Webサイトの仕組み

Webサイトは主にHTML CSSの２つの言語で作られている

HTMLは、Webサイトの文書を見出しや段落に分けるなど、文書を組み立てることが役割

CSSは、HTMLで組み立てた文書に色をつけたりレイアウトを調整したりするなど、文書を装飾することが役割

まとめ

HTMLは文書を組み立てる

CSSは要素の色や位置を整える

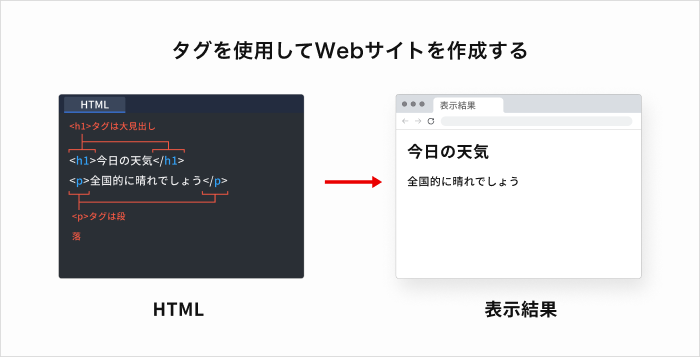
WebサイトはHTML CSSを組み合わせて作られている

HTMLとは（Hyper Text Markup Language）

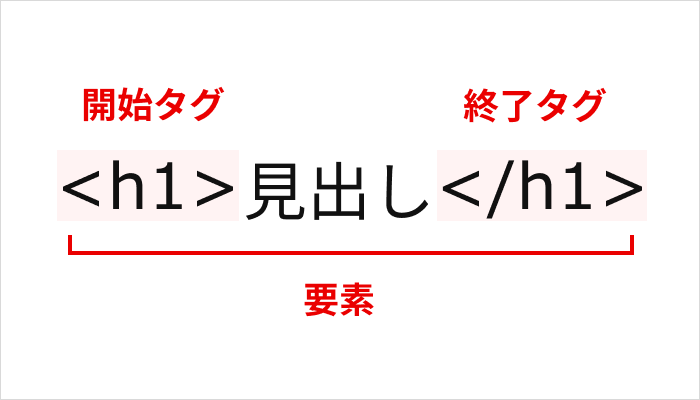
マークアップ言語の1つでWebサイトの文書や見出しや段落に分けるなど、文書を組み立てることが役割

HTMLタグという書式を使ってWebサイトを組み立てていく

HTMLを記述するにはテキストエディタを利用する



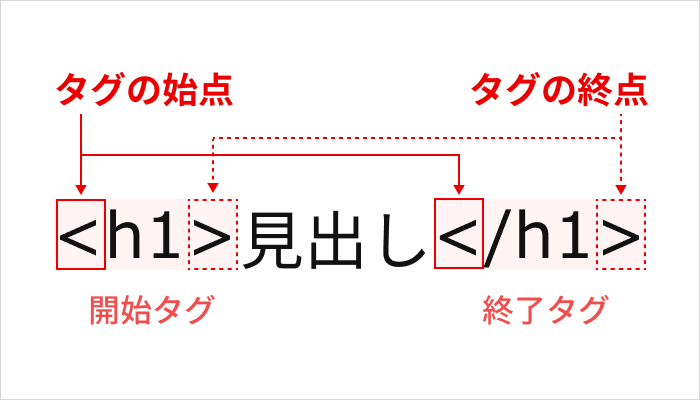
文書の見出しを表示するにはHTMLで以下のように記述する



「見出し」を囲んでいる<h1>と</h1>は「大見出し」を表すタグ

タグで囲まれている領域を要素という

タグは以下のように、開始タグと終了タグで囲んで記述する



<h1>が開始タグ

</h1>が終了タグ

このように様々なタグを用いて文字を囲み出力していく

「ここからここまでが見出し」「ここからここまでが段落」といったことがわかるようにHTMLを記述していく

空要素

終了タグがなく開始タグだけで利用できる要素のこと

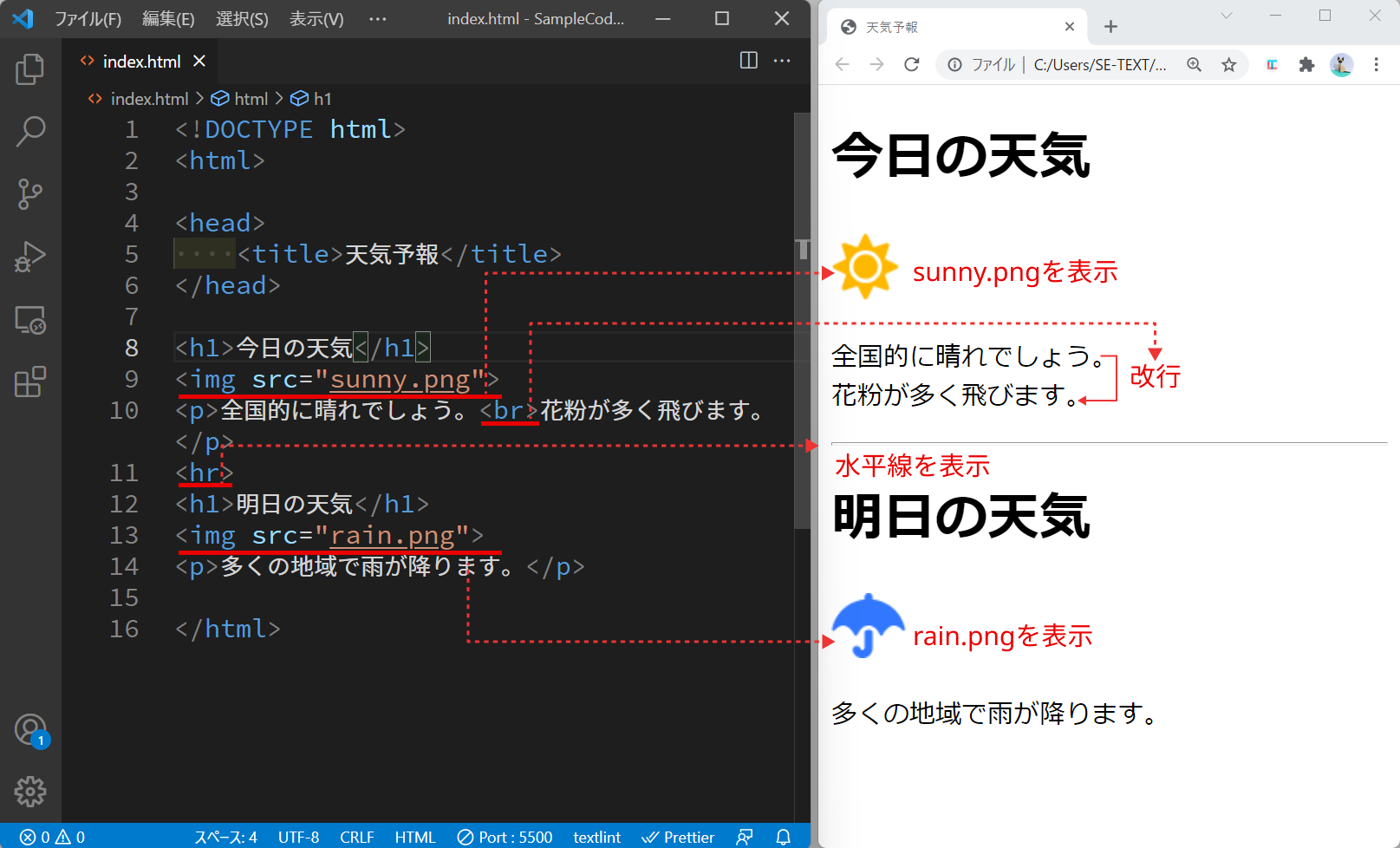
例えば文章を改行するbrや水平線をつけるhrがその一例

改行したり水平線を表示したりする場合、<h1>×××</h1>のように「ここからここまで」という範囲を指定する必要がないため

開始タグのみ記述する

代表的な例

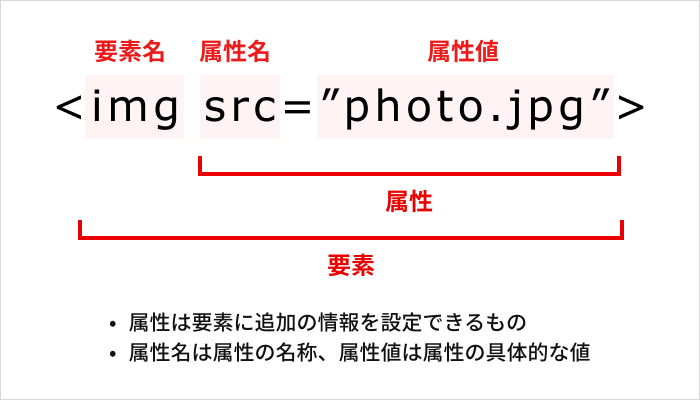
| **空要素** | **役割** | **語源** |
| --- | --- | --- |
| br | 文章を改行する | **br**eak |
| hr | 水平線を表示する | **h**orizontal **r**ule |
| img | 画像を表示する | **im**a**g**e |



要素の属性

属性とはHTMLの要素に追加の情報を設定するためのもの、例えば画像を表示するimg要素では以下の用にsrc属性を記述して表示したい画像ファイルを指定する

<img src="photo.jpg"> 2



属性の書き方

タグ名に続けて属性名と属性値を記述する、属性名と属性値はセットで記述する

1つのタグに対して複数の属性が存在する場合もある、imgaタグであれば、width heightといった属性がある

<img src="photo.jpg" width="200" height="100"> 2



インデントを使ってHTMLの構造を分かりやすくする

インデントとは、行頭に空白を設けて一定量ずらすこと、字下げともいう

Web制作の現場ではインデントを使いHTMLの入れ子構造をわかりやすくするのがマナー



まとめ

・HTMLタグの書き方　開始タグと終了タグで文字を囲む

・空要素は開始タグのみ記述する

・タグと要素の違い　タグで囲まれた中身（文字など）含めるかふぉうか

・属性の種類はタグによって異なる

DOCTYPE宣言

HTMLでまず最初に記述するタグはDOCTYPE宣言

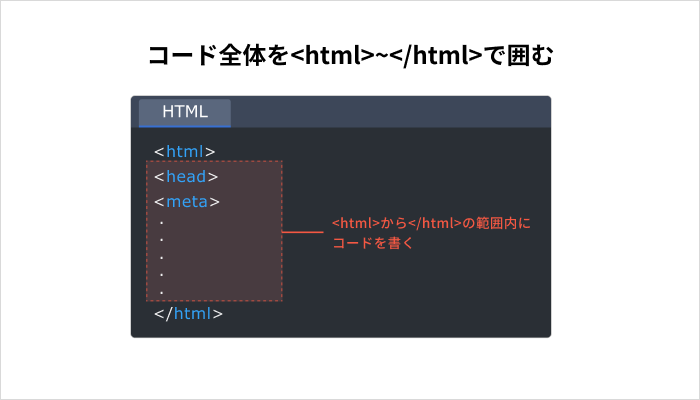
<!DOCTYPE html>　これのこと

「このファイルはHTMLである」ことをブラウザが認識できるようになる

HTML要素とは

<html>タグで囲まれた範囲がhtml要素で「ここからここまでがHTMLによる記述である」ことを意味する

Webサイトに表示したい内容はすべてこのhtmlタグの中に記述する



head要素とは

head要素はWebサイトのタイトルや説明文など、各設定用タグを記述する場所

head要素の中に記述した内容はブラウザには表示されない

**<!DOCTYPE html>**

**<html>**

**+ <head>**

**+ </head>**

**</html>**

titleタグ

そのwebサイトのタイトルを設定する、titleタグはheadタグの中に記述される

<!DOCTYPE html>

<html>

<head>

+ <title>天気予報</title>

</head>

</html>

titleタグで設定されたタイトルは、検索エンジンの検索結果の一覧にwebサイト名として表示される

metaタグ

Webサイトの説明文や文字コードの設定、画面の表示設定などができる

代表的な属性

description

metaタグのdescriptionはWebサイトの説明文を設定する、この説明文はブラウザには表示されず、検索エンジンの検索結果、SNSでシェアされたときに表示される

<!DOCTYPE html>

<html>

<head>

<title>天気予報</title>

+ <meta name="description" content="毎日の気象情報をお知らせします">

</head>

</html>

charset

charset属性は文字コードを設定する、文字コードとは数字しか理解できないコンピュータが文字を認識するための「文字と数字の対応表」

この対応表にはUTF-8、Shift-JISなどの種類があり、どの対応表を使って文字をコンピューターに認識させるのかを設定させる

最近ではUTF-8が主流、設定をミスると文字化けする

要は「文字化けを防ぐためにcharsetで文字コードを設定する必要がある」ということ

<!DOCTYPE html>

<html>

<head>

<title>天気予報</title>

<meta name="description" content="毎日の気象情報をお知らせします">

+ <meta charset="utf-8">

</head>

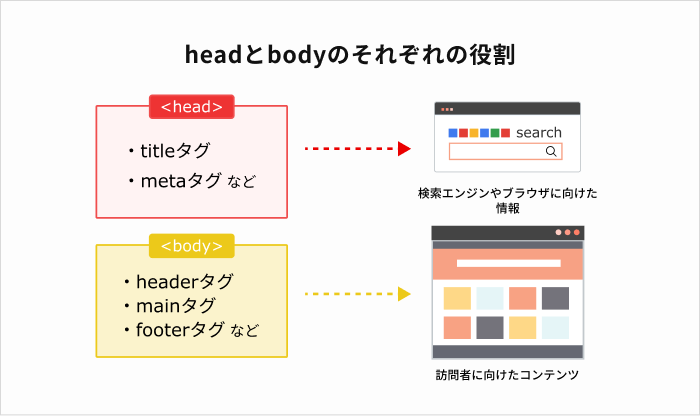
</html>

body要素

body要素にはブラウザに表示される内容を記述する

head要素の中に記述された内容はブラウザには表示されないが、body要素の中に記述された内容はブラウザに直接表示される

<!DOCTYPE html>

<!DOTYPE html>

ｌｌ

bodyタグ

<body></body>の間に見出しや文書、画像などを表示するタグを記述する

<!DOCTYPE html>

<html>

<head>

<title>天気予報</title>

<meta name="description" content="毎日の気象情報をお知らせします">

<meta charset="utf-8">

</head>

+ <body>

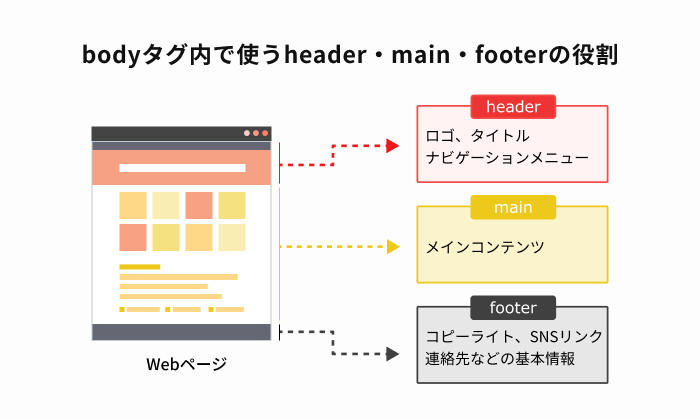
+ </body>

</html>

bodyタグは後述するheaderタグやmainタグ、footerタグと組み合わせて記述する

ブラウザ上のどこに何を表示させるかをブロックのように組み立てていく

| header | ロゴやWebサイトのタイトル |
| --- | --- |
| main | メインのコンテンツ、見た人に伝えたい内容 |
| footer | Webサイトのコピーライトや、ページ内リンクなどの付加情報 |



headerタグ

Webサイトのヘッダーを作るためのタグ、ヘッダーとはwebサイトのタイトルやロゴなどを表示する画面上部のエリアのこと

<!DOCTYPE html>

<html>

<head>

<title>天気予報</title>

<meta name="description" content="毎日の気象情報をお知らせします">

<meta charset="utf-8">

</head>

<body>

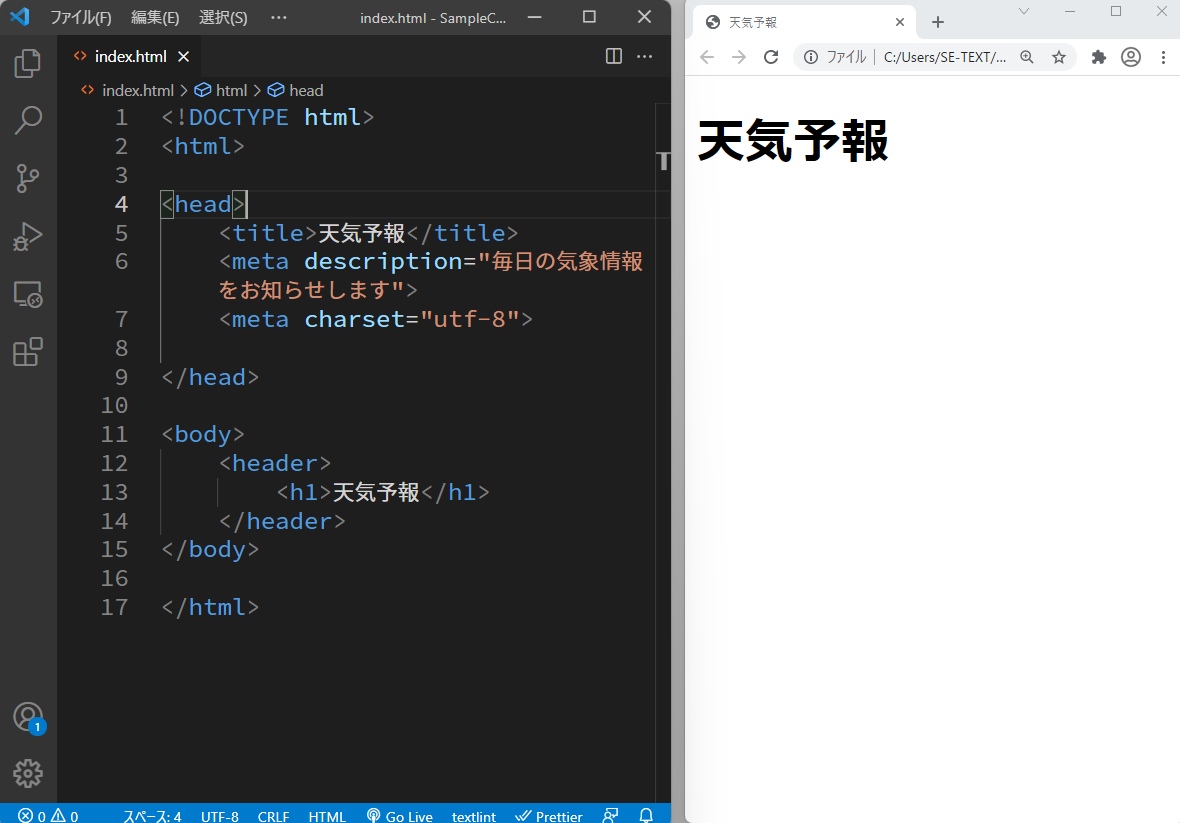
+ <header>

+ <h1>天気予報</h1>

+ </header>

</body>

</html>



mainタグ

Webサイトのメインコンテンツを作るためのタグ

mainタグの中にそのWebサイトを見た人に一番伝えたい見出しや段落、画像など並べていく

<!DOCTYPE html>

<html>

<head>

<title>天気予報</title>

<meta name="description" content="毎日の気象情報をお知らせします">

<meta charset="utf-8">

</head>

<body>

<header>

<h1>天気予報</h1>

</header>

+ <main>

+ <h2>今日の天気</h2>

+ <p>よく晴れるでしょう</p>

+ </main>

</body>

</html>



footerタグ

Webサイトのフッターを作るためのタグ、フッターとはコピーライト（著作権表示）や他のページへのリンクなどＷｅｂサイトの付加情報を表示すえう画面下部のエリアのこと

<!DOCTYPE html>

<html>

<head>

<title>天気予報</title>

<meta name="description" content="毎日の気象情報をお知らせします">

<meta charset="utf-8">

</head>

<body>

<header>

<h1>天気予報</h1>

</header>

<main>

<h2>今日の天気</h2>

<p>よく晴れるでしょう</p>

</main>

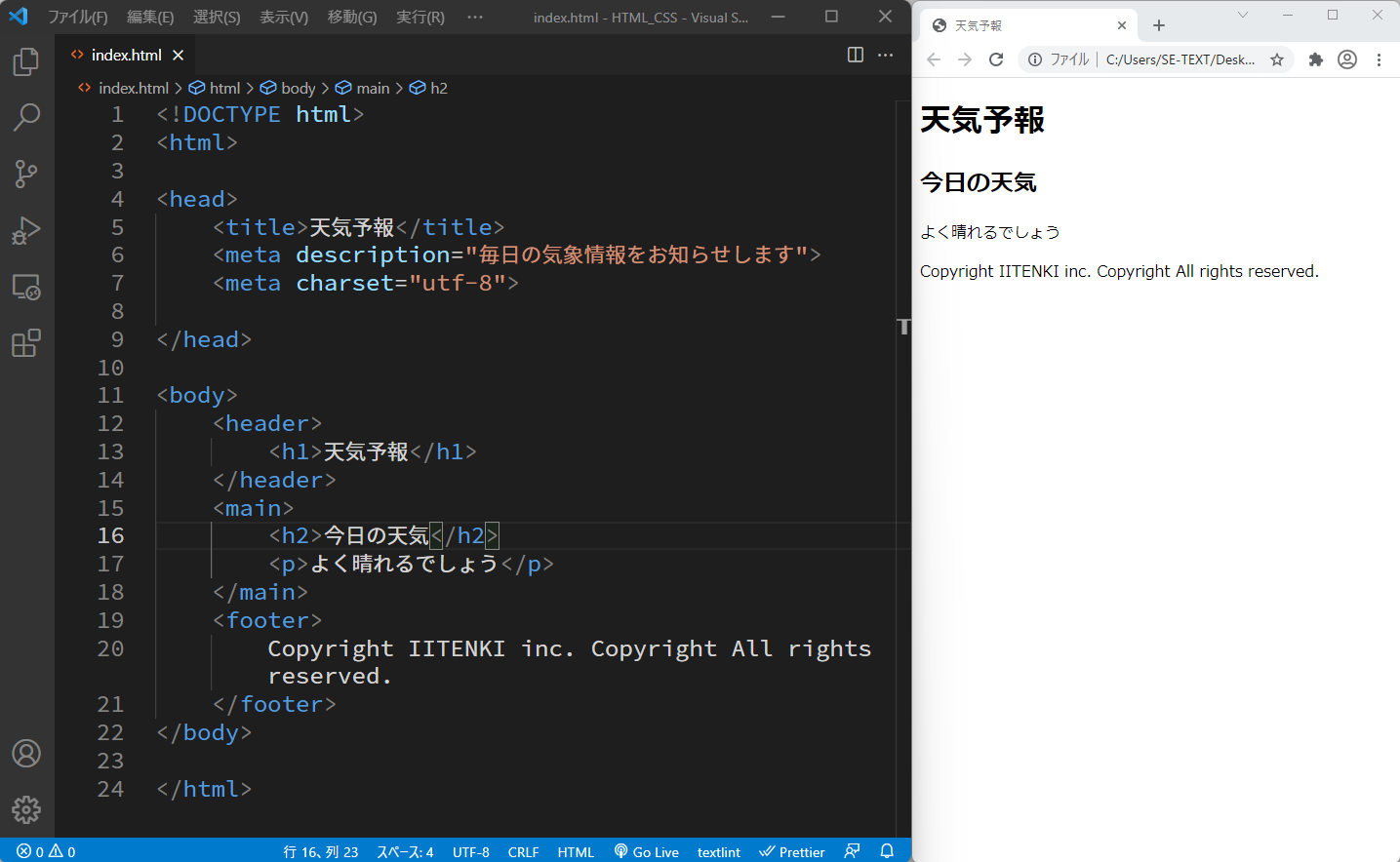
+ <footer>

+ <p>Copyright IITENKI inc. Copyright All rights reserved.</p>

+ </footer>

</body>

</html>



まとめ

・DECTYPE宣言で「このファイルがHTMLである」という情報をブラウザに伝える

・htmlタグを記述することで、その中身がHTMLであることを示す

・headタグ内にtitleタグやmetaタグ、charsetタグを記述してタイトルや説明文などを設定する

・headeraタグでタイトルやロゴ、mainタグでメインコンテンツ

footerタグでコピーライトなどを表示するエリアを作る

コメントアウト

+ <!DOCTYPE html>

+ <html>

+ <body>

+ <!--ここはヘッダーです-->

+ <header>

+ <h1>天気予報</h1>

+ <!--

+ <p>このホームページは全国天気予報を紹介します。</p>

+ <p>４時間ごとに最新情報に更新します。</p>

+ -->

+

+ </header>

+ </body>

+ </html>

pタグに書かれた文章は一時的にコメントアウトして非表示にしている

<!--と-->で囲んだ部分がコメントアウトされた状態となり、ブラウザ非表示になる

pタグ

文章を段落にするタグ

+ <!DOCTYPE html>

+ <html>

+ <body>

+ <p>

+ 今日は全国的に雨でしょう。

+ 明日には雨は止んで晴れますが、非常に冷え込むので注意してください。

+ 週末は気持ちよく晴れるでしょう。

+ </p>

+ </body>

+ </html>

pタグで囲んだ文章が段落としてブラウザに表示される

hタグ（見出し）

文章を見出しにするタグ

+ <!DOCTYPE html>

+ <html>

+ <body>

+ <h1>気象情報</h1>

+ <h2>全国天気</h2>

+ <p>全国的に晴れのところが多いです。</p>

+ <h3>愛知県</h3>

+ <p>愛知県西部は曇りところにより雨、東部は晴れでしょう。</p>

+ <h4>愛知県西部</h4>

+ <p>海側のエリアで一時雨が降ります。</p>

+ <h5>名古屋市</h5>

+ <p>一部の区で雨が降ります。</p>

+ <h6>名古屋市港区</h6>

+ <p>午後は雨が降ります。</p>

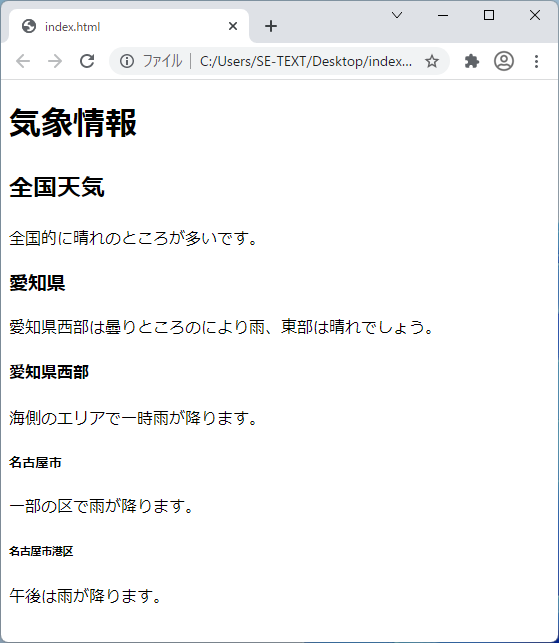
+ </body>

+ </html>

ｈタグで囲んだ文章には見出しとして表示され、ｐタグで囲んだ段落と比べると大きな文字になる

ｈタグにはｈ１～ｈ６まで種類があり、ｈ１が大見出しで、ｈ２移行が小見出し

それぞれの見出しの違い



ul/ol/liタグ（箇条書き）

ul は文章を箇条書き（行頭に「・」がつく）にするタグ

ol　は文書を番号付きの箇条書き（行頭に「１、２、３」がつく）タグ

ulの使用例

箇条書きにするにはulタグとliタグを組み合わせて記述する

ulタグを記述すると、行頭に「・」が表示される

+ <!DOCTYPE html>

+ <html>

+ <body>

+ <h1>今週の天気予報</h1>

+ <ul>

+ <li>明日は晴れです</li>

+ <li>明後日は雨です</li>

+ <li>週末は曇りでしょう</li>

+ </ul>

+ </body>

+ </html>



olの使用例

番号付きの箇条書きにするには、olタグとliタグを組み合わせて記述する

olタグを記述すると、行頭に１から順番に番号が表示される

+ <!DOCTYPE html>

+ <html>

+ <body>

+ <h1>即席ラーメンの作り方</h1>

+ <ol>

+ <li>ふたを開ける</li>

+ <li>お湯を注ぐ</li>

+ <li>ふたを閉める</li>

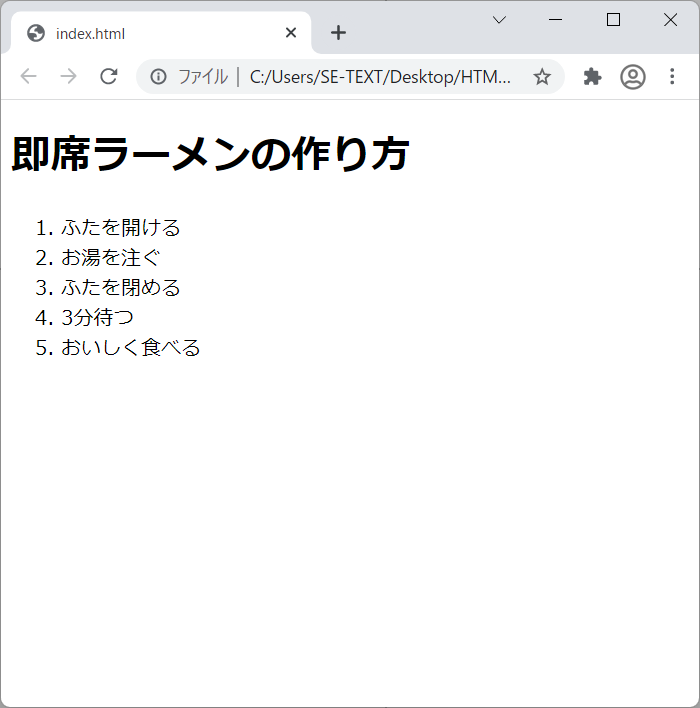
+ <li>3分待つ</li>

+ <li>おいしく食べる</li>

+ </ol>

+ </body>

+ </html>



hrタグ

hrは水平線を表示するタグ

段落と段落の間に水平線を入れる事でWebサイト内で区切りとして利用する

+ <!DOCTYPE html>

+ <html>

+ <body>

+ <h2>午前のニュース</h2>

+ <p>動物園でゾウの赤ちゃんが生まれました。</p>

+ <hr>

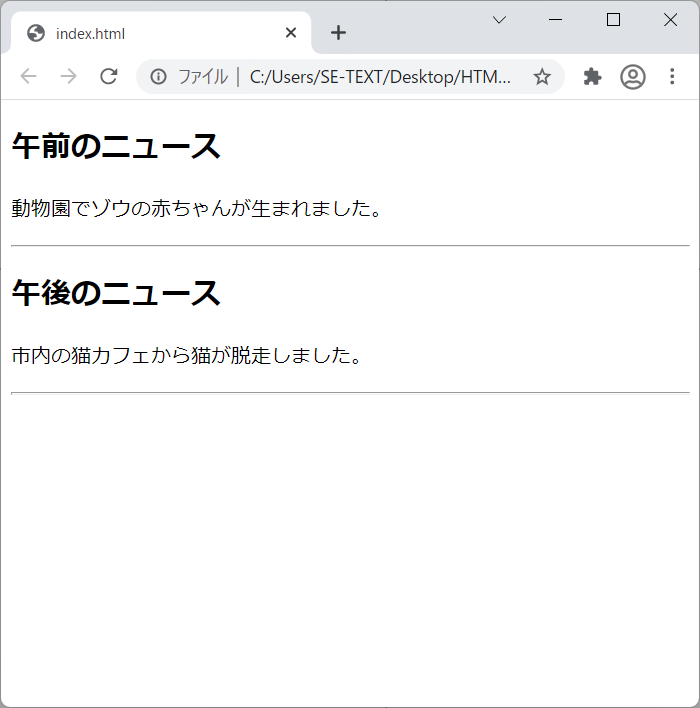
+ <h2>午後のニュース</h2>

+ <p>市内の猫カフェから猫が脱走しました。</p>

+ <hr>

+ </body>

+ </html>



brタグ

改行するタグ

+ <!DOCTYPE html>

+ <html>

+ <body>

+ <p>国道1号線が交通事故の影響で渋滞しています。<br>迂回路は県道48号線です。</p>

+ </body>

+ </html>



divタグ

複数の要素をdivタグで囲むことで、要素をグループ化できる

webサイトのレイアウトを組み立てたり、複数の要素をグループ化してひとまとめにしたりする用途で使われる

なぜdivタグで要素をグループ化するのかというと、CSSでまとめて装飾を適用できるから

要は「divタグでグループ化しておけば装飾が楽になる」というイメージ

+ <!DOCTYPE html>

+ <html>

+ <body>

+ <div>

+ <h2>午前のニュース</h2>

+ <p>動物園でゾウの赤ちゃんが生まれました。</p>

+ </div>

+ <div>

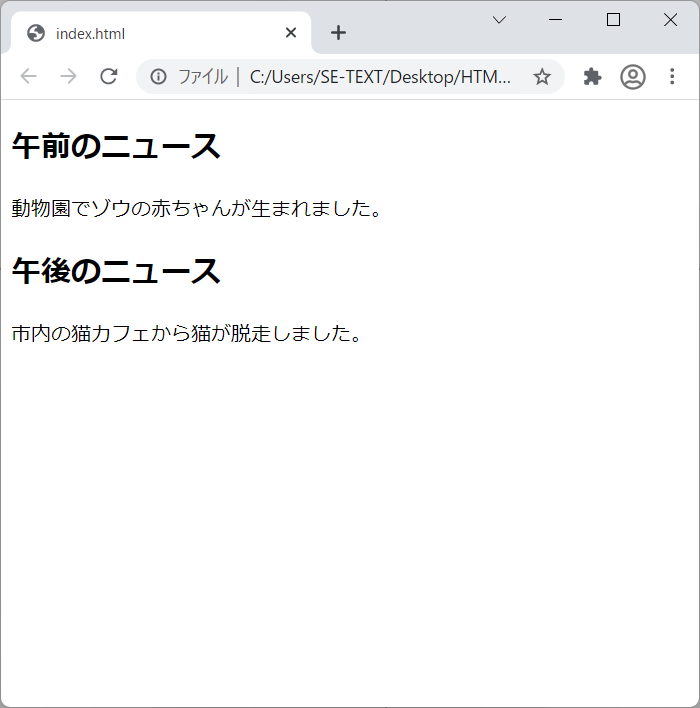
+ <h2>午後のニュース</h2>

+ <p>市内の猫カフェから猫が脱走しました。</p>

+ </div>

+ </body>

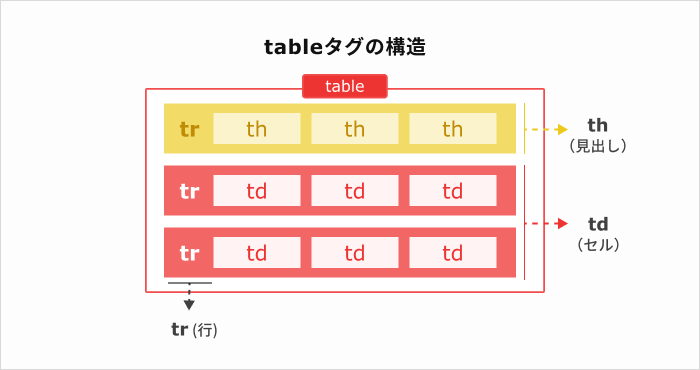
+ </html>



tableタグ

表を作るタグ、以下のタグを組み合わせて記述していく

| **タグ** | **用途** | **語源** |
| --- | --- | --- |
| tr | 行 | **t**able **r**ow |
| th | 見出し | **t**able **h**eader |
| td | セル | **t**able **d**ata |



tableタグ

表の作成を宣言するタグ、表全体をtableタグで囲む

trタグ

表の行を作る

thタグ

表の見出しを作る

tdタグ

表のセルを作る

+ <!DOCTYPE html>

+ <html>

+ <body>

+ <h1>ランチメニュー</h1>

+ <table border="1">

+ <tr>

+ <th>平日</th>

+ <th>土日祝日</th>

+ </tr>

+ <tr>

+ <td>Aランチ：唐揚げ定食</td>

+ <td>Aランチ：コロッケ定食</td>

+ </tr>

+ <tr>

+ <td>Bランチ：ハンバーグ定食</td>

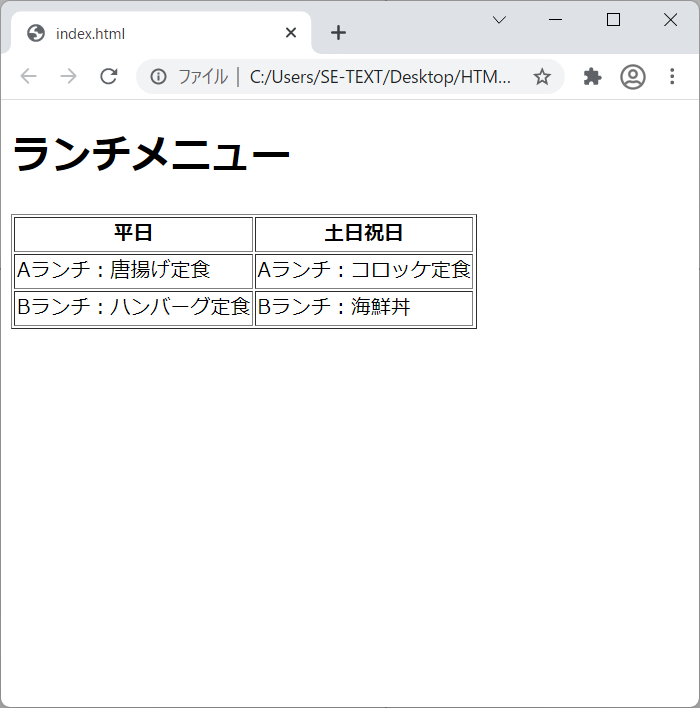
+ <td>Bランチ：海鮮丼</td>

+ </tr>

+ </table>

+ </body>

+ </html>



| **タグ** | **用途** |
| --- | --- |
| p | 段落にする |
| h1~h6 | 大見出し、小見出しにする |
| ulとli | 箇条書きにする |
| olとli | 番号付きの箇条書きにする |
| hr | 水平線を表示する |
| br | 改行する |
| div | 要素をグループ化する |
| table | 表を作る |

このファイルに記載されている内容

decoration

fontfamily

fontstyle

display

width min max

height min max

box-model

幅のauto

.parent {

width:auto;

background-color: aqua;

}

幅の最小値

.child {

width: 50%;

min-width: 300px;

background-color: brown;

}

幅の最大値

.parent {

width: 80%;

max-width: 600px;

background-color: brown

}

高さの最小値 最大値

.parent {

height: auto;

min-height: 100px;

max-height: 100px;

}

最大値最小値は指定した高さの数値によって高さや幅が決まる

　 子要素の高さが指定した数値を越えるとと子要素がはみ出る

基本的なプロパティ

color 文字色を指定する

font-family フォントの種類を指定する

font-size フォントの大きさを指定する

font-weight フォントの太さを指定する

background 背景を指定する

text-align 横方向の揃え方を指定する（左・右・中央）

line-height 行の高さを指定する

width 要素の幅を指定する

height 要素の高さを指定する

padding 要素のパディングを指定する

margin 要素のマージンを指定する

border 要素のボーダーの色や太さを指定する

position 要素の配置方法を指定する

display 要素の表示形式を指定する

複数のプロパティを参照する

・複数のプロパティを指定するには、以下のようにセミコロン;のあとに続けて書くだけです。

例

p {

color: red;

font-size: 48px;

text-align: right;

}

/\* CSSのコメントアウト\*/

コメントアウトのショートカットキーは Ctrl+/

CSSのコード内で使われる半角スペース・タブ・改行は、ブラウザ上の表示に影響しない

全角スペースはNG

decoration

fontfamily

fontstyle

.decoration {

font-size: 60px;

}

/\* デコレーション色々 \*/

p{ ←セレクタ運用する対象「どこ」＝「今回はｐタグ」

color:red;

color=プロパティ「何を」=「今回は色を」

red=値 運用の仕方「どうする」=「赤にする」

CSSは上記のようにセレクタ、プロパティ、値の３つを書く

セレクタ：スタイルを適用する場所

プロパティ：適用するスタイルの内容

値：スタイルの適用の仕方

CSSにおける大きさの単位

「px パソコンやスマホの画面を構成する最小単位のこと」

メリット：画面幅やデバイスに関係なく大きさを固定したいときに便利

デメリット：大きさが固定なので、レスポンシブデザインには不向き

「em 親要素のフォントサイズを１とする単位、つまり同じ1emでも親要素のフォントサイズによって大きさが変わる」

例えば親要素のフォントサイズが16pxであれば「1em=16px」になり、親要素のフォントサイズが12pxであれば

1em＝12pxになる

メリット：サイズが相対的に変わるので、レスポンシブデザインに向いている

デメリット：親・子・孫要素といった入れ子状態で各要素にemを使うと計算が複雑になる

「rem emが親要素のフォントサイズを基準とするのに対し、remはルート要素のフォントサイズを基準とする」

ルート要素とは、そのファイルにおける最上位の親要素のことで、他のすべての要素を囲んでいます。HTMLファイルにおけるルート要素はhtml要素

rem＝root emの略で「root（根本、ルート）のem」という意味

「％　親要素の大きさを基準として割合を指定する」

親要素が16pxのとき、50%を指定すれば8pxになり、200%を指定すれば32pxになります

「vw Viewport Widthの略で、ビューポート（Webサイトの表示領域）の横幅を基準とする単位です。」

100vwで表示領域の横幅いっぱい（100%）に広がり、1vwで表示領域の1%の横幅になります。

画面に表示されているWebサイトの幅が例えば300pxであれば、「100vw＝300px」「1vw＝3px」となります。

メリット：親要素の影響を受けないため、大きさの計算がしやすい

デメリット：ビューポートの横幅にはスクロールバーが含まれるため、100vwを指定すると想定よりも幅が大きくなることがある

「vh Viewport Heightの略で、ビューポート（Webサイトの表示領域）の高さを基準とする単位です。」

100vhで表示領域の高さいっぱい（100%）に広がり、1vhで表示領域の1%の高さになります。

メリット：親要素の影響を受けないため、大きさの計算がしやすい

デメリット：スマートフォンにおけるアドレスバー（開いているページのURLなどが表示されている領域）の高さが考慮されないため、100vhを指定しても画面の高さいっぱいに表示されない（アドレスバーの高さ分下に押し出される）

太字を指定

font-weight: bold;

斜体を指定

font-family: "游ゴシック体", "YuGothic", "游ゴシック", "Yu Gothic";

font-style: italic;

フォントの種類によっては斜体にならないため、ここではfont-familyプロパティでフォントの種類を指定しています

行間を変える

line-height: 4.5;

文字の間隔を変える

letter-spacing: 1.5em;

値の指定にはフォントサイズと同じようにpxなどが使えますが、emを使うことで、フォントサイズに応じて相対的に字間が決まるのでおすすめです。

例えば値に1.5emを指定すれば、フォントサイズの1.5倍の字間になります。

文字に下線を引く

text-decoration: underline;

取り消し線を引く

text-decoration: line-through;

文字を枠線で囲む

{border: solid 3px red;}

solid=線の種類

3px=線の太さ

red=線の色

dotted=点線で囲む

dashed=破線で囲む

double=二重線で囲む

上にだけ線を引く=border-top

下にだけ線を引く=border-bottom

左にだけ線を引く=border-left

右にだけ線を引く=border-right

文字の背景色を指定する=background-color:

文字に影をつける=text-shadow: 1px 1px 2px gray;

1px=影の右方向の長さ 1px=影のした方向の長さ 2px=影のぼけ具合 gray=影の色

文字の位置を変える text-align: 〇〇;

left=左揃え

center=中央揃え

right=右揃え

文字の一部だけ装飾する

<span>

使い方

<p>文字の<span class="text-bold">一部</span>だけ装飾する</p>

フォントの種類を指定する

body{

font-family: "Arial", "Meiryo", sans-serif;

}

基本的にfont-familyはWebページ全体の文字に対してまとめて適用するため、上記の例ではbody要素をセレクタに指定しています。

フォントファミリー名とは、「Arial」や「Meiryo（メイリオ）」などフォントの具体的な名前のことです。

font-familyプロパティに指定できる代表的な総称フォント名は以下のとおりです。

sans-serif（ゴシック体）

serif（明朝体）

cursive（筆記体）

fantasy（装飾体）

monospace（等幅）

優先したいフォントを先に書く

フォントを指定するときは、その順番にも注意が必要です。

font-familyプロパティでは、前に書かれたフォントが優先的に使われます。よって、優先的に表示させたいフォントを先に書くようにしましょう。

body {

font-family: "Meiryo", "Hiragino Kaku Gothic ProN";

}

英語フォントを先に書く

フォントには英語だけを表示できる英語フォントと、英語と日本語の両方を表示できる日本語フォントがあります。

英語フォントと日本語フォントを使い分けたい場合は、英語フォントを先に書くようにしましょう。

body {

font-family: "Meiryo", "Arial";

}

しかし、以下のように英語フォントであるArialを先に書けば、英語はArial、日本語はメイリオで表示されます。

body {

font-family: "Arial", "Meiryo";

}

最後に総称フォント名を書く

指定したフォントがすべて表示されなかった場合に備え、sans-serif（ゴシック体）やserif（明朝体）といった総称フォント名を最後に書くようにしましょう。

例えば以下のようにフォントを指定した場合、WindowsであればＭＳ Ｐゴシックが表示されますが、ＭＳ ＰゴシックがインストールされていないMacではsans-serif（ゴシック体）のフォントが表示されます。

Cssファイル見本

body{

font-family: "MS Pゴシック", sans-serif;

HTMLファイル見本

<!DOCTYPE html>

<html>

<head>

<title>タイトルです</title>

<meta name="description" content="説明文です">

<meta charset="utf-8">

<link rel="stylesheet" href="font-family.css">

</head>

<body>

<p>Englishと日本語です</p>

<p class="decoration">Englishと日本語です</p>

</body>

</html>

font-familyプロパティで指定するフォントはデバイスにインストールされていなければ表示されません。よって、複数指定するといった工夫が必要でした。

この問題を解決する方法が他にもあります。それがWebフォントです。

Webフォントとは、インターネットのサーバー上に置かれたフォントのことです。

Webページを表示するときにインターネット上から呼び出して使うため、ユーザーのデバイスに依存しない（デバイスにそのフォントがインストールされていなくてもOK）という大きなメリットがあります。

ただしデメリットとして、利用料がかかる場合や表示速度が遅くなる場合があります。よって、Webフォントを利用するかどうかはWebサイトの制作を依頼したクライアント（依頼主）の意向次第となります

CSSファイル見本

.google-fonts {

ここにGoogleフォントを指定

}

Goolgleフォントの指定の仕方

ブラウザからGoogle fontsにアクセス

Languageを選択

指定したいフォントのソースコードをコピーして貼り付け

font-family: "Noto Sans JP", sans-serif;

#second-paragraph {

padding: 100px;

border: solid 5px yellow;

}

content 文字や画像など、要素の中身が表示される領域

コンテンツの大きさは、width（幅）height（高さ）

border 　コンテンツの周辺にある境界線のこと、初期設定だとボーダーが0なので表示されない

padding ボーダーの内側の余白、大きさはpaddingで指定できる

margin ボーダーの外側の余白、大きさはmarginで指定できる

余白の値を上→右→下→左の順で一度に指定する

p{

padding: 20px 15px 10px 5px;

margin: 15px 10px 5px 0px;

このように上記の順番で別々に指定できる。「上から時計周り」と覚える

余白の値を上下に左右に分け指定する

「上下の余白は20px」「左右の余白は10px」という感じで、上下と左右に分けて指定することもできる

p{

padding: 20px 10px;

margin: 40px 20px;

}

上下左右の余白の一部だけを指定する

上だけ指定する　padding-top margin-top

下だけ指定する padding-bottom margin-bottom

左だけ指定する padding-left margin-left

右だけ指定する padding-right margin-right

marginのを利用して中央揃えにする

p{

margin-left: auto;

margin-right: auto;

}

上下に0、左右にautoを指定している

p{

margin: 0 auto;

}

display 要素の表示形式（要素をどのように表示するか）

displayプロパティの値

block 要素が縦に並んでいく

inline 要素が横に並んでいく

inline-block 要素が横に並んでいくが、blockと同じような性質を持つ

none 要素が非表示になる

display: block;

以下のような性質を持っている

1.要素が縦に並んでいく

2.width heightで幅と高さを指定できる

3.padding margin上下左右に余白を指定できる

#id-name {

width: 200px

background-color: yellow;

}

display: inline;

以下のような性質を持っている

1.要素が横に並んでいく

2.幅と高さを指定できない

3.左右の余白のみ調整できる　margin

inlineの要素にpaddingを指定する時は左右だけにしておく（不具合でるらしい

display: inline-block;

1.要素が横に並んでいく

2.width heightで幅と高さを調整できる

3.padding marginで上下左右に余白を設定できる

「要素が横に並んでいく」という以外はblockと同じような性質

初期値がdisplay: block　であるp要素にdisplay: inline;

を指定すれば、そのp要素はblockではなくinlineになる

幅と高さを指定したinline-block

#id-name {

display: inline-block;

width: 100px;

height: 100px;

background-color: color-name

}

上下左右に余白を指定したinline

#id-name {

display: inline-block;

width: 100px;

height: 100px;

padding: 10px;

margin: 10px;

background-color: color-name

}

display none

その要素は非表示になるもの

どういった時に用いられるかというとレスポンシブデザインを作るとき

つまり「PCではナビメニューを表示させて、スマホでは非表示にしたい」ときなどに用いられる

#id-name{

display: none;

background-color: color-name;

}

ｌ