

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

```
html>  
ng="ja">  
 charset="utf-8">  
a name="viewport" content="width=device-width,initial-scale=1.0"  
a name="format-detection" content="telephone=no">  
ta name="robots" content="noindex,nofollow,noarchive">  
tle>My Profile</title>  
nk rel="stylesheet" href="./css/style.css">  
ad>  
ly>  
v class="wrapper">  
<header class="header">  
  <h1>My Profile</h1>  
  <figure class="prof_icon"></figure>
```

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、ご自由にしてください。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

はじめに	1
ソースコードのダウンロードについて	2

1. 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	10
ドキュメントタイプ宣言をする	10
HTML構造の大枠をつくる	11
html要素	11
head要素	12
body要素	13
head要素の中に追記する	13
meta要素でビューポートを追加する	13
meta要素でformat-detectionを設定する	14
meta要素で検索クローラーのクロールを拒否する	14
link要素でCSSファイルを読み込む	15
ここまでに記述したコード	16
参考情報など	16

2. 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	20
マークアップ言語とは	20
コンテンツ構造の確認	21
コンテンツの大枠となる箱を用意する	21
ヘッダー部分	21
メイン部分	22
フッター部分	22
各要素をグループ化する	23
各HTML要素にclass属性を付与する	24
ここまでに記述したコード	27
参考情報など	28
2-1. header要素のマークアップ	29

ページのタイトル見出しをつける	29
h要素（見出し要素）	29
表示確認	30
プロフィール画像を表示する	30
figure要素（本文の補足となる図表などを指定する要素）	30
img要素（画像要素）	31
表示確認	33
プロフィール説明文を追加する	33
p要素 / br要素（段落要素 / 改行要素）	34
表示確認	34
ここまで記述したコード	34
参考情報など	35
2-2. main要素のマークアップ	37
コンテンツのタイトル見出しをつける	37
表示確認	37
リンク一覧を作成する	38
ul要素 / ol要素 / li要素（リスト要素）	38
リンクリストをマークアップする	39
リンクリストにワンポイントアイコンを追加する	43
ここまで記述したコード	46
参考情報など	47
2-3. footer要素のマークアップ	49
コピーライト表記を入れる	49
small要素（附随コメント要素）	49
表示確認	49
ここまで記述したコード	50
参考情報など	52

3. スタイルシートを書く

3. スタイルシートを書く	54
---------------------	----

スタイルシートとは.....	54
HTML文書にCSSを反映させる方法.....	54
1. 外部ファイルとしてHTML文書内で読み込む.....	54
2. style要素で記述する.....	56
3. HTMLタグにstyle属性で記述する.....	56
ブラウザのデフォルトスタイル.....	57
CSSの記述方法.....	57
セレクタのネスト.....	58
複数のセレクタに同じスタイルをあてる.....	59
CSSの優先順位.....	60
CSSの継承について.....	61
CSSで利用できる単位について.....	62
レスポンシブ対応について.....	63
HTML要素のボックス領域.....	64
おわりに.....	65
参考情報など.....	65
3-1. リセットCSSを準備する.....	67
CSSファイルを作成する.....	67
CSSの文字コードを指定する.....	67
@charset (アットルール).....	67
文字コードの指定について.....	67
スタイルをリセットする.....	68
コメントの書き方.....	69
余白の調整.....	69
HTML要素の横と高さの計算方法を指定する.....	70
画像 (img要素) のスタイルをリセットする.....	71
表示確認.....	73
ここまでに記述したコード.....	73
参考情報など.....	74
3-2. ベースをつくる.....	75
ベースを整える.....	75
html要素にページ内スクロールの挙動を設定する.....	75
body要素に文字まわりやサイト背景色の情報を設定する.....	76
link要素にリンク表示時のスタイルを設定する.....	79
コンテンツの表示幅を整える.....	82
marginプロパティ.....	82
widthプロパティ / max-widthプロパティ.....	84
表示確認.....	84
ここまでに記述したコード.....	85
参考情報など.....	87

3-3. header部分のCSSを書く	88
余白を調整する	88
ページタイトルにスタイルをあてる	88
Google Fontsを読み込む	88
h1 要素にスタイルをあてる	90
表示確認	93
プロフィールアイコンにスタイルをあてる	93
width プロパティの値fit-contentとは	94
border-radius プロパティ	94
overflow プロパティ	95
表示確認	96
CSS変数（カスタムプロパティ）を利用する	97
CSS変数の定義	97
CSS変数の呼び出し（var()関数）	98
カラーコード記述箇所をCSS変数に置き換える	98
ここまで記述したコード	100
参考情報など	102
3-4. main部分のCSSを書く	104
余白を調整する	104
コンテンツタイトルにスタイルをあてる	104
見出しの重複する値をまとめて記述する	105
表示確認	106
リンクリストにスタイルをあてる	107
リストのマーカーを消す	107
リンク部分のスタイルを調整する	109
リンク部分のスタイルを調整する（レイアウト崩れを防ぐ）	111
見出しとテキストのスタイルを調整する	113
矢印アイコンの位置を調整する	114
矢印アイコンの位置を調整する（リスト項目枠内に収める）	118
ここまで記述したコード	121
参考情報など	124
3-5. footer部分のCSSを書く	126
テキスト位置を調整する	126
表示確認	126
ここまで記述したコード	127
参考情報など	130
3-6. スマホでの見た目を調整する	131

ブラウザの検証ツールでスマホの見た目を確認する	131
検証ツールの表示方法：Google Chromeの場合	131
検証ツールの表示方法：Safariの場合	132
スマホ表示用にCSSを調整する	134
メディアクエリとは	134
スマホ用のCSSを調整する	136
表示確認	136
ここまでに記述したコード	137
参考情報など	141
おわりに.....	142

付録

付録：最終的なHTMLとCSSのコード	144
ディレクトリ構成.....	144
HTML文書	144
CSSファイル.....	145

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ういかに思います。

`<!-- -->` 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

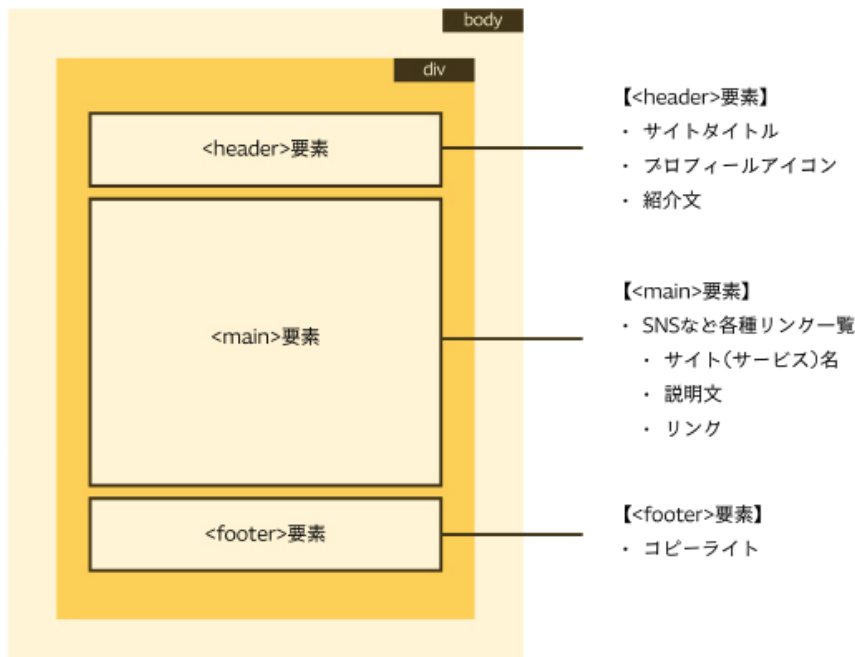
Word ソフトなどで文書を作成する際は、見出しや段落テキスト、リストや表などを使用して文書を読みやすく整形しながら作成しますが、HTML 文書のマークアップもそれと似たようなものです。なお、Word 文書作成における「テキストのサイズ変更」や「テキストの右寄せ」、「表のデザイン変更」が、のちのチャプターで行う「スタイルシート (CSS) 記述によるレイアウトやデザイン調整」に該当します。

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。