんと同じように、memberList.htmlにリスト14-10の太字の部分を追記してください。

リスト14-10: memberList.html (更新版)

～省略～

```
<section>
```

```
  <h3>会員リスト</h3>
```

～省略～

```
</section>
```

```

<section>
  <h3>会員新規登録</h3>
  <p>新規登録は<a href="/websamples/chap14/memberAdd. ⇒
html">こちら</a>から行ってください。</p>
</section>
  ~省略~

```

追記が終了したらmemberList.htmlを再表示させてください。図14-15のように表示されます。

図 14-15：新規登録へのリンクが追加された会員リスト画面

これから学ぶシステム画面

ダッシュボード

会員管理

受注管理

商品管理

マスター管理

ログイン情報

会員管理

会員リスト

会員ID	会員氏名	会員種類	電話番号	操作
4153587	田中麻衣	通常	09012345678	詳細表示
5644124	中田忠義	特別	09098745632	詳細表示
3354644	村田裕子	通常	08023456789	詳細表示
1452246	石田悦子	通常	07034567891	詳細表示
2457712	坂東由紀	特別	09045678923	詳細表示

会員新規登録

新規登録は[こちら](#)から行ってください。

▶▶▶ 新規登録画面を作成

このリンクをクリックした後に表示される会員情報新規登録画面を作成しましょう。入力項目としては、14-2-2項記載の会員情報のデータ項目とします。ただし、会員IDに関しては、システム側で自動採番することになっているので、入力欄は不要です。また、すべての項目を必須項目とします。



ひとつ注意しなければならないのは、詳細表示では「会員氏名」としてひとつの欄で姓と名をくっつけて表示させていたけど、入力欄では当然別々になるということね。

あ!危うくひとつにしてしまうところでした。



でしょうね。データ項目が姓と名で別々なら、入力欄に関してはそれに合わせて別々にする必要があるんだ。次に、性別はどういった入力部品を使う?

これは、9-3-2項で扱っているので覚えています。ラジオボタンですね。



じゃあ、会員種類は?

うーん。これもラジオボタンですか? 単一選択だからチェックボックスはあり得ないですし。



うん。ラジオボタンでもいいけど、会員種類が増えた場合とかを考えると、ドロップダウンリストでもいいね。

selectタグですね!



どちらでもいいけど、どちらで作る?

せっかくなので、selectタグを使ってみます。



全体のレイアウトとしては、どうする?

詳細表示と同じ縦並びの表がいいのかなと思います。表ですけど、14-2-3項で学んだように、dlを使ってみます。



うん。それでいいよ。じゃあ、やってみよう。

詳細表示と同じレイアウトですので、ベース部分はmemberDetail.htmlです。memberDetail.htmlをコピーしてmemberAdd.htmlとして、mainタグ内のsectionタグ内をリスト14-11のように丸々変更してください。

リスト14-11: memberAdd.html (一部)

```
<section>
  <h3>会員新規登録</h3>
  <p>
    必要事項を入力して、登録ボタンをクリックしてください。<br>
    なお、会員IDはシステム側で自動採番します。
  </p>
  <form action="/websamples/chap14/memberList.html" ⇒
method="POST"> ←①
    <dl class="detail">
      <dt><label for="memberNameLast">会員姓</label></ ⇒
dt> ←②
      <dd>
        <input type="text" name="memberNameLast" ⇒
id="memberNameLast" placeholder="例) 柴田" required> ←③
      </dd>
      <dt><label for="memberNameFirst">会員名</ ⇒
label></dt>
      <dd>
        <input type="text" name="memberNameFirst" ⇒
id="memberNameFirst" placeholder="例) 愛理" required>
      </dd>
      <dt><label for="memberNameLastKana">会員姓ふりがな ⇒
</label></dt>
      <dd>
        <input type="text" name="memberNameLastKana" ⇒
id="memberNameLastKana" placeholder="例) しばた" required> ⇒
      </dd>
      <dt><label for="memberNameFirstKana">会員名ふりが ⇒
な</label></dt>
      <dd>
        <input type="text" name="memberNameFirstKana" ⇒
id="memberNameFirstKana" placeholder="例) あいり" required>
```

```

        </dd>
        <dt><label for="memberTel">電話番号</label></dt>
        <dd>
            <input type="tel" name="memberTel" ⇒
            id="memberTel" placeholder="例)09045785547" required> ←④
        </dd>
        <dt>性別</dt>
        <dd>
            <label for="memberSexFemale"><input ⇒
            type="radio" name="memberSex" id="memberSexFemale" ⇒
            value="0" checked>女</label>
            <label for="memberSexMale"><input ⇒
            type="radio" name="memberSex" id="memberSexMale" ⇒
            value="1">男</label>
        </dd>
        <dt>会員種類</dt>
        <dd>
            <select name="memberType" id="memberType" ⇒
            required>
                <option value="">選択してください</option>
                <option value="1">通常</option>
                <option value="2">特別</option>
                <option value="3">優良</option>
            </select>
        </dd>
    </dl>
    <p class="submitP"> ←⑤
        <button type="submit">登録</button>
    </p>
</form> ←①
</section>

```

main.cssへ追記しなければならないのは、リスト14-11⑤のPタグで指定しているsubmitPクラスセレクタだけです。リスト14-12の内容をmain.cssに追記してください。

リスト14-12: main.css (追記分)

～省略～

```
/*submitPボックス内をセンタリングに設定*/  
.submitP {  
    text-align: center;  
}
```

このスタイルは、コメントにある通り、リスト14-11の⑤のpタグ内の登録ボタンをセンタリングさせるためです。

この段階で、一度会員リストの新規登録画面へのリンクをクリックしてmemberAdd.htmlを表示させてください。図14-16のように表示されます。

図14-16: 表示させた会員情報新規登録画面

これから学ぶシステム画面

[ダッシュボード](#)[会員管理](#)[受注管理](#)[商品管理](#)[マスター管理](#)[ログイン情報](#)

会員管理

会員新規登録

必要事項を入力して、登録ボタンをクリックしてください。
なお、会員IDはシステム側で自動採番します。

会員姓	例)栗田
会員名	例)愛理
会員姓ふりがな	例)しばた
会員名ふりがな	例)あいり
電話番号	例)09045785547
性別	☒ 女 ☐ 男
会員種類	選択してください

登録

▶▶▶ 入力欄全体をformで囲む

レイアウト全体としては、詳細表示画面と同じくdlを使って縦表のように表示されています。スタイルは同じものを流用しています。ここでのポイントは、入力画

面となっているので、このdlタグ全体をformタグで囲む必要があることです。それがリスト14-11の❶です。また、データを送信するためのボタンが必要ですので、それをdlタグの次にpタグで囲んで記述しています（リスト14-11の❷）。ここまでがformタグの範囲です。

なお、このformタグのmethod属性としてはPOSTを指定しています。9-1-2項でも解説したように、ここで入力するような個人情報に当たるものは、原則POSTで送信するようにします。

▶▶▶ label内に入力部品がない場合は必ずidを指定

そのformタグ内に記述された入力部品はすべてChapter 9の復習です。ここでは会員姓の入力欄を例として説明します。dtタグに記述された入力項目名は、そのまま入力欄のラベルとなるのでlabelタグで囲みます（リスト14-11の❸）。ただ、Chapter 9で紹介したサンプルでは、labelタグ内にinputタグが含まれていました。ここでは、そのinputタグはタグをまたいだddタグに記述する必要があるので、labelタグ内には記述できません。そこで、「会員姓」のようなラベル名称だけを記述し、必ずfor属性としてidを指定し、別タグ内に記述した入力部品と結びつけています。

▶▶▶ placeholderやrequired属性を活用

その会員姓の入力部品を実際に記述しているのが、リスト14-11の❹です。ここでは、placeholder属性を活用し、入力例を記述しています。さらに、すべてのデータ項目が必須ですので、required属性も付与しています。このplaceholderとrequired属性はその後のテキストボックスにはすべて付与しています。

なお、リスト14-11❺の電話番号入力欄に関しては、type属性としてtelを指定しています。

▶▶▶ dlだとレスポンスブレイアウト

新規登録画面が完成したところで、14-2-3項の最後にdlで作成した縦向きの表がレスポンス対応になっていることを説明しましたが、それを確認してみましょ

う。この画面をスマホサイズまで狭めてください。ナビゲーションが縦並びになることはChapter 13で体験済みです。dlで記述した表部分も図 14-17のように縦並びになっていることがわかります。このように柔軟なレイアウトはtableでは難しいです。

図 14-17：入力欄のdlが縦並びに変更された画面

これから学ぶシステム画面

ダッシュボード

会員管理

受注管理

商品管理

マスター管理

ログイン情報

会員管理

会員新規登録

必要事項を入力して、登録ボタンをクリックしてください。
なお、会員IDはシステム側で自動採番します。

会員姓

例)柴田

会員名

例)愛理

会員姓ふりがな

例)しばた

会員名ふりがな

例)あいり

電話番号

例)09045785547

性別

☒女 ☐男

会員種類

選択してください ▼

登録



わあ!これはすごいですね。確かに、tableタグではこうはいかないですね。

でしょ?



フレックスボックスおそろべしです。ところで、先生、気になっていたのですが、ここまで作ってきた画面って、今自分がどこにいるのかわからなくなります?例えば、詳細画面からリスト画面に戻るには、どうすればいいのかなって。確かに、ナビゲーションの[会員管理]をクリックすればリストに戻るのですが、もっとわかりやすい方法ってないものかと思って。

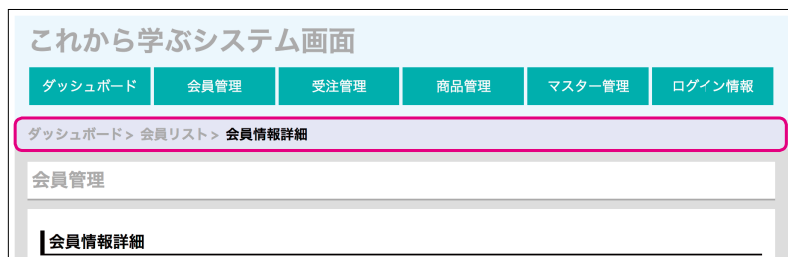
いいところに気づいたね。パンくずリストという、いいのがあるんだ。それを最後にやっておこうか。



[14-2-5 パンくずリスト]

会話の中でワトソン先生が紹介していた**パンくずリスト**というのは、図14-18の赤で囲まれた部分の表示です。

図 14-18 : 詳細画面に追加されたパンくずリスト



この「パンくず」というのは、グリム童話の『ヘンゼルとグレーテル』で森の中に入るときに、森で迷わないようにパンくずを置いて道しるべにしたという話から来ています。これと同様に、WebサイトやWebシステムの中で自分がどこにいるの

か、どういう経路でこの画面に来たかを表す図14-18のようなリストを**パンくずリスト**と呼んでいます。

▶▶▶ パンくずリストもナビゲーションの一種

では早速、詳細画面にこのパンくずリストを追記しましょう。memberDetail.htmlのheaderタグとmainタグの間にリスト14-13の内容を追記してください。

リスト14-13: memberDetail.html (追記分)

```
～省略～
</header>
<nav id="breadcrumb">
  <ul>
    <li><a href="/websamples/chap14/dashboard.html">ダ⇒
    ッシュボード</a></li>
    <li><a href="/websamples/chap14/memberList.html">会⇒
    員リスト</a></li>
    <li>会員情報詳細</li>
  </ul>
</nav>
<main>
～省略～
```

リスト14-13を見てもわかるように、パンくずリストもナビゲーションの一種なので、navタグで囲み、ulタグとliタグを使います。これを図14-18のように表示させるのがCSSの働きです。

▶▶▶ 各リンク間の「>」はスタイルとして設定

リスト14-13で追記したnavタグには、パンくずリスト用のnavであることを明示するためにid属性としてbreadcrumbを記述しています。idセレクトアとしてbreadcrumbを記述することで、パンくずリストとして表示されるようにしていきます。main.cssにリスト14-14の内容を追記してください。

リスト14-14: main.css (追記分)

～省略～

```
/*idがbreadcrumbボックス(パンくずリストボックス)のスタイル設定*/
#breadcrumb {
    /*背景色を設定*/
    background:lavender;
}
/*パンくずリスト内のulボックスのスタイル設定*/
#breadcrumb ul {
    /*外余白を0pxに設定*/
    margin: 0px;
    /*内余白の上下を5px、左右を15pxに設定*/
    padding: 5px 15px;
}
/*パンくずリスト内のliボックスのスタイル設定*/
#breadcrumb li {
    /*インラインとして表示するように設定*/
    display: inline; ←①
    /*リストのスタイルを無効に設定*/
    list-style: none;
    /*太字に設定*/
    font-weight: bold;
}
/*パンくずリスト内のliボックスの後に「>」を表示するように設定*/
#breadcrumb li::after{ ←②
    content: ">";
    padding: 0 3px;
    color: darkgrey;
}
/*一番最後のliボックスの後には「>」を表示しないように設定*/
#breadcrumb li:last-child::after{ ←③
    content: "";
}
/*liボックス内のリンクのスタイル設定*/
#breadcrumb li a {
    /*文字装飾を無効に設定*/
    text-decoration: none;
```

```

/*文字色を設定*/
color: darkgrey;
}
/*liボックス内のリンクにマウスが重なった時のスタイル設定*/
#breadcrumb li a:hover {
/*文字色を設定*/
color: coral;
}

```

追記が終了したらmemberDetail.htmlを再表示させてください。図14-19のように表示されます。

図 14-19：入バンくずリストが表示された詳細画面

これから学ぶシステム画面

[ダッシュボード](#)
[会員管理](#)
[受注管理](#)
[商品管理](#)
[マスター管理](#)
[ログイン情報](#)

ダッシュボード> 会員リスト> 会員情報詳細

会員管理

会員情報詳細

会員ID	4153587
会員氏名	田中 麻衣
会員氏名ふりがな	たなか まい
電話番号	09012345678
性別	女
会員種類	通常

この会員情報を編集する場合は[こちら](#)から
この会員情報を削除する場合は[こちら](#)から

リスト14-14のポイントは、まず①でインライン設定をしていることです。これで、各liタグが横並びになります。

さらに、②と③で各リンク間にある「>」をスタイル設定だけで表示するようにしていることです。実は、②の記述は12-3-4項で紹介したクリアフィックスとほぼ同じです。疑似要素の::afterを利用してそこに「>」を表示させています。

では、❸はどういう記述でしょうか。これは、コメントにあるように、一番最後のli要素のさらに後ろに「>」が表示されないようにしています。疑似セクタ:last-childを使って、一番最後の要素を取得し、さらに、疑似要素の::afterを利用してそこに「>」と空文字を設定することで、「>」を表示させないようにしています。



このパンくずリスト、めっちゃ見たことあります！

でしょ？WebサイトやWebシステムでは必須の表示といえるわね。



リスト14-14で追記したセクタは、パンくずリストを表示させるための典型的な記述なので、他のでパンくずリストを表示させたい場合は、これをベースに改編していけばいいよ。じゃあ、他の2画面にもパンくずリストを追加しようか。

会員リスト画面、会員情報新規登録画面にもパンくずリストを追記しましょう。

まず、会員リスト画面です。memberDetail.html同様、memberList.htmlのheaderタグとmainタグの間にリスト14-15の内容を追記してください。

リスト14-15：memberList.html（追記分）

```

～省略～
</header>
<nav id="breadcrumb">
  <ul>
    <li><a href="/websamples/chap14/dashboard.html">ダ⇒
    ッッシュボード</a></li>
    <li>会員リスト</li>
  </ul>
</nav>
<main>
～省略～

```

次に、会員情報新規登録画面です。同様に、memberAdd.htmlのheaderタグとmainタグの間にリスト14-16の内容を追記してください。

リスト14-16: memberAdd.html (追記分)

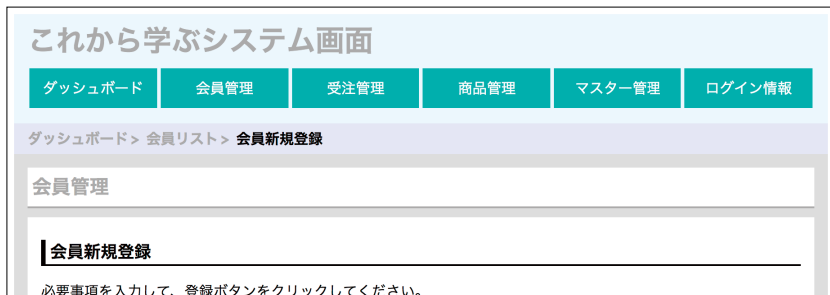
```
～省略～
</header>
<nav id="breadcrumb">
  <ul>
    <li><a href="/websamples/chap14/dashboard.html">ダ⇒
    ッッシュボード</a></li>
    <li><a href="/websamples/chap14/memberList.html">会⇒
    員リスト</a></li>
    <li>会員新規登録</li>
  </ul>
</nav>
<main>
～省略～
```

追記が完了したら、それぞれの画面に図14-20や図14-21のようにちゃんとパンくずが表示されているか確認してください。

図 14-20: 会員リスト画面に追加されたパンくずリスト



図 14-21: 会員情報新規登録画面に追加されたパンくずリスト





以上で、本当にこの講座はすべて終了です。

おつかれさま。



ありがとうございました。

番外編のこのChapter 14はどうでした？



ものすごく実践的でびっくりしました。でも、本編で学んできたことが、このようにつながっていくんだというのが実感できておもしろかったです。

そう言ってもらえたら、こちらも解説した甲斐がある。これを活かして次に進んでいってね。



はい！

皆様も、本編に引き続き、この番外編までお付き合いのほどを、ありがとうございました。

