

# Web プログラミング

第四回 : CSS(1)

# CSS(1) - CSS の導入

css（Cascading Style Sheets）は、  
Web ページの外見を定義するコードです。

```
p {  
  color: red;  
}
```

# CSS(1) - CSS の導入

head タグの中に **style タグ** を書き、  
その中に **css** を記述することができます。

```
<head>  
  <meta charset="UTF-8" />  
  <title> 初めての HTML</title>  
  <style>  
    p {  
      color: red;  
    }  
  </style>  
</head>
```

## CSS(1) - CSS の導入

**セレクタ** : スタイルを適用する対象の HTML 要素を指定

**プロパティ** : 変更する見た目の種類を指定

**値** : 具体的な見た目を指定

```
p {  
  color: red;  
}
```

||

p 要素の色 (color) を赤 (red) にする

# CSS(1) - CSS の導入

色々書いてみましょう

```
body{
  background-color: black;
}
p{
  color: red;
  background-color: white;
  font-weight: bold;
  font-size: 24px;
  text-align: center;
  width: 500px;
}
```

# CSS(1) - CSS の導入

css ファイルを別途作成して、  
それを読み込むこともできます。

index.html

```
<head>
  <meta charset="UTF-8" />
  <title> 初めての HTML</title>
  <link rel="stylesheet" href="style.css">
</head>
```

style.css

```
body{
  background-color: black;
}
p{
  color: red;
  background-color: white;
  font-weight: bold;
  font-size: 24px;
  text-align: center;
  width: 500px;
}
```

## CSS(1) - CSS の導入

html タグに **class 属性**をつけることで、**セレクタ**を更に絞り込むことができます。

index.html

```
<p> 普通のテキスト </p>  
<p class="red"> 赤いテキスト </p>  
<p class="blue"> 青いテキスト </p>
```

style.css

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

## CSS(1) - CSS の導入

セレクタは、**class** 名だけで指定することもできます。

index.html

```
<h1 class="red"> 赤い見出し </h1>  
<p class="red"> 赤いテキスト </p>
```

style.css

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



## CSS(1) - CSS の導入

**セレクタ**は、入れ子構造を指定することもできます。

index.html

```
<p><strong> 赤い強調文字 </strong></p>  
<strong> 普通の強調文字 </strong>
```

style.css

```
p strong{  
  color: red;  
}
```

# CSS(1) - CSS の導入

## 練習問題

「css セレクタ練習」の HTML を編集して、  
青いリスト項目の中のリンクのみ赤くする css を書いてみよう

# CSS(1) - ブロックとインライン

HTML 要素には  
「ブロック」要素と  
「インライン」要素があります。

# CSS(1) - ブロック要素について

ブロック要素は、コンテンツの「かたまり」を定義するものが多いです。  
そのため、ブロック要素は縦方向に並びます。

index.html

```
<p>p 要素はブロックレベル要素です </p>  
<p> ブロックレベル要素は縦に並びます </p>
```

ブラウザでの表示例

p 要素はブロックレベル要素です  
ブロックレベル要素は縦に並びます

# CSS(1) - ブロック要素について

ブロック要素は、以下のような特徴があります。

- ・「かたまり」を定義するので、縦方向にならぶ
  - ・ 幅と高さを持つことができる
  - ・ 幅は、暗黙的に親要素の 100% になる

## 代表的なブロック要素

`<h1>~<h6>`、`<p>`、`<ol>`、`<ul>`、`<li>`、`<div>` など

# CSS(1) - インライン要素について

インライン要素は、主にテキストの一部に意味付けするものが多いです。  
そのため、前後にそのままコンテンツが続き、改行などは入りません。

index.html

代表的なインライン要素には `<em>` 強調 `</em>` や `<a href="">` リンク `</a>` などがあります。

ブラウザでの表示例

代表的なインライン要素には 強調 や [リンク](#) などがあります。

# CSS(1) - インライン要素について

インライン要素は、以下のような特徴があります。

- ・「一部の文章の意味」を定義するので、横方向にならぶ
  - ・ 幅と高さを持つことができない
- ・ 原則として、ブロック要素を子要素に入れることができない（例外あり）

## 代表的なインライン要素

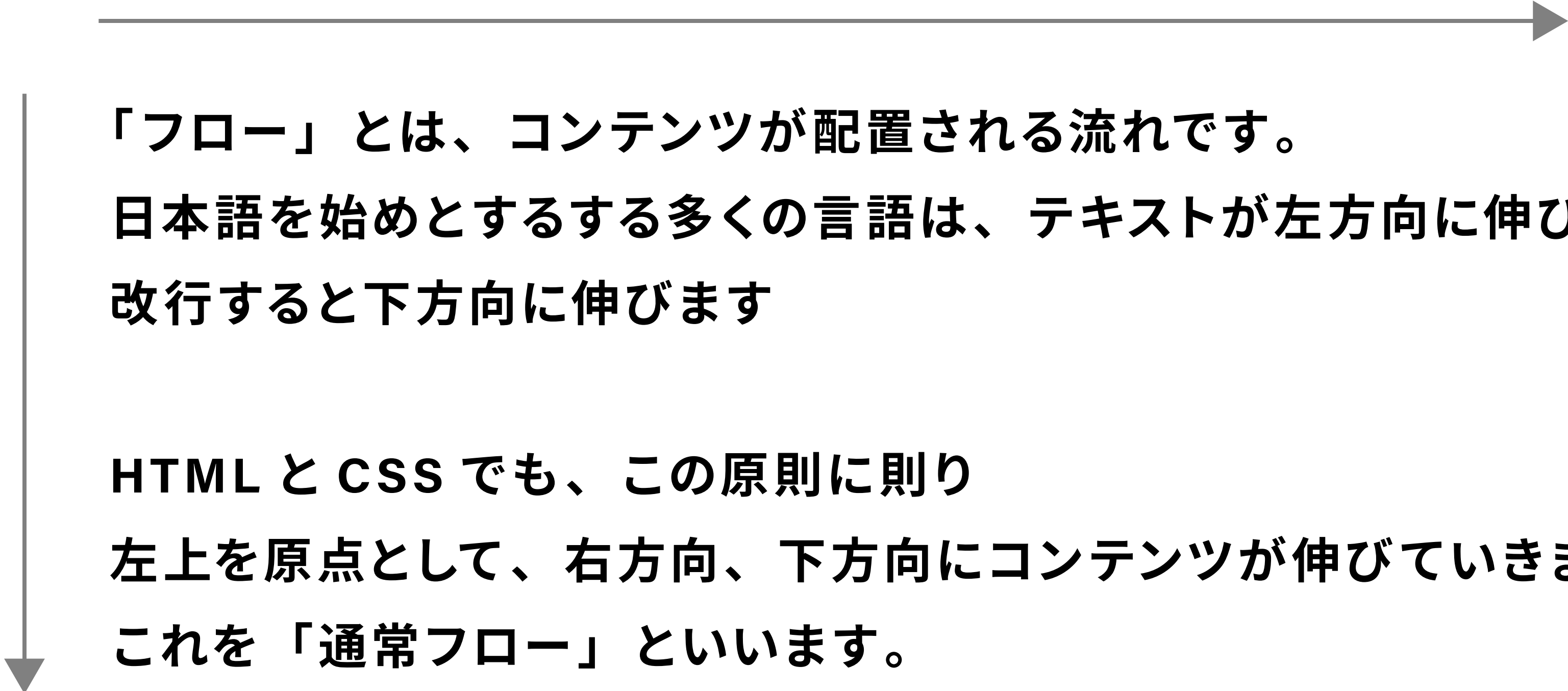
`<a>`、`<strong>`、`<em>`、`<img>`、`<br>`、`<span>` など

# CSS(1) - ブロックとインライン

**インライン要素は「横方向」に並ぶ**  
**ブロック要素は「縦方向」に並ぶ**  
という原則を覚えておきましょう。



## CSS(1) - フローについて

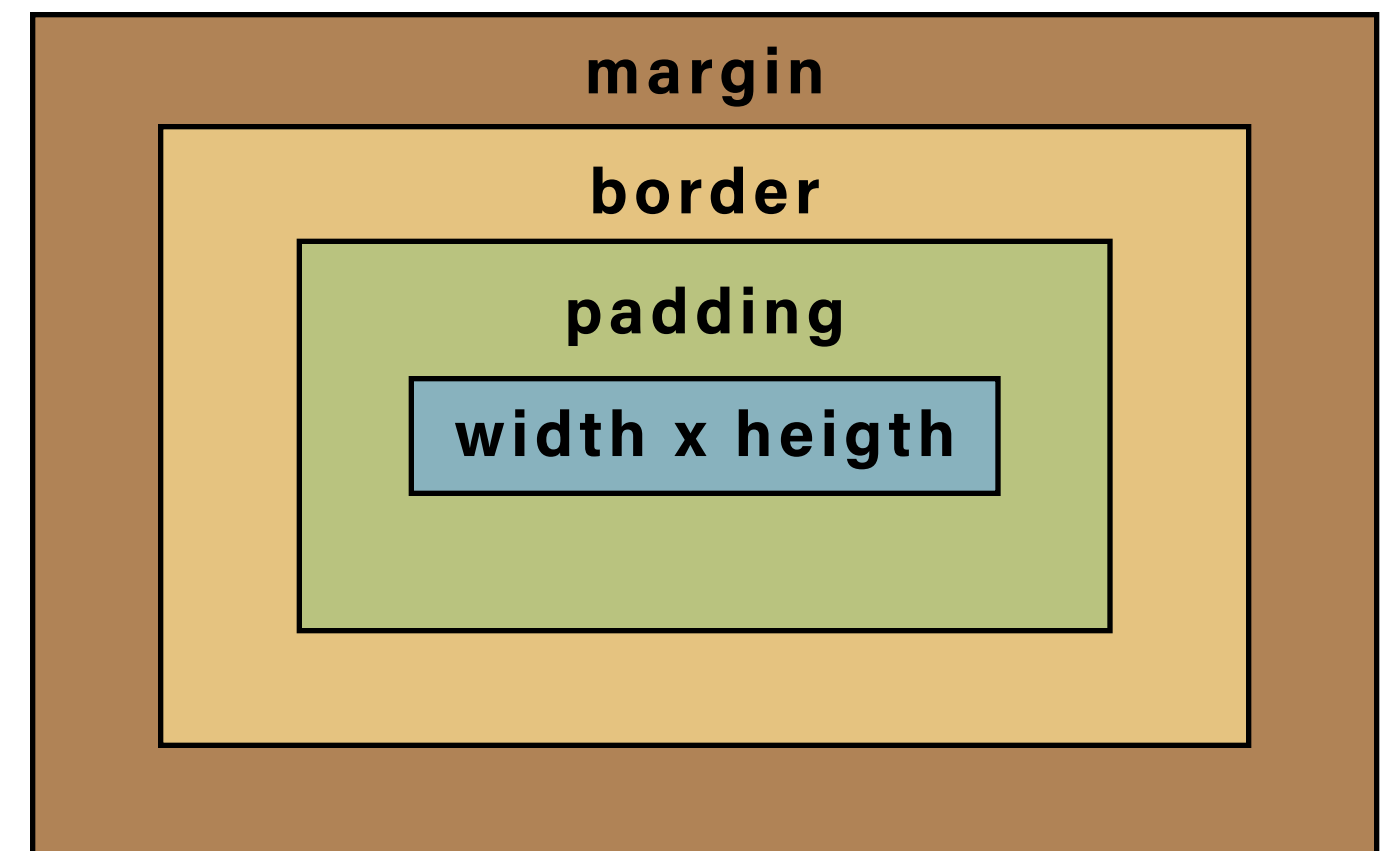


**「フロー」とは、コンテンツが配置される流れです。**  
**日本語を始めとする多くの言語は、テキストが左方向に伸び**  
**改行すると下方向に伸びます**

**HTML と CSS でも、この原則に則り**  
**左上を原点として、右方向、下方向にコンテンツが伸びていきます。**  
**これを「通常フロー」といいます。**

# CSS(1) - ボックスモデルについて

「ボックスモデル」は、ブロック要素の幅や余白などを定義する時の基本です



# CSS(1) - 幅と高さ

width と height を指定して、ブロックの幅と高さを指定できる

```
p {  
  width: 400px;  
  height: 200px;  
}
```

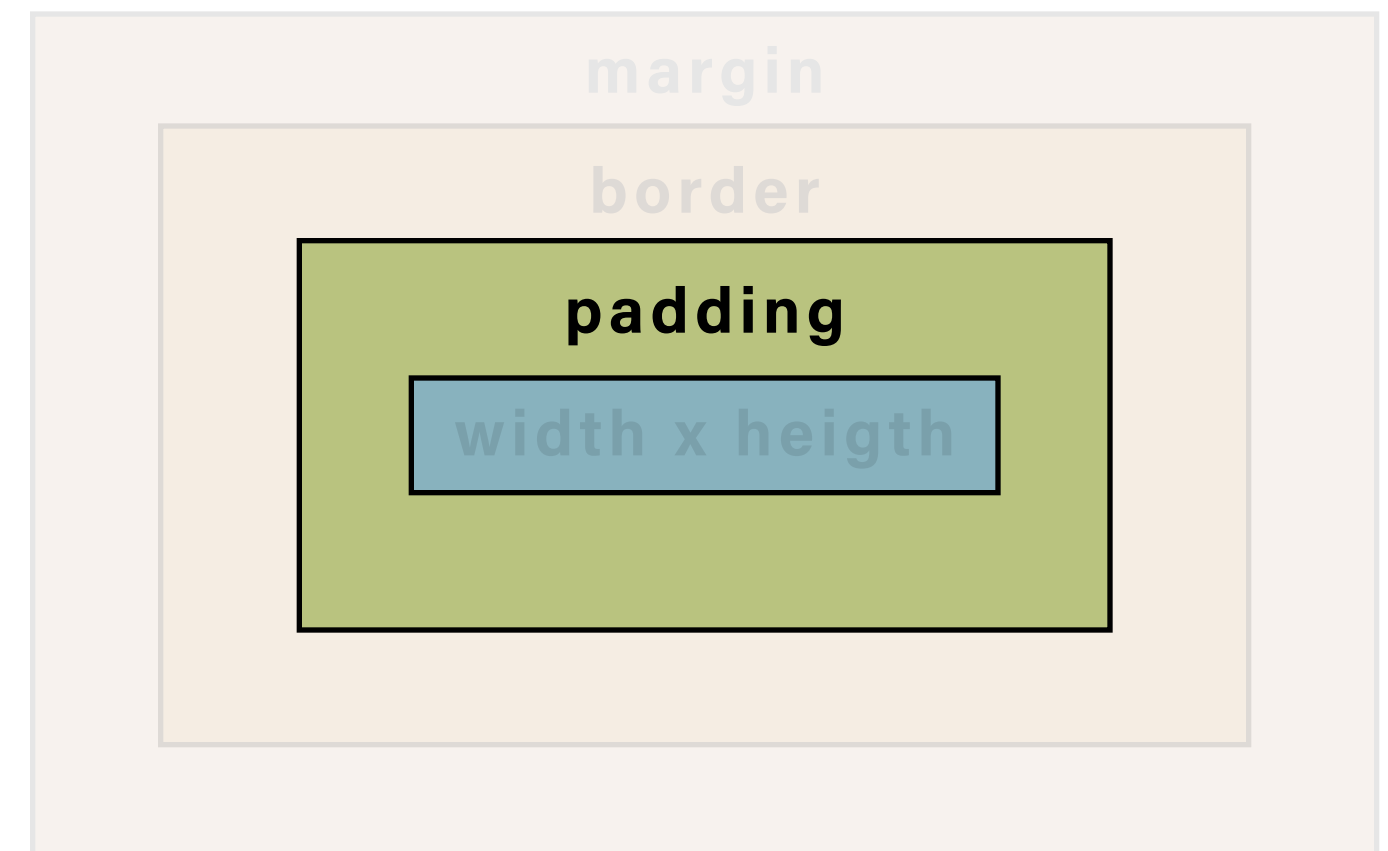


# CSS(1) - パディング

padding を指定して、ブロックの内側の余白を指定できる

```
p {  
  padding-top: 10px;  
  padding-right: 20px;  
  padding-bottom: 30px;  
  padding-left: 40px;  
}
```

```
p {  
  padding: 10px 20px 30px 40px;  
}
```

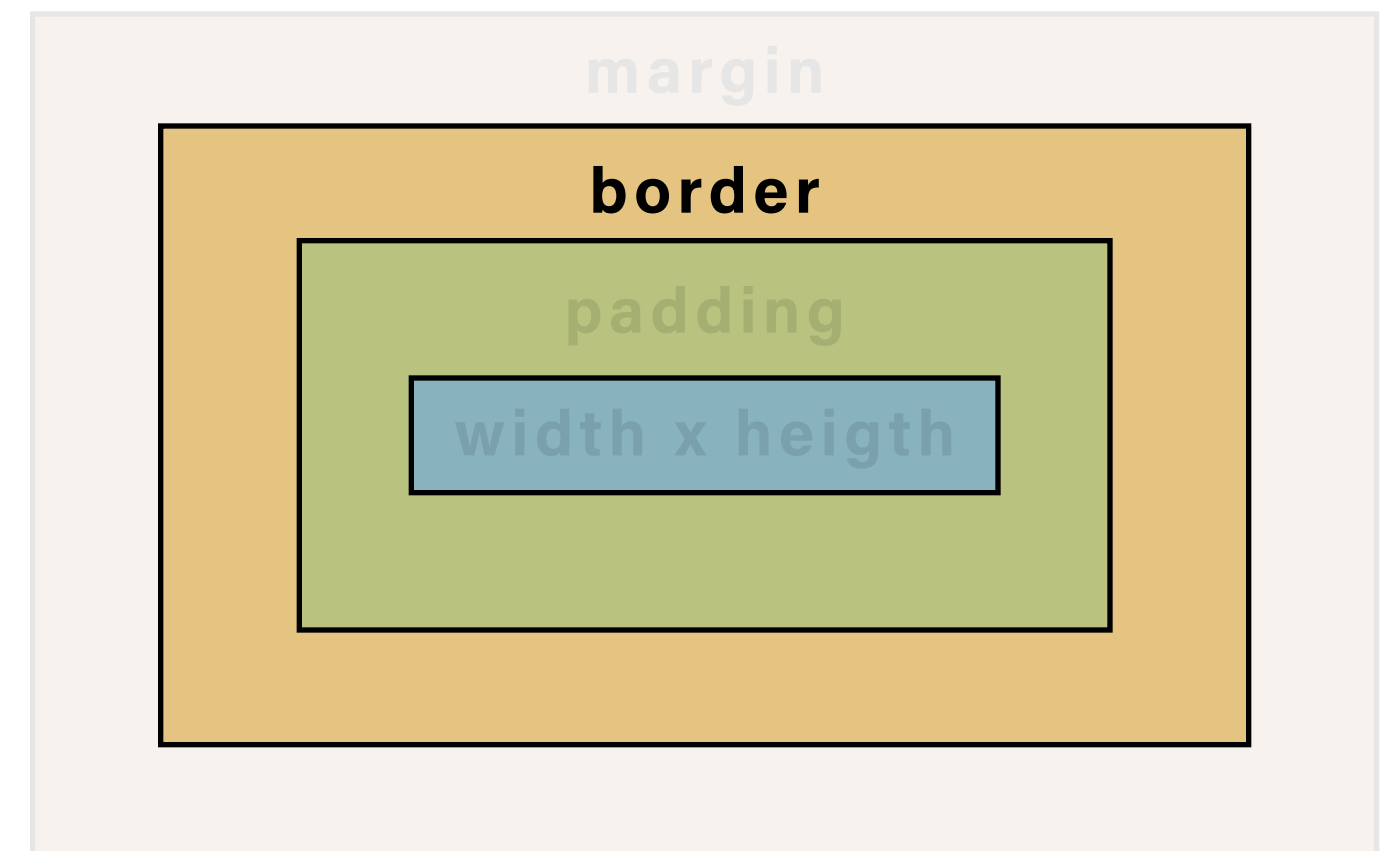


# CSS(1) - ボーダー

**border** を指定して、ブロックに枠線を付与できる

```
p {  
  border: 1px solid #000;  
}
```

```
p {  
  border-top: 1px solid #000;  
  border-right: 1px solid #000;  
  border-bottom: 1px solid #000;  
  border-left: 1px solid #000;  
}
```

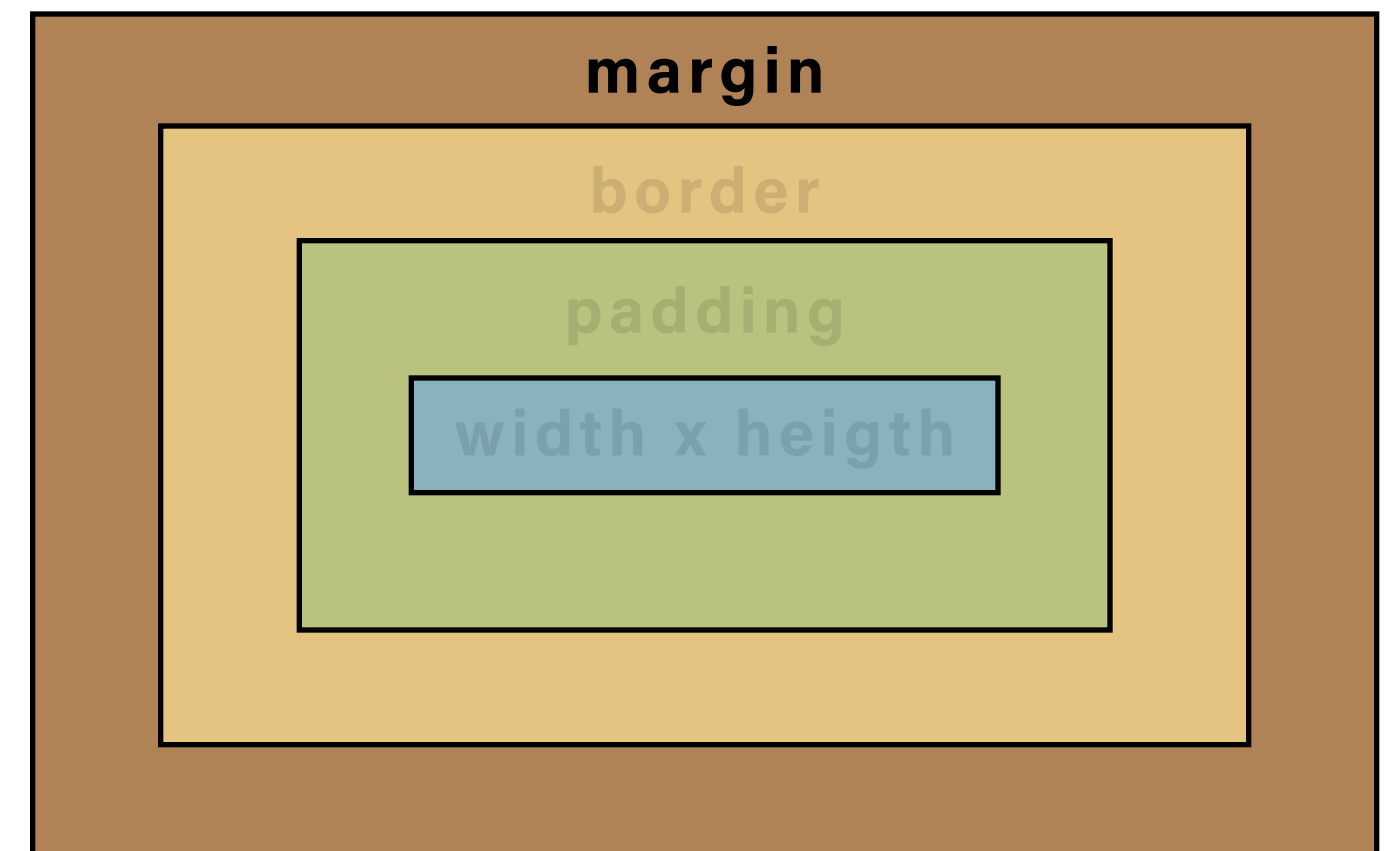


# CSS(1) - マージン

**margin** を指定して、ブロック外側の余白（他要素とのスキマ）を指定できる

```
p {  
  margin-top: 10px;  
  margin-right: 20px;  
  margin-bottom: 30px;  
  margin-left: 40px;  
}
```

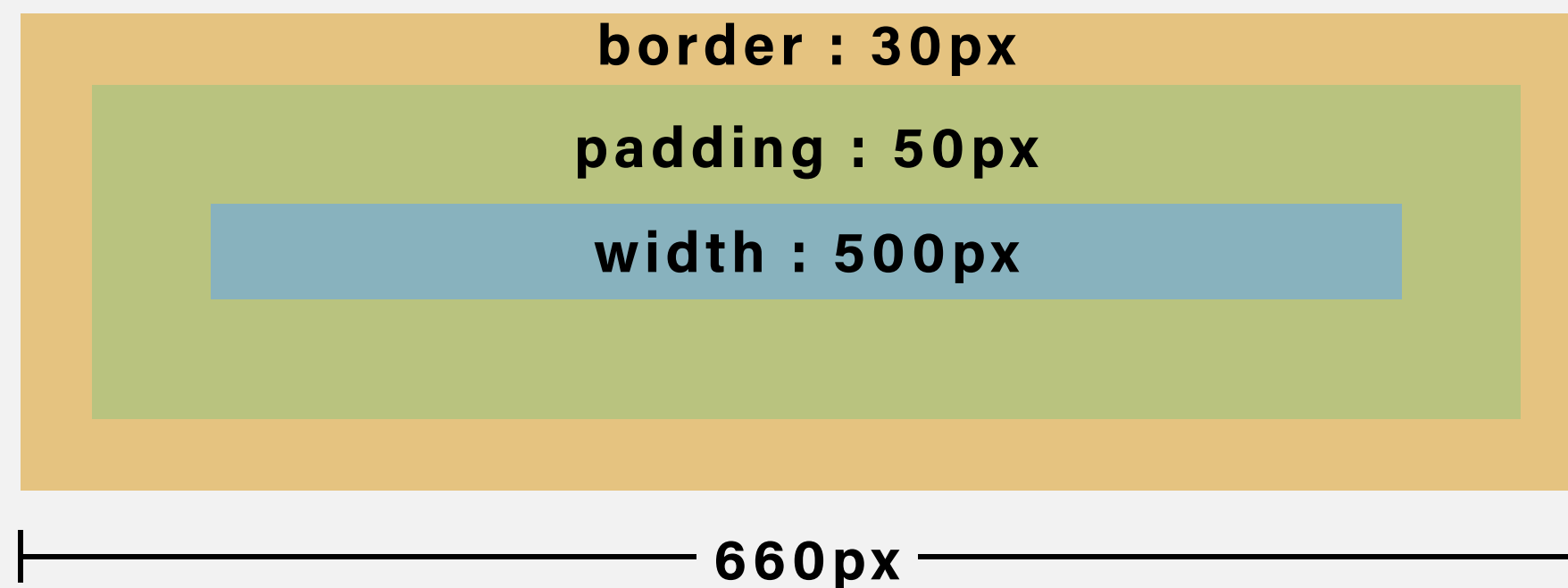
```
p {  
  margin: 10px 20px 30px 40px;  
}
```



## CSS(1) - ボックスのサイズ計算

最終的なボックスの幅と高さは、  
`width/height` + `padding` + `border` のサイズになる

例 : `width:500px` 、 `padding:50px` 、 `border: 30px` の場合  
最終的な幅は 660px になる



`margin` は、他要素とのスキマなので、ボックスの幅と高さには含まない

## CSS(1) - ボックスのサイズ計算

css で `box-sizing: border-box` を指定すれば、  
`width/height` に `padding` と `border` を含めたサイズにできる

```
p {  
    box-sizing: border-box;  
}
```

例 : `width:500px` 、 `padding:50px` 、 `border: 30px` 、 `box-sizing:border-box` の場合  
最終的な幅は 500px になり、コンテンツエリアの幅は 340px になる





# CSS(1) - インライン要素におけるボックスモデル

インライン要素では、ボックスモデルの扱いは特殊になる

- width と height は指定しても無視される
- padding は指定できるが、上下の padding はレイアウトに影響しない
- border は指定できる
- margin は左右のみ有効で、上下は無視される

インライン要素では、幅や高さなどは基本考えない方が無難

## CSS(1) - インライン要素をブロックレベルにする

a 要素など、インライン要素をブロック要素のように扱いたいときがある。

css で `display: block` を設定すると、ブロック要素のように取り扱えるようにある。

```
a {  
    display: block;  
}
```

逆に、ブロック要素をインライン要素のようにすることもできる

```
p {  
    display: inline;  
}
```

# CSS(1) - 練習問題

## 練習問題

a 要素を「ボタン」にしてみよう



- 幅 300px くらい
- 上下パディング 15px くらい
- 枠線 2px、黒色（#000000）
- 文字色は白色（#FFFFFF）
- フォントサイズは 24px くらい
- テキスト中央揃え
- 背景色グレー（#B3B3B3）