Accessibility Metadata Best Practices Guide

Version 1.0.1, December 29, 2013

Table of Contents

About this Guide

Although no prior knowledge of schema.org or semantic tagging is necessary in order to read this guide, to get the most from it you should have a working knowledge of these technologies.

The main body of this guide walks through the accessibility metadata and assumes a general understanding of microdata and schema.org. You can read through the body without any knowledge of the underlying technologies, but the language and examples may seem foreign.

For those completely new to these technologies, it is recommended that you start with the semantic markup primer. The primer covers only as much of the technologies as you need to know, without bogging you down in technical details. Once you're comfortable with the basic concepts, the tagging of the accessibility metadata will be much easier to understand.

Introduction

Semantic data and semantic markup are two concepts that go hand-in-hand. Semantic data is the end goal of everyone who maintains content on the web—rich, meaningful data that can be processed by search engines and browsers—while semantic markup is the means by which this information is layered into HTML content. Both are at the heart of making digital content accessible, as it is only through meaningful structure that otherwise arbitrary strings of text begin to assume meaning.

The simplicity of markup that underlies the web makes it difficult for machines to extract meaning, however. Search engines are adept at matching the text string you enter against text strings found in web content, but they're still a long way from inferring the meaning of those strings. Language is, put simply, complex.

The goal of semantically enriching content is to provide more nuanced information that can simplify processing and making connections. The RDFa and Microdata technologies allow content authors to clearly identify properties of the content that might otherwise be misinterpreted, or not stated at all. The machine still may not comprehend the text in the way a human does, but with these semantics the programmers behind those tools can extract and present the information in meaningful ways.

And that's where semantics and accessibility intersect in the realm of internet searching. The richer the data is, the better it can be presented to users with differing needs and preferences. No more having to access each resource in order to manually discover its nature.

Not surprisingly, then, the goal of the Accessibility Metadata Project is to improve the results that search engines can provide by defining semantic properties within schema.org, the new framework for expressing search semantics. The project does not attempt to address the underlying (in)accessibility of formats and protocols on the web; it only seeks to help users in the rapid discovery of resources that meet their needs.

By identifying accessibility features (enhanced content to enable multiple modalities) and other key aspects of the content, search engines can accurately relay the nature of these resources on to users. The ambiguity of searching by keywords—where the presence of search words on a page, regardless of context, determines relevance—will finally be taken out of the equation.

Overview

Properties

The Accessibility Metadata Project contributed a set of four properties to the schema.org CreativeWork type:

- `accessibilityFeature`
- `accessibilityHazard`
- `accessibilityControl`
- `accessibilityAPI`

By adding the properties to this super-type in the schema.org hierarchy, they are inherited by all the content specializations that are commonly sought out for compatibility, such as Book, Article, Blog and Movie. They will also be available in any future subtypes of CreativeWork.

These properties provide a basis for augmenting web content with accessibility information. Their use is not limited only to what lives on the web; you can use them to describe any resource that is referenced from a web page, as well. For example, the metadata can be used to enhance an online library catalogue of available works, even though the works themselves may only be available physically.

This guide reviews each of these properties, providing guidance on how to use them to make accessibility statements about your content.

Values

The property values listed in this guide are recommended for use at the time of writing, but are not intended to be read as a closed list; new values can be added at any time. To view the most up-to-date value list, please refer to the W3C Web Schemas wiki.

If you need to use a value that is not yet defined, there is no restriction against defining your own. You are encouraged, however, to bring missing values to the attention of the working group using the google groups mailing list for consideration of inclusion in the formal set.

You will also notice as you read through this guide that more than one property value may apply to the content you are describing. This is perfectly normal, as there is no limit to the number of values you can use.

When including multiple media features, the one rule you must follow is to tag each separately. For example, a textbook that includes alt text and descriptions for images, transcripts for audio

and video content, and MathML equations, might include all of the following accessibility features:

```
<div itemscope="" itemtype="http://schema.org/Book">
       <meta itemprop="accessibilityFeature" content="alternativeText"/>
       <meta itemprop="accessibilityFeature" content="longDescription"/>
       <meta itemprop="accessibilityFeature" content="transcript"/>
       <meta itemprop="accessibilityFeature" content="MathML"/>

       <h1 itemprop="name">Introduction to Math</h1>
       ...
</div>
```

Another convention you will find in this guide is the subclassification of values using slashes ('x/y'). For example, the `braille` value for `accessibilityFeature` provides only a general indication that braille resources are available. To be more precise about whether the braille is contracted or uncontracted, you could append that detail to the value as `braille/grade1`.

This guide covers the most common values for extension, and provides guidelines on how to extend them, but you can extend any value. You should choose extensions over creating new top-level values whenever you need more precision than an existing value provides. But note that using extension values is not required, and unless the additional information is critical to discovery it is better to avoid them.

The accessibilityFeature property

The first property we'll look at is the most fundamental for expressing the accessible qualities of your content: `accessibilityFeature`. This property describes the content features of the resource that make it accessible to a broader range of users.

Although the richer content being provided may seem obvious, it's not necessarily the case that a search engine will be able to understand or expose this information for users. For example, the HTML5 `video` element allows you to attach caption tracks. In theory, a search engine could inspect the element and discover if such a track were present. The problem is that search engines don't perform these kinds of manual tests, and in many cases the presence of captions will not be apparent from the markup (e.g., in flash-based video).

Likewise, a web page may have MathML markup directly embedded in it, but to find the same in an ebook would require digging into the packaging. The very point of this accessibility metadata is to provide an easily identifiable set of information, so when you include any accessibility feature make sure to make note of it using this property.

It is not a requirement that the accessibility feature be located on the same page as the resource, but users must have easy access to the feature in order for it to be listed. For

example, you could provide a transcript for a speech separately from the page hosting the recording provided a link to the transcript is included.

The following subsections detail the predefined values for this property.

alternativeText

The `alternativeText` value indicates that alternative text is provided for images.

For HTML-based content, alternative text is provided in the `img` element's `alt` attribute. Other formats, like Microsoft Word, have similar proprietary mechanisms for annotating images with alternate text.

A judgement call may be necessary when setting this value if only some of the visual content includes alternative text. The schema.org metadata does not include completeness of coverage, so you will have to determine whether the lack of alternative text represents a significant barrier to comprehension. If it does, do not set the property.

annotations

The `annotations` value indicates that the work is annotated. Annotations may be an intrinsic part of the work (i.e., included by the author, translator or similar) or could be included by an instructor to provide additional guidance or explanation.

This value can also be used to identify annotations for a work independent of the work (e.g., sold commercially).

audioControl

The `audioControl` value indicates that the user has full control over audio playback. When setting this value, the user must have control over play, pause, stop, resume, movement forward and backward through the timeline. Control of the volume, independent of the operating system, is also necessary.

Control is required for audio that plays in the foreground and the background. For background sounds (e.g., mood music), the ability to disable playback is the only necessary feature, but control of the volume is strongly suggested.

audioDescription

The `audioDescription` value indicates that audio description tracks are available for video content. Audio descriptions are an additional track that provides context to the visual component of the content (e.g., explaining actions going on in a video between instances of dialogue).

You can attach audio descriptions to HTML5 `video` elements using the `track` element. It is also possible to synchronize them with a video using SMIL.

For more information, please see WCAG Technique G78.

bookmarks

The `bookmarks` value indicates that bookmarks to facilitate access to key locations in the work are included. Bookmarks are typically only found in electronic books and documents.

braille

The `braille` value indicates that braille content or alternatives are available.

You can extend this property to include more information about the precise nature of the braille provided. For example:

- `braille/grade1` (uncontracted braille)
- `braille/grade2` (contracted braille)
- `braille/ascii` (ASCII-encoded 6 dot braille)

The code to which the braille conforms can also be specified:

- `braille/BANA` (Braille Authority of North America)
- `braille/SEB` (Standard English Braille)
- `braille/UEB` (Unified English Braille)

And it is also possible to indicate features of the braille content:

- `braille/nemeth` (Nemeth-encoded math)
- `braille/music`
- `braille/chemistry` (Braille Code for Chemical Notation)

The Accessibility Metadata project working group has not attempted to enumerate all possible subclassifications, as these are better developed by braille authorities and producers.

captions

The `captions` value indicates that captions are available for audio and video content.

This setting does not guarantee that captions will be viewable in all user agents, as support for closed captions using the `track` element is not yet universal in HTML5-capable browsers.

ChemML

The `ChemML` value indicates that the content includes chemical information encoded in the ChemML markup language.

displayTransformability

The `displayTransformability` value provides a general indication that aspects of the rendering can be modified by the user, whether by the application of custom style sheets or through controls in the application.

You should set this property only when the ability to change the styles will have a meaningful impact on the overall accessibility of the content. For example, the ability to change fonts or the background and foreground colors is typically not helpful when the content is image-based.

If only certain properties are modifiable, or will meaningfully impact on the accessibility, it is possible to extend this value using specific CSS property names:

- `displayTransformability/font-size`
- `displayTransformability/background-color`

(Note that the use of CSS property names is for standardization only; it is not necessary that CSS be the formatting technology used.)

highContrastAudio

The `highContrastAudio` value indicates that there is high contrast between background noise and foreground human speech in pre-recorded audio.

Additional refinements can be made to this property to qualify the level of background noise as follows:

- enhancedAudio/noBackground - indicate that there is no background noise that will interfere with the foreground speech
- enhancedAudio/reducedBackground - indicates that there is at least 20db difference between the foreground speech and the background noise
- enhancedAudio/switchableBackground - indicates that the background noise can be turned off

For more information, please see WCAG Understanding SC 1.4.6.

highContrastDisplay

The `highContrastDisplay` value indicates that the content meets minimum high-contrast display requirements defined in WCAG SC 1.4.3.

If the content does not meet requirements for high contrast reading, but the format of the content allows the foreground and background colors to be modified (e.g., through CSS), or alternate high-contrast options are provided, set the `displayTransformability` value instead.

index

The `index` value indicates that an index to the work is available.

If you include more than one index, you can optionally identify the specific type(s) using the extension mechanism. For example, `index/topical` is used to identify a topical index.

largePrint

The `largePrint` value indicates that the resource is formatted for large print reading.

This value must not be set when the font size can be increased either through zooming or CSS manipulation. It is used only to indicate that the resource was designed for large print reading.

latex

The `latex` value indicates that math content is provided in LaTeX format.

longDescription

The `longDescription` value indicates that descriptions are provided for image-based content and/or complex structures such as tables.

MathML

The `MathML` value indicates that math content is provided in MathML format.

printPageNumbers

The `printPageNumbers` value is only applicable for electronic books with print equivalents. The value indicates that page markers are included in the work, allowing users to locate print page equivalent locations (e.g., for mixed print-digital environments such as classrooms or book clubs).

readingOrder

The `readingOrder` value is primarily applicable to electronic books and indicates that the markup is structured to facilitate access to the logical reading order. Only set this value if sidebars, figures and other secondary material has been properly tagged using HTML5 elements that allow the content to be automatically skipped or manually escaped from.

signLanguage

The `signLanguage` value indicates that sign language translation is available for aural content.

structuralNavigation

The `structuralNavigation` value indicates that the content is structured for easier navigation (e.g., using headings), a table of contents is available or similar high-level document navigation is provided.

tableOfContents

The `tableOfContents` value indicates that a complete table of contents to the work is provided. Do not set this property if only a reduced table of contents is provided.

taggedPDF

The `taggedPDF` value indicates that a PDF is tagged so that structures such as headings, lists and tables can be identified.

tactileGraphic

The `tactileGraphic` value indicates that tactile graphics are included. For example, as described in the BANA Guidelines and Standards for Tactile Graphics.

tactileObject

The `tactileObject` value indicates that tactile 3D objects are included, or instructions to build them are available.

timingControl

The `timingControl` value indicates that timed interfaces are controllable by the user. For example, users who require more time to complete a task are able to reset the timer or pause the countdown while seeking assistance.

transcript

The `transcript` value indicates that transcripts are provided for audio and video content.

unlocked

The `unlocked` value indicates that no digital rights management or other content restriction protocols have been applied to content that would impede access.

videoControl

The `videoControl` value indicates that the user has full control over video playback. When setting this value, the user must have control over play, pause, stop, resume, movement forward and backward through the timeline. Control of the volume, independent of the operating system, is also necessary.

The accessibilityHazard property

Some digital media can be physically dangerous to those who access it. For example, rates of flashing faster than 3Hz (3 times per second) can cause seizures, as can loud repetitive sounds. Nausea is another hazard that can be brought on by motion simulation in visual resources, such as 3D gaming emulated on the HTML `canvas` element.

When a resource is known to contain such physiological hazards, the user needs to discover their presence before accessing the content. Although adding warnings to the page may seem sufficient, such warnings are easily missed by users and difficult to extract by search engines.

The `accessibilityHazard` property allows these dangers to be identified so that a search engine can report the hazard before the user attempts to access the resource. It has three predefined values to account for the above-mentioned situations:

- `flashing`
- `motionSimulation`
- `sound`

For example, a video could be identified as having a flashing hazard as follows:

```
<div itemtype="http://schema.org/VideoObject" itemscope="">
      <p>Warning: This video contains
         <span itemprop="accessibilityHazard">flashing</span> that may
         cause seizures in some people.</p>
      <video src="rapidFire.webm"/>
</div>
```

Note that providing access to an alternative less hazardous adaptation does not change the need to indicate that the hazard exists.

Adding the hazards when they exist is useful, but there is also a question of how to know whether the hazards don't exist or whether the resource hasn't been checked. To indicate that the content has been checked and is safe, the opposite values are included:

- `noFlashingHazard`
- `noMotionSimulationHazard`
- `noSoundHazard`

The previous example can now be fleshed out to indicate that while it has a flashing hazard, it does not have sound or motion simulation hazards:

```
<div itemtype="http://schema.org/VideoObject" itemscope="">
       <meta property="accessibilityHazard" content="noMotionSimulationHazard"/>
       <meta property="accessibilityHazard" content="noSoundHazard"/>
       <p>Warning: This video contains
          <span itemprop="accessibilityHazard">flashing</span> that may
          cause seizures in some people.</p>
       <video src="rapidFire.webm"/>
</div>
```

The accessibilityControl property

If you take the time to ensure that your content can be accessed effectively by users with different preferred modalities, you should also indicate such compatibility in your metadata. The `accessibilityControl` property enables the tagging of this information and has five predefined values:

- `fullKeyboardControl`
- `fullMouseControl`
- `fullSwitchControl`
- `fullTouchControl`
- `fullVoiceControl`

This property identifies which input methods allow access to all of the application functionality. Multiple methods can be specified, as appropriate.

A good measure for keyboard control is WCAG 2.0 Guideline 2.1. If your content is compliant to this guideline, you can safely set this control flexibility in your metadata:

```
<meta itemprop="accessibilityControl" content="fullKeyboardControl"/>
```

A similar guideline for other forms of accessibility does not exist, but the requirement is generally the same in terms of the user being able to effectively access and control your content by one of these methods alone:

```
<meta itemprop="accessibilityControl" content="fullMouseControl"/>
<meta itemprop="accessibilityControl" content="fullSwitchControl"/>
<meta itemprop="accessibilityControl" content="fullTouchControl"/>
<meta itemprop="accessibilityControl" content="fullVoiceControl"/>
```

If you are not certain if your content is accessible by an input method, do not assume it must be true because the content is accessible by another (e.g., mouse access does not translate into keyboard access). If your content has any scripted controls or dynamic features, you will need to

test to ensure operability.

Note that the need to input text does not necessarily make content inaccessible by non-keyboard methods. Users may have an on-screen keyboard available, for example.

The accessibilityAPI property

While you may have been led to believe so far that accessibility metadata applies only to media objects (books, video, etc.), it also applies to programs, whether mobile apps, flash content, Java applets or HTML pages. Noting whether this content is compatible with accessibility APIs is important because it allows users to determine whether they will be able to interact with it via the device they're using:

```
<meta itemprop="accessibilityAPI" content="MSAA"/>
```

This property includes the following predefined API values:

- `AndroidAccessibility`
- `ARIA`
- `ATK`
- `AT-SPI`
- `BlackberryAccessibility`
- `iAccessible2`
- `iOSAccessibility`
- `JavaAccessibility`
- `MacOSXAccessibility`
- `MSAA`
- `UIAutomation`

To create a more complete picture, you can also pair this property with the `operatingSystem` property from the SoftwareApplication type, especially if the content is known to only work on certain systems that support the accessibility API:

```
<span itemprop="operatingSystem"
      itemtype="http://schema.org/SoftwareApplication"
      itemscope="">Windows 7</span>
```

In general, you should not need to set the OS-based APIs for HTML, but `JavaAccessibility` and `ARIA` should be set when appropriate.

This property is useful when you are describing a page that uses proprietary plugins, or when

you are describing the accessibility of non-web formats, where such formats may only work on certain platforms.

Book Formats

The accessibility metadata proposal also defines the following extensions to the BookFormatType enumeration:

- `EBook/DAISY202`
- `EBook/DAISY3`
- `EBook/EPUB2`
- `EBook/EPUB3`

These formats provide information that might otherwise not be easily obtainable. For example, EPUB files are not distinguishable from their file extensions, but require looking into the packaging to find out which version the file conforms to. Likewise, DAISY books often are zipped up for online distribution, again making it hard to determine their nature without loading them or looking into the content.

An online library could, for example, indicate that a resource is a DAISY 2.02 book by setting this property as follows:

```
<link itemprop=”bookFormat” href=”http://schema.org/EBook”DAISY202/>
```

Appendix A – Quick-Start Templates

The following basic templates can be used to quickly apply the accessibility metadata to some common creative works.

Note that the use of `meta` tags is not required, but done for simplicity. If the accessible nature of the content can be exposed as text content, the meta tags should be reformulated to a more appropriate form.

Video

The only accessibility property that must be set for all videos is the `visual` access mode. All other metadata is conditional.

```
<body itemscope="" itemtype="http://schema.org/VideoObject">
      <!-- if captions have been provided: -->
      <meta itemprop="accessibilityFeature" content="captions"/>

      <!-- if a transcript is available: -->
      <meta itemprop="accessibilityFeature" content="transcript"/>

      <!-- if an audio description for the soundtrack: -->
      <meta itemprop="accessibilityFeature" content="audioDescription"/>

      <!-- if sign language transcription provided: -->
      <meta itemprop="accessibilityFeature" content="signLanguage"/>

      <!-- if a keyboard accessible player: -->
      <meta itemprop="accessibilityControl" content="fullKeyboardControl"/>

      <!-- if a mouse accessible player: -->
      <meta itemprop="accessibilityControl" content="fullMouseControl"/>

      <!-- if a flashing hazard in the video: -->
      <meta itemprop="accessibilityHazard" content="flashing"/>
      <!-- or if no flashing hazard in the video: -->
      <meta itemprop="accessibilityHazard" content="noFlashingHazard"/>

      <!-- if a motion hazard in the video: -->
      <meta itemprop="accessibilityHazard" content="motionSimulation"/>
      <!-- or if no motion hazard in the video: -->
      <meta itemprop="accessibilityHazard" content="noMotionSimulationHazard"/>

      <!-- if a sound hazard in the video: -->
```

```
        <meta itemprop=”accessibilityHazard” content=”sound”/>
        <!-- or if no sound hazard in the video: -->
        <meta itemprop=”accessibilityHazard” content=”noSoundHazard”/>

        <video .../>
</body>
```

Audio

The only accessibility property that must be set for all audio clips is the `auditory` access mode. All other metadata is conditional.

```
<body itemscope=”” itemtype=”http://schema.org/AudioObject”>
        <!-- if captions have been provided: -->
        <meta itemprop=”accessibilityFeature” content=”captions”/>

        <!-- if a transcript is available: -->
        <meta itemprop=”accessibilityFeature” content=”transcript”/>

        <!-- if an audio description for the soundtrack: -->
        <meta itemprop=”accessibilityFeature” content=”audioDescription”/>

        <!-- if sign language transcription provided: -->
        <meta itemprop=”accessibilityFeature” content=”signLanguage”/>

        <!-- if a keyboard accessible player: -->
        <meta itemprop=”accessibilityControl” content=”fullKeyboardControl”/>

        <!-- if a mouse accessible player: -->
        <meta itemprop=”accessibilityControl” content=”fullMouseControl”/>

        <!-- if a sound hazard: -->
        <meta itemprop=”accessibilityHazard” content=”sound”/>
        <!-- or if no sound hazard: -->
        <meta itemprop=”accessibilityHazard” content=”noSoundHazard”/>

        <!-- no other hazards: -->
        <meta itemprop=”accessibilityHazard” content=”noFlashingHazard”/>
        <meta itemprop=”accessibilityHazard” content=”noMotionSimulationHazard”/>

        <audio .../>
</body>
```