HTML/CSS実習

07. CSSの基本

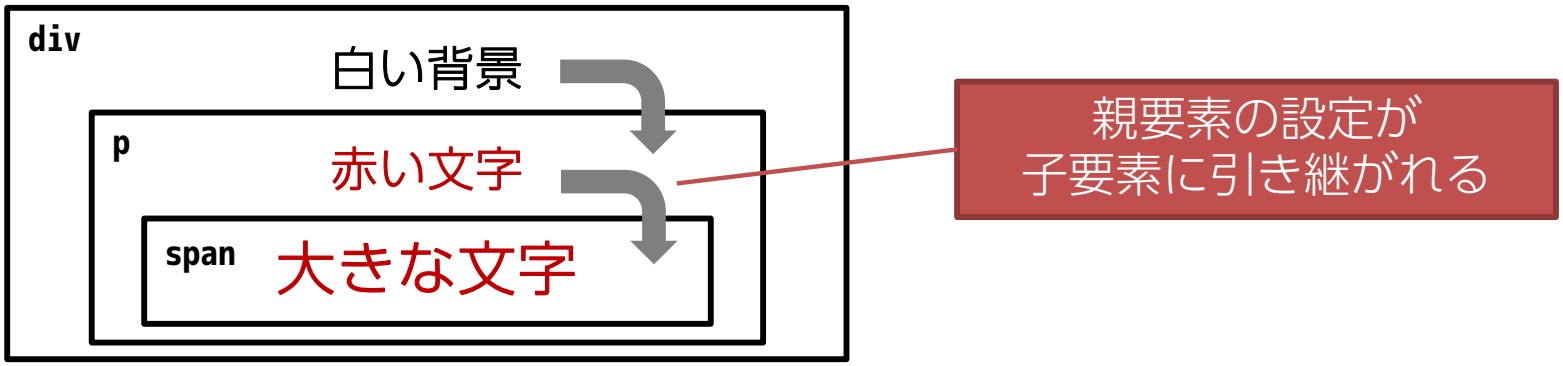
株式会社ジードライブ

今回学ぶこと

- CSSの基本書式
- セレクタ
- div要素とspan要素
- CSSの記述場所と優先順位
- 補足: 様々なセレクタ

CSSとは

- HTMLで記述された文書に対して、装飾やレイアウト(デザイン)を指定するためのデータ
- **C**ascading **S**tyle **S**heet の略
  - Cascadingとは、連続的に繋がっているというような意味で、デザインの指定が子要素に引き継がれることを示す。Style Sheetは「装飾用ファイル」という意味
  - CSSを記述したファイルの拡張子は **.css**



CSSのバージョン(仕様書)

- 1998年にCSS Level 2が、2011年にその改訂版であるCSS 2.1(CSS Level 2 Revision 1)が勧告された
- 現在はCSS Level 3が策定中だが、全体としてのバージョンコントロールではなく、モジュールごとに仕様が定義されている
  - 色、フォント、背景、変形、アニメーション…といった細かなモジュールに分かれている
  - 例えば、色についてはCSS Color Module Level 3が勧告、Level 4が草案状態にあるが、アニメーションについては、CSS Animations Level 1が草案の状態にある
- ターゲットとするブラウザが利用したいモジュールに対応しているか否かは、以下で確認することができる
  - https://caniuse.com/

CSSの書式

- CSSの基本書式

```
セレクタ {
  プロパティ: 値;
}
```

セレクタ：
装飾の対象。タグ名やクラス名、id名などで指定する

プロパティと値：
どのような装飾を行うかを指定する
プロパティと値はコロン(:)で区切る
値の後にはセミコロン(;)をつける

例：見出し(h1)の文字色を赤くして、大きさを48pxにする

```
h1 {
  color: red;
  font-size: 48px;
}
```

読みやすいコード

- CSSは1行で記述することも可能だが、読みづらくなることが多い

```
h1{color:red;font-size:10px;font-weight:bold;font-family:arial;}
```

- 適切な改行やスペースを入れることで読みやすいコードになる

```
h1 {
  color: red;
  font-size: 10px;
  font-weight: bold;
  font-family: arial;
}
```

CSSファイル

- ファイル名は自由だが、style.cssというファイル名がよく利用される
  - CSSファイルは「css」という名前のフォルダに配置されることが多い
- 1行目には文字コードの設定を記述する

文字コードの設定

```
@charset "utf-8";

h1 {
  color: red;
}

p {
  font-size: 24px;
}
```

セミコロン抜けに注意

charsetの後に半角スペース

CSSのコメント

- CSSコメントは /* この間に記述する */

```
/*-----------------------------
      トップページ
-----------------------------*/

/* ヘッダのイメージ画像 */
#keyVisual {
      height: 0;  /* 比率をパディングで調整 */
      padding: 40%;
      /* background-color: #333; */
}
/*
header {
    max-width: 960px;
}
header p {
    font-size: 24px;
}
*/
```

コメントアウト
コメントにすることで、
一時的にCSSを無効にする

CSSファイルの読み込み

- CSSファイルは、linkタグを使ってHTMLに読み込む
- linkタグはhead要素内に配置する空要素で、rel属性とhref属性を伴う
  - 複数のCSSを読み込む場合は、複数のlinkタグを記述する

```
<html>
<head>
    <title>CSS</title>
    <link rel="stylesheet" href="css/style.css">
</head>
<body>
…
```

rel属性は定型

href属性値はCSSのファイルパスを記述

キャッシュのクリア

- ページの効率的な表示のため、過去に表示した情報(HTML，CSS，画像等のデータ)は、「キャッシュ」としてブラウザに一時保存される
- CSSを変更しても、ブラウザに反映されない場合は、キャッシュを削除することで解決する場合がある
  - Shiftを押しながら、ページを読み込み直すことで、簡易的なキャッシュ削除を行うことができる

練習

- 練習07-1

セレクタの基本３種

- タイプセレクタ
- クラスセレクタ
- idセレクタ

タイプセレクタ

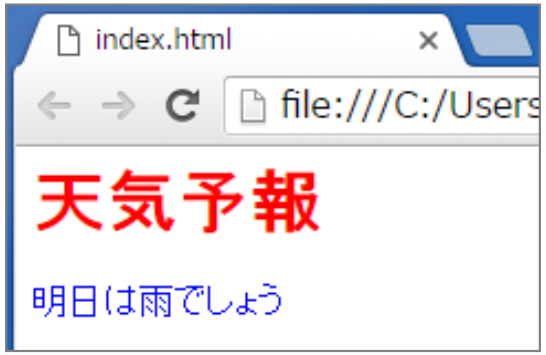
- HTMLの要素をセレクタとして記述する
  - h1～h6, p, img, a など

HTML

```
<h1>天気予報</h1>
<p>明日は雨でしょう</p>
```

CSS

```
h1 {
  color: red;
}
p {
  color: blue;
}
```



クラスセレクタ

- クラス属性を「．(ドット)」で表し、その後にクラス名を記述する

HTML

```
<h1>天気予報</h1>
<h2 class="rainy">東京</h2>
<p class="rainy">明日は雨でしょう</p>
```

CSS

```
h1 {
  color: red;
}
.rainy {
  color: blue;
}
```

IDセレクタ

- id属性を「#(ハッシュ)」で表し、その後にid名を記述する

HTML

```
<h1 id="logo">天気予報</h1>
<p>明日は雨でしょう</p>
```

CSS

```
#logo {
  color: red;
}
p {
  color: blue;
}
```

id属性とclass属性

- id属性／class属性は、どの要素にも設定できる
- 属性値には半角英数字、ハイフン(-)、アンダーバー(_)を使用することができる
  - 大文字・小文字は区別される
  - 数字で始まる属性値は設定できない

id属性とclass属性

- 同じID名はページ内で一度しか使えない
  ⇒ ページ内リンクや優先したい固有の設定がある場合に利用

index.html ✕

```
<h1>天気予報</h1>
<h2 id="area">関東地方</h2>
<p>明日は雨でしょう</p>
<h2 id="area">関西地方</h2>
<p>明日は晴れでしょう</p>
```

kanto.html ○

```
<h1>天気予報</h1>
<h2 id="area">関東地方</h2>
<p>明日は雨でしょう</p>
```

kansai.html

```
<h1>天気予報</h1>
<h2 id="area">関西地方</h2>
<p>明日は晴れでしょう</p>
```

- 同じクラス名はページ内で何回でも使うことができる
  ⇒同じ設定がある場合に利用

index.html ○

```
<h1>天気予報</h1>
<h2 class="area">関東地方</h2>
<p>明日は雨でしょう</p>
<h2 class="area">関西地方</h2>
<p>明日は晴れでしょう</p>
```

id属性とclass属性

- class属性は1つの要素に対して複数設定**できる**
  - 複数の値(クラス名)を半角空白で区切る

HTML

```
<p class="info sunny">今日は晴れ</p>
<p class="info rainy">明日は雨</p>
```

半角スペース

CSS

```
.info {
    font-size: 18px;
    border-radius: 5px:
}
.sunny {
    color: red;
}
.rainy {
    color: blue;
}
```

- id属性は1つの要素に対して複数設定**できない**

```
<p id="info sunny">今日は晴れです。</p>
```

id属性とclass属性

- セレクタとして、要素名とid名やクラス名を連結して記述することができる

HTML

```
<h1>天気予報</h1>
<h2 class="rainy">関東地方</h2>
<p class="rainy">明日は雨でしょう</p>
```

CSS

```
h2.rainy {
  color: mediumblue;
}
p.rainy {
  color: dodgerblue;
}
```

要素名＃ID名　　例）h1#logo
要素名.クラス名　例）h2.area

div要素とspan要素

- レイアウトや装飾のために使用される要素
- id属性やclass属性を伴って使用されることが多い
- divは見出しや段落、ヘッダーやフッターなどを内包することができる
  - header, main, footer などの中で利用することも可能
- spanは見出しや段落などの中で使用する

```
<h1>天気予報</h1>
<div class="kanto">
  <h2>関東地方</h2>
  <p>明日は<span class="rainy">雨</span>でしょう</p>
</div><!-- end of div.kanto -->
<div class="kansai">
  <h2>関西地方</h2>
  <p>明日は<span class="sunny">晴れ</span>でしょう</p>
</div><!-- end of div.kansai -->
```

練習

- 練習07-2

応用的なセレクタ

グループセレクタ

- 複数のセレクタを「, (カンマ)」区切りで記述
  ⇒ 同様の設定をする場合に有効

HTML

```
<h1>天気予報</h1>
<h2>関東地方</h2>
<p>明日は<span class="rainy">雨</span>でしょう</p>
```

CSS

```
h1, h2, .rainy {
  color: mediumblue;
}
```

※セレクタが冗長で読みづらくなる場合は「，」の後に改行する

子孫セレクタ

- 階層構造を利用してセレクタを指定
  - 内包するセレクタを半角スペースを空けて記述

HTML

```
<header>
  <h1>天気予報</h1>
  <p>全国の天気をお知らせします</p>
</header>
<section id="kanto">
  <h2>関東地方</h2>
  <p>明日は<span>雨</span>でしょう</p>
  <section>
    <h3>東京：<span>雨</span></h3>
    <p>明日は<span>雷雨</span>です</p>
  </section>
</section>
```

CSS

```
header p {
  color: green;
}
#kanto p {
  color: royalblue;
}
h1, #kanto h3 span {
  color: mediumblue;
}
#kanto section p span, h2 {
  color: orange;
}
```


```
#kanto section p span
#kanto section span
#kanto p span
#kanto span
```

省略可能

スペース・カンマによる違い

記述の仕方	説明	記述例	記述例の説明
半角スペースで区切る	子孫セレクタ。 「A要素の**中の**B要素」といった**内包**を表す。	`h3 .area` 空白あり	h3要素が**内包する**areaというクラス名の要素
くっつけて記述	「A**で且つ**B」といった**複合条件**を表す。IDやクラスなどと一緒に用いられる。	`h3.area` 空白なし	h3で**且つ**areaというクラス名が付いている要素
カンマで区切る	グループセレクタ。 「A要素**と**B要素**と**C要素」のように、**複数のセレクタ**をまとめて指定する場合に用いる。	`h3, .area`	h3要素**と**areaというクラス名が付いている要素

親子セレクタ >

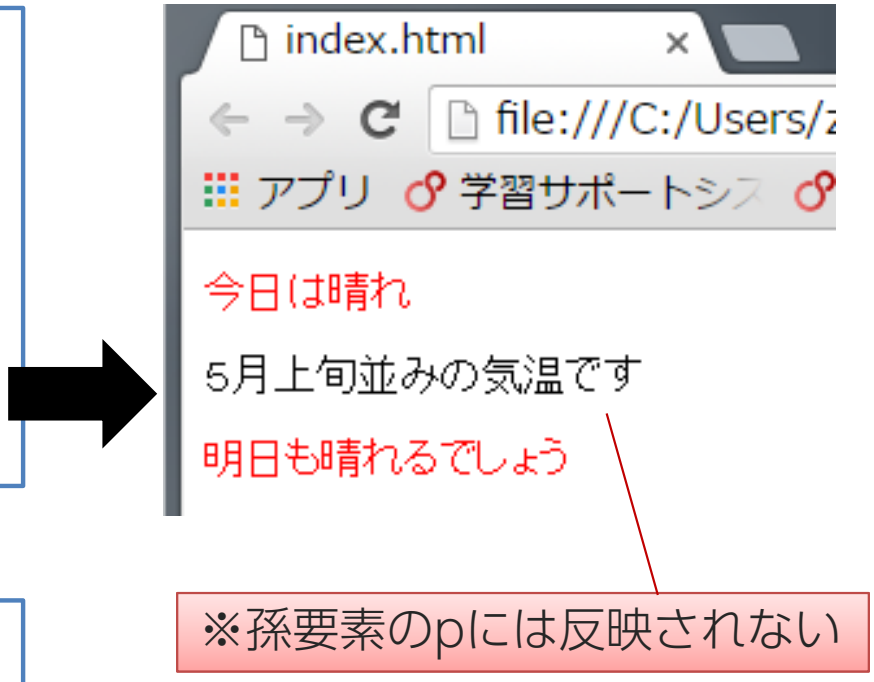
- 直接の子要素を選択するためのセレクタ

HTML

```
<div class="weather">
  <p>今日は晴れ</p>
  <div class="detail">
    <p>５月上旬並みの気温です</p>
  </div>
  <p>明日も晴れるでしょう</p>
</div>
```

CSS

```
.weather > p {
  color: red;
}
```



隣接セレクタ +

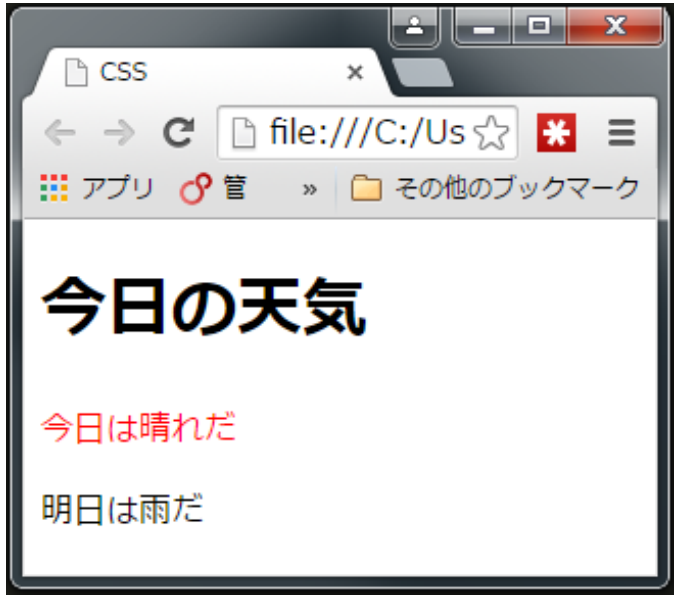
- 同じ階層の次の要素を示す

HTML

```
<h1>今日の天気</h1>
<p>今日は晴れだ</p>
<p>明日は雨だ</p>
```

CSS

```
h1 + p {
  color: red;
}
```



全称セレクタ *

- 全ての要素を指す
  - ユニバーサルセレクタとも呼ばれる
  - 各要素が初期値としてもっているスタイルの打消しに利用されることが多い

```
* {
  margin: 0;
  padding: 0;
  font-weight: normal;
  list-style-type: none;
  text-decoration: none;
  color: #000;
  font-size: 18px;
}
```

要素間の余白をなくす

太字の打消し
リストマークをなくす
下線をなくす

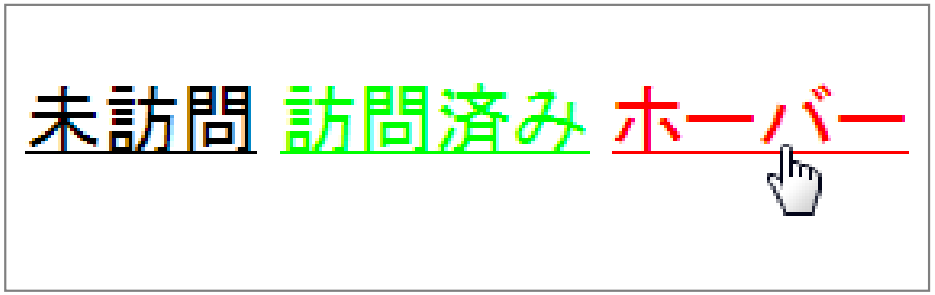
文字の色とサイズの統一

疑似クラス

- 要素が特定の状態にある場合のスタイルを指定するためのクラス指定を疑似クラスと呼ぶ

例：アンカー要素の疑似クラス

```
a:link { color: #000; }    /* デフォルト状態 */
a:visited { color: #0f0; } /* 訪問済み */
a:hover { color: #f00; }   /* マウスが重なった状態 */
a:active { color: #00f; }  /* マウスボタンが押されている状態 */
```



練習

- 練習07-3

CSSの記述場所と優先順位

CSSの記述位置

① HTMLタグのstyle属性として記述する
- 他の記述方法よりも、優先される
- JavaScriptと連携させて活用する

② HTMLページ内にstyle要素として記述する
- head要素内に記述
- 特定のページにのみ適用するスタイルの設定に使用することがある

③ 外部ファイルに記述する
- メンテナンスがしやすい
- もっとも一般的な記述方法

これまで学んできた方法

CSSの記述①：style属性

- HTMLタグのstyle属性としてCSSを記述する
- インラインCSSと呼ばれる

```
<h1 style="color:red;">天気予報</h1>
<p style="font-size:18px; color:blue;">明日は雨でしょう</p>
```

プロパティ
区切り コロン
値
指定終了 セミコロン

あまり利用されないが、他の記述方法よりも設定が優先される
優先したい設定がある場合やJavaScriptと連携させる場面などで活用する

CSSの記述②：style要素

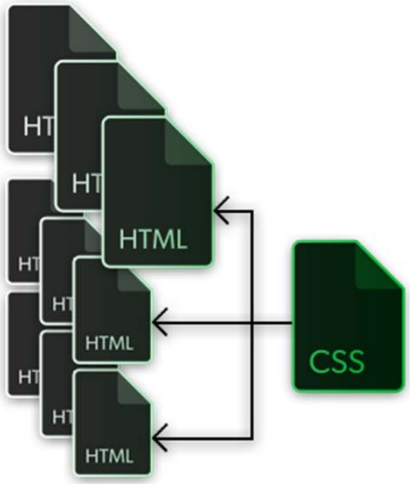
- head内にstyle要素としてCSSを記述する

```
<head>
<style>
h1 {
  color: red;
}
p {
  font-size: 18px;
  color: blue;
}
</style>
</head>
<body>
<h1>天気予報</h1>
<p>明日は雨でしょう</p>
</body>
```

特定のページにのみ適用するスタイルを設定する場合に使用することがある

CSSの記述③：外部ファイル

- 外部CSSファイルのメリット
  - 外部にCSSファイルを作成し、複数のHTMLファイルにリンクさせることが可能
  - CSSファイルを修正することで、リンクした全てのHTMLファイルにデザインの変更が反映されるようになり、効率的に作業できる



HTML

```
<head>
<link rel="stylesheet" href="style.css">
</head>
<body>
<h1>天気予報</h1>
<p>明日は雨でしょう</p>
</body>
```

＋

CSS

```
@charset "utf-8";
h1 {
  color: red;
}
p {
  font-size: 18px;
  color: blue;
}
```

＝

CSSの優先順位①

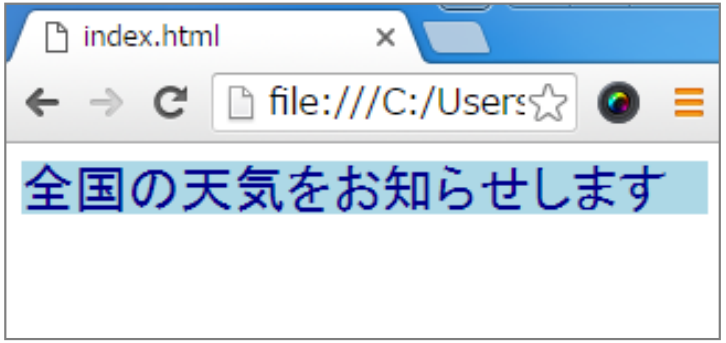
・上から下に読み込まれ、設定が追加／上書きされる
⇒ 下の方の記述が優先（基本ルール）

HTML

```
<p>全国の天気をお知らせします</p>
```

CSS

```
p {
  color: red;
  font-size: 24px;
}
p {
  color: darkblue;
  background-color: lightblue;
}
```



red を上書きした

背景色の設定が追加された

CSSの優先順位②

・より詳細なセレクタが優先される

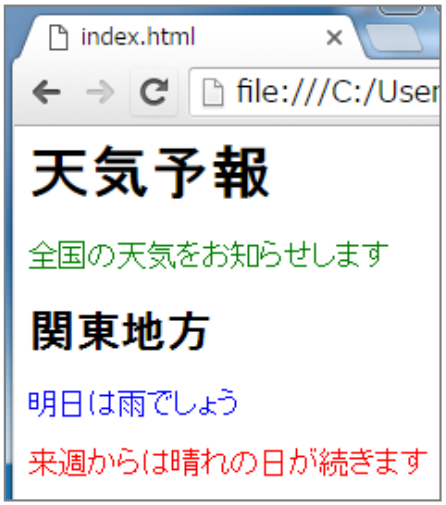
⇒ タグは1点、クラスは10点、idは100点と決められており、
合計点（詳細度）が高いセレクタが優先される

HTML

```
<header>
  <h1>天気予報</h1>
  <p>全国の天気をお知らせします</p>
</header>
<section class="kanto">
  <h2>関東地方</h2>
  <p id="rainy">明日は雨でしょう</p>
  <p>来週からは晴れの日が続きます</p>
</section>
```

CSS

100点
```
#rainy {
  color: blue;
}
```
11点
```
.kanto p {
  color: red;
}
```
1点
```
p {
  color: green;
}
```



上から下への基本ルールではなく、
より**詳細度**が高いものが優先されている

CSSの優先順位③

・**!important** が最優先。style属性が第二優先

HTML

```
<h2 id="area" style="color:red;">関東地方</h2>
<p id="rainy">明日は雨でしょう</p>
<p id="sunny" style="color:red;">
   来週からは晴れの日が続きます
</p>
```

CSS

```
p {
  color: orange !important;
}
#area, #rainy {
  color: blue;
}
#sunny {
  color: green;
}
```

id指定よりもstyle属性が優先

id指定よりも!importantが優先

style属性よりも!importantが優先

!important や style属性は最終手段なので、最低限の使用に留める

CSSの優先順位④

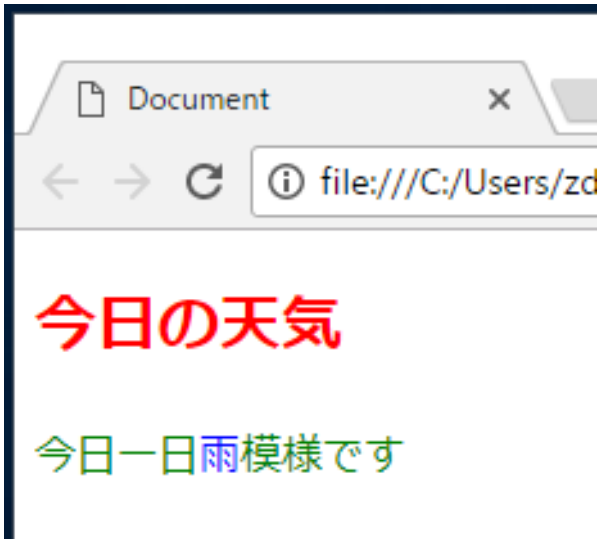
・文字のスタイルは、子要素の設定が優先される

HTML

```
<section>
  <h1>今日の天気</h1>
  <p>今日一日<span>雨</span>模様です</p>
</section>
```

CSS

```
span {
  color: blue;
}
p {
  color: green;
}
section {
  color: red;
}
```



練習

- 練習07-4

補足: 様々なセレクタ

セレクタに関するドキュメント
https://developer.mozilla.org/ja/docs/Learn/CSS/Building_blocks/Selectors

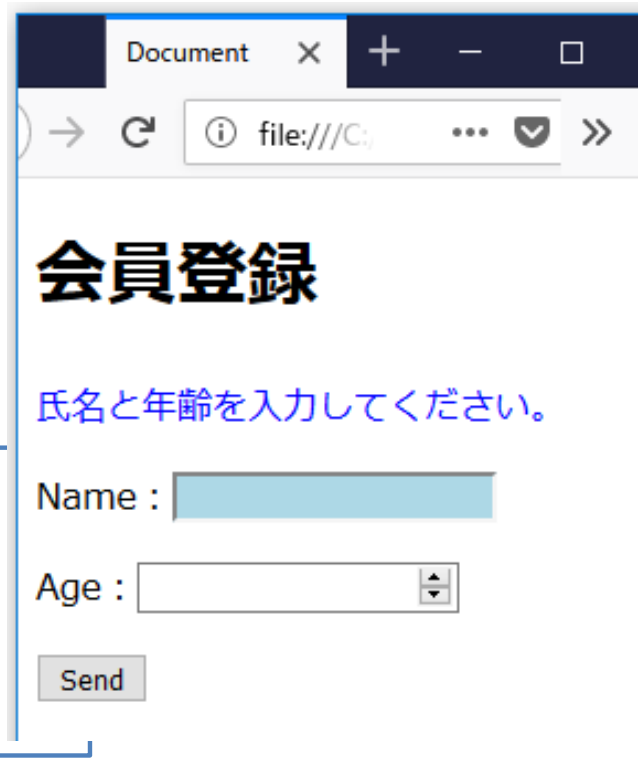
属性セレクタ [ ]

- 属性を元に、対象を選択するためのセレクタ
  - 属性値が〇〇で始まる、△△を含む、といった表現も記述可能

参考:https://developer.mozilla.org/ja/docs/Web/CSS/Attribute_selectors

```
p[lang="ja"] {
    color: blue;
}
[type="text"] {
    background: lightblue;
}
```

```
<h1 lang="ja">会員登録</h1>
<p lang="ja">氏名と年齢を入力してください。</p>
<p>Name：<input type="text"></p>
<p>Age：<input type="number"></p>
<input type="submit" value="Send">
```



疑似クラス

- コロン（：）で始まり、要素のある状態や構造などを示すセレクタ
  - a要素にマウスオーバーした状態
  - フォームの入力欄にカーソルが入った状態
  - 偶数番目のテーブル行　…など

  参考：https://developer.mozilla.org/ja/docs/Web/CSS/Pseudo-classes

疑似クラスの記述例	説明
a**:hover**	マウスオーバーした状態のa要素
input**:focus**	カーソルが入った状態のinput要素
:checked + span	チェックが入ったラジオボタン（またはチェックボックス）の次のspan要素
tr**:nth-child(2n)**	偶数番目のtr要素
div**:has(h4)**	子孫としてh4要素をもつdiv要素

a要素の疑似クラス

- 以下の疑似クラスは、基本的にa要素に対して指定する
  - :hoverはa要素以外でも使用可能

  ⇒ :hoverのみでは、body要素も含めた全要素に影響してしまうため、a:hover のように要素を限定して使用する

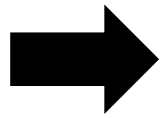
表記法	説明
:link	リンク先のページに未訪問の状態
:visited	リンク先のページに訪問済みの状態
:hover	マウスオーバー時の状態
:active	マウスでプレスしている時の状態

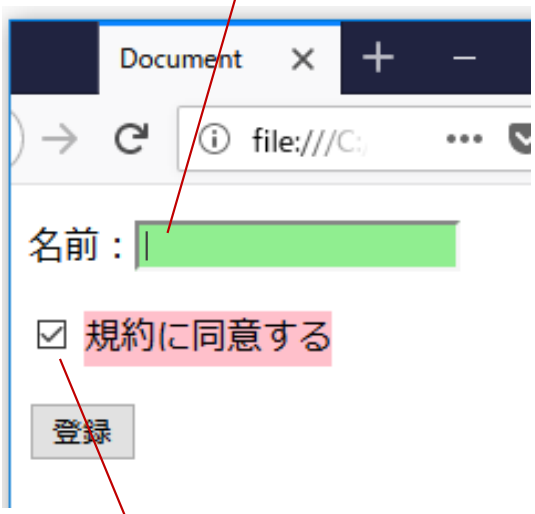
※ a要素に対して、上記すべての疑似クラスを設定する必要はない
複数の疑似クラスを設定する場合は、上記の順序で記述しないと機能しない

フォームの疑似クラス

```
:focus {
    background: lightgreen;
}
:checked + span {
    background: pink;
}
```

```
<p>名前：
<input type="text" autofocus="focus">
</p>
<p>
<input type="checkbox" name="agree">
<span>規約に同意する</span>
</p>
<input type="submit" value="登録">
```





:nth-child / :nth-of-type

- N番目の要素を指定するセレクタ

表記法	説明
:nth-child(**N**)	同じ階層内でn番目の要素 タグの種類に関係なく数える
:nth-of-type(**N**)	同じ階層内でn番目の要素 タグの種類ごとに数える

※ Nには整数、自然数nを含む式、またはキーワードを入れる

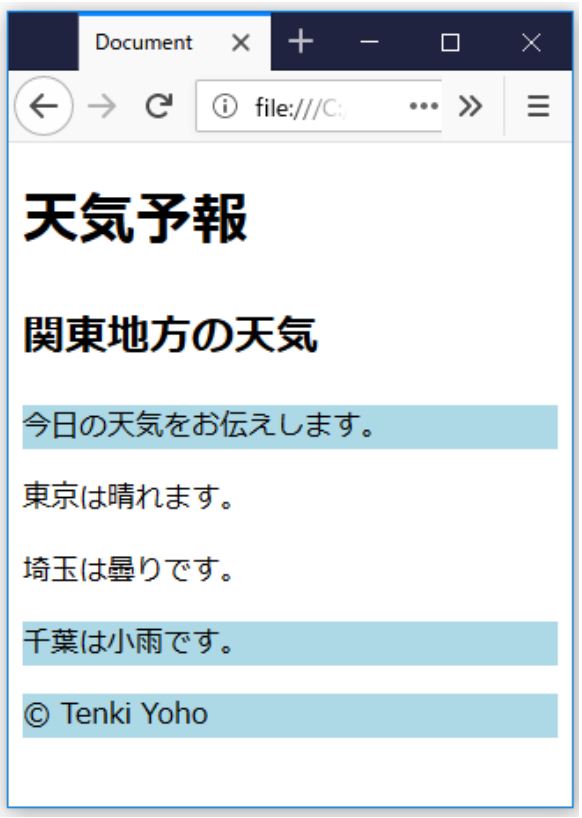
```
:nth-child(2)        ⇒ ２番目の要素
:nth-child(2n)       ⇒ 0, 2, 4, 8, . . .番目の要素（２の倍数・偶数）
:nth-child(2n + 1)   ⇒ 1, 3, 5, 7, . . .番目の要素（２の倍数+1・奇数）
:nth-child(-n + 5)   ⇒ ５番目までの要素
:nth-child(n + 5)    ⇒ ５番目以降の要素
:nth-child(n + 3):nth-child(-n + 8) ⇒ ３～８番目までの要素
:nth-last-of-child(3) ⇒ 最後から数えて３番目の要素
:first-child         ⇒ 1番目の要素。nth-child(1)と同じ
:last-child          ⇒ 最後の要素
```

:nth-childの挙動

```
:nth-child(3) {
    background: lightblue;
}
```

```
<body>
<header>
    <h1>天気予報</h1>                    ①
    <h2>関東地方の天気</h2>              ②
    <p>今日の天気をお伝えします。</p>    ③
</header>
<main>
    <p>東京は晴れます。</p>   ①
    <p>埼玉は曇りです。</p>   ②
    <p>千葉は小雨です。</p>   ③
</main>
<footer>© Tenki Yoho</footer>
</body>
```

①
②
③

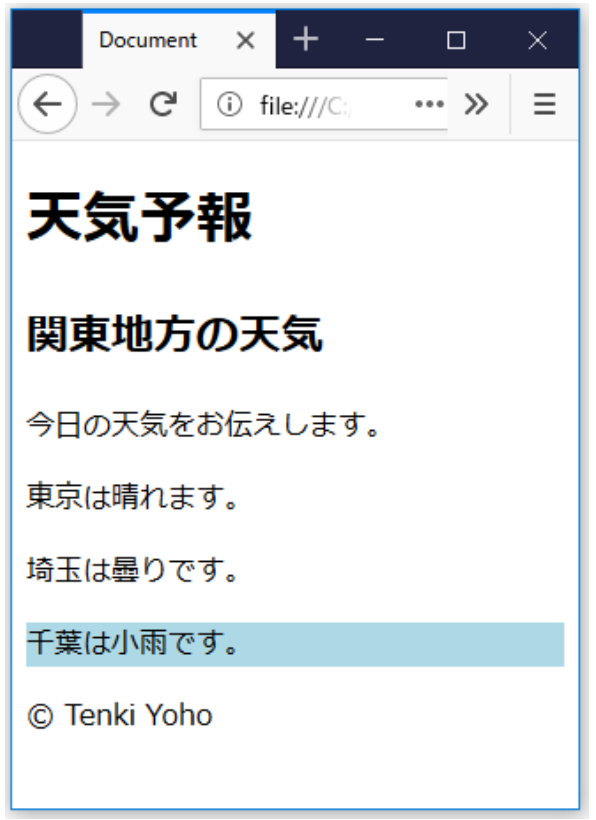


タグの種類に関係なく
階層ごとに数える

:nth-of-typeの挙動

```
:nth-of-type(3) {
    background: lightblue;
}
```

```
<body>
<header>
    <h1>天気予報</h1>                          ①
    <h2>関東地方の天気</h2>                    ①
    <p>今日の天気をお伝えします。</p>          ①
</header>
<main>
    <p>東京は晴れます。</p>                    ①
    <p>埼玉は曇りです。</p>                    ②
    <p>千葉は小雨です。</p>                    ③
</main>
<footer>© Tenki Yoho</footer>
</body>
```

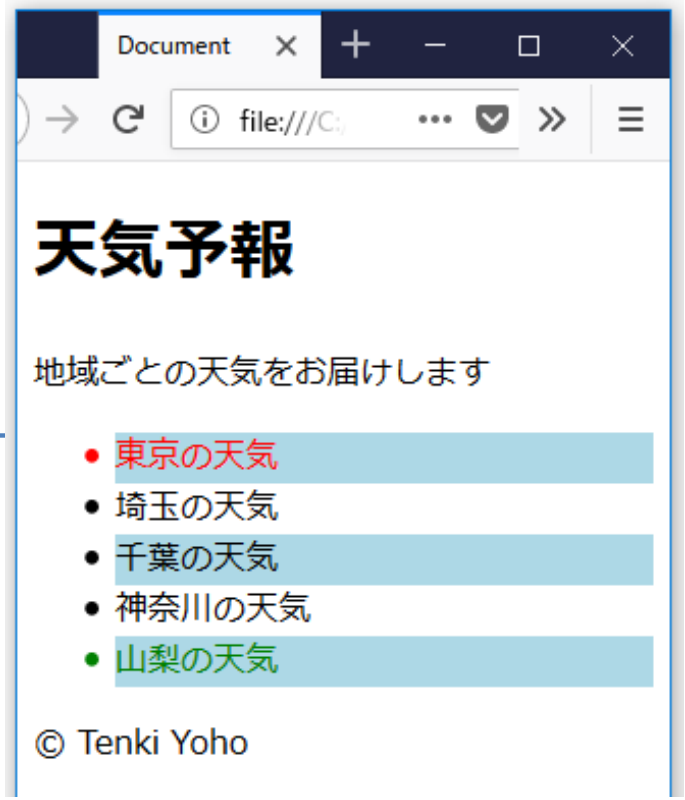


階層ごと、タグの種類ごとに数える

:nth-childの使用例

```
.area > :nth-child(2n+1) {
    background: lightblue;
}
li:first-child {
    color: red;
}
li:last-child {
    color: green;
}
```

```
<body>
<h1>天気予報</h1>
<p>地域ごとの天気をお届けします</p>
<ul class="area">
    <li>東京の天気</li>
    <li>埼玉の天気</li>
    <li>千葉の天気</li>
    <li>神奈川の天気</li>
    <li>山梨の天気</li>
</ul>
<p>© Tenki Yoho</p>
</body>
```



範囲を限定するために親子セレクタなどと
併用することが多い

:has, :not, :is, :where

- 引数としてセレクタをとる疑似クラスとして、以下のものが存在する

表記法	記述例・説明
:has(**Selector**)	子孫要素を利用して、要素を選択できる header**:has(**h1**)** ⇒ 子孫要素としてh1要素をもつheader要素
:not(**Selector**)	否定を利用して、要素を選択できる div**:not(**:has(p)**)** ⇒ 子孫要素としてp要素をもたないdiv要素
:is(**...Selectors**) :where(**...Selectors**)	セレクタを列挙してまとめることができる。列挙されたセレクタのいずれかに当てはまる要素を選択できる。:is()は詳細度をもつが、:where()は詳細度をもたない **:is(**section, article**)** p ⇒ section内のp要素またはarticle内のp要素

疑似要素

- 疑似要素はダブルコロン（::）で始まるセレクタで、CSS側から疑似的にHTML内に要素を作り出す

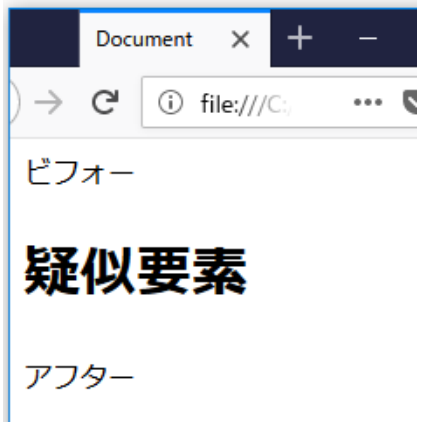
参考：https://developer.mozilla.org/ja/docs/Web/CSS/Pseudo-elements

表記法	説明
::before	要素内の最初に疑似的な要素を追加
::after	要素内の最後に疑似的な要素を指定

※ダブルコロンは疑似要素を、シングルコロンは疑似クラスを表す
⇒シングルコロンで記述しても、ブラウザが解釈し、表示される

```
div::before {
    content: "ビフォー";
}
div::after {
    content: "アフター";
}
```

```
<div>
    ::before 疑似的な要素を生成
    <h1>疑似要素</h1>
    ::after 疑似的な要素を生成
</div>
```



::before / ::afterの使用例

- contentプロパティを記述し、疑似要素として表示させる文字や画像を指定する

```
.info > p::before {
    content: url(icon.png);
    position: relative;
    top: 7px;
}
.new::after {
    content: "NEW";
    font-size: 75%;
    color: #fff;
    background: #f00;
}
```

追加した疑似要素にスタイルを適用することも可能

```
<div class="info">
    <p class="new">新宿でグループ展を開催します。</p>
    <p>四谷で個展を開催しました。</p>
    <p>写真集「風の谷」を発売しました。</p>
</div>
```