

# Web講習会2021 ワールドワイドウェブ基礎

第4回: CSS (2)



Arthur

# この回の目標

- CSSの外部ファイル読み込みの方法を知る
- ボックスの概念について理解する
- レスポンシブデザインを実現する仕様について理解する

今回の説明で使用するコードはpr2.zipとして配布します

# [再掲] 次回予告

CSS編はまだまだ続きます 😞

- CSSの外部ファイル化
- CSSのプロパティ(2) ボックス
- 疑似クラス、疑似要素
- レスポンシブデザイン

# CSSの読み込み方法

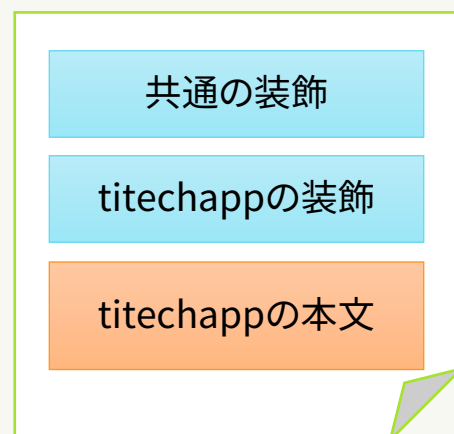
---

# 複数ページはコピペ地獄？

複数ページに共通の装飾をつけたいとき



index.html



titechapp.html



titechinfo.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>titechinfo</title>
    <style>
      /* 共通のCSS */
    </style>
    <style>
      /* titechinfoのCSS */
    </style>
  </head>
  <body>
    <p>titechinfo</p>
  </body>
</html>
```



共通のCSSを各ページに  
コピペする？

# CSSの外部ファイル化

CSS(<style>の中身)を外部ファイルに書き出し、各ページで読み込むことができる

link要素 … 外部リソースへのリンク

href: パス

rel: 参照先がどのような関係か (relation)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>linktest</title>
    <link href="hello.css" rel="stylesheet">
  </head>
  <body>
    <p>hello!link!</p>
  </body>
</html>
```

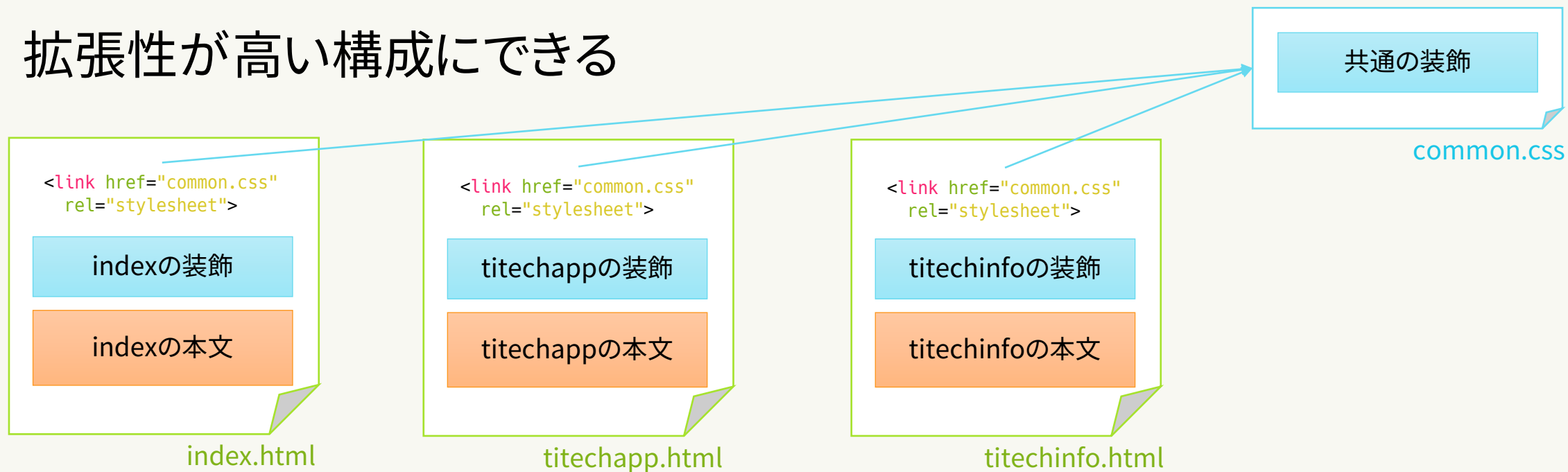
linktest.html

```
/* <style>タグは不要 */
p {
  color: red;
}
```

hello.css

# コピペ地獄からの脱却

拡張性が高い構成にできる



# インラインスタイル

## グローバル属性: **style**

HTMLの要素の属性として、ある要素だけへのスタイルを定義できる

インラインスタイルの詳細度は、idセレクトタ(100)よりも高い(1000)

ただし、!importantのほうが優先

```
/* 詳細度100 */  
#hello {  
  color: blue;  
}
```

```
<p id="hello" style="color: red;">helloinline!</p>
```

セレクトタ不要

※HTMLに文書構造と装飾が混在する書き方になるので、推奨しない

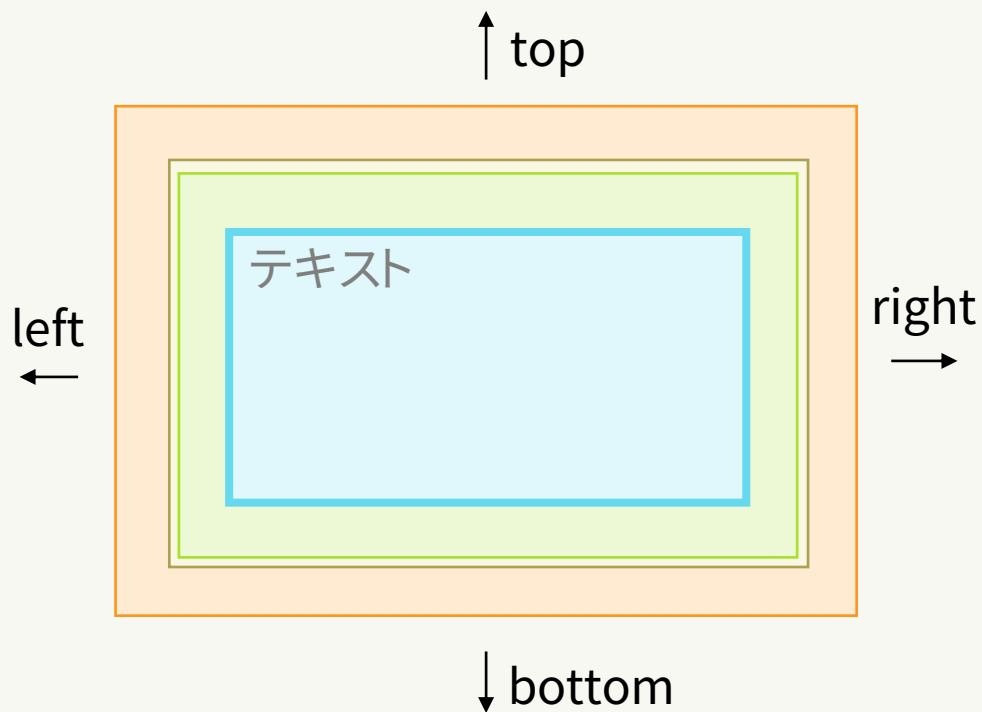


# CSSのプロパティ(2)

ボックス

# ボックス

ボックス…要素の描画領域



**content:** テキストや画像などの内容を表示する

**padding:** 余白(背景が塗られる)

**border:** 境界線

**margin:** 余白(背景が塗られない)

※説明を簡潔にするため、以降しばらくはブロックレベル要素のボックスに限って説明する

# width・height

**width:** contentの幅を定める

デフォルト値は100% (親要素のwidthと同じ)

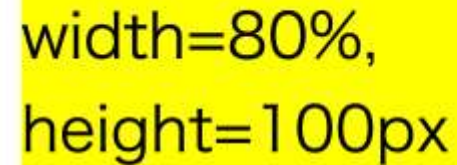
html要素のデフォルト幅は画面幅と同じ

**height:** contentの高さを定める

デフォルト値はauto (中のテキストや要素によって可変)

```
#box {  
  width: 80%; /* 相対指定 */  
  height: 200px; /* 絶対指定 */  
  background: yellow;  
}
```

```
<div id="box">width, height</div>
```



width=80%,  
height=100px

# margin: ボックスの外側の余白

**margin-(方向):** 方向の向きに余白を取る

ex) margin-top: 30px;

marginの部分にbackgroundは適用されない

```
#question {  
  margin-bottom: 30px;  
  background: yellow;  
}
```

```
<div id="question">How are you?</div>  
<div id="answer">I'm fine.</div>
```

How are you?

I'm fine.

# クイズ①

#boxAと#boxBの距離は？

```
.box {  
  background: yellow;  
}  
#boxA {  
  margin-bottom: 30px;  
}  
#boxB {  
  margin-top: 50px;  
}
```

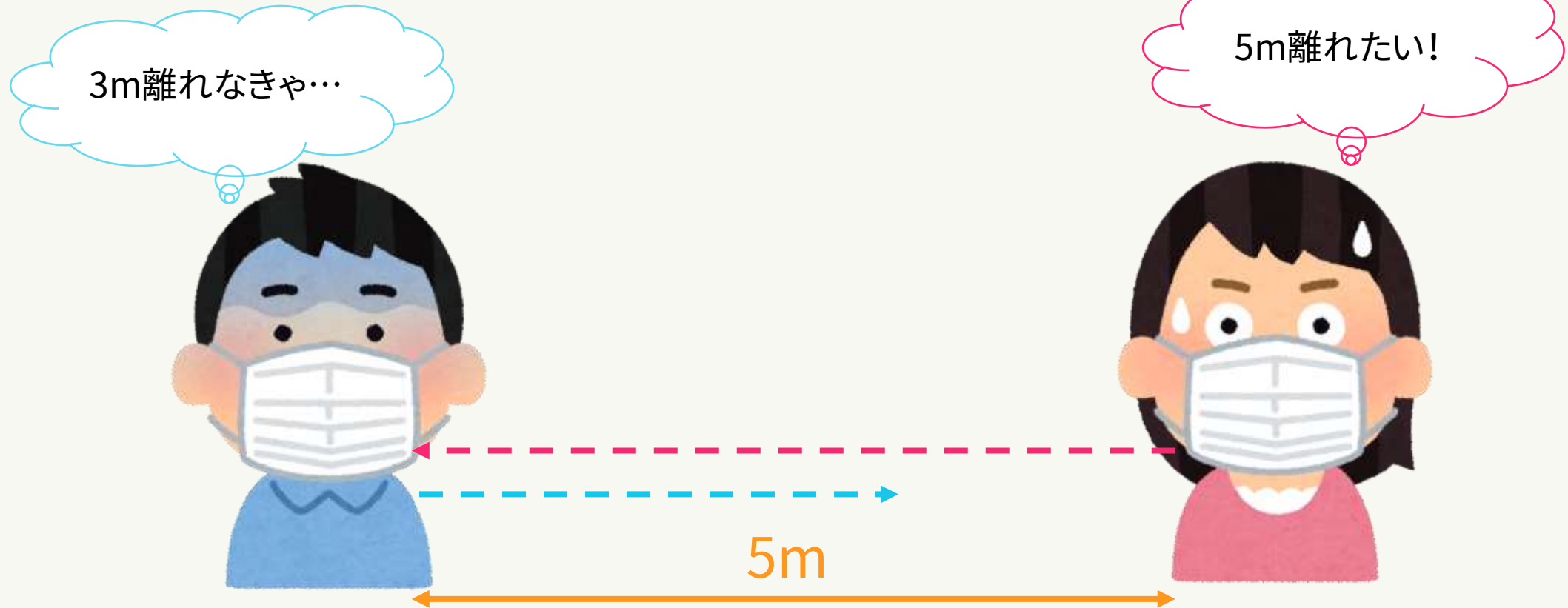
```
<div id="boxA" class="box">boxA</div>  
<div id="boxB" class="box">boxB</div>
```

boxA

boxB

# marginのイメージ

## ソーシャルディスタンス



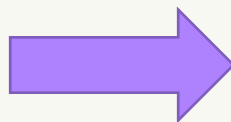
# ショートハンド

同じ種類のプロパティをまとめて指定できるalias

今まで使ってきたbackgroundも実はショートハンド

background-color, background-repeat, etc

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



```
#box {  
  margin: 10px 20px 30px 40px;  
}
```

# ショートハンドの順序

複数の方向に対して一度に指定するショートハンドでは、値の順番が決まっている

○値1つの場合

`(top/bottom/left/right);`

○値2つの場合

`(top/bottom) (left/right);`

○値3つの場合

`(top) (left/right) (bottom);`

○値4つの場合

`(top) (right) (bottom) (left);`



# padding: ボックスの内側の余白

**padding-(方向):** 方向の向きに内側の余白を取る

ex) padding-top: 30px;

paddingの部分にbackgroundは適用される

```
#question {  
  padding-bottom: 30px;  
  background: yellow;  
}
```

```
<div id="question">How are you?</div>  
<div id="answer">I'm fine.</div>
```



How are you?

I'm fine.

## クイズ②

#boxAと#boxBのcontent間の距離は？

```
#boxA {  
  background: yellow;  
  padding-bottom: 30px;  
}  
#boxB {  
  background: pink;  
  padding-top: 50px;  
}
```

```
<div id="boxA">boxA</div>  
<div id="boxB">boxB</div>
```



# paddingのイメージ

人体と人体の距離を考える  
服の内側の空間は交わらない



# border: ボックスの枠線

**border-(方向)-style:** 線の種類

solid 実線、double 二重線、dashed 破線

**border-(方向)-width:** 線の太さ

**border-(方向)-color:** 線の色

```
#title {  
  border-bottom: solid 1px black;  
}
```

```
<div id="title">Hello, world.</div>
```

Hello, world.

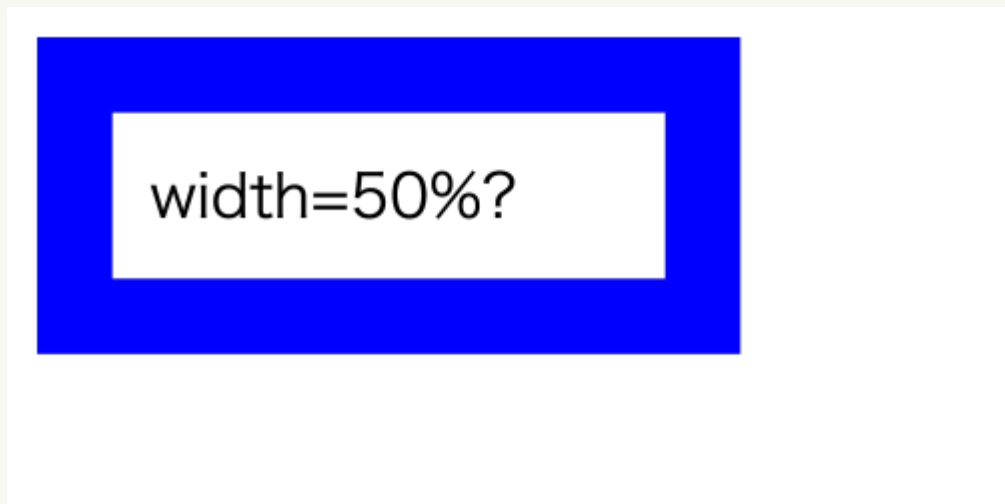
---

# ボックスの膨らみ

widthやheightで指定するのはcontentのサイズ  
paddingやborderの分だけ見た目が膨らむ

```
#box {  
  width: 50%;  
  padding: 10px;  
  border: solid 20px blue;  
}
```

```
<div id="box">width=50%?</div>
```

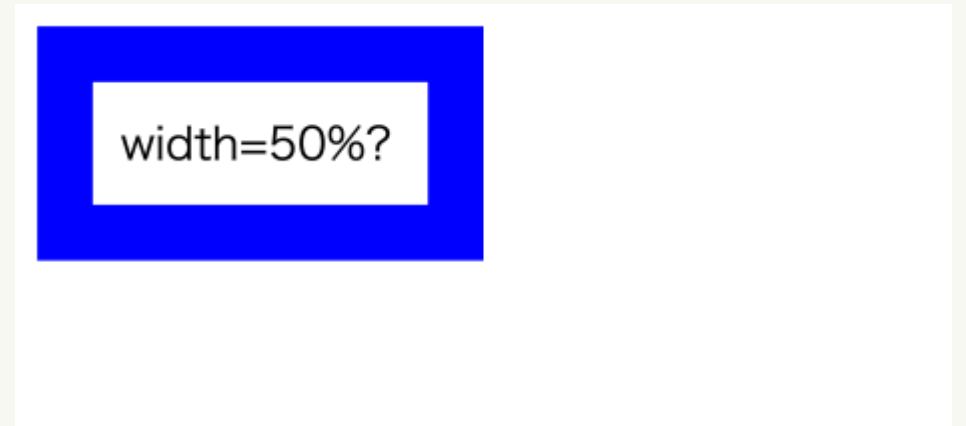


# 対処法1: calc()で計算

paddingとborderの分だけwidthの数字を減らす  
絶対数値と相対数値の間の計算をcalc()で行う

```
#box {  
  width: calc(50% - 10px * 2 - 20px * 2);  
  padding: 10px;  
  border: solid 20px blue;  
}
```

```
<div id="box">width=50%?</div>
```

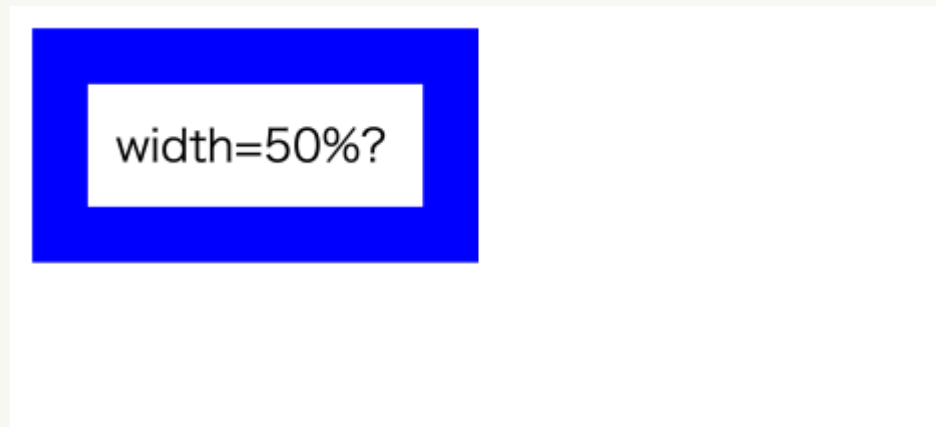


# 対処法2: box-sizing

**box-sizing**: ボックスのサイズを計算する基準を決める  
content-box(デフォルト値) → widthやheightはcontentのサイズ  
border-box → widthやheightはborderまでを含めたサイズ

```
#box {  
  width: 50%;  
  padding: 10px;  
  border: solid 20px blue;  
  box-sizing: border-box;  
}
```

```
<div id="box">width=50%?</div>
```



# [再掲] 歴史的な要素の区別

HTML4.01以前に存在した、かつての要素の分類

## ブロックレベル要素

文書を構成するひとかたまり。改行が入る

ex) p, h1, ul, ol, dl, table

## インライン要素

ブロックレベル要素の中身。改行が入らない

ex) img, a

現在でも、ブラウザのデフォルトの描画にはこれらの名残が見られる



# ボックスの種類

ブロックレベル要素とインライン要素のレイアウトの違いは、ブラウザのデフォルトスタイルの`display`プロパティによる

`display: block;`  
前後に改行が入る

`display: inline;`  
前後に改行が入らない。`width`や`height`が適用されない

# 応用的なセレクタ

疑似クラス・疑似要素

# 擬似クラス

擬似クラス … セレクタに付加して特定の状態を指す  
:から始まる

```
.hoverme {  
  color: blue;  
}  
  
.hoverme:hover {  
  color: red;  
}
```

```
<div class="hoverme">Hover me!</div>
```

:hover

カーソルがあたっているとき

# 擬似要素

擬似要素 … セレクタに付加して要素内の特定の部分を指す  
::から始まる

```
.headline {  
  color: blue;  
}  
  
.headline::first-letter {  
  color: red;  
}
```

```
<div class="headline">Headline</div>
```

::first-letter  
最初の文字

Headline

# レスポンスイブデザイン



# 端末の違いとブラウザ

パソコンとスマートフォンで画面幅など端末の性質が異なる

モバイル端末でも見やすいWebサイトにすることは必須

Googleは2018年にモバイルファーストインデックスを開始

# Yahoo! Japanの場合

PCとスマホで表示するサイトを分けている  
HTMLやCSSが異なる



# Titech App Project Webの場合

画面幅が変わると自動でレイアウトが変わる  
→レスポンスブデザイン





# レスポンシブデザインの実現

レスポンシブデザインを実現するには以下のような方法がある

- ボックスサイズをルート要素から相対指定していく
- メディアクエリの利用

スマホブラウザで意図するサイズになるように、以下のおまじないをheadに書く

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

# 相対指定の利用

html要素のデフォルトwidthは画面幅と同じである

その値を基準に相対指定することで、画面幅が変わっても比率が一定になる

html → body → hoge → … → poyoと相対指定で繋いでいく

```
img {  
  display: block; /* widthやheightを適用するためblockに */  
  width: 100%; /* 親要素の100%の幅 */  
  max-width: 600px; /* 大きすぎないように */  
  height: auto; /* アスペクト比を崩さず自動拡張 */  
  margin: auto; /* 左右中央寄せ */  
}
```

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

# メディアクエリの利用

## メディアクエリ

表示するメディア(の種類や画面幅など)を絞ってスタイルを宣言できる

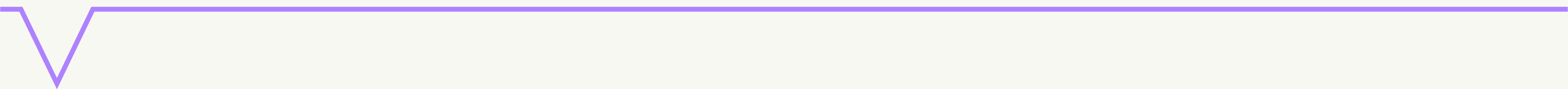
```
.hidden-sp, .hidden-pc {  
  display: block;  
}  
@media screen and (max-width: 599px) {  
  .hidden-sp {  
    display: none; /* 非表示 */  
  }  
}  
@media screen and (min-width: 600px) {  
  .hidden-pc {  
    display: none; /* 非表示 */  
  }  
}
```

```
<p class="hidden-pc">  
  この文章はスマホから閲覧できます。  
</p>  
<p class="hidden-sp">  
  この文章はPCから閲覧できます。  
</p>
```

# 講習会で扱わない大事な内容

- flexbox  
要素を横並びにするのにtable-cellやfloatを使っていたら時代遅れ
- positionプロパティ  
要素の位置を絶対指定することもできる。固定フッターなど
- transitionプロパティ  
かんたんなアニメーションに利用。プロパティ値の変化をなめらかに
- vw, vh  
長さの単位。安易な使用には注意が必要

# CSSライブラリ

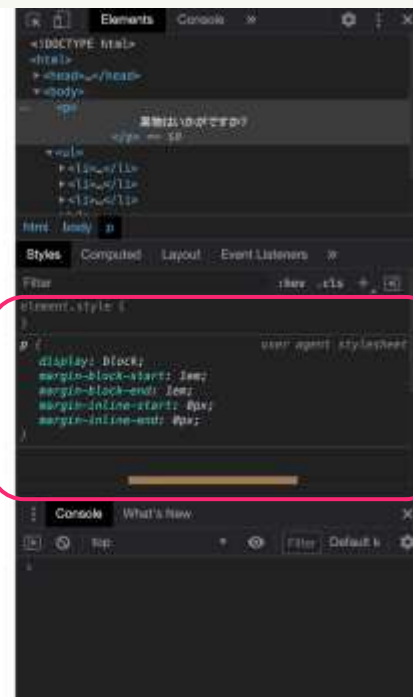


# デフォルトスタイル

ブラウザは様々な要素にデフォルトの装飾をつけている  
オリジナルのデザインでは、これが邪魔になることがある  
また、異なるブラウザ間で表示が変わってしまう原因になる

果物はいかがですか？

- オレンジ
- りんご
- ぶどう



# リセットCSS

## リセットCSS

デフォルトスタイルを無効化したり、ブラウザ間の差異をなくすようなスタイルシートが公開されている

- reset.css

昔からある本家リセットCSS。デフォルトスタイルをすべて打ち消す  
似ているもの: destyle.css, HTML5 Doctor Reset CSS

- Normalize.css

ブラウザ間の差異をなくすだけに留め、スタイルの再定義を防ぐ  
似ているもの: A Modern CSS Reset, ress.css, Reboot.css

# CSSフレームワーク

## CSSフレームワーク

Webページでよく使用されるコンポーネントのスタイルが定義されている  
公開されたCSSを読み込み、クラスを付与するだけで装飾できる

- Bootstrap

いちばん有名なフレームワーク

Arthurがこれまで制作したWebアプリのほとんどに利用している

- Tailwind CSS

1装飾ごとのレベルで細分化されたクラスを組み合わせで装飾

Bootstrapより自由度が高い分、コーディングに難あり



# 次回予告

## CSS編の最終回

- 良いマークアップとは
- CSSアーキテクチャ

また、HTTPについて説明します