なかしまぁ先生のHTML5教室

最初に

講師紹介

- 中島俊治。別名「なかしまぁ先生」
- 岡山県出身。「SoftBank」や「GeoCities　Japan」「Yahoo!　JAPAN」などのネットベンチャの後に独立。現在は「サイバー大学　IT総合学部」で准教授。HTML5を教えている。
- 一般社団法人　Web技術振興協会の代表理事として、HTML5の試験を実施と、初心者向け講座（まるごとどっぷりHTML5教室）を開催。
- マイクロソフトより「Microsoft MVP (Developer Technologiesカテゴリ)」受賞。

HTML5について

HTML5はモダンなWeb技術の総称です。HTMLの他、デザインのCSS、動きのJavaScript等を含みます。

単なるWebサイトから、より豊かなデザイン、動きのあるWebアプリケーションへ。そんなHTML5をぜひ楽しんでください。

本講座について

本講座は2021年11月現在「１：HTML編」「２：CSS編」「３：JavaScript編」の３部に分かれています。最初から受講されることをおすすめします。

2：CSS編

1時限目：CSS

CSSは「Cascading Style Sheets」の略です。

・Style は「見た目」

・Sheets は「場所」

. 授業では単純に「要素のスタイルを記述した場所」とする

---

1. CSSの書き方

例えば次のp要素について

```
<p>今日はいい天気です</p>
```

「p要素の中のテキストを青色」にしたければ、CSSでは、

```
<style>
  p{color:blue;}
</style>
```

となります。

. 「〜の」をセレクタ、「〜は」をプロパティ、「〜です」を値と呼ぶ

. CSSは「セレクタの{プロパティは:値です;}」と覚えればよい

. コロン「:」はイコールの意味。セミコロン「;」は文の区切りの意味。「:」と「;」を間違えないように

. 「セレクタ」は４種類ある

. 「プロパティ」は数多くあり日々増加している

. 「値」は「100px」や「50%」などの単位のついた数値、「blue」や「red」などのキーワード、たまに文字列などがある

---

2. CSSの配置

CSSを配置する方法には3種類あり、それぞれ特徴があります。

1.style要素を使う

head要素内のstyle要素内にCSSを書込む方法

```
<style>
  p{color:blue;}
</style>
```

. そのページの中にあるp要素の中のコンテンツはすべて青色になる

. ファイルが一つで編集しやすいので、授業ではこの方法で学んでいく

2.外部スタイルシートを使う

最初に、htmlファイルとは別に次のような外部ファイル(style.css)を用意。拡張子は.css。styleタグの記述は不要

```
p{color:blue;}
```

次に、htmlファイルのhead要素内に次のコードを書込む

```
<link rel="stylesheet" href="style.css">
```

"style.css" を "stylesheet" として定義する（=読み込む）

これだと、何万ページあっても、スタイリングを変更したいときはcssファイル1枚のCSSを書換えればいい。多くのページを持った一般的なWebサイトはこの設置の方法が便利です。

3.style属性を使う

style属性を用いてスタイリングすることも可能です。ただし、ピンポイントでの指定しかできませんので、非効率。HTML5では非推奨です。

```
<p style="color:blue;">今日はいい天気です</p>
```

---

3. 配置の使い分け

普段は「2.外部スタイルシート」で構いません。Webアプリケーションを作ろうとすると「1.style要素」のほうが便利な場合がありますので、適宜使い分けたり組み合わせて設置することを推奨します。

2時限目：セレクタ

セレクタには４種類あります。

1．要素セレクタ

```
<style>
  p{color:green;}
</style>
```

CSSが有効な範囲内で、要素の各々全てをスタイリングするためのセレクタ。セレクタに要素名(ここではp要素)をそのまま書く

2．classセレクタ

同じ要素でも、スタイリングを変えることもあり、異なる要素でもスタイリングを同じにしたいことがある

その場合は、「classセレクタ」で指定。classは「共通の性質・性格＝スタイリング」という意味

HTMLの要素にclass属性でクラス名を設定し、CSSではクラス名の前にclassであることを示す「.(ドット)」を付ける

```
<style>
  p{color:green;}
  .css1{color:blue;}
</style>
⋮
<h1>見出し1</h1>
<p>本文1</p>
<p class="css1">本文2</p>
<p class="css1">本文3</p>
```

3．idセレクタ

JavaScriptでは、要素を操作するための名前(識別子=id）を用いる。CSSではそのidを利用してセレクタにすることが可能

HTMLの要素にid属性でid名を設定し、CSSではid名の前にidであることを示す「#(シャープ)」を付加

JavaScriptで扱うので、そのidはページ内でひとつでなければならないことに注意

```
<style>
  p{color:green;}
  .css1{color:blue;}
  #id1{color:red;}
</style>
⋮
<h1>見出し1</h1>
<p>本文1</p>
<p class="css1" id="id1">本文2</p>
<p class="css1">本文3</p>
```

4. セレクタの強さ

上のサンプルでは、本文2に、同時に「要素セレクタ(文字色は緑)」「classセレクタ(文字色はblue)」と「idセレクタ(文字色はred)」を指定しているが、赤色になった。セレクタには次のように優先順位がある

要素セレクタ ＜ classセレクタ ＜ idセレクタ

5. ユニバーサル(全称)セレクタ

ユニバーサルというのは全てに共通するという意味で、全称セレクタとも言う。セレクタは「*」

```
<style>
   *{color:yellow;}
   p{color:green;}
   .css1{color:blue;}
   #id1{color:red;}
</style>
```

ユニバーサルセレクタはid・class・要素セレクタよりも弱く、一番弱いセレクタ

例えば、HTMLのページ。文字の色やh1要素の大きさ、リンクの下線、文字の行間や余白、ページの背景色はブラウザ自身の持っているCSSによっている

例えばブラウザが勝手に隙間があけていたら困るので、ブラウザの持っているCSSをリセットするために用いるとよい。例えばCSSの最初に以下のCSSを書いたとします

```
<style>
   *{
      margin:0;
      padding:0;
      font-size:24px;
      text-decoration:none;
   }
 ⋮
</style>
```

. marginは要素の外の余白、paddingは要素の内側の余白、font-sizeは文字の大きさ、text-decorationは下線などの装飾

3時限目：結合子・グループ化

セレクタは４種類ですが、それを組み合わせると、複雑な指定も可能。組み合わせるために、結合子を用います。

1. 子孫セレクタ

親要素内にある全てのセレクタ。階層に関係ない。記号は半角空白。「A B{〜:〜;}」

```
<style>
   body *{border:1px solid gray;margin:3px;}
   #box1 p{background-color:red;}
</style>
⋮
<div id="box1">

   <p>#box1の子要素p</p>

   <div>
      <p>ここは２階層の孫要素p</p>
   </div>
</div>
```

2. 子セレクタ

親要素内の一階層下の子要素を示す。記号は「>」。「A>B{〜:〜;}」

```
<style>
   body *{border:1px solid gray;margin:3px;}
   #box1>p{background-color:red;}
   #box1>*>p{color:blue;}
</style>
⋮
<div id="box1">
   <p>#box1の子要素p</p>
   <div>
      <p>ここは２階層の孫要素p</p>
   </div>
</div>
```

もし２階層下を指定したければ、「A>*>B」と指定

3. 隣接セレクタ

子要素のすぐ隣のセレクタを指定。記号は「+」。「A+B{〜:〜;}」

```
<style>
   body *{border:1px solid gray;margin:3px;}
   h1+p{color:red;}
</style>
⋮
```

```
<h1>隣接セレクタ</h1>
<p>隣接しています</p>
<p>隣接していません</p>
```

---

4. 間接セレクタ

同一階層の、すべての要素を指定。記号は「~」。「A~B{〜:〜;}」

```
<style>
   *{border:1px solid gray;}
   h1~p{color:red;}
</style>
⋮
<h1>間接セレクタ</h1>
<p>隣接・間接しています</p>
<p>隣接していませんが間接しています</p>
```

---

5. グループ化

複数のセレクタにCSSを適用させる場合、個々に書くのは効率が悪いので、セレクタをグループ化することができる

```
<style>
   #wrap,h1,p{
      border:1px solid gray;
      margin:3px;
      color:red;
   }
</style>
⋮
<div id="wrap">
   <h1>グループ化</h1>
   <p>CSSのセレクタはグループ化することができます</p>
</div>
```

セレクタをカンマ「,」でつなげて記述

4時限目：擬似クラス

HTML5ではコンテンツを操作するのでコンテンツが増えたり減ったりします。例えば最初の要素にclassを指定してその要素がなくなったら困りますね。HTMLソースにclassを書込まないで、要素を指定する方法が必要です。

1. :first-of-type　:last-of-type

セレクタに対する最初、最後の要素を指定

```
<style>
   li:first-of-type{color: red;}
   li:last-of-type{color: blue;}
</style>
⋮
<ol>
   <li>値1</li>
   <li>値2</li>
   <li>値3</li>
   <li>値4</li>
   <li>値5</li>
</ol>
```

- :first-of-type、:last-of-type　は指定したセレクタに対して、それぞれ最初の要素、最後の要素を選ぶ

2. :nth-of-type(n)　:nth-last-of-type(n)

セレクタに対して、指定した番目の要素を指定

```
<style>
   li:nth-of-type(2){color: red;}
   li:nth-last-of-type(3){color: blue;}
</style>
⋮
<ol>
   <li>値1</li>
   <li>値2</li>
   <li>値3</li>
   <li>値4</li>
   <li>値5</li>
</ol>
```

- nth-of-type(n)、:nth-last-of-type(n)は、指定したセレクタに対して、最初から何番目、最後から数えて何番目を選ぶ
- nth-of-type(3n)　とすると3の倍数。nth-of-type(3n+1)　とすると3の倍数に1を加えた番目の要素を選ぶ

3. :nth-of-type(odd)　:nth-of-type(even)

指定したセレクタに対して、奇数・偶数で要素を指定

```
<style>
   li:nth-of-type(odd){color: red;}
   li:nth-last-of-type(even){color: blue;}
</style>
⋮
<ol>
   <li>値1</li>
   <li>値2</li>
   <li>値3</li>
   <li>値4</li>
   <li>値5</li>
</ol>
```

・:nth-of-type(odd)、:nth-of-type(even)は、指定したセレクタに対して、奇数、偶数を選ぶ

5時限目：属性セレクタ

属性セレクタは、要素の属性の値によってCSSを指定することができます。

1. 要素[属性]

設定した属性を持つ要素に対してスタイルをつける

```
<style>
  p[class]{color:red;}
</style>
⋮
<p class="css1">文章１</p>
<p>文章２</p>
```

2. 要素[属性="属性値"]

設定した属性の値が完全一致する要素に対してスタイルをつける

```
<style>
  input[type="text"]{color:red;}
</style>
⋮
<p>入力１：<input type="text"></p>
<p>入力２：<input type="tel"></p>
```

3. 属性値のパターンマッチ

属性値のパターンがマッチする要素に対してスタイルをつける

```
<style>
  div[class~="css1"]{font-size:30px;}
  div[class^="c"]{background-color:green;}
  div[class$="2"]{color:blue;}
  div[class*="a"]{font-weight:bold;}
</style>
⋮
<div class="css1">文章１</div>
<div class="css2">文章２</div>
<div class="css1 css3">文章３</div>
<div class="aaa">文章４</div>
```

- [class~="css1"]は、class属性の値が複数あるうちでcss1を含んでいる要素です
- [class^="c"]は、class属性の値がcで始まる要素です
- [class$="2"]は、class属性の値が2で終わる要素です
- [class*="a"]は、class属性の値にaが含まれている要素です

6時限目：CSSプロパティ

CSSのプロパティはとても多いので、まずは基本的なものを紹介します。

1. border プロパティ

div要素に枠線を付ける

```
<style>
   div{
      border-width:1px;
      border-style:solid;
      border-color:green;
   }
</style>
⋮
<div>テキスト1</div>
<div>テキスト2</div>
<div>テキスト3</div>
```

- border-widthが「枠線の太さ」のプロパティ
- border-styleが「枠線のスタイル」のプロパティ
- border-colorが「枠線の色」のプロパティ

次のようにまとめることができる。

```
<style>
   div{
      border:1px solid green;
   }
</style>
```

- これを「一括指定プロパティ（shorthand property）」と言う。短縮して書けるのでとても便利なので、この形のプロパティを覚えるとよい

ただし次のようなことが起こるので注意が必要。

```
<style>
   div{
      border-color:red;
      border:1px solid;
   }
</style>
```

黒色になる。

- 一括指定プロパティでは、値を省略するとその値は初期化(=黒)されるので注意が必要

2. コンテンツをまとめる要素は「重箱」

HTML編の第５時限でててきたコンテンツをまとめる要素（div要素、header、nav、ol、li、main、section、h1、h2、p、footer）などに枠線を設定すると、すべて横に伸び、縦は中のコンテンツの高さになる。

3. コンテンツの要素は「おかず」

HTML編の６時限でててきたコンテンツの要素は横に伸びないで縮こまる

```
<style>
   span,a,em{
      border:1px solid green;
   }
</style>
⋮
<p>きょうは<span>くもりの天気</span>でした。明日は<span>晴れのいい天気</span>
になるでしょう。</p>
<p>私のWebページは<a href="https://example.html">こちらのページ</a>です。
</p>
<p>きのうは<em>暑かった！</em></p>
```

4. 文字のプロパティ

次のようなHTMLのコンテンツにCSSを適用させる

```
<p>こんにちは。</p>
```

color

文字の色を指定

```
p{color:gray;}
```

- 色はキーワード（red、green、blue）等の他、カラーコード（例えば「#ff0000」は赤）を使用できる

font-size

文字の大きさを指定

```
p{font-size:16px;}
```

- 文字の大きさは16pxが標準。その他一文字分の文字の大きさの単位「em」などがある

font-weight

文字の太さを指定

```
p{font-weight:bold;}
```

- boldは太字。数値でも指定できるがフォントが対応していないことが多いので、実質boldしかない

text-decoration

文字の装飾を指定

```
p{text-decoration:none;}
```

- 「underline」は下線、「overline」は上線、「line-through」は取り消し線
- 例えばa要素では自動的に下線がついているので、それを無効にするには値は「none」

line-height

行高を指定

```
p{line-height:20px;}
```

．１行目と２行目の間が空くので文章が見やすくなる。段落と段落の間はmarginを使う

7時限目：ボックスモデル

枠線をスタイルしましょう。

1. ページは「四角形」でできている

要素の形は四角形。私達は四角形をデザインできればページのデザインができる。

2. widthとheight

width はコンテンツの横幅 のプロパティ、height はコンテンツの縦幅 のプロパティ

```
  <style>
     *{margin:0px; padding:0px;}

     div{
        border:1px solid blue;
        width:200px;
        height:200px;
     }
  </style>
  ⋮
  <div>ボックス1</div>
```

3. padding

padding プロパティは枠線とコンテンツとの間の(内側)余白

```
     div{
        ⋮
        padding:10px;
     }
```

ちなみに次のように値を4つにすると上から時計回りの内側余白になります

```
     div{
        ⋮
        padding:10px 10px 10px 10px;
     }
```

4. margin

marginプロパティは外余白（枠線の外側と外部の要素などとの隙間）

```
     div{
        ⋮
        margin:100px;
     }
```

値を4つにすると上から時計回りの外余白

```
     div{
```

```
        ⋮
        margin:100px 100px 100px 100px;
    }
```

次のようにすると左右の真ん中揃えになる(値が2つの場合は上下・左右の値を指定したことになる。auto：自動調節)

```
    div{
        ⋮
        margin:100px auto;
    }
```

---

5. marginの相殺

marginは相殺されることがある。上下の箱の間に注意

```
<div>ボックス1</div>
<div>ボックス2</div>
```

8時限目：Flexible box

要素を横並べするのが「Flexible box」です。

1．Flexible box

グローバルナビゲーションをサンプルにする

```
<nav>
   <ul>
      <li class="css1"><a href="page1.html">ページ1</a></li>
      <li class="css2"><a href="page2.html">ページ2</a></li>
      <li class="css3"><a href="page3.html">ページ3</a></li>
   </ul>
</nav>
```

・li要素を横並べにすることが目標

・li要素はul要素でグループ化。この場合、ul要素は「親要素」li要素は「子要素」と言う

親要素（ul要素）に次のように記述

```
ul{
   border:3px solid red;
   height:100px;
   display:flex;
}
```

・displayは、表示形式のプロパティ

・値のflexは一般に柔軟にという意味ですが、ここでは「横並べ」とする

隙間をなくすために子要素に次のように追加

```
li{
   border:3px solid green;
   flex:1;
}
```

・flexプロパティは隙間なく埋める「重み」

・以下のCSSを追加することで整えることができる

```
li{
   border:2px solid green;
   flex:1;
   text-align:center;
   list-style:none;
}
```

flexを子要素に移動しそれぞれの重みを変えることも可能

```
li.css1{flex:2;}
li.css2{flex:1;}
li.css3{flex:1;}
```

9時限目：背景のスタイル

画像を配置するにはimg要素で画像として表示する方法と、CSSで背景画像として表示する方法があります。ここでは背景のスタイルについて学びましょう。

1. background

次のHTMLを用意

```
<style>
   #box1{
      border:15px dashed gray;
      width:500px;height:300px;
      margin:100px auto;
   }
</style>

<div id="box1">
   <p>box1</p>
</div>
```

では、#box1に次のCSSを順番に指定

```
<style>
   #box1{
      ⋮
      background-color:blue;
      background-clip:content-box;

      background-image:url(image1.png);
      background-origin:content-box;

      background-repeat:no-repeat;
      background-size:cover;
      background-position:top left;
      background-attachment:fixed;
   }
</style>
```

以上で準備が完了

2. background-color

背景色を指定。背景色には

・red、greenなどのキーワード

・#ff00ff などの16進数値

・rgb(100, 100, 100)、rgba(100, 100, 100,0.5) などのRGB値

・hsl(50, 33%, 25%)、hsla(50, 33%, 25%, 0.75)などのHSL値

3. background-clip

背景を境界のどこまで拡張するかを指定

・border-box：境界の外側の辺まで

・padding-box：paddingの外辺まで

・content-box：コンテンツボックスの中のみ

4. background-image

背景画像を指定。値はurl("画像ファイル")

背景画像では値を複数持つことが可能。次の背景は、image1の前にimage2が表示。image2は透過画像でなければならない

```
background-image:
   url("image2.png"),
   url("image1.png");
```

5. background-origin

背景画像の基準位置を指定。background-clipと同様、境界のどこから始めるのかを指定

・border-box：borderの外側境界まで

・padding-box：paddingの内側

・content-box：コンテンツの範囲のみ

6. background-repeat

背景画像の繰り返しを指定

・repeat-x：横方向に繰返し

・repeat-y：縦方向に繰返し

・repeat：縦横に繰返し

・no-repeat：繰返さない

7. background-size

背景画像のサイズを指定

・横のpx値(又は%) と 縦のpx値(又は%)の組み合わせ

・contains：縦横比はそのままで領域内に収まる最大サイズ。ただし空白が出る可能性がある

・cover：縦横比はそのままで領域全体を隙間がないようにカバー

8. background-position

背景画像の基準点を指定。例えば拡大縮小時にピンを打つ際のピンの位置です。

・キーワード：top,left,bottom,right,center とその組み合わせ（「top right」等）

・横のpx値(又は%) と 縦のpx値(又は%)の組み合わせ

---

9. background-attachment

スクロール時の背景の移動の有無を指定

・scrollは移動

・fixedは固定

10.backgroundのショートハンド

一括して表示することも可能

```
background:
   border-box border-box
   gray
   url("image1.png")
   repeat-x
   center/cover
   scroll;
```

. background-positionとbackground-sizeの値は「/」を使って組み合わせる

. background-origin と background-clip の値は、まとめて1つだけ書けば両方を指定します。2回書くと background-origin 、 background-clipの順番に指定します

10時限目：影のスタイル

要素に影を付けてみましょう。

次のHTML、CSSを用意

```
<style>
   div{
      border:1px solid gray;
      width:200px;height:200px;
      margin:100px auto;
      background-image:url("bgimg.png");
   }
   .css1{

   }
</style>
⋮
<div class="css1"></div>
```

1．box-shadow

影のプロパティは次の通り

```
   .css1{
      box-shadow:3px 3px 3px gray;
   }
```

．値は最初から、横方向の影のズレ具合　縦方向の影のズレ具合　ぼかし具合　影の色　になる

．横と縦のズレ具合はマイナス値も可能

．写真とか、ボタンとかで目立たせたり、section要素を少し浮き立たせるのにいい

ぼかし具合の後に値を挿入すると、影の大きさを変えることも可能

```
   .css1{
      box-shadow:3px 3px 3px 5px gray;
   }
```

そして、「inset」で内側に影を表示することが可能

```
   .css1{
      box-shadow:inset 3px 3px 3px 5px gray;
   }
```

さらに影を複数指定することも可能。カンマで区切って値を指定

```
   .css1{
      box-shadow:
         3px 3px 3px 5px pink,
         -3px -3px 3px 5px skyblue;
   }
```

2. filter:drop-shadow()

透過画像に影をつける

```
.css1{
   filter:drop-shadow(3px 3px 3px gray);
}
```

. filterは画像を調整するプロパティ

. drop-shadowは影を落とす値。カッコ内の値はbox-shadowと同じ

3. text-shadow

要素内のテキストに影をつける

```
.css1{
   text-shadow:3px 3px 3px gray;
}
```

. text-shadowは文字の影のプロパティ

. 値は横のズレ具合、縦のズレ具合、ぼかし具合に色の順番

11時限目：filter

filterは画像を調整するプロパティです。

1．filterプロパティ

filterプロパティの値には次のようなものがある。値を半角あけて組み合わせることも可能

明るさ

```
filter: brightness(200%);
```

コントラスト

```
filter: contrast(200%);
```

グレースケール

```
filter: grayscale(80%);
```

彩度

```
filter: saturate(30%);
```

セピア

```
filter: sepia(60%);
```

色相回転

```
filter: hue-rotate(90deg);
```

階調反転

```
filter: invert(75%);
```

ぼかし

```
filter: blur(5px);
```

透明度

```
filter: opacity(75%);
```

12時限目：角を丸める

角を丸めるプロパティです。

まず、ボックスを一つ作っておく

```
<style>
   div{
      border:1px solid gray;
      width:200px;height:200px;
      margin:100px auto;
      background-image:url("bgimg.png");
   }
   .css1{

   }
</style>
⋮
<div class="css1"></div>
```

1. border-radius

border-radiusはカドを丸めるプロパティ。radiusは角を丸めたときの半径

```
   .css1{
      border-radius:10px;
   }
```

４つの角を別々に指定

```
   .css1{
      border-radius:10px 10px 10px 10px;
   }
```

. 左上のカドから時計回りに指定

この四角形をまんまるにする

```
   .css1{
      border-radius:50%;
   }
```

13時限目：グラデーション

グラデーションのプロパティです。

以下のファイルを準備

```
<style>
   div{
      border:1px solid gray;
      width:200px;height:200px;
      margin:100px auto;
   }
   .css1{

   }
</style>
⋮
<div class="css1"></div>
```

1. linear-gradient

最初は線形のグラデーション。線形(linear)とは徐々にとか段々にいう意味

```
   .css1{
      background:linear-gradient(white,gray);
   }
```

次にグラデーションの範囲を変える

```
   .css1{
      background:linear-gradient(white 30%,gray 70%);
   }
```

. パーセントは、whiteを0%、grayを100%とした位置になる

. このコードではwhiteが0%から30%まで占め、grayは100%から70%まで占め、あいだの30%から70%でのグラデーション

グラデーションの角度を変える

```
   .css1{
      background:linear-gradient(90deg,white,gray);
   }
```

. degをつけると180度ずれて指定される

2. radial-gradient

radial-gradientは放射状のグラデーション

```
   .css1{
      background:radial-gradient(white,gray);
   }
```

範囲を変える

```
.css1{
   background:radial-gradient(white 30%,gray 70%);
}
```

中心を変えてみましょう

```
.css1{
   background:
      radial-gradient(circle at top left,white,gray);
}
```

・at が位置を指定。横と縦の位置をtop、right、bottom、left、center、px、%で指定

circleは形状。これがないと楕円(ellipse)になる

グラデーションの大きさを変えてみましょう

```
.css1{
   background:radial-gradient(
      closest-side circle at 30% 30%,white,gray
   );
}
```

・closest-sideは、グラデーションの中心から終点の大きさを決めます

・大きさには次の4つがあります

・closest-side：中心から最も近い辺まで

・closest-corner：中心から最も近い頂点まで

・farthest-side：中心から最も遠い辺まで

・farthest-corner：中心から最も遠い頂点まで

14時限目：変形（回転・伸縮・移動・傾き）

要素を回転・伸縮・移動・傾かせることを変形と呼びます。変形は、トランジション・アニメーションで必要になる重要な技術です。

. 以下のファイルを準備

```
<style>
   div{
      border:1px solid gray;
      width:200px;height:200px;
      margin:100px auto;
      background-image:url("bgimg.png");
   }
   .css1{

   }
</style>
⋮
<div class="css1"></div>
```

1. transform

transformは変形のプロパティです。値には二次元の場合には次の4つの値を持ちます。

・回転：rotate()

・伸縮：scale()

・移動：translate()

・傾き：skew()

2. 回転　rotate()

回転の値です。丸括弧内に角度を入れます。

```
   .css1{
      transform:rotate(20deg);
   }
```

3. 伸縮　scale()

伸縮の値です。丸括弧内には倍率を入れますが、単位はありません。

```
   .css1{
      transform:scale(2,1);
   }
```

値を scale(0,0) に変えるとなくなる

値を scale(-1,1) にすると反転する

4. 移動　translate()

移動の値です。

```
.css1{
   transform:translate(100px,50px);
}
```

. 丸括弧内には横方向、縦方向の移動の長さ。マイナスも可能

5. 傾き　skew()

傾きの値です。

```
.css1{
   transform:skew(40deg,0deg);
}
```

. 丸括弧内には横方向の傾き角度、縦方向の傾き角度を入れる

15時限目：トランジション

時間をかけてスタイルを変化させるには２つの方法があり、今回はその中のトランジションを学ぶ

次の要素を準備

```
<style>
   div{
      border:1px solid gray;
      width:100px;height:100px;
   }
   .css1{

   }
</style>
⋮
<div class="css1">ボックス１</div>
```

1. transition

transitionは時間の変化のプロパティです。

- マウスのポインタが要素の上に乗ったときに、テキストの色を変えたいとするならば擬似クラス(:hover)を用いる
- :hoverの他にも:activeや、JavaScriptできっかけを作ります

```
   .css1{transition:color 1s linear 0s;}
   .css1:hover{color:green;}
```

- transitionプロパティの値は、
  - 対象となるプロパティ：ここでは「color」を指定
  - 変化する時間：「1s」は1秒間のこと
  - 変化のスタイル：「linear」は等速。「ease-in-out」に変えれば自然な状態でゆっくり始まりゆっくり終わります
  - 遅延の時間：きっかけの後、何秒後に始まるかを指定。ここでは0秒

背景色をトランジションしましょう。

```
   .css{
      transition:
         color 1s linear 0s,
         background-color 1s linear 0s;
   }
   .css1:hover{
      color:green;
      background-color:gray;
   }
```

- transitionプロパティにカンマで区切って値が２つ入ることに注意

2. 変形をトランジション

変形をトランジションする。（対象プロパティはtransform）

```
.css1{
   transition:transform 1s linear 0s;
}
.css1:hover{
   transform:rotate(360deg);
}
```

. 変形の値(rotate()、scale()、translate()、skew())も試してみると良い

3. 変形を合成

変形を組み合わてみましょう。回転しながら、伸縮など。

```
.css1{
   transition:transform 1s linear 0s;
}
.css1:hover{
   transform:rotate(360deg) scale(2,2);
}
```

値の位置・順番によって変化の様子が違うので注意しましょう。

```
.css1{
   transition:transform 1s linear 0s;
}
.css1:hover{
   transform:
      translate(100px,0px) rotate(360deg) scale(2,2);
}
```

16時限目：アニメーション

アニメーションはトランジションと同じで要素を時間をかけて変化させるプロパティです。

. ボックスを一つ作っておく

```
<style>
   div{
      border:1px solid gray;
      width:200px;height:200px;
      margin:100px auto;
      background-image:url("bgimg.png");
   }
   .css1{

   }
</style>
⋮
<div class="css1"></div>
```

1. animation

アニメーションのプロパティです。

```
   .css1{
      animation: mystory 3s linear 0s infinite normal;
   }
```

animationプロパティの値は、

. アニメーションの名前：名前は任意です。ここではmystoryとしました

. アニメーションの時間：ここでは3s（3秒）

. アニメーションのスタイル：linearは等速。ease-in-outとするとゆっくり始まりゆっくり終わる

. アニメーションの開始時間：ページを開いてから何秒後に動作するかです。ここでは、0秒としています

. アニメーションの繰返し回数：infiniteは永久に繰り返します。1や2の数字にするとその回数だけ繰返しします

. アニメーションの方向：normalは一方通行です。alternateとすると、巻き戻しを行います

2. keyframes

アニメーションの台本です。

要所要所の変化点をkeyframeという

. 「mystory」はアニメーションの名前

. 0%がアニメーションの最初、100%がアニメーションの最後の変化点

. 複数のフレームなので「keyframes」。keyframesの前には「@」を付ける

. @は「アットマークルール」という。規則を定義するCSSの書式

```
@keyframes mystory{
   0%{transform:rotate(0deg);}
   100%{transform:rotate(360deg);}
}
```

移動させましょう。

```
@keyframes mystory{
   0%{transform:translate(-300px,0px);opacity:0;}
   40%{transform:translate(0px,0px);opacity:1;}
   60%{transform:translate(0px,0px);opacity:1;}
   100%{transform:translate(300px,0px);opacity:0;}
}
```

- opacityプロパティは透過。左から現れて真ん中で止まり、右に消えていく

伸縮させましょう。

```
@keyframes mystory{
   0%{transform:scale(1,1);}
   50%{transform:scale(1.4,1.4);}
   100%{transform:scale(1,1);}
}
```

---

3. animation-fill-mode

次のCSSを実行してみましょう。

```
.css1{
   animation: mystory 3s linear 0.5s 1 normal;
}

@keyframes mystory{
   0%{transform:rotate(30deg);}
   100%{transform:rotate(330deg);}
}
```

- このアニメーションは、繰返しが1回。0.5秒後に始まり、3秒経過後終了。アニメーションの最初が30度、終わりが330度と傾いたまま
- 一方、アニメーションの始まる前と終わった後は、アニメーションの時間ではないので、角度は0度のまま。なのでつながっていないように見える

それを解決するのが animation-fill-mode です。

```
.css1{
   animation: mystory 3s linear 0.5s 1 normal;
   animation-fill-mode:both;
}
```

- 始まる前も終わった後も両方ともなので、値は「both」

17時限目：三次元

CSSは三次元が可能です。

. まず、次のソースを準備

```
<style>
   .parent1{
      border:1px solid green;
      width:200px;height:200px;
      margin:100px auto;
   }
   .child1{
      border:1px solid red;
      width:200px;height:200px;
      background-image:url("bgimg.png");
      margin:0 auto;
   }
</style>
⋮
<div class="parent1">
   <div class="child1"></div>
</div>
```

子要素を三次元表示にするため、２つのことをします。

・親要素を三次元化

・子要素を三次元で変形

---

1. 親要素を三次元化

親要素に次のCSSを記述します。

```
   .parent1{
      transform-style:preserve-3d;
      perspective:105px;
   }
```

. transform-styleは次元を設定するプロパティ

. 値をpreserve-3d(=三次元を保存)とすれば三次元（二次元はflat）

. perspectiveは奥行きの深さ

---

2. 子要素を三次元で変形

子要素を三次元で変形します。

. 三次元の変形もtransformを使う。ただし、三次元はX軸、Y軸、Z軸があるので、値は以下のようになる

回転(単位はdeg)

・**rotateX**(x軸の回転角度)

・**rotateY**(y軸の回転角度)

・**rotateZ**(z軸の回転角度)

・**rotate3d**(x軸の回転角度，y軸の回転角度，z軸の回転角度)

伸縮(単位はない)

・**scaleX**(x方向の縮尺比率)

・**scaleY**(y方向の縮尺比率)

・**scaleZ**(z方向の縮尺比率)

・**scale3d**(x方向の縮尺比率，y方向の縮尺比率，z方向の縮尺比率)

移動(単位はpx)

・**translateX**(x方向の距離)

・**translateY**(y方向の距離)

・**translateZ**(z方向の距離)

・**translate3d**(x方向の距離，y方向の距離，z方向の距離)

例えば、Y軸を中心に回転したければ

```
.child1{
   transform:rotateY(40deg);
}
```

組み合わせも可能です。

```
.child1{
   transform:rotateY(40deg) rotateX(20deg);
}
```

18時限目：三次元アニメーション

「三次元」「アニメーション」を組み合わせてみましょう。

. 次のソースを準備

```
<style>
   .parent1{
      border:1px solid green;
      width:200px;height:200px;
      margin:100px auto;
   }
   .child1{
      border:1px solid red;
      width:200px;height:200px;
      background-image:url("bgimg.png");
      margin:0 auto;
   }

   /*アニメーションのCSS*/
   .parent1{
      transform-style:preserve-3d;
      perspective:105px;
   }

   .child1{
      transform:rotateY(40deg);
   }
</style>
⋮
<div class="parent1">
   <div class="child1"></div>
</div>
```

1. 子要素にanimationプロパティ

子要素のtransformプロパティをanimationプロパティに差し替えてみよう。

```
   .child1{
      animation: mystory 3s linear 0s infinite normal;
      animation-fill-mode:both;
   }
```

mystoryを記述して回転させてみよう。

```
   @keyframes mystory{
      0%{transform:rotateY(0deg);}
      100%{transform:rotateY(360deg);}
   }
```

移動させてみよう。

```
    @keyframes mystory{
       0%{
          transform:rotateY(30deg) translateX(-500px);
          opacity:0;
       }

       30%,70%{opacity:1;}

       100%{
          transform:rotateY(30deg) translateX(500px);
          opacity:0;
       }
    }
```

伸縮させてみよう。

```
    @keyframes mystory{
       0%{transform:rotateY(30deg) scaleX(0.5) scaleY(1);}
       50%{transform:rotateY(30deg) scaleX(1) scaleY(0.5);}
       100%{transform:rotateY(30deg) scaleX(0.5) scaleY(1);}
    }
```

19時限目：メディアクエリ

レスポンシブWebデザインというのはPCでもスマホでもそれぞれに適したページを表示するデザインです。その中でももっとも大事なのがメディアクエリです。

. コンテンツの横幅が800pxのページを準備

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset=utf-8>
      <title>メディアクエリ</title>
      <meta name="viewport" content="width=device-width">
      <style>
         #wrap{
            width:800px;
            margin:0 auto;
            border:5px solid gray;
            box-sizing:border-box;
         }
      </style>
   </head>
   <body>
      <div id="wrap">
         <p>これはダミーです。これはダミーです。これはダミーです。
         これはダミーです。これはダミーです。これはダミーです。
         これはダミーです。これはダミーです。これはダミーです。
         これはダミーです。これはダミーです。これはダミーです。
         これはダミーです。これはダミーです。これはダミーです。
         これはダミーです。これはダミーです。これはダミーです。
         これはダミーです。これはダミーです。これはダミーです。</p>
      </div>
   </body>
</html>
```

1. メディアクエリ

IDがwrapのスタイルを次のようにするには

・PCで表示したときは横幅800px

・タブレットで表示したら横幅は100%の可変

```
   @media(max-width:800px){
      #wrap{width:100%;}
   }
```

. mediaは媒体。ここではスマホやPC

. max-widthはメディア属性。このコードでは表示領域(viewport)が800px以下だったら {} 内のCSSを有効にする働きがある

背景色を変えてみよう。

```
@media(max-width:800px){
   #wrap{
      width:100%;
      background-color:gray;
   }
}
```

これに準じて、タブレットやスマホでのCSSの異なる部分を書き足していきましょう。

令和３年10月　第1版　発行

著者：中島俊治