



CSS Nite in **Osaka**, Vol.52

# Modern Coding

きちんと理解して活用する SVG と CSS Grid

powered by **MEBIC**

2019.9.7 



# 現場で働くコーダーのための CSS Grid + モダンコーディング

株式会社ICS 鹿野 壮

2019年9月7日 CSS Nite in Osaka, vol.52 「Modern Coding」



# 自己紹介



株式会社ICS

福岡県出身

鹿野 壮 (かの たけし)



@tonkotsuboy\_com

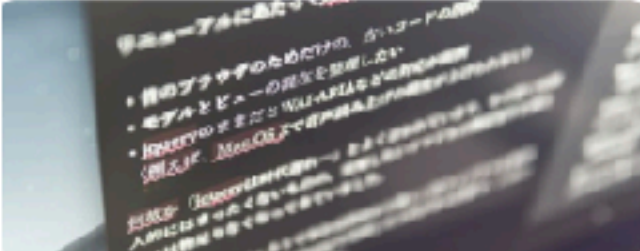


# オウンドメディア「ICS MEDIA」

ウェブデザイナーとフロントエンジニアのための情報メディア

## ICS MEDIA

### 話題になった記事



エンジニアのための文章校正テクニック  
池田 泰延 平成30年10月5日 ♥ 1399



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 泰延 平成30年5月17日 ♥ 4688



2018年に見直した現代的なJSの記法  
鹿野 壮 平成30年2月22日 ♥ 1352

### 記事一覧


🔍 記事を検索...

トレンド: CSS Grid Flexbox Adobe XD React webpack gulp

新着 人気



TypeScriptのビルド環境が3分でできるParcel入門  
鹿野 壮 令和元年3月28日 ♥ 293



WAI-ARIA対応のタブ型UIの作り方 (Vue.js編)  
池田 泰延 令和元年3月27日 ♥ 2



Adobe XDの「音声トリガー」でかんたん音声プロトタイピング  
加賀 篤史 令和元年2月29日 ♥ 59

ウェブデザイナーと  
フロントエンドエンジニアのための  
情報メディア



ics.media



# 「JavaScriptコードレシピ集」の著者



- ECMAScript、DOM APIなど  
JavaScriptの構文や使い方を解説
- スグに現場で使える  
サンプルを数多く準備
- 時代が変わっても  
長く使えることを目標としている



**CSS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**



# 現場で働くコーダーのためのCSS Grid + モダンコーディング

1. CSS Gridとボックスレイアウト

2. 抑えておきたいCSSの新機能

3. 情報のキャッチアップ方法



# 現場で働くコーダーのためのCSS Grid + モダンコーディング

## 1. CSS Grid

- 従来のボックスレイアウト手法

- CSS Gridの基本

- CSS Gridの便利な機能

- CSS GridのIE11対応方法

## 2. 抑えておきたいCSSの新機能

## 3. 情報のキャッチアップ方法



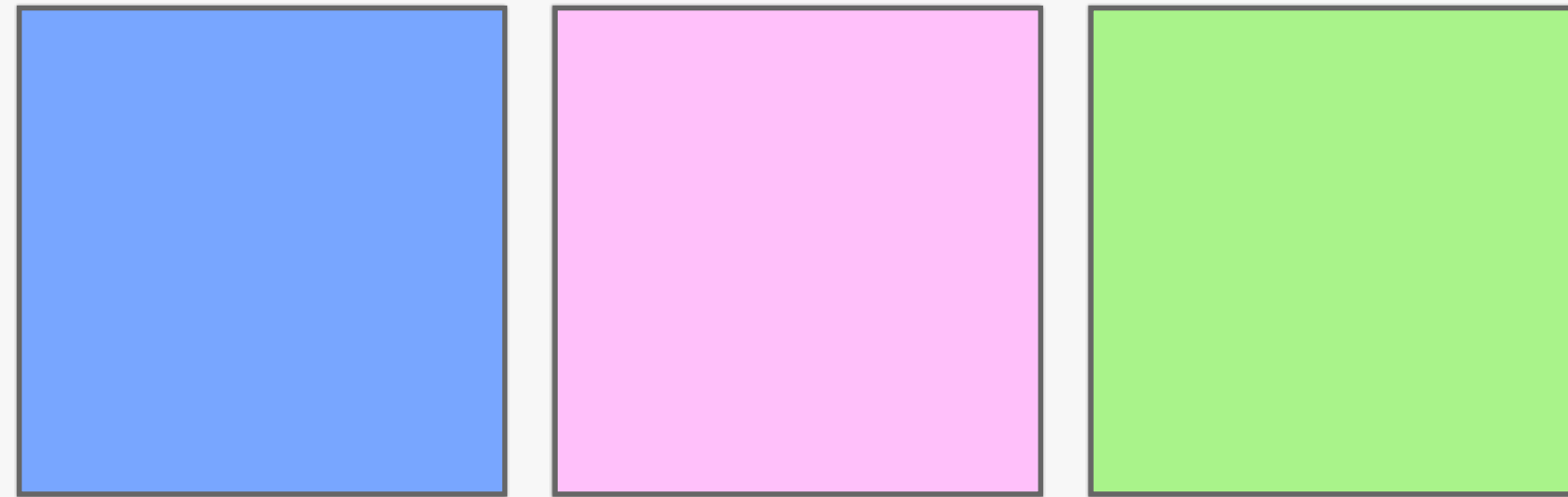
# float

もともとはテキストの回り込みのためのプロパティだった





# Flexbox: 1軸方向のレイアウト



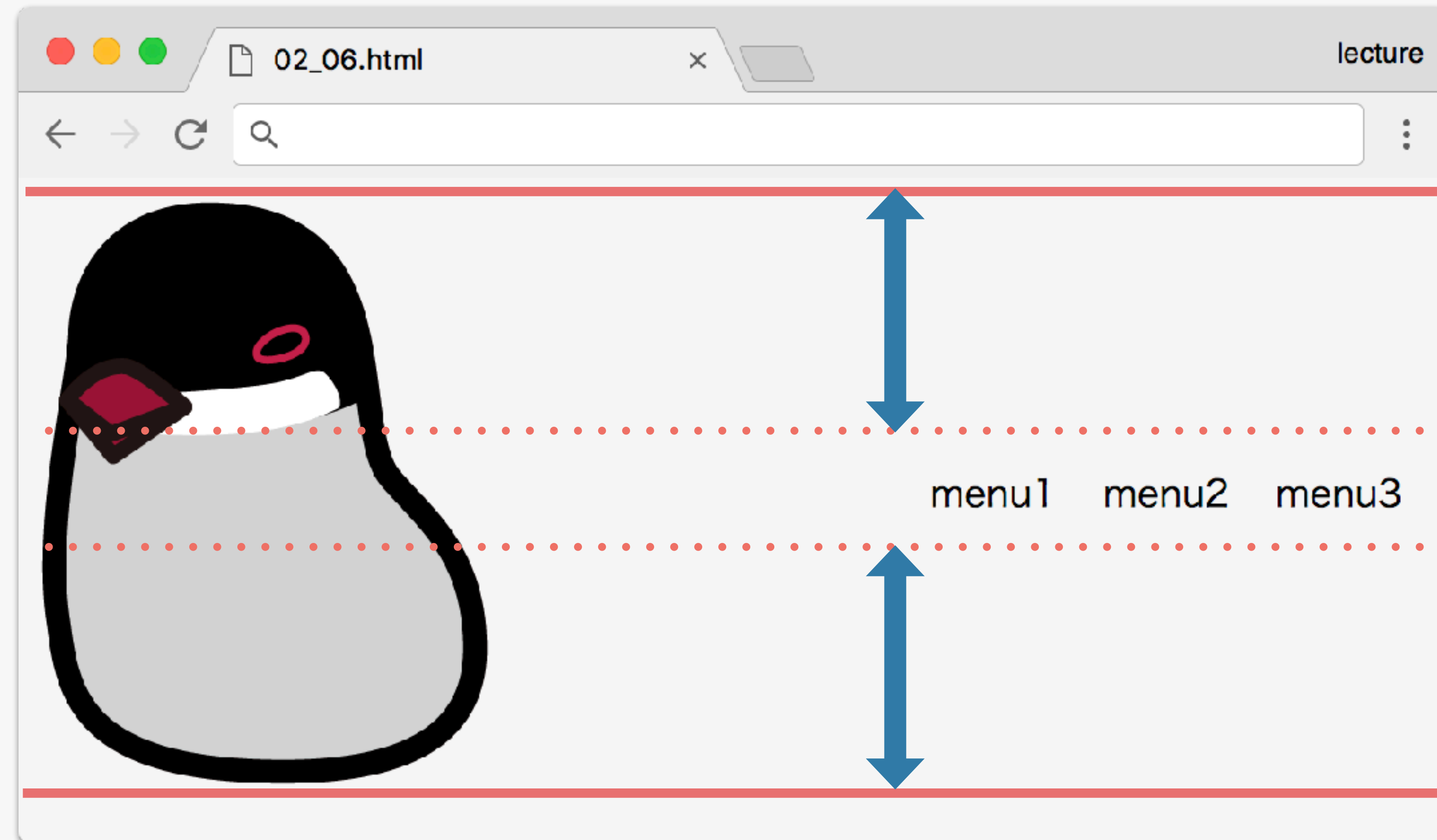
アイテム アイテム アイテム

コンテナ



# Flexbox: レイアウト用プロパティが豊富にある

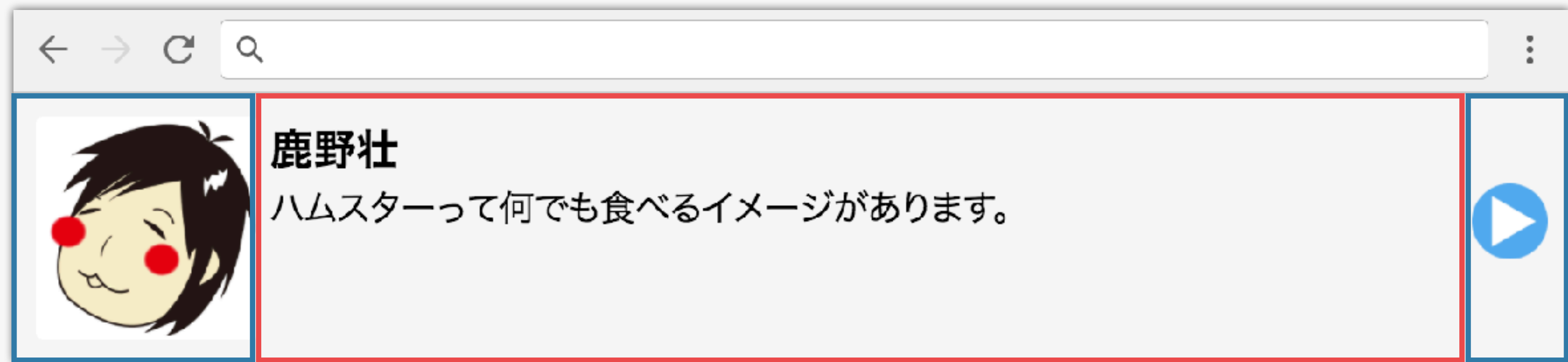
- justify-contentプロパティ: 主軸方向のレイアウト
- align-itemsプロパティ: 交差軸方向のレイアウト





# Flexbox: 固定幅・可変幅の組み合わせがラク

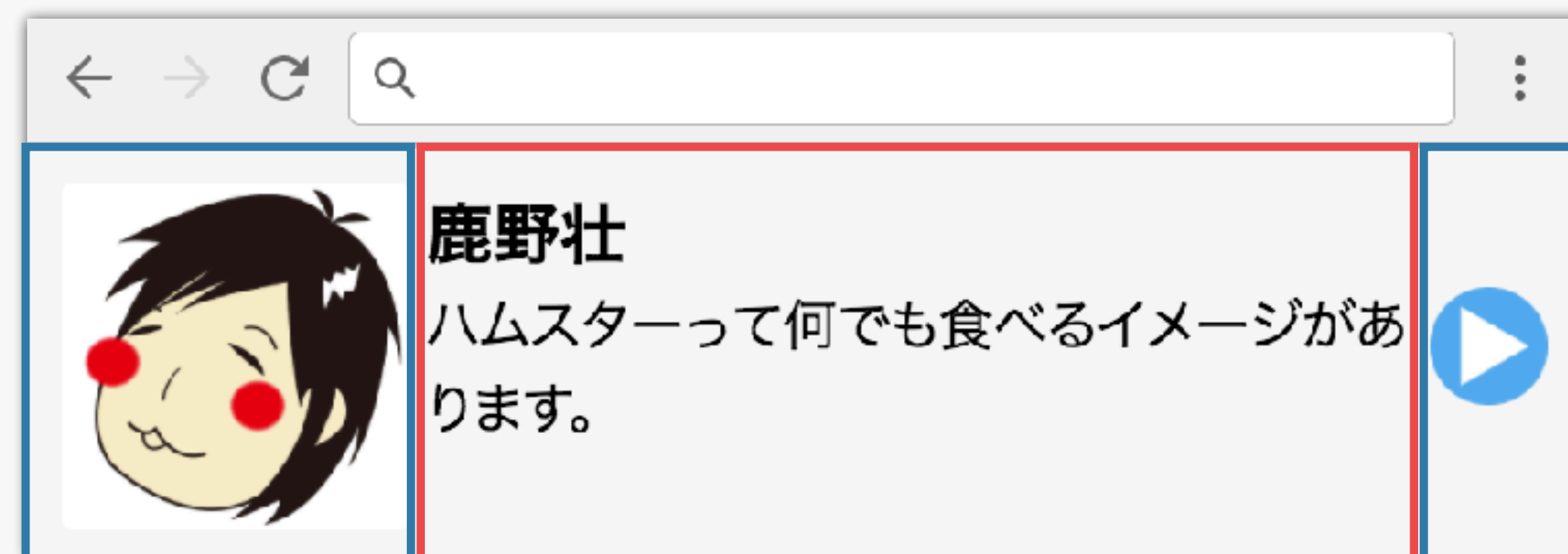
- flex-growプロパティ
- flex-shrinkプロパティ



固定幅

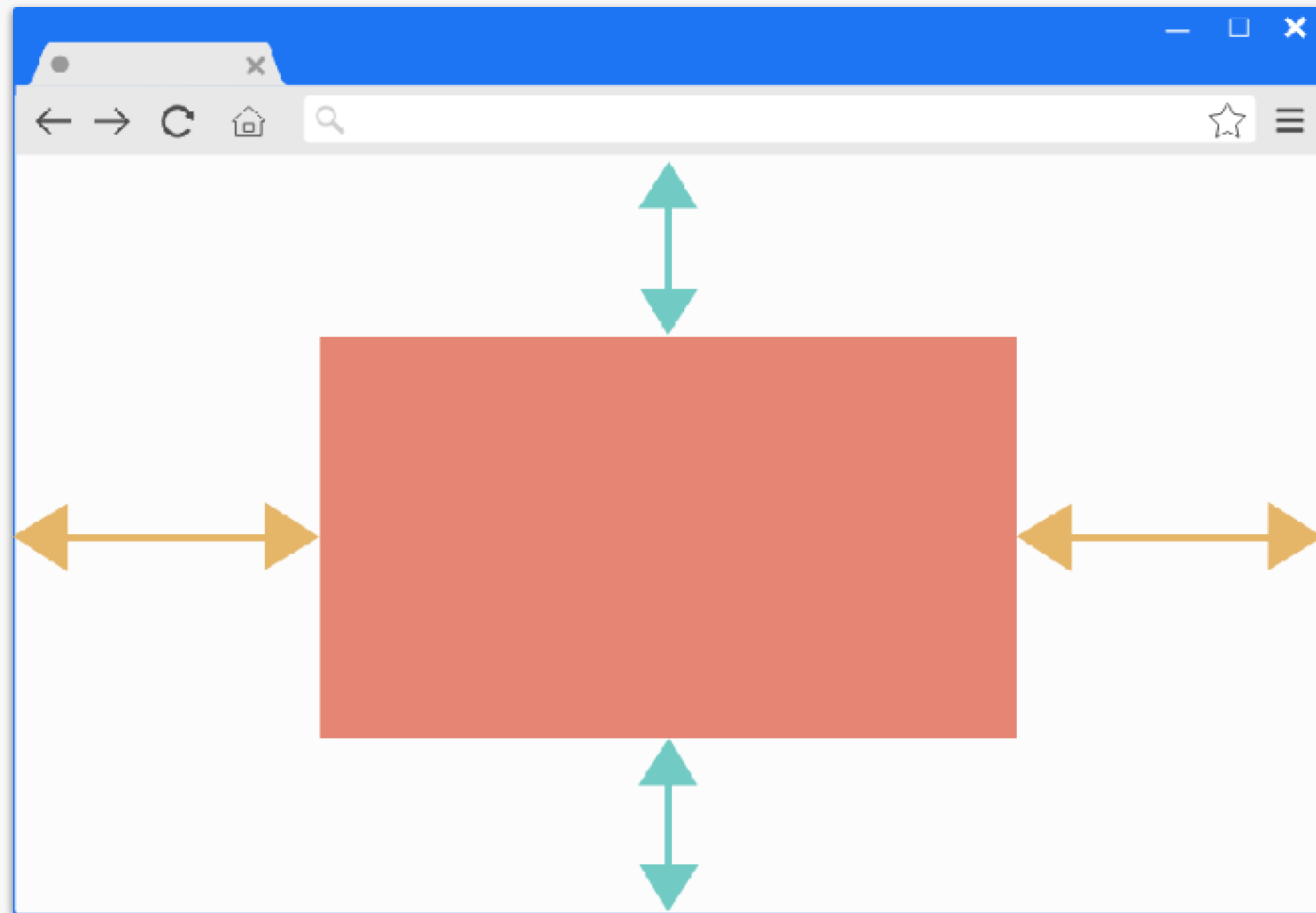
可変幅

固定幅





# Flexbox: 上下中央揃えがたった3行でできる



```
.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

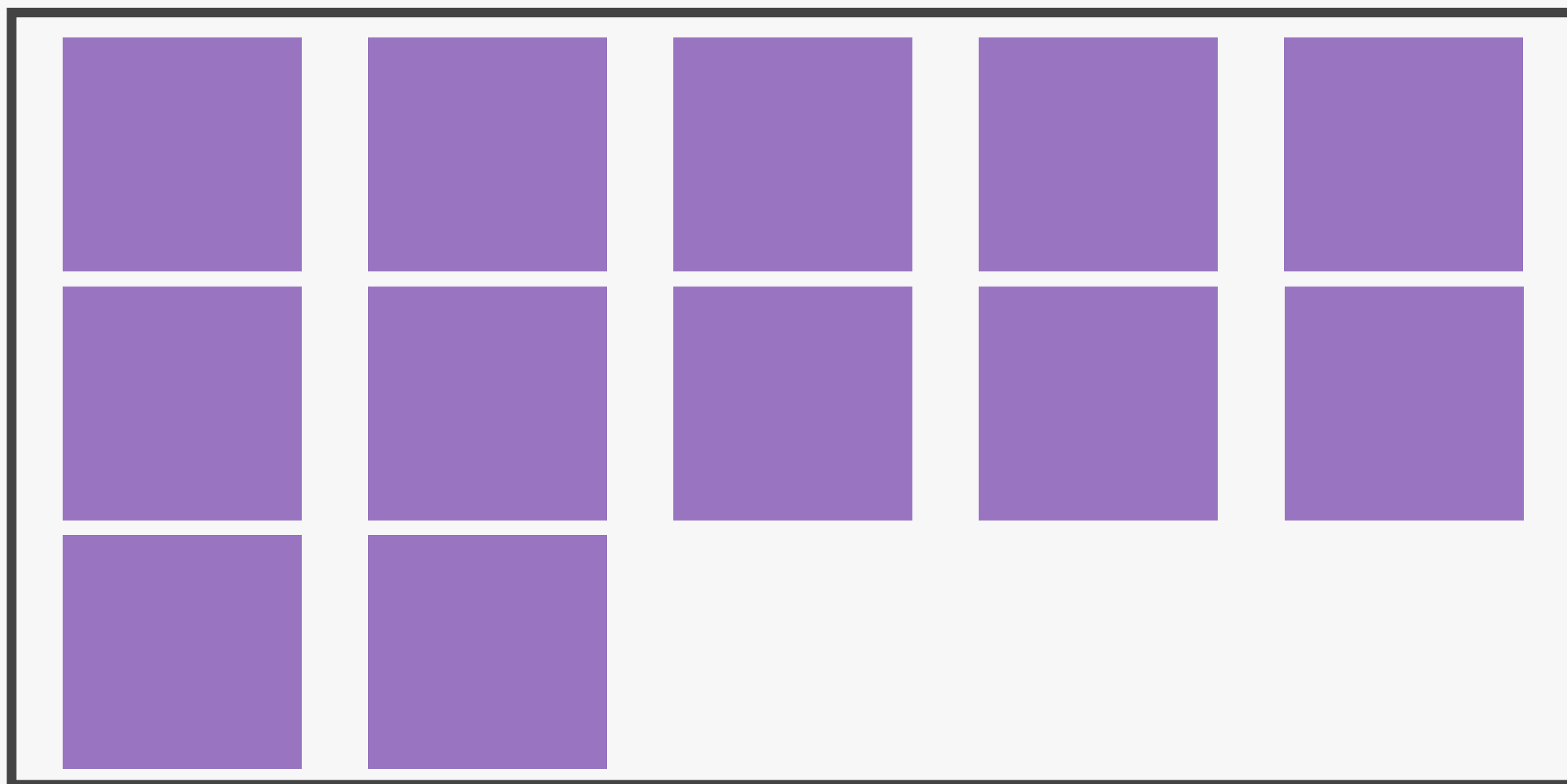


**Flexboxは  
複数行のレイアウトが苦手**

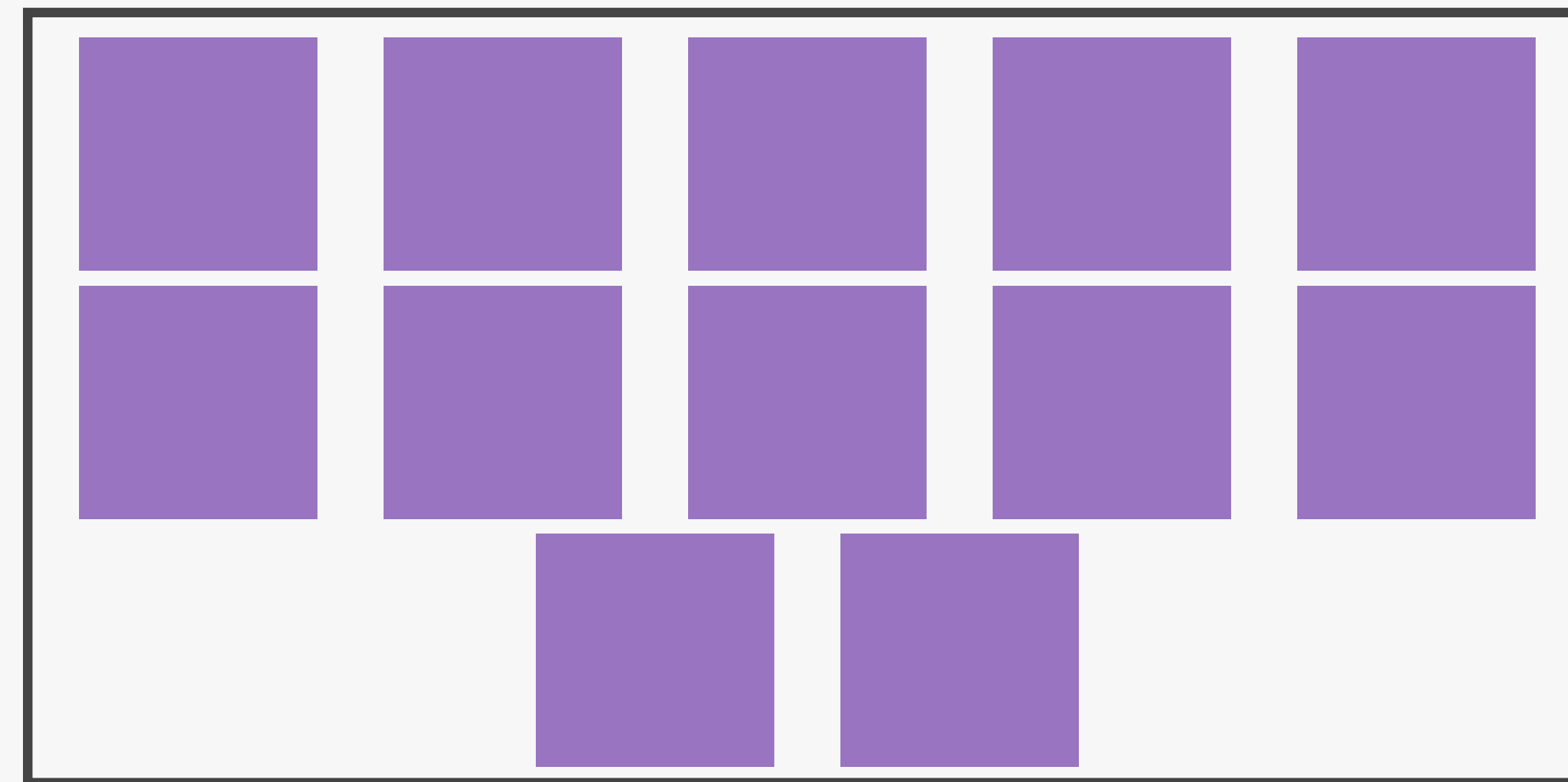


# Flexboxは複数行レイアウトがニガテ

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



理想

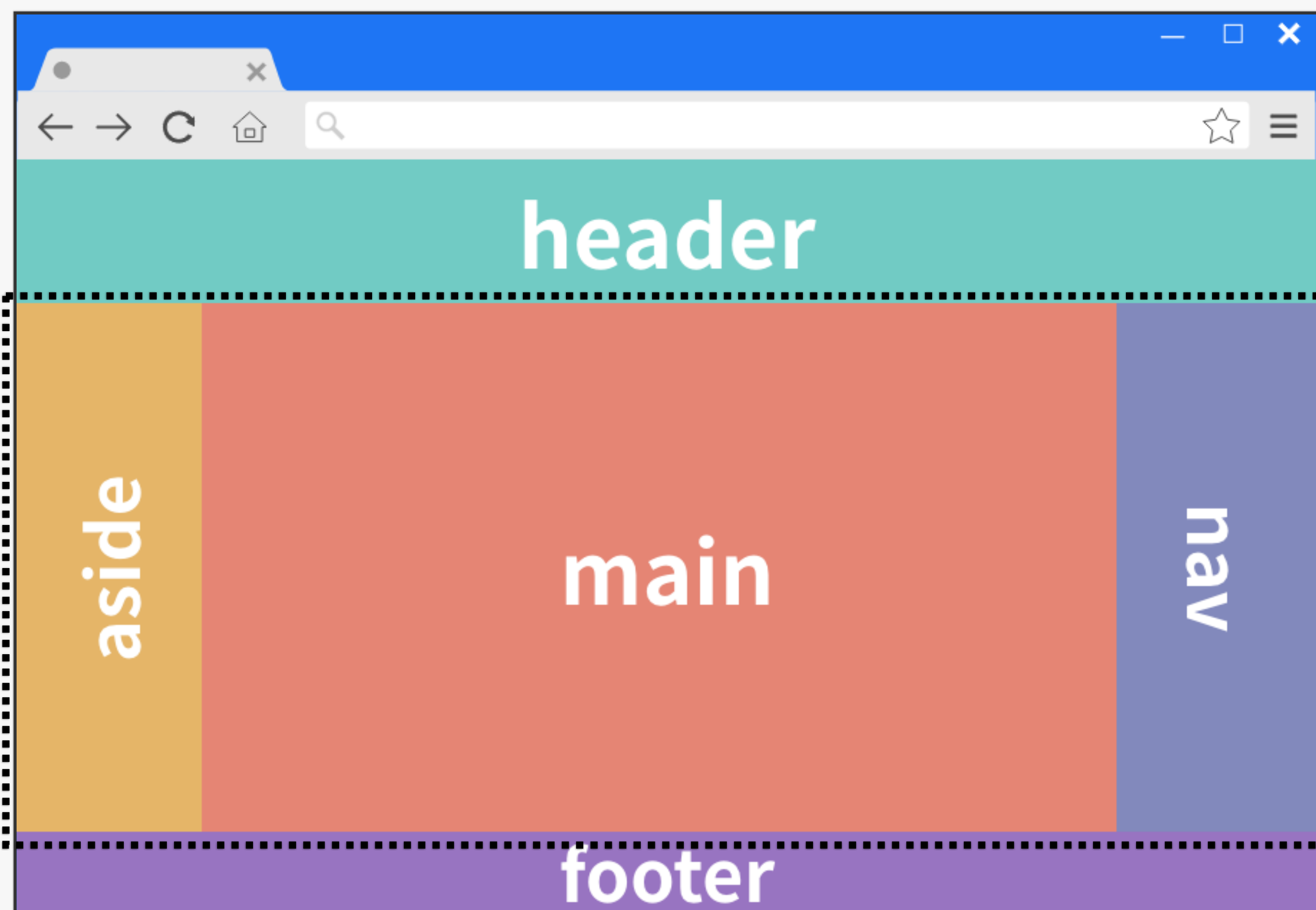


現実

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



# 現場で働くコーダーのためのCSS Grid + モダンコーディング

## 1. CSS Grid

- 従来のボックスレイアウト手法

- CSS Gridの基本

- CSS Gridの便利な機能

- CSS GridのIE11対応方法

## 2. 抑えておきたいCSSの新機能

## 3. 情報のキャッチアップ方法



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



# CSS Gridとは

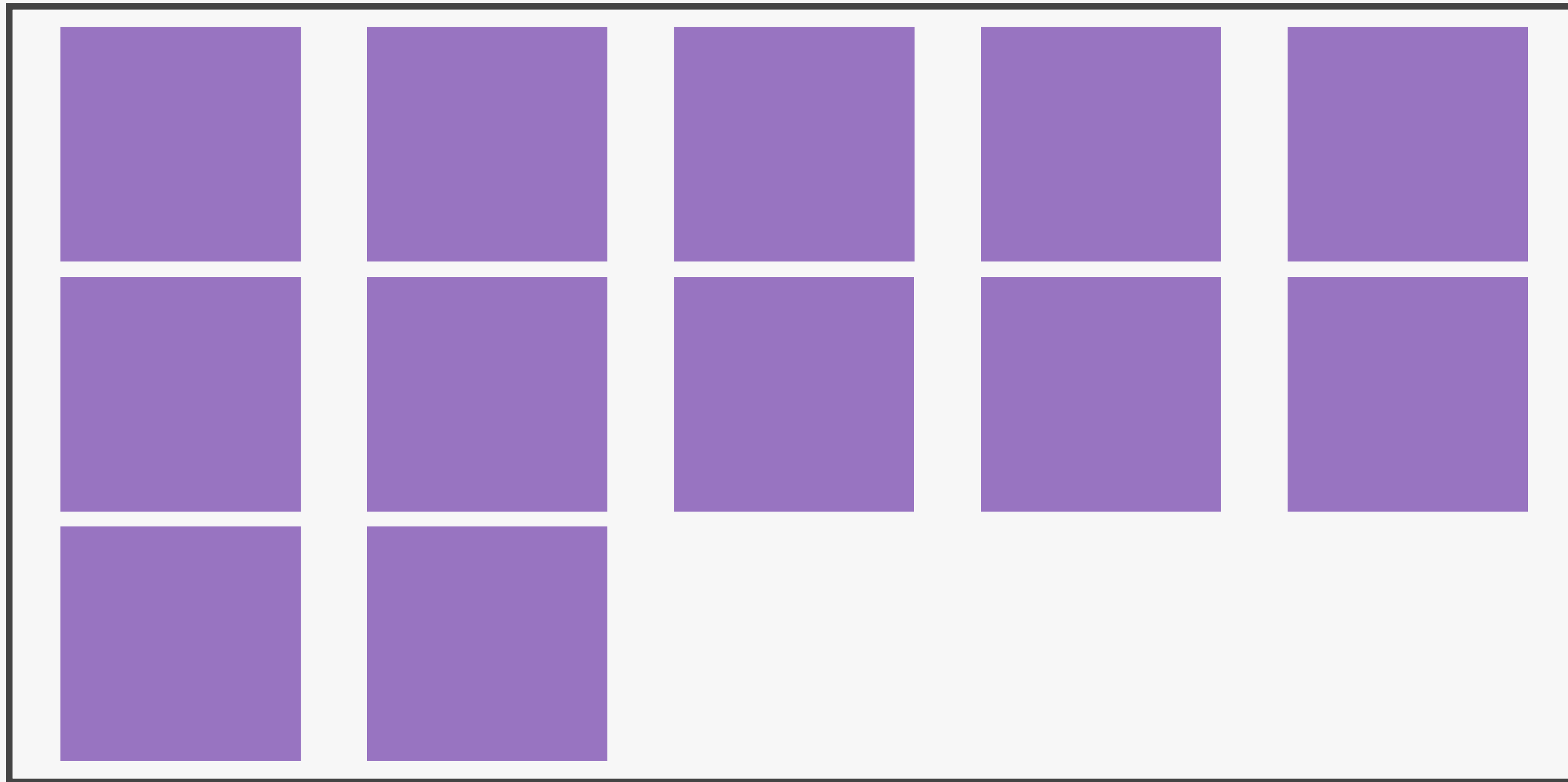
## 「行列」を使ったレイアウトのこと





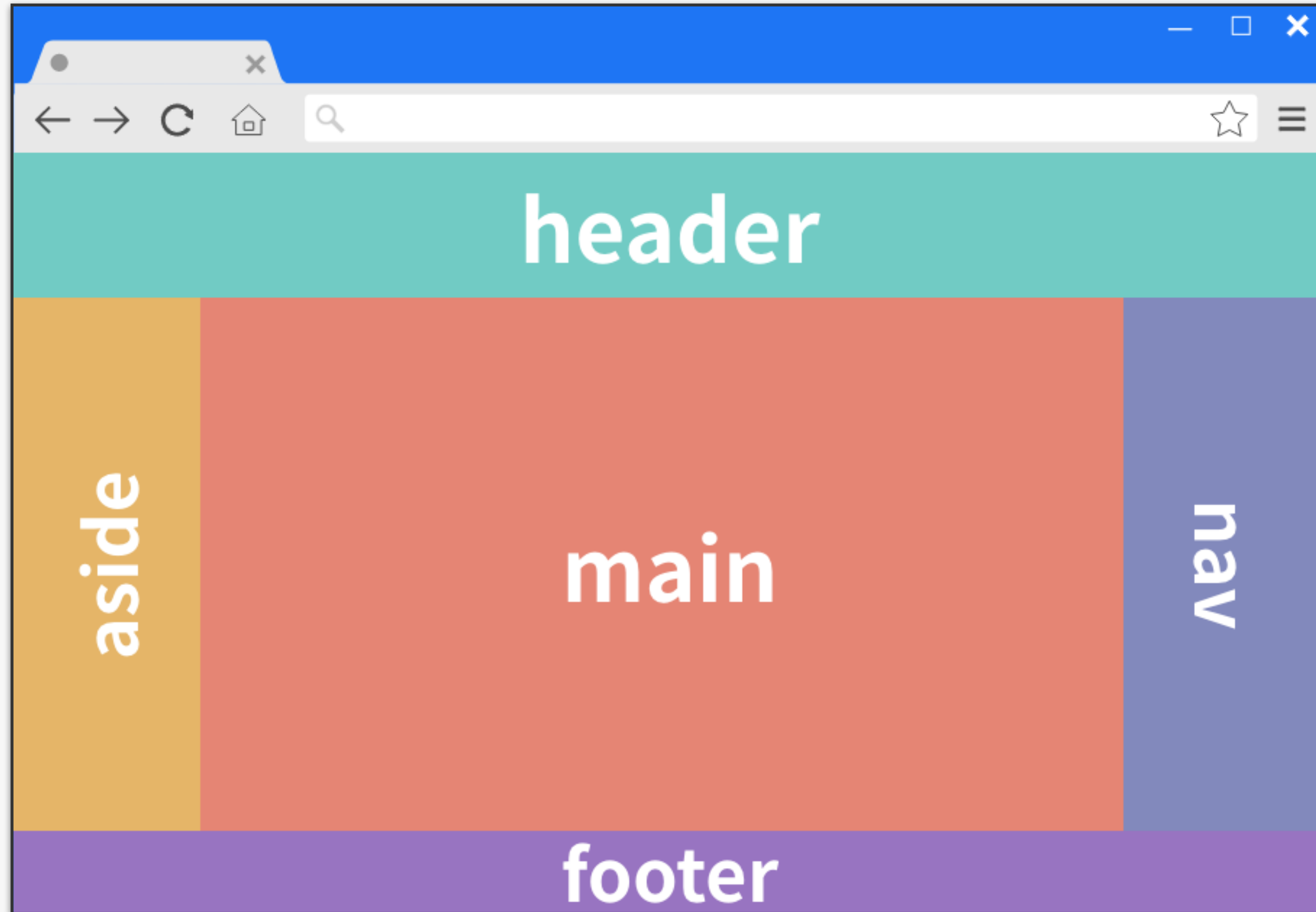

列

# CSS Gridでは複数行レイアウトの最終行が揃う

















# CSS Gridでは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は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	Safari (iOS版)	Chrome (Android版)
最新バージョン	76	68	12	18	12	76
Grid対応						

※ IE 11については、後のスライドで解説します



# 用途別レイアウトの使い分け

テキストの回り込み

**float**



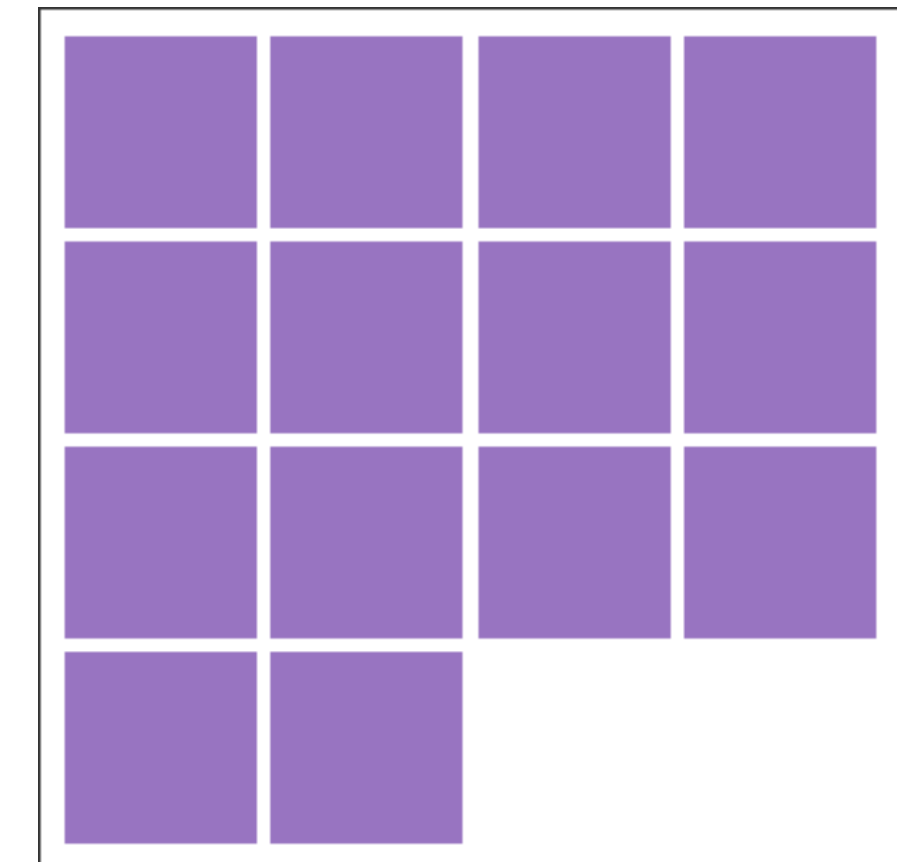
1行の横並び、縦並び

**Flexbox**



格子状のレイアウト  
ページ全体のレイアウト

**CSS Grid**



# CSS Gridを書くための3ステップ

1. コンテナを作る



2. 行と列を作る



3. アイテムを配置する





# 1. コンテナを作る



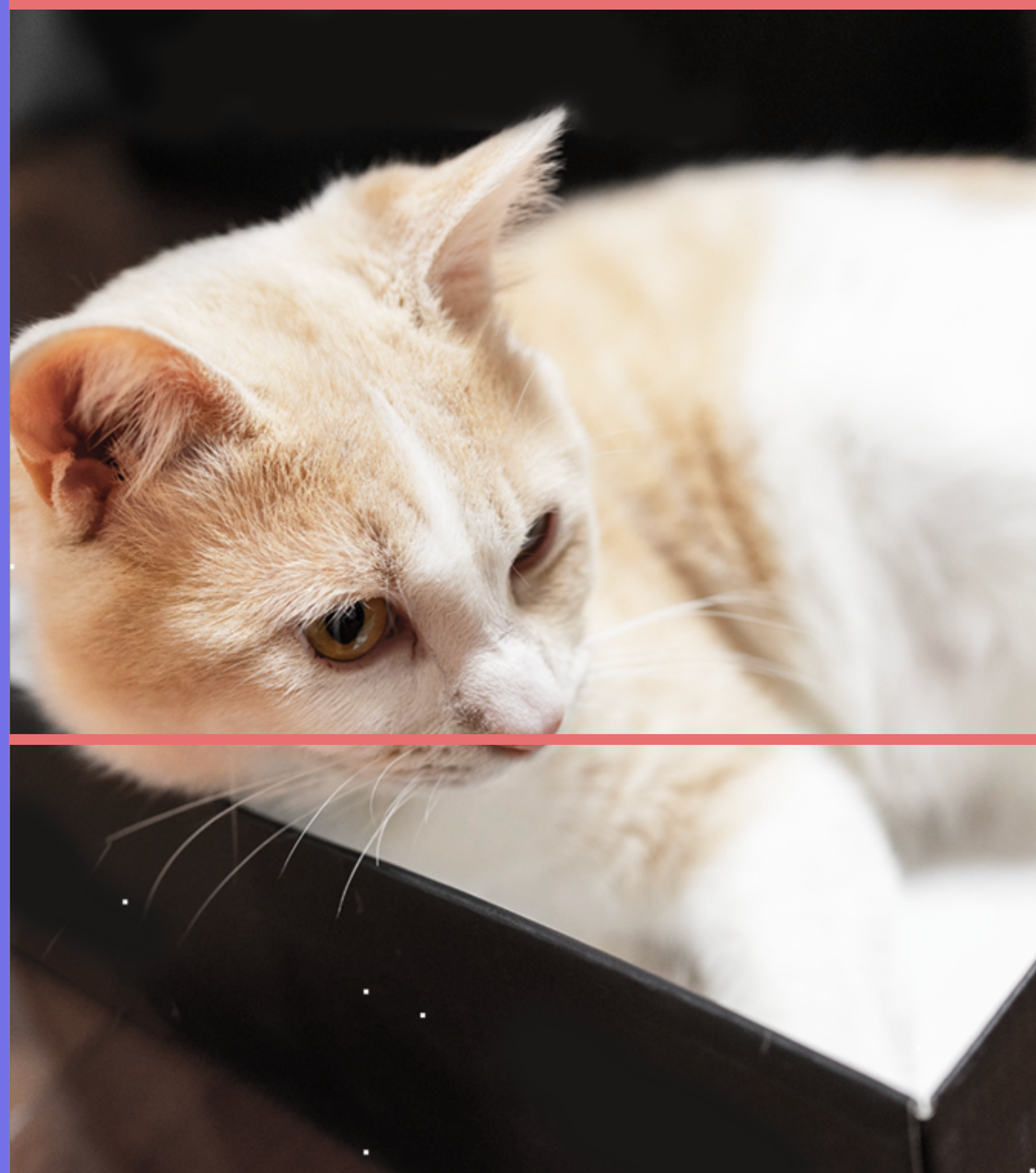
**UNI OF THE DAY**

今日のうにちゃん

コンテナ



## 2. 行と列を作る (テンプレートの作成)



列(column)



行(row)

UNI OF THE DAY

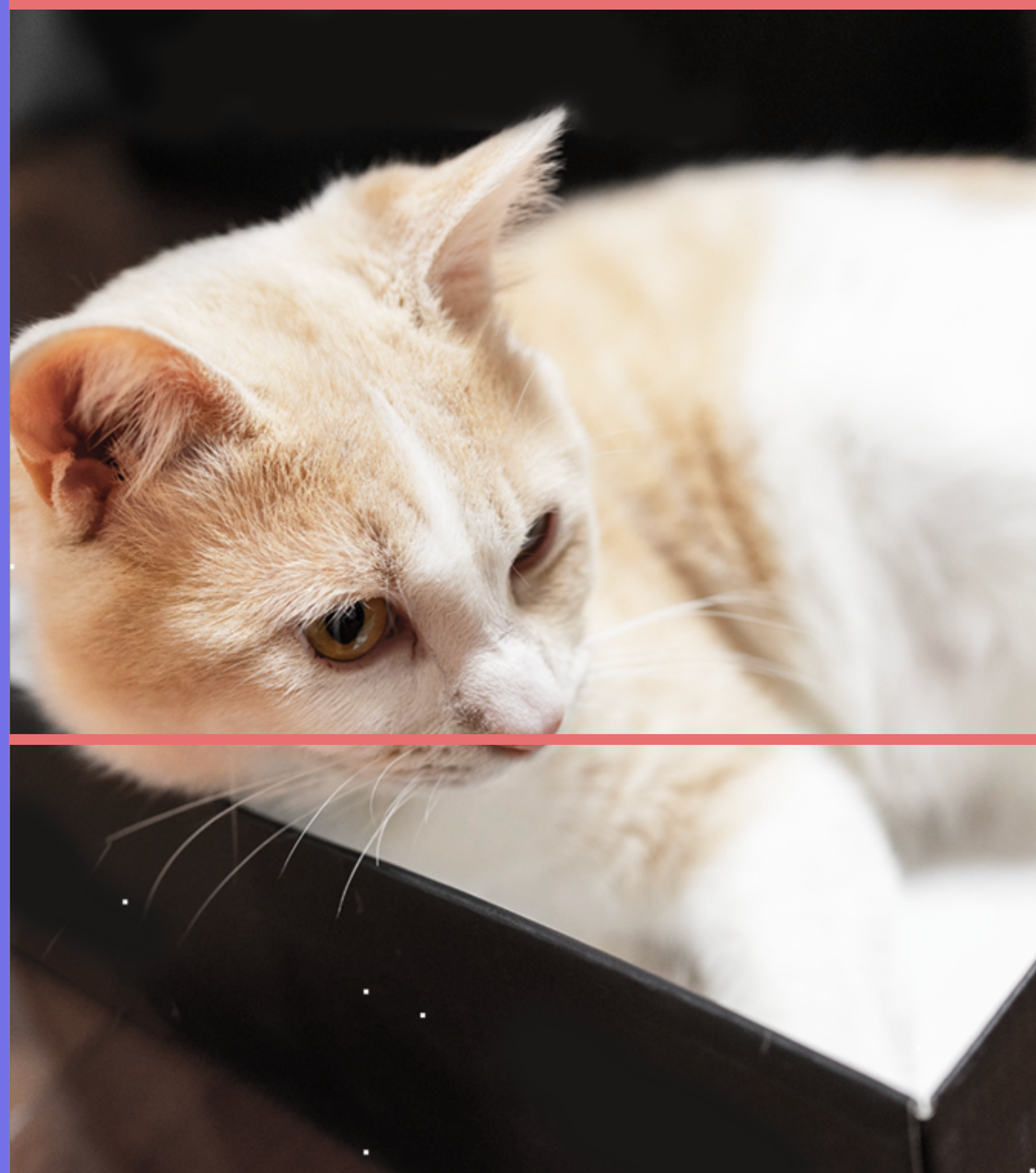
今日のうにちゃん

行(row)

列(column)



## 2. 行と列を作る (テンプレートの作成)



列(column)



行(row)

UNI OF THE DAY

今日のうにちゃん

行(row)

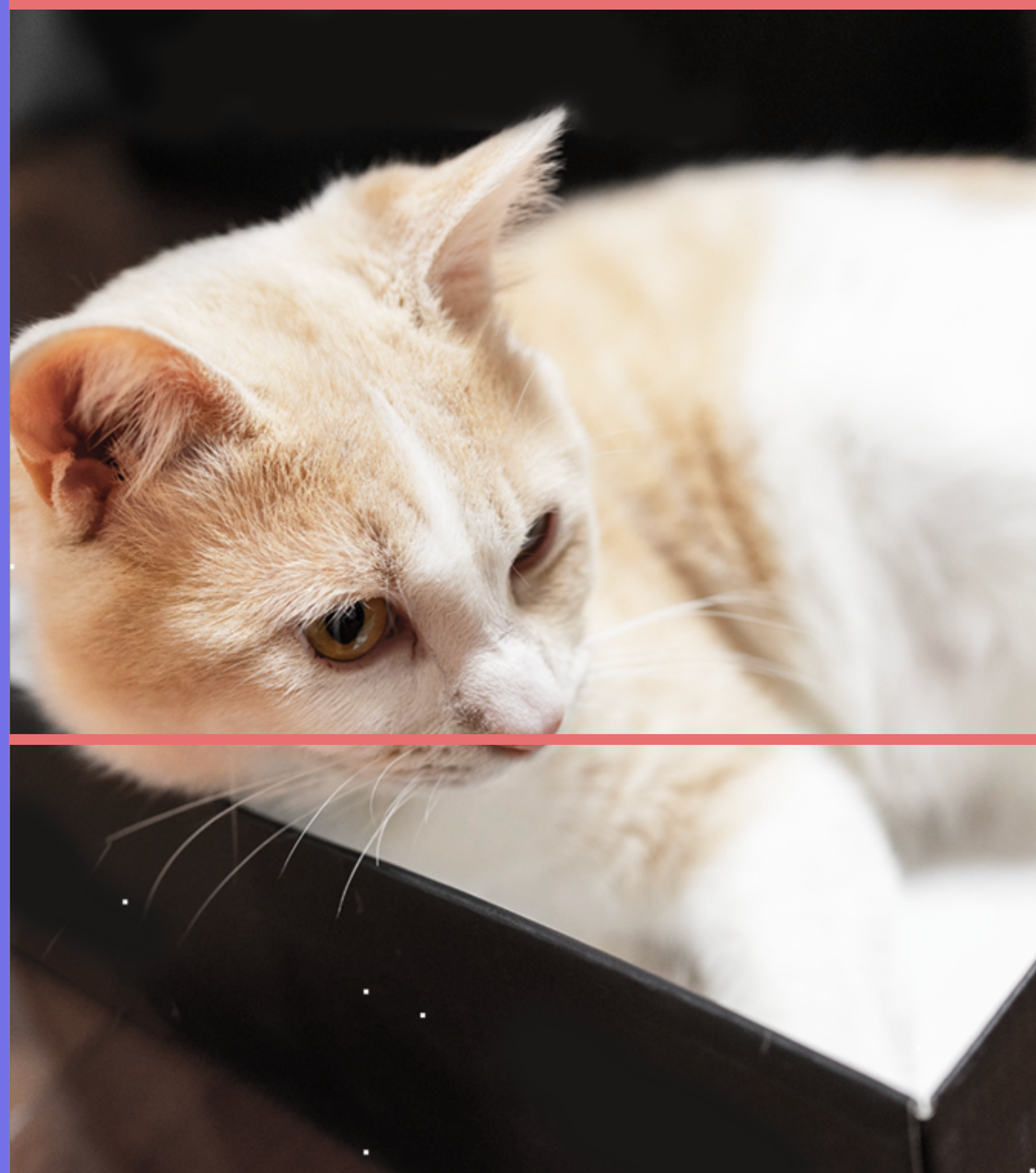
列(column)

**行列は**

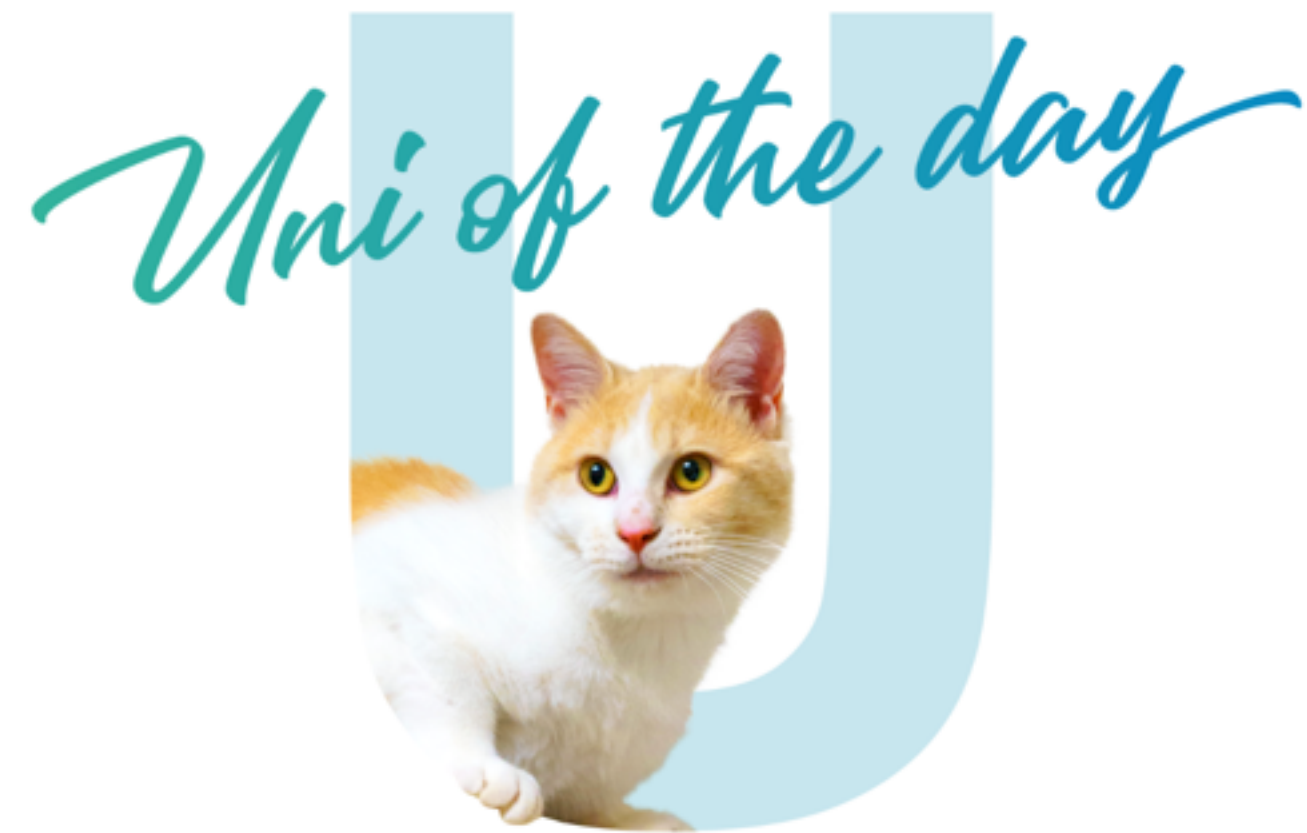
**「サイズ」と「エリアの名前」で決定する**



## 2-1. 行と列のサイズの決定



50%



70%

**UNI OF THE DAY**

今日のうにちゃん

30%

50%



## 2-2. 各エリアに名前をつける





# 行と列を作るためのCSSコード

```
.hero-area {  
  grid-template:  
    "main-visual logo" 70%  
    "main-visual title" 30% /  
    50% 50%;  
}
```

# 行と列を作るためのCSSコード

```
.hero-area {  
  grid-template:  
    "main-visual | logo" 70%  
    "main-visual | title" 30% /  
    50% 50%;  
}
```

エリア名 ↓

行のサイズ ↓

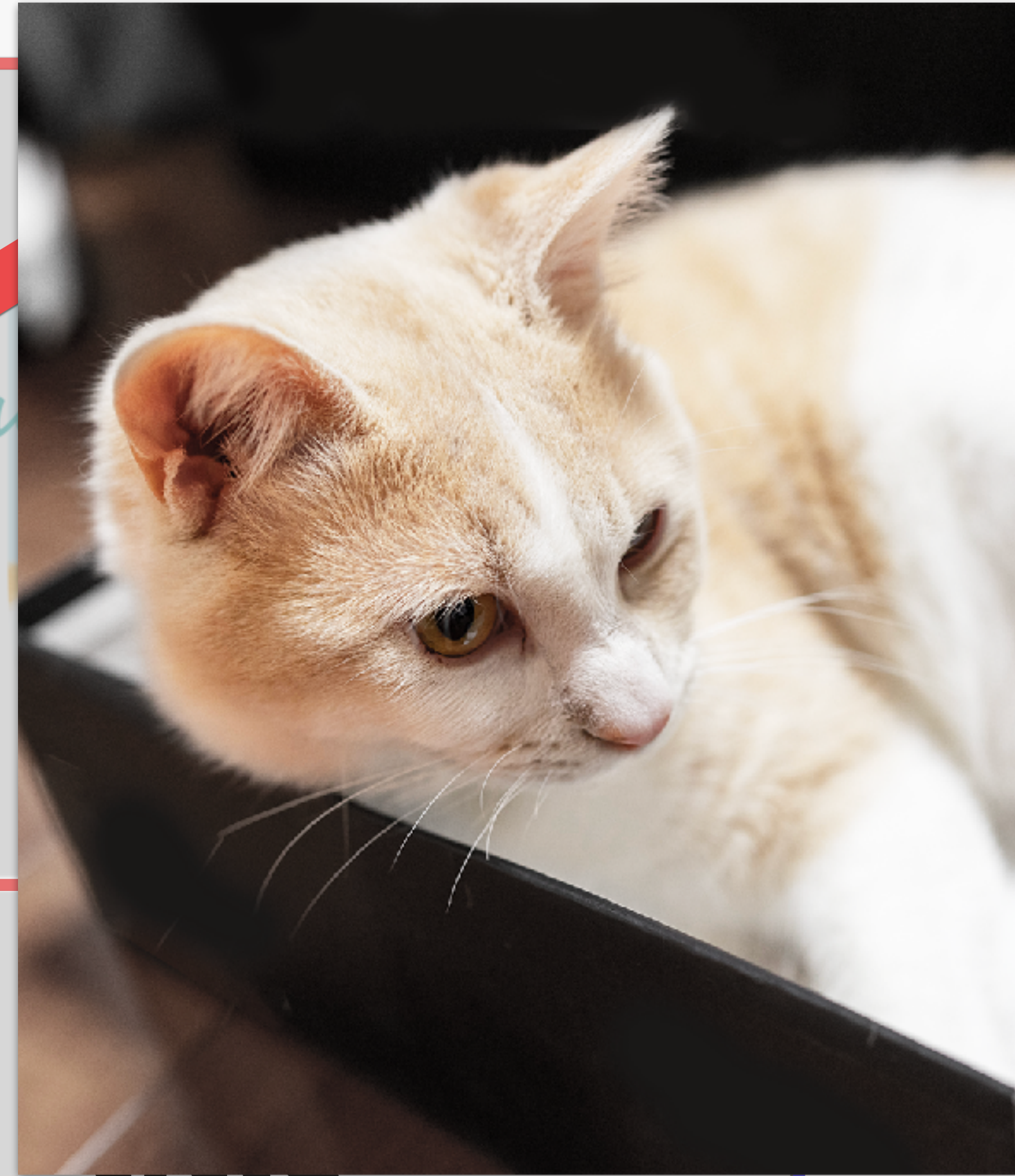
"main-visual   logo"	70%
"main-visual   title"	30% /

50% 50%; ← 列のサイズ

行と列を区切るスラッシュ



### 3. アイテムを配置する





# CSS Gridを書くための3ステップ

1. コンテナを作る

`display: grid`



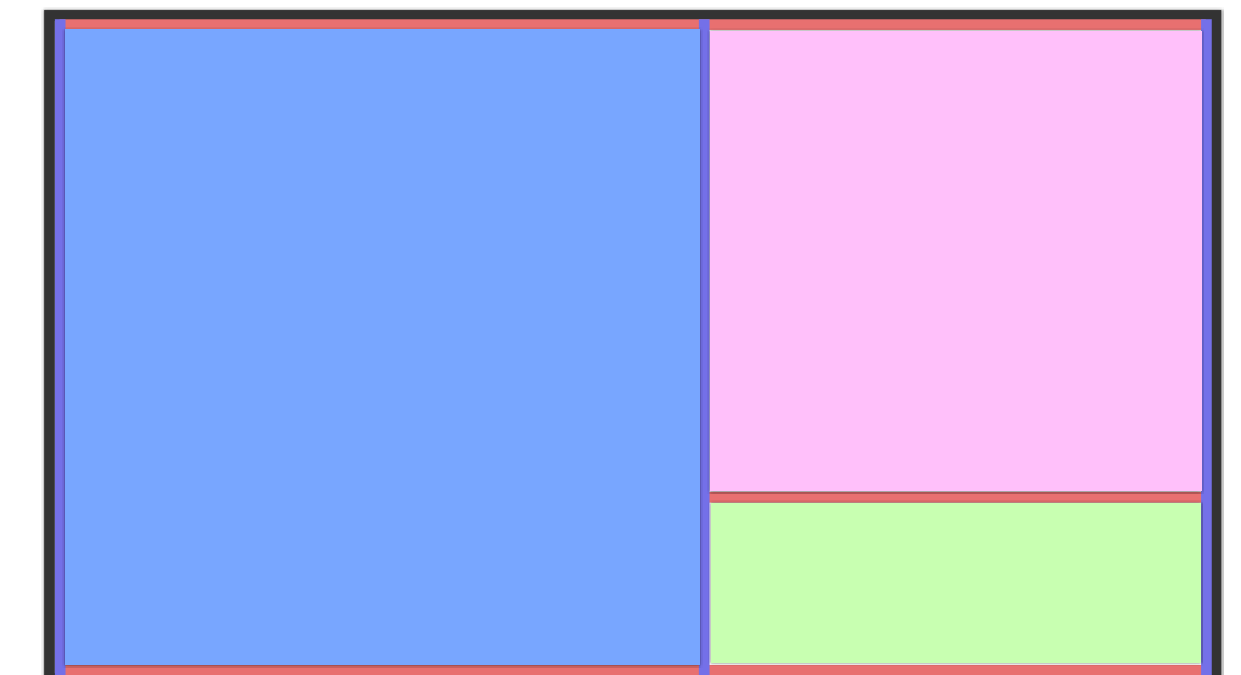
2. 行と列を作る

`grid-template`  
プロパティ



3. アイテムを配置する

`grid-area`  
プロパティ





■ サンプルファイル  
sample1.html  
sample1.css



# 現場で働くコーダーのためのCSS Grid + モダンコーディング

## 1. CSS Grid

- 従来のボックスレイアウト手法
- CSS Gridの基本
- CSS Gridの便利な機能
- CSS GridのIE11対応方法

## 2. 抑えておきたいCSSの新機能

## 3. 情報のキャッチアップ方法



# レスポンシブ対応

grid-templateを書き換えるだけ

```
@media (max-width: 520px) {  
  .hero-area {  
    grid-template:  
      "main-visual" 50%  
      "logo" 35%  
      "title" 1fr /  
      100%;  
  }  
}
```

# 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;  
}
```



# 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 コードを書きながら解説します

## ■ サンプルファイル

sample2.html

sample2.css



# 現場で働くコーダーのためのCSS Grid + モダンコーディング

## 1. CSS Grid

- 従来のボックスレイアウト手法
- CSS Gridの基本
- CSS Gridの便利な機能
- CSS GridのIE11対応方法

## 2. 抑えておきたいCSSの新機能

## 3. 情報のキャッチアップ方法



**Q. CSS Gridを使いたいけど  
IE11では使えないよね？**



**A. 使えます**

# IE 11はCSS Gridの古い記法に対応している

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

※ 記法が異なる



**Autoprefixerを使って  
IE 11に対応した記法に変換する**

# 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



# 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](https://qiita.com/tonkotsuboy_com/items/377913c51b1ac00deffe)

Sassの変換にオススメ! Parcel入門



<https://ics.media/entry/19580/>

# オンラインツールで自動変換する方法

Autoprefixer CSS online

Autoprefixer is a PostCSS plugin which parse your CSS and add vendor prefixes

Postcss: v7.0.2, autoprefixer: v9.1.5

```
.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;
}
```

```
.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;
}
```

Filter last 4 version Apply Select result

You can also see which browsers you choose by filter string on [browserl.ist](https://browserl.ist)

7 207

## Autoprefixer CSS online

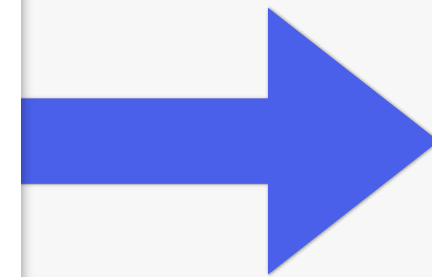
- 環境構築不要

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



# Autoprefixerで自動変換

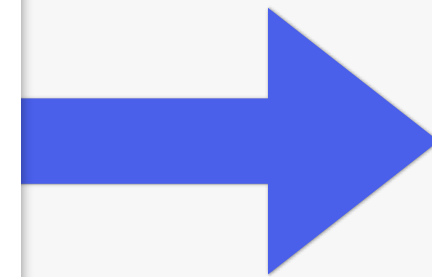
```
.container {  
  display: grid;  
  grid-template:  
    "visual number expression" 1fr  
    "visual other other" 220px /  
    40% 120px 1fr;  
}
```



```
.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;  
}
```

# Autoprefixerで自動変換

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



```
.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;  
}
```



# エリア名の変換

# 昔は、IE 11向けにエリアを指定できなかった

## 昔は番号指定が必要だった



### 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;  
}
```



# 今は、AutoprefixerでエリアをIE 11向けに変換可能

before

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

# 今は、AutoprefixerでエリアをIE 11向けに変換可能

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;  
}
```



# 行・列間の隙間 (gap) の IE11向け変換

# CSS Gridで行と列の間隔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;  
}
```

# CSS コードを書きながら解説します

## ■ サンプルファイル

`sample1.html`

`sample1.css`



**repeatをIE11向けに変換するときは  
ちょっと注意が必要**

# CSS Gridで行と列の間隔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;  
}
```

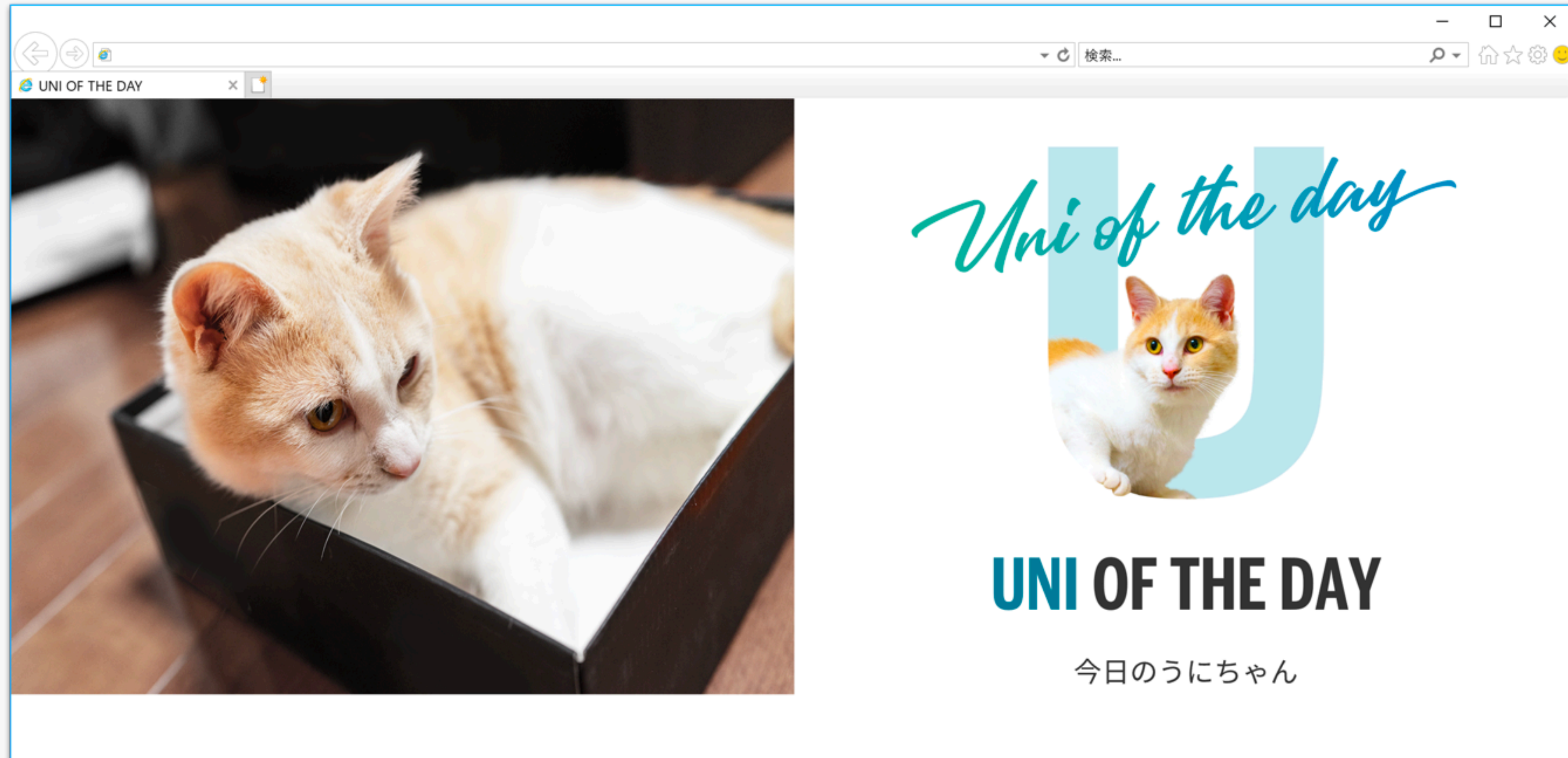


# AutoprefixerでgapをIE 11向けに変換

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;  
}
```

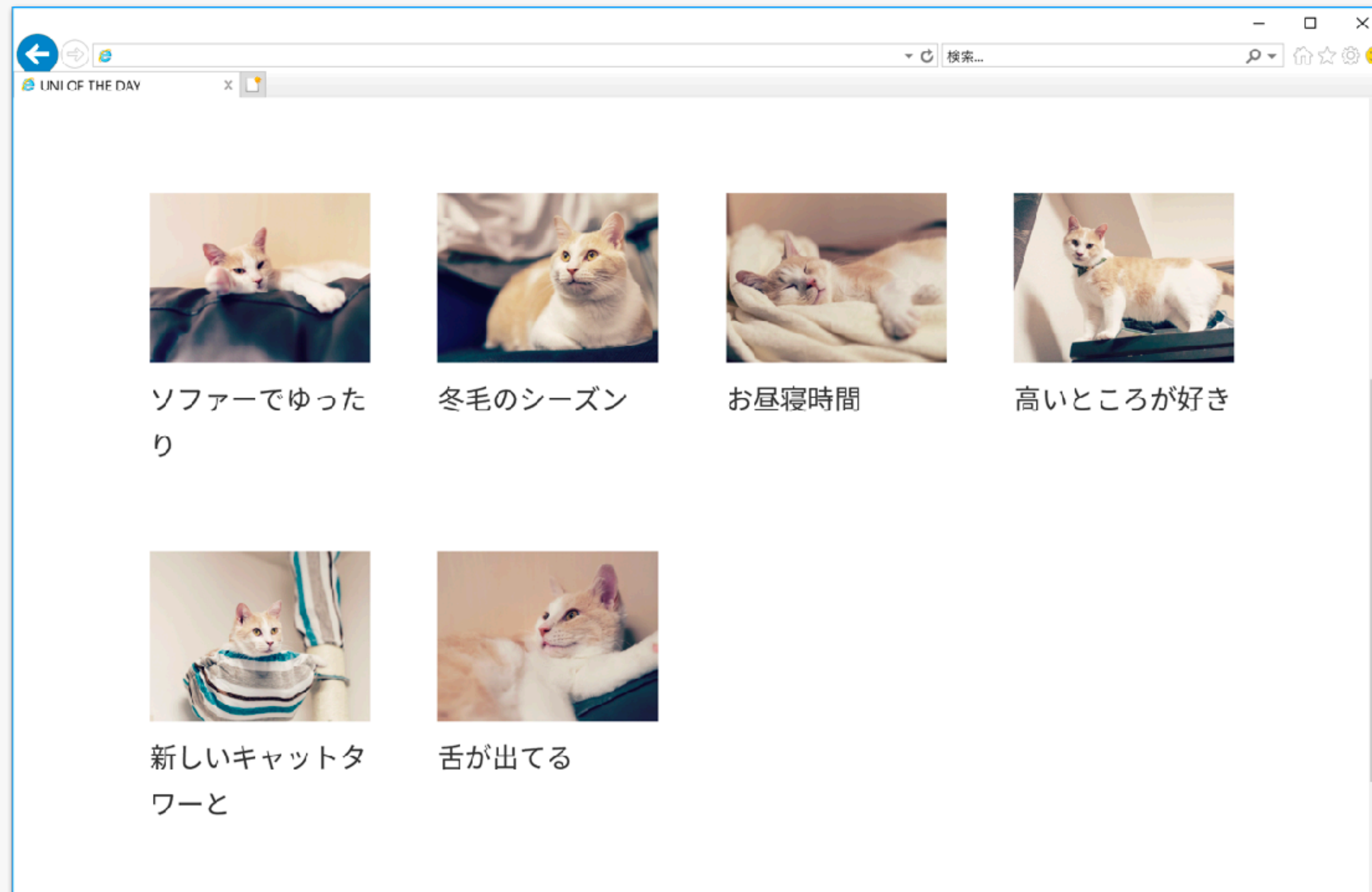
# sample1をIE 11に対応したものの



sample1\_ie11.html / sample1\_ie11.css



# sample2をIE 11に対応したものの



[sample2\\_ie11.html](#) / [sample2\\_ie11.css](#)

※ grid-templateの記述に注意が必要 (フォローアップ参照)



# 現場で働くコーダーのためのCSS Grid + モダンコーディング

## 1. CSS Grid

## 2. 抑えておきたいCSSの新機能

### 1. 最近のモダンブラウザで使えるCSSの機能

### 2. 近い将来使えるようになるCSSの機能

## 3. 情報のキャッチアップ方法



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

# CSS变数



# CSS変数: CSSで変数が見える

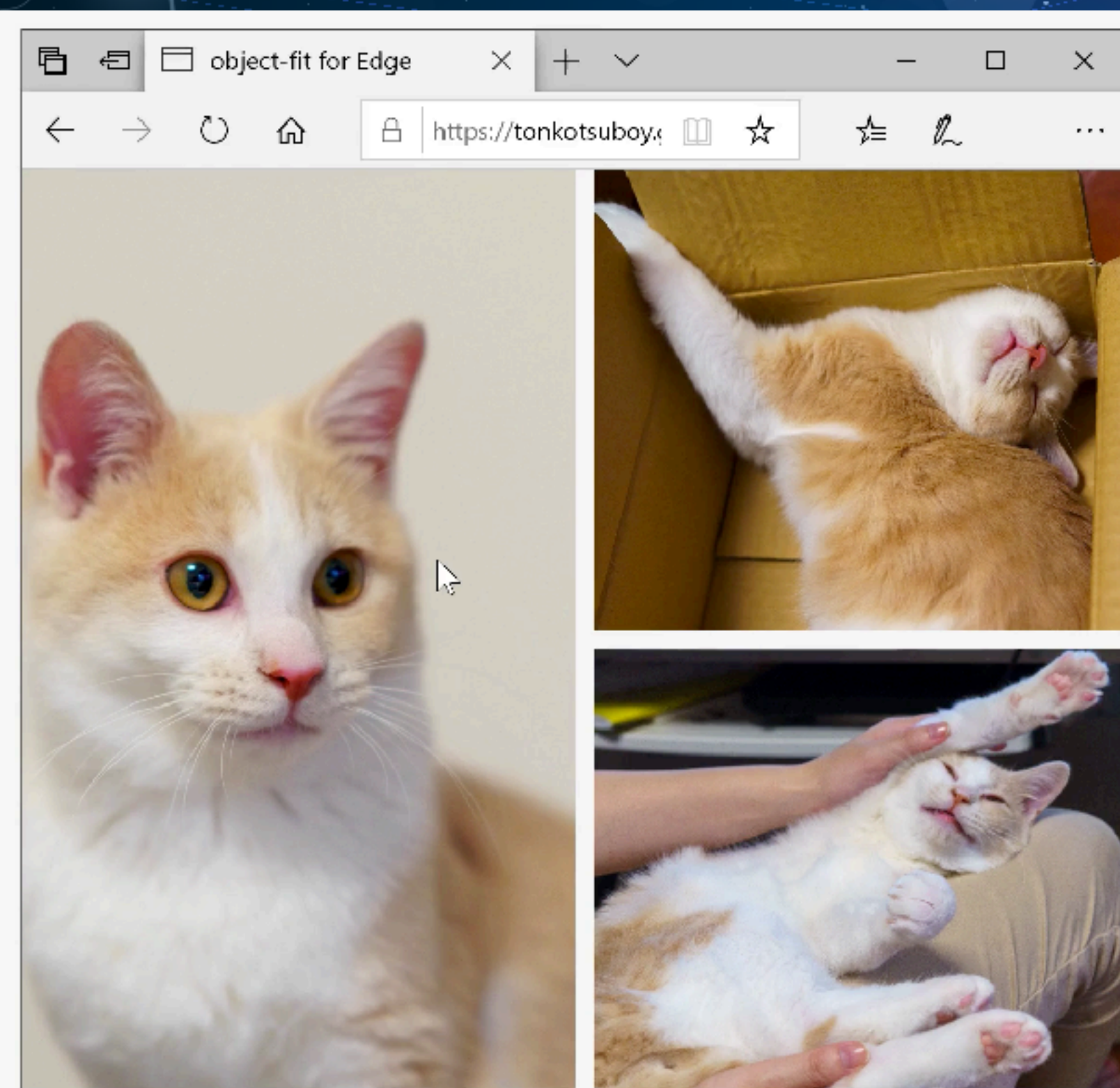
```
/* 通常時は黒色テキスト・白色背景 */  
:root {  
  --text-color: #000000;  
}  
  
/* ダークモードでは灰色テキスト・黒色背景を指定する */  
@media (prefers-color-scheme: dark) {  
  :root {  
    --text-color: #ffffff;  
  }  
}  
  
/* 通常時は黒色、ダークモードで白色に変わるテキスト */  
p {  
  color: var(--text-color);  
}
```

**画像の比率を保ちながら**

**変形できる**

**「object-fit」プロパティ**

# imgタグで設定した画像の比率を保つ



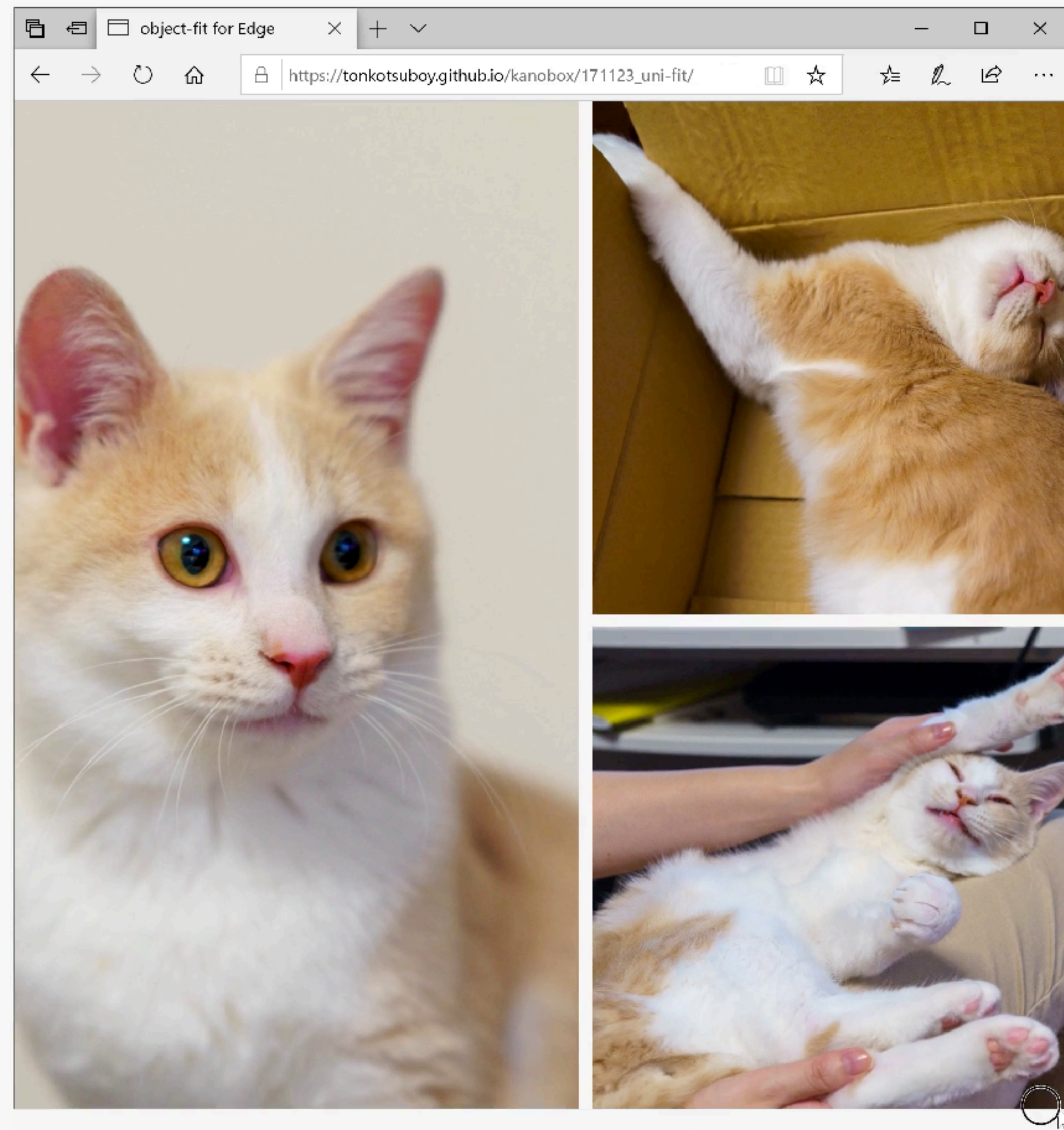
HTMLコード

```
  
  

```



# imgタグで設定した画像の比率を保つ



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

(2017年10月リリースのEdge v16で対応)

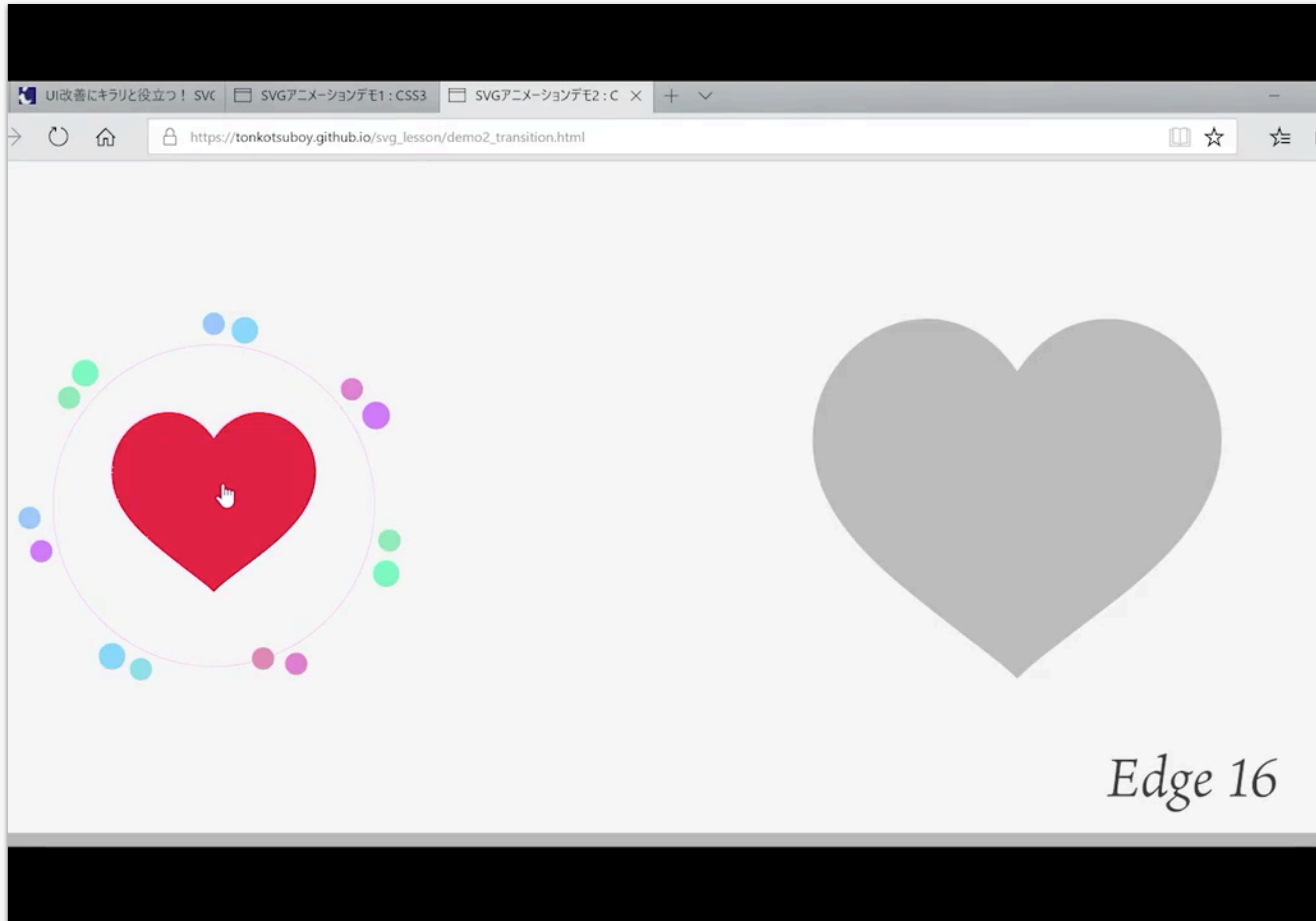
```
img {  
  object-fit: cover;  
}  
.image1 {  
  object-position: 30% 50%;  
}  
.image2 {  
  object-position: 0 0;  
}
```

うに fit

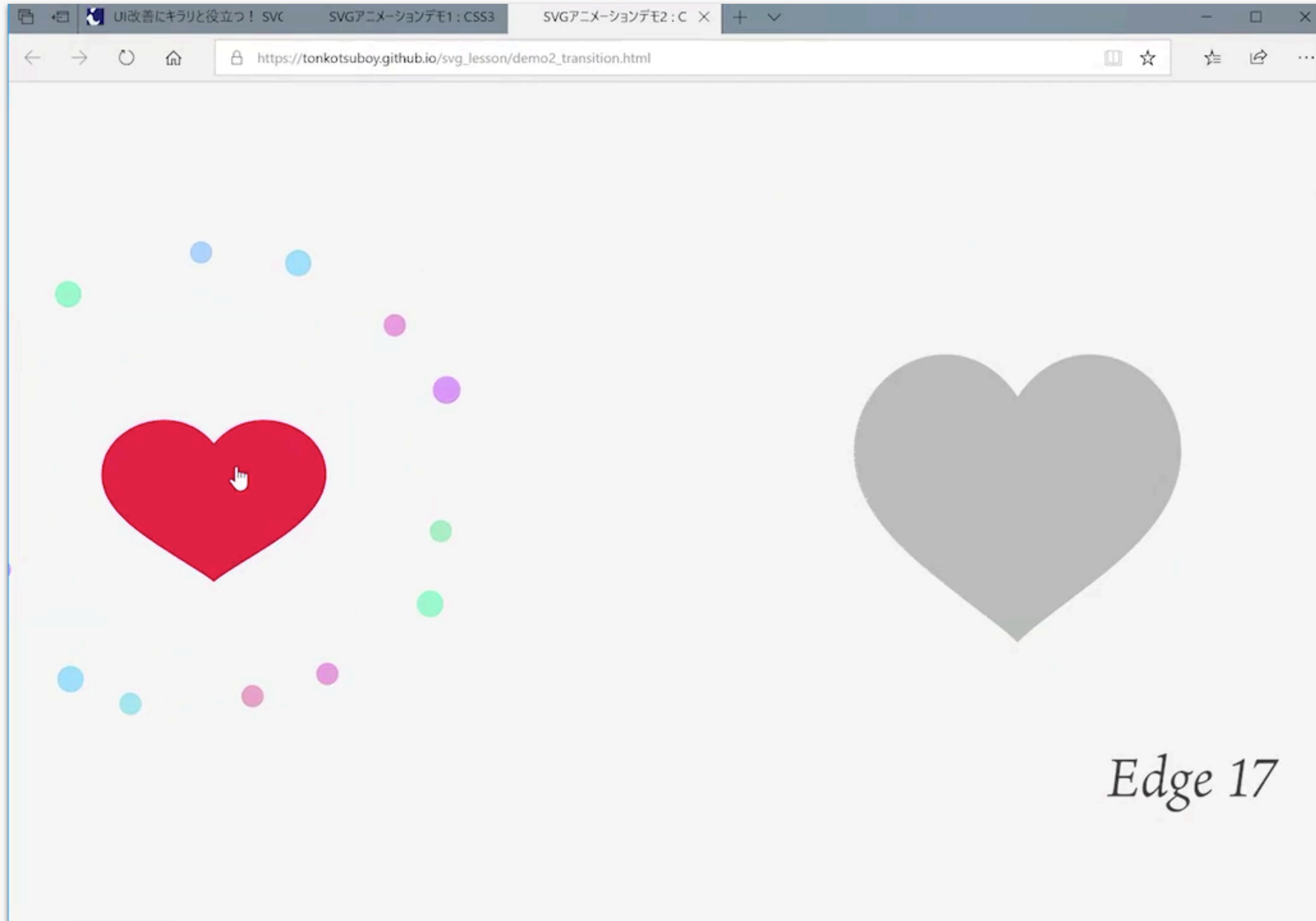
**SVG要素を  
CSS Transformsで変形する**



# 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  
maskプロパティ

# 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

(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



一つのフォントファイルが  
複数のフォントのように動作する  
「variable font」

# variable font: 1つのフォントから複数の見栄え

複数のフォントファイルを読み込まず、CSSで見栄えを制御可能



CSS Nite Osaka

Weight 226.75

xHeight 459.2



**CSS Nite Osaka**

Weight 712.00

xHeight 432.83

# variable font: 1つのフォントから複数の見栄え

複数のフォントファイルを読み込まず、CSSで見栄えを制御可能

```
p {  
  font-variation-settings:  
    "wght" 25.31,  
    "XHGT" 353,  
    "opsz" 13.03;  
}
```



# 現場で働くコーダーのためのCSS Grid + モダンコーディング

## 1. CSS Grid

## 2. 抑えておきたいCSSの新機能

### 1. 最近のモダンブラウザで使えるCSSの機能

### 2. 近い将来使えるようになるCSSの機能

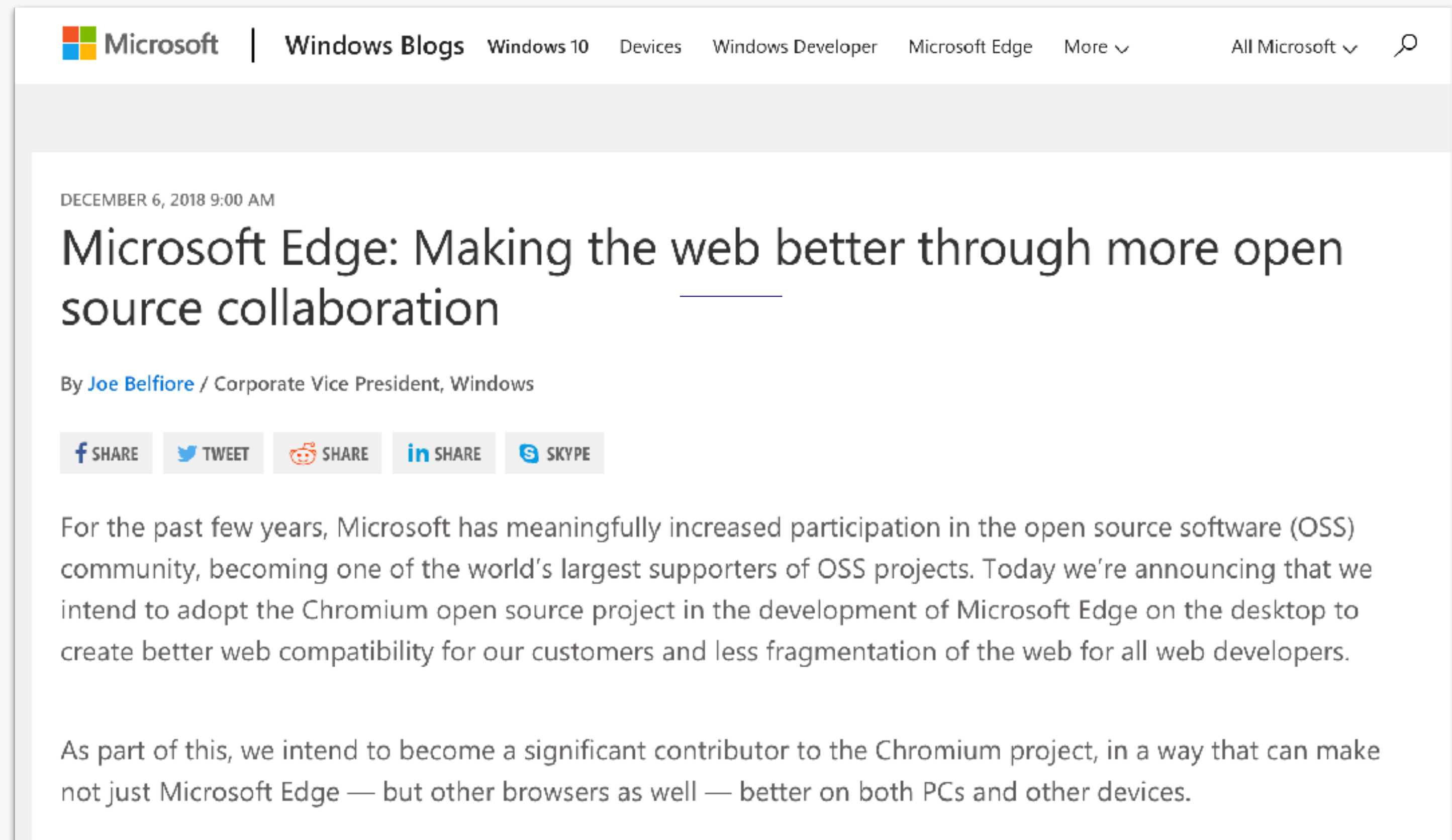
## 3. 情報のキャッチアップ方法



**Microsoft Edgeの次バージョンより  
全モダンブラウザ向けCSSの機能は  
更に増える**

# Microsoft EdgeがChromiumベースになる

## 2018年12月に発表



Windows Experience Blog (2018年12月)



# 「Chromium」とは？

## Chromium (クロミウム)

- Google Chromeのもととなっているプログラム
- HTML・CSSのレンダリング: Blink (ブリンク)
- JavaScriptの実行: V8



Chromiumのロゴ

# Edgeの次バージョンからはChromeと同じ構成になる

## Edge v18までの構成

- HTML・CSSのレンダリング: EdgeHTML
- JavaScriptの実行: Chakra (チャクラ)



## 次のEdgeからの構成

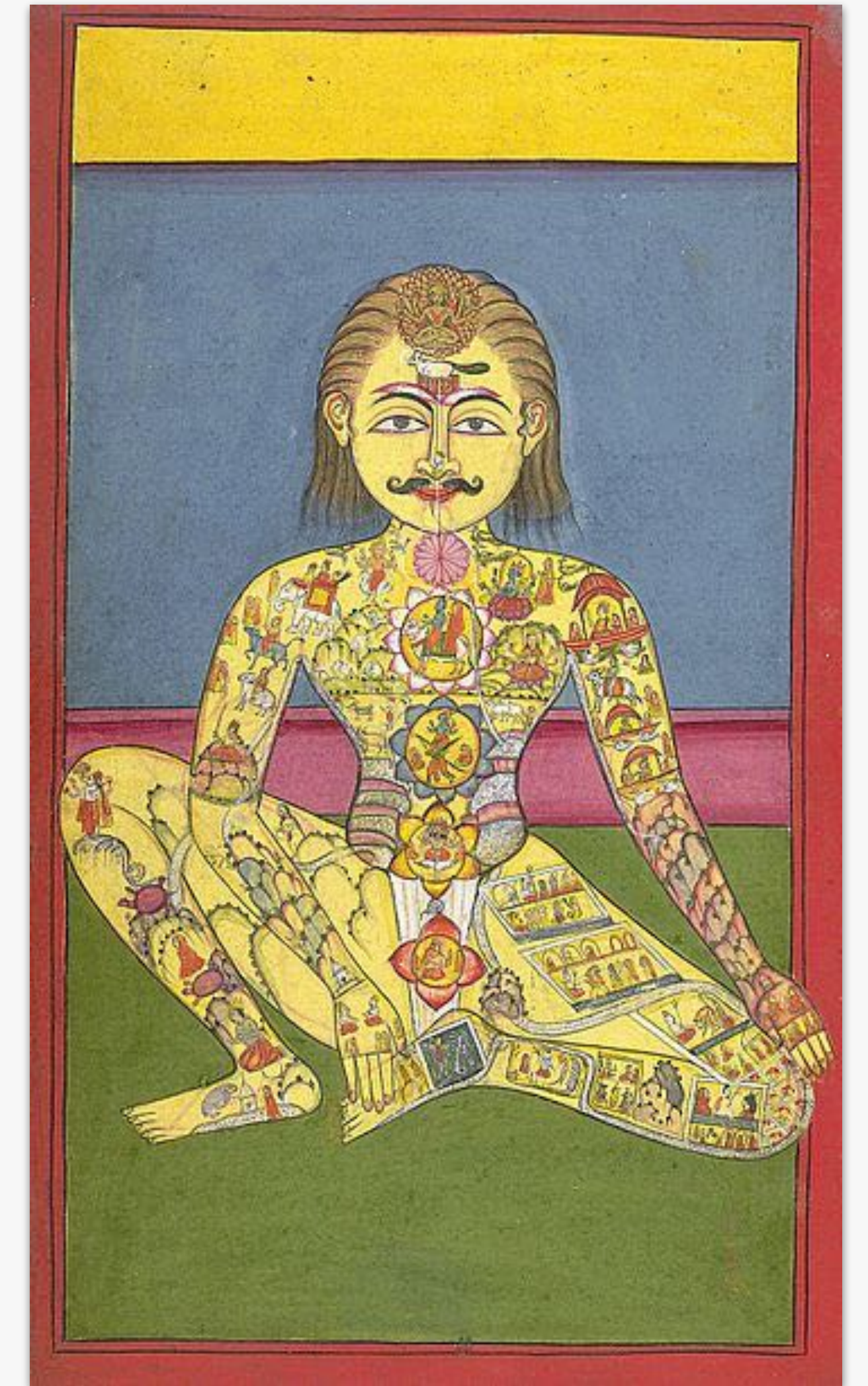
- HTML・CSSのレンダリング: Blink
- JavaScriptの実行: V8



## ちなみにチャクラとは…

サンスクリットで円、円盤、車輪、ろくろを意味する語。

ヒンドゥー教のタントラやハタ・ヨーガ、仏教の後期密教では、人体の頭部、胸部、腹部などにあるとされる中枢を指す言葉として用いられる。





# EdgeのChromium化によりCSSの対応が増える

## Edgeの次バージョンで使えるCSS機能の一例

will-change property

prefers-reduced-motion media query

prefers-color-scheme media query

display: flow-root

CSS text-orientation

CSSOM Scroll-behavior

#rrggbaa hex color notation

:placeholder-shown CSS pseudo-class

CSS Motion Path

CSS font-rendering controls

:focus-within CSS pseudo-class

CSS Environment Variables env()

:default CSS pseudo-class

CSS Conical Gradients

Case-insensitive CSS attribute selectors

CSS background-blend-mode

CSS all property

CSS Font Loading

:matches() CSS pseudo-class

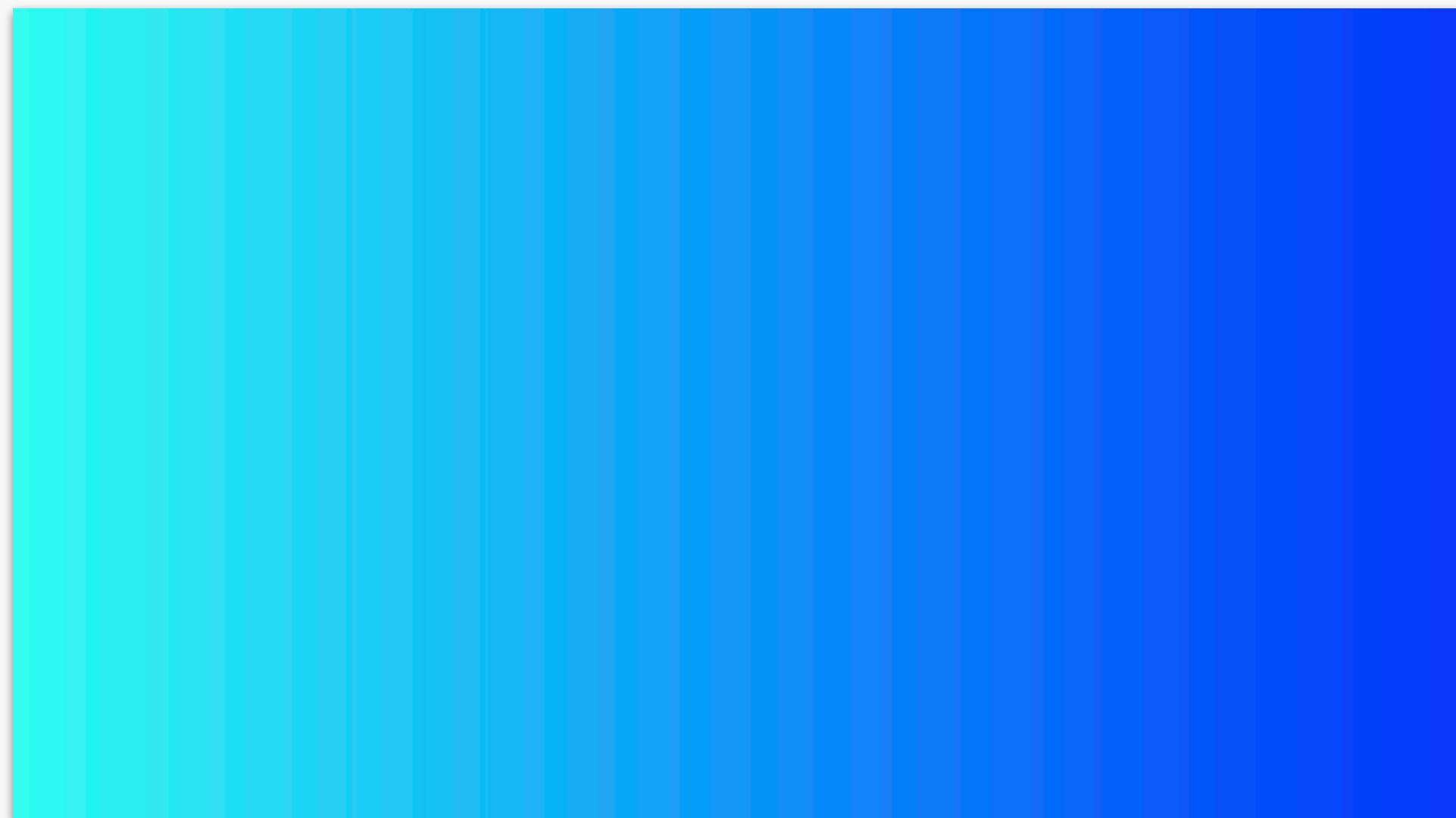
CSS display: contents

**「円錐状」のグラデーション**

**Conical Gradients**

# 従来のグラデーション

## 線形



```
.box {  
  background-image: linear-gradient(  
    to right,  
    hsl(177, 100%, 50%),  
    hsl(232, 100%, 50%)  
  );  
}
```



# 従来のグラデーション

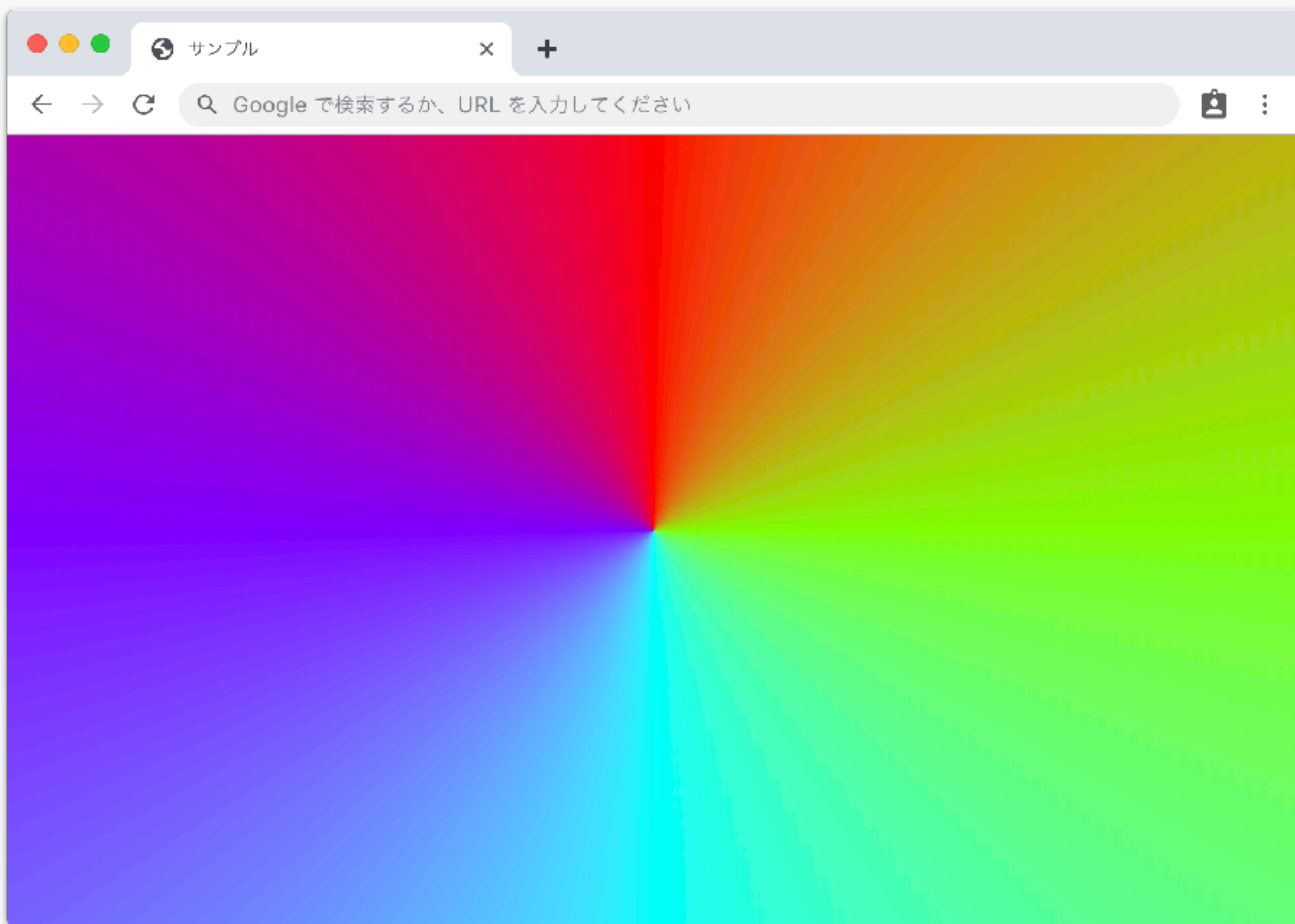
## 円形



```
.box {  
  background-image: radial-gradient(  
    hsl(177, 100%, 50%),  
    hsl(232, 100%, 50%)  
  );  
}
```

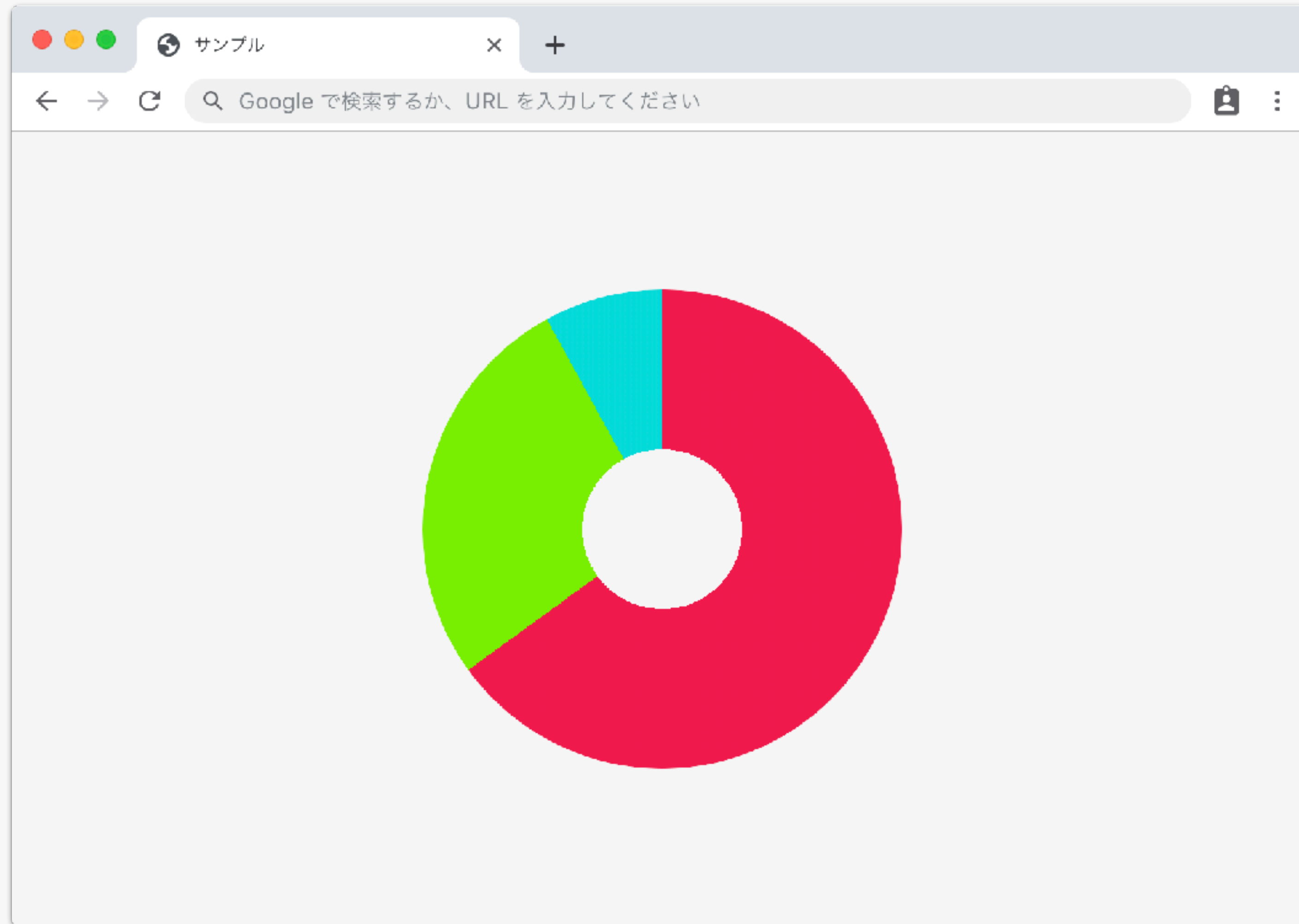
# Conical Gradients

## 「円錐状」のグラデーションを指定する値



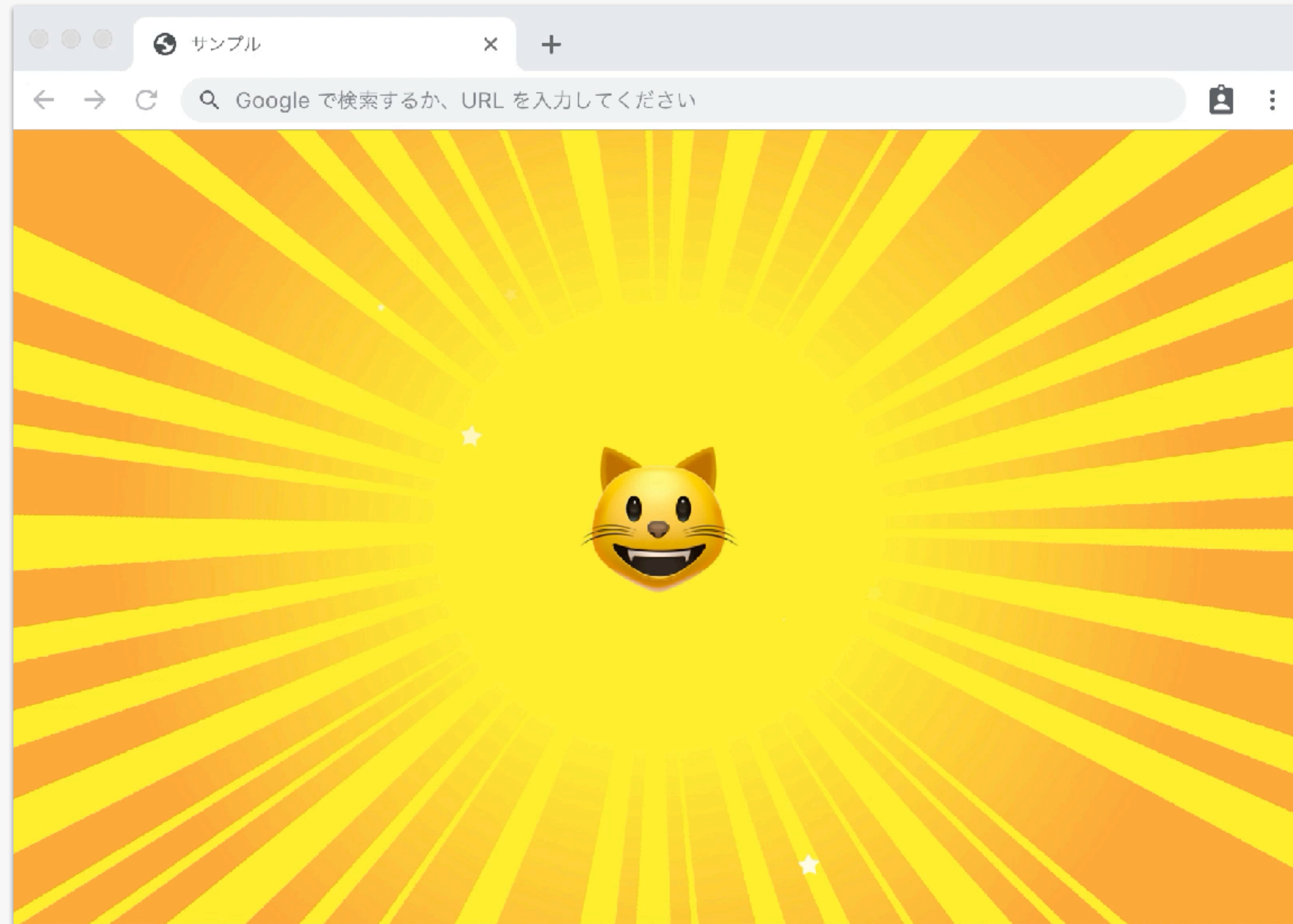
```
.box {  
  background-image: conic-gradient(  
    hsl(0, 100%, 50%) 0,  
    hsl(90, 100%, 50%) 90deg,  
    hsl(180, 100%, 50%) 180deg,  
    hsl(270, 100%, 50%) 270deg,  
    hsl(360, 100%, 50%) 360deg  
  );  
}
```

# Conical Gradientsで円グラフの表現





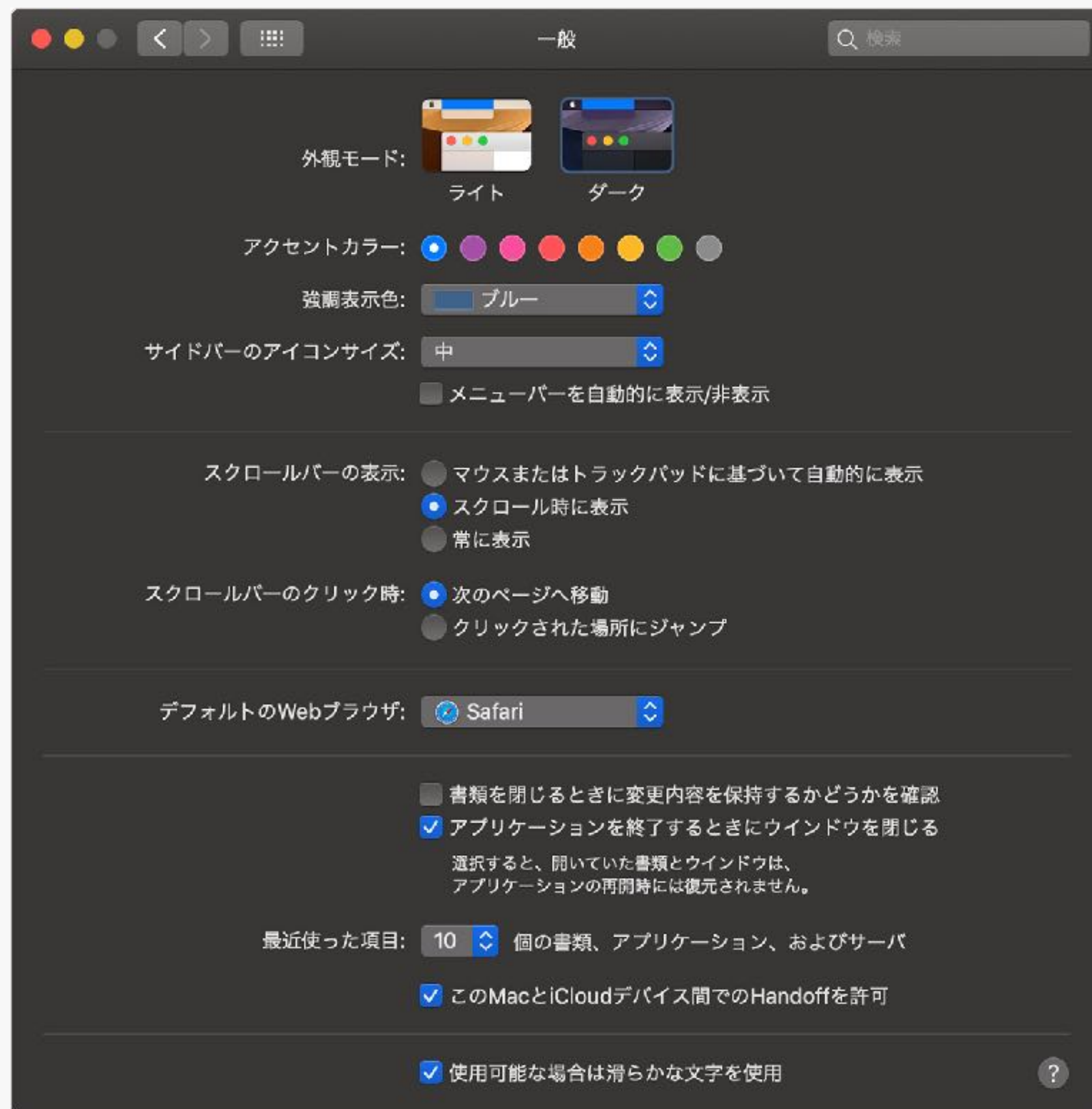
# Conical Gradientsで集中線の表現



**OSのダークモードを判別できる**

**prefers-color-scheme media query**

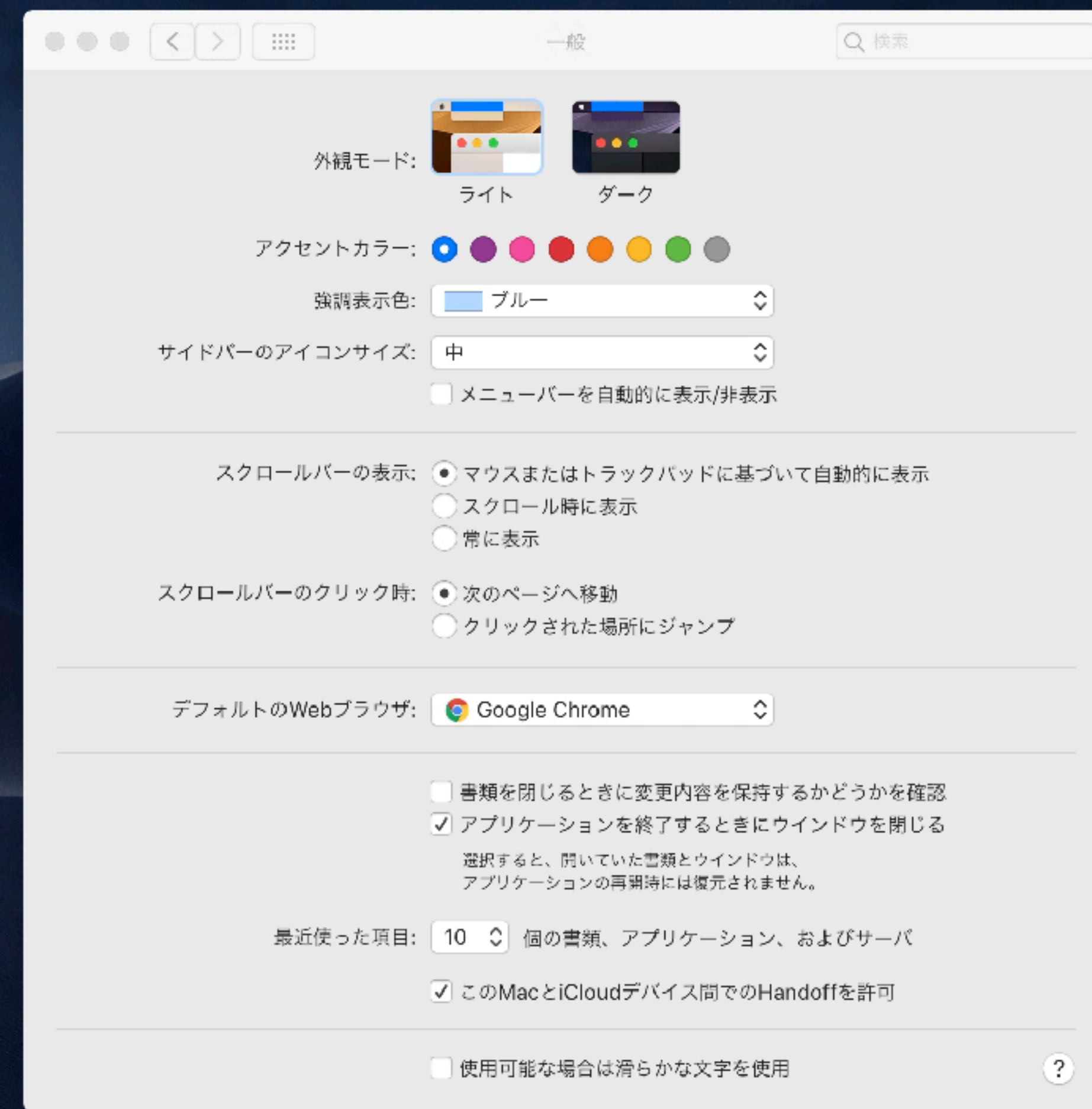
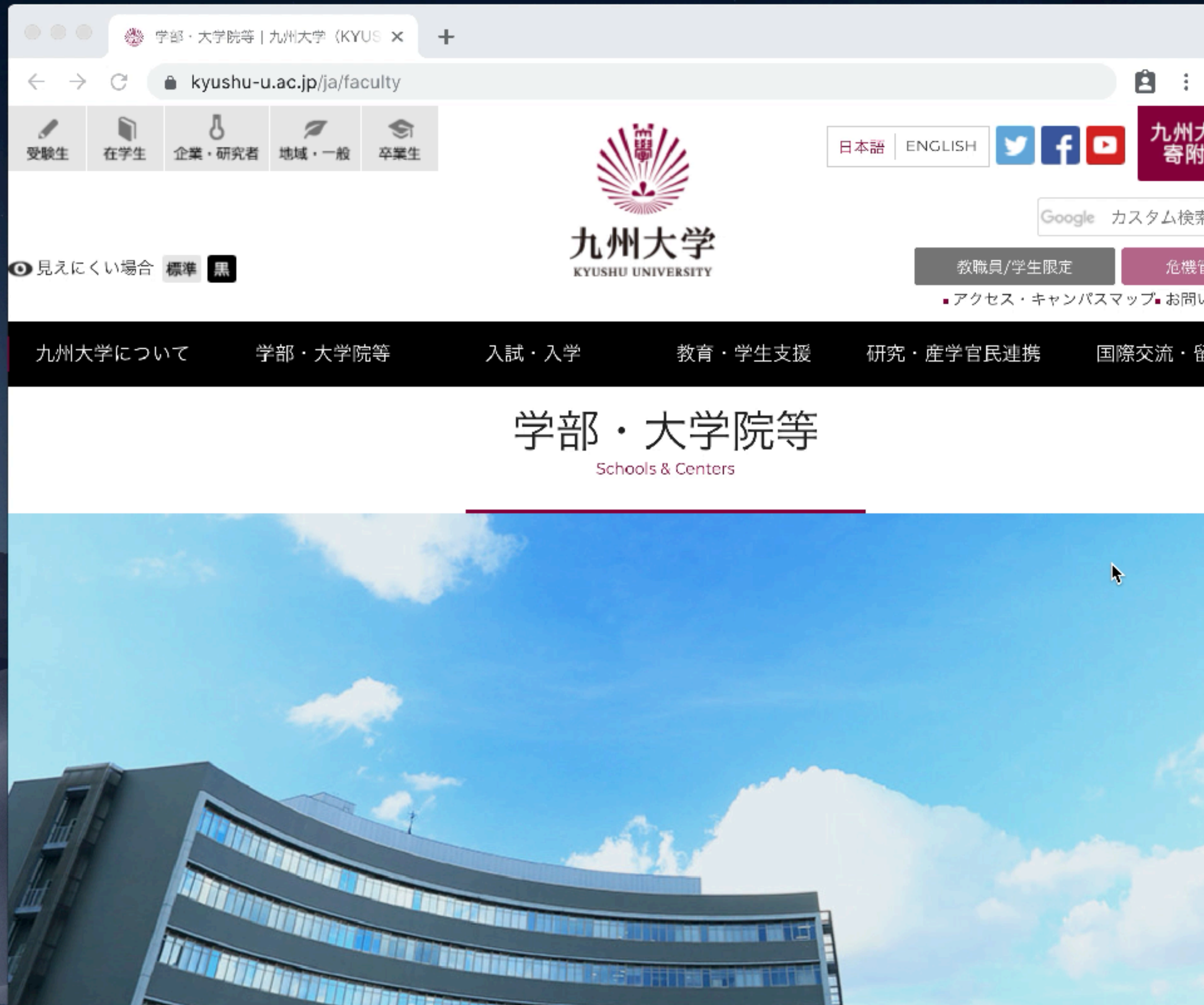
# 各OSにはダークモードがある



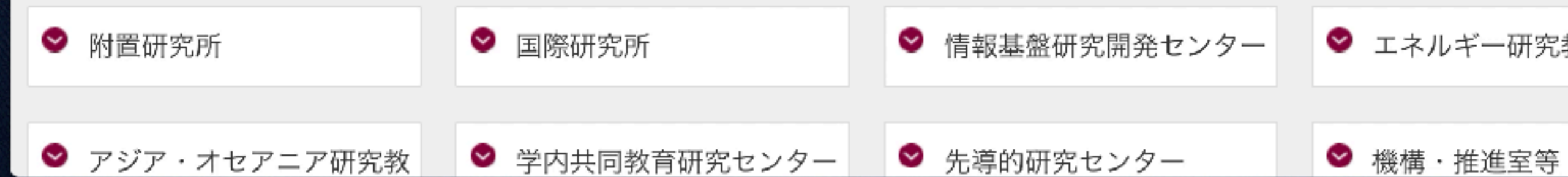
- Windows
- macOS
- iOS (13~)
- Android※

※ Androidはダークテーマという名称

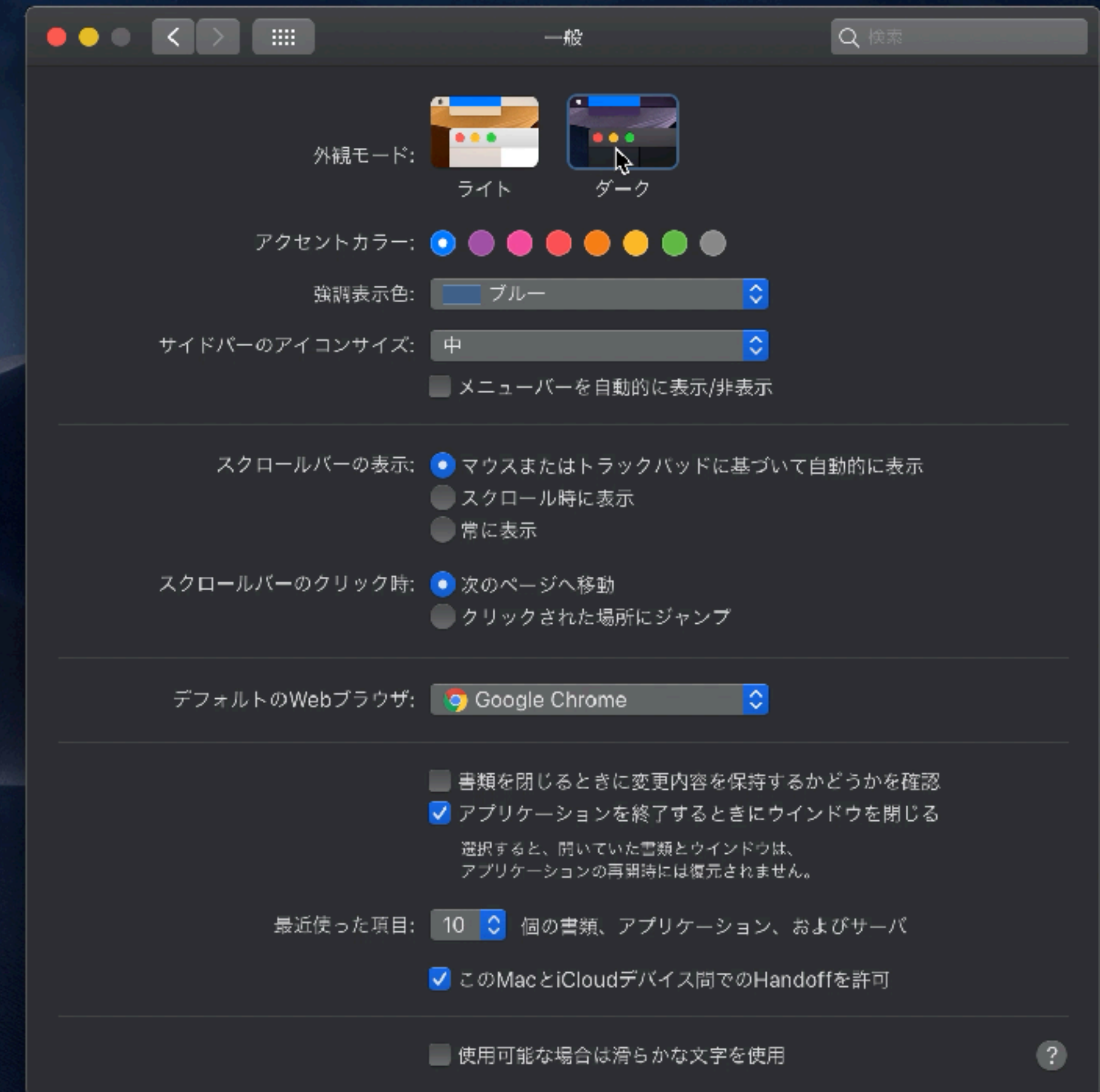
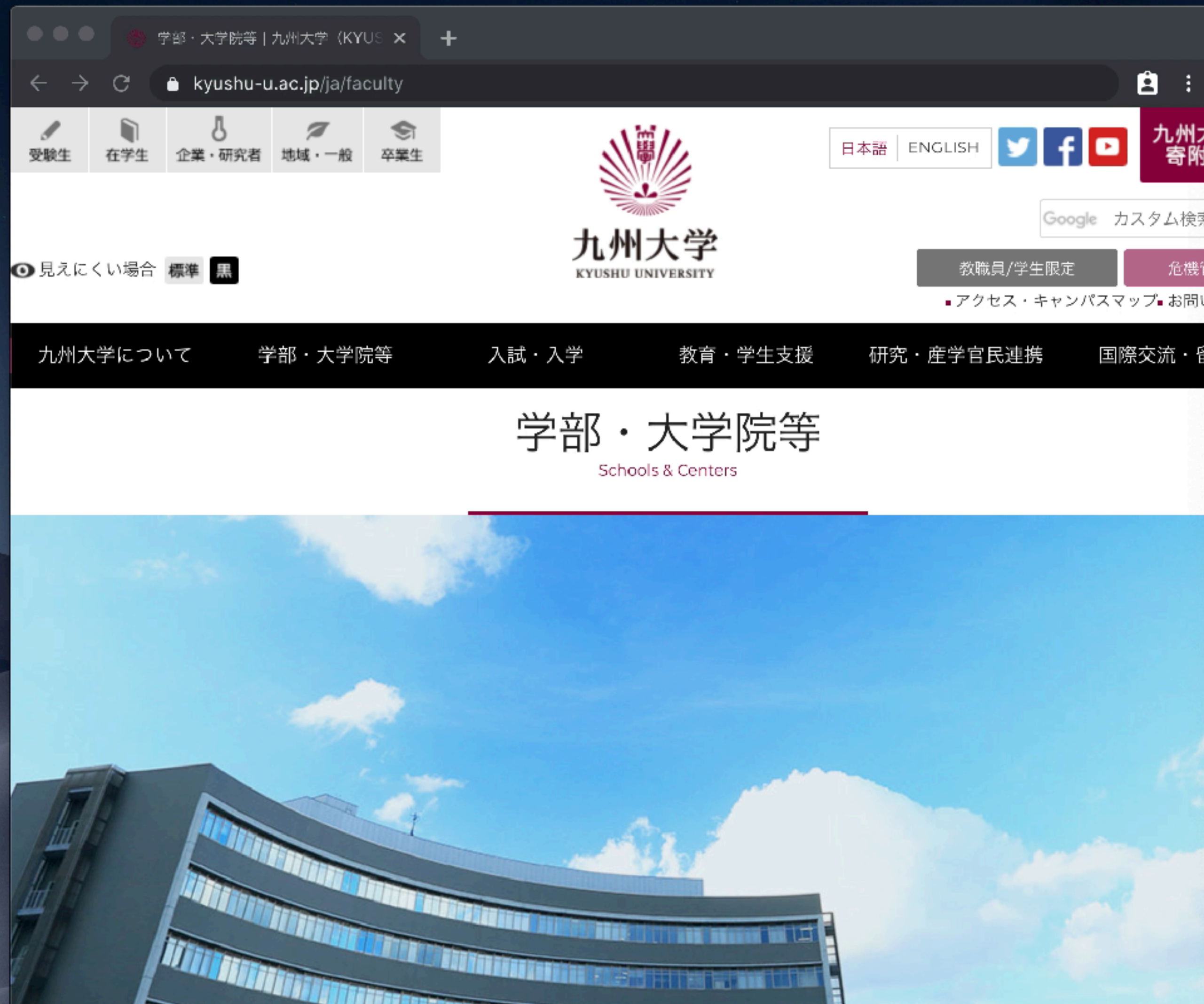




# ダークモードにしても、コンテンツの外観は明るいママ問題







# ダークモードにしても、コンテンツの外観は明るいママ問題





# prefers-color-schemeを使う場合

```
/* 通常は背景を白にする */  
body {  
  background-color: white;  
}  
  
/* ダークモードのときは背景を黒にする */  
@media (prefers-color-scheme: dark) {  
  body {  
    background-color: black;  
  }  
}
```



Chrome ファイル 編集 表示 履歴 ブックマーク ユーザー ウィンドウ ヘルプ


ICS MEDIA - Webデザイナー/フロント  
ics.media

ics.media


UIデザイナー HTMLコーダー フロントエンジニア クリエイティブコーダー 3Dデベロッパー アプリ開発者

ウェブデザイナーとフロントエンジニアのための情報メディア  
ICS MEDIA

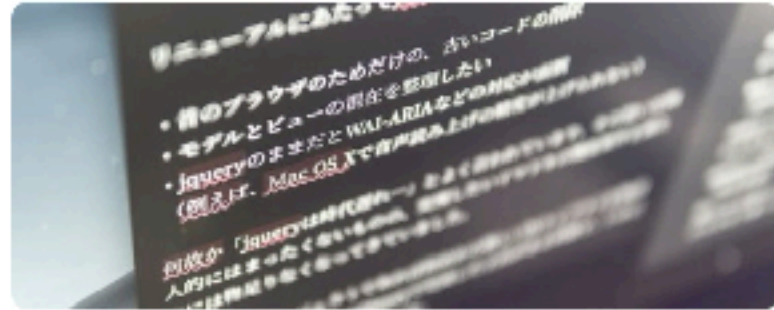
### 話題になった記事



「JavaScriptコードレシピ集」を執筆しました  
鹿野 社 平成31年1月21日 ♥ 1534



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 泰延 平成30年5月17日 ♥ 4695



エンジニアのための文章校正テクニック  
池田 泰延 平成30年10月5日 ♥ 1404

### 記事一覧

新着 人気

一般

外観モード: ライト ダーク

アクセントカラー: [Color Selection]

強調表示色: ブルー

サイドバーのアイコンサイズ: 中

メニューバーを自動的に表示/非表示

スクロールバーの表示:  マウスまたはトラックパッドに基づいて自動的に表示  
 スクロール時に表示  
 常に表示

スクロールバーのクリック時:  次のページへ移動  
 クリックされた場所にジャンプ

デフォルトのWebブラウザ: Google Chrome

書類を閉じるときに変更内容を保持するかどうかを確認  
 アプリケーションを終了するときウィンドウを閉じる  
選択すると、開いていた書類とウィンドウは、アプリケーションの再開時には復元されません。

最近使った項目: 10 個の書類、アプリケーション、およびサーバ

このMacとiCloudデバイス間でのHandoffを許可

使用可能な場合は滑らかな文字を使用

ダークモードに応じて、コンテンツの外観が暗くなる



ICS MEDIA - Webデザイナー/フリーランス

ics.media

配色

UIデザイナー HTMLコーダー フロントエンジニア クリエイティブコーダー 3Dデベロッパー アプリ開発者

ウェブデザイナーとフロントエンジニアのための情報メディア

ICS MEDIA

### 話題になった記事

「JavaScriptコードレシピ集」を執筆しました

鹿野 壮 平成31年1月21日 ♥ 1534

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 泰延 平成30年5月17日 ♥ 4695

エンジニアのための文章校正テクニック

池田 泰延 平成30年10月5日 ♥ 1404

記事一覧

新着 人気

一般

検索

外観モード: ライト **ダーク**

アクセントカラー: ● ● ● ● ● ● ● ●

強調表示色: ブルー

サイドバーのアイコンサイズ: 中

メニューバーを自動的に表示/非表示

スクロールバーの表示:
 

- マウスまたはトラックパッドに基づいて自動的に表示
- スクロール時に表示
- 常に表示

スクロールバーのクリック時:
 

- 次のページへ移動
- クリックされた場所にジャンプ

デフォルトのWebブラウザ: Google Chrome

書類を閉じるときに変更内容を保持するかどうかを確認

アプリケーションを終了するときにウィンドウを閉じる

選択すると、閉じていた書類とウィンドウは、アプリケーションの再開時には復元されません。

最近使った項目: 10 個の書類、アプリケーション、およびサーバ

このMacとiCloudデバイス間でのHandoffを許可

使用可能な場合は滑らかな文字を使用

ダークモードに応じて、コンテンツの外観が暗くなる



# 「CSS変数」と組み合わせせて更に便利に

```
/* 通常時は黒色テキスト・白色背景 */
:root {
  --text-color: #000000;
}

/* ダークモードでは灰色テキスト・黒色背景を指定する */
@media (prefers-color-scheme: dark) {
  :root {
    --text-color: #ffffff;
  }
}

/* 通常時は黒色、ダークモードで白色に変わるテキスト */
p {
  color: var(--text-color);
}
```



**OSのアニメーションが**

**OFFかをどうかを判別できる**

**prefers-reduced-motion media query**

# prefers-color-scheme

```
/* 通常は各要素にアニメーションを設定する */  
  
.element1,  
.element2 {  
  animation: 1s myAnimation;  
}  
  
/* OSのアニメーションがOFF設定の場合は  
CSSアニメーションもOFFにする */  
  
@media (prefers-reduced-motion: reduce) {  
  .element1,  
  .element2 {  
    animation: none;  
  }  
}
```

**display: contents**



# 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;
}
```

# EdgeのChromium化によりCSSの対応が増える

## Edgeの次バージョンで使えるCSS機能の一例

will-change property

prefers-reduced-motion media query

prefers-color-scheme media query

display: flow-root

CSS text-orientation

CSSOM Scroll-behavior

#rrggbaa hex color notation

:placeholder-shown CSS pseudo-class

CSS Motion Path

CSS font-rendering controls

:focus-within CSS pseudo-class

CSS Environment Variables env()

:default CSS pseudo-class

CSS Conical Gradients

Case-insensitive CSS attribute selectors

CSS background-blend-mode

CSS all property

CSS Font Loading

:matches() CSS pseudo-class

CSS display: contents



# 現場で働くコーダーのためのCSS Grid + モダンコーディング

1. CSS Gridとボックスレイアウト

2. 抑えておきたいCSSの新機能

3. 情報のキャッチアップ方法



**どうやってCSSの  
最新機能や対応状況を追うか？**

# Platform Status

The screenshot shows the Chrome Platform Status website. The header includes the Chrome logo, the title "Chrome Platform Status", and navigation links for "All features", "Releases", "Samples", and "Stats". Below the header, it displays "Features: 1397" with a search filter and a "Subscribe" button. The main content area is titled "IN DEVELOPMENT" and lists several features with their descriptions and category tags:

- <dialog>: add DialogShowParams to show()/showModal()** (DOM) - Adds parameters to change how show() or showModal() is called.
- <virtual-scroller>** (DOM) - The <virtual-scroller> element maps a provided set of JavaScript objects onto DOM nodes, and renders only the DOM r
- Accessible Object Model (AOM)** (DOM) - This effort aims to create a JavaScript API to allow developers to modify (and eventually explore) the accessibility tree
- Add purpose member to the Web App Manifest** (Misc) - Chrome will use the purpose member in the Web App Manifest to determine which icon to use in the status bar for not
- Allow PaymentRequest.show() to take optional detailsPromise** (Misc) - Some users may not know the total or line items at the time of attempting to open the payment sheet with show() - th
- Alternative Text in CSS Generated Content** (CSS) - Currently there is no way to supply alternative text for CSS generated content. This change would allow alternative text: The content property can be used to provide semantically important information to the user. Alternative text is used by
- AnimationEvent.pseudoElement** (CSS) - AnimationEvent.pseudoElement is a string that starts with "::", containing the name of the pseudo-element the animati
- Antialiased Clipping in Canvas** (Graphics) - The Canvas spec seems intentionally vague about the antialiasing of clips but: a) performing antialiased draws but ali

- ブラウザで開発中の機能や実装済みの機能が確認可能
- キャプチャーはChromeのもの

<https://www.chromestatus.com/features>



# 各ブラウザ向けのPlatform Status

- **Chrome Platform Status**

<https://www.chromestatus.com>

- **WebKit Feature Status**

<https://webkit.org/status/>

- **Firefox Platform Status**

<https://platform-status.mozilla.org/>

- **Microsoft Edge web platform features status**

<https://developer.microsoft.com/microsoft-edge/platform/status/>

# Can I Use

The screenshot shows the Can I Use website interface. At the top, there are navigation links: Home, News, September 3, 2019, New feature: WebXR Device API, Compare browsers (highlighted), and About. Below this is a header with the text 'Can I use' and a 'Settings' link. The main content area is titled 'Select browsers to compare' and includes a 'Reset selected browsers' button. On the left, there is a 'Categories' list with checkboxes for All, CSS, HTML5, JS, JS API, Other, Security, and SVG. The main area contains a grid of browser versions with checkboxes for selection. The browsers listed are IE, Edge, Firefox, Chrome, Safari, Opera, and iOS Safari. The versions are listed in columns: IE (6-11), Edge (12-76), Firefox (2-20), Chrome (4-24), Safari (3.1-12.1), Opera (10.0-10.1, 11.5-32), and iOS Safari (3.2-13). There are also buttons for 'Opera Mini' and 'Android Browser'.

- 特に、Compare Browsers機能が便利
- ブラウザの新バージョンリリース時はチェックしておく

<https://caniuse.com>

# 各ブラウザの次バージョンを使う

## Microsoft Edge Insider Channels

さまざまなチャンネルについて確認し、ダウンロードして、使用を開始しましょう。



### Beta Channel

6週間ごとにメジャーアップデートを実施

Beta Channel は、最も安定した Microsoft Edge プレビュー エクスペリエンスを提供します。6週間ごとにメジャーアップデートを実施し、各リリースには Dev ビルドおよび Canary ビルドから得られた知見や改善が反映されています。

ダウンロード  
macOS 向け

ダウンロード中にエラーが生じたか?



### Dev Channel

毎週更新

Dev ビルドは過去 1 週間の改善を最も良く表しています。Microsoft Edge チームによってテストされ、一般的に Canary よりも安定しています。

ダウンロード  
macOS 向け

ダウンロード中にエラーが生じたか?



### Canary Channel

毎日更新

昨日の作業内容を確認したいと思われませんか。Canary はほとんど毎晩、自動的にリリースされるので、最新の進捗状況を確認できます。

ダウンロード  
macOS 向け

ダウンロード中にエラーが生じたか?



まとめ



# まとめ

- **CSSには、  
新しい機能が増えてきている**
- **新しい機能とは、  
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、  
コンテンツのブラッシュアップに時間をかけよう**

**当たり前だと思っていた手法を見直して  
より便利にウェブ開発をしよう!**



# 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**

**@tonkotsu\_boy\_com**