TECHNICAL SPECIFICATION

ISO/IEC TS 30135-3

First edition
2014-11-15

Information technology — Digital publishing — EPUB3 —

Part 3: Content Documents

Technologies de l'information — Publications numériques — EPUB3 —

Partie 3: Documents de contenu

Reference number
ISO/IEC TS 30135-3:2014(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

In other circumstances, particularly when there is an urgent market requirement for such documents, the joint technical committee may decide to publish an ISO/IEC Technical Specification (ISO/IEC TS), which represents an agreement between the members of the joint technical committee and is accepted for publication if it is approved by 2/3 of the members of the committee casting a vote.

An ISO/IEC TS is reviewed after three years in order to decide whether it will be confirmed for a further three years, revised to become an International Standard, or withdrawn. If the ISO/IEC TS is confirmed, it is reviewed again after a further three years, at which time it must either be transformed into an International Standard or be withdrawn.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC TS 30135 series were prepared by Korean Agency for Technology and Standards (as KS X 6070 series) with International Digital Publishing Forum and were adopted, under a special “fast-track procedure”, by Joint Technical Committee ISO/IEC JTC 1, Information technology, in parallel with its approval by the national bodies of ISO and IEC.

ISO/IEC TS 30135 consists of the following parts, under the general title *Information technology — Document description and processing languages — EPUB 3:*

- *Part 1: Overview*
- *Part 2: Publications*
- *Part 3: Content Documents*
- *Part 4: Open Container Format*
- *Part 5: Media Overlay*
- *Part 6: Canonical Fragment Identifier*
- *Part 7: Fixed-Layout Documents*

EPUB Content Documents 3.0

Recommended Specification 11 October 2011

THIS VERSION

http://www.idpf.org/epub/30/spec/epub30-contentdocs-20111011.html

LATEST VERSION

http://www.idpf.org/epub/30/spec/epub30-contentdocs.html

PREVIOUS VERSION

http://www.idpf.org/epub/30/spec/epub30-contentdocs-20110908.html

A diff of changes from the previous draft is available at this link.

Please refer to the errata for this document, which may include some normative corrections.

Editors

Markus Gylling, DAISY Consortium

William McCoy, International Digital Publishing Forum (IDPF)

Elika J. Etemad, Invited Expert

Matt Garrish, Invited Expert

TABLE OF CONTENTS

1 Overview

1.1 Purpose and Scope

This section is informative

This specification, EPUB Content Documents 3.0, defines profiles of HTML5, SVG, and CSS for use in the context of EPUB® Publications.

This specification is one of a family of related specifications that compose EPUB 3, the third major revision of an interchange and delivery format for digital publications based on XML and Web Standards. It is meant to be read and understood in concert with the other specifications that make up EPUB 3:

- The EPUB 3 Overview [EPUB3Overview], which provides an informative overview of EPUB and a roadmap to the rest of the EPUB 3 documents. The Overview should be read first.
- EPUB Publications 3.0 [Publications30], which defines publication-level semantics and overarching conformance requirements for EPUB Publications.
- EPUB Open Container Format (OCF) 3.0 [OCF3], which defines a file format and processing model for encapsulating a set of related resources into a single-file (ZIP) EPUB Container.
- EPUB Media Overlays 3.0 [MediaOverlays30], which defines a format and a processing model for synchronization of text and audio.

This specification supersedes Open Publication Structure (OPS) 2.0.1 [OPS2]. Refer to [EPUB3Changes] for information on differences between this specification and its predecessor.

1.2 Relationship to Other Specifications

This section is informative

1.2.1 Relationship to HTML5

The XHTML document type defined by this specification is based on W3C [HTML5], and inherits all definitions of semantics, structure and processing behaviors from the HTML5 specification unless otherwise specified.

In addition, this specification defines a set of extensions to the W3C HTML5 document model that Authors may include in XHTML Content Documents.

This specification defines a simplified processing model that does not require Reading Systems to support scripting, HTML5 forms or the HTML5 DOM. EPUB Reading Systems conformant with this specification are only required to be able to process a conforming EPUB Content Document. As support for scripting and HTML5 forms are optional Reading System features, a conformant Reading System might not be a fully-conformant HTML5 User Agent (i.e., it might not implement the complete HTML5 processing model).

› 1.2.2 Relationship to SVG

This specification defines a restricted subset of SVG 1.1 to represent vector graphics inline in XHTML Content Documents and as standalone SVG Content Documents.

› 1.2.3 Relationship to CSS

The CSS profile defined in this specification has CSS 2.1 [CSS2.1] as its baseline. Any CSS Style Sheet that conforms to CSS 2.1 may be used in the context of an EPUB Publication, except as noted in CSS 2.1.

This specification also incorporates features defined by CSS3 Modules and introduces EPUB-specific CSS constructs.

› 1.2.4 EPUB 3 Versioning Strategy

EPUB 3 references W3C specifications that are not yet final, and incompatible changes to them may occur in the future that would cause EPUB 3 Content Documents that were previously conformant to no longer be conformant to the latest versions of the referenced specifications.

The IDPF anticipates revising the EPUB 3 specifications if and when such incompatible changes occur, updating the normative constraints defined herein as necessary and incrementing the minor version number of EPUB 3 (e.g., publishing an EPUB 3.0.n).

› 1.3 Terminology

EPUB Publication (or Publication)

A logical document entity consisting of a set of interrelated resources and packaged in an EPUB Container, as defined by this specification and its sibling specifications.

Publication Resource

A resource that contains content or instructions that contribute to the logic and rendering of the EPUB Publication. In the absence of this resource, the Publication might not render as intended by the Author. Examples of Publication Resources include the Package Document, EPUB Content Documents, EPUB Style Sheets, audio, video, images, embedded fonts and scripts.

With the exception of the Package Document itself, Publication Resources must be listed in the manifest [Publications30] and must be bundled in the EPUB container file unless specified otherwise in Publication Resource Locations [Publications30].

Examples of resources that are not Publication Resources include those identified by the Package Document link [Publications30] element and those identified in outbound hyperlinks that resolve outside the EPUB Container (e.g., referenced from an [HTML5] a element href attribute).

Core Media Type Resource

A Publication Resource that is a Core Media Type and may therefore be included in the EPUB Publication without the provision of fallbacks [Publications30].

EPUB Content Document

A Publication Resource that conforms to one of the EPUB Content Document definitions (XHTML or SVG).

An EPUB Content Document is a Core Media Type, and may therefore be included in the EPUB Publication without the provision of fallbacks [Publications30].

XHTML Content Document

An EPUB Content Document conforming to the profile of [HTML5] defined in XHTML Content Documents.

XHTML Content Documents use the XHTML syntax of [HTML5].

SVG Content Document

An EPUB Content Document conforming to the constraints expressed in SVG Content Documents.

EPUB Navigation Document

A specialization of the XHTML Content Document, containing human- and machine-readable global navigation information, conforming to the constraints expressed in EPUB Navigation Documents.

Scripted Content Document

An EPUB Content Document that includes scripting or an XHTML Content Document that contains HTML5 forms elements.

Refer to Scripted Content Documents for more information.

Top-level Content Document

An EPUB Content Document referenced directly from the spine

Core Media Type

A set of Publication Resource types for which no fallback is required. Refer to Publication Resources [Publications30] for more information.

Package Document

A Publication Resource carrying bibliographical and structural metadata about the EPUB Publication, as defined in Package Documents [Publications30].

Manifest

A list of all Publication Resources that constitute the EPUB Publication.

Refer to manifest [Publications30] for more information.

Spine

An ordered list of Publication Resources, typically EPUB Content Documents, representing the default reading order of the Publication.

Refer to spine [Publications30] for more information.

Text-to-Speech (TTS)

The rendering of the textual content of an EPUB Publication as artificial human speech using a synthesized voice.

EPUB Style Sheet (or Style Sheet)

A CSS Style Sheet conforming to the CSS profile defined in *EPUB Style Sheets*.

Viewport

The region of an EPUB Reading System in which the content of an EPUB Publication is rendered visually to a User.

CSS Viewport

A Viewport capable of displaying CSS-styled content.

SVG Viewport

A Viewport capable of displaying SVG images.

EPUB Container (or Container)

The ZIP-based packaging and distribution format for EPUB Publications defined in [OCF3].

Author

The person(s) or organization responsible for the creation of an EPUB Publication, which is not necessarily the creator of the content and resources it contains.

User

An individual that consumes an EPUB Publication using an EPUB Reading System.

EPUB Reading System (or Reading System)

A system that processes EPUB Publications for presentation to a User in a manner conformant with this specification and its sibling specifications.

› 1.4 Conformance Statements

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

All sections of this specification are normative except where identified by the informative status label "This section is informative". The application of informative status to sections and appendices applies to all child content and subsections they may contain.

All examples in this specification are informative.

› 1.5 Namespace prefix mappings

For convenience, the following namespace prefix mappings [XMLNS] are used throughout this specification:

prefix	namespace URI
epub	http://www.idpf.org/2007/ops
m	http://www.w3.org/1998/Math/MathML
pls	http://www.w3.org/2005/01/pronunciation-lexicon
ssml	http://www.w3.org/2001/10/synthesis
svg	http://www.w3.org/2000/svg

2 EPUB Content Documents

2.1 XHTML Content Documents

This section defines a profile of [HTML5] for creating XHTML Content Documents. An instance of an XML document that conforms to this profile is a Core Media Type and is referred to in this specification and its sibling specifications as an XHTML Content Document.

Unless otherwise specified, this specification inherits all definitions of semantics, structure and processing behaviors from the [HTML5] specification.

CAUTION

The EPUB 3 XHTML Content Document definition references features in the W3C [HTML5] specification that are still works in progress and may change in incompatible ways. When utilizing such features, authors should consider the inherent risks in terms of the potential impact on interoperability and document longevity.

2.1.1 Content Conformance

An XHTML Content Document must meet all of the following criteria:

Document Properties

- It must meet the conformance constraints for XML documents defined in XML Conformance [Publications30].
- It must use the XHTML syntax [HTML5].
- It must be valid to the XHTML Content Document schema as defined in XHTML Content Document Schema.
- For all document constructs used that are defined by [HTML5], it must conform to the conformance criteria defined for those constructs in that specification, unless explicitly overridden in HTML5 Deviations and Constraints.
- It must conform to all content conformance constraints defined in HTML5 Extensions and Enhancements.

File Properties

- The XHTML Content Document filename should use the file extension `.xhtml`.

NOTE

All Publication Resources referenced from an XHTML Content Document must conform to the constraints for Publication Resources defined in EPUB Publication — Content Conformance [Publications30]

2.1.2 Reading System Conformance

A conformant EPUB Reading System must meet all of the following criteria for processing XHTML Content Documents:

- Unless explicitly defined by this specification or its sibling specifications as overridden, it must process XHTML Content Documents using semantics defined by the [HTML5] specification and honor any applicable User Agent conformance constraints expressed therein.
- It must meet all Reading System conformance criteria defined in HTML5 Extensions and Enhancements.
- It must recognize and adapt behaviorally to the constraints defined in HTML5 Deviations and Constraints.
- It must meet the Reading System conformance criteria defined in Scripted Content Documents — Reading System Conformance.
- It must support visual rendering of XHTML Content Documents as defined in EPUB Style Sheets — Reading System Conformance.
- It should recognize embedded ARIA markup and support exposure of any given ARIA roles, states and properties to platform accessibility APIs [WAI-ARIA].

2.1.3 HTML5 Extensions and Enhancements

This section defines EPUB 3 XHTML Content Document extensions to the underlying [HTML5] document model.

2.1.3.1 Semantic Inflection

2.1.3.1.1 Introduction

This section is informative

Semantic inflection is the process of attaching additional meaning about the specific purpose and/or nature an element plays in an XHTML Content Document. In the context of EPUB Publications, the `epub:type` attribute is typically used to express domain-specific semantics, with the inflection(s) it carries complementing the underlying [HTML5] host vocabulary. The applied semantics always refine the meaning of their containing elements, never override their nature (e.g., the attribute can be used to indicate a `section` is a chapter in a work, but cannot be used to turn `p` elements into list items to avoid proper list structures).

Semantic metadata is not intended for human consumption; it instead provides a controlled way for Reading Systems and other User Agents to learn more about the structure and content of a document, providing them the opportunity to enhance the reading experience for Users.

This specification defines a method for semantic inflection using *the attribute axis*: instead of adding new XML elements to the XHTML Content Document vocabulary, the `epub:type` attribute can be appended to

existing elements to inflect the desired semantics. A mechanism to identify external vocabularies that provide controlled values for the attributes is also defined.

› 2.1.3.1.2 The `epub:type` Attribute

The `epub:type` attribute inflects semantics on the element on which it appears. Its value is one or more space-separated terms stemming from external vocabularies associated with the document instance, as defined in Vocabulary Association.

The inflected semantic must express a subclass of the semantic of the carrying element. In the case of semantically neutral elements (such as [HTML5] `div` and `span`), the inflected semantic must not attach a meaning that is already conveyed by an existing element (e.g., that a `div` represents a paragraph or section). Reading Systems must ignore inflected semantics that conflict with the carrying element.

NOTE

The `epub:type` attribute is intended to be functionally equivalent to the W3C Role Attribute [Role], but with restrictions as specified in Vocabulary Association.

Attribute Name

`type`

Namespace

`http://www.idpf.org/2007/ops`

Usage

May be specified on all elements.

Value

A space-separated list of property [Publications30] values, with restrictions as defined in Vocabulary Association.

› 2.1.3.1.3 Vocabulary Association

This specification adopts the vocabulary association mechanisms defined in Vocabulary Association Mechanisms [Publications30], with the following modifications:

Default Vocabulary

The default vocabulary for Content Documents is defined to be the EPUB 3 Structural Semantics Vocabulary.

Reserved Vocabularies

This specification does not reserve any prefixes.

The `prefix` Attribute

The `prefix` attribute definition is unchanged, but the attribute is defined to be in the namespace `http://www.idpf.org/2007/ops` when used in Content Documents.

Examples

The following example shows the `epub:type` *attribute used to inflect footnote and note reference semantics. The properties used are defined in the default vocabulary.*

```
<html … xmlns:epub="http://www.idpf.org/2007/ops">
    …
    <p> … <a epub:type="noteref" href="#n1">1</a> … </p>
    …
    <aside epub:type="footnote" id="n1">
        …
    </aside>
    …
</html>
```

The following example shows the `epub:type` *attribute used to inflect glossary semantics on an HTML5 definition list. The property used is defined in the default vocabulary.*

```
<html … xmlns:epub="http://www.idpf.org/2007/ops">
    …
    <dl epub:type="glossary">
        …
    </dl>
    …
</html>
```

The following example shows the `epub:type` *attribute used to inflect source publication pagebreak semantics. The property used is defined in the default vocabulary. (Note that the dc:source [Publications30] element provides a means of identifying the source publication to which the given pagination information applies.)*

```
<html … xmlns:epub="http://www.idpf.org/2007/ops">
   …
  <p> … <span epub:type="pagebreak" title="234"/> … </p>
   …
</html>
```

› 2.1.3.1.4 Processing Requirements

A Reading System must process the `epub:type` attribute as follows:

- It may associate specialized behaviors with none, some or all of the terms defined in the default vocabulary.
- It may associate specialized behaviors with terms given in vocabularies other than the default.
- It must ignore terms that it does not recognize.

When Reading System behavior associated with a given `epub:type` value conflicts with behavior associated with the carrying element, the behavior associated with the element must be given precedence.

› 2.1.3.2 SSML Attributes

› 2.1.3.2.1 Overview

The W3C Speech Synthesis Markup Language [SSML] is a language used for assisting Text-to-Speech (TTS) engines in generating synthetic speech. Although SSML is designed as a standalone document type, it also defines semantics suitable for use within other host languages.

This specification recasts the SSML 1.1 `phoneme` element as two attributes — `ssml:ph` and `ssml:alphabet` — and makes them available within EPUB XHTML Content Documents.

Reading Systems with Text-to-Speech (TTS) capabilities should support the SSML Attributes as defined below.

NOTE

For more information on EPUB 3 features related to synthetic speech, refer to Text-to-speech [EPUB3Overview].

› 2.1.3.2.2 The `ssml:ph` attribute

The `ssml:ph` attribute specifies a phonemic/phonetic pronunciation of the text represented by the element to which the attribute is attached.

Attribute Name

`ph`

Namespace

`http://www.w3.org/2001/10/synthesis`

Usage

May be specified on all elements with which a phonetic equivalent can logically be associated (e.g., elements that contain textual information).

Must not be specified on a descendant of an element that already carries this attribute.

Value

A phonemic/phonetic expression, syntactically valid with respect to the phonemic/phonetic alphabet being used.

This attribute inherits all the semantics of the SSML 1.1 `phoneme` element `ph` attribute, with the following addition:

› When the `ssml:ph` attribute appears on an element that has text node descendants, the corresponding document text to which the pronunciation applies is the string that results from concatenating the descendant text nodes, in document order. The specified phonetic pronunciation must therefore logically match the element's textual data in its entirety (i.e., not just an isolated part of its content).

NOTE

Reading Systems that support the SSML Attributes and PLS Documents must honor the defined precedence rules for these two constructs.

› 2.1.3.2.3 The `ssml:alphabet` attribute

The ssml:alphabet attribute specifies which phonemic/phonetic pronunciation alphabet is used in the value of the ssml:ph attribute.

Attribute Name	alphabet
Namespace	http://www.w3.org/2001/10/synthesis
Usage	Global, may be specified on any element.
Value	The name of the pronunciation alphabet used in the value of ssml:ph (inherited).

This attribute inherits all the semantics of the SSML 1.1 phoneme element alphabet attribute, with the following addition:

> The value of the ssml:alphabet attribute is inherited in the document tree. The pronunciation alphabet used in a given ssml:ph attribute value is determined by locating the first occurrence of the ssml:alphabet attribute starting with the element on which the ssml:ph attribute appears, followed by the nearest ancestor element.

Reading Systems that support the SSML Attributes feature of this specification should support the ipa alphabet.

2.1.3.3 Content Switching

2.1.3.3.1 Introduction

This section is informative

The switch element provides a simple mechanism through which Authors can tailor the Publication content displayed to Users, one that isn't dependent on the scripting capabilities of the Reading System.

Reading System developers may choose to support XML vocabularies and new HTML elements that are not valid in XHTML Content Documents. The switch mechanism encourages this type of development and experimentation, but at the same time provides Authors who wish to take advantage of it the security of knowing that their content will still display on any compliant Reading System (i.e., it maintains the baseline requirement that all XHTML Content Documents be valid if none of the specialized markup is supported).

Content switching is not just about encouraging future development, however; it can also be used to create Publications that maintain a level of compatibility with older Reading Systems unable to handle the new features of EPUB 3. For example, instances of MathML, now a native type, could be added using switch elements so that EPUB 2 Reading Systems could instead provide fallback images or text.

2.1.3.3.2 Definition

2.1.3.3.2.1 The epub:switch Element

The switch element allows an XML fragment to be conditionally inserted into the content model of an XHTML Content Document.

Element name

switch

Namespace

http://www.idpf.org/2007/ops

Usage

In Flow and Inline content. Repeatable.

Attributes

id [optional]

The ID [XML] of this element, which must be unique within the document scope.

Content Model

In this order: case [1 or more], default [exactly 1].

A Reading System must individually process each switch element in a document to determine whether it can render any of the child case elements (as determined by the value of their required-namespace attributes).

For each switch encountered, the Reading System should render the content of the first case it supports, but is free to select from any of the available options. If the Reading System does not support the markup contained in any of the child case elements, it must render the contents of the default element.

The [HTML5] object element should be used to embed custom (non-core) content types in XHTML Content Documents. Custom markup should be wrapped in a switch element only when the content it represents is an integral part of the document and depends on the context of the document to be properly processed.

Examples

An example of ChemML markup inserted using the switch *element.*

```
<epub:switch id="cmlSwitch">

   <epub:case required-namespace="http://www.xml-cml.org/schema">
      <cml xmlns="http://www.xml-cml.org/schema">
         <molecule id="sulfuric-acid">
            <formula id="f1" concise="H 2 S 1 O 4"/>
         </molecule>
      </cml>
   </epub:case>

   <epub:default>
      <p>H<sub>2</sub>SO<sub>4</sub></p>
   </epub:default>

</epub:switch>
```

An example of adding MathML markup for compliance with EPUB 2 Reading Systems.

```
<epub:switch id="mathmlSwitch">

   <epub:case required-namespace="http://www.w3.org/1998/Math/MathML">
      <math xmlns="http://www.w3.org/1998/Math/MathML">
         <mrow>
            <mn>2</mn>
            <mo> &#x2061;<!--INVISIBLE TIMES--></mo>
            <mi>x</mi>
         </mrow>
         <mrow>
            <mo>+</mo>
            <mi>y</mi>
            <mo>-</mo>
            <mi>z</mi>
         </mrow>
      </math>
   </epub:case>

   <epub:default>
      <p>2x + y - z</p>
   </epub:default>

</epub:switch>
```

› 2.1.3.3.2.2 The `epub:case` Element

The `case` element contains an instance of markup from an XML vocabulary. The included markup may be natively supported in XHTML Content Documents (in the case of MathML and SVG), but such support is not a requirement.

Element name

`case`

Namespace

`http://www.idpf.org/2007/ops`

Usage

Required first child of `switch`. Repeatable.

Attributes

`id` [optional]

The ID [XML] of this element, which must be unique within the document scope.

`required-namespace` [required]

An extension identifier in URI form [RFC2046] that identifies the XML vocabulary or extension that the Reading System must support in order to process the content of the `case` element.

Content Model

An XML fragment conforming to the markup vocabulary identified in the `required-namespace` attribute.

Each case element must contain an alternate representation of the same content. To ensure the best rendering of their content, Authors should order case elements by to their optimal rendering format.

If the case element contains markup that is valid in an XHTML Content Document (e.g., MathML), that content must be valid at the point where the switch element has been inserted (i.e., its addition must not result in an invalid document).

Foreign markup in a case element must be well formed, but does not have to be valid at its point of insertion. Authors should ensure that any foreign markup matches the context in which it is used (e.g., a block element should not be included in a switch element inserted in an inline context).

NOTE

The IDPF maintains an informative registry of common extension identifiers for use in the required-namespace attribute at http://www.idpf.org/epub/switch/.

› 2.1.3.3.2.3 The epub:default Element

The default element provides markup that is valid in any XHTML Content Document for when a Reading System cannot render any of the case elements.

Element name

default

Namespace

http://www.idpf.org/2007/ops

Usage

Required last child of epub:switch.

Attributes

id [optional]

The ID [XML] of this element, which must be unique within the document scope.

Content Model

An [HTML5]-compliant markup fragment.

The default element acts as a fallback for the switch and must include a representation of the content that is valid in XHTML Content Documents.

The default element must not include content that would invalidate the document at the point where the switch has been inserted: XHTML Content Documents must be valid if all the switch elements are replaced by their child default elements.

› 2.1.3.3.3 Processing

EPUB Reading Systems must support the switch element.

This specification does not require a specific rendering approach for switch elements. A Reading Systems may choose to apply CSS styling to render each switch, for example, but may use any other

approach as appropriate. All Reading Systems must present the content of only one `case` element or the `default` element per `switch` for rendering, however.

The `switch` element must be processed as though all of its children but one have the HTML5 hidden attribute set (i.e., all the same processing rules and requirements outlined for that attribute should be applied to the content not to be rendered).

> NOTE
>
> As the content that may be rendered depends on the capabilities of the User's Reading System, linking can be guaranteed only to the `switch` element. Deep referencing into the `switch` element is not recommended.

> NOTE
>
> The occurrence of `switch` elements in XHTML Content Document is indicated in the Package Document manifest through the `switch [Publications30]` property.

› 2.1.3.4 The `epub:trigger` Element

The `trigger` element enables the creation of markup-defined user interfaces for controlling multimedia objects, such as audio and video playback, in both scripted and non-scripted contexts.

Element name

`trigger`

Namespace

`http://www.idpf.org/2007/ops`

Usage

As a child of `head` and in Flow content. Repeatable.

Attributes

`id` [optional]

The ID [XML] of this element, which must be unique within the document scope.

`action` [required]

The action to perform for this event.

Allowed values: `show` | `hide` | `play` | `pause` | `resume` | `mute` | `unmute`

`ref` [required]

An IDREF [XML] that identifies the element that is the object of the action.

`ev:event` [required]

The applicable event for this trigger, as defined in [XML Events].

`ev:observer` [required]

The source object for this trigger, as defined in [XML Events].

Content Model

Empty.

The `trigger` element associates an `event` from a specified source object (`observer`) with a desired action to be performed with a specified target object (`ref`).

The semantics of the defined `action` values are:

- `show` – set the element's DOM visibility [CSS2.1] property to visible.
- `hide` – set the element's DOM visibility [CSS2.1] property to hidden.
- `play` – play the associated resource from the beginning (only applicable to video or audio elements).
- `pause` – pause playing (only applicable to video or audio elements).
- `resume` – resume playing (only applicable to video or audio elements).
- `mute` – mute sound (only applicable to video or audio elements).
- `unmute` – unmute sound (only applicable to video or audio elements).

Reading Systems that support video or audio playback must support the `epub:trigger` element.

Sample markup of a video player that uses `trigger` *elements to control playback and muting.*

```
<html xmlns="http://www.w3.org/1999/xhtml"
  xmlns:epub="http://www.idpf.org/2007/ops"
  xmlns:ev="http://www.w3.org/2001/xml-events">
  <head>
    <epub:trigger ev:observer="pause" ev:event="click" action="pause"
ref="test"/>
    <epub:trigger ev:observer="resume" ev:event="click" action="resume"
ref="test"/>
    <epub:trigger ev:observer="mute" ev:event="click" action="mute"
ref="test"/>
    <epub:trigger ev:observer="mute" ev:event="click" action="show"
ref="muted"/>
    <epub:trigger ev:observer="unmute" ev:event="click" action="unmute"
ref="test"/>
    <epub:trigger ev:observer="unmute" ev:event="click" action="hide"
ref="muted"/>
  </head>
  <body>
    <video id="test" src="birds.mp4" width="320" height="240"/>
    <p>
      <span class="button" id="resume">Play/Resume</span>
      <span class="button" id="pause">Pause</span>
      <span class="button" id="mute">Mute</span>
      <span class="button" id="unmute">Unmute</span>
      <span id="muted">MUTED</span>
    </p>
  </body>
</html>
```


› 2.1.3.5 Alternate Style Tags

In accordance with [AltStyleTags] , the `link` element `class` attribute may include any of the following values: `horizontal`, `vertical`, `day` and `night`. These values inherit the semantics defined by that

specification for their use.

Reading Systems should select and utilize such tagged style sets as appropriate, and as described in that specification.

› 2.1.4 HTML5 Deviations and Constraints

This section defines deviations and/or constraints in EPUB 3 XHTML Content Documents to the underlying [HTML5] document model.

› 2.1.4.1 Embedded MathML

› 2.1.4.1.1 Introduction

This section is informative

XHTML Content Documents support embedded [MATHML] but limit its usage to a restricted subset of the full MathML markup language.

This subset is designed to ease the implementation burden on Reading Systems and to promote accessibility, while retaining compatibility with [HTML5] User Agents.

NOTE

The `mathml` [Publications30] property of the manifest `item` element indicates that an XHTML Content Document contains embedded MathML.

› 2.1.4.1.2 Content Conformance

Any occurrence of MathML markup in XHTML Content Documents must conform to the constraints expressed in the MathML specification [MATHML], with the following additional restrictions:

Presentation MathML

› The `m:math` element must contain only Presentation MathML, with the exception of the `m:annotation-xml` element as defined below.

Content MathML

› Content MathML may be included within MathML markup in XHTML Content Documents, and, when present, must occur within an `m:annotation-xml` child element of an `m:semantics` element.

› When Content MathML is included as per the previous condition, the given `m:annotation-xml` element's `encoding` attribute must be set to either of the functionally-equivalent values `MathML-Content` or `application/mathml-content+xml`, and its `name` attribute must be set to `contentequiv`.

Deprecated MathML

› Elements and attributes marked as deprecated in [MATHML] must not be included within MathML markup in XHTML Content Documents.

XHTML Content Document fragments

› XHTML Content Document fragments may be included within MathML markup in XHTML Content Documents, and, when present, must occur within an `m:annotation-xml` child element of an

m:semantics element.

› When an XHTML Content Document fragment is included as per the above paragraph, the given m:annotation-xml element's encoding attribute must be set to application/xhtml+xml and its name attribute must be set to alternate-representation.

› Any included XHTML Content Document fragments must not themselves contain MathML markup.

› Any included XHTML Content Document fragments must conform to the content model in which the ancestor m:math element occurs, such that if the m:math element is replaced by the given XHTML Content Document fragment the document remains valid.

Alternative Content

› Alternative content should be included, and, when present, must be represented as defined in Alternative Content.

› 2.1.4.1.3 Reading System Conformance

A conformant EPUB Reading System must meet all of the following criteria for processing MathML embedded in XHTML Content Documents:

› It must support processing of Presentation MathML, and may support processing of Content MathML, using semantics defined by the MathML 3.0 specification [MATHML].

› If it has a Viewport, it must support visual rendering of Presentation MathML.

› When producing alternative textual content for MathML markup, it should be able to dynamically generate such content from the given Presentation MathML, and if not, must give preference to XHTML Content Document fragments followed by the alttext attribute on the m:math element.

› It must regard the mathml [Publications30] property of the Package Document manifest item element as the authoritative definition of whether an XHTML Content Document includes embedded MathML.

› 2.1.4.1.4 Alternative Content

Reading Systems should be able to generate any necessary alternative textual renditions dynamically using the given Presentation MathML markup (e.g., as output to Text-to-Speech (TTS) engines). To support Reading Systems that are not so capable, alternative textual content should be included with each occurrence of the m:math element in XHTML Content Documents.

The alttext attribute on the m:math element should be used for this purpose primarily when shorter alternative text runs are sufficient. When more extensive alternative text is required, XHTML Content Document fragments should be used. (Note that Reading Systems query these two alternative text locations using a defined preference order.)

For Reading System forward compatibility purposes, fallback images may be provided using the altimg attribute on the m:math element. It is recommended that the dimension and alignment attributes (altimg-width, altimg-height and altimg-valign) be used in conjunction with the altimg attribute.

NOTE

All referenced Publication Resources must conform to the constraints for Publication Resources defined in EPUB Publication — Content Conformance [Publications30].

› 2.1.4.2 Embedded SVG

XHTML Content Documents support the embedding of SVG 1.1 document fragments *by reference* (embedding via reference, for example, from an `img` or `object` element) and *by inclusion* (embedding via direct inclusion of the `svg:svg` element in the XHTML Content Document) [SVG].

The content conformance constraints for SVG embedded in XHTML Content Documents are the same as defined for SVG Content Documents in Restrictions on SVG 1.1.

Reading Systems must process SVG embedded in XHTML Content Documents as defined in SVG Content Documents — Reading System Conformance.

> NOTE
>
> The `svg` `[Publications30]` property of the manifest `item` element indicates that an XHTML Content Document contains embedded SVG.

› 2.1.4.2.1 Embedded SVG and CSS

For the purposes of styling SVG embedded in XHTML Content Documents *by reference*, Reading Systems must not apply CSS style rules of the containing document to the referenced SVG document.

For the purposes of styling SVG embedded in XHTML Content Documents *by inclusion*, Reading Systems must apply applicable CSS rules of the containing document to the included SVG elements.

> NOTE
>
> SVG included *by reference* is processed as a separate document, and may include its own CSS style rules just like an SVG Content Document would. Note that this is consistent with situations where an [HTML5] `object` element references an external [HTML5] element.

› 2.1.4.3 Unicode Restrictions

This section lists restrictions on the Unicode character repertoire.

Private Use Characters and Embedded Fonts

Any included characters that map to a code point within one of the Private Use Area (PUA) ranges as defined in [Unicode] must occur within a string that is styled or attributed in a manner that includes a reference to an embedded font that contains an appropriate glyph for that code point.

› 2.1.4.4 Discouraged Constructs

The `rp` Element

› The [HTML5] `rp` element is intended to provide a fallback — an optional parenthesis display around `ruby` markup — for older version Reading Systems that do not recognize ruby markup. As EPUB 3 Reading Systems are ruby-aware, and can provide fallbacks, the use of `rp` elements in Content Documents is discouraged.

The `embed` Element

› Since the [HTML5] `embed` element does not provide intrinsic facilities to provide fallbacks for Reading Systems that do not support scripting, its use is discouraged when the resource referenced has scripting components. Authors should use the `object` element instead.

› 2.2 EPUB Navigation Documents

› 2.2.1 Introduction

This section is informative

The EPUB Navigation Document is a required component [Publications30] of EPUB Publications. It provides the Author with a mechanism to include a human- and machine-readable global navigation layer in the Publication, thereby ensuring increased usability and accessibility for the User.

The EPUB Navigation Document is an adaptation of XHTML Content Document and is, by definition, a valid XHTML Content Document instance. All Content and Reading System conformance requirements that apply to XHTML Content Documents also apply to the EPUB Navigation Document.

The navigation features of this adaptation are expressed through specializations of the [HTML5] `nav` element. Each `nav` element in an EPUB Navigation Document represents a data island — an embedded source of specialized information within the general markup — from which Reading Systems can retrieve navigational information. Unlike typical XML data islands, however, the information within the `nav` element remains human readable as an [HTML5] document.

To facilitate machine readability, the content model of `nav` elements in EPUB Navigation Documents is restricted relative to what is allowed in general XHTML Content Documents.

> NOTE
>
> The EPUB Navigation Document is identified in the Package Document manifest through the `nav` [Publications30] property.

> NOTE
>
> The EPUB Navigation Document supersedes the NCX document type as defined in [OPF2].
>
> Information on how EPUB 3 Publications may include an NCX document for EPUB 2 Reading System forwards compatibility purposes is available in NCX Superseded [Publications30].

› 2.2.2 Content Conformance

A conformant EPUB Navigation Document must meet all of the following criteria:

Document Properties

› It must conform to all content conformance constraints for XHTML Content Documents as defined in XHTML Content Documents — Content Conformance.

› It must be valid to the EPUB Navigation Document schema as defined in EPUB Navigation Document Schema and conform to all content conformance constraints specific for EPUB Navigation Documents expressed in EPUB Navigation Document Definition.

› As a conforming XHTML Content Document, it may be included in the Publication spine, but may also be provided independently of it.

› 2.2.3 Reading System Conformance

A conformant EPUB Reading System must meet all of the following criteria for processing EPUB Navigation Document:

› When requested by a User, Reading Systems must provide access to the links and link labels in the `nav` elements of the EPUB Navigation Document in a fashion that allows the User to activate the links provided. When a link is activated, the Reading System must relocate the application's current reading position to the destination identified by that link.

› Reading Systems must honor the above requirement irrespective of whether the EPUB Navigation Document provided in a Publication is part of the spine.

› 2.2.4 EPUB Navigation Document Definition

› 2.2.4.1 The `nav` Element: Restrictions

This specification restricts the content model of the `nav` element and its descendants in EPUB Navigation Documents as follows:

› Each `nav` element may contain an optional heading indicating the title of the navigation list. The heading must be one of the [HTML5] `h1` through `h6` elements or an `hgroup`.

› The optional heading must be followed by a single `ol` ordered list; no other elements are permitted as direct children of the `nav` element. This ordered list represents the primary level of content navigation.

› Each list item (`li`) of the ordered list represents a primary heading, structure or other point of interest within the Publication and must contain either a child `a` element or a child `span` element. The `a` element describes the target within the Content Document that the link points to. The `span` element serves as a heading for breaking down lists into distinct groups (for example, a large list of illustrations can be segmented into several lists, one for each chapter).

› Each child `a` or `span` element of a list item may contain any valid HTML5 phrasing content, but must not result in a zero-length text string after concatenation of all child content and application of whitespace normalization rules.

› If the `a` element contains instances of HTML5 embedded content that do not provide intrinsic text alternatives, it must also include a `title` attribute with an alternate text rendition of the link label.

› The relative IRI reference provided in the `href` attribute of the `a` element must resolve to an EPUB Content Document or fragment therein.

› The `a` element may optionally be followed by an `ol` ordered list representing a subsidiary content level below that heading (e.g., all the subsection headings of a section). The `span` element must be followed by an `ol` ordered list: it cannot be used in "leaf" `li` elements. Regardless of whether an `a` or `span` element precedes it, this sublist must adhere to all the content requirements defined in this section for constructing the primary navigation list, and recursively (for each additional level of the Publication's hierarchy represented in this manner).

› The `ol` element represents an ordered list. In the context of this specification, the default display style of list items must be equivalent to CSS `list-style: none` (Reading Systems with no CSS support must not show list item numbering). Authors may specify alternative list item styles using CSS, but these would obviously be ignored by Reading Systems that do not support Cascading Style Sheets.

The following example shows a partial `lot` *("list of tables")* `nav` *element, with* `span` *elements used as link-less headings for grouping the sublists.*

```
<nav epub:type="lot">
    <h2>List of tables, broken down into individual groups, one per major
section of the publication content</h2>
```

```
    <ol>
        <li><span>Tables in Chapter 1</span>
            <ol>
                <li><a href="chap1.xhtml#table-1.1">Table 1.1</a>
                </li>
                <li><a href="chap1.xhtml#table-1.2">Table 1.2</a></li>
            </ol>
        </li>

        <li><span>Tables in Chapter 2</span>
            <ol>
                <li><a href="chap2.xhtml#table-2.1">Table 2.1</a>
                </li>
                <li><a href="chap2.xhtml#table-2.2">Table 2.2</a></li>
                <li><a href="chap2.xhtml#table-2.3">Table 2.3</a></li>
            </ol>
        </li>
        ...
        <li><span>Tables in Appendix</span>
            <ol>
                <li><a href="appendix.xhtml#table-a.1">Table A.1</a>
                </li>
                <li><a href="appendix.xhtml#table-a.2">Table B.2</a></li>
            </ol>
        </li>
    </ol>
</nav>
```

› 2.2.4.2 The nav Element: Types

The nav elements defined in an EPUB Navigation Document are distinguished semantically by the value of their epub:type attribute. By default, values of epub:type are drawn from the EPUB 3 Structural Semantics Vocabulary [StructureVocab], but values drawn from other vocabularies are also allowed. Refer to The epub:type Attribute for more information.

› 2.2.4.2.1 The toc nav Element

The toc nav element defines the primary navigational hierarchy of the EPUB Publication. It conceptually corresponds to a table of contents in a printed work (i.e., it provides navigation to the structural sections of the Publication).

For usability and accessibility reasons, Authors should provide a comprehensive table of contents: the toc nav should not exclude references based solely on their nesting depth within the document hierarchy, as is often the case in print works (particularly in reduced tables of contents).

In the case of Publications that exclusively reference XHTML Content Documents from their spines, the toc nav will typically correspond to the aggregation of HTML5 outlines of those documents (excluding any subtrees that do not contribute to the primary Publication outline).

The order of li elements contained within the toc nav element must match the order of the targeted elements within each targeted EPUB Content Document, and must also follow the order of Content Documents in the Publication spine.

The toc nav element must occur exactly once in EPUB Navigation Documents.

NOTE

The `toc nav` element corresponds to the `navMap` element in the superseded NCX [OPF2].

2.2.4.2.2 The `page-list nav` Element

The `page-list nav` element is a container for pagination information. It provides navigation to positions in the Publication content that correspond to the locations of page boundaries present in a print source being represented by this EPUB Publication.

The `page-list nav` element is optional in EPUB Navigation Documents and must not occur more than once.

The order of `li` elements contained within a `page-list nav` structure must match the order of the actual pages inside each targeted EPUB Content Document and must also follow the order of Content Documents in the Publication spine.

The `page-list nav` element should contain only a single `ol` descendant (i.e., it should be a flat list, not a nested structure of navigation items).

NOTE

The `page-list nav` element corresponds to the `pageList` element in the superseded NCX [OPF2]

NOTE

The dc:source [Publications30] element provides a means of identifying the source publication to which the given pagination information applies.

2.2.4.2.3 The `landmarks nav` Element

The `landmarks nav` element identifies fundamental structural components of the publication in order to enable Reading Systems to provide the User efficient access to them.

The structural semantics of each link target within the `landmarks nav` element is determined by the value of the epub:type attribute on the `a` element descendants. The `epub:type` attribute is required on `a` element descendants of the `landmarks nav` element.

The `landmarks nav` element extends the suggested HTML context of terms from the EPUB Structural Semantics Vocabulary to include the `a` element.

The following example shows a `landmarks nav` element with structural semantics drawn from the EPUB Structural Semantics Vocabulary.

```
<nav epub:type="landmarks">
    <h2>Guide</h2>
    <ol>
        <li><a epub:type="toc" href="#toc">Table of Contents</a></li>
        <li><a epub:type="loi" href="content.html#loi">List of
Illustrations</a></li>
        <li><a epub:type="bodymatter"
href="content.html#bodymatter">Start of Content</a></li>
    </ol>
</nav>
```

The `landmarks nav` element is optional in EPUB Navigation Documents and must not occur more than once.

> NOTE
>
> The `landmarks nav` element corresponds to the deprecated OPF `guide` element. Refer to guide [Publications30] for more information.

› 2.2.4.2.4 Other `nav` Elements

EPUB Navigation Documents optionally may include one or more `nav` elements in addition to the `toc`, `page-list` and `landmarks nav` elements defined above. Such additional `nav` elements should have an `epub:type` attribute to provide a machine-readable semantic, and must have a human-readable heading as their first child.

This specification imposes no restrictions on the semantics of such additional `nav` elements: they may be used to represent navigational semantics for any information domain, and they may contain link targets with homogeneous or heterogeneous semantics.

› 2.2.4.3 The `hidden` attribute

In some cases, Authors may wish to hide parts of the navigation data within the content flow (i.e., the Reading System's principal rendering of the spine contents). A typical example is the list of page breaks, which usually isn't rendered as part of the content flow but instead exposed to the User separately in a dedicated navigation user interface.

While the CSS `display` property can be used to control the visual rendering of EPUB Navigation Documents in Reading Systems with CSS Viewports, not all Reading Systems provide such an interface. To control rendering across all Reading Systems, authors must use the [HTML5] `hidden` attribute to indicate which (if any) portions of the navigation data are excluded from rendering in the content flow. The `hidden` attribute has no effect on how navigation data is rendered outside of the content flow (such as in dedicated navigation user interfaces provided by Reading Systems).

The following example shows a partial `page-list nav` *element. The presence of the* `hidden` *attribute on the root indicates that the entire list is excluded from rendering in the content flow.*

```
<nav epub:type="page-list" hidden="">
    <h2>Pagebreaks of the print version, third edition</h2>
    <ol>
        <li><a href="frontmatter.xhtml#pi">I</a></li>
        <li><a href="frontmatter.xhtml#pii">II</a></li> … <li><a
href="chap1.xhtml#p1">1</a></li>
        <li><a href="chap1.xhtml#p2">2</a></li> … </ol>
</nav>
```

The following example shows a partial `toc nav` *element where the* `hidden` *attribute is used to limit content flow rendering to the two topmost hierarchical levels.*

```
<nav epub:type="toc" id="toc">
  <h1>Table of contents</h1>
  <ol>
    <li>
      <a href="chap1.xhtml">Chapter 1</a>
      <ol>
        <li>
          <a href="chap1.xhtml#sec-1.1">Chapter 1.1</a>
```

```
          <ol hidden="">
            <li>
              <a href="chap1.xhtml#sec-1.1.1">Section 1.1.1</a>
            </li>
            <li>
              <a href="chap1.xhtml#sec-1.1.2">Section 1.1.2</a>
            </li>
          </ol>
        </li>
        <li>
          <a href="chap1.xhtml#sec-1.2">Chapter 1.2</a>
        </li>
      </ol>
    </li>
    <li>
      <a href="chap2.xhtml">Chapter 2</a>
    </li>
  </ol>
</nav>
```

2.3 SVG Content Documents

2.3.1 Introduction

This section is informative

The Scalable Vector Graphics (SVG) 1.1 (Second Edition) specification [SVG] defines a format for representing final-form vector graphics and text.

Although an EPUB Publication typically uses XHTML Content Documents as the top-level document type, the use of SVG Content Documents is also permitted. SVGs are typically only used in certain special circumstances, such as when final-form page images are the only suitable representation of the content (as may be the case, for example, in the context of manga or comic books).

This section defines a profile for [SVG] documents. An instance of an XML document that conforms to this profile is a Core Media Type and is referred to in this specification and its sibling specifications as an SVG Content Document.

NOTE

This section defines conformance requirements for SVG Content Documents. Refer to Embedded SVG for conformance requirements for SVG embedded in XHTML Content Documents.

2.3.2 Content Conformance

An SVG Content Document must meet all of the following criteria:

Document Properties

- It must meet the conformance constraints for XML documents defined in XML Conformance [Publications30].
- It must be an SVG 1.1 document fragment valid to the SVG Content Document schema as

defined in SVG Content Document Schema and conform to all content conformance constraints expressed in Restrictions on SVG 1.1.

› It should adhere to the accessibility guidelines given in [SVG Access].

File Properties

› The SVG Content Document filename should use the file extension `.svg`.

NOTE

All Publication Resources referenced from an SVG Content Document must conform to the constraints for Publication Resources defined in EPUB Publication — Content Conformance [Publications30]

› 2.3.3 Restrictions on SVG 1.1

This specification restricts the content model of SVG Content Documents and SVG embedded in XHTML Content Documents as follows:

› The [SVG] Animation Elements and Animation event attributes must not occur.

› The [SVG] `svg:foreignObject` element must contain only valid XHTML Content Document Flow content, and its `requiredExtensions` attribute, if given, must be set to `http://www.idpf.org/2007/ops`.

› The [SVG] `svg:title` element must contain only valid XHTML Content Document Phrasing content.

› 2.3.4 Reading System Conformance

A conformant EPUB Reading System must meet all of the following criteria for processing SVG Content Documents and SVG embedded in XHTML Content Documents:

› It must support the language features of SVG that correspond to the feature string `http://www.w3.org/TR/SVG11/feature#SVG-dynamic` minus the `http://www.w3.org/TR/SVG11/feature#Animation` and `http://www.w3.org/TR/SVG11/feature#AnimationEventsAttribute` features (see Feature strings) [SVG].

› It must meet the Reading System conformance criteria defined in Scripted Content Documents — Reading System Conformance.

› If it has an SVG Viewport, it must support the visual rendering of SVG using CSS as defined in Section 6 of [SVG], and it should support all properties defined in Appendix N of that specification. In the case of embedded SVG, it must also conform to the constraints defined in Embedded SVG and CSS.

› It should support User selection and searching of text within SVG elements.

› It must recognize the value `http://www.idpf.org/2007/ops` of the `requiredExtensions` attribute when appearing on the `svg:switch` and `svg:foreignObject` elements as representing the occurrence of XHTML Content Document fragments.

› It must regard the svg [Publications30] property of the Package Document manifest `item` element as the authoritative definition of whether an EPUB XHTML Content Document includes embedded SVG.

› 2.4 Scripted Content Documents

EPUB Content Documents may contain scripting using the facilities defined for this in the respective underlying specifications ([HTML5] and [SVG]). When an EPUB Content Document contains scripting, it is referred to in this specification and its sibling specifications as a Scripted Content Document. This label also applies to XHTML Content Documents when they contain instances of HTML5 forms.

› 2.4.1 Scripting Contexts

This specification defines two contexts in which scripts may appear:

spine-level

> An instance of the [HTML5] `script` element included in a Top-level Content Document.

container-constrained

> An instance of the [HTML5] `script` element included in an EPUB Content Document that is embedded in a parent Content Document using one of the [HTML5] `object`, `iframe` or `embed` elements.

In both of the above-defined contexts, whether the JavaScript code is embedded directly in the `script` element or referenced via its `src` attribute makes no difference to the executing context.

Which context a script is used in determines the rights and restrictions that a Reading System may place on it. Refer to Content Conformance and Reading System Conformance for some specific requirements that must be adhered to (not all Reading Systems may provide the same scripting functionality).

Example

Consider the following example Package Document:

```
<package …>
    …
    <manifest>
        …
        <item id="chap01"
            href="scripted01.xhtml"
            media-type="application/xhtml+xml"
            properties="scripted"/>
        <item id="inset01"
            href="scripted02.xhtml"
            media-type="application/xhtml+xml"
            properties="scripted"/>
        <item id="slideshowjs"
            href="slideshow.js"
            media-type="text/javascript"/>
    </manifest>

    <spine …>
        <itemref idref="chap01"/>
        …
    </spine>
    …
</package>
```

and the following file `scripted01.xhtml`:

```
<html …>
    <head>
        …
        <script type="text/javascript">
            alert("Reading System name: " +
navigator.epubReadingSystem.name);
        </script>
    </head>
    <body>
        …
        <iframe src="scripted02.xhtml" … />
        …
    </body>
</html>
```

and the following file `scripted02.xhtml`:

```
<html …>
    <head>
        …
        <script type="text/javascript" href="slideshow.js"></script>
    </head>
    <body>
        …
    </body>
</html>
```

From these examples, it is true that:

- the code in the `script` element in the `head` in `scripted01.xhtml` is a spine-level script because the document is referenced from the spine;
- the code in the `script` element in `scripted02.xhtml` is a container-constrained script because the HTML file it occurs in is included in `scripted01.xhtml` via the `iframe` element.

› 2.4.2 Content Conformance

Container-constrained scripts

› A container-constrained script must not contain instructions for modifying the DOM of the parent Content Document or other contents in the Publication, and must not contain instructions for manipulating the size of its containing rectangle.

Spine-level scripts

› EPUB Content Documents that include spine-level scripting must utilize the *progressive enhancement technique*, which for the purposes of this specification has the following definition: when the document is rendered by a Reading System without scripting support or with scripting support disabled, the top-level document content must retain its integrity, remaining consumable by the User without any information loss or other significant deterioration.

Accessibility

EPUB Content Documents that include scripting — using any inclusion model — should employ › relevant accessibility techniques to ensure that the content remains consumable by all Users. [WAI-ARIA] [WCAG20]

Fallbacks

› EPUB Content Documents that include scripting — using any inclusion model — may provide fallbacks for such content, either by using intrinsic fallback mechanisms (such as those available for the [HTML5] `object` and `canvas` elements) or, when an intrinsic fallback is not applicable, by using a manifest-level [Publications30] fallback.

NOTE

The `scripted` [Publications30] property of the manifest `item` element indicates that an EPUB Content Document is a Scripted Content Document.

› 2.4.3 Reading System Conformance

EPUB Reading System support for scripting is optional. A Reading System that supports scripting must meet the following criteria:

› It must support container-constrained scripting and may support spine-level scripting.

› It may render Scripted Content Documents as an interactive, scripted User Agent according to [HTML5].

› It must not allow a container-constrained script to modify the DOM of the parent Content Document or other contents in the Publication, and must not allow it to manipulate the size of its containing rectangle. (Note: Even if a script is not container-constrained, the Reading System may impose restrictions on modifications (see also the dom-manipulation feature).)

› It may place additional limitations on the capabilities provided to scripts during execution (e.g., limiting networking).

› It must implement the JavaScript `navigator` extension object `epubReadingSystem` defined in Appendix B, *JavaScript epubReadingSystem Object*. It also must support the `dom-manipulation` and `layout-change` features defined in Features in container-constrained scripting contexts.

› It must regard the `scripted` [Publications30] property of the Package Document manifest `item` element as the authoritative definition of whether an EPUB Content Document includes scripting.

A Reading System that does not support scripting must meet the following criteria:

› It must process fallbacks for scripted content as defined in Fallbacks for Scripted Content Documents.

NOTE

Reading Systems may render Scripted Content Documents in a manner that disables other EPUB capabilities and/or provides a different rendering and User experience (e.g., by disabling pagination).

Authors choosing to restrict the usage of scripting to the container-constrained model will ensure a more consistent User experience between scripted and non-scripted content (e.g., consistent pagination behavior).

Authors should use declarative techniques whenever practical to increase the interoperability, longevity and accessibility of their Publications, and avoid the inclusion of scripting whenever practical.

› 2.4.4 Security Considerations

This section is informative

All EPUB Authors and Reading System developers need to be aware of the security issues that arise when scripted content is executed by a Reading System. As the underlying scripting model employed by Reading Systems and browsers is the same, the same kinds of issues encountered in Web contexts must be taken into consideration.

Each Reading System should establish if the scripts in a particular document are to be trusted or not. It is recommended that all scripts be treated as untrusted (and potentially malicious), and that all vectors of attack be examined and protected against. In particular, the following should be considered:

- an attack against the runtime environment (e.g., stealing files from a User's hard drive);
- an attack against the Reading System itself (e.g., stealing a list of a User's books or causing unexpected behavior);
- an attack of one Content Document against another (e.g., stealing data that originated in a different document);
- an attack of an unencrypted script against an encrypted portion of a document (e.g., an injected malicious script extracting protected content);
- an attack against the local network (e.g., stealing data from a server behind a firewall).

The following recommendations are provided as a guide to handling untrusted scripts:

- Reading Systems should behave as if a unique domain were allocated to each Content Document, as browser-based security relies heavily on document URLs and domains. Adopting this approach will isolate documents from each other and from other Internet domains, thereby limiting access to external URLs, cookies, DOM storage, etc.

  Reading Systems that enable scripting and network access should also consider including methods to notify the user that network activity is occurring and/or that allow them to disable it.

  NOTE

  In practice, Reading Systems may share domains across documents, but they still should maintain isolation between documents.

  If parts of a document are encrypted and parts are not, or if different encryption keys are used for different parts of the document, a unique per-document domain might not provide sufficient protection.

- If a Reading System allows persistent data to be stored, that data should be treated as sensitive. Scripts may save persistent data through cookies and DOM storage, but Reading Systems may block such attempts. Reading Systems that do allow data to be stored must ensure that it is not made available to other unrelated documents (e.g., ones that could have been spoofed). In particular, checking for a matching document identifier (or similar metadata) is not a valid method to control access to persistent data.

  Reading Systems that allow local storage should also provide methods for Users to inspect, disable, or delete that data. The data should be destroyed if the corresponding EPUB Publication is deleted.

Note that compliance with these recommendations does not guarantee protection from the possible attacks listed above; developers must examine each potential vulnerability within the context of their Reading System.

› 2.4.5 Event Model Considerations

This section is informative

Reading Systems should follow the DOM Event model as per [HTML5] and pass UI events to the scripting environment before performing any default action associated with these events. Reading System implementers should ensure that scripts cannot disable critical functionality (such as navigation) to constrain the extent to which a potentially malicious script could impact their Reading Systems. As a result, although the scripting environment should be able to cancel the default action of any event, some events either might not be passed through or might not be cancelable.

Authors should take into account the wide variety of possible Reading System implementations when adding scripting functionality to their Publications (e.g., not all devices have physical keyboard, and in many cases a soft keyboard is only activated only for text input elements). Consequently, relying on keyboard events alone is not recommended; alternative ways to trigger the desired action should always be provided.

› 3 EPUB Style Sheets

This section defines a profile for Cascading Style Sheets (CSS) intended to be used for styling of XHTML Content Documents. An instance of a CSS Style Sheet that conforms to this profile is a Core Media Type and is referred to in this specification and its sibling specifications as an EPUB Style Sheet.

CAUTION

The EPUB 3 CSS Profile references CSS specifications that are still works in progress and may change in incompatible ways. When utilizing features from such specifications, authors should consider the inherent risks in terms of the potential impact on interoperability and document longevity.

NOTE

The EPUB 3 CSS Profile employs the usage of the `-epub-` prefix for a number of CSS3 property names, as detailed below. As the CSS3 modules that define these properties mature and stabilize, EPUB authoring guidelines may encourage authors to also include unprefixed equivalents of these properties in EPUB 3 Style Sheets.

› 3.1 Content Conformance

A conformant EPUB Style Sheet must meet all of the following criteria:

› It must adhere to all content restrictions given in EPUB 3 CSS Profile.

› It may include constructs not explicitly identified in the EPUB 3 CSS Profile, but should be authored so that rendering fidelity does not depend on such additional constructs.

› It must be UTF-8 or UTF-16 encoded.

NOTE

All Publication Resources referenced from a CSS Style Sheet must conform to the constraints for Publication Resources defined in EPUB Publication — Content Conformance [Publications30]

3.2 Reading System Conformance

› Reading Systems with a CSS Viewport should support — render as defined by the corresponding specification in the Viewport — all CSS constructs included in this profile unless detailed otherwise in EPUB 3 CSS Profile.

› Reading Systems may support additional CSS constructs not explicitly identified in the EPUB 3 CSS Profile, and must handle any unsupported constructs as defined in [CSS2.1].

NOTE

Reading Systems have varying capabilities with regards to CSS rendering support, so may ignore some or all style information of an EPUB Style Sheet.

In addition, even when a Reading System does have a CSS Viewport, it is likely to render content in a manner that differs from typical HTML5 User Agents (e.g., paginating content rather than providing a infinitely scrolling surface).

3.3 EPUB 3 CSS Profile

3.3.1 CSS 2.1

The style baseline of the EPUB 3 CSS Profile is Cascading Style Sheets Level 2 Revision 1 [CSS2.1]. The profile includes all style sheet constructs normatively defined in [CSS2.1], with the following exceptions:

- The `fixed` value of the `position` property is not part of the EPUB 3 CSS Profile. To avoid potential rendering and interoperability issues, it should not be included in an EPUB Style Sheet.
- The `direction` and `unicode-bidi` properties must not be included in an EPUB Style Sheet. Authors should use appropriate [HTML5] markup to express directionality information instead.

Reading Systems that have a CSS Viewport must support the `font-family` property.

NOTE

The ability of Reading Systems to paginate absolutely positioned layouts is not guaranteed, so reliance on absolute positioning is discouraged. Reading Systems might not support these property values.

3.3.2 CSS 2.0

The EPUB 3 CSS Profile includes the following values for the `list-style-type` property as defined in [CSS2.0]:

- `cjk-ideographic`
- `hebrew`
- `hiragana`
- `hiragana-iroha`
- `katakana`
- `katakana-iroha`

› 3.3.3 CSS 3.0 Speech

The EPUB 3 CSS Profile includes `-epub-` prefixed versions of the following properties from the CSS3 Speech Module [CSS3Speech] using syntax as defined in [CSS3Speech-20110818] and semantics as defined in [CSS3Speech]:

- `-epub-cue`
- `-epub-pause`
- `-epub-rest`
- `-epub-speak`
- `-epub-speak-as`
- `-epub-voice-family`

> NOTE
>
> For more information on EPUB 3 features related to synthetic speech, refer to Text-to-speech [EPUB3Overview].

› 3.3.4 CSS Fonts Level 3

The EPUB 3 CSS Profile includes `@font-face` rules and descriptors as defined in the CSS Fonts Module Level 3 [CSS3Fonts] specification, using syntax as defined in [CSS3Fonts-20110324] and semantics as defined in [CSS3Fonts].

Reading Systems with a CSS Viewport must support OpenType [OpenType] and WOFF [WOFF] fonts embedded using the `@font-face` rule.

> NOTE
>
> Refer to Embedded Font Intrinsic Fallback [Publications30] for font fallback processing requirements.

In addition, Reading Systems must support at least the following `@font-face` font descriptors.

- `font-family`
- `font-style`
- `font-weight`

- src
- unicode-range

For forwards compatibility with EPUB 2 Reading Systems that do not support @font-face rules, authors should reference a generic font using the font-family property.

NOTE

Refer to Font Obfuscation [OCF3] for Reading System font obfuscation requirements.

› 3.3.5 CSS Text Level 3

The EPUB 3 CSS Profile includes -epub- prefixed versions of the following properties from the CSS Text Level 3 [CSS3Text] specification using syntax as defined in [CSS3Text-20110412] and semantics as defined in [CSS3Text].

- -epub-hyphens*
- -epub-line-break
- -epub-text-align-last
- -epub-text-emphasis
- -epub-text-emphasis-color
- -epub-text-emphasis-style
- -epub-word-break

* The -epub-hyphens property does not include support for the value all.

In addition, the EPUB 3 CSS Profile includes the unprefixed text-transform property from CSS Text Level 3 using semantics as defined in [CSS3Text] and syntax as defined in [CSS3Text-20110412], with the exception that the fullwidth and fullsize-kana values are prefixed in the EPUB 3 CSS Profile (-epub-fullwidth and -epub-fullsize-kana, respectively).

› 3.3.6 CSS Writing Modes

With exceptions for the direction and unicode-bidi properties as noted below, the EPUB 3 CSS Profile includes all of the features defined in the CSS Writing Modes Module Level 3 [CSS3WritingModes] specification using -epub- prefixed property names, syntax as defined in [CSS3WritingModes-20110428] and semantics as defined in [CSS3WritingModes].

The direction and unicode-bidi properties from [CSS3WritingModes] are not included in the EPUB 3 CSS Profile. Authors should use appropriate [HTML5] markup to express directionality information instead.

› 3.3.7 Media Queries

The EPUB 3 CSS Profile includes @media and @import rules with media queries as defined in the Media Queries [MediaQueries] specification.

› 3.3.8 CSS Namespaces

The EPUB 3 CSS Profile includes the `@namespace` rule defined in [CSS Namespaces] for declaring the default namespace for a style sheet and for binding prefixes to namespaces.

› 3.3.9 CSS Multi-Column Layout

The EPUB 3 CSS Profile includes all of the features defined in the CSS Multi-column Layout Module [CSSMultiCol] specification with the exception of the `column-span` property.

> CAUTION
>
> Authors should not rely on column behavior in overflow conditions as this behavior is unstable and may change.

> CAUTION
>
> Pagination algorithms are not fully defined in CSS. Authors should therefore expect exact pagination points to vary from Reading System to Reading System.

Reading Systems must treat the `oeb-column-number` property as an alias for the `column-count` property. The use of the `oeb-column-number` property in EPUB Style Sheets is deprecated; this conformance requirement may be removed in the next major version of EPUB.

› 3.3.10 Ruby Positioning

The EPUB 3 CSS Profile includes the `-epub-ruby-position` property as defined below:

Name:	-epub-ruby-position
Value:	`over` | `under` | `inter-character`
Initial:	over
Applies to:	ruby text elements
Inherited:	yes
Percentages:	N/A
Media:	visual
Computed value:	as specified

This property controls the placement of ruby text with respect to its base text. Values have the following meanings:

`over`

Ruby text is positioned on the over side of the ruby base.

`under`

Ruby text is positioned on the under side of the ruby base.

`inter-character`

Ruby text is positioned on the right side of the base text. (This value is typically used for Zhuyin Fuhao (Bopomofo) ruby.)

NOTE

The `-epub-ruby-position` property will become an alias for the `ruby-position` property in the CSS Ruby Module [CSS3Ruby].

› 3.3.11 Display Property Values `oeb-page-head` and `oeb-page-foot`

In addition to the standard values defined for the `display` property in Section 9.2.4 of [CSS2.1], EPUB Style Sheets may specify the values `oeb-page-head` and `oeb-page-foot`.

Reading Systems should present the content of an element assigned `display: oeb-page-head` only as a header, and the content of an element assigned `display: oeb-page-foot` only as a footer. Neither should be presented simply as if it were `inline` or `block`. The way Reading Systems present headers and footers is not defined by this specification (e.g., they may render them in fixed positions as per print layouts or pop them up on demand if only limited screen space is available).

For the purposes of page layout, these display values are similar to block boxes with an absolute position (i.e., a position value of `fixed` or `absolute`). That is, they are removed from the normal flow and a new block box is created with its own flow. Margins, padding, and other block characteristics are determined as if the element had `position: fixed` set.

An element assigned `display: oeb-page-head` or `display: oeb-page-foot` must not be considered in effect while any markup specified before such an element is still being rendered in the same context (for example, if it is on the same page in a paginated context, or in the viewport for a scrolled context). Once in effect, the element must remain in effect until either of the following conditions is true:

- another header or footer (respectively) is in effect instead; or
- no part of its parent element remains presented.

For example, when rendered to a screen with appropriate style settings, the `myhead`-classed `div` element in the following example would become the page header as soon as nothing preceding the containing `div` is displayed, and go out of effect when that `div` is no longer visible:

```
<div>
  <div class="myhead" style="display: none; display: oeb-page-head">
     The OEB Publication Structure: Introduction
  </div>
  <h2>Introduction</h2>
  <p>…</p>
</div>
```

NOTE

The `display` property has its value set to `none` in the preceding example before setting it to `oeb-page-head` to ensure that Reading Systems that do not support this feature do not display the content. This approach is recommended whenever setting the `oeb-page-head` or `oeb-page-foot` values.

4 PLS Documents

4.1 Overview

This section is informative

The W3C Pronunciation Lexicon Specification [PLS] defines syntax and semantics for XML-based pronunciation lexicons to be used by Automatic Speech Recognition and Text-to-Speech (TTS) engines.

The following sections define conformance criteria for PLS documents when included in EPUB Publications, and rules for associating PLS Documents with XHTML Content Documents.

NOTE

For more information on EPUB 3 features related to synthetic speech, refer to Text-to-speech [EPUB3Overview].

4.2 EPUB Publication Conformance

A conformant EPUB Publication must meet all of the following criteria for inclusion of PLS Documents:

- PLS Documents may be associated with XHTML Content Documents. Each XHTML Content Document may contain zero or more PLS Document associations.
- PLS Documents must be associated with the XHTML Content Document to which it applies using the [HTML5] `link` element with its `rel` attribute set to `pronunciation` and its `type` attribute set to the PLS media type (`application/pls+xml`).
- The `link` element `hreflang` attribute should be specified on each PLS `link`, and its value must match the language for which the pronunciation lexicon is relevant [PLS] when specified.
- PLS Documents must meet the content conformance criteria defined in PLS Documents — Content Conformance.
- PLS Documents must be represented and located as defined in EPUB Publication — Content Conformance [Publications30].

Examples

The following example shows two PLS Documents (one for Chinese and one for Mongolian) associated with an XHTML Content Document.

```
<html … >
    <head>
        …
        <link rel="pronunciation" type="application/pls+xml"
hreflang="zh" href="../speech/zh.pls"/>
        <link rel="pronunciation" type="application/pls+xml"
hreflang="mn" href="../speech/mn.pls"/>
    </head>
    …
</html>
```