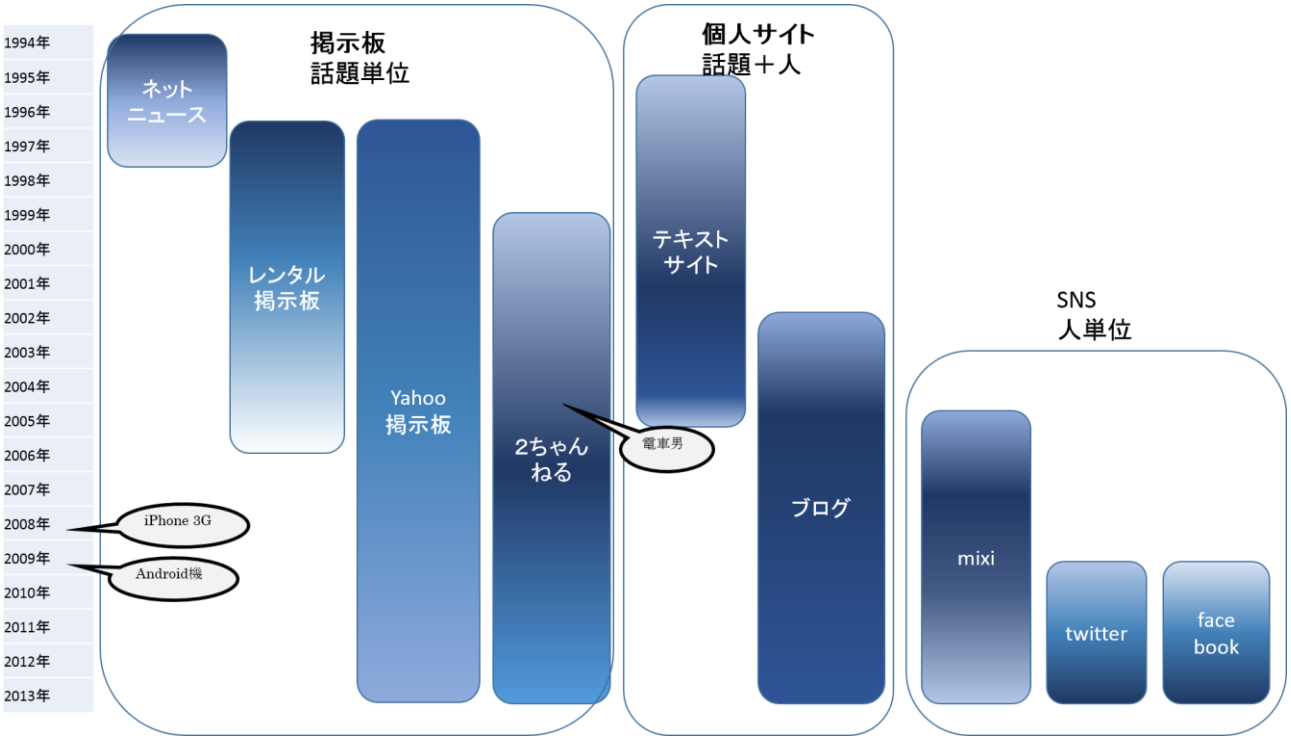
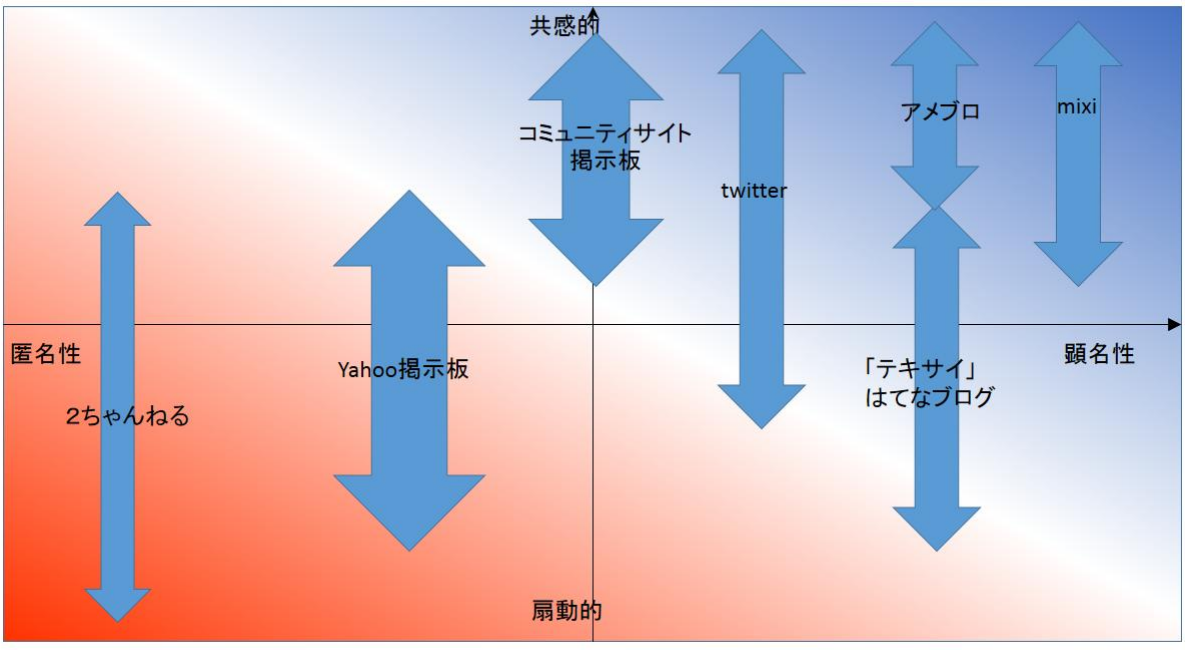
インターネットにおける情報発信技術の歴史





- ✓ 話題（トピック）単位で情報にアクセス→＋人単位で情報にアクセス
  - ✧ 情報発信・収集とコミュニケーション
- ✓ 匿名性と顕名性との揺らぎ、バランス取り
- ➢ 考察してみよう
  - ✧ インターネットにおいて匿名性が尊重された理由とその限界を考えてみよう
  - ✧ ２ちゃんねる隆盛期と日本社会の状況との関係を考察してみよう

HTML 入門

ネット上に情報発信を行う技術は上記ブログ、掲示板、SNS ほか CMS(Contents manegiment System)など様々にあり、その多くはウェブ技術の詳細を知る必要がない。イイタイコト、情報さえ持っていれば、既存のシステムを用いるだけで情報が発信できるようになっている。

そうした状況の中で、あえてウェブ技術を学ぶとすれば、その意義は

- ✧ 既存のシステムに縛られないより自由な形式で情報発信を行いたい
- ✧ システムを提供する立場として仕事をしたい

などが考えられる。

ウェブ技術を学ぶことはウェブによる情報発信の可能性をさらに広げるその礎となるものである。

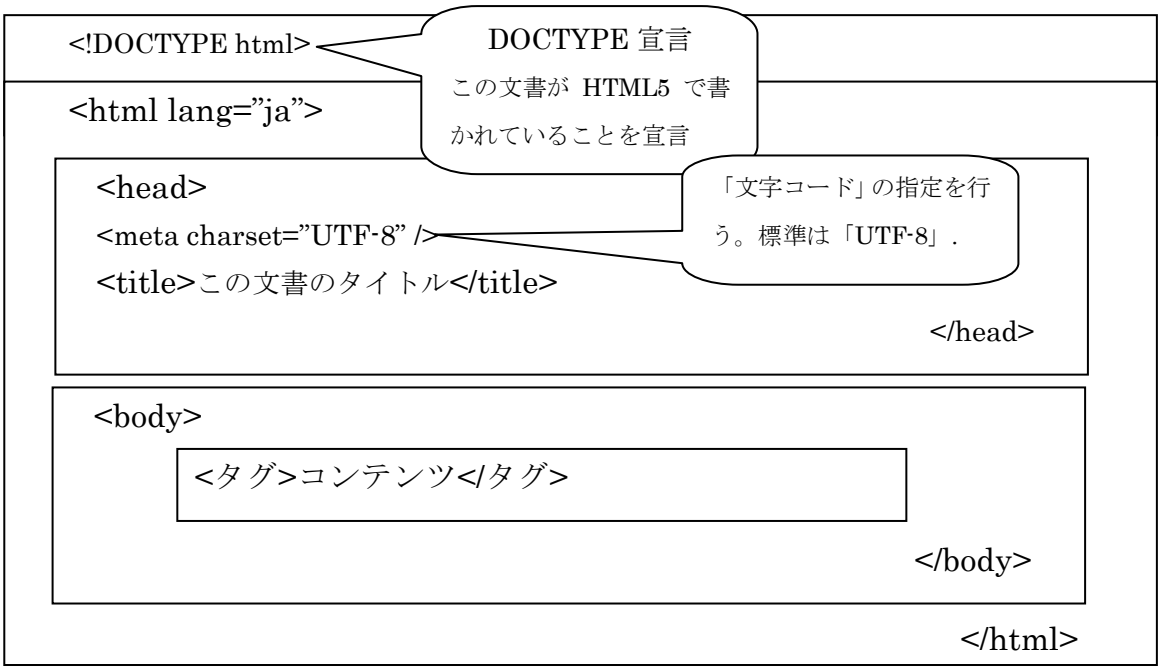
将来ウェブ技術を開発するシステムエンジニアになるとか、ウェブデザイナーになるとかそういう方向性を可能性として考えるためには最低限のウェブ技術には触れておかなければならない。またそうした専門家になるのでなくとも、様々な情報を発信していく広範な立場において、より広い視野でウェブ技術に触れることで情報発信する際のインスピレーションがさらにかき立てられることもあるだろう。

繰り返すが、一般的に情報発信を行うだけならば、ウェブ技術の中身まで知る必要はさほどない。しかしそれに触れることで少しでも自由（な発想）を得られるのだとすれば、一度は学んでおいて損することはない。

そしてさらにウェブ技術の基本をなす html(HyperText Markup language)にはインターネットという技術・世界に人々が託した（情報に関する）様々な理想が込められている。その理想を少しでも垣間見られるとすれば、それはそれでとても意義のある事だと思いませんか？

- ➢ html3.2→html4.01→html5 という html の歴史を学んでみよう。

1.1. HTML の基本構造



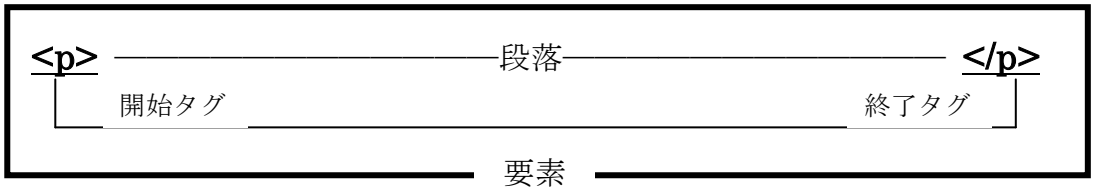
1.2. HTML タグ一覧(html5)

header	ヘッダー領域	em	強調
nav	ナビゲーション領域	strong	強い強調
article	記事領域	small	注釈・細目
aside	付加情報領域	s	現在正しくない古い情報
footer	フッター領域	b	キーワード
address	作成者情報	i	心の中で思ったことなど
section	セクション	cite	作品タイトル
h1～h6	見出し	q	引用文
p	段落	br	改行
hr	区切り	span	ひとかたまりの範囲
blockquote	引用・転載	a	リンクを指定する
ol・ul	リスト:序列あり・なし		
li	リストの中身		
dl	定義リスト		
dt	定義語		
dd	説明文		
div	上記以外の何らかの塊		
table	表	img	画像を表示する
tr	行	audio	音声を再生する
th・td	セル:見出し・データ	video	映像を再生する

表のように HTML は原則的には文書構造を記述するためのタグであって、どのように表示されるかを定義するためのものではない。表示設定はスタイルシートを用いる。

1.3. HTML(文書構造のマークアップ)の書き方

「開始タグ」と「終了タグ」

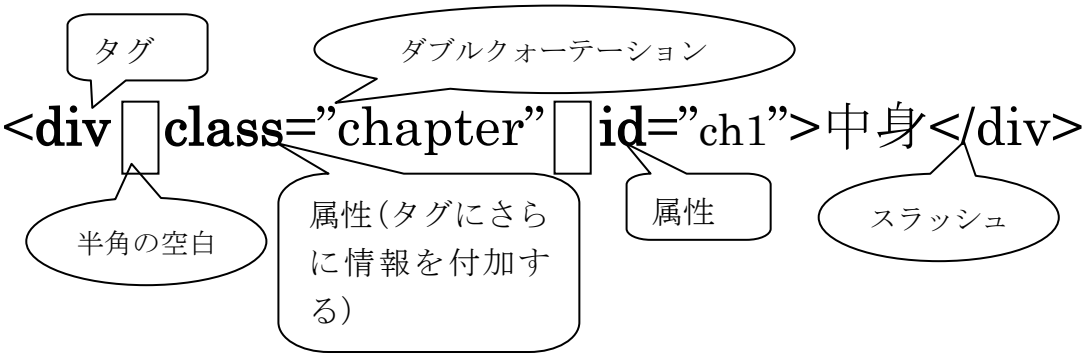


HTML のタグは基本的には「開始タグ」と「終了タグ」を一組で使用する。

> ** **
> 例外的なタグ（hr と br）を除いて、基本的にタグは「**開いたら閉じる**」を忘れないようにしましょう。
> ** **

<hr>と<img>,
には終了タグは必要ない（<hr />,<img />,
と書いても良い）。

HTML の書式

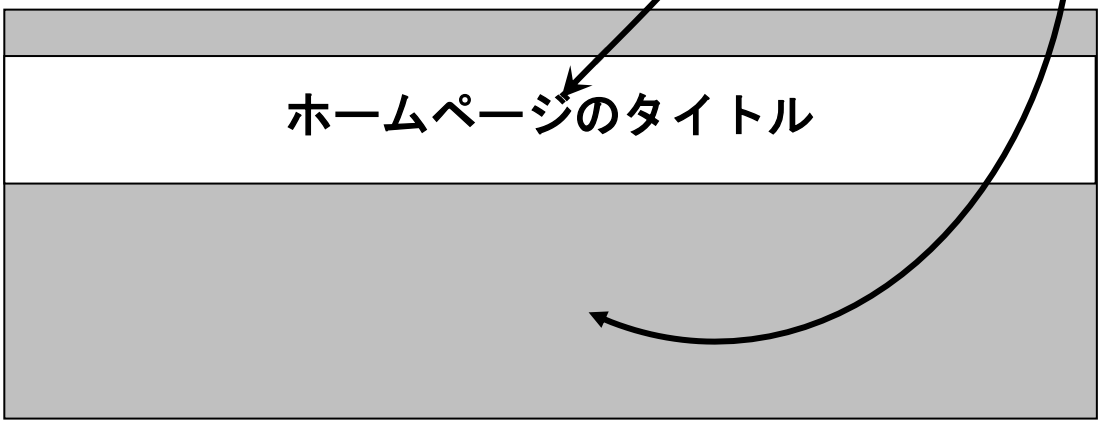


1.4. CSS（Cascading Style Sheets:表示設定）の書き方

```
<head>
<title>ホームページのタイトル</title>
    <style>
        body {
            background: silver;
        }

        h1 {
            background: white;
            font-size: 2em;
            text-align: center;
        }
    </style>
</head>
<body>
<header>
<h1>ホームページのタイトル</h1>
</header>
```

これが「**スタイルシート（CSS)**」です。
文字の色や大きさ、背景色などなどを自由に設定できる。



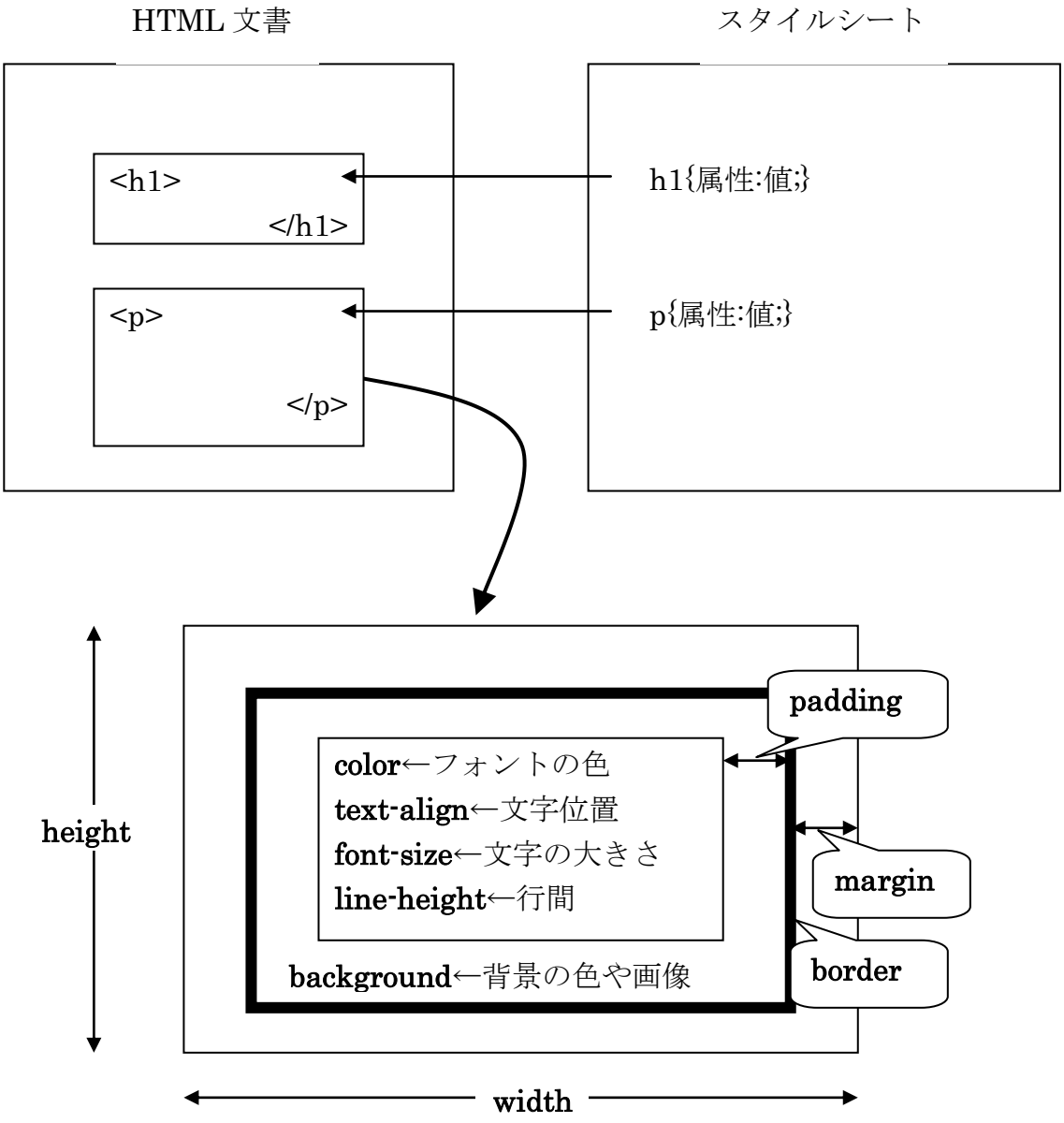
CSS の指定方法

```
<head>
<style>
        h1 {
            font-size: フォントの大きさ ;
        }
</style>
</head>
```

タグ名　ハイフン　コロン　セミコロン

ボックスへの表示設定

CSS（スタイルシート）は HTML の諸要素に対して表示設定を施すが、このとき HTML の要素は「ボックス」という単位で扱われる。HTML は白紙の上にボックスを重ねていくイメージで書かれている。CSS はこのボックスの様態（大きさ、余白、ボーダーライン、色など）と、その中のコンテンツ（文字の色、文字位置など）に対してさまざまな表示設定を行なうことができる。



1.5. HTML タグ辞典

以下、HTML でさしあたり覚えておけばよい代表的なタグを紹介する。すべてを丁寧に覚えたりする必要はない。大切なのは HTML のソースを読んで自分なりに理解できるようになることである。いろいろなタグやその詳細なコントロールはそのつど調べ、次第に覚えていけばよい。

文書構造 (header,nav,article,section,footer など)

```
<header>
<nav>サイト内ナビゲーション</nav>
見出しなど
</header>
<article>
記事本体（内部を section などで区切る）
</article>
<footer>著作者情報など</footer>
```

ページ内の文書構造の大枠を指定する。

見出し (h **eading**)

```
<h1>見出し 1 </h1>
   <h2>見出し 2</h2>
      ・・・・・・・
         <h6>見出し 6</h6>
```

文章のタイトルから、章、節、項、といった具合にセクションごとに見出しをつけるのに用いる。ワープロソフトの「アウトライン」機能に相当する。

このとき見出しは **h1** から順番に使わなければならない。**h1** が大きく表示されすぎるからといって **h2** からはじめるという使用方法は不可である。**見出しタグは文字の大きさを決めるためのタグではない**。

文字の大きさは CSS で自由にコントロールできる。また中央に持ってきたり、色を変えたりするのも CSS で行なうことができる。

CSS で文字サイズを指定する h1{font-size: 1.6em;}
CSS で文字を中央（右）に持ってくる h1{text-align: center(right);}
CSS で文字の色を変える（赤） h1{color: red;}

段落（p **aragraph**)

```
<p>
　日本語では段落という概念が希薄ですが、小学校のときに「改行して一文字あける」と習った、あれのことです。HTML では「一文字あける」には一手間必要ですが、考え方は変わりません。
</p>
```

p は改行をするためのタグではない。あくまで文章の内容的なひとまとまりである「段落」の範囲を示すためのタグである。もちろん開始タグと終了タグは必須。

日本語の段落表示の伝統的手法として「一文字あける」（インデント）のは CSS を用いる。

```
CSS で一文字分のインデントを設定する
p {text-indent: 1em;}
```

```
CSS で行間を決める(1.4 倍に広げる)
p{line-height: 140%;}
```

class 属性と id 属性

p に限らずほとんどのタグで使用可能な属性として class 属性と id 属性がある。たとえば p は段落を意味するが、段落は段落でもさらに細かな意味―前書き・本文・日付など―を付けることができる。そのときに用いる属性が class 属性と id 属性である。

- class 属性は**その塊の意味を定義する**ための属性。
- id 属性は**文書の特定の場所を指示する**ための属性。

id 属性は HTML 文書内でユニークな値を持つ必要がある。

```
CSS で class、id に対して書式を設定する
.クラス名{属性: 値;}
#id 名{属性: 値;}
```

引用 (blockquote)

```
  <p>枕草子の冒頭の一文です。</p>
<blockquote>
  <p>春は、あけぼの。やうやう白くなりゆく、山ぎは少し明りて、紫だち
  たる雲のほそくたなびきたる。</p>
</blockquote>
  <p>中学校のときに暗記したりしましたね。</p>
```

blockquote はある程度まとまった内容を引用するときに用いる。**blockquote** 要素の中には段落 **p** などの要素を配置する必要がある。cite 属性には引用元のアドレスを指定する。

標準的なブラウザでは **blockquote** は左右にスペース（マージン）をとるが、マージンをとりたいがために引用ではない内容に **blockquote** を用いるのは不可。

地の文にマージンをつけたい場合は、CSS を用いる。

```
CSS でマージンをとる（上下なし、左右 3 文字分）
div{margin: 0 3em;}
```

作成者情報(address)

```
<address>
    Written by Natsumi Takahashi<br>
    E-mail: n_takahashi@ryukoku.ac.jp
</address>
```

HTML 文書の作成者に関しての基本的な情報（連絡先 address）を記述するのに使われるのが **address** である。作成者名とメールアドレスを記述するのが一般的。

ブラウザ上では斜体で表示されるのが一般的だが、CSS で設定を変更することができる。

```
CSS で斜体を指定（解除）する
address{font-style: italic(normal);}
```

リスト（箇条書き）

序列のないリスト(ul: Unordered List)

```
<h2>目玉焼きの材料</h2>
<ul>
    <li>新鮮なたまご</li>
    <li>サラダ油</li>
</ul>
```

序列のあるリスト(ol: Ordered List)

```
<h2>作り方</h2>
<ol>
    <li>熱したフライパンに油を入れる</li>
    <li>焼く</li>
</ol>
```

箇条書きを作成するときに用いるのがリストタグ。リストタグは ol、ul のいずれかを用い、その中に具体的なリスト項目 li(list item)を入れる。li 要素の中には他の要素を入れることもできる。

ol を用いた場合はリスト項目の先頭に連番が振られる。ul のばあいには傍点がつく。

定義リスト(dl: Definition List)

```
<dl>
    <dt>カレーライス</dt>
    <dd>カレー（ルー）とライス（ご飯）が別の皿に盛られたもの</dd>
    <dt>ライスカレー</dt>
    <dd>ご飯の上にカレー（ルー）をかけたもの</dd>
</dl>
```

定義リスト（Definition List）とは定義語(dt: Definition Term)とそれに対する説明(dd: Definition Description)とをペアでリストにするためのものである。

- dl は dt と dd
- dt はテキスト・そのほかの要素
- dd はテキスト・そのほかの要素

を持つことができる。

表 (table)

```
<table >
    <caption>夕食の献立</caption>
    <tr>
        <th>日付</th><th>月曜日</th><th>火曜日</th>
    </tr>
    <tr>
        <th>メニュー</th><td>炒り卵</td><td>カレーライス</td>
    </tr>
</table>
```

夕食の献立

日付	月曜日	火曜日
メニュー	炒り卵	カレーライス

table のなかには **tr**(Table Row:行)が必須で、**tr** のなかには **td**(Table Data:列に相当)が入りる。**td** のなかにはそのほかの要素を入れることができる。

表にタイトルをつけたいときは **caption** を、項目見出しをつけたいときには **td** の代わりに **th** を用いる。

属性

td・th 要素

rowspan=”結合したいセルの数” セルを縦方向に連結する。

colspan=”結合したいセルの数” セルを横方向に連結する。

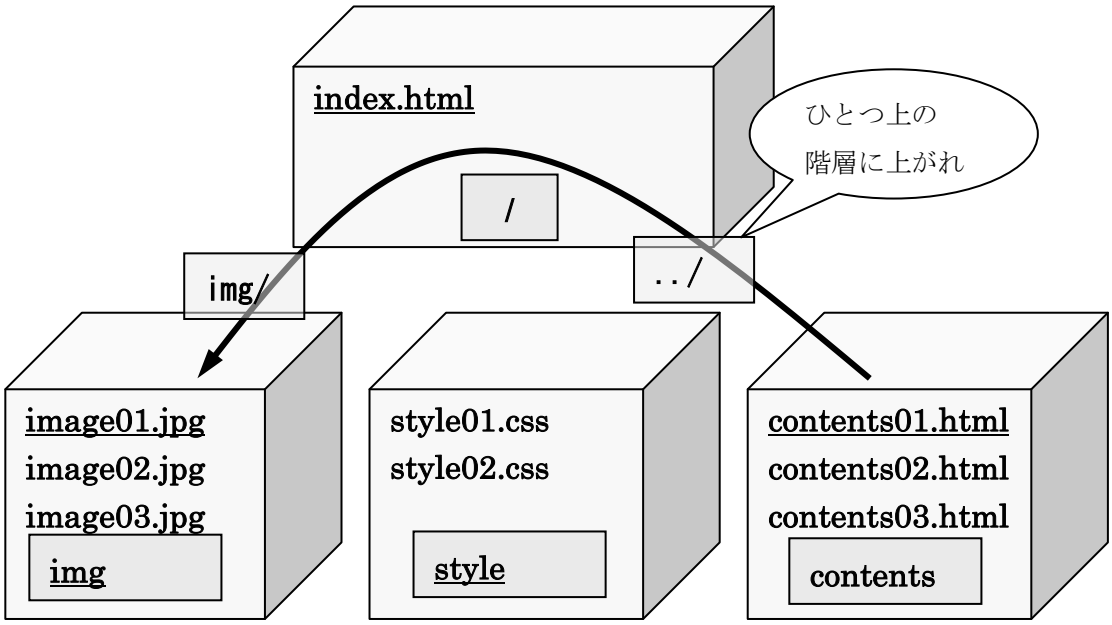
1.6. 他ファイルとの連携

HTML 文書は複数のファイルと連携して、全体を構成する。連携するファイルは HTML 文書、画像、音声、動画などさまざまである。いずれの場合においても、連携先のファイルの場所（URI: Uniform Resource Indicator）をしっかり把握しておく必要がある。

絶対 URI と相対 URI

絶対 URI とはインターネットのどの場所から参照してもひとつの場所を指し示す場所指定のことである。「**http://www.aited.jp/CITA/img/image01.jpg**」が絶対 URI の例。

他方相対 URI とは参照元の HTML 文書から見た参照先の文書の位置を示す場所指定。下図を参照。



「**index.html**」から見て、**image01.jpg** は「**img/image01.jpg**」となる。
「**contents01.html**」から見て、**image01.jpg** は「**../img/image01.jpg**」。

index.html から見た場合

"img/image01.jpg"　"style/style01.css"　"contents/contents01.html"

contents01.html から見た場合

"../img/image01.jpg"　"../style/style01.css"　"../index.html"

画像 (img)

「img」フォルダの中にある curry.jpeg という画像ファイルを HTML 文書に表示する。

```
<p>
名物カレーピラフだ！。<br>
<img src="img/curry.jpeg" alt="とてもおいしそうなカレーピラフの写真">
</p>
```

画像が表示できない環境の訪問者への配慮などの意味で画像自体がなくても、内容が把握できるようにする必要がある。**alt** 属性はそのために必須。画像がただの飾りである場合は alt 属性は空白(“”)にする。

- html5 で策定された video タグを使ってみよう。

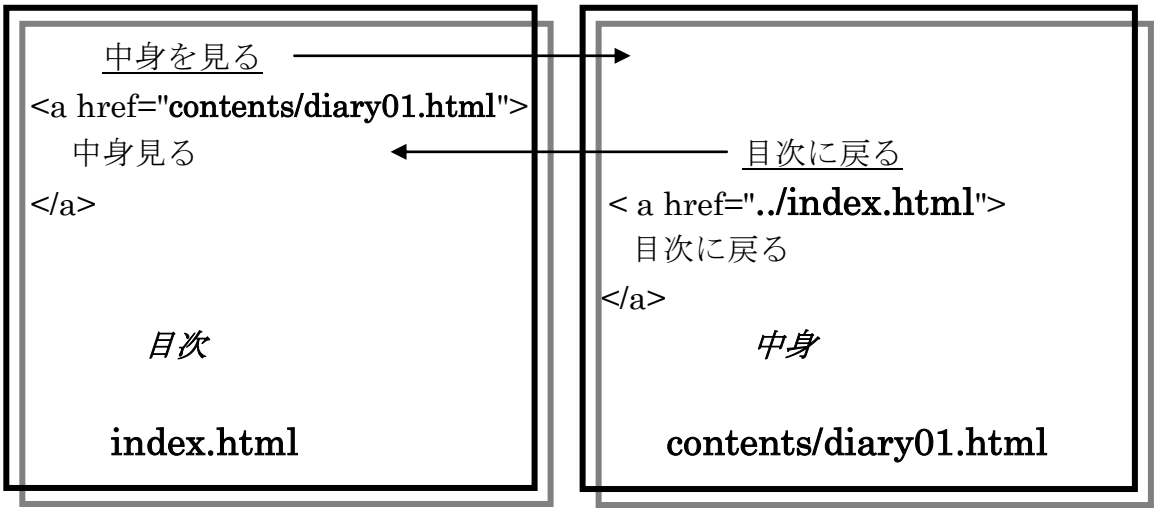
リンク

「contents」フォルダの中にある diary01.html、diary02.html にリンクを貼る。

```
<ol>
    <li><a href="contents/diary01.html">一月分の日記</a></li>
    <li><a href="contents/diary02.html">二月分の日記</a></li>
</ol>
```

リンクを貼るためにはリンクの始点となる「a: anchor」でリンクを張りたい文字列（画像）を囲み、href 属性でリンクの終点を URI（絶対・相対いずれも可）で指定する。

上の例では「一月分の日記」という文字列をマウスでクリックすると「contents」ディレクトリ内の diary01.html というファイルを参照することができる。



スタイルシートを共通で使う

「スタイルシート」はHTMLファイルの中に書くこともできるが、同じデザインを複数のページに指定したい場合は、**一つのスタイルシートを**外部の専用ファイル(*.css)に書き出し、**複数の HTML ファイルから、参照できる**ようにもできる。そうすると簡単に**サイト内のデザインを統一する**ことができる。

さらにページごとに細かな指定をしたいときは、HTMLファイル内に記述しておけばよい（スタイルシートの優先順位:ページ内スタイルシート→外部スタイルシート)。

index.html

```
<html lang=”ja”>
    <head>
        <title>ホームページのタイトル</title>
        <style>
            h1 {
                background : yellow ;
                color: red;
            }
        </style>
</head>
    <body>
```

index.html

```
<html lang=”ja”>
    <head>
        <title>ホームページのタイトル</title>
        <link rel="stylesheet"    href="style/style.css">
    </head>
    <body>
```

style.css

```
h1 {
    background : yellow ;
    color: red;
}
```

1.7.「正しい」html の話

由緒正しい（今の）HTML では、文字データ、表、絵（画像）などすべて「塊」として共通に扱う。単に文字を書きたい場合でも、**いちいちきちんと「塊」にしなくてはならない**。HTML で文書を作るというのは、文字列や絵や表や何やらをすべて「塊」でくくる、という作業に他ならない。そして面倒なことに、この「塊」には一つ一ついちいち「意味」「役割」をもたせ、それを明示しなくてはならない。

「ここの塊は見出しですよ」。「ここの塊は引用ですよ」。「ここの塊は段落ですよ」。などなど。

そういうことをいちいち意識しないと HTML では文章を書いてはならん！というのだ。はぁ、面倒くさい。書きたいことを書きたいように書きたいのであって、いちいちそんなこと、意識していられるか！！！

何故そんな面倒な話になっているのだろう？もちろんそれはそれなりに理由がある。それは、、、

Web ページというのは当然最初から電子データになっている。情報を単に人間が目で見るだけでなく、それ以外の手段でも処理できる、ということだ。

例えば、視覚障害を持った人のことを考えてみよう。一般的に単にだらだらと書き連ねられた文字列というのは、なかなか頭に入ってこないもので、読み手はそれなりに一群の文字列を「塊」にして把握しようとしている。ところが今まではアバウトな「見かけ」で「塊」を表現してきた。なんとなくこの文字は真中に持ってこよう。なんとなくこの辺は文字を大きくしてみよう。この辺で一行余分にあければ見やすいかな。

電子データというのはありがたいことに、容易に「読み上げ」を機械にやらせることができる。ところがこの場合、「見かけ」によって表現された「塊」をうまく伝達することはできない。そうなればただただ文字列をひたすら読み上げる、ということになってしまう。作り手は「塊」を表現しているつもりでも、それは一部の人にしか伝わらない。それは情報伝達の偏りを招く。

そうであるなら「見かけ」は後回しにして、「塊」化をもっと明示的にやったらいいでしょう、ということだ。**あるページの見出しがどの文字列であるのかは、文字の大きさや派手さで決まるのではなくて、「見出しだよ」という HTML 側の宣言で決まる**。これならどんな環境にあっても、みんな公平にその部分が「見出し」だということがわかる。後は目でそれを確認したい人はそのように「スタイルシート」で目立たせればよい。耳で聞きたい人は、機械に「この文章の見出しは、、、」というように（それもまた設定次第ですが）しゃべらせればよいのだ。

ある時期まで、見かけだけきらびやかな、でも**読み手によってはとても扱いにくい** Web ページが作りつづけられてきた。そうした状況に対する反省として、今の HTML の仕様が策定されたのだ。それは情報への平等なアクセスを標榜していたインターネット元来の精神を取り戻すことでもある。新しい今の HTML の流儀を勉強することはその精神を学ぶことでもあるのだ。