DEPART コーディングガイドライン

Table of contents:

    - 必須 CSSハック
    - 必須 大文字は使用しない
    - 必須 ブロック内のインデント
    - 必須 宣言終了のセミコロン
    - 必須 プロパティ名の直後
    - 必須 宣言ブロックの区切り
    - 必須 セレクタおよび宣言の分離
    - 必須 ルールの分離
    - 必須 CSSの引用符
- JavaScript
    - 必須 厳格モード
    - 必須 グローバル変数・関数の追加禁止
    - 必須 変数宣言のキーワード
    - 必須 カンマ区切りの変数宣言
    - 必須 比較演算子
    - 必須 動的なコード評価
    - 必須 setTimeout()、setInterval()の第1引数
    - 必須 外部データのDOM挿入
    - 必須 非推奨の機能、廃止された機能
    - 必須 セミコロン
    - 必須 コロン
    - 必須 演算子
    - 必須 ブレース（波括弧）
    - 推奨 JSDoc
- アクセシビリティ
    - 必須 構文
    - 必須 ランドマークの明示
    - 必須 セマンティクス
    - 必須 言語コード
    - 必須 テキストを含む画像
    - 推奨 SVGの代替テキスト
    - 推奨 特定の感覚に依存しない表現
    - 推奨 複数の視覚的要素を用いた表現
    - 推奨 適切なリンクテキスト
    - 推奨 適切なフォーカス順序
    - 推奨 フォーム・コントロールのラベル
    - 推奨 フォーカス時の挙動
    - 推奨 入力時の挙動
    - 推奨 テキスト情報によるエラーの特定
    - 推奨 動き、点滅、スクロールを伴うコンテンツ
    - 推奨 WAI-ARIAは必要十分かつ最小限
- SEO
    - 必須 画像の代替テキスト
    - 推奨 アンカーテキストの最適化
    - 推奨 ナビゲーションの画像
    - 推奨 別サイトからのコンテンツ引用
    - 必須 検索結果に表示したくないページ
    - 推奨 XMLサイトマップ
    - 推奨 URLの正規化（PCとモバイルが別ページの場合）
    - 推奨 URLの正規化（複数のURLでアクセスできる場合）
    - 推奨 多言語・多地域サイトの対応
    - 推奨 パンくずリスト

本ガイドラインについて

目的

本ガイドラインは、Webサイトの制作・運用に必要な基本ルールを定義したものです。
定義されたルールに準拠することでWebサイト全体の品質維持と運用効率の向上を目的とします。

各項目の準拠レベル

各項目は `必須` と `推奨` の準拠レベルに分類しています。

`必須` Webサイトの品質基準を満たすために必ず遵守すべきルール

`推奨` Webサイトの品質や運用効率を向上するために推奨されるルール

対象範囲

本ガイドラインで示している内容は、株式会社デパート（以下、当社）のWebサイト制作における標準ルールです。
本ガイドラインは、当社が制作するWebサイト全般を対象としますが、当該Webサイトに個別ルールがある場合はこの限りではありません。

内容の見直し

利用者の閲覧環境の変化、業界トレンドや制作技術の変化に対応するため、必要性に応じて随時更新します。

改訂履歴

1.0.0

2023-01-13

- DEPART コーディングガイドラインのリリース

動作環境

下記の端末・ブラウザを動作環境とする。

- Windows 10
  - Edge 最新
  - Chrome 最新
  - Firefox 最新
- Mac OS X
  - Chrome 最新
  - Safari 最新
- iOS 13 以上
  - Chrome 最新
  - Safari 最新
- Android 7 以上
  - Chrome 最新

※ 当社がサポートする標準環境であり、制作する各Webサイトの動作環境は別途定義します。

全般

必須 プロトコル

絶対パスでのリソース読み込みは可能な限りHTTPS（`https:`）を使用する。

<details>
<summary>詳細</summary>

画像やその他のメディアファイル、スタイルシート、スクリプトには、HTTPS経由で利用できない場合を除き、常にHTTPS（`https:`）を使用します。

BAD

```
<!-- プロトコル省略は非推奨 -->
<script src="//code.jquery.com/jquery-3.6.3.min.js"></script>

<!-- HTTPの使用は非推奨 -->
<script src="http://code.jquery.com/jquery-3.6.3.min.js"></script>
```

GOOD

```
<script src="https://code.jquery.com/jquery-3.6.3.min.js"></script>
```

</details>

必須 インデント

インデントは半角スペース（U+0020）2つを1単位とする。

<details>
<summary>詳細</summary>

タブ（U+0009）でインデントしたり、タブと半角スペースを混在させることはできません。

GOOD

```
<ul>
  <li>Fantastic</li>
  <li>Great</li>
</ul>
```

GOOD

```
.example {
  color: blue;
}
```

必須 行末尾の空白

行末尾の空白は削除する。

詳細

行末尾の空白は不要であり、差分がわかりにくくなるため削除します。

BAD

```
<p>行末尾に半角スペースあり</p>
```

GOOD

```
<p>行末尾に半角スペースなし</p>
```

必須 ファイル末尾の改行

ファイル末尾は改行する。

詳細

差分がわかりにくくなるためファイル末尾の行では改行します。

BAD

```
...
</html>
```

GOOD

```
...
</html>
```

必須 エンコーディング

UTF-8（BOMなし）を使用する。

詳細

エディターの文字エンコードがUTF-8（BOMなし）に設定されていることを確認します。
HTMLの場合は`<meta charset="utf-8">`でHTML文書のエンコーディングを指定します。
スタイルシートはUTF-8を想定しているためエンコーディングを指定しません。

（エンコーディングの詳細とその指定方法は、W3CのHandling character encodings in HTML and CSSを参照してください。）

必須 改行コード

LFを使用する。

詳細

エディターの改行コードがLFに設定されていることを確認します。

> NOTE
>
> Windowsの場合はGit操作時に改行コードが自動変換されないよう.gitconfigの`core.autocrlf`の設定も確認します。
>
> Git - git-config Documentation

推奨 コメント

必要に応じてコメントでコードを説明する。

詳細

基本方針としては、コメントを記述しなくても読み手が理解できるコードを書くように心がけます。
コメントを記述する必要がある場合は、そのコードが何を意図しているか、どんな目的を果たしているか、なぜその解決アプローチが使用されているかを記述します。

必須 ライセンス遵守

ライブラリやフレームワーク（CSSリセットなども含む）を利用する際はライセンスを遵守する。

<details>
<summary>詳細</summary>

ライセンスが商用利用可能であることを確認します。

商用利用が可能なライセンス

- MIT
- Apache-2.0
- BSD-3-Clause
- BSD-2-Clause
- ISC
- W3C-20150513

商用利用が不可なライセンス

- GNU LGPL v2.1
- GNU LGPL v3
- GNU GPL v2
- GNU GPL v3
- GNU AGPL v3

</details>

必須 ライセンス表記

ライブラリやフレームワーク（CSSリセットなども含む）を利用する際、ライセンス表記のコメントは削除しない。

<details>
<summary>詳細</summary>

コードをコンパイル/圧縮した際にライセンス表記のコメントが削除されていないか必ず確認します。

</details>

HTML

必須 HTMLの妥当性

可能な限り妥当性のあるHTMLを使用する。

詳細

HTMLの妥当性はW3C HTML validatorなどのツールを使用して検証します。
validatorにてエラーがあってもHTML Living Standardの仕様に準拠していれば許容します。

BAD

```
<!-- DOCTYPE宣言がない -->
<title>ページタイトル</title>
<main>メインコンテンツ <!-- 閉じタグがない -->
```

GOOD

```
<!DOCTYPE html>
<head>
<meta charset="utf-8">
<title>ページタイトル</title>
</head>
<body>
<main>メインコンテンツ</main>
</body>
```

必須 ドキュメントタイプ

HTML（HTML Living Standard）を使用する。

詳細

`<!DOCTYPE html>`で宣言し、`text/html`としてHTMLを使用します。

XHTMLは2018年3月27日に廃止されたため、XHTMLの使用は禁止します。

また、W3Cが策定していたHTML5は2021年1月28日に廃止されました。
現状はWHATWGが策定しているHTML Living StandardがHTMLの標準仕様のため、HTML Living Standardの仕様に準拠します。

BAD

```
<!-- XHTMLが使用されている -->
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

GOOD

```
<!-- HTMLが使用されている -->
<!DOCTYPE html>
```

必須 言語コード

htmlタグのlang属性にページで使用する主な言語（自然言語）の言語コードを指定する。

詳細

GOOD

```
<html lang="ja">
```

言語・地域コード一覧

- 言語コード（ISO 639-1）一覧
- 地域コード（ISO 3166-1）一覧

必須 表示領域の最適化

`meta`タグで`viewport`を指定する。

詳細

`user-scalable=no`や`minimum-scale=1`と`maximum-scale=1`の併記は表示領域の拡大ができなくなり、アクセシビリティ・ユーザビリティ上の問題があるため禁止します。

BAD

```
<meta name="viewport" content="user-scalable=no, minimum-scale=1, maximum-scale=1">
```

GOOD

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

必須 セマンティクス

目的に応じたHTML要素を使用する。

詳細

文書構造や目的に適した要素を使用してマークアップします。
例えば、見出しは`h1`-`h6`要素、段落は`p`要素、リンクは`a`要素を使用します。

BAD

```
<section>
<b>HTMLガイドライン</b>
<div>段落テキスト <span onclick="goToDetail();">ガイドラインについての詳細</span></div>
</section>
```

GOOD

```
<section>
<h2>HTMLガイドライン</h2>
<p>段落テキスト <a href="/guideline/detail.html">ガイドラインについての詳細</a></p>
</section>
```

必須 空要素の末尾

空要素は末尾スラッシュ文字なし（HTML構文）で記述する。

詳細

BAD

```
<br />
<img src="image.jpg" alt="" />
```

GOOD

```
<br>
<img src="image.jpg" alt="">
```

必須 開始タグ・終了タグの省略

開始タグ・終了タグは省略しない。（仕様上、開始タグ・終了タグが省略可能な要素でも必ず記述する）

<details>
<summary>詳細</summary>

BAD

```
<p>終了タグが省略可能な要素
<ul>
  <li>終了タグが省略可能な要素
  <li>終了タグが省略可能な要素
</ul>
```

GOOD

```
<p>終了タグが省略可能な要素</p>
<ul>
  <li>終了タグが省略可能な要素</li>
  <li>終了タグが省略可能な要素</li>
</ul>
```

</details>

必須 引用符

属性値を囲む引用符は省略せず、ダブル・クォート（`""`）を使用する。

<details>
<summary>詳細</summary>

属性値を囲む引用符は省略せずに記述します。
また、シングル・クオート（`''`）ではなくダブル・クオート（`""`）を使用します。

BAD

```
<a href=index.html class='c-button'>一覧へ戻る</a>
```

GOOD
</details>

```
<a href="index.html" class="c-button">一覧へ戻る</a>
```

必須 属性値の省略

特別な理由がない限り、真偽属性（`disabled`や`checked`など）の値は省略する。

詳細

真偽属性（`disabled`や`checked`など）の値は省略しますが、
真偽属性ではない属性（`alt`など）の値は空文字列の場合でも省略しません。

BAD

```
<!-- 真偽属性 -->
<script src="script.js" async=""></script>
<input type="text" disabled="disabled">

<!-- 真偽属性ではない -->
<img src="example.png" alt>
```

GOOD

```
<!-- 真偽属性 -->
<script src="script.js" async></script>
<input type="text" disabled>

<!-- 真偽属性ではない -->
<img src="example.png" alt="">
```

必須 パス表記

サイト内のリソースへのパス表記はルートパス（`/`から始まる表記）で記述する。

詳細

BAD

```
<!-- 相対パスになっている -->
<img src="example.png" alt="">
<img src="./example.png" alt="">
```

```
<!-- 絶対パスになっている -->
<img src="https://example.com/sabpage/example.png" alt="">
```

GOOD

```
<img src="/subpage/example.png" alt="">
```

※ランディングページなど独立したコンテンツでのみ使用するリソースの場合は、相対パスを使用しても問題ありません。

必須 文字参照

文字参照（数値文字参照 / 文字実体参照）は使用しない。

詳細

UTF-8でエンコードされたファイルであれば`&copy;`、`&#9312;`のような文字参照を使用する必要はありません。
文字参照を使用することで可読性が低下するため、文字参照は使用せず「©」「①」のように記述します。
ただし、HTMLで特別な意味を持つ文字（「<」や「&」など）は例外として文字参照を使用します。

BAD

```
<p>Copyright &copy; 2021 depart Inc. All Rights Reserved.</p>
<ul>
  <li>&#9312;リスト項目</li>
  <li>&#9313;リスト項目</li>
</ul>
```

GOOD

```
<p>Copyright © 2021 depart Inc. All Rights Reserved.</p>
<ul>
  <li>①リスト項目</li>
  <li>②リスト項目</li>
</ul>
```

必須 `type`属性

CSSファイルとJavaScriptファイル読み込み時の`type`属性は省略する。

詳細

HTMLはデフォルトで`text/css`と`text/javascript`を暗示しているため、`type`属性を明示的に指定する必要はありません。

BAD

```
<link rel="stylesheet" href="/assets/css/common.css" type="text/css">
```

GOOD

```
<link rel="stylesheet" href="/assets/css/common.css">
```

BAD

```
<script src="/assets/js/script.js" type="text/javascript"></script>
```

GOOD

```
<script src="/assets/js/script.js"></script>
```

必須 `id`属性

不要な`id`属性は使用しない。

詳細

不要な`id`属性は使用は避けて、スタイリングやスクリプトには`class`属性を使用します。

WAI-ARIAの使用などにより`id`属性が必要な場合、JavaScriptの識別子構文と一致しないよう常にハイフンを含めます。
例えば、単に`profile`や`userProfile`とするのではなく、`user-profile`と記述します。

要素に`id`属性値を使用した場合、その`id`属性値がグローバルな名前空間(`window`オブジェクト)のプロパティとして扱われ、グローバルに要素が参照できるようになります。
これにより予期しない動作が発生する可能性があるため、id属性値にハイフンを含めることでグローバルなJavaScript変数としては参照できないようにします。

BAD

```
<!-- グローバルに（`window.userProfile`で）div要素を参照できるようになるため、非推奨 -->
<div id="userProfile"></div>
```

GOOD

```
<!-- id属性が必要な場合は属性値にハイフンを含める -->
<div id="user-profile"></div>
```

推奨 `img`要素のサイズ指定

`img`要素には`width`属性と`height`属性を指定する。

詳細

画像の読み込みを待たずにレイアウトが計算されるように、`img`要素には`width`属性と`height`属性を指定します。

また、`picture`要素を使用する場合は`source`要素にも`width`属性と`height`属性を指定します。
（W3Cのバリデーターではエラーになりますが、HTML Living Standardでは仕様として定義されているため、エラーを許容します。）

GOOD

```
<img src="example.png" alt="" width="400" height="200">

<picture>
<source srcset="example_pc.png" media="(min-width: 768px)" width="600" height="300">
<img src="example_sp.jpg" alt="" width="300" height="300">
</picture>
```

推奨 画像のデコード処理 / 遅延読み込み

ファーストビューに含まれる画像には`decoding="async"`を指定する。 ファーストビューに含まれない画像には`loading="lazy"`を指定する。

詳細

画像のデコード処理によって後続のレンダリングがブロックされないよう、ファーストビューに含まれる画像は`decoding="async"`を指定してデコード処理を非同期化します。

ファーストビューの表示に必要がない画像は初期表示時のリソース取得リクエスト数を減らすため、`loading="lazy"`を指定して読み込み処理を遅延させます。

GOOD

```
<!-- ファーストビューに含まれる画像 -->
<img src="first-view-image.png" alt="" width="400" height="200" decoding="async">
```

```
<!-- ファーストビューに含まれない画像 -->
<img src="lazy-load-image.png" alt="" width="400" height="200" loading="lazy">
```

推奨 svg要素のコード最適化

svg要素から不要な属性やスタイル定義を削除する。

詳細

デザインツールから書き出したSVGコードはid属性やスタイル定義が含まれる場合があります。（書き出し設定による）
id属性の重複エラーやコードの冗長化・肥大化にもつながるため、不要なコードは削除します。

BAD

```
<!-- デザインツールから自動で付与されるid属性は不要 -->
<svg id="icon_arrow_next" xmlns="http://www.w3.org/2000/svg" width="16" height="16" viewBox="0 0 16 16">
  <!-- スタイル定義は不要 path要素などのグラフィックス要素にfill属性やstroke属性で指定する -->
  <defs>
    <style>
      .cls-1 {
        fill: #528fe2;
      }

      .cls-2 {
        fill: none;
      }
    </style>
  </defs>
  <path id="パス_2" data-name="パス 2" class="cls-1" d="M192.952,367.814l-1.414-1.414,6.585-6.585-6.585-6.585-
6.585,1.414-1.414,8,8Z" transform="translate(-187.891 -351.815)">
  <!-- 16x16のサイズ領域を確保するためにグループ化されていた見えない要素は不要 -->
  <rect id="長方形_11" data-name="長方形 11" class="cls-2" width="16" height="16">
</svg>
```

GOOD

```
<svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" viewBox="0 0 16 16">
  <path d="M192.952,367.814l-1.414-1.414,6.585-6.585-6.585-6.585-6.585,1.414-1.414,8,8Z"
transform="translate(-187.891 -351.815)" fill="#528fe2">
</svg>
```

必須 要素名や属性の表記

HTML要素名や属性は小文字のみを使用する。

詳細

HTML要素名、属性、属性値（`text/CDATA` や文字列は除く）には大文字を使用しません。

BAD

```
<A HREF="/">トップページ</A>
```

GOOD

```
<a href="/">トップページ</a>
```

推奨 終了タグ識別用のコメント

構造が読み取りにくい場合は終了タグの後ろに識別用のコメントを記述する。

詳細

開始タグに対応する終了タグがコード上で離れている場合や繰り返し同じマークアップが続く場合は、要素の構造が読み取りにくくなります。
構造の理解を助けるため、識別用のコメントを記述します。

GOOD

```
<div class="complex-structure-block">
  ...
</div><!-- /complex-structure-block -->
```

CSS

必須 CSSの妥当性

可能な限り妥当性のあるCSSを使用する。

<details>
<summary>詳細</summary>

CSSの妥当性はW3C CSS validatorなどのツールを使用して検証します。

</details>

必須 IDセレクター

IDセレクターは使用しない。

<details>
<summary>詳細</summary>

`id`属性は、ページ全体で一意であることが期待されますが、同じページに多くのコンポーネントが含まれている場合や複数のエンジニアが作業する場合は、常に`id`属性が一意であることを保証するのが困難です。
そのため、スタイリングではIDセレクターの使用は避け、クラスセレクターを使用します。

BAD

```
#title {}
```

GOOD

```
.title {}
```

</details>

必須 タイプセレクター

タイプセレクターは使用しない。

<details>
<summary>詳細</summary>

メンテナンス性が低下するため、タイプセレクターを使用したスタイリングは禁止します。
また、パフォーマンスの理由から、必要な場合以外はクラスセレクターと一緒に要素名（タイプセレクター）を使用しないようにします。

BAD

```
ul {}

div.content {}
```

GOOD

```
.list {}

.content {}
```

必須 命名規則

意味のある、または一般的なクラス名を使用する。

詳細

見た目だけを表す名前や機械的な名前ではなく、要素の目的や役割に即した名前もしくは、一般的な名前を使用します。
ただし、CSSフレームワークを使用している（既定の命名規則がある）場合や、ユーティリティ目的の調整用スタイルのクラス名（機械的に命名するのが妥当な場合）は対象外とします。

詳細は命名規則を参照してください。

BAD

```
/* 見た目だけの名前 */
.large-text {}

/* 機械的で意味のない名前 */
.v-001 {}
```

GOOD

```
/* 目的や役割に即した名前 */
.heading {}

/* 一般的な名前 */
.hero {}
```

必須 クラス名の記法

単語の連結記法はハイフン区切りのケバブケースを使用する。

詳細

理解しやすく検索しやすいようにするため、単語はハイフンで区切ります。

BAD

```
/* 単語を分けず、そのまま連結している */
.buttonprimary {}

/* アンダースコアが使用されている */
.search_result {}
```

GOOD

```
.button-primary {}

.search-result {}
```

必須 単語の略語

クラス名で使用する単語は理解可能な範囲で省略し、できるだけ簡潔にする。

詳細

クラス名はできるだけ短くしつつ、目的や役割が読み手に伝わるようにします。
「要素名や属性名などで使われる一般的な略語か」が、読み手が理解できるかどうかの1つの判断基準になります。

BAD

```
/* 長くて可読性がよくない */
.navigation {}

/* 省略しすぎて目的や役割が理解できない */
.hdg {}
```

GOOD

```
/* 短くても目的や役割が伝わる */
.nav {}

/* 目的や役割を伝えるために省略しない */
.heading {}
```

推奨 ショートハンドプロパティ

ショートハンドで記述できるプロパティは省略して記述する。

詳細

1つの値のみを明示的に設定する場合でも、効率と理解しやすさためショートハンドプロパティを使用します。
ただし、`font`や`background`など省略しないほうが分かりやすいものは例外とします。（ショートハンドでの記述も可）

BAD

```
/* border-top を消すために詳細なプロパティを個別に指定 */
border-top-style: none;

/* padding を個別に指定 */
padding-bottom: 2em;
padding-left: 1em;
padding-right: 1em;
padding-top: 0;
```

GOOD

```
/* border-top を消すスタイルを一番短く記述 */
border-top: 0;

/* padding をまとめて指定 */
padding: 0 1em 2em;

/* font: 1rem/1.6 sans-serif; と同義だが、分けて書いたほうが理解しやすいので省略しない */
font-family: sans-serif;
font-size: 1rem;
line-height: 1.6;
```

必須 0と単位

必要な場合を除き、「0」の単位指定は省略する。

詳細

GOOD

```
flex: 0px; /* flexプロパティの指定が曖昧になってしまうため、flex-basisで0pxを指定するには単位が必要 */
margin: 0;
padding: 0;
```

必須 カラーコードの16進表記

可能な限り3文字の16進表記を使用する。

▼ 詳細

より短くより簡潔にするため、3文字で表現できる場合は3文字の16進表記を使用します。

BAD

```
color: #eebbcc;
```

GOOD

```
color: #ebc;
```

必須 `!important`宣言

`!important`宣言は使用しない。

▼ 詳細

`!important`は、CSSの継承や優先度などの一般的なルールを無視してしまうため、レイアウトが崩れた場合など原因の特定が困難になります。
`!important`を使用する代わりに、セレクターの詳細度を使用してスタイルを上書きします。

ただし、ユーティリティ目的の調整用スタイル（`mt-0`で上マージンを0にする場合など）は対象外とします。

BAD

```
.example {
  font-weight: bold !important;
}
```

GOOD

```
.foo .example {
  font-weight: bold;
}

/* 調整用スタイルは !important を許容 */
.mt-0 {
  margin-top: 0 !important;
}
```

必須 ベンダープレフィックス

ベンダープレフィックスは必要最小限の範囲で記述する。

詳細

ベンダープレフィックスはプロジェクトの対象ブラウザで必要なプロパティのみを記述します。
また、ブラウザからベンダープレフィックスのサポートが消えた時のために、接頭辞なしのプロパティも必ず記述します。

※ ベンダープレフィックスは基本的に手動では設定せず、Autoprefixerで自動出力するようにします。手動で設定する場合は、Can I use...を参考に設定します。

BAD

```
/* プロジェクトの対象ブラウザにSafariが含まれている場合 */
.foo {
  /* 不要なベンダー接頭辞を記述している（-webkit- は不要） */
  -webkit-box-shadow: 1px 1px 0 rgba(0, 0, 0, 0.5);
  box-shadow: 1px 1px 0 rgba(0, 0, 0, 0.5);

  /* 必要なベンダー接頭辞を記述していない（-webkit- が必要） */
  backdrop-filter: blur(2px);
}
```

GOOD

```
/* Safariが対象ブラウザに含まれている場合の例 */
.foo {
  box-shadow: 1px 1px 0 rgba(0, 0, 0, 0.5);

  -webkit-backdrop-filter: blur(2px);
  backdrop-filter: blur(2px);
}
```

必須 CSSハック

特定のブラウザ/バージョンにのみスタイルを適用させるCSSハックは使用しない。

<details>
<summary>詳細</summary>

CSSハックを使用した条件分岐のスタイリングは、長期的には効率性・メンテナンス性の低下に繋がります。
JavaScriptを利用してclass属性値を付与するなど、別の手段での解決するようにします。

</details>

必須 大文字は使用しない

プロパティ値のコードは小文字のみを使用する。

<details>
<summary>詳細</summary>

プロパティ値（文字列を除く）には大文字を使用せず、小文字のみを使用します。

BAD

```
color: #E5E5E5;
```

GOOD

```
color: #e5e5e5;
```

</details>

必須 ブロック内のインデント

すべてのブロック内をインデントする。

<details>
<summary>詳細</summary>

階層をわかりやすくするために、すべてのブロック内をインデントします。

GOOD

```
@media screen, projection {

  html {
    background: #fff;
    color: #444;
  }
```

</details>

```
}
```

必須 宣言終了のセミコロン

すべての宣言の後にセミコロンを使用する。

詳細

一貫性と拡張性の理由から、すべての宣言をセミコロンで終了します。

BAD

```
.test {
  display: block;
  height: 100px
}
```

GOOD

```
.test {
  display: block;
  height: 100px;
}
```

必須 プロパティ名の直後

プロパティ名のコロンの後にスペースを入れる。

詳細

一貫性を保つために、プロパティと値の間には必ず半角スペース（U+0020）を1つ使用します。
プロパティとコロンの間にはスペースを入れません。

BAD

```
h3 {
  font-weight:bold;
}
```

GOOD

```
h3 {
  font-weight: bold;
}
```

必須 宣言ブロックの区切り

最後のセレクターと宣言ブロックの間にスペースを入れる。

詳細

最後のセレクターと宣言ブロックを開始する左中括弧の間には、常に1個のスペースを使用します。
左中括弧は、記述するルールの最後のセレクターと同じ行に置きます。

BAD

```
/* スペースがない */
.video{
  margin-top: 1em;
}

/* 不要な改行 */
.video
{
  margin-top: 1em;
}
```

GOOD

```
.video {
  margin-top: 1em;
}
```

必須 セレクタおよび宣言の分離

セレクタおよび宣言は改行で区切る。

詳細

それぞれのセレクタと宣言は必ず改行して新しい行に記述します。

BAD

```
a:focus, a:active {
  position: relative; top: 1px;
}
```

GOOD

```
h1,
h2,
h3 {
  font-weight: normal;
  line-height: 1.2;
}
```

必須 ルールの分離

ルールは空行で区切る。

詳細

ルールの間には必ず空白行を1つ入れます。

GOOD

```
html {
  background: #fff;
}

body {
  margin: auto;
  width: 50%;
}
```

必須 CSSの引用符

属性セレクタとプロパティ値には、二重引用符（`""`）を使用する。
また、URI値（`url()`）に引用符を使用しない。

詳細

BAD

```
html[lang='ja'] {
  font-family: 'Hiragino Kaku Gothic ProN', sans-serif;
  background: url('/img/bg.svg');
}
```

GOOD

```
html[lang="ja"] {
  font-family: "Hiragino Kaku Gothic ProN", sans-serif;
  background: url(/img/bg.svg);
}
```

JavaScript

必須 厳格モード

厳格モード（Strict mode）を使用する。

詳細

厳密なエラー解釈によるケアレスミスの防止やセキュリティリスク軽減のため、`"use strict";`宣言による厳格モードを使用します。

モジュールの場合、常に厳格モード有効のため`"use strict";`宣言は不要です。
モジュール以外の場合、複数ファイル連結時に厳格モードの影響範囲が意図せず拡大することがあるため、グローバルスコープでの`"use strict";`宣言は禁止します。

> 参考
>
> - 厳格モード - JavaScript | MDN

必須 グローバル変数・関数の追加禁止

可能な限りグローバルスコープへ変数や関数は追加しない。

詳細

グローバルスコープの変数・関数を意図せず上書きすることで、不測のエラーや依存ライブラリへ影響を及ぼす可能性があります。
JavaScriptコードは基本的に全体を包含する即時関数の中で定義するか、モジュールの中で定義します。

例外として、サイトやアプリケーション全体の名前空間を意図した変数は許容します。

> BAD

```
// グローバルスコープで変数を定義している
var global = "";

(() => {
  // ...
})();
```

> GOOD

```
(() => {
  "use strict";
```

```
  // 即時関数の中で定義
  const local = "";
})();

// サイト全体の名前空間は例外
window.MY_NAME_SPACE = {};
```

module-sample.js

```
// モジュール（module-sample.js）の中で定義
export const binding = "";
const local = "";
```

必須 変数宣言のキーワード

変数宣言には`const`もしくは`let`を使用する。

詳細

変数宣言は基本的に`const`を使用し、値を変更する必要がある場合のみ`let`を使用します。
`var`は仕様上、意図しない再代入や再宣言の可能性があるため禁止します。

BAD

```
var foo = ""; // 変数をvarで宣言している
let BRAKEPOINT = 768; // 値を変更しない変数をletで宣言している
```

GOOD

```
const items = [1, 3, 5, 7, 11];
let result = "";

for (const item of items) {
  if (item === 3) {
    result = "The world famous Nabeatsu.";
  }
}
```

必須 カンマ区切りの変数宣言

カンマ区切りの変数宣言は使用しない。

詳細

変数宣言はカンマ区切りでまとめて書かず、1つの宣言キーワードにつき変数は1つだけ宣言します。

BAD

```
let foo = "", bar = false;
```

GOOD

```
let foo = "";
let bar = false;
```

必須 比較演算子

比較は厳密な演算子（`===`、`!==`）を使用する。

詳細

等価演算子（`==`）や不等価演算子（`!=`）は異なるデータ型を暗黙的に型変換をしてから比較するため、バグを作りこむ要因となります。
`==`や`!=`ではなく、厳密な比較ができる`===`と`!==`を使用します。

BAD

```
const foo = 0;

// 数値（0）と文字列（"0"）の比較だが、結果はtrueになる
if (foo == "0") {
  console.log('equality');
}
```

GOOD

```
const foo = 0;

if (foo === 0) {
  console.log('strict equality');
}
```

必須 動的なコード評価

`eval()`、`new Function()`を使用しない。

<details>
<summary>詳細</summary>

`eval()`や`new Function()`は任意の文字列をJavaScriptとして実行できるため、セキュリティリスクとなります。

JSONのパースには`JSON.parse()`を使います。
文字列から数値の変換には`parseInt()`, `parseFloat()`, `Number()`を使います。

BAD

```
const json = `{ "users": [], "count": 0 }`;

console.log(eval("(" + json + ")"));
```

GOOD

```
const json = `{ "users": [], "count": 0 }`;

console.log(JSON.parse(json));
```

</details>

必須 `setTimeout()`、`setInterval()`の第1引数

`setTimeout()`、`setInterval()`の第1引数に文字列は使用しない。

<details>
<summary>詳細</summary>

`setTimeout()`、`setInterval()`の第1引数には関数の代わりに文字列を指定することも可能ですが、`eval()`や`new Function()`と同様にセキュリティのリスクがあります。
`setTimeout()`、`setInterval()`の第1引数には文字列ではなく関数を指定します。

BAD

```
setTimeout("console.log('Hello World!');", 500);
```

GOOD

```
setTimeout(function() {
  console.log('Hello World!');
}, 500);
```

必須 外部データのDOM挿入

外部データをDOMに追加する場合、必ずエスケープ処理をする。

詳細

外部から操作できるデータをDOMに追加する場合、悪意のあるJavaScriptが実行される可能性があります（XSS）。
以下のいずれかの対応で適切な処理をします。

- HTMLエスケープしてから追加、もしくはHTMLではなくテキストとして追加する
- スクリプトを含まない必要な要素のみを抽出し、DOMオブジェクトとして組み立てて追加する

※ DOMオブジェクトとして組み立てて追加する場合、処理が複雑になるためDOMPurify等のライブラリの使用を推奨します。

BAD

```
/*
以下のURLでアクセスしている想定
https://example.com/?keyword=%3Cimg+src%3D%22%22+onerror%3D%22alert%28%27hello%27%29%22%3E

keywordの値をURLデコードすると <img src="" onerror="alert('hello')">
*/

const root = document.querySelector("...");

const searchParams = new URLSearchParams(location.search);
const keyword = searchParams.get("keyword");

root.innerHTML = keyword; // JavaScriptが実行される
```

GOOD

```
/*
以下のURLでアクセスしている想定
https://example.com/?keyword=%3Cimg+src%3D%22%22+onerror%3D%22alert%28%27hello%27%29%22%3E

keywordの値をURLデコードすると <img src="" onerror="alert('hello')">
*/

const root = document.querySelector("...");

const searchParams = new URLSearchParams(location.search);
const keyword = searchParams.get("keyword");

// HTMLではなくテキストとして追加する
root.textContent = keyword;

// DOMオブジェクトとして組み立てて追加する（ライブラリを利用してサニタイジング）
```

```
const html = `<span>${keyword}</span>`;
root.innerHTML = DOMPurify.sanitize(html);
```

必須 非推奨の機能、廃止された機能

非推奨の機能、廃止された機能は使用しない。

詳細

古いコードとの互換性のために残された機能は将来的に廃止される可能性があるため、使用を禁止します。

非推奨の機能については以下を参照

参考

- 非推奨の機能、廃止された機能 - JavaScript | MDN
- ECMAScript 仕様書 付録 B

必須 セミコロン

文の最後は必ずセミコロン（`;`）を記述する。

詳細

Strictモードでもセミコロン（`;`）は省略可能ですが、予期せぬエラーを避けるため省略は禁止します。

また、セミコロン（`;`）の直後では必ず改行します。
（例外：`for`文の条件部分におけるセミコロン（`;`）の直後は半角スペースでも）

主にエラーになりやすい部分としては、即時関数を用いる時や、配列を変数に格納せず`[]`でそのまま使用するなどの場合に直前に文末でセミコロンを使用していないとエラーまたは予期せぬ結果になります。

BAD

```
const foo = () => {
  // ...
} // 式文の終わりにセミコロンがない
const baz = {
  checked: true,
} // 式文の終わりにセミコロンがない

function qux() {
  bar.push('d') // 文の終わりにセミコロンがない
}
if (baz.checked) { foo(); } // セミコロンの後に改行がない
```

```
const x = 10 // エラー

(function(n){
  y = n
})(100)
```

```
const x = 10

['aaa', 'bbb'].push('ccc') // エラー
```

GOOD

```
const foo = () => {
  // ...
};
const bar = ['a', 'b', 'c'];
const baz = {
  checked: true,
};

// 以下、関数宣言のようにブロックで終わる文の末尾にはセミコロン不要

function qux() {
  bar.push('d');
}

if (baz.checked) {
  foo();
}

// for文の条件ではセミコロンの後に改行しなくてもよい
for (let i = 0; i < bar.length; i++) {
  // ...
}
```

```
const x = 10;

(function(n){
  y = n
})(100);
```

```
const x = 10;

['aaa', 'bbb'].push('ccc');
```

必須 コロン

コロン（`:`）の前後に適切な半角スペースまたは改行を記述する。

表記スタイルは以下「詳細」を参照

<details>
<summary>詳細</summary>

- コロン（`:`）の前は、半角スペースを記述しない
- コロン（`:`）の後は、半角スペースを記述する

BAD

```
// コロンの後に半角スペースがない
const sample1 = {
  foo:"",
  bar:0,
};

// コロンの前に半角スペースがある
const sample2 = {
  foo : "",
  bar : 0,
};

// コロンの前に半角スペースがある
switch (type) {
case "success" :
  // ...
  break;
case "error" :
  // ...
  break;
default :
  break;
}
```

GOOD

```
const sample1 = {
  foo: "",
  bar: 0,
};

switch (type) {
case "success":
  // ...
  break;
case "error":
  // ...
  break;
default:
  break;
}
```
</details>

必須 演算子

演算子の前後に適切な半角スペースまたは改行を記述する。

表記スタイルは以下「詳細」を参照

<details>
<summary>詳細</summary>

- 演算子の前後は、改行もしくは半角スペースを記述する
  - 例外：`++`、`--`、`!`、単項マイナス演算子（数値の前にある`-`）とオペランドの間には、半角スペースを記述しない

BAD

```
// 演算子の前後に半角スペースがない
const result = 123+456;

// 単項マイナス演算子とオペランドの間には半角スペースを記述しない
let sum = - 2;

// ++とオペランドの間には半角スペースを記述しない
sum ++;
```

GOOD

```
const result = 123 + 456;

let sum = -2;

sum++;
```

</details>

必須 ブレース（波括弧）

文や関数のブレース（波括弧）は省略しない。

表記スタイルは以下「詳細」を参照

<details>
<summary>詳細</summary>

文や関数のブレース（波括弧）は（本文が1つの文であっても）省略せず、下記の表記スタイルで記述する。

- 開始括弧の前で改行しない
- 開始括弧の後で改行する
- 終了括弧の前で改行する
- 終了括弧の後で文や関数が終了する場合は改行する
- 終了括弧の後に`else`、`catch`、`while`、カンマ（`,`）、セミコロン（`;`）が続く場合は改行しない

BAD

```
// ブレース（波括弧）が省略されている
for (let i = 0; i < foo.length; i++) bar(foo[i]);

if (someCondition()) // 開始括弧の前で改行している
{
  // ...
}
else // 終了括弧の後に else があるが改行している
{
  // ...
}

// 終了括弧の後で文が終了しているが、改行していない
function bar() { /* ... */ } function baz() { /* ... */ }
```

GOOD

```
for (let i = 0; i < foo.length; i++) {
  bar(foo[i]);
}

if (someCondition()) {
  // ...
} else {
  // ...
}

function bar() {
  /* ... */
}
function baz() {
  /* ... */
}
```

推奨 JSDoc

すべてのクラス、メソッド、関数にJSDocの形式でコメントを記述する。

詳細

コードの読み手がプログラムの仕様や処理の内容を理解し、正しく使用するため、必要十分なコメントを残します。
コメントはJSDocの形式で記述し、ドキュメントを自動生成できるようにします。

GOOD

```
/** クラスの説明 */
class MyClass {
```

```
  /**
   * @param {string=} someString 引数（デフォルト引数あり）の説明
   */
  constructor(someString = 'default string') {
    /** @private @const {string} */
    this.someString_ = someString;
    /** @type {number} */
    this.someProperty = 4;

    // プロパティにすべてコメントを残すのはコストがかかるため任意
  }

  /**
   * メソッドの説明
   * @param {!MyClass} obj 引数の説明
   * @return {boolean} 戻り値の説明
   */
  someMethod(obj) {
    // ...
  }
}
```

アクセシビリティ

このページでは、アクセシビリティを確保するために最低限達成するべき「WCAG 2.1適合レベルA」のうち、基本的な内容のみ定義しています。
適合レベルAの達成を目標としますが、プロジェクトの要件や工期、デザインの実現性など、バランスをみて対応範囲を調整します。

必須 構文

仕様に準拠した、文法的に正しいHTMLを使用する。

> **WCAG 2.1 関連項目**
>
> - 4.1.1 構文解析

<details>
<summary>詳細</summary>

スクリーン・リーダーなどの支援技術が、Webページを正確に解析できるよう文法的に正しいHTMLをマークアップします。

HTMLの妥当性はW3C HTML validatorなどのツールを使用して検証します。

</details>

必須 ランドマークの明示

ページを構成する領域を適切なHTML要素でマークアップする。

> **WCAG 2.1 関連項目**
>
> - 1.3.1 情報及び関係性

<details>
<summary>詳細</summary>

スクリーン・リーダーなどの支援技術がページの構成を適切にユーザーに提示できるようにするため、`header`、`main`、`nav`、`footer`の各要素を適切にマークアップします。

BAD

```
<div class="header">
  ...
</div>
<div class="nav">
  ...
</div>
<div class="main">
  ...
</div>
```

```
<div class="footer">
  ...
</div>
```

GOOD

```
<header class="header">
  ...
</header>
<nav class="nav">
  ...
</nav>
<main class="main">
  ...
</main>
<footer class="footer">
  ...
</footer>
```

必須 セマンティクス

文書構造を適切にマークアップする。

WCAG 2.1 関連項目

- 1.3.1 情報及び関係性

詳細

見出し、強調、リスト、段落、テーブルなどのセマンティクスは適切なHTML要素でマークアップします。

全角スペースやカッコ、記号文字 (「【】」「＜＞」「・」「■」など) のテキストを視覚的な表現のために使用することはできません。視覚的な表現はスタイルシートで実装します。

BAD

```
<!-- 見出しにh1-h6が使われていない -->
<span style="font-size: 32px; margin: 21px 0;">■ 見出しテキスト</span>

<!-- br要素の連続によって段落また余白が表現されている -->
<p>1つめの段落テキスト<br><br><br>
2つめの段落テキスト</p>

<!-- リストにol、ul、dl要素が使われていない -->
<div>・親リスト項目</div>
<div>　・子リスト項目</div><!-- 全角スペースでインデントを表現している -->
<div>　・子リスト項目</div>

<!-- 見出し行と見出し列にth要素がない -->
```

```
<table>
  <tbody>
    <tr>
      <td></td>
      <td>主格</td>
      <td>所有格</td>
      <td>目的格</td>
    </tr>
    <tr>
      <th>一人称単数</th>
      <td>I</td>
      <td>my</td>
      <td>me</td>
    </tr>
    <tr>
      <th>一人称複数</th>
      <td>We</td>
      <td>our</td>
      <td>us</td>
    </tr>
  </tbody>
</table>
```

GOOD

```
<h1>見出しテキスト</h1>

<p>1つめの段落テキスト</p>
<p>2つめの段落テキスト</p>

<ul>
  <li>親リスト項目
    <ul>
      <li>子リスト項目</li>
      <li>子リスト項目</li>
    </ul>
  </li>
</ul>

<table>
    <thead>
        <tr>
            <td></td>
            <th>主格</th>
            <th>所有格</th>
            <th>目的格</th>
        </tr>
    </thead>
    <tbody>
        <tr>
            <th>一人称単数</th>
            <td>I</td>
            <td>my</td>
            <td>me</td>
        </tr>
        <tr>
            <th>一人称複数</th>
            <td>We</td>
            <td>our</td>
```

```
            <td>us</td>
        </tr>
    </tbody>
</table>
```

必須 言語コード

`html`要素の`lang`属性にページで使用する主な言語（自然言語）の言語コードを指定する。

> **WCAG 2.1 関連項目**
>
> - 3.1.1 ページの言語

詳細

多言語対応している読み上げ環境にてページを読み上げる場合、適切な言語の音声エンジンで読み上げられるよう`lang`属性に言語を指定します。

GOOD

```
<!DOCTYPE html>
<html lang="ja">
  ...
```

必須 テキストを含む画像

画像に含まれるテキストと同じ内容の代替テキストを`alt`属性に指定する。

> **WCAG 2.1 関連項目**
>
> - 1.1.1 非テキストコンテンツ

詳細

スクリーン・リーダーのユーザーが画像化されたテキストにアクセスできるよう`alt`属性に代替テキストを設定します。

GOOD

```
<img src="..." alt="代替テキスト">

<input type="image" alt="代替テキスト">
```

```
<map name="example">
  <area shape="rect" coords="..." href="..." alt="代替テキスト">
  <area shape="circle" coords="..." href="..." alt="代替テキスト">
</map>
```

推奨 SVGの代替テキスト

SVGが意味のある画像の場合、`role="img"`と`title`要素で代替テキストを提供する。

> **WCAG 2.1 関連項目**
>
> - 1.1.1 非テキストコンテンツ

詳細

SVGが意味のある画像の場合、`role="img"`で画像として認識させた上で`title`要素内に代替テキストを記述します。

GOOD

```
<svg role="img">
  <title>代替コンテンツ</title>
  <path d="...">
</svg>
```

推奨 特定の感覚に依存しない表現

形や色など特定の感覚だけを前提とした表現をしない。

> **WCAG 2.1 関連項目**
>
> - 1.3.3 感覚的な特徴

詳細

形状、色、大きさ、視覚的な位置、方向、音などが分からないと理解できないような表現は避けます。

BAD

- 「赤字の項目は必須項目です」
- 「右の表を参照してください」
- 「青いボタンを押してください」

GOOD

- 「赤い※印のついた項目は必須項目です」
- 「右の表（表3）を参照してください」
- 「青い「保存」ボタンを押してください」

推奨 複数の視覚的要素を用いた表現

文字色や図形の色で意図を表現している場合、色以外の要素と併用して表現する。

WCAG 2.1 関連項目

- 1.4.1 色の使用

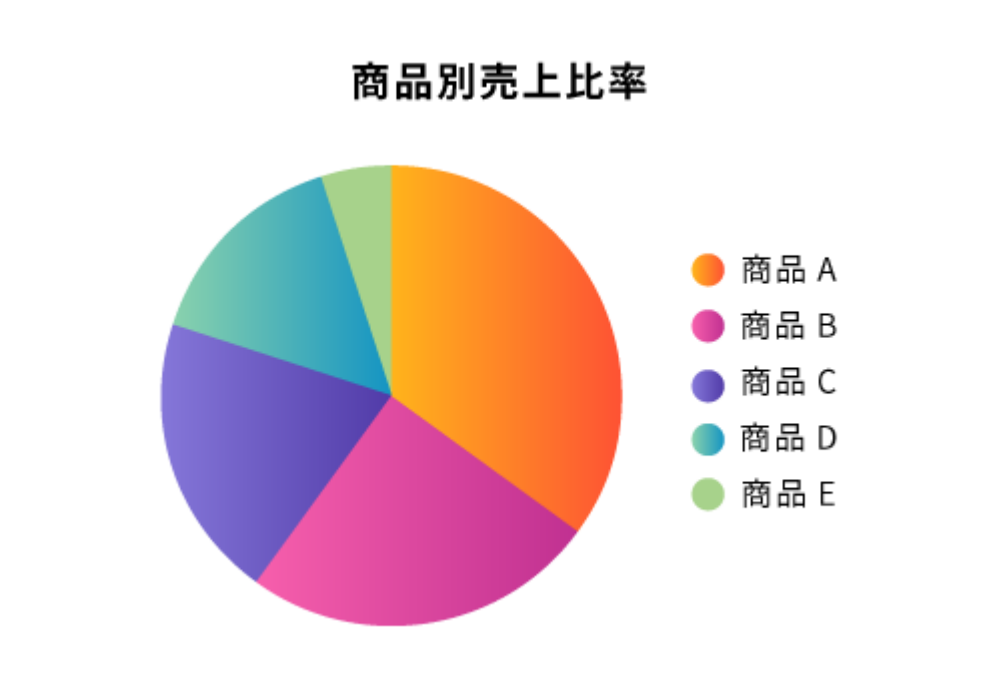
詳細

強調、引用など、何らかの意図を文字色や図形の色を変えることによって表現している場合、書体など他の視覚的な要素も併せて用い、色が判別できなくてもその意味を理解できるようにする。

リンク、グラフなどの図版、入力フォームの必須項目やエラー表示など

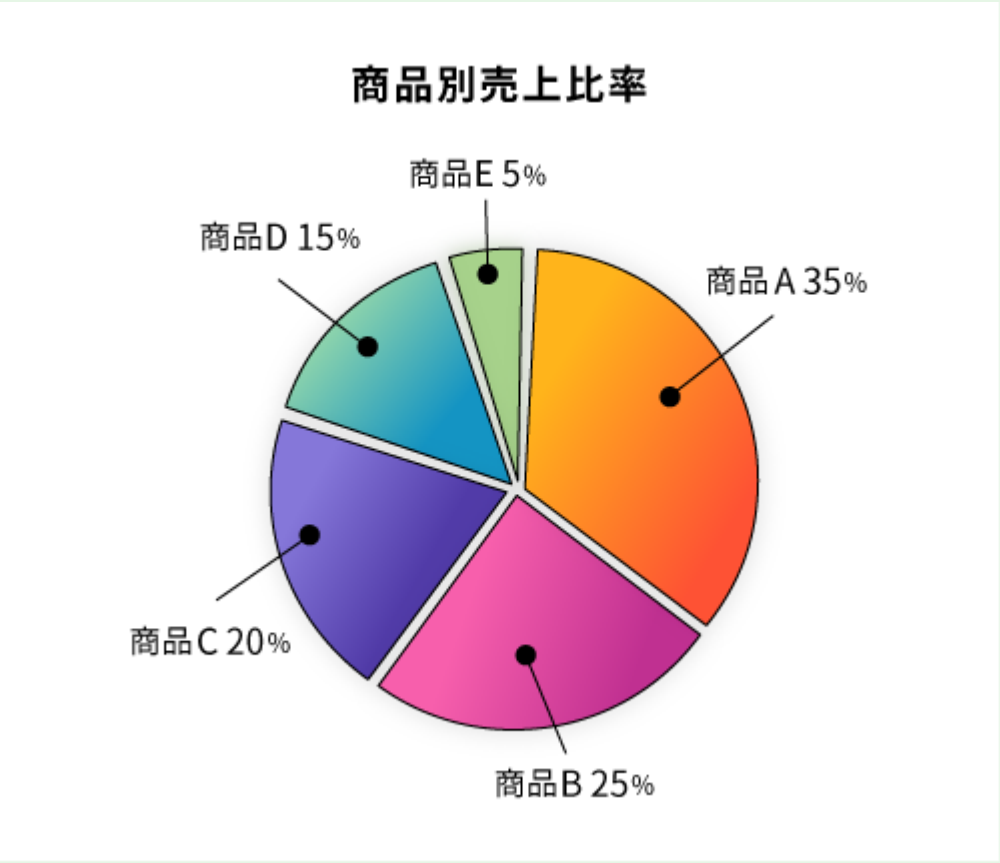
BAD

項目とデータの対応関係が色のみで表現されている

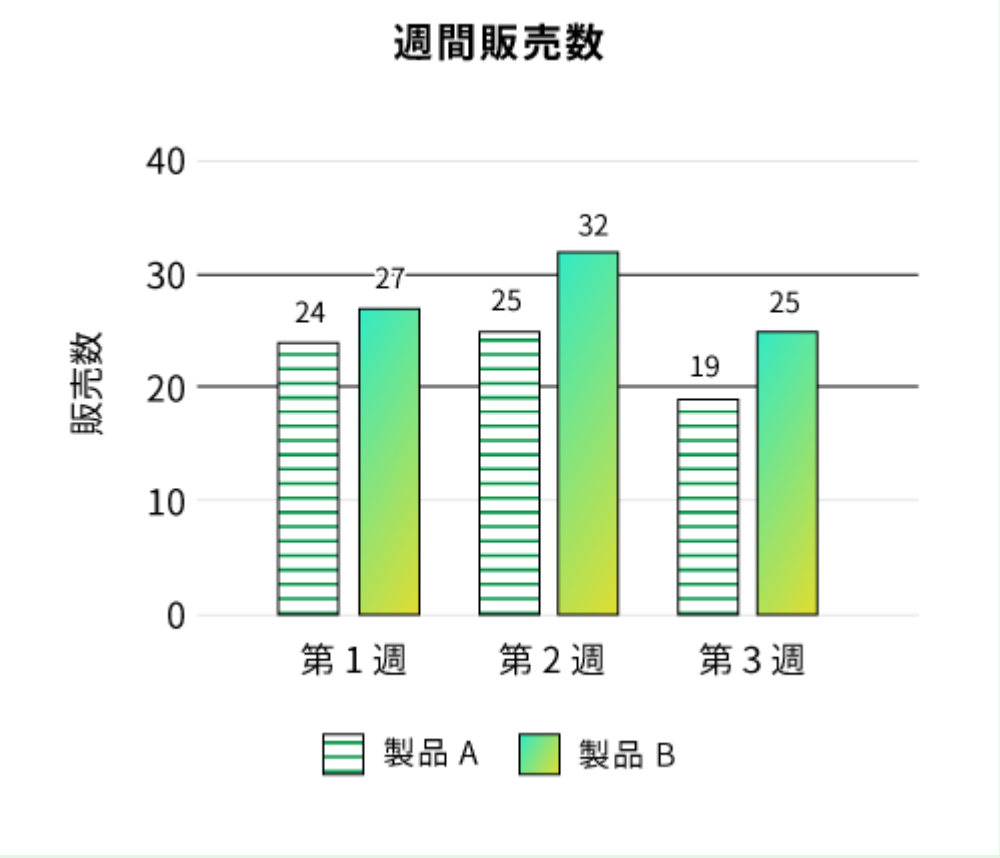


GOOD

項目とデータを続けて記載したり引き出し線を引くことで、テキストと色で情報が伝えられている



模様と色で情報が伝えられている



推奨 適切なリンクテキスト

リンクテキストは、そのリンクの目的を判断できるものにする。

> **WCAG 2.1 関連項目**
>
> - 2.4.4 リンクの目的 (コンテキスト内)

<details>
<summary>詳細</summary>

リンクテキストが「こちら」などはせず、リンクテキストの内容から遷移先をある程度推測できるようします。
`a`要素の中に画像しかない場合は必ず代替テキストを設定します。

BAD

```
<!-- 「こちら」のみリンクが設定されているため遷移先が予測できない -->
<p>おすすめ保険商品は<a href="detail.html">こちら</a></p>

<!-- a要素の中には画像しか含まれていないが、画像のalt属性値が空 -->
<a href="detail.html"><img src="..." alt=""></a>
```

GOOD

```
<p><a href="detail.html">おすすめ保険商品はこちら</a></p>

<a href="detail.html"><img src="..." alt="おすすめ保険商品"></a>
```

</details>

推奨 適切なフォーカス順序

コンテンツの意味に合った適切な順序でフォーカスを移動させる。

> **WCAG 2.1 関連項目**
>
> - 2.4.3 フォーカス順序

<details>
<summary>詳細</summary>

TabキーとShift+Tabキーでフォーカスを移動させる際、文脈やレイアウト、操作手順に即した自然な順序でフォーカスが移動するように設定します。

BAD

- 1ソースのレスポンシブデザインページにてSPとPCで要素の順序が異なる場合

- JavaScriptによるフォーカス制御や`tabindex`属性によりフォーカス順序が固定されている場合

推奨 フォーム・コントロールのラベル

フォーム・コントロールには、表示されているテキストをラベルとして明示的に関連付ける。

WCAG 2.1 関連項目

- 1.1.1 非テキストコンテンツ
- 1.3.1 情報及び関係性
- 2.4.6 見出し及びラベル
- 2.5.3 名前 (name) のラベル
- 3.3.2 ラベル又は説明

詳細

視覚障害者が、フォーム・コントロールの目的を容易に判断することができるよう、`label`要素で画面上に表示されているテキスト（または代替テキストが付加された画像）をラベルとして設定します。

該当するラベルがない場合は`title`属性により入力欄の目的を設定します。

入力例、入力形式、「必須」「任意」などの補足情報は入力欄より前（見た目上の位置、ソース上の位置）にあることが望ましいです。

BAD

```
<!-- ラベルで関連付けされていない -->
<p>氏名 <input type="text"></p>

<!-- ラベルで関連付けされていない -->
<!-- ラベルがない入力テキスト（2つめのinput要素）にtitleが設定されていない -->
<!-- 「必須」の補足情報がラベルの後にある -->
<table>
<tbody>
<tr>
<th>郵便番号 <strong>必須</strong></th>
<td><input type="text"><input type="text"></td>
</tr>
</tbody>
</table>

<!-- ラベルがない入力テキストにtitleが設定されていない -->
<select>
  <option value="japan">日本</option>
  <option value="other">日本以外</option>
</select>
```

GOOD

```
<!-- label要素で囲う -->
<p><label>氏名 <input type="text"></label></p>

<!-- label要素とinput要素が離れている場合はfor属性でidを指定する -->
<!-- ラベルがない入力テキスト（2つめのinput要素）にはtitle属性で入力欄の目的を設定する -->
<!-- label要素1つに対してinput要素が2つある場合は最初の1つと関連付ける -->
<table>
<tbody>
<tr>
<th><label for="zip"><strong>必須</strong> 郵便番号</label></th>
<td><input type="text" id="zip" title="郵便番号 冒頭3桁"><input type="text" title="郵便番号 末尾4桁"></td>
</tr>
</tbody>
</table>

<!-- ラベルがない入力テキストにはtitle属性で入力欄の目的を設定する -->
<select title="国籍">
  <option value="japan">日本</option>
  <option value="other">日本以外</option>
</select>
```

推奨 フォーカス時の挙動

Tabキーによるフォーカス移動時に、新しいウィンドウが開いたり、別のページに遷移したり、ページ全体がリロードされるなどの状況の変化を起こさない。

> **WCAG 2.1 関連項目**
>
> - 3.2.1 フォーカス時

詳細

TabキーとShift+Tabキーでフォーカスを移動させる際、以下のような変化を発生させないようにします。

- フォームの送信
- レイアウトの変更
- ページの遷移
- モーダル・ダイアログの表示
- 表示内容の大幅な変更、など

推奨 入力時の挙動

フォームに入力/選択する操作により、新しいウィンドウが開いたり、別のページに遷移したり、ページ全体がリロードされるなどの状況の変化を起こさない。

WCAG 2.1 関連項目

- 3.2.2 入力時

詳細

フォームに入力/選択する操作により、コンテンツの意味の変更、ページ全体に及ぶような変化が発生する場合は事前にユーザーに知らせる必要があります。

入力値や選択値に応じてページの一部分が書き換わる（入力欄が増えたり選択肢が変化する）場合は該当しません。
ただし、そのような可変の入力欄がある場合はソース上の位置やフォーカス順序に注意する必要があります（1.3.2 意味のある順序、2.4.3 フォーカス順序）

BAD

- 選択メニューを選択すると同時に別のページに遷移、あるいはファイルのダウンロードがはじまる場合
- フォームの全項目を入力し終えた時や最後の入力項目からフォーカスがはずれた時に自動的にページ遷移する場合

GOOD

- 選択メニューを選択してから実行ボタンを押すことにより別のページに遷移、あるいはファイルのダウンロードがはじまる

推奨 テキスト情報によるエラーの特定

入力エラーがある場合、エラー箇所とエラー内容をテキストで知らせる。

WCAG 2.1 関連項目

- 3.3.1 エラーの特定

詳細

視覚障害者、色弱者がエラー箇所を特定できるように、エラー内容が分かる具体的なテキスト情報を表示します。

BAD

- 「必須項目が入力されていません」のような具体的な項目名を示していないメッセージ
- 「エラーがあります」のような、エラーの理由（必須項目を入力していない、入力形式が違っている、桁数が違っているなど）が不明確なメッセージ
  - ※ 認証画面などでセキュリティ上の理由で具体的なエラー内容を書かないほうが良いとされる場合もあります

GOOD

- 「お名前」は必須入力項目です。

- 「パスワード」の入力形式が不正です。全角文字を使用することはできません。
- 「郵便番号 冒頭3桁」をご入力ください。
  - 入力欄が複数分かれる場合

推奨 動き、点滅、スクロールを伴うコンテンツ

同じページ上に、自動的に開始し5秒以上継続する、点滅やスクロールを伴うコンテンツと、他のコンテンツを一緒に配置しない。
5秒以内に自動的に停止しない場合は、ユーザーが一時停止、停止、または非表示にすることができるようにする。

> **WCAG 2.1 関連項目**
>
> - 2.2.2 一時停止、停止、非表示

詳細

ウィンドウ全面に表示される広告やローディング画面など、その他の情報とは同時に表示されないコンテンツの場合は該当しません。

BAD

- 自動的に情報が更新され、一時停止、停止、もしくは非表示にする手段のないニュースティッカー
- 一定時間で表示が切り替わり、一時停止、停止、もしくは非表示にする手段のないカルーセル

推奨 WAI-ARIAは必要十分かつ最小限

WAI-ARIAを利用する場合は必要十分かつ最小限に留める。

詳細

WAI-ARIAは利用方法を誤ると支援技術のユーザーに情報が伝わらなかったり操作できなくなったりするため、HTMLで表現できるセマンティクスはHTMLで表現します。

BAD

```
<span role="button" tabindex="0">ボタン</span>
```

GOOD

```
<button type="button">ボタン</button>
```

SEO

必須 画像の代替テキスト

意味を持つ画像に代替テキストを設定する。

<details>
<summary>詳細</summary>

キーワードを適切に使用して、ページのコンテンツの文脈に沿った代替テキストを設定します。
ただし、`alt`属性にキーワードを羅列すること（キーワードの乱用）は避けてください。これによって、ユーザーエクスペリエンスが低下し、サイトがスパムとみなされる場合があります。

BAD

```
<!-- alt属性がない -->
<img src="puppy.jpg"/>
```

```
<!-- キーワードの乱用 -->
<img src="puppy.jpg" alt="puppy dog baby dog pup pups puppies doggies pups litter puppies dog retriever
labrador wolfhound setter pointer puppy jack russell terrier puppies dog food cheap dogfood puppy food"/>
```

GOOD

```
<img src="puppy.jpg" alt="Dalmatian puppy playing fetch"/>
```

</details>

推奨 アンカーテキストの最適化

アンカーテキストにリンク先ページの内容を示す。

<details>
<summary>詳細</summary>

リンク先を一言で説明する内容をアンカーテキストに設定することで、検索エンジンがリンク先ページのテーマ(専門性の高さ)を把握するのに役立ちます。
キーワードに対してサイトの専門性が高いほど、ランキングで有利になります。

BAD

```
<p>おすすめ保険商品は<a href="detail.html">こちら</a></p>
```

GOOD

```
<p><a href="detail.html">おすすめ保険商品はこちら</a></p>
```

推奨 ナビゲーションの画像

可能な限りナビゲーションは画像ではなくテキストで表現する。

詳細

検索エンジンの理解を助けるため、グローバルナビゲーションなど重要なナビゲーションはテキストリンクで表現します。
テキストでの実装が難しく、画像を使用する場合は`alt`属性を設定します。

BAD

```
<nav>
  <ul>
    <li><a href="about.html"><img src="nav-about.png" alt=""></a></li>
    <li><a href="news.html"><img src="nav-news.png" alt=""></a></li>
  </ul>
</nav>
```

GOOD

```
<nav>
  <ul>
    <li><a href="about.html">私たちについて</a></li>
    <li><a href="news.html"><img src="nav-news.png" alt="お知らせ"></a></li>
  </ul>
</nav>
```

推奨 別サイトからのコンテンツ引用

別サイトからコンテンツを引用する場合、`blockquote`要素を使用する。

詳細

コンテンツコピーによるペナルティを避けるため、他ページのコンテンツ（文章、画像）を利用したい場合は`blockquote`要素を使用し、別サイトからの引用を明示します。

BAD

```
<p>Words can be like X-rays, if you use them properly—they'll go through anything. You read and you're pierced.</p>
```

GOOD

```
<blockquote cite="https://www.huxley.net/bnw/four.html">
    <p>Words can be like X-rays, if you use them properly—they'll go through anything. You read and you're pierced.</p>
</blockquote>
```

必須 検索結果に表示したくないページ

検索結果に表示したくないページは`noindex`を使用して検索インデックス登録をブロックする。

詳細

会員限定ページなど検索結果で見せたくないページは`noindex`を使用して検索インデックス登録をブロックします。

GOOD

```
<head>
...
<meta name="robots" content="noindex">
...
</head>
```

推奨 XMLサイトマップ

XMLサイトマップ（sitemap.xml）を設定し、定期的に更新する。

詳細

Sitemap.xmlはクローラーの地図になるため設定されていない場合、サイト内でクロールされないページが発生する可能性があります。
検索エンジンが効率的にクロールできるように、XMLサイトマップ（sitemap.xml）を設置します。

GOOD

sitemap.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
  <url>
    <loc>http://www.example.com/foo.html</loc>
    <lastmod>2018-06-04</lastmod>
  </url>
</urlset>
```

推奨 URLの正規化（PCとモバイルが別ページの場合）

`rel="canonical"`と`rel=alternate`を使用してURLを正規化する。

詳細

ページが重複コンテンツとして扱われることや、評価が分散してしまうことを避けるためURLを正規化します。

GOOD

PCページのURLに正規化する（検索結果に表示したいURLがPCページ）場合

https://mobile.example.com/index.html

```html
<head>
...
<!-- rel="canonical"を記載し、PCページを検索エンジンに伝える -->
<link rel="canonical" href="https://example.com/">
...
</head>
```

https://example.com/index.html

```html
<head>
...
<!-- rel="alternate"を記載し、モバイルページを検索エンジンに伝える -->
<link rel="alternate" media="only screen and (max-width: 640px)" href="https://mobile.example.com/" />
...
</head>
```

推奨 URLの正規化（複数のURLでアクセスできる場合）

301リダイレクトや`rel="canonical"`を使用してURLを正規化する。

詳細

ページが重複コンテンツとして扱われることや、評価が分散してしまうことを避けるためURLを正規化します。

URLの重複はサーバーの301リダイレクトで行うことが望ましいですが、サーバー側でURLを正規化できない場合は`rel=canonical`を使ってURLを正規化します。

複数のURLでアクセスできる例

- プロトコルによる重複
  - http://example.com/index.html
  - https://example.com/index.html
- www.有無による重複
  - https://www.example.com/index.html
  - https://example.com/index.html
- index.html有無による重複
  - https://example.com/index.html
  - https://example.com/

GOOD

```
<head>
...
<link rel="canonical" href="https://example.com/">
...
</head>
```

推奨 多言語・多地域サイトの対応

`hreflang`属性を指定した`rel="alternate"`を使用して、異なる言語・地域用のページを明示する。

※ページ内容が全く同じ場合のみ指定する

詳細

異なる言語・地域用のページがある場合は、`hreflang`属性を指定した`rel="alternate"`を使用して、異なる言語・地域用のページのURLを検索エンジンに伝えます。

GOOD

- 日本語ページ
  - `https://example.com/ja/`
- アメリカ向け英語ページ
  - `https://example.com/en-us/`

- イギリス向け英語ページ
  - `https://example.com/en-gb/`
- その他の英語圏向け英語ページ
  - `https://example.com/en/`

の場合、下記の設定で日本語圏で検索した場合は日本語ページ、アメリカで検索した場合はアメリカ向け英語ページ…が、検索結果に表示されます。

```
<head>
...
<!-- 日本語ページ -->
<link rel="alternate" href="https://example.com/ja/" hreflang="ja">
<!-- アメリカ向け英語ページ -->
<link rel="alternate" href="https://example.com/en-us/" hreflang="en-us">
<!-- イギリス向け英語ページ -->
<link rel="alternate" href="https://example.com/en-gb/" hreflang="en-gb">
<!-- その他の英語圏向け英語ページ -->
<link rel="alternate" href="https://example.com/en/" hreflang="en">
...
</head>
```

言語・地域コード一覧

- 言語コード（ISO 639-1）一覧
- 地域コード（ISO 3166-1）一覧

推奨 パンくずリスト

JSON-LDを使用して構造化データをマークアップする。

詳細

ページ上のパンくずリストは人間の目から見てパンくずリストだと判断できますが、検索エンジンには判断が難しい可能性があります。
パンくずリストを表示する場合は、パンくずリストの構造化データマークアップを使用します。

GOOD

```
<head>
  <title>Award Winners</title>
  <script type="application/ld+json">
  {
    "@context": "https://schema.org",
    "@type": "BreadcrumbList",
    "itemListElement": [{
      "@type": "ListItem",
      "position": 1,
      "name": "Books",
      "item": "https://example.com/books"
```

```
    },{
      "@type": "ListItem",
      "position": 2,
      "name": "Science Fiction",
      "item": "https://example.com/books/sciencefiction"
    },{
      "@type": "ListItem",
      "position": 3,
      "name": "Award Winners"
    }]
  }
  </script>
</head>
```

構造化データのテスト

マークアップした構造化データは、下記から検証ができます。

- Google 検索セントラル「構造化データ マークアップをテストする」

推奨 SNSシェア用の設定

SNSシェアボタンを設置したページにはOGP（Open Graph Protcol）を設定する。

▼ 詳細

SNSシェアした際に意図しない画像やテキストが抜粋されてしまうことを避けるため、OGP（Open Graph Protcol）で表示されるテキストや画像を設定します。

GOOD

```
<!-- Open Graph -->
<meta property="og:title" content="ページのタイトル">
<meta property="og:type" content="website">
<meta property="og:url" content="https://example.com/">
<meta property="og:site_name" content="サイト名">
<meta property="og:description" content="トップページの説明">
<meta property="og:locale" content="ja_JP">
<meta property="og:image" content="https://example.com/assets/img/ogimage.jpg">

<!-- Twitter固有設定 -->
<meta name="twitter:card" content="summary_large_image">
<meta name="twitter:site" content="@twitter_acount">
```

OGPの仕様

サンプルコード以外のOGPパラメータは、下記を参照してください。

- The Open Graph protocol

命名規則

命名規則リスト

CSS

CSS 命名規則

CSSでよく使うパターン

Javascript

JS 命名規則

ツール・サービス

JS 命名規則

CSS

CSS クラス名リスト

出典元:
https://github.com/manabuyasuda/coding-guidelines/blob/master/css/css-naming-list.md

> 名前をつけることは難しいですが、とても重要なことです。 CSSには設計思想が必要ですが、実践するにあたり、名前と機能の意味がとおり、名前のつけ方にブレがないようにするべきです。
>
> このドキュメントでは、CSSでよく使われる単語を分類し、意味や機能を短くまとめています。 ただし、見た目やUIの機能をクラス名にするのではなく、デザインの意図やそのコンポーネントの役割をクラス名にすることを推奨します。

上記をDEPARTルールとしてアレンジ 複数語はキャメルケース。

形容詞

名詞の性質や状態を修飾する品詞。「〜の」「〜な」。

- `main` - 主要な。
- `primary` - 主要な。
- `secondary` - 補助的な・第二の。
- `tertiary` - 第三の。
- `quaternary` - 第四の。
- `common` - 共通の。
- `global` - 全体的な。
- `local` - 局所的な。
- `general` - 一般的な。
- `brand` - ブランドの。
- `site` - サイトの。

分類

サイトのページやカテゴリとして使える名詞。

- `about` - 〜について。
- `work` - 仕事・任務。
- `product` - 製品。
- `service` - サービス。
- `news` - お知らせ・近況。
- `event` - 行事・出来事。
- `history` - 歴史・沿革。
- `archive` - 保存・保管・記録。
- `concept` - 構想・概念・考え。
- `recommend` - おすすめ・推奨。

- `toc` - 目次。
- `index` - 索引・指標。
- `contact` - 問い合わせ・連絡。
- `inquiry` - 問い合わせ・質問・調査。
- `access` - 交通手段。
- `shop` - 店舗。
- `related` - 関連のある。
- `privacy` - 個人情報の利用・保護の方針。
- `faq` - Frequently asked questions（よくある質問）の略。
- `input` - フォームの入力画面。
- `confirm` - フォームの確認画面。
- `finish` - フォームの完了画面。
- `search` - 検索結果画面。
- `cart` - 購入するアイテムを一時的に保存する画面。
- `checkout` - 保存していたアイテムを購入する画面。

Block

BEMのBlockで使われるような名詞。

- `section` - 区分・区画。
- `content` - 文書の内容。
- `article` - 記事。
- `post` - 投稿。
- `top` - 頂上・上部。
- `home` - トップページ。
- `sidebar` - 補足記事。
- `wrapper` - 内包する。
- `wrap` - `wrapper`の略語。
  - `container` - `wrapper`の類語。容器・入れ物。`wrapper`はレイアウト的に、`container`は意味的に使うことが多い。
- `group` - 集まり。
- `area` - 特定の場所・範囲。
- `emphasis` - 強調・重視。
- `catch` - 興味を惹く・関心をつかむ。
- `note` - 注釈。
- `description` - 概要。
  - `desc` - `description`の略語。
- `introduction` - 前置き・導入。
  - `intro` - `introduction`の略語。
- `announce` - お知らせ。
- `information` - 情報。
  - `info` - `information`の略語。
- `action` - Call To Action。重要度の高い。

- `more` - もっと多くの。
- `feature` - 主要なもの。
- `detail` - 詳細・細部。
- `summary` - 概要・要約。

Element

BEMのElementで使われるような名詞や形容詞など。

- `inner` - 内側の。
- `outer` - 外側の。
- `body` - 主要部分。
- `head` - 上部。
- `foot` - 下部。
- `heading` - 見出し・表題。
- `title` - 表題・題名。
- `lead` - 見出しの補足・記事の要約。
- `list` - 一覧・表。
- `menu` - 一覧・表。
- `item` - （表やデータなどの）項目。
- `unit` - 1つの単位・1セット。
- `column` - 縦列。
  - `col` - `column`の略語。
- `text` - 本文。
- `caption` - 画像・図表の補足文。
- `thumbnail` - 縮小した画像。
  - `thumb` - `thumbnail`の略語。
- `avatar` - 人の顔を示す画像。
- `feature` - 特徴を補足する画像。
- `tel` - 電話番号。
- `address` - 住所。
- `date` - 日付。
- `time` - 日時。
- `category` - 分類・部類。
  - `cat` - `category`の略語。
- `tag` - 識別子。
- `next` - 次の。
- `previous` - 前の。
  - `prev` - `previous`の略語。
- `mask` - 覆い隠す。
- `overlay` - かぶせる・上書きする。
- `delimiter` - アイテムの範囲や境界線を示すインターフェイス。
  - `separator` - `delimiter`の類語で混ぜないように分離する目的で使います。

    - `divider` - `delimiter`の類語でグルーピングするように分割する目的で使います。

Modifier

BEMのModifierで使われるような名詞や形容詞など。

- `success` - 成功。
- `alert` - 注意・警戒。
- `attention` - 配慮・気配り。
- `error` - 間違い・失敗。
- `caution` - 用心・警告。
- `warning` - 警告・予告。
- `danger` - 危険・驚異。
- `tiny` - とても小さい。
    - `xs` - `tiny`の類語でExtra small（smallよりさらに小さい）こと。
- `small` - 小さい。
- `medium` - 中間。
- `large` - 大きい。
- `huge` - とても大きい。
    - `xl` - `huge`の類語でExtra large（特大・超大型）のこと。
- `reverse` - 反転する。
- `push` - 前方に押す。
- `pull` - 自分の方に引く。
- `offset` - 相殺する・埋めあわせる。
- `left` - 左側の。
- `center` - 真ん中。
- `right` - 右側の。
- `top` - 上部。
- `middle` - 真ん中。
- `bottom` - 下部。
- `round` - 角丸。
- `circle` - 円形。

State

状態を示す動詞や形容詞など。 is-xxxxで使うことが多い。

- `show` - 見せる。
- `hide` - 隠す。
- `open` - 開く。
- `close` - 閉じる。
- `current` - 現在の。
- `active` - 活動中・有効な。
- `disabled` - 無効になっている。

UI

利用者に情報を提供するためのインターフェイス。

Text

- `link` - アンカーテキスト。検索する
- `label` - 分類する・項目名。検索する
- `tag` - 符号・識別子。検索する
- `badge` - 残数を示す数値。検索する
- `copyright` - 著作権表示。検索する
- `tooltip` - マウスオーバー時に補足情報を表示するインターフェイス。検索する
- `button` - オン・オフの選択に使うインターフェイス。検索する
  - `btn` - `button`の略語。

Image

- `image` - 画像。検索する
  - `img` - `image`の略語。
- `icon` - 対象の内容や機能などを小さな絵柄で表したもの。検索する
- `loading` - 読み込み中であることを示すインターフェイス。検索する
- `logo` - 社名や製品名などを図案化・装飾化した文字・文字列。検索する
- `map` - 地図。検索する
- `chart` - 理解しやすいような方法で情報を示すリストやグラフのこと。検索する
  - `graph` - `chart`の類語で視覚表現のための手段のこと。検索する
- `hero` - キービジュアルになる大型の画像。検索する
- `banner` - （主に宣伝用の）画像をともなうリンク。検索する
- `carousel` - 画像などのコンテンツを上下左右にスライドさせて切り替えるインターフェイス。検索する
  - `slider` - `carousel`の類語。検索する
  - `ticker` - `carousel`の類語で自動でアイテムを左右に流しながら表示する。ユーザーは基本的にコントロールできない。検索する
- `modal` - モーダル。主な用途はサムネイル画像をクリックしてモーダルを開き画像を大きく見せること。
  - `lightbox` - モーダルのjQueryプラグインのこと。

Navigation

- `navigation` - 情報へ誘導するリンク。検索する
  - `nav` - `navigation`の略語。
  - `globalNavigation` - ほとんどの画面で表示されている主要なナビゲーション。検索する
  - `localNavigation` - あるカテゴリ内専用のナビゲーション。グローバルナビゲーションの中や下に配置する場合や、サイドバーに配置する場合がある。検索する
- `megaMenu` - 深い階層のカテゴリやページへのナビゲーション。ドロップダウン（クリックやマウスオーバー）で階層を表示していく。カテゴリやページ数の多いサイトのグローバルナビゲーションに使われることが多い。検索する
- `breadcrumb` - パンくずリスト。トップページから現在ページまでの階層構造を示したリンク。検索する
  - `topicpath` - `breadcrumb`の類語。検索する
- `springboard` - 同じサイズのリンクを2×2や3×3のように配置した同じ重要度を持つ並列なナビゲーション。検索する
- `listMenu` - 縦に並んだリスト型のリンクで、階層構造をもったナビゲーション。検索する
- `dashboard` - グラフやステータスなどを一つの画面にまとめたインターフェイス。検索する

- `pagination` - 昇順にしたページ番号付きのナビゲーション。検索する
- `linearNavigation` - その画面の前後に移動するためのナビゲーション。`pagination`との違いはページ指定ができないこと。検索する
- `backToTop` - ページトップに戻るためのページ内リンク。検索する
- `tab` - 書類などのインデックスタブを模した、画面内のコンテンツ表示を切り替えるインターフェイス。検索する
  - `tabbar` - おもに画面下部に固定されたタブで、画面全体を切り替えるインターフェイス。検索する
  - `segmentedControl` - 機能的には`tab`と同じで、見た目がタブではなくボタンである点が違う。検索する
- `off-canvas` - 表示領域外から画面の大半を覆い隠したり（オーバーレイ）、ずらすようにスライドしながら表示するナビゲーション。検索する
  - `sideDrawer` - `offCanvas`の類語。drawerは「引き出し」の意味。検索する
- `toggleMenu` - 同一の操作で二つの状態を交互に切り換えるインターフェイスで、操作対象になるボタンを基準にナビゲーションを表示することが多い。検索する
- `sitemap` - サイト内のすべてのページへのリンクをリスト化したもの。検索する
- `sns` - ソーシャルネットワーキングサービス。検索する

Form

- `form` - 送信フォーム。
- `login` - ユーザー認証をするためのフォーム。検索する
  - `signin` - `login`の類語。検索する
  - `logout` - `signout` - ユーザー認証を解除すること。検索する
- `registration` - ユーザー登録をするためのフォーム。検索する
  - `signup` - `registration`の類語。検索する
- `stepNavigation` - 複数画面にわたるフォームの順序や、現在地を示したナビゲーション。検索する
- `searchBox` - キーワード検索するためのフォームエリア。検索する
- `textField` - `input[type="text"]`のような利用者が編集する改行なしのテキストフィールド。検索する
- `textarea` - `textarea`のような利用者が編集する複数行のテキストフィールド。検索する
- `checkbox` - `input[type="checkbox"]`のような1つ以上の項目を選択するインターフェイス。検索する
- `radio` - `input[type="radio"]`のような1つの項目を選択するインターフェイス。検索する
- `select` - `select`のような1つの項目を選択するインターフェイス。ラジオボタンと違い、（`dropdown`のように）項目が最初は隠れているのが特徴。検索する
- `submit` - 送信ボタン。検索する
- `reset` - リセット（取り消し）ボタン。検索する
- `modify` - 修正ボタン。検索する

Other

- `cards` - トランプのような積み重なったカードのメタファーをもつインターフェイス。検索する
- `dropdown` - 複数の項目を表示して、1つの項目を選択するインターフェイス。検索する
  - `pulldown` - `dropdown`の類語。検索する
- `accordion` - 折りたたまれたコンテンツを選択することで表示させるインターフェイス。検索する
- `scrollTab` - 表示領域よりも横に長いナビゲーションで、左右にスクロールすることで非表示部分を見ることができるインターフェイス。検索する
- `dialog` - （主に）注意や警告を知らせるために使用されるメッセージで、ユーザーに操作を要求するのが特徴。検索する
  - `toast` - `dialog`の類語で、`dialog`との違いは時間が経つと自動的に消えること。検索する
  - `popupWindow` - `dialog`の類語で、リンク先を別の小さなウィンドウで表示するインターフェイス。検索する

- `toolbar` - ユーザーが利用できるツールやアクションをまとめたインターフェイス。検索する
- `comment` - 記事に対する反応。検索する
- `table` - テーブル・図表。検索する
- `timeline` - チャットや年表のように時系列に並べたリスト。検索する

jsonファイル

css.json

エディタのスニペットなどや辞書登録しやすくなればと思ってjsonファイルも用意しました。

CSSでよく使うパターン

案件などの制作でよく使うパターン

- `cardLists` - カードコンポーネントを囲う部分が`cards`では表現的に乏しい場合など。
- `aboutBlock` - `aboutSection`どうしても`about`などコンテンツ名だけを使えないがブロックを現したい場合は、後ろに`Block`や`section`を付け足すと良い。
- `serviceFlow` - `serviceStep` フローなのかステップなのかは意味合い的には大差ない。
- `feature` - 機能の紹介。Google翻訳すると`Introduction of functions`だが、近しい意味の特徴に変えて、`feature`とする方がよかったりする。
- `toc` - 目次。 `Table of Contents`だが、略語を用いるケースもある。ただし、一般的な略語のみにしないと意味がわからなくなるので注意。

実際の案件から抜粋

- `commentArea`, `inputArea` - 要素名などに`Area`を足してそれを行う場所を意味させる方法。`Block`や`Section`を使うケースも多い。
- `communityArticle`, `featureDetail` - コンテンツの詳細ページを表す場合に`Article`や`Detail`で表す場合もある。
- `keyVisual`, `cartStep` - 一つの単語で説明が難しいもの、複数単語で成り立つもの。
- `postSlider`, `shareLink` - `Area`などよりも詳細度が高い一部分に対してつけることでネストの深さを緩和したり、他のCSSの記述に影響させないようにする。
- `modalBlock`, `modalInner`, `modalWrap` - modalのようなUIパーツの構造を表す場合。
- `inputCheckToggle` - 三つの単語で機能まで名前に含める場合。

フレームワークでのコンポーネントに関して

コンポーネントベースのフレームワークを利用する場合には、ほとんどの場合CSS Modulesとしての設定がされるため、内部的なBlockが複数語ではなくても可能です。 その場合においても最初にコンポーネント名を使用するようにし、予期せぬ衝突を避けるようにしてください。

componentsNameコンポーネント内

```
<div class="componentsName">
  <ul class="componentsName__lists">
    <li class="componentsName__list"></li>
  </ul>
</div>
```

```
// componentsName.scss
.componentsName{
  &__lists{
  }
  &__list{
  }
}
```

または下記も可能

```
<div class="componentsName">
  <ul class="lists">
    <li class="list"></li>
  </ul>
</div>
```

```
// componentsName.scss
.componentsName{
  .lists{
  }
  .list{
  }
}
```

Javascript

Javascript 命名リスト

記法形式

記法	記法名	例
コンポーネント名	パスカルケース	UserForm
変数名	キャメルケース	sampleFunction
定数名	スネークケース	API_URL
メソッド名	キャメルケース	addNumber
プロパティ名	キャメルケース	userName
クラス名	パスカルケース	MyCar
型	パスカルケース	MyType

その他はフレームワークやライブラリの方針に沿った形での命名規則とする。

命名で使われる単語リストと意味

※更新予定

下記外部記事などを参考にしてください。

https://qiita.com/KeithYokoma/items/2193cf79ba76563e3db6 https://php-archive.net/php/words-in-function-names/ https://arakan-pgm-ai.hatenablog.com/entry/2019/04/15/000000

ツール・サービス

お役立ちツール・サービス

命名にどうしても迷う場合のツール

翻訳

codic
プログラミングで使われそうな単語に翻訳してくれます。

辞書

プログラミング英語検定
プログラマーに求められる英語力を高めるというサイトで、600単語以上が掲載されています。