CSSNITE

Coder's High 2018

2018.9.29

もう一歩踏み込んで現場で使うCSS Grid

株式会社ICS 鹿野 壮

自己紹介

鹿野 壮 (かの たけし)

ics.media

@tonkotsuboy_com

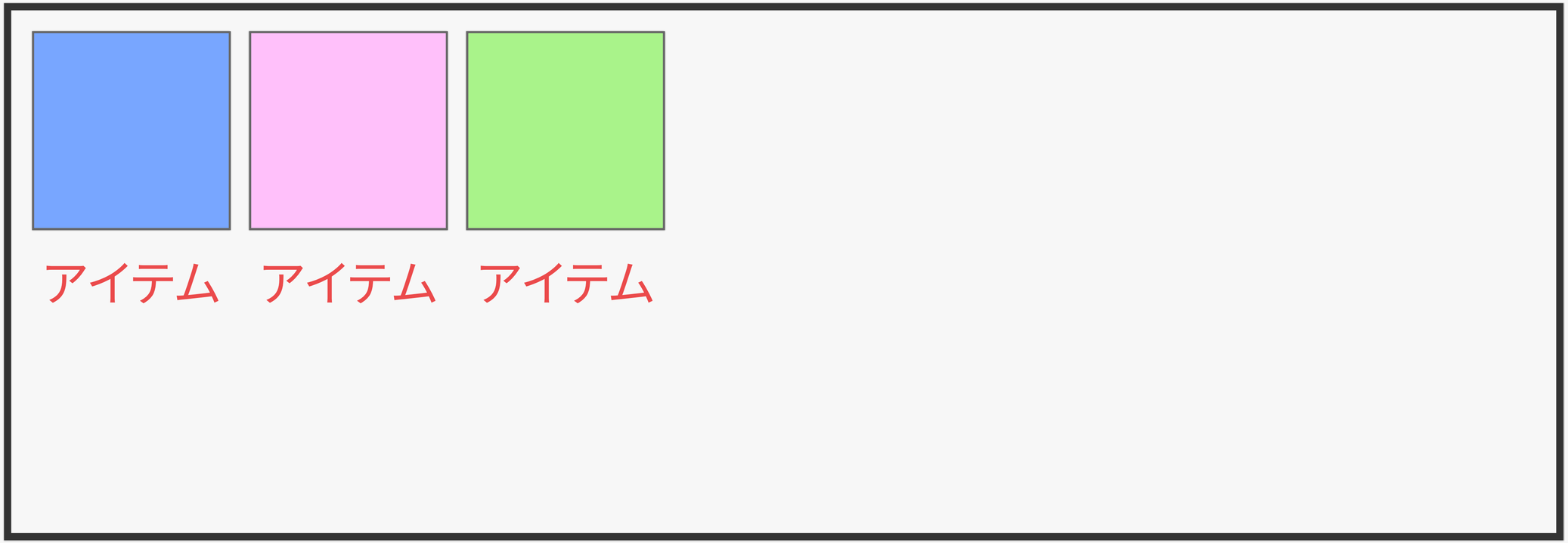
もう一歩踏み込んで現場で使うCSS Grid

1. CSS Gridの基本
2. CSS Gridのオススメの使い方
3. 弊社案件における採用事例
4. 2018年はIE 11対応がラクになった
5. 未来のCSS Gridと周辺技術
6. まとめ

よくあるボックスレイアウトの手法

float：テキストの回り込み

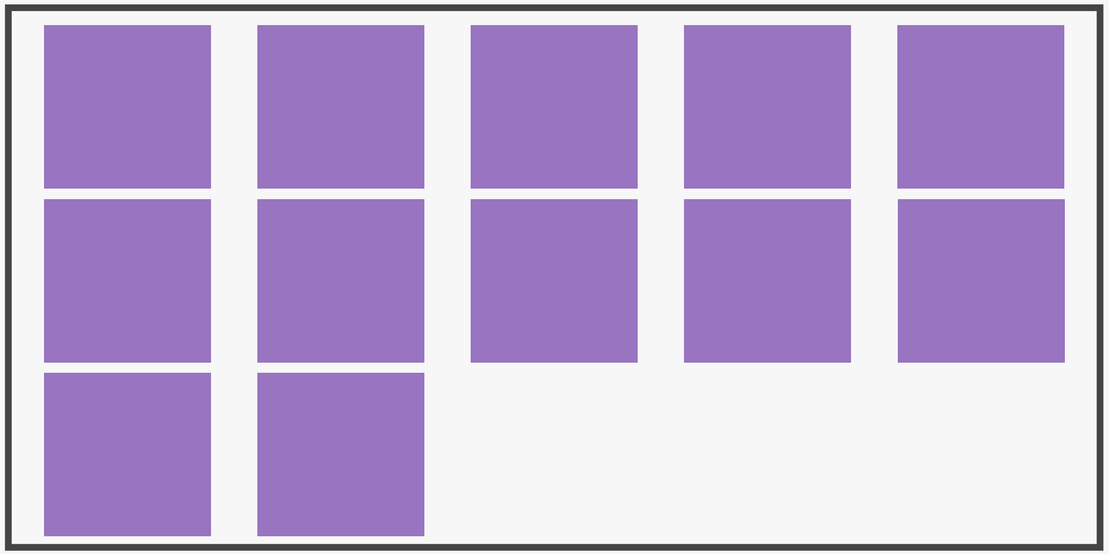
Flexbox：1軸方向のレイアウト



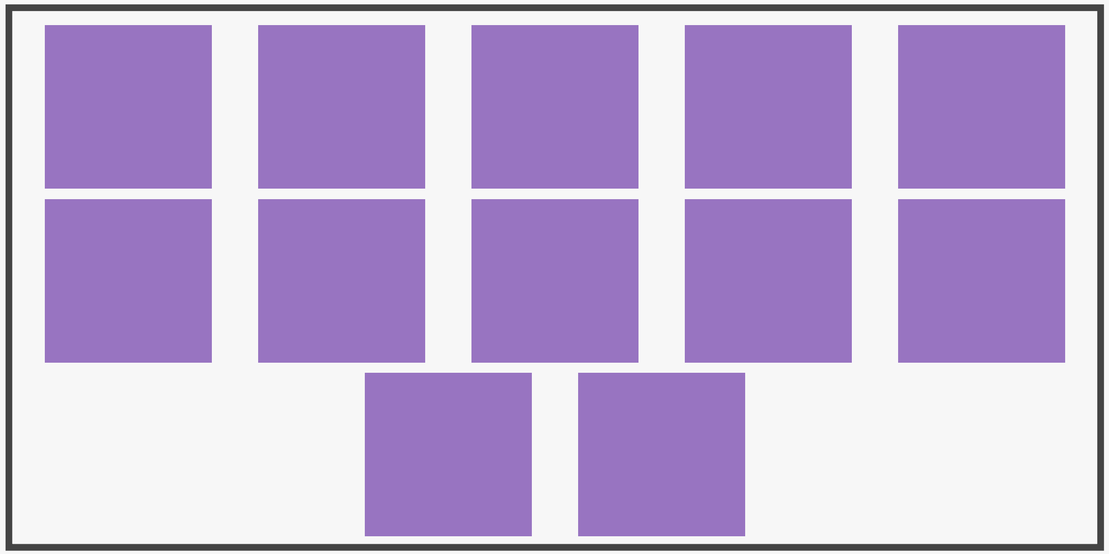
Flexboxは
複数行のレイアウトが苦手

複数行レイアウトの均等配置の最終行が揃わない

justify-content: space-around



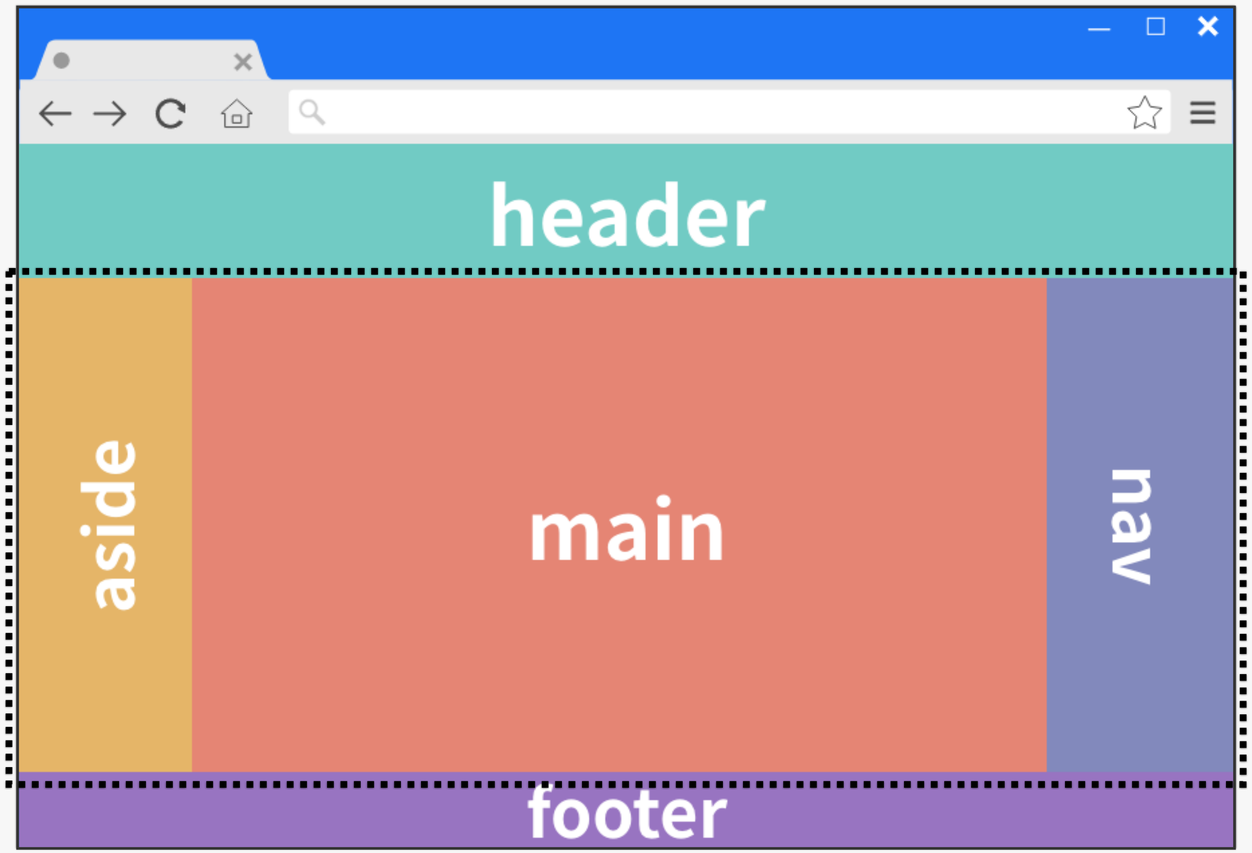
理想



現実

※ 透明なボックスを配置したり、flex-startとcalc()を組み合わせて解決可能だが、煩雑

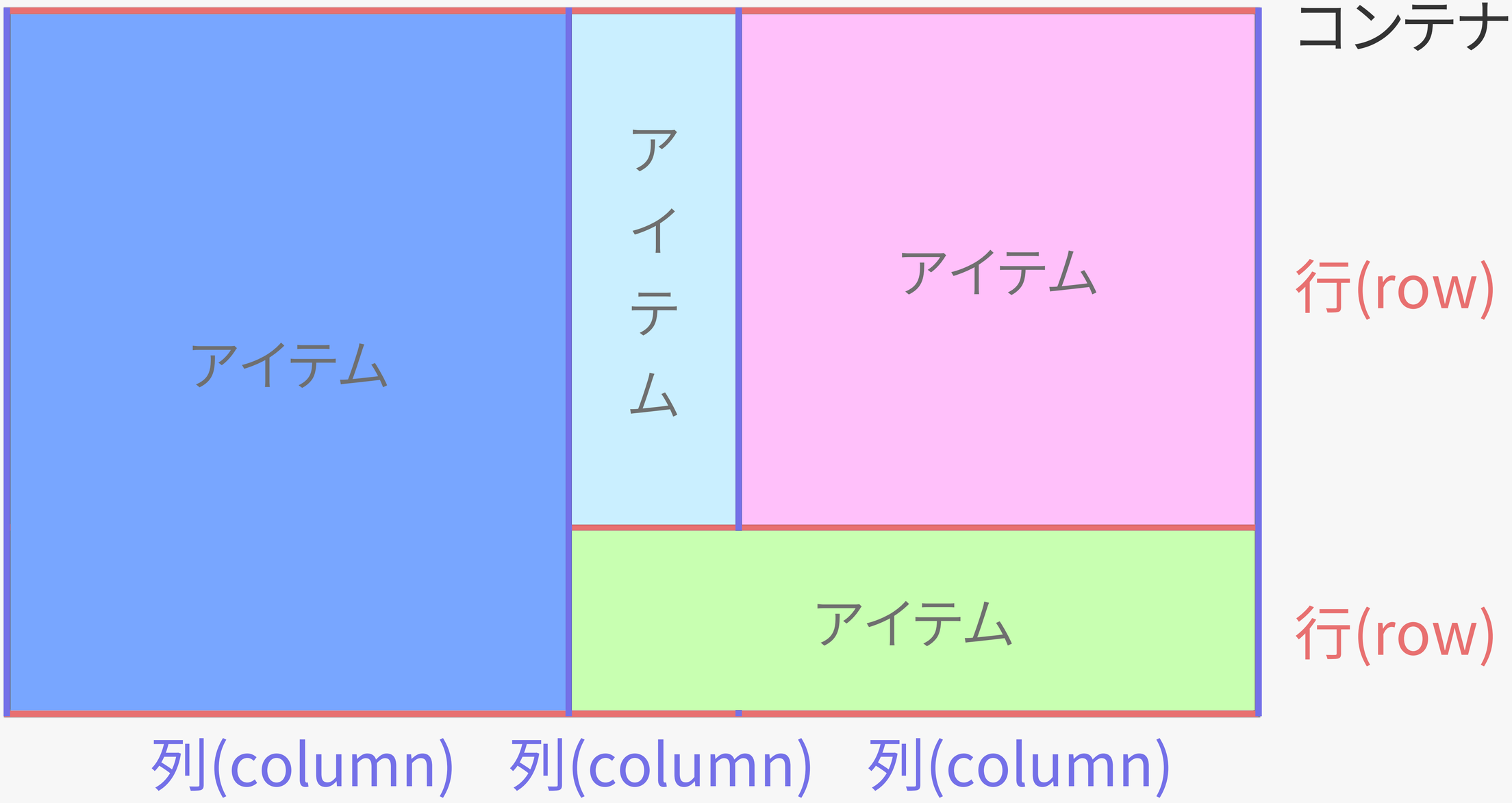
レイアウトのために入れ子が増える



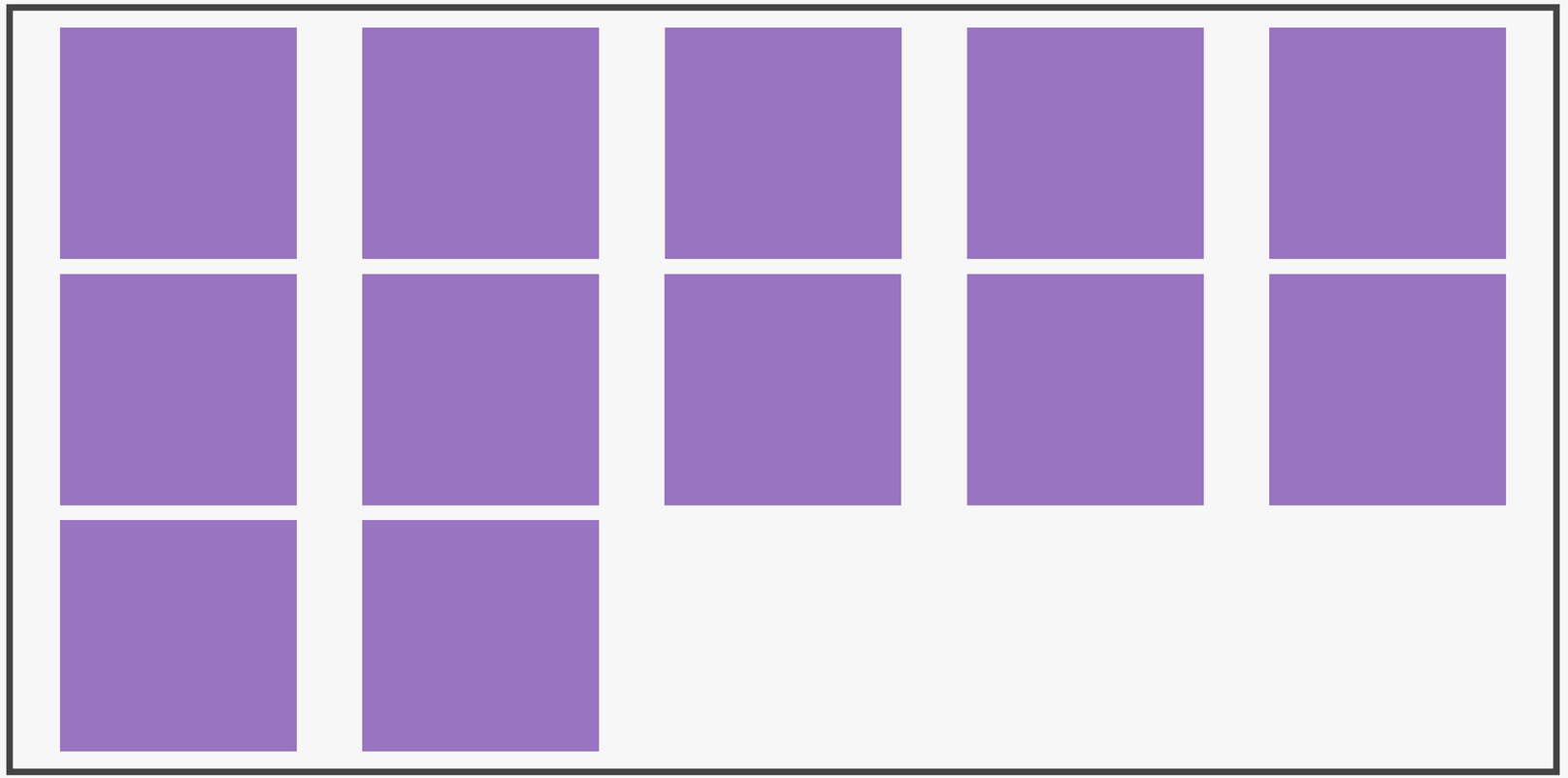

```
<div class="container">
  <header>header</header>
  <div class="middle">
    <aside>aside</aside>
    <main>main</main>
    <nav>nav</nav>
  </div>
  <footer>footer</footer>
</div>
```

複数行のレイアウトに強い
CSS Gridが登場

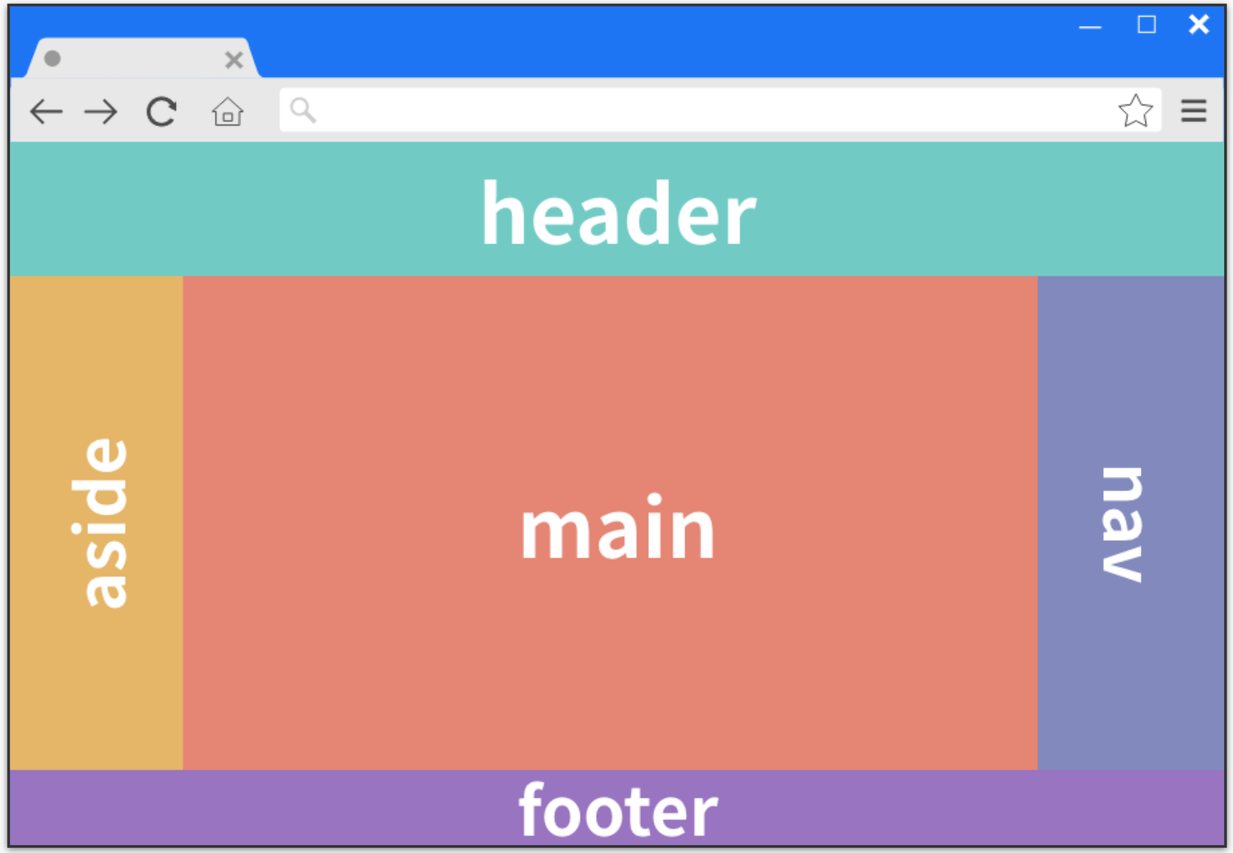
CSS Grid：行と列を使ったレイアウト



複数行レイアウトの最終行が揃う



レイアウトのためのHTMLの入れ子は増えない




```
<div class="container">
  <header>header</header>
  <aside>aside</aside>
  <main>main</main>
  <nav>nav</nav>
  <footer>footer</footer>
</div>
```

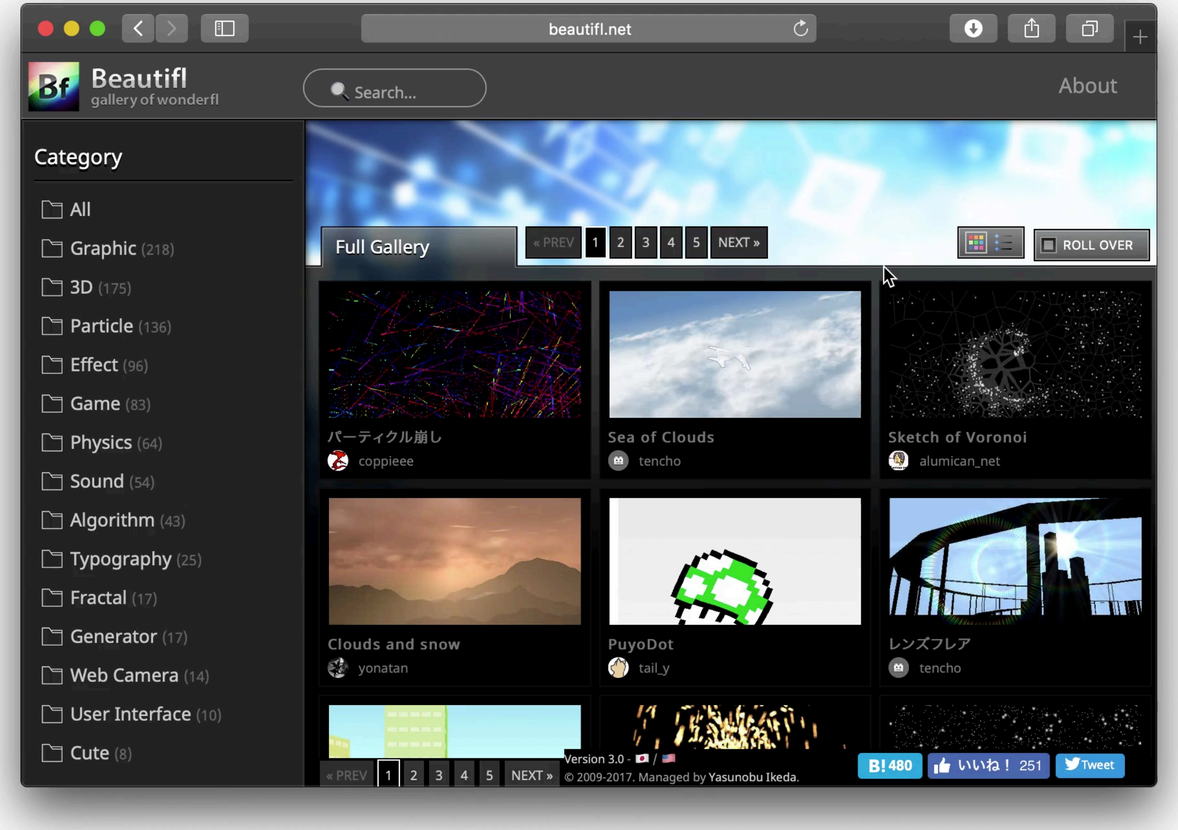
CSS Gridを採用しているサイト

Qiita

https://qiita.com/

- プログラミング知識の共有サービス
- 個別記事の大枠にGridを採用

CSS Gridを採用しているサイト

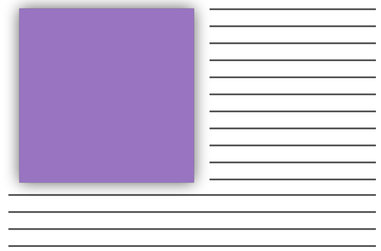
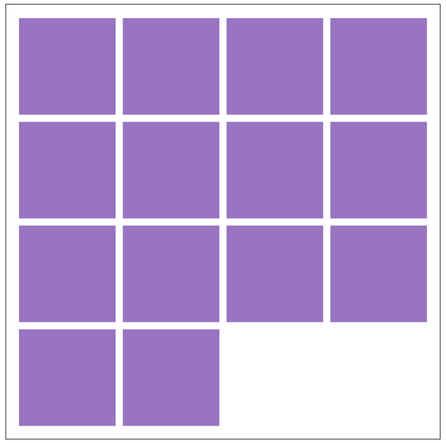


Beautifl

http://beautifl.net/

- Flash作品の紹介サイト
- レイアウトの大枠にGridを採用

用途別レイアウトの使い分け

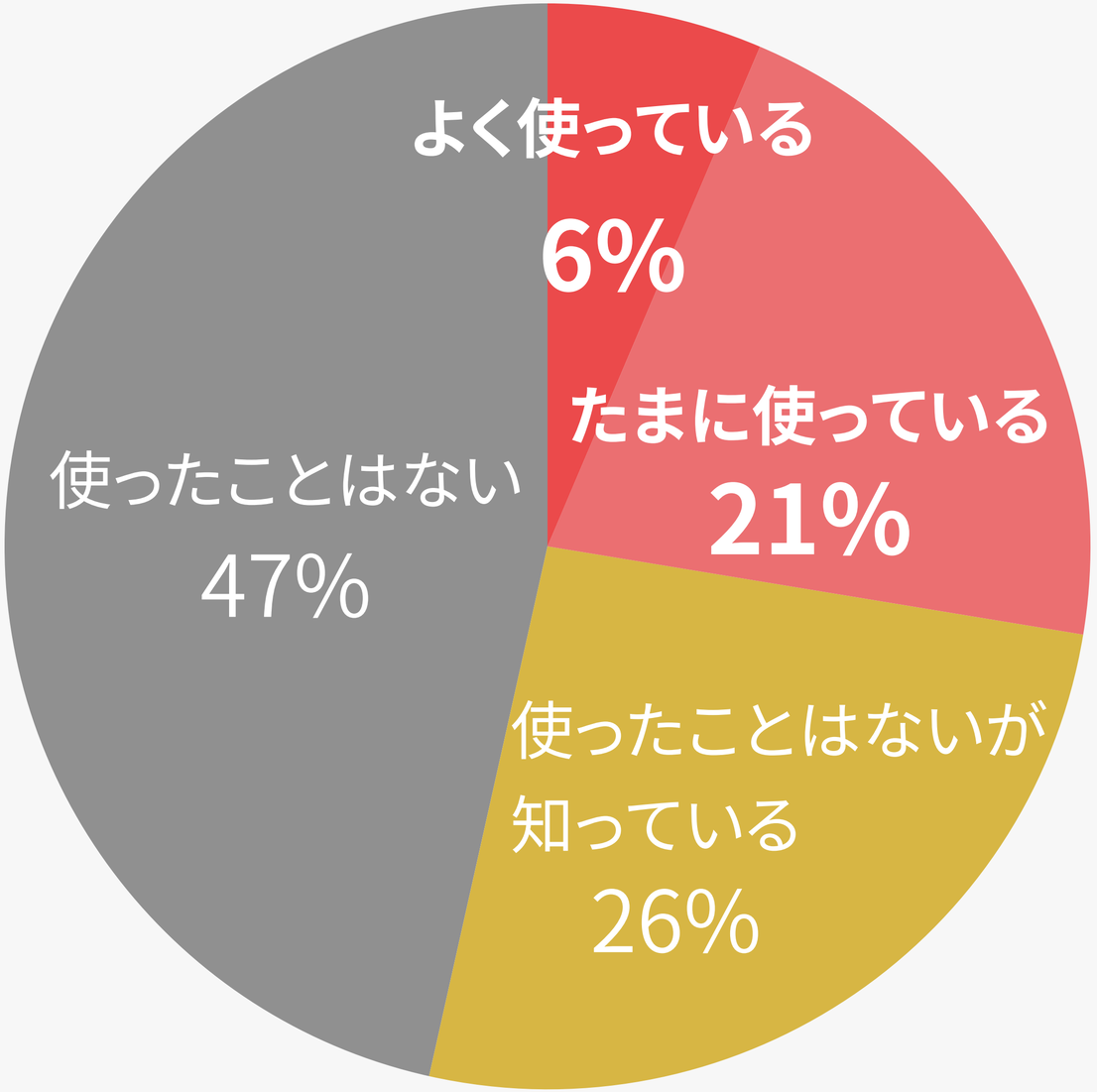
テキストの回り込み	float	
1行の横並び、縦並び	Flexbox	
格子状のレイアウト ページ全体のレイアウト	Grid	

CSS Gridの対応状況

							
ブラウザ名	Chrome	Firefox	Safari	Edge	Internet Explorer	Safari (iOS版)	Chrome (Android版)
最新バージョン	69	62	12	17	11	12	69
Grid対応	○	○	○	○	△※	○	○

※ 部分対応

2018年、CSS Gridはどれくらい使われているのか？



回答者数173人 / Googleフォームでのアンケート

採用する理由（上位2項目）

- レイアウトに便利だから
- モダンブラウザで使えるから

採用しない理由（上位2項目）

- floatやFlexboxで充分だから
- 古いブラウザで使えないから

私もFlexboxで充分だと思っていたが
CSS Gridを使うと世界が変わった

もう一歩踏み込んで現場で使うCSS Grid

1. CSS Gridの基本
2. CSS Gridのオススメの使い方
3. 弊社案件における採用事例
4. 2018年はIE 11対応がラクになった
5. 未来のCSS Gridと周辺技術
6. まとめ

このページの作り方を通して、CSS Gridの使い方を学びます

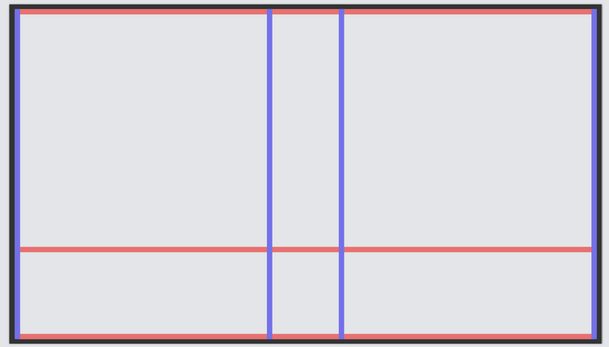
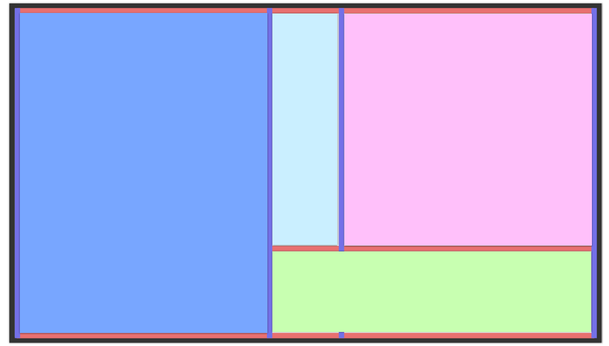
02

今日のうにちゃん

1才になった愛猫「うに」。

家の中のどこに行ってもついてくる甘えん坊。

CSS Gridの基本的な書き方

1. コンテナを作る	display: grid	
2. 行と列を作る	grid-template プロパティ	
3. アイテムを配置する	grid-area プロパティ	

HTMLコーディング

```
<section class="container">
  <div class="visual">（メインビジュアル）</div>
  <div class="number">（数字）</div>
  <div class="expression">（テキスト）</div>
  <div class="other">（３枚の写真）</div>
</section>
```

1. コンテナを作る

コンテナ

1. コンテナを作る


```
.container {
  display: grid;
}
```

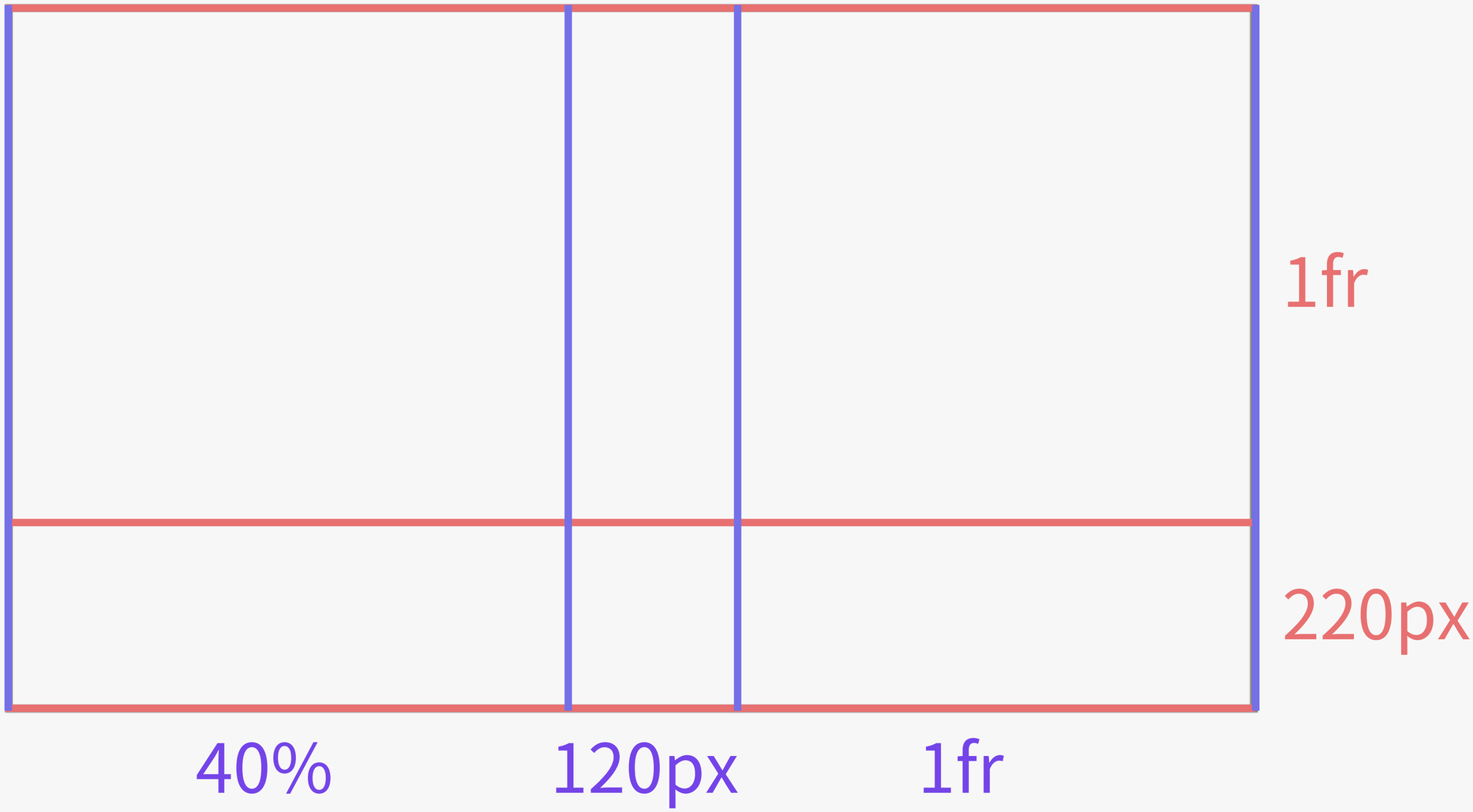
2. 行と列を作る（テンプレートの作成）

行(row)

行(row)

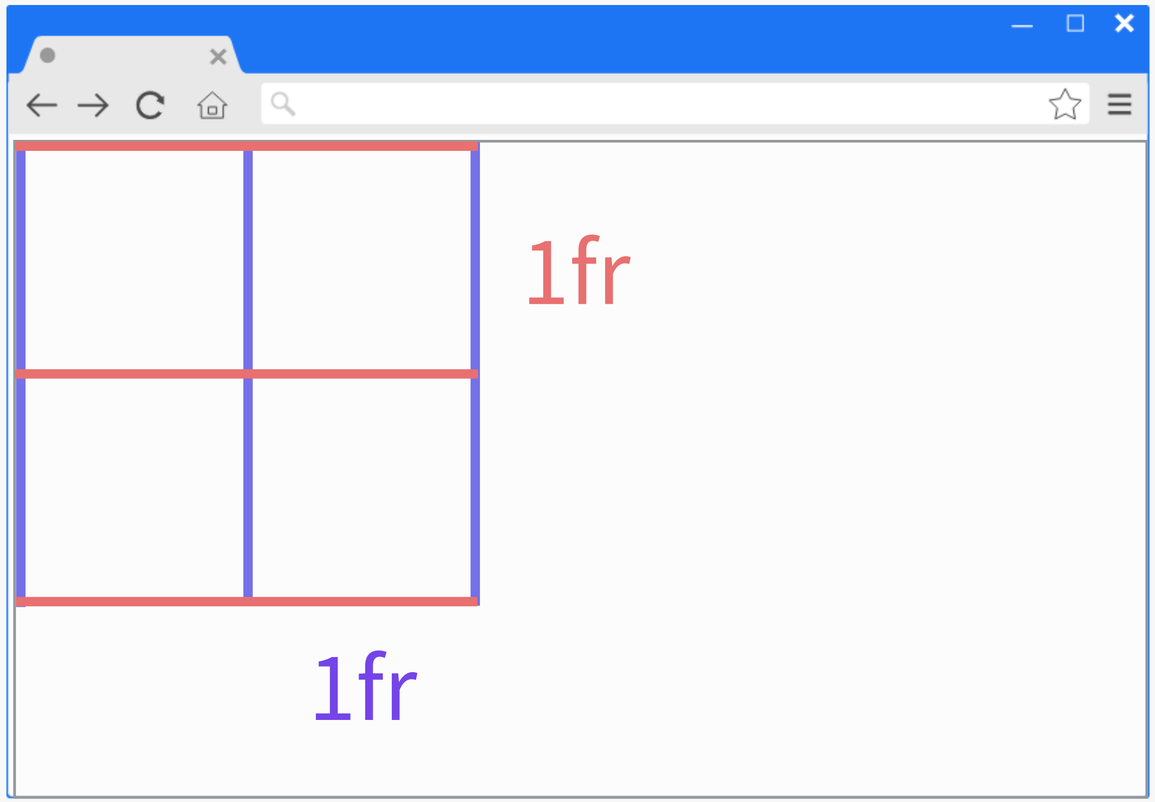
列(column)　列(column)　列(column)

2-1. 行と列のサイズを決定する



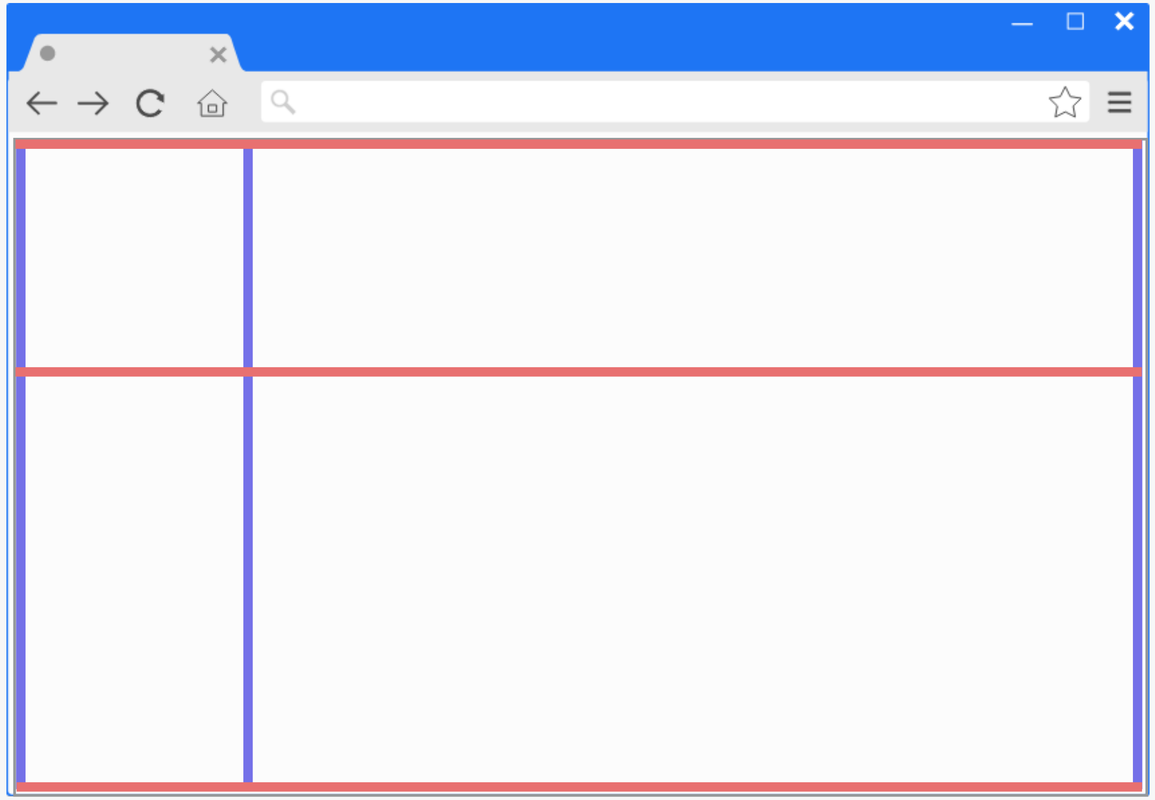
行と列のサイズには「fr」単位が指定できる

コンテナの余白いっぱいに領域を広げる

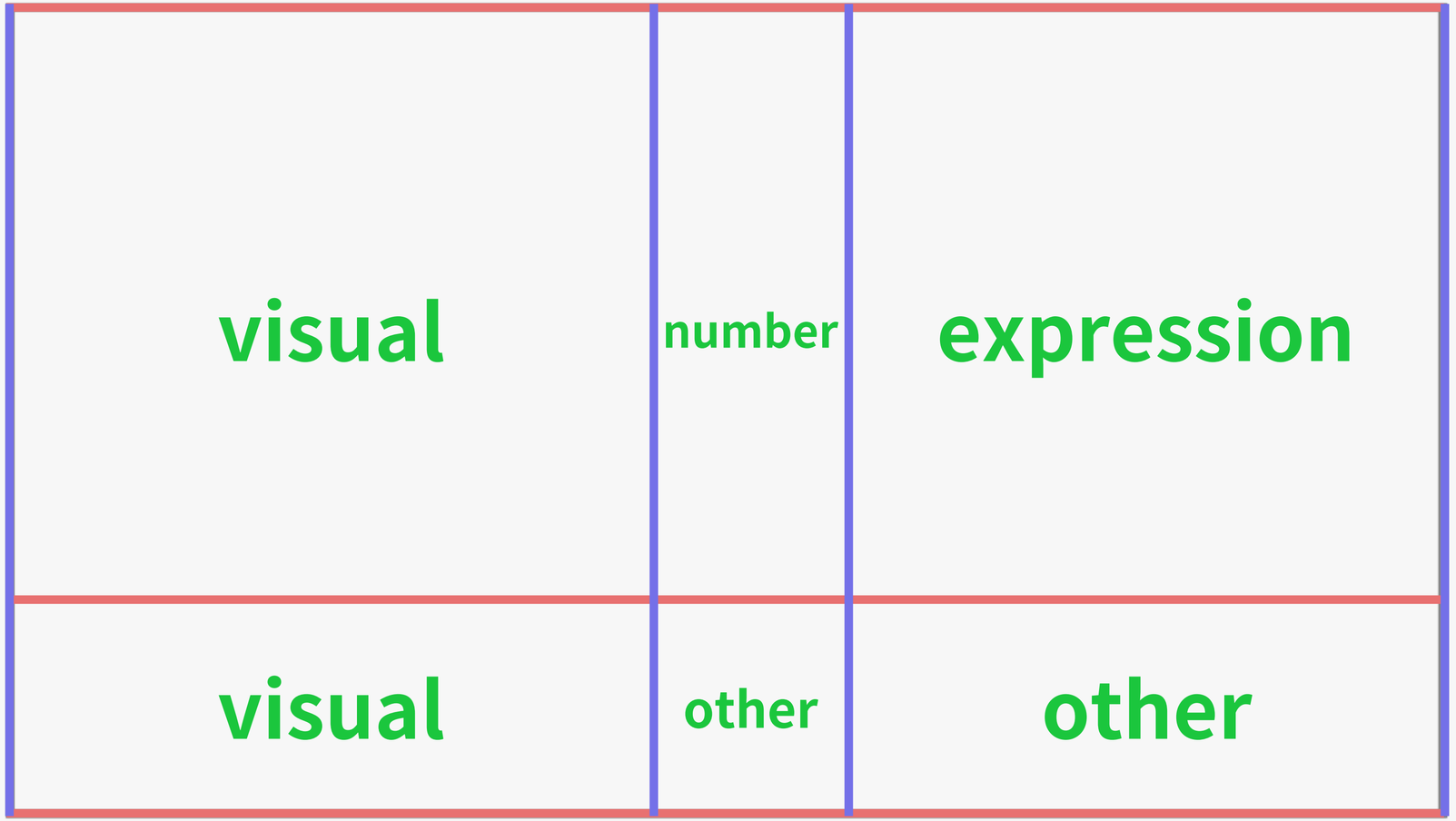


行と列のサイズには「fr」単位が指定できる

コンテナの余白いっぱいに領域を広げる



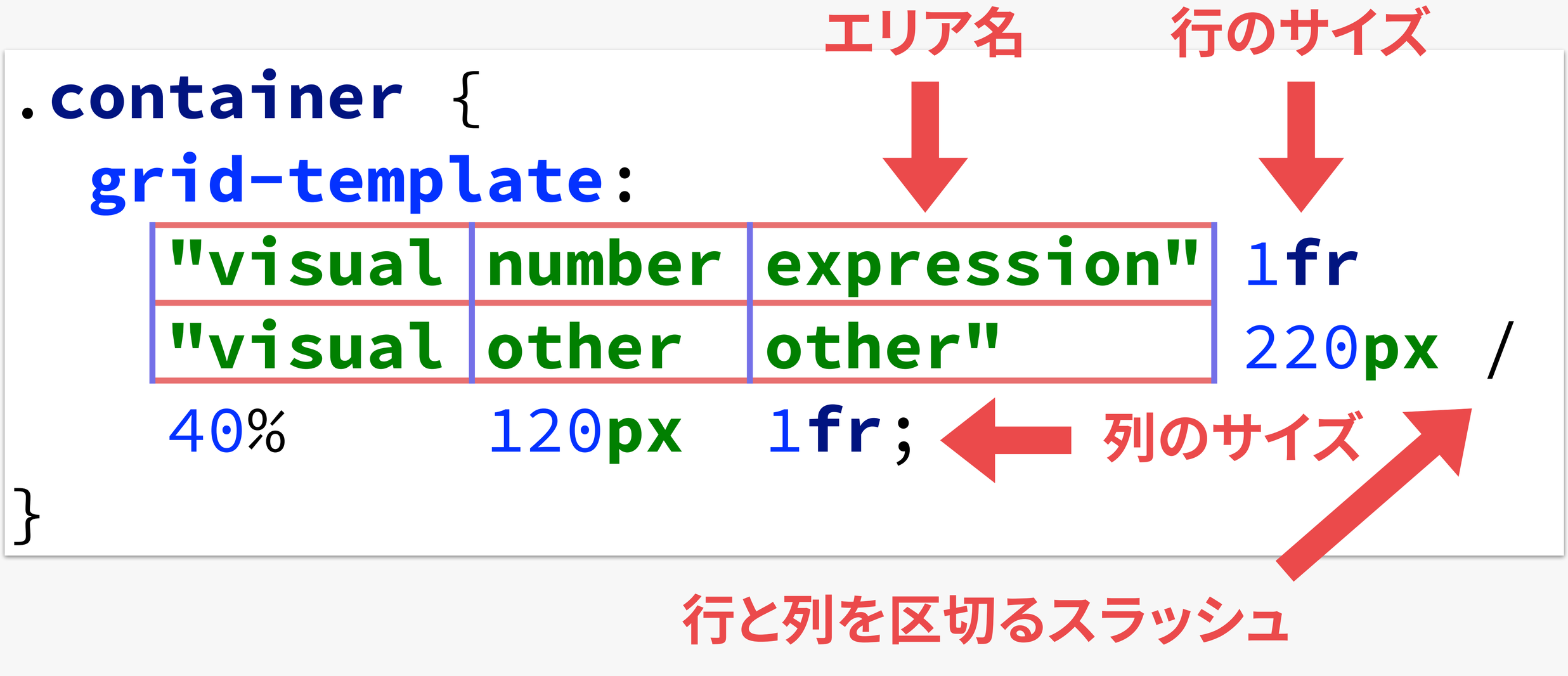
2-2. エリアに名前をつける

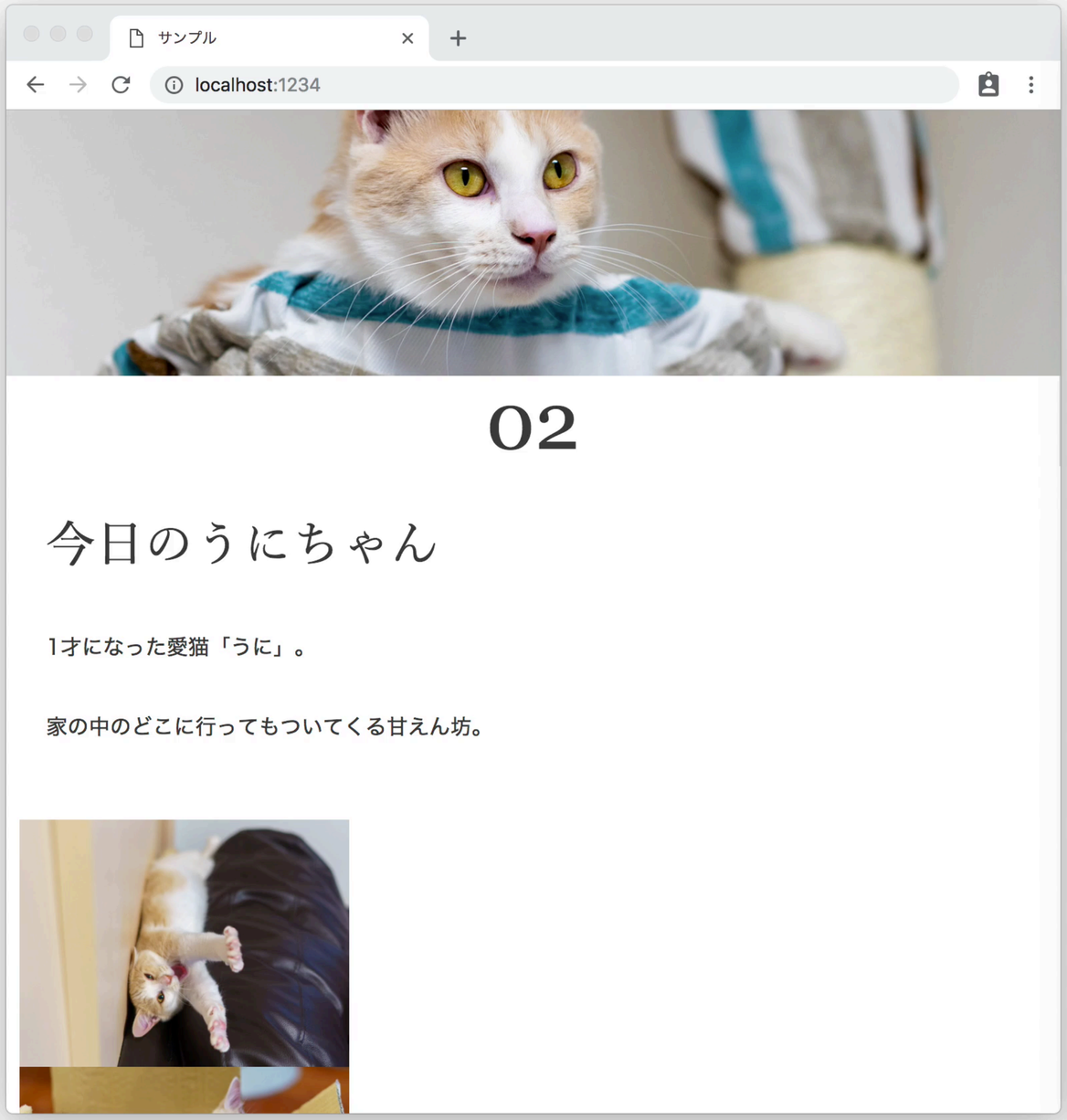


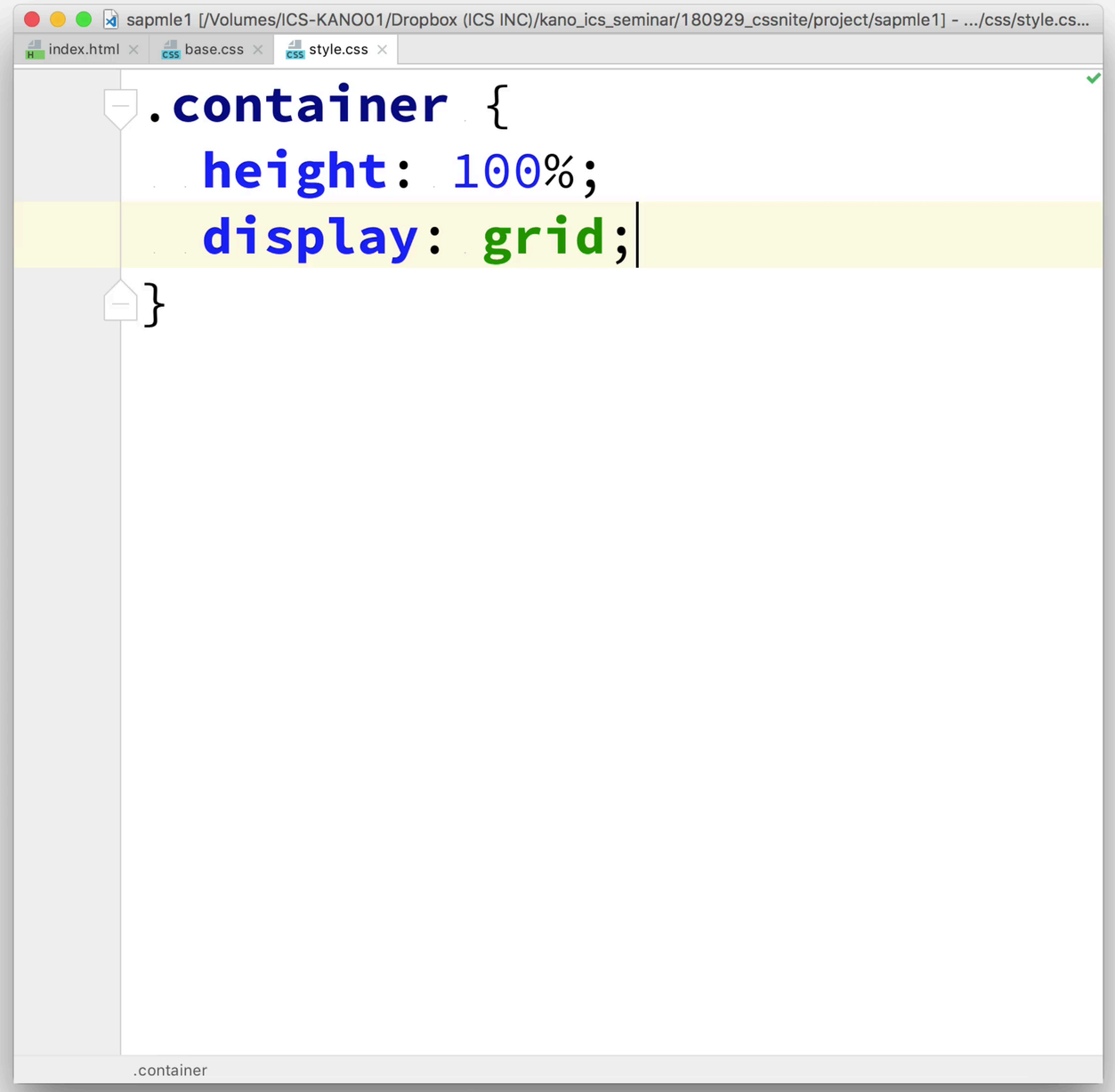
2-3. grid-templateで行列を設定する

```
.container {
  grid-template:
    "visual number expression" 1fr
    "visual other  other"      220px /
    40%      120px  1fr;
}
```

2-3. grid-templateで行列を設定する

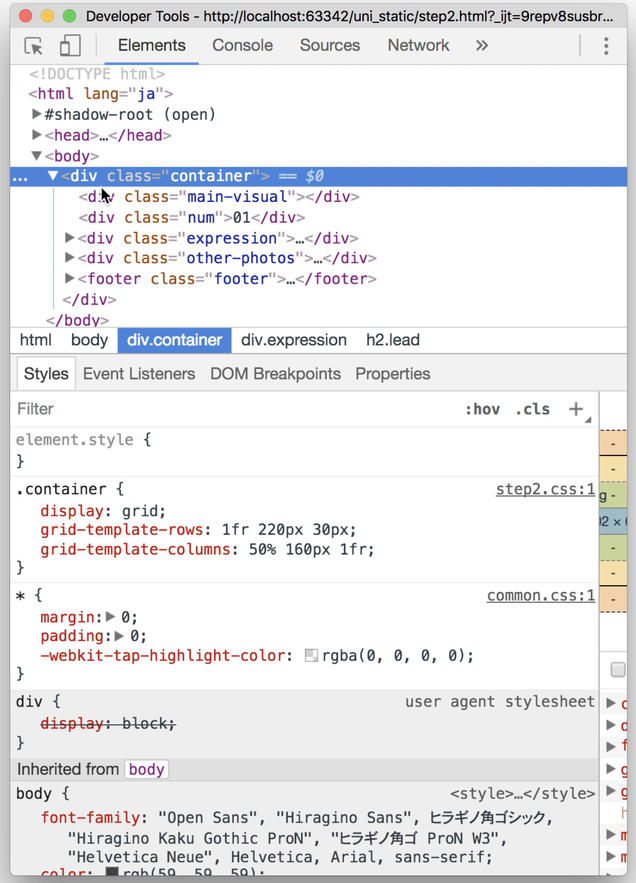



サンプル
localhost:1234
02
今日のうにちゃん
1才になった愛猫「うに」。
家の中のどこに行ってもついてくる甘えん坊。

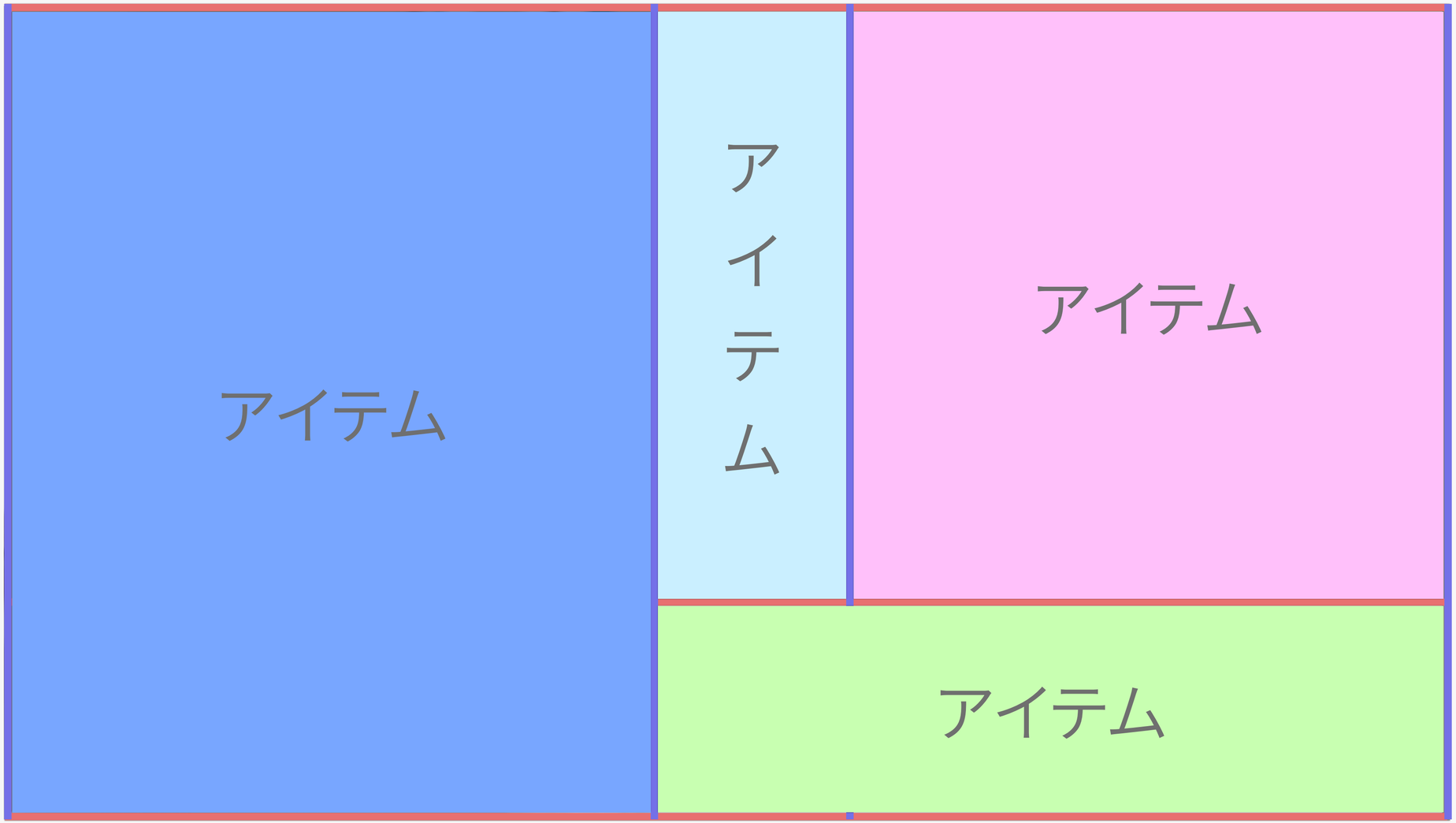

sapmle1 [/Volumes/ICS-KANO01/Dropbox (ICS INC)/kano_ics_seminar/180929_cssnite/project/sapmle1] - .../css/style.cs...
index.html
base.css
style.css
.container {
height: 100%;
display: grid;
}
.container

テンプレートの確認

ChromeのDevToolsやFirefoxの開発ツールで
行と列の設定状況を確認

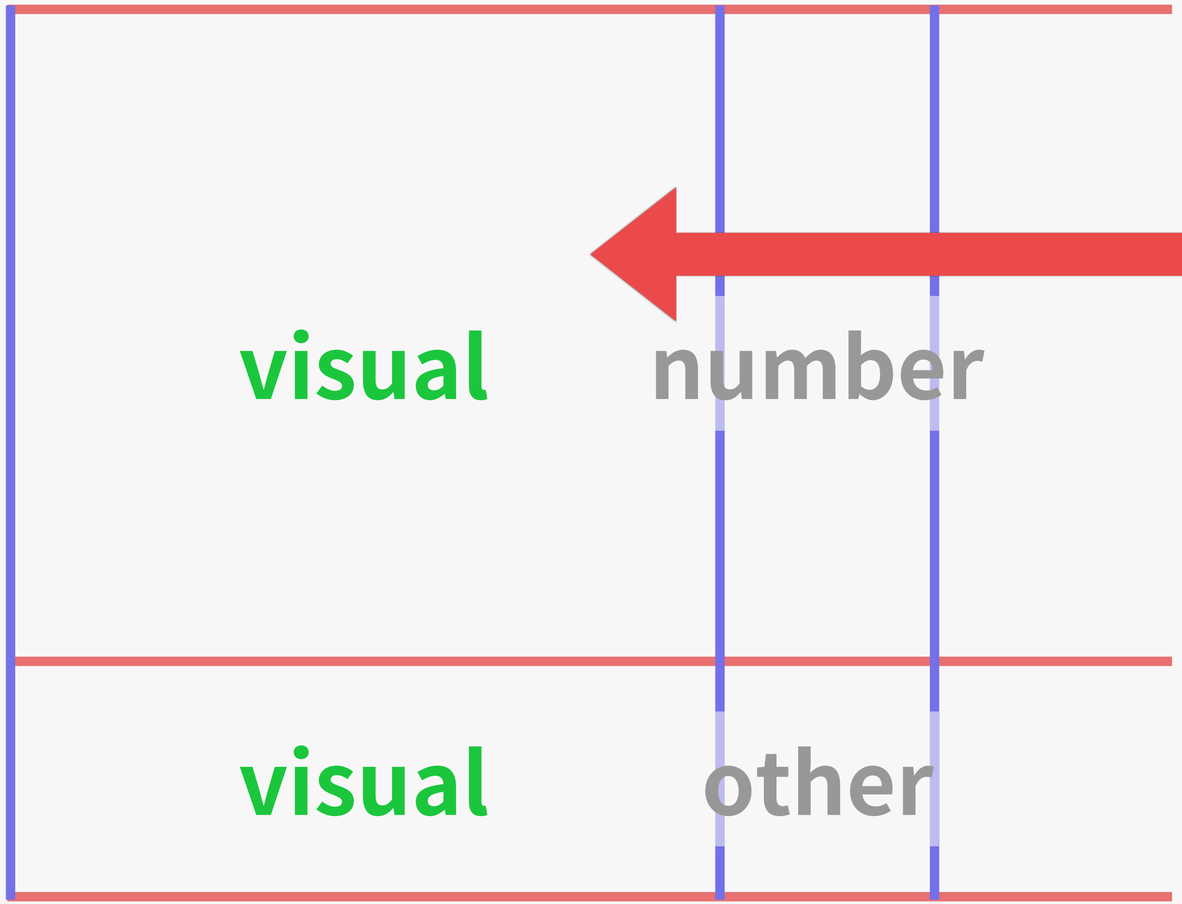


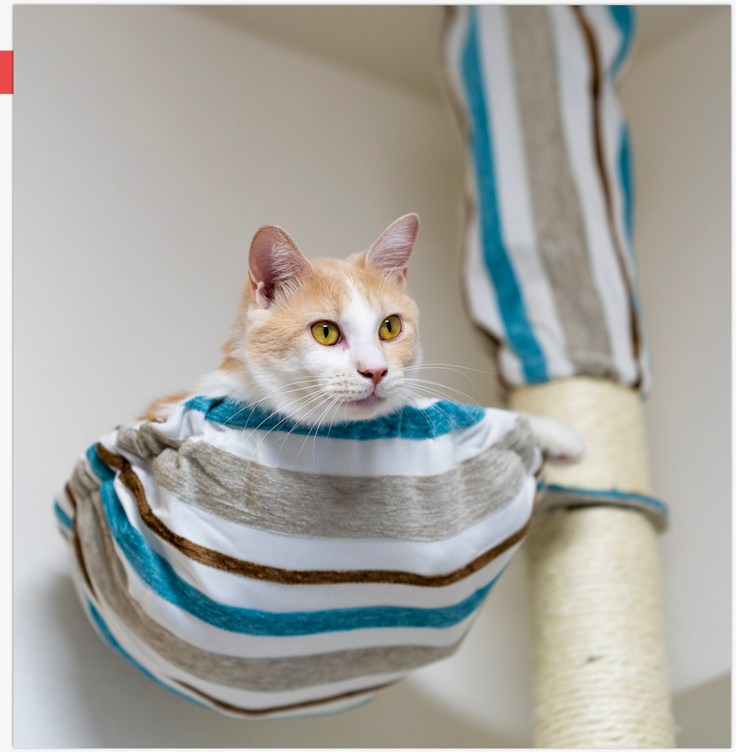
3. アイテムの配置場所の決定

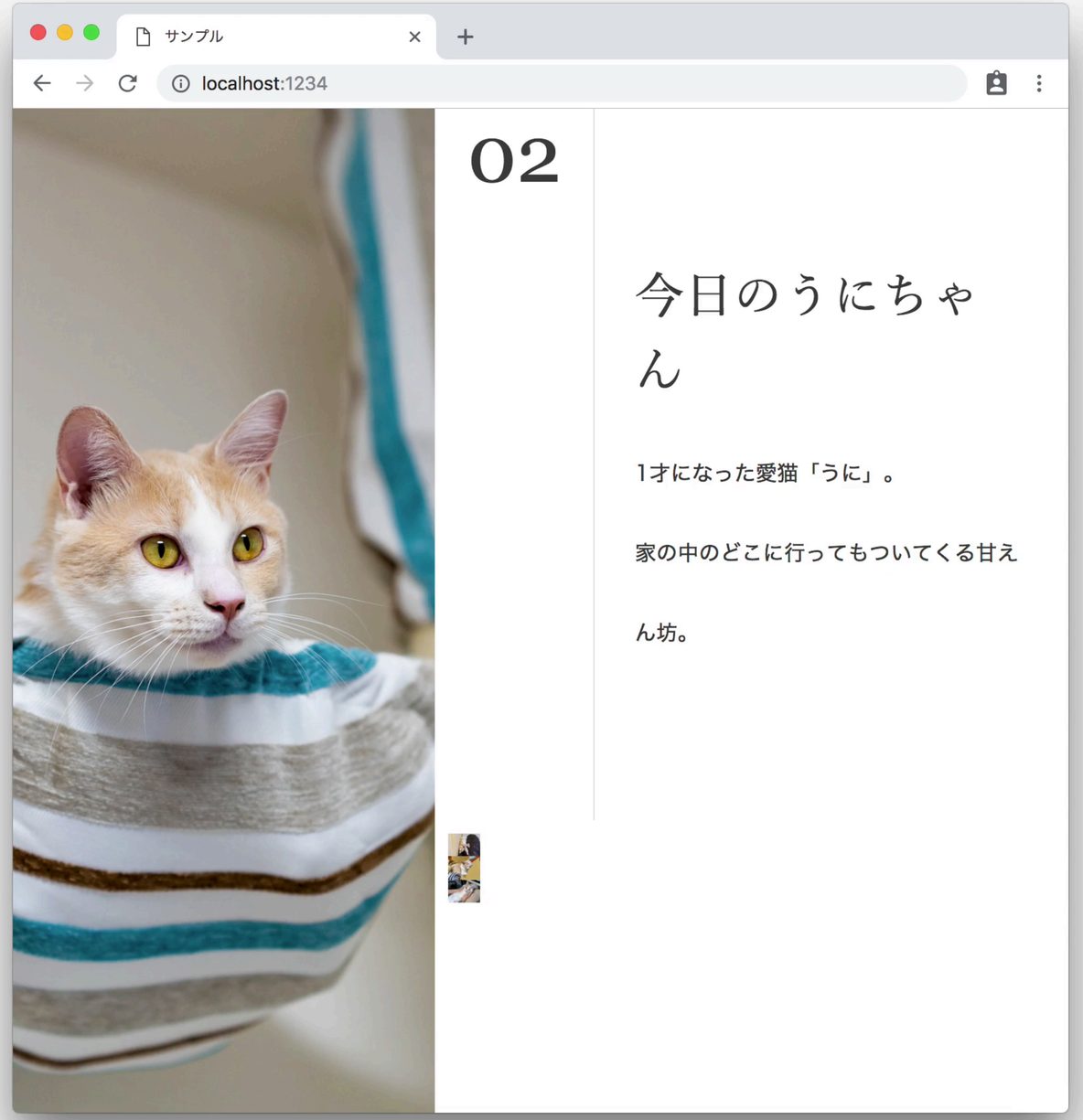


3-1. grid-areaで配置したいエリア名を指定

```
.visual {
   grid-area: visual;
}
```







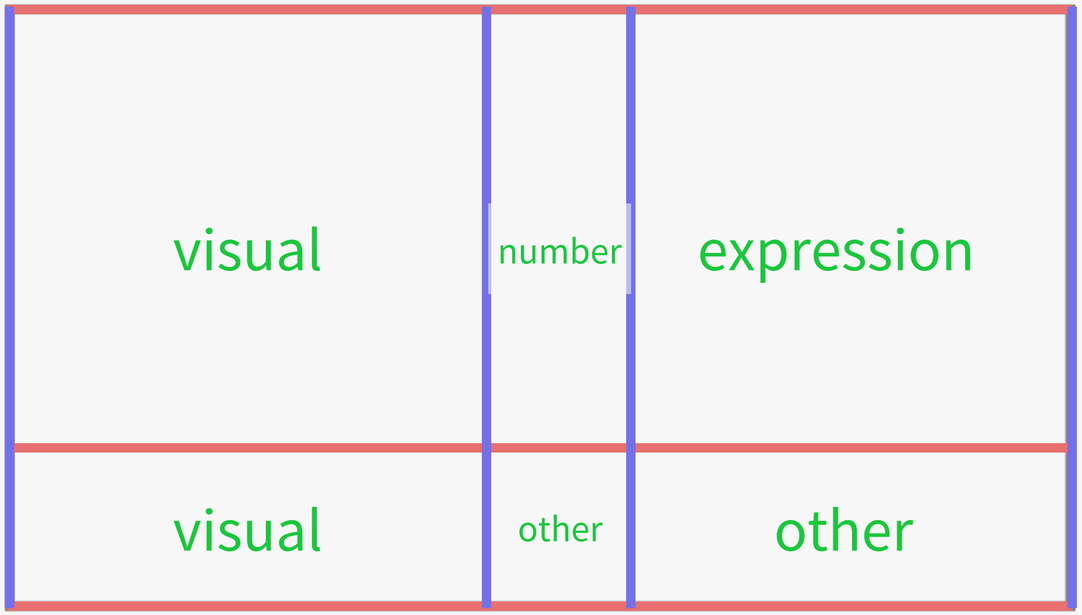
sapmle1 [/Volumes/ICS-KANO01/Dropbox (ICS INC)/kano_ics_seminar/180929_cssnite/project/sapmle1] - .../css/style.cs...

index.html | base.css | style.css

```css
.container {
  height: 100%;
  display: grid;
  grid-template:
    "visual number   expression"
    "visual other    other"
    40%     120px    1fr;
}

.visual {
  grid-area: visual;
}
```

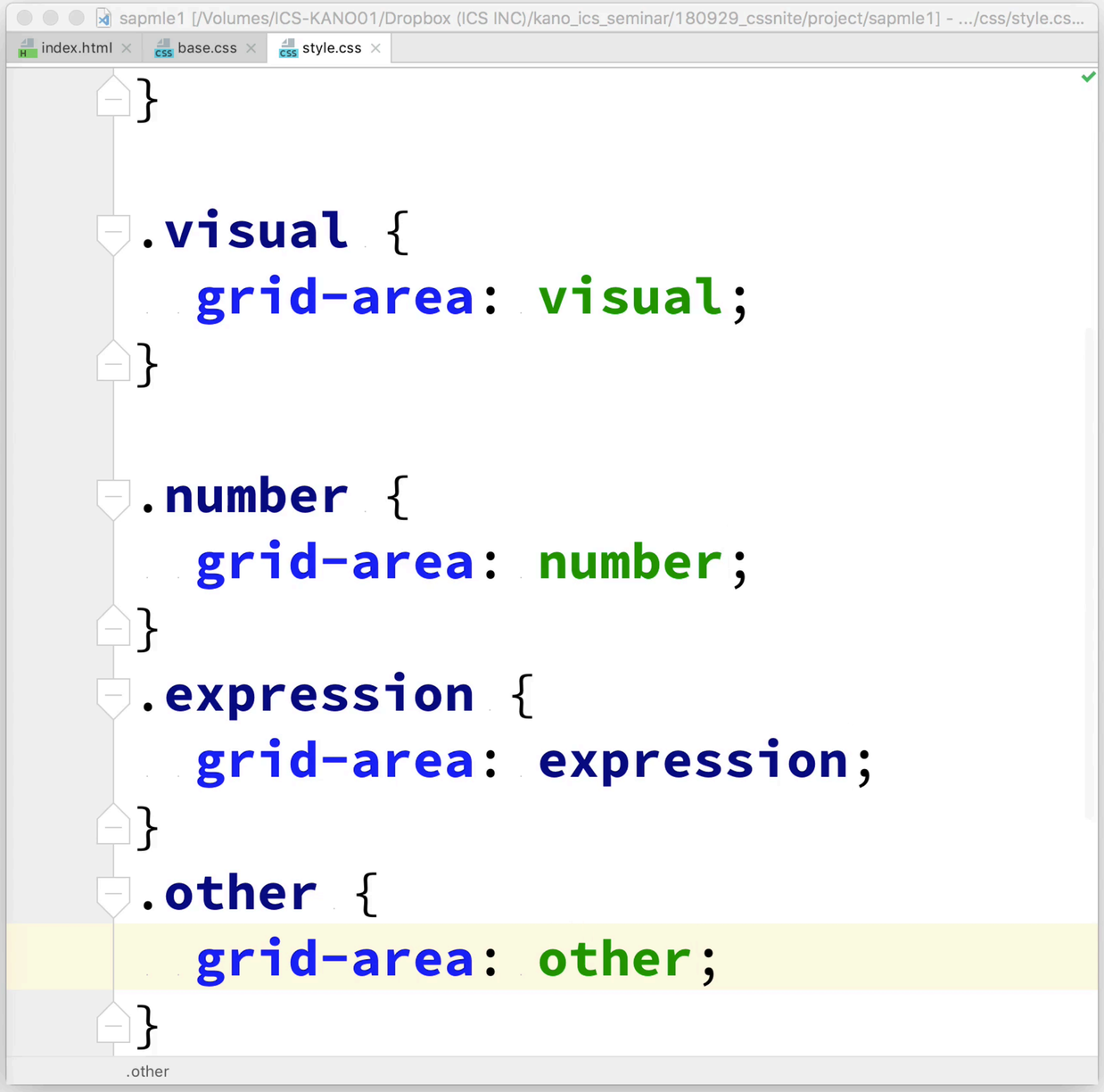
.visual

3-2. 全アイテムに配置エリアを指定する




```
.visual {
  grid-area: visual;
}

.number {
  grid-area: number;
}

.expression {
  grid-area: expression;
}

.other {
  grid-area: other;
}
```


サンプル
localhost:1234
02
今日のうにちゃん
1才になった愛猫「うに」。
家の中のどこに行ってもついてくる甘えん坊。


sapmle1 [/Volumes/ICS-KANO01/Dropbox (ICS INC)/kano_ics_seminar/180929_cssnite/project/sapmle1] - .../css/style.cs...
index.html
base.css
style.css
}
.visual {
grid-area: visual;
}
.number {
grid-area: number;
}
.expression {
grid-area: expression;
}
.other {
grid-area: other;
}
.other

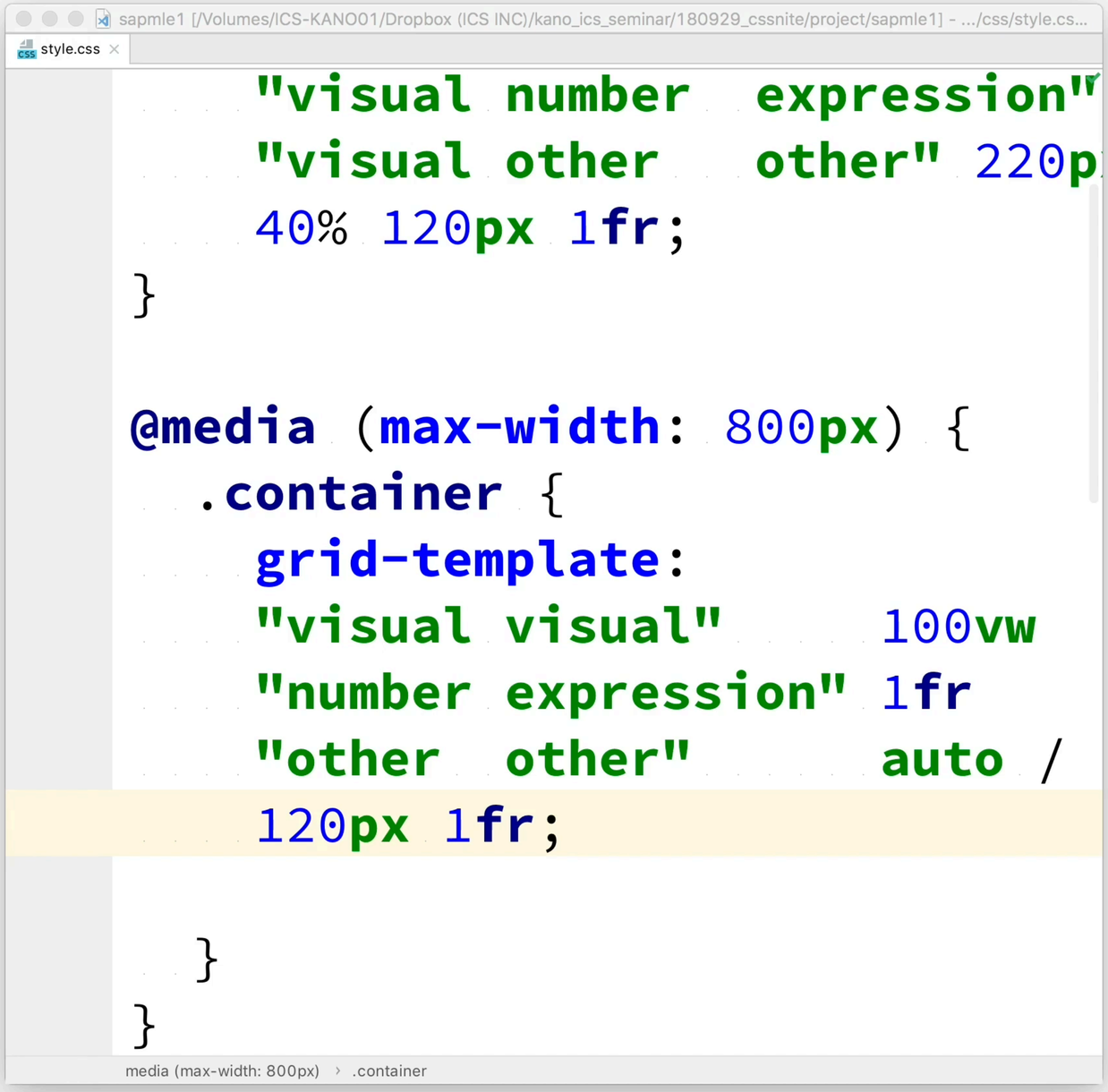

```
sapmle1 [/Volumes/ICS-KANO01/Dropbox (ICS INC)/kano_ics_seminar/180929_cssnite/project/sapmle1] - .../css/style.cs...
index.html  base.css  style.css

    grid-area: visual;
}

.number {
    grid-area: number;
}
.expression {
    grid-area: expression;
}
.other {
    grid-area: other;
    display: flex;
    justify-content: space-betwee
    align-items: center;
}

.other
```

4. レスポンシブ対応するには、行と列を更新する

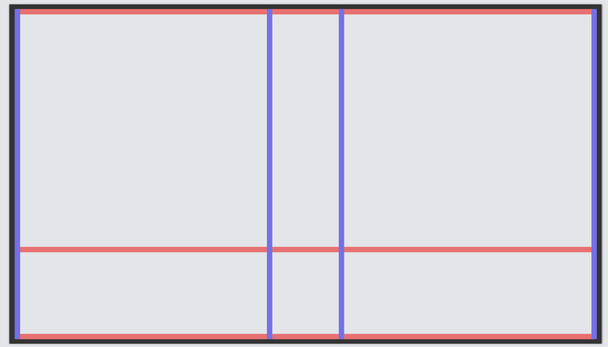
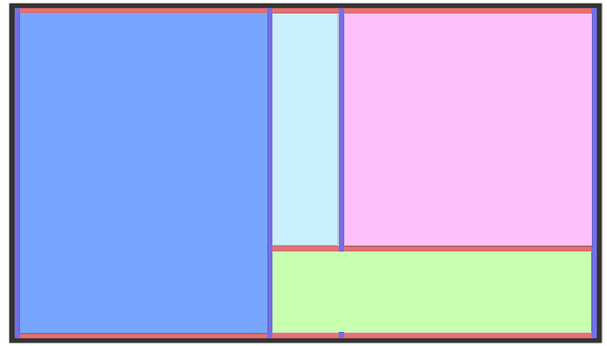
画面サイズが800px以下のときに、2行3列にする設定

```
@media (max-width: 800px) {
   .container {
      grid-template:
         "visual visual"      100vw
         "number expression" 1fr
         "other other"        auto /
         120px  1fr
   }
}
```

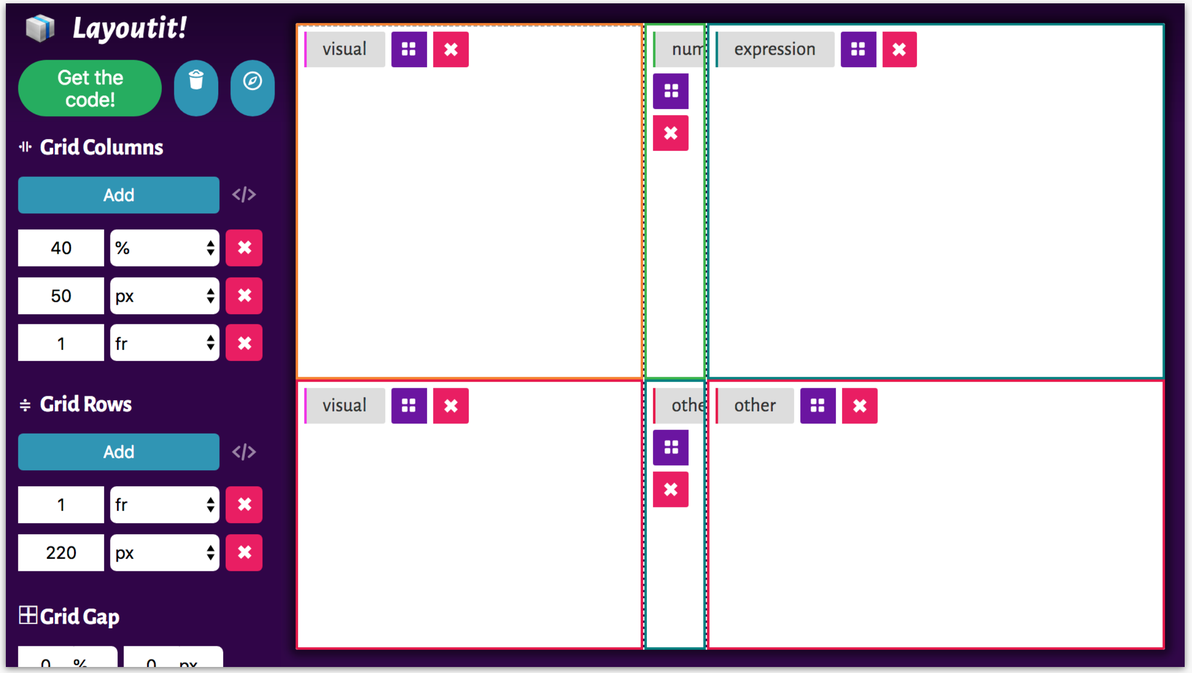

サンプル
localhost:1234
02
今日のうにちゃん
1才になった愛猫「うに」。
家の中のどこに行ってもついてくる甘えん坊。


sapmle1 [/Volumes/ICS-KANO01/Dropbox (ICS INC)/kano_ics_seminar/180929_cssnite/project/sapmle1] - .../css/style.cs...
style.css
"visual number expression"
"visual other other" 220p
40% 120px 1fr;
}
@media (max-width: 800px) {
.container {
grid-template:
"visual visual" 100vw
"number expression" 1fr
"other other" auto /
120px 1fr;
}
}
media (max-width: 800px) › .container

CSS Gridの基本的な書き方まとめ

1. コンテナを作る	display: grid	
2. 行と列を作る	grid-template プロパティ	
3. アイテムを配置する	grid-area プロパティ	

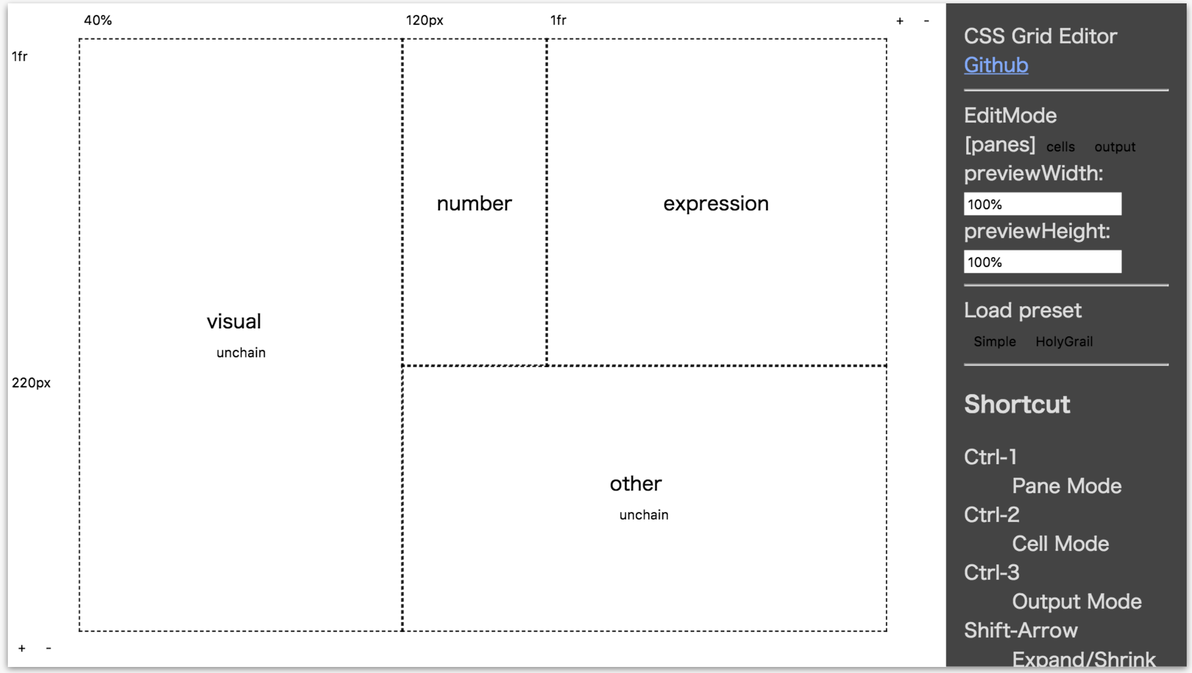
オンラインのツールを使う方法



CSS Grid Layout Generator

by Leniolabs

https://www.layoutit.com/grid



CSS Grid Layout Generator

by @mizchi

https://mizchi-sandbox.github.io/grid-generator/

もう一歩踏み込んで現場で使うCSS Grid

1. CSS Gridの基本
2. CSS Gridのオススメの使い方
3. 弊社案件における採用事例
4. 2018年はIE 11対応がラクになった
5. 未来のCSS Gridと周辺技術
6. まとめ


SOUND ON
SOUND OFF
とんこつBOY
Lv. 24
あと3240exp.
つけ麺部隊
第5編隊 先 総合力 1300
821266
125360
MENU
行動力 67/112
回復まで02:13:43
トップ
司令室
ストーリー
ガチャ
戦闘機
操縦士
編隊
クエスト
4
新着情報
999
贈り物
13
任務
交友
イベント
撃退完了
報酬受取9月29日(土) 12:59まで
戦闘リストや討伐報酬はこちら!


吹雪 舞弥
吹雪舞弥であります！　それより！　ちゃんと出されたものは全部食べるであります！
AUTO OFF
SKIP
LOG


Wave 1/1
3233
撃隊!!
撃隊!!
撃隊!!
ギギッ。
もっと強くないと、あたいとは張り合えないから！

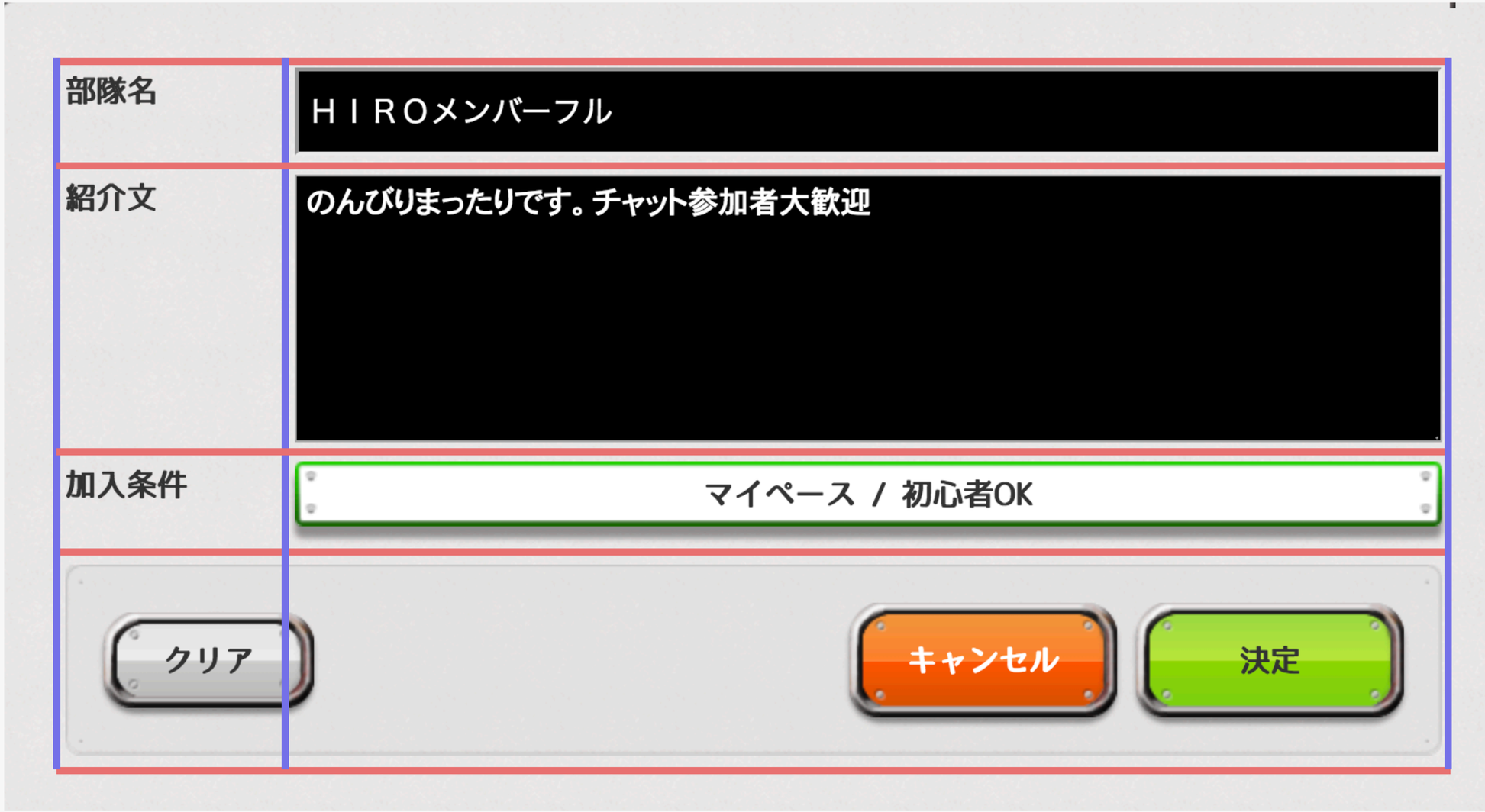
編隊少女 -フォーメーションガールズ-

- 2016年に
  株式会社アイオウプラスより
  リリースされたFlashゲーム
- 2018年夏、HTML製SPA
  としてリニューアル
- IE 11でも動作

http://henjo.jp/

採用した主な技術

CSS Gridで便利だった点1：複雑なレイアウト



CSS Gridで便利だった点1：複雑なレイアウト

grid-templateプロパティ

```
grid-template:
  "name-label          name" auto
  "introduction-label introduction-text" 220px
  "condition-label    condition-button" auto
  "button-list         button-list" auto /
  120px 1fr;
```

CSS Gridで便利だった点2：行や列同士のスペース

CSS Gridで便利だった点2：行や列同士のスペース

gapプロパティ（旧名grid-gapプロパティ）

- floatやFlexboxにはない、CSS Gridの魅力の一つ

```
.container {
  gap: 10px;
}
```

CSS Gridで便利だった点3：繰り返しの記述

CSS Gridで便利だった点3：繰り返しの記述

行や列を指定サイズで繰り返すrepeat()メソッド

- repat(繰り返し数, 繰り返しのサイズ)

```
.container {
  grid-template-rows: repeat(5, 50px);
  grid-template-columns: repeat(2, 50%);
}
```

CSS Gridを採用して苦労した点

- 自動配置がIE 11で使えなかった
- チームメンバーへの教育コストがかかった
- iOS 10.2以下をサポート外にし、
  10.3以上をサポート端末にした

もう一歩踏み込んで現場で使うCSS Grid

1. CSS Gridの基本
2. CSS Gridのオススメの使い方
3. 弊社案件における採用事例
4. 2018年はIE 11対応がラクになった
5. 未来のCSS Gridと周辺技術
6. まとめ

IE 11のサポート期間とシェア

IEのサポート期間

- Windows 10　/ IE 11 / **2025年10月まで**
- Windows 8.1　/ IE 11 / 2023年1月まで
- Windows 7　/ IE 11 / 2020年1月まで

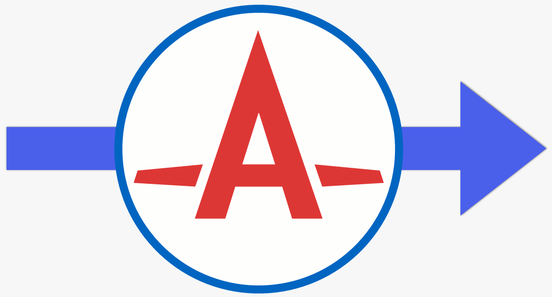
2018年現在のIE 11のシェア（ statcounter 調べ）

- **日本 11%**
- 世界3%

IE 11でCSS Gridを使おうとすると
古い記法にしか対応していないので
変換が必要

Autoprefierで自動変換

```
.container {
  display: grid;
  grid-template:
    "visual number expression" 1fr
    "visual other  other" 220px /
    40% 120px 1fr;
}

.visual {
  grid-area: visual;
}

.number {
  grid-area: number;
}

.expression {
  grid-area: expression;
}

.other {
  grid-area: other;
}
```



```
.container {
  display: -ms-grid;
  display: grid;
  -ms-grid-rows: 1fr 220px;
  -ms-grid-columns: 40% 120px 1fr;
  grid-template:
    "visual number expression" 1fr
    "visual other  other" 220px /
    40% 120px 1fr;
}

.visual {
  -ms-grid-row: 1;
  -ms-grid-row-span: 2;
  -ms-grid-column: 1;
  grid-area: visual;
}

.number {
  -ms-grid-row: 1;
  -ms-grid-column: 2;
  grid-area: number;
}

.expression {
  -ms-grid-row: 1;
  -ms-grid-column: 3;
  grid-area: expression;
}

.other {
  -ms-grid-row: 2;
  -ms-grid-column: 2;
  -ms-grid-column-span: 2;
  grid-area: other;
}
```

2017年冬〜2018年に AutoprefixerのIE 11向け変換が 大きくパワーアップ

エリア名の変換

2017年12月リリースのv7.2〜

grid-templateは2017年にはおすすめできなかった

IE 11の存在があったから

- エリア名の指定に未対応なので、番号指定が必要
- 番号指定はエリアの位置がイメージしづらく、
  レイアウト変更時に書き換えが大変

```
.container {
 grid-template-rows: 1fr 220px;
 grid-template-columns: 40% 120px 1fr;
}
```

行と列の定義

```
.main-visual {
  grid-row-start: 1;
  grid-row-end: 3;
  grid-column-start: 1;
  grid-column-end: 2;
  -ms-grid-row-span: 2;
}
```

アイテムの配置

AutoprefixerでIE11対応コードに自動変換

before

```
.container {
  display: grid;
  grid-template:
    "visual number expression" 1fr
    "visual other  other" 220px /
    40% 120px 1fr;
}

.visual {
  grid-area: visual;
}
```

AutoprefixerでIE11対応コードに自動変換

after

```
.container {
  display: -ms-grid;
  display: grid;
  -ms-grid-rows: 1fr 220px;
  -ms-grid-columns: 40% 120px 1fr;
  grid-template:
    "visual number expression" 1fr
    "visual other  other" 220px /
    40% 120px 1fr;
}

.visual {
  -ms-grid-row: 1;
  -ms-grid-row-span: 2;
  -ms-grid-column: 1;
  grid-area: visual;
}
```

gapプロパティの変換

2018年6月リリースのv8.6〜

gapプロパティは2017年にはおすすめできなかった

IE 11の存在があったから

- gapに未対応
- marginやpaddingで無理やり間隔を調整していた

```
.number,
.expression,
.other {
  padding: 10px;
}
```

AutoprefixerでIE11対応コードに自動変換

before

```
.container {
  display: grid;
  grid-template:
    "visual number expression" 1fr
    "visual other  other" 220px /
    40% 120px 1fr;
}

.visual {
  grid-area: visual;
}
```

AutoprefixerでIE11対応コードに自動変換

after

```
.container {
  /* 中略 */

  gap: 10px;
  -ms-grid-rows: 1fr 10px 220px;
  -ms-grid-columns: 40% 10px 120px 10px 1fr;
  grid-template:
    "visual number  expression" 1fr
    "visual other   other"      220px /
    40%     120px   1fr;
}

.visual {
  -ms-grid-row: 1;
  -ms-grid-row-span: 3;
  -ms-grid-column: 1;
  grid-area: visual;
}
```

Autoprefixerの使い方

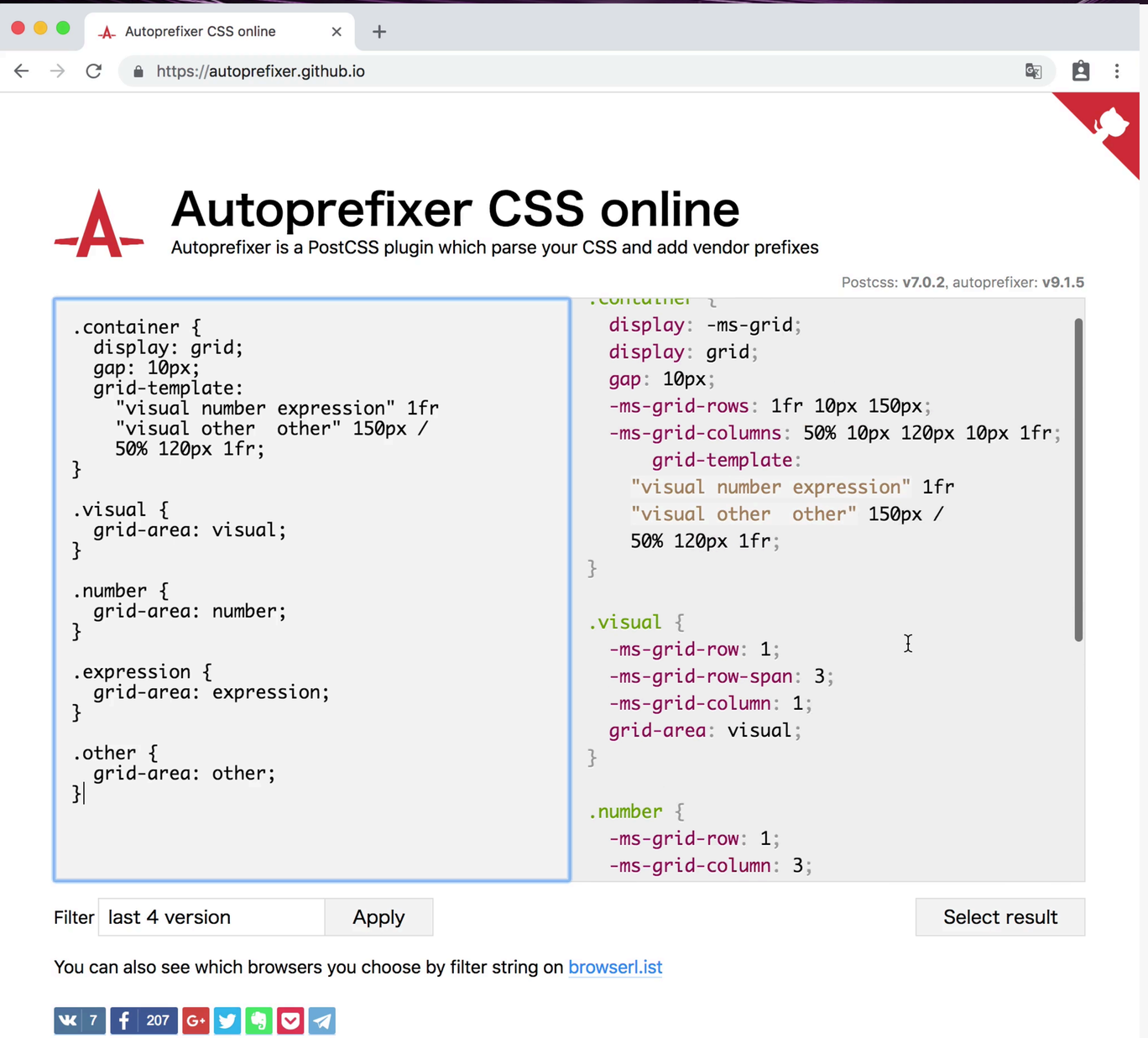
各ツールと連携する方法

最新版で学ぶwebpack 4入門 - スタイルシート取り込む方法
https://ics.media/entry/17376

CSSベンダープレフィックスを今この瞬間に辞める為Autoprefixerの導入
https://qiita.com/tonkotsuboy_com/items/377913c51b1ac00deffe

独自設定ファイルは不要。Parcelでコンパイルする SassとAutoprefixer
https://qiita.com/tonkotsuboy_com/items/2f96263294fad7661a82

オンラインツールで自動変換する方法



Autoprefixer CSS online

- 環境構築不要

https://autoprefixer.github.io/

多くのCSS GridプロパティをIE 11で変換できる

プロパティ	IE 11	Autoprefixer
grid-template	×	○
grid-template-areas	×	○
grid-template-columns grid-template-rows	○※	○
grid-column grid-row	○※	○
grid-row-start grid-column-start	○※	○
grid-row-end grid-column-end	×	○

プロパティ	IE 11	Autoprefixer
grid-auto-flow	×	×
grid-auto-columns grid-auto-rows	×	×
gap	×	○
column-gap row-gap※	×	○
align-self	○※	○
justify-self	○※	○

※ 記法が異なる

もう一歩踏み込んで現場で使うCSS Grid

1. CSS Gridの基本
2. CSS Gridのオススメの使い方
3. 弊社案件における採用事例
4. 2018年はIE 11対応がラクになった
5. 未来のCSS Gridと周辺技術
6. まとめ

未来のCSS Gridと周辺技術

1. アイテムの自動配置と敷き詰め
2. セマンティックなコーディングとCSS Grid
3. さらに複雑なレイアウトのために

アイテムの自動配置と敷き詰め

grid-autoflowプロパティ

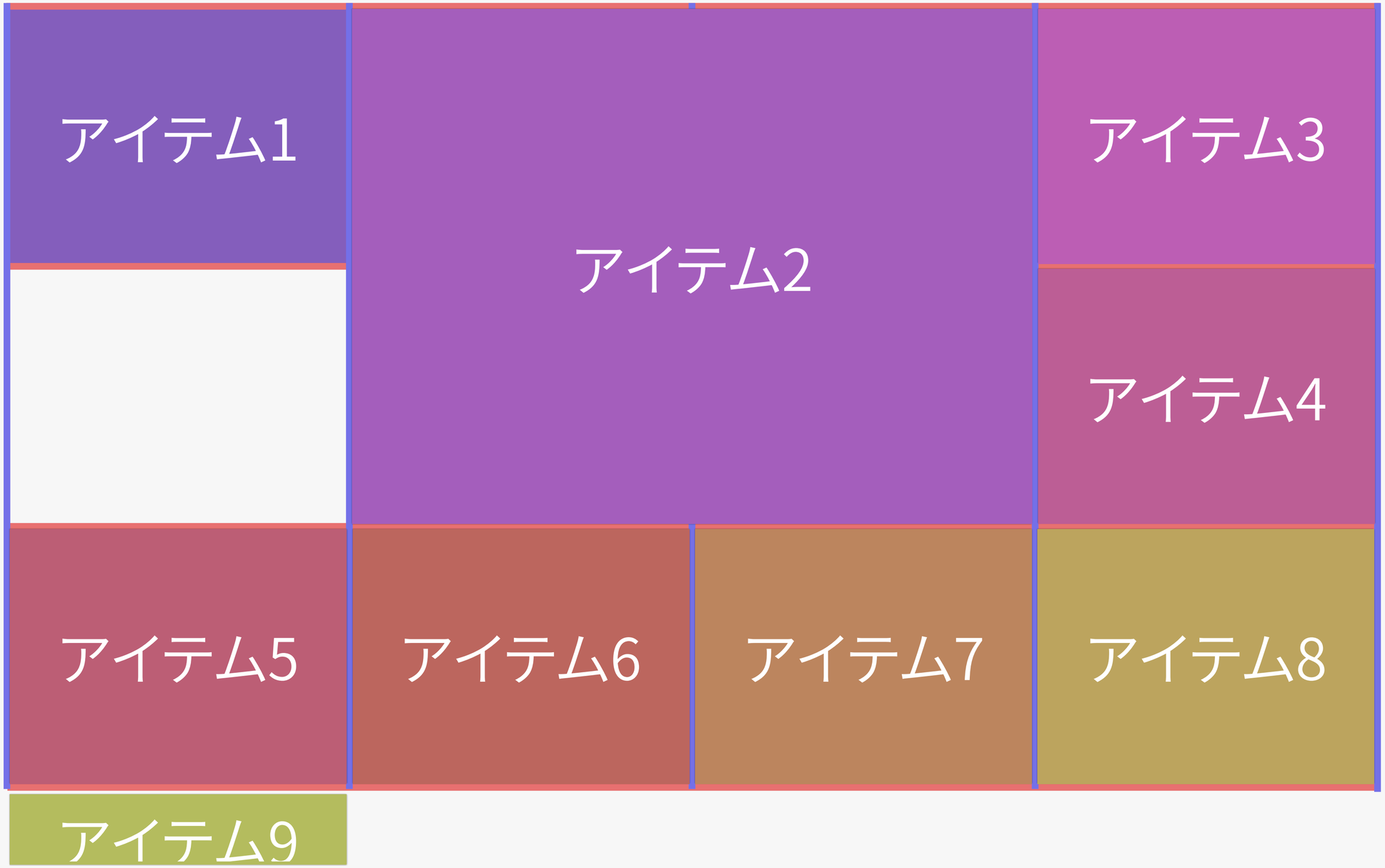
CSS Gridでは、アイテムは左上から自動的に配置される

アイテム1	アイテム2	アイテム3	アイテム4
アイテム5	アイテム6	アイテム7	アイテム8
アイテム9	アイテム10	アイテム11	アイテム12

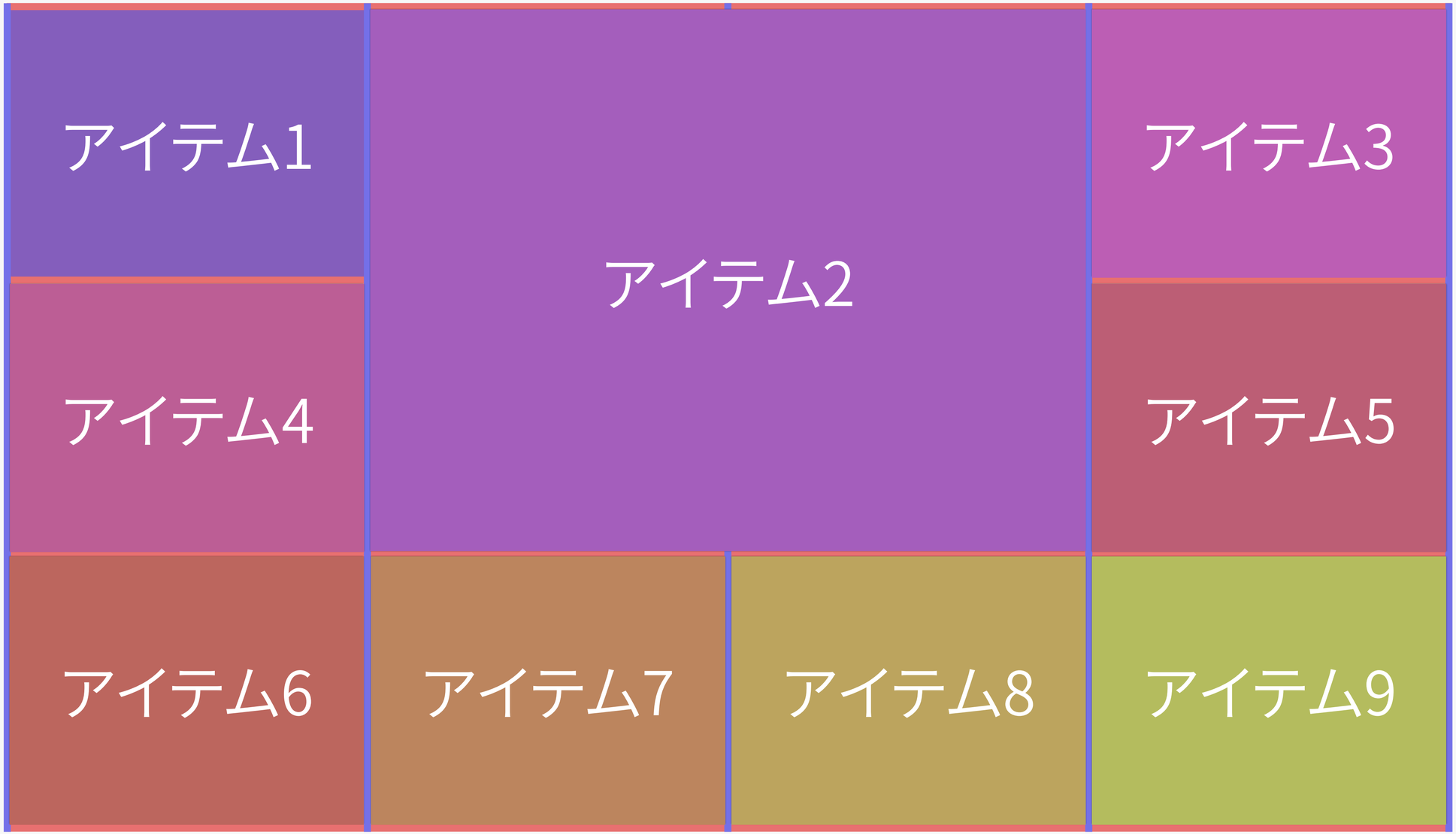
grid-autoflow: columnで列順に

アイテム1	アイテム4	アイテム7	アイテム10
アイテム2	アイテム5	アイテム8	アイテム11
アイテム3	アイテム6	アイテム9	アイテム12

大きいアイテムがあると隙間があく



grid-autoflow: denseで敷き詰める



grid-autoflowプロパティの対応状況

							
ブラウザ名	Chrome	Firefox	Safari	Edge	Internet Explorer	Safari (iOS版)	Chrome (Android版)
最新バージョン	69	62	12	17	11	12	69
Grid対応	○	○	○	○	×	○	○

2018/09/29 現在

セマンティックなコーディングと CSS Grid

CSS Gridを使いつつ、情報の意味付けをしたい

大見出し	
中見出し1	中見出し2
テキスト1	テキスト3
テキスト2	テキスト4

CSS Gridを使いつつ、情報の意味付けをしたい

理想のHTML

```
<div class="container">
  <h1>大見出し</h1>
  <section>
    <h2>中見出し</h2>
    <p>テキスト1</p>
    <p>テキスト2</p>
  </section>
  <section>
    <h2>中見出し</h2>
    <p>テキスト3</p>
    <p>テキスト4</p>
  </section>
</div>
```

CSS

```
.container {
  grid-template:
    "header header" auto
    "title1 title2" 100px
    "text1  text3 " 1fr
    "text2  text4 " 40px /
    50%     50%;
}
```

アイテムはすべて並列に並べなければいけない

現実のHTML

```
<div class="container">
  <h1>大見出し</h1>
  <h2>中見出し</h2>
  <p>テキスト1</p>
  <p>テキスト2</p>
  <h2>中見出し</h2>
  <p>テキスト3</p>
  <p>テキスト4</p>
</div>
```

CSS

```
.container {
  grid-template:
    "header header" auto
    "title1 title2" 100px
    "text1  text3 " 1fr
    "text2  text4 " 40px /
    50%     50%;
}
```

display: contentsで解決

子要素（または疑似要素）によって置換されるボックス

```
section {
  display: contents;
}
```

Gridを使いつつ、セマンティックなコーディングが実現

HTML

```
<div class="container">
  <h1>大見出し</h1>
  <section>
    <h2>中見出し</h2>
    <p>テキスト1</p>
    <p>テキスト2</p>
  </section>
  <section>
    <h2>中見出し</h2>
    <p>テキスト3</p>
    <p>テキスト4</p>
  </section>
</div>
```

CSS

```
.container {
  grid-template:
    "header header" auto
    "title1 title2" 100px
    "text1  text3 " 1fr
    "text2  text4 " 40px /
    50%     50%;
}

section {
  display: contents;
}
```

display: contentsの対応状況

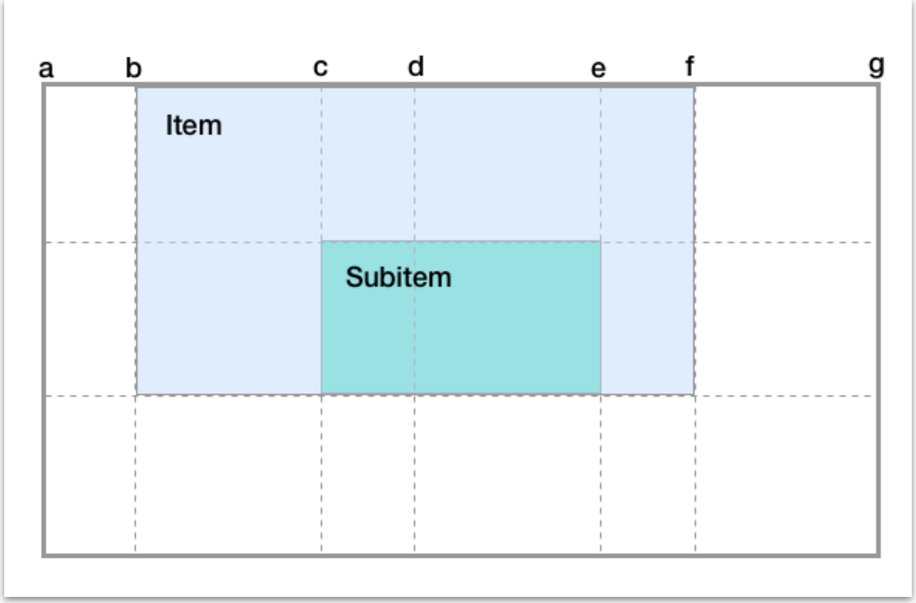
							
ブラウザ名	Chrome	Firefox	Safari	Edge	Internet Explorer	Safari (iOS版)	Chrome (Android版)
最新バージョン	69	62	12	17	11	12	69
Grid対応	△	○	△	×	×	△	△

2018/09/29 現在

さらに複雑なレイアウト

サブグリッド機能

- 親の行・列を用いて入れ子のグリッドを配置する機能
- CSS Grid Level 2で仕様策定中



「CSS Grid Level 2: Here Comes Subgrid」 より

さらに複雑なレイアウト

CSS Layout API

- JavaScriptで作成したレイアウトを、CSSから使える機能
- CSS Houdini（フーディーニ）の一種

```
.box {
  /* my-layout-logicは、  */
  /* JavaScriptで定義したレイアウト */
  display: layout(my-layout-logic);
}
```

もう一歩踏み込んで現場で使うCSS Grid

1. CSS Gridの基本
2. CSS Gridのオススメの使い方
3. 弊社案件における採用事例
4. 2018年はIE 11対応がラクになった
5. 未来のCSS Gridと周辺技術
6. まとめ

まとめ

1. 無駄な入れ子を作らずに2次元のレイアウトができる
2. エリア名を使った書き方がオススメ
3. IE 11対応案件でも安心して使える
4. 未来のCSS Gridには、便利な機能が待っている

CSS Gridを今日から使おう!

ご清聴ありがとうございました

ICS MEDIAやTwitterでもウェブテクノロジーの情報を発信中

ics.media

鹿野 壮
@tonkotsuboy_com