

# 3. 実践編

ポートフォリオサイトを作ろうハンズオン  
@newt239

# このセクションのゴール

- HTML/CSS に慣れる
- 実用的なデザインを作れるようになる
- サイト訪問者を増やすための工夫を知る

# 目次

1. flexbox について
2. position プロパティについて
3. レスポンシブデザイン
4. SNS で注目してもらうための工夫
5. アクセス解析の導入
6. ユーザーフレンドリーなサイトを作るために

# 03-1. flexbox について

# flexbox とは

```
<div class="container">
  <div class="item">
    <h2>Work 1</h2>
    
    <p>This is description.</p>
  </div>
  <div class="item">
    <h2>Work 2</h2>
    
    <p>This is description.</p>
  </div>
  <div class="item">
    <h2>Work 3</h2>
    
    <p>This is description.</p>
  </div>
</div>
```

- 要素の配置を柔軟に行うためのレイアウトモデル
- 横並べ、縦並べ、折り返し、均等配置などを簡単に実現できる

ポートCSSで親要素に対して `display:flex;` を指定することで有効になる

# flexbox の基本

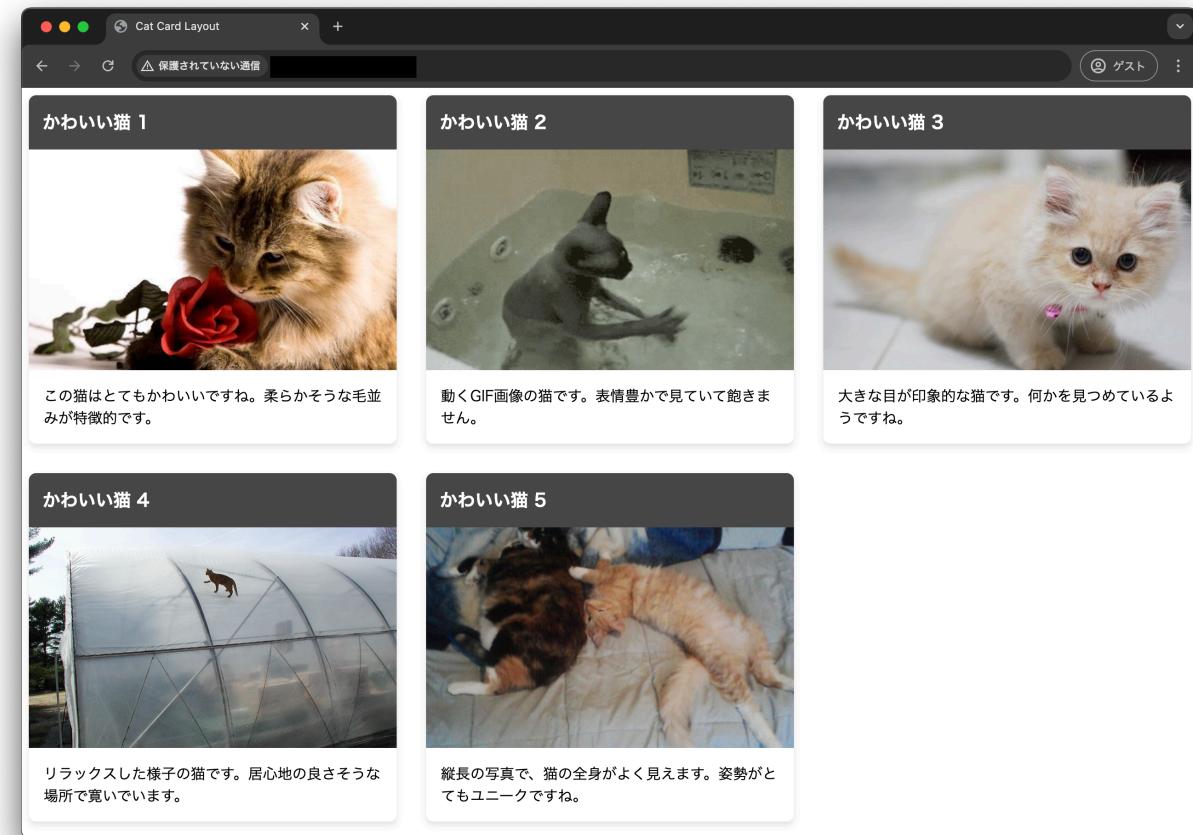
```
.container {  
  display: flex; /* flexbox を有効にする */  
  flex-wrap: wrap; /* 要素の折り返すかどうか */  
  flex-direction: row; /* 要素の並び方向 */  
  gap: 1rem; /* 要素間の間隔 */  
  justify-content: space-between; /* 横方向の配置 */  
}
```

- このあたりは良質な記事がたくさん出ているので、詳細な解説はそちらへ譲ります。
  - [日本語対応！CSS Flexbox のチートシートを作ったので配布します | Web クリエイター ボックス](#)
  - [Flexbox 入門 - 横並びを実現する定番の CSS - ICS MEDIA](#)

# 実装例：カードデザイン

コード量が多いので、以降ソースコードは Gist で公開します。

- <https://gist.github.com/newt239/82bccbf2234114c8f65f69d1f3a6a206>
- `overflow: hidden;` (12 行目)
  - 要素の内容がはみ出たときに、はみ出た部分を非表示にする
- `object-fit: cover;` (29 行目)
  - 画像のアスペクト比を保ったまま、要素に収まるように拡大・縮小



## 03-2. position プロパティについて

# position プロパティ

- 要素の配置方法を指定するプロパティ
- static (デフォルト) , relative , absolute , fixed , sticky の 5 種類がある
- ある要素を基準にして別の要素を配置したい時に利用する

# 実装例：見出しの装飾

- <https://gist.github.com/newt239/a4ad4b5565e46a2c7ef8416a0377bd47>
- `transform` プロパティ
  - `rotate` で要素を回転
  - `translate` で要素を移動 (91 行目)
  - `scale` で要素を拡大・縮小
  - `skew` で要素を傾ける (53 行目)

`position`プロパティと疑似要素を使った見出しデザイン

## デザイン1: 下線付き見出し

このデザインは、見出しの下にグラデーションの下線を配置しています。

## デザイン2: サイドバー付き見出し

このデザインは、見出しの左側にカラーバーを配置しています。

## デザイン3: 背景付き見出し

このデザインは、見出しに傾斜した背景を配置しています。

## デザイン4: 二重線付き見出し

このデザインは、見出しの上下に線を配置しています。

## ★デザイン5: 角アイコン付き見出し

このデザインは、見出しの左側にアイコンを配置しています。

# 実装例：固定されたヘッダー

- <https://gist.github.com/newt239/b7576ef08e2770392d94fbb437836630>
- ページ内リンク ([HTML 17 行目](#))
  - a タグで id を指定した要素に飛ぶ
- ページ内リンクのスクロール方法 ([1 行目](#))
  - `scroll-behavior: smooth;` でスクロールを滑らかにする
  - `scroll-margin-top` でスクロール位置を調整 (ヘッダーの高さ分ずらす)
- `z-index` プロパティ ([20 行目](#))
  - 要素の重なり順を指定。大きいほど手前に表示される

The screenshot shows a mobile browser interface. At the top, there are navigation icons (back, forward, search) and a status bar with Japanese text: '△ 保護されていた' (Protected), a guest icon, and a more options icon. Below this is a dark header bar with white text: 'ロゴ' (Logo) on the left, and 'ホーム' (Home), 'About', 'サービス' (Services), and 'お問い合わせ' (Contact) on the right. The main content area has a large title '固定ヘッダーのデモ' (Fixed Header Demo). Below it is explanatory text: 'このページは固定ヘッダーのデモンストレーションです。下にスクロールしてみてください。' (This page is a demonstration of a fixed header. Please scroll down). There is also a section titled 'スクロールしてみてください' (Please scroll) with the text: 'ページをスクロールしても、ヘッダーは常に画面上部に固定されています。' (Even if you scroll the page, the header remains fixed at the top). The bottom right corner of the screen shows the number '11'.

## 03-3. レスポンシブデザイン

# レスポンシブデザインとは

- 画面サイズに応じてデザインを変更すること
- モバイルファースト（スマホサイトを優先して作る）が主流
- CSS のメディアクエリという機能を使って実装する
- 必ず head タグ内に `<meta name="viewport" content="width=device-width, initial-scale=1.0">` を記述しておくこと

# メディアクエリ

```
/* 例：画面幅が 600px 以下の時 */
@media screen and (600px <= width) {
    /* ここにスタイルを記述 */
}
```

- `screen` : 通常の画面表示用
  - 他に印刷時用の `print` や音声読み上げ用の `speech` などがある（滅多に使わない）
- `600px <= width` : 画面幅が 600px 以下の時
  - このスコープ内に書かれたスタイルは、画面幅が 600px 以下の時のみ適用される
  - 従来は `max-width: 600px` と書く必要があったが、最近この記法が導入され、`600px <= width < 900px` のような範囲指定も可能に

# calc 関数

- CSS の計算式を記述する関数
- 画面幅に応じて要素のサイズを変化させるのに便利

```
.container {  
  width: calc(100% - 20px); /* 画面幅から 20px を引いた幅 */  
  height: calc(100vh - 4rem); /* 画面高さから 4rem を引いた高さ */  
}
```

# min 関数・max 関数・clamp 関数

- CSS でより小さい値や大きい値を指定するための関数

```
.container {  
  width: min(100%, 600px); /* 画面幅と 600px のうち小さい方 */  
  height: max(100vh, 600px); /* 画面高さと 600px のうち大きい方 */  
  font-size: clamp(  
    1rem,  
    2vw,  
    2rem  
  ); /* 1rem ~ 2rem の範囲で、画面幅に応じて変化 */  
}
```

- これらの関数の中では calc 関数のように複数の単位を組み合わせることができる
- 関数を組み合わせることもできる

# 実装例：レスポンシブなレイアウト

- <https://gist.github.com/newt239/652a82904916f690b5b0095b03a1239b>



山田 太郎

ウェブ開発歴10年のフルスタックエンジニア。HTML, CSS, JavaScriptのスペシャリスト。

## 目次

- [はじめに](#)
- [HTMLの役割](#)
- [CSSの役割](#)
- [JavaScriptの役割](#)
- [三要素の連携](#)
- [まとめ](#)

## はじめに

ウェブ開発において、HTML、CSS、JavaScriptは三位一体の要素として知られています。これら3つの技術は、それぞれ異なる役割を果たしながら、協調して動作することで、魅力的で機能的なウェブサイトを作り出します。本記事では、各要素の役割と、そ

## 03-4. SNS で注目してもらうための工夫

# OGP とは

- Open Graph Protocol の略
- Facebook が提唱したメタデータの一つ
- SNS にシェアした時に表示されるタイトル、画像、説明文などを指定できる



ポートフォリオサイトを作ろうハンズオン by newt239

# OGP の設定方法

- head タグ内に以下のタグを追加

```
<meta name="description" content="newt239's portfolio site" />
<meta property="og:title" content="newt239.dev" />
<meta property="og:type" content="website" />
<meta property="og:site_name" content="newt239.dev" />
<meta property="og:description" content="newt239's portfolio site" />
<meta property="og:image" content="https://newt239.dev/og-image.webp" />
<meta property="og:url" content="https://newt239.dev" />
<meta property="twitter:card" content="summary_large_image" />
<meta property="twitter:title" content="newt239.dev" />
<meta property="twitter:description" content="newt239's portfolio site" />
<meta property="twitter:image" content="https://newt239.dev/og-image.webp" />
<meta property="twitter:site" content="@newt239" />
<meta property="twitter:creator" content="@newt239" />
<meta property="twitter:domain" content="newt239.dev" />
```

# 各タグの意味

プロパティ	説明
description	ページの説明文。Google の検索結果でページタイトルの下に表示される文章だが、設定されていなければ Google が適当な分を抽出してくれるので、中身のない文を書くよりは設定しないほうが良いかも
og:title	ページタイトル。 <code>&lt;title&gt;</code> タグと同じものを書くことが多い
og:type	コンテンツの種類。 <code>website</code> で Web サイトのトップページ、 <code>article</code> で記事ページ等
og:site_name	サイト名
og:description	説明文。 <code>description</code> と同じ
og:image	サムネイル画像
og:url	サイトの URL

# Twitter 用の設定

プロパティ	説明
twitter:title	ページタイトル
twitter:description	説明文
twitter:image	サムネイル画像
twitter:card	Twitter カードの種類。summary か summary_large_image を指定する
twitter:site	サイトの Twitter ID
twitter:creator	作成者の Twitter ID
twitter:domain	ドメイン

# OGP 画像を作るときのコツ

- SNS によって表示される画像のサイズが異なるため、見せたい部分が切れないよう注意
  - 画像サイズを 1200px × 630px にする (Twitter の推奨サイズ)
  - 余白を十分にとり、重要な要素（テキストなど）を中心に配置
- [OGP 確認 | ラッコツールズ](#)などの OGP チェッカーを使うと良い
- Twitter の場合、一度 URL をツイートしてしまうと OGP がキャッシュされるため、変更が反映されないことがある
  - ツイートの文を書いただけでキャッシュされる
  - キャッシュされると約 1 週間は変更が反映されないので注意！
- 画像の URL は絶対パスで指定する必要がある
  - <img> タグの src 属性のようにファイル名だけの指定はできない
  - 画像が配置されるはずの URL を指定する
  - images フォルダに og-image.webp という名前で画像を配置し Github Pages で公開する場合、  
<https://ユーザー名.github.io/リポジトリ名/images/og-image.webp> と指定する

## 03-5. アクセス解析の導入

# アクセス解析とは

- Web サイトのアクセス状況を分析すること
- ユーザー数、ページビュー数、滞在時間、リファラー（どのサイトから来たか）などを把握できる
- Google Analytics が有名



# Google Analytics の導入方法

1. Google Analyticsにアクセス
2. 新しいプロパティを作成し、トラッキング ID（G- から始まるもの）を取得
3. head タグ内に以下のタグを追加

```
<script
  async
  src="https://www.googletagmanager.com/gtag/js?id=トラッキングID"
></script>
```

```
<script>
  window.dataLayer = window.dataLayer || [];
  function gtag() {
    dataLayer.push(arguments);
  }
  gtag("js", new Date());
  gtag("config", "トラッキングID");
</script>
```

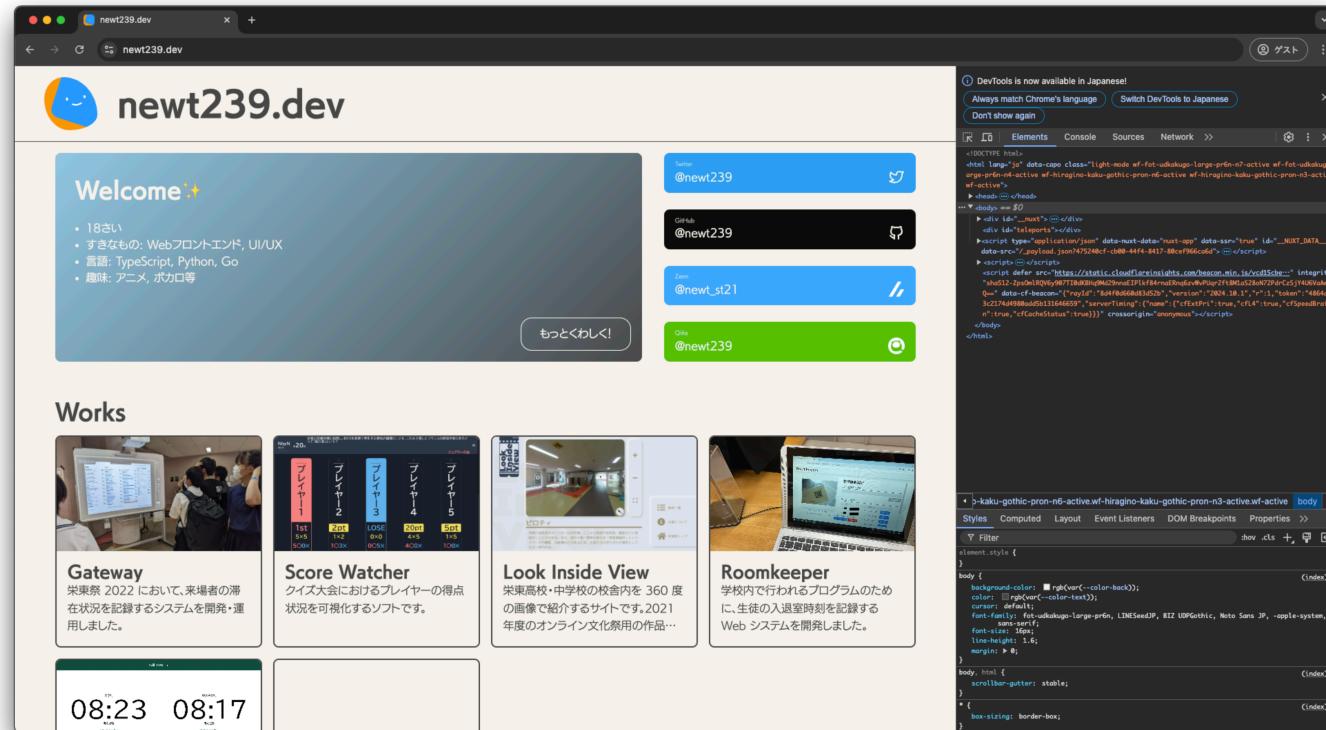
# 導入時の注意点

- トラッキング ID はサイトごとに別のものを使う
- 反映に時間がかかることがある
- 導入できたか確認するために[Google Analytics Debugger](#)という拡張機能を使うと良い

## 03-6. ユーザーフレンドリーなサイトを作 るために

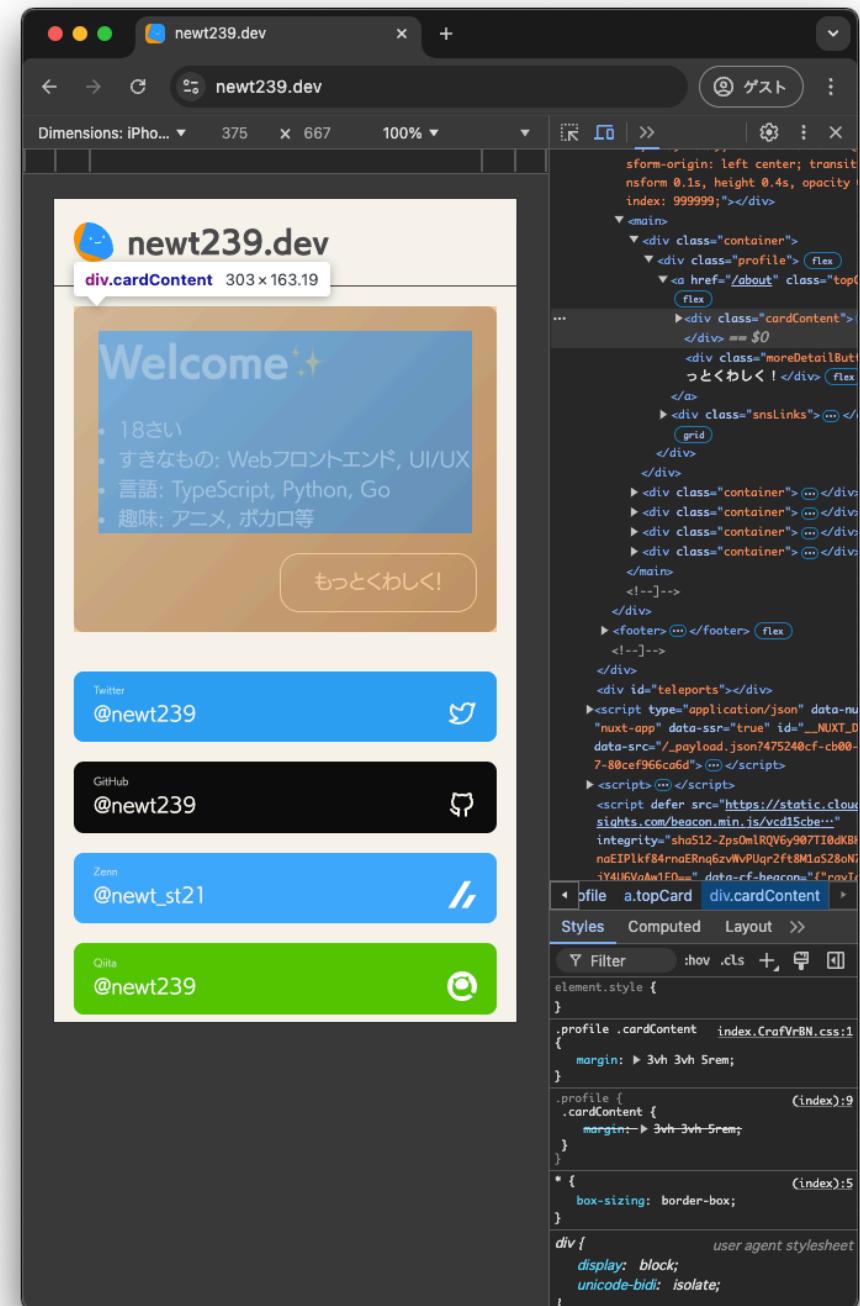
# 開発者ツール ①

- ブラウザには開発者ツールという機能が搭載されており、Web サイト のデバッグや検証に使える
- Google Chrome・Microsoft Edge の場合
  - 右クリック → 「検証」または `Ctrl + Shift + I` または `F12`



# 開発者ツール ②

- Elements タブで HTML 要素の構造を確認
- 要素を選択すると、下にその要素のスタイルが表示される
- その場で要素のスタイルを変更して検証することができる
- スマートフォンでの表示をエミュレーションすることも可能



# 画像を軽量化

- ユーザーはあらゆる環境からアクセスする可能性がある
- モバイルデータ通信を利用しているユーザーのためにサイトの読み込みに必要な帯域を減らしてあげよう
- Web サイトに表示する画像はできるだけ軽量化する
  - WebP という形式に変換
  - Google が提供している [Squoosh](#) というツールがおすすめ

# 画像には代替テキストをつける

- ・ 視覚障害者がコンテンツの内容を把握するためには有効
  - ページにおける役割だけでなく、画像がどのように見えるかを言語化して説明するとなお良い
- ・ 通信速度が遅く画像を読み込めない場合や、リンクが切れている場合にも代替テキストが表示される
- ・ Google 検索のクローラがサイトを評価する際の助けにもなる



# アクションに対するフィードバックを明確に

- リンクやボタンをホバー・クリックした際に、対象の要素に動きをつけることで、正しく反応できていることをユーザーに伝える
- :hover や :click などの疑似クラスを使って実装する

```
a:hover {  
  color: red;  
}  
button:click {  
  transform: scale(1.1);  
}
```

# アイコンを活用する

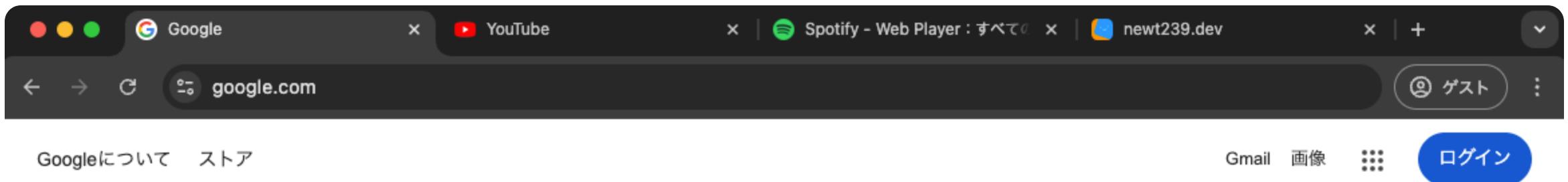
- フリーで利用できるアイコンセットが多く公開されているため、アクセントとして活用すると良い
- 大量に使う場合は CDN を利用する方法もあるが、SVG ファイルをダウンロードし `img` タグで読み込むのがラク

## 有名なアイコンセット

- [Font Awesome](#)
- [Material Icons](#)
- [Lucide](#)
- [Phosphor](#)

# ファビコンを設定する

- ブラウザのタブに表示される小さなアイコン
- OS やブラウザによって表示されるサイズが異なるため、複数のサイズのアイコンを用意する必要がある
- 以下の記事が非常に分かりやすい
  - [ずばら私の 2023 年のファビコン事情 \(SVG でダークモード対応\)](#)



第3回の内容は以上です。お疲れ様でした！