UI 設計のための HTML デザイン パターン

日野 克哉
株式会社東芝
ソフトウェア技術センター

鷲崎 弘宜
早稲田大学 基幹理工学研究科
情報理工学専攻

HTML は本来、文書を記述するための言語であり、現在の Web アプリケーションやサイトに見られるような高度なインタラクションを伴う UI を実現するためのタグは用意されていない。そのため、HTML で用意された既存のタグを組み合わせて使用し、様々な UI を実現している。

ここで、世の中に存在する UI のパターンは限られているのにも関わらず、HTML におけるそれらの実装方法が何通りもあり、提供したい UX に対する UI を実現する最適な実装を瞬時に選択できないという問題がある。

この要求に応えるために、Web アプリケーションやサイトでよく見られる UI のパターンを抽出し、HTML でアクセシビリティを確保しつつ実装する方法を HTML UI Design Patterns としてまとめる。

これにより、様々な閲覧環境でユーザビリティの高い HTML を素早く実装できるようになるほか、パターンを適用した HTML 間では、HTML の構造が共通となるため、これに適用できる CSS や JavaScript を使いまわせるようになる。

1. はじめに

Web アプリケーションやサイトは閲覧環境が多様でユーザビリティの確保が困難であり、またサイトが提供する情報の意味的な構造化がユーザ エージェントや検索エンジンなどからの評価において度々重要視される。よって、Web アプリケーションやサイトの UI は、一環境における完成図だけを意識して作成するのではなく、これらの副次的な事項も考慮して作成することが望ましい。しかし Web アプリケーションやサイトの UI 実装に使用される HTML は記述の自由度が高く、一つの UI を実現するだけでも複数の実装方法が存在する。そのため開発者はユーザビリティや構造化が考慮された最適な実装を瞬時に選択することが困難となっている。

本論文では、HTML で記述された Web アプリケーションおよびサイト向けの UI 設計時にみられる HTML 設計のパターンを「HTML UI Design Patterns」として紹介する。HTML UI Design Patterns で抽出されるパターンは、HTML5 を活用し、より意味的な構造化がなされた HTML を設計することを基本的な指針としている。これにより、様々な閲覧環境におけるアクセシビリティの向上や検索エンジン最適化（SEO）への対策といった効果が

期待できる。

HTML UI Design Patterns の位置づけを図 1 に示す。Designing Interfaces [1]は Web というメディアに限らない汎用的な UI を網羅したデザイン パターンである。Web で使用される UI パターンも含まれるが、HTML の設計については触れられていない。そのため Web 特有の多様な閲覧環境に対応したアクセシビリティについての考慮は含まれていない。一方で HTML UI Design Patterns は、Designing Interfaces で紹介されたパターンのうち Web アプリケーションおよびサイトでよく使用されるパターンを Web 特有のアクセシビリティを考慮しつつ HTML での実装例も交えて紹介している。パターンには UI を HTML で実現する際の HTML 構造設計方法と、必要に応じて UI の機能に必要最小限の外見を定義しない CSS と JavaScript の記述が定義されている。また、Pro CSS and HTML Design Patterns [2]は HTML および CSS を使用して表現できる外見や UI の記述方法を網羅したデザイン パターンである。こちらは各パターンに具体的なコードが HTML および CSS 共に定義されていて、パターンが対象とする解決策は構造から外見まで含んだ具体的なものであるため、その性質は GUI 部品ライブラリに近い。また、比較的内容が古いため、最新の HTML5 や CSS3 への対応、検索エンジン最適化、アクセシビリティ対策などに対応していない。

なお、HTML UI Design Patterns が扱う UI の粒度は、画面部品（またはコントロール）の粒度から、それらを複数組み合わせたものくらいの間を対象としている。

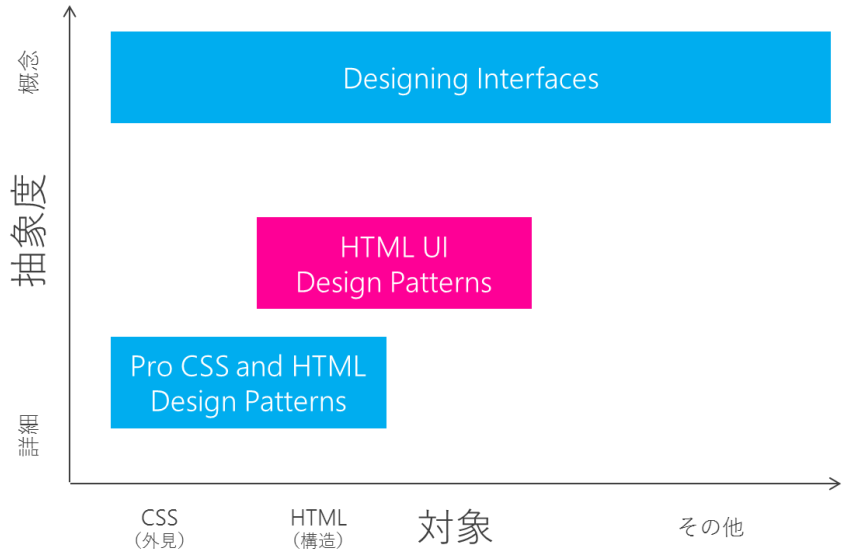


図 1 類似デザイン パターン内での HTML UI デザインパターンの位置づけ

2. HTML による UI パターンの設計について

HTML で UI を設計するには、HTML で用意されている HTML 要素を組み合わせる。HTML 要素には基本的なものしか用意されておらず、高度な機能やインタラクションを実

現するものは、HTML 要素を組み合わせて作成しなければならない。

HTML UI Design Patterns で紹介する各パターンでは、HTML で当該 UI パターンを実現する際の最低限必要な HTML 要素を示している。デザイン パターンを参照する者は、ここで示している以外の要素を必須要素の内外に追加して、各自の UI を設計する。

また、各必須要素には推奨クラス名が定義されている。推奨クラス名をつけると、JavaScript による動作定義が必要なパターンにおいて、自前で JavaScript を実装することなく、HTML UI Design Patterns に対応した JavaScript ライブラリをそのまま適用できる。クラス名以外にも、必須要素には条件が指定されることがある。条件には、HTML 要素型、必須属性、子要素保持の可否、同じ要素内に同時に存在できる数が挙げられる。

3. パターンの分類とパターン間の関連

HTML UI Design Patterns では、UI パターンを次の 6 つのグループに分類した。

- レイアウト
  画面全体の配置を決定するパターン。
- ナビゲーション
  画面遷移の手段を提供するパターン。
- アクション
  機能や操作の手段を提供するパターン。
- データ表示
  データの様々な形式で表示するパターン。
- 入力
  データの入力手段を提供するパターン。
- 外見
  外見の変化を司るパターン。

本論文では、これらの分類のうち、ナビゲーションおよび入力に属するパターンについて紹介する。

また、HTML UI Design Patterns の参照者が適用すべきパターンを選択する際の手助けとして、HTML UI Design Patterns で紹介する各パターン間の関連と、図 1 で紹介した類似するデザイン パターンで紹介されているパターンとの関連を図 2 に示した。

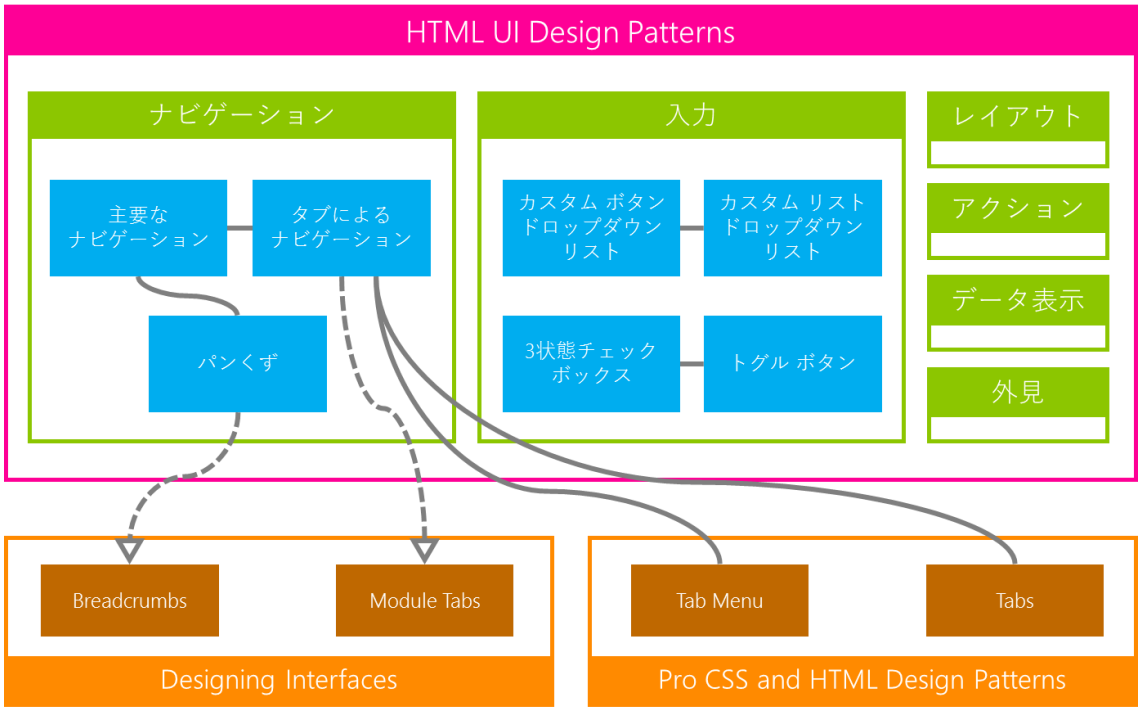


図 2 パターンの関連。パンくずは Breadcrumbs の HTML での実装、タブによるナビゲーションは Module Tabs の HTML での実装、その他の関連は類似パターン。レイアウト、アクション、データ表示、および外見の分類に関しては、本論文では省略する。

4. デザイン パターン

本章では、HTML UI Design Patterns に含まれるパターンを紹介する。なお、パターンの紹介は Canonical Form[1]に照らし合わせ、名前・分類・文脈・問題・フォース・解決策・解決後の状況・事例・関連するパターンの項目を記述している。

4.1. 主要なナビゲーション（Major Navigation）

Web Images Videos Maps News

分類

ナビゲーション

文脈

目の不自由な利用者を想定した Web アプリケーションやサイトを構築している。目の不自由な利用者はスクリーン リーダという文字の読み上げ機能を使用して画面上の情報を読

[1] http://c2.com/cgi/wiki?CanonicalForm

み取る。目視では本文や興味のある位置までの読み飛ばしが容易なため気づかないが、一つの画面に記載される文字量は思いのほか多く、これらすべてが読み上げられると利用者は必要な情報を選別することが困難となる。

問題

Web アプリケーションやサイトには、本文の内容とは直接関係のない様々なリンクが存在し、そのリンクもサイト内の主要なページへ遷移するリンクのようにユーザにとって使用頻度の高いものから、免責事項記載へのリンクなど繰り返し何度も遷移しないリンクのように使用頻度の低いものまである。これらをすべて読み上げられてもユーザは使用頻度の高いリンクを容易に扱えない。

フォース

音声読み上げでの利用を想定する。しかし目視における一覧性も引き続き確保する必要がある。

音声読み上げ時の利便性を向上させる必要がある。しかし同一内容で音声読み上げ用のページを作成し同期するといった手間はかけられない。

使用頻度の高いリンクへのアクセスをしやすくする必要がある。しかし使用者にとって使用頻度の高いリンクがページ内のあらゆる場所に分散している。

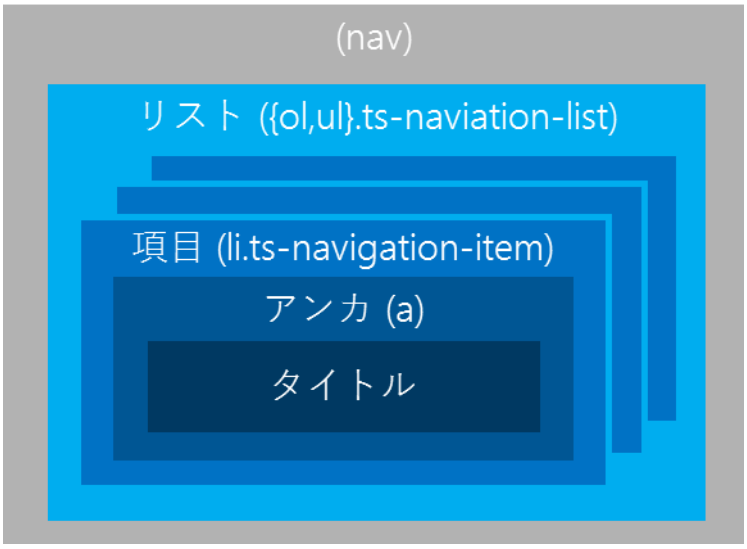
デザインの都合上、音声読上げ時にも本文やナビゲーションへのアクセスを簡単にする必要がある。しかし、目視の際のデザインを重視する必要があり、ページ内にアクセスを簡単にするためのリンクを導入できない。

目視での利便性を優先して HTML 構造を作成するため、読み上げ順を考慮して HTML 要素を再配置できない。

解決策

使用頻度の高い比較的少数のリンクを「主要なナビゲーション」として配置し、これを nav 要素で囲う。nav 要素で囲うとスクリーン リーダは nav 要素だけの読み上げや nav 要素の読み飛ばしができるようになる。

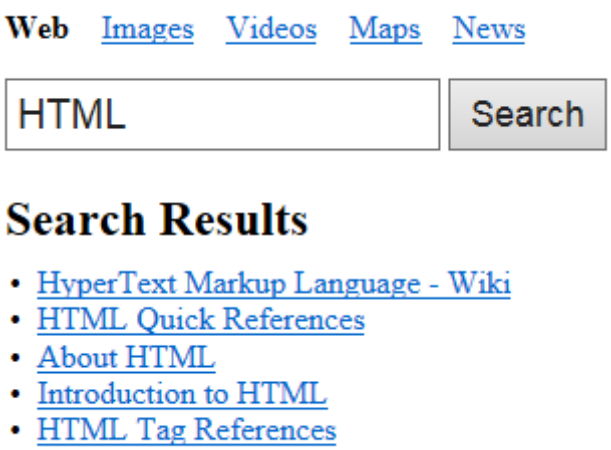
リンクはリストを表現する要素（ol または ul 要素）を使用する。各ページへのナビゲーションをリストの項目（li 要素）で表現する。リストの項目には、リンクの実体であるアンカ（a 要素）を入れる。



解決後の状況

スクリーン リーダは「主要なナビゲーション」の位置を把握でき、ユーザは使用頻度の高いページへの遷移を容易にできるようになり、また遷移を必要としない場合はリンクを読み飛ばすことができる。

事例

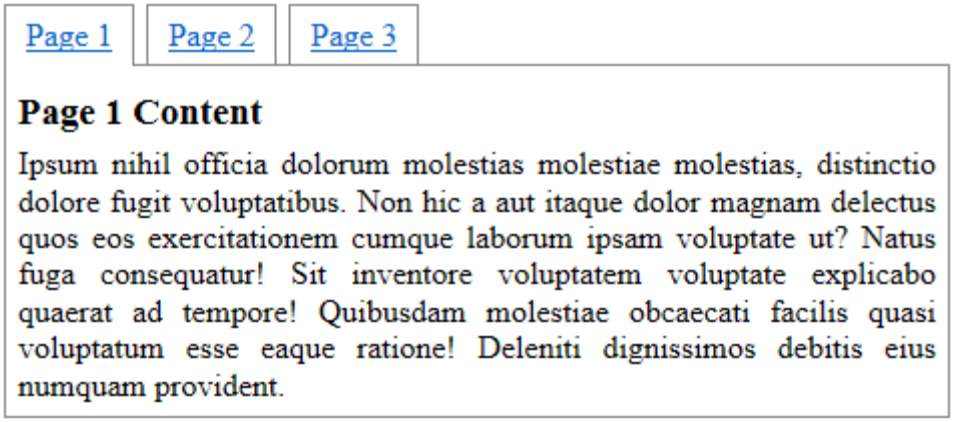


最近のインターネット検索サイトでは検索対象のカテゴリを切り替えられる。例の上部にあるカテゴリを切り替えるためのリンク集には本パターンを適用する。一方、検索結果のリンク集は検索サイトのコンテンツであるので、本パターンは適用できない。

関連するパターン

- ナビゲーション先に階層構造がある場合は「パンくず」パターンを検討する

4.2. タブによるナビゲーション（Tab Navigation）



分類

ナビゲーション

文脈

画面の一部分で複数のタイトル付きの表示内容を素早く切り替えられるタブ型の UI を作成している。ページ全体を通常のハイパーリンクで遷移させると HTTP のトランザクションが発生し、ページ全体の読み込みと描画が起こり、素早い切り替えが実現できない。そのため JavaScript を使用して HTML 要素の表示・非表示を切り替えることで素早い切り替えを実現するが、表示内容の切り替えが JavaScript に依存してしまう。

問題

JavaScript が無効の環境で閲覧した時に表示内容が切り替えられなくなり、表示できない内容が生じてしまう。

フォース

JavaScript が無効の環境で閲覧される可能性もある。

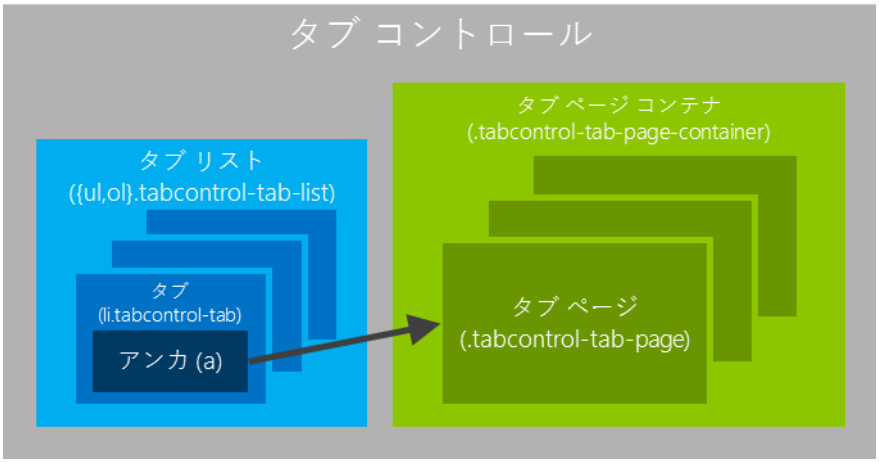
JavaScript が無効の環境でもすべての表示内容が表示される必要がある。切り替えが働かずにすべての内容が表示されても、引き続きタブによるナビゲーションの利便性も確保したい。

解決策

切り替え対象の表示内容のうち、初期に表示されない表示内容の非表示化を JavaScript によって行う。すると、JavaScript 無効の環境では初期の非表示化が行われないため、すべての表示内容が表示されたままになる。さらにタイトルを表現する HTML 要素と表示内容を表現する HTML 要素とをアンカで対応づけることで、JavaScript が無効の場合には、タイトルがそのまま表示内容へのナビゲーション リンクとして機能する。これを実現するには、次の構造で HTML を作成する。

タブ リストですべてのタブを保持する。

タブ ページ コンテナですべてのタブ ページを保持する。
各タブを各タブ ページに対応づける。
タブ ページに id 属性を指定する。
アンカの href 属性に「'#'+タブ ページの id」を指定する。
タブがクリックされたら、対応するタブ ページ以外に、タブ ページが非表示となるようなクラスを JavaScript で指定する。



解決後の状況

JavaScript が無効の時では、すべてのページが表示されるため、アクセスできない内容が生じない。
JavaScript が無効の時では、タブのアンカをクリックすると、対応するページの位置が表示されるため、タブによるナビゲーションができる。

事例

Page 1 | Page 2 | Page 3

Page 1 Content

Ipsum nihil officia dolorum molestias molestiae molestias, distinctio dolore fugit voluptatibus. Non hic a aut itaque dolor magnam delectus quos eos exercitationem cumque laborum ipsam voluptate ut? Natus fuga consequatur! Sit inventore voluptatem voluptate explicabo quaerat ad tempore! Quibusdam molestiae obcaecati facilis quasi voluptatum esse eaque ratione! Deleniti dignissimos debitis eius numquam provident.

Page 1 | Page 2 | Page 3

Page 1 Content

Ipsum nihil officia dolorum molestias molestiae molestias, distinctio dolore fugit voluptatibus. Non hic a aut itaque dolor magnam delectus quos eos exercitationem cumque laborum ipsam voluptate ut? Natus fuga consequatur! Sit inventore voluptatem voluptate explicabo quaerat ad tempore! Quibusdam molestiae obcaecati facilis quasi voluptatum esse eaque ratione! Deleniti dignissimos debitis eius numquam provident.

Page 2 Content

Elit quasi delectus corporis voluptate nesciunt. Amet assumenda perferendis nostrum non similique. Sequi ipsum laborum et consequatur tempore consequuntur doDolor asperiores sunt itaque in minima iusto suscipit aliquam veniam ipsum autem earum dignissimos. Sunt atque placeat earum corrupti nobis? Reprehenderit tempora sapiente tenetur qui eos odit vero tempore, sapiente! que iusto voluptas.

Page 3 Content

Amet amet id necessitatibus sunt doIpsum odio eos aut quis quo? Neque quod illum aliquam odit nam? Magni facilis distinctio placeat expedita illo impedit voluptas. Sint cum voluptates quia consequuntur reiciendis repellat eligendi tempora adipisci. que! Impedit corrupti ab facere saepe sed officia nulla voluptatum vero illum totam quam veritatis ab, magni. Accusamus et asperiores architecto voluptatem vero nostrum quam.

「タブによるナビゲーション」を使用して複数の内容を切り替えて表示できる領域を作成した。右は JavaScript が無効な場合で、すべてのページの内容が表示され、タブをクリックすると該当するページの内容が表示されている位置までスクロールする。

関連するパターン

- ページ全体のナビゲーションは「主要なナビゲーション」パターンを検討する
- Web に限らない、より汎用的なパターンについては、Designing Interfaces の「Module Tabs」パターン[1]を参照する
- JavaScript を使用せず、複数のページをタブの外見でナビゲーションするときには、Pro CSS and HTML Design Patterns の「Tab Menu」パターン[2]を参照する
- Pro CSS and HTML Design Patterns の「Tabs」パターン[2]は本パターンと似ているが、JavaScript が無効の環境では一部の表示内容が表示されず、また表示内容の非表示化に画面外の座標を指定する方法が取られており、SEO においてペナルティとなる可能性がある

4.3. パンくず（Breadcrumbs）

分類

ナビゲーション

文脈

ページが階層化された Web サイトを構築している。各ページには、そのページの階層を示し、さらに上位階層へ遷移するためのナビゲーションを、a 要素を使用したリンクで作成している。単なるリンクなので、これが階層を表現していることは UI を目視している人間にしかわからない。

問題

このナビゲーションが階層を表現していることが UI を目視している人間にしかわからないため、検索エンジンによって当該ページが検索結果に表示される際に当該ページの階層情報が表示されず、検索エンジンからの訪問者のユーザビリティが低下する。

フォース

検索エンジンからサイト内のサブ ページへの流入が想定される。
階層表示が原因でユーザが期待しないページへアクセスすることは避けたい。しかし、ユーザの利便性を維持するために階層表示は廃止できない。

解決策

検索エンジンのクローラが階層情報を機械的に読み取ることができるように、階層情報を表現できる Breadcrumb という microdata を埋め込んだ「パンくず」ナビゲーションを配置する。

各ページへのリンクのリストなので、全体をリスト（ul 要素）で表現する。各ページへのナビゲーションをリストの項目（li 要素）で表現する。リストの項目には、リンクの実体であるアンカ（a 要素）を入れる。アンカ内には span 要素を入れ、その中にタイトルを記入する。

このナビゲーションが、当該サイト内の移動における重要な手段である場合は、スクリーンリーダでナビゲーションしやすくするためにさらに全体を nav 要素で囲い、ナビゲーションのためのリンクであることを表現する。

検索エンジンがページを結果に表示する際、サイト内における位置も同時に表示されるようにする。これには検索エンジンにパンくずであることを伝える必要がある。microdata を埋め込むことで、パンくずであることを表現できる。リストの項目には、itemscope 属性を指定し、1 つのリンクを表す HTML 上での範囲を明示する。リストの項目には、次の属性と属性値のペアを指定し、この項目がパンくずの 1 項目であることを明示する:

itemtype="http://data-vocabulary.org/Breadcrumb"

アンカには、次の属性と属性値のペアを指定し、これが当該項目の URL を示していることを明示する:

itemprop="url"

タイトルには、次の属性と属性値のペアを指定し、これが当該項目のページ タイトルを示していることを明示する:

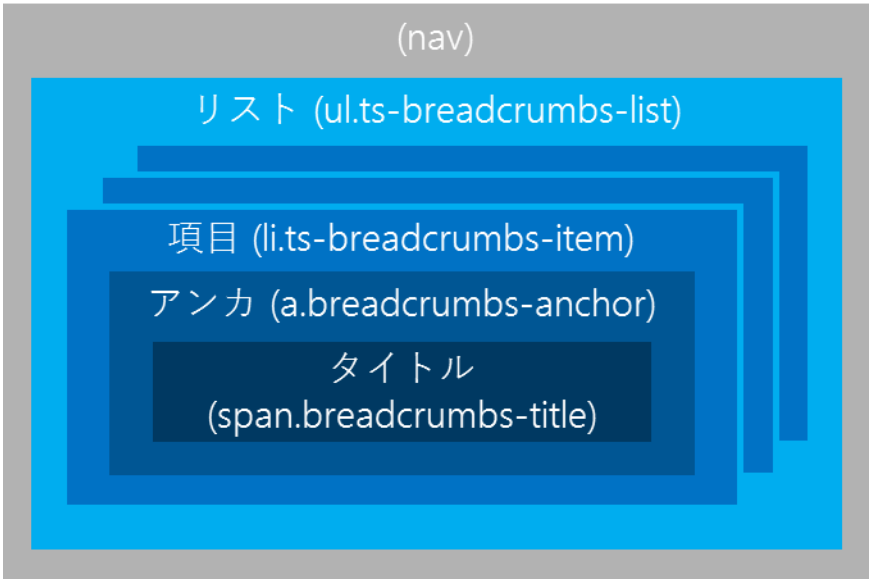
itemprop="title"

項目間の区切り文字（「>」など）は、CSS を使用して表示することが望ましいが、項目内のアンカ外領域にテキストで記入することもできる。

解決後の状況

検索エンジンの結果に当該ページのサイト内における位置も表示され、ページを訪れる前にユーザがサイトの構造を把握できる。

検索エンジンの結果から、ホーム ページなど、サイト内にある当該ページ以外のページにも直接たどれるようになる。



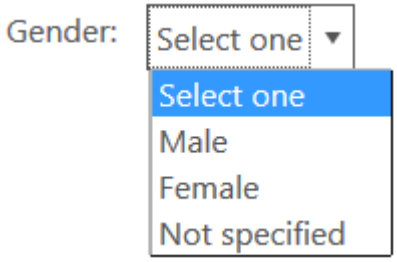
事例

ショッピング サイトでは、商品ページがカテゴリ別に分けられている。各ページの上部には、現在表示しているページのサイト全体における位置を表示するリンク リストが表示される。このリストには本パターンを適用できる。

関連するパターン

- 階層構造のないフラットなナビゲーションは「主要なナビゲーション」パターンを検討する
- Web に限らない、より汎用的なパターンは Designing Interfaces の「Breadcrumbs」パターン[1]を参照する

4.4. カスタム ボタン ドロップダウン リスト（Custom Button Drop-down List）



分類

入力

文脈

外見のデザインを重視した Web アプリケーションやサイトに含まれるフォーム上にドロップダウン リストを配置しようとしている。フォームでは、ドロップダウン リストは select 要素で実現される。

問題

多数の Web ブラウザでは、select 要素の収縮時における外見を変更できないため、ドロップダウン リストだけ Web ブラウザで描画されるデフォルトの外見になってしまう。そのため、他の部分の外見と統一感のあるデザインが実現できない。

フォース

すべてのブラウザで同様の収縮時における見た目を実現する。

タッチ スクリーン デバイスに搭載されているブラウザから閲覧する可能性がある。しかしこれらの環境では収縮時の外見をマウス環境のものと統一してしまうと使い勝手が悪くなる。

解決策

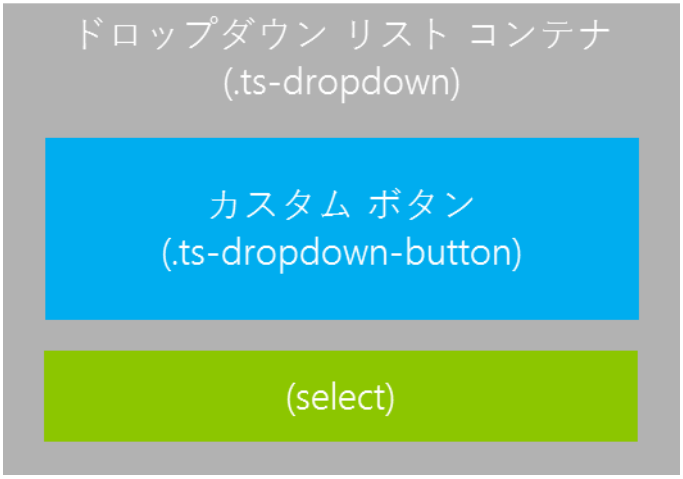
ドロップダウンを表現する select 要素を使いつつ、収縮時の外見を独自に作成して実現する。

select 要素は CSS で透明にし、その下に収縮時のドロップダウン リストの外見を表現する「カスタム ボタン」を重ねる。

カスタム ボタンは適当な要素（span 要素など）を使って表現する。

ドロップダウンはブラウザ ネイティブのものを使用する。そのためには、select 要素を透明にし（opacity: 0）、カスタム ボタンの上に同じ大きさで重ねる。

なお、カスタム ボタンと select 要素をドロップダウン リスト コンテナ（div 要素など）で囲い、高さと幅を 100%に指定すると、2 つの要素を同じ大きさにできる。



解決後の状況

ブラウザ間で収縮時におけるドロップダウン リストの見た目が統一され、入力フォームな

どの体裁が崩れなくなる。
選択画面はブラウザのネイティブのものを使うので、スマートフォンならスマートフォンに適した選択画面が表示される。

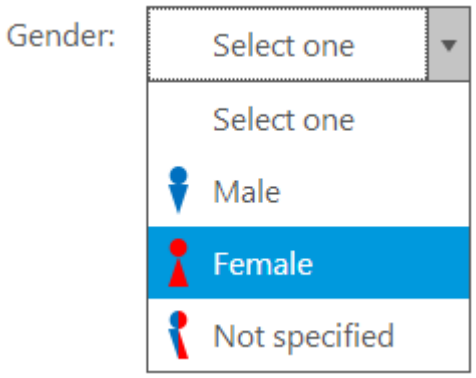
事例

性別を選択するドロップダウン リストを、select 要素を使ったものと本パターンを適用したものとで上下に並べ、Internet Explorer 11（左）と Firefox 26（右）で表示させたところ。本パターンを適用すると、異なるブラウザでも同様の外見を実現できる。

関連するパターン

- ドロップダウン リストの展開時の外見も統一する場合は「カスタム リスト ドロップダウン リスト」パターンを検討する

4.5. カスタム リスト ドロップダウン リスト（Custom List Drop-down List）



分類

入力

文脈

外見のデザインを重視した Web アプリケーションやサイトに含まれるフォーム上にドロップダウン リストを配置しようとしている。フォームでは、ドロップダウン リストは select 要素で実現される。

問題

多数の Web ブラウザでは、select 要素のドロップダウン時における外見を変更できないため、ドロップダウン リストだけ Web ブラウザで描画されるデフォルトの外見になってしまう。そのため、他の部分の外見と統一感のあるデザインが実現できない。

フォース

すべてのブラウザで同様のドロップダウン時における見た目を実現する。

ドロップダウン時に表示される項目として、画像などの文字列以外のものも扱う。

解決策

「カスタム ボタン ドロップダウン リスト」において、さらにドロップダウン時に出現するリストを表現する要素を独自に作成する。

ドロップダウン時に出現するリストは、リスト要素（ol または ul 要素と、li 要素）を使って表現する。

次のスタイルを指定して、select 要素の機能を保ったまま非表示にする:

position: absolute;

clip: rect(0,0,0,0);

select 要素に tabindex="-1"属性を追加し、ネイティブのドロップダウン リストがフォーカスを得ないようにする。

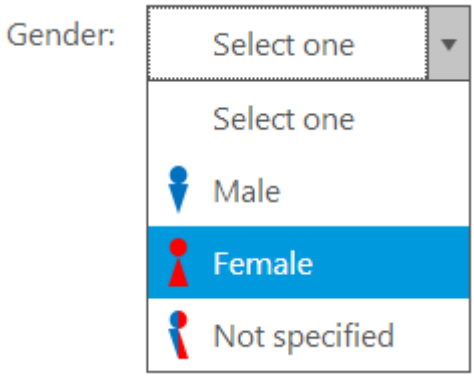
代わりにドロップダウン リスト コンテナの tabindex 属性に 0 以上を指定し、カスタム リスト ドロップダウン リストがフォーカスを得られるようにする。

解決後の状況

ブラウザ間でドロップダウン時のドロップダウン リストの見た目が統一される。

リストの項目に文字以外の項目が追加できて、表現力が向上する。

事例

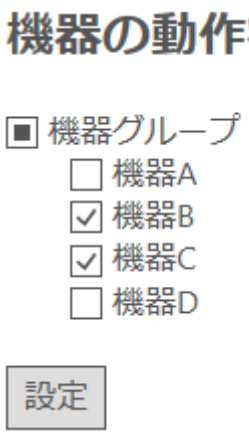


性別を選択するドロップダウン リストに本パターンを適用させ、項目にアイコンを追加した。

関連するパターン

- タッチ スクリーン デバイスなどで閲覧される可能性がある場合は「カスタム ボタン ドロップダウン リスト」パターンを検討する

4.6. 3 状態チェック ボックス（Three-state Check Box）



分類

入力

文脈

Web アプリケーションやサイト内のフォームで、ユーザに「チェック」・「未チェック」・「チェックでも未チェックでもない」という 3 状態を入力させようとしている。ユーザは 3 状態をとるチェック ボックスの使用に慣れており、Web においてもこれの使用を期待している。

問題

type 属性が”checkbox”と指定された input 要素はフォームで使用できるチェック ボックスであるが、これは「チェック」・「未チェック」の 2 状態しか入力できない。

フォース

HTML では 3 状態を入力するためにドロップダウン リストが使用される。しかし、ドロップダウン リストは画面上を占める面積が広く、これを設置する余分なスペースがない。

解決策

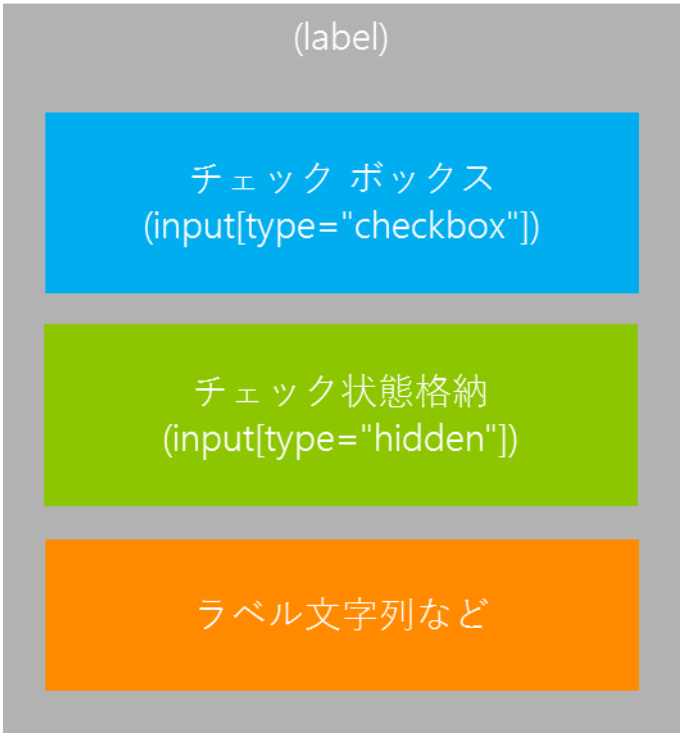
type 属性が”checkbox”と指定された input 要素を使用しつつ、チェック ボックスの外見を独自に作成し、JavaScript で 3 状態の制御を行うことで「3 状態チェック ボックス」を実現する。

チェック状態を格納する要素（非表示 input 要素）を配置し、そこにチェック状態を数値などで格納する。

チェック状態格納要素に name 属性を指定すると、フォームでこの値が取得できる。

チェック ボックスがクリックされた時、チェック ボックスの外見を JavaScript で 3 状態に変更する。

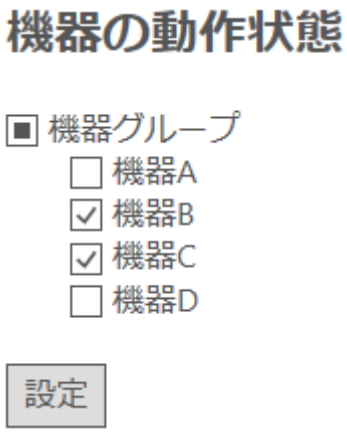
同時にチェック状態格納の value 属性も JavaScript で変更する。



解決後の状況

他の input 要素と同じように使える、3 状態チェック ボックスが設置される。

事例



機器の動作状態を 2 値で設定する画面において、複数の機器にまとめて値を設定したい場合がある。このような場合、複数の機器をまとめたグループにチェック ボックスを設ければよいが、グループ内の機器の状態が統一されていない場合、そのチェック ボックスは「チェック」でも「未チェック」でもない値をとる。このような UI を実現するには、本パターンを適用する。

関連するパターン

- 「トグル ボタン」パターンは、「3 状態チェック ボックス」パターンと同様にフォーム上で状態をユーザに入力させるが、入力させる状態は 2 状態のうちどちらかの状態である

4.7. トグル ボタン（Toggle Button）

分類

入力

文脈

Web アプリケーションで、「オン」・「オフ」の 2 状態を切り替えて制御する UI を作成している。チェック ボックスやラジオ ボタンでも「オン」・「オフ」の 2 状態を切り替えられるが、押された状態を保持するボタンでこれを表現しようとしている。

問題

HTML でボタンを表現する button 要素や input 要素では押された状態を保持できないため、「オン」・「オフ」状態を表現できない。また、チェック ボックスやラジオ ボタンを表現する input 要素は多くのブラウザで外見を変更できないため、これをボタンのような外見にすることも困難である。

フォース

JavaScript でトグル ボタンを実現すると、HTML のチェック ボックスとの互換性が損なわれる。

すべての Web ブラウザで同じ外見を実現する。

JavaScript が無効の環境で閲覧される可能性がある。

解決策

type 属性が”checkbox”または”radio”と指定された input 要素は、フォームの要素として使用できるチェック ボックスまたはラジオ ボタンである。これを非表示にして、クリックによって 2 値を切り替えて保持する要素として使用する。一方ボタンの外見は独自に作成し、input 要素の値に応じてボタンの凹凸が変化させる。

次のスタイルを指定して、input 要素をクリック可能かつ非表示にする:

position: absolute;

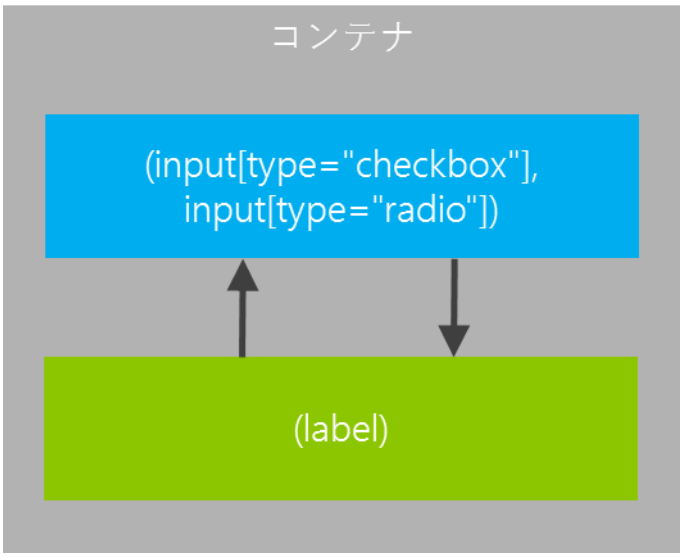
clip: rect(0,0,0,0);

CSS のセレクタでは、+演算子で直前の要素の状態を利用できるため、input 要素の直後に label 要素を配置することで label 要素の外見を input 要素の状態に応じて変更できるようにする。

input 要素に任意の id 属性を指定し、label 要素には for 属性に input 要素の id 属性値を指定する。

label 要素は、input 要素のチェック状態に応じて、オン・オフそれぞれの見た目が切り替わるようにする。

label 要素の働きで、label 要素がクリックされると for 属性で指定された input 要素のチェック状態が変更される。



解決後の状況

JavaScript が無効でも機能するトグル ボタンが設置でき、通常のチェック ボックスやラジオ ボタンと同様に使用できる。

事例

ストリーミング動画などを録画するアプリケーションで、録画の開始と停止という機能が同一のボタンに割り当てられ、そのボタンが録画中か否かの状態をもつことは珍しくない。このようなボタンには本パターンが適用できる。

関連するパターン

- 「3 状態チェック ボックス」パターンは「トグル ボタン」パターンと同様にフォーム上で状態をユーザに入力させるが、入力させる状態は 3 状態のうちいずれかの状態である

参考文献

1. Jenifer Tidwell, Designing Interfaces, 2nd Edition, O'Reilly Media, 2010
2. Michael Bowers, Pro CSS and HTML Design Patterns, Apress, 2007