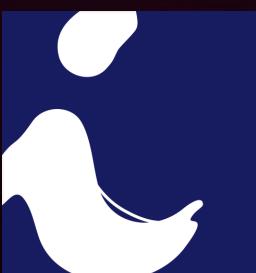


Coders
High
2018

2018.9.29

もう一步踏み込んで 現場で使うCSS Grid



株式会社ICS 鹿野 壮

自己紹介

ICS MEDIA

記事を検索 ブッシュ通知 f t n

ics.media
インタラクションデザイン × ウェブテクノロジー
最先端のリッチ表現やイノベーション、
制作に役立つ新しいトピックスを紹介する ICS INC. による次世代メディア

ウェブ制作 3D表現 現場で使えるフロントエンド解説 デザイン アプリ開発

話題の記事

現場で使えるフロントエンド解説 Visual Studio Codeを使いこなせ! フロントエンジニア必須の拡張機能7選

渡邊 2018.07.03 ❤ 2448

ウェブ制作 若い世代が知らない2000年代のHTMLコーディングの地獄

池田 2018.05.17 ❤ 3502

ウェブ制作 2018年に見直した現代的なJavaScriptの記法を紹介するぜ

鹿野 2018.02.22 ❤ 2078

デザイン 人工知能Adobe Senseiの画像処理が凄すぎてAdobe MAX 2017の会場は狂喜乱舞に

池田 2017.10.20 ❤ 4965

新着記事

記事を検索...

人気のワード : CSS Grid Adobe XD webpack gulp Electron React WebGL Unity

Adobe XDとPhotoshopのアセット連携が便利! 快適なデザイン制作フローを実現しよう

加賀 2018-09-26 ❤ 103

オフスクリーンキャンバスを使ったJSでのマルチスレッド描画 -スムーズなユーザー操作実現の切り札

川勝 2018-09-18 ❤ 404

Chromeの最新機能が楽しいぞ! CSSで円グラフや集中線が描けるconic-gradient入門

鹿野 2018-09-11 ❤ 406

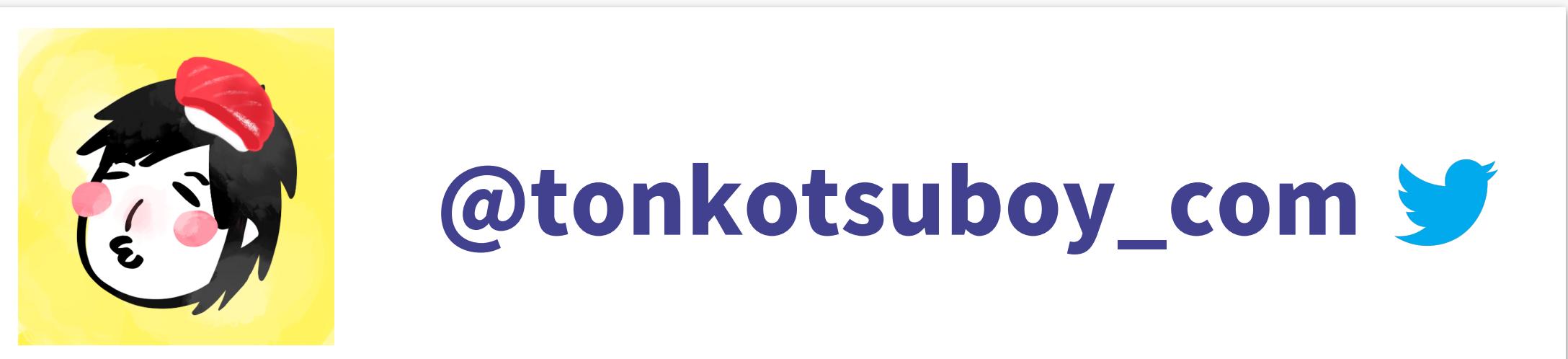
CEDEC2018発表資料「編隊少女 -フォーメーションガールズ-」における3Dレンダリング技術解説 Babylon.js と BISHAMON WebGL版の合成

鈴木 2018-09-07 ❤ 60

ICS MEDIAは週1~2回の頻度でウェブ制作の記事を公開。SNSで新着記事をチェックしよう!

RSS f t n ブッシュ通知

鹿野 壮 (かの たけし)



もう一歩踏み込んで現場で使うCSS Grid

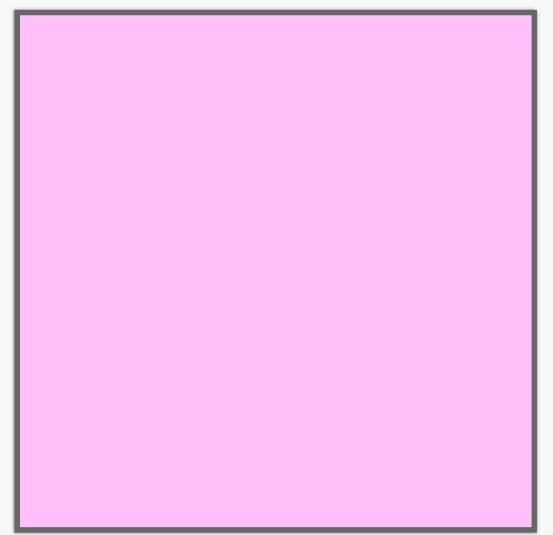
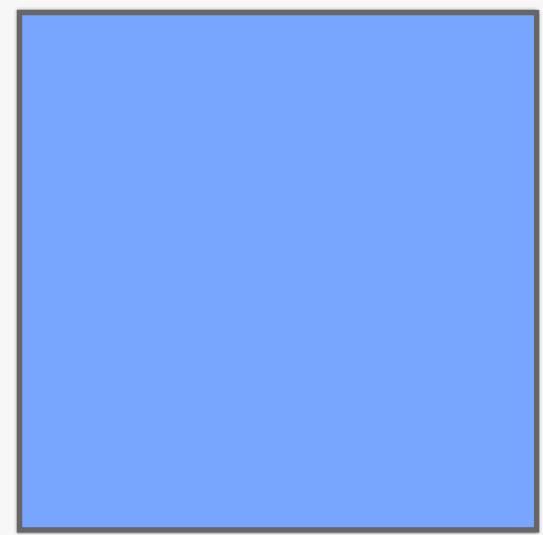
1. CSS Gridの基本
2. CSS Gridのオススメの使い方
3. 弊社案件における採用事例
4. 2018年はIE11対応がラクになった
5. 未来のCSS Gridと周辺技術
6. まとめ

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



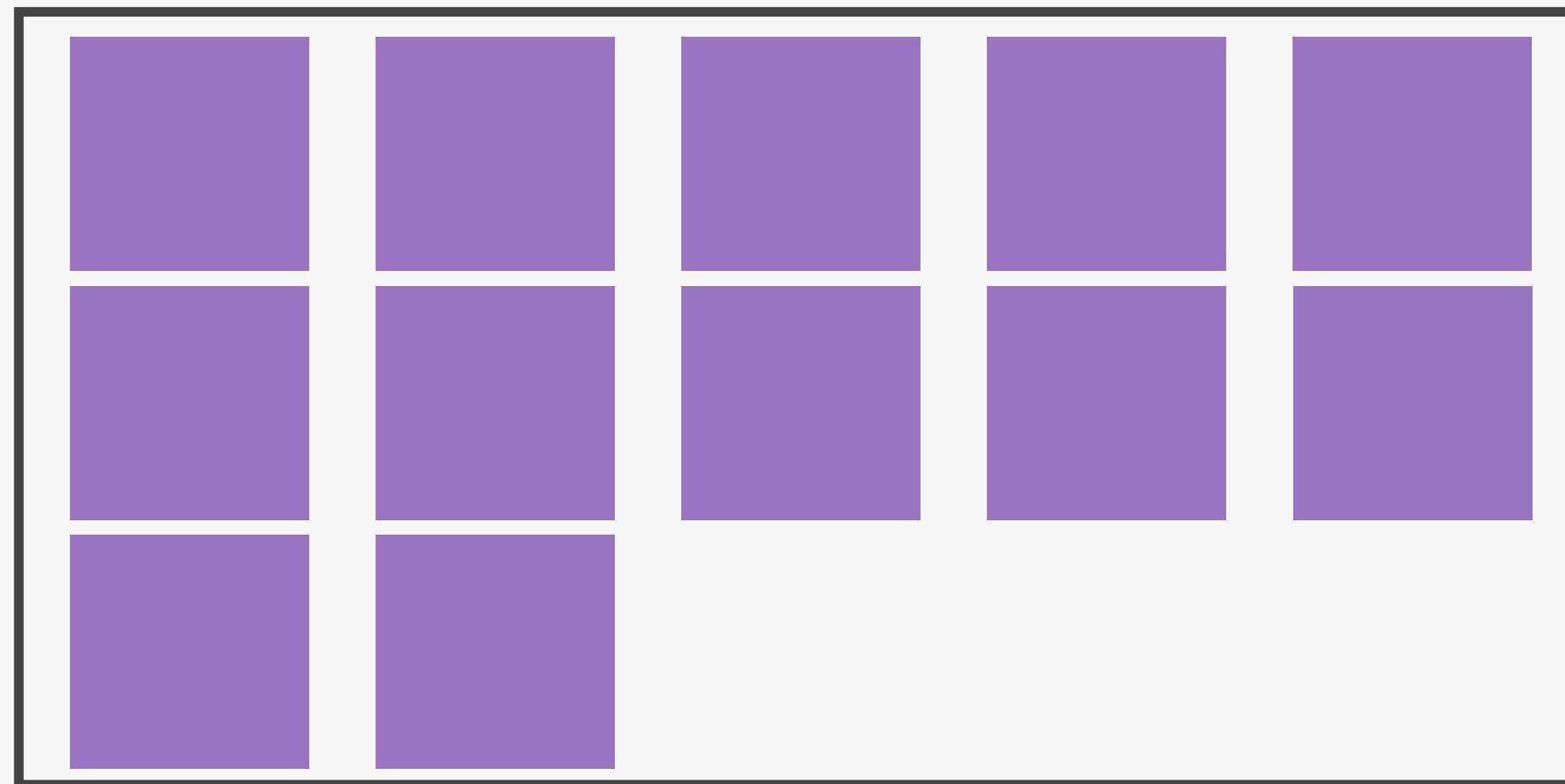
アイテム アイテム アイテム

コンテナ

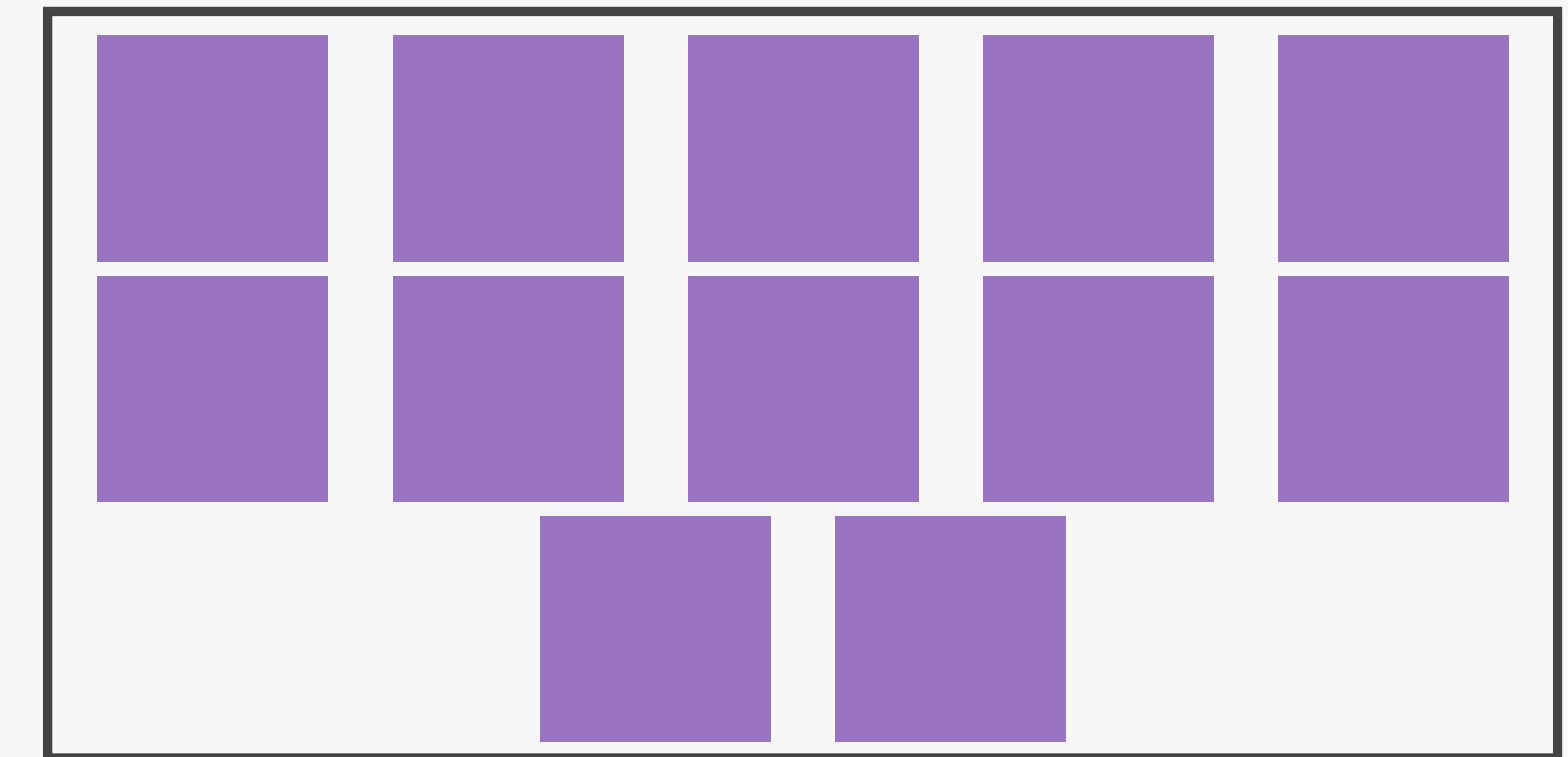
**Flexboxは
複数行のレイアウトが苦手**

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



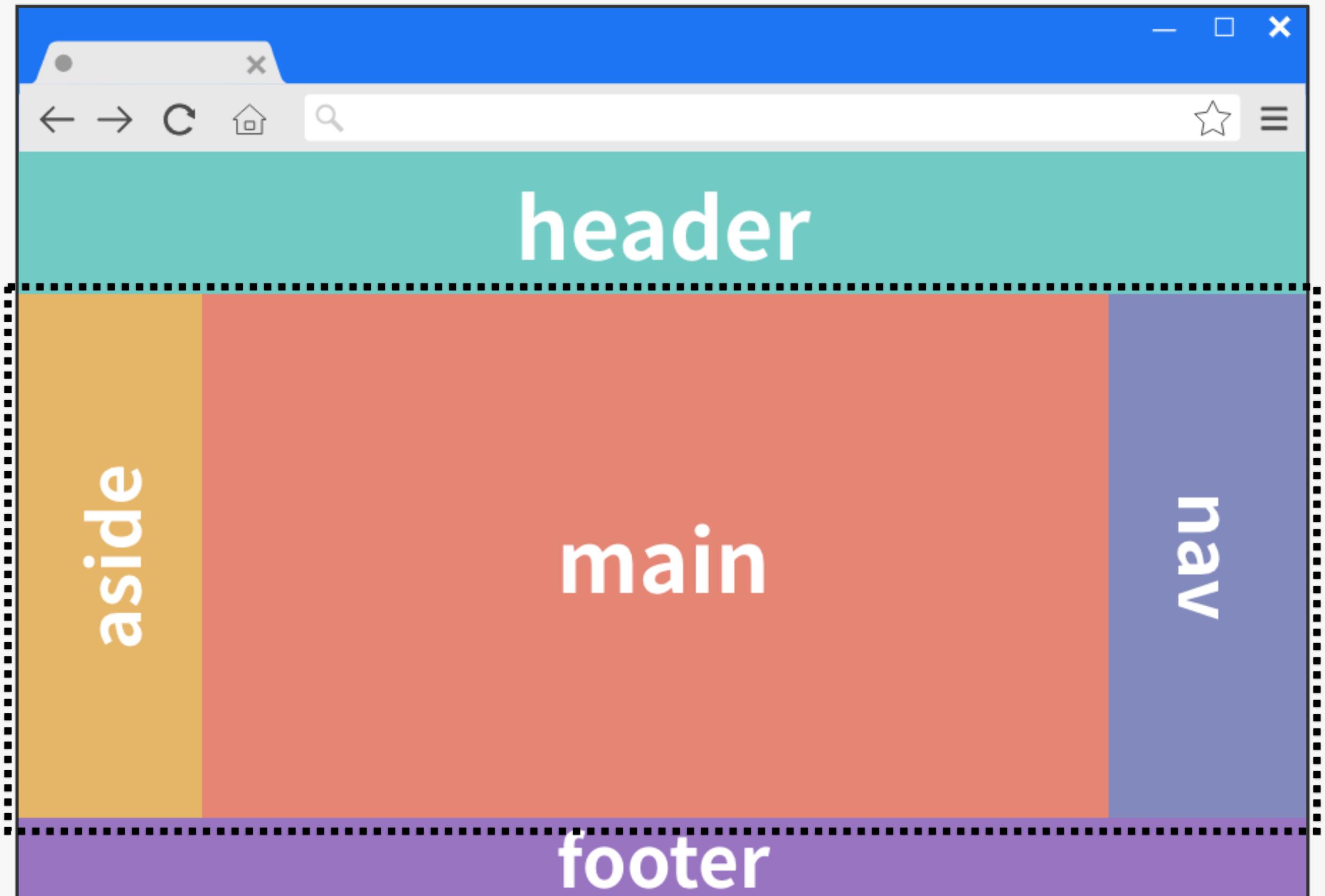
理想



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

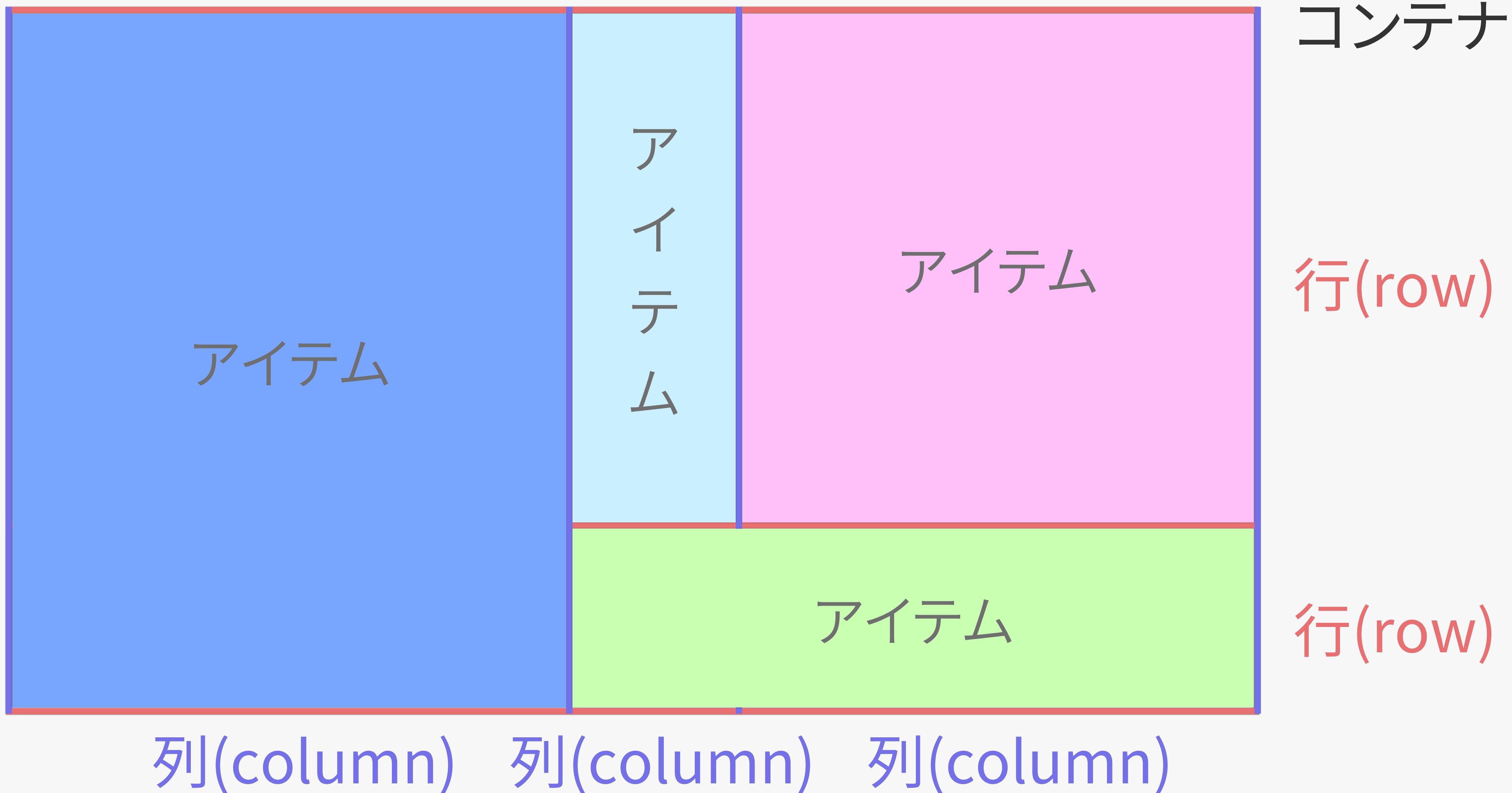
レイアウトのために入れ子が増える



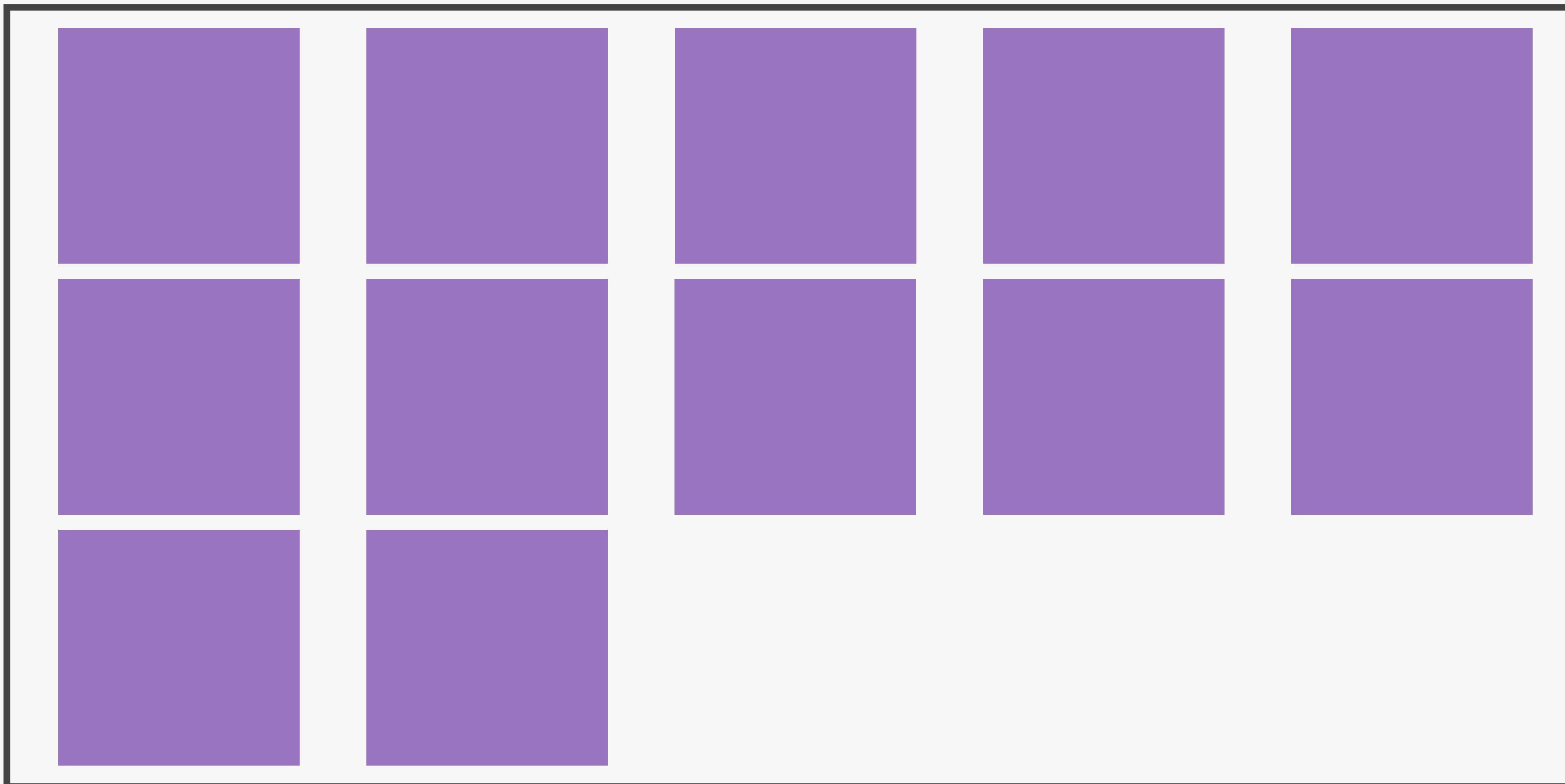
```
<div class="container">
  <header>header</header>
  <div class="middle">
    <aside>aside</aside>
    <main>main</main>
    <nav>nav</nav>
  </div>
  <footer>footer</footer>
</div>
```

**複数行のレイアウトに強い
CSS Gridが登場**

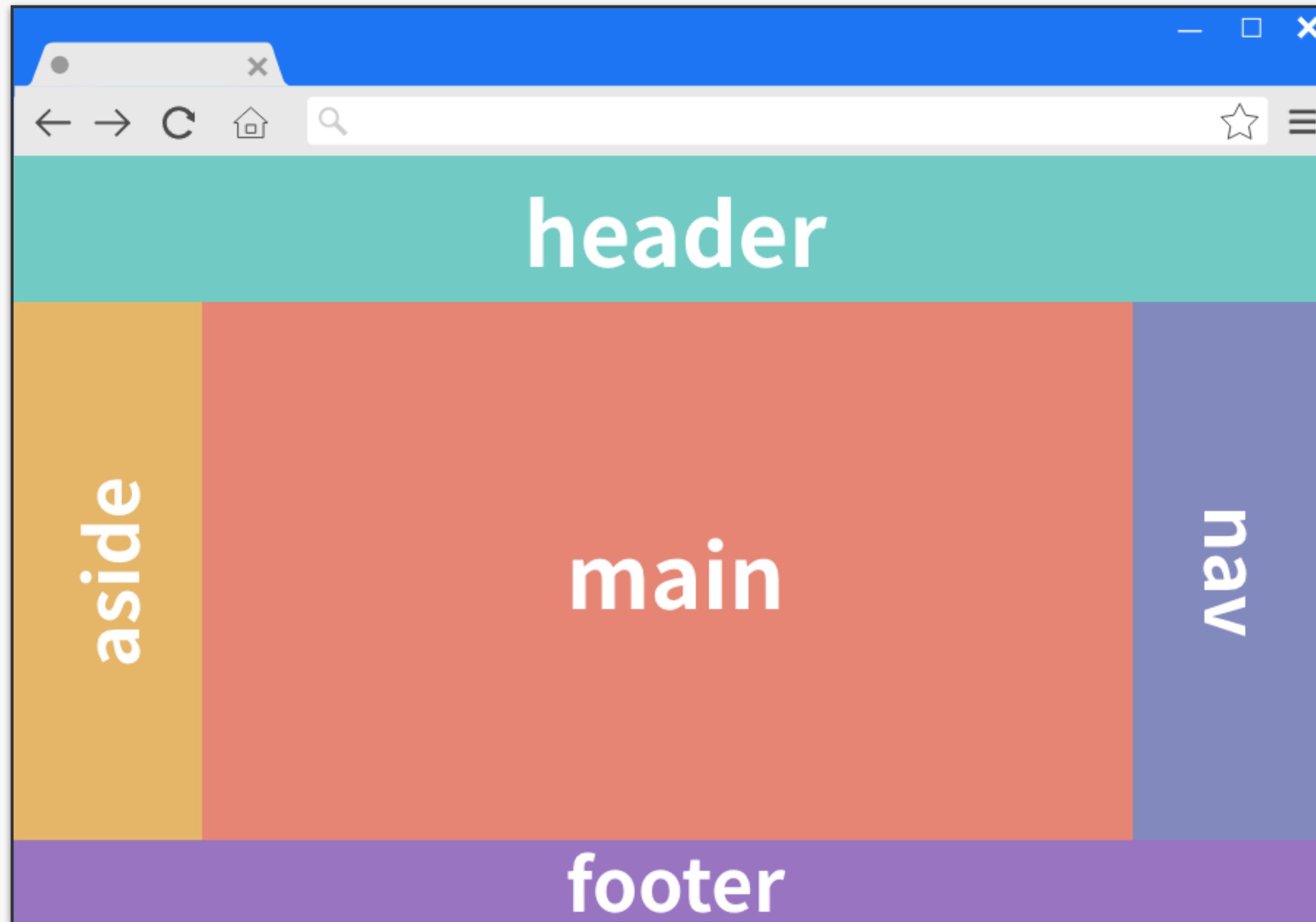
CSS Grid: 行と列を使ったレイアウト



複数行レイアウトの最終行が揃う

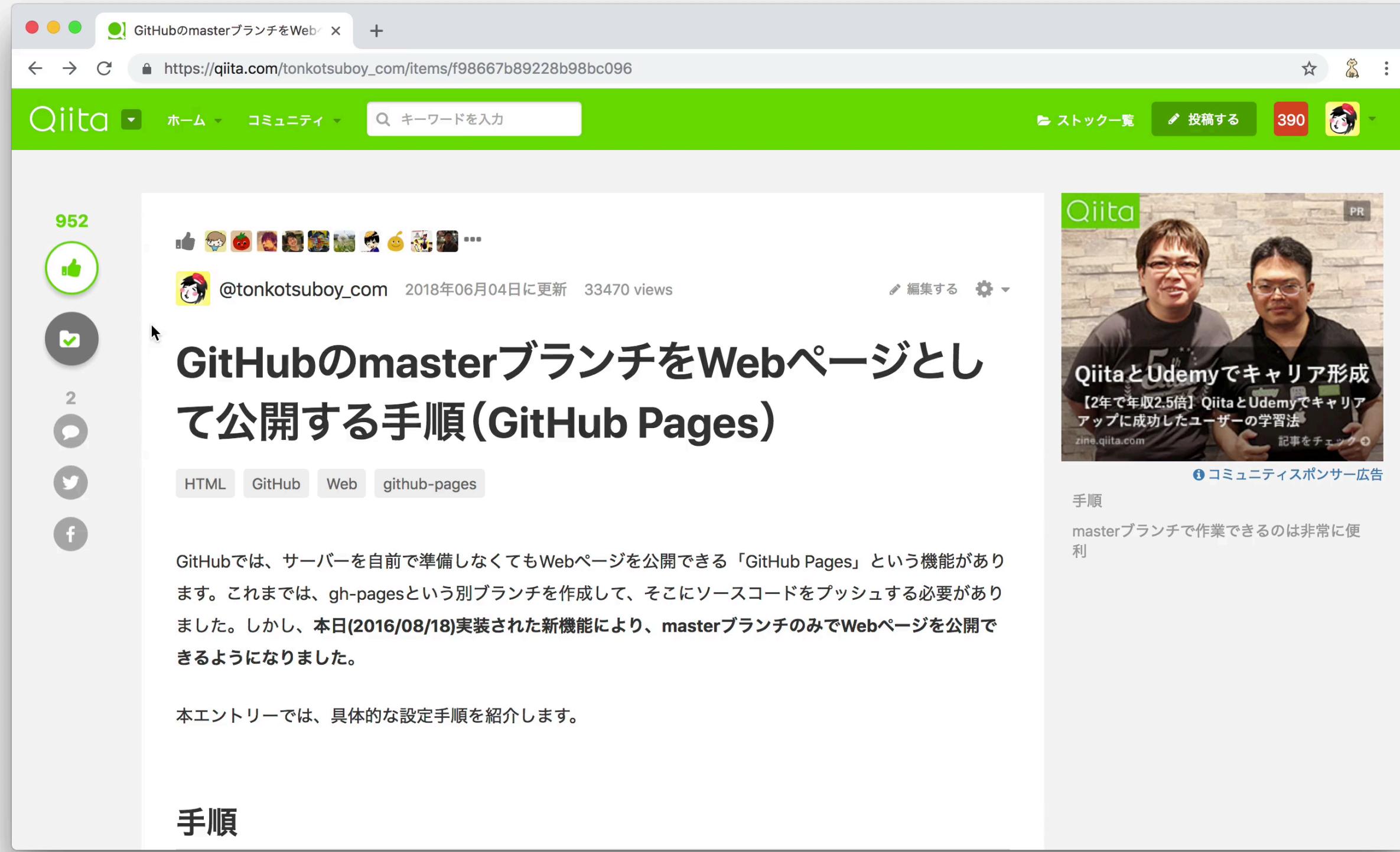


レイアウトのためのHTMLの入れ子は増えない



```
<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

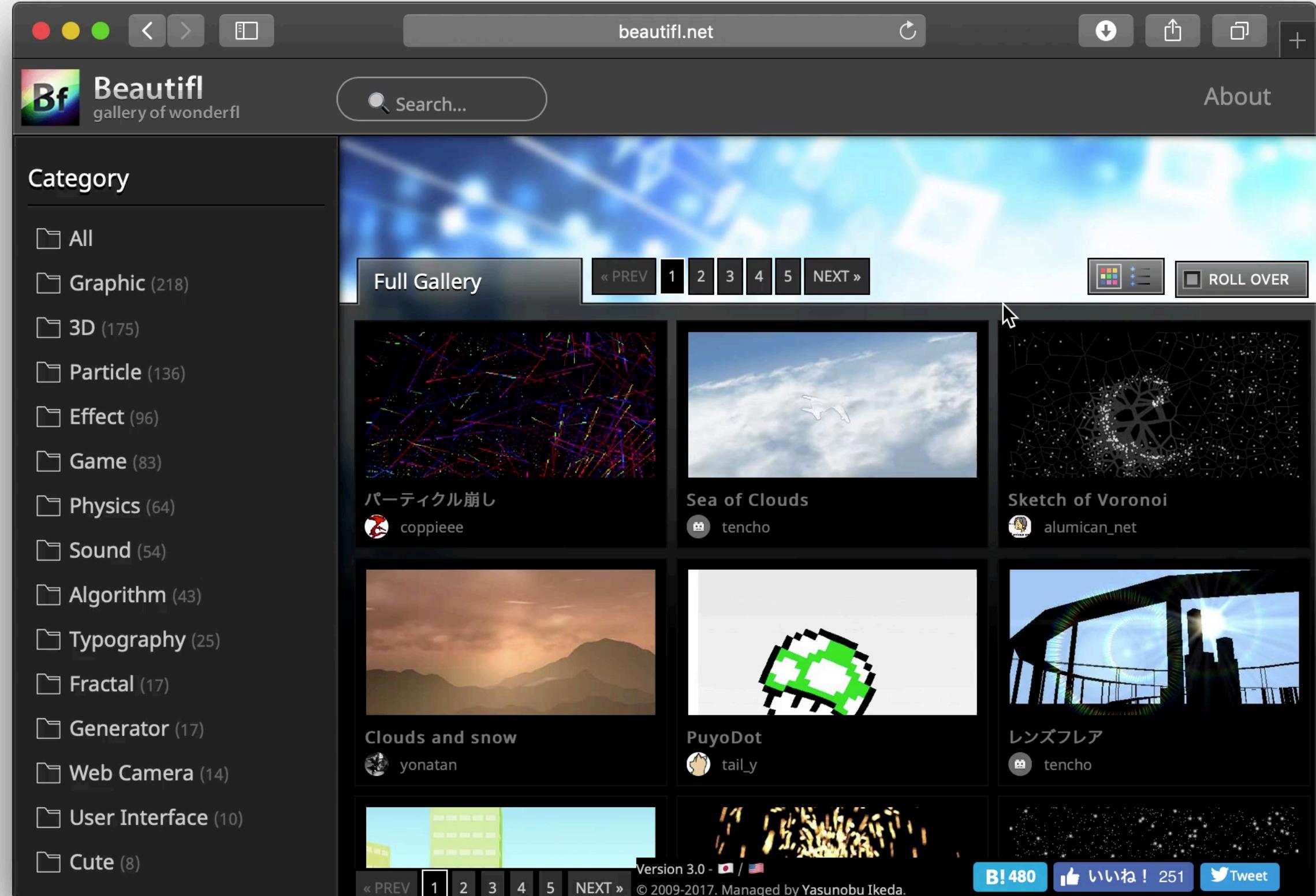


Qiita

<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

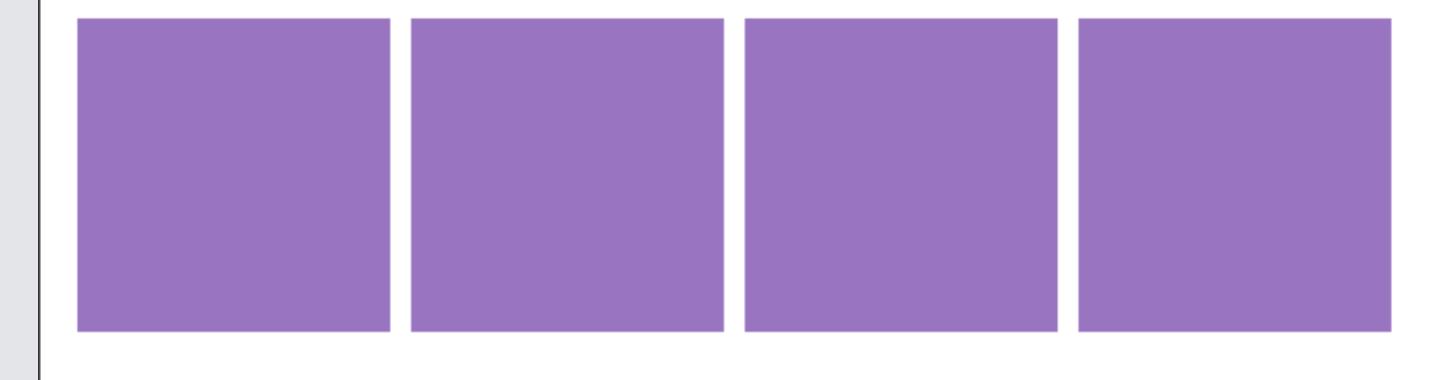
テキストの回り込み

float



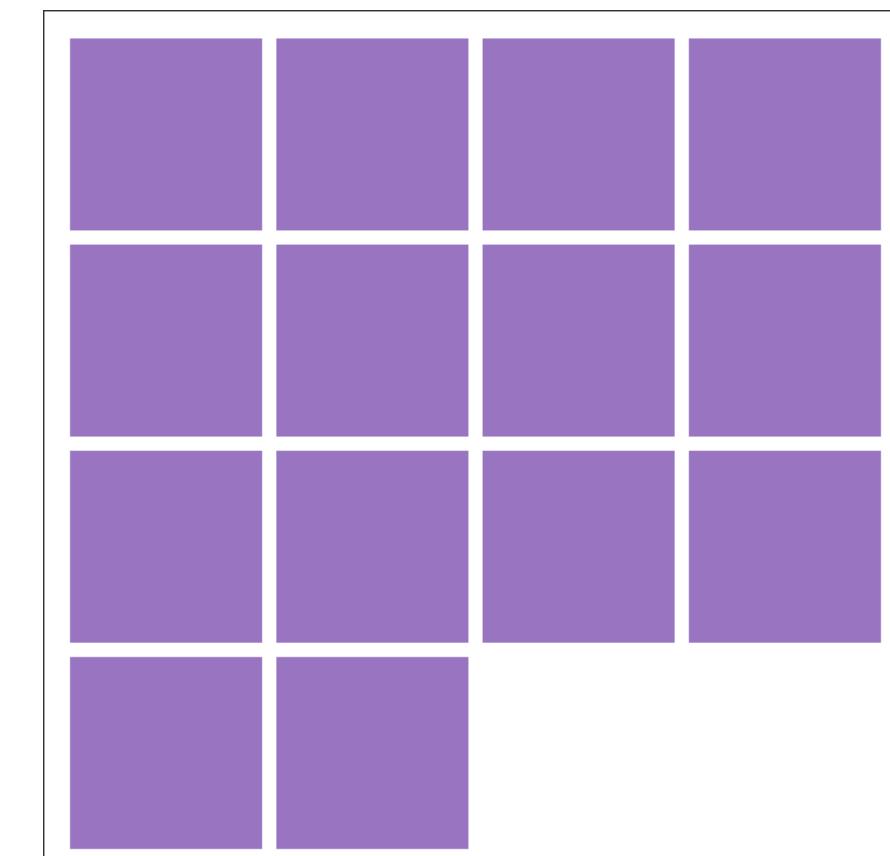
1行の横並び、縦並び

Flexbox

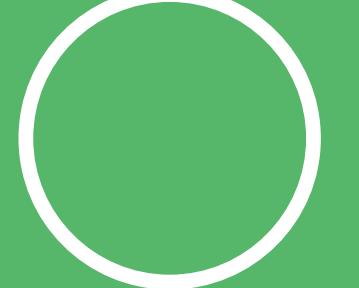
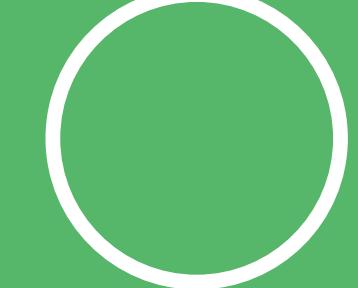
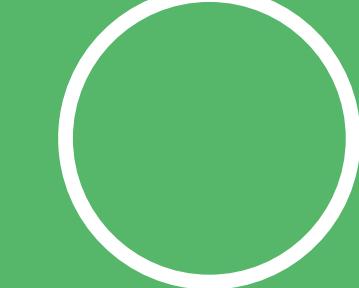
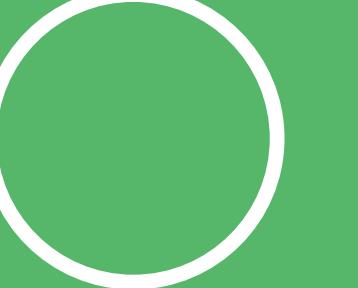
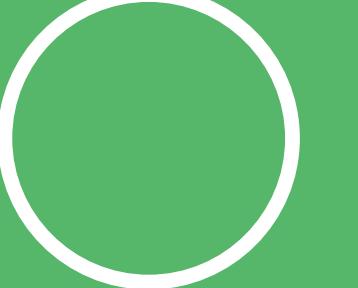


格子状のレイアウト
ページ全体のレイアウト

Grid

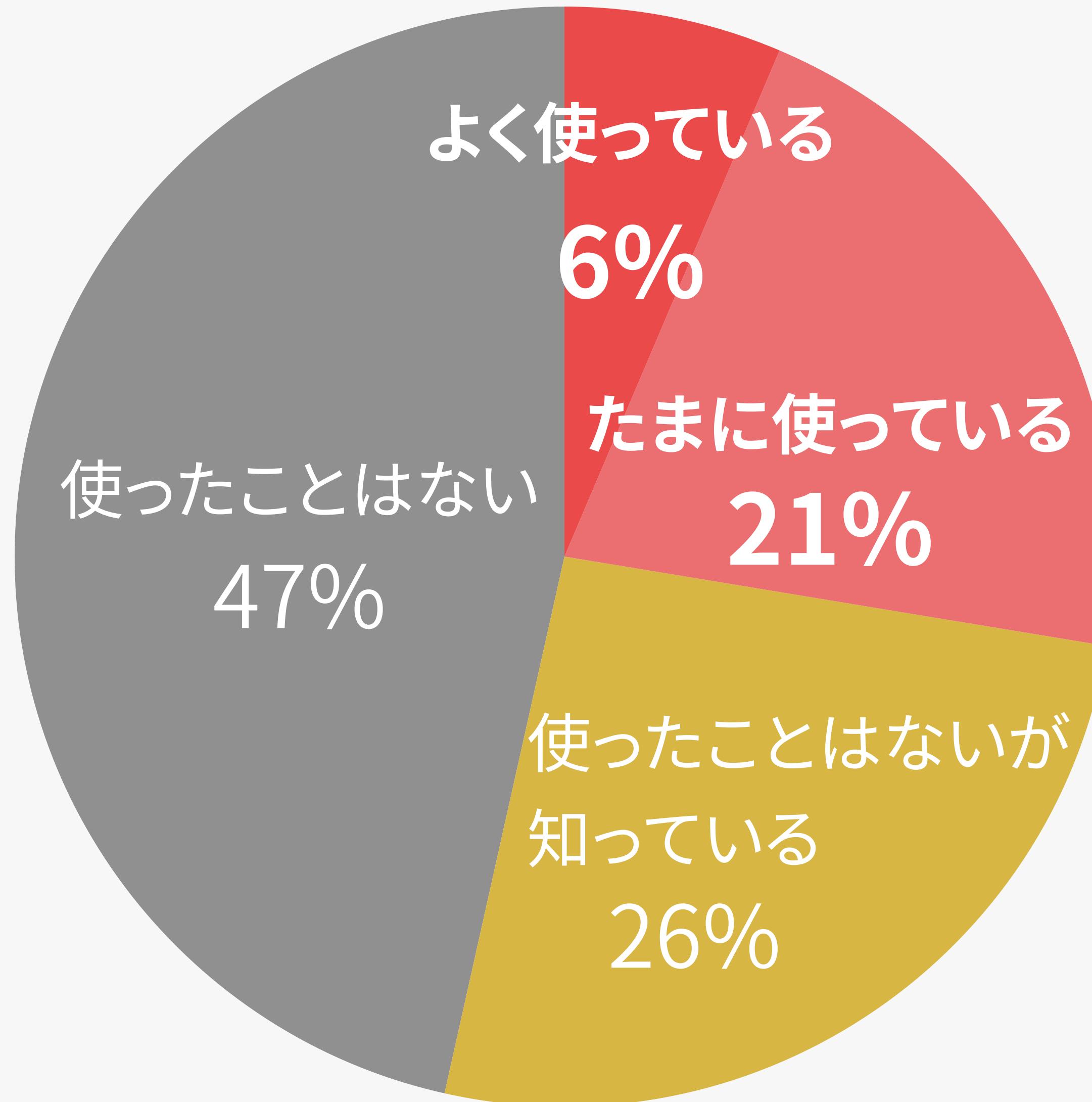


CSS Gridの対応状況

							
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IE11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

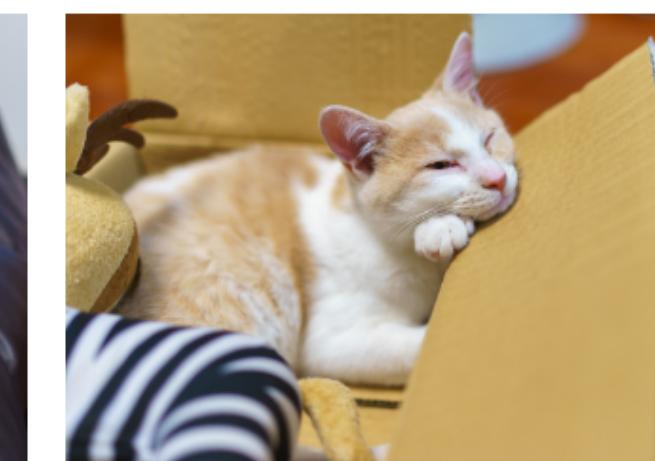
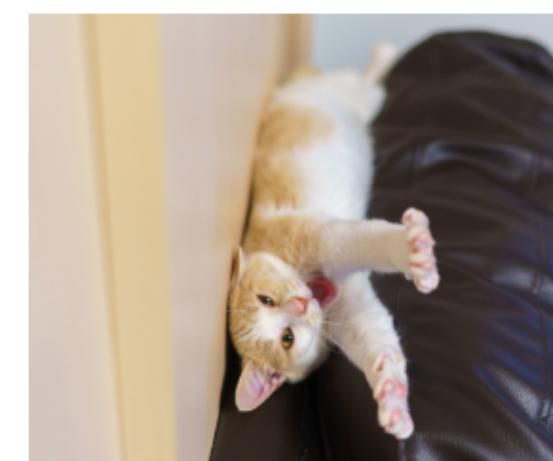


02

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`



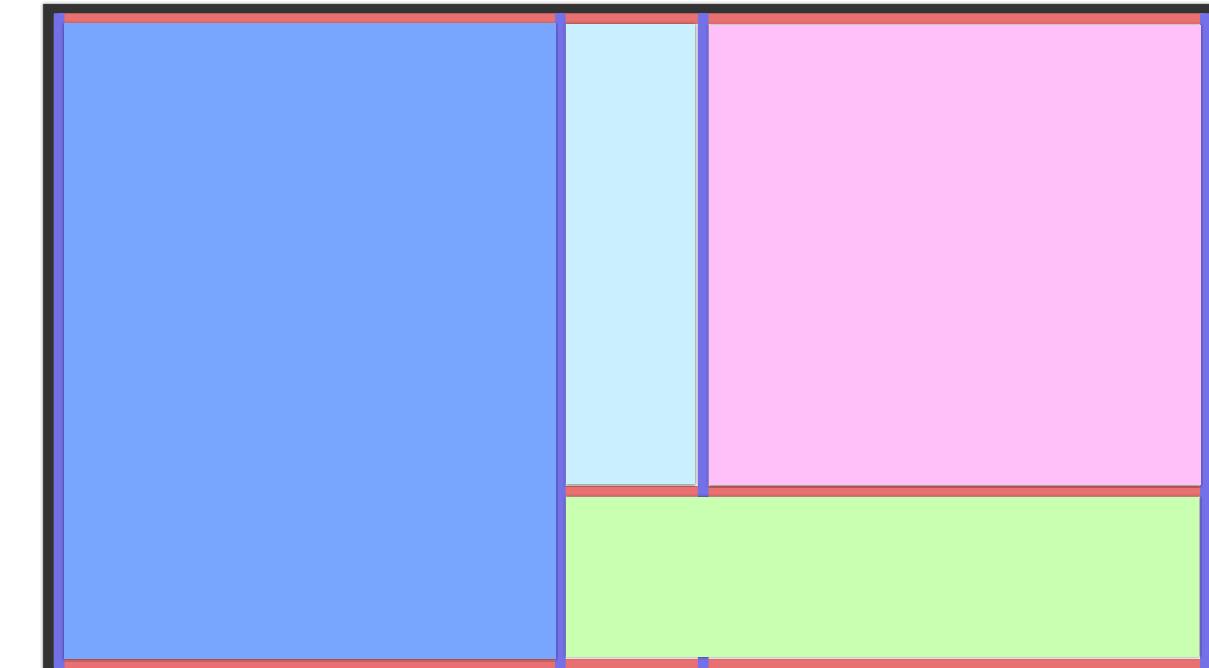
2. 行と列を作る

`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

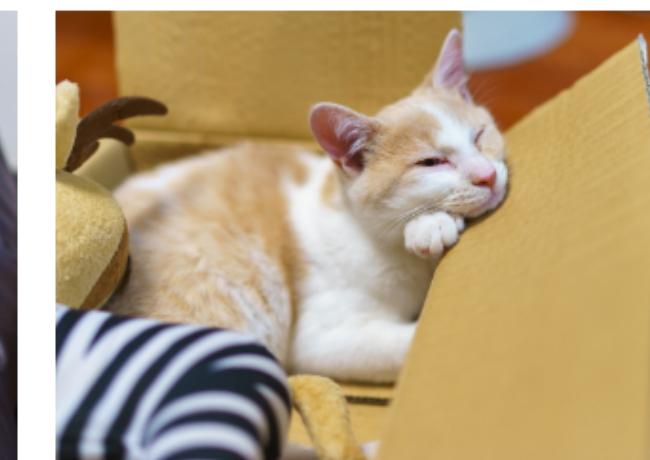
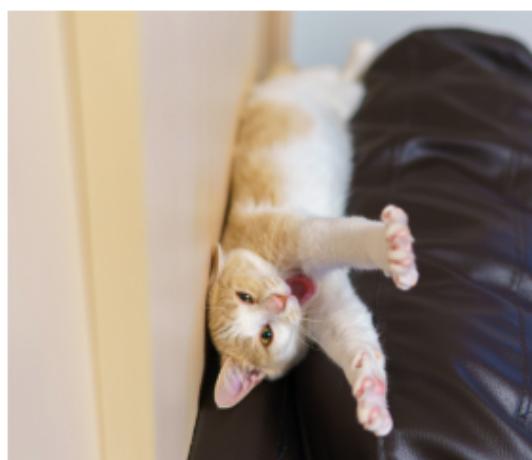


02

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。



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)

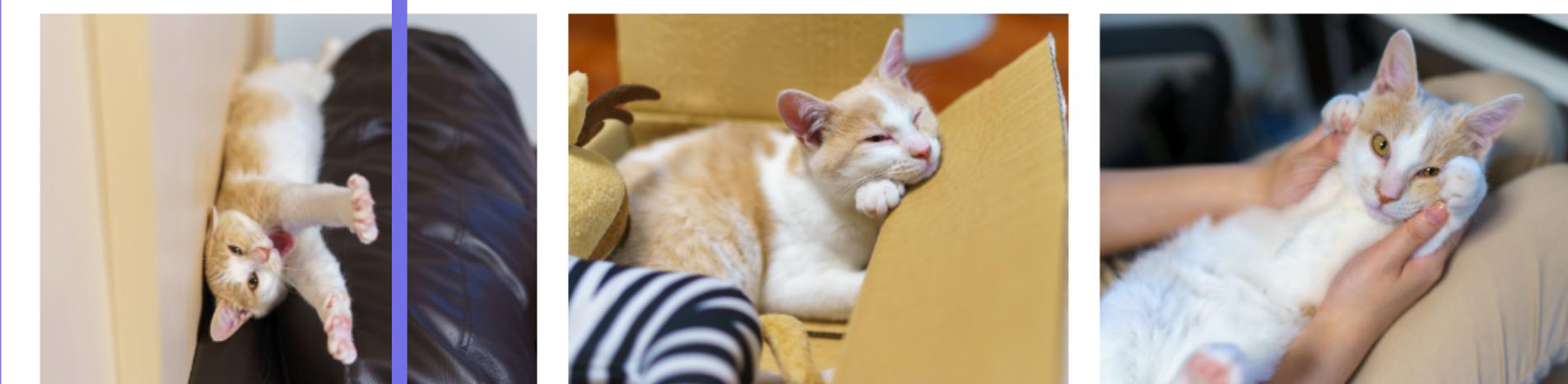


02

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。



行(row)

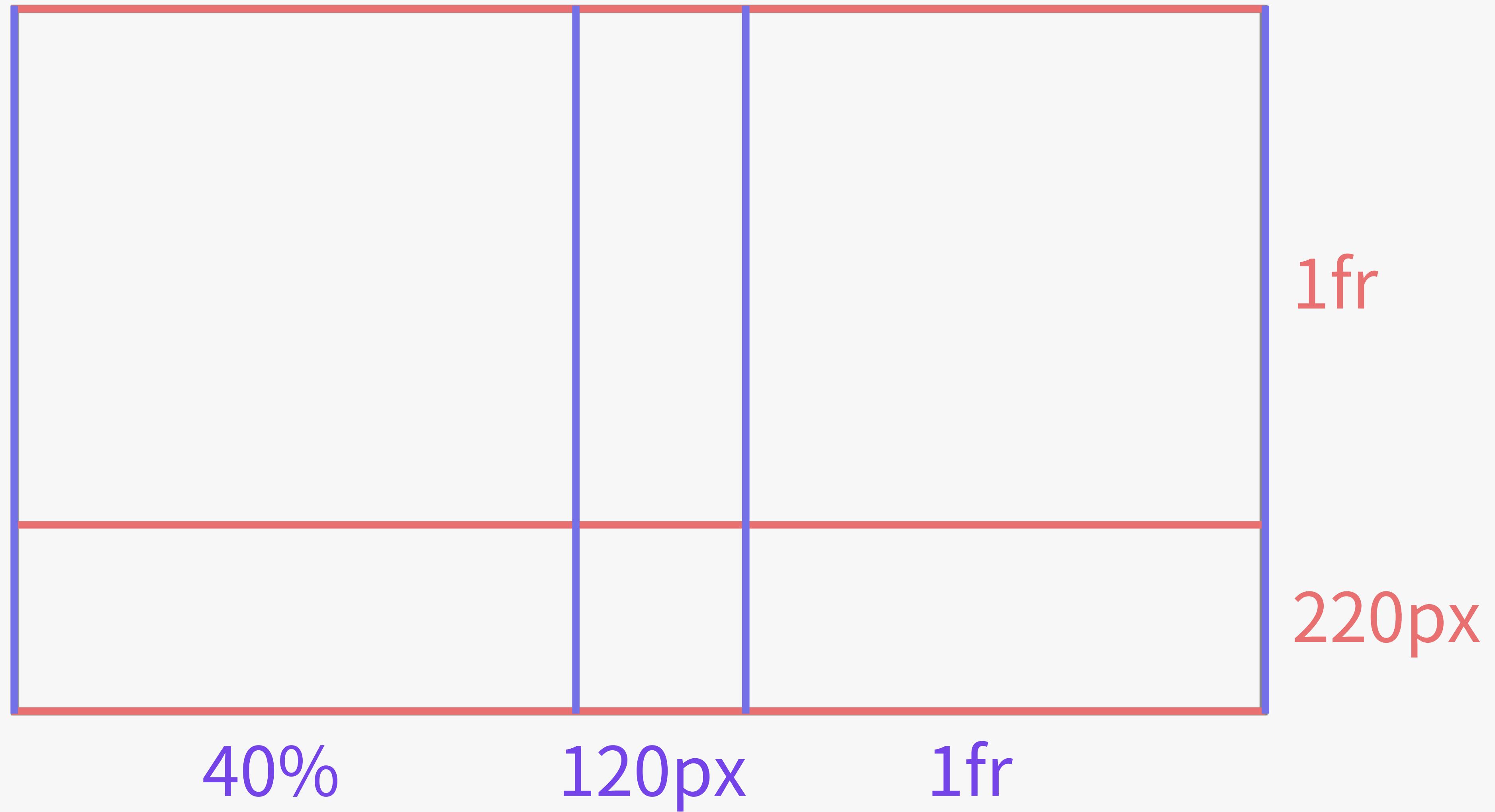
行(row)

列(column)

列(column)

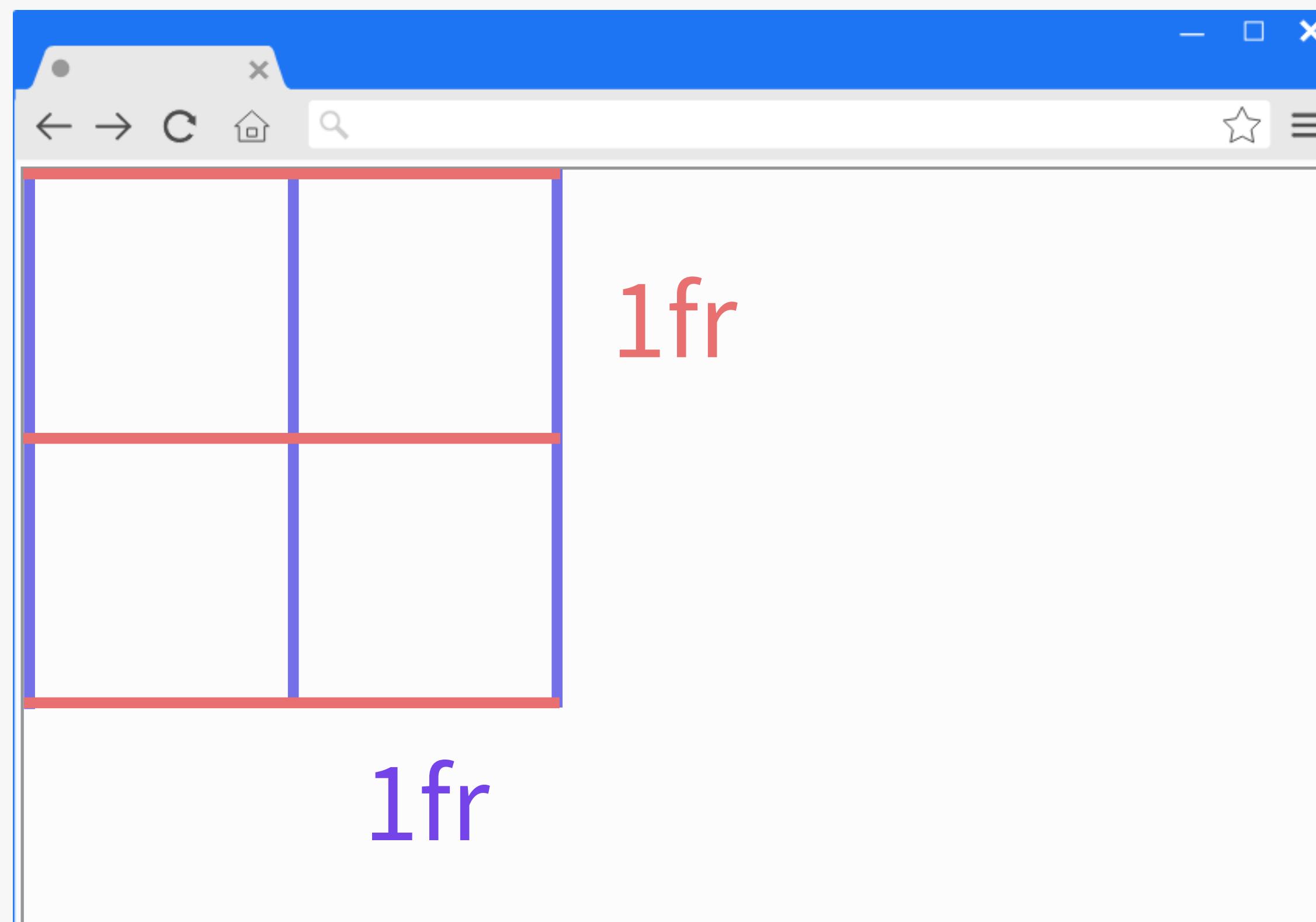
列(column)

2-1. 行と列のサイズを決定する



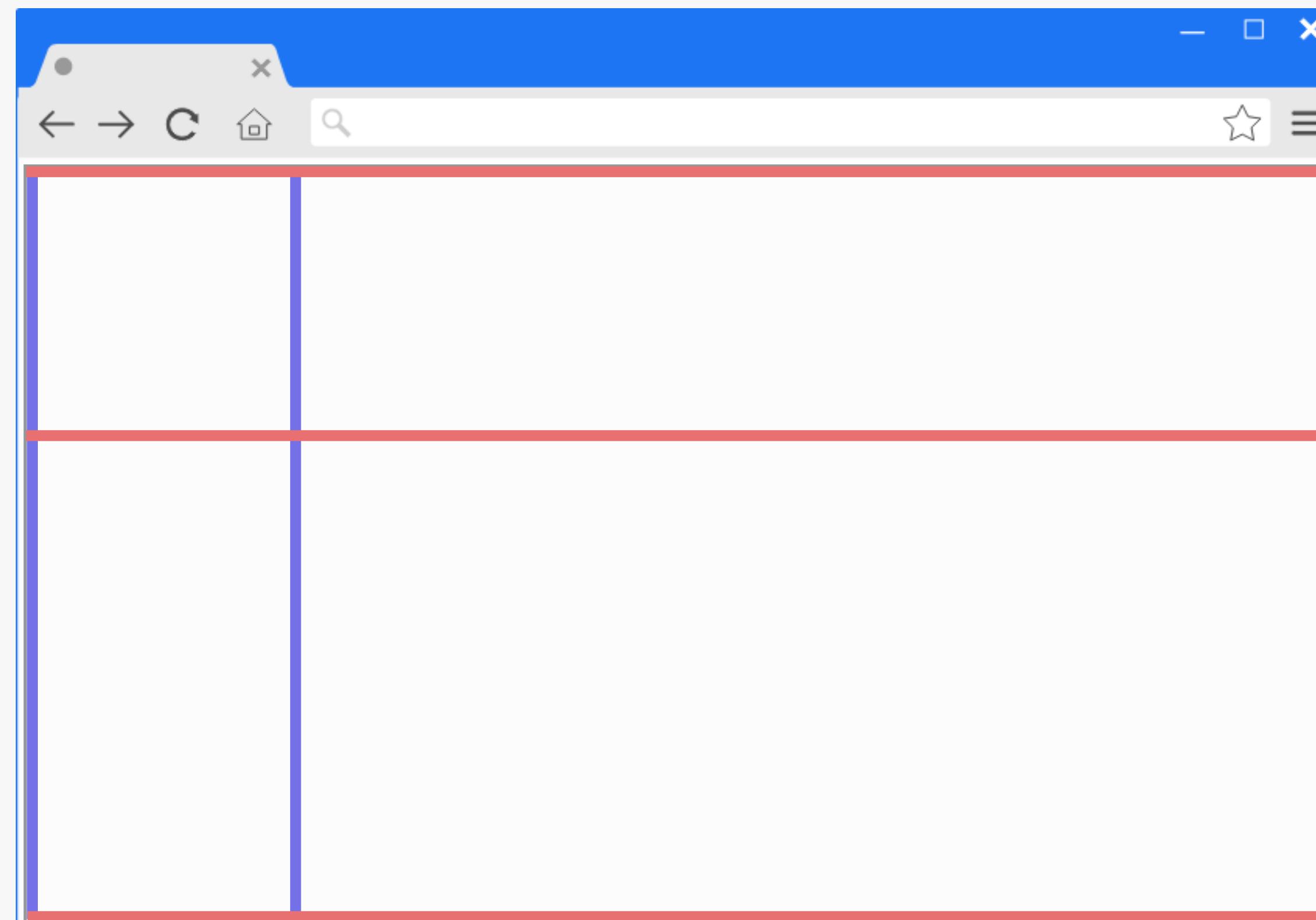
行と列のサイズには「fr」単位が指定できる

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

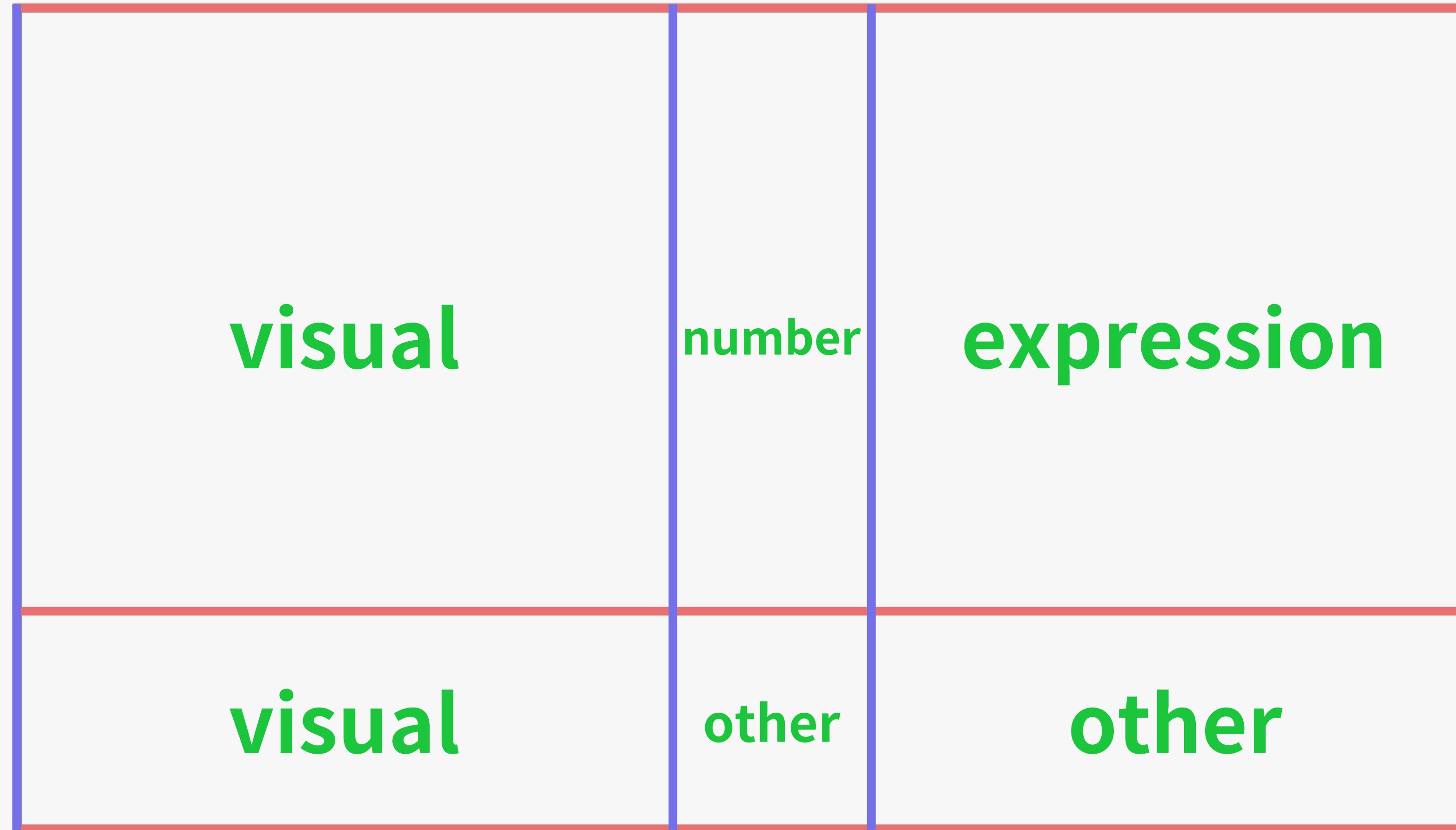
コンテナの余白いっぱいに領域を広げる



1fr

1fr

2-2. エリアに名前をつける



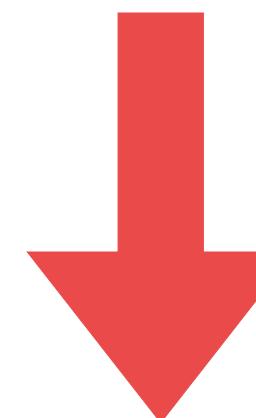
2-3. grid-templateで行列を設定する

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

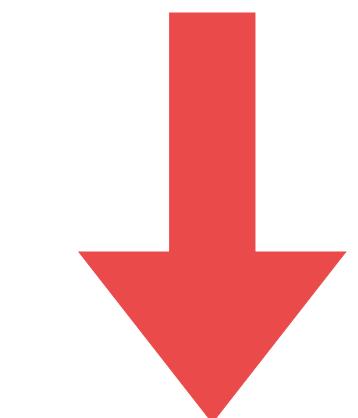
2-3. grid-templateで行列を設定する

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

エリア名



行のサイズ



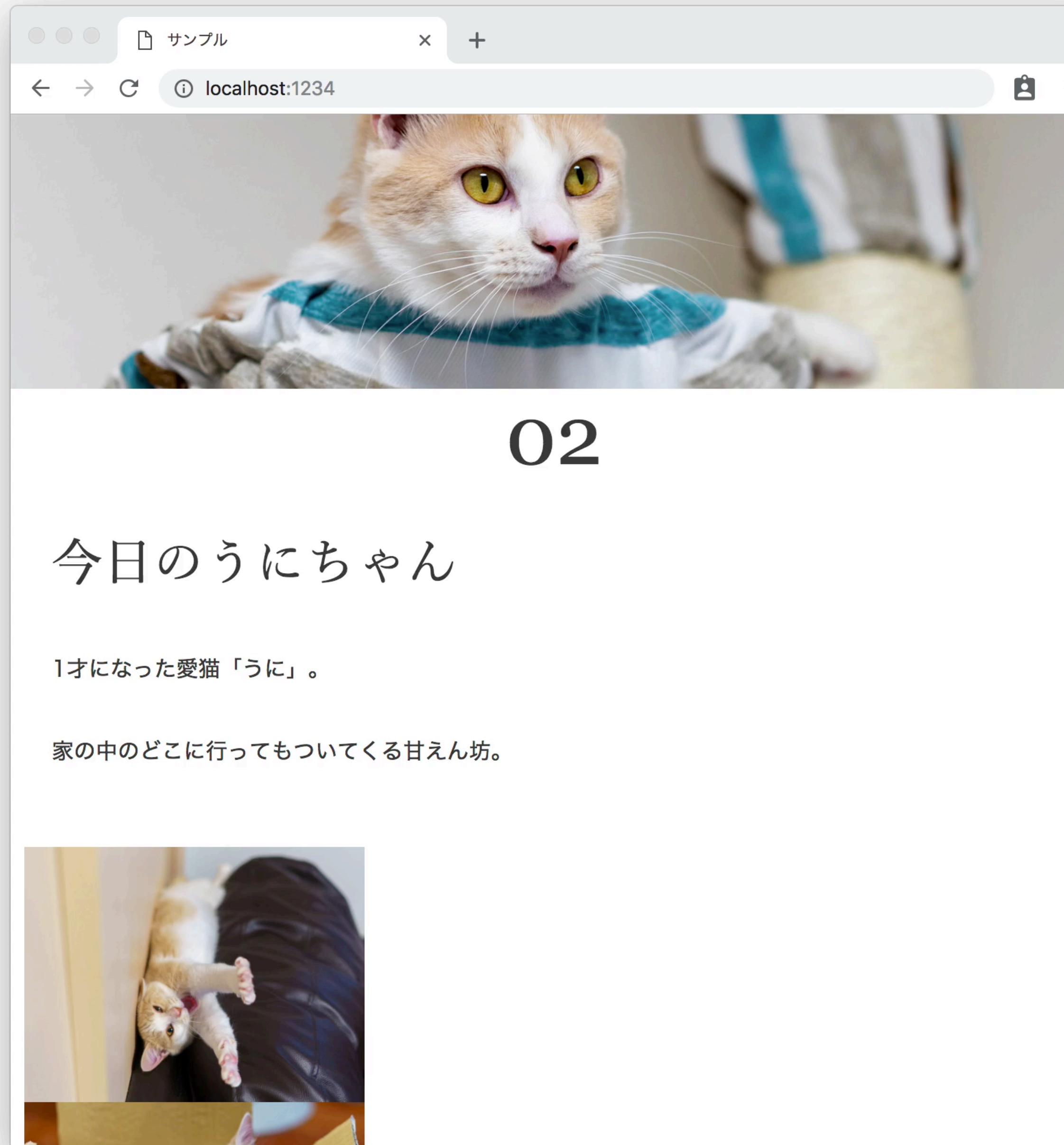
other

1fr

220px /

列のサイズ

行と列を区切るスラッシュ

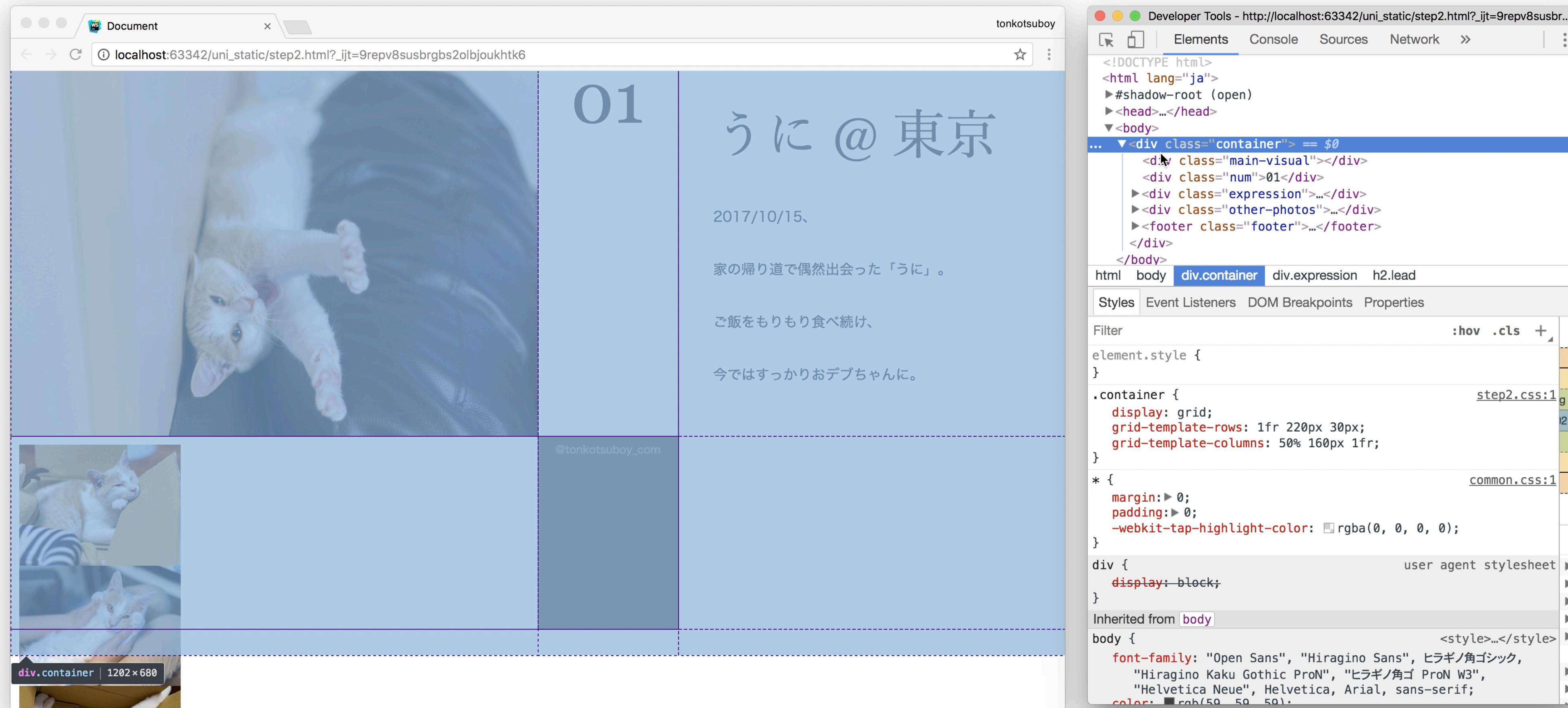
A screenshot of a code editor window titled "sapmle1 [/Volumes/ICS-KANO01/Dropbox (ICS INC)/kano_ics_seminar/180929_cssnite/project/sapmle1] - .../css/style.cs...". The editor shows three tabs: "index.html", "base.css", and "style.css". The "style.css" tab is active and contains the following CSS code:

```
.container {  
    height: 100%;  
    display: grid;  
}
```

The word "display" is highlighted in green, while the other parts of the code are in blue. The ".container" selector is at the bottom of the code editor.

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



The screenshot shows a web browser window with a cat image and some text. The browser's address bar shows the URL `localhost:63342/uni_static/step2.html?_jst=9repv8susbr...
tonkotsuboy`. The page content includes a large image of a white cat, a title "01 うに @ 東京", a date "2017/10/15," and some descriptive text about the cat.

On the right, the browser's developer tools are open, specifically the Elements tab. It displays the DOM structure:

```
<!DOCTYPE html>
<html lang="ja">
  <head>...</head>
  <body>
    ... <div class="container"> == $0
      <div class="main-visual"></div>
      <div class="num">01</div>
      <div class="expression">...</div>
      <div class="other-photos">...</div>
      <div class="footer">...</div>
    </div>
  </body>
```

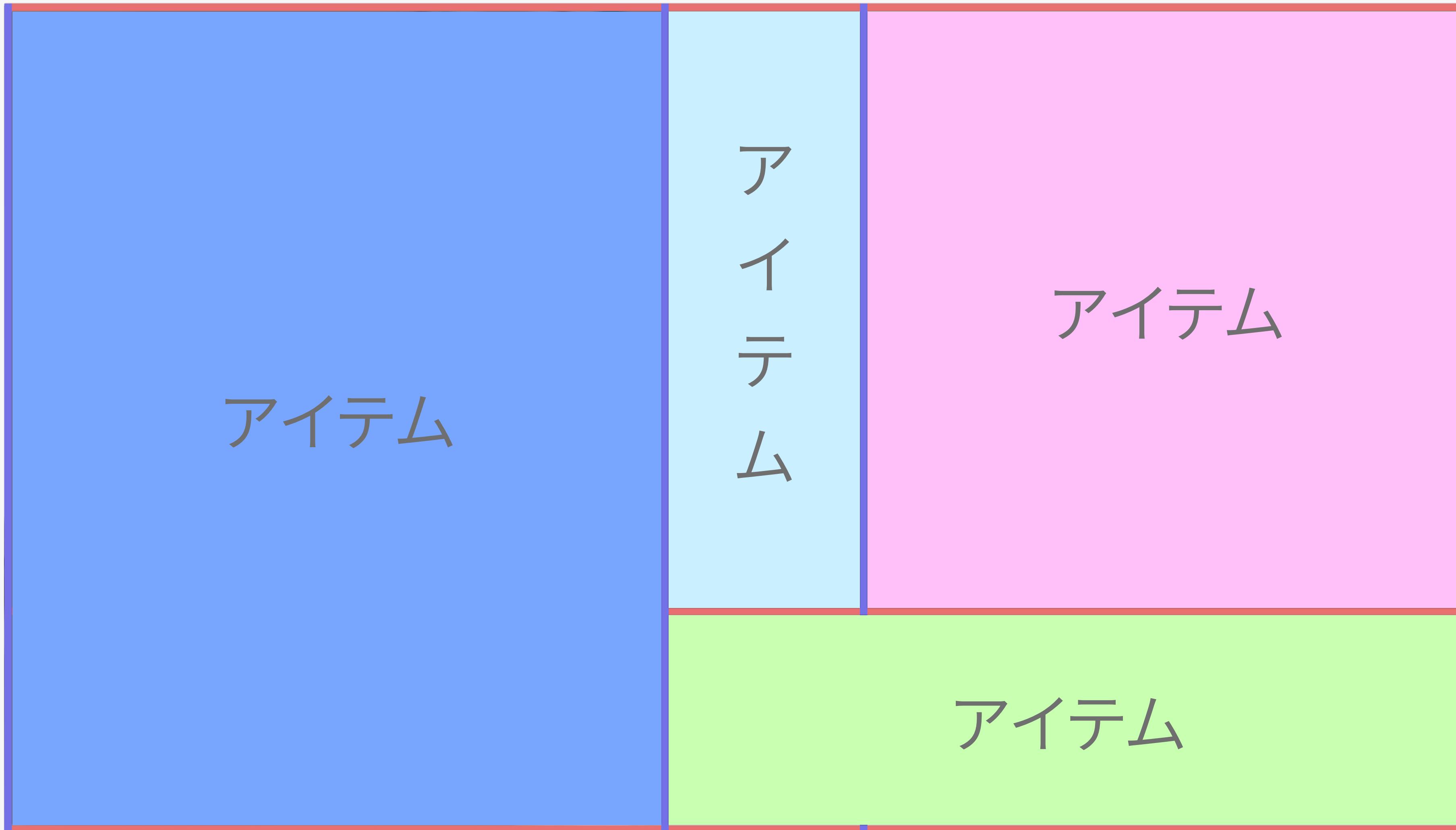
The "Styles" tab is selected in the developer tools. It shows the CSS rules applied to the elements, including:

```
element.style { }
.container {
  display: grid;
  grid-template-rows: 1fr 220px 30px;
  grid-template-columns: 50% 160px 1fr;
}
* {
  margin: 0;
  padding: 0;
  -webkit-tap-highlight-color: transparent;
}
div {
  display: block;
}
```

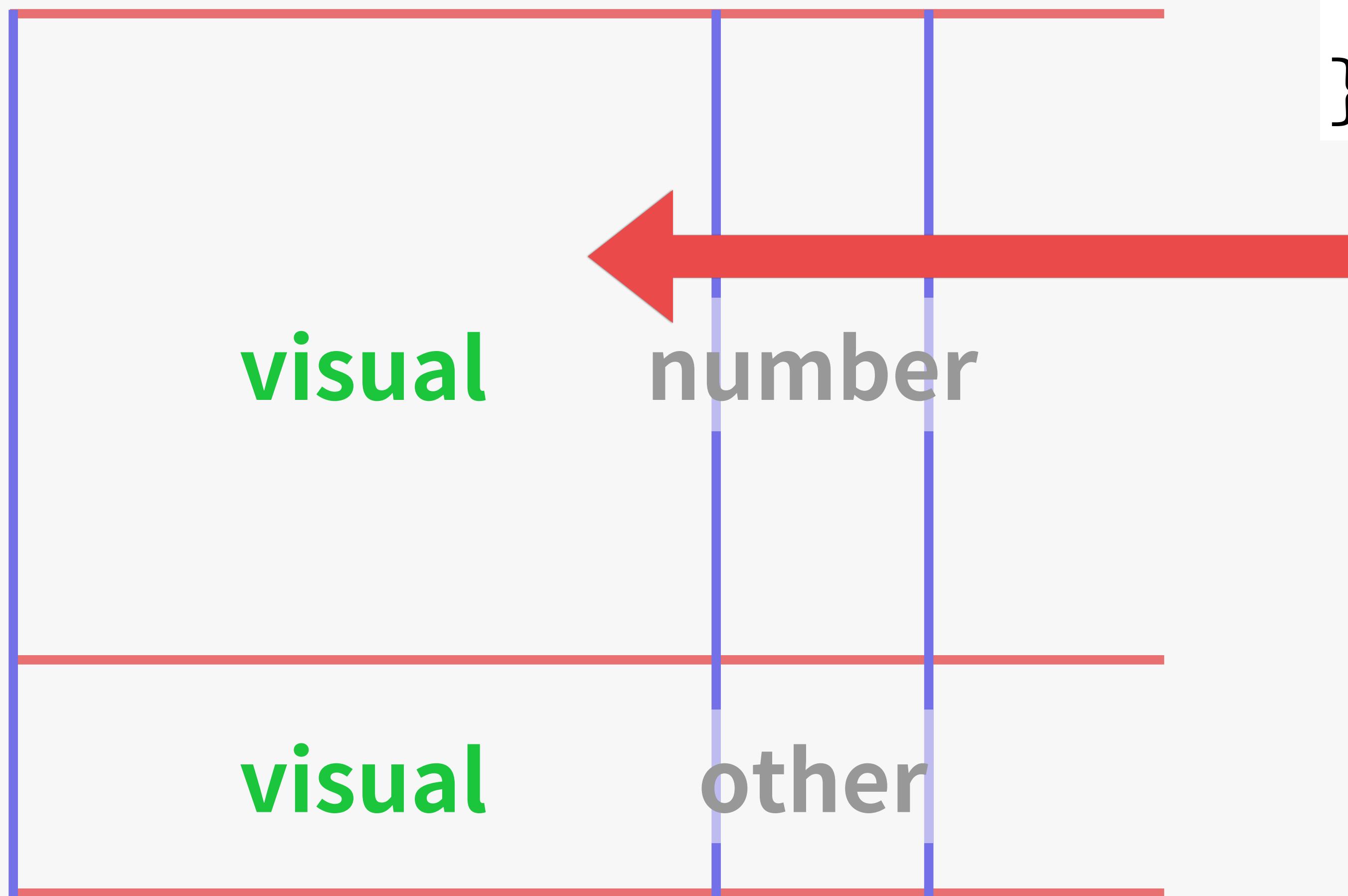
The "Inherited from body" section shows the font settings:

```
body {
  font-family: "Open Sans", "Hiragino Sans", ヒラギノ角ゴシック,
  "Hiragino Kaku Gothic ProN", "ヒラギノ角ゴ ProN W3",
  "Helvetica Neue", Helvetica, Arial, sans-serif;
  color: #rgb(50, 50, 50);
```

3. アイテムの配置場所の決定

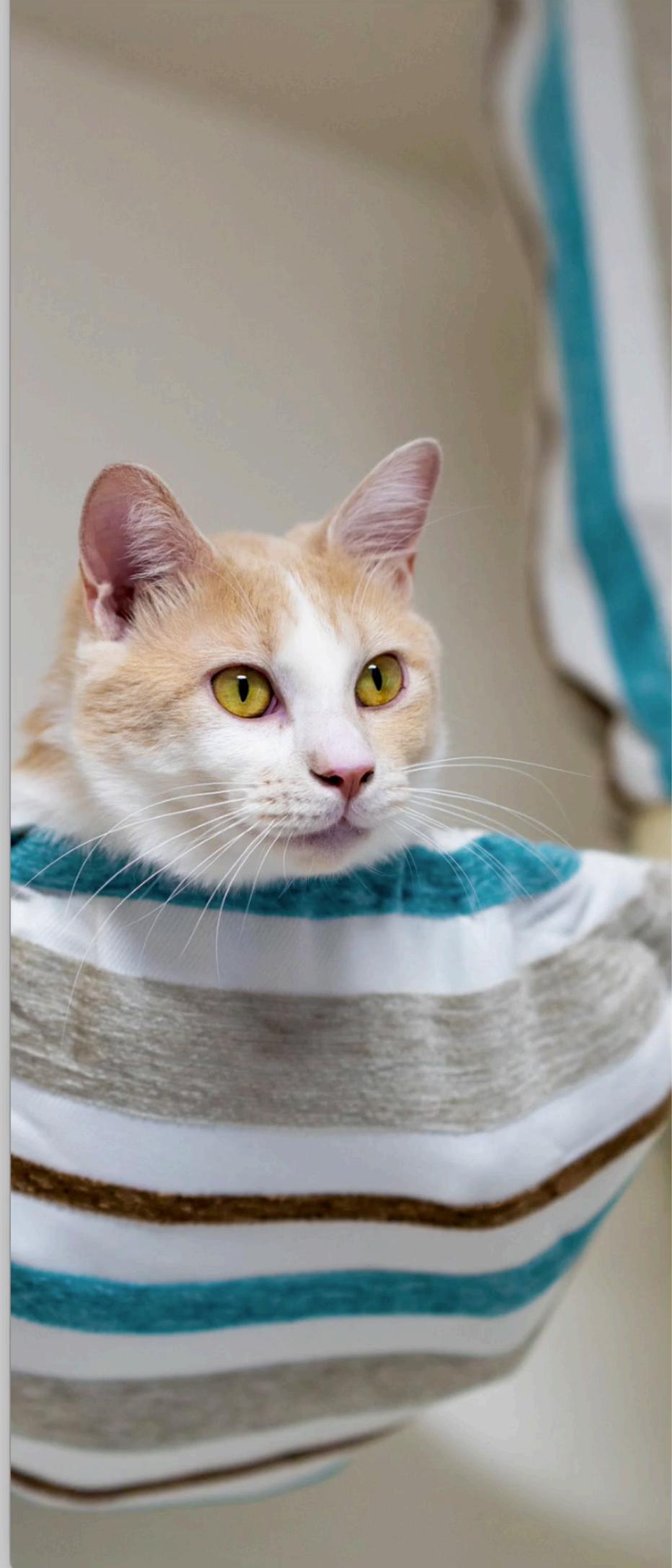


3-1. grid-areaで配置したいエリア名を指定



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





02

今日のうにちゃん

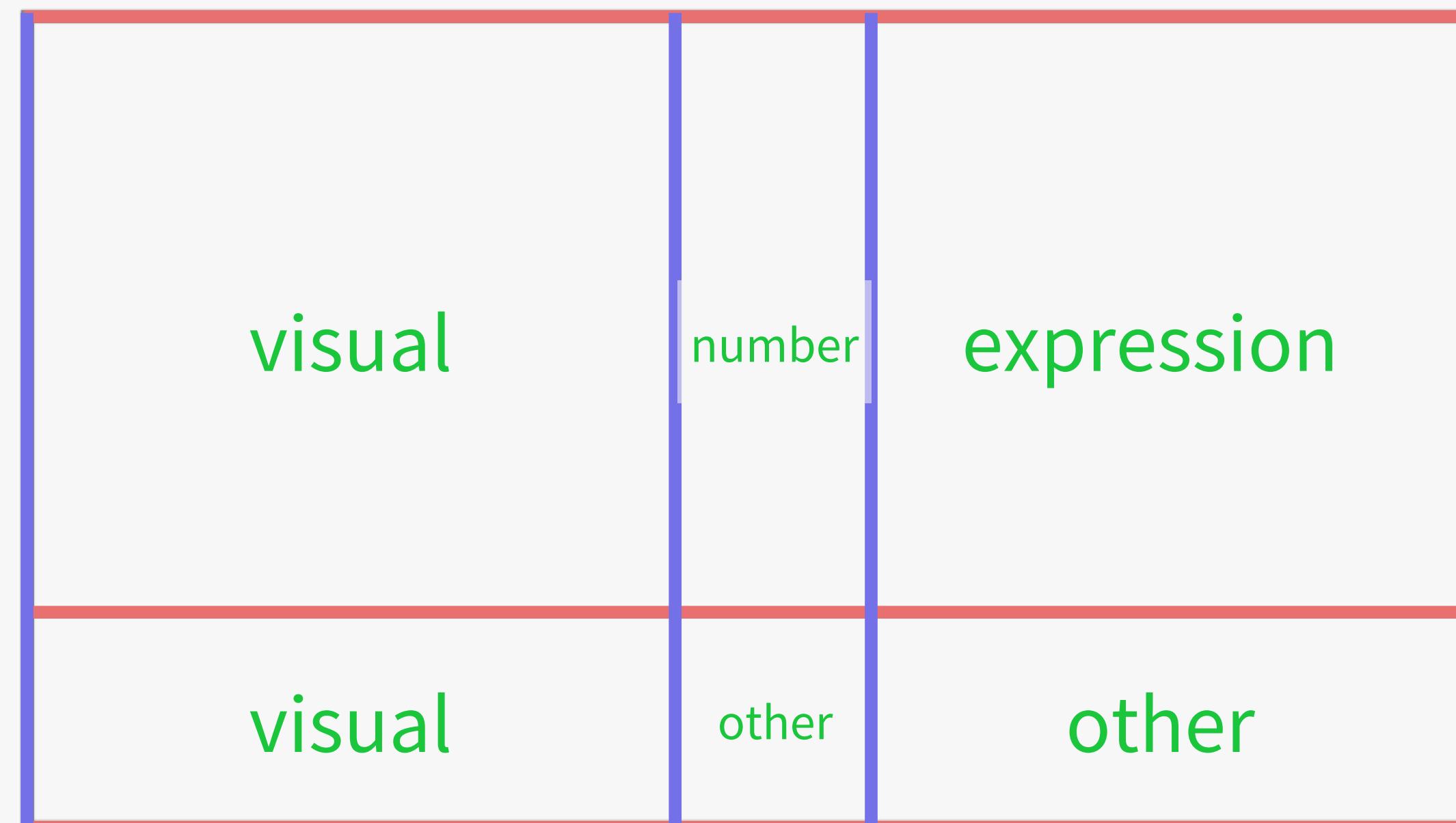
1才になった愛猫「うに」。
家の中のどこに行ってもついてくる甘えん坊。

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

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

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

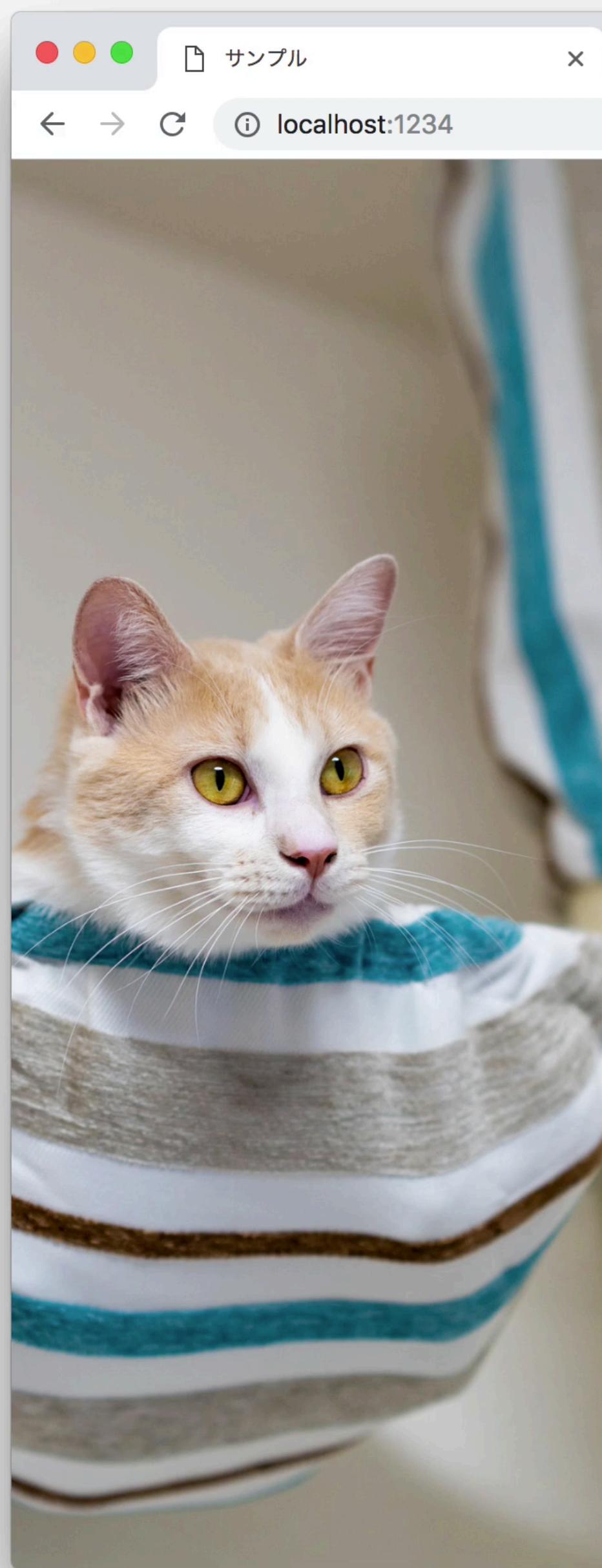
02

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。

```
.visual {  
    grid-area: visual;  
}  
.number {  
    grid-area: number;  
}  
.expression {  
    grid-area: expression;  
}  
.other {  
    grid-area: other;  
}
```



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 ✓

grid-area: visual;
}

.number {
  grid-area: number;
}

.expression {
  grid-area: expression;
}

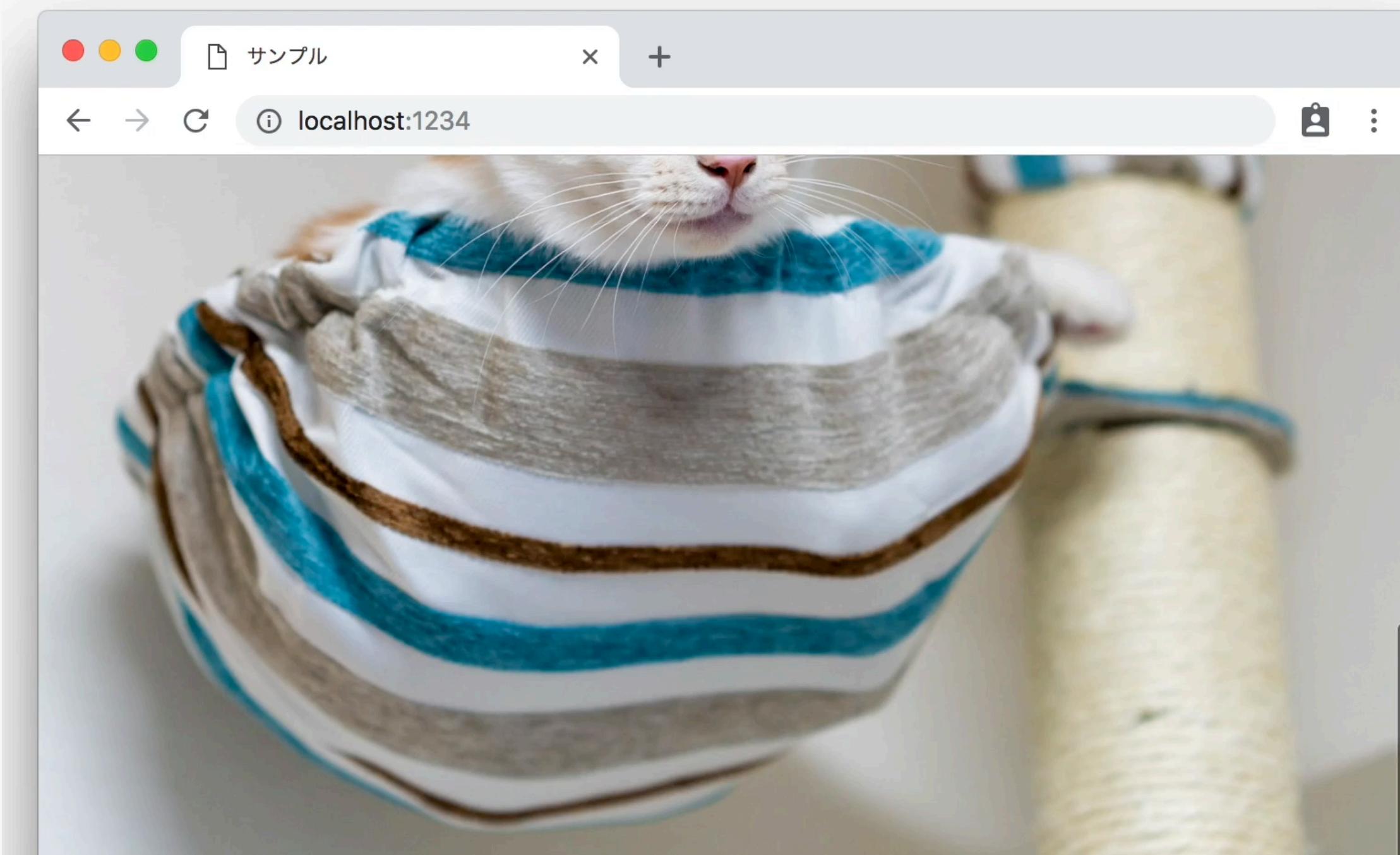
.other {
  grid-area: other;
  display: flex;
  justify-content: space-between;
  align-items: center;
}

.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  
  }  
}
```

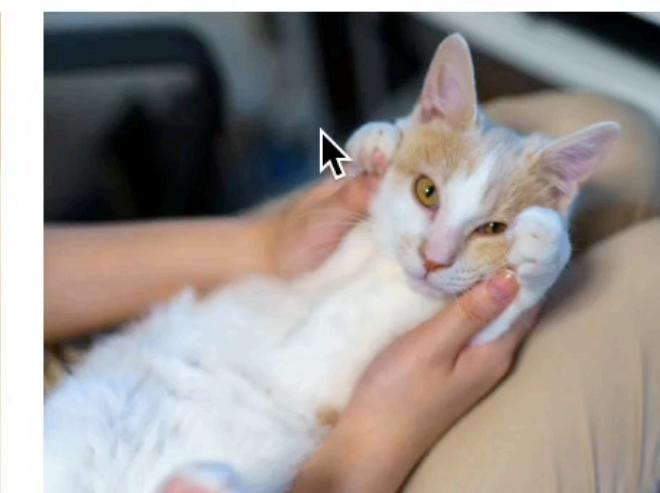
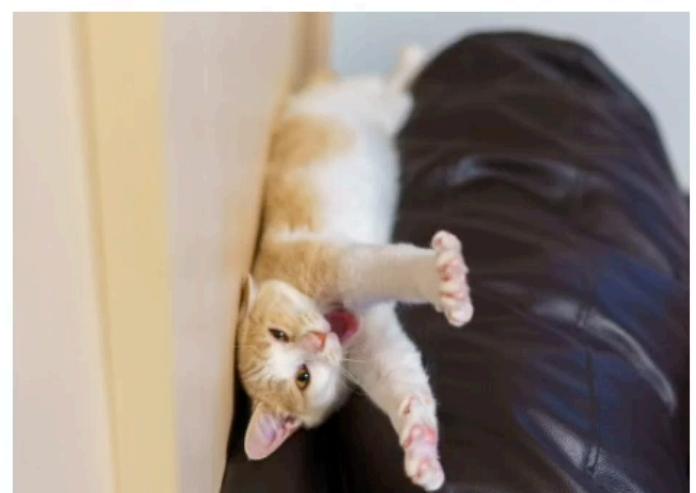


02

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。



```
samp1 [/Volumes/ICS-KANO01/Dropbox (ICS INC)/kano_ics_seminar/180929_cssnite/project/samp1] - .../css/style.css...  
style.css x  
"visual number expression"  
"visual other" other" 220px;  
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`



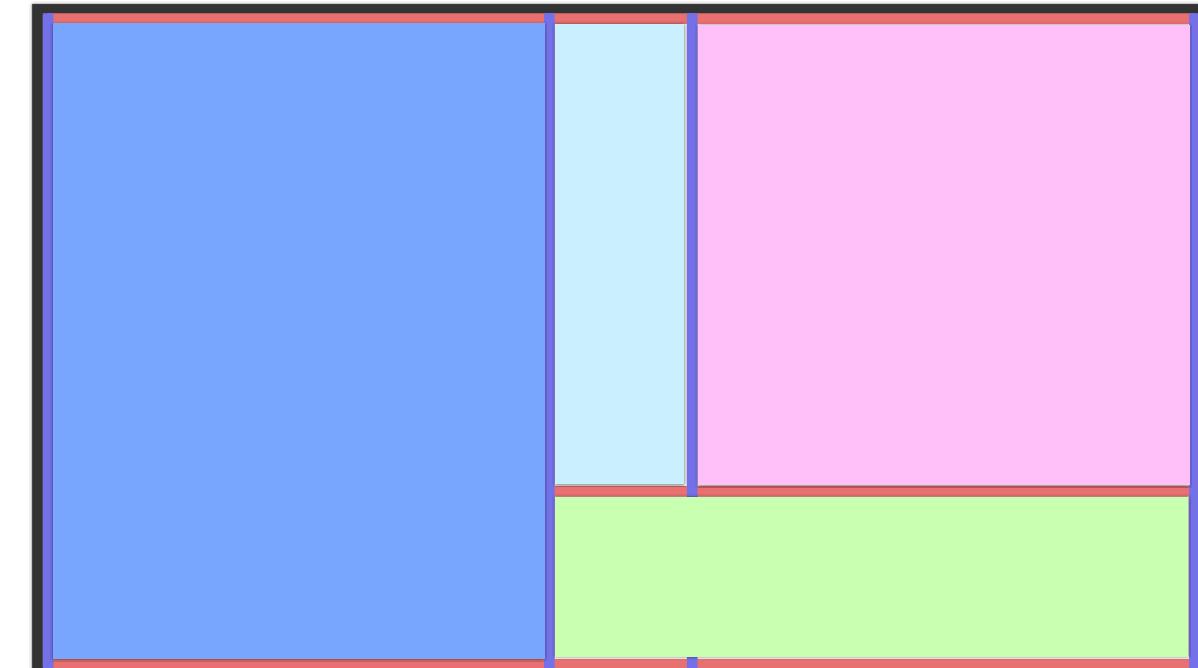
2. 行と列を作る

`grid-template`
プロパティ

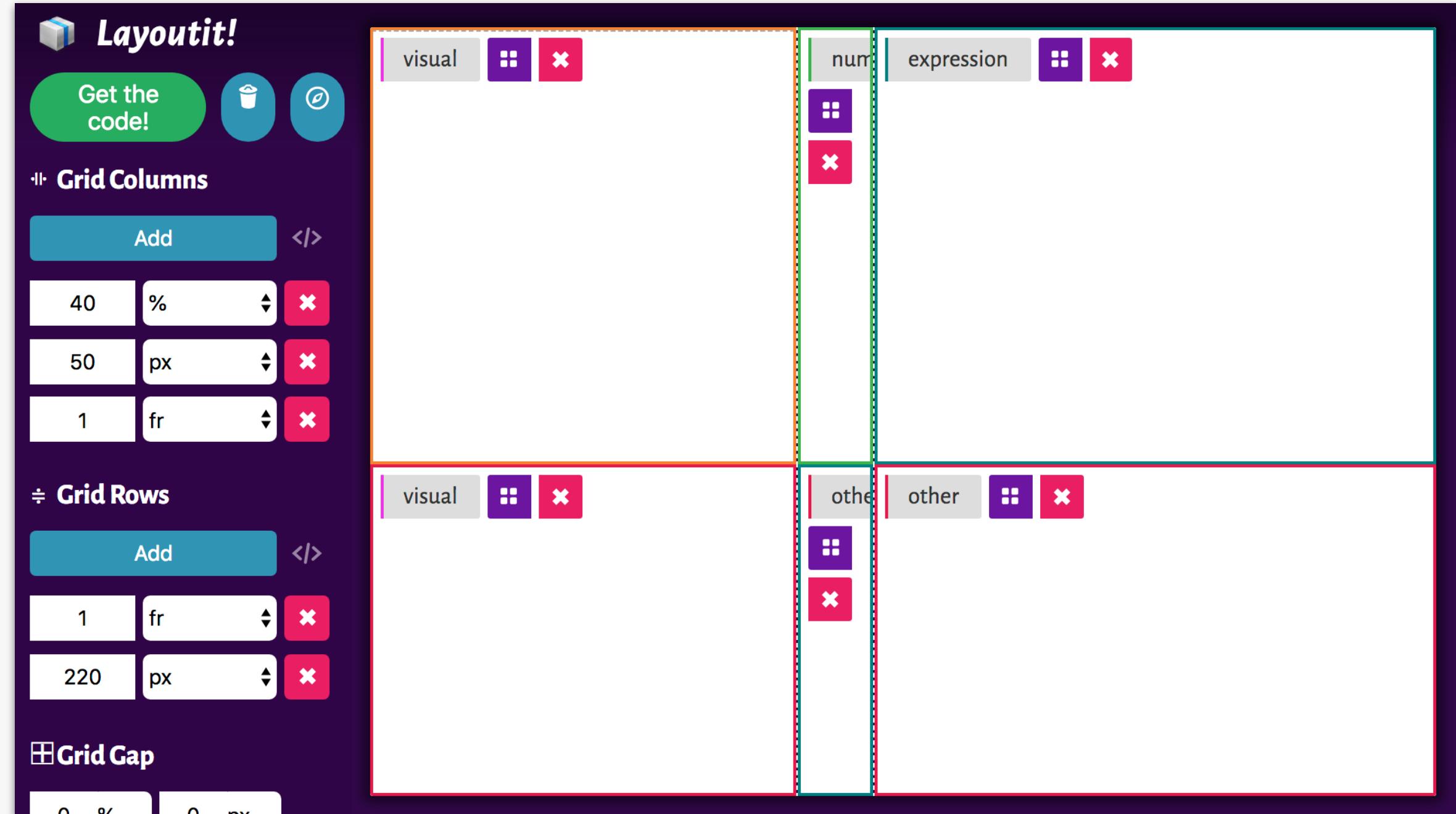


3. アイテムを配置する

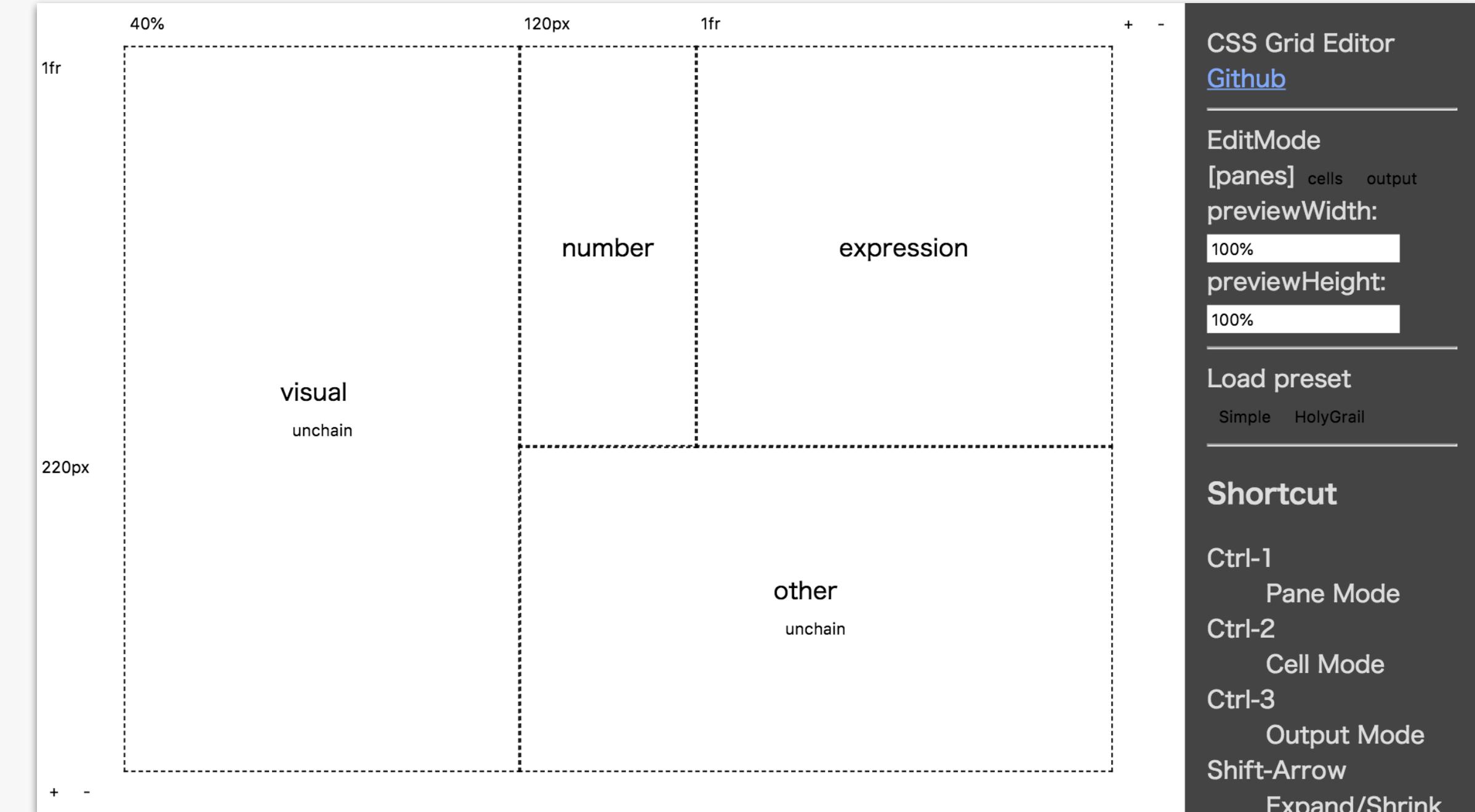
`grid-area`
プロパティ



オンラインのツールを使う方法



The screenshot shows the Layoutit! tool interface. On the left, there are sections for "Grid Columns" and "Grid Rows", each with "Add" buttons and input fields for values like "40%", "50px", and "1fr". The main area displays a grid with three columns and two rows. The first column is labeled "visual" and contains a "number" cell. The second column is labeled "expression" and contains an "expression" cell. The third column is labeled "other" and contains another "other" cell. Each cell has a "Get the code!" button and a trash icon.



The screenshot shows the CSS Grid Editor by @mizchi. It features a visual representation of a grid with dimensions 40% by 220px. The grid is divided into four cells: top-left (1fr, 120px) labeled "number", top-right (1fr) labeled "expression", bottom-left (220px) labeled "other", and bottom-right (1fr) also labeled "other". The "other" cells are labeled "unchain". On the right side, there's a sidebar with "EditMode" settings for panes, cells, and output, and a "Shortcut" section with keys for Pane Mode, Cell Mode, Output Mode, Shift-Arrow, and Expand/Shrink. There are also links for "CSS Grid Editor" and "Github".

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IE11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

821266

125360 +

MENU



トップ



司令室



ストーリー



ガチャ



戦闘機



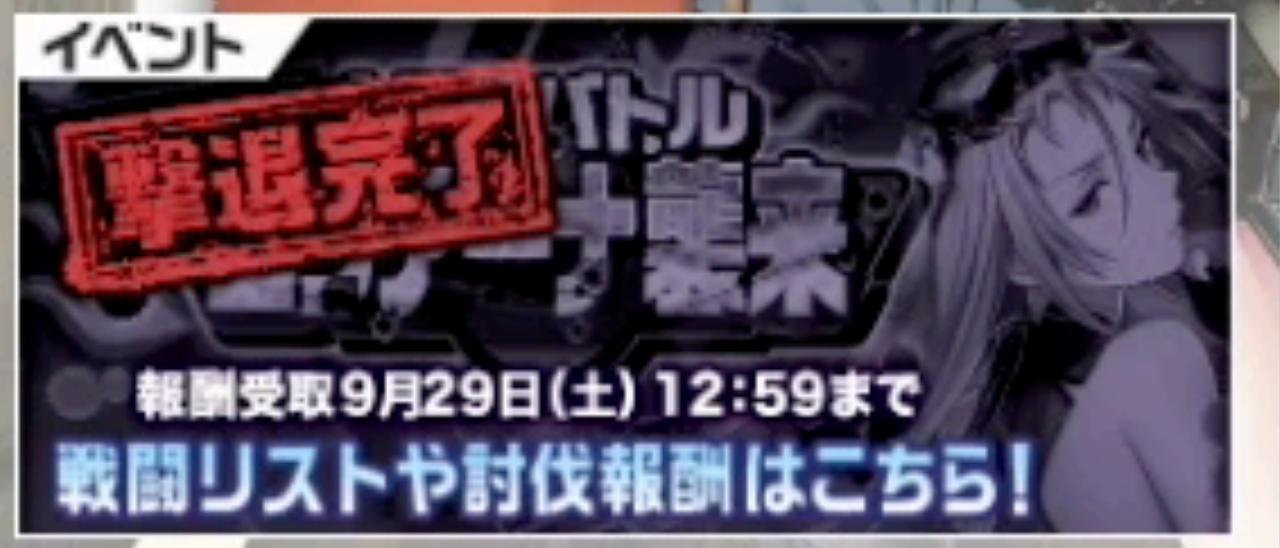
操縦士



編隊



クエスト





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
≡



編隊少女 -フォーメーションガールズ-

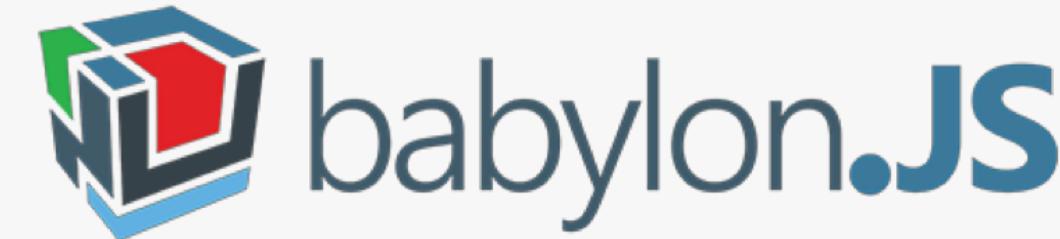


- 2016年に
株式会社アイオウプラスより
リリースされたFlashゲーム
- 2018年夏、HTML製SPA
としてリニューアル
- IE 11でも動作

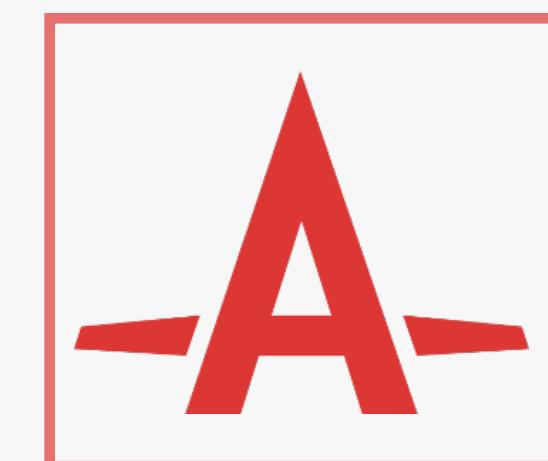
<http://henjo.jp/>

採用した主な技術

TypeScript



PixiJS



CSS Gridで便利だった点1：複雑なレイアウト

部隊名	HIROメンバーフル
紹介文	のんびりまったりです。チャット参加者大歓迎
加入条件	マイペース / 初心者OK
<input type="button" value="クリア"/>	<input type="button" value="キャンセル"/> <input type="button" value="決定"/>

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

- repeat(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IE11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**

Autoprefixerで自動変換

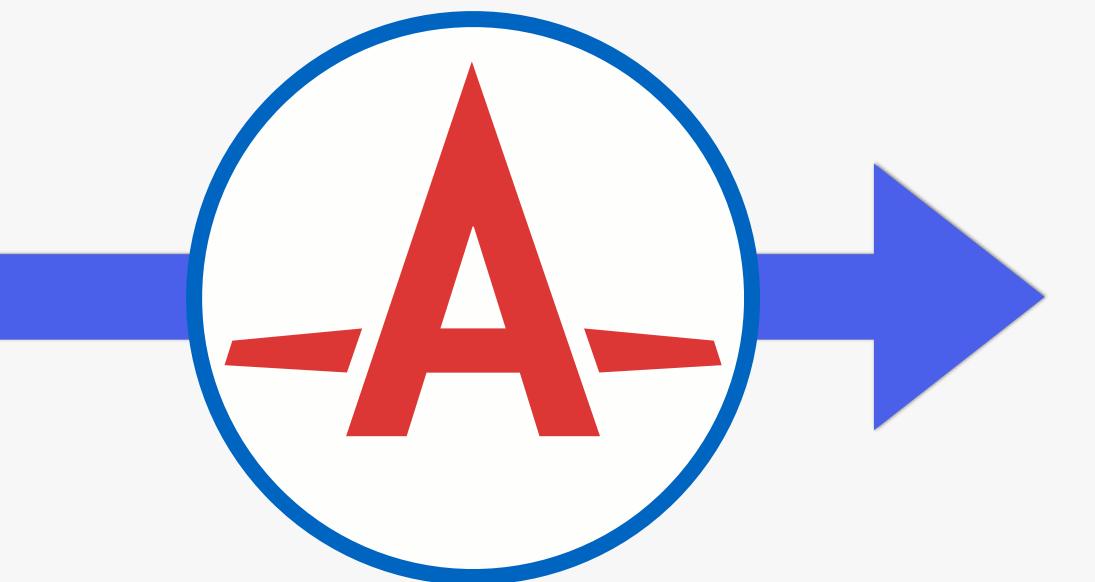
```
.container {
  display: grid;
  grid-template:
    "visual number expression" 1fr
    "visual other other" 220px /
    40% 120px 1fr;
}

.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つた

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

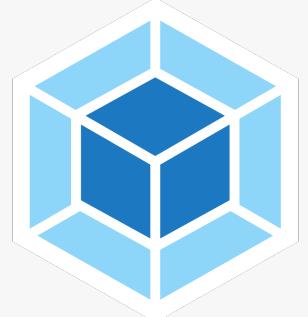
```
.container {
    /* 中略 */

    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

The screenshot shows a browser window titled "Autoprefixer CSS online" at the URL <https://autoprefixer.github.io>. The page displays two blocks of CSS code. The left block is the original input:

```
.container {  
  display: grid;  
  gap: 10px;  
  grid-template:  
    "visual number expression" 1fr  
    "visual other other" 150px /  
    50% 120px 1fr;  
}  
  
.visual {  
  grid-area: visual;  
}  
  
.number {  
  grid-area: number;  
}  
  
.expression {  
  grid-area: expression;  
}  
  
.other {  
  grid-area: other;  
}
```

The right block shows the output after being processed by Postcss v7.0.2 and autoprefixer v9.1.5, which adds vendor prefixes for Microsoft Edge:

```
.container {  
  display: -ms-grid;  
  display: grid;  
  gap: 10px;  
  -ms-grid-rows: 1fr 10px 150px;  
  -ms-grid-columns: 50% 10px 120px 10px 1fr;  
  grid-template:  
    "visual number expression" 1fr  
    "visual other other" 150px /  
    50% 120px 1fr;  
}  
  
.visual {  
  -ms-grid-row: 1;  
  -ms-grid-row-span: 3;  
  -ms-grid-column: 1;  
  grid-area: visual;  
}  
  
.number {  
  -ms-grid-row: 1;  
  -ms-grid-column: 3;  
}
```

At the bottom, there are filters for "last 4 version" and "Apply" buttons, along with a "Select result" button and a note about browser compatibility via [browserlist](#).

Autoprefixer CSS online

- 環境構築不要

<https://autoprefixer.github.io/>

You can also see which browsers you choose by filter string on [browserlist](#)

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IE11対応がラクになった
5. 未来のCSS Gridと周辺技術
6. まとめ

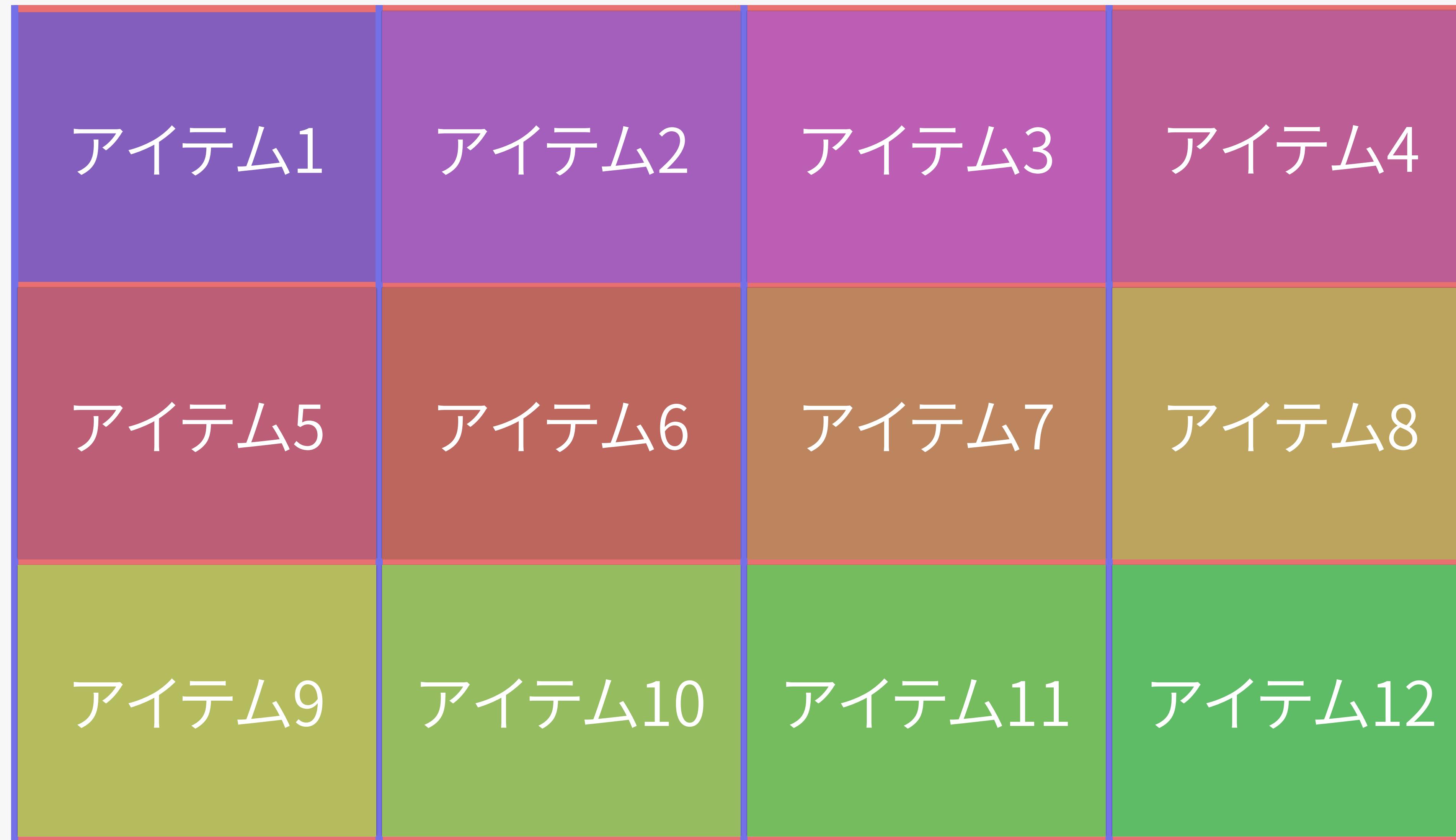
未来のCSS Gridと周辺技術

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

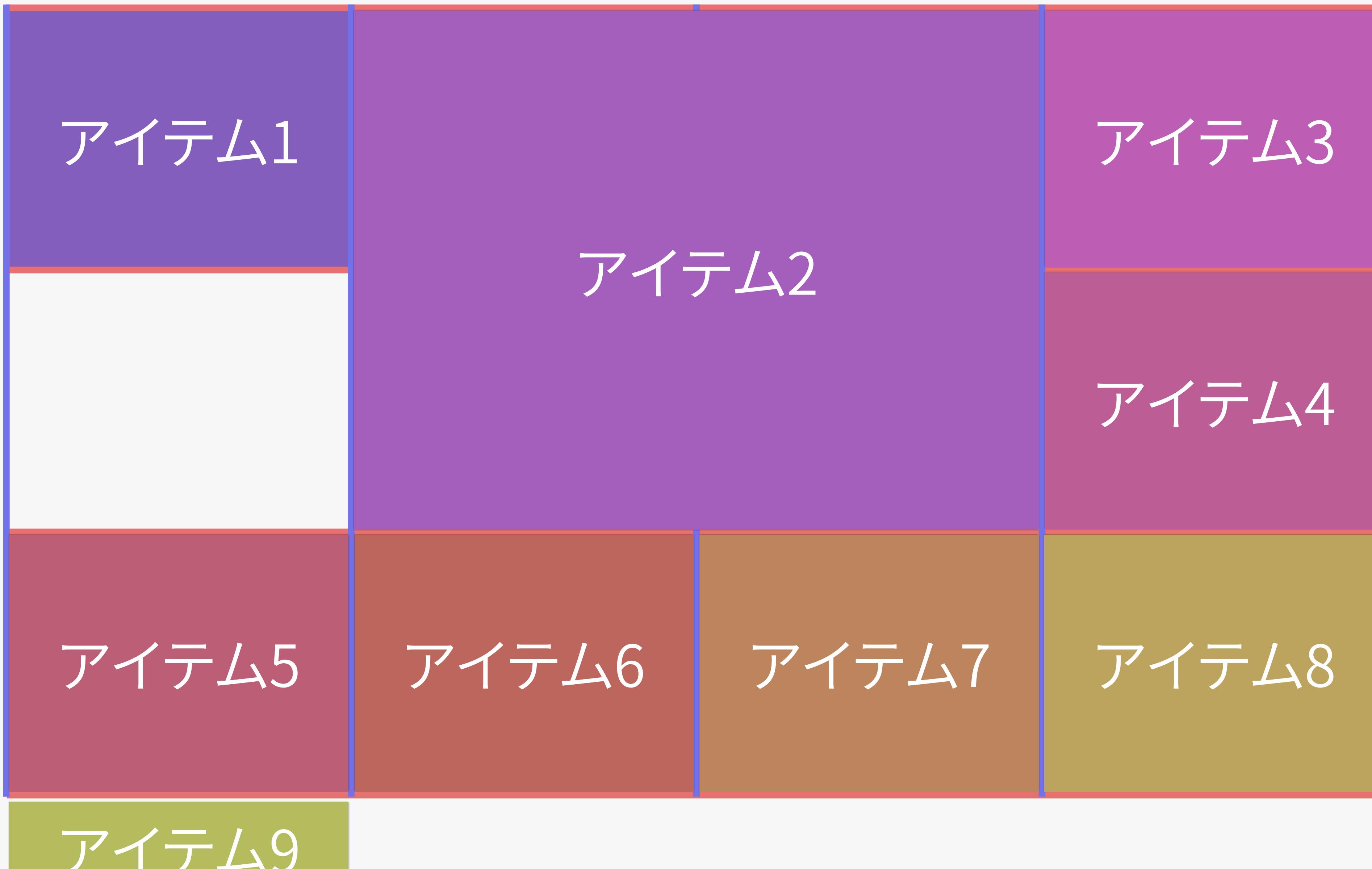
CSS Gridでは、アイテムは左上から自動的に配置される



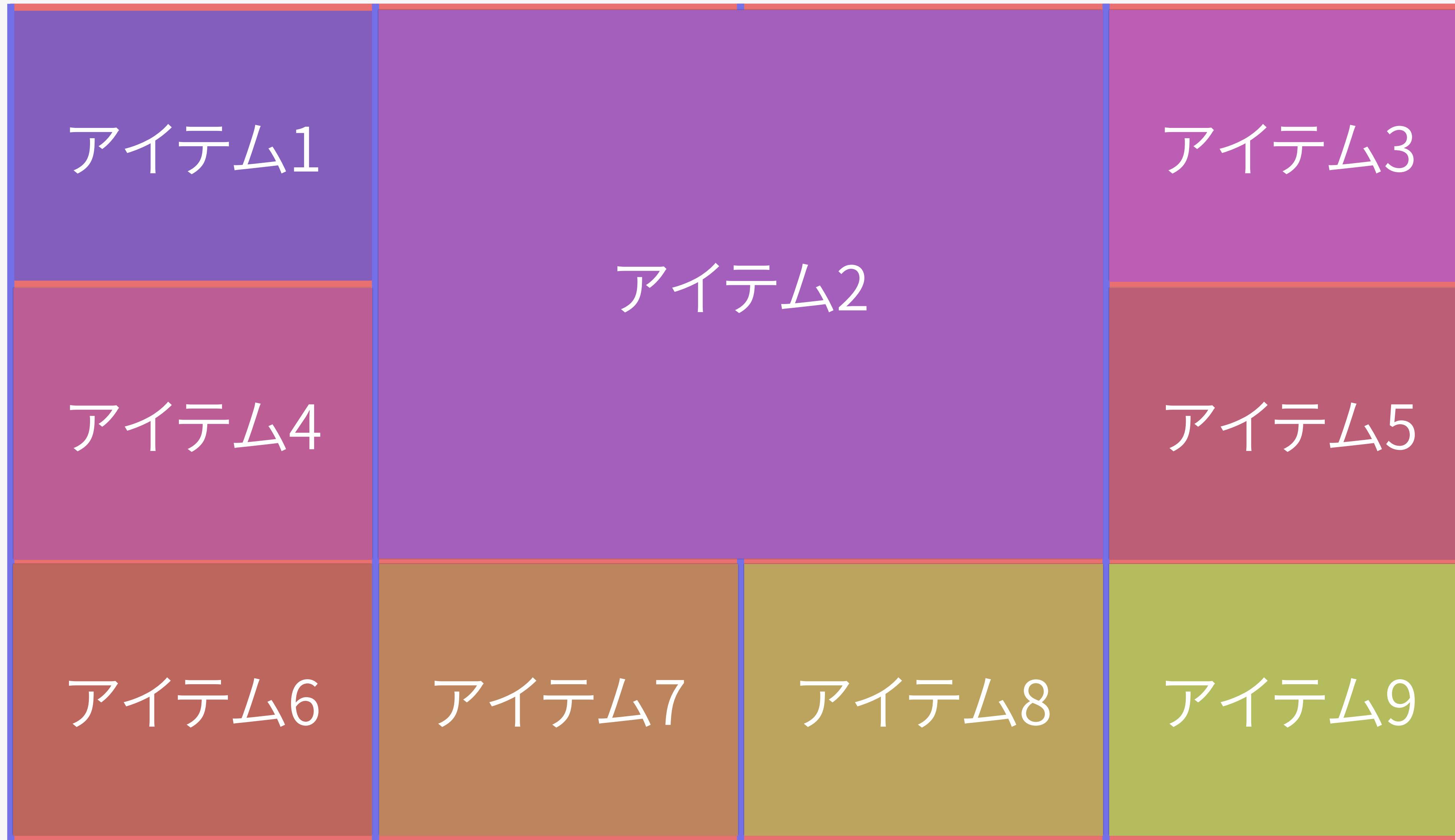
grid-autoflow: columnで列順に

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

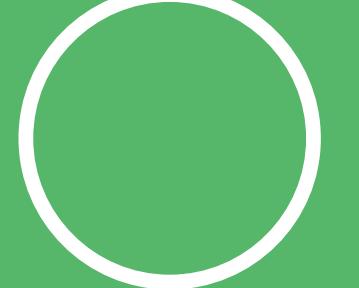
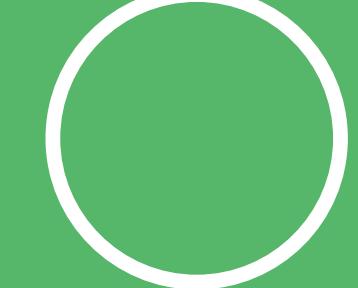
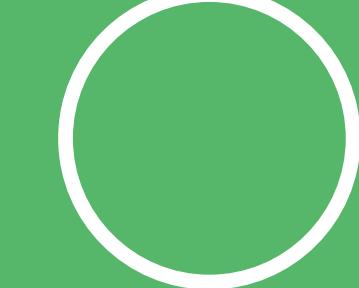
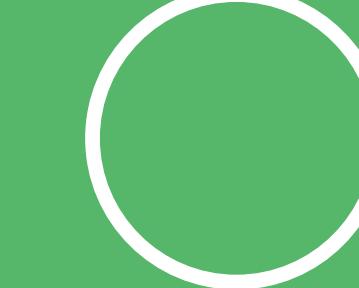
大きいアイテムがあると隙間があく



grid-autoflow: denseで敷き詰める



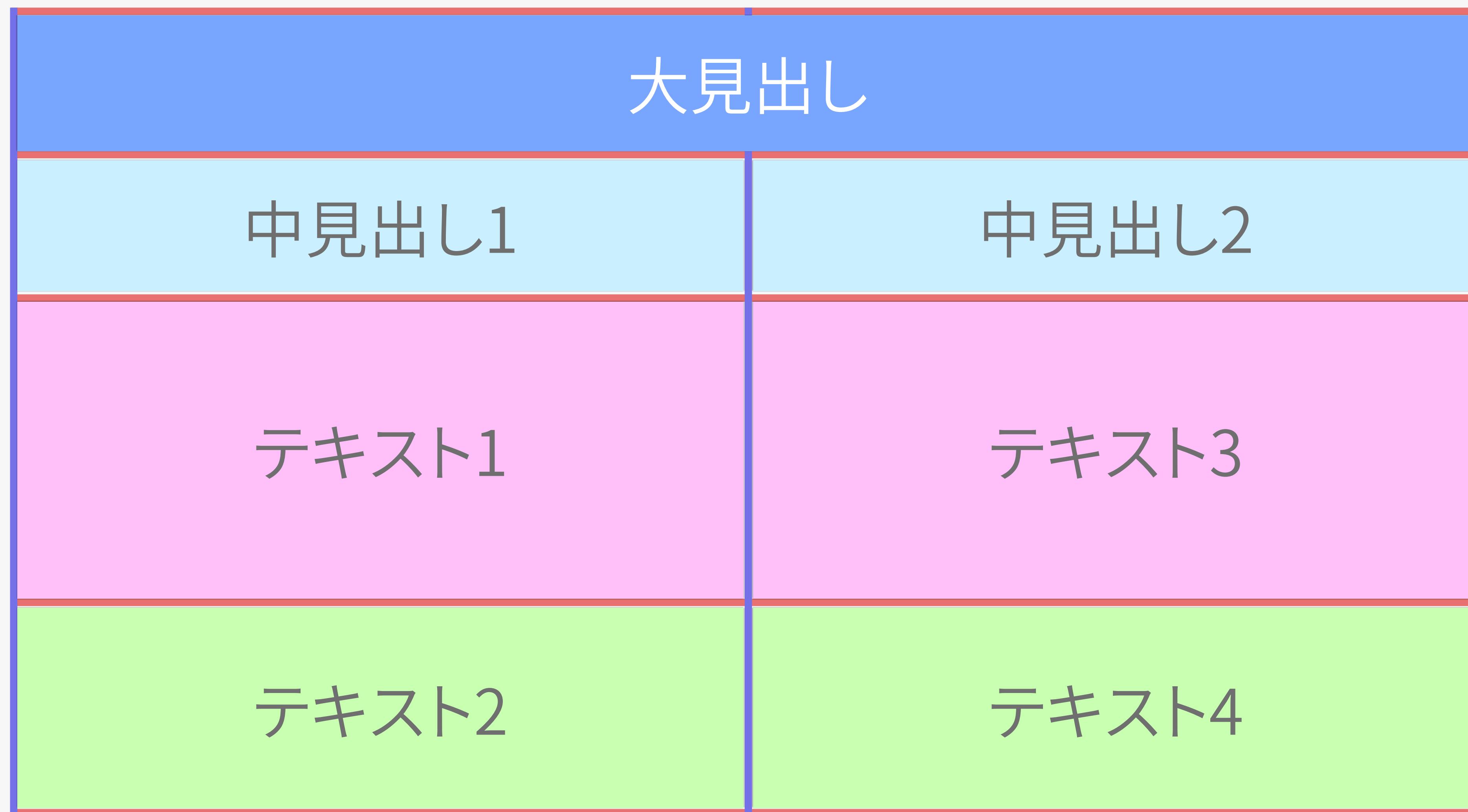
grid-autoflowプロパティの対応状況

							
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



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>
```

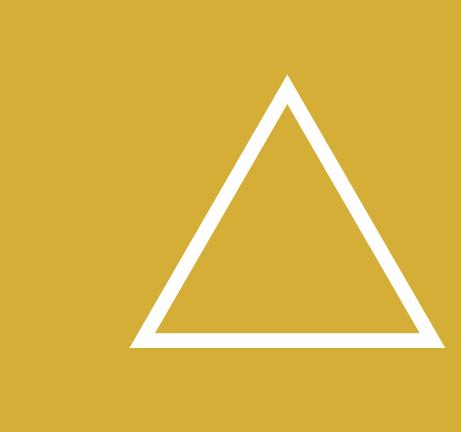
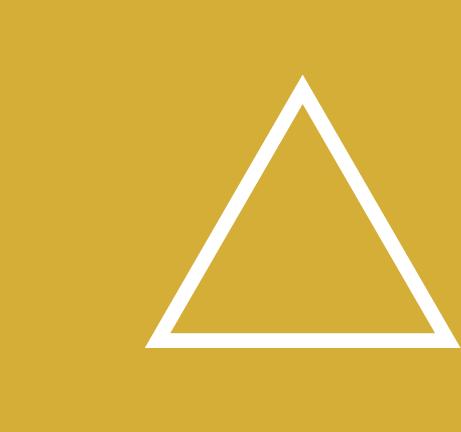
CSS

```
.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

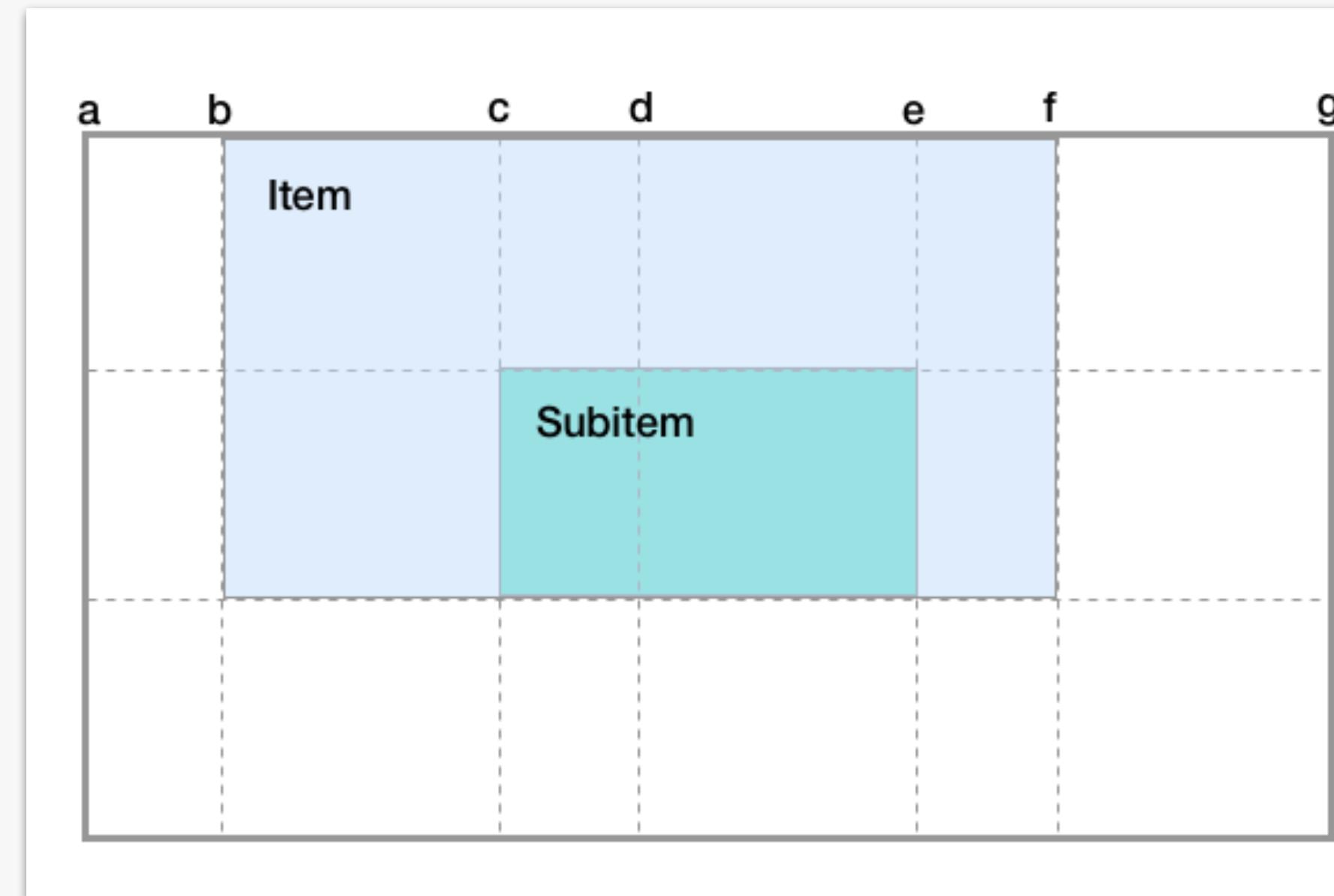
							
ブラウザ名	Chrome	Firefox	Safari	Edge	Internet Explorer	Safari (iOS版)	Chrome (Android版)
最新バージョン	69	62	12	17	11	12	69
Grid対応							

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IE11対応がラクになった
5. 未来のCSS Gridと周辺技術
6. まとめ

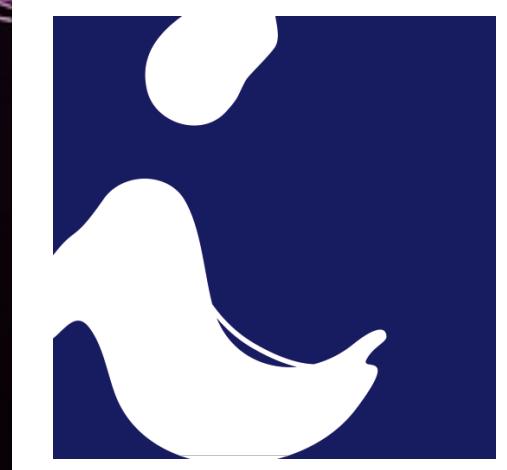
まとめ

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

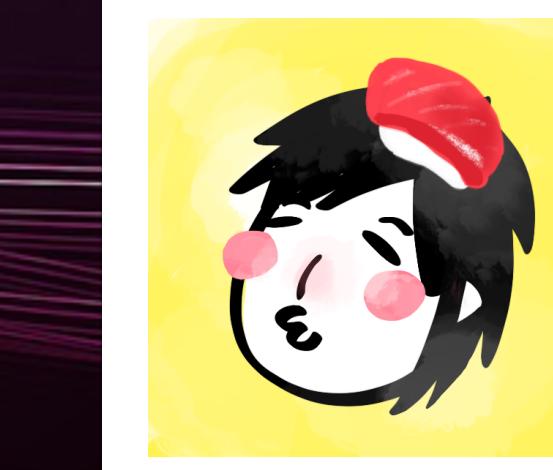
CSS Gridを今日から使おう！

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 