

発展HTML

前の資料では、HTMLの基本的なタグを使ってWebページの骨組みを作る方法を学びましたね。今回は、さらに一歩進んで、Webページにもっと豊かな意味を持たせたり、インタラクティブな機能を追加したりするためのHTMLタグを見ていきましょう。

基本的なタグだけでもWebページは作れますが、これから紹介するタグを使いこなすことで、より構造が分かりやすく、ユーザーにとっても、検索エンジン（Googleなど）にとっても、そして開発者自身にとっても「良い」Webページを作ることができるようになります。

1. セマンティックタグ：意味のある構造を作る

HTML5から、ページの各部分が「何を表しているのか」をより明確に示すためのタグがたくさん導入されました。これらを **セマンティックタグ** と呼びます。「セマンティック」とは「意味の」という意味です。

以前学んだ `<div>` タグは、単なる「箱」でしかなく、それ自体に意味はありませんでした。セマンティックタグを使うことで、コンピューター（ブラウザや検索エンジン）も「ここがヘッダーだな」「ここがナビゲーションだな」と理解しやすくなります。

代表的なセマンティックタグを見てみましょう。

- `<header>`**：ページやセクションの導入部分、ヘッダー情報を入れる場所です。サイトのロゴ、タイトル、ナビゲーションなどがよく入ります。
- `<footer>`**：ページやセクションのフッター情報（著者、コピーライト、関連リンクなど）を入れる場所です。
- `<nav>`**：主要なナビゲーションリンク（メニュー）を入れる場所です。サイト内の他のページへのリンクや、ページ内の特定セクションへのリンクなど。
- `<main>`**：そのページの主要なコンテンツ（本文）を入れる場所です。`main` 要素は、1つのページに1つだけ使うのが原則です。
- `<article>`**：それ自体で完結する独立したコンテンツ（ブログ記事、ニュース記事、フォーラムの投稿など）を入れる場所です。`article` は入れ子にすることもできます。
- `<section>`**：文書内のテーマごとの区切りを表します。通常、見出し（`<h1>` ～ `<h6>`）と一緒に使われます。`article` の中をさらに `section` で区切ったり、関連するコンテンツをグループ化したりします。
- `<aside>`**：メインコンテンツとは直接的な関係が薄い補足情報や関連情報（サイドバー、広告、補足説明など）を入れる場所です。

なぜセマンティックタグが重要？

1. **SEO（検索エンジン最適化）に有利**：Googleなどの検索エンジンは、これらのタグを見てページの内容や構造をより正確に理解し、検索結果のランキングに役立てます。
2. **アクセシビリティ向上**：スクリーンリーダー（視覚障害者がWebページを読むためのソフト）などが、ページの構造を理解しやすくなり、より多くの人が情報にアクセスしやすくなります。
3. **コードの可読性向上**：`<div>` だらけのコードよりも、`<header>`、`<nav>`、`<main>` などが使われている方が、開発者自身も後からコードを読んだときに構造を把握しやすくなります。

開発者の視点：最初は `<div>` とどう使い分けるか迷うかもしれません

が、「この部分は何のためのエリアか？」を考えて、適切なセマンティックタグを選ぶ癖をつけると良いでしょう。完璧を目指す必要はありませんが、意識することが大切です。

```
<!DOCTYPE html>
<html lang="ja">
  <head>
    <meta charset="UTF-8" />
    <title>セマンティックなページ</title>
    <style>
      ul {
        list-style: none;
      }
      nav li {
        display: inline;
      }
      nav a{
        display: inline-block;
      }
      article {
        float: left;
        width: 68%;
      }
      aside {
        float: right;
        width: 28%;
      }
      footer {
        text-align: center;
        clear: both;
      }
    </style>
  </head>
  <body>
    <header>
      <h1>サイトタイトル</h1>
      <nav>
        <ul>
          <li><a href="/">ホーム</a></li>
          <li><a href="/about">概要</a></li>
          <li><a href="/contact">お問い合わせ</a></li>
        </ul>
      </nav>
    </header>
  </body>
</html>
```

```
</nav>
</header>

<main>
  <article>
    <h2>記事タイトル</h2>
    <p>記事の本文です。...</p>
    <section>
      <h3>セクション1</h3>
      <p>セクション1の内容です。...</p>
    </section>
    <section>
      <h3>セクション2</h3>
      <p>セクション2の内容です。...</p>
    </section>
  </article>

  <aside>
    <h3>関連リンク</h3>
    <ul>
      <li><a href="#">関連情報1</a></li>
      <li><a href="#">関連情報2</a></li>
    </ul>
  </aside>
</main>

<footer>
  <p>&copy; 2025 サイト名</p>
</footer>
</body>
</html>
```

2. フォーム要素の発展：もっとインタラクティブに

基本的な `<form>` , `<input type="text">` , `<input type="submit">` は前の資料で見ましたが、HTMLにはもっと多様な入力形式や便利なフォーム関連タグがあります。

`<input>` の様々な `type` 属性

`<input>` タグは `type` 属性を変えるだけで、色々な入力欄を作れます。

- `type="email"` : メールアドレス入力用。ブラウザによっては入力形式をチェックしてくれます。
- `type="password"` : パスワード入力用。入力文字が `*` などで隠されます。
- `type="number"` : 数値入力用。スピンボタン（上下矢印）が表示されることも。
- `type="date"` : 日付入力用。カレンダーが表示されることも。

- `type="color"` : 色選択用。カラーピッカーが表示されることも。
- `type="range"` : 特定範囲の値を選択するスライダー。
- `type="checkbox"` : 複数選択可能なチェックボックス。
- `type="radio"` : 複数の中から一つだけ選択するラジオボタン。同じ `name` 属性を持つラジオボタンがグループになります。
- `type="file"` : ファイルを選択するためのボタン。

```
<form>
  <label for="email">メール:</label>
  <input type="email" id="email" name="user_email"><br>

  <label for="pass">パスワード:</label>
  <input type="password" id="pass" name="user_password"><br>

  <label for="num">数量:</label>
  <input type="number" id="num" name="quantity" min="1" max="10"><br> <!-- min/maxで範囲指定 -->

  <label for="bday">誕生日:</label>
  <input type="date" id="bday" name="birthday"><br>

  <label for="favcolor">好きな色:</label>
  <input type="color" id="favcolor" name="favorite_color"><br>

  <label for="vol">音量:</label>
  <input type="range" id="vol" name="volume" min="0" max="100"><br> <!-- min/maxで範囲指定 -->

  <p>興味のある分野（複数選択可）:</p>
  <input type="checkbox" id="tech" name="interest" value="technology">
  <label for="tech">技術</label><br>
  <input type="checkbox" id="art" name="interest" value="art">
  <label for="art">芸術</label><br>

  <p>好きな季節（一つ選択）:</p>
  <input type="radio" id="spring" name="season" value="spring">
  <label for="spring">春</label><br>
  <input type="radio" id="summer" name="season" value="summer">
  <label for="summer">夏</label><br>

  <label for="profile_pic">写真:</label>
  <input type="file" id="profile_pic" name="profile_picture"><br>

  <input type="submit" value="送信">
</form>
```

その他のフォーム関連タグ

- `<textarea>` : 複数行のテキスト入力欄。

- `<select>` と `<option>` : ドロップダウンリスト（選択肢）。
 - `<optgroup>` : `<option>` をグループ化してラベルを付けられます。
- `<datalist>` : `<input>` に入力候補のリストを提供します。ユーザーは候補から選ぶことも、自由に入力することもできます。
- `<label>` : 入力欄（`<input>` , `<textarea>` , `<select>` など）と、その説明ラベルを結びつけます。`for` 属性の値と、対応する入力要素の `id` 属性の値を一致させます。
 - **開発者の視点:** `<label>` は非常に重要です！ ラベルをクリックすると対応する入力欄がフォーカスされるため、使いやすさが向上します。また、スクリーンリーダーが「何の入力欄か」を正しく読み上げるためにも不可欠です。必ず使いましょう。
- `<fieldset>` と `<legend>` : 関連するフォーム部品をグループ化し、`<legend>` でそのグループにタイトルを付けます。フォームが長くなる場合に整理するのに役立ちます。

```
<form>
  <fieldset>
    <legend>個人情報</legend>
    <label for="name">名前:</label>
    <input type="text" id="name" name="user_name"><br>

    <label for="message">メッセージ:</label><br>
    <textarea id="message" name="user_message" rows="4" cols="50"></textarea><br>
  </fieldset>

  <fieldset>
    <legend>その他</legend>
    <label for="country">国:</label>
    <select id="country" name="user_country">
      <option value="">選択してください</option>
      <optgroup label="アジア">
        <option value="jp">日本</option>
        <option value="kr">韓国</option>
      </optgroup>
      <optgroup label="ヨーロッパ">
        <option value="fr">フランス</option>
        <option value="de">ドイツ</option>
      </optgroup>
    </select><br>

    <label for="browser">よく使うブラウザ:</label>
    <input list="browsers" id="browser" name="user_browser">
    <datalist id="browsers">
      <option value="Chrome">
      <option value="Firefox">
      <option value="Safari">
      <option value="Edge">
    </datalist><br>
  </fieldset>
```

```
<input type="submit" value="登録">
</form>
```

フォームバリデーション属性

HTMLだけで簡単な入力チェック（バリデーション）を行うための属性もあります。

- required** : 入力必須項目にします。
- pattern** : 正規表現を使って、入力可能な文字パターンを指定します。
- min**, **max** : 数値や日付の最小値・最大値を指定します。
- step** : 数値や範囲入力の刻み幅を指定します。
- minlength**, **maxlength** : 入力可能な最小・最大文字数を指定します。

```
<form>
  <label for="username">ユーザー名（必須，5文字以上）:</label>
  <input type="text" id="username" name="username" required minlength="5"><br>

  <label for="postal_code">郵便番号（例：123-4567）:</label>
  <input type="text" id="postal_code" name="postal_code" pattern="\d{3}-\d{4}" placeholder="123-4567"><br>

  <input type="submit" value="送信">
</form>
```

開発者の視点： HTMLのバリデーションは手軽ですが、あくまで補助的なものです。悪意のあるユーザーはこれを簡単に回避できてしまうため、**サーバー側でのしっかりとしたバリデーションは別途必須** です。フロントエンド（HTML/JavaScript）のバリデーションは、ユーザーの入力ミスを減らし、使い勝手を良くするためのものと考えましょう。

3. メディア要素：動画と音声の埋め込み

HTML5では、プラグイン（昔はFlashなどが使われていました）なしで動画や音声をWebページに埋め込めるようになりました。

<video> タグ

動画ファイルを埋め込みます。

- src** : 動画ファイルのパスまたはURL。
- controls** : 再生、一時停止、音量などのコントロールパネルを表示します。通常は付けた方が親切です。

- `width` , `height` : 動画の表示サイズを指定します。CSSで指定する方が柔軟です。
- `autoplay` : ページ読み込み時に自動再生します。多くの場合、ユーザー体験を損ねるので注意が必要です。 `muted` 属性と一緒にないとブロックされることが多いです。
- `loop` : 動画を繰り返し再生します。
- `muted` : 最初からミュート（消音）状態で再生します。自動再生したい場合に必要になることが多いです。
- `poster` : 動画が読み込まれる前や、再生開始前に表示される画像のパスまたはURL。

```
<!-- コントロール表示あり -->
<video src="movie.mp4" controls width="640" height="360" poster="thumbnail.jpg"></video>

<!-- 自動再生（ミュート必須の場合が多い）、ループ -->
<video src="background.mp4" autoplay muted loop width="100%"></video>
```

<audio> タグ

音声ファイルを埋め込みます。属性は `<video>` と似ています。

```
<!-- コントロール表示あり -->
<audio src="music.mp3" controls></audio>

<!-- 自動再生（非推奨）、ループ -->
<audio src="bgm.ogg" autoplay loop></audio>
```

<source> タグ

異なるブラウザが対応している動画/音声フォーマットは様々です。 `<source>` タグを使うと、複数のフォーマットのファイルを用意しておき、ブラウザが再生可能なものを自動で選択させることができます。 `<video>` または `<audio>` タグの中に入れます。

```
<video controls width="640" height="360" poster="thumbnail.jpg">
  <source src="movie.webm" type="video/webm">
  <source src="movie.mp4" type="video/mp4">
  <!-- 対応ブラウザがない場合のメッセージ -->
  お使いのブラウザは動画再生に対応していません。
</video>

<audio controls>
  <source src="music.opus" type="audio/opus">
```

```
<source src="music.mp3" type="audio/mpeg">
お使いのブラウザは音声再生に対応していません。
</audio>
```

<track> タグ

動画や音声に、字幕、キャプション、説明などを追加するためのタグです。Web VTT（.vtt）という形式のファイルでテキストと表示タイミングを指定します。アクセシビリティ向上のために重要です。

```
<video src="lecture.mp4" controls>
  <track kind="subtitles" src="subtitles_ja.vtt" srclang="ja" label="日本語字幕" default>
  <track kind="captions" src="captions_en.vtt" srclang="en" label="English Captions">
</video>
```

4. その他の便利なタグ

他にも、特定の目的で役立つタグがあります。

<details> と <summary>

クリックすると詳細が表示/非表示される、アコーディオンのようなUI（ユーザーインターフェース）を簡単に作れます。<summary> が常に表示されるタイトル部分、<details> の中の <summary> 以外の部分が詳細コンテンツになります。

```
<details>
  <summary>もっと詳しく見る</summary>
  <p>ここが詳細な内容です。画像やリストなども入れられます。</p>
  <ul>
    <li>項目1</li>
    <li>項目2</li>
  </ul>
</details>
```

<iframe>

インラインフレーム（Inline Frame）の略で、現在のHTMLページの中に、別のHTMLページを埋め込むことができます。Google マップや YouTube 動画の埋め込みによく使われます。


```
<!-- Google マップの埋め込み例 -->
<iframe
  src="https://www.google.com/maps/embed?pb=!1m18!1m12!..."
  width="600"
  height="450"
  style="border:0;"
  allowfullscreen=""
  loading="lazy"
  referrerpolicy="no-referrer-when-downgrade">
</iframe>

<!-- YouTube 動画の埋め込み例 -->
<iframe
  width="560"
  height="315"
  src="https://www.youtube.com/embed/VIDEO_ID"
  title="YouTube video player"
  frameborder="0"
  allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture"
  allowfullscreen>
</iframe>
```

開発者の視点:`<iframe>` は便利ですが、セキュリティリスク（クリッ

クジャッキングなど）や、ページの読み込み速度に影響を与える可能性もあります。むやみに使わず、信頼できるコンテンツの埋め込みに限定しましょう。

`sandbox` 属性や`allow` 属性で、埋め込みコンテンツができることを

制限することも検討しましょう。

`<figure>` と `<figcaption>`

図版（画像、図、コード片など）とそのキャプション（説明文）をセットで示すためのタグです。 `<figure>` で図版全体を囲み、 `<figcaption>` でキャプションを記述します。

```
<figure>
  
  <figcaption>図1: 最近の売上推移を示すグラフ</figcaption>
</figure>

<figure>
  <pre><code>function hello() {
  console.log("Hello, world!");
}</code></pre>
  <figcaption>コード例1: JavaScriptでの挨拶表示</figcaption>
</figure>
```

<canvas> (概要)

JavaScript と連携して、図形を描画したり、アニメーションを作成したりするための領域を提供します。ゲームやグラフィックツールなど、高度なグラフィック表現に使われますが、HTMLだけでは何も表示されず、JavaScriptでのプログラミングが必須になります。

```
<canvas id="myCanvas" width="200" height="100" style="border:1px solid #000;"></canvas>  
<!-- この後、JavaScriptで描画処理を書く -->
```

5. まとめ：より豊かな表現力を持つHTMLへ

お疲れ様でした！今回は、基本的なHTMLに加えて、より発展的なタグを見てきました。

- **セマンティックタグ**（`<header>`，`<nav>`，`<main>` など）を使うことで、ページの構造が明確になり、SEOやアクセシビリティ、コードの可読性が向上します。
- **フォーム要素**（`<input>` の多様な `type`，`<textarea>`，`<select>`，`<label>` など）を使いこなすことで、ユーザーが情報を入力しやすい、インタラクティブなページを作れます。
- **メディア要素**（`<video>`，`<audio>`，`<source>`）で、動画や音声を簡単に埋め込みます。
- **その他のタグ**（`<details>`，`<iframe>`，`<figure>` など）も、特定の場面で役立ちます。

これらのタグを適切に使うことで、単なる情報の羅列ではなく、構造化され、意味を持ち、インタラクティブで、豊かな表現力を持つWebページを作成することができます。

もちろん、すべてを一度に覚える必要はありません。「こういうことができるんだな」と知っておき、実際にページを作る際に「あの機能を使いたいな」と思ったら、改めて調べ直してみる、という進め方で大丈夫です。

次の資料では、いよいよCSSの発展的な内容に入っていきます。より高度なセレクトクや、Webページに動きや視覚効果を加える方法などを学んでいきましょう！