Engineer

ウェブコンポーネント

🖼 再利用できる独自のHTML要素を定義するブラウザー機能

どのようなものか

ウェブコンポーネントは、英語ではweb componentsや大文字を使ったWeb Componentsと書き、制作者が独自のHTML要素を定義できるブラウザー機能群の一般的な呼び名です。ページでは、その要素を他のタグと同じように使います。標準は各部分をそれぞれの名前で扱います。HTML Standardのカスタム要素、`template`要素と`slot`要素、DOM Standardのシャドーツリーです。総称は2014年7月24日付のW3C作業部会ノート*Introduction to Web Components*の題名にもなっていますが、W3Cはその後この文書の維持を打ち切り、参照すべきでない文書としています。

カスタム要素は、`HTMLElement`を継承するJavaScriptクラスで定義し、`customElements.define()`で名前を登録します。その定義はHTMLパーサーに要素の構築方法と変更への反応を伝えます。それ以降は、マークアップ、`document.createElement()`、`new`で要素を作れます。名前は小文字のASCII英字で始まり、大文字のASCII英字を含まず、ハイフンを含まなければなりません。ハイフンは独自の名前を言語自体から区別します。HTML Standardは、HTML、SVG、MathMLにはハイフンを含む名前の要素を今後追加しないと定めています。

種類は2つあります。標準自体の例である`<flag-icon country="nl">`のような自律型カスタム要素は、組み込みの意味を持たない通常の要素から始めます。カスタマイズされた組み込み要素は、既存の要素を拡張します。`HTMLButtonElement`を継承するクラスを`{ extends: "button" }`で登録し、`<button is="plastic-button">`として使うと、フォーカスの動作、フォーム送信、`disabled`属性など、ボタンをボタンとして成り立たせるすべての機能を保てます。

シャドーツリーは、ホストと呼ばれる要素に取り付ける独立したノードの木です。DOM Standardはホスト側の通常の木をライトツリーと呼びます。`attachShadow()`でopenまたはclosedモードのシャドールートを作り、ホストの

`shadowRoot` プロパティーがそれを返すのはopenモードだけです。シャドーツリー内の `slot` 要素は、ホスト自身の子を表示する位置を示します。割り当てられたノードを表示し、なければ自分自身の内容を表示します。名前付きスロットは、 `slot` 属性にその名前を持つ子を受け取ります。 `template` 要素は、表示されず、スクリプトで複製できるHTML断片を保持します。 `shadowrootmode` 属性を持つ `template` は宣言的シャドールートを表すため、シャドーツリーをマークアップへ直接書くこともできます。

概念を判断に生かす

「ウェブコンポーネントとは？」を実際の判断に生かしましょう。その概念を、開発または購入するシステムに結び付けて考えるお手伝いをします。 読み込みが速く、スマートフォンでも使え、お客様がどのブラウザーを使っていても同じように動く画面を提供します。

サービスを見る お問い合わせ →

用語の背後にある前提を検証する

解説とリンク先の実績を参考に、プロジェクトについての質問を整理してください。ツールを選ぶ前に、前提を一緒に検討できます。 フルスタックのGIS地図開発を指揮 — DBからブラウザーまでのフルスタック地図開発

フルスタックのGIS地図開発を指揮 お問い合わせ →

なぜ重要なのか

以前は独自に作ったタグに用途はほとんどありませんでした。HTML Standardも、そのような要素を無効で実用性に乏しかったものとして振り返っています。カスタム要素は文書の正式な一員です。パーサーが構築し、DOMメソッドが作成し、属性が変わったときやページへ出入りするときの動作をクラスが決めます。

定義が先に到着している必要はありません。パーサーが既に作った要素も、その名前の定義が登録されるとカスタム要素になります。HTML Standardはこれをアップグレードと呼びます。それまでは `span` に似た未定義要素として扱われ、中の文字とリンクはそのまま表示されます。標準は、アップグレードによってカスタム要素のコンテンツを段階的に拡張できると明記しています。

カスタム要素はフォームやアクセシビリティツリーにも参加できます。クラスで `static formAssociated = true` を設定して `attachInternals()` を呼ぶと、 `ElementInternals` オブジェクトが得られます。その `setFormValue()` で要素の値を設定すると、フォームは他のコントロールと同じように送信します。同じオブジェクトでARIAの既定ロールと既定状態を設定でき、ページの制作者が `role` 属性や対応する `aria-*` 属性で上書きしない限り、それらが適用されます。また、スタイルシートの `:state()` 疑似クラスで照合する独自状態も保持できます。

チームにとって、部品を自己完結させるのはシャドーツリーです。ページの通常のセレクターは内部のシャドーツリーへ入り込まず、外側から装飾できる部分はコンポーネント自身が決めます。ただし継承値とカスタムプロパティーは内部へ届き、スロットへ配置したライトツリーの内容にはページのセレクターが引き続き適用されます。複数のライブラリーがページを共有する場合、HTML Standard は `new CustomElementRegistry()` で作るスコープ付きレジストリーを用意しています。これにより、ライブラリー同士で名前を調整しなくても、それぞれの要素を共存させられます。

実際の活用例

商品カードを例にすると、各部分の連携が分かります。ページはコンテンツを通常のマークアップとして持ち、名前付きスロットに入れる要素へ `slot` 属性を付けます。

```
<product-card>
  <span slot="name">Oak stool</span>
  <a href="/shop/oak-stool/">See the details</a>
</product-card>
<script type="module" src="product-card.mjs"></script>
```

モジュールが要素を定義します。テンプレートは1回だけ作られ、各カードのopenモードのシャドールート内へ、それぞれ複製を配置します。

```
const template = document.createElement("template");
template.innerHTML = `
  <h2 part="title"><slot name="name">Unnamed product</slot></h2>
  <slot></slot>
`;

class ProductCard extends HTMLElement {
  constructor() {
    super();
    const root = this.attachShadow({ mode: "open" });
    root.append(template.content.cloneNode(true));
  }
}

if (!customElements.get("product-card")) {
  customElements.define("product-card", ProductCard);
}
```

ページは、コンポーネントが公開する1つのパーツを通じて題名を装飾します。

```
product-card::part(title) {
  font-size: 1.25rem;
}
```

`product-card.mjs` が実行される前、カードは未定義要素で、読者には商品名とリンクが通常のインラインコンテンツとして見えます。モジュールが動くと `define()` がクラスを登録し、既にページ内にあるカードがアップグレードされます。コンストラクターはシャドールートを取り付け、テンプレートの内容の複製を入れます。 `span` は `name` というスロットへ割り当てられ、見出しの中に現れます。リンクには `slot` 属性がないため、名前のない既定スロットに入ります。名前のないカードには代わりに、スロット自身の内容である「Unnamed product」（名前のない商品）が表示されます。 `customElements.get()` による確認は、別のバンドルから同じモジュールの2つ目のコピーが届いたときに、 `define()` が例外を投げるのを防ぎます。

共有部品は、使われて初めて役に立ちます。このサイトのポートフォリオにも、その事例があります。iOS 26の「Liquid Glass」デザインシステムは、新しい画面ごとに少しずつ違うボタン、バッジ、カードを作り直すUIが出発点です。その結果、変更のたびに5つの個別実装を触る状態になっていました。記録された解決策は、すべての画面で組み合わせて使う正規の部品群です。Card、Label、Button、GlassIconButton、DirectoryGridなどを揃え、手作りのバッジやチップを拒否して正規部品を案内するlint規則も設けました。そのページは部品がカスタム要素かどうかを述べていません。ただし、部品ごとの定義を1つにし、レビュー担当者の記憶でなくツールで徹底する方針は共通です。カスタム要素レジストリーは、同じ名前の2つ目の定義を拒否することで前半を守ります。

問題が起きるところ

ハイフン付きの名前でロールを代用する。 HTML Standardは、「自律型カスタム要素には特別な意味はなく、その子を表す」と説明しています。例の `taco-button` は必要な作業を示します。ボタンとして動くには、 `tabindex` 、ARIAロール `button` 、アクセシブルな名前、 `aria-disabled` 、EnterとSpaceをクリックに変えるキーハンドラー、無効状態のスタイルが必要です。それでもカスタム要素は既定のフォーカス動作を宣言できず、 `tabindex="0"` がそのプラットフォームのボタン動作に一致しないこともあります。既存の要素を基にするなら、標準は組み込み要素の拡張によって、制作者と利用者の双方の負担を減らせると説明しています。自律型要素では、各ページへ設定を任せるのではなく、 `ElementInternals` で既定ロールと状態を設定します。

コンストラクターですべてを処理する。 HTML Standardはカスタム要素のコンストラクターに規則を定めています。引数なしの `super()` を最初に呼びます。アップグレードではなく新規作成された要素には属性も子もないため、それらを調べてはいけません。また、要素自身に属性や子を追加してはいけません。この規則を破ると、パーサーやDOMメソッドで作れない要素になる場合があります。コンストラクターでは初期状態、既定値、イベントリスナー、シャドールートを用意し、データ取得と描画は `connectedCallback` へ移します。

`connectedCallback` **が1回しか動かないと思い込む。** 複数回動く場合があります。既定では、要素を移動すると `disconnectedCallback` の後に `connectedCallback` が動き、削除して挿入し直したかのように状態を初期化します。1回だけ必要な準備はフラグで守り、 `disconnectedCallback` では、 `connectedCallback` が開始した処理を元に戻します。 `moveBefore()` による移動で状態を保ちたい場合は、 `connectedMoveCallback` を定義します。本体が空でも、存在するだけで既定の2つのコールバックに代わります。

同じ名前を2回定義する。 `customElements.define()` は、名前またはクラスが登録済みなら `NotSupportedError` を投げ、ハイフンがない、大文字を含むなど無効な名前なら `SyntaxError` を投げます。定義前に `customElements.get()` で確認し、他と共存する必要のあるライブラリーには専用のスコープ付きレジストリーを与えます。

closedモードを保護機能と考える。 closedモードはホストの `shadowRoot` プロパティーからルートを隠します。また、そのclosedモードのシャドーツリーの外にあるリスナーから見たイベントの合成パスでは、内部のノードを隠します。CSS Shadow Partsは、シャドーツリーは意図的にセキュリティ境界ではなく、ページ制作者の便宜のためだけのものだと明記しています。部品に秘密を入れず、closedモードは施錠手段としてではなく、外側のコードへの影響で選びます。

使えない要素で `attachShadow()` **を呼ぶ。** DOM Standardがシャドールートを認めるのは、有効なカスタム要素名を持つ要素と、限られた組み込み要素だけです。後者には `div` 、 `section` 、 `p` 、 `span` などが含まれますが、 `button` は含まれません。カスタマイズされた組み込みボタンで呼ぶと、 `NotSupportedError` になります。既にルートを持つ要素でも動作が異なります。そのルートが宣言的で、同じモードなら、呼び出しは中身をすべて削除し、空になったルートを返します。それ以外は例外になります。先に既存のシャドールートを確認し、サーバーから宣言的ルートとして送ったマークアップを消さずに保ちます。

ページのスタイルが内側まで届くと思い込む。 CSS Shadow Partsはこの境界を装飾上の障壁と呼び、ページ側で装飾したい場合には部品の装飾を難しくするとも指摘しています。ページが装飾してよい要素を `part` で公開し、外側から `::part()` で装飾します。部品の中に別の部品がある場合、そのパーツをページへ渡すには、内側のホストに `exportparts` が必要です。色や間隔をページに合わせるには、ホストへ設定したカスタムプロパティーを読み取ります。継承によってシャドーツリーへ届くためです。

スクリプトが動くまでコンテンツが存在しない。 JavaScriptですべてのテキストを書き込む部品は、定義の読み込みが終わるまで何も読者へ見せず、スクリプトが失敗すれば何も表示しません。商品カードのようにライトツリーへコンテンツを残してスロットへ割り当て、アップグレードが既存の内容を拡張するようにします。表示を待つ必要がある場合、HTML Standard自身の例は、`customElements.whenDefined()` が、 `:not(:defined)` に一致するすべての要素について解決するまで隠しています。

参考資料

- HTML Standard：カスタム要素 — 命名規則、コンストラクター要件、アップグレード、フォーム関連要素、 `ElementInternals` を具体例付きで説明します。
- DOM Standard：シャドーツリー — シャドールートとホスト、スロットと割り当てられたノード、 `attachShadow()` の手順。
- HTML Standard：スクリプト — `template` 要素の `shadowrootmode` 属性と、`slot` 要素。
- CSS Shadow Parts — `part` 、 `exportparts` 、 `::part()` 疑似要素を定める、2025年12月16日付のW3C作業草案。

用語集

実際の制約に合わせて選ぶ

業務の流れ、既存のインターフェース、変更を必要とする利用者をお知らせください。検証可能な納品内容と、お客様のチームが保守を続けられる引き継ぎ内容を定められます。

料金を見る お問い合わせ →

用語を実務に結び付ける

「ウェブコンポーネントとは？」がお客様の組織にどう当てはまるかをご相談ください。現在のシステムの具体例が一つあれば、話を始められます。

サービスを見る

welcome@engineer.company +45 50 10 61 84 デンマーク、コペンハーゲン

このページを持ち歩く

連絡先カード

心を込めて丁寧に制作： D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

著作権 © 2025 – 現在 · Engineer ApS · すべての権利を留保します。

印刷元： https://engineer.company/ja/glossary/web-components/