

## 4.2 CSS 基礎

- CSS
- CSS の基本文法
- 代表的なプロパティ
- レイアウト

# 目次

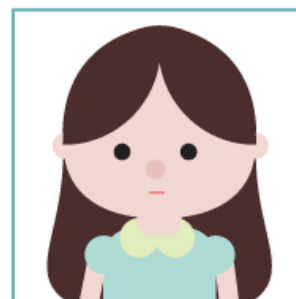
- 1 CSS
- 2 CSS の基本文法
- 3 代表的なプロパティ
- 4 レイアウト

# CSS

- **CSS** (Cascading Style Sheets) とは、HTML ページの**スタイル**を記述するためのスタイルシート言語です。
- CSS は、HTML 要素のサイズや色など、ブラウザでの表示方法を具体的に指定するものです。
- 化粧品は人がいないと意味がないように、CSS のコードだけでは意味がありません。



**HTML は**  
**顔のパーツ**



**CSS は**  
**お化粧**

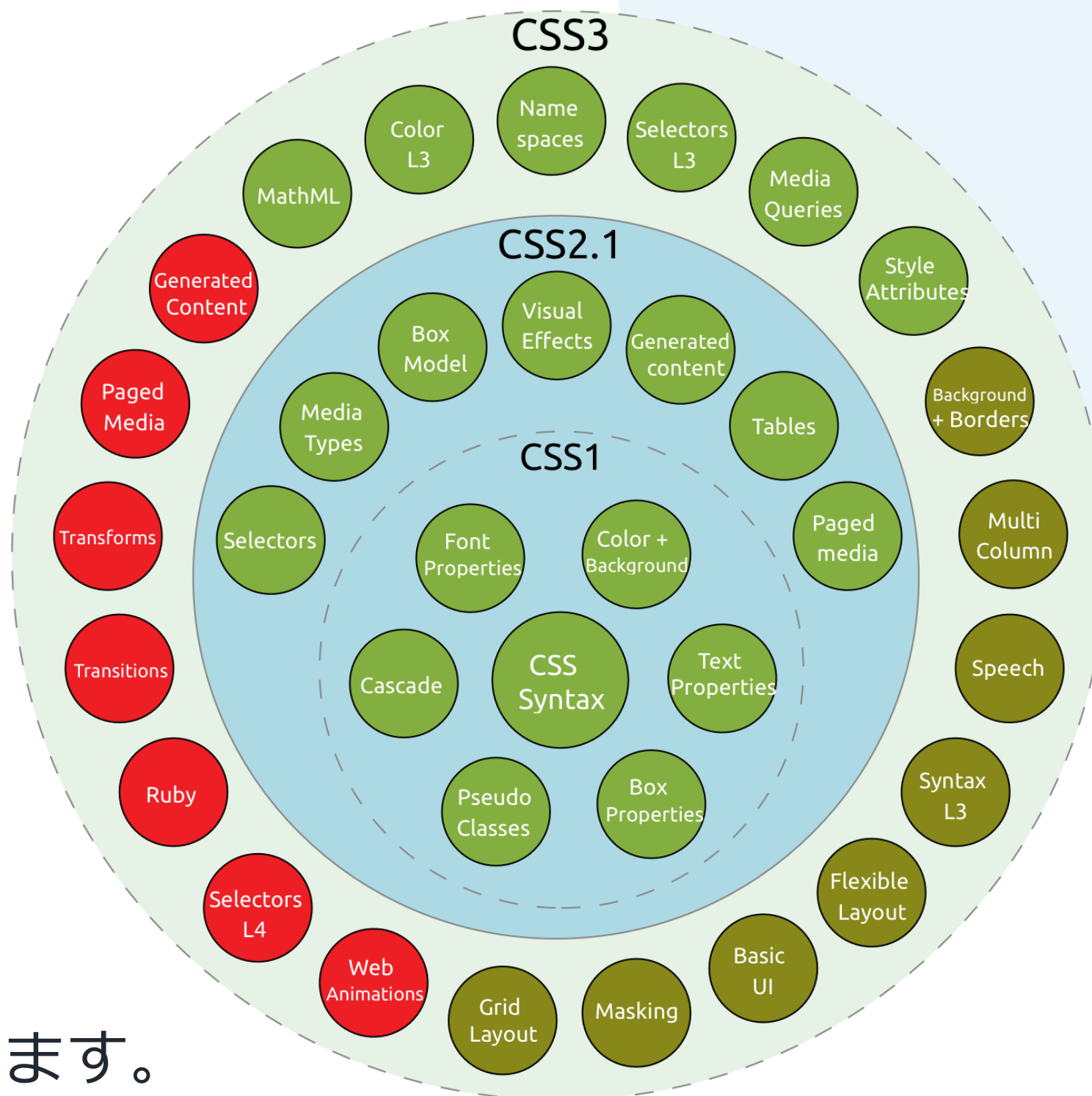


**JavaScript は**  
**動き**





# CSS の歴史



- CSS3 を紹介します。

# CSS の使用

- CSS で HTML の要素のスタイルを変更する方法は、**3 つ**あります：
  - `<link>` タグで**外部 CSS ファイル**を取り込む。
  - `<style>` タグを使って、HTML **文書の中に直接 CSS コード**を記述する。
  - `style` 属性を使用して、**要素を直接変更**する。

# 外部 CSS ファイルの読み込み

- CSS のコードは、拡張子が「**.css**」のファイルに書くことが一般的です。
- **外部 CSS** ファイルを取り込むには、**<link>** タグを使用します（通常は **<head>** タグの中に置く）：

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

- **rel** 属性の値である「stylesheet」は、スタイルシートを取り込むことを示します。**href** 属性の値は、取り込みたい **CSS ファイルの URL** となります。

# 内部 CSS コード

- CSS コードは、**<style>** タグ（通常は <head> タグの中）に**直接記述**することもできます：

```
1 <style>
2   p {
3     color:green;
4   }
5 </style>
```

# インライン CSS

- 任意の要素の **style** 属性で、その要素のスタイルを直接変更することができます：

```
<button style="color:blue">This is a button</button>
```

- これは**インライン**<sup>[Inline]</sup> **CSS** ともいいます。

Try <sup>01011  
11010  
01011</sup>  
style フォルダ



# スタイルの優先順位

- 同じ要素のスタイルを異る CSS コードで同時に変更した場合、最終的にどの CSS スタイルが適用されるのでしょうか？
- 複数の CSS スタイルが競合する場合は、以下の流れで適用されるスタイルを決めます：
  1. 要素にインラインスタイルがある場合、**インラインスタイル**が優先に適用されます。
  2. 要素にインラインスタイルがなかったら、**最後に**実行される（タグがページの一番下の）スタイルが適用されます。

Try 01011  
11010  
01011  
order フォルダ

# CSS の利用方法の選択

- どの方法で要素のスタイルを変更するかには、厳密な基準などありません。必要に応じて最も便利な方法を選択しましょう。
- 一般的に言えば：
  - **外部 CSS** を利用して、**サイト全体**で一貫したスタイルを指定する。
  - **内部 CSS** は、**単独のページ全体**で一貫したスタイルを指定する。
  - **インライン CSS** は、**単独の要素**に特化したスタイルを指定する。
- 一般に、統一性や拡張しやすさを高めるために、できるだけ**外部 CSS ファイル**を使うのが望ましいです。

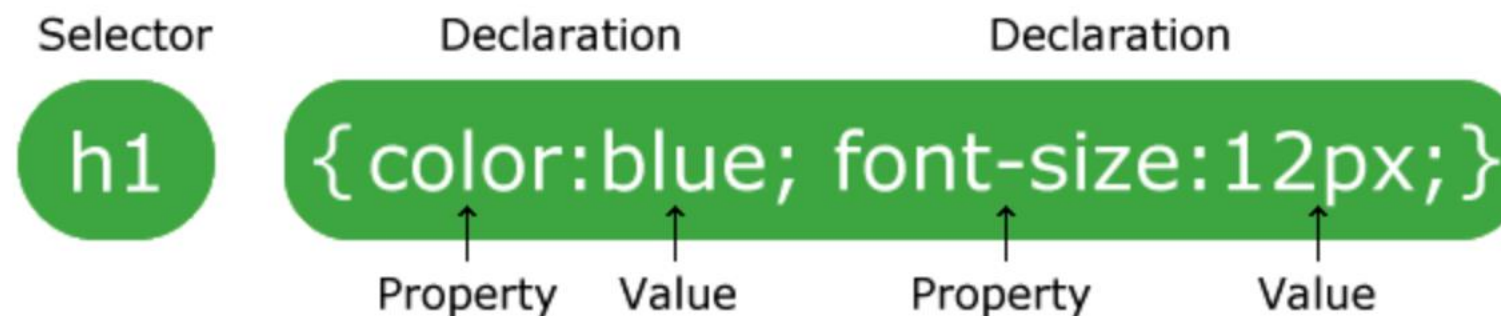
# Q&A

# 目次

- 1 CSS
- 2 CSS の基本文法
- 3 代表的なプロパティ
- 4 レイアウト

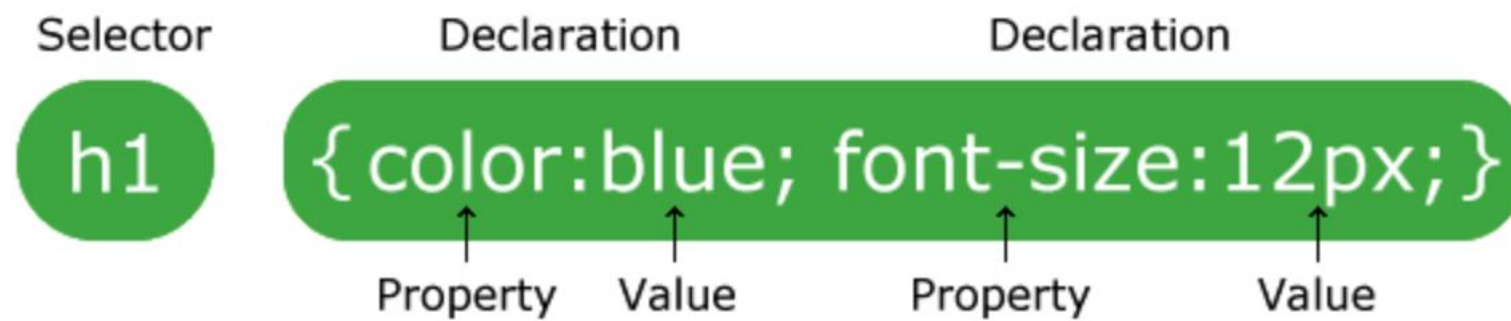
# CSS の基本構文

- CSS は、複数の**ルールセット** [Rule Set] から構成されます：



- ルールセットは、「{ }」の中で宣言されたスタイルを適用する要素を指定する**セレクター** [Selector] から始まります。この例では <h1> 要素を指定しているので、すべての <h1> 要素はこれらのスタイルになる、と意味しています。





- 中括弧「`{ }`」で囲まれた部分は**宣言ブロック** [Declaration Block] です。複数の**宣言** [Declaration] を含めることができます。複数の宣言は、セミコロン「`;`」で区切られます。
- 宣言は**プロパティ** [Property] とプロパティに設定させる値によって構成されます。コロン「`:`」の前にはプロパティの名前で、後ろはプロパティ値です。

# セレクター

- ルールセットでどの要素がスタイルを使用するかを指定するために、**セレクター**を使用します。
- セレクターは大きく分けて 5 種類があります：
  - 基本セレクター：名前、ID、クラスに基づいて要素を選択
  - 属性セレクター：HTML タグの属性に基づいて要素を選択
  - 結合子<sup>[Combinator]</sup>セレクター：他の要素との関係性に基づいて要素を選択
  - 擬似クラス<sup>[Pseudo-class]</sup>セレクター：要素の特定の状態を選択
  - 擬似要素<sup>[Pseudo-Element]</sup>セレクター：要素の特定の部分を選択

# 要素セレクター

- 基本的なセレクターには、要素の**タグ名**、**ID**、**クラス**のいずれかによって要素を指定するものがあります。
- **要素セレクター**は、**要素名**（タグ名）をもとに特定の要素を選択するものです。要素を選択するには、その要素の名前を書くだけです。
- 次のコードは、ページ上のすべての `<p>` 要素のテキストを赤にします：

```
1 p {
2   color: red;
3 }
```

# ID セレクター

- **ID セレクター**は、要素の **id 属性**によって要素を指定します。
- 原則として、各要素の id 属性は、ページ内で**一意になるように設定**する必要があります。したがって、ID セレクターは特定の **1 つの要素**を選択するために使用されます。特定の ID をもつ要素を選択するには、ハッシュタグ「**#**」の直後に要素の ID を記述します。
- 次のコードは、ID が「text-1」の要素だけを赤にします：

```
1 #text-1 {
2     color: red;
3 }
```

# クラスセレクター

- **クラスセレクター**は、要素の **class 属性**に基づく特定の要素を選択します。
- id 属性とは異なり、複数の要素の class 属性に同じ値を設定することができます。したがって、クラスセレクターは、同じクラスに属する**複数の要素**を選択できます。

## Tips💡

Java の変数名とは異なり、HTML 要素の ID 名とクラス名にはハイフン「-」を使用することができます。

次へ



- 特定のクラスを持つ要素を選択する場合は、ピリオド「.」の直後に要素のクラス名記述します。
- 次のコードは、ページ上のすべての class-1 の要素のテキストを赤くします：

```
1 .class-1 {  
2     color: red;  
3 }
```

# 要素名とクラスの組み合わせ

- 要素名とクラス名を「.」で繋げることで、その要素名で、**かつ**そのクラスに属する要素を指定できます。
- 次のコードは、ページ上の class-1 に属するすべての <p> 要素のテキストを赤にします：

```
1 p.class-1 {  
2   color: red;  
3 }
```

# 複数のクラスに所属する要素

- 同じ要素が**複数のクラス**に属することもできます。
- class 属性を設定する場合、**一つ**の「'''」の中で、複数のクラス名をスペースで区切って書くことで複数のクラス名を設定できます。
- 例えば、次の <p> 要素は「class-1」と「class-2」の両方に属して、両方のスタイルの影響を受けます：

```
<p class="class-1 class-2">Two Classes</p>
```

# 全称セレクター

- **全称セレクター**はページ上の**全ての要素**を選択するときに使用します。アスタリスク「**\***」で表現します。
- 次のコードは、ページ上のすべての要素のテキストを赤にします：

```
1 * {
2   color: red;
3 }
```

Try 01011  
11010  
01011  
selector.html

## Note

異なるセレクターの適用優先順位（上のほうが優先）：

1. ID セレクター
2. クラスセレクター
3. 要素名セレクター
4. 全称セレクター

# セレクトーリスト

- 複数のセレクトーが同じ（または部分が共通の）スタイルを定義したいときがあります。
- 複数のセレクトーをリストで一括に表現できます。
- セレクトーリストを使うためには、複数のセレクトーをカンマ「**,**」で区切ります。
- 次のコードは、ページ上のすべての `<h1>` 要素、`<h2>` 要素、および ID が `text-1` の要素を赤に変更します：

```
1 h1, h2, #text-1 {
2     color: red;
3 }
```



# 結合子セレクター

- 結合子は、複数のセレクターを組み合わせ、**他の要素との関係**に基づいて要素を選択できます。
- 主に 4 種類の結合子が使われます：
  1. **子孫**[Descendant]**結合子**：要素の**中にある**要素を選択。スペース「」で表現。
  2. **子**[Child]**結合子**：要素の中に**直接含む**要素（子要素）を選択。大なり記号「`>`」で表現。
  3. **隣接兄弟**[Adjacent Sibling]**結合子**：要素の**すぐ後ろにある**、同レベルの要素（兄弟要素）を選択。プラス「`+`」で表現。
  4. **一般兄弟**[General Sibling]**結合子**：要素と**同レベル**のすべての要素（兄弟要素）を選択。チルダ「`~`」で表現。

Try 01011  
11010  
01011  
combinator.html

# 擬似クラスセレクター

- **擬似クラスセレクター**は**特定の状態**である要素のスタイルを指定します。
- 例えば、ハイパーリンクの要素 `<a>` は、**通常状態** (link)、**クリックした状態** (visited)、**カーソルが上にある状態** (hover)、**クリックしている状態** (active) などの状態に分けることができます。これらの状態を**擬似クラス**と呼びます。
- 擬似クラスセレクターを使用するには、通常のセレクターに続いて、コロン「`:`」と擬似クラス名 (状態名) を書きます。

Try 

pseudo-class.html

# コメント

- CSS の**コメント**は、Java の複数行コメントに似ており、「**/\***」で始まり、「**\*/**」で終わります：

```
1 /*  
2  * This is a CSS comment.  
3  */  
4 p { color: red; }
```

# Q&A

# 目次

- 1 CSS
- 2 CSS の基本文法
- 3 代表的なプロパティ
- 4 レイアウト



# テキストの色

- これまでの多くの例で見てきたように、**color** プロパティは要素内の**テキストの色**を設定します：

```
1 <h1 style="color:red">This Is A Heading</h1>
2 <p style="color:green">This is a paragraph.</p>
3 <button style="color:blue">This is a button</button>
```

**This Is A Heading**

This is a paragraph.

This is a button

# 背景の色

- 要素の背景の色は、**background-color** プロパティで指定することができます：

```
1 <h1 style="background-color:red">This Is A Heading</h1>  
2 <p style="background-color:green">This is a paragraph.</p>  
3 <button style="background-color:blue">This is a button</button>
```

**This Is A Heading**

This is a paragraph.

This is a button

# 色の表現

- CSS で色を表現する方法は数多くあります：

- 色の名称：tomato

- RGB 表現：rgb(255, 99, 71)

- 16 進表現：#ff6371（カラーコード）

- HSL 表現：hsl(9, 100%, 64%)

- RGBA 表現：rgba(255, 99, 71, 0.5)

- .....

rgb(255,99,71)


#ff6371

hsl(9,100%,64%)

rgba(255,99,71,0.5)

hsla(9,100%,64%,0.5)

# CSS 色名

- CSS3 は 147 の色名があります。
- 色名の一覧は、こちらのページでご覧いただけます：  
 <https://www.w3.org/wiki/CSS/Properties/color/keywords>



# テキスト関連プロパティ

- 次表は、テキストに関連する代表的なプロパティを紹介：

プロパティ	定義されるもの
color	テキストの色
text-align	テキストの水平位置の配置方法
vertical-align	テキストの垂直位置の配置方法
text-decoration	文字装飾（下線、取り消し線など）
text-overflow	要素内にテキストを表示しきれないときの制御
font	フォント

# 水平位置の配置

- **text-align** プロパティは、テキストの**水平方向の位置合わせ**を設定します。左、右、中央のいずれかに揃えることができます：

```
1 <p style="text-align:left">This is left-aligned.</p>
2 <p style="text-align:center">This is center-aligned.</p>
3 <p style="text-align:right">This is right-aligned.</p>
```

This is left-aligned.

This is center-aligned.

This is right-aligned.

# フォント

- **font** プロパティは、**フォント**の種類、サイズ、およびその他のスタイルを定義します。一般的なフォント関連のプロパティは以下の通りです：

プロパティ	定義されるもの
font-family	フォントの種類
font-size	フォントのサイズ
font-style	特殊な書式（太字、斜体など）
font-weight	フォントの太さ



# フォントファミリー

- **font-family** プロパティは、テキストがどのフォントファミリーに属しているかを定義します。
- フォントファミリーは、基本的にフォントの種類を表すものだとして理解すればいいでしょう。ただし、1つのファミリーには、例えば太さが違う別のバージョンなど、複数のフォントを含むことができます。

次へ 

- また、**総称ファミリー**<sup>[Generic Family]</sup>と呼ばれる特殊なファミリーも存在します。大体のフォントは、ある種の総称ファミリーに属しています。代表的な総称ファミリーには、**Serif**、**Sans-serif**、**monospace** などがあります。

F

Sans-serif

F

Serif

F

Serif  
(red serifs)

# フォントファミリーを指定

- **font-family** プロパティの後に、**複数のファミリー名**をカンマ「**,**」で区切って指定することができます：

```
1 p {
2   font-family: Arial, Helvetica, sans-serif;
3 }
```


- ブラウザは、まず 1 番目のフォントが利用可能かどうかを調べ、利用可能でない場合は 2 番目のフォントを調べ、といった具合になります。そのため、ブラウザが確実に適切なフォントを見つけるように、**最後のファミリー名を総称ファミリー**のいずれかに設定することをお勧めします。
- ファミリー名が複数の単語を含む場合、"Times New Roman " のように、「"」で囲む必要があります。

# ウェブフォントを使おう

- ブラウザーが直接サポートするフォントは限られていますが、いくつかの**ウェブフォント**[Web Font]を使ってページを装飾することができます。
- フォントを提供してくれるサイトはたくさんあります。例えば、Google Fonts、Fonts.com、Adobe Fonts、TypeSquare などがあります。

# Google Fonts

- 例として、**Google Fonts** の使い方を紹介します：

1.  <https://fonts.google.com/> で必要なフォントを検索。
2. ページ右側の `<link>` タグを HTML 文書にコピー。
3. このフォント名を用いて、対応する要素の `font-family` プロパティを CSS で設定

Try  font.html

Use on the web

To embed a font, copy the code into the `<head>` of your html

☒ `<link>`
☐ `@import`

```

<link rel="preconnect" href="https://
fonts.googleapis.com">
<link rel="preconnect" href="https://
fonts.gstatic.com" crossorigin>
<link href="https://fonts.googleapis.
com/css2?family=VT323&display=swap" r
el="stylesheet">


```

CSS rules to specify families

```
font-family: 'VT323', monospace;
```

# アイコン

- Google Fonts を使えば、**アイコン**<sup>[Icon]</sup>も簡単に追加できます：

1.  <https://fonts.google.com/icons?icon.set=Material+Icons> で使いたいアイコンの名前を探す。
2. <head> タグの中で以下のスタイルシートをリンク：

```
<link href="https://fonts.googleapis.com/icon?family=Material+Icons"
      rel="stylesheet">
```

3. アイコンを入れたい場所に次のようなタグを入れる：

```
<span class="material-icons">アイコンの名称</span>
```

4. この後、CSS でアイコンの色などを変更することもできる。



# 文字サイズ

- **font-size** プロパティは、テキストの**サイズ**を設定します。
- 単位は px（ピクセル）、em（文字サイズとの相対比率）、rem（ルート要素の文字サイズとの相対比率）など複数用意されています：

font-size:16px

font-size:32px

font-size:1.5rem

## Note

font-size を、見出しとテキストを区別するために**使用しないでください**。見出しは `<h1>` ~ `<h6>` タグで指定すること。



# <span> タグを使ってプロパティを設定

- CSS のプロパティを <h1> や <p> などのタグに適用すると、<h1> や <p> タグ内のすべてのテキストに影響が与えられます。
- テキストの中の**一部だけ**のスタイルを変更したい場合、**<span>** タグでこの部分を囲み、CSS でこの <span> タグのプロパティを設定すればいいです。
- 例えば、以下のコードでは、テキストの「赤い」の部分だけが赤色に設定されます：

```
<h1>この部分だけは<span style="color:red">赤い</span>です。</h1>
```

# 高さと幅

- **height** と **width** プロパティを使って、それぞれ要素の**高さ**と**幅**を設定することができます。
- height と width の値にも、px（ピクセル）、em（文字サイズとの相対比率）、%（パーセント）など指定できます。

# まとめ : CSS のサイズの単位

## Sum Up

単位	意味
px	ピクセル（画素）数
cm、mm、inch など	センチメートル、ミリメートル、インチなど物理的な単位
em	文字サイズに対する比率
rem	ルート要素の文字サイズに対する比率
%	親要素の幅・高さに対する比率
vw / vh	ビューポートの幅・高さ

# 背景

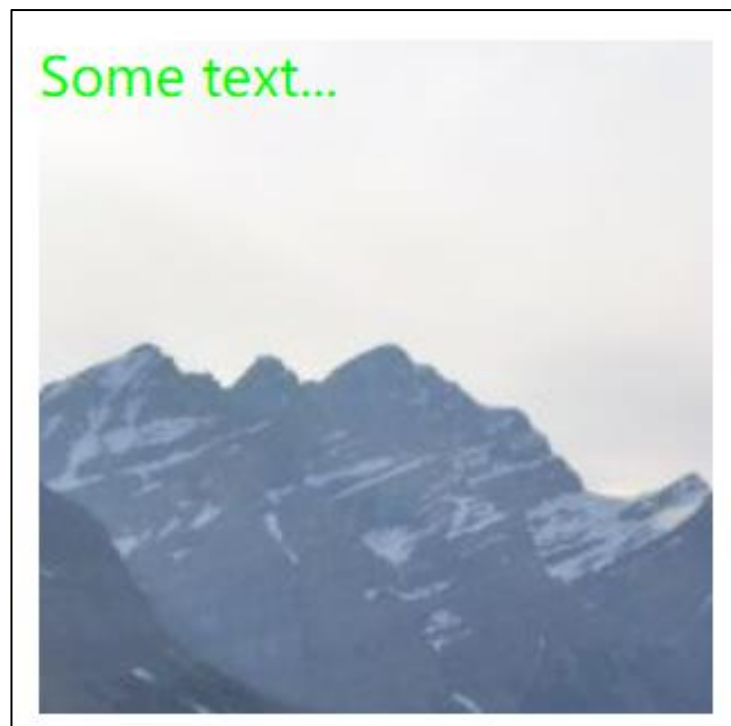
- 背景関連のプロパティは以下の通り：

プロパティ	定義されたもの
background-color	背景の色
background-image	背景画像
background-size	画像のサイズ
background-position	画像の位置
background-repeat	画像の繰り返しの仕方
background-attachment	ページがスクロールするとき背景画像も一緒に移動するか

# 背景画像

- **background-image** プロパティは、要素の背景となる画像を指定します：

```
1 #a {
2   background-image: url(img/mountains.jpg);
3 }
```



# 背景画像のサイズ

- **background-size** プロパティは画像の**サイズの設定方法**をブラウザに伝えます。以下の値に設定することができます：

値	効果
contain	可能な限りフル画像を表示（縦横比は変えない）
cover	可能な限り要素をカバー（縦横比は変えない）
サイズ値 サイズ値 例：200px 300px	画像を指定された幅と高さにする

## Tips

拡大縮小された画像を繰り返したくない場合は  
repeat: no-repeat を使えばいい。

# 背景の一括指定

- 先ほど紹介したプロパティを **background** プロパティで一括に定義することができます：

```
1 #first {
2   background-image: url("img/mountains.jpg");
3   background-repeat: no-repeat;
4   background-attachment: fixed;
5 }
```

```
1 #first {
2   background: no-repeat fixed url("img/mountains.jpg");
3 }
```

- CSS には、このような**一括指定**<sup>[Shorthand]</sup>プロパティは他にもたくさんあります。

Try  [background.html](#)



# その他のプロパティ

- opacity : 汎用プロパティ。要素の透明度を設定。
- animation : 汎用プロパティ。アニメーション効果を設定。
- inherit : 汎用的なプロパティの値。親要素から継承。
- linear-gradient、radial-gradient : 色プロパティの値。グラデーションカラーを設定する関数。
- CSS の公式ドキュメントは参考しづらいので、Mozilla 社のドキュメントを参考するのはおすすめします :

<https://developer.mozilla.org/en-US/docs/Web/CSS>



# Q&A

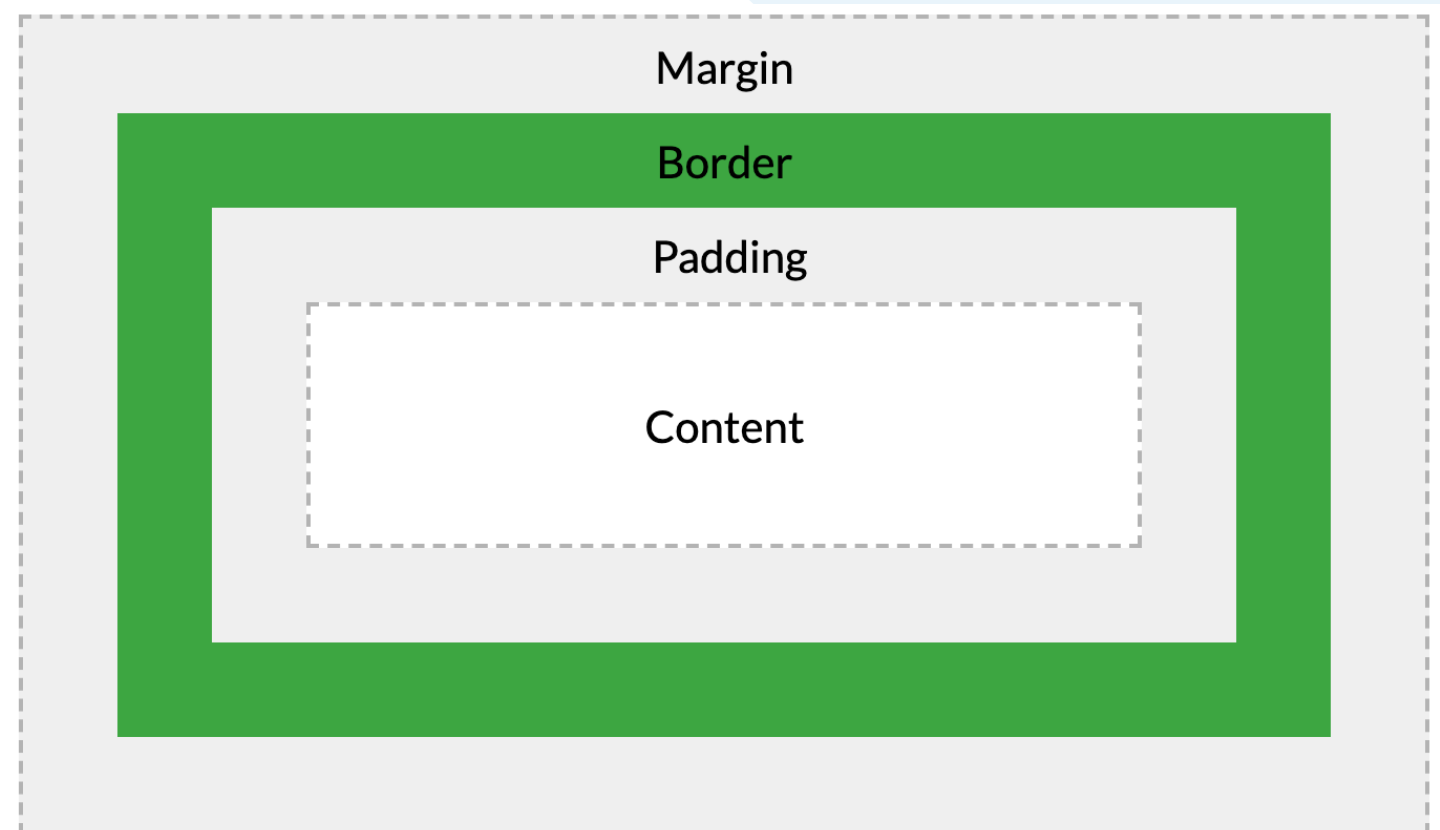
# 目次

- 1 CSS
- 2 CSS の基本文法
- 3 代表的なプロパティ
- 4 レイアウト

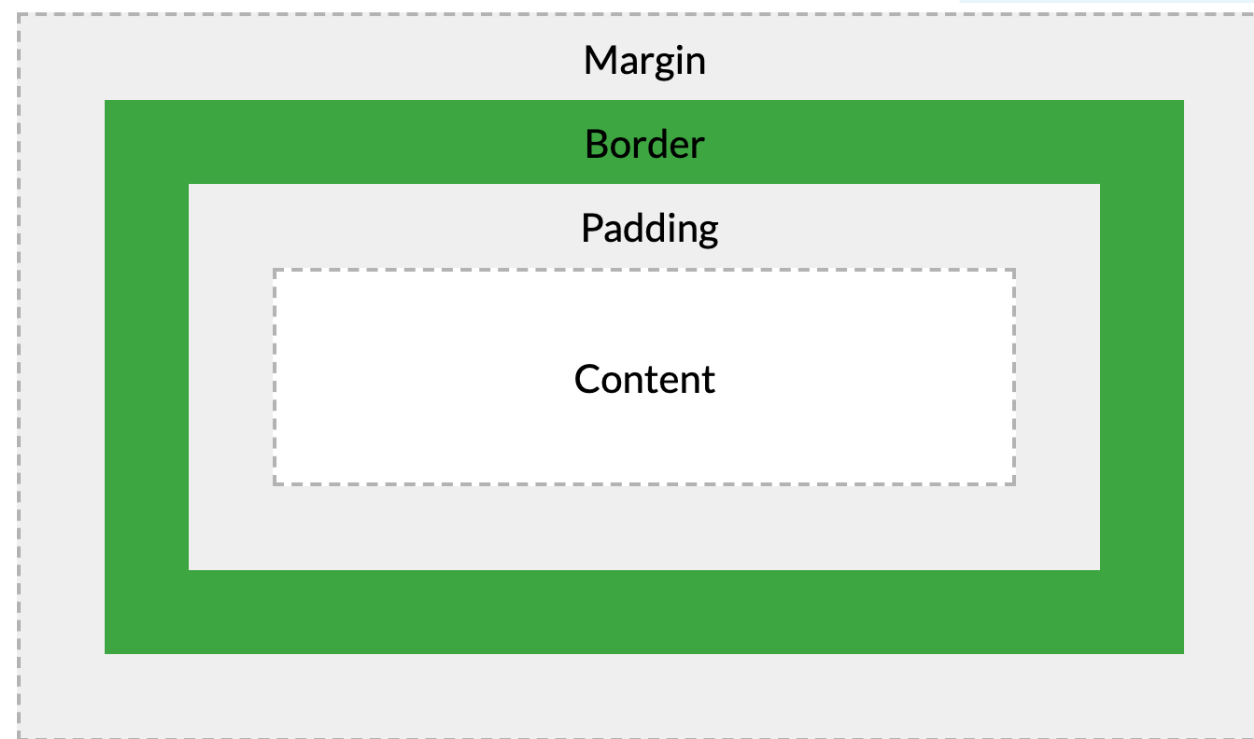
# ボックスモデル

- CSS では、デザインやレイアウトに関して、「**ボックスモデル**[Box Model]」という言葉が使われます。ボックスモデルは、各要素を 4 つの部分に分解します：


- **コンテンツ**[Content]、
- **パディング**[Padding]、
- **ボーダー**[Border]、
- **マージン**[Margin]。



# ボックスモデル



- **コンテンツ**：内容が実際に表示する部分
- **パディング**：コンテンツとボーダーの間の空白
- **ボーダー**：境界線の部分
- **マージン**：ボーダーの外側にある空白、他の要素から分離するために利用

Try  [box-model.html](#)

# ボーダー

- まずは、**ボーダー**から学びましょう。要素のボーダーに関連する共通のプロパティは以下の通りです：

プロパティ	定義されるもの
border-style	ボーダーのスタイル（種類）
border-width	ボーダーの太さ
border-color	ボーダーの色
border-radius	ボーダーの角丸の半径

# ボーダーの種類

- **border-style** プロパティは、表示される**ボーダーの種類**を指定します。例えば、solid に設定すると、実線のボーダーになり、dashed に設定すると、破線のボーダーになります：

`border-style: solid`

`border-style: dashed`

`border-style: dotted`

`border-style: double`

`border-style: groove`

`border-style: ridge`

`border-style: inset`

`border-style: outset`

# ボーダーの太さ

- **border-width** プロパティは、ボーダーの**太さ**を指定します。
- 10px、1em などの具体的な値にも設定できれば、「thin」「medium」「thick」のいずれかにも設定できます：

```
border-width: 5px
```

```
border-width: thin
```

```
border-width: medium
```

```
border-width: thick
```



# ボーダーの色

- **border-color** プロパティは、ボーダーの色を指定します：

```
border-color: red
```

```
border-color: yellow
```

```
border-color: green
```

```
border-color: blue
```

# 上下左右を一括設定

- ボーダーは上下左右の **4 つの辺** で構成されています。先ほどのプロパティが、1 つの値しか含まない場合、**全ての辺** に適用されます：

```
border-color: red
```

- 2 つの値が含まれる場合、それぞれ**上下・左右**の辺に適用：

```
border-color: red lime
```

- 3 つの値が含まれる場合、それぞれ**上・左右・下**に適用：

```
border-style: solid double dotted
```

- 4 つの値が含まれる場合、それぞれ**上・右・下・左**に適用：

```
border-width: 2px 3px 4px 5px
```

# 上下左右を個別設定

- border-top-、border-right-、border-bottom-、border-left- で始まるプロパティを使って、**個別の辺**のスタイルを設定することも可能です：

```
border-style: solid;
border-top-color: red;
border-right-style: dotted;
border-left-width: 5px;
border-left-color: blue;
border-bottom-style: none;
```

# ボーダー関係スタイルの一括指定

- **border** プロパティは、先ほど説明した一括指定プロパティです。これを利用して、ボーダーの種類、太さ、色を一括に設定することができます：

```
border: solid 2px red
```

```
border: dashed 5px blue
```

- border-top、border-right、border-bottom、border-left を使用して、**個別の辺を一括指定**できます：

```
border-right: double 3px green;
border-bottom: solid 2px red;
```

# ボーダーの角を丸くする

- **border-radius** プロパティで、ボーダーの角を丸くすることができます。大きくするほど、角が丸くになります：

```
border-radius: 2px
```

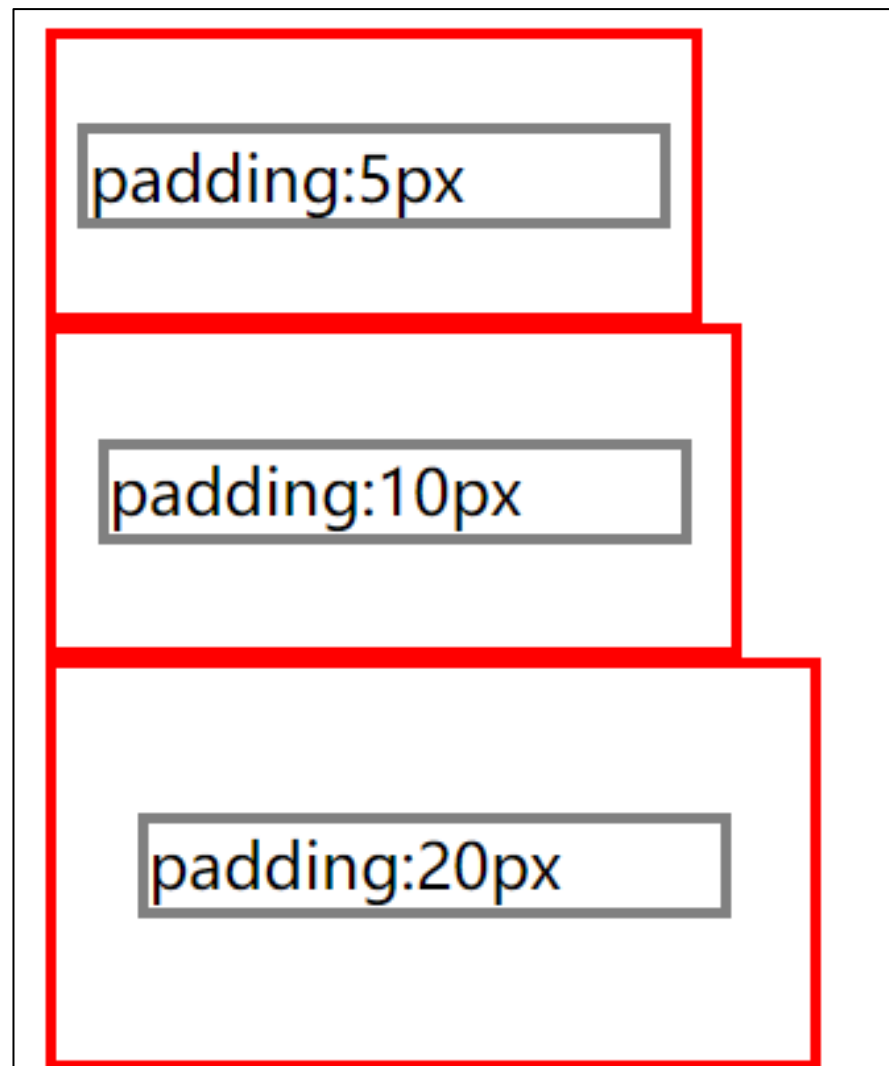
```
border-radius: 5px
```

```
border-radius: 20px
```

Try `border.html`

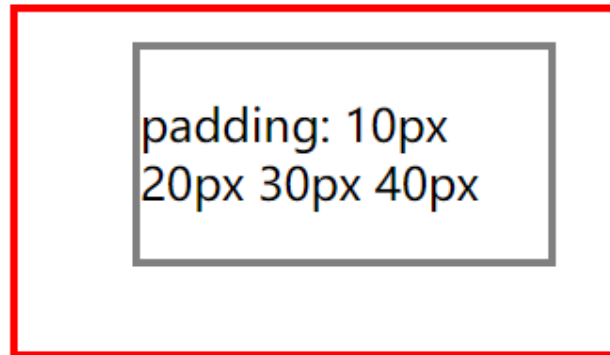
# パディング

- **padding** プロパティでパディングを設定できます：

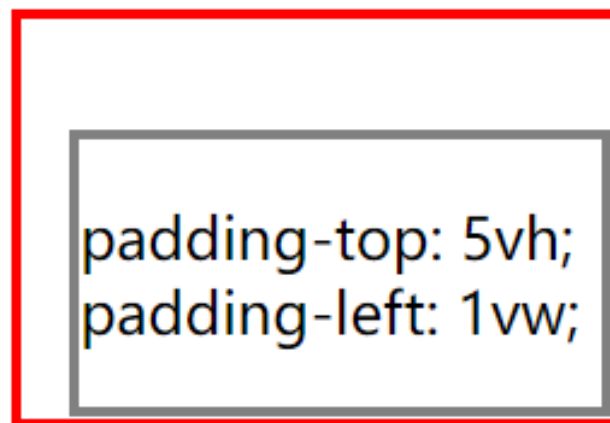


# 上下左右のパディング

- border と同様に、上下左右を一括設定できます：

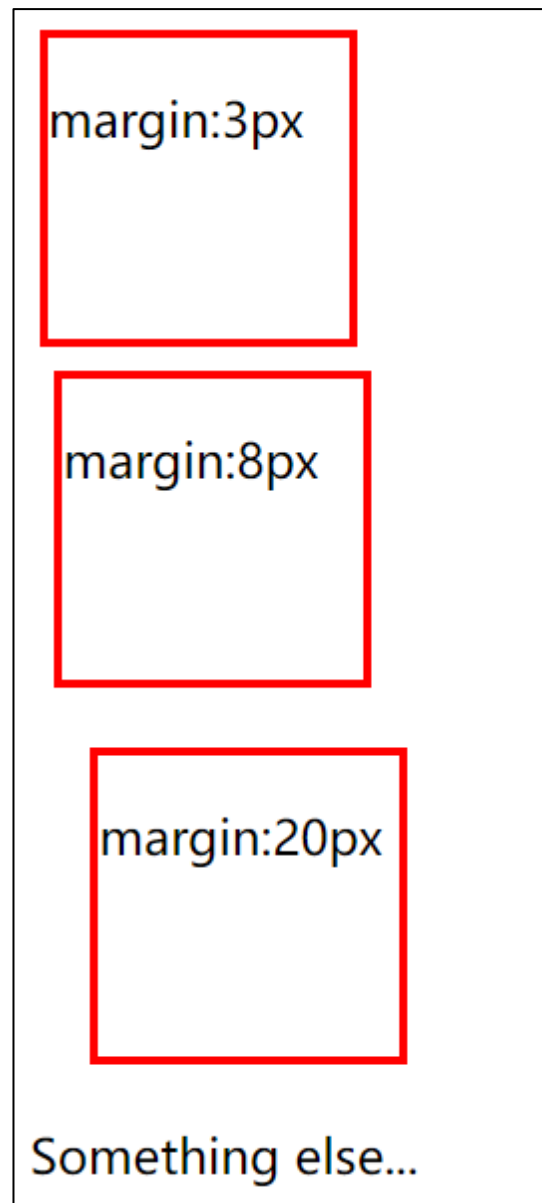


- また、padding-top などのプロパティを使って、特定の方  
向を個別に設定することも可能です：



# マージン

- **margin** プロパティは、要素のマージンを指定します：





# 上下左右のマージン

- ボーダーやパディングと同じように、上下左右の値を一括か個別に指定することもできます：

Something else...

`margin:3% 1% -20px`

Something else...

Something else...

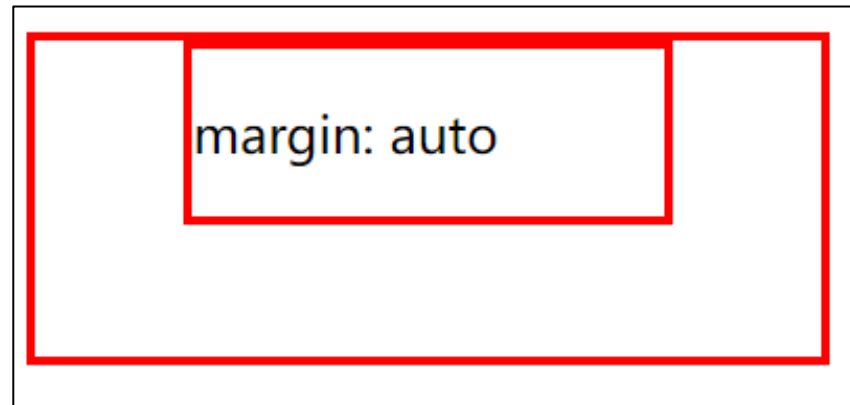
`margin-top: 50px;`  
`margin-bottom: 10px;`

Something else...

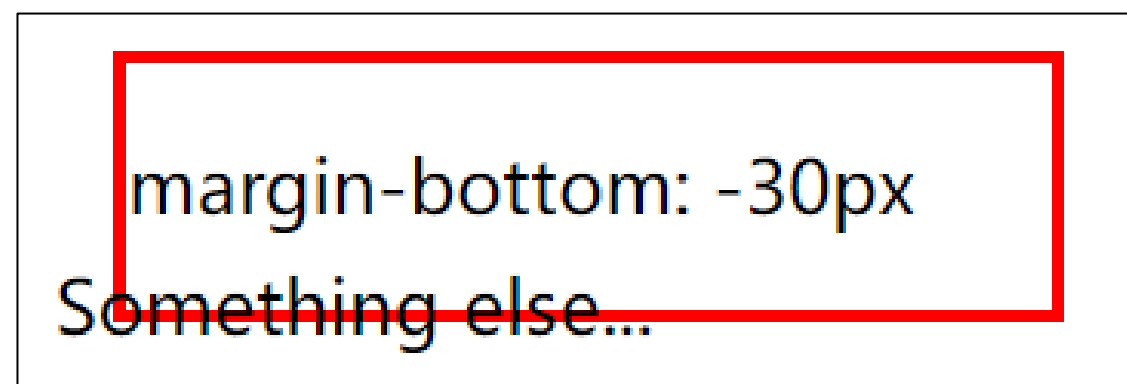
Try `margin.html`

# 特殊のマージンの値

- margin に「**auto**」を指定すると、要素を**水平で中央に揃**えることができます：



- margin プロパティに**負の値**を設定することで、他の要素と「**重なり合う**」ようにすることもできます：



# ボックスモデルの高さと幅

- 注意してください：この前紹介した height と width のプロパティは、要素のコンテンツ部分だけの高さと幅を設定します。
- つまり、それらのサイズ値には、デフォルトではパディング、ボーダー、マージンの部分は含まれません。正しく要素のフルサイズを計算するには、各部分の太さも加える必要があります。



# 高さ・幅の計算

- 例えば、この要素をご覧ください：

```
1 #box {
2   width: 320px;
3   padding: 10px;
4   border: 5px solid red;
5   margin: 0;
6 }
```

- この要素の実際の横幅はどのくらいありますか？
  - $320\text{px}$ （コンテンツの幅） +  $2 \times 10\text{px}$ （左右のパディング） +  $2 \times 5\text{px}$ （左右のボーダー） =  $350\text{px}$
- マージンに関するルールはもっと複雑ですが、ここでは割愛しますが、ここを参考してください：

 [https://developer.mozilla.org/en-US/docs/Web/CSS/CSS\\_Box\\_Model/Mastering\\_margin\\_collapsing](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Box_Model/Mastering_margin_collapsing)

# Q&A

# 要素の位置を配置

- CSS で要素の位置を決めるためには、2 つのステップが必要です：
  1. position プロパティで、要素がどのように配置されるかを決定する（動きの起点を決定）
  2. top、left、bottom、right の各プロパティを使って、オフセット値を決定する（どの方向に、どのくらい動かすかを決定）

# 位置の配置方法

- **position** プロパティは、位置決めの**方法**を指定します。
- 位置を決める方法は 5 種類あります：
  - **static**（デフォルト）：ブラウザが自動的に位置を決め、位置を調整することはできない
  - **relative**：自動に決めた位置から調整
  - **fixed**：ウィンドウ（ビューポート）に対する位置を調整
  - **absolute**：親要素に対する位置を調整
  - **sticky**：最初は、ウィンドウ外にあるかもしれないが、この要素が見える時から、fixed のようになる

# オフセット

- `top`、`right`、`bottom`、`left` プロパティは、位置の**オフセット**（移動する距離）を指定します。
- 例えば、`top` は、指定された基準（例えば、`fixed` はウィンドウ、`absolute` は親要素）の**上**の境界からどれだけ離れているかを指定します。`right`、`bottom`、`left` も同様に。

## Note

position を `absolute` に設定するには、**親要素の position が `static` 以外**のものにしなければなりません。

Try   
layout.html



# displayプロパティ

- displayプロパティは、「要素の表示形式」を決めるものです。
- displayに対する値としてまずは、次の5つを覚えましょう。

➤ block

➤  inline

p,div,ul,h1~h6などのタグの初期値はこれです。

➤  inline-block

a,span,imgなどのタグの初期値はこれです。

➤ none

➤ flex

次へ 

- 他にもdisplayの値がありますので、下記リンクから確認してみましょう

 <https://developer.mozilla.org/en-US/docs/Web/CSS/display?v=example>

- ただし、まずはこの5つを覚えておけば良いかと思います。  
この中で「pタグはblock」、「aタグはinline」というように  
に  
タグごとにdisplayプロパティの初期値が決まっています。  
ほとんどのタグはblockもしくはinlineが初期値となっています。

# それぞれの値のイメージ

- displayプロパティにより要素の表示のされ方や高さ・幅の設定が変わってきます。下図の値の表示のされ方のまとめを見てみましょう

displayに対する値	
block	<div>要素</div> <div>要素</div>
inline	<div>要素</div> <div>要素</div> <div>要素</div>
inline-block	<div>要素</div> <div>要素</div> <div>要素</div>
none	非表示

まずは、イメージを理解する

- **block**⇒要素が横いっぱいになり「縦」に並ぶ
- **inline**⇒要素が「平ぺったく横」に並ぶ
- **inline-block**⇒blockとinlineの中間
- **none**⇒非表示となる

次へ

# display プロパティの違いを確認

- inline

Try  inline.html

- block

Try  block.html

- none

Try  none.html

- inline-block

Try  Inline-block.html

- flex

Try  flex.html

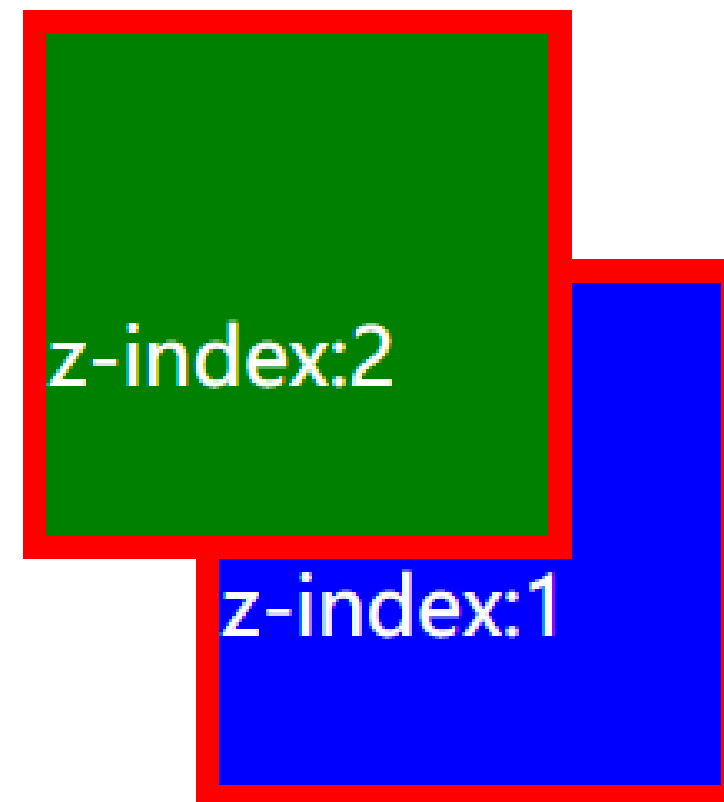
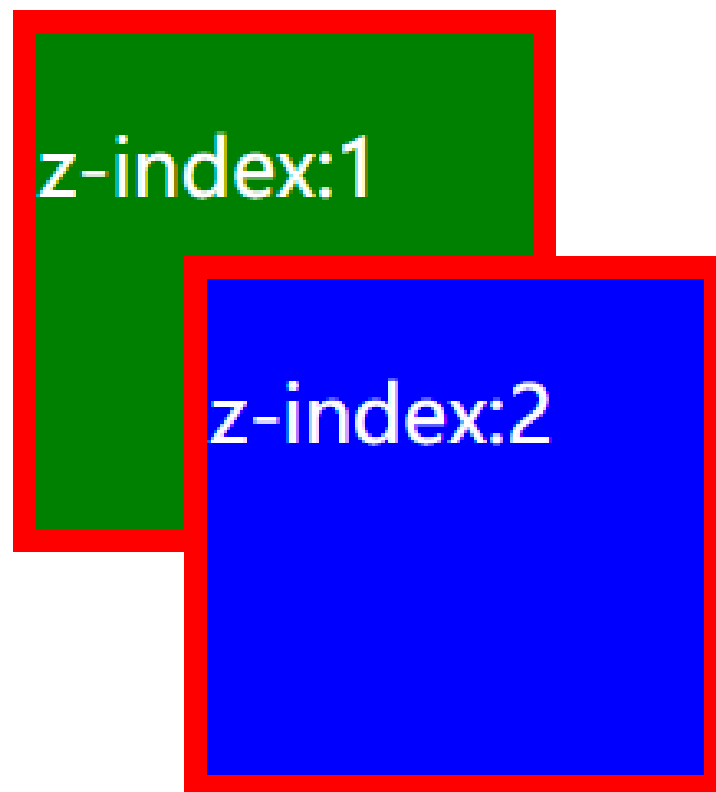
# transform プロパティ

- **transform** プロパティは、移動、拡大縮小、回転、せん断など、要素の様々な**変形**を設定できます。
- 特別な小技として、top と left のプロパティと組み合わせ、水平や垂直方向で**中央に並べる**ことができます：

```
position: relative;  
top: 50%;  
left: 50%;  
transform: translate(-50%, -50%);
```

# Z-index


- **z-index** プロパティは、要素の **z 軸の座標**を指定します。  
z 軸の座標が大きいほど、その要素はより「高く」なります（低い要素を覆います）：



# レスポンスなレイアウト

- CSS では、**@media** という特殊な構文で、レスポンスデザインに対応できます。特定の CSS スタイルを、画面サイズが特定の条件を満たしたときのみ動作するように設定することで、異なるサイズのデバイスに動的に対応できます。
- 例えば、`<p>` に対する以下のスタイル宣言は、画面サイズが 576 ピクセルより大きい場合のみ有効になります：

```
1 @media (min-width: 576px) {
2   p {
3     color: red;
4   }
5 }
```

Try   
responsive.html

- 実際の開発では、サイズの異なる複数のデバイスに対応するため、複数の @media ルールを組み合わせで利用します。

# Q&A



# まとめ

Sum Up



- 1.CSS を HTML に適用する 3 つの方法。
- 2.CSS の基本構文：セレクターとプロパティ。
- 3.CSS の常用プロパティ：
  - ① テキスト関連プロパティ、
  - ② 背景関連プロパティ、
  - ③ ボックスモデル関連プロパティ、
  - ④ レイアウト関連プロパティ。



*Light in Your Career.*

**THANK YOU!**