CSS の基本

CSS は文書を（スタイルやレイアウトを）どのように表現するか指定する言語です。
CSS の作成方法と HTML との連携について説明します。

HTML と CSS の連携方法は 3 種類あります。
1．HTML と CSS を別なファイルで作成して連携する。
2．HTML 内に STYLE タグを用いて記述する。
3．タグの STYLE 属性に記述する。

現在では1の方法のみで作成されることが多いです。メリットは次の通りです。
- HTML と CSS を別のファイルで記述できる
- 複数の HTML にひとつの CSS を連携できる
- ひとつの HTML に複数の CSS を連携できる
- HTML と CSS の組み合わせを自由に変更できる

実際に作成してみましょう。HTML と CSS を個別のファイルで作成します。

HTML のサンプル（basic/index.html）

```
<!DOCTYPE html>
<html lang="ja">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS の基本</title>
    <link rel="stylesheet" href="./css/style.css">
</head>
<body>
    <p class="red">赤</p>
    <p class="blue">青</p>
    <p class="green">緑</p>
</body>
</html>
```

CSS のサンプル(basic/css/style.css)

```
@charset "utf-8";

.red {
    color: red;
}
.blue {
    color: blue;
}
.green {
    color: green;
}
```

解説
HTML と CSS との連携は LINK 要素を使います。

1.CSS とのリンク

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

<link>を用います。属性は次の通りです。

rel 属性
リンクの種別を指定します。ここではスタイルシートを指定しています。種別の一覧はこちらです。

href 属性
リンクしたリソースの URL を指定します。URL は絶対・相対のどちらでもかまいません。

2.対象の要素を識別できるようにする
次に対象となる要素にグローバル属性である id 属性もしくは class 属性を設定します。

id 属性
文書全体で一意でなければならない識別子（ID）を定義します。

class 属性
要素のクラスを空白区切りで並べたリストで、大文字小文字を区別します。
これにより CSS の対象を複数の要素に指定できます。

例）

```
<p class="red">赤</p>
<p class="blue">青</p>
<p class="green">緑</p>
```

それぞれの<p>にたいして red,blue,green のクラス名が指定されました。
これを用いて css の対象を選択できます。

次に CSS 側の記述を説明します。

```
@charset "utf-8";

.red {
    color: red;
}
```

@charset
タイルシートで使う文字エンコーディングを定義します。この規則はスタイルシートの最初の要素でなければならず、これより前には文字を一切記述してはいけません。

cssの文法
cssは次の文法があります。

```
セレクタ {
    プロパティ: 値;
}
```

セレクタ
cssの適用対象を記述します。セレクタの書き方は様々です。こちらを見てまとめましょう。
「.red」はクラスの指定ですので、「class="red"」と記述した要素が対象になります。

プロパティ「color」は文字の色（前景色）を指定します。これは種類が多いため、こちらを参考にまとめます。

値についてはキーワードや数値で指定します。単位などについてはこちらを参考にまとめます。

これらを読み取ると、「クラス名 red の要素の文字色を赤にする」と読み取れます。

文字に関する CSS

文字に関する CSS からまとめます。文字はページの印象が変わるほど重要な部分です。
基本的な事柄から理解をしましょう。

要素内のテキストは要素のコンテンツボックス内にレイアウトされます。コンテンツボックスとはその要素が描画する矩形の領域の事です。そして次のルールに従って表示されます。

- 左から右へ表示
- コンテンツボックスの右端で折り返して次行に表示
- それ以外では改行されない

HTML のサンプル（font-style/index.html）

```
<!DOCTYPE html>
<html lang="ja">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>テキストについて</title>
</head>
<body>
    <p>適当な文字をたくさん</p>
</body>
</html>
```

ブラウザの表示幅を狭めると改行されたように見えます。意図して改行を挿入する場合は
を用います。


```
<p>適当な文字をたくさん<br>これは改行されます</p>
```

では文字に関する CSS を見ていきましょう。
テキストを装飾するために使用される CSS プロパティは、一般的に次の 2 つのカテゴリに分かれます。

- フォントスタイル
  テキストに適用されるフォントに影響するプロパティで、適用するフォント、大きさ、太字、斜体などに影響します。次のプロパティがあります。

  color text-decoration font-family font-size font-style font-weight
  text-transform text-shadow

- レイアウトスタイル
  テキストの間隔やその他のレイアウト機能に影響するプロパティで、例えば、行間や文字間のスペースや、コンテンツボックス内でのテキストの配置方法などを操作できます。
  次のプロパティがあります。

  text-align line-height letter-spacing word-spacing

それでは具体的に CSS を記述してみましょう。
外観に直結した重要なプロパティを例にしてみましょう

HTML の追記

```
    <p id="font1">cyan 24px serif bolid italic</p>
    <p id="font2">#00ffff 1.2rem sans-serif lighter italic</p>
    <p id="layout1">text-align: right line-height: 2em</p>
    <p id="layout2">text-align: left line-height: 2em</p>
```

CSS のサンプル(font-style/css/style.css)

```
@charset "utf-8";

#font1 {
    color: cyan;
    font-size: 24px;
    font-family: serif;
    font-weight: bold;

}

#font2 {
    color: #00ffff;
    font-size: 1.5rem;
    font-family: sans-serif;
    font-weight: lighter;
    font-style: italic;
}

#layout1 {
    text-align: right;
    line-height: 2em;
}

#layout2 {
    text-align: center;
    line-height: 160%;
}
```

それぞれの効果は次の通り

font1	font2	layout1	layout2
前景色:シアン フォントサイズ:24px 書体:明朝 太さ:太字	前景色:#00ffff フォントサイズ:1.5rem 書体:ゴシック 太さ:細字	水平揃え:右揃え 行の高さ:2 文字分	水平揃え:中央揃え 行の高さ:160%

- フォントの一括指定
  font プロパティを用いて様々なフォント関連プロパティを設定できます。
  詳細はリンクから確認をして欲しいのですが、重要なポイントはまとめます。

  font-size と font-family が必須です。
  font-style, font-variant, font-weight は font-size よりも前に記述する。
  line-height は font-size の直後に、"/" で区切って、"16px/2" のように指定します。
  font-family は指定される最後の値である必要があります。

  htmlの追記

```
    <p id="font3">bolid 16px/2 "メイリオ"</p>
```

  cssの追記

```
#font3 {
    font: bolid 16px/2 "メイリオ";
}
```

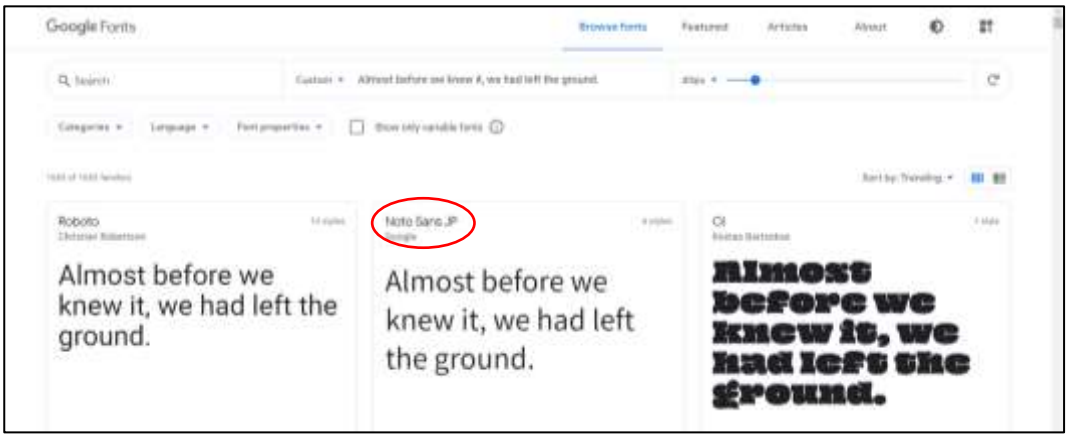
  フォントに関してはサイトのイメージを左右するので、ルート要素で設定するのが定番です。

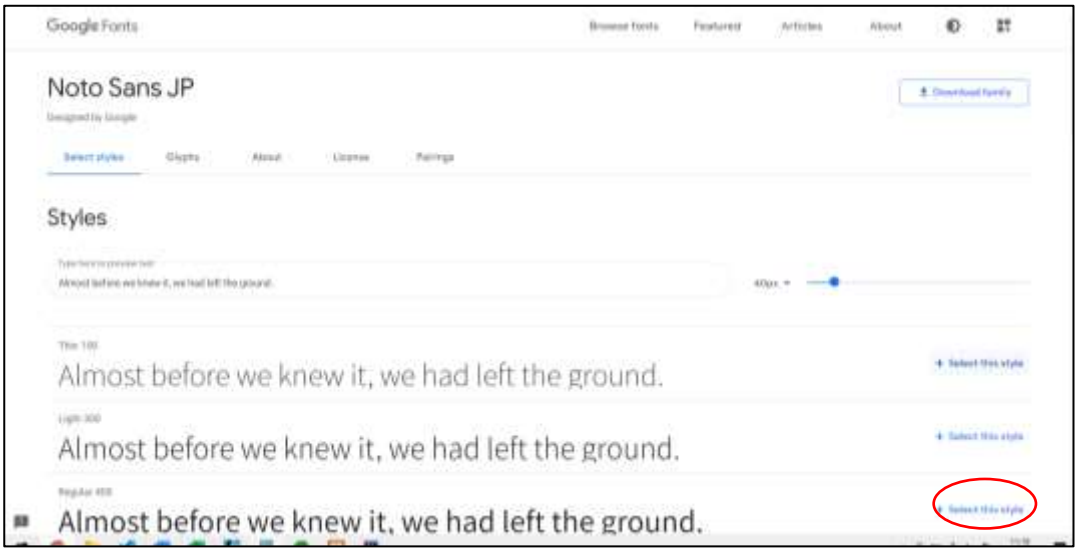
```
body {
    font-family: "遊ゴシック","ヒラギノ角ゴ ProN", sans-serif;
}
```

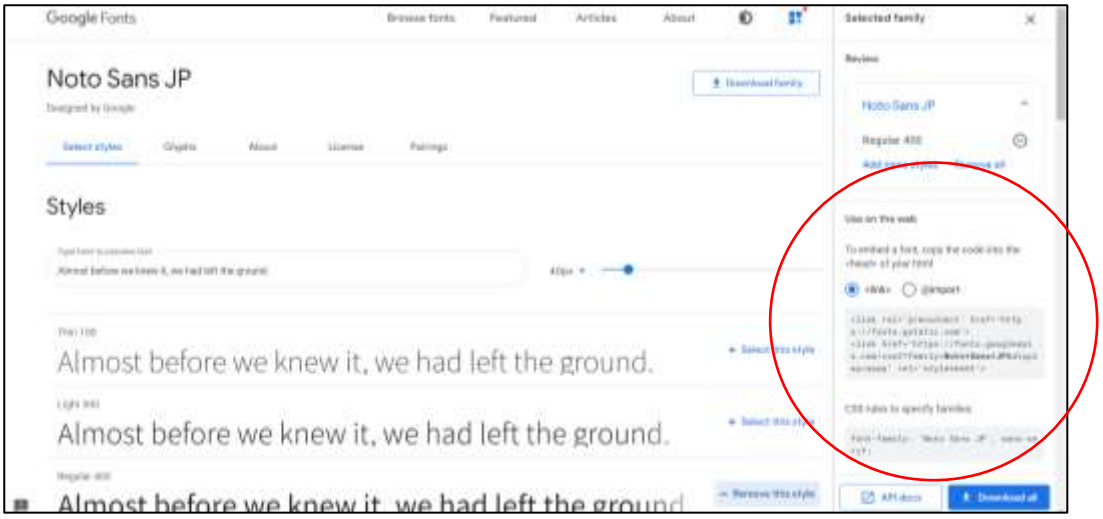
  そして各OSに対応したフォントを設定していきます。最後はゴシック体のどれかという保険的な記述です。

- ウェブフォントついて
  インターネット上に公開されているフォントであり、表示するユーザーが所持していない場合にでも期待されるフォントで表示されるメリットがある。

  メジャーなフォントに GoogleFonts がある。ここではウェブフォントの設定について理解しよう。
  ここでは日本語フォントである Noto Sans JP を設定してみる。







最後の画像に link 情報と css が表示されるのでコピーして使用します。

次の見本を参考に作成してみましょう。

ウェブフォントの適用の見本（web-font/index.html）

```
<!DOCTYPE html>
<html lang="ja">
    <head>
        <meta charset="UTF-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>ウェブフォントの設定</title>
        <link rel="preconnect" href="https://fonts.gstatic.com">
         <link
        href="https://fonts.googleapis.com/css2?family=Noto+Sans+JP&display=swap"
        rel="stylesheet">
        <link rel="stylesheet" href="./css/style.css">
    </head>
    <body>
        <h1>見出し</h1>
        <p>内容</p>
    </body>
</html>
```

ウェブフォントの適用の CSS 見本（web-font/css/style.css）

```
@charset "utf-8";

html {
    font-family: 'Noto Sans JP', sans-serif;
}
```

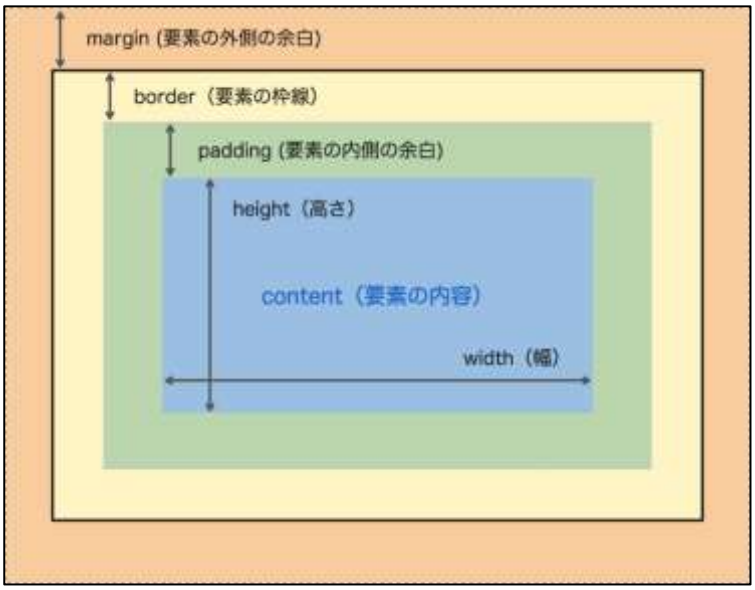
開発者ツール（F12）でウェブフォントに関する CSS が適用されているのを確認しよう。

ボックスモデルについて

CSS にはボックスの概念があります。これを理解することで様々なレイアウトを作成できます。
では基本的な知識であるボックスモデルについて理解しましょう。

1. ボックスモデルとは
   HTML の各要素は幅と高さがある矩形の形で描画される特徴があります。これをボックスモデルと呼びます。

2. ボックスモデルの構成要素
   ボックスの構成要素は次の 4 つです。



- コンテンツ(content)
  要素内に記述された文字・マルチメディア(画像・動画)そのもののこと。
  幅(width)と高さ(height)の情報を基に描画される。幅と高さを指定しないと、文字や素材のサイズで描画される。

- 要素の内側の余白(padding)
  コンテンツの周囲に表示される余白のこと。上下左右の大きさを個別に指定できる。

- 要素の枠線(border)
  パディングの周囲に描画される枠線のこと。上下左右の大きさを個別に指定できる。

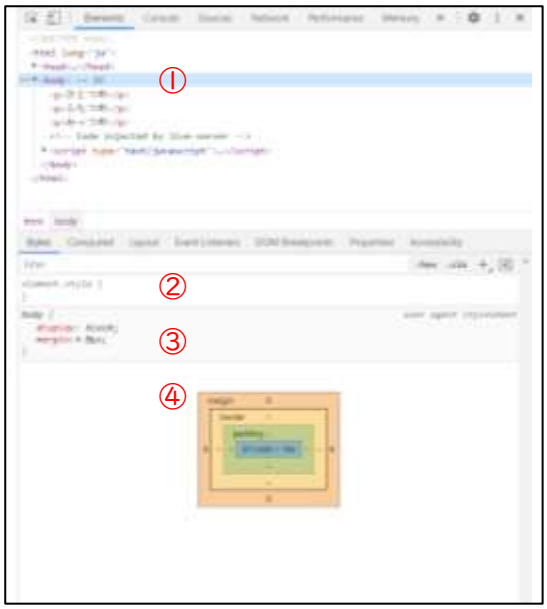
- 要素の外側の余白(margin)
  ボーダーの周囲に表示される余白のこと。上下左右の大きさを個別に指定できる。

ボックスモデルを理解するためにサンプルを表示してみよう。

ボックスモデルのサンプル（box-model1/index.html）

```
<!DOCTYPE html>
<html lang="ja">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>ボックスモデルの基本</title>
</head>
<body>
    <p>ひとつめ</p>
    <p>ふたつめ</p>
    <p>みっつめ</p>
</body>
</html>
```

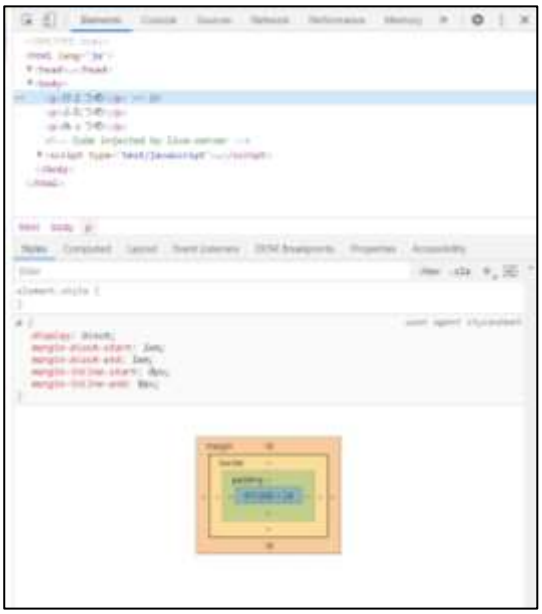
これを表示した後に F12 のキーを押してみましょう。開発者モードで表示されます。
デフォルトの設定ですとウィンドウの右側に次のように表示されます。



Elements のタブが選択されている場合
HTML の要素についての情報を閲覧できます。

① 閲覧する対象のタグを選択します
　現在は<body>が選択されていています
② 要素の CSS が表示されています
③ ユーザーエージェントの CSS が表示されます
④ ボックスモデルの表示イメージです

それでは対象のタグを変更してみましょう。マウスを使ってひとつめの<p>を選択しましょう。



<p>にもユーザーエージェントの CSS が適用されています。

margin-block は新しい記述方法で対応していないブラウザもあります。

ここでは単位について学習しましょう。

参考:CSS の単位

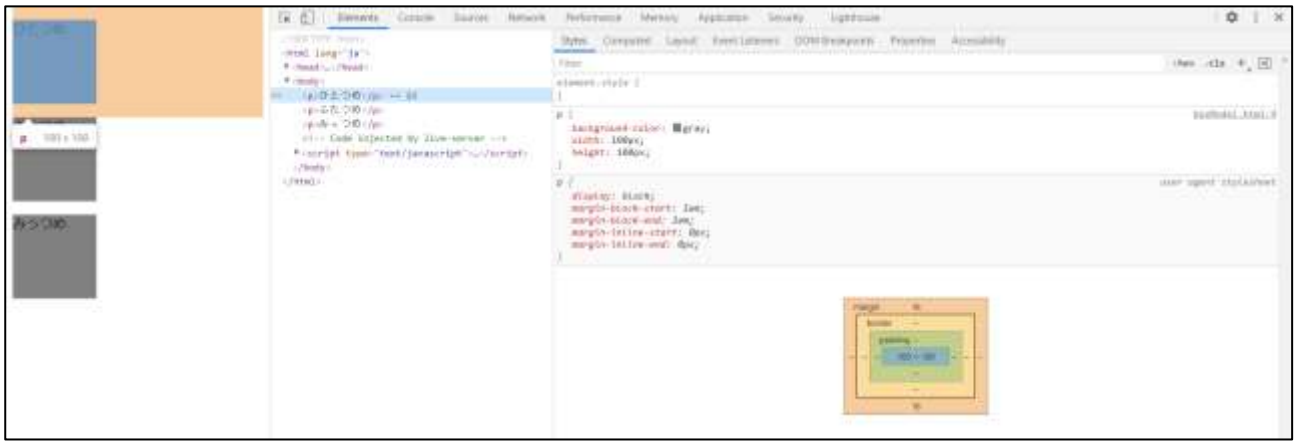
ここでは絶対長のpx、相対長の rem と%をしっかりと理解すること。

1em とは親要素のフォントサイズの 16px です。

では CSS を記述してボックスの幅と高さと背景色を設定をしてみましょう。
サンプルの追記（box-model1/index.html）

```
    <title>ボックスモデルの基本</title>
    <style>
        p {
            background-color: gray;
            width: 100px;
            height: 100px;
        }
    </style>
```

background-color
背景色に関するプロパティです。サンプルは灰色（gray）をキーワードで指定しています。
Width
コンテンツの幅に関するプロパティです。サンプルは 100px と絶対長で指定しています。
height
コンテンツの高さに関するプロパティです。サンプルは 100px と絶対長で指定しています。



次にパディング・ボーダー・マージンを設定してみましょう。

サンプルの変更（box-model1/index.html）

```
    <title>ボックスモデルの基本</title>
    <style>
        p {
            background-color: gray;
            width: 100px;
            height: 100px;
            padding: 1rem;
            border: solid 1em black;
            margin: 1rem;
        }
    </style>
```

padding

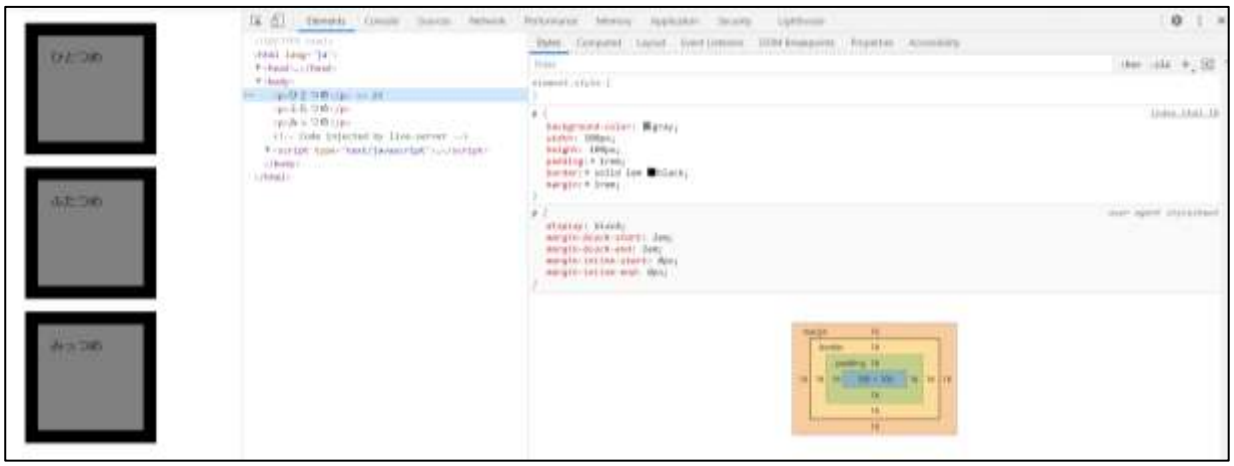
コンテンツの周囲に表示される余白に関するプロパティです。サンプルは 1rem と相対長で指定しています。
ここでは root（ルート要素）の フォントサイズである 16px（デフォルトのサイズ）が上下左右に適用されます。
この様に上下左右すべてのサイズを一括指定することをショートハンドプロパティと呼びます。

border

ボーダーに関するプロパティです。サンプルは線種を solid（実線）、線の太さを 1em、線の色を黒として設定しています。これらは border-style（線種）、border-width（線の太さ）、border-color（線の色）の一括設定を、さらに一括設定できるプロパティです。これら3つのプロパティの記述順序には指定がありません。

margin

マージンに関するプロパティです。1rem としての相対長で設定しています。
これも一括指定プロパティです。



3. ボックスのサイズ計算

ボックスのサイズ計算は構成要素をすべて足したサイズになります。

例）幅100px 高さ100px パディング すべて10px ボーダー 4px マージン 10px
幅　100 + 10 * 2 + 4 * 2 + 10 * 2 = 148px
高さ 100 + 10 * 2 + 4 * 2 + 10 * 2 = 148px
このように構成要素のサイズから計算して求める必要がありました。

box-sizing プロパティ
このプロパティの値が content-box の場合は上記の計算方法になる（従来の計算方法）。
このプロパティの値が border-box の場合は次のような計算になる。

例）幅100px 高さ100px パディング すべて10px ボーダー 4px マージン 10px
幅　100px + 10 * 2 = 120px
高さ 100px + 10 * 2 = 120px
マージン以外の構成要素を含めて幅と高さのサイズとする。

サイズ計算のサンプル（box-model/index.html に追記）

```
                    margin: 1rem;


                }
                .content-box {
                    box-sizing: content-box;
                }
                .border-box {
                    box-sizing: border-box;
                }
    </style>
```

```
    <p class="content-box">ふたつめ</p>
    <p class="border-box">みっつめ</p>
```

サイズの計算方法が変わります。

全ての適用する場合のCSSは次のように記述する。

```
*, *::before, *::after {
    box-sizing: border-box
}
```

ユニバーサルセレクタと疑似要素の組み合わせで、すべての要素に適用する形で記述します。

4. ボックスモデルの種類

これらのボックスモデルは css の display プロパティにより描画のされ方が異なる。

display		block	inline	inline-block
width（幅）の初期値		親要素の幅	コンテンツの幅	コンテンツの幅
width（幅）の設定		○	×	○
height（高さ）の設定		○	×	○
改行		○	×	○
表示位置への影響	マージン	すべて	左右	すべて
	ボーダー	すべて	左右	すべて
	パディング	すべて	左右	すべて

※ブロックは幅と高さが設定できて、要素は並ばない。

※インラインは幅と高さが設定できず、要素が横に並ぶ。

ボックスモデルの HTML サンプル（box-model2/index.html）

```
<!DOCTYPE html>
<html lang="ja">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>ボックスレイアウト</title>
    <link rel="stylesheet" href="./css/style.css">
</head>
<body>
    <p>abcdef</p>
    <p>ghi</p>
    <p>jklm</p>
</body>
</html>
```

ボックスモデルの CSS サンプル（box-model2/css/style.css）

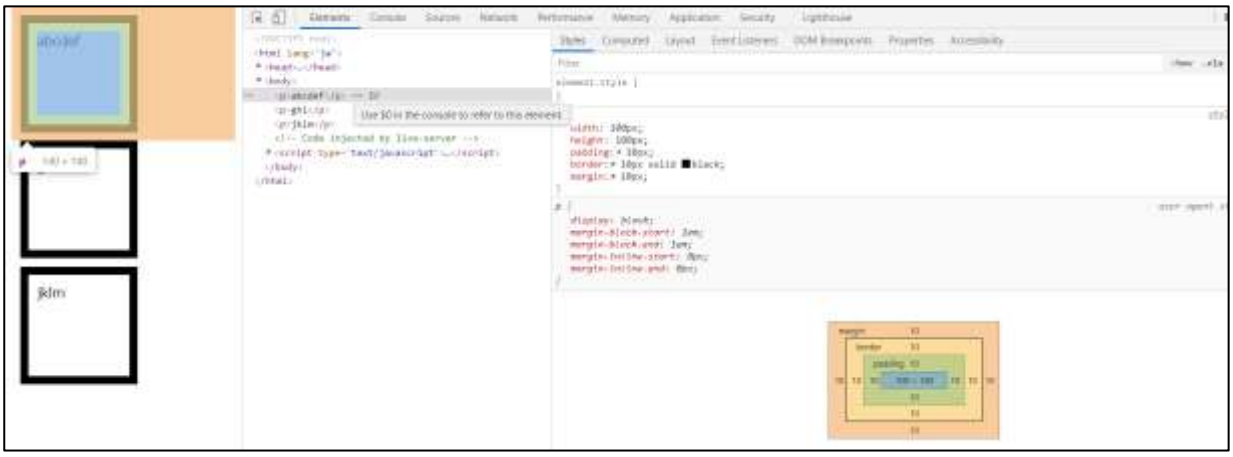
```
@charset "utf-8";

.display-inline {
    display: inline;
}

.display-inline-block {
    display: inline-block;
}

p {
    width: 100px;
    height: 100px;
    padding: 10px;
    border: 10px solid black;
    margin: 10px;
}
```

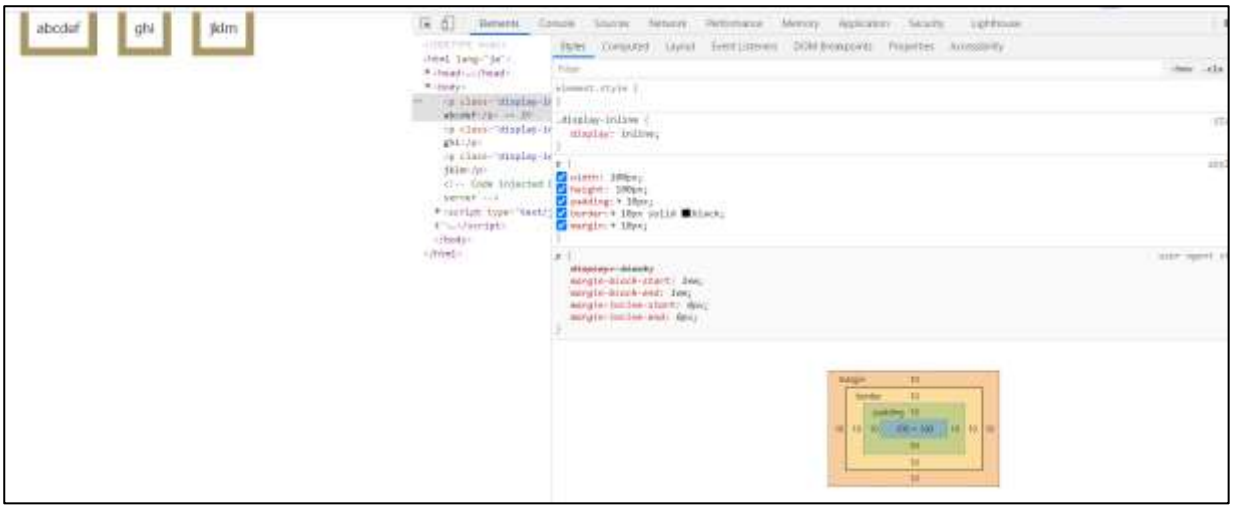
ブロックの状態



インラインの状態

```
    <p class="display-inline">abcdef</p>
    <p class="display-inline">ghi</p>
    <p class="display-inline">jklm</p>
```

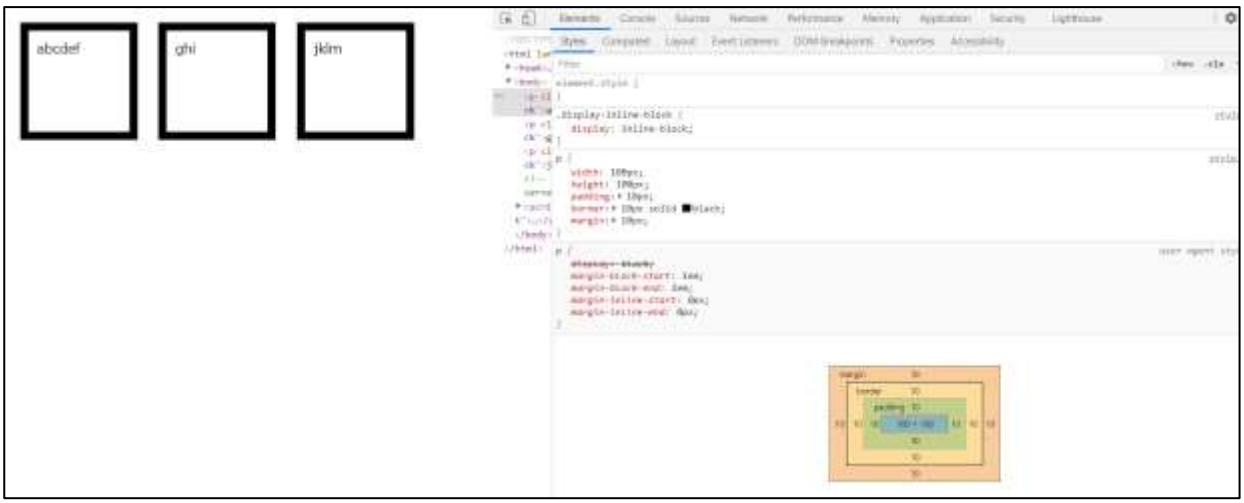
マージン・ボーダー・パディングの上下の部分は要素の表示位置に影響しない。



インラインブロックの状態

```
    <p class="display-inline-block">abcdef</p>
    <p class="display-inline-block">ghi</p>
    <p class="display-inline-block">jklm</p>
```

幅と高さが設定できて、要素は横に並ぶ。



フレックスについて

段組みのレイアウトを作成したい時にはフレックスを用いると良い。
サンプルを次に示します。

フレックスサンプル HTML(flex/inde.html)

```
<!DOCTYPE html>
<html lang="ja">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>フレックス</title>
    <link rel="stylesheet" href="./css/style.css">
</head>
<body>
    <div class="container">
        <ul class="flex">
            <li class="item"><a href="">1</a></li>
            <li class="item"><a href="">2</a></li>
            <li class="item"><a href="">3</a></li>
            <li class="item"><a href="">4</a></li>
        </ul>
    </div>
</body>
</html>
```

フレックスサンプルの CSS（flex/css/style.css）

```
@charset "utf-8";
/* flex コンテナの最小設定 */
.flex {
    display: flex;
}
/* flex アイテムの最小設定 */
.item {
    flex: auto;
}
/* アンカーの一般設定 */
a {
    color: black;
    text-decoration: none;
    display: block;
    text-align: center;
}
/* リストの一般設定 */
li {
    list-style: none;
}
/* 順序なしリストの一般設定 */
ul {
    padding: 0;
}
```

CSS の解説

フレックスコンテナを設定する。子要素を並べて表示する親要素になります。

```
display: flex;
```

フレックスアイテムを設定する。フレックスコンテナの子要素の設定をします。

```
flex: auto;
```

これは flex の一括設定で、拡大比率・縮小比率・基本の幅を設定できます。

flex:auto は flex 1 1 auto と記述したのと同じで拡大比率 1、縮小比率 1、基本幅自動と同じです。

ポジションについて

position は CSS のプロパティで、文書内で要素がどのように配置されるかを設定します。
top, right, bottom, left の各プロパティが、配置された要素の最終的な位置を決めます。

Position プロパティー覧

属性値	包含ブロック	特徴
static	直近の祖先要素のブロックコンテナ	初期値、通常のフローに従って配置されます。
relative	直近の祖先要素のブロックコンテナ	通常のフローに従って配置されます。 top, right, bottom, left の値に基づいて自分自身からの相対オフセットで配置されます。他の要素の配置には影響を与えません。
absolute	position の値が static 以外のコンテナ	通常のフローから除外されます。 要素のための空間が作成されません。 top, right, bottom, left の値に基づいて基準位置からの相対オフセットで配置されます。ボックスのマージンは、他の要素のマージンと相殺されません。
Fixed	ビューポート または ページ領域	通常のフローから除外されます。 要素のための空間が作成されません。 top, right, bottom, left によって位置が決まります・
sticky	直近の祖先要素のブロックコンテナ	通常のフローに従って配置されます。 top, right, bottom, left の値に基づいて相対配置されます。オフセットは他の要素の配置には影響を与えません。

Static 以外は重ね合わせコンテキストを作る場合があります。

関連するプロパティ

top　包含ブロックの上の基準位置からの距離
right　包含ブロックの右の基準位置からの距離
bottom　包含ブロックの下の基準位置からの距離
left　包含ブロックの左の基準位置からの距離
z-index　重ね合わせコンテキストの重なりの設定