

# 1. HTML/CSS 編

ポートフォリオサイトを作ろうハンズオン

2024-12-16 / [@newt239](#)

# このセクションのゴール

- Web サイトの基本的な構造を理解する
- HTML/CSS が書けるようになる

# 目次

1. HTML とは
2. CSS とは
3. Q&A

# 02-1. HTML とは

# HTML とは

- HyperText Markup Language
  - 「マークアップ言語」であって「プログラミング言語」ではない
- 文書の構造を記述するための言語
- タグで囲まれた要素を使って構造を表現
- `index.html` というファイル名に特別な意味を持つ
  - `https://example.com/index.html`へのアクセスは `https://example.com/` と同じ

HTML5は 2014 年 10 月に W3C によって勧告された HTML のバージョン。2021 年 1 月に廃止され、以降は **HTML Living Standard** が有効な規格となっている

# HTML の基本構造

```
<!DOCTYPE html>
<html lang="ja">
  <head>
    <meta charset="UTF-8" />
    <title>ページのタイトル</title>
  </head>
  <body>
    <!-- これはコメントで、実際には表示されません -->
    <h1>見出し</h1>
    <p>段落</p>
  </body>
</html>
```

これだけで表示することができる！

# HTML の基本的なルール

- 開始タグを書いたら必ず終了タグを書く
  - < で始まり > で終わっている部分がタグで、とくに </> で始まっているものが終了タグ
  - <img/> タグや <br/> タグなど、一部例外あり
- Python などとは異なり、インデントの大きさは問わない
- コメントは <!--> と <--> で囲む
- 1 行目の <!DOCTYPE html> は 必ず書く
- <html> タグの中に <head> と <body> がある (この 3 つは必須)
- <head> タグの中にはページの情報を書く
- <body> タグの中にはページの内容を書く
  - ページを開いた時に表示されるのはこの部分

# おもに使うタグ

タグ	説明
<h1> - <h6>	見出しを表現するタグ
<p>	段落を表現するタグ
<a>	ハイパーアリンクを作成するタグ
<img>	画像を表示するタグ
<ul> , <ol> , <li>	リストを表現するタグ
<table> , <tr> , <td>	表を表現するタグ
<div> , <span>	ブロック要素とインライン要素を表現するタグ

# 別のページにリンクする

- VSCode で新しく `about.html` を作成し、以下のコードを書き込む

```
<!DOCTYPE html>
<html lang="ja">
  <head>
    <meta charset="UTF-8" />
    <title>aboutページ</title>
  </head>
  <body>
    <h1>aboutページ</h1>
    <p>これはaboutです。</p>
    <a href=". ./index.html">トップページに戻る</a>
  </body>
</html>
```

- `index.html` の `body` タグ内に以下のコードを追加

```
<a href=". ./about.html">aboutページへ</a>
```

# 画像を表示する

- 画像をダウンロードして、HTML ファイルと同じフォルダに保存
  - 分かりやすくするために `images` フォルダなどを作成して配置することが多い
- `index.html` の `body` タグ内に以下のコードを追加
  - `your-image-name.png` はダウンロードした画像のファイル名

```

```

- 必ず `alt` 属性をつけること
  - 画像が表示できない場合に代替テキストとして表示される

# 箇条書きを作る

- index.html の body タグ内に以下のコードを追加

```
<ul>
  <li>リスト1</li>
  <li>リスト2</li>
  <li>リスト3</li>
</ul>
```

- ul は unordered list の略で、順序のないリストを表現
- li は list item の略で、リストの各項目を表現

```
<ol>
  <li>リスト1</li>
  <li>リスト2</li>
  <li>リスト3</li>
</ol>
```

- ol は ordered list の略で、順序のあるリストを表現

# 表を作る

- index.html の body タグ内に以下のコードを追加

```
<table>
  <thead>
    <tr>
      <th>見出し1</th>
      <th>見出し2</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>1行目1列目</td>
      <td>1行目2列目</td>
    </tr>
    <tr>
      <td>2行目1列目</td>
      <td>2行目2列目</td>
    </tr>
  </tbody>
</table>
```

- table は表を表現
- thead は表の見出し部分
- tbody は表の本体部分
- tr は table row の略で、表の行を表現
- th は table header の略で、表の見出しを表現
- td は table data の略で、表のデータを表現

# 補足: タグのネスト

- これまでの例にもあったように、タグは入れ子にすることができる
- 例えば、リストの中にリストを入れることもできる

```
<ul>
  <li>リスト1</li>
  <li>
    リスト2
    <ul>
      <li>リスト2-1</li>
      <li>リスト2-2</li>
    </ul>
  </li>
</ul>
```

- ただし、許容されていない組み合わせもあるので注意
  - 例えば、`a` タグの中に `a` タグを入れたり、`p` タグの中に `div` タグを入れたりするのは避ける

## 02-2. CSS とは

# CSS とは

- Cascading Style Sheets
  - スタイルシート言語
- 文書のデザインを記述するための言語
- セレクタとプロパティを使ってデザインを記述

# CSS の基本構造

```
h1 {  
  color: red;  
  font-size: 32px;  
}  
p {  
  background-color: lightgray;  
  text-align: center;  
}
```

- セレクタ { プロパティ: 値; } の形式で記述
- セレクタはタグ名や id、class など

# h1 タグの文字色を変えてみよう

- `style.css` というファイルを作成
- `index.html` を以下のようにする
- `style.css` に以下のコードを書き込む

```
<!DOCTYPE html>
<html lang="ja">
  <head>
    <meta charset="UTF-8" />
    <title>ページのタイトル</title>
    <link rel="stylesheet" href=".style.css" />
  </head>
  <body>
    <h1>見出し</h1>
    <p>段落</p>
  </body>
</html>
```

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

- `head` タグ内に `<link rel="stylesheet" href=".ファイル名.css" />` を追加することで CSS ファイルを読み込むことができる

# id と class

- id はページ内で一意の要素を指定するための属性
  - 1 ページで 1 回しか使うことができない
- class は複数の要素に同じスタイルを適用するための属性
  - 複数の要素に同じスタイルを適用したい場合に使う

```
<h1 id="title">文字色が赤になる</h1>
<p class="description">ここの背景がグレーになる</p>
```

```
#title {
  color: red;
}
.description {
  background-color: lightgray;
}
```

- CSS では名前の前に、 id は # 、 class は . をつける
  - バグを防ぐため id は使わず class を使うことが多い

# 詳細度 ①

```
<h1 id="title_id" class="title_class">見出し</h1>
```

```
#title_id {  
    color: blue;  
}  
.title_class {  
    color: green;  
}  
h1 {  
    color: red;  
}
```

このとき、見出しは何色になる？

## 詳細度 ②

- 複数のセレクタが同じ要素を指定している場合、詳細度が高いものが優先される
- 基本的には最後に書かれたスタイルが優先されるが、詳細度によっては逆転する場合がある
- 基本的に id セレクタ > class セレクタ > タイプセレクタ の順に詳細度が高い

実際のところ詳細度の計算はより複雑で、CSSが難しいとされる理由の一つだったりします。

# おもに使うプロパティ

プロパティ名	説明
color	文字色を変更
background-color	背景色を変更
font-size	文字サイズを変更
padding	
margin	
border	枠線に関する設定

# padding と margin

- padding は要素の内側の隙間、margin は要素の外側の隙間

```
<div>一つ前の要素</div>
<div class="target">要素</div>
<div>一つ後の要素</div>
```

- 何も指定しないとき

```
// 見やすさのため色のみ設定
.target {
  background-color: red;
  color: white;
}
```

一つ前の要素

要素

一つ後の要素

- padding だけ指定したとき

```
.target {
  background-color: red;
  color: white;
  padding: 16px;
}
```

一つ前の要素

要素

一つ後の要素

- margin だけ指定したとき

```
.target {
  background-color: red;
  color: white;
  margin: 16px;
}
```

一つ前の要素

要素

一つ後の要素

# 色に関するプロパティ

- `color`、`background-color` など
- 指定方法はさまざま
  - キーワード: `red`, `blue`, `green` など
  - RGB: `rgb(255, 0, 0)`
  - HEX(いわゆるカラーコード): `#ff0000`
  - HSL: `hsl(0, 100%, 50%)`

色に困ったときはMaterial Colorsなどのカラーパレットサイトを使うと便利です。  
<https://materialui.co/colors>

# 数値を扱うプロパティ

- `px` ピクセル
- `%` パーセント
- `em` 親要素のフォントサイズに対する相対的な大きさ
  - `1em` は親要素のフォントサイズと同じ
- `rem` ルート要素のフォントサイズに対する相対的な大きさ
  - `1rem` は `<html>` タグのフォントサイズと同じ

最初は `px` を使っても良いが、レスポンシブデザインを考えると `%` や `rem` を使うほうが良い

レスポンシブデザイン：デバイスの画面サイズに合わせて（パソコンやスマホに応じて）デザインを変えること

# 枠線をつける

- border: 線の太さ 線の種類 線の色 で指定できる

```
.target {  
  border: 1px solid black;  
}
```

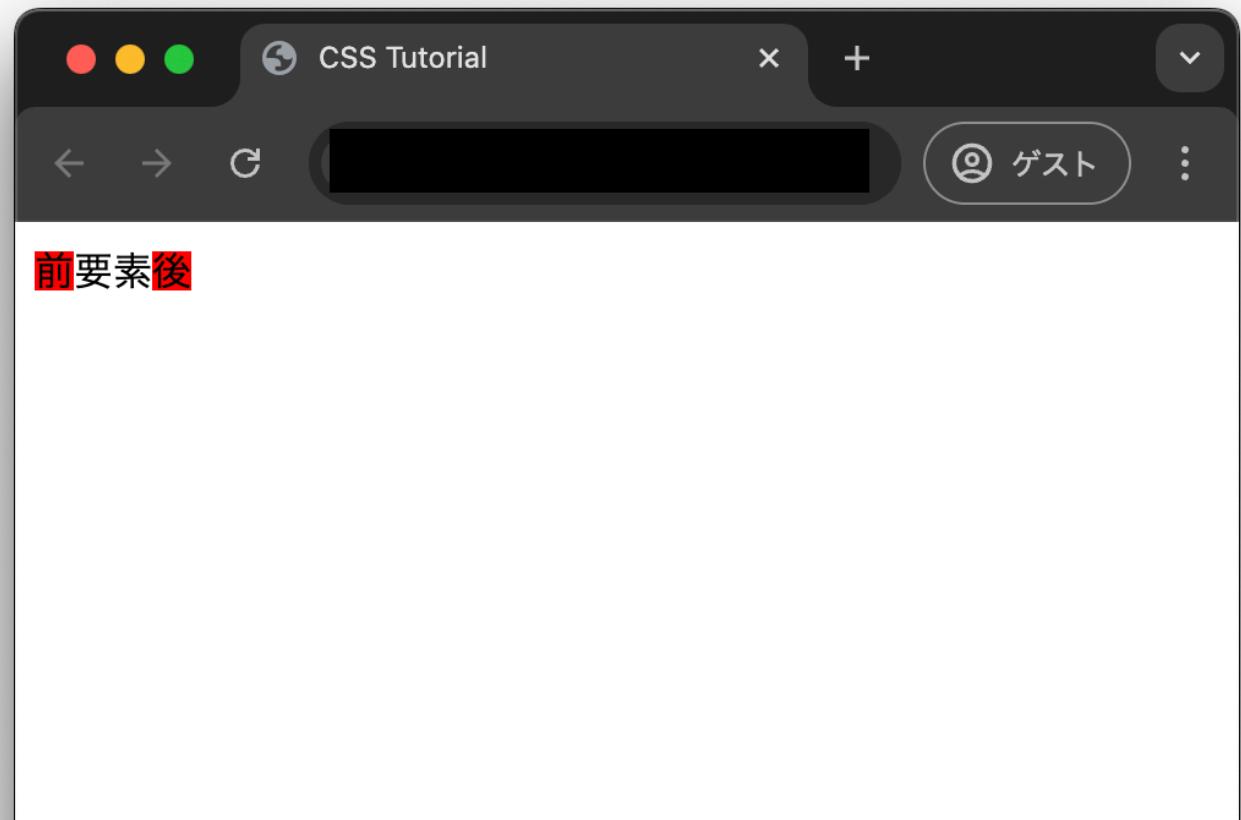
- 線の種類
  - solid : 実線
  - dotted : 点線
  - dashed : 破線
  - double : 二重線
  - none : 線なし

# 疑似要素

- `::before`、`::after`などの疑似要素を使うことで、要素の前後に要素を追加できる
- メリットとしては HTML を変えることなく装飾を加えることができる
- 実際には `content` の値を空にして、`<div>` タグのように装飾することが多い

```
<div class="target">要素</div>
```

```
.target::before {  
  content: "前";  
  background-color: red;  
}  
.target::after {  
  content: "後";  
  background-color: red;  
}
```



# 実例: 見出しのデザイン

```
h1 {  
    font-size: 3rem;  
    text-align: center;  
    padding: 1rem;  
    margin: 1rem;  
    color: white;  
    background: linear-gradient(to bottom right, blue, pink);  
    border-radius: 1.5rem;  
}
```



- `text-align: center` で中央揃え
- `linear-gradient` でグラデーションを作成
  - 角度、起点の色、終点の色を指定
  - 詳細は[linear-gradient\(\) - CSS: カスケーディングスタイルシート | MDN](#)を参照
- `border-radius` で角を丸く

# ここまで変更を GitHub Pages に反映

- VSCode 左上のメニューから「ターミナル」→「新しいターミナル」を選択
- 以下のコマンドを入力

```
git add .
git commit -m "first commit"
git push origin main
```

- コマンドは 1 行ずつコピーして貼り付け、実行 (Enter)
- 最後のコマンドに失敗する場合、`main` を `master` に変更
- 今後の変更も同様の手順で反映できる

## 02-3. Q&A

# Q. h1, p, div とかたくさんあるけど同じじゃダメ？

A. 機械が Web ページの構造を理解しやすいよう、適切なタグを使いましょう。

Google や Bing などの検索エンジンは、Web ページの構造を理解するために HTML のタグを読み取ります。適切なタグを使うことで、検索エンジンが Web ページを正しく理解しやすくなります。

この他、スクリーンリーダーをはじめとする支援技術を使うユーザーにとっても、適切なタグを使うことで Web ページの構造を理解しやすくなります。

# Q. フォントサイズはどのくらいがいい？

A. 本文のサイズは 16px にするのが一般的です。

body 要素に対して font-size:16px を指定したうえで、他の要素に対しては rem を使って相対的なサイズを指定するのが一般的です。

```
html {  
  font-size: 16px;  
}  
h1 {  
  font-size: 2rem; /* 32px */  
}  
small {  
  font-size: 0.75rem; /* .75rem = 12px */  
}
```

第 2 回の内容は以上です。お疲れ様でした！