Vol.2016-SE-192 No.8
Vol.2016-EMB-41 No.8
2016/6/3

アクセシビリティ向上を目的とした HTML 属性記述の分類と系統化

松井 真子[1]　松延 拓生[2]　満田 成紀[2]　福安 直樹[2]　鯵坂 恒夫[2]

概要：近年，ウェブで利用されているアプリケーションは Rich Internet Application（RIA）が主流となっている．しかし，RIA は同一ページ内で動的に情報が更新されるため，支援技術を用いてウェブを利用するユーザには変化が伝わらない可能性がある．このようなアクセシビリティに関する問題を解消するには多大な労力を必要とする．そこで本研究では，HTML を対象に動的なオブジェクトの状態を把握するために WAI-ARIA の仕様を基に必要な HTML 属性記述を検討し，属性の記述方法の分類と系統化を行った．実際にユーザインタフェース（UI）を 22 種類作成し，UI の使用目的毎に分類した結果，入力系・表示系・操作系・ナビゲート系・部品系の大きく 5 種類に分類された．その後，分類毎に標準化すべき属性の記述方法を検討した．JavaScript ライブラリによって，Role 属性から UI の種類を判断し，必須の属性値を動的に付与するサポートが可能であると考えられる．

Classification and Systematization of HTML Attribute Descriptions for Accessibility Enhancement

Mako Matsui[1]　Takuo Matsunobe[2]　Naruki Mitsuda[2]　Naoki Fukuyasu[2]　Tsuneo Ajisaka[2]

Abstract: In recent years, Rich Internet Application（RIA）is a mainstream technology for Web applications. However, people with assistive technologies sometimes can't notify change of information, because RIA can renew contents dynamically in a single page. Web developers need much cost to solve such accessibility problems. This paper considers appropriate HTML attribute descriptions to grasp states of dynamic contents based on the WAI-ARIA specification. These attribute descriptions are classified and systematized to enhance accessibility. The 22 types of UIs were classified into 5 groups of "input", "display", "operate", "navigate" and "control". Attribute descriptions for each classes were normalized. This paper also discusses the JavaScript library which support to develop accessible contents efficiently.

1. はじめに

動的に変化するコンテンツを含んだウェブを Rich Internet Application（RIA）と呼び，デスクトップアプリケーションのようなウェブサービスを作成できるようになり，ユーザにリッチな体験を与えられるようになった．しかし，非同期通信によりページの一部を更新する RIA はスクリーンリーダといった支援技術と相性が悪く，支援技術を用いてウェブを利用するユーザはウェブサービスを利用できないというアクセシビリティの問題が生じた．そのため，2008 年に RIA をアクセシブルにするための仕様書が策定された．それが WAI-ARIA である [1]．この仕様は RIA をアクセシブルにするために Role，State，Property の 3 つの属性を定義している．しかし，WAI-ARIA は詳細に細分化されており，アクセシビリティの問題を解消するには多大な労力とコストを必要とする．また，JavaScript ライブラリを用いれば容易に動的なコンテンツを生成できるが，WAI-ARIA への対応は少なく，属性値を動的に変更させるには開発者が記述しなければならない．

このように，全てのコンテンツをアクセシブルにするためには，多大な労力と時間やコストを要する．そのため，

[1] 和歌山大学大学院システム工学研究科
Sakaedani930, Wakayama 640–8510, Japan
[2] 和歌山大学システム工学部
Sakaedani930, Wakayama 640–8510, Japan

アクセシビリティの対応が困難になっていると考えられる．この問題を解決するためには動的なコンテンツの状態を把握するための属性が HTML の標準技術となれば良い．そこで本研究では動的なオブジェクトの状態を把握するために必要な属性とその値を検討し，HTML における属性記述の分類と系統化を行う．

2. ウェブアクセシビリティとユニバーサルデザイン

2.1 ウェブアクセシビリティに関する研究

ウェブ開発者と支援技術の提供者が独自技術の条件でウェブアクセシビリティに対応すると，正しくアクセシビリティに対応がなされずユーザに混乱を与えてしまう．ゆえに，ウェブアクセシビリティの要件を満たすために求められる条件は標準化されたものであるべきである [3].

2.1.1 ウェブアクセシビリティに関するガイドライン

1999 年 5 月に W3C は，アクセシビリティ向上のためにウェブ開発者が配慮すべき点をまとめたガイドラインである Web Content Accessibility Guidelines(WCAG)1.0 [4] を発表した．世界中で注目され，日本では 2004 年 6 月に，JIS X 8341-3 [5] 日本工業規格として刊行された．

渡辺 [6] は，ガイドラインに頼りすぎると利用者視点が失われがちになるため，ガイドラインだけを重視するべきではないと考えた．そこで，阻害する要因を総合的に検討し，5 つの問題を指摘した．これを基に今後のウェブアクセシビリティに必要な要件として，教育，研究，記述の向上と標準化への準拠，戦略という 4 点の重要性を指摘した．

渡辺ら [7] は，JIS の内容に詳しい 4 名の全盲の視覚障がい者に対して普段の利用から優先すべき項目を調査し，アクセシビリティの観点からは比較的小さな問題であっても配慮が必要だと考えていることがわかった．

2.1.2 RIA のアクセシビリティに関する仕様

JavaScript で実装された RIA が増加し，WAI-ARIA が策定された．この仕様は RIA をアクセシブルにするために Role，State，Property の 3 つの属性を定義している．現在も議論がなされており，頻繁に更新されている．

松田ら [8] は iGoogle および Google ドキュメントを利用する際の問題点を調べ，共通するアクセシビリティの問題が大きく 3 つあることを示した．スクリーンリーダも WAI-ARIA に完全には対応していないことがわかった．

WAI-ARIA で定義された属性を設定するためのモデルやツールが不足している．そこで，Linaje ら [9] は，RUX 方を拡張したモデルベースの RIA UI の作成を提案した．しかし，この手法を適用するためには 1 から RIA を構築する必要があり，既に構築されているものには適用できない．

2.2 ユニバーサルデザイン実践ガイドライン

アクセシビリティを考える上でユニバーサルデザイン (以下，UD と表記) の概念は重要である．支援技術は個別支援が求められるが，情報を扱う HTML などのコンテンツは UD である必要がある．UD については日本人間工学会アーゴデザイン研究部会がまとめたユニバーサルデザイン実践ガイドラインがある [10].

2.2.1 ユーザ分類表

ユーザ分類表とは，UD の対象となるユーザ，特別な配慮を必要とするユーザなどをできるだけ抽出し，配慮すべきユーザの種類やユーザグループを整理するためのものである．配慮すべきユーザがある程度明確である場合に有効である．また，問題点や要求事項を簡単に知りたい場合にも有効である．

3. HTML 属性記述の分類と系統化の方法

3.1 対象とする UI

本研究では WAI-ARIA Authoring Practices 1.1[11] の Design Patterns で挙げられているものを対象とする．WAI-ARIA Authoring Practices は，アクセシブルな RIA を作成するために WAI-ARIA の使用方法の理解を提供するための文書である．動的なウェブコンテンツやキーボードナビゲーションに関するアクセシビリティの配慮すべき点が挙げられている．

Design Patterns には 36 種類の UI が挙げられているが，動的な状態を含まないものやアプリケーション依存が強く標準化が困難なものがある．また，他の UI を複数組み合わせることにより構成されているものがある．そのような他の UI 要素で構成された UI は UI 毎に分析を行うことによって，アクセシビリティについて考慮できると考えられる．UI が複合したことにより考えなければならないアクセシビリティの問題が生じるかもしれないが，その問題はアプリケーション依存が強いものである．そのため，本研究ではそれらの UI を対象から除外する．対象である 22 種類の UI を以下に示す．

(1) Accordion
(2) Alert
(3) Autcomplete
(4) Button
(5) Check Box
(6) Combo Box
(7) Date Picker
(8) Dialog Modal
(9) Dialog Nonmodal
(10) Drag&Drop
(11) Grid
(12) Link
(13) List Box
(14) Radio Group
(15) Site Navigator-General
(16) Slider
(17) Slider Multi-Thumb
(18) Spin Button
(19) Tab Panel
(20) Tooltip
(21) Tree View
(22) Wizard

ただし，対象とする UI であっても，アプリケーションに依存する部分は議論しない．

3.2 UI の作成と HTML 属性記述

想定されるユーザグループをユーザ分類表から抽出する．さらに詳しくアクセシビリティを検討するために，3P タスク分析の情報取得と操作の観点を利用する．その後 UI を作成し，利用目的から UI と HTML 属性記述の分類を実施し，HTML 属性記述に考えられるアクセシビリティの共通点から，HTML 属性記述の系統化を試みる．

3.2.1 UI の作成方法

まず，各 UI の定義を Authoring Practices の Description と Example から定義付ける．次に，UI を作成する上で配慮すべきアクセシビリティの観点の分析を行う．

配慮すべきアクセシビリティの観点を分析するために，ユーザ分類表から「視覚に頼ることができないユーザ」を抽出し，情報提示における「見えない」と「情報の全体量を前もって知りたい」，操作における「ボタンスイッチなどの位置がわからない」という例を参考に UI を操作する上で想定される問題点や要求事項などを検討する．その際に，UI の操作とアクセシビリティの問題点や要求事項は 3P タスク分析の観点を利用して分析する．

Authoring Practices1.1 の Example に挙げられている例を参考に UI を作成する．さらに，Example に挙げられている例以外にも UI の利用方法があれば，追加で作成する．ただし，同じ見た目や動作をする UI であっても，作成のために用いる HTML 要素や JavaScript の処理方法は多様である．また，Example で挙げられている例は jQuery を用いて作成されたものが多い．jQuery などの JavaScript ライブラリを用いれば容易に UI を作成することは可能である．しかし，本研究では動的な状態に対する属性値の記述方法を検討するため，本研究では JavaScript ライブラリは使用せずに作成する．また，見た目も重要なアクセシビリティ要素はあるが，本研究の対象は UI の動的な状態である．そのため，使用される HTML 要素・CSS・JavaScript での処理の実装方法は考慮しない．なお，JavaScript の処理の実装方法とは，処理の内容が同じであるがアルゴリズムは異なるような実装方法のことである．

3.2.2 HTML 属性記述の分類と系統化方法

対象 UI の HTML 属性記述をアクセシビリティの項目・UI の目的・構成要素・コンテンツ外の要素との対応関係の 4 つの観点から分類を試みる．まず，アクセシビリティの項目では，属性値として設定する値が質的な情報であるのか，量的な情報であるのか，操作に関係するものの 3 つから分類を行う．次に，UI の目的では，属性記述としてどのような情報を記述すべかの観点から分類する．さらに，構成要素とは対象 UI がどのような要素から構成されているかの観点であり，具体的には，ヘッダとパネルから構成される UI，自身単体もしくは複数から成る UI，テキストボックスやボタンなどの他の UI から構成される UI，その他に分類する．最後に，コンテンツ外の要素との対応関係では，自身を含むコンテンツと外部の UI・要素との対応関係を表す属性を含んでいるのかという観点から分類を行う．

その後，対象 UI の分類ごとに HTML 属性記述の共通性を整理し，属性記述の系統化を行う．このとき，対象 UI を構成する要素を全て含んだ全体の要素と各構成要素から共通性を整理し，分類における共通の属性記述をまとめる．ただし，UI 特有の属性が多いものは系統化を行えないため，属性記述を整理する対象からは除外する．

4. HTML 属性記述の分類と系統化の結果

3.2.1 に示した方法で 22 種類の UI を作成した．22 種類のうち Accordion と Tan Panel を例として，把握するべき動的な状態とそのための属性値を検討した結果を示す．HTML 属性記述を分類した結果を 4.5 に，系統化した結果を 4.6 に示す．

4.1 利用したアクセシビリティ項目

各 UI 毎に，定義，アクセシビリティの項目と属性値の関係を示していく．なお，アクセシビリティの項目は「見えない」「全体量を前もって知りたい」「操作関係」の観点に分けて列挙する．以下に各項目の説明を示す．

4.1.1 見えない

情報提示に関する項目であり，ユーザは視覚に頼ることができないため，その観点からアクセシビリティを検討する．

4.1.2 情報の全体量を前もって知りたい

「見えない」と同様に，情報提示に関する項目である．その中でも，視覚に頼ることができないため，長い文章を音声情報で聞き取るには労力を要する．ため，事前に知らせるべき情報の全体量の観点からアクセシビリティを検討する．

4.1.3 操作関係

視覚に頼ることができないため，操作に関するオブジェクトの配置や操作方法がわからない．そのような，操作や操作のために必要となるオブジェクトの位置関係の観点からアクセシビリティを検討する．

4.2 属性記述とアクセシビリティ項目の対応付け

列挙した項目と UI に記述すべき属性との関係を表にまとめる．ただし，「見えない」「全体量を前もって知りたい」は情報を提示の項目であるため，類似している．そのため，質的な情報と量的な情報の観点からアクセシビリティの項目をまとめる．量的な情報とは，オブジェクトの数や日付，数字の値などの数字を属性値として設定し支援技術に伝えることにより，状態を把握する情報のことである．質的な情報とは UI の種類や対象 UI を構成する要素などの数字以外の値を支援技術に伝えることにより，状態を把握できる情報のことである．

4.3 Accordion

Accordion は図 1 のようにヘッダとパネルから構成され，パネルを折りたたむとヘッダのみが表示されるため，少ない領域で多くのコンテンツをユーザに提供することができる UI である．

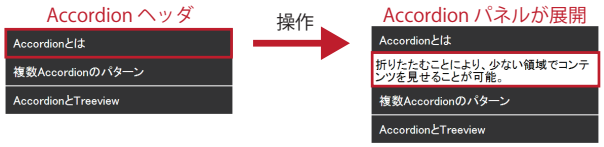

図 1 Accordion の要素

Example では，Accordion の複数個組み合わせたものを例として挙げている．その例を参考に，単一パネルのみ展開する Accordion と複数のパネルを展開できる Accordion を作成した．1 つのパネルが展開されている状態で別のヘッダを操作した場合の Accordion の状態を図 2 に示す．それ以外にも，単体の Accordion を作成した．

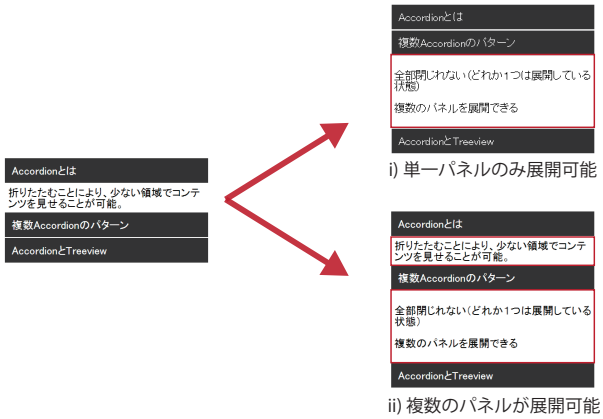

図 2 Accordion の状態

図 2 の Accordion や単一の Accordion を作成する際に検討したアクセシビリティの項目を「見えない」「情報の全体量を前もって知りたい」「操作関係」の観点に分けて列挙する．ただし，Accordion パネルの内容はアプリケーション依存が強いものは考慮すべき点から除外する．

4.3.1 見えない

- 共通
  - UI の種類
  - UI の使用用途
  - 折りたたまれているのか，展開されているのか
  - 選択されているか
- 複数の Accordion
  - 他の Accordion とパネルの展開の関係

4.3.2 情報の全体量を前もって知りたい

- 共通
  - ヘッダとパネルの領域
- 複数の Accordion
  - Accordion の数
  - 最初と最後の Accordion

4.3.3 操作関係

- 共通
  - 展開・折りたたみ方
- 複数の Accordion
  - Accordion 同士の移動方法

4.3.4 アクセシビリティ項目の属性

列挙したアクセシビリティの観点を HTML 記述に反映させるために用いた属性を表 1 に示す．また，操作関係において，ショートカットキーやキーボードでの操作方法は現在の WAI-ARIA Authoring Practices で挙げられているもので良いと判断した．操作に必要な状態を把握するための属性記述を他のアクセシビリティの項目で示しているため，操作関係の属性値は空欄にしている．

4.4 Tab Panel

Tab Panel はファイルのタブのような表現を用いて，少ない領域で複数のコンテンツを提供することができる UI である．作成した UI を図 3 に示す．

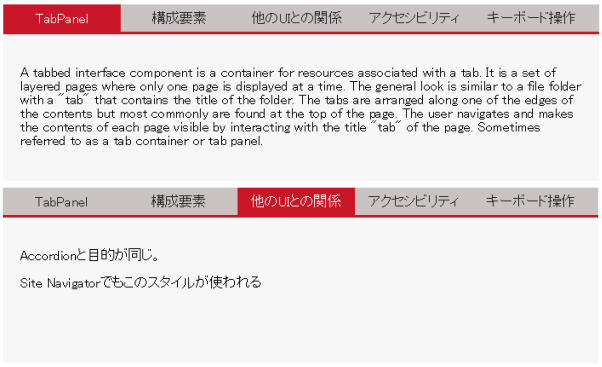

図 3 Tab Panel

図 3 を作成する際に検討したアクセシビリティの項目を「見えない」「情報の全体量を前もって知りたい」「操作関係」の観点に分けて列挙する．

4.4.1 見えない

- UI の種類
- UI の使用用途
- 今表示されているコンテンツはどのタブのものなのか

4.4.2 情報の全体量を前もって知りたい

- タブの個数と名前（コンテンツ内容）

表 1 Accordion に関するアクセシビリティを考慮した属性記述

UI 構成要素	動的状態	アクセシビリティの項目				属性記述			
		情報提示			操作関係	ROLE	STATE	PROPERTY	その他
		質的		量的					
全体	初期状態	UI の種類				accordion			
		UI の目的							alt 文字列
		構成要素	ヘッダ					aria-owns ヘッダの ID	
			パネル					パネルの ID	
ヘッダ	初期状態	名前				header		aria-controls パネルの ID	ID
								aria-label 文字列	title 文字列
	展開されているか						aria-expanded true/false		
					開閉方法				
	選択されているか						aria-selected true/false		
パネル部分	初期状態					panel			ID
	展開されているか						aria-hidden true/false		
複数の Accordion 全体	初期状態	UI の目的							alt 文字列
				Accordion の数				aria-setsize* 整数	
ヘッダ	初期状態			Accordion の番号				aria-posinset* 整数	
				最初の Accordion					
				最後の Accordion					

*定義と異なる使用をした属性

表 2 Tab Panel に関するアクセシビリティを考慮した属性記述

UI 構成要素	動的状態	アクセシビリティの項目				属性記述			
		情報提示			操作関係	ROLE	STATE	PROPERTY	その他
		質的		量的					
全体	初期状態	UI の種類				tablist			
		UI の目的							alt 文字列
		構成要素	タブ					aria-owns ID	
			パネル						
				タブの総数				aria-setsize 整数	
					タブ間の移動				
					タブとパネル間の移動				
タブ	初期状態	名前				tab			ID
								aria-label 文字列	title 文字列
		パネルとの対応関係		タブの番号				aria-posinset 整数	
								aria-controls パネルの ID	
	選択されているか						aria-selected true/false		
	表示されているか						aria-hidden true/false		
パネル						tabpanel			ID

4.4.3 操作関係

- タブ同士の移動
- タブとコンテンツの移動

4.4.4 アクセシビリティ項目の属性

列挙したアクセシビリティの観点を HTML 記述に反映させるために用いた属性を表 2 に示す．また，操作関係において，ショートカットキーやキーボードでの操作方法は現在の WAI-ARIA Authoring Practices で挙げられているもので良いと判断した．操作に必要な状態を把握するための属性記述を他のアクセシビリティの項目で示しているため，操作関係の属性値は空欄にしている．

4.5 HTML 属性記述の分類

属性記述の結果を対象 UI において共通であるアクセシビリティ項目，UI の目的，構成要素，コンテンツ外の要素との対応関係の 4 つの観点から分類を試みた．1 つ目のアクセシビリティ項目は，表に示していた質的・量的・操作関係があり，その UI において配慮すべき割合が多いものに分類した．2 つ目の UI の目的は，属性において UI の目的を設定すべきかどうかの観点である．このとき，設定せ

表 3　HTML 属性記述の分類結果

	UI	アクセシビリティの項目	UIの目的	構成要素	コンテンツ外要素との対応関係
ナビゲート	Site Navigator-General	量的	コンテンツ内容	ヘッダとパネル	あり
ナビゲート	Tree View	量的	コンテンツ内容	ヘッダとパネル	あり
ナビゲート	Wizard	量的	コンテンツ内容	ヘッダとパネル	あり
部品	Button	質的	UIの役割	自UIの集まり	あり
部品	Link	質的	UIの役割	自UIの集まり	あり
部品	Checkbox	量的	UIの役割	自UIの集まり	あり
部品	List Box	量的	UIの役割	自UIの集まり	あり
部品	Radio Group	量的	UIの役割	自UIの集まり	あり
操作	Drag&Drop	操作関係	不要	その他	あり
表示	Accordion	質的	コンテンツ内容	ヘッダとパネル	なし
表示	Dialog Modal	質的	コンテンツ内容	ヘッダとパネル	なし
表示	Tab Panel	質的	コンテンツ内容	ヘッダとパネル	なし
表示	Dialog Nonmodal	量的	コンテンツ内容	ヘッダとパネル	なし
表示	Grid	量的	コンテンツ内容	ヘッダとパネル	なし
表示	Alert	質的	不要	自UIの集まり	なし
表示	Tooltip	質的	不要	自UIの集まり	なし
入力	Autocomplete	量的	不要	他UIの集まり	なし
入力	ComboBox	量的	不要	他UIの集まり	なし
入力	Date Picker	量的	不要	他UIの集まり	なし
入力	Slider	量的	不要	他UIの集まり	なし
入力	SliderMulti-thumb	量的	不要	他UIの集まり	なし
入力	Spin Button	量的	不要	他UIの集まり	なし

ずとも目的が定まっているものを不要，設定しなければならない UI のうち，その UI を使用する目的がコンテンツを作成するためのものであればコンテンツ内容を，他の UI を制御するために用いられる UI には UI の役割とした．3 つ目の構成要素は，タイトル部分と本部分のような構成から成る「ヘッダとパネル」と自身が集まった「自 UI の集まり」，複数の UI が複合して成る「複数の UI の複合」，「その他」の 4 種類に分類した．4 つ目のコンテンツ外の要素との対応関係は，対象 UI が自身を含むコンテンツ領域外の UI と対応関係を持つかどうかから分類を試みた．分類した結果を表 3 に示す．

4.6　HTML 属性記述の系統化

表 3 に示した分類毎に HTML 属性記述の一般化を検討し，系統化を試みた．ただし，Drag& Drop はアクセシビリティの項目で操作関係，構成要素においてその他に分類されており，他の対象 UI とは異質な UI である．また，API が公開され，標準化の検討が十分にされている．そのため，記述の一般化についての議論は行わない．

Accordion と Tab Panel と Dialog Modal では，共通の属性が多い．しかし，Modal には aria-modal 属性を設定する必要があり，かつ重要である．ただ，この属性が必要となる UI は Dialog のみである．そのため，共通の属性記述にする際に障害が生じると判断し，Accordion と Tab Panel のみで記述の整理を行った．

Accodrion と Tab Panel は全体の要素とヘッダ（タブ），パネルの要素から構成されており，全体の要素では role，alt，aria-owns，aria-setsize 属性が共通であった．また，ヘッダでは，role，id，aria-controls，aria-label，aria-posinset，aria-expanded，aria-selected 属性が共通であった．パネルでは，role，id，aria-hidden 属性が共通であった．整理した属性記述を図 4 に示す．

5.　考察

5.1　対象 UI の種類と使用目的の属性記述

コンテンツ内の UI に対する操作や状態の変化が把握できることがアクセシビリティに繋がる．そのため，コンテンツ内の UI の種類を支援技術に伝えることが必要である．それが，UI の種類やその UI を構成する要素が WAI-ARIA の Role 属性に定義されている理由であると考えられる．したがって，Role 属性の値は対象 UI と同じものが与えられるはずである．しかし，以下に示す対象 UI において Role 属性は定義されていなかった．

- Accordion
- Autocomplete
- DatePicker
- Slider Multi-thumb
- Dialog Modal
- Dialog Nonmodal
- Wizard

また，Site Navigator において，navigation という Role 属性が定義されているが，使用される UI の種類は定まっていなかった．個々の対象 UI と Role 属性について考察を以下に示す．

5.1.1　Accordion

Accordion は Role 属性は定義されていないが，展開さ

```
<全体 role=”UI の種類” alt=”UI の目的” aria-owns=”ヘッダ ID パネル ID” aria-setsize=”ヘッダの総数”>
  <ヘッダ role=”header” id=” ヘッダ ID” aria-controls=”パネル ID” alia-selected=”true/false”
     aria-label=”ヘッダタイトル” aria-expanded=”true/false” aria-posinset=”番号”>タイトル</ヘッダ>
  <パネル id=”パネル ID” role=”panel” aria-hidden=”true/false”> </パネル>
</全体>
```

図 4 限られた領域で多くのコンテンツを提示する UI

れているかを表す aria-expanded 属性が定義されている．この属性を要素に与えることにより，Accordion のような折りたたんだり，展開する UI の状態が把握できる．そのため，Role 属性が定義されていないと考えられる．しかし，類似する Treeview は tree という Role 属性が存在する．Accordion もヘッダとパネルの要素を必ず必要とするため，Role 属性が必要であると考える．

5.1.2 Autocomplete

Autocomplete は WAI-ARIA Authoring Practices の Design Patterns において，Role 属性に combobox，aria-autocomplete 属性を付与するように記述されている．ただし，表示されるリストは入力に応じて変化するはずであるため，変化の通知を設定する aria-relevant 属性と aria-atomic 属性を必ず設定する必要がある．

5.1.3 Date Picker

Date Picker は Role 属性として定義されているべき UI であるが，定義されていない．HTML5 の INPUT 要素で作成することは可能であるが，JavaScript で作成する可能性は大いに考えられる．そのため，カレンダーを用いて日付を入力するような UI の Role 属性が定義されるべきである．

5.1.4 Dialog Modal/Nonmodal

Dialog Modal と NonModal は，Dialog の Role 属性があり，aria-modal がモーダルであるかを表す property 属性であるため，Dialog の Role 属性を 2 つ用意する必要がない．

5.1.5 Slider Multi-Thumb

Slider Multi-thumb は slider を Role 属性に設定することで UI の種類を判断することが可能である．ただし，これはあくまで Slider であり，サムが何個あるのか，それぞれのサムが何の値を設定するものなのかがわからない．そのような Property 属性を付与することで UI の種類や動作を判断することが可能である．

5.1.6 Wizard

Wizard は WAi-ARIA Authoring Practices の Design Patterns 内で Role，State，Property 属性が記述されていない．RIA において，進行状況に応じてページ遷移せずにコンテンツの内容が変化する UI は重要である．Design Pattern 内に挙げられている UI には Wizard 以外に工程に応じてコンテンツが変化する UI は存在しない．さらに，定義されている Role 属性で進行状況を表す Role は progressbar のみである．そのため，工程においてコンテンツが切り替わるような wizard という Role 属性が必要であると考える．

5.1.7 Site Navigator

UI の種類は Site Navigator に分類されるが，UI の表現部分には Tree View や Tab Panel 以外の対象 UI である可能性がある．その場合どちらを Role 属性に定義すれば良いのか混乱を招く可能性がある．そこで，HTML5 には新規に NAVIGATION 要素が追加された．Role 属性の変わりに NAV 要素で UI を作成し，Role 属性に使用する UI の種類を設定すれば良いと考えられる．

5.1.8 Role 属性と alt 属性の役割

WAI-ARIA で定められている Role 属性は，あくまでも支援技術に対してどのような動作を行う UI であるかを伝えるための情報であって，ユーザに UI の種類を伝えてもその UI がどのように使われているのかまではわからない．また，同じ UI であっても異なる使われ方であったり，異なる UI であっても使われ方が同じである可能性がある．これは，開発者は視覚的な見栄えを目的としてその UI を使用する可能性があるからである．現在の RIA では支援技術を用いてサービスを利用するユーザにはどのようなコンテンツなのかを理解するためには時間がかかる．そのため，UI にその UI を使用する目的（コンテンツ内容）を付与すべきである．

ユーザ視点であれば，Role が使用目的であるべきだが，UI の目的は開発者に依存し，全てを網羅した一般化は困難である．そのため，Role 属性は UI の種類を設定し，HTML の alt 属性に UI の目的（コンテンツ内容）を設定すれば良いと考える．ただし，開発者やユーザによっては Site Navigator のことを Menu であると認識している可能性がある．そのため，使用する用語の注意が必要である．

5.2 JavaScript ライブラリを用いたサポート

HTML 属性記述を分類・系統化した結果の活用として，図 5 に示すように，開発者による属性の記述や動的な属性変更の管理をサポートすることが考えられる．

Role などの属性値から UI を特定することで，その UI に必要な一部の属性を自動的に付与することができれば，開発者が記述する必要がなくなり，アクセシビリティへの対応の負担が軽減される．また，UI を特定し，その UI に

適用される CSS を解析することで，UI の視覚的な状態を把握することができれば，動的に変更されるべき属性値も自動的に付与することが可能である．さらに，UI ごとに属性値の切り替え処理を行う JavaScript をライブラリ化しておき，ページに組み込むことで，属性値の動的な変更処理を記述する必要もなくなる．

JavaScript ライブラリの開発においても，属性記述の分類・系統化の結果を活用することができる．UI の各分類に共通する処理と，個々の UI 特有の処理を認識することで，共通する処理をテンプレートとして用意し，UI 特有の処理をその中に組み入れることで，ライブラリの開発を効率化することができる．

これらのサポート技術により，WAI-ARIA やアクセシビリティに対する知識が少なくても，アクセシビリティの高いウェブアプリケーションを作成することが可能になる．

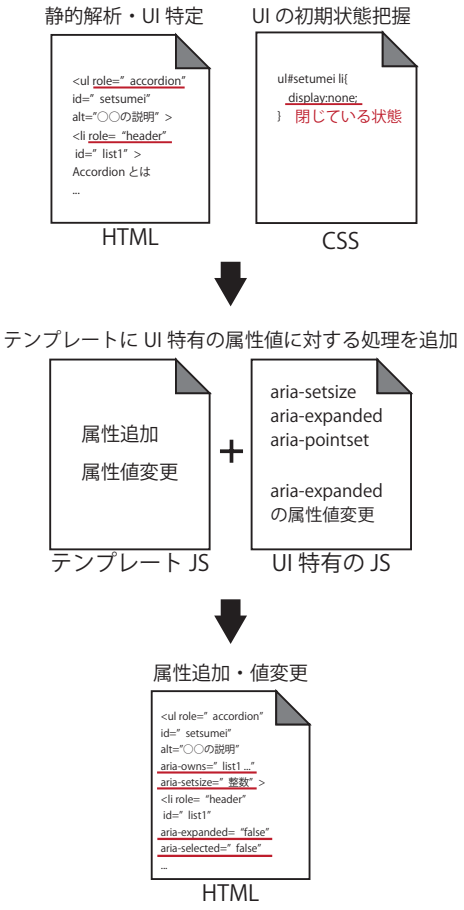

図 5 期待されるツールによるサポート

6. おわりに

本研究では，動的なコンテンツの状態を把握するための属性が HTML の標準技術となれば良いと考え，22 種類の UI を対象に動的な状態を把握するために必要な属性とその値を検討した．その後，作成した UI の HTML 属性記述の分類と系統化を行った．その結果，22 種類の UI を大きく 5 種類に分類し，分類毎に属性記述の共通性を整理し，対応する記述方法をまとめた．これにより，UI には Role 属性として UI の種類を設定し，alt 属性として UI の使用目的や役割を記述すべきであると考えられる．

本研究では，36 種類の UI のうち，他対象 UI が複数組み合わさって構成されるような UI やアプリケーション依存が強い UI は対象から除外した．しかし，それらの UI は RIA において重要な UI である．そのため，今後の課題として残りの UI をあわせて分類と系統化を行う必要がある．また，各 UI を作成する上で，使用する HTML 要素は考慮しなかった．しかし，標準化を目指すのであれば，UI を作成する上で最適な HTML 要素を調査し，属性記述に反映させる必要がある．さらに，本研究では HTML 属性記述のみを議論し，JavaScritpt での処理内容について議論しなかった．現状では開発者が 1 から処理内容を記述する必要があるため，HTML 要素を調査後，JavaScript ライブラリ作成も今後の課題である．

参考文献

[1] W3C/WAI, Accessible Rich Internet Applications (WAI-ARIA) 1.1, November 2015, <https://www.w3.org/TR/wai-aria-1.1/ >(参照日 2016.2.13).
[2] 富士通株式会社 総合デザインセンター著，よくわかるウェブ・アクセシビリティ＆ユーザビリティ誰もが使いやすいウェブサイトへ，FOM 出版 (2014).
[3] 山田 肇：ウェブアクセシビリティの標準化と普及への課題，科学技術動向，vol.122，pp.20-35(2011).
[4] W3C, Web Content Accessibility Guidelines(WCAG)2.0, December 2008 <https://www.w3.org/TR/2008/REC-WCAG20-20081211/ >(参照日 2016.2.12).
[5] 日本規格協会，JIS X 8341-3:2010：高齢者・障害者等配慮設計指針-情報通信における機器，ソフトウェア及びサービス-第 3 部：ウェブコンテンツ (2004 年 6 月制定，2010 年 8 月改正).
[6] 渡辺 隆行：ウェブ・アクセシビリティ向上の要件，インターネットコンファレンス (2005).
[7] 渡辺 昌行，橋本 遼，濱口 菜々，浅野 陽子：視覚障がい者視点のウェブデザイン，電子情報通信学会技術研究報告. HCS, ヒューマンコミュニケーション基礎，vol.113, No.426，pp.187-192(2014).
[8] 松田 理紗, 渡辺 隆行：JavaScript を利用した動的な Web のアクセシビリティ，電子情報通信学会技術研究報告. WIT, 福祉情報工学, vol.107, No.368, pp.111-118(2007).
[9] Marino Linaje, Adolfo Lozano-Tello, Miguel A. Perez-Toledano, Juan Carlos Preciado, Roberto Rodriguez-Echeverria, Fernando Sanchez-Figueroa：Providing RIA user interfaces with accessibility properties , Journal of Symbolic Computation, vol.46, No.2, pp207-217(2011).
[10] 人間工学（編）：ユニバーサルデザイン実践ガイドライン，共立出版株式会社 (2003).
[11] W3C/WAI, WAI-ARIA Authoring Practices 1.1, November 2015, <https://www.w3.org/TR/wai-aria-practices-1.1/ >(参照日 2015.12.21).