HTMLハンズオン&リファレンス

1ページの簡易プロフィールページを作成する

はじめに

本書は、Parts & Tipsというサイト内のコンテンツ「HTMLハンズオン＆リファレンス（β）[*1]」で公開している「1ページの簡易プロフィールページを作成する」というハンズオン記事の内容をまとめたものです。このハンズオンは主にHTMLやCSS初学者の方に向けて、できるかぎりわかりやすい解説となるよう心がけています。

また、書籍用にレイアウトや構成を変更していますが、ハンズオンの掲載内容そのものに変更はありません。具体的には、以下の箇所を書籍用に変更しています。

- 画像の大きさやトリミング位置の変更
- 一部本文の紙面向け説明

ハンズオンでは、1ページの簡易プロフィールページを作成します。ページ構成はおおまかに以下のとおりです。

- ページタイトル
- プロフィールアイコン
- 紹介文
- サイト、SNSなど各種リンク
  - 名称
  - 説明文
- ページ制作者（著作権）情報

構造はシンプルですが、サイト制作の基本を詰め込んでいます。スタイルシートでのカスタマイズもしやすいかと思いますので、ぜひ挑戦してみてください。

なお、このハンズオンの中ではGoogle Fonts[*2]とPhosphor Icons[*3]を使用しています。ハンズオン記事内でも簡単に利用方法を説明していますが、詳細については当サイトで公開している以下の記事もご参考ください。

- サイト内でGoogle Fontsを利用する[*4]
- Phosphor Iconsを利用する[*5]

---

1. https://parts-tips.com/ho_ref/
2. https://fonts.google.com/
3. https://phosphoricons.com/
4. https://parts-tips.com/posts/google-fonts
5. https://parts-tips.com/posts/phosphor-icons

ソースコードのダウンロードについて

このハンズオンで作成するHTML/CSSのソースコードは、「HTMLハンズオン＆リファレンス（β）」サイトからダウンロードできます。ソースコードは改変やご自身のサイトでの公開など、ご自由にしていただいて問題ありません。ただし、改変したものを含め再配布はおやめください。

また、ソースコードの利用に際し、当サイトへのリンクなどは不要です。

目次

はじめに

1. HTML文書の基本構造をつくる

2. コンテンツをマークアップする

3. スタイルシートを書く

付録

Chapter1

HTML文書の基本構造をつくる

1. HTML文書の基本構造をつくる

このチャプターでは、サイト制作のうえで基本となるHTMLの構造について解説しています。

さっそくHTMLを書いていきます。適当なフォルダを作成し、その中に`index.html`ファイルを作成します。

ドキュメントタイプ宣言をする

`index.html`の1行目で、まずはじめに「HTMLのバージョン」を宣言を行います。VS Codeなど補完機能がついているエディターでは、最初の一文字目を打つと候補に出てくるかと思います。

```
<!DOCTYPE html>
```

これはHTMLの一番新しい仕様である「HTML Living Standard」の宣言で、現在においては必須の記述です。いま新しくHTMLページを作成する場合は何も考えずに`<!DOCTYPE html>`と書いておいて問題ありません（この仕様のひとつ前のHTML5も同じ記述です）

古いサイトだと、以下のような記述がされているものもあります。いずれも過去のHTMLの仕様に基づいた宣言です（ここに挙げたもの以外もあります）

```
<!-- HTML 4.01 Strict -->
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">

<!-- HTML 4.01 Transitional -->
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<!-- XHTML 1.0 Strict -->
<!DOCTYPE HTML PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

ブラウザはページを表示する際、このドキュメントタイプ宣言を見て、そのページで表示するHTMLのモード（解釈）を自動で切り替えます。宣言がない場合もページは表示されますが、意図しない表示となる可能性もあるので、必ず宣言するようにします（宣言がないことでとんでもない表示になることは稀ですが...）

`<!DOCTYPE>` は慣例的に大文字で書かれることが多いですが、小文字でも構いません。とりあえず、HTMLの１行目ではまずこの宣言を記述します。

HTML構造の大枠をつくる

つぎに、HTML構造の大枠をつくります。といっても、必要最低限の記述は以下のみです。

```
<!DOCTYPE html>

<html lang="[メイン言語]">
<head>
  <meta charset="[文字コード]">
  <title>[サイト名などブラウザのタブバーなどに表示される名称]</title>
</head>
<body>

<!-- ここに記述された内容がブラウザの画面に表示される -->

</body>
</html>
```

実際のウェブサイトでは `head` の中身をもうすこし記述しますが、ブラウザにページを表示するために最低限必要な情報はこれだけであり、テキストや画像、リンクなどをそのまま表示するだけであれば、上記でじゅうぶん機能します。先述のドキュメントタイプ宣言とこの `<html>...</html>` の構造は、ひとつの構文として覚えておくとよいかと思います。

`<!-- -->` で書かれた部分は、HTML文書やRSSフィードなどで利用されるXML文書におけるコメント表記です。コメントアウトともいい、この記述部分はブラウザのソース表示では確認できますが、実際の画面上には反映されません。

html要素

要素としての `html` は、HTML文書における起点になる要素であり、他のすべてのHTMLはこの配下に記述します。ここでは以下を指定しています。

lang属性

`lang` 属性の値には、「そのページで使用するメイン言語」を指定します。日本語であれば `ja` 、英語は `en` です。

Chapter2

コンテンツをマークアップする

2. コンテンツをマークアップする

このチャプターでは、ブラウザに表示するためのコンテンツのマークアップ手法と、HTML要素（HTMLタグ）の意味などについて解説しています。

マークアップ言語とは

HTMLの正式名称は「HyperText Markup Language（ハイパーテキスト・マークアップ・ランゲージ）」といいます。HTMLはマークアップ言語と呼ばれており、HTMLを記述することを「HTMLをマークアップする」ともいいます。

マークアップ言語は「文書の構造と体裁を明確に記述するためのデータ記述言語」のことで、「マークアップする」ことは「文書の構造と体裁を明確に記述する」ことに相当します。

…と、難しく書きましたが、ざっくり言うと「ブラウザがこれから表示するページを正しく解釈できるように、用途に見合う正しいHTML要素を使用してHTMLを書く（マークアップする）」ということです。見出しに該当する項目は`h1`～`h6`要素、段落テキストは`p`要素、リストに該当する項目は`ul`要素（番号付きリストであれば`ol`要素）を使用してマークアップするといった具合です。

HTML文書を適切にマークアップすることで、以下のようなメリットがあります。

- 音声読み上げ機能などアクセシビリティに配慮することができる
- スタイルシート（CSS）が読み込まれない状況でも、閲覧者が文書として内容を的確に把握することが可能になる

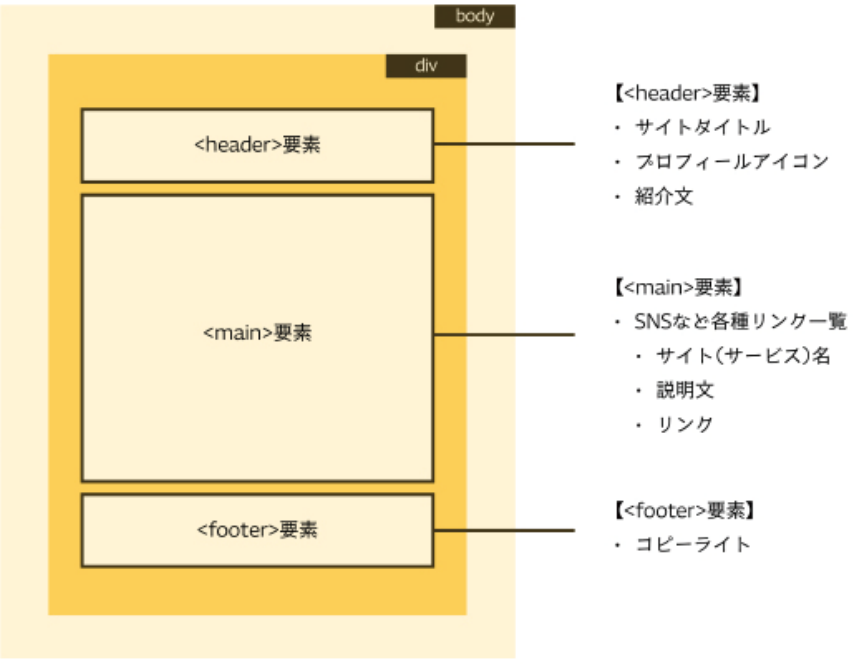
Wordソフトなどで文書を作成する際は、見出しや段落テキスト、リストや表などを使用して文書を読みやすく整形しながら作成しますが、HTML文書のマークアップもそれと似たようなものです。なお、Word文書作成における「テキストのサイズ変更」や「テキストの右寄せ」、「表のデザイン変更」が、のちのチャプターで行う「スタイルシート（CSS）記述によるレイアウトやデザイン調整」に該当します。

とはいえ、実際にマークアップをしていると「このテキストはどのHTML要素を使ってマークアップすればいいのだ…？」ということも結構あり、すべてを正しくマークアップすることは実際には難しいです。とりあえず最初のうちはあまり難しく考えずに、とくにこのハンズオンではだいたいのふわっとした理解と意味づけでマークアップするでOKです。

なお余談になりますが、マークアップ言語と呼ばれるものはHTMLの他にもいくつかあり、RSSフィードやサイトマップの配信などで利用される「XML（Extensible Markup Language / エクステンシブル・マークアップ・ランゲージ）」もマークアップ言語のひとつです。

コンテンツ構造の確認

まずは、このハンズオンで制作するプロフィールページの構造を確認します。大枠で以下の構造をしています。



ここで制作するプロフィールページの構造と内容はとてもシンプルなものなので、この程度であれば `header` `main` `footer` と各要素に分ける必要もそこまでないのですが（とくに `header` 要素と `footer` 要素には中に入る内容がそこまで多くないので）、HTMLの構造としては分けておいたほうが見やすいので、このようにしています。

それでは、HTMLのマークアップをはじめていきます。

コンテンツの大枠となる箱を用意する

ブラウザの画面に表示させる内容は、基本的にはすべて `body` 要素の中に記述します。まずは大枠の要素を追加します。

ヘッダー部分

ヘッダー部分の箱を用意します。ヘッダーは主にサイト名やサイト内コンテンツへのナビゲーションメニューなど、サイト全般にかかるメニューなどが置かれる領域です。

Chapter3

スタイルシートを書く

3. スタイルシートを書く

このチャプターでは、スタイルシートとは何か、またスタイルシートを書くにあたり事前に知っておくとよい情報などについて解説しています。

スタイルシートとは

スタイルシートの本来の意味としては、HTMLやXMLといった構造化文書などの表示を制御するしくみのことを指します。

スタイルシートと同義で扱われることの多い「CSS」は、正式名称を「Cascading Style Sheets（カスケーディング・スタイル・シーツ）」といい、主にウェブサイトを構成するHTML文書やXML文書のスタイリングに利用されます。

CSSは構造化文書の見た目を記述する「スタイルシート言語」のうちのひとつで、コンピュータ言語のひとつでもあります。そういう意味では、CSSはJavaScriptやPHPなどと同じプログラミング言語と言えるかもしれません。

厳密にはスタイルシートとCSSは必ずしもイコールではないですが、当コンテンツでは同じものとして扱います。また、これ以降は「CSS」と記載します。

HTML文書にCSSを反映させる方法

HTML文書にCSSを反映する方法としては、主に3つ方法があります。そのうちサイト制作においては、最初に紹介する「外部ファイルとしてHTML文書内で読み込む」方法がもっとも推奨されている方法です。

1. 外部ファイルとしてHTML文書内で読み込む

HTML文書とは別で用意したCSSファイル（拡張子は`.css`）をHTML文書内で読み込んでスタイルを反映します。

CSSファイルの文字コードは、HTML文書と同じ文字コードを使用します。現在では`utf-8`を使用することが一般的です。

```
<head>
  <link rel="stylesheet" href="[CSSファイルのパス]">
</head>
```

この方式は現在もっとも一般的なCSSの反映方法であり、推奨されている方法です。CSSファイルは通常HTML文書の`head`要素内で読み込まれることが一般的ですが、状況により`body`要素内で読み込まれる場合もあります。やむを得ない場合を除き、基本的には`head`内での読み込みを推奨します。

また、CSSファイルは複数読み込みが可能です。この場合、読み込むCSSファイルの数だけ`link`を記述します。それぞれのCSSファイル内に同じ記述がある場合、あとから読み込まれたものが優先（前に読み込んだCSSに追加や上書き）されます。

```
<head>
  <link rel="stylesheet" href="./css/style_base.css">
  <link rel="stylesheet" href="./css/style_common.css">
  <link rel="stylesheet" href="./css/home.css">
</head>
```

CSSのファイルパスはこれまでのチャプターでも取り上げた画像やリンクの記述同様、CSSファイルを読み込むHTML文書ファイルから見た相対パスのほか、`https://`からはじまる絶対パス、サイトのドキュメントルート（`public_html`など）である`/`からはじまるルートパスで指定します。

ローカルPC上での作業時は、CSSファイルに限らず相対パスでの記述を推奨します。この記述方式はレイアウトの変更や修正をする際、「現在表示しているHTML文書の置いてある場所を起点に」指定したファイルのパスをつなげるため、ローカル上・サーバー上関係なくファイルを読み込むことが可能です。

ちなみに、絶対パスはCDN（Contents Delivery Network）など外部のライブラリファイルなどを読み込む際に、またルートパスはプログラムで動くページで利用されていることが多いです。

なお、`link`要素に`media`属性を付与することで、後述する「メディアクエリ」と同じ指定することが可能です。これを利用すると、たとえば以下のように「プリント時に追加でCSSを読み込む」ということが可能です。

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

`media`属性の値としては`all` `screen` `print`という値のほか、後述する`screen and (...)`という表示幅での指定できるので、スマホ表示とPC表示で読み込むCSSを変えるということが可能になります。