

作って学ぶ HTML & CSS モダンコーディング

モバイルファースト&レスポンスなサイト作成を
ステップ・バイ・ステップでマスターする

エビスコム 著

特典PDF

HTML& CSS 簡易リファレンス



本 PDF は下記書籍の特典 PDF です。

PDF は GitHub (<https://github.com/ebisucom/html-css-modern-coding/>) で配布しています。



作って学ぶ

HTML & CSS モダンコーディング

<https://book.mynavi.jp/ec/products/detail/id=124054>

<https://ebisu.com/html-css-modern-coding/>

<https://amzn.to/2XsZHoU>

HTML & CSS の各種機能についてのより詳細な情報は、下記の書籍を参考にしてください。



HTML5 & CSS3 デザイン 現場の新標準ガイド

<https://book.mynavi.jp/ec/products/detail/id=117364>

<https://ebisu.com/html5-css3-practical-design-guide-2/>

<https://amzn.to/378x17B>

- ・ 本書に記載された内容は、情報の提供のみを目的としております。したがって、本書を用いての運用はすべてお客様自身の責任と判断において行ってください。
- ・ 本書の制作にあたっては正確な記述につとめました。著者や出版社のいずれも、本書の内容に関してなんらかの保証をするものではなく、内容に関するいかなる運用結果についてもいっさいの責任を負いません。あらかじめご了承ください。
- ・ 本書中に掲載している画面イメージなどは、特定の設定に基づいた環境にて再現される一例です。ハードウェアやソフトウェアの環境によっては、必ずしも本書通りの画面にならないことがあります。あらかじめご了承ください。
- ・ 本書は 2021 年 8 月段階での情報に基づいて執筆されています。本書に登場するソフトウェアのバージョン、URL、製品のスペックなどの情報は、すべてその原稿執筆時点でのものです。執筆以降に変更されている可能性がありますので、ご了承ください。
- ・ 本書中に登場する会社名および商品名は、該当する各社の商標または登録商標です。本書では®および TM マークは省略させていただきます。

CONTENTS

HTMLリファレンス

■ HTML の基本文法	5
要素	5
入れ子構造	5
属性	5

■ HTML の基本設定	6
DOCTYPE 宣言	6
<html> (ルート要素)	6
<head>	6
<body>	6

■ メタデータ	6
<title>	6
<style>	6
<link>	7
<script>	7
<meta>	7
■ エンコードの種類	7
■ ビューポートの設定	7

■ セクション	8
<section>	8

<nav>	8
<aside>	8
<article>	8
<body>	8

■ セクションに関する情報	9
<h1> / <h2> / <h3> / <h4> / <h5> / <h6>	9
<header> / <footer>	9

■ コンテンツ	10
<p>	10
<div> / 	10
<i>	10
 / 	10
<button>	10
<a>	10
<figure>	11
	11
■ レスポンシブイメージ	11
<picture>	11

CSSリファレンス

■ CSS の基本文法	12
ルール	12
■ セレクタ	13
セレクタの記述形式	13
シンプルセレクタ	13
疑似要素	14
詳細度	14

■ @ 規則 (@ ルール)	15
エンコードの指定 @charset	15
メディアクエリ @media	15
機能クエリ @supports	16
コンテナクエリ @container	16
■ ボックス	18
ボックスモデル	18

CONTENTS

■ margin / padding	18
■ border	18
■ width / height	18
■ aspect-ratio	18
画像	19
■ object-fit	19
背景	19
■ background-color	19
■ background-image	19
■ background-size / background-position	19
ポジション	20
ボックスの種類	20
ボックス内のコンテンツのデザイン	21
■ color	21
■ font-family	21
■ font-size	21
■ font-weight	21
■ line-height	21
■ text-align	21
■ text-decoration	21
■ text-shadow	21
■ list-style	21
■ cursor	21
■ Flexbox	22
Flexbox の基本	22
flex-direction	22
flex-wrap	22
order	22
justify-content	23
align-content	23
flex-basis	23
align-items	23
flex-shrink	24
flex-grow	24
■ CSS Grid	25
CSS Grid の基本	25
grid-column / grid-row	26
grid-auto-flow	26
order	26
justify-content	27
align-content	27
justify-items	28
align-items	28
■ CSS 関数	29
比較関数	29
■ min()	29
■ max()	29
■ clamp()	29
変数 (カスタムプロパティ)	30
■ var()	30
数式	30
■ calc()	30
フィルタ	30
トランスフォーム	30
■ 単位 / 値	31
単位	31
色の値	31
デフォルトの設定にする値	31

HTML

HTMLリファレンス

HTML Basic Syntax

HTMLの基本文法

HTML の基本的な文法（構文規則）です。

要素

HTMLでは開始タグと終了タグでコンテンツを囲み、コンテンツの中身が何であるかを示します。囲んだ部分全体を「要素」と呼びます。

単独で機能するものは「void 要素」と呼ばれます。コンテンツのマークアップは行わず、右のように開始タグのみを記述します。末尾に「/」をつけることもできます。

要素名。

`<h1> 記事のタイトル </h1>`

開始タグ。

コンテンツ。

終了タグ。

要素。

``

``

入れ子構造

複数のタグでマークアップする場合、親要素内に子要素を収め、「入れ子構造」で記述することができます。

`<div><h1> 記事のタイトル </h1></div>`

子要素
親要素

属性

属性を使うと各種情報を付加することができます。値はクォーテーション（" または '）で囲むか、クォーテーションなしで記述します。

`<html lang="ja"> ... </html>`

属性名。 値。

HTMLの基本設定

Web ページを構成する HTML の基本的な設定です。

DOCTYPE宣言

HTML で記述したページであることを明示しています。

<html>（ルート要素）

コード全体を囲み、最上位階層（ルート）を構成します。
lang 属性で日本語のページであることを明示しています。

<head>

<head> 内にはメタデータ（ページに関する情報）を記述します。

<body>

<body> 内にはブラウザ画面に表示するコンテンツを記述します。

```
<!DOCTYPE html>
<html lang="ja">
<head>

</head>
<body>

</body>
</html>
```

Metadata

メタデータ

ページに関する各種情報や設定です。<head> 内に記述します。

<title>

ページのタイトルを指定します。記述は必須となっています。

```
<title> ... </title>
```

<style>

HTML 内に CSS の設定を記述するために使用します。

```
<style> ... </style>
```

<link>

外部リソースの明示に使用します。

外部 CSS ファイルを読み込む場合は `rel` 属性を「stylesheet」と指定し、`href` 属性でファイルの URL を指定します。

```
<link href="～.css" rel="stylesheet">
```

Google Fonts の設定（書籍 P.28）に含まれている「preconnect」は、`href` 属性で指定した外部サイトへの接続処理をできるだけ早く開始するようにブラウザに提案するものです。

```
<link rel="preconnect"
href="https://fonts.gstatic.com">
```

<script>

Web ページ内にスクリプトの設定を記述するために使用します。`src` 属性で外部スクリプトファイルを読み込むことも可能です。

```
<script> … </script>
```

```
<script src="～.js"></script>
```

<meta>

さまざまな情報や設定の明示に使用します。

■ エンコードの種類

HTML ファイルのエンコードの種類を明示します。

```
<meta charset="UTF-8">
```

■ ビューポートの設定

デバイスに合わせた画面サイズ（ビューポートサイズ）でページを表示するために必要な設定です。

```
<meta name="viewport"
content="width=device-width">
```



ビューポートの設定がない場合、PC版が縮小表示されます。



ビューポートの設定がある場合、モバイル版での表示になります。

セクション

コンテンツのまとまり(セクション)を明示するタグです。「セクショニング・コンテンツ」とも呼ばれます。

<section>

コンテンツのまとまりであることを明示します。視覚的なデザインを適用する目的でコンテンツをグループ化する場合は <div> を使用します。

```
<section> ... </section>
```

<nav>

ナビゲーションであることを明示します。

```
<section> ... </section>
```

<aside>

補足・関連情報であることを明示します。

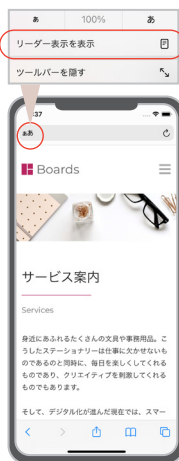
```
<aside> ... </aside>
```

<article>

1つの完結したコンテンツであることを明示します。ブログやニュースの記事、SNSの投稿などの明示に使用することができます。

```
<article> ... </article>
```

Safariのリーダー機能では <article> で明示された記事が抽出して表示されます。



抽出された記事。記事が短い場合などには抽出されません。



<body>

「セクショニング・ルート」とも呼ばれ、ページ全体のコンテンツを明示します。

```
<body> ... </body>
```


セクションに関する情報

セクションに関する情報を明示するタグです。

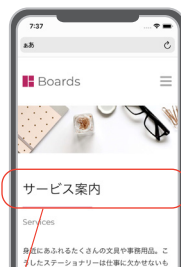
`<h1>` / `<h2>` / `<h3>` / `<h4>` / `<h5>` / `<h6>`

セクションの見出しであることを明示します。見出しのレベルに応じて `<h1>` ～ `<h6>` を使い分けます。

たとえば、`<article>` 直下に `<h1>` を記述すると、「記事のタイトル」であることを明示できます。

Safari のリーダー機能でも `<article>` 内の `<h1>` がタイトルとして抽出され、リーダーの一番最初に表示されることがわかります。

```
<article>
  <h1> ... </h1>
</article>
```



`<article>` 直下の `<h1>` でマークアップした箇所。



記事のタイトルとして抽出されます。

`<header>` / `<footer>`

セクションのヘッダーおよびフッターであることを明示します。たとえば、`<body>` 直下に記述すると、ページ全体のヘッダー、フッターであることを明示できます。

※ヘッダーは上部に付加するもので、サイト名やタイトルなどの情報が含まれます。

※フッターは下部に付加するもので、コピーライトなどの情報が含まれます。

```
<body>
  <header> ... </header>
  ...
  <footer> ... </footer>
</body>
```



コンテンツ

さまざまなコンテンツを明示するタグです。

<p>

段落をはじめとしたテキストのまとまりであることを明示します。

```
<p> ... </p>
```

<div> /

特別な意味を持たない汎用タグです。セマンティクスを明示したくない場合や、他に適切なタグがない場合に使用します。

```
<div> ... </div>
```

```
<span> ... </span>
```

<i>

雰囲気の異なる語句や慣用句など、他と区別したい語句を示します。Font Awesome(書籍 P.43)ではアイコンの記述に使用されています。

```
<i> ... </i>
```

 /

リスト(箇条書き)形式の情報であることを明示します。リストの各項目をで、全体をでマークアップします。

```
<ul>  
  <li> ... </li>  
  <li> ... </li>  
  <li> ... </li>  
</ul>
```

<button>

マークアップしたコンテンツをボタンとして機能させます。書籍(P.296)ではonClick属性を使用し、クリックしたときに実行する処理をJavaScriptで指定しています。

```
<button onClick=" ~ "> ... </button>
```

<a>

リンクを設定します。リンク先のURLはhref属性で指定します。

```
<a href=" ~ "> ... </a>
```

<figure>

図版や表などの完結したコンテンツであることを明示します。

```
<figure> ... </figure>
```


画像を表示します。画像の URL は src 属性で、代替テキストは alt 属性で指定します。width と height 属性では画像のオリジナルサイズを指定し、レイアウトシフト（書籍 P.168）が発生するのを防ぎます。

```

```

■ レスポンシブイメージ

デバイスに応じて最適なサイズの画像で表示するためには、書籍 P.247 のように srcset 属性でサイズの異なる画像セットを指定します。sizes 属性では画像セットの中から画像を選択する条件を指定します。

```

```

<picture>

レスポンシブイメージでさらに細かく画像の選択肢を用意するためには、書籍 P.249 のように <picture> を使用します。

<picture> では <source> で画像の選択肢を用意します。右の設定では WebP フォーマットの画像を選択肢にしています。

<source> で指定された画像が使用できなかった場合には、 で指定した画像が使用されます。

```
<picture>
  <source
    type="image/webp"
    srcset="service400.webp 400w,
            service800.webp 800w,
            service.webp 1600w"
    sizes="(max-width: 1600px) 100vw,
            1600px">

  
</picture>
```

CSS

CSSリファレンス

CSS Basic Syntax

CSSの基本文法

CSS の基本的な文法（構文規則）です。

ルール

CSS ではセレクトアで指定した箇所の視覚的なデザインを設定します。デザインはプロパティと値の組み合わせ（宣言）で指定していきます。

たとえば、右の設定では `<h1>` でマークアップした箇所のフォントサイズ（`font-size`）を 20 ピクセルに指定しています。

`{ ~ }` は「宣言ブロック」と呼ばれ、ブロック内には複数の宣言を記述することができます。たとえば、右の設定では `color`、`font-size`、`text-align` プロパティの宣言を記述しています。

なお、セレクトアと宣言ブロックを含めた設定全体は「ルール」と呼ばれます。

セレクトア。

`h1 { font-size: 20px; }`

プロパティ。

値。

宣言。

宣言ブロック。

ルール。

```
h1 {  
  color: #ff0000;  
  font-size: 20px;  
  text-align: center;  
}
```

宣言ブロック

セクタ

セクタでは CSS の適用先をさまざまな形で指定することができます。

セクタの記述形式

シンプルセクタを組み合わせで記述します。

記述形式	記述例	適用先
基本形	<code>section {…}</code>	<code><section></code> に適用
混合セクタ	<code>section.hero {…}</code>	クラス名が「hero」と指定された <code><section></code> に適用
子孫セクタ	<code>article h1 {…}</code>	<code><article></code> 内の <code><h1></code> に適用
子セクタ	<code>article > h1 {…}</code>	<code><article></code> 直下の <code><h1></code> に適用
兄弟セクタ	<code>h1 + p {…}</code>	<code><h1></code> に続けて記述された <code><p></code> に適用
セクタリスト	<code>h1, p, figure {…}</code>	<code><h1></code> 、 <code><p></code> 、 <code><figure></code> のそれぞれに適用

シンプルセクタ

シンプルセクタ	記述例	適用先
タイプセクタ	<code>section {…}</code>	指定した名前の要素（ここでは <code><section></code> ）に適用
ユニバーサルセクタ	<code>* {…}</code>	すべての要素に適用
IDセクタ	<code>#id {…}</code>	IDが「id」と指定された要素に適用
クラスセクタ	<code>.hero {…}</code>	クラス名が「hero」と指定された要素に適用
構造疑似クラス	<code>:first-child {…}</code>	同一階層の最初の要素に適用
	<code>:last-child {…}</code>	同一階層の最後の要素に適用
ダイナミック疑似クラス	<code>:hover {…}</code>	カーソルを重ねた要素に適用
否定疑似クラス	<code>:not(h1) {…}</code>	指定した条件と一致しない要素（ここでは <code><h1></code> 以外の要素）に適用。詳細度は <code><h1></code> の詳細度で処理されます。
Matches-Any 疑似クラス	<code>:is(h1, p, div.hero) {…}</code>	<code><h1></code> 、 <code><p></code> 、 <code><div class="hero"></code> のそれぞれに適用。詳細度は最も高くなる「div.hero」の詳細度で処理されます。
Specificity-adjustment 疑似クラス	<code>:where(h1, p, div.hero) {…}</code>	<code><h1></code> 、 <code><p></code> 、 <code><div class="hero"></code> のそれぞれに適用。詳細度は「0」で処理されます。

疑似要素

シンプルセクタの末尾に付加して使用します。
挿入する要素のコンテンツは content で指定します。

```
h1::before {  
  content: '新着情報';  
}
```

疑似要素	記述例	適用先
::before/::after疑似要素	h2::before {…}	<h2>内のコンテンツの前に要素を挿入して適用
	h2::after {…}	<h2>内のコンテンツの後に要素を挿入して適用

詳細度

同じ要素に適用した CSS が重複しているケースでは、セクタの詳細度が同じ場合は後から記述した設定が使用されます。異なる場合は詳細度の高い設定が使用されます。

```
section { color: red; } /* 詳細度 1 */  
section { color: blue; } /* 詳細度 1 */
```

詳細度が同じ場合

使用される設定。

```
section.hero { color: red; } /* 詳細度 11 */  
section { color: blue; } /* 詳細度 1 */
```

詳細度が異なる場合

使用される設定。

詳細度はセクタの構成に応じて a、b、c をカウントし、3 つの数字 a-b-c を連結して求めます。

a = ID セクタの数。

b = クラスセクタ / 疑似クラスの数。

c = タイプセクタ / 疑似要素の数。

※ユニバーサルセクタ(*)はカウントされません。

例	a	b	c	詳細度
section {…}	0	0	1	1
section.hero {…}	0	1	1	11
section.hero h1 {…}	0	1	2	12
#id .hero {…}	1	1	0	110

@規則 (@ルール)

CSS に関する各種設定や、適用条件の指定などを行います。

エンコードの指定 @charset

CSS ファイルの 1 行目には @charset でエンコードの種類を記述します。

```
@charset "UTF-8";  
...
```

メディアクエリ @media

メディアクエリ @media は次のような形式で記述し、デバイスの特性に応じて CSS を適用します。ここではビューポート (画面) の横幅が 600 ピクセル以上かつ 768 ピクセル以下の場合に CSS の設定を適用するように指定しています。

```
@media (min-width: 600px) and (max-width: 768px) { ... }
```

特性名。 値。 特性名。 値。 CSSの設定。

特性。 特性。

条件(メディアコンディション)。

特性名	指定できる値	適用条件
min-width	数値	ビューポート (画面) の横幅が指定したサイズ以上なら適用
max-width	数値	ビューポート (画面) の横幅が指定したサイズ以下なら適用
pointer	coarse	デバイスの標準の入力方法がタッチ操作なら適用
	fine	デバイスの標準の入力方法がキーボードやマウスなら適用
hover	none	マウスを重ねるホバーができないなら適用
	hover (省略可)	マウスを重ねるホバーができるなら適用

機能クエリ @supports

@supports を利用すると、特定の CSS の機能に対応しているかどうかに応じて CSS を適用できます。条件は「プロパティ: 値」の形で指定します。

```
@supports (aspect-ratio: 3 / 2) {…}
```

aspect-ratioの指定に対応しているなら適用

```
@supports not (aspect-ratio: 3 / 2) {…}
```

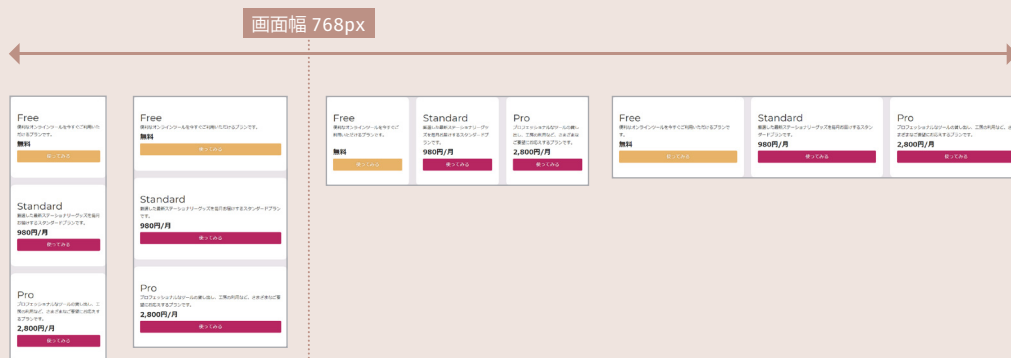
aspect-ratioの指定に未対応なら適用

※値の指定が必要なので、ここでは「3 / 2」と指定しています。

コンテナクエリ @container

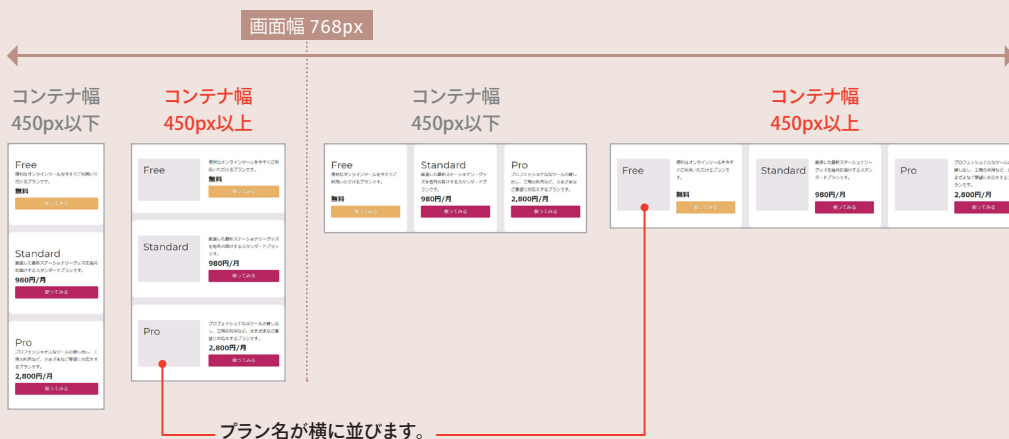
コンテナクエリ @container を利用すると、コンテナ要素の特性に応じて CSS を適用できるようになります。

たとえば、「プラン」パーツではメディアクエリ @media を利用して、画面幅に応じて3つのプランの縦並びと横並びを切り替えています。こうしたパーツでは、画面幅によって横幅が大きくなる箇所が出てきます。



書籍ではパーツ全体の最大幅を指定し、横幅が大きくなりすぎるのを防いでいますが、コンテナクエリを利用して対処することも考えられます。

各プランをマークアップした `<div class="plan">` をコンテナクエリのコンテナにして、コンテナ幅が 450px 以上になった場合はプラン名を横に並べるレイアウトに切り替えると次のようになります。



```
.plan {
  container-name: plan;
  container-type: inline-size;
}

@container plan (min-width: 450px) {
  /* プラン名を横に並べる設定 */
  .plan > * { ... }
  .plan h3 { ... }
}
```

```
<section class="plans">
  <div class="plans-container">
    <div class="plan">
      <h3>Free</h3> ...
    </div>
    <div class="plan">
      <h3>Standard</h3> ...
    </div>
    <div class="plan">
      <h3>Pro</h3> ...
    </div>
  </div>
</section>
```

`<div class="plan">` をコンテナにするため、`container-name` でコンテナ名を「plan」、`container-type` でコンテナの種類を「inline-size (横幅を基準に処理)」に指定しています。これで「`@container plan`」を使用すると、plan コンテナのコンテンツ幅 (パディングなどを含まない横幅) に応じて CSS を適用できるようになります。ただし、`<div class="plan">` 内の要素にしか適用できません。

※コンテナクエリにはChrome 92以降が実験的な機能として対応しています。
利用するためには「chrome://flags/」を開き、「CSS Container Queries」を有効化します。

※ここでは2021年7月に改定されたコンテナクエリの仕様 (<https://drafts.csswg.org/css-contain-3/>) に従って設定を記述しており、動作確認にはChrome 93以降が必要です。

ボックス

要素が構成するボックス関連のプロパティです。レイアウトや装飾の基本となります。

ボックスモデル

■ margin / padding

margin ではボーダーの外側、padding では内側の余白サイズを指定します。上下左右の値は次の形式で指定できます。

margin: 上下左右 ;

margin: 上下 左右 ;

margin: 上 右 下 左 ;

margin: 上 左右 下 ;

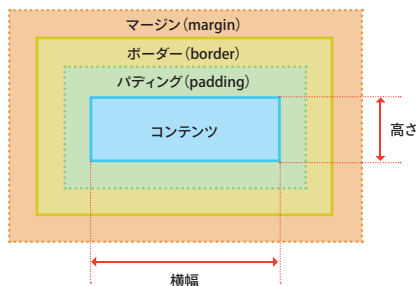
※個別のプロパティ (margin-top / margin-right / margin-bottom / margin-left など) でも指定できます。

■ border

ボーダー (罫線) のデザインを指定します。たとえば、「solid 1px #ffaa00」と指定すると、太さ 1 ピクセルのオレンジ色 (#ffaa00) の実線 (solid) になります。指定できるボーダーの種類は以下のようになっています。

border: 種類 太さ 色 ;

値	罫線の種類	表示
dotted	点線	
dashed	破線	
solid	実線	
double	二重線	



■ width / height

コンテンツの横幅と高さを指定します。





width: 横幅 ; height: 高さ ;

※パディングとボーダーを含めたサイズにする場合は box-sizing を「border-box」と指定します。

■ aspect-ratio

ボックスの縦横比を指定します。16:9 の縦横比にする場合は次のように指定します。

aspect-ratio: 16 / 9 ;

値	罫線の種類	表示
groove	立体枠	
ridge	立体枠	
inset	立体枠	
outset	立体枠	

画像

■ object-fit

`` が構成するボックスのサイズも `width`、`height`、`aspect-ratio` で指定できます。このとき、`object-fit` を使用するとボックスのサイズに合わせて画像を切り出すことが可能です。



`` が構成する
ボックス

```
img {  
  object-fit: cover;  
  width: 100%;  
  aspect-ratio: 1 / 1;  
}
```

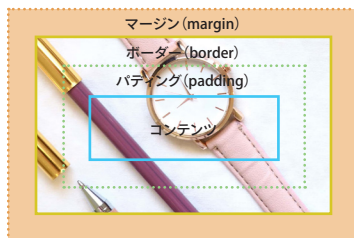
背景

背景色や背景画像はボックスのコンテンツ、パディング、ボーダーの範囲に表示されます。

■ background-color

背景色を指定します。

```
div {background-color: #ffffff;}
```



■ background-image

背景画像を指定します。画像はオリジナルサイズで表示され、左上がボックスで切り出された形になります。



`<div>` が構成する
ボックス

```
div {  
  background-image: url(helpful.jpg);  
  aspect-ratio: 1 / 1;  
}
```

■ background-size / background-position

`background-size` と `background-position` を利用すると、画像をボックスのサイズに合わせ、中央を切り出すことができます。



`<div>` が構成する
ボックス

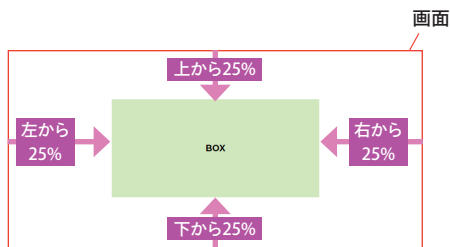
```
div {  
  background-image: url(helpful.jpg);  
  background-position: center;  
  background-size: cover;  
  aspect-ratio: 1 / 1;  
}
```

ポジション

`position` を利用すると、親要素や画面の上下左右からの表示位置を指定できます。他の要素とは別扱いになり、重ねて表示されます。重なり順は `z-index` で調整します。

```
.box {  
  position: fixed;  
  inset: 25%;  
}
```

※上下左右からの距離は `inset` や `top` / `right` / `bottom` / `left` で指定します。



positionの値	inset (top / right / bottom / left) で指定される距離
absolute	position (static 以外) が適用された直近の親要素、または画面の上下左右からの距離
fixed	画面の上下左右からの距離
sticky	overflow が適用されたスクロール可能な直近の親要素、または画面の上下左右からの距離 ※ロード時は position 未指定時の位置に表示され、スクロールに応じて指定位置に配置されます
relative	position 未指定時の表示位置からの距離
static	位置調整なし (position 未指定時と同じ表示)

ボックスの種類

`display` を利用すると、ボックスの種類を指定できます。

```
.box {display: block;}
```

displayの値	ボックスの種類	要素
block	ブロック	ブロックとして扱われるボックス。親要素に合わせた横幅になります。 <div>、<h1>、<p> などが標準でブロックボックスを構成します。
inline	インライン	テキストと同じように扱われるボックス。中身に合わせた横幅になります。 width、height、上下 margin は表示に反映されません。上下 padding は反映されますが、他の要素との位置関係に影響を与えません。 、<a>、<button> などが標準でインラインボックスを構成します。
inline-block	インラインブロック	基本的にインラインと同じですが、width、height、上下 margin が表示に反映され、上下 padding も他の要素との位置関係に影響を与えるようになります。 などがインラインブロックボックスに相当するものとして処理されます。
none	なし	ボックスを構成しません。

ボックス内のコンテンツのデザイン

■ color

テキストの色を指定します。

```
color: #ffffff;
```

■ font-family

フォントファミリーやフォントの系統（ゴシック系など）を指定します。

```
font-family: "Montserrat", sans-serif;
```

■ font-size

フォントサイズを指定します。

```
font-size: 16px;
```

■ font-weight

フォントの太さ（100～900）を指定します。「normal」は400、「bold」は700の太さになります。

```
font-weight: bold;
```

■ line-height

行の高さを指定します。フォントサイズの1.5倍の高さにする場合は「1.5」と指定します。

```
line-height: 1.5;
```

■ text-align

テキストの行揃え（left、center、right）を指定します。

```
text-align: center;
```

■ text-decoration

テキストの装飾（underline など）を指定します。下線を消す場合は「none」と指定します。

```
text-decoration: none;
```

■ text-shadow

テキストに影をつけます。

```
text-shadow: 0 0 6px #00000052;
```

横オフセット 縦オフセット ブラー 影の色

■ list-style

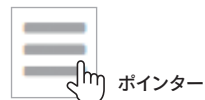
リストマークの種類（disc など）を指定します。削除する場合は「none」と指定します。

```
list-style: none;
```

■ cursor

カーソルの種類（pointer など）を指定します。

```
cursor: pointer;
```



Flexbox

Flexbox 関連のプロパティです。

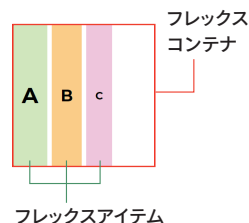
Flexboxの基本

display を「flex」と指定するとフレックスコンテナが構成され、直近の子要素（フレックスアイテム）のレイアウトを Flexbox でコントロールできるようになります。

ここではフォントサイズと色が異なる3つのアイテム（A～C）を用意しています。標準では横並びになり、各アイテムの横幅は中身に合わせたサイズに、高さはコンテナに合わせたサイズになります。gap ではアイテムの間隔を調整しています。

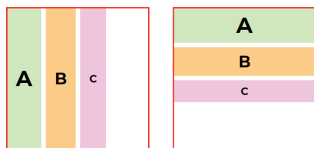
```
.container {
  display: flex;
  gap: 10px;
  aspect-ratio: 1 / 1;
  border: solid 2px red;
}
```

```
<div class="container">
  <div class="boxA">A</div>
  <div class="boxB">B</div>
  <div class="boxC">C</div>
</div>
```



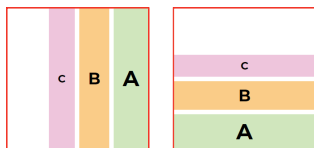
flex-direction

アイテムを並べる方向を指定します。



row

column

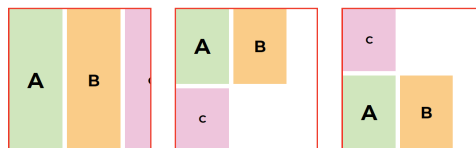


row-reverse

column-reverse

flex-wrap

コンテナに収まらないアイテムを折り返します。



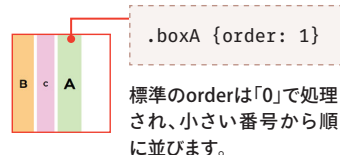
nowrap

wrap

wrap-reverse

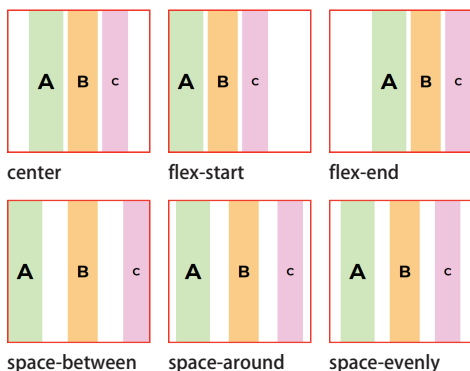
order

アイテムの並び順を指定します。



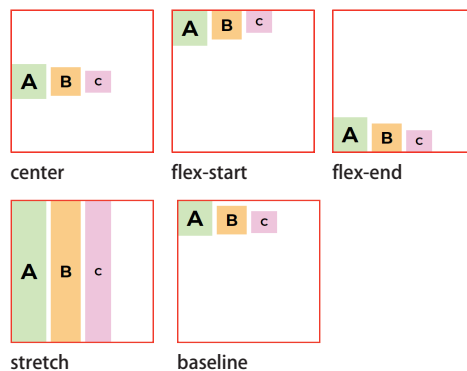
justify-content

アイテムの横方向の配置を指定します。



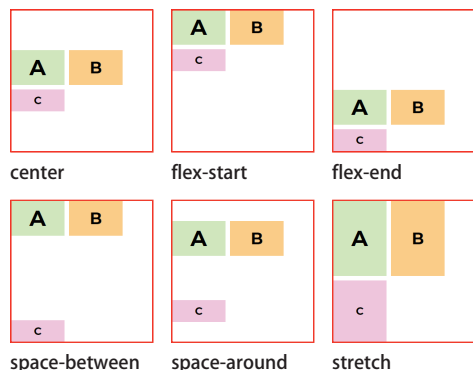
align-items

アイテムの縦方向の配置を指定します。

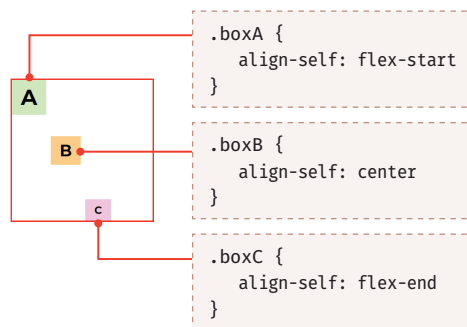


align-content

「flex-wrap: wrap」を適用したときの縦方向の配置を指定します。

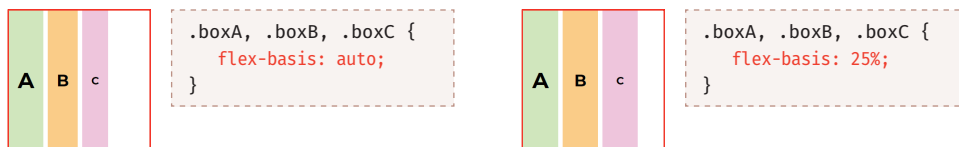


※アイテムごとに縦方向の配置を変えたい場合、
align-selfで個別に指定します。
指定できる値は **align-items** と同じです。



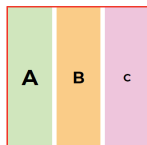
flex-basis

アイテムの横幅を指定します。標準では「auto」で処理され、中身に合わせた横幅になります。
「auto」で処理されている場合、widthで横幅を調整することも可能です。



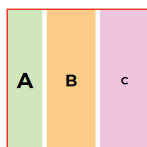
flex-shrink

アイテムを縮小し、コンテナに収まるようにする機能です。標準では「1」で処理され、必要に応じて縮小処理が行われます。



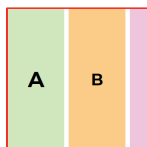
```
.boxA, .boxB, .boxC {  
  flex-basis: 40%;  
  flex-shrink: 1;  
}
```

3つのアイテムの横幅を40%に指定していますが、コンテナに収まるように縮小処理が行われます。



```
.boxA, .boxB, .boxC {  
  flex-basis: 40%;  
  flex-shrink: 1;  
}  
.boxA {  
  flex-shrink: 3;  
}
```

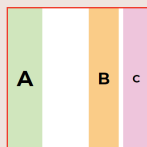
縮小処理の比重を変えることもできます。ここではflex-shrinkを「3」にしたAの横幅が他より短くなっています。



```
.boxA, .boxB, .boxC {  
  flex-basis: 40%;  
  flex-shrink: 0;  
}
```

flex-shrinkを「0」にすると縮小されなくなり、コンテナからオーバーフローします。

flex-grow を「0」以外に指定したアイテムがなく、margin を「auto」と指定した箇所がある場合、そこに余剰スペースが分配されます。

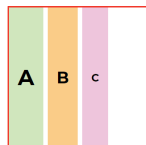


```
.boxA, .boxB, .boxC {  
  flex-basis: auto;  
}  
.boxA {  
  margin-right: auto;  
}
```

Aの右マージンを「auto」に指定。

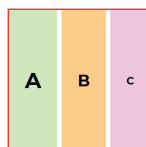
flex-grow

コンテナ内の余剰スペースをアイテムの横幅に分配する機能です。標準では「0」で処理され、余剰スペースは分配されません。



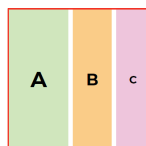
```
.boxA, .boxB, .boxC {  
  flex-basis: auto;  
  flex-grow: 0;  
}
```

余剰スペース。



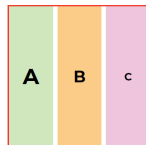
```
.boxA, .boxB, .boxC {  
  flex-basis: auto;  
  flex-grow: 1;  
}
```

余剰スペースを1:1:1の比率で分配するように指定。各アイテムの元の横幅に追加されます。



```
.boxA, .boxB, .boxC {  
  flex-basis: auto;  
  flex-grow: 1;  
}  
.boxA {  
  flex-grow: 3;  
}
```

余剰スペースを3:1:1の比率で分配するように指定。Aに分配されるサイズが大きくなります。



```
.boxA, .boxB, .boxC {  
  flex-basis: 0;  
  flex-grow: 1;  
}
```

||

```
.boxA, .boxB, .boxC {  
  flex: 1;  
}
```

flex-basisでアイテムの横幅を「0」と指定すると、コンテナの横幅からgap、margin、paddingを除いたサイズが余剰スペースとなります。その上で、flex-growを「1」と指定すると、アイテムが1:1:1で等分割した横幅になります。この指定は「flex: 1」と記述することもできます。

CSS Grid

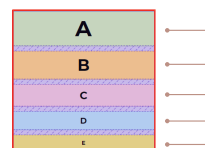
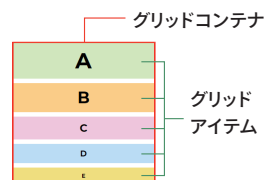
CSS Grid 関連のプロパティです。

CSS Gridの基本

`display` を「grid」と指定するとグリッドコンテナが構成され、子要素（グリッドアイテム）のレイアウトを CSS Grid でコントロールできるようになります。ここではフォントサイズと色が異なる5つのアイテム（A～E）を用意しています。そのため、1列×5行のグリッドが生成され、縦並びで配置されます。行列のサイズは標準では「auto」で処理され、配置したアイテムに合わせたサイズになります。gap では行列の間隔を調整しています。

```
.container {
  display: grid;
  gap: 10px;
  aspect-ratio: 1 / 1;
  border: solid 2px red;
}
```

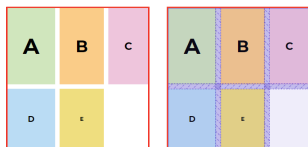
```
<div class="container">
  <div class="boxA">A</div>
  <div class="boxB">B</div>
  <div class="boxC">C</div>
  <div class="boxD">D</div>
  <div class="boxE">E</div>
</div>
```



グリッドの構造
(1列×5行)

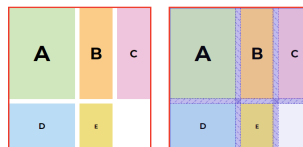
アイテムの中身に合わせた
行の高さになっています。

グリッドの構造は行列のサイズを `grid-template-columns` (列) と `grid-template-rows` (行) で指定して調整します。たとえば、次のように指定すると3列×2行の構造にできます。



```
.container {
  display: grid;
  gap: 10px;
  grid-template-columns: auto auto auto;
  grid-template-rows: auto auto;
  ...
}
```

列と行のサイズを「auto (中身に合わせたサイズ)」に指定。



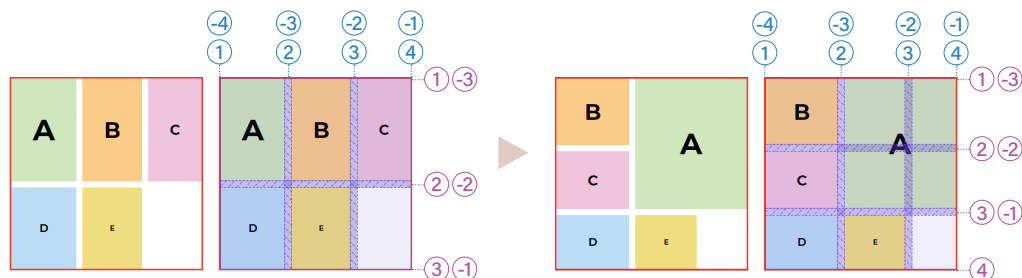
```
.container {
  display: grid;
  gap: 10px;
  grid-template-columns: 2fr 1fr 1fr;
  grid-template-rows: 2fr 1fr;
  ...
}
```

列を2:1:1、行を2:1のサイズに指定。

※同じ値の繰り返しはrepeat()でまとめて記述することもできます。たとえば、「auto auto」は「repeat(2, auto)」と記述します。

grid-column / grid-row

アイテムの配置先は `grid-column` と `grid-row` で指定できます。グリッドの行列を区切るラインに割り振られたライン番号を使用し、A の配置先を列 ②～④、行 ①～③ に指定すると次のようになります。B～C は自動配置されますが、3 列×2 行では配置先が足りないため、3 行目が自動生成されます。



```
.container {  
  grid-template-columns: auto auto auto;  
  grid-template-rows: auto auto;  
}
```

```
.container {  
  grid-template-columns: auto auto auto;  
  grid-template-rows: auto auto;  
}
```

```
.boxA {  
  grid-column: 2 / 4;  
  grid-row: 1 / 3;  
}
```

```
.boxA {  
  grid-column: 2 / span 2;  
  grid-row: 1 / span 2;  
}
```

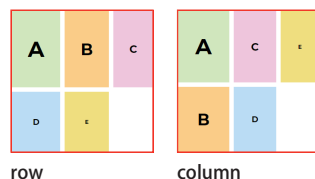
A の配置先は「列 2 から 2 行分」、「行 1 から 2 行分」という形で指定することもできます。

```
.boxA {  
  grid-column: 2 / -1;  
  grid-row: 1 / -1;  
}
```

マイナスのライン番号を使って指定することもできます。ただし、マイナスのライン番号は `grid-template-rows` で指定していない、自動生成された行には割り振られません。

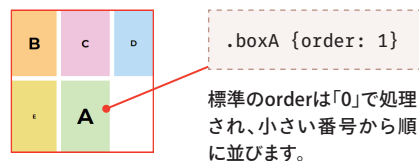
grid-auto-flow

アイテムを自動配置する方向を指定します。



order

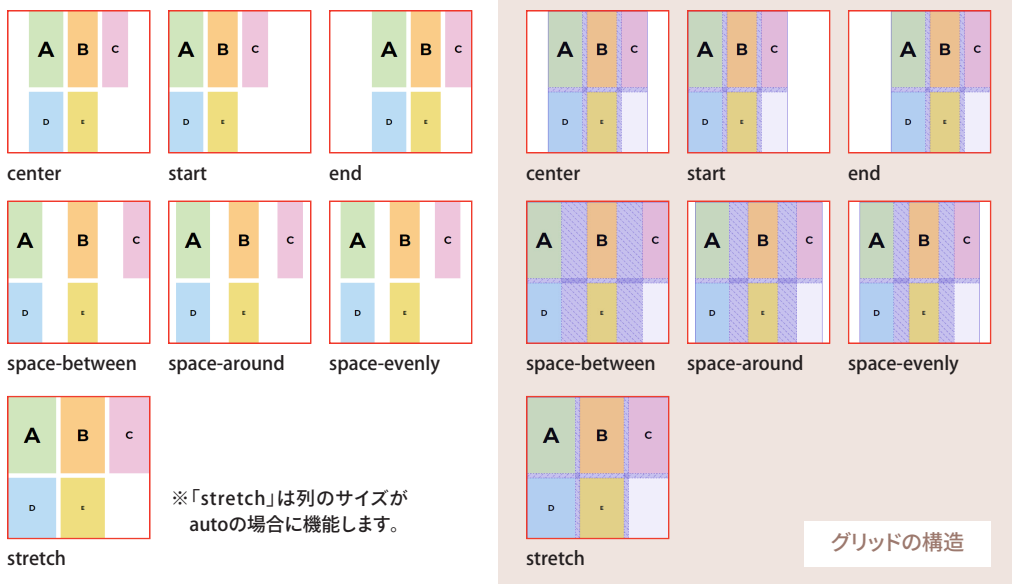
アイテムの並び順を指定します。



標準の order は「0」で処理され、小さい番号から順に並びます。

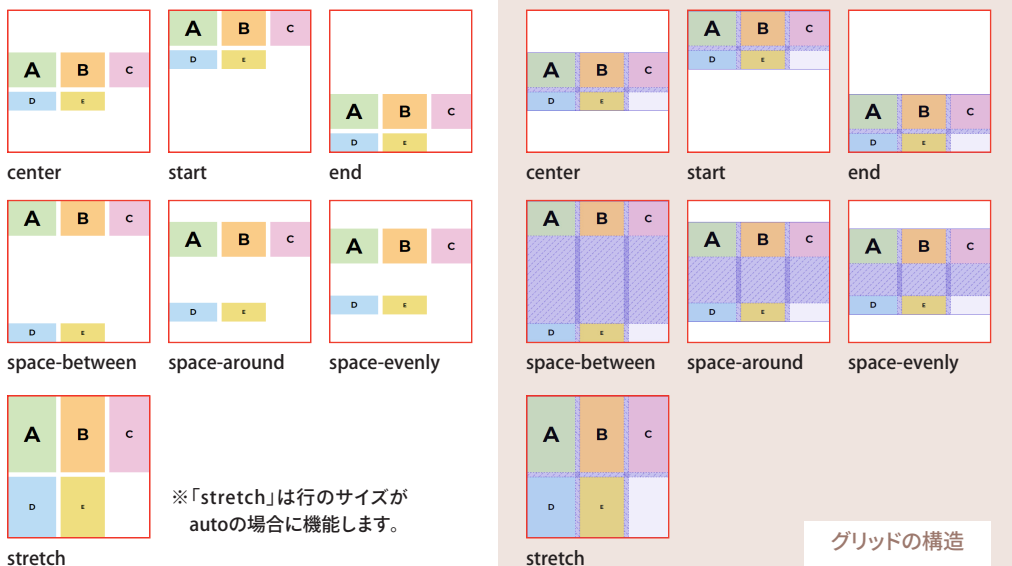
justify-content

グリッドコンテナ内の横方向のグリッドの配置を指定します。



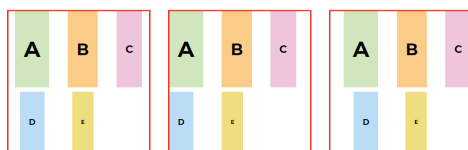
align-content

グリッドコンテナ内の縦方向のグリッドの配置を指定します。



justify-items

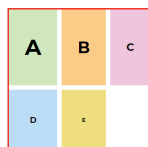
グリッドアイテムの横方向の配置を指定します。



center

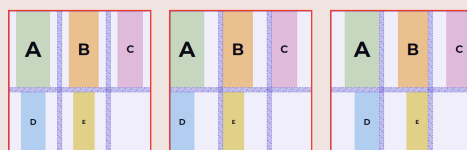
start

end



stretch

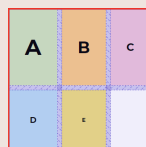
※「stretch」はアイテムの横幅が auto の場合に機能します。



center

start

end

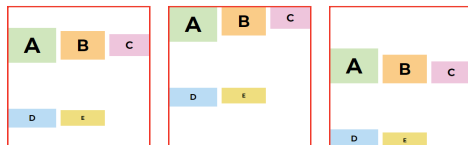


stretch

グリッドの構造

align-items

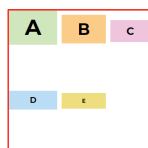
グリッドアイテムの縦方向の配置を指定します。



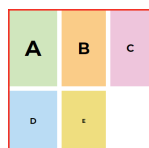
center

start

end

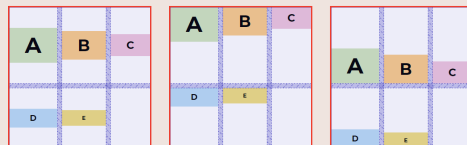


baseline



stretch

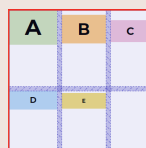
※「stretch」はアイテムの高さが auto の場合に機能します。



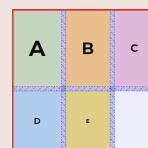
center

start

end



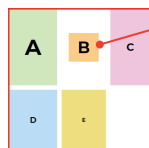
baseline



stretch

グリッドの構造

アイテムごとに配置を変えたい場合、justify-self と align-self で個別に指定します。



```
.boxB {
  justify-self: center;
  align-self: center;
}
```

配置に関する設定は次のプロパティでまとめて指定できます。

プロパティ	まとめて指定できる設定
place-content	justify-content と align-content の設定
place-items	justify-items と align-items の設定
place-self	justify-self と align-self の設定

CSS関数

CSS 関数では、与えた値などに応じて特定の処理結果が返ってきます。

比較関数

■ min()

与えた値の中から最小になる値を返します。

```
width: min(100vw, 500px);
```

処理結果

画面幅(100vw)が
499px以下の場合

```
width: 100vw;
```

画面幅(100vw)が
500px以上の場合

```
width: 500px;
```

■ max()

与えた値の中から最大になる値を返します。

```
width: max(100vw, 500px);
```

画面幅(100vw)が
500px以下の場合

```
width: 500px;
```

画面幅(100vw)が
501px以上の場合

```
width: 100vw;
```

■ clamp()

与えた値を返しますが、最小値と最大値の範囲に収めます。

```
width: clamp(360px, 100vw, 500px);
```

最小値

値

最大値

画面幅(100vw)が
360px以下の場合

```
width: 360px;
```

画面幅(100vw)が
361~499pxの場合

```
width: 100vw;
```

画面幅(100vw)が
500px以上の場合

```
width: 500px;
```

変数（カスタムプロパティ）

■ var()

「--（2つのダッシュ）」で始まる変数（カスタムプロパティ）の値を返します。

```
:root {--primary: red;}  
h1 {color: var(--primary);}
```



処理結果

```
h1 {color: red;}
```

数式

■ calc()

数式の計算結果を返します。

```
:root {--base: 16px;}  
h1 {font-size: calc(var(--base) * 4);}
```

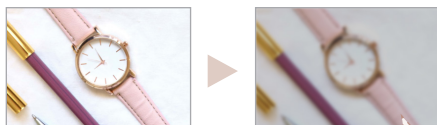


処理結果

```
h1 {font-size: 64px;}
```

フィルタ

フィルタ効果をかける関数です。filter プロパティで使います。



```
img {filter: brightness(80%) blur(3px);}
```

明るさとブラーのフィルタを適用。

関数	フィルタ
blur()	ブラー
brightness()	明るさ
contrast()	コントラスト
grayscale()	グレースケール
hue-rotate()	色相
saturate()	彩度
invert()	階調の反転
opacity()	不透明度
sepia()	セピア
drop-shadow()	ドロップシャドウ

トランスフォーム

変形処理をかける関数です。transform プロパティで使います。



```
img {transform: translate(50%, 0) rotate(-30deg);}
```

移動と回転の処理を適用。

関数	変形処理
translate()	移動
scale()	拡大縮小
rotate()	回転
skew()	スキュー(シア)

フィルタやトランスフォーム関数の設定と処理結果は、オンラインツールを使うと簡単に確認できます。

Filter CSS Generator

<https://cssgenerator.org/filter-css-generator.html>

Transform CSS Generator

<https://cssgenerator.org/transform-css-generator.html>

単位 / 値

単位

数値でサイズを指定するときに使用できる単位です。

※Safariは未対応

単位		単位		単位	
px	ピクセル	pc	パイカ (1pc=12pt)	ch	1ch=数字「0(ゼロ)」の横幅
cm	センチメートル	em	1em=フォントサイズ	vw	100vw=画面の横幅
mm	ミリメートル	ex	1ex=小文字xの高さ	vh	100vh=画面の高さ
in	インチ (1in=約2.54cm)	rem	1rem=ルート要素のフォントサイズ	vmin	vwとvhの小さい方
pt	ポイント	Q	級数 (1Q=1/40cm=0.25mm) ※	vmax	vwとvhの大きい方

色の値

色の指定に使用できる値です。

色の値

色名	色名 (red など)
#RRGGBB / #RGB	RGB16進数 (#ff000 / #f00 など)
#RRGGBBAA / #RGBA	RGBA16進数 (#ff00077 / #f007 など)
rgb(R,G,B) / rgb(R G B)	RGB (rgb(255,0,0) / rgb(255 0 0) など)
rgba(R,G,B,A) / rgba(R G B / A)	RGBA (rgb(255,0,0,1) / rgb(255 0 0 / 1) など)
hsl(H,S,L) / hsl(H S L)	HSLカラー (hsl(360,100%,50%) / hsl(360 100% 50%) など)
hsla(H,S,L,A) / hsl(H S L A)	HSLAカラー (hsl(360,100%,50%,1) / hsl(360 100% 50% / 1) など)
currentColor	colorプロパティの値
transparent	透明

デフォルトの設定にする値

デフォルトの設定にするための値です。すべてのプロパティで指定できます。

値	処理
initial	プロパティの初期値に設定。
inherit	親から継承した値に設定。
revert	ブラウザが標準で適用するUAスタイルシートの値に設定。
unset	値を継承する場合は「inherit」、それ以外の場合は「initial」で処理。

■著者紹介

エビスコム

<https://ebisu.com/>

さまざまなメディアにおける企画制作を世界各地のネットワークを駆使して展開。コンピュータ、インターネット関係では書籍、デジタル映像、CG、ソフトウェアの企画制作、WWW システムの構築などを行う。

主な編著書：『HTML5 & CSS3 デザイン 現場の新標準ガイド【第2版】』マイナビ出版刊
『Web サイト高速化のための 静的サイトジェネレーター活用入門』同上
『CSS グリッドレイアウト デザインブック』同上
『WordPress レッスンブック 5.x 対応版』ソシム刊
『フレキシブルボックスで作る HTML5&CSS3 レッスンブック』同上
『CSS グリッドで作る HTML5&CSS3 レッスンブック』同上
『HTML&CSS コーディング・プラクティスブック 1～7』エビスコム電子書籍出版部刊
『グーテンベルク時代の WordPress ノート テーマの作り方（入門編）』同上
『グーテンベルク時代の WordPress ノート テーマの作り方
（ランディングページ&ワンカラムサイト編）』同上

ほか多数

作って学ぶ HTML & CSS モダンコーディング【特典 PDF】

HTML&CSS 簡易リファレンス

2021 年 9 月 20 日 ver.1.0 発行