

# Antenna House Formatter V7.4 紹介

大幅強化された

CSS 組版で簡単  
レイアウト

アンテナハウス株式会社  
横田達朗

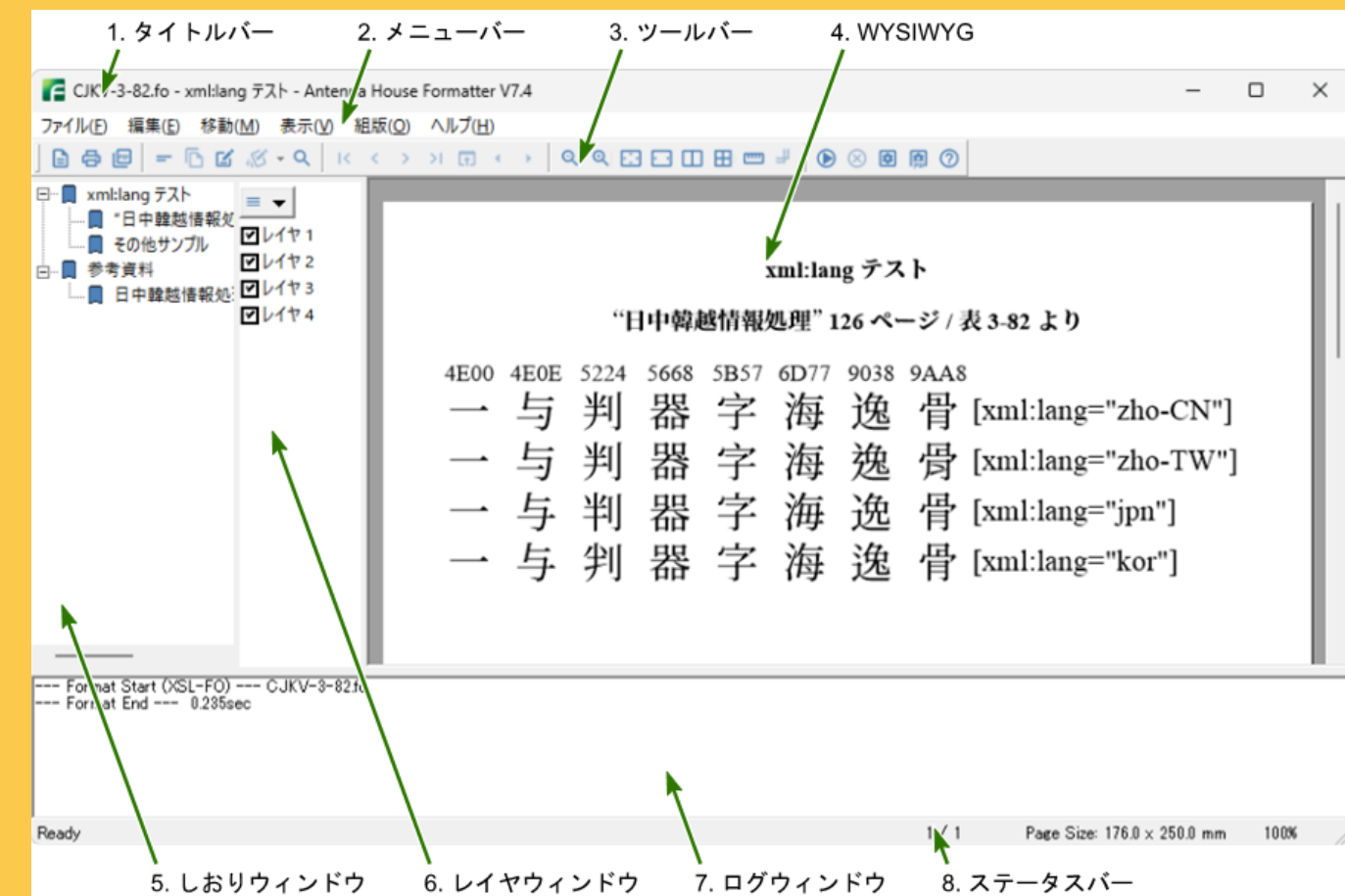
# Formatter

- Antenna House Formatter
- なぜ専用ソフトを使うのか？
- V7.4
- CSS 機能が強化されると.....

# Antenna House Formatter

ふたつのスタイルシート（XSL 1.1・CSS）に対応した自動組版・印刷ソフトウェア

- このスライドも HTML+CSS を Formatter でレイアウト
  - 過去ウェビナーでは XSL 製もある



# なぜ専用ソフトを使うのか？

流し込んでみないとどうなるか分からないものを美しく処理できる

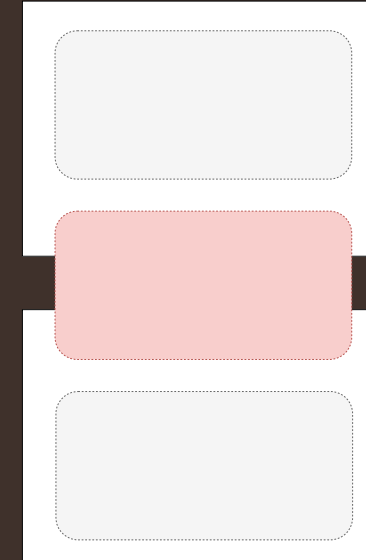
「このページは m 章だ  
(ヘッダ・フッタ)」

3節 Grid

**3節 Grid**

この節では……

「改ページの前（後）に  
この部分は収めたい」

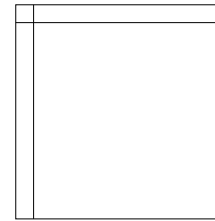


# なぜ専用ソフトを使うのか？

「.....についてはn ページを参照」

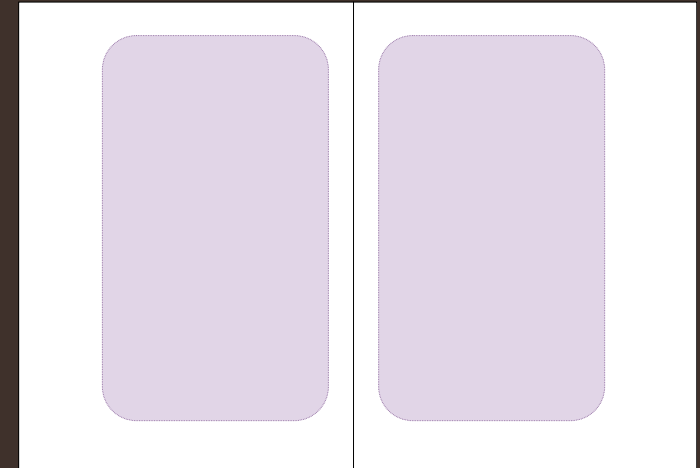
30ページの表を見  
てください

19



30

「左右のページで余白を  
変える」



# V7.4

V7.4 が 2023-12 にリリース

- 02/08 に改訂 1 版 (MR1)
- **CSS 機能が大幅強化**
- **数年来カスタマー要望のあった機能が実装**



# CSS 機能が強化されると.....

素朴で、でも難しかった要望のレイアウトが簡単に！

並べた図版がいい感じ  
に自動的に並んでほし  
い

特定位置に配置したい

目的の HTML 要素を指  
定するのが大変

- CSS 自体は慣れている → 慣れた最近の書き方ができる
- CSS 自体もあまり経験がない → 簡単になった最近の方法だけ学べばよい

# アジェンダ



# アジェンダ

1. Formatter
2. FlexibleBox と CSS Grid のレイアウト例
3. セレクタ指定を柔軟に
4. その他
5. まとめ

# FlexibleBox と CSS Grid のレイアウト ト例

- コンテナとアイテム
- Flexible Box
- CSS Grid

# 配置を揃えるコンテナとアイテム

コンテンツをコンテナとアイテムという構造に対応させて扱う

## コンテナ

アイテムを配置する箱

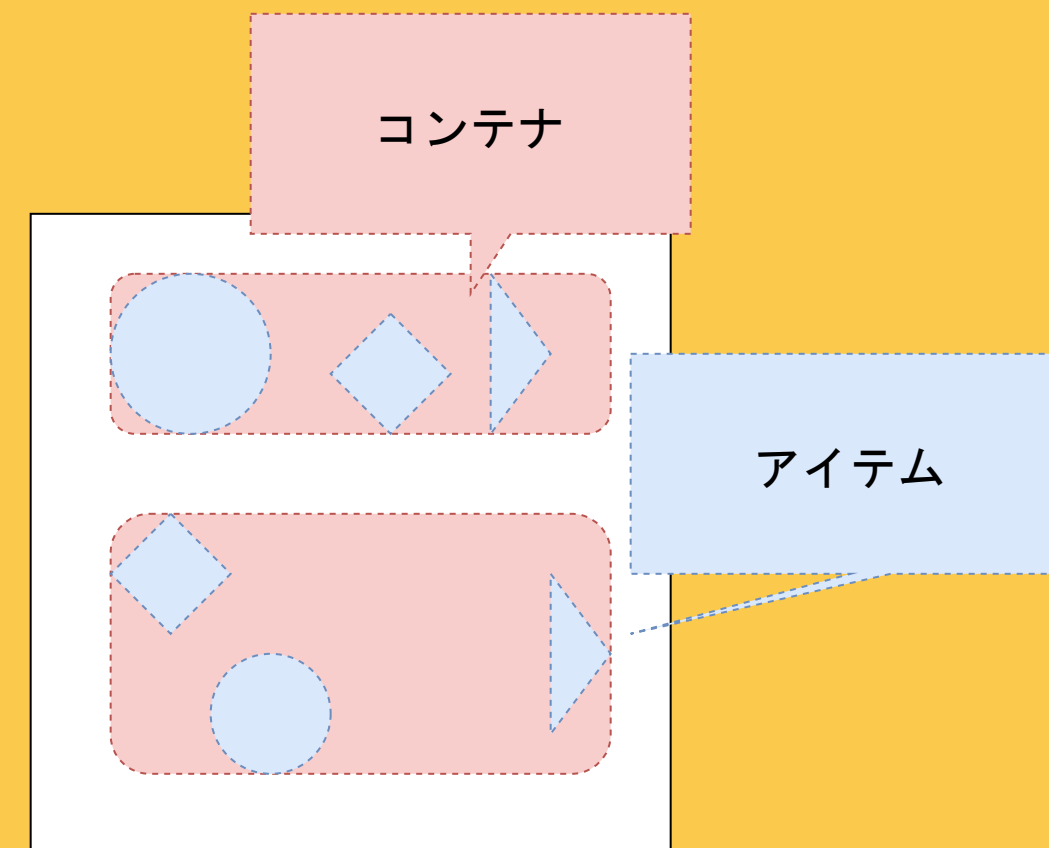
`display: flex,`

`display: grid`

(`inline-flex`, `inline-grid` は  
行内用)

## アイテム

コンテナ内に配置される箱



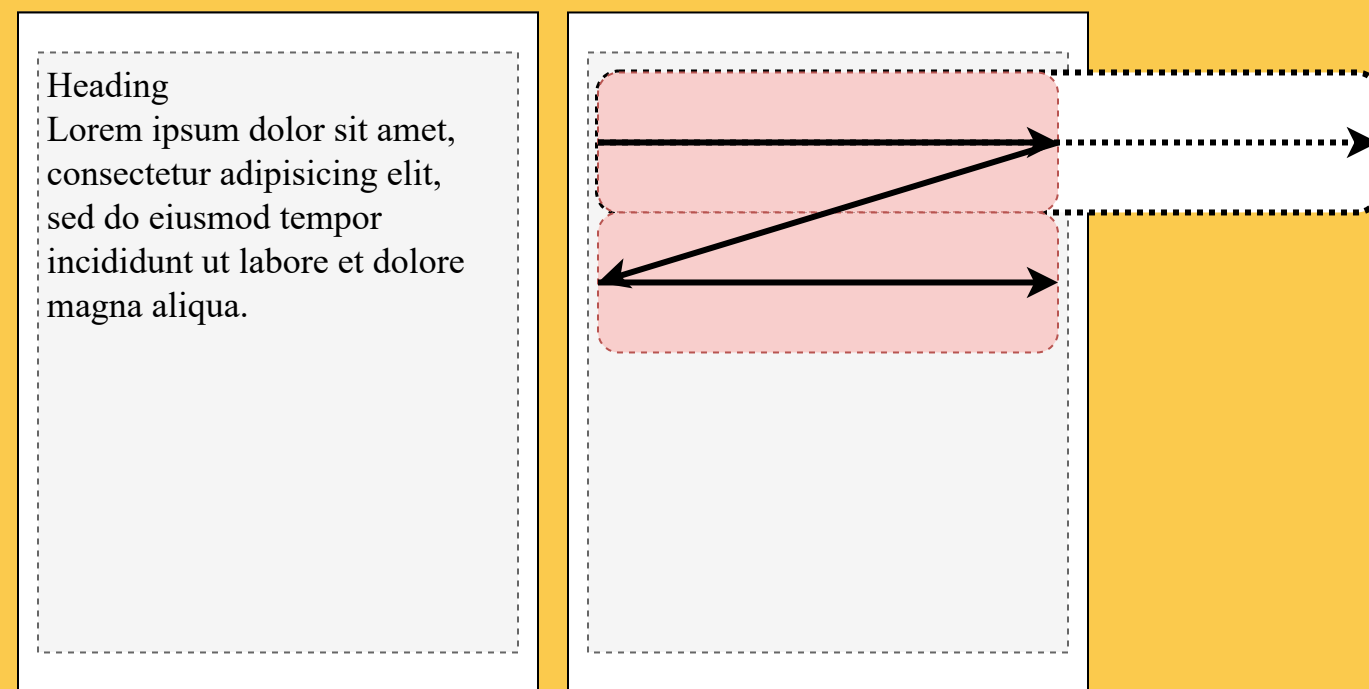
# 配置を揃えるコンテナとアイテム

```
(display: flex-item,  
display: grid-item)
```

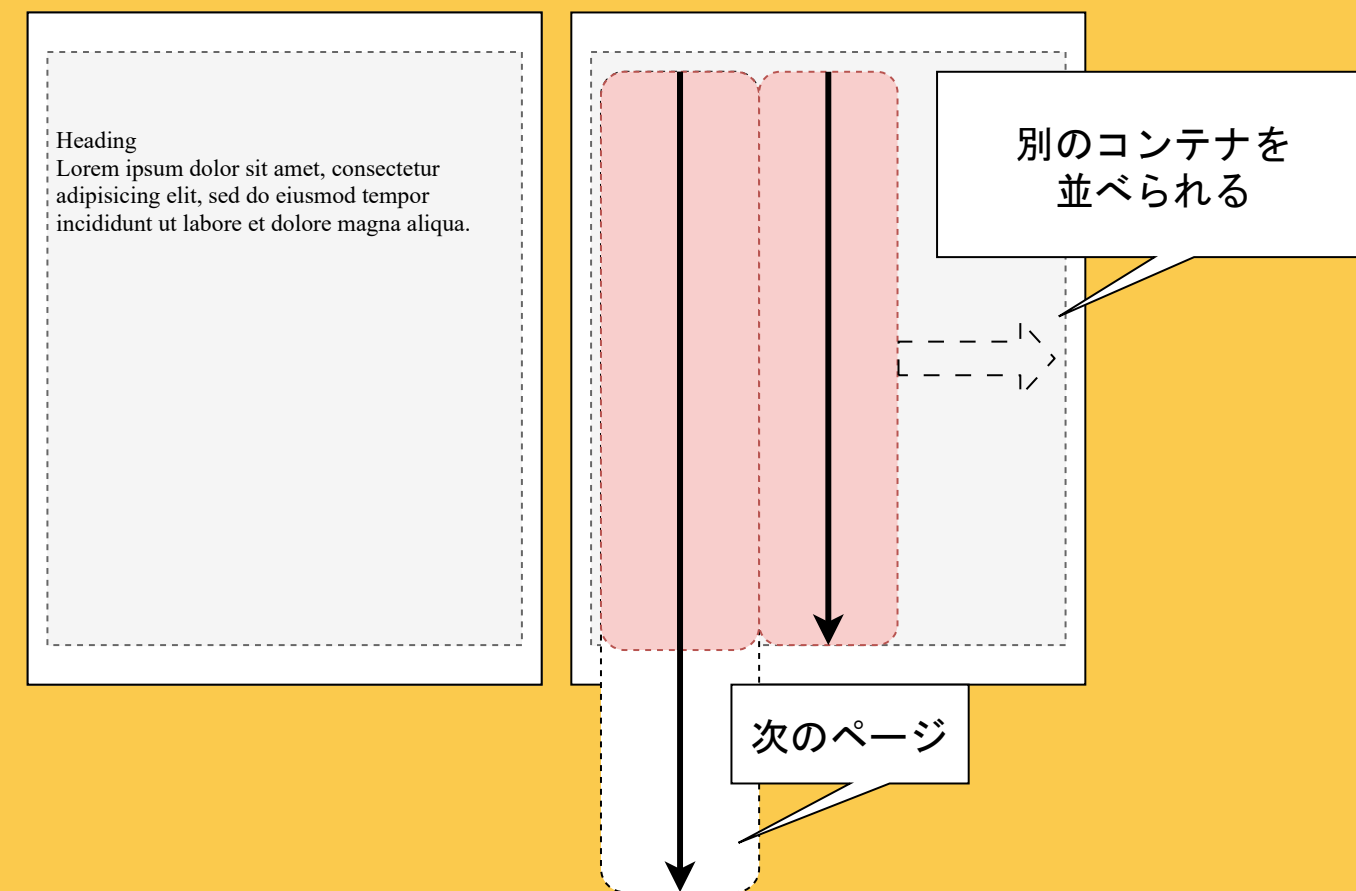
# Flexible Box

アイテムの要素が順に並んでいく。アイテムの並べ・揃え・余白の取り方を外側（コンテナ）で制御できる

## 並べ方



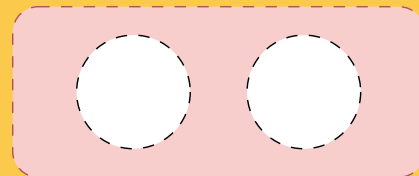
```
flex-direction: row;  
flex-wrap: wrap
```



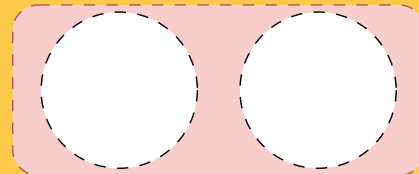
```
flex-direction: column
```

# Flexible Box

## 揃え方



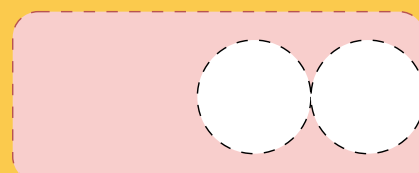
space-evenly



space-around

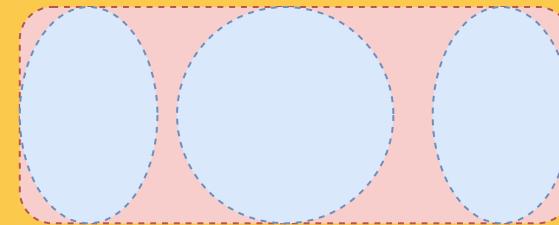


space-between

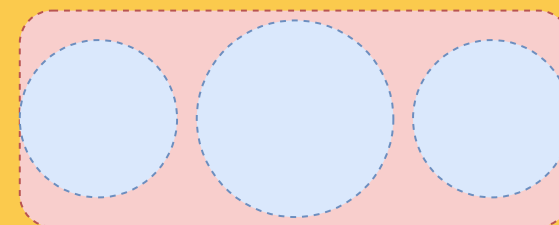


flex-end

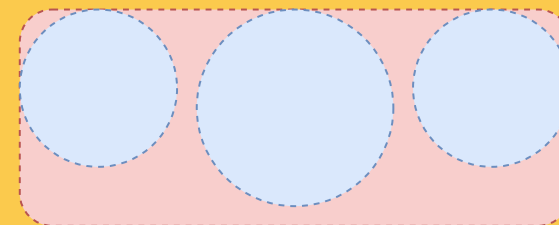
justify-content



stretch



center



flex-start

align-items

# Flexible Box

## 余白の取り方

- 揃え方と関連
- `flex-basis` • `flex-grow` • `flex-shrink`
- レスポンシブでない CSS 組版では、揃え方がバッチリ一度決まれば出番は低め

# Flexible Box

## フレックスのポイント

- コンテンツの横並べが inline-block などより格段に簡単
- 並べたアイテムの縦配置も変更し易い
- 縦に並べ変えるのが簡単
- Web ページと違ってスクロールできないため、`flex-wrap: wrap` が重要
- 隙間のある空間でピンポイントの配置はできない→グリッドの出番



# table 要素では.....

CSS Grid 以前から Web サイトでコンテンツを自在に配置する要求はあったが、table や frameset といった HTML 要素をいじることで実現していた

## table

テーブル	
A	B
	C

```
<table>
  <thead>
    <tr><th colspan="2">テーブル</th></tr>
  </thead>
  <tbody>
    <tr>
      <th rowspan="2">A</th>
      <td>B</td>
    </tr>
    <tr><td>C</td></tr>
  </tbody>
</table>
```

# table 要素では.....

グリッドなら

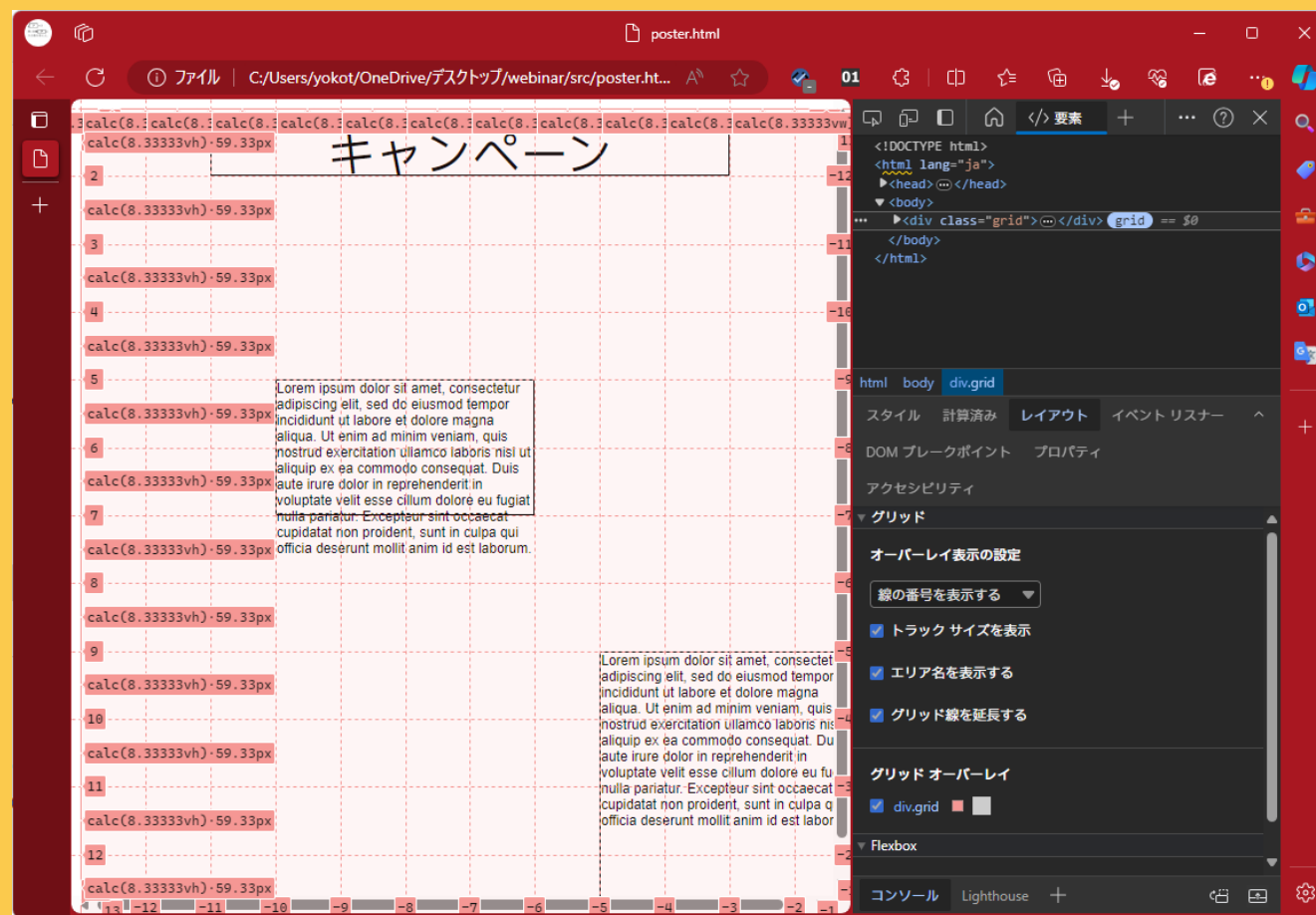
HTML 上は単純な構造のまま

grid	
A	B
	C

```
<div id="grid-ex" >  
  <div>grid</div>  
  <div>A</div><div>B</div>  
  <div>C</div>  
</div>]
```

# CSS Grid

トラックを設定し、*行と列の格子線*・テンプレートでアイテムを配置



ブラウザの開発者モードでグリッドが確認できる

# CSS Grid

## トラックを設定する

グリッドの縦横に用意する帯（または筋）。帯の幅の中にアイテムを配置する

# CSS Grid

```
.grid {display:grid;
  grid-template-rows: 1cm 1cm ...;
/*inline-start-end の端から端まで引かれる帯の各幅*/
  grid-template-columns: 1cm 1cm ...;
/*brock-start-end の端から端まで引かれる帯の各幅*/
/*全長から固定幅でない幅サイズを等分した fr 単位指定も可能*/
  row-gap: 0pt; /*inline-start-end のトラック間の余白*/
  column-gap: 0pt; /*brock-start-end のトラック間の余白*/
}
```

# CSS Grid

## 格子線（グリッドライン）

アイテムの開始・終了位置をトラックの端の<sup>すう</sup>数で示す

```
.item {  
  /*先頭の row トラック*/  
  grid-row-start: 1;  
  grid-row-end: 2;  
  /* 2 番目の column トラック開始位置から  
  3 番目の column トラック終了位置 */  
  grid-column-start: 2;  
  grid-column-end: 4; }
```

- 12 本のトラックを引いたら、線の番号は 1-13
- gap が設定されていると数え方が若干ややこしい

# CSS Grid

## 名前付きエリアを設定する

設定したトラック上で、矩形に区切った範囲を名前付けし、名前で位置を指定する

```
.grid {  
  /*いわゆる grail 型のコンテンツ配置例*/  
  grid-template-areas:  
    "header header header"  
    "nav main ."  
    "footer footer footer";  
}  
header {grid-area: header;}  
footer {grid-area: footer;}
```

- 使用しない領域は「.」を記述
- 矩形の形状のみ（L字などは不可）

# CSS Grid

## 自動配置について

- `grid-auto-rows`・`grid-auto-columns` で未指定の箇所に来た場合の追加トラックを設定
- ページレイアウト（スクロール・画面伸縮できないもの）と折り合いが悪く、ページの端を越えていくこともある
- グリッド位置を指定しないアイテムは格子線の順に配置されていく。これも伸縮しないページメディアと折り合いが悪い



# inline-grid

行内でどのようなときにグリッドがあると便利か？

思<sub>ヒ</sub> 学<sub>ビテ</sub>  
而<sub>ニ</sub> 而<sub>ニ</sub>  
不<sub>レ</sub> 不<sub>レバ</sub>  
学<sub>ハ</sub> 思<sub>ハ</sub>  
則<sub>デ</sub> 則<sub>スニテ</sub>  
殆<sub>ロシ</sub> 罔<sub>ムク</sub>

- ex. 漢文配置
  - `template-area` で返り点や送り仮名位置の要素を配置
  - 返り点位置は他では結構難しい

# グリッドのポイント

- トラックの設定と、格子線またはテンプレート領域配置の 2 段階
- レスポンシブではないページメディアでは、自動配置は慎重に
- 長さが分からないコンテンツには向かない（矩形のサイズ予想ができるものがよい）
  - Formatter の overflow 処理拡張である程度対応可能
- 
- （線名など、更に複雑な話は今回省いている）

# セレクタ指定を柔軟に

- セレクタが充実する利点
- `:is()`, `:where()`
- `:has()`
- `:not()`
- `::before(n)`, `::after(n)` 拡張

# セレクタが充実する利点

レイアウトが簡潔に

```
p { /* p 要素すべて */  
#fig1 { /* id が fig1 の要素 */  
:last-child,  
:last-child > :last-child {  
    /* 最後の要素とその子の最後の要素 */  
}
```

- セレクタで当てる対象を指定する
- ID 指定など、構造と見た目が離れたセレクタ指定は読みにくい
- 単純なセレクタのみでは記述が複雑に

# :is(),:where()

複数の指定をまとめる

```
section h1 span,  
section h2 span {sectionの子孫の h1 の子孫の span と  
                sectionの子孫の h2 の子孫の span  
}
```

```
/*詳細度の違いがある。  
慣れるまでは where をまず使うのがおすすめ*/  
section :where(h1,h2) span {  
...}
```

# :has()

○○:has(～)で「～を持つ○○要素」を対象にできる

```
section > h2 {  
  background-color: cadetblue;  
}  
section:has(>h2) {  
  /*h2 レベルの section だけを指定*/  
  border: solid 1pt black;  
  padding: 0.5rem;  
}  
section:has(>h3) {  
  background-color: lightpink;  
}
```

```
<section><h2>h2 レベル</h2>  
  <section><h3>h3 レベル</h3>  
  ...  
</section>  
</section>
```

# :not()

○:not (~) で「～でない○要素」を指定。CSS4 の:not () では複数を指定可能に

```
li:not(.special, .veryspecial) {  
    /*複数のセレクタを同時指定など*/  
    ...  
}  
p:not(:first-child) {  
    /*2 つめ以降の p*/  
}
```

# ::before(n),::after(n)

要素の前後に *複数* のコンテンツを追加する

**::before, ::after**

要素の前後にコンテンツを表示する *疑似要素*

- 1つの要素に各1つずつ指定可能
- 装飾を変えたい場合、HTML タグを増やすことになる



# ::before(n),::after(n)

::before(n),::after(n) 擴張

```
h2::before(1) {  
    content: '第' counter(sect) '節';  
    text-decoration: underline;  
    font-size: x-large }  
    h2::before(2) {  
    content: '第' counter(chap) '章';  
    font-size: xx-large;  
    }
```

# その他の 強化機能

- axf:border-radius: diagonal
- OSDC 連携

# (-ah-)border-radius: diagonal

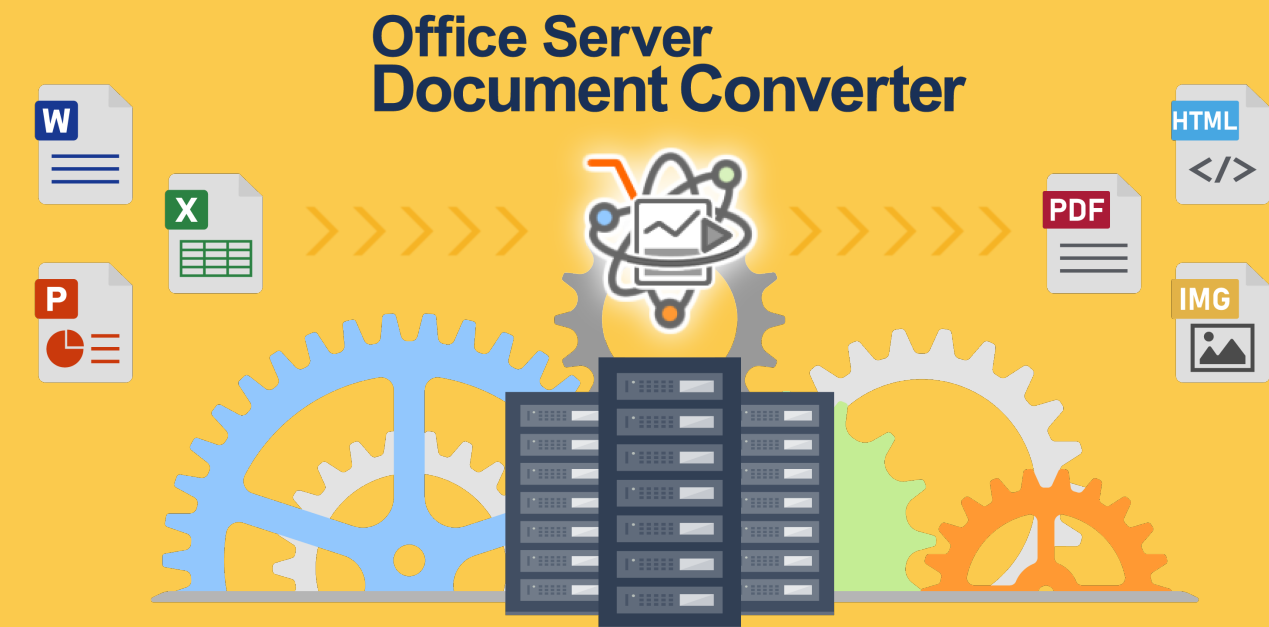
角を丸めではなく対角線を引いた形状に。このスライドでも多用

```
.card {  
  /*右下の角にだけ作用させる */  
  border-bottom-right-radius: 30pt 30pt diagonal;  
}
```

# Office Server Document Converter

MS Office がなくても Word,Excel,PowerPoint を PDF にできる

- 怪しい Office 文書のマクロを排除して閲覧
- PDF で最終文書管理
- MS Office のサーバ運用はライセンスが.....



# axf:image-preproc

Office 文書を外部グラフィック参照に指定すると、OSDC で PDF 化した結果を埋め込める

- 特定ページのみも指定可能
- Formatter のために PDF と元の Office 文書を二重管理しなくともよい
- Formatter の他の機能と組み合わせることも可能！

# まとめ

# まとめ

## Formatter V7.4 は CSS 機能を中心に大強化

- フレックス・グリッドで今まで面倒だったレイアウトも簡単に
  - コンテナ内にアイテムを配置
  - フレックスはフローに沿った配置
  - グリッドは格子線で配置を指定
- セレクタが拡充され、指定の見通しがよくなる
- border-radius に角の切り取り指定追加
- OSDC 連携で Office 文書を画像のように参照
- おまけ:未紹介機能の一部
  - XML Catalog で Formatter からパス書き換えが可能に
  - CSS デバグツリーでレイアウト指定箇所のデバッグ
  - 行分割のハイフネーションを行頭側にも表示可能に
  - URL の行分割ルールをシカゴマニュアルのように変更可能に
  - ファイル添付注釈に data スキーム指定可能に
  - トンボを画像など（ページ内容）の上に描画可能に
  - ほか