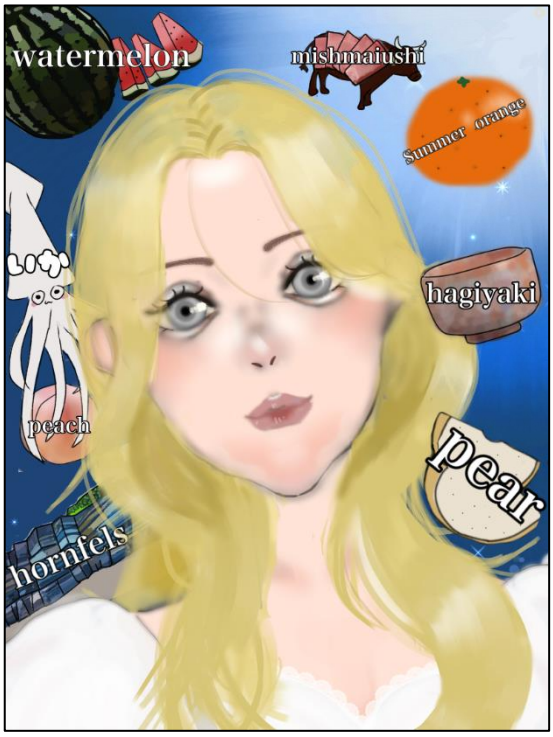
ホームページ制作　知識・実践編



UI8 デジタルアートコンテスト２０２４　入賞作品

令和７年５月

特別支援教育推進室

INDEX

知識・実践編

Ⅰ　基礎知識と準備

学習内容

- ✓ Web の基礎知識
- ✓ Web サイトの構成
- ✓ 作成ツールアプリの利用

Web に関する基本知識を理解し、Web 制作の準備をしましょう。

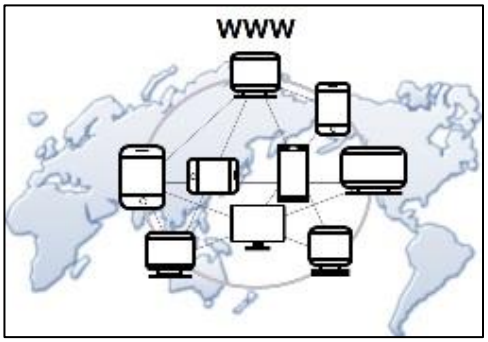
Web の基礎知識

インターネット上の情報を活用

世界に網のように張り巡らされているネットワーク網のことを「**WWW：World Wide Web**」といいます。WWW の Web に注目してみましょう。

Web を利用し、インターネット上に文書や画像、動画などの情報発信をしたり、必要な情報を閲覧したりすることができます。

例えば、メール・LINE を使い、家族や友人と連絡を取ることができます。Web サイトでは、ニュース、天気予報、ウィキペディア、YouTube、Instagram、Facebook、X（旧 Twitter）、インターネットショッピング、クックパッド、お得なクーポンなど、生活に役立つ情報を活用することができます。



ブラウザとは

パソコンやスマホを使って、ネット上で公開されている Web ページやシステムなどを閲覧・操作するときに利用するアプリのことです。

主なブラウザアプリとピクトグラムは以下のとおり（PC 内にアプリのピクトグラムがない場合は、別途インストールする必要があります）

- Google Chrome  （今回こちらを利用します）
- Firefox  
- Microsoft Edge 
- Safari 

Web サイト閲覧

パソコンやスマホ・タブレットなどでインターネット上の Web ページの情報を画面上に表示するためには、ブラウザアプリを利用します。

Google Chrome  アプリのピクトグラムで、**ブラウザ**を開いてみましょう。

サイトの表示および閲覧方法例

画像では、Google サイトを例として説明します。

- **アドレスバー利用方法**
  閲覧したいサイトの URL を直接入力して［Enter］キーを押す
  （例：「天気ドット jp」サイトであれば、【tenki.jp】と URL を入力してから［Enter］キーを押す）
  → サイト内容が表示される

- **検索窓でキーワード検索する方法**
  ブラウザで Google や Yahoo の「検索サイト」を利用する
  - 「検索窓」に「キーワード」（天気予報など）を入力する
    複数のキーワード入力例（空白文字で区切る）：天気　東京　港区
    → ［Enter］キーを押し検索
  - 「検索結果」の一覧表示から、利用したい項目（リンクテキスト）を選んでクリックする
    → サイト内容が表示される（必要な情報が見つかるまで繰り返す）

上記の手順で、インターネット上にあるページを開くことができ、知りたい情報を得たり活用したりすることができます。

Web サイトの構成

ファイルと拡張子

Web サイトでは、主に以下の 2 種類のファイルを利用します。
それぞれに役割があり、ソースコードを記述・編集してファイルを作成します。

- **HTML** ファイル（拡張子：「.html」）：
  文章構成の土台と枠組みを決める

- **CSS** ファイル（拡張子：「.css」）：
  デザイン・配色・レイアウトなどの、見栄えを決める

💻用語解説

・ファイル：
データそのもので、データやプログラムの個々のまとまりのこと

・拡張子：
ファイルの種別を表すもので、ファイル名の最後の「.」（ドット）より後に続く部分のことでアプリ、ソフトと関連付けられ、それぞれに固有の拡張子がある
例：　「Word」→「**.docx**」　「Excel」→「**.xlsx**」

💻確認

「**拡張子**」の表示はされていますか。

ファイル名の後に表示される「**.html**」や「**.css**」などの事です。

表示されていない場合は、以下の手順で設定します。

💻手順

1. 「マイドキュメント」などのウィンドウを開く
2. ウィンドウの 「表示」タブ をクリックする
3. 「表示／非表示」グループ の 「ファイル名拡張子」 部分に「レ」チェックを入れる
4. 表示確認：ファイル名の後に拡張子が表示

   HTML ファイルなら「.html」、CSS ファイルなら「.css」の拡張子が表示される

   他ファイルの場合：Word なら「.docx」、Excel なら「.xlsx」など

   ファイル名の後に拡張子が表示されていれば、設定完了となる

Web サイト公開の仕組み

公開用の HTML ファイルと CSS ファイルが完成したら、FTP アプリを利用して Web サーバー（別途、契約が必要）へファイルアップロードを行うことで、公開できます。

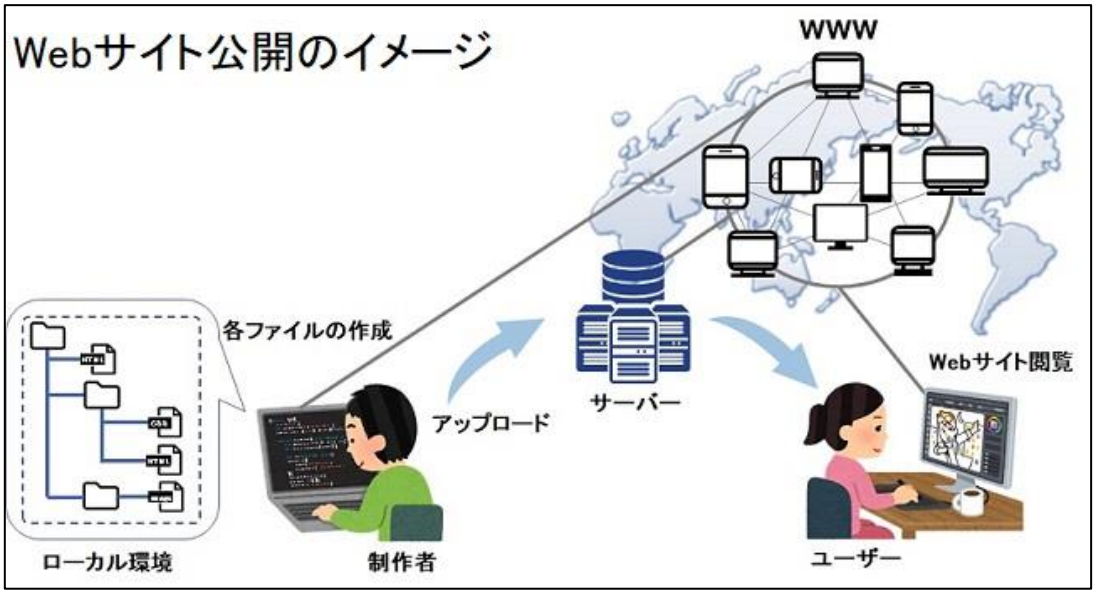
Web サーバーへのアップロードが完了したら、その情報はインターネットにつながっている世界中のユーザーが閲覧可能になります。

※ 本教材では公開まで行いませんので FTP アプリやサーバーの契約は不要です。

💻**用語解説**

- **FTP：**

  ファイル転送用アプリ。サーバーへファイルアップロードまたは、サーバーからファイルダウンロード時に利用するアプリケーション



作成ツールアプリの利用

パソコンがあれば HTML や CSS のコードを書き、Web ページの制作を進めることができます。

以下の 2 つのアプリを利用して、Web 制作を行います。

- **テキストエディタ**　HTML や CSS を編集するために必要（専用のアプリを利用します）
- **ブラウザ**　HTML を表示するために必要

📖 テキストエディタとは

テキストファイル（中身が文字だけのファイル）を作成・編集時に使うアプリのことです。

なお、"edit"は「編集」で、"editor" とは「編集者」の意味です。

今回の学習では、HTML ソースの編集のために利用します（CSS ファイルの編集時も利用します）。

「メモ帳」でも作成可能ですが、テキストエディタを使用する方がメモ帳よりも見やすく、機能的で利用しやすいので便利です。

本教材では、**「Visual Studio Code（ヴィジュアル スタジオ コード）」**アプリを利用して解説します。

なお、既に利用されているエディタがあれば、そちらをご利用ください。

2　HTML による文書作成

学習内容

- ✓　HTML を学ぼう
- ✓　文章構造の理解
- ✓　マークアップの基本ルール
- ✓　段落、見出しについて
- ✓　リストについて
- ✓　テーブルについて
- ✓　画像表示について
- ✓　リンク設定について
- ✓　id、class 名について

文章構造とマークアップのタグ要素を学びましょう。

HTML を学ぼう

Web サイトを表示させるために必要となる知識を学んでいきましょう。

HTML は、「JavaScript」などのプログラミング言語に比べると、容易に理解しやすいものです。

「難しそう」という先入観をひとまず捨てて、まずは基本について学習していきましょう。

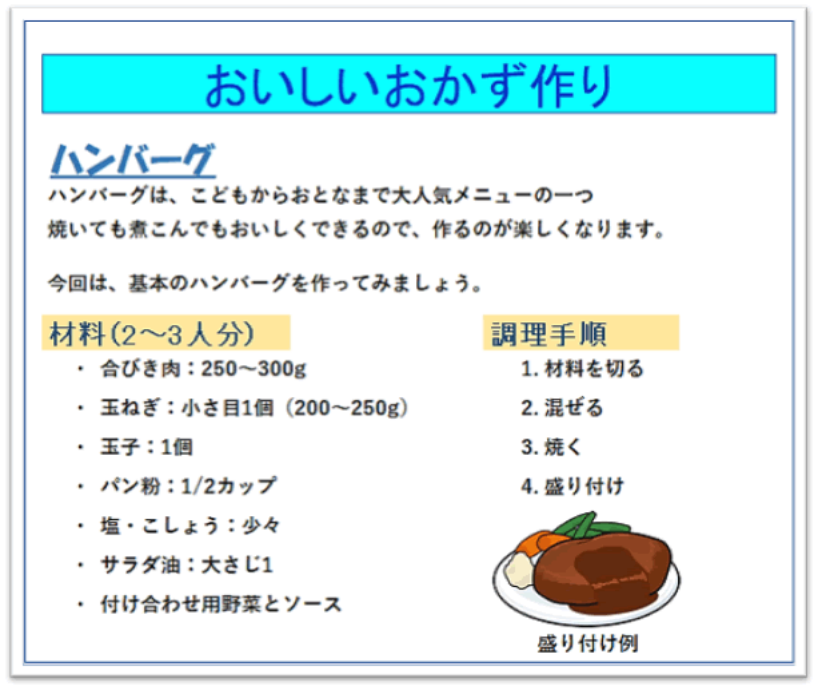
最初に、「おいしいおかず作り」のチラシを HTML 化してみましょう。

📖 HTML とは

"Hyper Text Markup Language"の略称です。

Web ページを作成するときにコンピュータに対して情報を伝えるために意味付け（マークアップ）を行う言語です。

文章構造の理解



「おいしいおかず作り」のチラシ

チラシの主な内容

- 「おいしいおかず作り」では、「ハンバーグ」の作り方を紹介
- ハンバーグを作る材料と、2～3 人分の分量を示す
- 調理手順を示す
- 盛り付け例としてハンバーグの完成例の画像を表示

「見出し」を利用して、各項目の内容を分類しているのが分かります。

また、「段落」が「箇条書き」になっているので、何を説明しているのかが分かりやすく表現されています。

HTML 化するには、ブラウザに表示させる全てのテキストに対して、文章それぞれの役割を示すために**マークアップ**（意味付け）していきます。

文章の役割を大まかな例で示すと以下のとおりです。
まずは文章の表示から内容を読み取り、分類していきましょう。

- 見出し
- 段落
- 箇条書き
- 表組み

📖 マークアップとは

コンピュータは、文章の内容を問わず全てをテキスト（文字列）として認識するため、文章構造は判断できません。
そのため、「これがタイトル」「ここは段落」などのように、コンピュータに文章構造が理解できるように、文章に役割の意味付け・定義付けを行います。
そのことをマークアップと言います。
意味付けを行うだけなので、記述した「タグ要素」などは、ブラウザに表示されません。

マークアップの基本ルール

HTML 基本設定（head 部分など）

HTML ファイル作成時に必要な記述があります。
12pの 1 行目の<!DOCTYPE html> ～ 8 行目の</head>部分については、呪文のように記載する内容だと思ってください。
それぞれ意味がありますが、現時点では、「**とりあえず記述する**」ことが分かれば良いです。

- 基本設定では、サイトの情報がたくさん記述される
- ブラウザに表示されるのはほんの一部のみだが、重要な項目ばかりである
- title タグに記載の内容は、ブラウザのタブ部分に表示されるページのタイトルに
- CSS ファイルの読み込みについては、後ほど学習する内容になる

💻HTML 必須記述内容

設定内容 詳細

1	<html>	・1 行目：html タグ開始
2	<head>	・2 行目：head タグ開始
3	＜title＞●タイトル●</title>	・3 行目：title タグ（ブラウザのタブ部分に表示される内容）
4	</head>	・4 行目：head タグ終了
5	<body>	・5 行目：body タグ開始
6		
7	ブラウザに表示される部分です	ブラウザに表示される部分です
8		
9	</body>	・9 行目：body タグ終了
10	</html>	・10 行目：html タグ終了

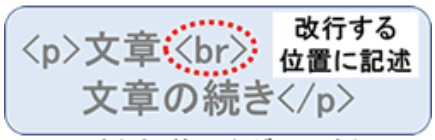
※タグについては大文字、小文字の区別はありません。

基本設定（ブラウザ表示したいテキストをタグではさむ例）：
<h1>見出し</h1>
** 本文内容や文章 **

- **基本：**右図の上側参照
  ブラウザ表示させる全テキストに対して、タグではさむ
  - 文章の前に**開始タグ**を記述
  - 文章の最後尾に**/（スラッシュ）付きの終了タグ**を記述

- **例外：**右図の下側参照
  空要素といい**終了タグ**が不要な要素
  例：以下の 3 つの要素
  - **
：強制改行タグ　改行したい場所に
**タグの記述を行う
  - **<img>**：画像挿入タグ
  - **<hr>**：水平線タグ　段落と段落の間などのテーマの区切りで利用

基本構文例：タグ要素



例外：終了タグ不要例

段落・見出しについて

ここでは、【段落】と【見出し】をマークアップしていきます。
要素名の違いはありますが、マークアップ方法は同じです。
段落内で利用する改行（空要素）も含めて、一緒に説明します。

- **「段落」**に利用したい文章テキストは、 と ではさむ
- 文章テキストを**「改行」**したい場合は、改行したい部分に
を記述
- 文章テキストを**「強調」**したい場合は、そのテキスト部分をとではさむ
- **「見出し１」**に利用したいテキストは、<h1>と</h1>ではさむ（見出しは１～６まで）

・見出しの文字サイズは、<h1>が一番大きく表示され、<h2>→<h6>の順に小さく表示される

・見出しの文字は、太文字で表示される

・見出しレベルは上位から順に h1、h2、h3、h4 と順に利用

・見出しのテキストの上下に余白が付き、単独の行として表示

・文章の構文として見出しを利用するため、文字サイズを大きく見せるための「見出し」の利用ではない事を理解する

文章中のテキストが、「段落」または「見出し」のどちらの要素にあたるのかを明確に判断していきましょう。

✎ HTML 記述例

```
<p>ブラウザに表示させる「段落」テキストを p 要素のタグではさむ。</p>
<p>本文に改行が入ります。<br>改行後の本文です。</p>
<p>本文の中で<strong>強調部分</strong>に利用。</p>

<h1>「見出し 1」マークアップ方法を学ぶ</h1>
<h2>「見出し 2」段落と見出しのマークアップ方法は同じ</h2>
<h3>「見出し 3」見出し要素のレベルは 1～6 まで</h3>
<h4>「見出し 4」見出しのレベルの順序は意味がある</h4>
<h5>「見出し 5」文字サイズのために見出しは利用しない</h5>
<h6>「見出し 6」文字サイズが小さくても、見出しです</h6>
```

💻 ブラウザ表示例

ブラウザに表示させる「段落」テキストを p 要素のタグではさむ。

本文に改行が入ります。
改行後の本文です。

本文の中で**強調部分**に利用。

「見出し 1」マークアップ方法を学ぶ

「見出し 2」段落と見出しのマークアップ方法は同じ

「見出し 3」見出し要素のレベルは 1～6 まで

「見出し 4」見出しのレベルの順序は意味がある

「見出し 5」文字サイズのために見出しは利用しない

「見出し 6」文字サイズが小さくても、見出しです

💻参考：見出しの構成について＜重要＞

見出しがあることで、文章をまとめやすくなります。
例えば、カテゴリーによる分類、過去・現在・未来の時系列や順序を示す時は、見出しがあると分りやすいですね。

章立て とは

「部（タイトル）」「章（第 1 章）」「節（1.2）」「項（1.2-1）」「目（1.2-1-1）」のような構成です。
文章のまとまりを分かりやすく構成するために、「見出し」で表現しています。

見出しの使い分けについて（ <h1> ～ <h6> ）:「**章立て**」を理解できると分かりやすくなります。

章立ての例

- タイトル
  - 序章
  - 第１章
    - 第１節
    - 第２節
      - １項
        - １目
      - ２項
  - 第２章
  - 最終章

見出しの例

- 誰もが分かる Web 制作（h1）
  - Web 制作をはじめるみなさまへ（h2）
  - 第１章：Web の基礎知識（h2）
    - Web サイト閲覧（h3）
    - Web 制作で必要なもの（h3）
      - テキストエディタ（h4）
        - VisualStudioCord（h5）
      - ブラウザ（h4）
  - 第２章：HTML を学ぼう（h2）
  - 終わりに（h2）

リストについて

順序リスト・順不同リスト

リストとは、箇条書きのことです。
ここでは、【順序リスト】と【順不同リスト】をマークアップしていきます。

- **<ol>** は、**順序通りに行う操作手順や項目数を示す**箇条書き （順序リスト）
- **<ul>** は、**順序が不要**な箇条書き（順不同リスト）
- **
** と **
** の直下には、**
** しか置けない
  （リスト項目内には、入れ子で他タグを入れることが可能）

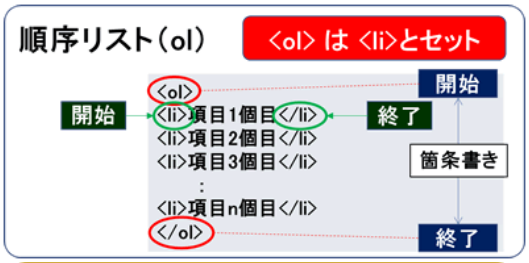
リストの基本設定：

```
<ol>
    <li>項目</li>
    ：  項目数分 追加
</ol>
```

※ 箇条書きの内容に合わせて、**順序リスト**や**順不同リスト**を使い分けましょう。
～ または、
～

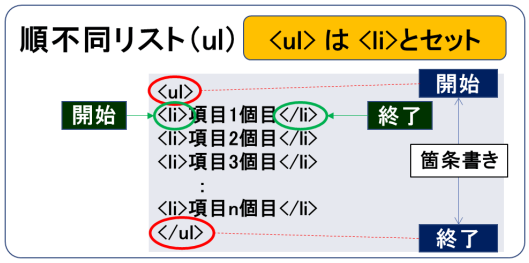
リストの要素タグの設定方法は、順序リストも順不同リストの要素名の違いはありますが、マークアップ方法は同じです。

順序リスト



を、順序リストの開始部分に置く
を、順序リストの最後尾に置く
●
部分が、各リスト項目
ブラウザでは、各項目の前に数字が表示
なお、リスト項目は、複数設置可能

順不同リスト



を、順不同リストの開始部分に置く
を、順不同リストの最後尾に置く
●
部分が、各リスト項目
ブラウザでは、各項目の前にリストマークが表示
なお、リスト項目は、複数設置可能

HTML 記述例（順序リスト）

```
<p>北陸新幹線の駅名（東京～金沢）
</p>
<ol>
<li>東京駅</li>
<li>上野駅</li>
<li>大宮駅</li>
<li>熊谷駅</li>
   :
<li>金沢駅</li>
</ol>
```

HTML 記述例（順不同リスト）

```
<p>新幹線の路線名</p>
<ul>
<li>東海道新幹線</li>
<li>山陽新幹線</li>
<li>東北新幹線</li>
<li>北陸新幹線</li>
<li>上越新幹線</li>
      :
</ul>
```

💻ブラウザ表示例（順序リスト）

北陸新幹線の駅名（東京～金沢）

1．東京駅
2．上野駅
3．大宮駅
4．熊谷駅
：
18．金沢駅

💻ブラウザ表示例（順不同リスト）

新幹線の路線名

- 東海道新幹線
- 山陽新幹線
- 東北新幹線
- 北陸新幹線
- 上越新幹線
：

リストの入れ子

リストの中にさらにリストを入れて表現したい場合には、入れ子構造で記述します。

1．リストを（
～ または ～ の構文一式）2 つ用意する
2．リスト項目のテキストの後に、入れ子の対象となるリストを入れ込む
例：
北陸新幹線■
（■部分に入れ子リスト一式を入れる）

HTML 記述例（順序リスト）

```
<p>新幹線の路線名と駅名</p>
<ul>
<li>東海道新幹線</li>
<li>山陽新幹線</li>
<li>東北新幹線</li>
<li>北陸新幹線
  <ol>
  <li>東京駅</li>
  <li>上野駅</li>
  <li>大宮気</li>
      :
  <li>金沢駅</li>
  </ol></li>
<li>上越新幹線</li>
      :
</ul>
```

💻ブラウザ表示例（順不同リスト）

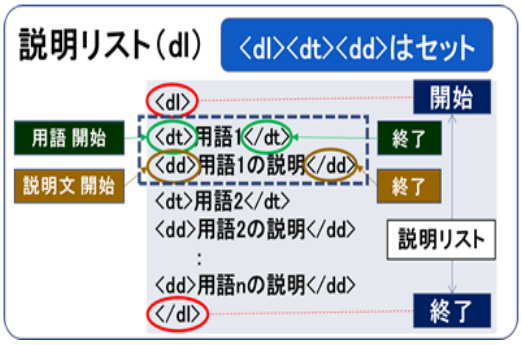
新幹線の路線名と駅名

- 東海道新幹線
- 山陽新幹線
- 東北新幹線
- 北陸新幹線
  1．東京駅
  2．上野駅
  3．大宮駅
  ：
  18．金沢駅
- 上越新幹線
：

説明リスト

説明リストは、用語説明や注釈などで使用します。

<dl> を、説明リストの開始部分に置く
</dl> を、説明リストの最後尾に置く
<dt>●</dt> 部分が、用語
<dd>●</dd> 部分が、説明文の内容
なお、リスト項目は、複数設置可能



<dl> の直下には、<dt></dt> と <dd></dd> しか置けません。

説明リストの基本設定：

```
<dl>
    <dt>用語</dt>
    <dd>説明文</dd>
    ：　用語 項目数分 追加
</dl>
```

HTML 記述例

```
<dl>
<dt>用語</dt>
<dd>用語の説明文</dd>

<dt>曇り</dt>
<dd>空が雲で 9 割以上覆われていて、降水現象がない天気状態のこと</dd>
<dt>晴れ</dt>
<dd>雲が少ないか全く無い天気のこと</dd>
        :
</dl>
```

💻ブラウザ表示例

用語
　　用語の説明文
曇り
　　空が雲で９割以上覆われていて、降水現象がない天気状態のこと
晴れ
　　雲が少ないか全く無い天気のこと
　　：

テーブルについて

テーブルは、表のように縦列と横行の関係性を利用して表現したい場合に使用します。

＜table＞ をテーブルの開始部分に置き、＜/table＞ をテーブルの最後尾に置く
<tr>～各セル～</tr> で囲われた部分が、表内の１行分
<th>●</th> 部分がテーブルの見出し用セルとなり、太文字でセルの中央揃えで表示
<td>●</td> 部分が、テーブルのデータ用セル

テーブルの基本設定：

```
<table>
    <tr>
        <td>データ</td> ～必要 列 の追加～ <td>データ</td>
    </tr>
     :
  必要 行分 の追加
     :
</table>
```

※ 表の**最上段の１行**や**左端の１列**を、**見出し**として利用することがあります。
　　<td>データ</td> → <th>見出し</th>

表の内容に合わせて、縦横の列数や行数を増減して対応します。
見出し項目は、最上段や左列で利用されます。
表の内容が見出しではない場合は、<td> を利用します。

最上段の 1 行が全て見出し項目になる場合は、1 行目の対象となる <tr>～各セル～</tr> を <thead></thead> で囲います。
その場合には、2 行目から最終行までを <tbody></tbody> でひとくくりで囲います。

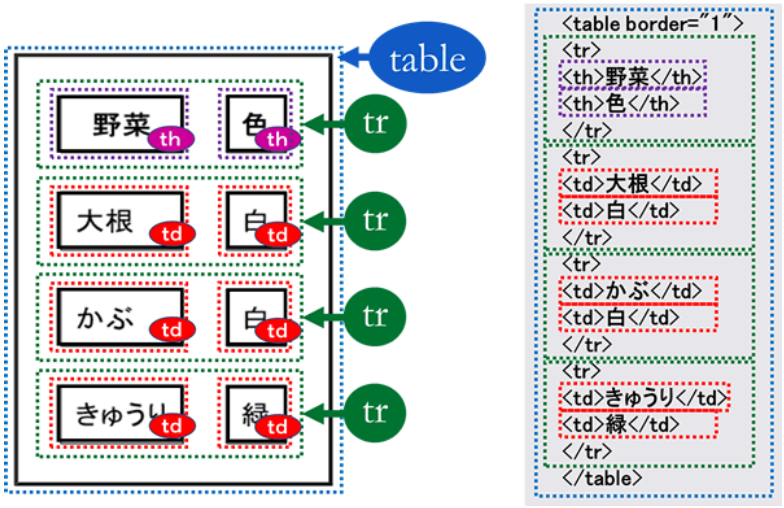
テーブルの構成

セルには、それぞれ**データ**を入れます。（表の内容によって、**見出し <th>** や**データ <td>** として扱います。）
セルが複数並び 1 行分になります。（**1 行分 <tr>～</tr>** 各行ともセルの個数は同数になるように設定）
複数の行を全て含めた全体が、1 つの表です。（ **<table>～</table>** ）

★ 入れ子構造としてのイメージ

テーブルという大きな箱（青い点線枠）の中に、細長い 1 行分の箱（緑色の点線枠）が入ります。【複数行可】
1 行分の細長い箱（緑色の点線枠）の中には、データを入れるセル箱（赤、紫色の点線枠）が入ります。【複数列可】
（各セル項目内には、入れ子で他タグを入れることが可能）

イメージ図を以下に示しました。野菜と色の表は、2 列 4 行の表です。



テーブルの構造のイメージ図とタグ要素関連図

テーブルの基本構文

```
<table border="1">
<tr>
<th>見出し</th>
<th>見出し 1</th>
<th>見出し 2</th>
</tr>
<tr>
<th>見出し a</th>
<td>データ a1</td>
<td>データ a2</td>
</tr>
<tr>
<th>見出し b</th>
<td>データ b1</td>
<td>データ b2</td>
</tr>
</table>
```

3 列 3 行の表

ブラウザ表示例

見出し	見出し 1	見出し 2
見出し a	データ a1	データ a2
見出し b	データ b1	データ b2

HTML 記述例

```
<table border="1">
<tr>
<th>野菜</th>
<th>色</th>
</tr>
<tr>
<td>大根</td>
<td>白</td>
</tr>
<tr>
<td>かぶ</td>
<td>白</td>
</tr>
<tr>
<td>きゅうり</td>
<td>緑</td>
</tr>
</table>
```

2 列 4 行の表

ブラウザ表示例

野菜	色
大根	白
かぶ	白
きゅうり	緑

セルの結合

<th> や **<td>** のタグ内に**セル結合**の属性を追加します。

いくつ分のセル（横方向・縦方向）を結合するのかを指定することでセル結合が可能となります。

結合を行なった場合、不要となるセルを削除する必要があります。削除しないと表示したときにレイアウトが崩れてしまうので、必ずブラウザで確認しましょう。

- **colspan="結合したいセル数"**：横方向のセル結合（右側の複数セルをまとめて１セルに結合）
- **rowspan="結合したいセル数"**：縦方向のセル結合（下側の複数セルをまとめて１セルに結合）

セル結合例の基本のテーブル

セル１	セル２	セル３
セル４	セル５	セル６
セル７	セル８	セル９

✎ セル結合成功の記述例
（セル 2・3／セル 5・8 の結合）

```
<table border="1">
<tr>
  <td>セル 1</td>
  <td colspan="2">セル 2</td>
    <!-- セル 2 と 3 の 2 セルを結合 -->
  <!-- <td>セル 3</td>
    （セル 3 は削除） -->
</tr>
<tr>
  <td>セル 4</td>
  <td rowspan="2">セル 5</td>
    <!-- セル 5 と 8 の 2 セルを結合 -->
  <td>セル 6</td>
</tr>
<tr>
  <td>セル 7</td>
  <!-- <td>セル 8</td>
    （セル 8 は削除） -->
  <td>セル 9</td>
</tr>
</table>
```

✎ セル削除忘れの記述例
（セル 3／セル 8 の未削除）

```
<table border="1">
<tr>
  <td>セル 1</td>
  <td colspan="2">セル 2</td>
    <!-- セル 2 つ分を結合 -->
  <td>セル 3</td>
  <!-- (セル 3 が残る) -->
</tr>
<tr>
  <td>セル 4</td>
  <td rowspan="2">セル 5</td>
    <!-- セル 2 つ分を結合 -->
  <td>セル 6</td>
</tr>
<tr>
  <td>セル 7</td>
  <td>セル 8</td>
    <!-- (セル 8 が残る) -->
  <td>セル 9</td>
</tr>
</table>
```

💻 ブラウザ表示例

<table>
<tr><td>セル 1</td><td colspan="2">セル 2</td></tr>
<tr><td>セル 4</td><td rowspan="2">セル 5</td><td>セル 6</td></tr>
<tr><td>セル 7</td><td>セル 9</td></tr>
</table>

💻 ブラウザ表示例

<table>
<tr><td>セル 1</td><td colspan="2">セル 2</td><td>セル 3</td></tr>
<tr><td>セル 4</td><td rowspan="2">セル 5</td><td>セル 6</td><td></td></tr>
<tr><td>セル 7</td><td>セル 8</td><td>セル 9</td></tr>
</table>

💻ブラウザ表示例

上記のソースの構文内に**コメント**を記述しています。
HTML ソース の記述時に、制作者側としてコメントを付けることで「なぜ、このような記述を行ったのか」などのソース内容を理解しやすくできます。
他にも一時的にブラウザ上で記載内容を非表示にしたい場合などに利用します。

コメントの記述方法：
`<!--` ■ `-->` で囲まれた範囲の■部分にコメントの内容を記述します。
コメントはブラウザ上には表示されません。

コメント記述例：
「 `<!--` セル２と３の２セルを結合 `-->` 」
上記の table のセル結合部分のコメント内容は、ブラウザに表示されないことを確認しましょう。
なお、コメントで記述した内容については、「コメントアウト」ともいいます。

画像表示について

テキストのみの説明だけでなく、画像があるとより内容を伝えやすいサイトになります。
ここでは、画像表示の要素をマークアップしていきます。

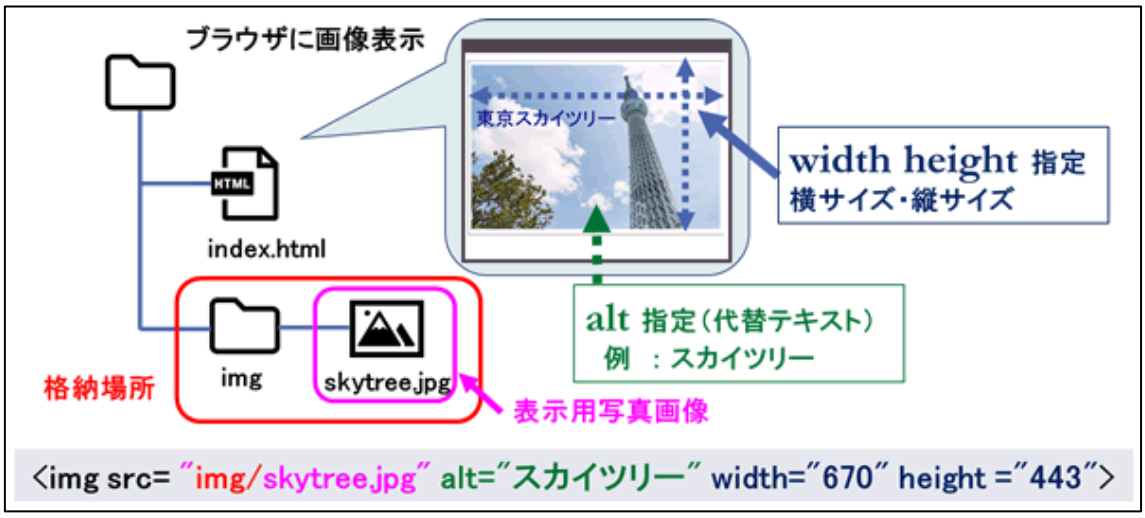
画像表示 基本設定について

img 要素でタグを設定します。img タグは空要素なので終了タグは不要です。
img タグは、段落タグや見出しタグなどの中にも設置が可能です。
画像の表示には、以下の４つの設定が必要です。

1. ブラウザ表示用の「**画像ファイル**」をサイトのフォルダ内に格納
2. 画像ファイルへのパス指定：「**場所の指定**」「**画像ファイル名**」【 src 属性 】
3. 画像ファイルが表示されない場合の「**代替テキストの指定**」:【 alt 属性 】
4. 画像ファイルの「**表示サイズ**」（横サイズ、縦サイズ）【表示サイズ設定】

画像表示の基本設定：

<img src="フォルダ名/画像ファイル名" alt="●●のイメージ" width="■" height="■">



画像の格納場所について

画像を表示するページの HTML から画像ファイルが格納されているフォルダ経路をたどる（パスの指定）

例：img フォルダ内の ▼.gif の画像ファイルを表示　→　src="img/▼.gif"

代替テキスト alt 属性について

画像が表示できない場合に、【×】マークの横に「テキスト」で表示される

表示例：

画像ファイルのサイズの確認方法

1. 画像の入ったフォルダを開き、「表示」タブから「詳細」を選択
2. 「名前」「更新日時」部分を右クリックして「大きさ」を選択
   （もし、表示されない場合は「その他」をクリックしてから「大きさ」を選択）
3. 「670×433」のように画像の大きさが表示されるのでこの数値を利用する（「横サイズ×縦サイズ」で表示）

✏ HTML 記述例

```
1 <p><img src="img/skytree.jpg" alt="スカイツリー" width="300" height ="198"></p>
```

💻 ブラウザ表示例

リンク設定について

リンクとは、異なる Web ページ間や紐付けされたファイルをつなぐ仕組みのことです。
ここでは、リンク設定についてマークアップしていきます。

<a> タグは、Anchor（アンカー）の略で、リンクの出発点と到達点を指定するタグです。

href については（エイチレフ）と読みます。Hypertext Reference（ハイパーテキストリファレンス）の略語です。リンク先を設定するときに使用する属性です。
<a>タグは、「href で指定したファイル（場所）と、今設定しているファイルをつなぐ」役割を行います。

ファイル同士をつなげる設定を行うことで、a タグではさまれたテキストや画像部分は【リンクボタン】となります。クリックすると指定ページへ遷移します。

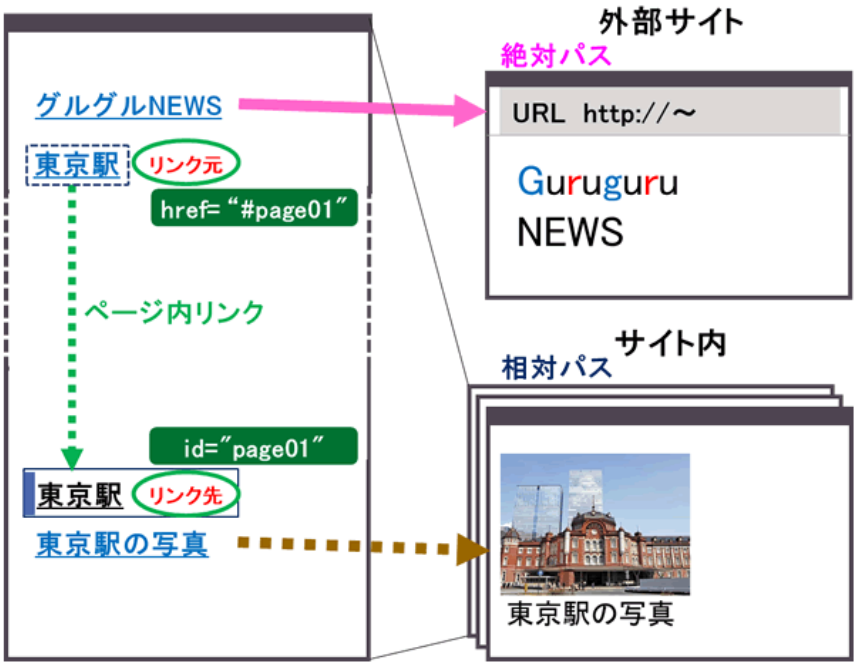
リンクの基本設定：
リンクテキスト

リンクの基本設定について

a タグを利用すれば、リンクしたいページなどにジャンプする設定が可能です。
a タグで囲った部分が、リンク用のボタンとなります。

リンク設定部分では、マウスポインタが指マークに変わります。クリックすることでリンク先へジャンプできます。確認してみましょう。

- リンク設定をしたい部分のテキストや画像などを　　のタグではさむ
- リンク先の設定は　href="リンクパス"　で記述する
  リンクパスの設定方法：URL もしくは、ジャンプ先ファイルまでの経路（相対パス参照）
- ジャンプ先は、以下の 3 種類で設定を行う
  1. **別サイトへのリンク【絶対パス：URL**（http://www～）】
     URL（Uniform Resource Locator）：ホームページを表示するための住所
  2. **サイト内ページへのリンク【相対パス】**
  3. **ページ内リンク**：ページ内の指定した場所へのリンク



リンクの種類のイメージ図

相対パスについて

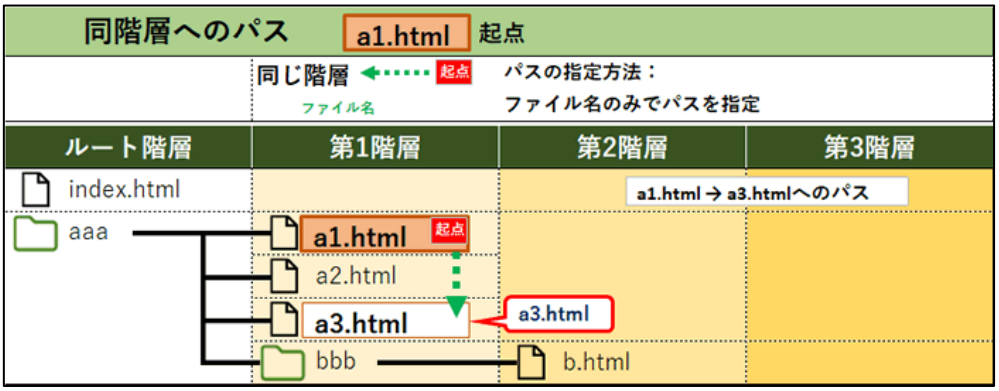
起点となるファイルから表示したいファイルまでの 階層経路 と ファイル名 を確認して「**階層経路 /（区切り）ファイル名**」のように記述します。

- 同階層へのパス
- 下の階層へのパス
- 上の階層へのパス

それぞれの階層のパスについて、設定方法を確認していきましょう。

同階層へのパス

ファイル名のみをそのまま記述

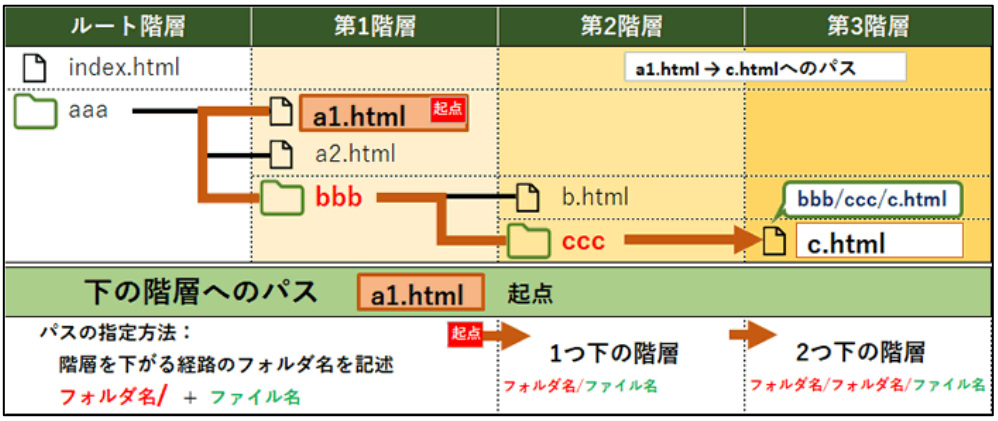


HTML 記述例（同階層へのパス）

```
1 <a href="a3.html">同階層のパス設定</a>
```

下の階層へのパス

経由する**フォルダ名**を「**/（スラッシュ）**」で区切って記述
表示したいファイルに到達したら**ファイル名**を記述

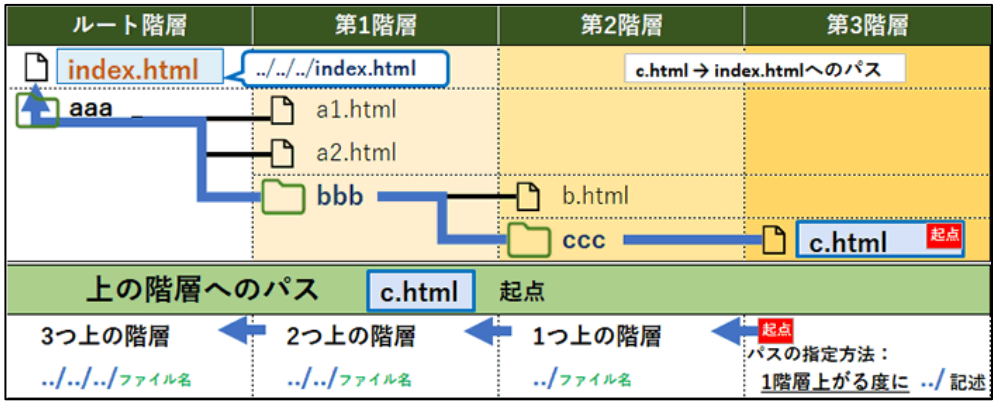


HTML 記述例（下の階層へのパス）

```
1  <a href="bbb/ccc/c.html">下の階層へのパス設定</a>
```

上の階層へのパス

上の階層に上がるたびに「**../**」で区切る（フォルダ名の記述は不要）
表示したいファイルに到達したら**ファイル名**を記述
※ さらに下階層へのパスが必要な場合には、「下の階層へのパス」を参照



HTML 記述例（上の階層へのパス）

```
1  <a href="../../../index.html">上の階層へのパス設定</a>
```

ページの途中部分へリンク設定をすることも可能です。

- **リンクの到達先**となる任意の要素に、固有の識別子（id 名）を指定
  記述例：<h3 **id="id 名"**>リンク先の見出し</h3>

- **リンク元**の設定では、「#」の後に id 名（到達先と同じ名称）を記述
  記述例： 「2 行上の記述例部分」へジャンプ

※ 他ページ内の途中部分にリンク設定する場合は、ファイル名の後に「#id 名」を追記（リンクページ先の到達先部分にも id 設定が必要）

id、class 名について

ここでは、id 名、class 名について説明します。

おもに id、class ともに「特定の要素にだけ CSS（見栄え・デザインの設定）で適用させたい場合」に使用します。（例：文字色を 1 部分だけ赤色表示したいなど）
HTML 文書のタグに記述することで、CSS 記述時にデザインを切り分けることができます。

※ **id 名**および **class 名**の指定については、**半角英数字**で指定しましょう。
※ id については、ページ内リンク設定での利用方法もありましたね。併せて、理解をしましょう。

id について

同じ id 名は 1 ページの中で、**1 度しか**使用できません。

```
<h1 id="sample01">id 名の記述例です。</h1>
<p id="sample01">id 名の記述例です。</p> <!-- × id 名の重複使用は NG -->
<p id="sample02">id 名の記述例です。</p> <!-- ○ id 名が重複していなければ OK！ -->
```

class について

同じ class 名は 1 ページの中で、**何度でも**使用できます。絞り込んだデザインを複数箇所に適用させたい場合に使用します。

```
<p class="sample02">class 名の記述例 1 です。</p>
<p class="sample02">class 名の記述例 2 です。</p>
<p class="sample02">class 名の記述例 3 です。</p> <!-- class 名は重複使用 OK！ -->
```

Web ページを作成する際は、CSS で様々なデザインを設定していきます。id、class の設定は必ず必要になるので押さえておきましょう。

💻注意点

id 名や class 名を命名する際、使用できる文字には条件があります。

- 半角のアルファベットと数字のみ
- 記号は半角の -（ハイフン）と_（アンダースコア）の 2 つ
- 1 文字目は必ずアルファベットにする

上記条件に合っていないと、スタイルが反映されない等が起こります。

```
<p id="sample01">OK です。</p>
<p class="sample_01">OK です。</p>
<p class="2sample">NG です。</p>
<p class="-sample">NG です。</p>
```

3　CSS による文書の装飾

学習内容

- ✓　CSS を学ぼう
- ✓　CSS の役割
- ✓　CSS コーディングルール
- ✓　CSS による見栄え設定
- ✓　CSS 設定と HTML ファイルとの関係
- ✓　CSS での【id、class】の活用
- ✓　主な CSS プロパティについて

見栄え設定の考え方と基本構文を学びましょう。

CSS を学ぼう

ここまで、HTML ファイルの**表示確認（ブラウザ）**と**ファイル作成（エディタ）**の２種類について学んできました。ここからは、上記２つに加え、見栄えとなるデザイン設定用として **CSS ファイルの作成（エディタ）**について学習します。

CSS の構文を理解し設定するスキルを学びましょう。

📖 CSS とは

"Cascading Style Sheets" の略称です。

Web ページのデザイン・レイアウトなどの見栄えを設定するためのコンピュータ言語です。

CSS の役割

CSS でのページデザイン例

見栄えの追加

- 左側は、HTML 構文のみを作成しブラウザで表示したもの
  右側は、CSS で見栄え設定を追加したデザイン例
- 見出しや段落部分のデザインが追加されている
- 文字色や文字サイズ、配置位置、背景色などの変化が確認できる
- 見栄えを整えて、ユーザー（閲覧者）が興味を引くページに仕上げていく

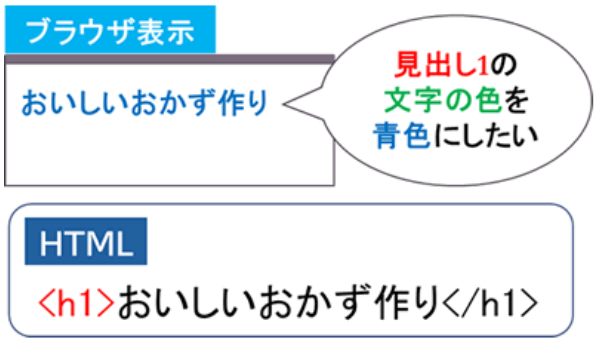
CSS コーディングルール

先頭行には、次の記述が必要です。２行目以降から、スタイルの設定が可能です。

```
@charset "utf-8";
```

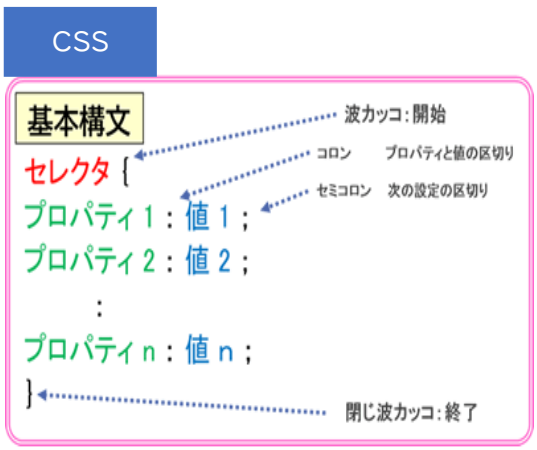
CSS のコーディングを行う場合、「セレクタ」「プロパティ」「値」の３点を以下のように考えます。

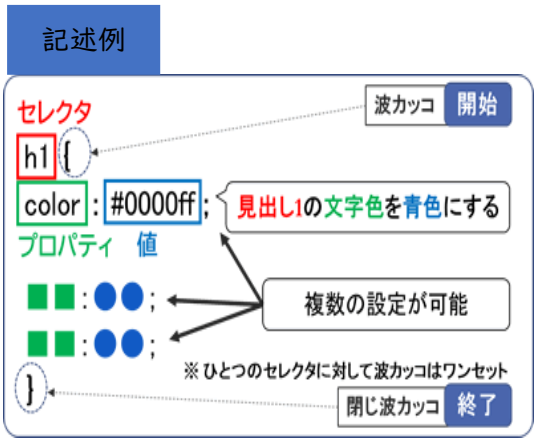
- **コーディングルール**
  - **・どこの**
    → セレクタ：要素など
  - **・なにを**
    → プロパティ：設定の対象
  - **・どうする**
    → 値：設定内容



CSS の基本構文

「セレクタ」記述後の「{ ～ }」の間に「プロパティ」と「値」を記述します。





スタイルデザインの考え方

HTML のみでコーディングされているブラウザ表示と、完成例の見本と見比べ、見栄えの違いを見つけましょう。

見出し 1（h1）の文字色（color）を、青色（#0000ff）にする

✎ CSS 記述

```
@charset "utf-8";
h1 {
color: #0000ff;




}
        :

```

💻ブラウザ表示例

おいしいおかず作り

「見出し 1 の文字色を青色にする」設定方法は分かりましたね。
ここからは、見出し 1 の見栄えについて、更に確認していきます。

- 見出し 1（h1）の**背景色**（background-color）を、水色（#ccffff）にする
- 見出し 1（h1）の**枠線の色**（border-color）を、…どうしたい？
- 見出し 1（h1）の**枠線の太さ**（border-width）を、…どうする？
- 見出し 1（h1）の**枠線の種類**（border-style）を、…どのような線にしたい？
  枠線は、**【色】【太さ】【線の種類】**の 3 つの設定が可能です
- 見出し 1（h1）の見栄えについて、他にもありますか？

見出し１の見栄えを CSS ファイルに追記していきます。
セレクタ・プロパティ・値の関係と CSS の基本構文を活用していきます。
同じセレクタの場合は、「{ ～ }」の間に、追記していきます。

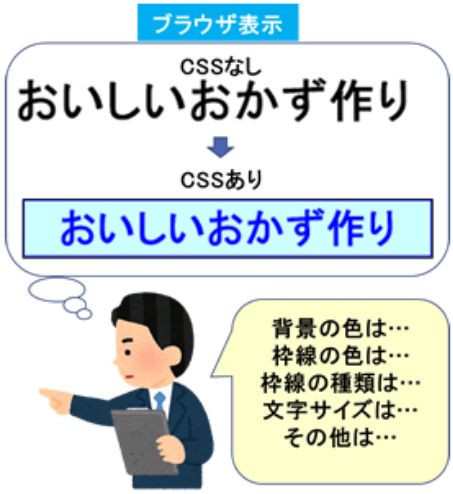
なお、新たなセレクタの設定を行う場合は、セレクタ・プロパティ・値について、新規に基本構文を設定します。

CSS 記述の仕方

```
@charset "utf-8";
h1 {
color: #0000ff;
background-color: #ccffff;
border-color: ●;
border-width: ●;
border-style: ●;
font-size: ●;
/* その他の設定があれば記述 */

}

/* h1 以外の設定があれば記述 */
```



CSS 記述例

```
@charset "utf-8";
h1 {
color: #0000ff;
background-color: #ccffff;
border: 3px solid #0000ff;
font-size: 160%;
text-align: center;
font-weight: bold;
line-height: 160%;
}
```

記述例 詳細

h1 の見栄えは以下のように複数設定
{ } の中に記載すれば、全てセレクタに
反映される

- 文字色を、**青色**（#0000ff）にする
- 背景色を、**水色（#ccffff）**にする
- 枠線を、3px の実線（solid）で青色にする
- 文字の大きさを、160%にする
- 文字の配置位置を、中央揃えにする
- 文字を、**太文字（bold）**にする
- 行間を、160%にする

💻ブラウザ表示例

おいしいおかず作り

CSS による見栄え設定

HTML と CSS ファイルを見比べて設定する手順

スタイルの設定を行う場合、**CSS ファイルのみ**を見て変更するだけでは、設定はできません。

HTML ファイルも同時に見ながら設定を行う必要があります。

では、それぞれのファイルごとに何を設定していくのか見ていきましょう。

注意：HTML ファイル／CSS ファイルの編集後には、保存されていることを確認すること

VSCode：タブの「ファイル名の右側」に「●」が表示された場合は、【保存】が必要

編集後の保存方法

「ファイル」-「保存」、または【Ctrl】＋【S】ボタンを押す

「自動保存」設定であれば、編集ごとに保存される

HTML ファイル側で行うこと

- 内容の**文章構造**を確認して、**タグの要素を決定**する
- ブラウザに表示させる、**全てのテキストなどをタグではさむ**

CSS ファイル側で行うこと

1. ブラウザで表示されている、**見栄えを変更したいテキスト部分を見つける**
2. HTML ソース内の**対象テキストに設定している【要素】や【id／class】**を確認する
   →「**セレクタ**」を見つける
3. 見栄えを変更したい「セレクタ」に対して、**「プロパティ」と「値」を記述**する

ブラウザで行うこと

1．ブラウザで**見栄えが設定通りに表示できているか**を確認する
2．設定内容が反映されていない場合は、編集したファイルを保存しているかを確認する

CSS 設定と HTML ファイルとの関係

スタイルの設定については、２つのファイルの記述方法があります。

1. CSS ファイルに記述する方法（①）
2. HTML ファイルに記述する方法（②、③は予備知識）

1. CSS ファイルに記述する方法

文書の構造を定義する **HTML** と見映えを制御する **CSS** は、**分けることが主流**です。本教材ではこちらの方法で進めていきます。

①CSS ファイル（外部ファイル）に記述する

メリット

- CSS ファイルの設定内容が、サイト内の全 HTML ファイルに反映される
- １つの CSS ファイルで設定を行うので、スタイルの一括編集が可能

デメリット

- CSS ファイル（外部ファイル）の管理が必要
- HTML ファイル内に CSS ファイルへの**紐付け設定**の記述が必要（6 行目）

HTML 記述例（css フォルダ内の style.css への紐付け）

```
1  <!DOCTYPE html>
2  <html lang="ja">
3  <head>
4  <meta charset="utf-8">
5  <title>●タイトル●</title>
6  <link rel="stylesheet" href="css/style.css">
7  </head>
8  <body>
9  <h1>おいしいおかず作り</h1>
10   :
11
```

見栄えを設定したい部分のセレクタを確認しましょう。
「おいしいおかず作り」のテキスト部分の要素は、h1 です。（9 行目）

h1 のスタイル設定を、CSS ファイルで行います。
スタイル設定のみの記述なので、とてもシンプルです。

CSS 記述例

```
1  h1 {
2  color: #0000ff;
3  }
```

２． HTML ファイルに記述する方法（予備知識）

- 要素のタグに直接記述 －－－ ②
- head タグ内の<style>～</style>の中に記述 －－－ ③

②要素のタグに直接記述する

メリット

限定的な１部分に対してのみ見栄えを設定するには便利です。

デメリット

同じ要素に対して全て設定する必要がある場合には、全てのタグを探して設定が必要という手間がかかる点（リニューアル時は、設定した全てのタグの変更が必要）

HTML 記述例

```
1  <h1 style="color:#0000ff">おいしいおかず作り</h1>
```

要素タグへの直接記述設定は、セレクタが、要素そのものとなります。
タグ内に直接 「style="プロパティ ： 値"」 を記述して設定します。

③head タグ内の<style>～</style>の中に記述する

メリット

限定的にそのページだけの見栄えを設定するには便利です。

デメリット

サイト全体のページに対して設定する場合には、サイト内全てのページの head 部分に記述設定する必要があり手間がかかる点。（リニューアル時は、設定した全てのページの変更が必要）

✎ HTML 記述例

```
1  <!DOCTYPE html>
2  <html lang="ja">
3  <head>
4  <meta charset="utf-8">
5  <title>●タイトル●</title>
6
7  <style>
8  h1 {
9  color: #0000ff;
10 }
11 </style>
12
13 </head>
14 <body>
15 <h1>おいしいおかず作り</h1>
16   :
17
```

見栄えを設定したい部分のセレクタを確認しましょう。
「おいしいおかず作り」のテキスト部分の 要素は、h1 です。(15 行目)

head タグ内（3～13 行目）に <style>～</style>（7～11 行目）を記述します。
この style 要素の中に CSS を記述します。(8～10 行目)
CSS の基本構文を以下のように記述してスタイルを設定します。

セレクタ {
プロパティ : 値;
}

CSS での【id、class】の活用

さて、HTML の文書作成で「id、class」に触れたことは覚えているでしょうか？
id、class に対しての CSS 記述も押さえておきましょう。

HTML 記述例

```
<h1 id="sample01">おいしいおかず作り</h1>
  :
<p class="sample02">焼肉</p>
<p class="sample02">焼き鳥</p>
<p class="sample02">トンカツ</p>
<p>カレー</p>
```

CSS 記述例

```
#sample01 { /* id の指定方法。id 名先頭に「#」を付与します。 */
color: #0000ff;
}

.sample02 { /* class の指定方法。class 名先頭に「.」（ドット）を付与します。
*/
color: #ff0000;
```

上記例の場合、id で指定した「おいしいおかず作り」の文字色が**青色（#0000ff）**になります。
一方、class を指定した「焼肉」「焼き鳥」「トンカツ」は文字色が**赤（#ff0000）**になります。
しかしながら、同じ<p>タグでありながら、「カレー」は class 名が付与されていないので適用されません。

要素をセレクタとしたデザイン作成には、限界があります。
id 名や class 名でのセレクタ設定を行うことで、自分でイメージしたデザインをいくらでも作成できるようになります。

主な CSS プロパティについて

CSS のスタイル設定において、よく利用されるプロパティについて学びましょう。

幅と高さを設定するプロパティ

指定したセレクト要素の**幅と高さを設定**する、主なプロパティは以下のとおりです。

box-sizing 以外のプロパティの値は、**単位付きの数値（px、em など）**もしくは**ブラウザ表示時の割合（%）**で指定することができます。また、値を「auto」にすると要素の大きさを自動で調整します。

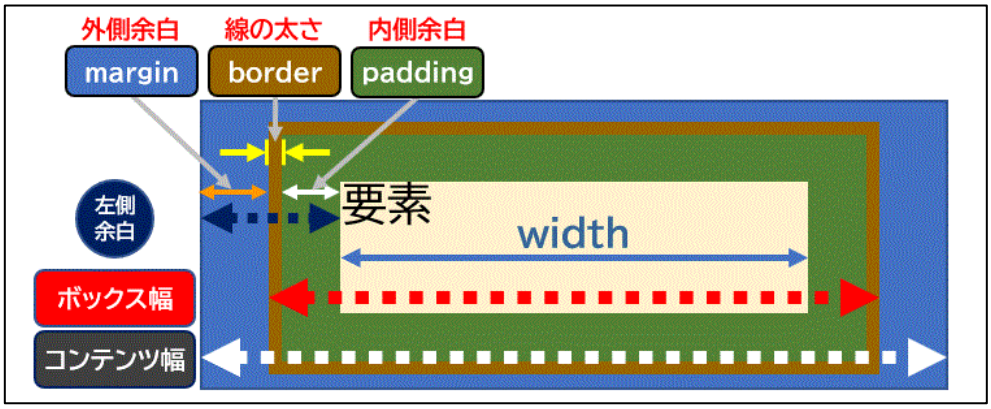
プロパティ名	効果
width	要素の**幅**を指定します。
height	要素の**高さ**を指定します。
min-width	要素の**最小の幅**を指定します。
min-height	要素の**最小の高さ**を指定します。
max-width	要素の**最大の幅**を指定します。
max-height	要素の**最大の高さ**を指定します。
box-sizing	要素の幅（width）と高さ（height）の中に padding と border を含めるかどうかを指定します。 値が「content-box」の場合、padding と border を幅と高さに含めません。初期値です。 値が「border-box」の場合、padding と border を幅と高さに含めます。

余白を設定するプロパティ

要素に余白を設定するプロパティには、padding プロパティと margin プロパティがあります。

padding プロパティは、指定したセレクト要素の**内部余白**を設定するプロパティです。

margin プロパティは、指定したセレクト要素の**外側余白**を設定するプロパティです。



padding プロパティと margin プロパティはともに、上下左右の余白を個別指定または一括指定することができます。

プロパティ名	スタイルを適用する場所	指定できる値の数
padding-top	上の余白	1
padding-right	右の余白	1
padding-bottom	下の余白	1
padding-left	左の余白	1
padding	上下左右の余白	4

※ margin プロパティの場合は、プロパティ名の padding の部分が margin に置き換わります。

padding または margin で一括で指定する場合、値の個数によって余白を設定する場所が変わります。

- １つ … [上下左右]　一括指定　（margin: 10px;）
- ２つ … [上下] [左右] の順で指定　（margin: 10px 20px;）
- ３つ … [上] [左右] [下] の順で指定　（margin: 5px 30px 10px;）
- ４つ … [上] [右] [下] [左] の順で指定　（margin: 5px 30px 10px 20px;）

値は、**単位付きの数値**もしくは**ブラウザ表示時の割合(%)**で指定することができます。また、値を「auto」にすると要素を中央配置に自動設定することができます。値と値の間は半角スペースで区切って指定します。

背景画像に関するプロパティ

background-image プロパティ

background-image プロパティは、**背景に画像を表示させる**プロパティです。
CSS で以下のように記述します。

セレクタ {
background-image: url(画像のパス);
}

画像のパスの部分には、画像への絶対パス（http～ファイル名）または相対パス（../../ファイル名、など）を記述し、どの画像を表示させるかを指定します。

background-size プロパティ

background-size プロパティは、背景に設定した画像の**表示サイズを設定する**プロパティです。
値は、１つまたは２つ指定できます。値を２つ指定した場合は、１つ目が幅、２つ目が高さの指定となります（2 つの値は半角スペースで区切って指定します）。

何も指定していない場合は、そのままのサイズで表示されます（auto と同じ）。

指定できる値	効果
auto	背景画像を元の大きさで表示します。**初期値**です。
contain	背景画像の**縦横比を保持して、要素内にすべて収まる最大サイズ**で表示させます。この値は単独でしか使用できません。
cover	背景画像の**縦横比を保持して、要素内を隙間なく覆う最小サイズ**で表示させます。この値は単独でしか使用できません。
数値+単位	任意の数値と単位で指定します。
数値+%	要素に対しての割合で指定します。

background-position プロパティ

background-position プロパティは、背景に設定した画像を**表示させる位置を設定する**プロパティです。

値は、半角スペースで区切って 2 つ指定するのが基本です。数値またはパーセンテージ（%）で指定した場合は、1 つ目が横方向の位置、2 つ目が縦方向の位置となります。1 つしか指定しなかった場合は、2 つ目の値に「center」が指定されたものとして処理されます。初期値は 0%、0%です。

指定できる値	効果
数値+単位	要素の左上からの位置を、任意の数値と単位で指定します。
数値+%	要素の左上からの位置を、要素に対しての割合で指定します。
top	要素の上端に表示します。縦方向の 0%と同じです。
bottom	要素の下端に表示します。縦方向の 100%と同じです。
left	要素の左端に表示します。横方向の 0%と同じです。
right	要素の右端に表示します。横方向の 100%と同じです。
center	要素の中央に表示させます。縦方向の 50%／横方向の 50%と同じです。

background-repeat プロパティ

background-repeat プロパティは、背景に設定した画像を**縦または横に繰り返して表示させるのかどうかを設定する**プロパティです。

指定できる値	効果
repeat	背景画像を繰り返し表示させます。**初期値**です。
no-repeat	背景画像を繰り返さず、１つだけ表示させます。
repeat-x	背景画像を X 方向、つまり横方向にだけ繰り返して表示させます。
repeat-y	背景画像を Y 方向、つまり縦方向にだけ繰り返して表示させます。

全てを覚える必要はありません。実際に作成をしながら少しずつ身に付けていきましょう。

CSS プロパティは、上記に挙げた以外にもまだまだあります。時間があるときにインターネット等で検索してみましょう。

4　CSS セレクタについて

学習内容

- ✓　セレクタの活用について
- ✓　タイプ（要素型）セレクタ
- ✓　id、class セレクタ
- ✓　疑似クラスセレクタ
- ✓　疑似要素セレクタ
- ✓　属性セレクタ
- ✓　その他のセレクタ

CSS セレクタの設定を活用して、デザインの幅の広がりを実感しましょう。

セレクタの活用について

CSS は、HTML に記述された要素に対してデザインを指定します。
CSS のデザイン設定をどの HTML 要素に適用させるかを指定するのに用いられるのが「**CSS セレクタ**」です。

通常は、h1 や p などの要素をセレクタで指定すると、**指定した要素全てに反映**されます。
CSS セレクタを使用することで、**細かく個別に指定**することができ、デザインの幅が広がります。

頻繁に使用する CSS セレクタを押さえておきましょう。

※ 表内の「使用例」の記述について
　p 要素、文字色のプロパティなどを一例として挙げています。
　他の「要素」「プロパティ」と「値」でも設定可能ですので、試してみましょう。

タイプ（要素型）セレクタ

セレクタの書式	スタイルを適用する対象	使用例
* （ユニバーサルセレクタ）	すべての要素	* {color: blue;}
要素名	要素名で指定した要素	p {color: blue;}

id、class セレクタ

HTML に記述している「ある 1 つの要素」について、見栄え（デザイン）を使い分けたい場合には、class 属性や id 属性を利用することで対応が可能です。

class 属性や id 属性のみでも、CSS 設定を行うことが可能です。
ただし、CSS の優先度によっては反映されない場合がありますので、セレクタの設定方法の理解が必要です。

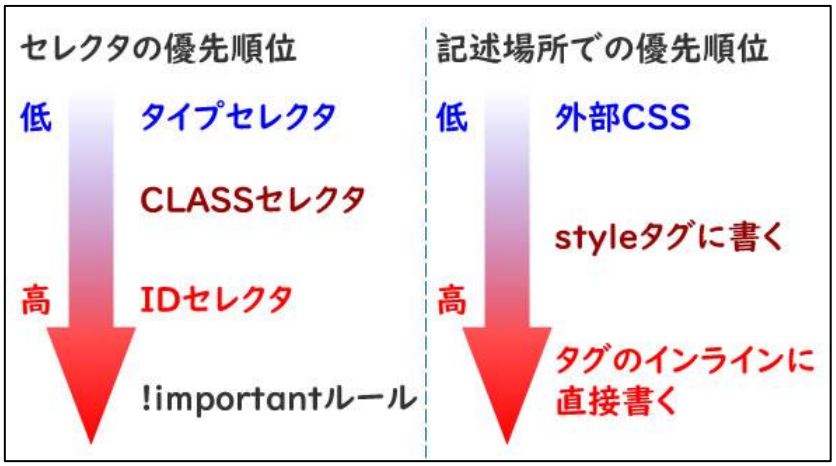
※ 要素に続く class 名・id 名の前には、**半角スペースは入れません。**
※ id 設定例：レイアウトに関する header、nav、footer などで利用する（ページ内で 1 度しか利用できないため）。

セレクタの書式	スタイルを適用する対象	使用例
.class 名	class 名を付けた全ての要素	.sample {color: blue;}
#id 名	id 名を付けた全ての要素	#sample {color: blue;}
要素名.class 名	class 名を付けた特定の要素	p.sample {color: blue;}
要素名#id 名	id 名を付けた特定の要素	div#sample {color: blue;}

CSS の優先順位について＜超重要！＞

基本的に CSS は、上から下に記述順で適用されます。同じセレクタであれば、CSS ファイル内で後ろ（より下に）に記述された指定が優先されるということです。

基本的な優先順位は、以下の図にあるとおりです。例えば、"h1 {color: red;}" と記述するよりも、"h1#sample {color: red;}" の方が優先されます。これは、記述する CSS ファイルが同じであれば、記述の後先は関係ないと言えるでしょう。
（補足ですが、h1 は書かずに"#sample"でも OK です。比較のため記述しています。）



なお、!important を指定したルールは何よりも最優先されます。便利といえば便利ですが、乱用すると CSS の破綻に繋がる恐れもあるので、計画的に利用しましょう。
まずは、!important を使用せずに組み立てることをおすすめします。

記述例は以下です（プロパティのあとに半角スペースを入れるのを忘れないようにしましょう）。

CSS 記述例

```
1 h1 {
2 color: #0000ff !important;
3 }
```

💻参考：セレクタ指定による優先順位

後続で同じセレクタに上書き設定を行う場合には、点数が高くないと反映されません。優先順位は、以下のようになります。

- **優先度高　id ＞ class ＞ 要素 ＞ *　優先度低**
  id 指定が一番高く、続いて class 指定、要素指定の順に優先される
- セレクタが詳細に宣言されていると優先順位が高くなる
  セレクタ例：.main p.font_type01 { … }
  （main クラス内の font_type01 クラスが付いた p 要素をセレクタとした設定）
  優先度点数：21 点（表参照）

CSS を設定しているのに反映されない場合、以下の表を参考に確認していきましょう。（覚える必要はありません！）
表内の「CSS セレクタ」以降の内容については、次のコンテンツで学習予定です。

注意：
桁の小さい 【class セレクタ】をたくさん集めて合計点数を増やした設定をしても、桁の大きい設定【id セレクタ】よりも、優先順位が高くなることはありません。

セレクタの指定	点数	例
タグ内に style 属性を追加指定 ※ 今回推奨しません	1,000 点	style="color: blue;"
id セレクタ	100 点	#sample01
class セレクタ	10 点	.sample02
属性セレクタ	10 点	a[href="career.com"]
疑似要素	1 点	::before など
要素名	1 点	p など
ユニバーサルセレクタ	0 点	*

擬似クラスセレクタ

擬似クラスとは、指定の要素が**特定の状態**である場合にスタイルを適用させるセレクタです。

文章構造の範囲外となる情報を利用して、スタイルを変化させることが可能です。

🔎 要素の**ある特定の状態**について設定

セレクタの書式	スタイルを適用する対象	使用例
要素名:link	未訪問のリンク	a:link {color: blue;}
要素名:visited	訪問済のリンク	a:visited {color: blue;}
要素名:hover	マウスカーソルが乗っている要素	a:hover {color: blue;}
要素名:active	クリック中の要素	a:active {color: blue;}

[hover]や[active]などは、マウス操作時のスタイル設定となります。

※ a 要素への擬似クラスセレクタの設定は、表の順序で設定すること

「なぜ、表の順序で設定が必要なのか」は、考えてみましょう。

ヒント：CSS は後から記述されると上書きされます。

🔎 要素内の**ある部分**について設定

セレクタの書式	スタイルを適用する対象	使用例
要素名:focus	テキストボックスにフォーカスさせたい要素 フォームページなどでの対応（学習範囲外）	input:focus {background: blue;}
要素名:first-child	要素内の最初の子要素	li:first-child {color: blue;}

要素名:last-child	要素内の最後の子要素	li:last-child {color: blue;}
要素名:nth-child(n)	親要素内の n 番目の要素	p:nth-child(2) {color: blue;} ※親要素配下の、2 番目の子要素にあたる要素の文字色が青になる。
	奇数番目の要素	p:nth-child(2n+1) {color: blue;} or p:nth-child(odd) {color: blue;}
	偶数番目の要素	p:nth-child(2n) {color: blue;} or p:nth-child(even) {color: blue;}
要素名:nth-of-type(n)	親要素内の n 番目の指定の要素	p:nth-of-type(2) {color: blue;} ※配下の 2 番目の p 要素の文字色が青になる。
	奇数番目の指定の要素	p:nth-of-type(2n+1) {color: blue;} or p:nth-of-type(odd) {color: blue;}
	偶数番目の指定の要素	p:nth-of-type(2n) {color: blue;} or p:nth-of-type(even) {color: blue;}

💻ポイント

「nth-child」と「nth-of-type」の違いは「要素の数え方」にあります。
例えば、親要素 ～ 内の子要素として、先頭に<h4>がある場合、p:nth-child(2)、p:nth-of-type(2) の例で考えてみましょう。

- p:nth-child(2) … <h4>は 1 つ目としてカウントされる
- p:nth-of-type(2) … （p 要素ではないので）<h4>はカウントされない

HTML 記述例

```
<div>
<h4>1 つ目の h4 要素</h4>
<p>1 つ目の p 要素</p>
<p>2 つ目の p 要素</p>
<p>3 つ目の p 要素</p>
</div>
```

nth-child : CSS 記述例

```
p:nth-child(2) {
color: blue;
}
```

nth-of-type : CSS 記述例

```
p:nth-of-type(2) {
color: blue;
}
```

nth-child : ブラウザ表示例

1 つ目の h4 要素
1 つ目の p 要素
2 つ目の p 要素
3 つ目の p 要素

親要素 ～ 内の 2 番目の**子要素**の文字色が青になる。

nth-of-type : ブラウザ表示

1 つ目の h4 要素
1 つ目の p 要素
2 つ目の p 要素
3 つ目の p 要素

親要素 ～ 内の 2 番目の**p 要素**の文字色が青になる。

擬似要素セレクタ

擬似要素とは、CSS への設定のみで HTML に記述しなくとも以下のような装飾が可能なセレクタです。

- 対象となる要素の一部を指定して装飾を適用させる
- 対象の要素に擬似的に要素を追加して装飾を適用させる

擬似要素の設定を行うことで、装飾忘れのミスを防げるメリットがあります。
※ 要素名に続く**半角コロン**は、**２つ入れる**のを忘れないようにしましょう。

セレクタの書式	スタイルを適用する対象	使用例
要素名::first-line	要素の最初の一行	p::first-line {color: blue;}
要素名::first-letter	要素の最初の一文字	p::first-letter {color: blue;}
要素名::before	要素の直前	blockquote::before {content: "『";}
要素名::after	要素の直後	blockquote::after {content: "』";}

属性セレクタ

属性セレクタは、**特定の属性**が入っている要素に対してだけ装飾を変えることが可能なセレクタです。

セレクタの書式	スタイルを適用する対象	使用例
要素名[属性名]	特定の属性を持つ要素	a[href] {color: blue;}
要素名[属性名="属性値"]	特定の属性値を持つ要素	a[target="_blank"] {color: blue;}

その他のセレクタ

セレクタの設定は、HTML 構造の「親要素」と「子要素」関係が分かると理解がしやすくなります。

「親子・兄弟関係」を活用した、限定的なスタイルの適用範囲の設定が可能になります。

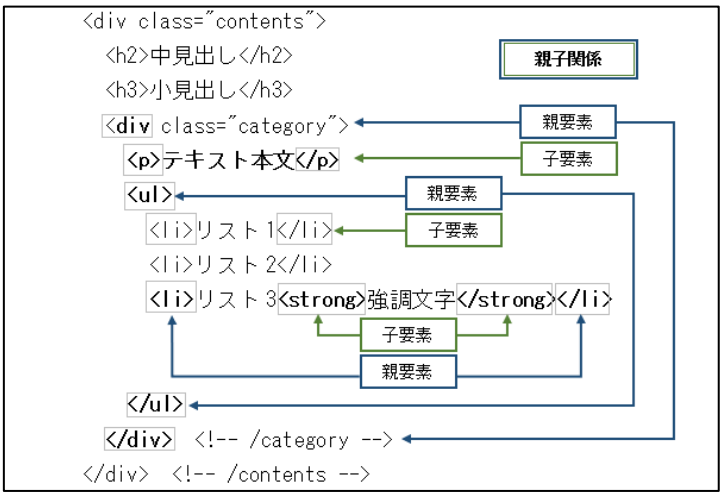
セレクタの書式	スタイルを適用する対象	使用例
セレクタ,セレクタ (セレクタ間はカンマで区切る)	「,」区切りで並べた複数セレクタ	h1,h2 {color: blue;} h1 と h2 要素に同じスタイルを設定可能
セレクタ■セレクタ ■ ＝ 半角スペースです： (セレクタ間は半角スペースで区切る)	下の階層の子孫要素	p strong {color: blue;} p 要素の中にある strong 要素にのみスタイルを設定可能 その他：class 設定をした div 要素内の p 要素（子要素）などに設定 ※親子関係の要素に設定する場合に有効
セレクタ>セレクタ (セレクタ間は>で区切る)	直下の階層の子要素にのみ	p>strong {color: blue;} p 要素の中にある strong 要素にのみスタイルを設定可能 その他：class 設定をした要素内の子要素のみの設定に有効 ※親子関係の要素に設定する場合に有効
セレクタ+セレクタ (セレクタ間は+で区切る)	同じ階層のすぐ後の要素にのみ	h1+p {color: blue;} h1 要素のすぐ後にある p 要素にのみスタイルを設定可能 ※親子・兄弟関係の要素に設定する場合に有効

参考：親子・兄弟関係

HTML 構文の「親子関係」

ブロック要素の中に包括されて記述する要素同士の関係は、**親子関係**になります。
以下の例を参考に、イメージをつかみましょう。

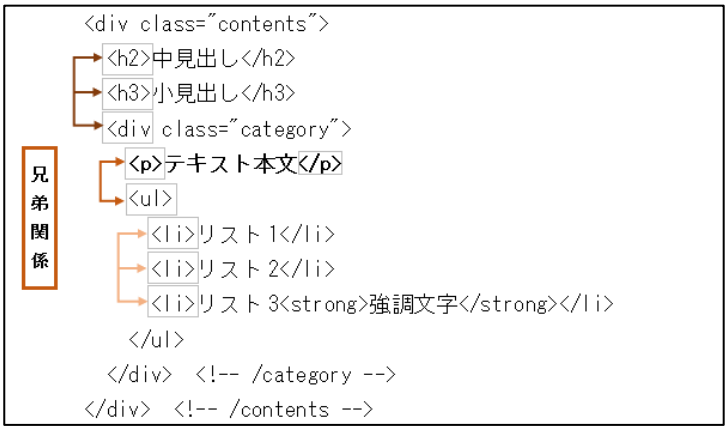
- の中の <p>
- の中の
- の中の



HTML 構文の「兄弟関係」

同階層に記述する要素同士の関係は、**兄弟関係**になります。

- 直下の <h2> と <h3> と
- 直下の <p> と
- 直下の <li> 3 つ



5　CSS の複数設定

学習内容

- ✓　複数の class 設定
- ✓　複数の class を 1 つのタグに指定
- ✓　入れ子構造を含む複数 class 設定

HTML 側で複数の class 設定ができることを学びましょう。

複数の class 設定

CSS を設定すると、たった 1 つのセレクタでもたくさんのスタイル設定を行うことになります。結果、1 ページ分のスタイル設定だけでも記述は多くなります。

例えばですが、あちらこちらに【文字色は青色】という同じ設定を繰り返し記述していませんか。**セレクタごとにスタイルを上手に分けて、class を複数設定**する方法を学習しましょう。

複数の class を 1 つのタグに指定

HTML では、1 つのタグ要素に class 属性を複数指定する事ができます。
1 つの要素に class を複数指定することで、各セレクタで設定した見栄えを適宜利用することが可能になります。

例えば、「文字設定」「余白設定」「枠線設定」などを別の要素に設定したい場合に、使い回しをする事ができます。ただし、HTML の記述が複雑になる場合があるので、見やすく、メンテナンスしやすいよう、工夫が必要です。

CSS 記述例

```
@charset "utf-8";
.fot_type01 {
color: #0000ff;
font-size: 150%;
}
.mar_type01 {
padding: 10px;
}
.bgc_type01 {
background-color: #bbb;
}
.bdc_type01 {
border: 3px solid #0000ff;
}
```

記述例 詳細

左記のように複数のスタイル設定があるとします。
それぞれの設定は、以下のとおりです。

- .fot_type01
  文字色を**青色（#0000ff）**
  文字サイズを 150%
- ..mar_type01
  余白を 10px
- .bgc_type01
  背景色を灰色（#bbb）
- .bdc_type01
  枠線を、3px の実線（solid）で**青色**

別々に設定したスタイルを、**HTML の class 指定部分**で、以下の記述のように**半角スペースで区切って**設定しましょう。

HTML 記述例

```
<p class="fot_type01 mar_type01 bgc_type01 bdc_type01">おいしいおかず作り</p>
```

ブラウザ表示例

おいしいおかず作り

入れ子構造を含む複数 class 設定

親要素に適用したスタイルは、子要素にも適用されます。これを**【スタイルの継承】**と言います。HTML の親子関係の設定と合わせて、スタイルの継承を利用しながら CSS を作成していくのがポイントです。

以下の例では、親要素で設定したスタイル（余白、背景色、枠線など）は、継承されて子要素にも適用されているのが確認できます。
例：親要素 に適用したスタイルが、子要素 <p> にも適用

CSS 記述例

```
@charset "utf-8";
.fot_type00 {
color: #ff0000;
font-size: 100%;
}
.mar_type01 {
padding: 10px;
}
.bgc_type01 {
background-color: #bbb;
}
.bdc_type01 {
border: 3px solid #0000ff;
}

.fot_type01 {
color: #0000ff;
font-size: 150%;
}
.fot_type02 {
color: #00ff00;
font-size: 80%;
}
```

記述例 詳細

左記のように親要素・子要素に対して、
複数のスタイルを設定します。
それぞれの設定は、以下のとおりです。

- **親要素：**
  - fot_type00
    文字色を**赤色（#ff0000）**
    文字サイズを 100%
  - .mar_type01
    余白を 10px
  - .bgc_type01
    背景色を灰色（#bbb）
  - bdc_type01
  - 枠線を、3px の実線（solid）で
    青色（#0000ff）
- **子要素：**
  - .fot_type01
    文字色を**青色**、サイズを 150%
  - .fot_type02
    文字色を**緑色（#00ff00）**
    文字サイズを 80%

子要素で「文字色」「文字サイズ」を
class 指定して上書きします。
基本設定スタイルに対して、**子要素の設定**から見栄えの変更が可能です。

HTML 記述例

```
<div class="bgc_type01 bdc_type01 mar_type01 fot_type00">
<p>おいしいおかず作り_00</p>
<p class="fot_type01">おいしいおかず作り_01</p>
<p class="fot_type02">おいしいおかず作り_02</p>
</div>
```

ブラウザ表示例

おいしいおかず作り_00

おいしいおかず作り_01

おいしいおかず作り_02

6　CSS レイアウトの基本

学習内容

✓　レイアウトの種類
✓　position レイアウト

　パソコンだけでなく、スマートフォンやタブレット端末のように様々なデバイスに対応した Web レイアウトを学びましょう

レイアウトの種類

　現在のインターネット閲覧環境は、パソコンだけでなく、スマートフォンやタブレット端末のように様々なデバイスに対応した Web レイアウトが求められています。
　ユーザーの Web ページを閲覧する方法が多様化しているため、制作側も、どのようなデバイスで閲覧しても見やすいことを重視しなければなりません。
　どのようなレイアウトがあるのか、大まかに押さえておきましょう。

固定レイアウト

　おもにピクセルベースで設計されており、**特定の幅でコンテンツを表示させる**レイアウト手法です。基本的に最初に起こしたデザインを、そのまま実現できるため、実現難易度は比較的容易と言えるでしょう。

メリット

　上記で挙げたように、デザインから設計までででズレが生じることが少ないため、比較的容易に実現できます。

デメリット

多種多様なデバイスに対応しづらい点が挙げられます。大型ディスプレイであれば、余白が目立ち、コンテンツ自体が窮屈に感じられたり、スマートフォンのような狭い画面だと、そもそも設計自体を見直す必要が出てきたりします。古めかしい印象を与えがちになることからも、固定幅自体を分けた方が良いでしょう。

参考：Yahoo Japan

可変レイアウト

ウィンドウ幅が変わると、それに合わせて**コンテンツの幅が伸縮する**レイアウト手法です。

ユーザーの閲覧環境に合わせて、レイアウトが変わるため、様々なデバイスで負担なく閲覧することが可能となります。レスポンシブデザイン、グリッドデザインが代表格です。

レスポンシブデザイン

デバイスの画面サイズに依存せず、柔軟に Web サイトを構築する手法です。デバイスのウィンドウ幅に「responsive(良く反応)」して、見やすい表示に切り替える仕組みをもつデザインを指します。実現には、CSS のメディアクエリを使用して、画面幅によってデザインを切り替えます。

メリット

パソコン用サイトとモバイル用サイトを別々に作る必要がありません。共通の Web サイトを 1 つ構築して、URL や HTML も、ページごとに 1 種類ずつ用意すれば OK です。ゆえにコンテンツを更新する場合は、HTML 1 つの更新で済むことも魅力です。

デメリット

HTML が 1 つなのでスマホとパソコンでコンテンツの順番を変えるなど、大幅にデザインを変えることが難しいです。また、CSS の記述が複雑になりやすいです。

参考：OKI ワークウェル

グリッドデザイン

別名、グリッドシステムやグリッドレイアウトとも呼ばれます。格子状のように配置し、コンテンツを整列させます。Web サイト内のコンテンツがグリッドレイアウトによりデザインされていると、ユーザーに対して情報がすっきりと整理された印象を与えることができます。

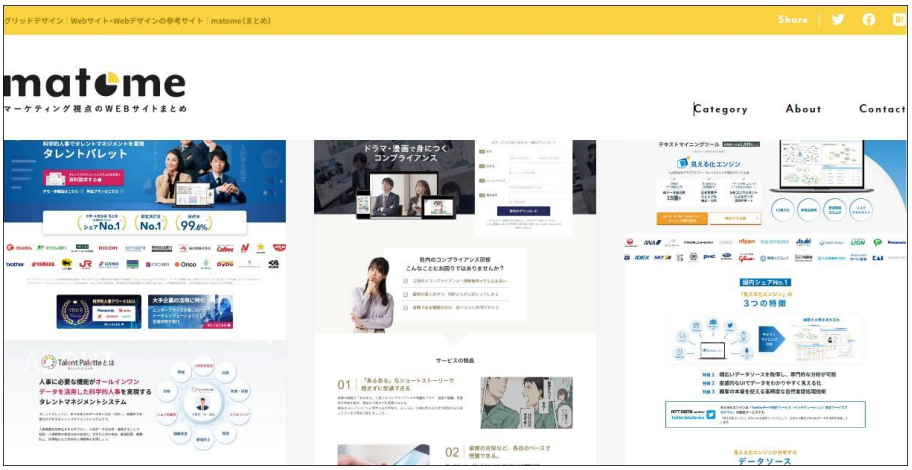
メリット

整列された配置のため、Web サイトに統一感が出ます。ブラウザが表示できる幅によってレイアウトが変化するため、閲覧状況に合わせて臨機応変に対応できます。

デメリット

マルチカラム同様に、デザインによってはメインコンテンツに集中しづらいことが挙げられます。『統一感 ＝ 重要な情報』とのメリハリを考慮しないと埋もれてしまう可能性があります。

参考：matome (Web サイトまとめ)



フリーレイアウト

一般的に**固定した幅で自由に**コンテンツを配置していくパターンです。多くの情報を好きなようにレイアウトできます。position プロパティを使用することが多いです。

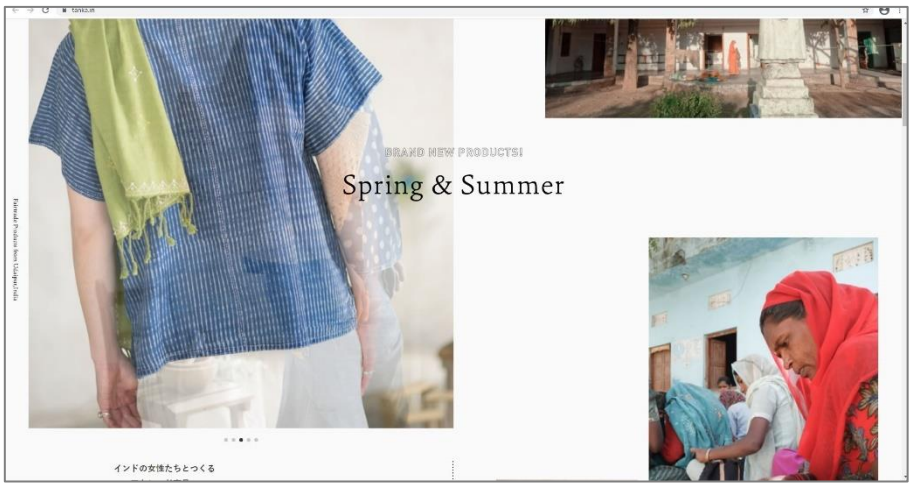
メリット

オリジナリティを意識したレイアウトを作り出すことができます。

デメリット

ユーザーのデバイス環境によってページが見づらくなってしまうことが最大のデメリットです。かつ、自由度が高いため、よりユーザー目線を失わずに制作することが求められます。

参考：tanka



上記のようなデザインレイアウトを実現するためには、CSS による「**position**」「**flex-box**」等を使いこなすスキルが必要となります。次の学習で身に付けていきましょう。

position レイアウト

CSS でレイアウト（配置）を設定します。
例えば、コンテンツを指定の位置に表示させることやバナーやボタンの固定表示などです。

ここでは、基本的なレイアウト手法の 1 つである「**position**」を学習します。
position プロパティを使いこなすことで、HTML コード上の記述順序に依存せずに自由なレイアウトが可能になります。

position プロパティの値	効果
static スタティック	特に配置方法を指定しません。この値のときには、top、bottom、left、right は適用されません。**初期値**です。
relative リラティブ	**相対位置**への配置となります。position プロパティで static を指定した場合に表示される位置が基準位置となります。relative のみ float と併用可能です。
absolute アブソリュート	**絶対位置**への配置となります。親ボックスに position プロパティの static 以外の値が指定されている場合に、親ボックスの左上が基準位置となります。親ボックスに static 以外の値が指定されていない場合には、ウィンドウ全体の左上が基準位置となります。
fixed フィックスド	絶対位置への配置となるのは absolute と同じですが、起点がブラウザの表示画面領域全体となります。スクロールしても位置が固定されたままとなり、**常時表示**されます。
sticky スティッキー	fixed は表示画面領域から見た位置に固定されるのに対し、sticky は**親・兄弟要素を起点として固定表示**されます。つまり、親・兄弟要素が無ければ反映されません。また、一部のブラウザでは対応されていないので、注意が必要です。

position 指定時と併せて使用する位置指定

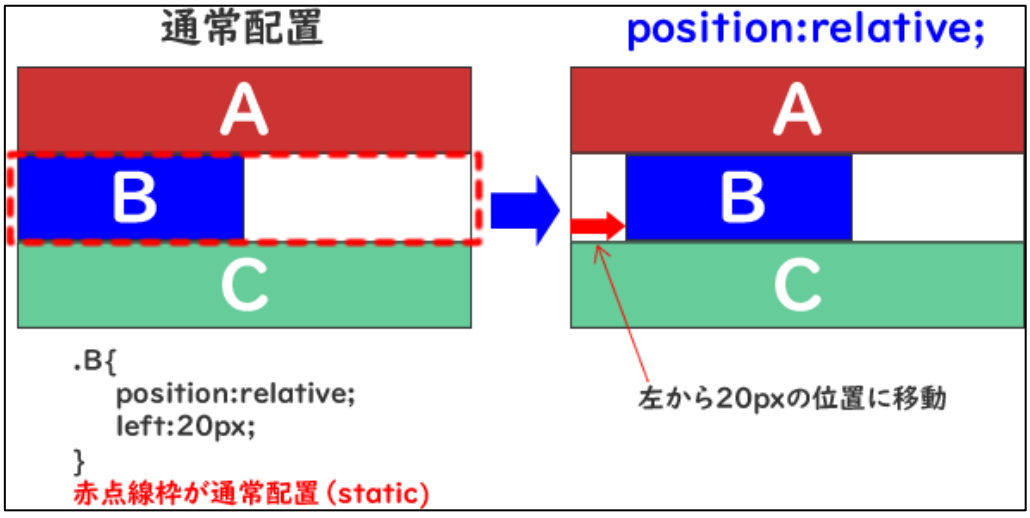
position 指定した場合、基準となる要素から「**どの位置に移動させるか**」を指定する必要がある場合が主です。その場合に使用するプロパティが **top**、**left**、**bottom**、**right** です。

top、left、bottom、right プロパティは単体で使用することは無く、position プロパティと併せて使用します。

位置指定のプロパティの値として使用できるのは、**px** や **em** または基準となる親要素の幅・高さに対する割合（**%**）などがあります。

relative での配置

position: relative; での配置は**相対位置指定**となります。相対位置指定とは、元々配置されている位置（position: static;）を基準にして指定する方法のことです。冒頭の表でも述べましたが、position: static; は CSS で特に指定が無ければ、初期値として設定されているので目にする機会は少ないかもしれません。



【relative イメージ】

以下のような HTML、CSS の記述があった場合、どのように表示されるのでしょうか。

position プロパティが記載されている場合とされていない場合を比較して、どのように表示位置が変更されるのか確認してみましょう。

✎ HTML 記述例

```
<div id="wrap">
	<div id="header">
		ヘッダー部分です。<br>
		ヘッダー部分です。<br>
		ヘッダー部分です。
	</div>

<div id="main">
	<div id="area01">
		コンテンツAです。コンテンツAです。コンテンツAです。コンテンツAです。コンテンツAです。コンテンツAです。コンテンツAです。コンテンツAです。コンテンツAです。コンテンツAです。コンテンツAです。コンテンツAです。コンテンツAです。コンテンツAです。コンテンツAです。コンテンツAです。コンテンツAです。コンテンツAです。コンテンツAです。コンテンツAです。コンテンツAです。コンテンツAです。コンテンツAです。コンテンツAです。コンテンツAです。コンテンツAです。コンテンツAです。コンテンツAです。コンテンツAです。コンテンツAです。コンテンツAです。コンテンツAです。コンテンツAです。コンテンツAです。コンテンツAです。コンテンツAです。コンテンツAです。コンテンツAです。コンテンツAです。コンテンツ
Aです。コンテンツAです。コンテンツAです。コンテンツAです。コンテンツAです。コンテンツ
Aです。
	</div>
</div><!-- /main -->

	<div id="footer">
		フッター部分です。<br>
		フッター部分です。<br>
		フッター部分です。
```

✎ CSS 記述例(position 指定なし)

```
@charset "UTF-8";
* {
margin: 0;
padding: 0;
}

#wrap {
width: 800px;
margin: 30px auto;
background-color: #ccc;
}

#main {
background-color: yellow;
}

#header {
background-color: #C00000;
}

#area01 {
width: 250px;
background-color: #92D050;
}

#footer {
background-color: #CC66FF;
}
```

💻 ブラウザ表示例（position 指定なし）

「コンテンツ A です。」の緑ボックスを移動させてみましょう。

CSS の「#area01」セレクタに、position: relative; **を追加**し、top、left、right、bottom プロパティを使って**位置を指定**します。

✏ CSS 記述例

```
#area01 {
position: relative; /*追加行*/
left: 20px; /*追加行*/
width: 250px;
background-color: #92D050;
}
```

※ 記載していないセレクタについては、変更なし。

💻 ブラウザ表示例

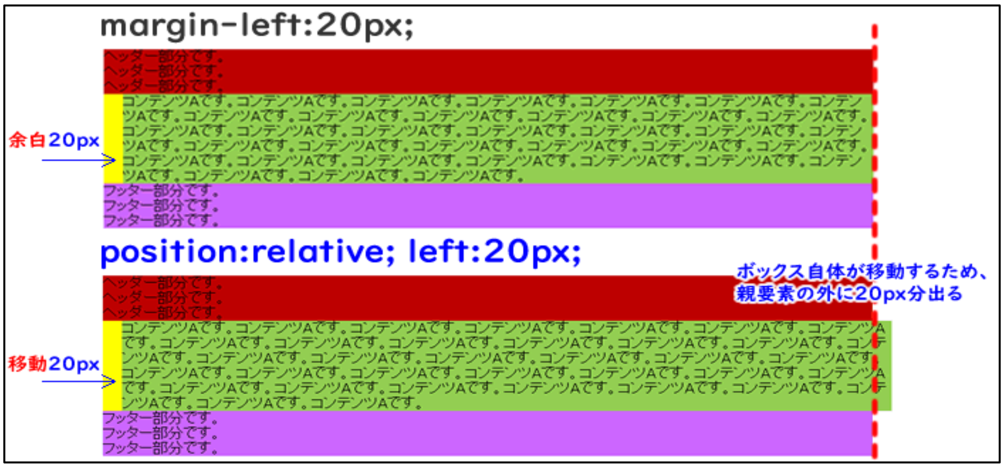
「コンテンツ A です。」の緑ボックスが 20px 右に移動しました。左側に 20px 分の黄色背景が見えるようになっているのが分かります。

left 等の値はマイナスの値も指定できます。20px の値を色々変更して、動作を確認してみましょう。

ここで、1 つ疑問に思うことがあるかもしれません。「**margin-left:20px;**」と何が違うのかと。

margin はボックスの外側余白を制御します。一方、position での位置指定はボックスそのものを動かし配置します。

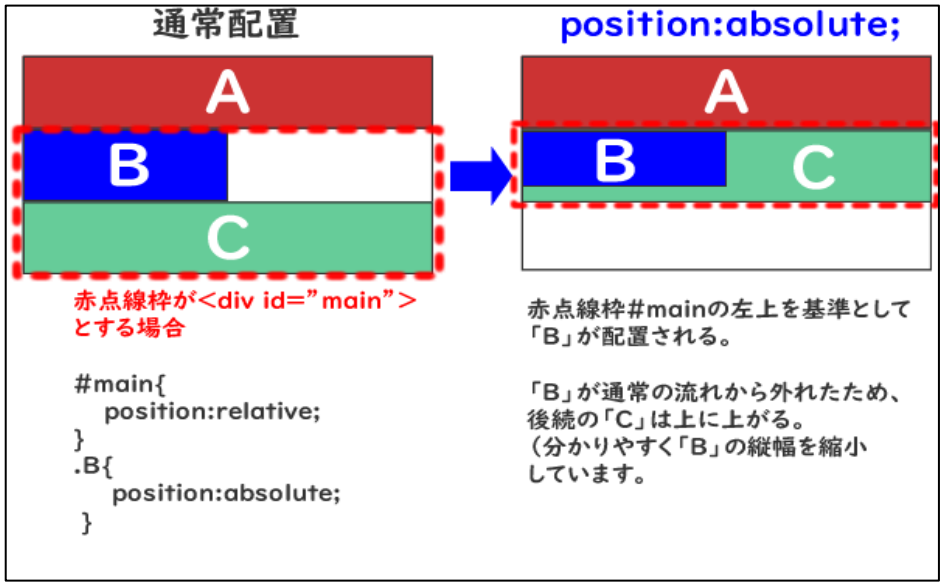
例を以下に示します（分かりやすいように「コンテンツ A」部分を親要素の横幅いっぱいに合わせます）。



absolute での配置

position: absolute; での配置は**絶対位置指定**となります。絶対位置指定とは、親要素や画面全体など、他の要素を基準にして位置を指定する方法のことです。

position: absolute; が設定されると、そのコンテンツは**通常の流れからは完全に切り離され**、基準となる要素を基点として自由に配置させることができます。特徴として、本来の表示領域が無くなり、後に続く要素が上に上がってきます。基準となる親要素には、**position: relative;** を設定します。



【absolute イメージ】

CSS 記述例

```
#main {
position: relative; /*追加行*/
background-color: yellow;
}

#area01 {
position: absolute; /*追加行*/
top: 0; /*追加行*/
left: 0; /*追加行*/
width: 250px;
background-color: #92D050;
}
```

💻 ブラウザ表示例

position: absolute; を設定した「コンテンツＡです。」ボックスが、通常の流れから切り離され、親要素「#main」を基準に位置が移動されました。切り離されたことにより、フッター部分が上がり、「コンテンツＡです。」の後ろに回り込みました。

フッターを「コンテンツＡです。」の前に表示させたい場合は、**z-index** プロパティを使用します。

✏ CSS 記述例

```
#area01 {
position: absolute;
top: 0;
left: 0;
width: 250px;
background-color: #92D050;
z-index: -1; /*追加行*/
}
```

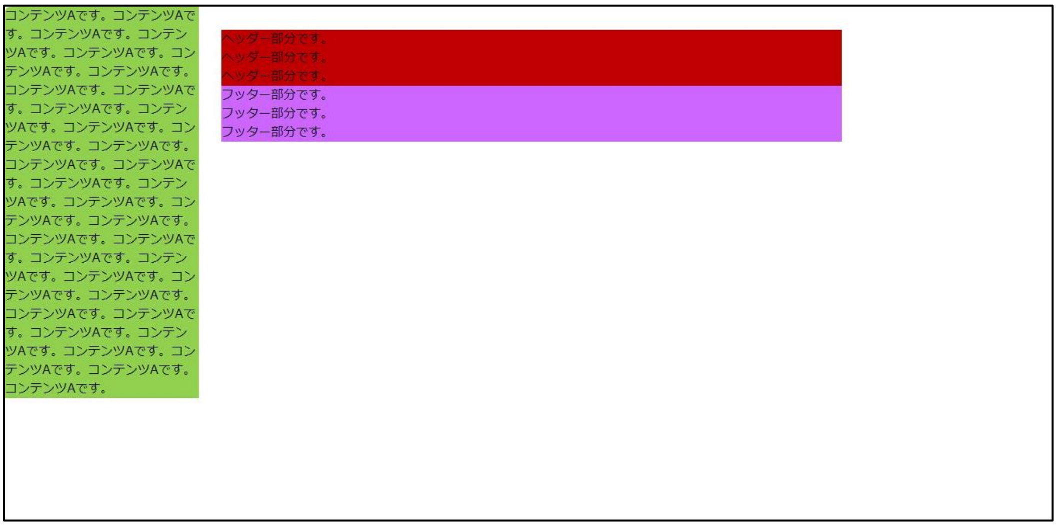
💻ポイント

z-index プロパティは、ボックスの重なりの順序を指定する際に使用します。
position プロパティで static 以外の値が指定されている要素に適用されます。
重なりの順序を整数値で指定します。0 を基準として、値が大きいものほど上になります。

もう 1 つ動作を確認しておきましょう。

基準となる relative 指定された親要素が無い場合、ウィンドウ画面全体の左上が基準位置となります。

💻 ブラウザ表示例



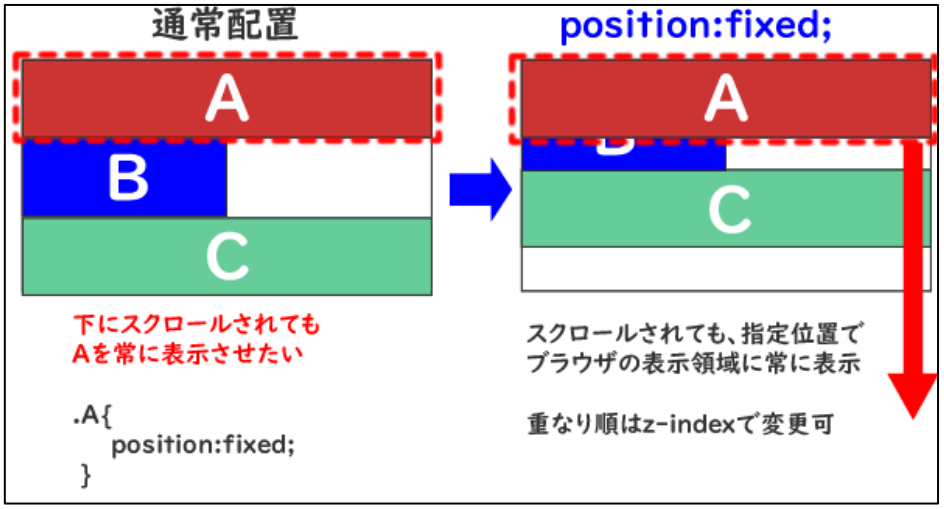
絶対位置への自由配置ではありますが、position: relative; 設定と position: absolute; 設定の組み合わせによる、親要素を基準とする関係性は重要ですので注意しましょう。

基本的には、親要素に static 以外を指定して基準を明示しましょう。

fixed での配置

position:fixed; も absolute 同様に通常の流れからは完全に切り離されます。ただし、１つ違う点があります。それはブラウザの表示領域に**常時表示**されることです。JavaScript※等でパーツに動きを付けた場合は別として、デフォルトではスクロールしても消えずに表示されます。グローバルナビゲーションや「トップに戻る」ボタンなどで使われることが多いです。

※ JavaScript：Web ページに動きをつけるためのプログラミング言語のこと



【fixed イメージ】

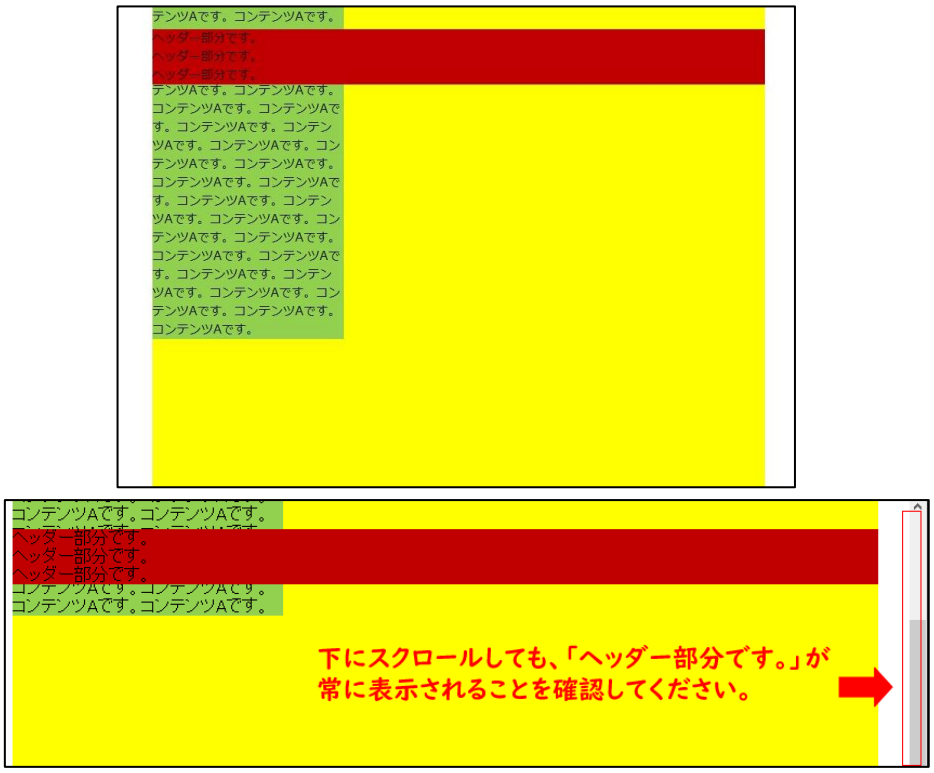
CSS 記述例

```
#main {
height: 2000px; /*追加行（スクロールを発生させる為）*/
background-color: yellow;
}

#header {
position: fixed; /*追加行*/
width: 800px; /*追加行*/
background-color: #C00000;
}

#area01 {
width: 250px;
background-color: #92D050;
}
```

ブラウザ表示例



sticky での配置

position: sticky; を設定した要素は、通常の位置に配置され、スクロールしたときに指定した位置に固定することができます。sticky は、fixed と似ていますが、fixed は常に画面上の同じ位置に要素を固定するのに対して、sticky は要素が**親要素の境界に達するまでスクロールし、その後指定された位置に固定**されます。つまりは、決められた範囲の中で固定表示と通常表示に切り替わるイメージです。

sicky を指定する場合は、必ず親要素が必要です。

「フッター部分です。」の紫ブロックに position: sticky; を設定してみましょう。
「フッター部分です。」（#footer）の親要素は、#wrap になります。

✎ CSS 記述例

```
#main {
background-color: yellow;
}

#header {
background-color: #C00000;
}

#area01 {
height: 1000px; /*追加行（スクロール時に固定表示されることを確認する為）
*/
width: 250px;
background-color: #92D050;
}

#footer {
margin-bottom: 300px; /*追加行（通常表示に切り替わることを確認する為）*/
background-color: #CC66FF;
position: -webkit-sticky; /*追加行*/
position: sticky; /*追加行*/
bottom: 0; /*追加行*/
```

実際にスクロールしてみると、途中まではフッター部分が下に張り付き、最後までスクロールすると通常の流れに戻ることが確認できます。

💻注意点

position: sticky; は、ブラウザによっては対応しておらず効かない場合があります。

Internet Explorer は、対応していません。

Safari で対応させるには、「position: -webkit-sticky;」の記載が必要です。

7　display の基本

学習内容

✓　ブロック要素とインライン要素
✓　CSS でブロック要素やインライン要素を操作する

　これまでの学習で、HTML・CSS の基本コーディングを知ってもらいました。学習内容を今後活かしていくためにも、コーディングに関する知識をさらに深めましょう。

ブロック要素とインライン要素

　HTML で定義されている要素のうち、使用される要素の多くは、**ブロック要素**か**インライン要素**に分類されます。
　この分類により、どの要素の内側にどの要素を配置できるかなどのルールが定められています。基本的にインライン要素はブロック要素を包括できません。

📖 ブロック要素とは

　見出し・段落・表・リストなど、文書を構成する要素で、１つのブロック（かたまり）としてブラウザに認識されます。一般的なブラウザでは前後に改行が入ることが特徴です。ブロックレベル要素とも呼ばれます。

📖 インライン要素とは

　おもにブロック要素の内容として用いられる要素で、文章の一部として扱われます。例えば、<p> 要素の中の のように、文章の中の一部を強調するような使われ方をする要素のことです。ブラウザでは前後に改行が入らず、文章の一部として表示されることが特徴です。

おもなブロック要素	<address> <blockquote> <dl> <fieldset> <form> <h1>～<h6> <hr> <noscript> <p> <pre> <table>
おもなインライン要素	<a> <b> <cite> <code> <i> <img> <input> <kbd> <label> <samp> <select> <small> <textarea>

※ ただし、<a> 要素はブロック要素を包括することができます。

CSS でブロック要素やインライン要素を操作する

上記で示したように、デフォルトの要素の値が決められていますが、CSS の設定で見た目上は変更することが可能です。**ただし、マークアップのルールを変えることは NG です。**

CSS で特に display を指定していなければ、各要素は固有の display プロパティが初期値として設定されています。
（例：p や div などのブロック要素であれば block、em や span であれば inline、table であれば table 等々）

display プロパティの値	説明
display: block	■ **ブロック要素**を形成（要素の上下に余白が入る） ■ 領域： 要素は横幅いっぱいの領域 ■ 幅高さ指定： width ／ height の設定可能 ■ 整列方向： 記述順で縦方向に並ぶ ■ 余白設定： margin と padding は、上下左右の余白設定が可能 ■ 要素例： h1～h6、p、ol、div 要素などのデフォルト値と同様

display: inline	■ **インライン要素**を形成（文中の内容のように横並び） ■ 領域： 要素内のテキスト部分や内容による ■ 幅高さ指定： width ／ height の設定不可 ■ 整列方向： 記述順に横方向に並ぶ ■ 余白設定： margin と padding は、左右のみ余白設定可能 ■ 要素例： img、strong、span 要素などのデフォルト値
display: inline-block	■ **インライン要素のように扱える、ブロック要素**を形成 ■ 領域： 要素内のテキスト部分や内容が領域 ■ 幅高さ指定： 要素への width ／ height の設定可能 ■ 整列方向： 記述順に横方向に並ぶ ■ 余白設定： margin と padding は、上下左右の設定が可能
display: none	要素を非表示にする（**表示領域自体が無くなる**）
display: flex	ブロックレベルで、フレックスコンテナのボックスを形成する 要素を一行に配置する
display: grid	ブロックレベルで、グリッドコンテナを形成する 「グリッドコンテナ」を「グリッドライン」で分割し「グリッドアイテム」を配置するレイアウト（自由配置）
display: list-item	li 要素のようなリスト形式のマーカーボックスを形成する
display: inherit	親要素の値を継承する
display: table	table のような要素を形成する
display: table-cell	table のセル（td）のような要素を形成する

上記に挙げただけでもかなりの数ですが、実はもっとあります…。
時間があるときに「CSS display」等で検索してみてください。

8　チャレンジコーディング課題

学習内容

✓　Web 制作時の画面表示の工夫について
✓　課題 1：チャレンジコーディング：ハンバーグページ制作
✓　課題 2：チャレンジコーディング：誰でもトイレ調査ページ制作

Web 制作時の画面表示を工夫し、実際に HTML や CSS を制作してみましょう！
課題を通して、Web 制作に必要な基礎的なスキルを身に付けましょう！

Web 制作時の画面表示の工夫について

Web 制作時は、複数の資料を見ながら作業を行います。
全画面表示は見やすいですが、各資料が隠れてしまうと表示に手間がかかります。

デスクトップ画面上に並べて配置するなど、自分に合った見やすい工夫を行いましょう。
資料を並べて表示するメリットは以下のとおりです。

- 指示内容や資料を見ながらの編集が可能
- 原稿内容の通りに編集・反映されているかの比較が容易
- 編集結果が「ブラウザ表示に反映されているか」の確認が容易
- 編集ファイルが最新状態なのかを「更新日時」で確認が可能

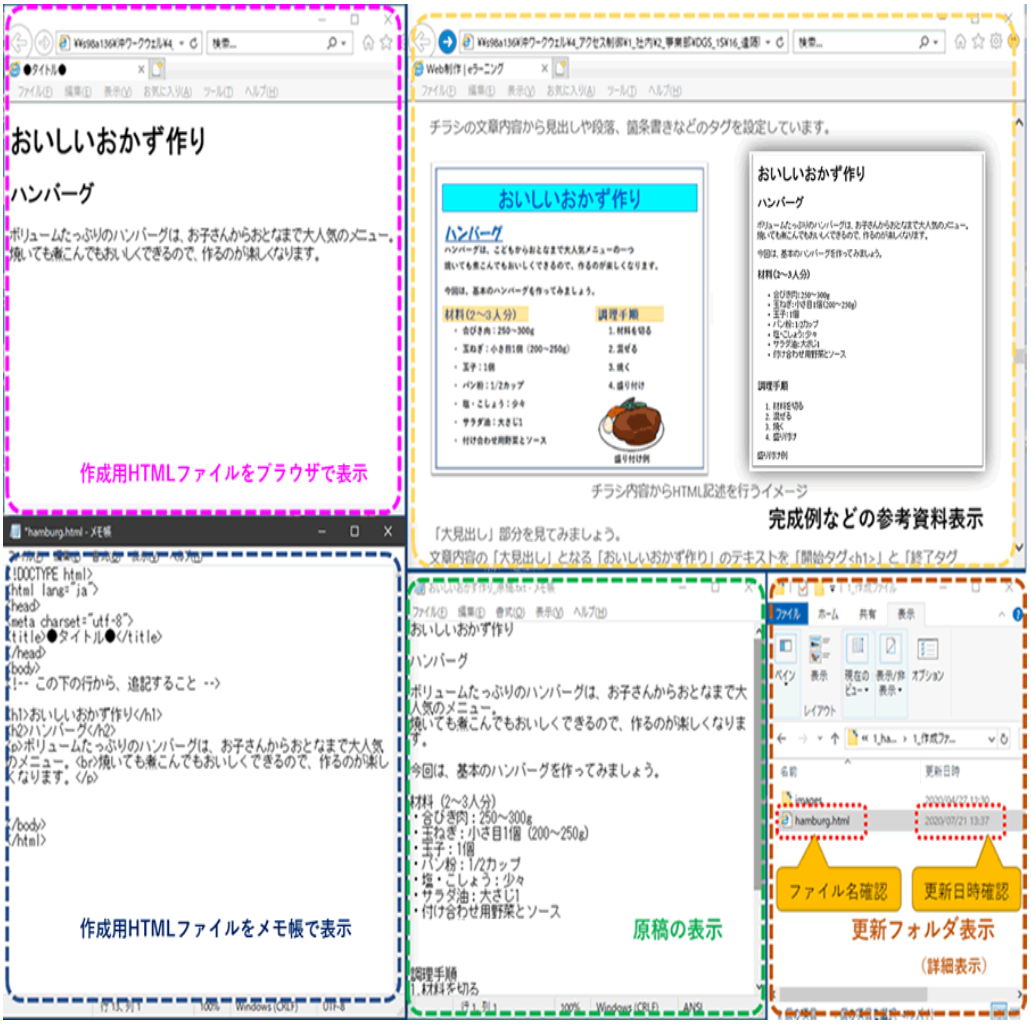
編集・比較・確認が行いやすくなりますので、結果としてスムーズな業務ができます。

各種資料： 原稿・指示内容・完成例など＜指示書に関する資料＞

メモ帳： 更新業務を行うファイル「HTML ファイル」や「CSS ファイル」

ブラウザ： 更新結果の表示確認「Google Chrome」

フォルダウィンドウ： 「更新日時」で最新ファイルの確認可能（「詳細表示」の設定が必要）



ウィンドウ画面に各ファイルを並べて Web 制作を行うイメージ例

課題 1：チャレンジコーディング：ハンバーグページ制作

実際に HTML と CSS を書いて、ハンバーグページを制作してみましょう。
キーボードで直接入力することで、理解は深まります。

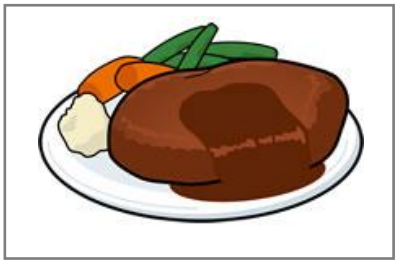
💻資料の準備

まずは、制作するためのファイルとフォルダを準備しましょう。
フォルダ構成は以下です。

- Lesson01
  - **hamburg.html** … ＜マークアップを行うファイル＞
  - css
    - **style.css** … ＜スタイル（見栄え）を設定するファイル＞
  - images
    - hamburg.jpg … ＜ハンバーグの完成イメージ画像＞

任意で画像も制作してみましょう。
ファイル名は「hamburg.jpg」
サイズは、200×130 としましょう。

※ 制作が難しい場合は、無くても問題ありません。



ハンバーグの完成イメージ画像

💻制作手順

1. 上記で示した「Lesson01」フォルダを準備する
2. **hamburg.html** と **style.css** を作成する
   HTML、CSS の作成方法：フォルダ内で右クリックし、新規作成からテキストドキュメントを作成する（拡張子を HTML の場合は【.html】、CSS の場合は【.css】に変更する）
3. **hamburg.html** をブラウザで開く（この時点では、何も表示されません）
4. 次に、**hamburg.html** と **style.css** をエディタで開く
5. 下記（86 ページ）の完成例を見ながら、エディタでコーディングを行う
6. コーディングを行ったら、上書き保存を行う
7. ブラウザで表示確認をする。確認結果でもしミスがあったら、エディタで修正を行う
   ブラウザ表示の再読み込み：ブラウザの再読み込みボタン C か、【F5】キーを押す
8. 4～7 を繰り返し行い、完成例どおり、ブラウザで問題なく表示できれば完成

🔎 作成のヒント！

文字色を**青色**（#0000ff）
背景色を水色（#ccffff）
枠線を 3px の実線で**青色**
文字を**太文字**、行間を 200%

左の枠線（border-left）を 8px の実線で**青色**

背景色を黄色（#ffff00）

背景色を茶色（#b8860b）と深緑色（#808000）
各セルの内側の余白を 8px

水平線を引く空タグ <hr> を使用
枠線を茶色（#b8860b）

💻完成例

以下のように、ブラウザに表示ができればコーディングは完成です。

おいしいおかず作り

ハンバーグ

ボリュームたっぷりのハンバーグは、お子さんからおとなまで大人気のメニュー。
焼いても煮こんでもおいしくできるので、作るのが楽しくなります。

今回は、基本のハンバーグを作ってみましょう。

材料（2～3人分）

- 合びき肉：250～300g
- 玉ねぎ：小さ目1個（200～250g）
- 玉子：1個
- パン粉：1/2カップ
- 塩・こしょう：少々
- サラダ油：大さじ1
- 付け合わせ用野菜とソース

調理手順

1. 材料を切る
2. 混ぜる
3. 焼く
4. 盛り付け

盛り付け例

ハンバーグメニューバリエーション

お好みの食材で様々なハンバーグをお楽しみください。

料理名	メニューポイント
ふわふわハンバーグ	はんぺんのみじん切り
シャキシャキハンバーグ	もやしのザク切りが歯ごたえあり
すき焼き風煮込みハンバーグ	割り下を作り、野菜と一緒に煮込む

→次のページにコーディングが完成した際の記述例を載せています。分からなくなった場合やうまくブラウザに表示できない場合は、確認してみましょう。
（あくまで記述例の一つであり、マークアップの方法は何通りもあります。）

✎ hamburg.html 記述

```
<!DOCTYPE html>
<html lang="ja">
<head>
<meta charset="utf-8">
<title>ハンバーグページ</title>
<link rel="stylesheet" href="css/style.css">
</head>
<body>

<h1>おいしいおかず作り</h1>

<h2>ハンバーグ</h2>
<p>ボリュームたっぷりのハンバーグは、お子さんからおとなまで大人気のメニュー。<br>
焼いても煮こんでもおいしくできるので、作るのが楽しくなります。</p>
<p>今回は、基本のハンバーグを作ってみましょう。</p>

<h3>材料（2～3 人分）</h3>
<ul>
<li>合びき肉：250～300g</li>
<li>玉ねぎ：小さ目 1 個（200～250g）</li>
<li>玉子：1 個</li>
<li>パン粉：1/2 カップ</li>
<li>塩・こしょう：少々</li>
<li>サラダ油：大さじ 1</li>
<li>付け合わせ用野菜とソース</li>
</ul>

<h3>調理手順</h3>
<ol>
<li>材料を切る</li>
<li>混ぜる</li>
<li>焼く</li>
<li>盛り付け</li>
```

```
35
36 <p class="image01"><img src="images/hamburg.jpg" alt="できたてのハンバーグ" width="200" height ="130"><br>盛り付け例</p>
37
38 <h3>ハンバーグメニューバリエーション</h3>
39 <p>お好みの食材で様々なハンバーグをお楽しみください。</p>
40 <table border="1">
41 <tr>
42 <th class="th01">料理名</th>
43 <th class="th01">メニューポイント</th>
44 </tr>
45 <tr>
46 <th class="th02">ふわふわハンバーグ</th>
47 <td>はんぺんのみじん切り</td>
48 </tr>
49 <tr>
50 <th class="th02">シャキシャキハンバーグ</th>
51 <td>もやしのザク切りが歯ごたえあり</td>
52 </tr>
53 <tr>
54 <th class="th02">すき焼き風煮込みハンバーグ</th>
55 <td>割り下を作り、野菜と一緒に煮込む</td>
56 </tr>
57 </table>
58
59 <hr>
60 <p class="copyright">Copyright &copy; キャリア教育プログラム. All Rights Reserved.</p>
61
62 </body>
63 </html>
```

✎ style.css 記述例

```
@charset "utf-8";
body {
background-color: #ffff00;
}

h1 {
color: #0000ff;
background-color: #ccffff;
border: 3px solid #0000ff;
text-align: center;
font-weight: bold;
line-height: 200%;
}

h2 {
color: #0000ff;
border-left: 8px solid #0000ff;
padding: 10px;
}

h3 {
color: #0000ff;
background-color: #ccffff;
padding: 10px;
}

p {
background-color: #fff;
padding: 10px;
}

ul,ol {
background-color: #fff;
border: 1px solid #000;
```

```
35  padding-top: 10px;
36  padding-bottom: 10px;
37  margin-left: 50px;
38  }
39
40  .image01 {
41  background-color: transparent;
42  text-align: center;
43  }
44
45  table {
46  border-collapse: collapse;
47  margin: 0 auto;
48  }
49
50  table th,table td {
51  border:1px solid #000;
52  padding: 8px;
53  }
54
55  table .th01 {
56  background-color: #b8860b;
57  color: #fff;
58  }
59
60  table .th02 {
61  background-color: #808000;
62  color: #fff;
63  }
64
65  table td {
66  background-color: #fff;
67  }
68
```

```
hr {
border-color: #b8860b;
}

.copyright {
margin: 0 25% 50px 25%;
text-align: center;
}
```

💻解説

- **background-color: transparent;** （style.css 41 行目）
  指定した要素の背景色を透明化します。すでに背景色が設定されている要素の場合、その背景色を無効化します。

- **border-collapse: collapse;** （style.css 46 行目）
  テーブルの隣接するセルの境界線を共有します。
  「border-collapse」は、テーブルの中のセルが境界を共有するか分離するかを設定することができるプロパティです。初期値は、「separate（分離）」です。

課題２：チャレンジコーディング：誰でもトイレ調査ページ制作

続いて、誰でもトイレ調査ページを制作してみましょう。
今回は、指示内容にも従って制作を進めてみよう！

💻資料の準備

まずは制作するためのファイルとフォルダを準備しましょう。
フォルダ構成は以下です。

- Lesson02
  - **toilet.html** … ＜マークアップを行うファイル＞
  - css
    - **style.css** … ＜スタイル（見栄え）を設定するファイル＞
  - images
    - head_img.jpg … ＜タイトルの背景画像＞
    - pencil.jpg … ＜リストマーク画像＞
    - toilet_lines.jpg … ＜フッター画像＞
    - toilet__pub.jpg … ＜公衆トイレの画像＞

画像も制作してみましょう。

- head_img.jpg（900×120）
- Pencil.jpg（18×18）
- toilet_lines.jpg（100×45）
- toilet_pub.jpg（300×307）


head_img.jpg


pencil.jpg


toilet_lines.jpg


toilet_pub.jpg

※ 制作が難しい場合は、画像ダウンロードサイトからフリー画像をダウンロードしましょう。

💻指示内容

1. ページのタイトルを、「誰でもトイレ調査」にしてください。（１２ページ参考）
2. 「調査結果にまとめました。」は、見出し「調査結果」に飛ぶ**ページ内リンク**を設定してください。（３２ページ参考）
3. 「誰でもトイレとは」の部分は、**説明リスト**を使用してください。（２０ページ参考）
4. トイレ画像に、**代替テキスト**「公衆トイレ」を設定してください。（２７ページ参考）

💻完成例

以下のように、ブラウザに表示ができればコーディングは完成です。

誰でもトイレ調査

調査のきっかけ

誰でもトイレを利用しようと思ったところ、汚い・破損・設備の不具合で使えないという様々な状況がありました。
地域にある誰でもトイレの**問題点**を調査結果にまとめました。

誰でもトイレとは

高齢者・車椅子利用者・乳幼児連れや妊婦・排泄器官障がい者など、様々な人が利用しやすいように設計されたトイレ。

調査対象

●●区内の駅・公園・地区センターなどの公共施設にある、誰でもトイレ（215ヶ所）

調査結果

問題点

問題点	数
設備の不具合	45ヶ所
便座・便器の破損	32ヶ所
汚い・臭い	65ヶ所
封鎖のため使用不可	11ヶ所

問題点の例

公衆トイレ

問題点詳細

1. 設備の不具合
   - 扉がきちんと閉まらない
   - 鍵が閉まらない
2. 便座や手すりの破損
   - 便座が燃やされている
   - 便座が割れている
3. 排泄物や嘔吐物が便座に付着し、悪臭がひどい
4. 封鎖されていて入室もできない（使用不可）

Web制作_キャリア教育プログラム

→次のページにコーディングが完成した際の記述例を載せています。

toilet.html 記述例

```
<!DOCTYPE html>
<html lang="ja">
<head>
<meta charset="utf-8">
<title>誰でもトイレ調査</title>
<link rel="stylesheet" href="css/style.css">
</head>
<body>

<div id="contents">
<h1>誰でもトイレ調査</h1>

<div id="main">
<h2>調査のきっかけ</h2>
<p>誰でもトイレを利用しようと思ったところ、汚い・破損・設備の不具合で使えないという様々な状況がありました。<br>
地域にある誰でもトイレの<span class="red">問題点</span>を<a href="#note01">調査結果にまとめました。</a></p>

<dl>
<dt>誰でもトイレとは </dt>
<dd>高齢者・車椅子利用者・乳幼児連れや妊婦・排泄器官障がい者など、様々な人が利用しやすいように設計されたトイレ。</dd>
</dl>

<h2>調査対象</h2>
<p>●●区内の駅･公園･地区センターなどの公共施設にある､誰でもトイレ(215ヶ所）</p>

<h2 id="note01">調査結果</h2>
<h3>問題点</h3>
<table border="1">
<tr>
<th>問題点</th>
```

```
</tr>
<tr>
<td>設備の不具合</td>
<td class="right">45 ヶ所</td>
</tr>
<tr>
<td>便座・便器の破損</td>
<td class="right">32 ヶ所</td>
</tr>
<tr>
<td>汚い・臭い</td>
<td class="right">65 ヶ所</td>
</tr>
<tr>
<td>封鎖のため使用不可</td>
<td class="right">11 ヶ所</td>
</tr>
</table>

<h4>問題点の例</h4>
<p class="image01"><img src="images/toilet_pub.jpg" alt="公衆トイレ"
width="300" height ="307"><br>公衆トイレ</p>

<h4>問題点詳細</h4>
<ol>
<li>設備の不具合<ul>
    <li>扉がきちんと閉まらない</li>
    <li>鍵が閉まらない</li>
</ul></li>
<li>便座や手すりの破損<ul>
    <li>便座が燃やされている</li>
    <li>便座が割れている</li>
</ul></li>
<li>排泄物や嘔吐物が便座に付着し、悪臭がひどい</li>
<li>封鎖されていて入室もできない（使用不可）</li>
```

```
66 </ol>
67 </div><!-- main -->
68
69 <div id="footer">
70 <p id="copyright">Web 制作_キャリア教育プログラム</p>
71 </div><!-- footer -->
72 </div><!-- contents -->
73
74 </body>
75 </html>
```

style.css 記述例

```
1  @charset "utf-8";
2  body {
3  background-color: #87ceeb;
4  margin: 0;
5  }
6
7  #contents {
8  width: 1200px;
9  padding: 0 50px;
10 margin: 0 auto;
11 }
12
13 #main {
14 background-color: #fff;
15 padding: 10px 20px;
16 }
17
18 h1 {
19 background-image: url(../images/head_img.jpg);
20 background-size: cover;
21 background-position: center;
```

```
color: #fff;
font-size: 80px;
border: 3px solid #0000cd;
border-left: 20px solid #0000cd;
padding: 10px 0 10px 30px;
margin: 0;
}

h2 {
background-color: #ffd700;
border: 2px solid #000080;
border-left: 5px solid #000080;
border-radius: 10px;
padding: 5px;
margin: 0;
}

h3 {
background-color: #ffdead;
color: #000080;
border-top: 2px solid #000080;
border-bottom: 2px solid #000080;
border-left: 2px solid #000080;
padding: 5px;
margin: 10px 0 10px 10px;
}

h4 {
background-color: #ffdead;
border-left: double #000080;
padding: 5px;
margin: 10px 0 10px 20px;
}

p {
```

```
padding-left: 20px;
}

p .red {
color: red;
font-weight: bold;
}

p a {
text-decoration: none;
}

dl {
padding: 0 20px;
}

dl dt {
font-weight: bold;
border-bottom: 2px solid #000;
width: 130px;
height: 20px;
padding-bottom: 0;
margin-bottom: 5px;
}

dl dd {
font-size: 13px;
}

table {
border-collapse: collapse;
border: 3px solid #000080;
width: 60%;
margin: 0 auto;
}
```

```
92
93  table th {
94  background-color: #ccc;
95  border: 3px solid #000080;
96  padding: 10px 20px;
97  }
98
99  table td {
100 border: 3px dotted #000080;
101 padding: 10px 20px;
102 }
103
104 table .right {
105 text-align: right;
106 }
107
108 .image01 {
109 text-align: center;
110 }
111
112 ol {
113 margin-left: 20px;
114 }
115
116 ul li {
117 list-style-image: url(../images/pencil.jpg);
118 }
119
120 #footer {
121 background-color: #fff;
122 border-top: 3px solid #000080;
123 background-image: url(../images/toilet_lines.jpg);
124 background-position: bottom left;
125 background-repeat: repeat-x;
126 height: 100px;
```

```
127 }
128
129 #copyright {
130 color: #000080;
131 text-align: center;
132 }
```

💻解説

- **border-radius: 10px;**（style.css 34 行目）
  「border-radius」は、要素の角を丸くするプロパティです。値は、単位付きの数値または（%）で指定でき、値が大きいほど角は丸くなります。

- **text-decoration: none;**（style.css 66 行目）
  「text-decoration」は、テキストに装飾を付けることができるプロパティです。a 要素に「text-decoration: none;」を設定することで、リンクの下線を消すことができます。

- **list-style-image: url(“画像のパス”);**（style.css 117 行目）
  「list-style-image」は、リストのマーカー部分に画像を設定することができるプロパティです。

ホームページ制作は以上で終わりです。