Engineer

DOM

🌲 ブラウザーがページから作る、動的なオブジェクトの木

どのようなものか

DOMは、Document Object Modelの略で、ブラウザーがページから作り、開いている間、メモリーに保持する、オブジェクトの木です。HTMLパーサーは、マークアップを読み、文書オブジェクトを出力します。各要素、連続したテキスト、コメントが、それぞれノードとなり、各ノードは、親、子、隣接するノードを知っています。スクリプトは、この木を通じてページを読み、変更します。ブラウザーが描くものも、木から決まります。

WHATWGは、DOM標準で、このモデルを維持しています。標準の要約は、特定のプラットフォームに結び付けず、ノードの木、イベント、進行中の処理の中止という、三つの対象を挙げます。名前は、標準より古いものです。元は、文書へアクセスして変更するインターフェースを意味しました。今は、木自体もDOMと呼ぶことが多く、両方の意味を日常的に使います。HTML標準は、その上に構築されています。HTMLは、要素とその動作を定義し、DOM標準は、ノード、要素、属性、イベントとは何かを定義します。

木は、ソースファイルではありません。ページのHTMLは、ネットワークから届いたテキストです。DOMは、HTML標準の規則に従ってすべての解析エラーを処理し、パーサーがそのテキストから作ったものを、その後のスクリプトが変更した結果です。パーサーは、忘れがちな細部も残します。二つのタグの間の空白が、独立したテキストノードになる場合があります。そのため、最初の子ノードは、必ずしも中の最初の要素ではありません。ただし、標準自身の例も、すべての空白がテキストノードになるわけではないと記しています。

ノードには、いくつかの種類があります。ルートには文書があり、その下に、文書型宣言、要素、テキストノード、コメントがあります。文書フラグメントは、ページの木の外にある、軽い入れ物です。ブロックを組み立ててから一度に挿入する場所として、便利です。要素は、シャドウツリーを持つこともできます。独立した別の木で、文書の通常のクエリーは、中へ届きません。

ノードの一覧には二種類あり、その違いが重要です。標準では、特記がなければ、コレクションはライブです。`childNodes` と、`getElementsByClassName()` が返すHTMLCollectionは、木が変わると、ともに変わります。`querySelectorAll()` は、明記した例外で、呼び出し時点のスナップショットである、静的な一覧を返します。

同じ標準が、イベントの伝わり方も定義します。イベントは、対象のノードへ送出され、三段階を通ります。木の上から対象の祖先を通って下るキャプチャー、対象自体、祖先を通って上へ戻るバブリングです。`eventPhase` プロパティーは、段階を1、2、3で表し、送出中でなければ0です。`addEventListener()` で加えるリスナーには、オプションがあります。`once` は、最初の呼び出し後に取り除き、`passive` は、`preventDefault()` を呼ばないと約束します。`signal` は、`AbortController` へ結び付け、コントローラーを中止すると、リスナーも外れます。

概念を判断に生かす

「DOM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

ページを変えるスクリプトは、言語やフレームワークに関係なく、DOMを通じて動きます。フレームワークは、独自の層を上に加えますが、読者が最後に見るものは、この木のノードです。Next.jsは、ハイドレーションを、ReactがDOMへイベントハンドラーを取り付けることと説明します。サーバーから届いたHTMLが、クリックに応答するページになる仕組みです。下にある木を理解したチームは、フレームワークへ頼んだ内容ではなく、実際の動作をデバッグできます。

木は、テキストとマークアップが分かれる場所でもあります。`textContent` へ代入した文字列は、要素の子を、一つのテキストノードで置き換え、マークアップとして解釈されることはありません。同じ文字列を `innerHTML` へ代入すると、HTMLパーサーへ渡します。HTML標準は、このHTMLの代入処理では、既定で危険なものを取り除かないと記しています。どちらを使うかで、訪問者の表示名が、テキストとして表示されるか、コードとして動くかが決まります。

読者の体験は、木だけでは決まりません。スタイルが、ノードの可視性、位置、ポインターで届くかどうかを決めます。ノードをたどるだけの道具からは、どれも見えません。この差が、自動検査のよく間違う部分で、文書だけでなく、描画されたページもテストする理由です。

最後に、DOMは、サーバーの仕事が終わった後のページと、それを読むすべてのものとの間の取り決めです。クリックして進むテスト、ブラウザー拡張、支援ソフトウェアは、すべてライブの木から始めます。この取り決めには、見落としやすい別の側面もあります。JavaScriptを動かさないクローラーは、送られたHTMLだけを見て、後からスクリプトが加えるノードは見ません。

実際の活用例

この記事用に書いた次のスクリプトは、データから注文の一覧を作ります。項目ごとではなく、一覧に一つだけクリックリスナーを置き、後で、一回の呼び出しで取り除けます。 `openOrder` は、注文を選んだときにページが行う処理を表します。

```
const list = document.querySelector("#orders");
const controller = new AbortController();

function render(orders) {
  const items = orders.map((order) => {
    const li = document.createElement("li");
    li.dataset.id = order.id;
    li.textContent = `${order.number}: ${order.status}`;
    return li;
  });
  list.replaceChildren(...items);
}

list.addEventListener(
  "click",
  (event) => {
    const li = event.target.closest("li");
    if (li) openOrder(li.dataset.id);
  },
  { signal: controller.signal },
);

// ビューを破棄するとき：
controller.abort();
```

どの部分も、標準に対応します。`createElement()`は、挿入するまで親を持たない要素を作ります。`textContent`は、注文番号をテキストとして置くので、山括弧を含む状態表示でも、状態表示のままです。`replaceChildren()`は、古い項目を消し、一回の操作で新しい項目を挿入します。項目のクリックは一覧までバブリングするので、一つのリスナーが、すべての項目に対応します。`closest()`は、クリックした場所から、最も近い`li`までたどります。コントローラーを中止すると、ページが関数の参照を残さなくても、リスナーを取り除けます。

このサイトのレビューで見つかった、ある種類の欠陥も、木から説明できます。2026年8月のアクセシビリティーレビューでは、自動ルールを通っても、十一件の欠陥がありました。どれも、マークアップの内容ではなく、描画に関わるものでした。フッターの八つのリンクは、`pointer-events`によりポインターでは操作できないのに、タブ移動の順序には残っていました。このプロパティーが取り除くのは、ポインターによる操作だけです。文書を読む道具には、普通のリンクです。ポインターは届かず、キーボードは、すべてで止まりました。スクロールに連動するアニメーションによって、三つのページではフッターが切れて表示されなくなっていました。レビューが残した四つの規則のうち、一つは、木だけを信じないための警告です。可視性は、文書の木だけでなく、描画したピクセルから確かめる必要があります。

問題が起きるところ

ライブのコレクションを回すループが、一つおきに飛ばします。 `getElementsByClassName()`のHTMLCollectionはライブなので、添え字で各要素のクラスを取り除くループは、途中でコレクション自体を縮め、一つおきに飛ばします。代わりに、静的な一覧をたどります。`querySelectorAll()`は静的な一覧を返し、`Array.from()`なら、先にライブのコレクションを通常の配列へ変えられます。

```
const pending = document.getElementsByClassName("pending");
for (let i = 0; i < pending.length; i++) {
  pending[i].classList.remove("pending"); // 一つおきに飛ばしてしまう
}

for (const el of document.querySelectorAll(".pending")) {
  el.classList.remove("pending"); // 静的な一覧なので、すべてを処理する
}
```

リスナーが、積み重なります。 描画のたびにリスナーを加え、取り除かないビューは、イベントのたびに、描画した回数だけハンドラーを動かします。三回訪問すると、一回のクリックで三つのリクエストを送ります。手で取り除くには、

加えたときと、まったく同じ関数オブジェクトが必要です。呼び出しの場でアロー関数を書くと、後で渡すものが残りません。先の例のように、ビューごとの一つの `AbortController` から `signal` を渡します。一回の `abort()` で、そのビューが加えたすべてのリスナーを外せます。一度だけ動かすものには、 `once: true` を使います。

テキストを、マークアップとして入れます。 データからHTML文字列を作り、 `innerHTML` へ代入すると、既定のサニタイズなしで、パーサーへ渡します。エラーハンドラー付きの画像タグなど、マークアップを含む表示名が、ハンドラーの付いた本物の要素としてページへ届きます。テキストは、 `textContent` か、 `createTextNode()` で作るテキストノードへ置きます。本当に文字列からマークアップを作る必要がある場合、HTML標準は、解析中にサニタイズし、危険な内容を取り除く `setHTML()` を定義しています。

オブザーバーが、自分自身を動かします。 `MutationObserver` は、監視するノードの変更を報告し、変更の各まとまりの後に、コールバックを動かします。カウンターを加えたり、クラスを直したりするために、監視中のサブツリーへ書くコールバックは、新しい変更レコードを生み、再び呼ばれます。ページが、自ら作るループで動かなくなる場合があります。必要なところだけを、 `attributeFilter` を使い、可能なら `subtree` なしで監視します。書く前に、すでに変更済みか確かめるようにするか、書く前に `disconnect()` 、後に `observe()` を呼びます。

参考資料

- DOM標準 — WHATWGによる、ノード、木、ライブと静的なコレクション、イベント、 `AbortController` 、 `MutationObserver` の定義です。
- HTML文書の解析 — HTMLパーサーが、文字の流れから、解析エラーも処理しながら、文書オブジェクトを作る方法です。
- 動的なマークアップの挿入 — `innerHTML` 、 `setHTML()` 、 `setHTMLUnsafe()` と、それぞれがサニタイズするかどうかです。
- サーバーとクライアントのコンポーネント — Next.jsによるハイドレーションの説明です。サーバーからHTMLが届き、その後、DOMにイベントハンドラーを取り付けます。



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

「DOM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/dom/