

2019年までに見直しておきたい CSS・JavaScriptの手法

株式会社ICS 鹿野 壮

平成30年12月8日 @ Frontend Conference Fukuoka 2018

#fec_fukuoka

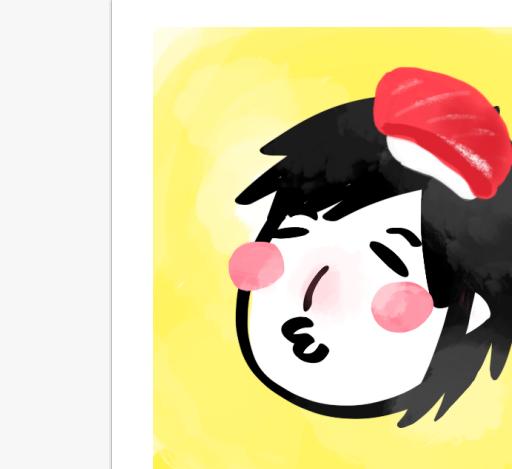
自己紹介



福岡県筑後市出身

九州大学芸術工学部音響設計学科卒

鹿野 壮 (かの たけし)



@tonkotsuboy_com

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

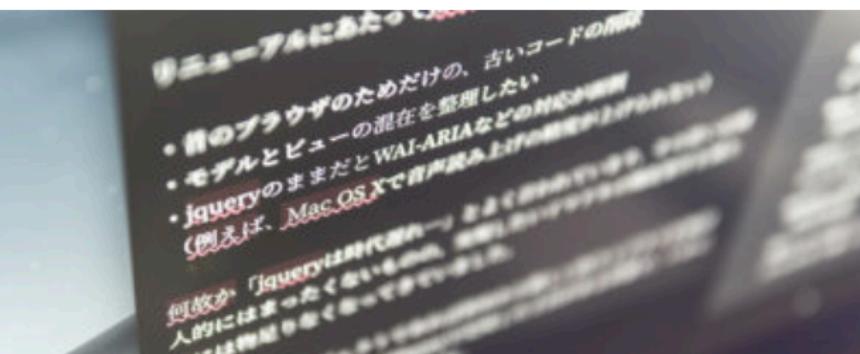
3D表現

現場で使えるフロントエンド解説

デザイン

アプリ開発

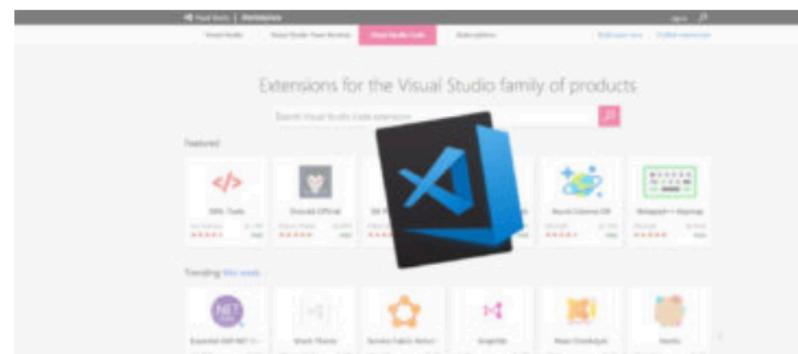
話題の記事



現場で使えるフロントエンド解説

エンジニアのための、いますぐ使える文章校正テクニック

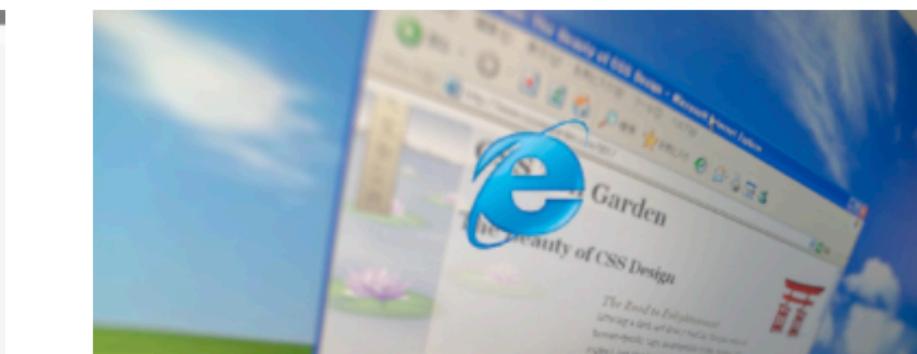
池田 2018.10.05 ❤ 2380



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 ❤ 3503



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 ❤ 2132

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

w

プッシュ通知

ICS MEDIA - Webデザイナー/フロントエンジニアのメディア
<https://ics.media/>

CSSやJavaScriptには、
便利な機能が次々と追加されている

当たり前だと思っていた手法を見直すことで
より便利にウェブ開発ができる

2019年までに見直しておきたいCSS・JavaScriptの手法

1. CSSの見直し
2. JavaScriptの見直し
3. ツールの見直し

2019年までに見直しておきたいCSS・JavaScriptの手法

1. CSSの見直し

- Flexbox
- CSS Grid
- モダンブラウザで使えるようになった機能
- CSS新機能のキャッチアップ

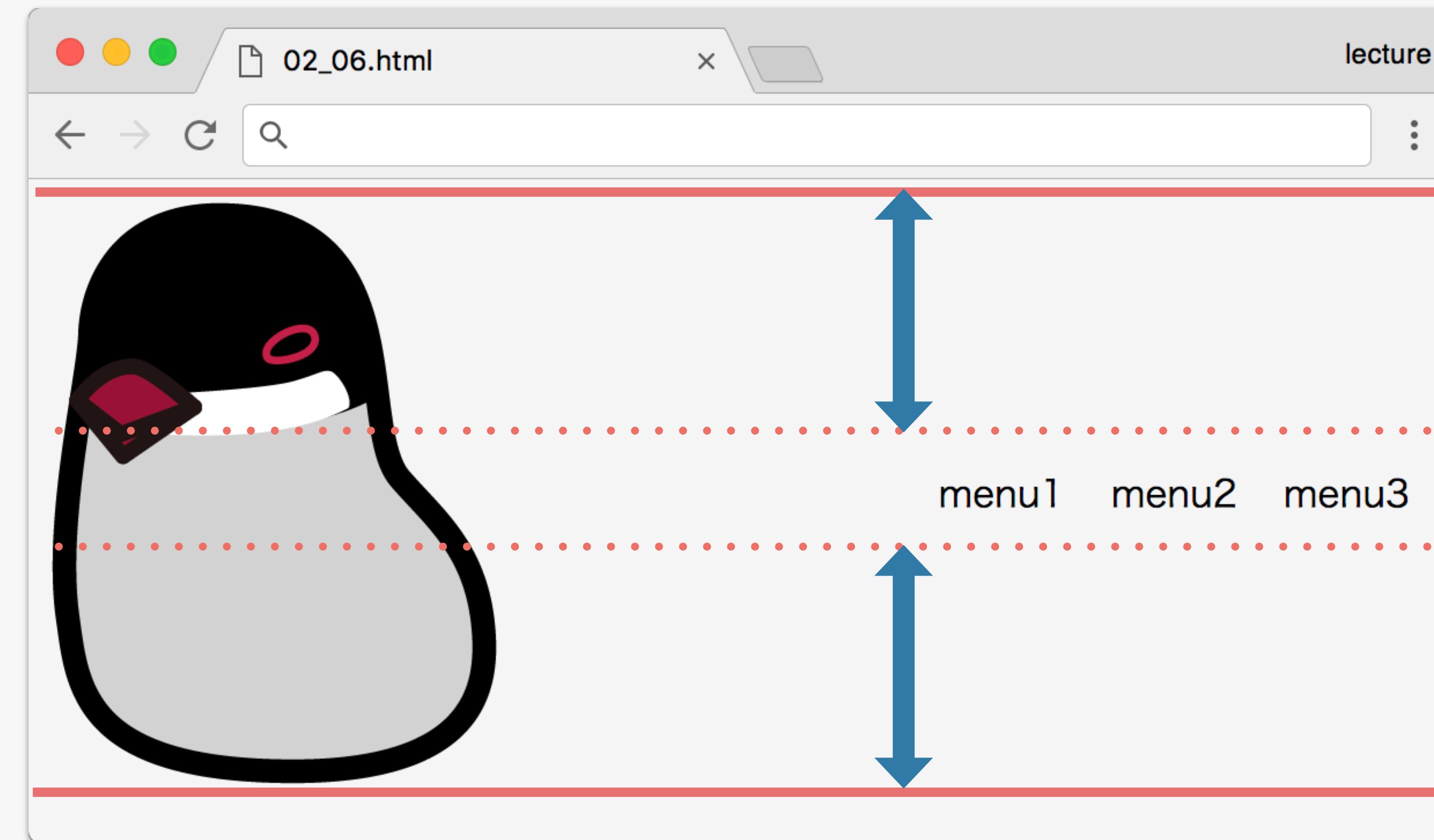
2. JavaScriptの見直し

3. ツールの見直し

ボックスレイアウトにはfloatよりも
FlexboxやCSS Gridを使う

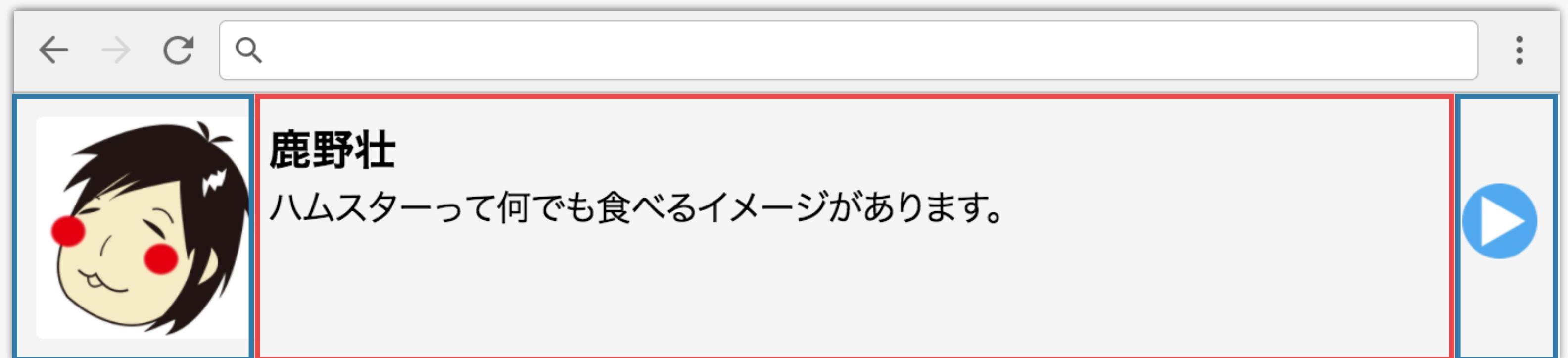
横向き、縦方向の並びや整列

- justify-contentプロパティ
- align-itemsプロパティ



固定幅、可変幅の組み合わせ

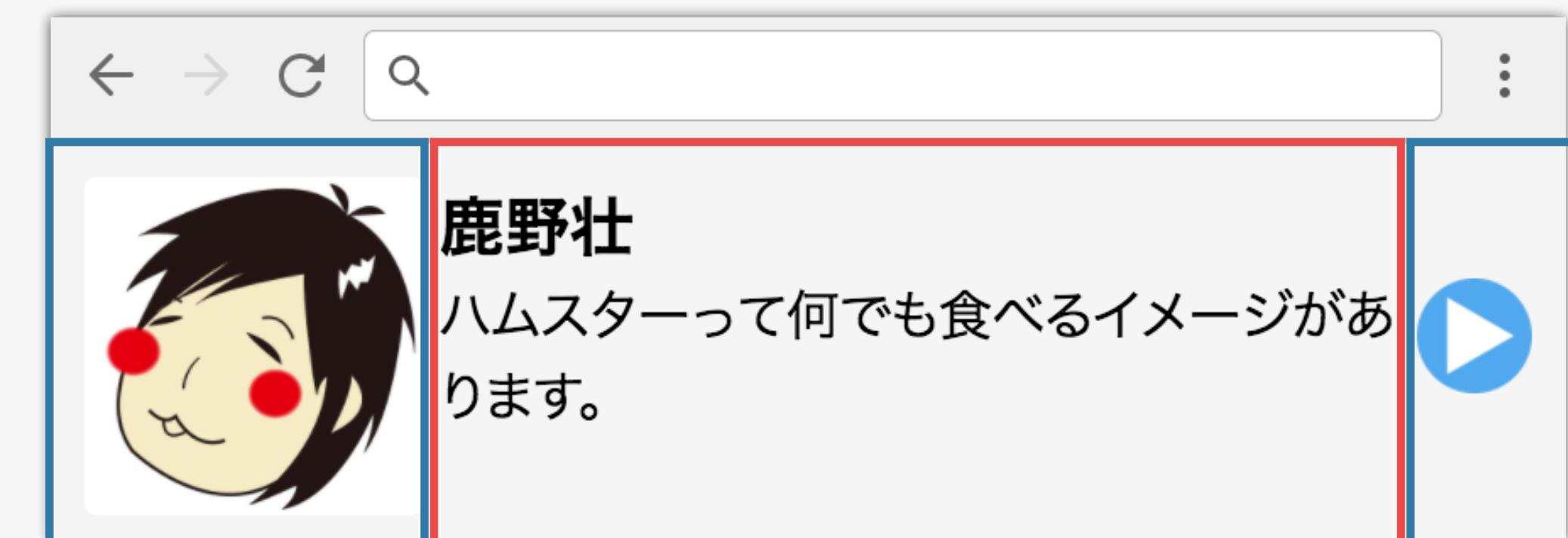
- flex-growプロパティ
- flex-shrinkプロパティ



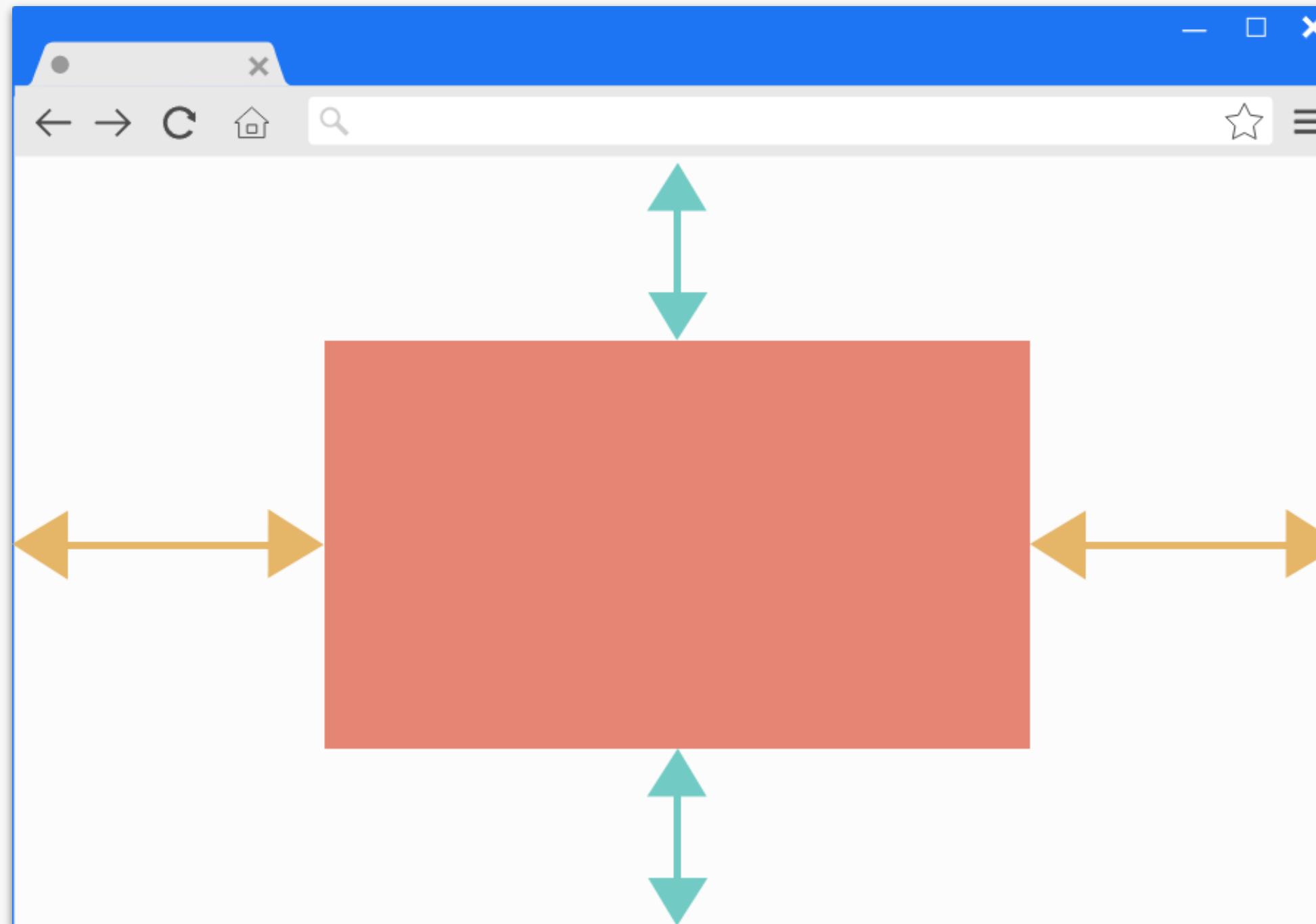
固定幅

可変幅

固定幅



上下中央揃えにも便利

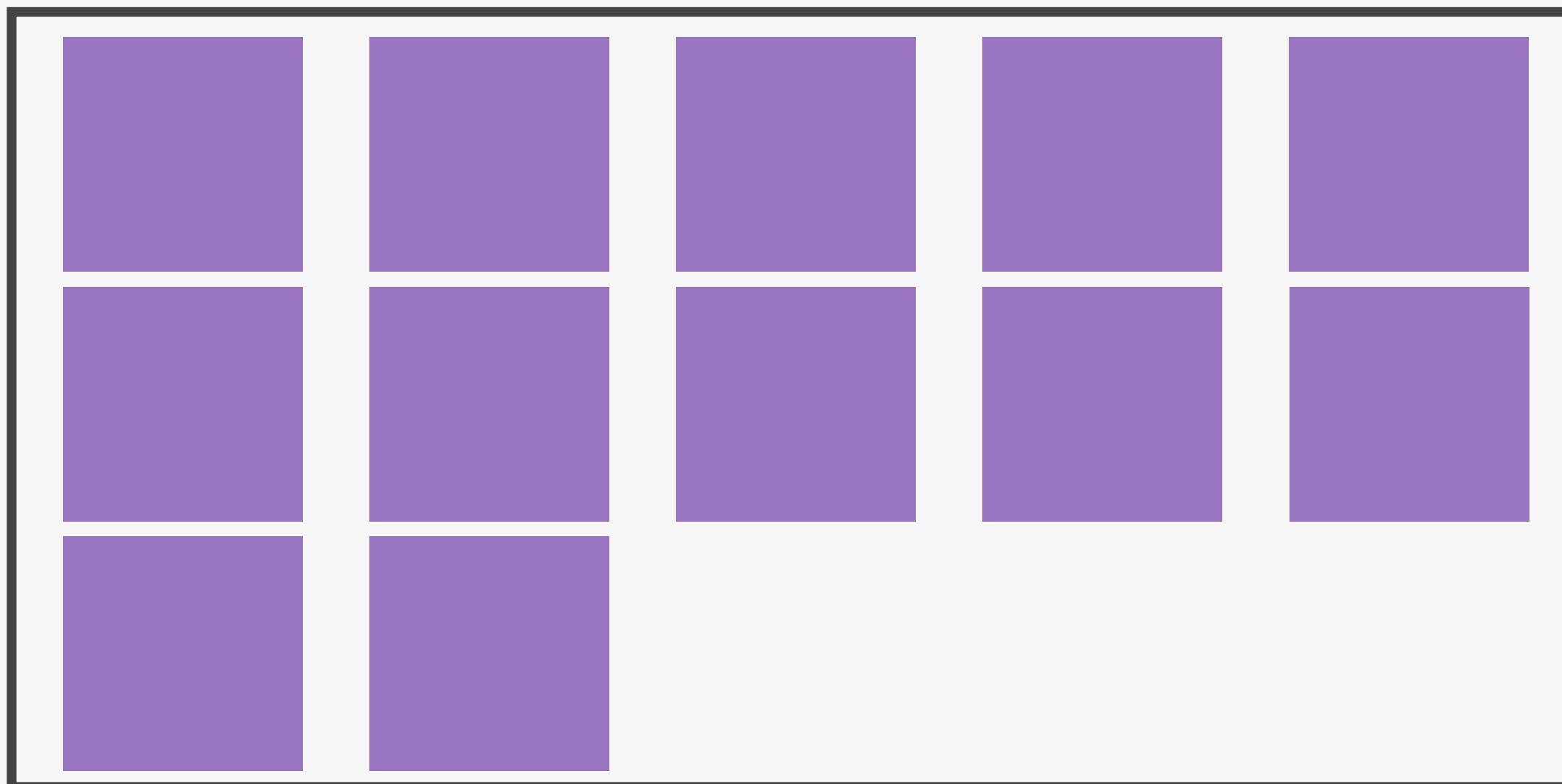


```
.container {  
  display: flex;  
  justify-content: center;  
  align-items: center;  
}
```

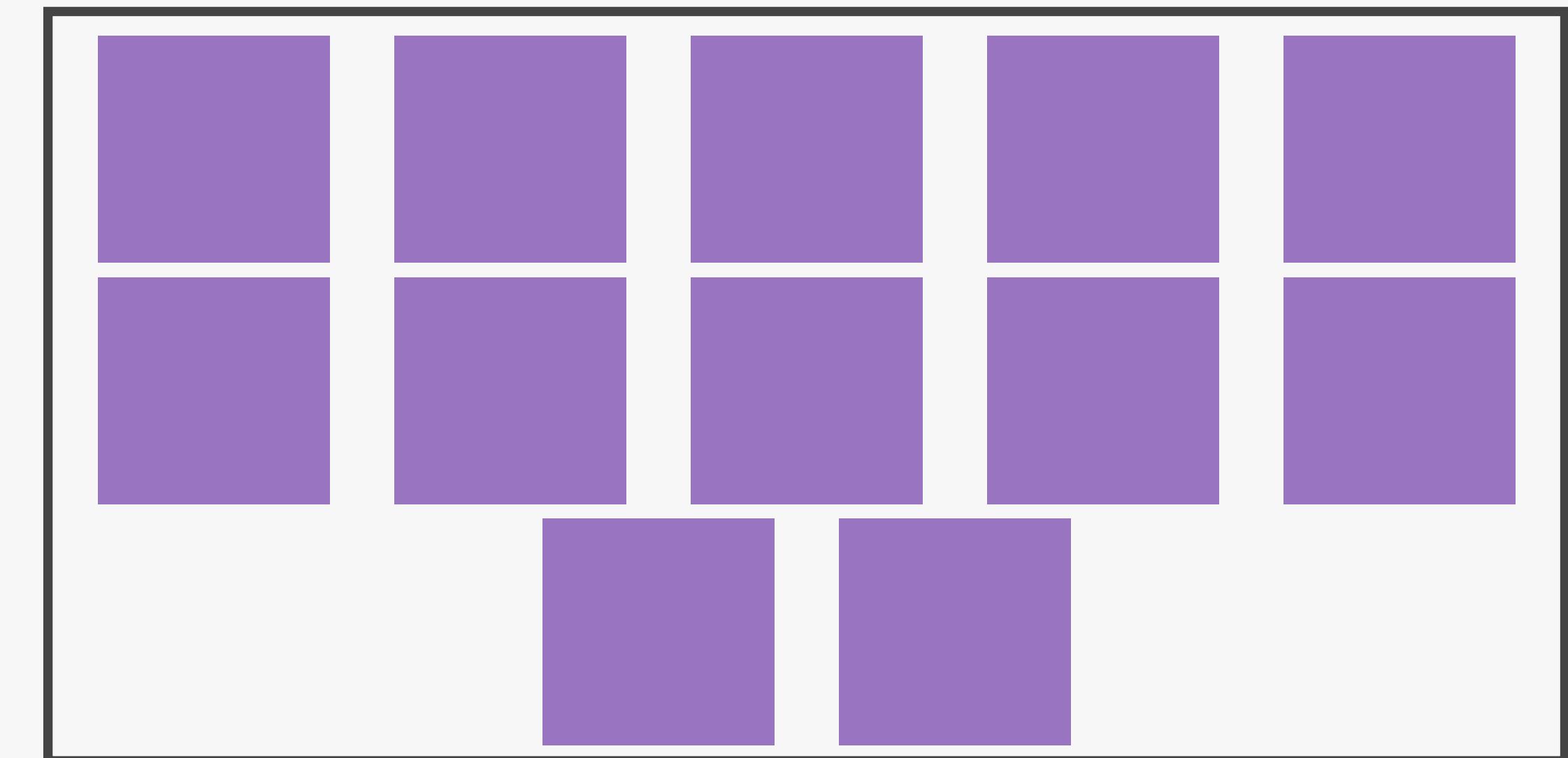
上下中央揃えのCSSまとめ。Flexboxがたった3行で最も手軽

Flexboxは複数行レイアウトがニガテ

`justify-content: space-around`で均等配置しても、
最終行が揃わない



理想

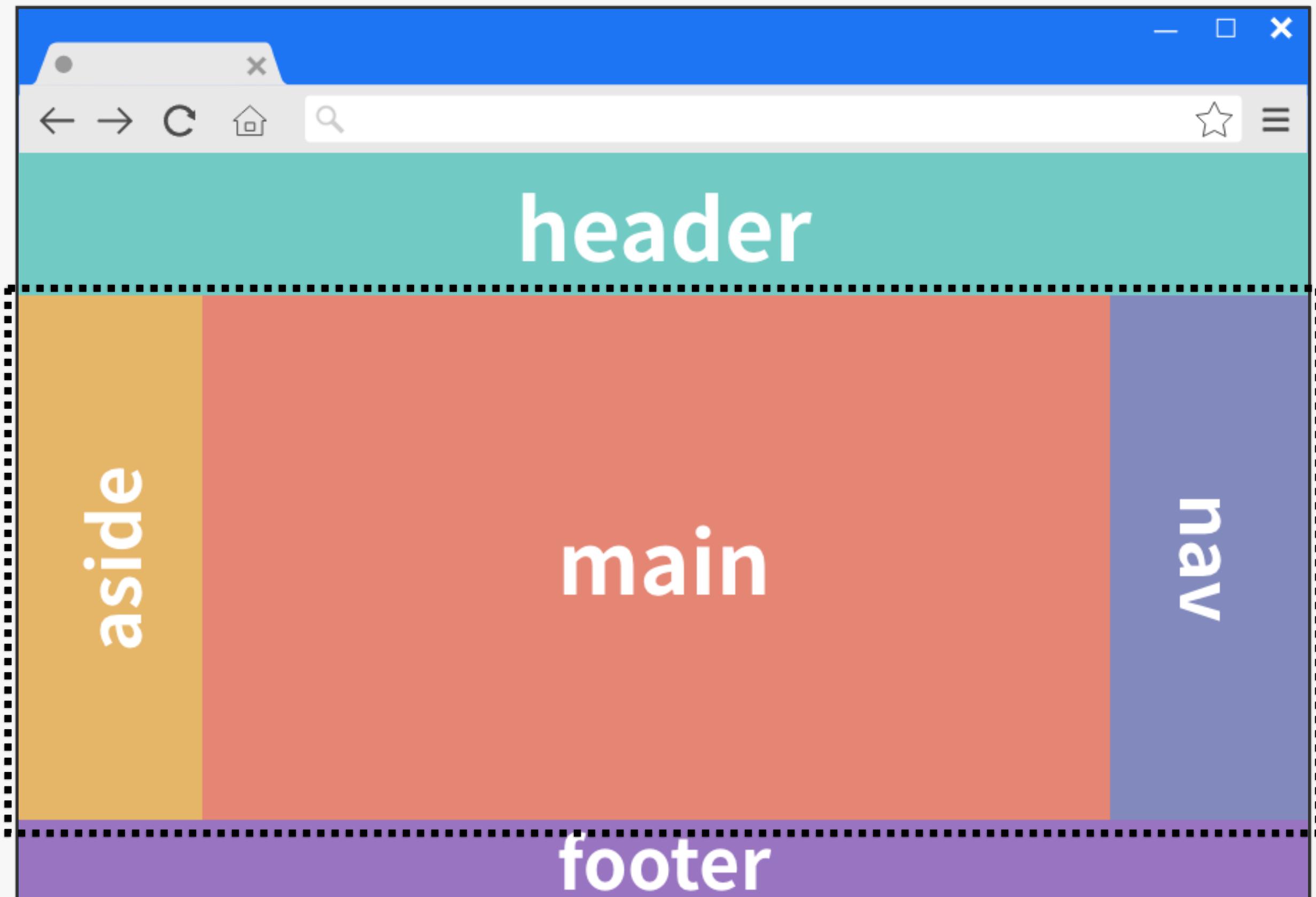


現実

※ 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

Flexboxは複数行レイアウトがニガテ

聖杯レイアウトを実現しようとすると要素の入れ子が必要



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

2019年までに見直しておきたいCSS・JavaScriptの手法

1. CSSの見直し

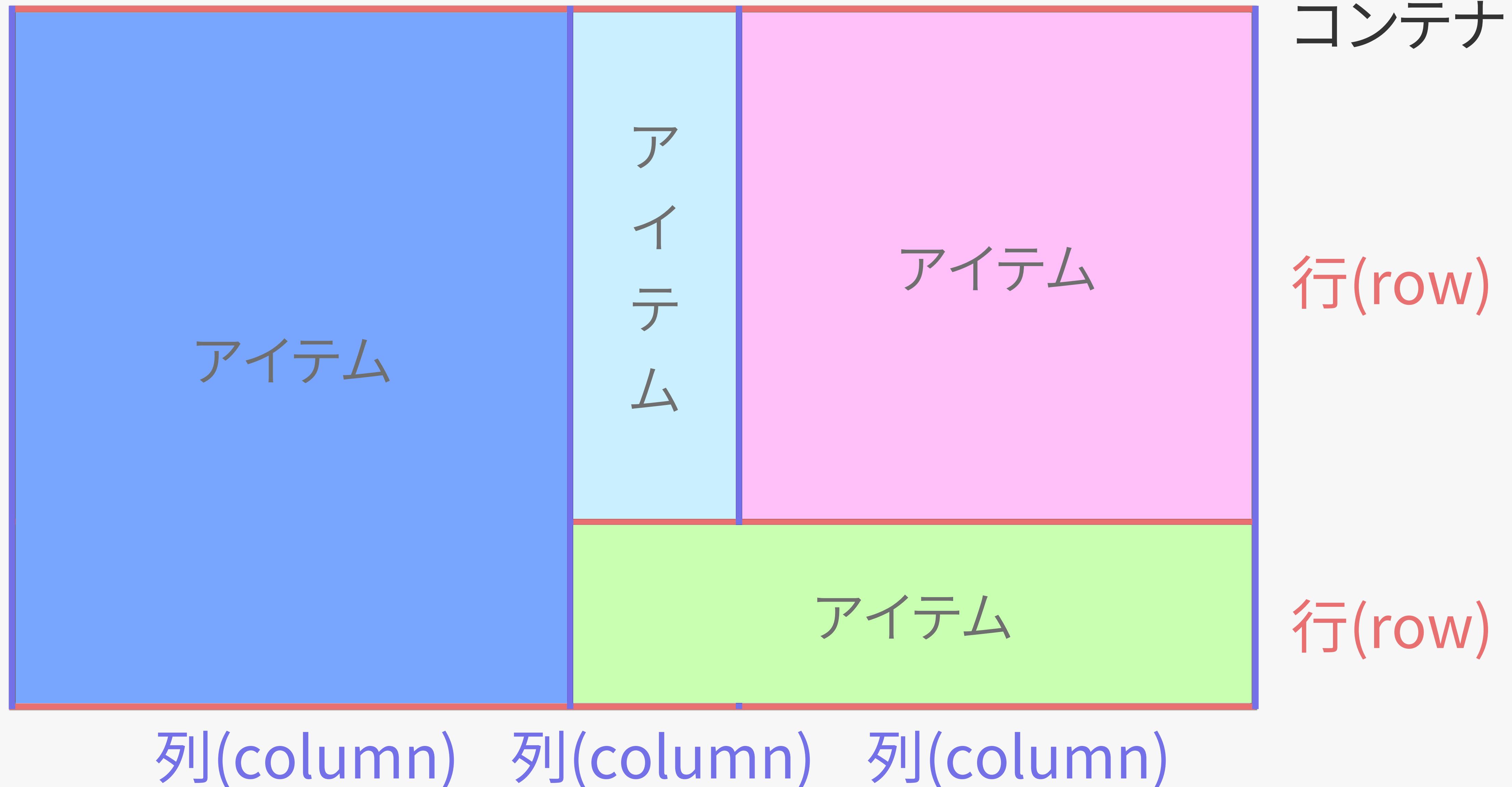
- Flexbox
- CSS Grid
- モダンブラウザで使えるようになった機能
- CSS新機能のキャッチアップ

2. JavaScriptの見直し

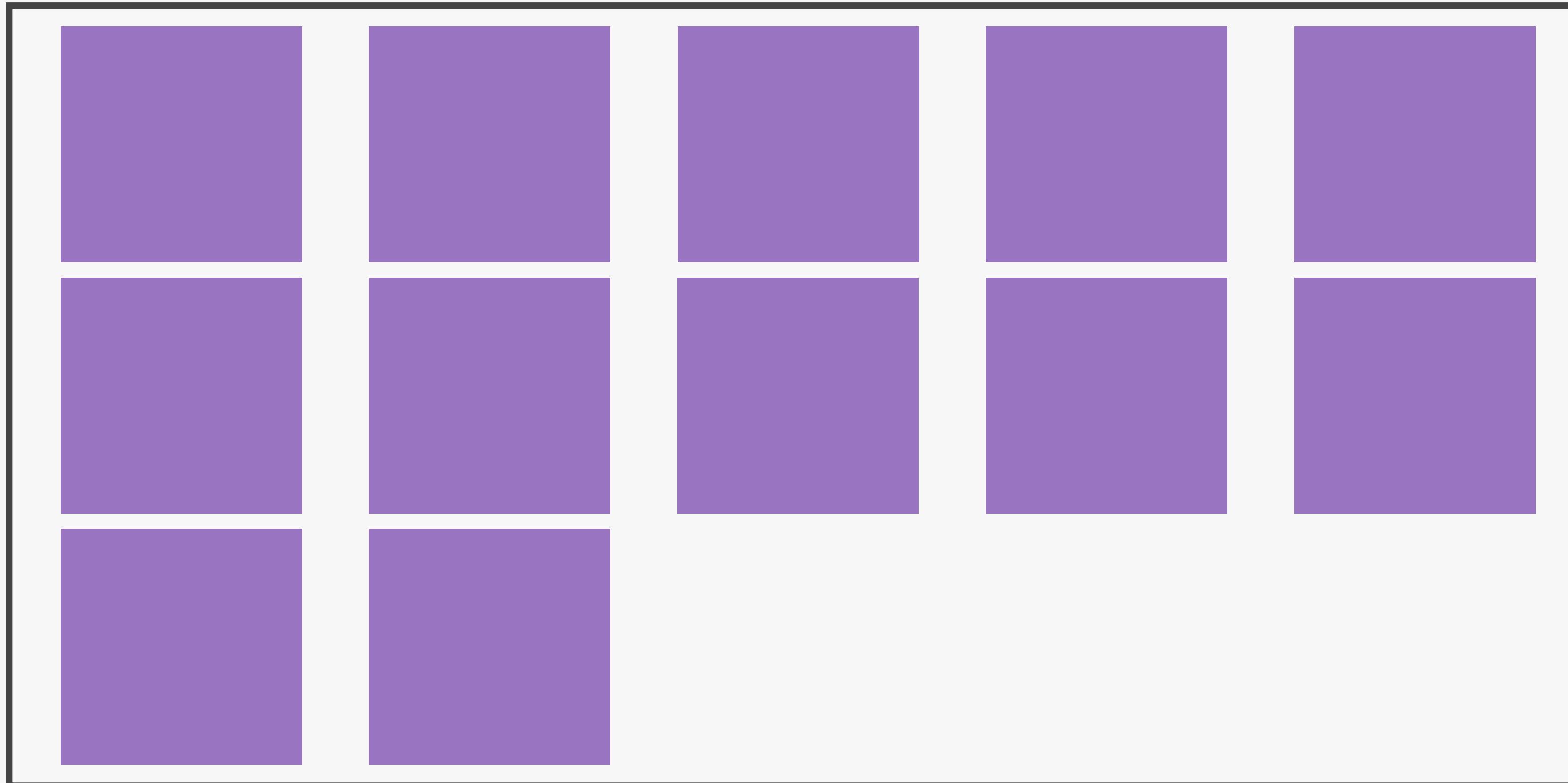
3. ツールの見直し

**複数行のレイアウトには
Flexboxではなく
CSS Gridを使う**

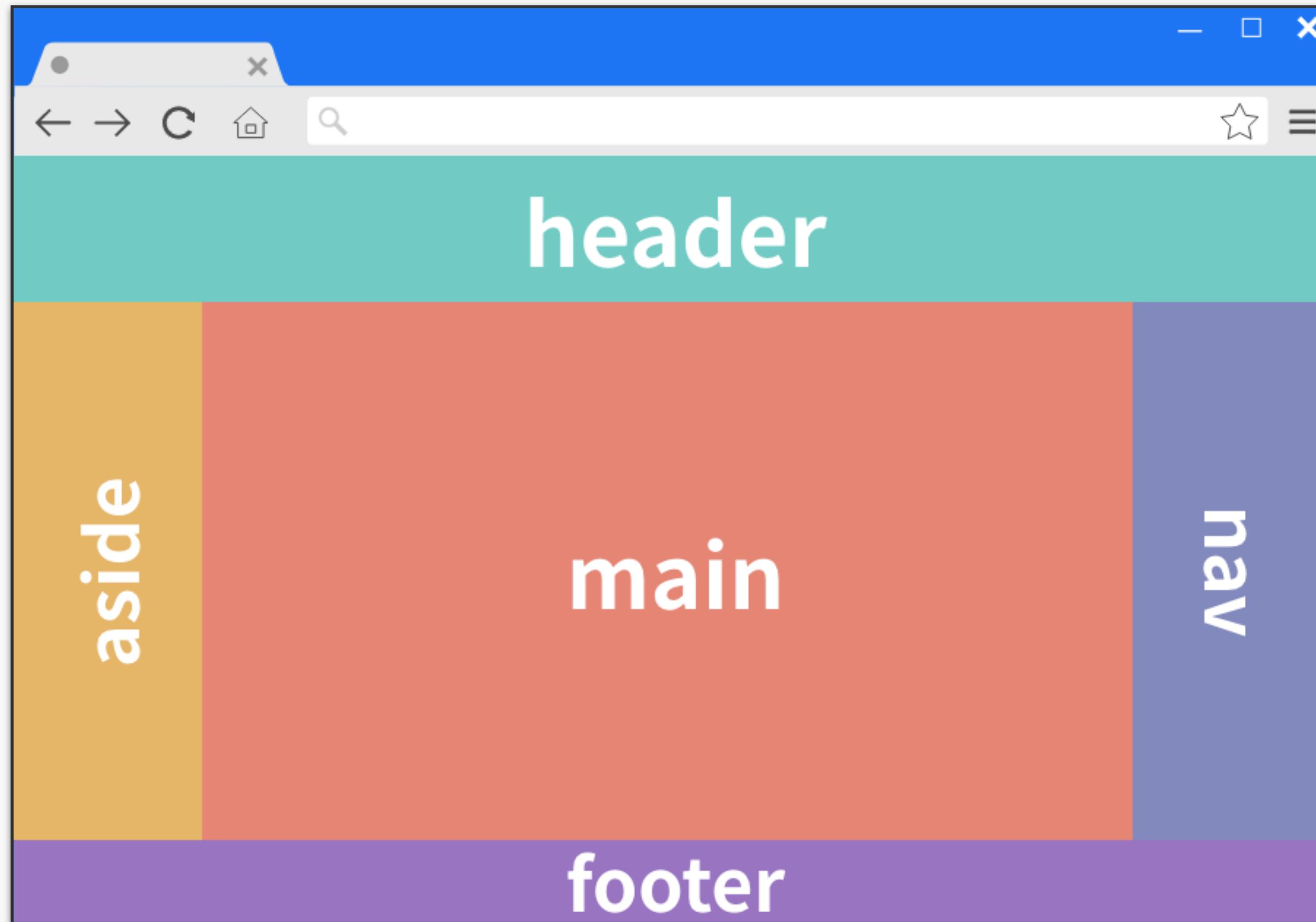
CSS Grid: 行と列を使ったレイアウト



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>
```

Edgeを含む全モダンブラウザで対応済み

							
ブラウザ名	Chrome	Firefox	Safari	Edge	Internet Explorer	Safari (iOS版)	Chrome (Android版)
最新バージョン	71	63	12	18	11	12	71
Grid対応							

※ 部分対応

このページの作り方を通して、CSS Gridの使い方を学びます

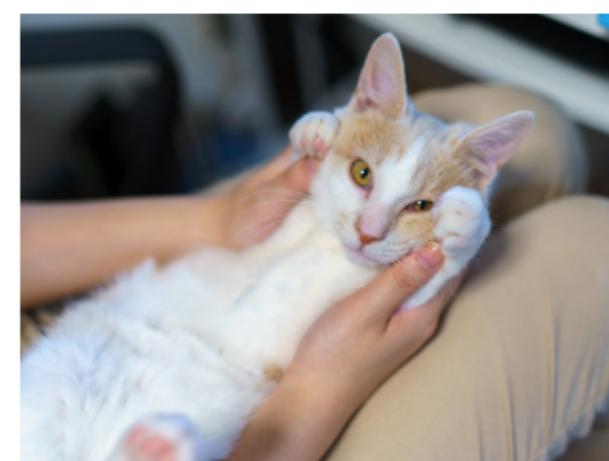
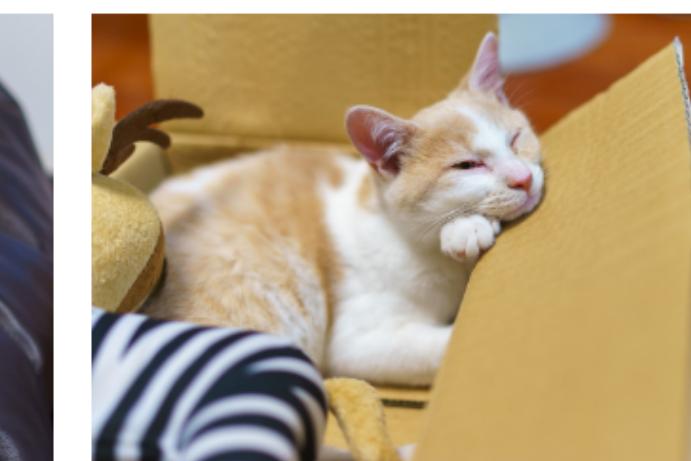
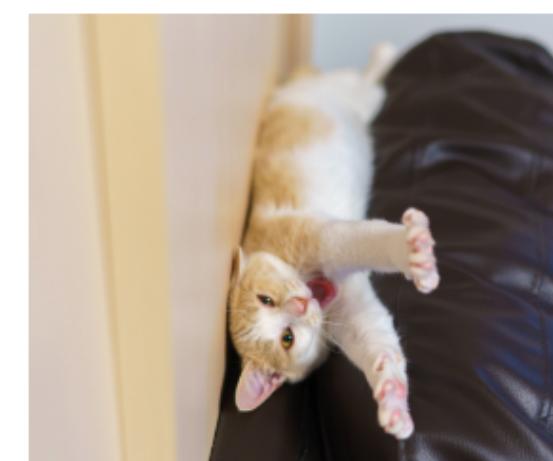


02

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`



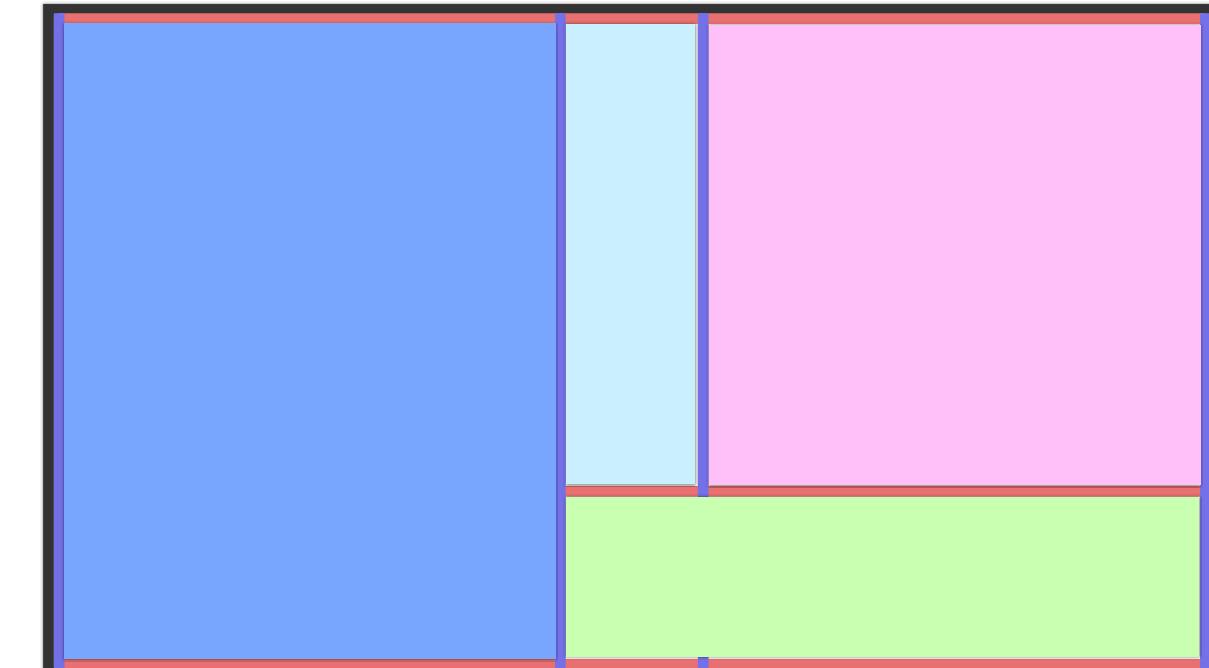
2. 行と列を作る

`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

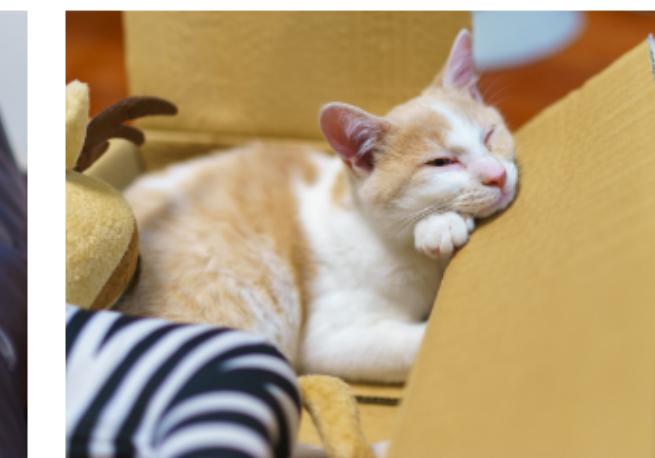
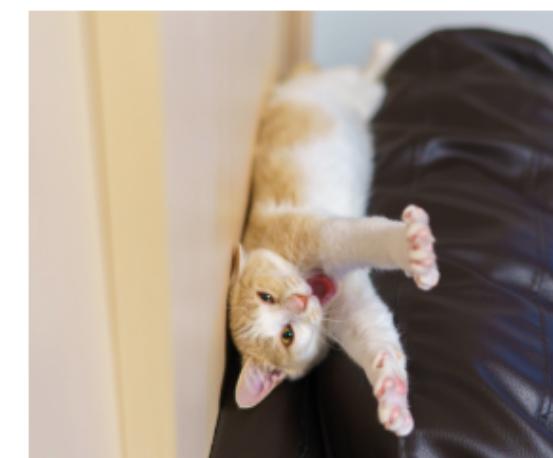


02

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)

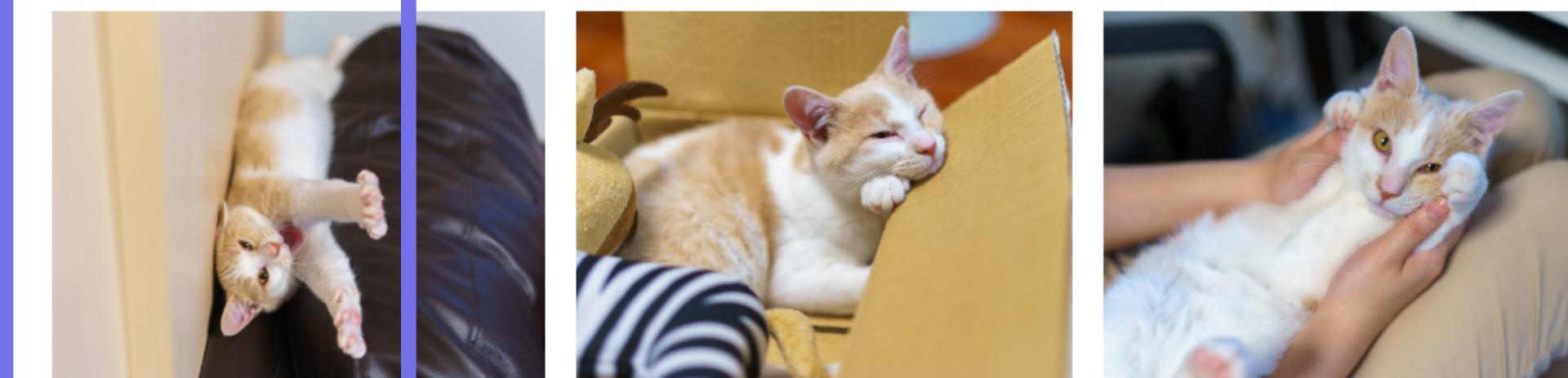


02

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)

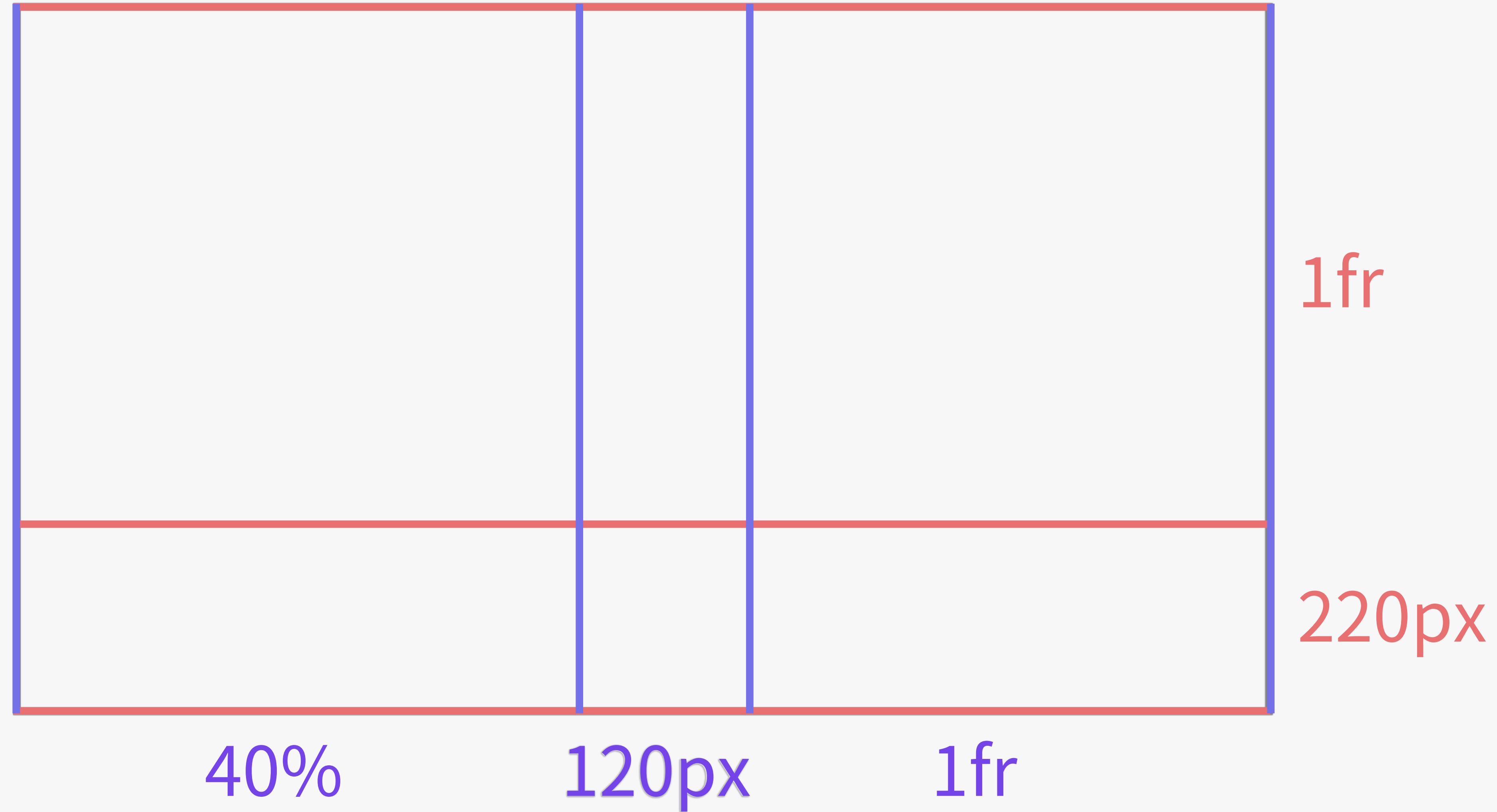
行(row)

列(column)

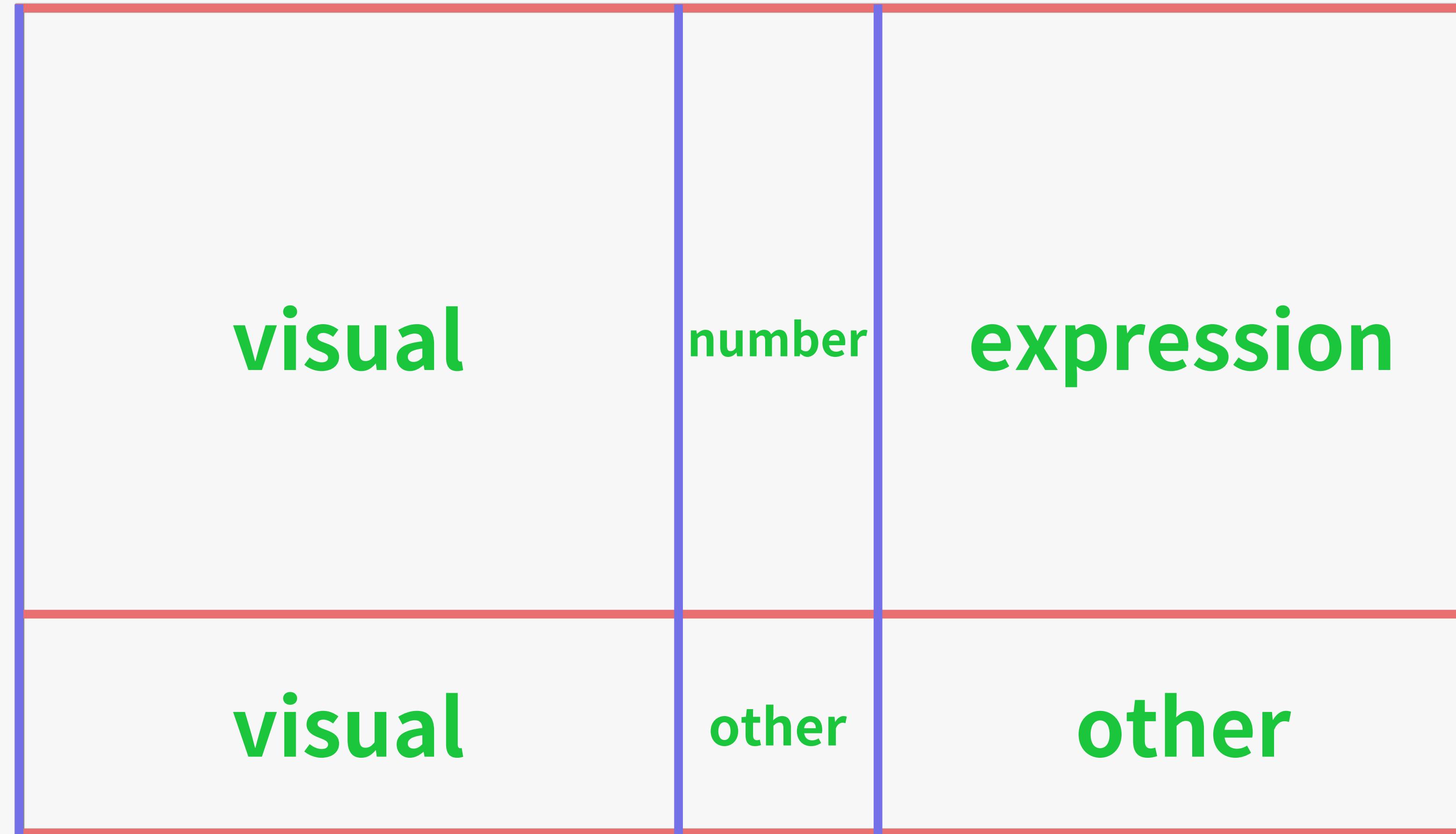
列(column)

列(column)

2-1. 行と列のサイズを決定する



2-2. エリアに名前をつける

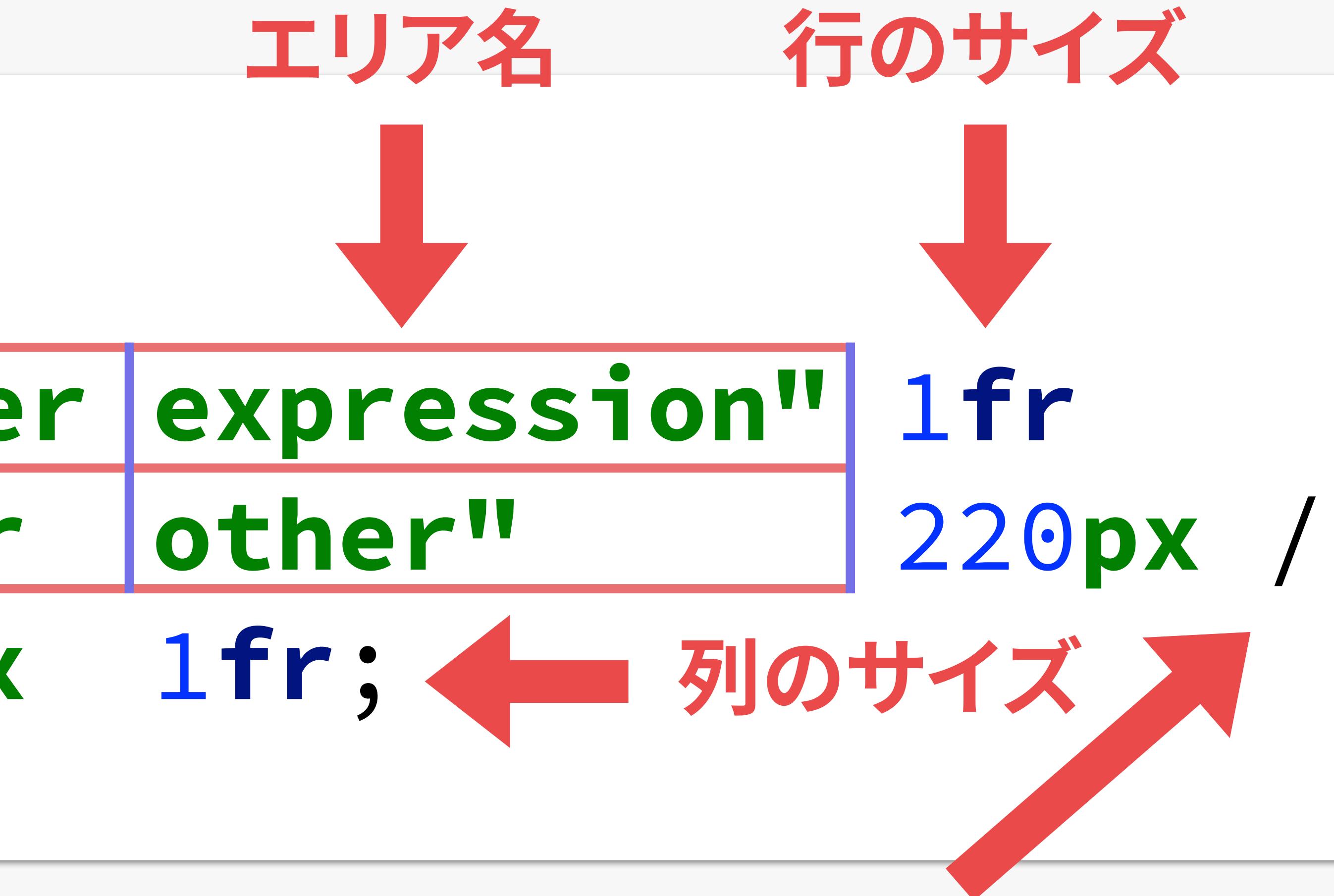


2-3. grid-templateで行列を設定する

```
.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;
```



行と列を区切るスラッシュ

3. アイテムの配置場所の決定

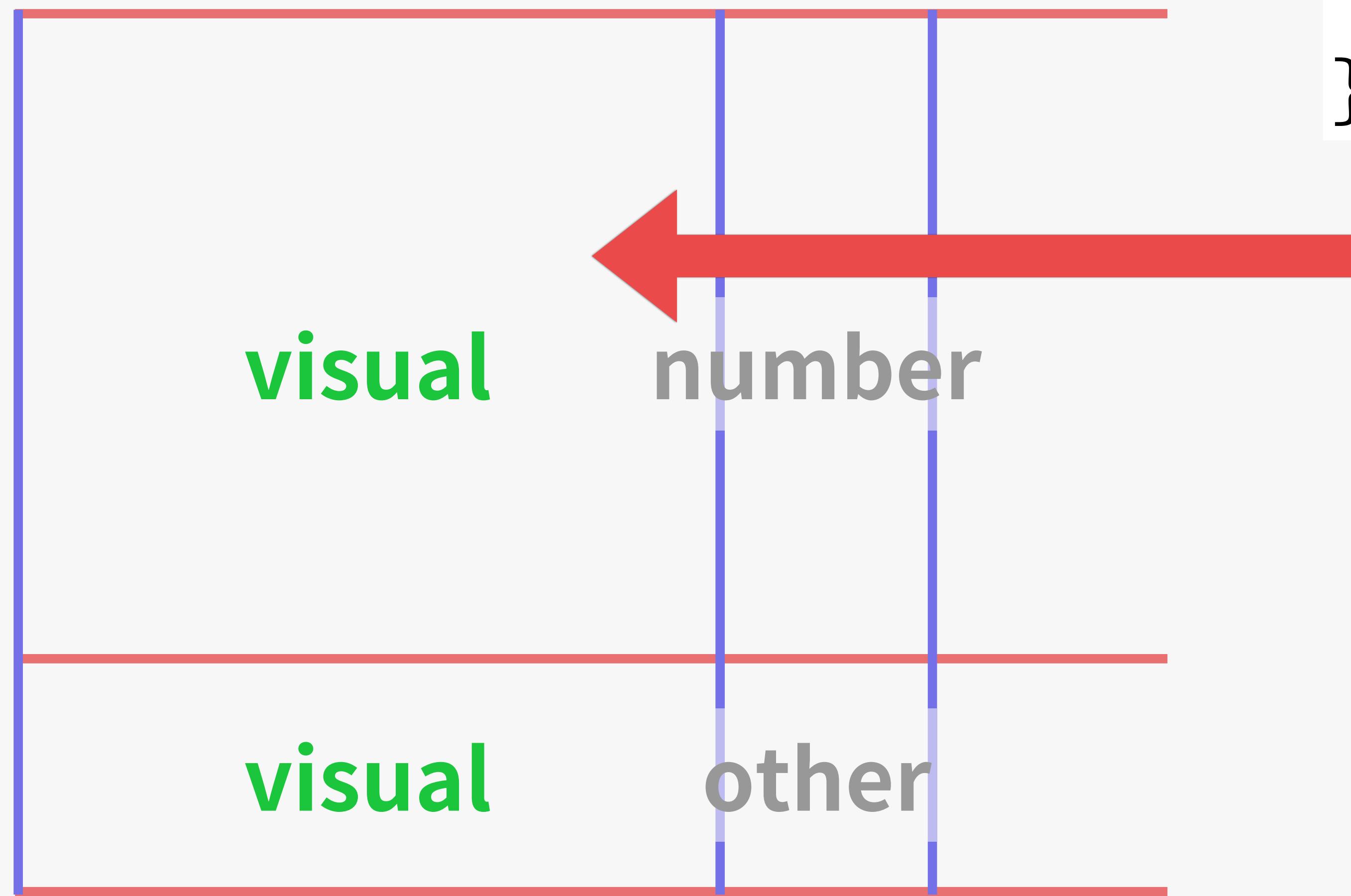
アイテム

アイテ
ム

アイテム

アイテム

3-1. grid-areaで配置したいエリア名を指定



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





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」。

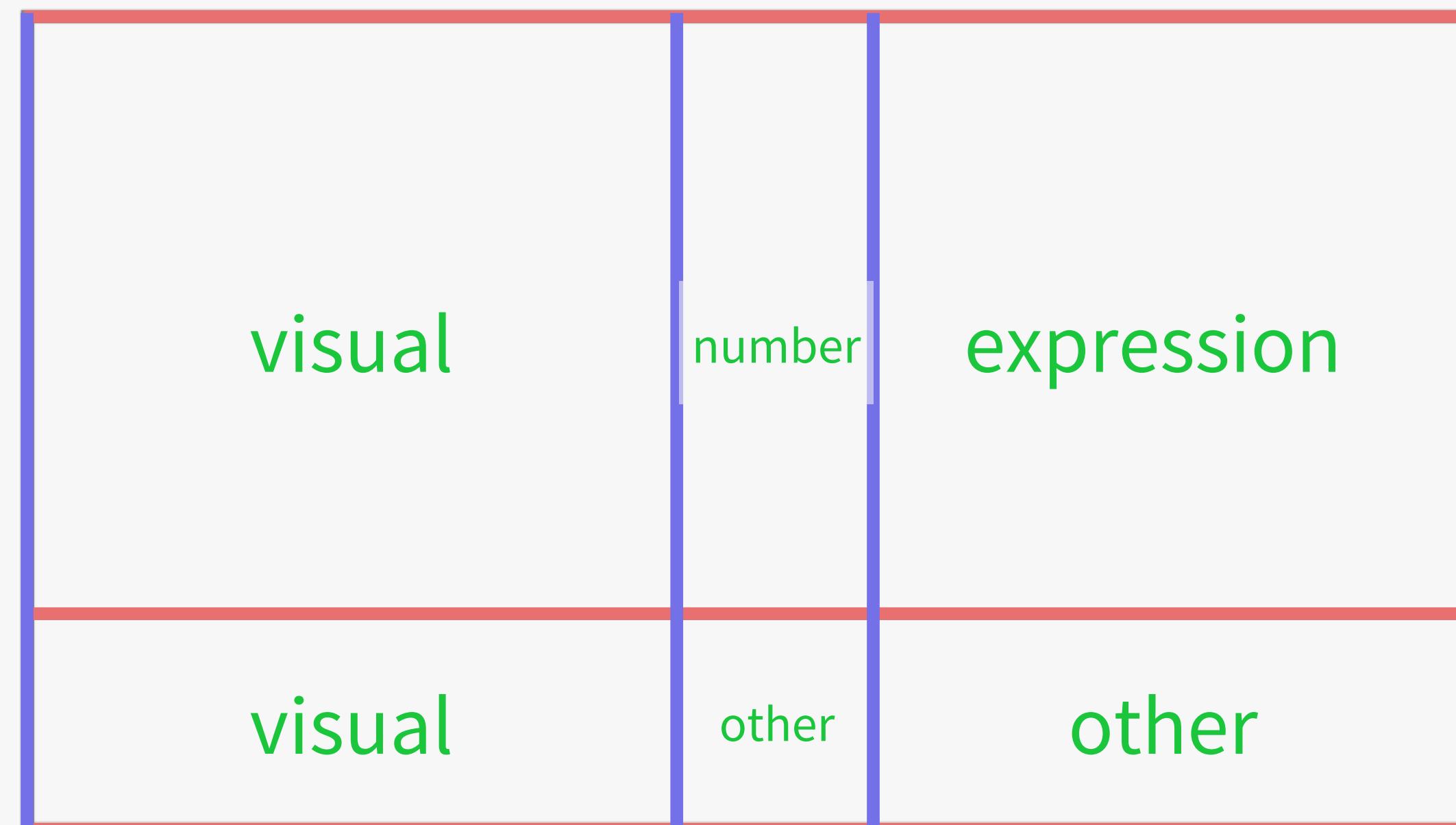
家の中のどこに行ってもついてくる甘えん坊。

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

index.html base.css style.css

```
.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。

```
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
```

```
}
```

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

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

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

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

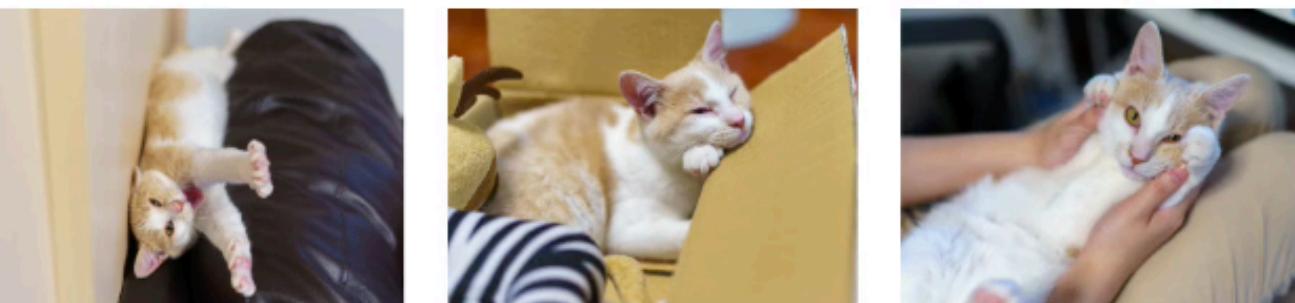
```
.other
```



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。



```
grid-area: visual;

}

.number {
    grid-area: number;
}

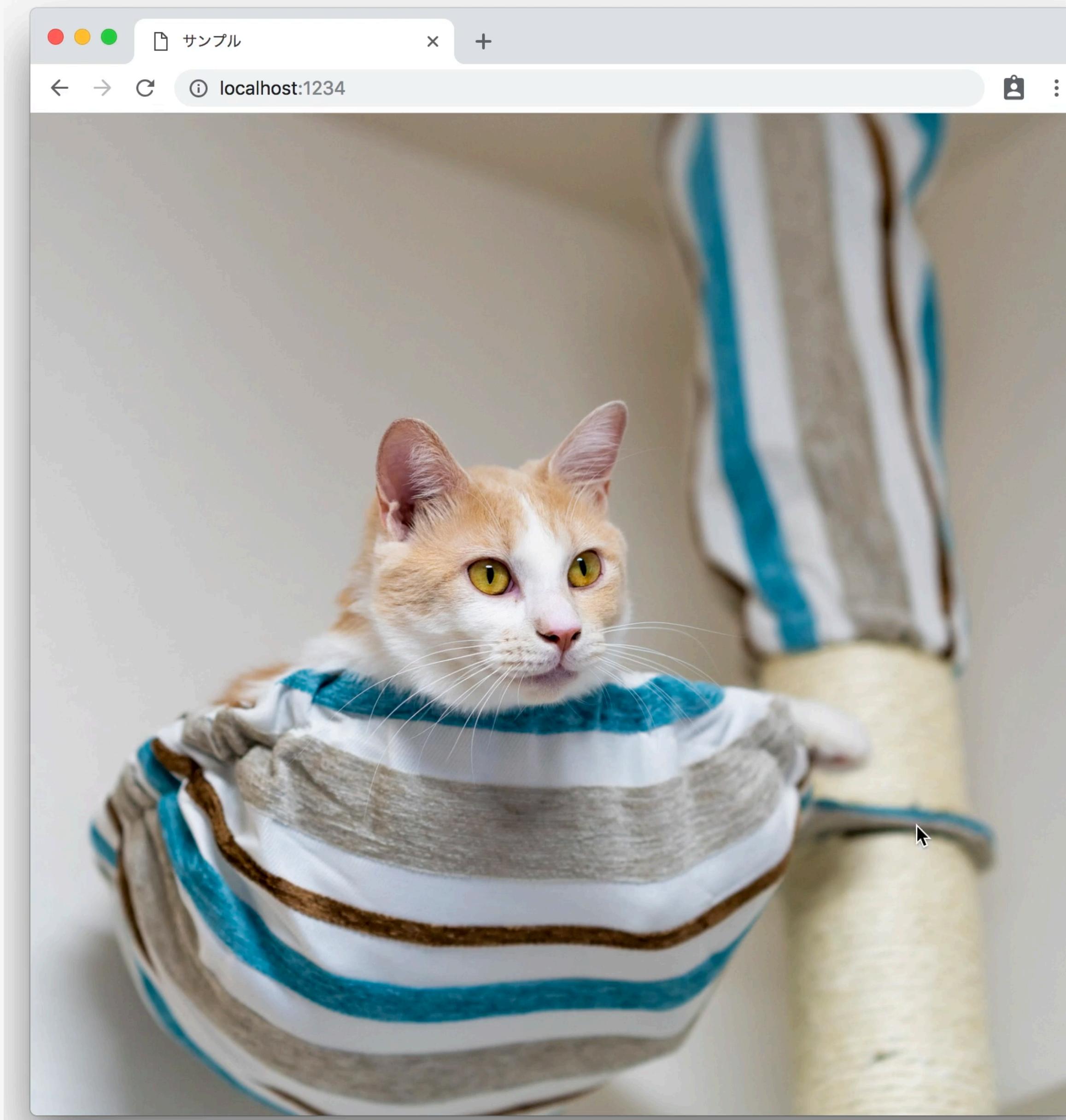
.expression {
    grid-area: expression;
}

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

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  
  }  
}
```



```
samp1 [/Volumes/ICS-KANO01/Dropbox (ICS INC)/kano_ics_seminar/180929_cssnite/project/samp1] - .../css/style.cs...
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

用途別レイアウトの使い分け

テキストの回り込み

float



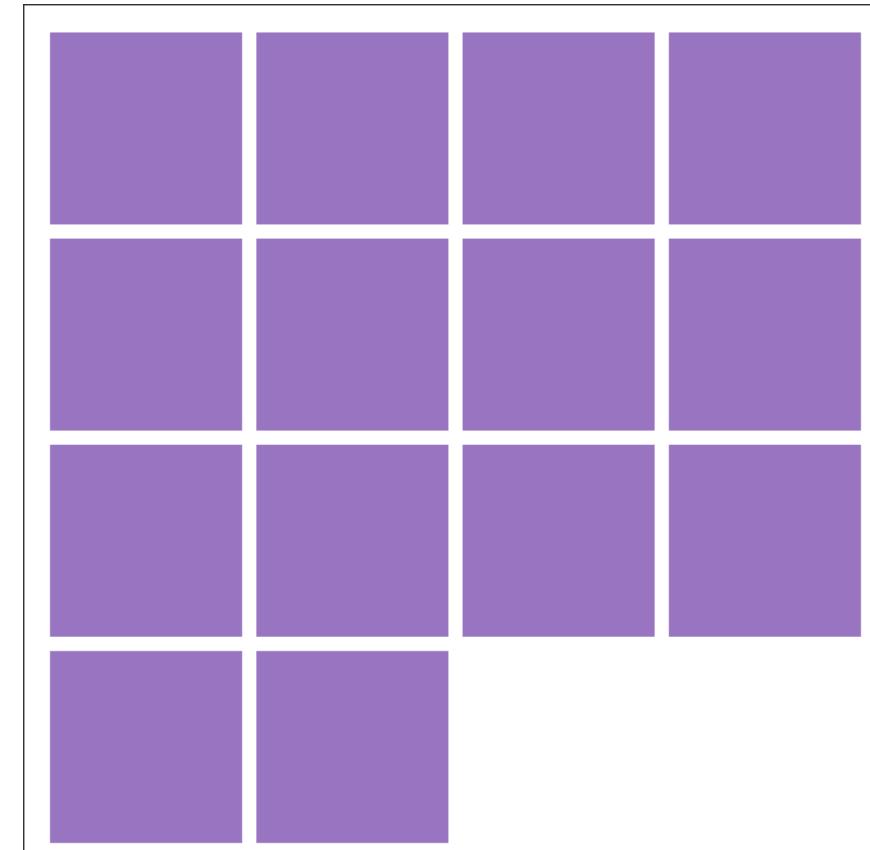
1行の横並び、縦並び

Flexbox



格子状のレイアウト
ページ全体のレイアウト

Grid



2019年までに見直しておきたいCSS・JavaScriptの手法

1. CSSの見直し

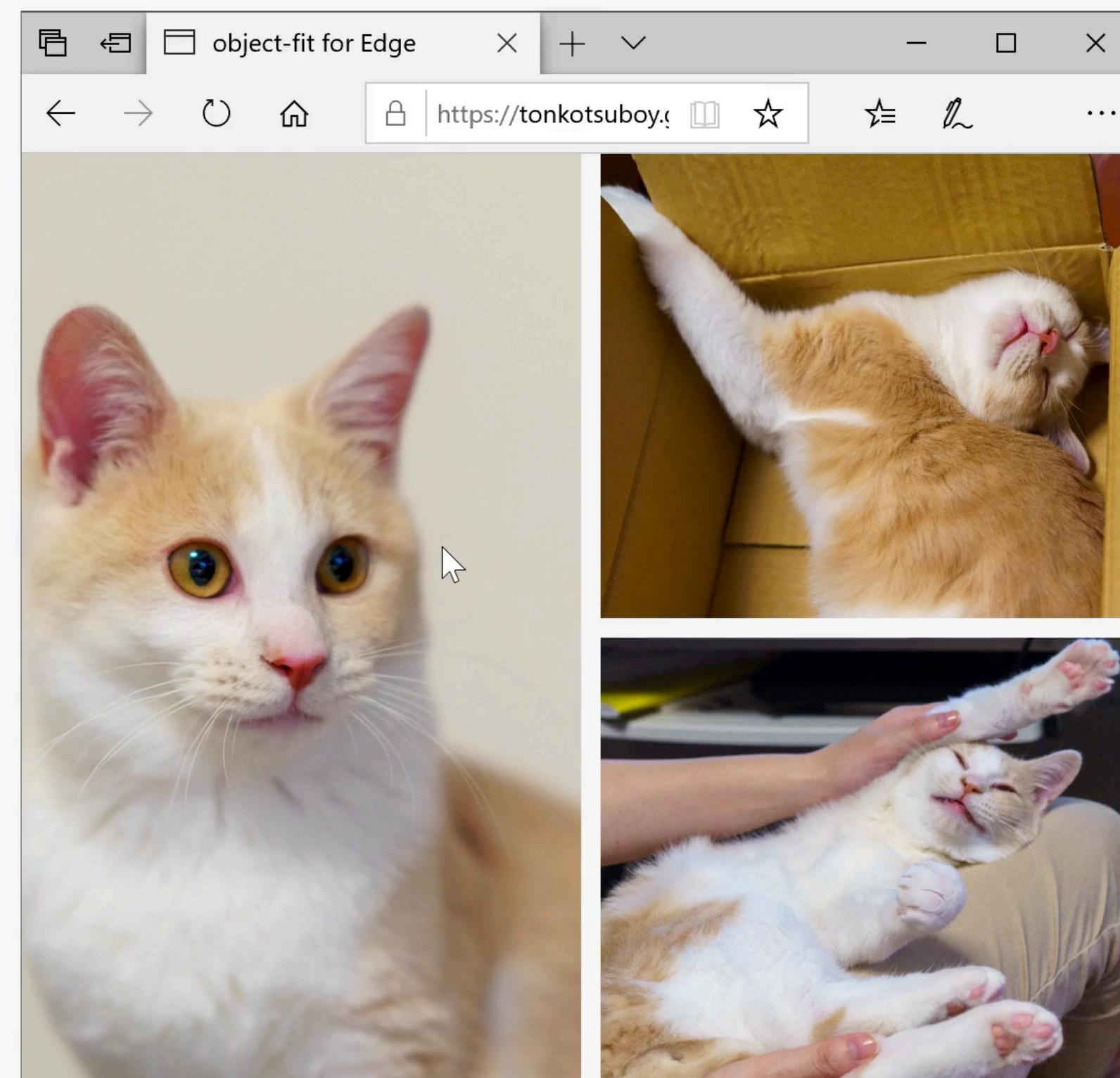
- Flexbox
- CSS Grid
- モダンブラウザで使えるようになった機能
- CSS新機能のキャッチアップ

2. JavaScriptの見直し

3. ツールの見直し

**Microsoft Edgeの対応により
全モダンブラウザーで使えるようになった
CSSの機能**

imgタグで設定した画像の比率を保つ



HTMLコード

```



```

imgタグで設定した画像の比率を保つ

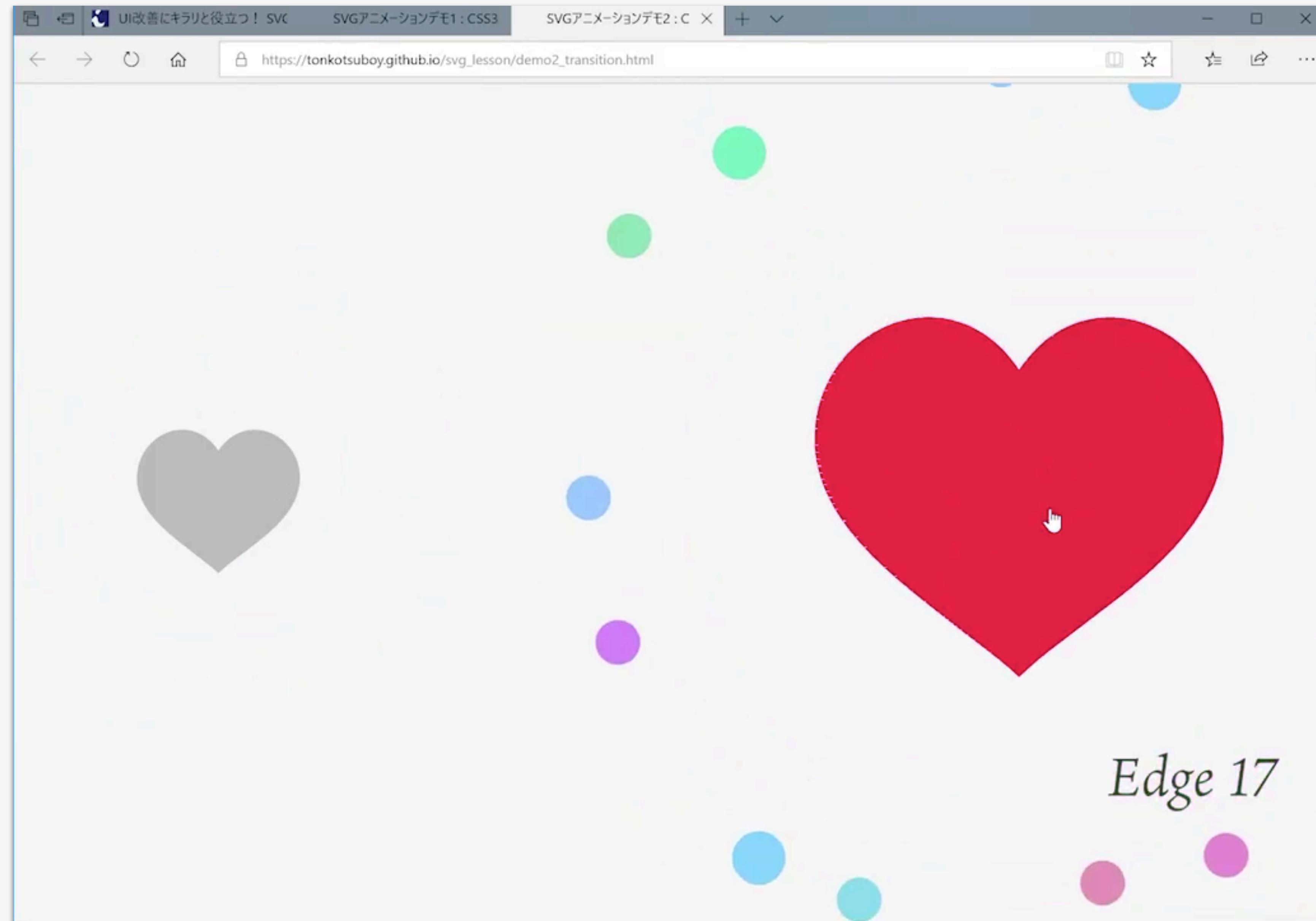
object-fitプロパティー

(2017年10月リリースのEdge v16で対応)

```
img {  
  object-fit: cover;  
}  
.image1 {  
  object-position: 30% 50%;  
}  
.image2 {  
  object-position: 0 0;  
}
```

うに fit

SVG要素をCSS Transformで変形する



SVG要素をCSS Transformで変形する

SVG要素をtransformで変形する

(2018年4月リリースのEdge v17で対応)

```
/* SVGのハート図形のアニメーション（抜粋） */  
  
@keyframes heartAnime {  
    0% {  
        transform: scale(0);  
    }  
    60% {  
        transform: scale(1.2, 1.2);  
    }  
    80% {  
        transform: scale(0.95, 1.05) translate(0%, -3%);  
    }  
    100% {  
        transform: scale(1.0, 1.0) translate(0%, 0%);  
    }  
}
```

画像にマスクをかける



画像にマスクをかける

maskプロパティー

(2018年10月リリースのEdge v18で対応)

```
.mask {  
  mask-image: url("../images/mymask.png");  
  mask-repeat: no-repeat;  
  mask-position: center;  
  mask-size: contain;  
}
```

#自分のお金で寿司が食べたい

モダンブラウザのCSS新機能の対応状況(抜粋)

Mixed support	Edge 18	Firefox 63	Chrome 70	Safari 12
CSS touch-action property	Yes	Yes	Yes	No
CSS widows & orphans	Yes	No	Yes	Yes
Media Queries: interaction media features	Yes	No	Yes	Yes
CSS zoom	Yes	No	Yes	Yes
CSS line-clamp	Yes	No	Yes <small>-webkit-</small>	Yes <small>-webkit-</small>
CSS color-adjust	No	Yes	Yes <small>-webkit-</small>	Yes <small>-webkit-</small>
CSS caret-color	No	Yes	Yes	Yes
CSS text-orientation	No	Yes	Yes	Yes <small>-webkit-</small>
#rrggbaa hex color notation	No	Yes	Yes	Yes
CSS font-rendering controls	No	Yes	Yes	Yes
:focus-within CSS pseudo-class	No	Yes	Yes	Yes
:default CSS pseudo-class	No	Yes	Yes	Yes
Case-insensitive CSS attribute selectors	No	Yes	Yes	Yes
CSS all property	No	Yes	Yes	Yes
:placeholder-shown CSS pseudo-class	No	Yes	Yes	Yes
CSS will-change property	No	Yes	Yes	Yes
CSS background-blend-mode	No	Yes	Yes	Yes
CSS3 font-kerning	No	Yes	Yes	Yes
CSS Shapes Level 1	No	Yes	Yes	Yes
CSS resize property	No	Yes	Yes	Yes
CSS Font Loading	No	Yes	Yes	Yes
Crisp edges/pixelated images	No	Yes <small>-moz-</small>	Yes	Yes
CSS overscroll-behavior	Partial	Yes	Yes	No
CSS3 text-align-last	Partial	Yes	Yes	No
Blending of HTML/SVG elements	No	Yes	Yes	Partial
CSS3 tab-size	No	Yes <small>-moz-</small>	Yes	Partial <small>-webkit-</small>
CSS Logical Properties	No	Yes	Yes	Partial <small>-webkit-</small>
text-emphasis styling	No	Yes	Partial <small>-webkit-</small>	Yes

 対応済
 一部対応
 未対応

Can I useで各モダンブラウザのCSS対応状況を比較した結果

EdgeがChromiumベースになる

CSSの新機能の普及が進む可能性がある

The screenshot shows a blog post from the Microsoft Windows Blogs. The title is "Microsoft Edge: Making the web better through more open source collaboration" by Joe Belfiore. The post discusses Microsoft's increased participation in the open source community and its intention to adopt the Chromium open source project for Microsoft Edge. It aims to create better web compatibility and reduce fragmentation. The post includes social sharing buttons for Facebook, Twitter, Reddit, LinkedIn, and Skype.

DECEMBER 6, 2018 9:00 AM

Microsoft Edge: Making the web better through more open source collaboration

By [Joe Belfiore](#) / Corporate Vice President, Windows

[f SHARE](#) [TWEET](#) [SHARE](#) [in SHARE](#) [S SKYPE](#)

For the past few years, Microsoft has meaningfully increased participation in the open source software (OSS) community, becoming one of the world's largest supporters of OSS projects. Today we're announcing that we intend to adopt the Chromium open source project in the development of Microsoft Edge on the desktop to create better web compatibility for our customers and less fragmentation of the web for all web developers.

As part of this, we intend to become a significant contributor to the Chromium project, in a way that can make not just Microsoft Edge — but other browsers as well — better on both PCs and other devices.

Windows Experience Blog (2018年12月)

ウェブ業界では称賛と懸念の両方の声があがっている

together  App Store Google Play キーワードを入力 検索
アプリ限定の機能知ってる? キーワード ユーザ名

① 限定公開でまとめを作れば、相互フォローやフォロワー限定でまとめを共有できます!

注目まとめ 今週の人気 新着まとめ イチオシ 編集部厳選 Togetterのリリースノート一覧

まとめトップ > カテゴリー > IT・Web 22時間前

Microsoft EdgeがChromiumベースになると公式に発表。多様性が失われるウェブへの懸念と祝福の声と

Microsoft EdgeがChromium(Chromeの元になっているもの)ベースになると公式が発表しました。

ウェブ業界では称賛と懸念の両方の声があがっています。

IE Chrome EDGE chromium Microsoft 多様性 

clockmaker 9709view 33コメント  138  42  14  お気に入り

2D6  42

まとめ 

ものえおさむ @osamum_MS 思ったより早くアナウンスが出たなア。
- Microsoft Edge: Making the web better through more open source collaboration - blogs.windows.com/windowsexperie... 2018-12-07 03:03:00

dynamis (でゅなみす) @dynamitter EdgeをChromiumベースのOSSブラウザとして作り直すものに切り替えて来年早期にはプレビュービルドを提供すると公式発表。
OSSになるのはめでたいのだけど、ブラウザエンジンの多様性が無くなるのは残念としか言えない。
blogs.windows.com/windowsexperie... 2018-12-07 04:29:37

深津 貴之 / THE GUILD @fladdict これ業界全体で大感謝ひらいて、一日中歌い踊ってシャンパンあけて良いやつ。企画したら日本中のフロントエンジニアが来たがるはず。大英断。
twitter.com/fladdict/statu... 2018-12-07 09:29:47

Microsoft EdgeがChromiumベースになると公式に発表。
多様性が失われるウェブへの懸念と祝福の声と - Togetter

2019年までに見直しておきたいCSS・JavaScriptの手法

1. CSSの見直し

- Flexbox
- CSS Grid
- モダンブラウザで使えるようになった機能
 - CSS新機能のキャッチアップ

2. JavaScriptの見直し

3. ツールの見直し

**どうやってCSSの
最新機能や対応状況を追うか？**

各ブラウザーの公式ブログ

- Chromium Blog
- Chrome Releases
- WebKit
- The Mozilla Blog
- Microsoft Edge Blog

Can I useのCompare browsers

[Home](#)[News](#)

November 29, 2018 - New feature: Picture-in-Picture

[Compare browsers](#)[About](#)

Can I use

[?](#) [Settings](#)

Detected your country as "Japan". Would you like to import usage data for that country?

[Import](#)[No thanks](#)

Index of features

Latest features

- Picture-in-Picture
- BigInt
- CSS overflow property
- CSS Environment Variables env()
- Promise.prototype.finally

Most searched features

1. Flexbox
2. CSS Grid
3. SVG
4. CSS transforms
5. CSS Filter Effects

Did you know?

You can import usage data from your **Google Analytics** account and see exactly how well a feature is supported among your own site's visitors. Look under the **Settings panel** to get started!

[Next](#)[Can I use...](#)

2019年までに見直しておきたいCSS・JavaScriptの手法

1. CSSの見直し
2. JavaScriptの見直し
3. ツールの見直し

ES2015は登場してから3年経った

ES2015 = ECMAScript2015

ECMAScript = JavaScriptの仕様

ES2015の新機能の一例

- `let, const`
- ブロックスコープ
- `String, Number, Array, Object`などの新しいAPI
- `Map, Set`
- `Promise`
- クラス構文
- テンプレート文字列
- アロー関数
- イテレータ、ジェネレータ
- `for of`

ES2015以降もJavaScriptは進化を続いている

バージョン	リリース時期
ES2015	2015年6月
ES2016	2016年6月
ES2017	2017年6月
ES2018	2018年6月

2019年までに見直しておきたいCSS・JavaScriptの手法

1. CSSの見直し

2. JavaScriptの見直し

1. ES2016～ES2018の新機能

2. 2019年以降に追加される機能

3. ツールの見直し

ES2016～ES2018の新機能

- ・配列内の要素の存在チェック
- ・ゼロパディング
- ・Promiseの取り扱い
- ・オブジェクトの複製

配列内の要素の存在チェック

```
const targetArray = ["鈴木", "田中", "高橋"];
```

に"田中"が含まれているかどうかを調べたい

配列要素存在チェックの従来コード

indexOf()でインデックスが0以上かどうかを調べる

```
if (targetArray.indexOf("田中") >= 0) {  
    console.log("田中が含まれています");  
} else {  
    console.log("田中は存在しません");  
}
```

配列要素存在チェックで見直したコード

includes()で存在チェックが可能

(2016年6月リリースのES2016で対応)

```
// 田中が存在するかどうか  
if (targetArray.includes("田中")) {  
    console.log("田中が含まれています");  
}  
else {  
    console.log("田中は存在しません");  
}
```

ES2016～ES2018の新機能

- ・配列内の要素の存在チェック
- ・ゼロパディング
- ・Promiseの取り扱い
- ・オブジェクトの複製

JSでゼロパディング

```
const second = new Date().getSeconds();
```

で取得した秒数をゼロパディングした文字列にしたい
(例: 00, 01, 02, ..., 12, 13)

ゼロパディングの従来のコード

数値が10未満の場合だけ"0"を冒頭に追加する

```
let paddedSecond = String(second);

// 10秒未満なら、冒頭に0を付与する

if (second < 10) {
    paddedSecond = "0" + paddedSecond;
}
```

ゼロパディングで見直したコード

`padStart()`を使えばゼロパディングが可能

(2017年10月リリースのES2017で対応)

```
const paddedSecond = String(second).padStart(2, "0");
```

ES2016～ES2018の新機能

- ・配列内の要素の存在チェック
- ・ゼロパディング
- ・Promiseの取り扱い
- ・オブジェクトの複製

Promiseの取り扱い

JSONをフェッチし、その中身を出力したい

Promiseの従来コード

then()でつなげる

```
fetch("./myjson.json")
  .then(response => {
    return response.json()
  })
  .then(
    json => {
      const myText = json.myText;
      console.log(myText);
    },
    error => {
      console.error(`error : ${error}`)
    }
  );
}
```

Promiseをasync/awaitで処理する

async/awaitで処理する
(2017年6月リリースのES2017で対応)

```
async function main() {
  try {
    const json = await (await fetch("./myjson.json")).json();
    console.log(Reflect.get(json, "mytext"));
  } catch (error) {
    console.log(`error : ${error}`);
  }
}

main();
```

ES2016～ES2018の新機能

- ・配列内の要素の存在チェック
- ・ゼロパディング
- ・Promiseの取り扱い
- ・オブジェクトの複製

オブジェクトの複製

```
const myObject = {  
  result: true,  
  members: [  
    { id: 1, name: "鈴木" },  
    { id: 2, name: "田中" },  
    { id: 3, name: "高橋" }  
  ]  
};
```

を複製したい

オブジェクト複製の従来コード

Object.assign()を使う

```
const copiedObject = Object.assign({}, myObject);
```

オブジェクト複製で見直したコード

スプレッド演算子を使う
(2018年6月リリースのES2018で対応)

```
const copiedObject = {...myObject};
```

2019年以降も
JavaScriptは新機能が追加される

2019年までに見直しておきたいCSS・JavaScriptの手法

1. CSSの見直し

2. JavaScriptの見直し

1. ES2016～ES2018の新機能

2. 2019年以降に追加される機能

3. ツールの見直し

2019年以降に追加されるJavaScriptの機能

- nullの可能性がある値へのアクセス
- 関数の合成
- モジュールの動的import

nullの可能性がある値へのアクセス

```
const data = {  
  user: {  
    address: {  
      area: "福岡県"  
    }  
  }  
};
```

"福岡県"を抽出したいが、
userやaddressはnullかもしれない

nullチェックの従来コード

一つずつ値をチェックしていく

```
let areaName;

if (data.user && data.user.address) {
    areaName = data.user.address.area;
}

// または

let areaName2 =
    data.user
    && data.user.address
    && data.user.address.area;
```

nullチェックの新コード

Optional Chaining(?演算子)

```
const areaName = data.user?.address?.area;
```

2019年以降に追加されるJavaScriptの機能

- nullの可能性がある値へのアクセス
- 関数の合成
- モジュールの動的import

関数の合成

```
function myFunction1(price) {  
    return Math.floor(price * 1.1);  
}  
  
function myFunction2(price) {  
    return `${price}円`;  
}
```

10%込の値段を計算し、"円"をつけて出力するプログラムを作りたい

例: 100 → 「110円」

従来のコード

結果を受け取って次の関数に渡すか、
関数の引数に関数を渡す

```
let result = myFunction1(100);
result = myFunction2(result);

// または

const result2 =
myFunction2(myFunction1(100));
```

関数合成の新コード

Pipeline Operator(|>演算子)

```
const result =  
  100  
  |> myFunction1  
  |> myFunction2;
```

2019年以降に追加されるJavaScriptの機能

- nullの可能性がある値へのアクセス
- 関数の合成
- モジュールの動的import

モジュールの動的import

ルート(main.js)

 | ページ1 (module1.js)

 | ページ2 (module2.js)

 | ページ3 (module3.js)

SPAを作る際、JSファイルの肥大化を防ぐため、

ページ1に遷移したときにmodule1.jsを読み込みたい

モジュールの動的import

モジュールの定義

▼ module1.js

```
export class Module1 {
    myMethod() {
        alert("モジュール1のメソッド");
    }
}
```

モジュールの動的import

import()を使ってモジュールを任意のタイミングで読み込む

```
function onPage1Loaded() {  
  import('./module1.js');  
}
```

* onPage1Loaded()はページ1が読み込まれたとき
実行される想定の架空のコード

モジュールの動的import

Promise形式で処理可能

```
async function onPage1Loaded() {  
  const { Module1 } = await import('./module1.js');  
  
  const module = new Module1();  
  module.myMethod();  
}
```

**その他にも多くの機能が
提案されている**

現在議論されているJSの新機能の一例

提案内容	意味	ステージ
BigInt	2**53以上の数を表現	3
flat(), flatMap()	多階層の配列を 一階層にする	3
クラスフィールド、 プライベートフィールド	フィールド変数の定義、 クラス外からアクセス不可能な フィールドの定義	3
Numeric separators	123_456のように数値を 区切って表現	2

ブラウザの実装を待つのではなく、AltJSで対応

- Optional Chaining、Pipeline OperatorはBabel v7で対応
(2018年8月リリース)
- TypeScriptとの組み合わせも可能

BABEL **TypeScript**

**どうやってJavaScriptの
最新機能や対応状況を追うか？**

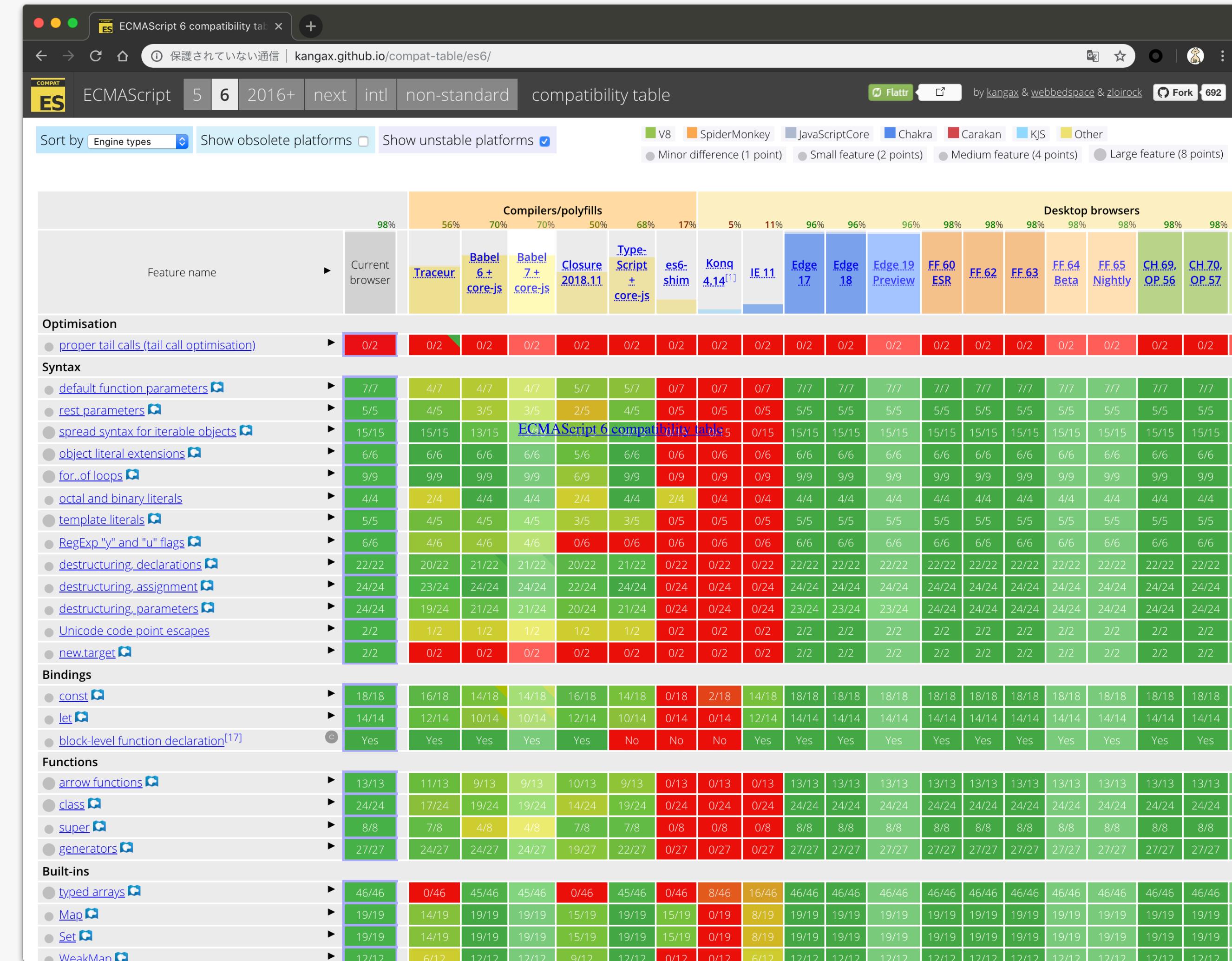
GitHubのtc39/proposals

The screenshot shows a web browser window displaying the [tc39/proposals](https://github.com/tc39/proposals) repository on GitHub. The page title is "ECMAScript proposals". Below the title, there are links to "Stage 0 Proposals", "Finished Proposals", and "Inactive Proposals". A note indicates that the list contains only stage 1 proposals and higher that have not yet been withdrawn/rejected, or become finished.

The main content area is titled "Active proposals" and shows a table of proposals categorized under "Stage 3". The table has columns for "Proposal", "Author", "Champion", and "Tests". Each proposal row includes a small icon representing its status.

Proposal	Author	Champion	Tests
<code>globalThis</code>	Jordan Harband	Jordan Harband	✓
<code>import()</code>	Domenic Denicola	Domenic Denicola	✓
<code>Legacy RegExp features in JavaScript</code>	Claude Pache	Mark Miller Claude Pache	✓
<code>BigInt</code>	Daniel Ehrenberg	Daniel Ehrenberg	✓
<code>import.meta</code>	Domenic Denicola	Domenic Denicola	?
<code>Private instance methods and accessors</code>	Daniel Ehrenberg	Daniel Ehrenberg Kevin Gibbons	?
<code>Array.prototype.{flat,flatMap}</code>	Brian Terlson Michael Ficarra	Brian Terlson Michael Ficarra	✓
<code>Class Public Instance Fields & Private Instance Fields</code>	Daniel Ehrenberg Kevin Gibbons	Daniel Ehrenberg Jeff Morrison Kevin Smith Kevin Gibbons	?
<code>Static class fields and private static methods</code>	Daniel Ehrenberg Kevin Gibbons Jeff Morrison Kevin Smith	Shu-Yu Guo and Daniel Ehrenberg	
<code>String.prototype.{trimStart,trimEnd}</code>	Sebastian	Sebastian Markbåge	✓

ECMAScript 6 compatibility table



各ブラウザーの公式ブログ

- Chromium Blog
- Chrome Releases
- WebKit
- The Mozilla Blog
- Microsoft Edge Blog

2019年までに見直しておきたいCSS・JavaScriptの手法

1. CSSの見直し
2. JavaScriptの見直し
3. ツールの見直し

2019年までに見直しておきたいCSS・JavaScriptの手法

1. CSSの見直し
2. JavaScriptの見直し
3. ツールの見直し

1. Autoprefixerの進化

2. コードフォーマットはPrettierで
3. パッケージマネージャーYarn
4. ビルド環境の変化

**IE 11対応案件でCSS Gridを使う場合は
古いコードへの変換が必要**

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 9%
- 世界 3%



Autoprefixerで自動変換

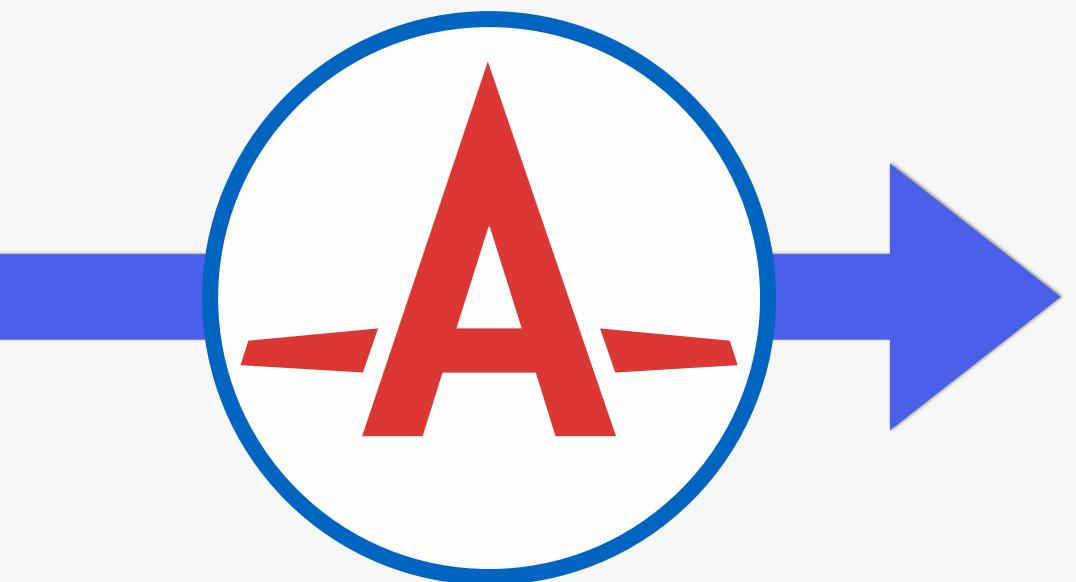
```
.container {
  display: grid;
  grid-template:
    "visual number expression" 1fr
    "visual other other" 220px /
    40% 120px 1fr;
}

.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-2018年のAutoprefixerの進化

- エリア名の変換
- 行・列間の隙間(gap)の変換
- 自動配置の変換(一部)

CSS Gridでエリアを指定する従来コード

IE 11はエリア名の指定に未対応なので、番号指定が必要



行と列の定義

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

アイテムの配置

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

AutoprefixerでエリアをIE 11向けに変換

before

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

AutoprefixerでエリアをIE 11向けに変換

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;  
}
```

2017年-2018年のAutoprefixerの進化

- エリア名の変換
- 行・列間の隙間(gap)の変換
- 自動配置の変換(一部)

CSS Gridで行と列の隙間を指定する従来コード

IE 11はgap(旧名grid-gap)に未対応なので、marginやpaddingで指定していた



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

AutoprefixerでgapをIE 11向けに変換

before

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

AutoprefixerでgapをIE 11向けに変換

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;
}

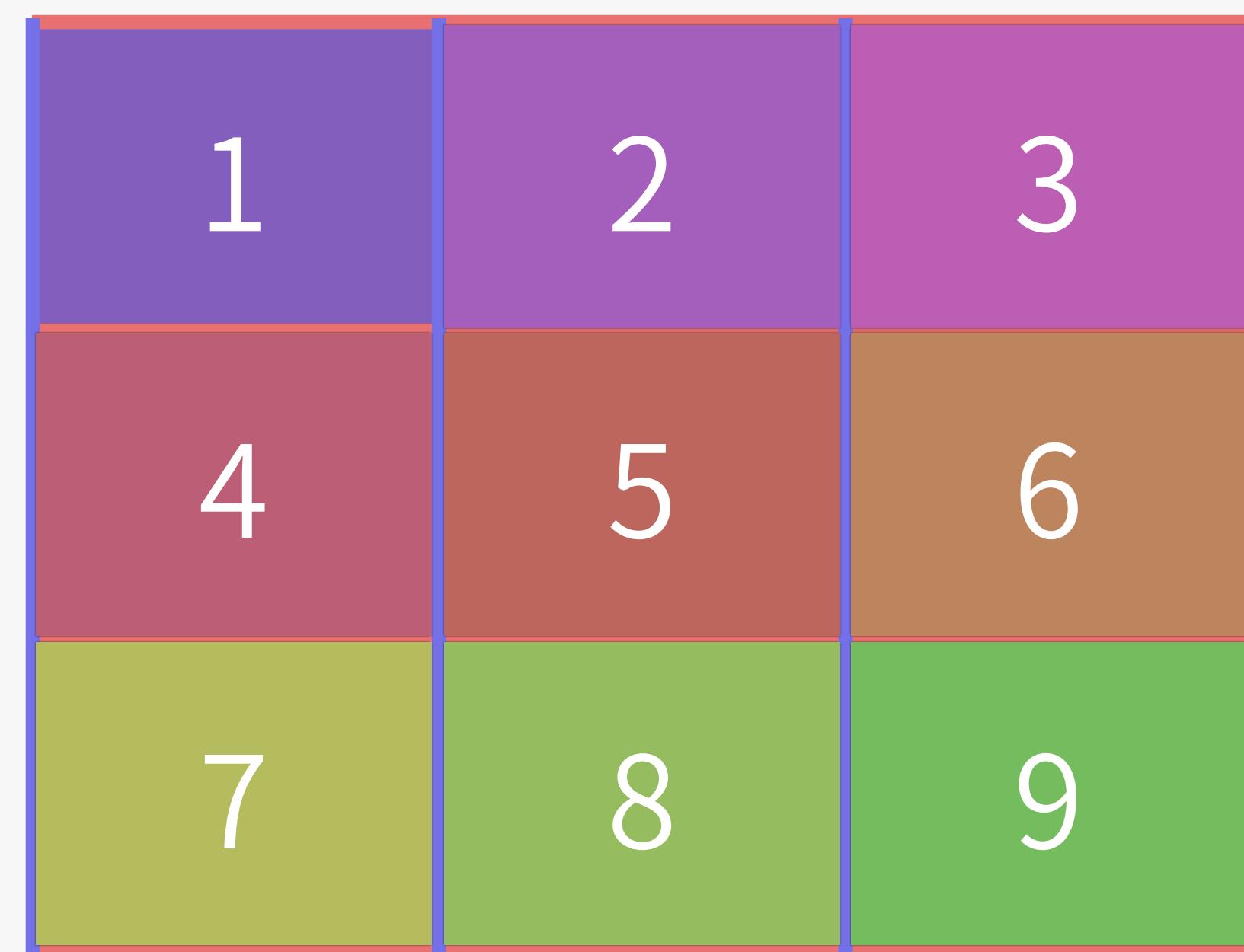
.visual {
    -ms-grid-row: 1;
    -ms-grid-row-span: 3;
    -ms-grid-column: 1;
    grid-area: visual;
}
```

2017年-2018年のAutoprefixerの進化

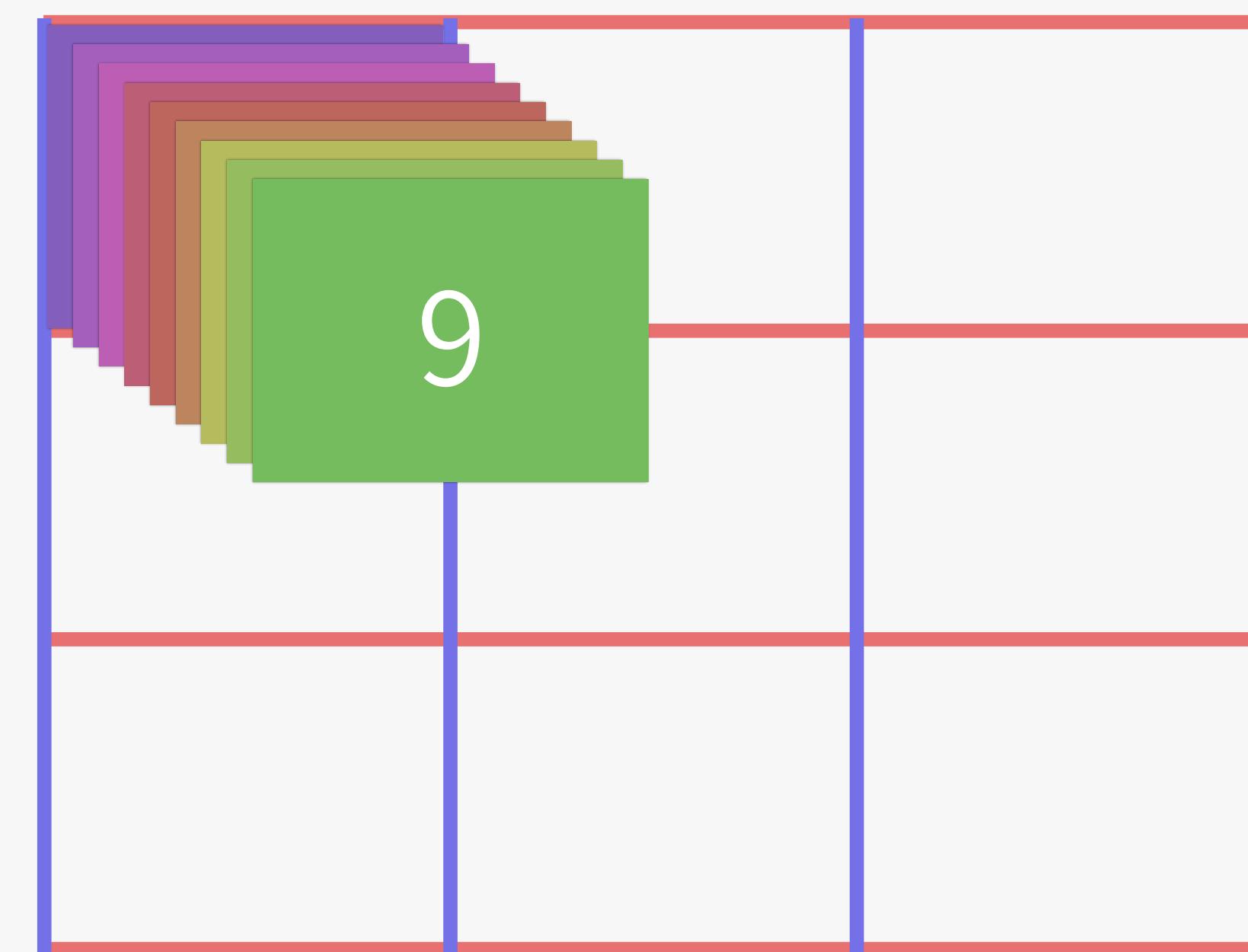
- エリア名の変換
- 行・列間の隙間(gap)の変換
- 自動配置の変換(一部)

IE 11におけるCSS Gridの自動配置問題

IE 11は自動配置に未対応なので、アイテムが最初のセルに重なる



モダンブラウザー



IE 11

- * アイテムは全て同じ位置に重なる
視覚的にわかりやすくするためにずらして表示した

Autoprefixerで自動配置をIE 11向けに変換可能に(一部)

before

```
/* autoprefixer grid: autoplacement */  
  
.container {  
    display: grid;  
    grid-template-columns: 1fr 1fr;  
    grid-template-rows: auto auto;  
    grid-gap: 20px;  
}
```

Autoprefixerで自動配置をIE 11向けに変換可能に(一部)

after

```
/* autoprefixer grid: autoplacement */

.container {
    display: -ms-grid;
    display: grid;
    -ms-grid-columns: 1fr 20px 1fr;
    grid-template-columns: 1fr 1fr;
    -ms-grid-rows: auto 20px auto;
    grid-template-rows: auto auto;
    grid-gap: 20px;
}

.container > *:nth-child(1) {
    -ms-grid-row: 1;
    -ms-grid-column: 1;
}

.container > *:nth-child(2) {
    -ms-grid-row: 1;
    -ms-grid-column: 3;
}

/* 省略 */
```

* 自動変換とrepeat()の組み合わせには未対応
なお、repeat()自体の変換には対応済

2019年までに見直しておきたいCSS・JavaScriptの手法

1. CSSの見直し
2. JavaScriptの見直し
3. ツールの見直し

1. Autoprefixerの進化

2. コードフォーマットはPrettierで

3. パッケージマネージャーYarn

4. ビルド環境の変化

コードフォーマットは
開発者任せではなく
Prettier任せにする

コードフォーマッターPrettier

各言語のフォーマットに対応

The screenshot shows the official Prettier website with a dark theme. At the top, there's a navigation bar with links to 'Playground', 'About', 'Usage', 'Blog', a search bar, and a 'GitHub' button. Below the navigation, a large heading says 'Works with the Tools You Use'. There are several sections featuring icons and lists of supported languages and tools:

- JavaScript:** ES2017, JSX, Flow, TypeScript, JSON.
- HTML:** HTML5, Vue, Angular.
- CSS:** CSS3+, Less, SCSS, styled-components, styled-jsx.
- GraphQL:** GraphQL, GraphQL Schemas.
- CommonMark:** CommonMark, GitHub Flavored Markdown, MDX.
- YAML:** YAML.
- Other:** Work in Progress (Elm via elm-format, Java, PHP, PostgreSQL, Python, Ruby, Swift).

コードフォーマッターPrettier

各エディターと連携

The screenshot shows the "Editor Support" section of the Prettier website. It features a grid of icons for various code editors and IDEs, each with its name and the corresponding Prettier package name.

Editor	Prettier Package
Atom	prettier-atom mprettier miniprettier
Emacs	prettier-emacs
Espresso	espresso-prettier
Sublime Text	JsPrettier
Vim	neofomat ale vim-prettier
Visual Studio	JavaScriptPrettier
VS Code	prettier-vscode unibeautify-vscode
WebStorm	Built-in support

[Prettier · Opinionated Code Formatter](#)

PrettierによるJavaScriptのコードフォーマット

- LintとPrettierの二段構え
- 細かいルールの指定ができるLintと
 - 各社のルールの適用がいい感じ
 - Google JavaScript Style Guide
 - JavaScript Standard Style
 - Airbnb

PrettierによるHTMLのフォーマット

- Prettier v1.5のHTMLのフォーマットに対応
- Angular, Vueなどフレームワークのテンプレートにも対応
- かなり強力に効くが、
ホワイトスペース問題があるので少し設定をいじると良い

.prettierrc

```
{  
  "htmlWhitespaceSensitivity": "strict"  
}
```

2019年までに見直しておきたいCSS・JavaScriptの手法

1. CSSの見直し
2. JavaScriptの見直し
3. ツールの見直し

1. Autoprefixerの進化
2. コードフォーマットはPrettierで
3. パッケージマネージャーYarn
4. ビルド環境の変化

Yarnが便利

Facebook製パッケージマネージャーYarn

- npmに似ているので学習コストが低い
- インストールが高速
- npmと一緒に使える
- ワークスペース機能が便利



Yarnは十分はやい

webpackのインストールで検証

パッケージマネージャー	かかった時間
npm	8.2秒
Yarn	3.5秒

npm v6.4.1、Yarn v1.10.1で検証

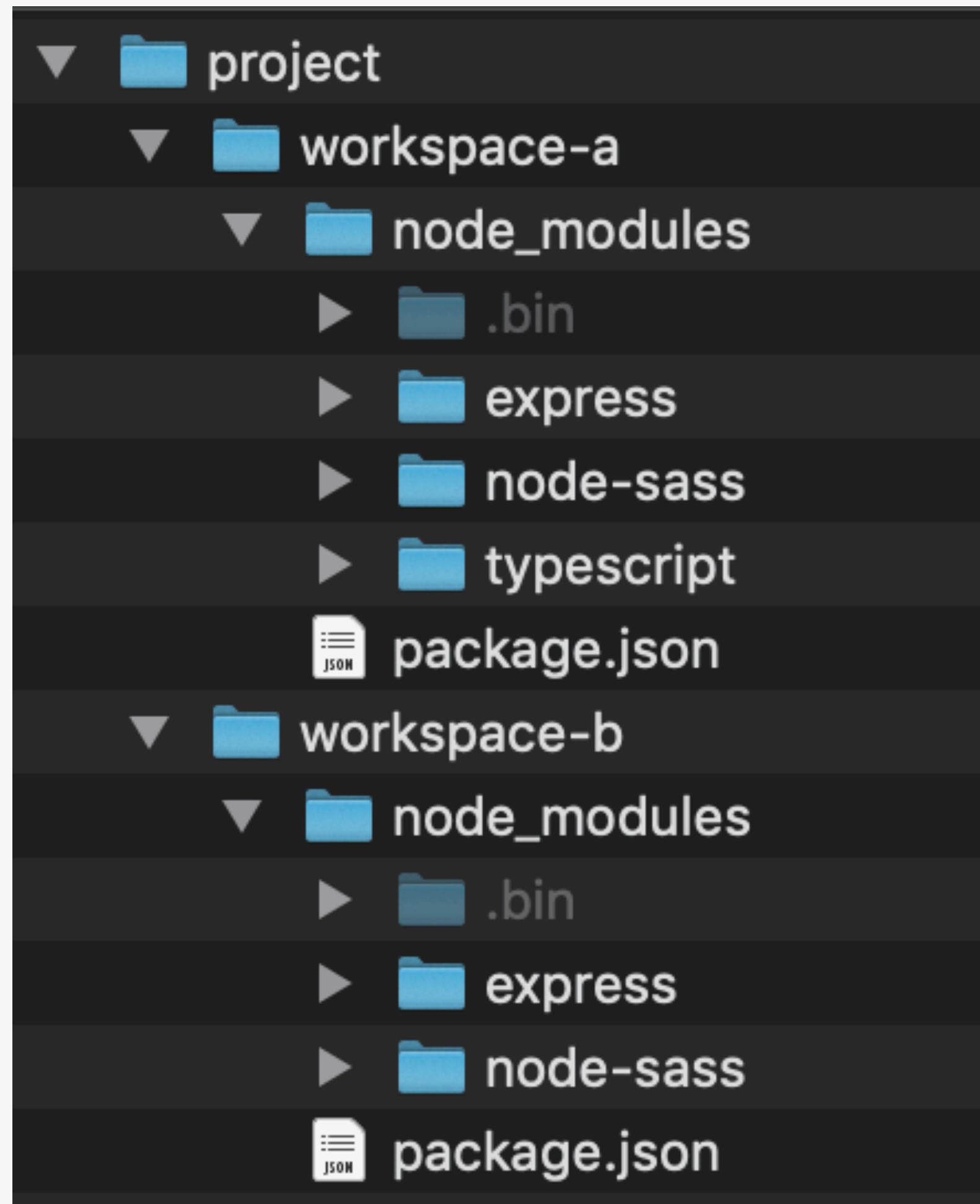
ワークスペース機能が便利

複数のプロジェクトでnode_modulesフォルダーを最適配置できる

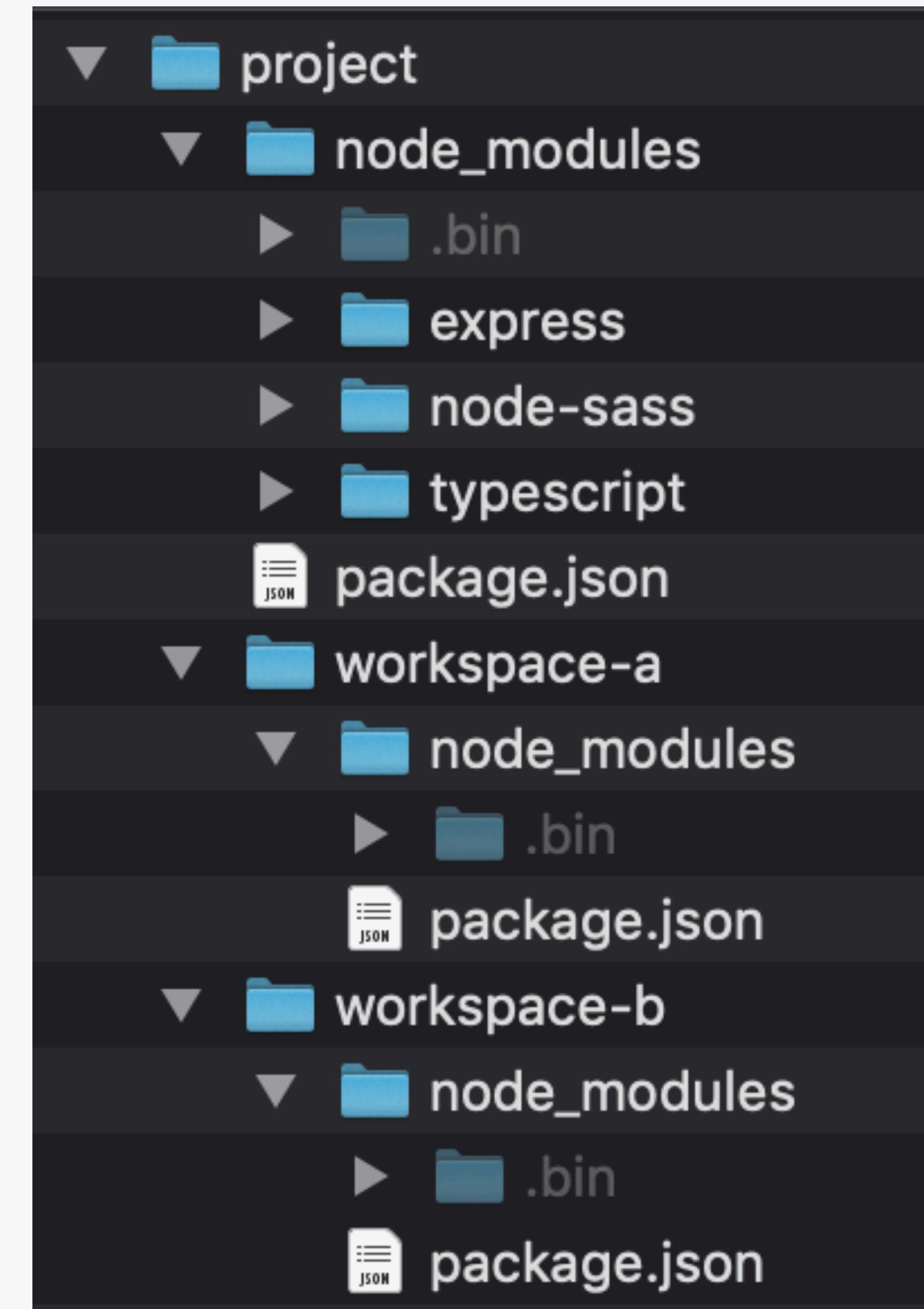
- node_modulesフォルダーの混沌と膨大なファイル管理が少し軽減され、精神衛生上良い

ワークスペースのイメージ

Yarnを使わない場合



Yarnを使う場合



ワークスペースの使い方

project/package.json

```
{  
  "private": true,  
  "workspaces": ["workspace-a", "workspace-b"]  
}
```

ワークスペースの使い方

project/workspace-a/package.json

```
{  
  "name": "workspace-a",  
  "dependencies": {  
    "express": "^4.16.4",  
    "node-sass": "^4.10.0",  
    "typescript": "^3.2.2"  
  }  
}
```

project/workspace-b/package.json

```
{  
  "name": "workspace-b",  
  "dependencies": {  
    "node-sass": "^4.10.0",  
    "typescript": "^3.2.2"  
  }  
}
```

ワークスペースの使い方

コマンドを実行

```
yarn install
```

2019年までに見直しておきたいCSS・JavaScriptの手法

1. CSSの見直し

2. JavaScriptの見直し

3. ツールの見直し

1. Autoprefixerの進化

2. コードフォーマットはPrettierで

3. パッケージマネージャーYarn

4. ビルド環境の変化

ビルド環境は個人でゼロから作るよりも
ツールに任せる

フレームワークを使うなら、環境構築はCLIに任せる

各フレームワークにはビルド環境が準備されている

- React: Create React App
- vue: Vue CLI
- Angular: Angular CLI

フレームワークを使うなら、環境構築はCLIに任せる

webpackの知識は必要

- ・ いずれのCLIも内部的にはwebpackを使用している
- ・ 案件に応じたカスタマイズをするケースは多い

フレームワークを使わないなら、Parcelを使うのも手

あらゆるリソースをバンドル可能

- Sass, Babel, TypeScriptなど、Alt言語のコンパイル
- モジュールのTreeShakingも可能になった



Parcel

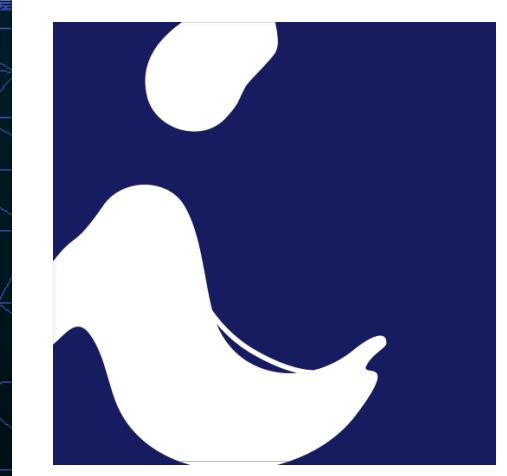
まとめ

まとめ

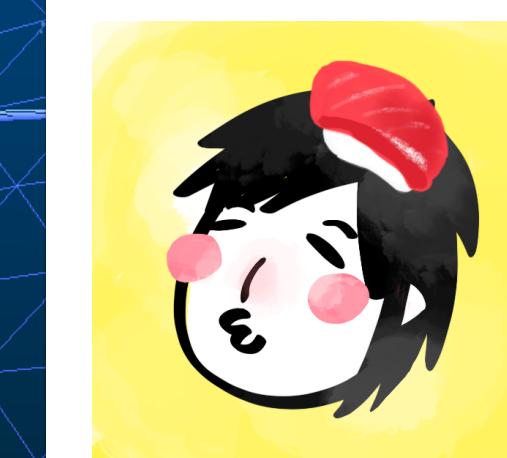
- CSS、JavaScript、ツール共に新しい機能が増えている
- 新しい機能はこれまでの開発のスピードを上昇させ、ラクにしてくれる
- 積極的に取り入れ、作業時間を減らし、作りこみに時間をかけよう

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