XSL-FO/CSS COMPARISON

A Data Usability Company
ANTENNA HOUSE

XSL-FO/CSS COMPARISON

Antenna House, Inc.

A Data Usability Company
ANTENNA HOUSE

AH CSS Formatter

XSL-FO/CSS Comparison

© 2022 Antenna House, Inc.

Antenna House, Inc.
500 Creek View Road
Suite 107
Newark, DE 19711
USA
Telephone +1 302-566-7225
sales@antennahouse.com
https://www.antennahouse.com/

This document compares XSL-FO and CSS, so it is available in two versions—formatted using XSL-FO and formatted using CSS—for you to compare. However, the results are so similar that for many pages, you will need the Antenna House Regression Testing System (AHRTS) to find the differences.

The text of this document is marked up in XHTML5 (XML-serialized HTML5). Before being formatted, the XHTML5 is augmented using XSLT to add the table of contents, side tabs, and syntax highlighting.

The CSS version applies CSS to the augmented XHTML5 using AH CSS Formatter to generate PDF. For the XSL-FO version, the augmented XHTML5 is transformed into XSL-FO markup using XSLT and is then formatted using AH XSL Formatter to generate PDF.

CONTENTS

INTRODUCTION

This document compares the specifications for XSL-FO and CSS and provides information about applicable AH Formatter extensions for one or both of XSL-FO and CSS. It is available in two versions—formatted using XSL-FO and formatted using CSS—for you to compare.

Comparing XSL-FO and CSS formatting is not straightforward. XSL implementations are not standing still: XSL formatters are still incrementally improving even though the XSL Recommendation has not been updated since 2006. CSS is definitely not standing still, although some of the modules most relevant to paged media are advancing slowly, if at all, and some paged media features have been removed in more recent Working Drafts.

XSL-FO and CSS with Antenna House Formatter

The Antenna House Formatter family is available in multiple variants with different capabilities: Antenna House Formatter can both format XML or HTML documents using CSS and format XSL-FO; Antenna House XSL Formatter can format documents using XSL-FO; and Antenna House CSS Formatter can format either XML or HTML using CSS. All of these are available as a Lite variant with reduced functionality. The standalone Windows GUI versions of these offer on-screen preview of formatted documents. Please see the Antenna House website for more information.

AH XSL Formatter and AH CSS Formatter cannot be compared strictly on the specifications for XSL-FO and CSS. AH Formatter uses a common layout engine for both XSL-FO and CSS formatting, so many of the features of one technology are implemented as extensions in the other technology. Some features, however, cannot be applied to the other technology because of design decisions made in each specification. For example, XSL-FO and CSS have very different approaches both to selecting page masters and to directing content to the headers and footers on a page.

In addition, AH Formatter implements a large number of original features as extensions to both XSL-FO and CSS. However, some of the extensions can be implemented for one technology but not the other. For example, the '-ah-attr-from()' function for retrieving an attribute value from a named ancestor element is implemented only for CSS because, for XSL-FO, the value to use could be determined by the XSLT transformation that generates the XSL-FO.

Audience

XSL-FO/CSS Comparison is intended for you if you are exploring whether to use XSL-FO or CSS, you know one of XSL-FO or CSS and want to learn more about the other, or if you work with both and want to see where are the differences.

Conventions used in this document

- Property names and property value keywords are enclosed in single quotes: for example, ‘counter-increment’ and ‘none’.
- Pseudo-elements, rule names, and functions are also enclosed in single quotes. Their type is also given, unless it is obvious from the context: for example, “‘::before’ pseudo-element” and “‘@page’ rule”.
- Element names are shown as start tags: for example, `<style>`.
- CSS fragments are shown as monospace text: for example, `float: left;`.
- Emphasized text is shown as text with a yellow background.
- Blocks of sample markup or sample CSS are shown in monospace text:

```
<p>Number of this page =
  <span style="content: counter(page)"></span>
</p>
<p>Total number of pages in this document=
  <span style="content: counter(pages)"></span>
</p>
```

Feedback

Help us to improve this document. Please send any comments, corrections, or suggestions for improvement to support@antennahouse.com.

1 HISTORY

Some of the significant events in the development of XSL-FO, CSS, HTML, and AH Formatter are shown in the following table.

Year	XSL-FO	Antenna House	CSS	HTML
1996	DSSSL		CSS 1	
1998			CSS 2	XHTML 1.0
1999			First CSS 3 drafts	
2001	XSL 1.0	AHF V1.1E		XHTML 1.1
2004				WHAT WG formed
2006	▪ XSL 1.1 ▪ XSL-FO 2.0 Workshop			HTML WG rechartered
2008		AHF V5.0		
2009	XSL-FO 2.0 Design Notes			
2011			CSS 2.1	
2012		AHF V6.0		
2018		AHF V6.6	CSS Snapshot 2018	
2019				W3C cedes HTML5 to WHAT WG
2020		AHF V7.0	CSS Snapshot 2020	

Year	XSL-FO	Antenna House	CSS	HTML
2021		▪ AHF V7.1 ▪ AHF V7.2		

Cascading Style Sheets, level 1, [CSS1] became a Recommendation in 1996. CSS was co-invented by Håkon Wium Lie and Bert Bos. CSS 1 built upon previous style sheet proposals, including earlier separate proposals by Lie and Bos. References to the earlier proposals are at: the W3C *Historical Style Sheet proposals* [HSSP] page; the *The CSS saga* [LIEBOS2E] chapter from Lie and Bos's CSS book: and Lie's Ph.D thesis [LIE]. The goals for CSS stated in 1995 [1995] include: "CSS supports stream-based (or 'incremental' formatting) where possible"; "CSS offers both readers and authors control over the style"; as well as "avoiding an uncontrolled growth of HTML extensions".

DSSSL [DSSSL], the stylesheet language for SGML, became an International Standard in 1996. DSSSL defines a tree transformation process followed by a formatting process. In practice, the tree transformation process was not widely used. However, the most widely-used DSSSL formatter had an extension for performing a transformation as an alternative to formatting.

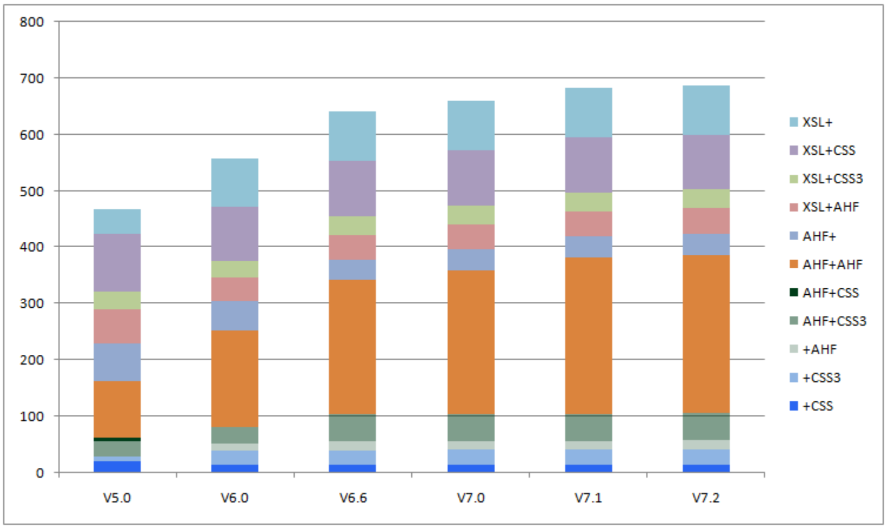
Styling for XML was always part of the development of XML. It was referred to by Jon Bosak, original XML Working Group (WG) Chair, in 1997 as "xml-style (Part 3 of the XML specification suite)" [XS] and "Part 3 of the W3C XML suite of specifications for the use of SGML, HyTime, and DSSSL subsets on the World Wide Web" [XSRTF].

'xml-style' became 'Extensible Stylesheet Language' (XSL). The XSL WG was formed in 1998, and its charter stated its intention "to define a style specification language that covers *at least* the formatting functionality of both CSS and DSSSL." [XSLCHARTER] XSL encompasses both transformation and formatting, but transformation proved generally useful, and the transformation component was broken out as the XSLT series of Recommendations. The bulk of the XSL 1.0 and XSL 1.1 Recommendations concern the formatting objects (FO) and their properties. The transformation component is covered by a short chapter that refers to the then-current XSLT 1.0 Recommendation. The XSL properties align as much as possible with the corresponding CSS properties, in keeping with the commitment in the XSL WG charter.

The need for consistency in properties was stated to be an architectural principle for the web in the *Consistency of Formatting Property Names, Values, and Semantics* TAG Finding [TAG] published in 2002.

Antenna House first proposed greater compatibility between XSL-FO and CSS, especially compatibility with the CSS 3 drafts, at *International Workshop on the future* *of the Extensible Stylesheet Language (XSL-FO) Version 2.0* [XSL2006] at Heidelberg, Germany, in 2006. That proposal was not supported by the workshop participants.

AH Formatter V5.0, released in 2008, was the first AH Formatter version to support both XSL-FO and CSS. AH Formatter is still the world's only XSL-FO and CSS formatter, and successive releases have added features for both XSL-FO and CSS. The following figure illustrates the relative proportions of the number of properties available for XSL-FO and CSS formatting. You can see that the majority of the properties available to either XSL-FO or CSS are AH Formatter original extensions or are properties from one technology made available in the other technology.



Key

Category	Definition
XSL+	XSL-FO property
XSL+CSS	Common to both XSL-FO and CSS 2
XSL+CSS3	Common to both XSL-FO and CSS 3
XSL+AHF	XSL-FO property also implemented as CSS extension
AHF+	AH Formatter extension for XSL-FO only
AHF+AHF	AH Formatter extension for both XSL-FO and CSS
AHF+CSS	CSS 2 property also implemented as XSL-FO extension

Category	Definition
AHF+CSS3	CSS 3 property also implemented as XSL-FO extension
+AHF	AH Formatter extension only for CSS
+CSS3	CSS 3 property
+CSS	CSS 2 property

2 VIEWPOINTS

How a user compares XSL-FO and CSS can depend on their initial exposure to markup and styling as much as or more than on the relative merits of either technology. This is an informal summary of how users of CSS can see XSL-FO, and vice-versa.

CSSer's view of XSL-FO

- *Source XML or HTML must be transformed into the XSL-FO vocabulary*

  Transformation has advantages and disadvantages. CSS was designed to support stream-based or incremental formatting [1995], which is part of why CSS selectors cannot match 'down' into the content of the current element or 'forward' to the structure of following elements. A transformation stage, on the other hand, typically (although less so after XSLT 3.0 added streaming for XSLT) requires the whole document to be available, but it does allow style decisions to be made based on the whole document. Transformation also allows the content to be duplicated and reordered to, for example: generate tables of contents and indexes; sort the rows of a table; or calculate subtotals.

  Formatting paged media using CSS 3 typically requires some form of transformation anyway. This includes generating the running elements that are taken out of the flow and used in headers and footers, as well as generating tables of contents and indexes.
- *Separate attributes for each property is verbose*

  XSL-FO is designed to be the result of an XSLT transformation. XSL-FO was not meant to be read, let alone authored, by humans. It does happen, of course. When XSL-FO is serialized as XML, it is straightforwardly usable in XML editors, etc., and it is as human-legible and as reasonably clear as any other XML or HTML document.

  When XSL-FO is generated using XSLT, the XSLT is less verbose than the XSL-FO that it creates. An FO and its properties can be generated from literal elements and attributes inside an XSLT template, but a single XSLT template can be used many times to generate multiple copies of the FO. Attribute sets in the XSLT are defined once and are used in multiple places in the XSLT. An XSLT template can contain more than just literal elements and attributes to copy to the result: the XSLT can include as much logic as is necessary to be able to conditionally generate the correct FOs, their correct properties, and the correct property values for any given context.

- *JavaScript could be used instead of XSLT*

  Yes, JavaScript can be used to generate tables of contents, indexes, and so on by manipulating the DOM of the document, but whatever JavaScript solutions exist have been bespoke code. Over the last 20 years, there hasn't been a standard, widely used, general purpose JavaScript library that matches both the path-matching ability of XPath and the declarative templating mechanism of XSLT. AH Formatter will format a correct XSL-FO file no matter how it was created.
- *XSL-FO properties inherit but do not cascade*

  The role of the cascade is taken by the XSLT transformation.

  In XSLT:
  - One XSLT stylesheet can import another, similarly to how '@import' imports another CSS style sheet
  - The 'match' patterns in XSLT templates have a default priority based on the specificity of their XPath pattern, similarly to more specific CSS selectors overriding more general ones. In addition, an XSLT template may be given an explicit numeric priority.
  - When there is more than one matching template with the same precedence, the one that occurs last is used, similarly to two CSS rules that have the same weight
  - An XSLT 'attribute set' is a named set of attribute definitions. The attribute definitions are reevaluated in each context where the attribute set is used. Multiple `<xsl:attribute-set>` with the same name are aggregated, with definitions for individual attributes in an `<xsl:attribute-set>` with higher precedence overriding definitions in other `<xsl:attribute-set>` that have lower precedence. An `<xsl:attribute-set>` can also reuse attribute definitions in other, named attribute sets. Attribute sets are not quite the same as on-the-fly building up of the properties to apply in a particular context based on the cascade of '@import' rules and specificity of CSS selectors, but they do make it easy to apply a group of properties in particular contexts.
- *FOs are like elements with fixed 'display' property values*

  In CSS, all that you have is the source document (unless, as noted previously, you have augmented the original source document to create tables of contents, etc.), and the 'display' property, like all properties, is applied as the document is formatted. With transformation before formatting, the decisions about how to format each part of the source document are part of the transformation, so there is no need for on-the-fly changes using a ‘display’ property.

- *XSL-FO does not have variables*

  CSS gained variables only recently, but variables are not needed in XSL-FO. XSLT has variables (which have scope and which can be passed between templates), so any calculations based on variables or any substitution of constant values can be handled in XSLT.
- *XSL-FO can’t be used for both web and print*

  XSL 1.1 defines how to handle a page that extends indefinitely in one or both dimensions and it also includes some interactive FOs, but neither of those has been widely implemented because few users have ever expressed interest in them.
- *CSS is easier to learn that XSL-FO*

  The basics of CSS syntax are easy to learn, but there is an expanding list of CSS selectors and pseudo-elements to be remembered, plus many CSS properties have their own micro-syntax for expressing their value. XSLT and XSL-FO are XML vocabularies, so most of their syntax is easily understood by anyone who can read XML or HTML. However, XSLT and XSL-FO attempt more than CSS, so it is not surprising that there is more to learn. XSLT is a Turing-complete declarative language for specifying transformations, whereas CSS is a declarative language for specifying styles. People who are used to imperative programming languages where they specify every step of the program can have trouble adapting to the XSLT model where the structure of the source document can determine the program flow.

  XSL-FO provides more control than CSS over, for example, the selection of page masters and the content of headers and footers, so it has more FOs and properties for those areas.
- *There are more CSS users than XSL-FO users*

  True. However, comparatively few CSS users are familiar with using CSS to generate paged media. If you want to learn more about CSS for paged media, see *Introduction to CSS for Paged Media* [ITCFPM], available from the Antenna House website.

XSL-FOer’s view of CSS

- *CSS ‘just decorates the tree’*

  Decorating the tree of elements fits with the original goal of CSS to support streaming or incremental formatting. [1995] XSLT, in contrast, can use any part of the document, or of a different document, when deciding which template to use in the current context. By using modes, XSLT can process the document or parts of the document multiple times in different ways.

The structure of the XSL-FO document does not need to match the structure of the source document: the XSLT stage can generate literal elements as well as copy all or part of any node in the source document. An XSLT template can select which nodes to process next rather that just processing the children of the current node.

- *CSS selectors won't look 'down' or 'forward'*

  CSS selectors can match on the current element, its class (or classes), its attribute values, its ancestor elements, and its preceding elements. Selectors won't, however, match on the string value of an element or on any aspect of the type and arrangement of an element's descendent elements. In contrast, the entire document (and possibly other, external documents) is available to the XSLT processor, and style decisions can be made based on more than just the context of the current element.
- *CSS only operates on elements*

  CSS selectors only apply to elements. In contrast, XSLT templates can match on text nodes, text nodes with particular values, or text nodes in particular contexts (possibly with particular values) and can generate FOs based on those text nodes. XSLT can similarly match on comments and processing instructions and generate FOs.

3 FEATURE COMPARISON

The following table provides an overview of the differences between XSL-FO and CSS. More details are provided in the sections following the table. For ease of comparison, the sequence in both the table and the rest of this document follows the chapter sequence in *Introduction to CSS for Paged Media* [ITCFPM].

Features that are AH Formatter extensions are shown as green text.

Feature comparison

Section	XSL-FO	CSS
Box layout	XSL and CSS both generate rectangular boxes by applying styles to markup. Their features are mostly identical.	
Page layout	▪ Page masters defined in `<fo:simple-page-master>` FOs ▪ Page masters selected in a predefined sequence using `<fo:page-sequence-master>` FOs ▪ AH Formatter supports nested `<fo:page-sequence>` as a child of `<fo:flow>` ▪ AH Formatter adds `<axf:spread-page-master>` for defining pages with regions that spread across two pages ▪ “Flow maps” allow content to flow into multiple separate regions on the same page	▪ Pages defined in ‘@page’ rules ▪ Elements can specify a ‘@page’ rule to use ▪ Adjacent or nested elements with different ‘@page’ selection causes a page break with the change to the new ‘@page’

Section	XSL-FO	CSS
Headers & footers	▪ Four 'outer' regions may be defined for each `<fo:simple-page-master>` ▪ Default name for each region may be overridden ▪ Block-level content from `<fo:static-content>` directed to named region (if defined for current page) ▪ Variable content specified with `<fo:marker>` and retrieved with `<fo:retrieve-marker>` ▪ Retrieved content does not inherit properties from its original location	▪ 16 page-margin boxes on every page ▪ Page-margin box names are fixed and each box has predefined default alignment ▪ Content is either retrieved from a running element using 'running()' or is any combination of fixed strings, counters, and strings retrieved using 'string()' ▪ Running elements inherit properties from their original location
Multiple columns	▪ Only in `<fo:region-body>` and, as AH Formatter extension, in `<fo:block-container>` ▪ 'column-count' specifies fixed number of columns ▪ AH Formatter extensions for column balancing and appearance of column rule	▪ Any block-level element ▪ 'column-count' specifies fixed number of columns ▪ Setting 'column-width' generates as many columns as will fit ▪ AH Formatter extensions for column balancing and appearance of column rule
Keeps & breaks	▪ Keywords plus numeric values to indicate weight ▪ AH Formatter extension for maximum height for a keep-together condition	▪ Keywords only ▪ AH Formatter extension for maximum height for a keep-together condition
Paragraph setting	▪ Same text alignment control, including AH Formatter extensions ▪ AH Formatter extensions for baseline grid ▪ Overflow extensions	▪ Same text alignment control, including AH Formatter extensions ▪ AH Formatter extensions for baseline grid

Section	XSL-FO	CSS
Footnotes & sidenotes	▪ A 'footnote-reference-area' is implicit in the area generated by an `<fo:region-body>` ▪ Footnote number expected to be included in XSL-FO ▪ Sidenotes are an AH Formatter extension ▪ Sidenote number expected to be included in XSL-FO	▪ '@footnote' rule for footnote area included in default `html.css` ▪ Footnote numbering is automatic ▪ '@sidenote' rule for sidenote area included in default `html.css` ▪ Sidenote numbering is automatic
Tables	▪ XSL 1.1 tables based on CSS 2 tables ▪ A table with a caption requires `<fo:table-and-caption>` containing both `<fo:table-caption>` and `<fo:table>` ▪ Precedence of collapsing borders can be set in the XSL-FO ▪ AH Formatter extensions for table footnotes, accessibility, and improved control of breaks	▪ CSS still uses CSS 2 tables ▪ AH Formatter implements XSL-FO properties for cell content alignment and table header and footer behavior at breaks ▪ AH Formatter extensions for accessibility and improved control of breaks ▪ '-ah-reference-orientation' applies to tables and table cells
Lists	▪ Multiple FOs for parts of a list ▪ List markers expected to be included in XSL-FO ▪ Numeric markers can be formatted using counter styles or other formats	▪ Any element may have `display: list-item;` to render as a list item ▪ `::marker` pseudo-element for list item marker ▪ Marker in an ordered list typically generated using a counter
Character setting	Equivalent capabilities. Many properties are common to XSL-FO and CSS. Properties that are not defined in a technology are implemented as AH Formatter extensions, plus there are multiple original AH Formatter extensions implemented for both XSL-FO and CSS.	

Section	XSL-FO	CSS
Japanese text composition	Equivalent capabilities. Many properties are common to XSL-FO and CSS. Properties that are not defined in a technology are implemented as AH Formatter extensions, plus there are multiple original AH Formatter extensions implemented for both XSL-FO and CSS.	
Cross-references	▪ `<fo:basic-link>` has ‘internal-destination’ and ‘external-destination’ properties ▪ Only one of ‘internal-destination’ and ‘external-destination’ should be specified ▪ Content of cross-reference, including section number, etc., is expected to be included in XSL-FO ▪ Page number of target generated using `<fo:page-number-citation>` ▪ `<fo:page-number-citation-last>` generates page number of last area generated by an FO ▪ AH Formatter extensions for physical page number, page numbers in reverse sequence, and relative page number difference	▪ `href` used for both internal and external links ▪ Text (but not styling or contained markup) of target can be retrieved using ‘target-text()’ ▪ Number (e.g., section number) and page number can be calculated using ‘target-counter()’ ▪ AH Formatter extensions for physical page number and page numbers in reverse sequence
Image positioning	AH Formatter provides equivalent extensions for both XSL-FO and CSS that provide better capabilities than what is defined in either XSL-FO or CSS.	
MathML & SVG graphics	▪ MathML and SVG are not part of XSL-FO but can be included using `<fo:instream-foreign-object>` or referred to using `<fo:external-graphic>` ▪ AH Formatter provides custom MathML3 and SVG renderers	▪ MathML and SVG are part of HTML5 ▪ AH Formatter provides custom MathML3 and SVG renderers

Section	XSL-FO	CSS
Counters	▪ XSL 1.1 has a limited ability to format page numbers using ‘format’ and other properties that are based on the *Number to String Conversion Attributes* defined in XSLT 1.0 [XSLT10] ▪ All other formatted numbers are expected to be included in the XSL-FO ▪ AH Formatter implements CSS 3 counter styles, including defining custom counter styles in the XSL-FO, to generate numbers when formatting ▪ Counter styles can be used on all block-level and inline-level formatting objects	▪ Numbers based on the element structure can be generated using ‘counter()’ ▪ Counters can be incremented or reset at arbitrary element starts or at page boundaries using ‘counter-increment’ and ‘counter-reset’ properties ▪ CSS Counter Styles 3 [CSS-CounterStyles] defines many formats for presentation of counter values as well as a mechanism for custom counter styles
Color	▪ RGB, RGBA, HSL, HSLA, CMYK, CMYKA colors ▪ Extended color names from CSS ▪ Gradient functions from CSS3 ▪ ICC color profiles	▪ RGB, RGBA, HSL, HSLA, CMYK, CMYKA colors ▪ Extended color names ▪ Radial and linear gradients ▪ ICC color profiles
Borders & background	Equivalent capabilities. Many properties are common to XSL-FO and CSS. Properties that are not defined in a technology are implemented as AH Formatter extensions, plus there are multiple original AH Formatter extensions implemented for both XSL-FO and CSS.	
PDF output	▪ PDF bookmarks from explicit `<fo:bookmark-tree>` ▪ Bookmarks can link to anywhere inside or outside the document ▪ Bookmarks can be built from document structure ▪ PDF forms ▪ Multiple PDF variants, including PDF/A, PDF/X, and PDF/UA	▪ PDF bookmarks from elements with ‘bookmark-level’ property ▪ Bookmarks can link to anywhere inside or outside the document ▪ PDF forms ▪ Multiple PDF variants, including PDF/A, PDF/X, and PDF/UA

Section	XSL-FO	CSS
	▪ XSL-FO can be output as multiple volumes	
Indexes	▪ Multiple FOs specific to indexes ▪ Properties for merging repeated and consecutive page numbers in index entries	▪ Extension properties for merging repeated and consecutive page numbers can be used on any block-level element
AH Formatter unique features	▪ Line numbers ▪ Tab stops ▪ Hyphenation information included in XSL-FO ▪ Formatter configuration settings in XSL-FO ▪ Two-pass formatting for large documents	▪ Line numbers ▪ '-ah-attr-from()' for value of attribute on an ancestor element

4 BOX LAYOUT

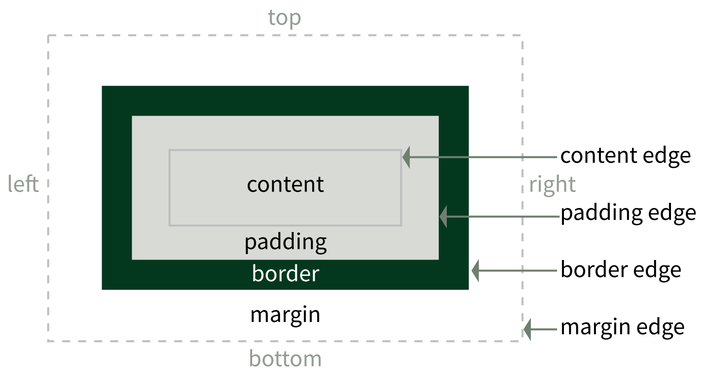
XSL and CSS both generate rectangular boxes by applying styles to markup. In XSL-FO, the FO name—such as `<fo:block>` or `<fo:list-item-label>`—determines what type of areas the FO generates as well as which properties apply to the generated areas. Some FOs, however, either do not generate any areas or simply return the areas generated by their descendent FOs. In CSS, the 'display' property determines the type of box or boxes that the element generates, which also determines which properties apply to the current element.

XSL requires that all FOs are present in the document: the XSLT stage is expected to generate all of the elements for the FOs. (XML does not allow tags to be omitted, but that is not relevant to XSL because FO documents were not expected to be serialized as XML documents before formatting.) The FOs, in turn, generate all of the areas in the formatted document. The main, barely-noticed exception is when a block-level FO contains both text and block-level FOs: any continuous run of text is contained by an 'anonymous block' so that areas generated by the outer FO contain only block areas and not a mixture of inline areas and block areas. Delimited runs of inline text are created for text that has a different orientation—such as Latin text in vertical Japanese text—or that has a different directionality—such as Latin text in an Arabic document.

HTML allows the start and/or end tags of some HTML elements to be omitted and the HTML parser will infer them. XHTML requires that start-tags and end-tags match (and that the single tag for an empty element ends with `/>`), but some elements can be inferred by other content: for example, `<table>` can either directly contain `<tr>` or contain `<tbody>` that contains `<tr>`.

CSS, similarly to XSL, will generate anonymous boxes around runs of text as required to ensure that a block box does not contain a mix of inline boxes and block boxes or inline boxes and text. CSS can also generate some anonymous boxes when formatting tables: for example, when an element with `display: table;` contains an element with `display: table-row;` that contains text, the formatter will generate an anonymous row-group box and an anonymous table-cell box so that the formatted elements are consistent with the table model.

Most boxes in both XSL and CSS can have a border. Padding can be specified to create space between the actual content of the box and its border, and margins can be specified to create space between the border and adjacent boxes. XSL defines these in terms of rectangles whereas CSS defines them in terms of boxes with edges, but they are compatible.



Areas and edges of a CSS box

A box's margin and padding as well as border color, style, and width can be individually specified for each side. In CSS and, with an AH Formatter extension, in XSL-FO, the box can have rounded corners. The radius of each side of each rounded corner can be individually specified. Shorthand properties allow the margin, padding, or one or more border properties for multiple sides to be set at once.

CSS originally defined the padding, etc., properties based on a box's appearance on the page (or screen). For example, 'padding-left' and 'padding-right' shorthand properties for the left and right paddings, respectively. For left-to-right text, such as English, these correspond to the start and end of a line of text, respectively. However, for right-to-left text, such as Arabic, they correspond to the end and start of a line of text, respectively, and for vertical Japanese text, they correspond to the sides but neither the start nor end of a line of text. There is a current CSS Working Draft that adds 'logical' properties—such as 'padding-inline-start' and 'padding-inline-end'—but that is not yet widely used. The logical properties map to the corresponding properties for the left, right, top, and bottom sides based on the values of the CSS 'writing-mode', 'direction', and 'text-orientation' properties. For example, specifying 'padding-inline-start' will generate the padding at the start side of a block irrespective of whether the current language is English, Arabic, or Japanese.

XSL defines padding, etc.—and even the position of the header and footer areas—in terms of relative directions. XSL defines 'padding-left' and 'padding-right', etc., in common with CSS, but also defines 'padding-start', 'padding-end', 'padding-before', and 'padding-after' (and so on for other border and padding properties). The correspondence between ‘padding-left’ and ‘padding-right’, etc., and the relative properties depends on the value of the XSL ‘writing-mode’ property. For the border and padding properties, if both an absolute and a relative property are specified for an FO, the absolute property has precedence. It is slightly more complicated for the margin properties because the values of the XSL ‘start-indent’ and ‘end-indent’ properties are a function of the corresponding margin, border, and padding widths.

XSL can change the orientation of a block by specifying the ‘reference-orientation’ property.

Boxes can be specified to have a width and/or height. XSL shares the ‘min-width’, ‘width’, ‘max-width’, ‘min-height’, ‘height’, and ‘max-height’ properties with CSS, but in XSL, these absolute properties are mapped to the relative ‘inline-progression-dimension’ and ‘block-progression-dimension’ properties. ‘inline-progression-dimension’ and ‘block-progression-dimension’ (and ‘leader-length’) can be specified as a <length-range> with minimum, optimum, and maximum components rather than a single fixed length. In CSS, non-‘auto’ width and height values specify the size of either the content rectangle or the border rectangle of the box, depending on the value of the ‘box-sizing’ property.

Boxes are ordinarily positioned one after the other down the page or along the line, with obvious exceptions for tables, list item labels, and floated elements. The ‘position’ property (which, in XSL, is a shorthand for the ‘relative-position’ and ‘absolute-position’ properties) allows a box to be offset relative to its position in the flow or positioned relative to its reference area or to the page. In CSS, ‘position’ can be specified for any element. In XSL, ‘relative-position’ can be specified on a range of FOs, but ‘absolute-position’ can only be specified on `<fo:block-container>`. The ‘z-index’ property specifies the relative layering of areas generated from positioned elements in CSS and from `<fo:block-container>` in XSL.

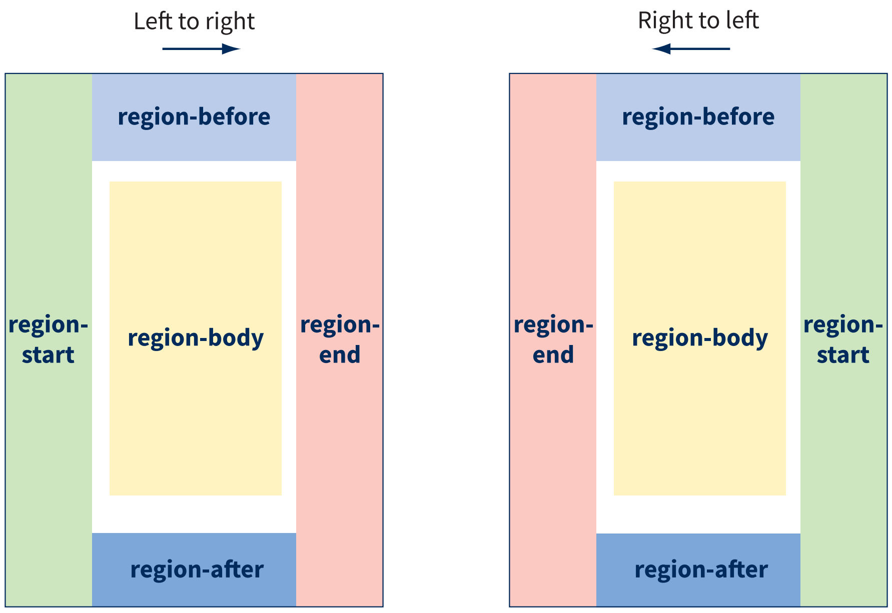
5 PAGE LAYOUT

Formatting on paged media obviously requires pages on which to do the formatting. XSL-FO and CSS have different mechanisms for defining the size and other characteristics of a page. Many documents use more than one type of page—for example, left-hand and right-hand pages can have different margins as well as different headers and footers—so XSL-FO and CSS each have mechanisms for choosing between page types.

Page master

A 'page master' is the specification of the size and overall layout of a page. A formatted document where all the pages are essentially the same could have just one page master. A formatted document could require multiple page masters if it has, for example: title page; different margins on right and left pages; one-column body text and a multi-column index; landscape pages for wide tables; or wider margins on table of contents pages.

Page masters in XSL-FO are defined using `<fo:simple-page-master>`. An `<fo:simple-page-master>` can define up to five types of regions. `<fo:region-body>` receives the main content of the document, and the other regions receive the content that makes up the headers and footers on each page. Omitted regions do not generate any areas when the `<fo:simple-page-master>` is used to generate pages. XSL 1.1 added the ability to define multiple `<fo:region-body>` on one page and to direct one or more flows to one or more regions on the page using `<fo:flow-map>`.



Page regions in XSL-FO

Page masters in CSS are defined using an '@page' rule. [CSS3-Page]

Page region layering

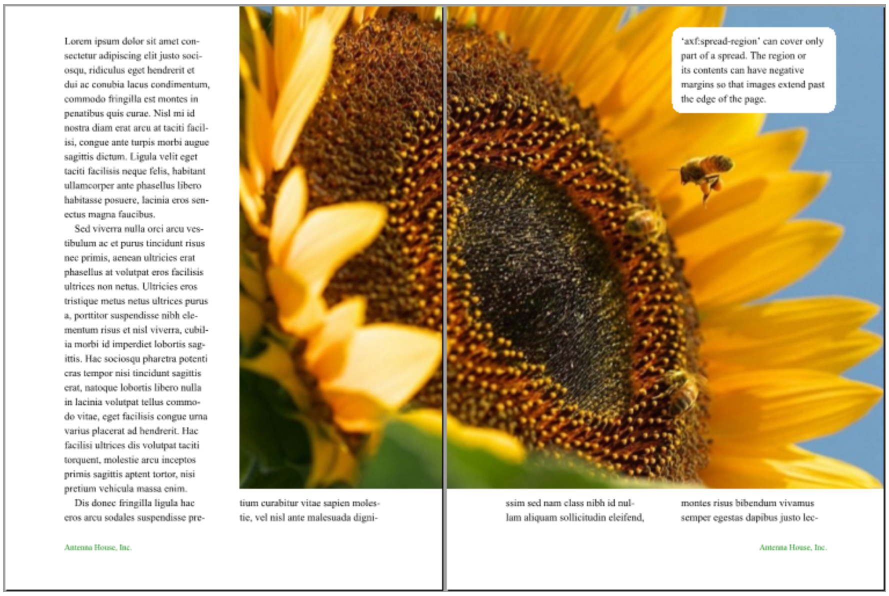
XSL 1.1 specifies that the regions follow a fixed sequence in the XSL-FO. AH Formatter relaxes this and allows the regions to appear in any sequence. AH Formatter renders the areas for the regions in the sequence in which the regions appear in the XSL-FO: 'z-index' does not apply to the regions, so the declaration sequence is a way to control which regions can overlap other regions. If `<fo:region-body>` is last, it can overlap content from the other regions.

'z-index' applies to CSS page-margin boxes. It can be used to control the layering of page-margin boxes that have content that overlaps.

Spreads

A 'spread' is two facing pages.

AH Formatter has an `<axf:spread-page-master>` extension for XSL-FO. The background of an `<axf:spread-page-master>` extends across both pages, and regions defined in the `<axf:spread-page-master>` can span across more than one page.



Lorem ipsum dolor sit amet consectetur adipiscing elit justo sociosqu, ridiculus eget hendrerit et dui ac conubia lacus condimentum, commodo fringilla est montes in penatibus quis curae. Nisl mi id nostra diam erat arcu at taciti facilisi, congue ante turpis morbi augue sagittis dictum. Ligula velit eget taciti facilisis neque felis, habitant ullamcorper ante phasellus libero habitasse posuere, lacinia eros senectus magna faucibus.

Sed viverra nulla orci arcu vestibulum ac et purus tincidunt risus nec primis, aenean ultricies erat phasellus at volutpat eros facilisis ultrices non netus. Ultricies eros tristique metus netus ultrices purus a, porttitor suspendisse nibh elementum risus et nisl viverra, cubilia morbi id imperdiet lobortis sagittis. Hac sociosqu pharetra potenti cras tempor nisi tincidunt sagittis erat, natoque lobortis libero nulla in lacinia volutpat tellus commodo vitae, eget facilisis congue urna varius placerat ad hendrerit. Hac facilisi ultrices dis volutpat taciti torquent, molestie arcu inceptos primis sagittis aptent tortor, nisi pretium vehicula massa enim.

Dis donec fringilla ligula hac eros arcu sodales suspendisse pretium curabitur vitae sapien molestie, vel nisl ante malesuada dignissim sed nam class nibh id nullam aliquam sollicitudin eleifend, montes risus bibendum vivamus semper egestas dapibus justo lec-

Antenna House, Inc.

'axf:spread-region' can cover only part of a spread. The region or its contents can have negative margins so that images extend past the edge of the page.

Antenna House, Inc.

CSS does not have any support for specifying spreads. The AH Formatter extension is not available with CSS because CSS does not provide the same control over the sequence of page masters as in XSL-FO.

Page size

Page size in XSL is specified with the ‘page-height’ and ‘page-width’ properties, or with the ‘size’ shorthand for setting both at once. The XSL 1.1 ‘size’ is based on the CSS 2 ‘size’ property, but that ‘size’ was removed in CSS 2.1 and not reinstated in CSS 2.2. AH Formatter implements a superset of the CSS Paged Media Module Level 3 ‘size’ property, which adds keywords for several standard page sizes.

Page size in CSS is specified using the ‘size’ property. The extended set of page size keywords is similarly available in CSS.

Selecting page masters

The content in an XSL-FO document is contained in one or more `<fo:page-sequence>` FO. Each `<fo:page-sequence>` specifies which `<fo:simple-page-master>` to use or, indirectly, which set of `<fo:simple-page-master>` to use. The `<fo:page-sequence>` can refer to a `<fo:page-sequence-master>` that sets out which `<fo:simple-page-master>` to use and in what sequence.

The sequence of page masters can be any sequence of:

- One page master, possibly with a limit to the number of times it can be repeated

- A choice between alternative page masters, possibly with a limit to the number of times that the choice can be repeated. The references to the alternative page masters each have three sub-conditions for deciding whether to use that alternative. When a new page is to be generated, the alternatives are considered in sequence, and the first alternative for which all three subconditions is true is chosen.

  The sub-conditions are:

  - The page is either first, last, only (both first and last), neither first nor last, or any page in its page sequence
  - The page number is either odd, even, or any (i.e., whether it is odd or even will not affect the choice)
  - The page is blank, is not blank, or is any (i.e., whether or not it is blank will not affect the choice)

  The sub-conditions are expressed as properties. The ‘any’ value is the default for each property, so when choosing an alternative, you only need to specify the sub-conditions that affect the choice.

For example, a document with different margins on odd and even pages could have two `<fo:simple-page-master>` named ‘OddPage’ and ‘EvenPage’. The `<fo:page-sequence-master>` for choosing the correct page master could be:

```
<fo:page-sequence-master master-name="PageMaster">
  <fo:repeatable-page-master-alternatives>
    <fo:conditional-page-master-reference master-reference="OddPage"
                                          odd-or-even="odd"/>
    <fo:conditional-page-master-reference master-reference="EvenPage"
                                          odd-or-even="even"/>
  </fo:repeatable-page-master-alternatives>
</fo:page-sequence-master>
```

(Note that `odd-or-even="even"` is not strictly necessary here because any page that did not match the previous `odd-or-even="odd"` is, by definition, an even page.)

CSS does not have a separate mechanism for predetermining the sequence of page masters. (Even in XSL-FO, it is only partially predetermined, because which page is the last page, or which pages are blank, is not known until the document is formatted.)

In CSS, any element that can cause a page break before or after itself can cause a change in ‘@page’ rule by specifying the ‘page’ property with a value that is different from the value of its parent or preceding sibling element. As an extension to the current CSS definition, AH Formatter allows a sequence of multiple names in the ‘page’ property, and AH Formatter uses each specified ‘@page’ rule in turn when generating pages. If the element generates more pages than there are names in the ‘page’ property value, AH Formatter continues to use the last name for the remainder of the pages.

Nested page sequences

XSL 1.1 does not allow page sequences to be nested: all of the page masters to use and conditions for their sequence of use are determined by the `<fo:page-sequence-master>` referred to by an `<fo:page-sequence>`.

AH Formatter allows `<fo:flow>` to contain children `<fo:page-sequence>` FOs. Formatting a nested `<fo:page-sequence>` is equivalent to ending the containing `<fo:page-sequence>`, formatting the `<fo:page-sequence>` as if it were another top-level `<fo:page-sequence>`, and then formatting the FOs after the `<fo:page-sequence>` as if in a copy of the original containing `<fo:page-sequence>`.

CSS does not differentiate between top-level elements and lower-level elements that can change the ‘@page’ that is being used. As stated previously, any element that can cause a page break before or after itself can cause a change in the ‘@page’ rule that is being used.

Multiple body regions on one page

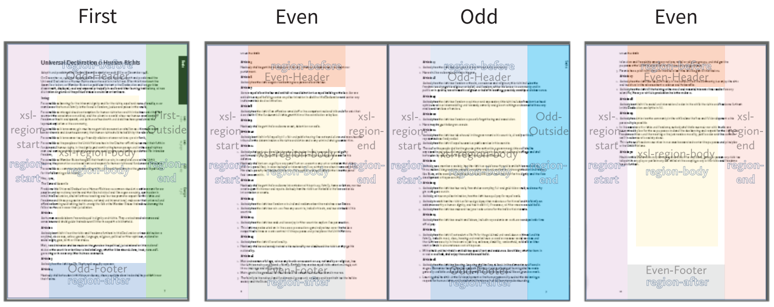
XSL 1.1 added the ability to define multiple `<fo:region-body>` in an `<fo:simple-page-master>` and also multiple `<fo:flow>` within an `<fo:page-sequence>`. One or more `<fo:flow>` can be directed to one or more `<fo:region-body>` on a page. Which `<fo:flow>` are directed to which `<fo:region-body>`, as well as the sequence in which multiple `<fo:flow>` are directed to one (or more) `<fo:region-body>`, is controlled by an `<fo:flow-map>`, which was also added in XSL 1.1.

CSS does not have a comparable feature at present.

6 HEADERS & FOOTERS

XSL-FO

The areas generated from the four 'outer' regions—`<fo:region-before>`, `<fo:region-after>`, `<fo:region-start>`, and `<fo:region-end>`—receive content from `<fo:static-content>`. Each region has a default name based on its type, but different 'region-name' values can be specified. An `<fo:static-content>` directs its content to a region with the same 'region-name' value as the 'flow-name' property value of the `<fo:static-content>`. This makes it possible to set up the static content for multiple sorts of pages and use only the applicable static content on each page. It also makes it possible to, for example, direct the contents of one `<fo:static-content>` to the outer, left-hand region on left-hand pages and to the outer, right-hand region on right-hand pages.



```
<fo:static-content flow-name="Even-Header">
  <fo:block>UDHR in Unicode</fo:block>
</fo:static-content>
<fo:static-content flow-name="First-Outside">
  <fo:block-container reference-orientation="270">
    ...
  </fo:block-container>
</fo:static-content>
<fo:static-content flow-name="Odd-Outside">
  <fo:block-container reference-orientation="270">
    ...
  </fo:block-container>
</fo:static-content>
```

```
<fo:static-content flow-name="Odd-Header">
  <fo:block>...</fo:block>
</fo:static-content>
<fo:static-content flow-name="Even-Footer">
  <fo:block><fo:page-number/></fo:block>
</fo:static-content>
<fo:static-content flow-name="Odd-Footer">
  <fo:block><fo:page-number/></fo:block>
</fo:static-content>
<fo:flow flow-name="xsl-region-body" id="sco">
...
</fo:flow>
```

Variable content that should appear in the header or footer, such as the current chapter title, is declared in an `<fo:marker>`. Any number of `<fo:marker>` can be included as the first children of `<fo:flow>` and of many of it descendent FOs. An `<fo:marker>` can contain any combination of text, inline-level FOs, and block-level FOs, but it does not generate any areas. Instead, its children are available to be retrieved and formatted by an `<fo:retrieve-marker>` that is within an `<fo:static-content>`.

As stated previously, the applicable `<fo:static-content>` provides the complete content for one of the outer regions on the page, and it is only those applicable `<fo:static-content>` that actually retrieve and format the children of an `<fo:marker>`. An `<fo:static-content>` contains one or more block-level FOs. One or more `<fo:retrieve-marker>` can be present as the only content of the `<fo:static-content>` or as siblings of block-level FOs. The block-level FO or FOs can contain any number of `<fo:retrieve-marker>` in any part of their content.

The correspondence between the 'marker-class-name' property on the `<fo:marker>` and the 'retrieve-class-name' property on the `<fo:retrieve-marker>` connects an `<fo:marker>` and an `<fo:retrieve-marker>`. Any name token can be used; for example, 'chapter-title' and 'xyzzy' are equally allowed. Sibling `<fo:marker>` cannot use the same 'marker-class-name' value.

6

If the `<fo:marker>` for the chapter title is:

```
<fo:marker
  marker-class-name="chapter-title">A<fo:inline
    font-variant="swash(1)">&</fo:inline>B</fo:marker>
```

then the `<fo:static-content>` of the header can be:

```
<fo:static-content flow-name="xsl-region-before">
  <fo:block font-size="10pt" text-align="center">
    <fo:retrieve-marker retrieve-class-name="chapter-title" />
  </fo:block>
</fo:static-content>
```

The content of the `<fo:marker>` is formatted in place of the empty `<fo:retrieve-marker>`. The content is formatted as if it had the same ancestors as the `<fo:retrieve-marker>`; that is, the content inherits property values from the context of the `<fo:retrieve-marker>` within its `<fo:static-content>` and not from the context of the `<fo:marker>` in its parent FO.

There can be multiple `<fo:marker>` with the same 'marker-class-name' in the FO document (just not with the same parent FO). Multiple FOs that have `<fo:marker>` children with the same 'marker-class-name' could generate areas on the same page. To continue with the example, unless each chapter starts on a new page, there could be both the end of one chapter and the start of the next chapter on the one page, and some books have complete chapters that are less than one page, so there could be part or all of three (or more) chapters on one page.

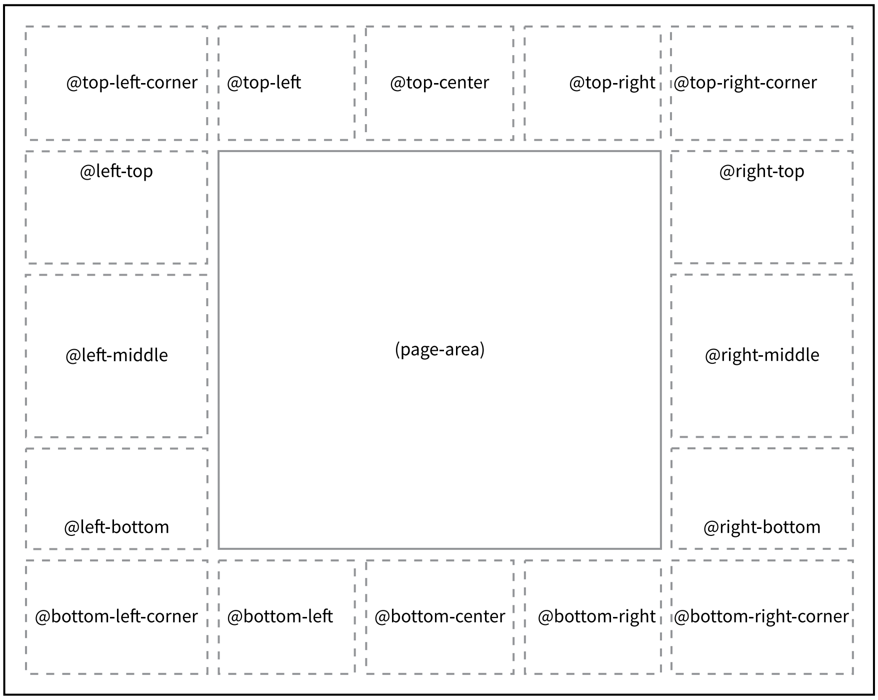
An XSL formatter considers `<fo:marker>` related to areas on the current page and, possibly, on preceding pages. It chooses a preferred `<fo:marker>` based on the 'retrieve-position' and 'retrieve-boundary' property values of an `<fo:retrieve-marker>`:

- 'retrieve-position' specifies a preference based on the areas generated by the parent FO of an `<fo:marker>` compared to the areas generated by other identically named `<fo:marker>`. 'retrieve-position' can have values 'first-starting-within-page', 'first-including-carryover', 'last-starting-within-page', and 'last-ending-within-page'.
- 'retrieve-boundary' specifies how far back to find a carried-over `<fo:marker>`. It can have values 'page', 'page-sequence', and 'document'. 'retrieve-boundary' can be useful to stop a marker 'leaking' into pages where it is not wanted: for example, an epilogue that does not need a running title in the header could use the same page masters as chapters if:
  - The epilogue is in a separate `<fo:page-sequence>`
  - The epilogue does not contain an `<fo:marker>` for the running title in the header
  - The `<fo:retrieve-marker>` for the running title in the header omits 'retrieve-boundary' to use its initial value of 'page-sequence'.

Page numbers in the header are generated using `<fo:page-number>`. The total number of pages shown in page numbers such as "Page 1 of 5" can be generated by using `<fo:page-number-citation-last>` with a 'ref-id' value that refers to the ID of the `<fo:page-sequence>` or other FO that encompasses all of the FOs that generate those pages.

CSS

In CSS, every page has 16 page-margin boxes that are always available to receive content. The names of the page-margin boxes are fixed.



Location of each page-margin box

A page-margin box is generated only if it has content; that is, if the value of its ‘content’ property is not ‘none’. ‘none’ is the computed default, so page-margin boxes ordinarily do not generate any boxes. This differs from XSL-FO, where the areas from `<fo:region-before>`, etc. can generate an area that has, for example, borders and a background color even when it has no content generated from an `<fo:static-content>` (provided that the ‘extent’ property of the region is not ‘0pt’).

Content can be either:

- Inline-level content that is any combination of literal strings, values of named strings, counter values, leaders, cross-reference text, or images
- A running element that has been removed from the flow using ‘position: running(…);’ and is retrieved using ‘content: element(…);’.

The ‘string-set’ property sets one or more named strings. For example, to set the ‘chapter’ named string to the text value of the current element:

```
H1 { string-set: chapter content(); }
```

The string name is followed by a list of string values that are concatenated to form the value of the named string. Setting a named string overwrites any previous value set earlier in the document. An element with ‘display: none;’ can still set named strings.

‘string(chapter)’ in the ‘content’ property of a page-margin box retrieves the value of the string named ‘chapter’. Multiple elements on one page can set the same named string, so ‘string()’ has an optional second argument to select one of the possible strings. The defined values are ‘first’ (default), ‘start’, ‘last’, and ‘first-except’.

Using `position: running(name);` is similar to using ‘string-set’, except that `position: running;` takes the current element out of the flow—similar to specifying `display: none;` . `content: element(name);` inserts the named running element into the page-margin box. ‘element()’ also has an optional second argument to select one of the possible running elements with the same name: ‘first’ (default), ‘start’, ‘last’, and ‘first-except’.

A running element inherits from its original position in the document. This is the opposite of `<fo:marker>`, where the marker contents inherit from the position of the `<fo:retrieve-marker>` in its `<fo:static-content>`.

‘counter(page)’ in the ‘content’ value of a page-margin box generates the current page number.

7 MULTIPLE COLUMNS

CSS offers more flexibility than XSL in which elements or FOs may have multiple columns, how multiple columns are specified, and which elements or FOs can span all of the columns in the area. For both XSL-FO and CSS, AH Formatter supports the same range of properties for specifying the column gap and the formatting of the column rule, if present.

XSL 1.1 allows multiple columns only in `<fo:region-body>`. AH Formatter extends this by also allowing multiple columns in `<fo:block-container>`. The ‘column-count’ property specifies the fixed number of columns in the `<fo:region-body>` or `<fo:block-container>`.

CSS allows any element that is a CSS ‘block container’ to possibly have multiple columns if either ‘column-width’ or ‘column-count’ is not ‘auto’. When ‘column-width’ is a length, it specifies the optimal column width. The actual columns can be wider, to fill the available space, or can be narrower, when the available space is less than the specified column width. When ‘column-count’ is an integer and ‘column-width’ is ‘auto’, ‘column-count’ specifies the number of columns, as with ‘column-count’ in XSL. However, when ‘column-count’ is an integer and ‘column-width’ is a length, ‘column-count’ specifies the maximum number of columns.

A block can either fit within one column or span all of the columns in the area in both XSL and CSS. In XSL, ‘span="all"’ on an `<fo:block>` or `<fo:block-container>` that is a child of an `<fo:flow>` (or, with AH Formatter, of an `<fo:block-container>`) will span all the columns of a multi-column region. In CSS, an in-flow block-level element with `column-span: all;` spans across all columns of the nearest ancestor multi-column area in the same block formatting context.

AH Formatter allows floats to span less than the full width of the area.

8 KEEPS & BREAKS

XSL-FO and CSS both have properties for keeping two adjacent areas together on the same page or column or for forcing a break between the areas. They also have a property or properties for keeping an area together on the same page or column.

XSL 1.1 defines 'break-before' and 'break-after' properties for specifying whether and how there should be a break before or after the areas for the FO, respectively. It also defines 'keep-with-previous' and 'keep-with-next' for specifying whether the first or last area for the FO should be kept with the preceding or following area, respectively, as well as 'keep-together' for specifying whether and how the areas for the FO should be kept together.

The value of the keep properties in XSL is a distinct 'keep' datatype that has three components: 'within-line', 'within-column', and 'within-page'. The value of each component is 'auto', 'always', or an integer. 'auto' and 'always' are self-explanatory. An integer value specifies the relative strength of the keep component. An XSL formatter should resolve the relative keep values that apply to an area so that conflicting specifications resolve, where possible, in favor of the stronger keep value. The components can be specified individually by appending the component name to the property name: for example, `keep-together.within-line="always"`. Specifying just the property name—for example, `keep-together="always"`—applies the same value to all three components. When both the property and one or more components are specified for an FO, the component values are used, and the property value is used for any components that were not explicitly specified. XSL-style 'keep-together' is available in CSS as the '-ah-keep-together' extension property.

CSS defines 'break-before' and 'break-after' properties that combine some of the features of the XSL 'break-before' and 'keep-with-previous' properties or 'break-after' and 'keep-with-next' properties. CSS also defines a 'break-inside' property that is similar to the XSL 'keep-together' property. CSS does not support integer values to indicate relative strengths

Both XSL and CSS 2 define 'page-break-before', 'page-break-after', and 'page-break-inside' properties. In XSL, these are shorthands for the other keep and break properties, and they were added to XSL only for compatibility with CSS 2. In CSS, these properties are now redefined as shorthands for some values of the other break properties.

The following tables show the values of the XSL and CSS break properties:

‘break-before’ and ‘break-after’

XSL	CSS	Description
‘auto’	‘auto’	Neither force nor forbid a break.
	‘avoid’	Avoid a break. Equivalent to `keep-with-previous="always"` or `keep-with-next="always"` in XSL.
‘column’	‘column’	Force a column break.
	‘avoid-column’	Avoid a column break. Equivalent to `keep-with-previous.within-column="always"` or `keep-with-next.within-column="always"` in XSL.
‘page’	‘page’	Force a page break.
	‘avoid-page’	Avoid a page break. Equivalent to `keep-with-previous.within-page="always"` or `keep-with-next.within-page="always"` in XSL.
‘even-page’		Force one or two page breaks so that the next page is an even page-area.
‘odd-page’		Force one or two page breaks so that the next page is an odd page-area.
	‘left’	Force one or two page breaks so that the next page is a left page.
	‘right’	Force one or two page breaks so that the next page is a right page.
	‘recto’	Force one or two page breaks so that the next page is a recto page in a two-page spread. Depending on the page progression, this could be a right-hand page or a left-hand page.
	‘verso’	Force one or two page breaks so that the next page is a verso page in a two-page spread. Depending on the

XSL	CSS	Description
		page progression, this could be a left-hand page or a right-hand page.
	'region'	Force a region break. Not implemented by AH Formatter.
	'avoid-region'	Avoid a region break. Not implemented by AH Formatter.

Note that in a left-to-right page progression, the first, odd-numbered page is a right-hand page, which is a recto page, and that in a right-to-left page progression, the first, odd-numbered page is a left-hand page, which is a also recto page. CSS explicitly allows, for example, '`html { break-before: always; }`' to force the first page of a document with left-to-right page progression to print on a left-hand page. There is no explicit support for this in XSL, but AH Formatter can generate documents that start on a left-hand page when, for example, '`<axf:document-info name="pagelayout" value="TwoPageLeft"/>`' is specified.

The 'region' and 'avoid-region' values are currently at risk of being dropped from the current Candidate Recommendation version of the specification because of an absence of interoperable implementations.

XSL 'keep-together' and CSS 'break-inside'

'keep-together'	'break-inside'	Description
'auto'	'auto'	No conditions or constraints.
'always'	'avoid'	Avoid breaks.
	'avoid-page'	Avoid a page break. Equivalent to `keep-together.within-page="always"` in XSL.
	'avoid-column'	Avoid a column break. Equivalent to `keep-together.within-column="always"` in XSL.
	'avoid-region'	Avoid a region break. No equivalent in XSL. Not implemented by AH Formatter.

AH Formatter provides 'axf:keep-together-within-dimension' and '-ah-keep-together-within-dimension' properties for specifying the maximum block height for which the keep on the block applies. When the block is higher than the limit, the block is broken normally as if no keep was specified. AH Formatter similarly provides 'axf:keep-together-within-inline-dimension' and '-ah-keep-together-within-inline-dimension’ properties for specifying the maximum width for which a ‘keep-together.within-line’ condition applies.

Both XSL and CSS define ‘widows’ and ‘orphans’ properties that determine how many lines must be kept together before or after a break. In XSL and CSS, an ‘orphan’ is a line that is left behind when part of a block of text is carried over after a break, and a ‘widow’ is a line carries on after a break. ‘orphans’ specifies the minimum number of orphan lines before a break, and ‘widows’ specifies the minimum number of widow lines after a break. If formatting a block would either leave too few lines or carry over too few lines, then the formatter will force a break so that the conditions are satisfied. A block that contains fewer lines than the sum of the ‘orphans’ and ‘widows’ values cannot be broken because doing so would not satisfy one or both of the constraints.

AH Formatter implements extensions for controlling orphan and widows rows of a table as well as for the minimum width of the last line of a paragraph.

9 PARAGRAPH SETTING

XSL-FO and CSS define both ‘text-align’ and ‘text-align-last’ properties for specifying the alignment of the text in a block and an overriding alignment for the text in the last line of a block, respectively. AH Formatter provides an extension property for specifying the alignment of the first line of a block.

XSL-FO and CSS define the ‘text-indent’ property for specifying the indentation of the first line of text in a block. AH Formatter provides an extension property for the text indent to use at the top of a page or column.

Separate styling of the first line of a block is specified using `<fo:initial-property-set>` in XSL-FO or the `first-line` pseudo-element in CSS. Drop capitals, raised capitals, and other styles for the initial letter in a block is implemented as the ‘axf:initial-letters’ extension property for `<fo:block>` in XSL-FO and the ‘-ah-initial-letters’ extension property for the `::first-letter` pseudo-element in CSS.

The height of the lines in a block is specified using the ‘line-height’ property in both XSL-FO and CSS. XSL defines ‘line-height-shift-adjustment’ for controlling whether the line height should change when the line includes content with a baseline-shift, such as superscripts or subscripts. This is available in CSS as the ‘-ah-baseline-shift-adjustment’ extension property. XSL-FO also defines ‘line-stacking-strategy’ for specifying the strategy for positioning adjacent lines relative to each other. This is available in CSS as the ‘-ah-line-stacking-strategy’ extension property.

AH Formatter provides extensions for aligning lines in multiple blocks to a common baseline grid. Blocks can establish their own baseline grid or align to the current grid, the root grid for the entire document, or no grid at all.

Leaders

XSL-FO and CSS both support leaders: XSL-FO using `<fo:leader>`, and CSS using ‘leader()’ in the value of the ‘content’ property. Leaders in XSL-FO can be unaligned or aligned to the reference area or page. AH Formatter does not support alignment to the page, but leaders can be aligned to the start, center, or end of their area. Leaders in CSS are always aligned to the end edge of their containing block. Leaders in

XSL-FO have a specified length (which can have minimum, optimum, and maximum components), whereas leaders in CSS always expand to fill the available space on the line. AH Formatter provides the ‘axf:leader-expansion’ extension property for XSL-FO leaders to expand to fill the line and the ‘-ah-leader-length’ extension property for CSS leaders to have a specified length.

Overflow

An `<fo:block-container>`, `<fo:inline-container>` or a CSS block can have a fixed height and/or width. It is possible for the content of the FO or CSS block to overflow the specified dimensions. On paged media, the content can overflow the available height or width, regardless of whether the area has a fixed size. XSL-FO and CSS both define the ‘overflow’ property that specifies whether and how the content is clipped when it overflows the area. XSL-FO extends the CSS ‘overflow’ by adding two more keywords.

AH Formatter extends the XSL-FO ‘overflow’ with more ways to recover from an overflow. The extended ‘overflow’ property is available with both XSL-FO and CSS. AH Formatter can either replace the text of the area or condense the existing text by modifying values of one or more properties until the text fits in the area. Properties that can be adjusted are ‘font-size’, ‘font-stretch’, ‘line-height’, and ‘letter-spacing’.

Hyphenation

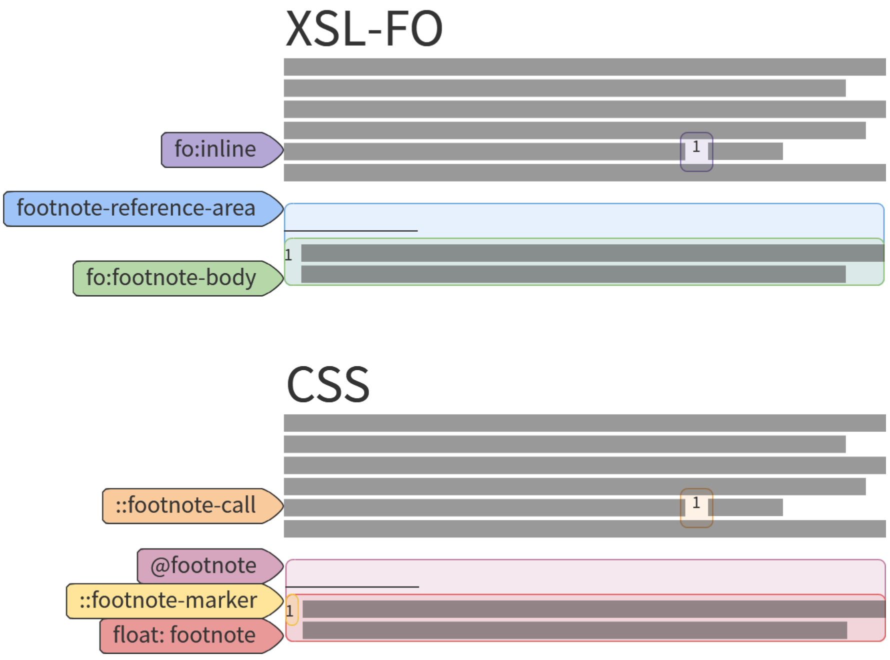
XSL-FO and CSS both support hyphenation: XSL-FO using ‘hyphenate’ and related properties, and CSS using ‘hyphens’ and related properties. AH Formatter provides some of the XSL-FO properties that do not have CSS equivalents as extension properties in CSS. AH Formatter provides additional extension properties for hyphenation in both XSL-FO and CSS.

Neither XSL-FO nor CSS define how a formatter determines where to hyphenate words. AH Formatter implements a hyphenation algorithm for multiple languages, and it can also use $\TeX$ hyphenation dictionaries. Hyphenation exceptions can also be specified in an external hyphenation information file and, for XSL-FO, `<axf:hyphenation-info>` elements can be included in `<fo:declarations>` in the XSL-FO document. The AH Formatter Option Setting File can also include a list of unbreakable words that, obviously, will not be hyphenated.

FOOTNOTES & SIDENOTES

XSL-FO and CSS both support footnotes. AH Formatter also supports sidenotes—notes that appear next to their relavent text instead of being collected in one place on the page or column—even though sidenotes are not defined in XSL 1.1 and have been removed from the current CSS Working Draft.

In both XSL-FO and CSS, the markup for sidenotes can be seen as a variation of the markup for footnotes.



XSL-FO

An `<fo:footnote>` contains both an `<fo:inline>` for the marker that is generated at the point where the `<fo:footnote>` occurs and an `<fo:footnote-body>`

that contains the actual footnote. The `<fo:footnote-body>` includes whatever marker should appear with the footnote: there is no specific FO for this, although the footnote body is often formatted as an `<fo:list-block>` so that a marker can be positioned correctly.

The content of the `<fo:footnote-body>` is ordinarily formatted in the footnote-reference-area of the bottom of the `<fo:region-body>`. The footnote-reference-area is implicitly present in every `<fo:region-body>` but it is generated only if the page contains one or more footnote areas. The footnote-reference-area cannot be defined as part of defining an `<fo:region-body>` in the `<fo:simple-page-master>`. If an footnote-reference-area is generated, it contains any areas generated by an `<fo:static-content flow-name="xsl-footnote-separator">` in the current page sequence followed by the areas generated from the `<fo:footnote-body>` from footnotes on the current page (as well as areas from any footnotes carried over from previous pages). The fo:static-content is used to, for example, generate a horizontal rule (often using an `<fo:leader>` with a fixed width) or force some white-space between the body text and the footnotes.

While XSL 1.1 defines footnotes as appearing only in the footnote-reference-area that spans the bottom of an `<fo:region-body>`, AH Formatter can generate footnotes that are page-wide or column-wide or are sidenotes at the ends of lines of body text. Page-wide footnotes can be placed on the current page or only on odd-numbered pages or only on even-numbered pages. Column-wide footnotes can be placed in the current column or the first, last, inside, or outside column on the page. Sidenotes can be placed in the `<fo:region-start>` or `<fo:region-end>` at the start, end, inside, or outside of the line containing the sidenote marker.

AH Formatter can also suppress duplicates of footnotes that would appear more than once in the same page or column.

XSL 1.1 does not define a way for footnote numbers to restart on each page. The text of the footnote marker in the `<fo:inline>` and any corresponding mark in the `<fo:footnote-body>` are expected to be generated by the XSLT that generated the XSL-FO. AH Formatter can format a provided numeric marker using any supported number format or counter style. Alternatively, AH Formatter can generate the footnote number to use and format that. If desired, the generated footnote numbers can be reset at page, odd-page, even-page, or column breaks. To reset footnote numbers, for example, for each chapter, `axf:footnote-number-initial="1"` can be specified on each `<fo:page-sequence>`.

CSS

Specifying `float: footnote;` for an element removes the element from the flow and causes it to be formatted as a footnote. Similarly, specifying `float: sidenote;` causes an element to be formatted as a sidenote. A `::footnote-call` (or `::sidenote-call`) pseudo-element is generated in place of the element. A `::footnote-marker` pseudo-element is placed at the beginning of the footnote element, and the combination is formatted in the footnote area. A `::sidenote-marker` is similarly created for a sidenote.

CSS, like XSL 1.1, defines that footnotes appear only at the bottom of a page. Sidenotes (when they were still in the GCPM Working Draft) were shown as floating to the margins of the page. The footnote area is defined by an '@footnote' rule in the applicable '@page' rule (and the sidenote area by an '@sidenote' rule). In CSS, the @footnote rule is expected (or required) to contain `float: bottom;`, which corresponds to or indicates the position of the footnotes. AH Formatter allows the 'float' property to be any supported value.

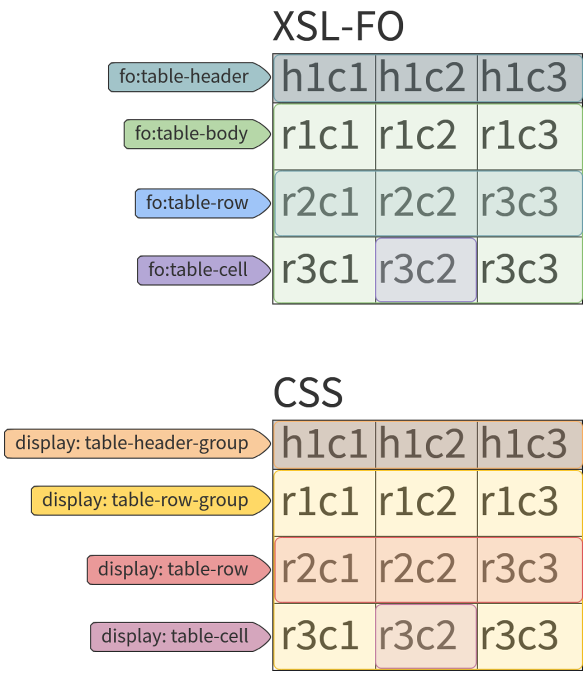
The default stylesheet styles the `::footnote-call` pseudo-element as a superscript digit and the `::footnote-marker` as an inline list-item marker. The default content of both pseudo-elements is the value of the 'footnote' counter. The current GCPM Working Draft states that the footnote counter may be reset for each new page. The AH Formatter default stylesheet does not reset the counter, so footnote numbers default to incrementing all through the document. Footnote numbers can be formatted using any supported counter style.

'-ah-suppress-duplicate-footnote' suppresses duplicate footnotes in the same page or column in CSS formatting.

11 TABLES

XSL-FO tables and CSS tables are very similar. XSL-FO tables are based on CSS2 tables, and the CSS handling of tables has not been significantly updated since CSS2. One difference is that CSS can infer “missing” elements around elements that are displayed as parts of a table and generate anonymous objects so that the CSS is working with the full set of boxes for a formatted table. In contrast, XSL allows almost no FOs to be omitted from the XSL-FO document. The exception is that `<fo:table-row>` can be omitted when the `<fo:table-cell>` specify ‘starts-row’ and ‘ends-row’ to indicate row breaks.

The following image shows some of the FOs and ‘display’ values for tables:



XSL-FO markup and CSS ‘display’ values for tables

Table header and footer

Tables can have header and/or footer rows in both XSL-FO and CSS. The header and footer rows are repeated when the table breaks across a page or column. XSL 1.1 defines ‘table-omit-header-at-break’ and ‘table-omit-footer-at-break’ properties that specify whether the table header or footer, respectively, should be omitted when the table breaks. AH Formatter adds the ‘column’ keyword for specifying that the header or footer should be omitted at column breaks but not page breaks. CSS does not define corresponding properties, but AH Formatter provides the ‘-ah-table-omit-header-at-break’ and ‘-ah-table-omit-footer-at-break’ extension properties for use with CSS.

An `<fo:table>` can contain `<fo:retrieve-table-marker>` as a descendent of `<fo:table-header>` or `<fo:table-footer>` or as a child of `<fo:table>` at any position where `<fo:table-header>` or `<fo:table-footer>` is allowed. `<fo:retrieve-table-marker>` behaves like `<fo:retrieve-marker>` but its scope is limited to the `<fo:table>`. It has properties ‘retrieve-class-name’, ‘retrieve-position-within-table’, and ‘retrieve-boundary-within-table’. `<fo:retrieve-table-marker>` can retrieve multiple rows (if that is the content of the selected `<fo:marker>`), and AH Formatter provides the ‘axf:retrieve-table-rows’ extension property to set the maximum number of retrieved rows.

A table header or footer can contain footnotes. AH Formatter provides the ‘axf:repeat-footnote-in-table-header’ and ‘axf:repeat-footnote-in-table-footer’ properties that specify whether footnotes should also be repeated when a table header or footer, respectively, is repeated when the table breaks. The ability to suppress repeated footnotes is not available with CSS.

Breaking tables

AH Formatter also provides the ‘axf:table-row-orphans’ and ‘axf:table-row-widows’ extension properties—and ‘-ah-table-row-orphans’ and ‘-ah-table-row-widows’ extension CSS properties—that specify the minimum number of rows that must remain at the bottom or top, respectively, of the page (or column) when a table breaks.

For a table cell that breaks across a page or column, AH Formatter provides extensions that either repeat the table cell contents after the break or generate provided content after the break. ‘axf:repeat-cell-content-at-break’ and ‘-ah-repeat-cell-content-at-break’ specify whether to generate another copy of the table cell’s content after a break. `<axf:table-cell-repeated-marker>`, if present in the XSL-FO, provides alternative content that is inserted after a break instead of the second copy of the table cell’s content. In CSS, `position: running(table-cell-repeated-marker);` on an element in a table cell removes the element from the flow. If its parent table cell breaks, the formatted element is inserted after the break instead of the second copy of the table cell's contents.

Table caption

Tables can have a caption in both XSL-FO and CSS: in XSL-FO, inside an `<fo:table-and-caption>` as an `<fo:table-caption>` that precedes the `<fo:table>`; in CSS, as an element with `display: table-caption;` that is the first child of the element for the table.

Column widths

XSL-FO and CSS both define the 'table-layout' property for specifying whether table column widths should be automatically determined from the table's content or are specified in the table's markup. AH XSL Formatter is the only XSL formatter that supports automatic table layout, and AH CSS Formatter and the browsers all support automatic table layout.

Table column widths in XSL-FO are specified by the 'column-width' property on `<fo:table-column>`, and in CSS, by the 'width' property of an element with `display: table-column;` (and in HTML, the `width` attribute on `<col>`).

Both 'column-width' and 'width' can be a length or percentage, but while 'column-width' in XSL specifies a fixed column width, 'width' in CSS specifies a minimum column width.

XSL, but not CSS, also defines the `proportional-column-width()` function for allocating column widths with fixed table layout: after the fixed components of the column widths are determined, the remainder of the available width is allocated in proportion to the `proportional-column-width()` values. AH Formatter makes `proportional-column-width()` available in CSS.

Properties from table columns

Although tables are a grid of rows and columns, the markup for both HTML and XSL-FO tables is 'row primary', where table cells are descendants of rows and not of columns. This obviously means that inherited properties cannot ordinarily be inherited from a column definition.

Both XSL and CSS define that border and background properties can apply to cells in the current table column. Border properties from the column definition can apply when 'border-collapse' is 'collapse' (or, in XSL-FO, also 'collapse-with-precedence').

CSS defines an algorithm for resolving the conflict when multiple border property specifications apply to an edge of a table cell; for example, conflicting border property specifications on the table cell, the table row, and the table column. In essence, the algorithm chooses the most 'eye catching' border style unless any of the elements specifies 'hidden'.

XSL adopted the same algorithm for use when ‘border-collapse’ is ‘separate’. When ‘border-collapse’ is ‘collapse-with-precedence’, conflicts are resolved based on the integer precedence value assigned to each FO that can supply the border of a table cell. The default precedence values give priority to a table’s ‘outer’ FOs as well as prioritizing column borders over row borders: `<fo:table>`, 6; `<fo:table-cell>`, 5; `<fo:table-column>`, 4; `<fo:table-row>`, 3; `<fo:table-body>`, 2; `<fo:table-header>`, 1; `<fo:table-footer>`, 0. However, each table FO can specify a different precedence value for each of the four edges, and can specify ‘force’ to override all other precedence values.

XSL defines the `from-table-column()` function that can be used on an `<fo:table-cell>` or its descendants to retrieve the inherited value of a property as specified on the corresponding `<fo:table-column>`.

Alignment in table cells

The first line of text in multiple table cells in a table row can have the same baseline in both XSL-FO and CSS, even when text in different cells have different font sizes. For XSL-FO, use `relative-align="baseline"`, and for CSS, use `vertical-align: baseline;`. CSS defines this behavior for ‘vertical-align’ only for table cells. Other than on table cells, `vertical-align: baseline;` in CSS aligns the alphabetic baseline of the element with the alphabetic baseline of its parent element. XSL 1.1 also defines ‘vertical-align’, but only as a shorthand for multiple XSL-FO properties and only to match the non-table cell usage in CSS.

Horizontal alignment of text in a table cell is the same in both XSL-FO and CSS when using AH Formatter. XSL 1.1 defined more values for ‘text-align’ than are defined in CSS2. AH Formatter also implements the XSL 1.1 values for CSS formatting. The CSS Text Module Level 3 defines some different keywords that achieve the same effects as the XSL 1.1 ‘text-align’ and related properties, except for one difference.

XSL 1.1 (and, consequently, CSS formatting in AH Formatter) allows the value of ‘text-align’ on descendents of a table cell to be a string. Each of the blocks in one table column that have a string value for ‘text-align’ will be aligned horizontally so that all of their strings are in the same vertical line. A typical usage is to align all of the numbers in a table column on their decimal point (.) by specifying `text-align='"."'` or `text-align: ".";`. AH Formatter also provides the ‘axf:text-align-string’ and ‘-ah-text-align-string’ extension properties for specifying the alignment of the aligned strings within their table cells.

Rotated tables and table cells

Entire tables and the content of individual table cells can be rotated both in XSL-FO and in CSS with an AH Formatter extension. In XSL-FO, an `<fo:table>` can be placed in an `<fo:block-container>` that specifies the ‘reference-orientation’ property, and an `<fo:table-cell>` can contain an `<fo:block-container>` that specifies the 'reference-orientation' property. In CSS, the '-ah-reference-orientation' extension property can be specified on both tables and table cells.

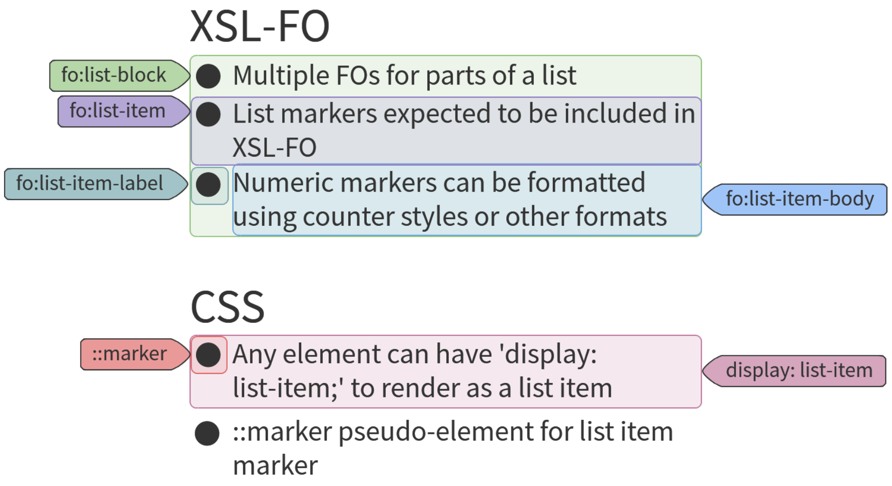
Accessibility

The row-and-column grid of a table can be difficult to convey using a screen reader or other assistive technology. For XSL-FO, AH Formatter provides 'axf:table-summary', 'axf:headers', and 'axf:scope' extension properties, and for CSS, the corresponding '-ah-table-summary', '-ah-headers', and '-ah-scope' properties. These properties bridge between attributes in HTML tables (or other XML markup) and PDF attributes in Tagged PDF and PDF/UA. For XML source that is transformed into XSL-FO, the properties can derive from the source XML or can be added by the XSLT stylesheet.

'axf:table-summary' and '-ah-table-summary' correspond to the `summary` attribute of `table` in HTML 4.01. (`summary` was deprecated in HTML5 in favor of `caption`, even though `caption` functions like a heading for the table and `summary` was meant to convey information about the organization of data in the table and to help users navigate the table. [WAI]) 'axf:table-summary' and '-ah-table-summary' also correspond to the `Summary` attribute for tables in Tagged PDF and PDF/UA that was introduced in PDF 1.7.

'axf:headers' and '-ah-headers' indicate the table head cells that apply to the current table cell. 'axf:scope' and '-ah-scope' work the other way and indicate whether the current table head cell applies to following cells in the same column, following cells in the same row, or all cells 'down' and 'to the right' of the current cell.

12 LISTS


XSL-FO
fo:list-block
Multiple FOs for parts of a list
fo:list-item
List markers expected to be included in XSL-FO
fo:list-item-label
Numeric markers can be formatted using counter styles or other formats
fo:list-item-body
CSS
::marker
Any element can have 'display: list-item;' to render as a list item
display: list-item
::marker pseudo-element for list item marker

XSL-FO markup and CSS ‘display’ values for lists

XSL-FO

XSL defines specialized FOs for representing lists. The FO structure for a two-item list, where indentation represents containment, is:

```
fo:list-block
  fo:list-item
    fo:list-item-label
      (%block;)+
    fo:list-item-body
      (%block;)+
  fo:list-item
    fo:list-item-label
      (%block;)+
    fo:list-item-body
      (%block;)+
```

`(block;)+` means one or more block-level FOs. This includes `<fo:block>` and `<fo:block-container>`, but it also includes, for example, `<fo:table>` and `<fo:list-block>`. It is possible to put structured content into an `<fo:list-item-label>` as well as into an `<fo:list-item-body>`.

XSL-FO defines the 'relative-align' property for aligning the baselines of the first lines of an `<fo:list-item-label>` and `<fo:list-item-body>`.

There is no automatic numbering of list items in XSL-FO. The list item label is expected to be generated by the XSLT transformation. The number could be copied from the source document or it could be generated based either on the structure of the source or on the logic in the stylesheet. XSLT defines a limited number of numbering schemes that can be used to format the list item label. AH Formatter provides the 'axf:number-transform' extension property, which supports more numbering schemes than are provided by XSLT.

'axf:number-transform' causes the number to which it applies to be formatted as ideographic numerals, as one of the CSS-style counter styles (including symbolic styles), or as one of the styles for the 'format' property. XSL 1.1 defines 'format' as matching the XSLT 1.0 [XSLT10] 'format' attribute, but AH Formatter extends this to support many more numeric, alphabetic, and symbolic sequences.

XSL-FO defines some properties and functions to make it easier to use the same margins for all of the items in the one list. The 'provisional-label-separation' property of `<fo:list-block>` is part of the formula for 'label-end()', which is used to determine the end-indent of an `<fo:list-item-label>`. Similarly, 'provisional-distance-between-starts' is part of the formula for 'body-start()', which is used to determine the start-indent of an `<fo:list-item-body>`. The XSL 1.1 Recommendation does not state why 'provisional' is included in the properties' names, but the properties' values are only one part of each formula. Also, the value of a property where the functions are used can be an expression that adds or subtracts other lengths or even multiplies the function value by some number.

CSS

An element with `display: list-item;` generates a block box plus a marker box. [CSS3-Display] The same specification also says that the `list-item` keyword generates a `::marker` pseudo-element with the content specified by the 'list-style' properties: 'list-style-type', 'list-style-position', and 'list-style-image', with 'list-style' as the shorthand property for all three.

The content of the marker box can be a combination of text and images. More specifically, it is the first of:

- The value of the 'content' property, if that is not `normal`

  This can be the value of an arbitrary counter that is formatted using any counter style.

- An image generated from ‘list-style-image’ (followed by a space character)
- String generated from ‘list-style-type’

  This can be a literal string or a counter style used for formatting the value of the built-in ‘list-item’ counter.

Otherwise, there is no marker box.

13 CHARACTER SETTING

The fundamental and most-used properties for setting characters—properties such as ‘line-height’, ‘font-family’, ‘font-style’, ‘font-variant’, etc.—were defined in CSS1 and CSS2 and, therefore, are also defined identically in XSL-FO. Character-level properties from CSS3 modules that AH Formatter implements are also implemented in XSL-FO as extensions. AH Formatter also implements its own extension properties for both XSL-FO and CSS. For example, CSS3-Text defines ‘text-underline-position’ for specifying the position of underlines. AH Formatter implements ‘text-underline-position’ for CSS and implements ‘axf:text-underline-position’ for XSL-FO. In addition, it implements extension properties for specifying the color, line style, and line width of underlines in both XSL-FO and CSS.

Per-document fonts

CSS defines the ‘@font-face’ rule that allows the stylesheet to automatically fetch and activate fonts instead of using only the fonts known to the formatter. XSL 1.1 does not define an equivalent feature, but AH Formatter implements the `<axf:font-face>` extension element for use with XSL-FO.

14 JAPANESE TEXT COMPOSITION

Requirements for Japanese Text Layout 日本語組版処理の要件（日本語版） [JLREQ] is a comprehensive desciption of requirements for general Japanese layout. XSL-FO and CSS both define broad support for a range of writing modes and scripts, but neither sets out to address the full range of Japanese text composition described in JLReq. CSS defines more support for Japanese than does XSL-FO, including: Japanese-specific counter styles; ruby; and horizontal-in-vertical tate-chu-yoko text.

AH Formatter provides a large set of extensions for both XSL-FO and CSS to support Japanese text composition. The extensions cover:

- Handling Japanese punctuation correctly
- Fine control of ruby and emphasis dots position and styling
- Mixing Japanese and Western text in the same composition
- Japanese-style crop marks

15 CROSS-REFERENCES

XSL-FO generates links from `<fo:basic-link>`, whereas CSS specifications assume that the browser implements the `<a>` element. Many cross-references include the title, page number, or section number, etc., of the subject of the cross-reference. XSL-FO assumes that all text other than page numbers is included in the FO tree, whereas CSS provides functions for retrieving text content or counter values from a target element.

Links

XSL 1.1 defines `<fo:basic-link>`, which has separate 'internal-destination' and 'external-destination' properties for referring to IDs within the FO tree and external resources, respectively. One of the properties, but not both, should be specified on an `<fo:basic-link>`.

`<fo:root>`, `<fo:page-sequence>`, `<fo:page-sequence-wrapper>`, and nearly every FO that can be used within an `<fo:page-sequence>` can have an 'id' property. `<fo:change-bar-begin>` and `<fo:change-bar-end>`, for example, do not have an 'id' property. AH Formatter supports 'xml:id' as an alternative to 'id' on FOs that can have 'id'.

CSS relies on a browser's implementation of the 'href' attribute of the `<a>` element to provide linking behavior. The HTML5 specification for a UA stylesheet [WHATWG] does not specify anything for the `href` attribute. The default `html.css` file for AH Formatter defines `href` as setting the '-ah-link' extension property:

```
a[href]   { -ah-link: attr(href url); }
```

'-ah-link' can similarly be used with other, non-HTML XML vocabularies to generate hyperlinks from other elements or attributes.

Generated text

XSL-FO assumes that references to the number or title of chapters, sections, tables, and figures, etc., will be generated by the XSLT that generated the XSL-FO. The full source document is available to the XSLT processor, and XSLT includes

`<xsl:number>`, which can generate multi-level numbers in a variety of formats. Consequently, XSL-FO does not include a mechanism to copy text from elsewhere in the FO tree.

The only text that XSL-FO can generate is for values that are not known before the document is formatted: page numbers and page number references (and the scaling factor of formatted graphics). `<fo:page-number>` generates the page number of the current page. `<fo:page-number-citation>` generates the page number of the page containing the first normal area from the referenced FO, and `<fo:page-number-citation-last>` generates the page number of the last page of the referenced FO.

AH Formatter extensions for page numbers in XSL-FO include:

- 'axf:reverse-page-number': page numbers and page number references are counted from the last page of the `<fo:page-sequence>`
- 'axf:physical-page-number': generates the physical page number, which will be different from the logical page number if, for example, page numbers restart after the front matter or for each chapter
- 'axf:origin-id': the generated page number is the difference between the page number of the FO referred to by 'axf:origin-id' and the page number that would ordinarily be generated by the current FO

CSS can generate text based on the content or position of another element in the document. The 'target-text()' function retrieves the text value of a target element or, alternatively, the text value of its `::before` or `::after` pseudo-element or its first letter. 'target-counter()' (and also 'target-counters()') retrieves the value of a named counter as seen by a target element. To obtain the page number of a target element, use target-counter() and specify the page counter:

```
content: target-counter(attr(href url), page);
```

The AH Formatter extensions for physical and reverse page numbers are available by adding a keyword. For example, to generate the physical page number of a target element:

```
content: target-counter(attr(href url), page physical);
```

CSS does not have an equivalent to `<fo:page-number-citation-last>`.

16 IMAGE POSITIONING

CSS2 defines the ‘float’ property, with values ‘left’, ‘right’, and ‘none’ (initial value), as applying to all elements. ‘left’ and ‘right’ float the block to the left edge and right edge, respectively, of the current block’s containing block. XSL 1.1 limits ‘float’ to apply to only `<fo:float>`. It adds the ‘before’ value specifying that the content of the `<fo:float>` is generated in the before-float-reference-area at the top of a body region. It also adds ‘start’, ‘end’, ‘inside’, and ‘outside’ values for specifying the horizontal position of the `<fo:float>`.

AH Formatter extends the ‘float’ property to be a shorthand for multiple extension properties—available with both XSL-FO and CSS—that provide greater control over the placement of floats. Floats ordinarily float within the current reference area, but they can be specified to float within the page area, multi-column area, or current column. A float can be forced to be placed in the current page or column, moved to the next page or column if there is not enough space in the current page or column, kept in the current page or column while the float’s anchor is moved to the next page or column, or floated to appear on a specific page. There are additional extension properties for how text can flow around a float and for the margins between a float and its surrounding text or its neighboring float. Floats can be made to span a subset of the columns in a multi-column area.

Footnotes and sidenotes in CSS are generated by specifying `float: footnote;` and `float: sidenote;`, respectively.

MATHML & SVG GRAPHICS

XSL 1.1 does not mention MathML, and it mentions SVG as a common format for `<fo:instream-foreign-object>` content but does not require that SVG is supported. SVG can be embedded in HTML5, and CSS 3 modules are written with the assumption that SVG graphics are supported.

AH Formatter provides custom MathML 3 and SVG renderers for including MathML and SVG in PDF output. Output of entire formatted documents in SVG is possible with the SVG Output Option add-on.

MathML 3

AH Formatter natively supports MathML 3, whereas browsers and MathJax implement a subset of MathML 2. Improvements between MathML 3 and MathML 2 include:

- Presentation elements for elementary maths layouts

$$\begin{array}{rcccccc}
 & & & & 5 & & \\
3 & & 4 & 3 & \cancel{6} & . & 3 \\ \cline{2-7}
\cancel{2}) & 1 & 3 & 0 & 6 & & \\
 & 1 & 2 & & & & \\ \cline{2-3}
 & & 1 & 0 & & & \\
 & & & 9 & & & \\ \cline{3-4}
 & & & 1 & 6 & & \\
 & & & 1 & 5 & & \\ \cline{4-5}
 & & & & 1 & . & 0 \\
 & & & & & & 9 \\ \cline{5-7}
 & & & & & & 1
\end{array}$$

Two-dimensional long division using MathML 3

- Improved linebreaking, spacing adjustment, and directionality control
- `<mglyph/>` for displaying images of non-standard symbols
- `href` added to common attributes to allow elements to link to a URI

18 COUNTERS

XSL 1.1 is designed to generate and format numbers only for page numbers and scaling value citations, because they are the only numbers that cannot be known before the document is formatted. All other numbers are assumed to have been generated by the XSLT stage.

The ‘format’ property on `<fo:page-sequence>` specifies the format both for the page numbers on pages generated by that `<fo:page-sequence>` and also for `<fo:page-number-citation>` that refer to an area on a page generated by that `<fo:page-sequence>`. `<fo:scaling-value-citation>`, added in XSL 1.1, is used to obtain the scale-factor that is applied to a cited `<fo:external-graphic>`.

Note that XSL 1.1 does not need to generate numbers for footnotes because it does not define a way for footnote numbers to restart on each new page or new column. Restarting footnote numbers is possible, however, with an AH Formatter extension.

CSS, for possibly every element, can generate a number based on the position of the element. The format of the number is determined by its ‘counter style’. A counter style is a particular algorithm for generating a representation of a number. CSS defines many more counter styles than XSL 1.1 defines number formats. It is also possible to define and use a new counter style within a CSS stylesheet, and any counter style can extend another counter style or fall back to using another counter style for numbers that are outside the range of the current counter style.

AH Formatter implements CSS-style counter styles for both CSS and XSL-FO, in addition to supporting the XSL 1.1 ‘format’ property for XSL-FO. For XSL-FO, AH Formatter uses the counter style to format an existing number, except for generated page numbers and `<fo:scaling-value-citation>` values.

Besides the predefined counter styles, a new counter style is defined in XSL-FO using the `<axf:counter-style>` FO within `<fo:declarations>`. `<axf:counter-style>` has identical properties to an ‘@counter-style’ rule in CSS. A counter style name, or any XSL 1.1 ‘format’ string, can be specified on any block-level or inline-level FO by specifying ‘axf:number-transform’. When ‘axf:number-transform’ is not ‘none’, any positive or negative integer sequence is transformed using the specified counter style or number format. Counter styles can also be used as the value of the ‘format’ property of `<fo:page-sequence>` to specify the format for page numbers of pages generated by that `<fo:page-sequence>`.

CSS counters

A CSS counter “is a special numeric tracker used, among other things, to automatically number list items in CSS. ” Counters are created, incremented, or set by specifying CSS properties. There are some predefined counters, such as the `page` counter for page numbers, that are always present and that are automatically incremented. For example, the `page` counter is incremented with each new page. The value of a predefined counter can also be modified by setting CSS properties.

Counter styles

CSS supports the following properties on an ‘@counter-style’ rule, which AH Formatter also supports on `<axf:counter-style>` in XSL-FO:

- ‘system’ : Specifies which algorithm to use to construct the counter’s representation.
- ‘negative’ : Defines how to alter the representation when the value is negative.
- ‘prefix’ : Specifies a symbol that is prepended to the marker representation.
- ‘suffix’ : Specifies a suffix that is appended to the marker representation.
- ‘range’ : Defines the ranges over which the counter style is defined.
- ‘pad’ : Specifies a symbol with which to pad counter representations that are not a minimum number of grapheme clusters.
- ‘fallback’ : Fallback counter style to be used when the current counter style cannot create a representation.
- ‘symbols’ : Symbols to be used by the marker-construction algorithm.
- ‘additive-symbols’ : Symbols to be used by an additive marker-construction algorithm.
- ‘speak-as’ : Describes how to synthesize the spoken form of a counter. (Not implemented by AH Formatter.)

The following example shows a ‘my-filled-circled-decimal’ counter style that is a based on the ‘filled-circled-decimal’ counter style from CSS Counter Styles Level 3. [CSS3-CounterStyles] As the name suggests, the counter style uses decimal numbers inside filled circles to represent decimal numbers. The generated symbol is followed by a space.

```
@counter-style my-filled-circled-decimal {
system: fixed;
         /* Using hex escape of Unicode character codes. */
symbols: '\2776' '\2777' '\2778' '\2779' '\277a' '\277b' '\277c'
         '\277d' '\277e';
/* symbols: '❶' '❷' '❸' '❹' '❺' '❻' '❼' '❽' '❾'; */
suffix: ' ';
}
```

The XSL-FO equivalent using `<axf:counter-style>` is:

```
<axf:counter-style
    name="my-filled-circled-decimal"
    symbols="'❶' '❷' '❸' '❹' '❺' '❻' '❼' '❽' '❾'"
    suffix=" " />
```

Use ‘axf:number-transform’ to represent numbers using the counter style:

```
<fo:list-item-label axf:number-transform="my-filled-circled-decimal">
 <fo:block>1</fo:block>
</fo:list-item-label>
```

‘format’ and related XSL properties

‘format’ and its related properties—‘grouping-separator’, ‘grouping-size’, ‘letter-value’, ‘country’, and ‘language’—take their definition from the corresponding XSLT 1.0 number-to-string conversion attributes. XSL is defined as both XSLT and XSL-FO, and XSLT 1.0 was the current XSLT version when XSL 1.0 became a Recommendation.

‘format’ in XSLT specifies the format of the numbers generated by `<xsl:number>`. `<xsl:number>` can generate a list of numbers (when ‘level="multiple"’), and the XSLT ‘format’ attribute contains a sequence of alphanumeric tokens for the format of the potentially multiple levels of number generated by `<xsl:number>`. These are separated by non-alphanumeric tokens that are repeated as the punctuation between the generated numbers.

‘format’ in XSL-FO is just one alphanumeric token that specifies the format for the page number or for the scaling value. `<xsl:number>` generates only integers, and page numbers are only integers and `<fo:scaling-value-citation>` generates only an integer percentage value. Any prefix or suffix on a page number is specified with `<fo:folio-prefix>` or `<fo:folio-suffix>`, respectively. Using separate FOs for the prefix and suffix allows them to be styled differently from the style for the page number, if desired. Any ‘%’ (or similar) suffix for an `<fo:scaling-value-citation>` would have to be included in the text of the XSL-FO document.

The number formats defined for ‘format’ in XSLT are aspirational rather than comprehensive. The format tokens are designed as a superset of the allowed values of the ‘type’ attribute for the `<OL>` element in HTML 4.0, but only five token formats are required:

- ‘1’ generates `1 2 ... 10 11 12 ...` (and ‘01’ generates `01 02 ... 09 10 11 12 ... 99 100 101 ...`, and so on)

    - In principle, any character that has a decimal digit value of 1 (as specified in the Unicode character property database) can be used
- ‘A’ generates `A B C ... Z AA AB AC...`
- ‘a’ generates `a b c ... z aa ab ac...`
- ‘i’ generates `i ii iii iv v vi vii viii ix x ...`
- ‘I’ generates `I II III IV V VI VII VIII IX X ...`

Any other character can be used to indicate a numbering sequence, provided that the processor supports it. AH Formatter supports an extensive range of format tokens for numbers in multiple scripts, as well as multiple styles of Roman numerals and even Chinese zodiac signs and the ‘*’, ‘†’, ‘‡’ sequence that is traditionally used for footnotes in English. However, the range is not extensible. If something else is required, you can specify one of the predefined CSS counter styles or specify the name of your own counter style as the value of the ‘format’ property. For more information, see “format” in the AH Formatter manual.

19 COLOR

XSL 1.1 supports only colors specified as RGB or as a color in an ICC (International Color Consortium) color profile using the ‘rgb-icc()’ function. A color profile defines the conversion between a standard color space and a native or device-dependent color space. An ICC color profile is a cross-platform standard for the color profile format.

In XSL 1.1, in common with CSS2:

- RGB colors are specified using `#` followed by three or six hexadecimal digits or using the ‘rgb()’ function
- 16 color names are supported: aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white, and yellow
- The ‘system-color()’ function returns the color corresponding to a name of a part of the operating system’s graphical environment

  (‘system-color()’ is seldom, if ever, used, and CSS deprecated its use in the CSS Color Module Level 3.)
- The ‘transparent’ keyword can be used in many places where a color is allowed: it specifies that the underlying color can show through

CSS Color Module Level 3 [CSS3-Color]:

- Supports colors with opacity by adding four-digit and eight-digit hexadecimal formats and the ‘rgba()’ function
- Increased the named colors to include all of the X11 colors that are supported by popular browsers with the addition of gray/grey variants from SVG 1.0
- Added ‘hsl()’ for specifying colors in terms of hue-saturation-lightness (HSL) as well ‘hsla()’ for specifying HSL colors with opacity
- Added ‘currentColor’ as the keyword for the computed value of the ‘color’ property

AH Formatter implements all of the color representations from CSS Color Module 3 as well as:

- ‘cmyk()’ from an earlier Generated Content in Paged Media [GCPM-2007] Working Draft for specifying colors as CMYK
- ‘cmyka()’ as an extension for specifying CMYK colors with opacity

- ‘k100’ color name as a shorthand for ‘cmyk(0, 0, 0, 1)’ . 100% black is often used for text because there is no risk of misregistration, unlike if the color was made from multiple tints.
- ‘rebeccapurple’ color name that is defined in CSS Color Module Level 4 drafts
- <meta name="color-profile"> specifies a color profile in HTML

AH Formatter implements ‘transparent’ text by reserving space for the text, but it does not generate any characters in the output. Use ‘rgba(0, 0, 0, 0)’ to generate invisible text that is included in the output.

More than 1,000 PANTONE® colors can be specified by name and printed as a spot color or be converted into the correct RGB or CMYK for rendering or printing when you have the AH Formatter PANTONE® Option

20 BORDERS & BACKGROUND

Similarly to the character setting properties, the fundamental and most-used properties for borders and backgrounds were defined in CSS1 and CSS2 and, therefore, are also defined identically in XSL-FO. Border and background properties from CSS3 modules that AH Formatter implements are also implemented in XSL-FO as extensions. AH Formatter also implements its own extension properties for both XSL-FO and CSS.

Rounded corners

CSS 3 provides properties for specifying rounded corners on any element that can have a border. XSL 1.1 does not define equivalent features, but rounded corners are available as an AH Formatter extension.

Multiple background images

XSL 1.1 and CSS 2 support a single background image for a box. CSS 3 allows a box to have multiple background images, which AH Formatter also supports in XSL-FO as an extension.

Background image position

CSS 2 defines the ‘background-position’ property for specifying the horizontal and vertical position of a background image, or, with CSS 3, background images. XSL 1.1 treats ‘background-position’ as a shorthand for the separate ‘background-position-horizontal’ and ‘background-position-vertical’ properties.

Background image repetition

CSS 2 defines the ‘background-repeat’ property for specifying whether and how a background image is repeated. CSS 3 defines additional keywords that provide more control over how a repeated image covers the available area.

AH Formatter implements the CSS 3 ‘background-position’ values for both XSL-FO and CSS and adds a ‘paginate’ value that is available when embedding a PDF as the background image of a page. Successive pages from the embedded PDF provide the backgrounds of successive pages in the output. The background images can be a subsequence of pages from the embedded PDF. If necessary, extra output pages are generated to match the number of embedded PDF page images.

21 PDF OUTPUT

AH Formatter supports the same range of PDF versions and the same range of PDF features, such as layers and Tagged PDF, for PDF output from both XSL-FO and CSS.

Bookmarks

XSL 1.1 added `<fo:bookmark-tree>` and related FOs to “hold a list of access points within the document such as a table of contents, a list of figures or tables, etc.” The bookmark tree is separate from the visible content of the document. If present, it is a child of `<fo:root>`, and it appears before the first `<fo:page-sequence>` (or `<fo:page-sequence-wrapper>`).

`<fo:bookmark-tree>` contains one or more `<fo:bookmark>`. An `<fo:bookmark>` contains an `<fo:bookmark-title>` and can also contain any number of subordinate `<fo:bookmark>`. An `<fo:bookmark>` can have either an ‘internal-destination’ property to refer to the ID of an FO in the current document or an `<fo:external-destination>` to refer to an external resource. The following figure shows a sample bookmark tree structure, where indentation represents containment:

```
fo:bookmark-tree
  fo:bookmark
    fo:bookmark-title
    fo:bookmark
      fo:title
      fo:bookmark
        fo:bookmark-title
        fo:bookmark
          fo:bookmark-title
    fo:bookmark
      fo:bookmark-title
```

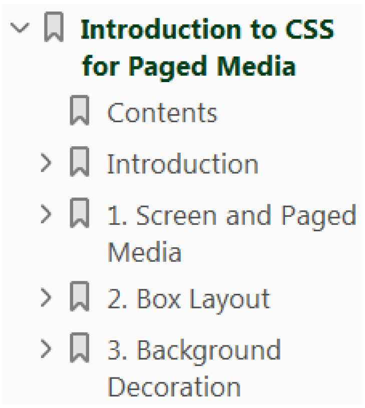
Sample bookmark tree structure

Because the bookmark tree is separate from the visible content and also because an `<fo:bookmark>` can refer to any destination, the bookmark tree does not have to represent the exact structure of the document. It is possible, for example, to refer to related documents or web pages from the bookmark tree or it can present multiple views of the document, such as listing sections and also tables in the bookmark tree.

An `<fo:bookmark>` can be specified as initially showing or hiding its subordinate `<fo:bookmark>`. An `<fo:bookmark-title>` can contain text only. Because the bookmark tree is typically generated using XSLT, the text can match the corresponding section title. In some cases, it can be necessary to transform superscript or subscript text into their corresponding plain text form. An `<fo:bookmark-title>` can have a specified color, its font weight can be normal or bold, and its font style can normal or italic.

Bookmarks in CSS follow the structure of the elements in the document that generate bookmarks. CSS defines an ‘bookmark-level’ property that applies to all elements. Its value is either ‘none’ (the default) or an integer, where ‘1’ is the highest level bookmark. The ‘bookmark-label’ property specifies the text content of the bookmark label. Its initial value is ‘content(text)’, which generates the text content of the element. The content that can be specified for ‘bookmark-label’ is the same as for the ‘string-set’ property: any combination of literal strings, counter values, the content of the current element (or its pseudo-elements), or the value of an attribute of the current element. AH Formatter also allows the ‘-ah-attr-from()’ and ‘-ah-attr-img()’ extension functions in ‘bookmark-label’.

Bookmarks in CSS automatically link from the bookmark to the element that generated the bookmark. The ‘-ah-outline-internal-destination’ extension property specifies an ID or a page number (with optional zoom factor) that is used instead of the default link. Alternatively, the ‘-ah-outline-external-destination’ extension property specifies a URL to use as the link destination.


Introduction to CSS for Paged Media
Contents
Introduction
1. Screen and Paged Media
2. Box Layout
3. Background Decoration

Screenshot of bookmarks in Adobe Acrobat

AH Formatter supports ‘-ah-outline-color’, ‘-ah-outline-font-style’, and ‘-ah-outline-font-weight’ extension properties in CSS to provide the same control over the appearance of the bookmark label as is available with XSL-FO.

A block-level element that generates a CSS bookmark label can have a ‘bookmark-state’ property. When ‘bookmark-state’ is ‘open’, the bookmark label and the bookmark labels of the nearest descendant elements that generate bookmark labels will be shown. Otherwise, the value is ‘closed’, and only the bookmark label of the current element is shown.

AH Formatter additionally supports its own extensions for generating bookmarks based on the structure of the document. In general, it is better to use the standard XSL-FO or CSS mechanism instead. The exception is when XSL-FO is formatted to generate multiple PDF volumes. When the AH Formatter extensions are used, a bookmark tree can be generated in the first volume or in each separate volume. The bookmark tree can either cover all the volumes and be the same in each volume or it can cover the current volume only. The AH Formatter extensions include a mechanism for optionally grouping bookmarks. When the bookmark tree is for only the current volume, any bookmark groups in the entire document can appear either in the bookmark tree in the last volume or in the bookmark tree of every volume.

22 INDEXES

The key features of a ‘back-of-the-book’ index are:

- Some books have multiple indexes
- An index is a sorted, and usually also grouped, set of headwords
- In paged media, headwords have an associated list of page numbers of the pages that contain content relevant to the headword
- The list of page numbers can include either or both of: sequences of consecutive page numbers for multiple separate instances of content that is relevant to the headword; and page ranges for relevant content that spans several pages
- Some page numbers can have different styles from others: for example, bold numbers to indicate the most important content, or italic numbers to indicate pages with figures or examples
- When starting from XML or HTML, the index terms are usually included in the markup for the body of the document, and these are sorted and collated to generate the markup for the index. Alternatively, the index could be maintained as an external document with cross-references to IDs in the document being indexed.
- Sorting and collating the index terms to produce a list with no duplicates can be done without formatting the document, but correct page number references to the original locations of the index terms can only be made once the document is formatted
- When an index term appears twice on the one page, the formatter must be able to merge duplicate page numbers
- In some cases, when the same index term appears on successive pages, the formatter must merge successive page numbers into a page range
- Some typographic traditions have shorthand notations for an index term appearing on two or three consecutive pages: for example, ‘8f.’ for an index term on pages 8 and 9, or ‘8ff.’ for an index term on pages 8, 9, and 10.

XSL-FO

XSL 1.1 expects that an XSLT transformation will convert index terms in the source XML into ‘index-key’ properties in the XSL-FO and also sort and collate the index terms to generate the FOs used for formatting the index (or indexes). The bulk of the FOs for a formatted index will be the general-purpose `<fo:block>`, and so on, for the list of headwords. The page references for each headword are generated from an `<fo:index-page-citation-list>`. These FOs are also expected to be generated by the XSLT transformation based on the index terms in the source XML. In fact, it is an error if a contained `<fo:index-key-reference>` does not match any index term in the document.

XSL 1.1 defines multiple FOs for the index reference and, optionally, any prefix or suffix to add to each page number reference. For example, the XSL 1.1 Recommendation shows the example of ‘[’ and ‘]’ around index references to figures. The page numbers, and any prefix or suffix, can be styled using inherited properties such as ‘font-style’ and ‘font-weight’.

There are also properties that specify when and how to merge sequential page numbers and whether to merge page numbers and page ranges across different `<fo:index-key-reference>`. This controls, for example, whether index references to figures can be merged with ‘ordinary’ index references to the same or adjacent pages. AH Formatter extends the ‘merge-sequential-page-numbers’ property to allow shorthand notations to be used for two and three successive pages.

CSS

All of the page number references in the index must be included in the document being formatted. Unlike with XSL-FO, there is no facility for generating page number references based on an index key.

AH Formatter provides the ‘-ah-merge-sequential-page-numbers’ extension property for merging sequential page numbers. When specified on a block-level element, adjacent identical page numbers are merged into one number, and sequences of sequential page numbers are merged into ranges. The page references must be to pages earlier in the document. Generating the shorthand notations for two or three consecutive pages is also possible with CSS.

Suppose the following CSS is specified.

```
p.index-page-citation-list {
 -ah-merge-sequential-page-numbers: merge;
}
span.index-item {
 content: target-counter(attr(href), page);
}
```

Then, suppose the following HTML exists.

```
<p class="index-page-citation-list">
 <span class="index-item" href="#id1"/>,
 <span class="index-item" href="#id2"/>,
 <span class="index-item" href="#id3"/>,
 <span class="index-item" href="#id4"/>,
 <span class="index-item" href="#id5"/>,
 <span class="index-item" href="#id6"/>,
 <span class="index-item" href="#id7"/>,
 <span class="index-item" href="#id8"/>,
 <span class="index-item" href="#id9"/>
</p>
```

Suppose the page numbers line up as follows:

```
1, 3, 4, 4, 5, 6, 8, 8, 9
```

The result will be:

```
1, 3-6, 8, 9
```

If '-ah-merge-sequential-page-numbers: merge-f;' is specified instead, the result will be:

```
1, 3-6, 8f.
```

23 AH FORMATTER FEATURES

This discussion of XSL-FO and CSS has already covered many features and extensions that make AH Formatter uniquely capable of formatting using either XSL-FO or CSS. AH Formatter has yet more features that do not fit neatly into the preceding classifications.

Unless stated otherwise, these features apply to both XSL-FO and CSS formatting.

Two-pass formatting

Two-pass formatting is a feature for formatting large documents that is only available for XSL. This is an AH Formatter feature that is outside the scope of either XSL or CSS.

In single-pass formatting, AH Formatter must keep a page in memory until the page numbers of all `<fo:page-number-citation>` on the page are resolved. All of the following pages are also kept in memory until the page that is the target of the last unresolved `<fo:page-number-citation>` is formatted.

In two-pass formatting, the first pass resolves and saves the `<fo:page-number-citation>` values but does not save any formatted pages. In the second pass, AH Formatter immediately resolves every `<fo:page-number-citation>` using the saved page number information and flushes each page from memory as soon as it is formatted. This allows extremely large documents to be formatted.

Multi-volume output

In XSL formatting but not CSS formatting, a document can be output as multiple PDF files. The bookmarks included in the PDF files can be the same in all volumes or can be different in each volume. The bookmarks in one volume can also refer to IDs in other volumes.

Font configuration

AH Formatter provides extensive control over the fonts that can be used.

The AH Formatter font configuration file specifies:

- Locations of folders containing fonts
- Font files to exclude from use

- Range of Unicode characters to use from a font
- Adjustments to font size, baseline, x-height and other font features
- Aliases that map font names to specific font files

The Option Setting File specifies:

- Mapping of the generic font family names—‘serif’, ‘sans-serif’, ‘monospace’, ‘cursive’ and ‘fantasy’—to specific fonts for individual scripts
- Whether to emulate italics and small-caps when they are not available in the selected font family
- Fonts to use with MathML

Line numbers

AH Formatter provides extension properties for the display of line numbers. Line numbers can be counted starting from: the current page sequence or a previous page sequence; current page; current column; previous column in the current table; or the current block. The position and appearance of the line number can be specified. The interval between displayed line numbers can be specified, as can a sequence of line numbers that must always be shown.

Line continuation marks

AH Formatter provides extension properties for the display of a mark when a ¬
line breaks.

Overprint

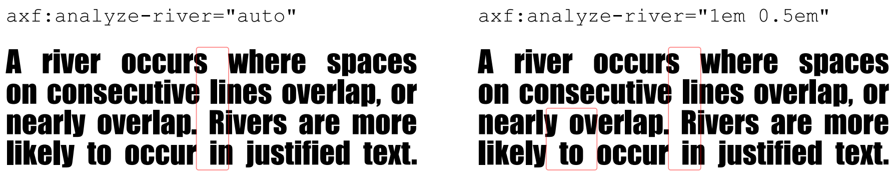
You can specify when a color or colors should overprint underlying colors instead of colors that are masked by other colors not printing at all.

Tabs

In XSL-FO but not CSS, it is possible to define horizontal alignment points—similar to tab stops in a word processor. An `<axf:tab>`, and optionally a tab character (U+0009), aligns its following text on the next tab stop. The start, end, or center of the text, or a decimal point within the text, can be aligned on the tab stop.

Automated analysis

AH Formatter can analyze the formatted text to identify potential typographic problems such as: lines with excessive white-space; rivers of white-space over successive lines; short lines at the end of paragraphs or at the start of a page; consecutive lines starting or ending with the same word; unbalanced spreads; consecutive lines that end with a hyphen; excessive blank pages at the end of the document; and too few lines before or after a block of text.


axf:analyze-river="auto"

A river occurs where spaces on consecutive lines overlap, or nearly overlap. Rivers are more likely to occur in justified text.

axf:analyze-river="1em 0.5em"

A river occurs where spaces on consecutive lines overlap, or nearly overlap. Rivers are more likely to occur in justified text.

Rivers reported in AH Formatter GUI

Automated analysis is quicker and more reliable than visually inspecting every formatted page. However, AH Formatter cannot fix the problems for you. Solving these problems usually requires editorial or stylistic changes, and sometimes both.

Barcode generation

The Barcode Generator Option allows you to embed barcodes, including QR codes, directly in the output as either an SVG or PNG image. Barcode fonts are not required.

```
<!-- XSL-FO example -->
<fo:external-graphic src="data:application/vnd.ah-barcode;type=QR;
scale=2,Hello%20World!"/>
```

```
<!-- HTML example -->
<img src="data:application/vnd.ah-barcode;type=QR;scale=2,
Hello%20World!"/>
```

24 CONCLUSION

XSL-FO and CSS have a lot of similarities because of their commitments to use common properties when XSL 1.0 and CSS2 were developed. More differences developed as XSL and CSS each added new features. AH Formatter smooths out most of the differences by providing many of the properties from one technology as extensions available to the other technology. In addition, AH Formatter provides a large number of original extensions that are, as much as possible, available in both technologies.

Whichever technology you use, or if you use both XSL-FO and CSS, AH Formatter is able to provide more and finer control over the formatted result than any comparable formatter.

REFERENCES

[1995]
: Bert Bos. Report on the W3C style sheet workshop, Paris ’95. URL: https://www.w3.org/Style/951106_Workshop/report1.html

[CSS1]
: World Wide Web Consortium. Cascading Style Sheets, level 1. W3C Recommendation 17 December 1996. URL: https://www.w3.org/TR/REC-CSS1/

[CSS3-Color]
: World Wide Web Consortium. CSS Color Module Level 3. W3C Recommendation, 19 June 2019. URL: https://www.w3.org/TR/css-color-3/

[CSS3-Content]
: World Wide Web Consortium. CSS Generated Content Module Level 3. W3C Working Draft, 2 August 2019. URL: https://www.w3.org/TR/css-content-3/

[CSS3-CounterStyles]
: World Wide Web Consortium. CSS Counter Styles Level 3, W3C Candidate Recommendation, 14 December 2017. URL: https://www.w3.org/TR/2017/CR-css-counter-styles-3-20171214/

[CSS3-Display]
: World Wide Web Consortium. CSS Display Module Level 3, W3C Editor’s Draft, 21 December 2020. URL: https://drafts.csswg.org/css-display-3/#valdef-display-list-item

[CSS3-Page]
: World Wide Web Consortium. CSS Paged Media Module Level 3. W3C Working Draft, 18 October 2018. URL: https://www.w3.org/TR/css-page-3/#page-size-prop

[DSSSL]
: Wikipedia. Document Style Semantics and Specification Language. URL: https://en.wikipedia.org/wiki/Document_Style_Semantics_and_Specification_Language

[GCPM-2007]
: World Wide Web Consortium. CSS3 module: Generated Content for Paged Media. 4 May 2007. URL: http://www.w3.org/TR/2007/WD-css3-gcpm-20070504

[HSSP]
: World Wide Web Consortium. Historical Style Sheet proposals. 6 January 2021. URL: https://www.w3.org/Style/History/ (archive)

[ITCFPM]
: Antenna House. Introduction to CSS for Paged Media, 15 February 2019. URL: https://www.antennahouse.com/css

[JLREQ]
World Wide Web Consortium. *Requirements for Japanese Text Layout 日本語組版処理の要件（日本語版）*. 11 August 2020. URL: https://www.w3.org/TR/jlreq/

[LIE]
Håkon Wium Lie. *Cascading Style Sheets*. Ph.D Thesis. 2005. URL: https://wiumlie.no/2006/phd/ (archive)

[LIEBOS2E]
Håkon Wium Lie and Bert Bos. *The CSS saga*. URL: https://www.w3.org/Style/LieBos2e/history/ (archive)

[TAG]
World Wide Web Consortium. Consistency of Formatting Property Names, Values, and Semantics. TAG Finding. 25 July 2002. URL: https://www.w3.org/2001/tag/doc/formatting-properties.html

[WAI]
World Wide Web Consortium. Caption & Summary in WAI Web Accessibility Tutorials. 27 July 2019. URL: https://www.w3.org/WAI/tutorials/tables/caption-summary/

[WHATWG]
WHAT-WG. The CSS user agent style sheet and presentational hints. 12 August 2021. URL: https://html.spec.whatwg.org/multipage/rendering.html#the-css-user-agent-style-sheet-and-presentational-hints

[XSL10]
World Wide Web Consortium. Extensible Stylesheet Language (XSL). W3C Recommendation. URL: https://www.w3.org/TR/2001/REC-xsl-20011015/

[XS]
Jon Bosak. XS discussion begins. 22 May 1997. URL: http://xml.coverpages.org/xs-970524.html (archive)

[XSL2006]
World Wide Web Consortium. Report from International Workshop on the future of the Extensible Stylesheet Language (XSL-FO) Version 2.0. 18 October 2006. URL: https://www.w3.org/Style/XSL/2006-Workshop/Report.html

[XSLCHARTER]
World Wide Web Consortium. Charter - XSL Working Group. 22 February 2002. URL: https://www.w3.org/Style/2000/xsl-charter.html

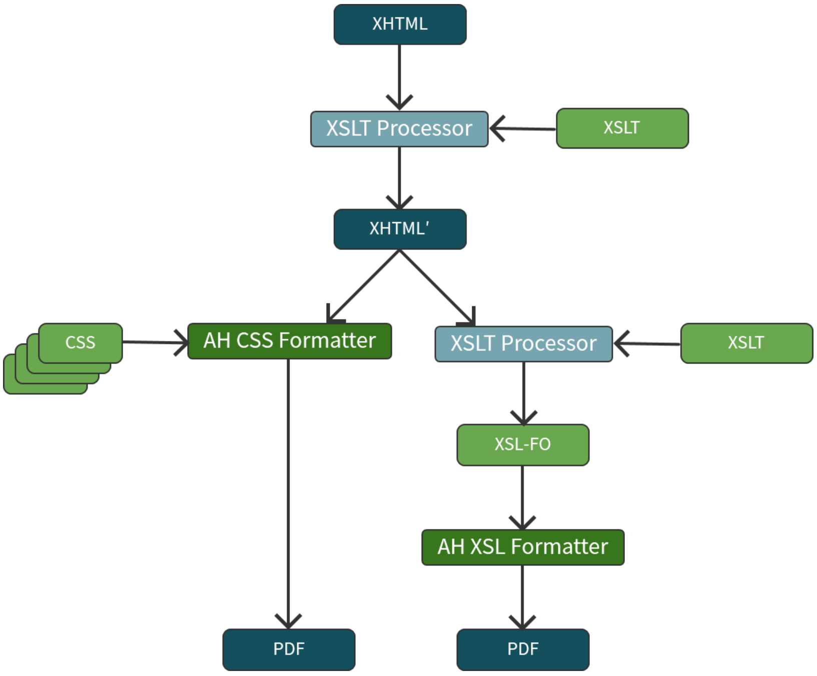
[XSLT10]
World Wide Web Consortium. XSL Transformations (XSLT). W3C Recommendation. 16 November 1999. URL: https://www.w3.org/TR/1999/REC-xslt-19991116

[XSRTF]

Jon Bosak. XML Part 3: Style [NOT YET] Version 1.0. URL: http://sunsite.unc.edu/pub/sun-info/standards/dsssl/xs/xs970522.rtf.zip (archive)

PRODUCTION NOTES

This document compares XSL-FO and CSS, so it is available in two versions—formatted using XSL-FO and formatted using CSS—for you to compare. However, the results are so similar that for many pages, you will need the Antenna House Regression Testing System (AHRTS) to find the differences.



Block diagram

The text of this document is marked up in XHTML5 (XML-serialized HTML5). Before being formatted, the XHTML5 is augmented using XSLT to add the table of contents and syntax highlighting.

The CSS version applies CSS to the augmented XHTML5 using AH CSS Formatter to generate PDF. For the XSL-FO version, the augmented XHTML5 is transformed into XSL-FO markup using XSLT and is then formatted using AH XSL Formatter to generate PDF.

The CSS version reuses the styles from the *Introduction to CSS for Paged Media* [ITCFPM], available from the Antenna House website. It also uses a separate stylesheet module for styles specific to this document, such as the '@page' rule for the landscape pages for the table of AH Formatter properties.

The XSLT stylesheet that generates XSL-FO markup is a custom XSLT 3.0 stylesheet designed to reproduce the appearance of the CSS version. The main stylesheet module imports a version of the `xhtml2fo.xsl` stylesheet from the Antenna House website to handle the basic conversion from XHTML to XSL-FO. The main module, therefore, mostly comprises stylistic overrides corresponding to the different `class` values in the XHTML document.

The high correspondence between XSL and CSS properties, and the even higher correspondence between AH Formatter extensions for XSL and CSS, simplified the task of recreating the CSS styles using XSL. However, because the CSS is divided into multiple modules, it was sometimes necessary to search across the modules to find all of the CSS rules that apply to some contexts.

XSL and CSS use different ways to define page masters, but XSL-FO can express the page layouts and headers and footers used in this document. If the conversion was the other way–from XSL-FO to CSS–then there could have been header and footer content that would require more work to convert. As it is, the content of tabs on the sides of pages are generated when the source XHTML5 is augmented using XSLT, and the augmented XHTML5 includes information on the vertical offset of each tab because CSS will not generate that from the source. The XSLT that generates the XSL-FO just uses that information instead of regenerating the same information from the structure of the XHTML5.

The change to landscape pages for the table of AH Formatter properties just requires `page: BackLandscapeWide;` in CSS, but in the XSL-FO, it required the nested page sequence AH Formatter extension.

XSL-FO/CSS PROPERTIES

The following table shows the correspondence between XSL-FO properties and CSS properties. A blank cell indicates that no corresponding property. To learn the current implementation status for each property, see the XSL-FO Conformance or CSS Conformance section in the AH Formatter manual. CSS Conformance also defines the CSS module abbreviations, such as [CSS3-GCPM], that are used in the table.

Showing properties as corresponding does not always mean that their specifications in XSL-FO and CSS are completely aligned. Some correspondences just indicate that the properties are functionally equivalent or mostly similar. Among the CSS extension properties that have -ah- just added to the property name of the XSL specification, those that have no link destination have the same specifications in XSL and CSS.

XSL	CSS	Description
axf:abbreviation-character-count	-ah-abbreviation-character-count	Specifies the maximum number of characters considered an abbreviation.
7.6.1 absolute-position	[CSS2.1] position	
axf:action-type	-ah-action-type	Specifies the action of External Link or Form Actions.
axf:additive-symbols	[CSS3-CounterStyle] (-ah-)additive-symbols	☞ <axf:counter-style>
axf:adjust-last-line-spacing	-ah-adjust-last-line-spacing	Adjusts the spacing on the last line.
7.14.1 alignment-adjust	[CSS2.1] vertical-align [CSS3-Line] (-ah-)alignment-adjust	
7.14.2 alignment-baseline	[CSS2.1] vertical-align [CSS3-Line] (-ah-)alignment-baseline	
7.15.1 allowed-height-scale	-ah-allowed-height-scale	
7.15.2 allowed-width-scale	-ah-allowed-width-scale	
axf:alttext	-ah-alttext [HTML] alt	Specifies the alternate text of an image or link.
axf:analyze-end-blank-page	-ah-analyze-end-blank-page	Specifies whether to perform end-blank-page analyzer checks on the current FO and its descendants.

XSL	CSS	Description
axf:analyze-hyphen	-ah-analyze-hyphen	Specifies whether to perform hyphen analyzer checks on the current FO and its descendants.
axf:analyze-line-end-repeat	-ah-analyze-line-end-repeat	Specifies whether to perform line-end-repeat analyzer checks on the current FO and its descendants.
axf:analyze-line-start-repeat	-ah-analyze-line-start-repeat	Specifies whether to perform line-start-repeat analyzer checks on the current FO and its descendants.
axf:analyze-lines-after	-ah-analyze-lines-after	Specifies whether to analyze the number of lines after the last area generated by this FO.
axf:analyze-lines-before	-ah-analyze-lines-before	Specifies whether to analyze the number of lines before the first area generated by this FO.
axf:analyze-page-widow	-ah-analyze-page-widow	Specifies whether to perform page widow analyzer checks on the current FO and its descendants.
axf:analyze-paragraph-widow	-ah-analyze-paragraph-widow	Specifies whether to perform paragraph widow analyzer checks on the current FO and its descendants.
axf:analyze-river	-ah-analyze-river	Specifies whether to perform river analyzer checks on the current FO and its descendants.

XSL	CSS	Description
axf:analyze-unbalanced-spread	-ah-analyze-unbalanced-spread	Specifies whether to perform unbalanced-spread analyzer checks on the current FO and its descendants.
axf:analyze-white-space	-ah-analyze-white-space	Specifies whether to perform white-space analyzer checks on the current FO and its descendants.
axf:annotation-author	-ah-annotation-author	Specifies the author of the annotation.
axf:annotation-border-color	-ah-annotation-border-color	Specifies the border color of the free text annotation.
axf:annotation-border-style	-ah-annotation-border-style	Specifies the border style of the free text annotation.
axf:annotation-border-width	-ah-annotation-border-width	Specifies the border width of the free text annotation.
axf:annotation-color	-ah-annotation-color	Specifies the color used for the background of the annotation.
axf:annotation-contents	-ah-annotation-contents	Specifies the content of the annotation.
axf:annotation-createdate	-ah-annotation-createdate	Specifies the annotation creation date.
axf:annotation-file-attachment	-ah-annotation-file-attachment	Specifies the file with which file attachment annotation is related.
axf:annotation-flags	-ah-annotation-flags	Specifies the flag of the annotation.

XSL	CSS	Description
axf:annotation-font-family	-ah-annotation-font-family	Specifies the font family of the free text annotation.
axf:annotation-font-size	-ah-annotation-font-size	Specifies the font size of the free text annotation.
axf:annotation-font-style	-ah-annotation-font-style	Specifies whether to make the font of the free text annotation italic.
axf:annotation-font-weight	-ah-annotation-font-weight	Specifies the font weight of the free text annotation.
axf:annotation-height	-ah-annotation-height	Specifies the height of the annotation.
axf:annotation-icon-name	-ah-annotation-icon-name	Specifies the name of the icon used for displaying the annotation.
axf:annotation-modifydate	-ah-annotation-modifydate	Specifies the annotation modification date.
axf:annotation-open	-ah-annotation-open	Specifies the initial state of the annotation.
axf:annotation-position-horizontal	-ah-annotation-position-horizontal	Specifies the horizontal position of the annotation.
axf:annotation-position-vertical	-ah-annotation-position-vertical	Specifies the vertical position of the annotation.
axf:annotation-text-align	-ah-annotation-text-align	Specifies the alignment of the free text annotation.
axf:annotation-text-color	-ah-annotation-text-color	Specifies the color of the free text annotation.
axf:annotation-title	-ah-annotation-title	Specifies the title of the annotation.
axf:annotation-type	-ah-annotation-type	Specifies the type of the annotation.

XSL	CSS	Description
axf:annotation-width	-ah-annotation-width	Specifies the width of the annotation.
axf:append-non-end-of-line-characters	-ah-append-non-end-of-line-characters	Specifies the non-end-of-line-characters to append.
axf:append-non-starter-characters	-ah-append-non-starter-characters	Specifies the non-starter-characters to append.
axf:assumed-page-number	-ah-assumed-page-number	Specifies the assumed page number.
axf:auto-letter-spacing	-ah-auto-letter-spacing	Changes letter-spacing depending on the number of characters.
axf:avoid-widow-words	-ah-avoid-widow-words	Specifies spacing behavior between words or characters so that the last line of the paragraph does not have only one word left (one character for CJK).
axf:avoid-widow-words-cjk-punctuation	-ah-avoid-widow-words-cjk-punctuation	Specifies whether to include the last punctuation mark and count them in one character when axf:avoid-widow-word="true" is specified in CJK.
7.31.1 background	[CSS2.1] background [CSS3-Background] (-ah-)background	
7.8.1 background-attachment	[CSS2.1] background-attachment [CSS3-Background] (-ah-)background-attachment	
axf:background-clip	[CSS3-Background] (-ah-)background-clip	

XSL	CSS	Description
7.8.2 background-color	[CSS2.1] background-color	
axf:background-content-type	-ah-background-content-type	Specifies the content type of a background image.
7.8.3 background-image	[CSS2.1] background-image [CSS3-Background] (-ah-)background-image	
axf:background-image-resolution	-ah-background-image-resolution	Specifies the resolution of a background image.
axf:background-origin	[CSS3-Background] (-ah-)background-origin	
7.31.2 background-position	[CSS2.1] background-position [CSS3-Background] (-ah-)background-position	
7.8.5 background-position-horizontal	[CSS2.1] background-position	
7.8.6 background-position-vertical	[CSS2.1] background-position	
7.8.4 background-repeat	[CSS2.1] background-repeat [CSS3-Background] (-ah-)background-repeat	
axf:background-size	[CSS3-Background] (-ah-)background-size	
axf:balanced-text-align	-ah-balanced-text-align	Specifies whether to balance the entire block including the last line.
axf:base-uri	-ah-base-uri [XML] xml:base	Specifies the location which becomes the base of relative URI.
axf:baseline-block-snap	-ah-baseline-block-snap	Specifies how to align blocks on the baseline grid.

XSL	CSS	Description
axf:baseline-grid	-ah-baseline-grid	Sets or clears the baseline grid.
7.14.3 baseline-shift	[CSS2.1] vertical-align [CSS3-Line] (-ah-)baseline-shift	
7.27.1 blank-or-not-blank		
axf:bleed	[CSS3-GCPM] (-ah-)bleed	Specifies the width of the bleed area for cutting off.
axf:bleed-bottom	-ah-bleed-bottom	Specifies the width of the bleed area on the bottom for cutting off.
axf:bleed-left	-ah-bleed-left	Specifies the width of the bleed area on the left for cutting off.
axf:bleed-right	-ah-bleed-right	Specifies the width of the bleed area on the right for cutting off.
axf:bleed-top	-ah-bleed-top	Specifies the width of the bleed area on the top for cutting off.
7.15.3 block-progression-dimension	-ah-logical-height	Specifies the block progression dimension.
axf:bookmark-include		Specifies how to include bookmarks in multi separate volume.
	[CSS3-GCPM] (-ah-)bookmark-label	
	[CSS3-GCPM] (-ah-)bookmark-level	

XSL	CSS	Description
	[CSS3-GCPM] (-ah-)bookmark-state	
7.31.3 border	[CSS2.1] border	
	-ah-border-after	Specifies the border of the after side.
7.8.10 border-after-color	-ah-border-after-color	Specifies the border color of the after side.
7.28.1 border-after-precedence		
7.8.11 border-after-style	-ah-border-after-style	Specifies the border style of the after side.
7.8.12 border-after-width	-ah-border-after-width	Specifies the border width of the after side.
	-ah-border-before	Specifies the border of the before side.
7.8.7 border-before-color	-ah-border-before-color	Specifies the border color of the before side.
7.28.2 border-before-precedence		
7.8.8 border-before-style	-ah-border-before-style	Specifies the border style of the before side.
7.8.9 border-before-width	-ah-border-before-width	Specifies the border width of the before side.
7.31.4 border-bottom	[CSS2.1] border-bottom	
7.8.22 border-bottom-color	[CSS2.1] border-bottom-color	
axf:border-bottom-left-radius	[CSS3-Background] (-ah-)border-bottom-left-radius	Specifies the radius of the bottom left rounded corners.

XSL	CSS	Description
axf:border-bottom-right-radius	[CSS3-Background] (-ah-)border-bottom-right-radius	Specifies the radius of the bottom right rounded corners.
7.8.23 border-bottom-style	[CSS2.1] border-bottom-style	
7.8.24 border-bottom-width	[CSS2.1] border-bottom-width	
7.28.3 border-collapse	[CSS2.1] border-collapse	
7.31.5 border-color	[CSS2.1] border-color	
axf:border-double-thickness	-ah-border-double-thickness	Specifies the line thickness of border-style="double".
	-ah-border-end	Specifies the border of the end side.
7.8.16 border-end-color	-ah-border-end-color	Specifies the border color of the end side.
7.28.4 border-end-precedence		
7.8.17 border-end-style	-ah-border-end-style	Specifies the border style of the end side.
7.8.18 border-end-width	-ah-border-end-width	Specifies the border width of the end side.
7.31.6 border-left	[CSS2.1] border-left	
7.8.25 border-left-color	[CSS2.1] border-left-color	
7.8.26 border-left-style	[CSS2.1] border-left-style	
7.8.27 border-left-width	[CSS2.1] border-left-width	

XSL	CSS	Description
	[CSS3-GCPM] (-ah-)border-length	
axf:border-radius	[CSS3-Background] (-ah-)border-radius	Specifies the radii of the rounded corners.
7.31.7 border-right	[CSS2.1] border-right	
7.8.28 border-right-color	[CSS2.1] border-right-color	
7.8.29 border-right-style	[CSS2.1] border-right-style	
7.8.30 border-right-width	[CSS2.1] border-right-width	
7.28.5 border-separation		
7.31.9 border-spacing	[CSS2.1] border-spacing	
	-ah-border-start	Specifies the border of the start side.
7.8.13 border-start-color	-ah-border-start-color	Specifies the border color of the start side.
7.28.6 border-start-precedence		
7.8.14 border-start-style	-ah-border-start-style	Specifies the border style of the start side.
7.8.15 border-start-width	-ah-border-start-width	Specifies the border width of the start side.
7.31.8 border-style	[CSS2.1] border-style	
7.31.10 border-top	[CSS2.1] border-top	
7.8.19 border-top-color	[CSS2.1] border-top-color	

XSL	CSS	Description
axf:border-top-left-radius	[CSS3-Background] (-ah-)border-top-left-radius	Specifies the radius of the top left rounded corners.
axf:border-top-right-radius	[CSS3-Background] (-ah-)border-top-right-radius	Specifies the radius of the top right rounded corners.
7.8.20 border-top-style	[CSS2.1] border-top-style	
7.8.21 border-top-width	[CSS2.1] border-top-width	
axf:border-wave-form	-ah-border-wave-form	Specifies the wave form of border-style="wave".
7.31.11 border-width	[CSS2.1] border-width	
7.6.4 bottom	[CSS2.1] bottom	
	[CSS3-Break] (-ah-)box-decoration-break	
axf:box-shadow	[CSS3-Background] (-ah-)box-shadow	Specifies the box shadow.
	[CSS3-UI] (-ah-)box-sizing	
7.20.1 break-after	[CSS3-Multicol] (-ah-)break-after	
7.20.2 break-before	[CSS3-Multicol] (-ah-)break-before	
7.28.7 caption-side	[CSS2.1] caption-side	
7.30.1 change-bar-class	[CSS3-GCPM] (-ah-)change-bar-class	
7.30.2 change-bar-color	[CSS3-GCPM] (-ah-)change-bar-color	

XSL	CSS	Description
7.30.3 change-bar-offset	[CSS3-GCPM] (-ah-)change-bar-offset	
7.30.4 change-bar-placement	[CSS3-GCPM] (-ah-)change-bar-side	
7.30.5 change-bar-style	[CSS3-GCPM] (-ah-)change-bar-style	
7.30.6 change-bar-width	[CSS3-GCPM] (-ah-)change-bar-width	
7.17.1 character		
7.19.1 clear	[CSS2.1] clear	
7.21.1 clip	[CSS2.1] clip	
7.18.1 color	[CSS2.1] color	
7.18.2 color-profile-name		
7.27.2 column-count	[CSS3-Multicol] (-ah-)column-count	
axf:column-fill	[CSS3-Multicol] (-ah-)column-fill	Specifies whether to balance the column height.
7.27.3 column-gap	[CSS3-Multicol] (-ah-)column-gap	
7.28.8 column-number		
axf:column-number-format		Specifies the format of column number.
axf:column-rule	[CSS3-Multicol] (-ah-)column-rule	Draws the column rules in the column gaps.
axf:column-rule-align	-ah-column-rule-align	Specifies the alignment of the column rule.

XSL	CSS	Description
axf:column-rule-color	[CSS3-Multicol] (-ah-)column-rule-color	Specifies the color of the column rule.
axf:column-rule-display	-ah-column-rule-display	Specifies whether to also display a rule at the place where column gaps do not exist.
axf:column-rule-length	-ah-column-rule-length	Specifies the length of the column rule.
axf:column-rule-style	[CSS3-Multicol] (-ah-)column-rule-style	Specifies the style of the column rule.
axf:column-rule-width	[CSS3-Multicol] (-ah-)column-rule-width	Specifies the width of the column rule.
7.28.9 column-width	[CSS3-Multicol] (-ah-)column-width	
	[CSS3-Multicol] (-ah-)columns	
axf:condensed-text-align-last	-ah-condensed-text-align-last	Specifies whether to set text-align-last="justify" automatically after condensing the overflow.
	[CSS2.1] content	
7.15.4 content-height	-ah-content-height	
7.30.7 content-type	-ah-content-type	
7.15.5 content-width	-ah-content-width	
	[CSS2.1] counter-increment	
	[CSS2.1] counter-reset	
7.10.1 country	-ah-country	

XSL	CSS	Description
axf:crop-area-visibility	-ah-crop-area-visibility	Specifies whether to display the area that is extended beyond the trim size.
axf:crop-offset	-ah-crop-offset	Specifies the distance from the physical end to the trim size of the output medium.
axf:crop-offset-bottom	-ah-crop-offset-bottom	Specifies the distance from the physical bottom edge to the trim size of the output medium.
axf:crop-offset-left	-ah-crop-offset-left	Specifies the distance from the physical left edge to the trim size of the output medium.
axf:crop-offset-right	-ah-crop-offset-right	Specifies the distance from the physical right edge to the trim size of the output medium.
axf:crop-offset-top	-ah-crop-offset-top	Specifies the distance from the physical top edge to the trim size of the output medium.
axf:destination-type	-ah-destination-type	Specifies the way a link opens in a link destination.
axf:diagonal-border-color	-ah-diagonal-border-color	Specifies the color of the diagonal border.
axf:diagonal-border-style	-ah-diagonal-border-style	Specifies the style of the diagonal border.
axf:diagonal-border-width	-ah-diagonal-border-width	Specifies the width of the diagonal border.
7.29.1 direction	[CSS2.1] direction	

XSL	CSS	Description
	[CSS2.1] display [CSS3-Box] (-ah-)display	
7.14.4 display-align	[CSS2.1] vertical-align -ah-display-align	
axf:display-alttext	-ah-display-alttext	Specifies whether to display the alternate text of an image.
axf:document-info-include		Specifies how to include document information in multi separate volume.
7.14.5 dominant-baseline	[CSS3-Line] (-ah-)dominant-baseline	
7.11.8 end-indent		
7.28.11 ends-row		
axf:except-non-end-of-line-characters	-ah-except-non-end-of-line-characters	Specifies the non-end-of-line-characters to eliminate.
axf:except-non-starter-characters	-ah-except-non-starter-characters	Specifies the non-starter-characters to eliminate.
axf:expansion-text	-ah-expansion-text	Specifies the expansion text for tags in Tagged PDF.
7.27.4 extent		
7.23.6 external-destination	[HTML] href [XML] xlink:href	

XSL	CSS	Description
axf:fallback	[CSS3-CounterStyle] (-ah-)fallback	☞ <axf:counter-style>
axf:field-button-face	-ah-field-button-face	Specifies the caption displayed in the push button field.
axf:field-button-face-down	-ah-field-button-face-down	Specifies the caption displayed when pressing the push button.
axf:field-button-face-rollover	-ah-field-button-face-rollover	Specifies the caption displayed when rolling over the push button.
axf:field-button-icon	-ah-field-button-icon	Specifies the icon displayed in the push button field.
axf:field-button-icon-down	-ah-field-button-icon-down	Specifies the icon displayed when pressing the push button.
axf:field-button-icon-rollover	-ah-field-button-icon-rollover	Specifies the icon displayed when rolling over the push button.
axf:field-button-layout	-ah-field-button-layout	Specifies the positioning between the caption and icon displayed in the push button field.
axf:field-checked	-ah-field-checked	Specifies the initial state of the check box and the radio button.
axf:field-checked-style	-ah-field-checked-style	Specifies the style of the check box and the radio button.

XSL	CSS	Description
axf:field-default-text	-ah-field-default-text	Specifies the text entered into the text field from the beginning.
axf:field-description	-ah-field-description	Specifies the descriptive text of the field.
axf:field-editable	-ah-field-editable	Specifies whether the value can be edited with the combo box.
axf:field-flags	-ah-field-flags	Specifies the flag of the field.
axf:field-font-size	-ah-field-font-size	Specifies the font size of the character string displayed in the form fields.
axf:field-format	-ah-field-format	Specifies the format of the text field.
axf:field-format-category	-ah-field-format-category	Specifies the format type of the text field.
axf:field-lock-document	-ah-field-lock-document	Specifies whether to lock the document at the time of signing with the digital signature field.
axf:field-maxlen	-ah-field-maxlen	Specifies the maximum number of characters which can be entered into the text field.
axf:field-multiline	-ah-field-multiline	Specifies whether the text field is a single-line enterable field or a multi-line enterable field.
axf:field-multiple	-ah-field-multiple	Specifies whether multiple items can be chosen in the list box.

XSL	CSS	Description
axf:field-name	-ah-field-name	Specifies the field name.
axf:field-name-suffix-page-number		Adds a page number to the field name.
axf:field-password	-ah-field-password	Specifies whether the text field requires the password or not.
axf:field-readonly	-ah-field-readonly	Specifies whether the field is read-only or not.
axf:field-required	-ah-field-required	Specifies whether the field is enter-required or not.
axf:field-scroll	-ah-field-scroll	Specifies whether the text field is scrollable or not.
axf:field-selected	-ah-field-selected	Specifies the first selected item in the list box, combo box.
axf:field-submit-coordinates	-ah-field-submit-coordinates	Specifies whether to send out the coordinates of the mouse when submitting a form field.
axf:field-submit-method	-ah-field-submit-method	Specifies the way to send the information when submitting a form field.
axf:field-text-align	-ah-field-text-align	Specifies the alignment of the text field.
axf:field-top-index	-ah-field-top-index	Specifies the first selected item in the list box, combo box.
axf:field-type	-ah-field-type	Specifies the field type.

XSL	CSS	Description
axf:field-value	-ah-field-value	Specifies the value used when submitting a form field, etc.
7.19.2 float	[CSS2.1] float	
axf:float	[CSS3-GCPM] (-ah-)float	This is a shorthand property for setting float related extension properties.
axf:float-centering-x	-ah-float-centering-x	Specifies whether the float is centered when the width for the text wrapping around the float is insufficient.
axf:float-centering-y	-ah-float-centering-y	Specifies whether the float is centered when the extent for the text placed before and after the float is insufficient.
axf:float-float-margin-x	-ah-float-float-margin-x	Specifies the space between the float and another neighboring float (in x-axis).
axf:float-float-margin-y	-ah-float-float-margin-y	Specifies the space between the float and another neighboring float (in y-axis).
axf:float-margin-x	-ah-float-margin-x	Specifies the space between the float and the text wrapping around the float (in x-axis).
axf:float-margin-y	-ah-float-margin-y	Specifies the space between the float and the text before and after the float (in y-axis).

XSL	CSS	Description
axf:float-min-wrap-x	-ah-float-min-wrap-x	Specifies the minimum width for the text wrapping around the float.
axf:float-min-wrap-y	-ah-float-min-wrap-y	Specifies the minimum extent for the text placed before and after the float.
axf:float-move	-ah-float-move	Specifies whether the float moves to the next page (or column).
axf:float-offset-x	-ah-float-offset-x	Specifies the offset placement for the float (in x-axis).
axf:float-offset-y	-ah-float-offset-y	Specifies the offset placement for the float (in y-axis).
axf:float-reference	-ah-float-reference	Specifies reference area where the float is placed.
axf:float-wrap	-ah-float-wrap	Specifies the text wrapping.
axf:float-x	-ah-float-x	Specifies horizontal (or vertical if writing-mode is vertical) float alignment.
axf:float-y	-ah-float-y	Specifies vertical (or horizontal if writing-mode is vertical) float alignment.
7.27.18 flow-map-name		
7.27.19 flow-map-reference		

XSL	CSS	Description
7.27.5 flow-name		
7.27.20 flow-name-reference		
axf:flush-zone	-ah-flush-zone	Adjusts the space at the end of the last line.
7.31.13 font	[CSS2.1] font	
7.9.2 font-family	[CSS2.1] font-family	
axf:font-feature-settings	[CSS3-Fonts] (-ah-)font-feature-settings	Controls the feature of OpenType fonts.
7.9.3 font-selection-strategy		
7.9.4 font-size	[CSS2.1] font-size	
7.9.6 font-size-adjust	[CSS3-Fonts] (-ah-)font-size-adjust	
7.9.5 font-stretch	[CSS3-Fonts] (-ah-)font-stretch	
7.9.7 font-style	[CSS2.1] font-style	
7.9.8 font-variant	[CSS2.1] font-variant [CSS3-Fonts] (-ah-)font-variant	
7.9.9 font-weight	[CSS2.1] font-weight	
axf:footnote-align		Specifies the alignment of the footnotes.
axf:footnote-keep	-ah-footnote-keep	Specifies whether to arrange a footnote and an anchor in the same page

XSL	CSS	Description
axf:footnote-max-height	-ah-footnote-max-height	Specifies the maximum height of footnote.
axf:footnote-number-format		Specifies the format of footnote number.
axf:footnote-number-initial		Specifies the initial footnote number.
axf:footnote-number-reset		Resets the footnote numbering.
axf:footnote-position		Specifies the location to place the footnote.
axf:footnote-stacking	-ah-footnote-stacking	Specifies the method to layout the footnote.
7.27.6 force-page-count		
7.26.1 format		
7.29.2 glyph-orientation-horizontal		
7.29.3 glyph-orientation-vertical		
7.26.2 grouping-separator		
7.26.3 grouping-size		
axf:hanging-punctuation	[CSS3-Text] (-ah-)hanging-punctuation	Specifies whether to hang Japanese punctuation characters or not.
axf:headers		Specifies the table header cells associated with this cell.
7.15.6 height	[CSS2.1] height	

XSL	CSS	Description
7.10.4 hyphenate	[CSS3-Text] (-ah-)hyphens	
axf:hyphenate-caps-word	-ah-hyphenate-caps-word	Specifies whether to hyphenate a word that consists of capital letters.
axf:hyphenate-hyphenated-word	-ah-hyphenate-hyphenated-word	Specifies whether to hyphenate the already hyphenated word or not.
7.10.5 hyphenation-character	[CSS3-GCPM] (-ah-)hyphenate-character	
7.16.1 hyphenation-keep	-ah-hyphenation-keep	
7.16.2 hyphenation-ladder-count	[CSS3-GCPM] (-ah-)hyphenate-lines	
axf:hyphenation-minimum-character-count	-ah-hyphenation-minimum-character-count	Specifies the minimum number of characters a word must have before it can be hyphenated.
7.10.6 hyphenation-push-character-count	[CSS3-GCPM] (-ah-)hyphenate-after	
7.10.7 hyphenation-remain-character-count	[CSS3-GCPM] (-ah-)hyphenate-before	
axf:hyphenation-zone	-ah-hyphenation-zone	Limits the range where a hyphenation is available.
7.30.8 id	[HTML] id [XML] xml:id	
	-ah-ignore-leading-newline	Specifies whether the newline right after the start tag is disregarded or not.

XSL	CSS	Description
axf:image-resolution	[CSS3-GCPM] (-ah-)image-resolution	Specifies the resolution of an image.
axf:image-smoothing	-ah-image-smoothing	Specifies whether to process anti-aliasing of an image on the screen.
axf:indent-here	-ah-indent-here	Aligns the indent position to the region position when a line break occurs.
7.24.1 index-class		
7.24.2 index-key		
	-ah-index-page-citation-range-f-suffix	Specifies the suffix when merging 2 consecutive page numbers.
	-ah-index-page-citation-range-ff-suffix	Specifies the suffix when merging 3 consecutive page numbers.
	-ah-index-page-citation-range-separator	Specifies the separator string when merging consecutive page numbers.
axf:initial-letters	-ah-initial-letters	Creates drop initials.
axf:initial-letters-color		Specifies the text color of a dropped initial.
axf:initial-letters-end-indent	-ah-initial-letters-end-indent	Specifies the space on the end side of a dropped initial.

XSL	CSS	Description
axf:initial-letters-first-line-head-height	-ah-initial-letters-first-line-head-height	Specifies where the height of a dropped initial should be adjusted.
axf:initial-letters-leading-punctuation	-ah-initial-letters-leading-punctuation	Specifies the size of the leading punctuation of a dropped initial, etc.
axf:initial-letters-leading-punctuation-position	-ah-initial-letters-leading-punctuation-position	Specifies the position of the leading punctuation of a dropped initial in the inline progression direction.
axf:initial-letters-leading-punctuation-shift	-ah-initial-letters-leading-punctuation-shift	Specifies the position of the leading punctuation of a dropped initial in the block progression direction.
axf:initial-letters-text-align		Specifies the alignment of a dropped initial.
axf:initial-letters-width		Specifies the width of a dropped initial.
7.27.7 initial-page-number		
axf:initial-volume-number		Specifies the initial volume number in multi separate volume.
axf:inline-overflow-align	-ah-inline-overflow-align	Makes adjustments when the blocks in <fo:inline-container> overflow.
7.15.7 inline-progression-dimension	-ah-logical-width	Specifies the inline progression dimension.
7.23.8 internal-destination	-ah-internal-destination	

XSL	CSS	Description
7.30.9 intrinsic-scale-value		
axf:intrude-into-punctuation	-ah-intrude-into-punctuation	Intrudes the inline element into the punctuation.
7.19.3 intrusion-displace	-ah-intrusion-displace	
axf:justify-nbsp	-ah-justify-nbsp	Specifies whether to justify NON-BREAKING SPACE or not.
axf:kansuji-grouping-letter	-ah-kansuji-grouping-letter	Specifies the grouping character used for Japanese numerals.
axf:kansuji-letter	-ah-kansuji-letter	Specifies the character used for Japanese numerals.
axf:kansuji-style	-ah-kansuji-style	Specifies the style used for Japanese numerals.
	[CSS3-Multicol] (-ah-)break-inside	
7.20.3 keep-together	-ah-keep-together	
axf:keep-together-within-dimension	-ah-keep-together-within-dimension	Specifies the upper limit height of the keep-together condition.
axf:keep-together-within-inline-dimension	-ah-keep-together-within-inline-dimension	Specifies the upper limit width of the keep-together.within-line condition.
7.20.4 keep-with-next		
7.20.5 keep-with-previous		

XSL	CSS	Description
axf:kerning-mode	-ah-kerning-mode	Specifies whether to process the kerning.
7.10.2 language	-ah-language	
7.16.3 last-line-end-indent	-ah-last-line-end-indent	
axf:layer	-ah-layer	Specifies to which layer the area is arranged.
axf:layer-settings	-ah-layer-settings	Defines layers.
7.22.1 leader-alignment		
axf:leader-expansion		Specifies whether to expand a leader forcibly.
7.22.4 leader-length	-ah-leader-length	
7.22.2 leader-pattern		
7.22.3 leader-pattern-width		
7.6.5 left	[CSS2.1] left	
axf:left-page-master-reference		Master name of the page master for the left page of a two-page spread.
7.17.2 letter-spacing	[CSS2.1] letter-spacing	
axf:letter-spacing-side	-ah-letter-spacing-side	Specifies on which side of the character the space by letter-spacing is distributed.
7.26.4 letter-value		

XSL	CSS	Description
axf:ligature-mode	-ah-ligature-mode	Specifies whether to perform the ligature processing.
axf:line-break	[CSS3-Text] (-ah-)line-break	Specifies the method of line breaking.
axf:line-continued-mark	-ah-line-continued-mark	Specifies whether to show line continued marks.
axf:line-continued-mark-background-color	-ah-line-continued-mark-background-color	Specifies the background color of line continued marks.
axf:line-continued-mark-color	-ah-line-continued-mark-color	Specifies the color of line continued marks.
axf:line-continued-mark-font-family	-ah-line-continued-mark-font-family	Specifies the font family of line continued marks.
axf:line-continued-mark-font-size	-ah-line-continued-mark-font-size	Specifies the font size of line continued marks.
axf:line-continued-mark-font-style	-ah-line-continued-mark-font-style	Specifies whether to make the font style italic.
axf:line-continued-mark-font-weight	-ah-line-continued-mark-font-weight	Specifies the font weight of line numbers.
axf:line-continued-mark-offset	-ah-line-continued-mark-offset	Specifies the offset of line continued marks.
7.16.4 line-height	[CSS2.1] line-height	
7.16.5 line-height-shift-adjustment	-ah-line-height-shift-adjustment	
axf:line-number	-ah-line-number	Specifies whether to show line numbers.
axf:line-number-background-color	-ah-line-number-background-color	Specifies the background color of line numbers.
axf:line-number-color	-ah-line-number-color	Specifies the color of line numbers.

XSL	CSS	Description
axf:line-number-display-align	-ah-line-number-display-align	Specifies the alignment, in the block-progression-direction, of line numbers in the line area.
axf:line-number-except-continued-line	-ah-line-number-except-continued-line	Specifies whether to add line numbers except for continued lines.
axf:line-number-font-family	-ah-line-number-font-family	Specifies the font family of line numbers.
axf:line-number-font-size	-ah-line-number-font-size	Specifies the font size of line numbers.
axf:line-number-font-style	-ah-line-number-font-style	Specifies whether to make the font style italic.
axf:line-number-font-weight	-ah-line-number-font-weight	Specifies the font weight of line numbers.
axf:line-number-format	-ah-line-number-format	Specifies the format of line numbers.
axf:line-number-initial	-ah-line-number-initial	Specifies the line number of the first line.
axf:line-number-interval	-ah-line-number-interval	Specifies the interval of line numbers.
axf:line-number-offset	-ah-line-number-offset	Specifies the offset of line numbers.
axf:line-number-orientation	-ah-line-number-orientation	Rotates line numbers.
axf:line-number-position	-ah-line-number-position	Specifies the position of line numbers.
axf:line-number-prefix	-ah-line-number-prefix	Sets the prefix of line number.
axf:line-number-reset	-ah-line-number-reset	Resets line numbering.
axf:line-number-show	-ah-line-number-show	Specifies the line number to be always output.

XSL	CSS	Description
axf:line-number-start	-ah-line-number-start	Specifies the starting line number.
axf:line-number-text-align	-ah-line-number-text-align	Specifies the alignment of line numbers in the line area.
axf:line-number-text-decoration	-ah-line-number-text-decoration	Specifies the test decoration of line numbers.
axf:line-number-width	-ah-line-number-width	Specifies the width of line numbers.
7.16.6 line-stacking-strategy	-ah-line-stacking-strategy	
7.16.7 linefeed-treatment	[CSS2.1] white-space -ah-linefeed-treatment	
	-ah-link	Generates a hyper-link.
	[CSS2.1] list-style	
	[CSS2.1] list-style-image	
	[CSS2.1] list-style-position	
	[CSS2.1] list-style-type [CSS3-Lists] (-ah-)list-style-type	Specifies the list style.
7.31.14 margin	[CSS2.1] margin	
7.11.2 margin-bottom	[CSS2.1] margin-bottom	

XSL	CSS	Description
	-ah-margin-break	Specifies how to treat the margin when the page/ column breaks.
7.11.3 margin-left	[CSS2.1] margin-left	
7.11.4 margin-right	[CSS2.1] margin-right	
7.11.1 margin-top	[CSS2.1] margin-top	
7.25.1 marker-class-name		
7.27.8 master-name		
7.27.9 master-reference		
7.15.8 max-height	[CSS2.1] max-height	
	-ah-max-logical-height	Specifies the maximum block progression dimension.
	-ah-max-logical-width	Specifies the maximum inline progression dimension.
7.15.9 max-width	[CSS2.1] max-width	
7.27.10 maximum-repeats		
axf:media-activation	-ah-media-activation	Specifies when to activate / deactivate the rich media.

XSL	CSS	Description
axf:media-duration	-ah-media-duration	Specifies the duration of a time period of the multimedia.
axf:media-extraction-policy	-ah-media-extraction-policy	Specifies whether the creation of temporary files is allowed or not when playing the multimedia.
axf:media-flash-context-menu	-ah-media-flash-context-menu	Specifies whether to display the context menu of Flash When embedding Flash in the rich media.
axf:media-flash-vars	-ah-media-flash-vars	Specifies the variable when embedding Flash in the rich media.
axf:media-play-mode	-ah-media-play-mode	Specifies the number of times to play the multimedia.
axf:media-skin-auto-hide	-ah-media-skin-auto-hide	Specifies whether to automatically hide rich media skins or not.
axf:media-skin-color	-ah-media-skin-color	Specifies the skin color of the rich media.
axf:media-skin-control	-ah-media-skin-control	Specifies the skin control of the rich media.
axf:media-transparent-background	-ah-media-transparent-background	Specifies whether to make the background transparent when embedding Flash in the rich media.
axf:media-volume	-ah-media-volume	Specifies the volume of the sound when playing the multimedia.

XSL	CSS	Description
axf:media-window-height	-ah-media-window-height	Specifies the height of the rich media window.
axf:media-window-width	-ah-media-window-width	Specifies the width of the rich media window.
7.24.6 merge-pages-across-index-key-references		
7.24.4 merge-ranges-across-index-key-references		
7.24.5 merge-sequential-page-numbers	-ah-merge-sequential-page-numbers	Specifies the page number reference to be merged.
7.15.10 min-height	[CSS2.1] min-height	
	-ah-min-logical-height	Specifies the minimum block progression dimension.
	-ah-min-logical-width	Specifies the minimum inline progression dimension.
7.15.11 min-width	[CSS2.1] min-width	
axf:multimedia-treatment	-ah-multimedia-treatment	Specifies whether to embed Multimedia in PDF.
axf:name		☞ <axf:document-info> <axf:counter-style>
axf:negative	[CSS3-CounterStyle] (-ah-)negative	☞ <axf:counter-style>
axf:normalize	-ah-normalize	Specifies the normalization of text.

XSL	CSS	Description
axf:normalize-exclude	-ah-normalize-exclude	Specifies whether Composition Exclusions are excluded or not when the normalization is specified.
7.28.12 number-columns-repeated		
7.28.13 number-columns-spanned	[CSS3-Tables] (-ah-)table-column-span [HTML] colspan	
7.28.14 number-rows-spanned	[CSS3-Tables] (-ah-)table-row-span [HTML] rowspan	
axf:number-transform	-ah-number-transform	Converts the number sequence in the character string.
axf:number-type		Specifies whether to output the page number or to output the column number.
	[CSS3-Images] (-ah-)object-fit	
	[CSS3-Images] (-ah-)object-position	
7.27.12 odd-or-even		
axf:origin-id		Specifies the origin of the page number.
7.20.6 orphans	[CSS2.1] orphans	

XSL	CSS	Description
axf:outline-color	-ah-outline-color	Specifies the color which appears as a title of bookmarks.
axf:outline-expand	-ah-outline-expand	Specifies whether to display the lower hierarchy of bookmark items or not.
axf:outline-external-destination	-ah-outline-external-destination	Sets the external link in the PDF bookmark.
axf:outline-font-style	-ah-outline-font-style	Specifies the font style which appears as a title of bookmarks.
axf:outline-font-weight	-ah-outline-font-weight	Specifies the font weight which appears as a title of bookmarks.
axf:outline-group	-ah-outline-group	Groups bookmark items, and outputs them collectively.
axf:outline-internal-destination	-ah-outline-internal-destination	Sets the internal link in the PDF bookmark.
axf:outline-level	-ah-outline-level	Indicates the hierarchy level of bookmark items.
axf:outline-title	-ah-outline-title	Specifies the string which appears as a title of bookmarks.
axf:output-volume-break		Separates the file in multi volume.
axf:output-volume-filename		Specifies the document file name in multi separate volume.

XSL	CSS	Description
7.21.2 overflow	[CSS2.1] overflow	
axf:overflow-align	-ah-overflow-align	Specifies the alignment of the overflowed block.
axf:overflow-condense	-ah-overflow-condense	Specifies how to condense the overflowed text within the region.
axf:overflow-condense-limit-font-size	-ah-overflow-condense-limit-font-size	Specifies the lower limit font size when axf:overflow-condense="font-size" is specified.
axf:overflow-condense-limit-font-stretch	-ah-overflow-condense-limit-font-stretch	Specifies the lower limit value when axf:overflow-condense="font-stretch" is specified.
axf:overflow-condense-limit-letter-spacing	-ah-overflow-condense-limit-letter-spacing	Specifies the lower limit value when axf:overflow-condense="letter-spacing" is specified.
axf:overflow-condense-limit-line-height	-ah-overflow-condense-limit-line-height	Specifies the lower limit value when axf:overflow-condense="line-height" is specified.
axf:overflow-limit	-ah-overflow-limit	Specifies the overflow limit value.
axf:overflow-limit-block	-ah-overflow-limit-block	Specifies the block overflow limit value.
axf:overflow-limit-inline	-ah-overflow-limit-inline	Specifies the inline overflow limit value.
axf:overflow-replace	-ah-overflow-replace	Specifies an alternative character string for the overflowed text.
axf:overprint	-ah-overprint	Specifies the overprint.

XSL	CSS	Description
axf:pad	[CSS3-CounterStyle] (-ah-)pad	☞ <axf:counter-style>
7.31.15 padding	[CSS2.1] padding	
7.8.32 padding-after	-ah-padding-after	
7.8.31 padding-before	-ah-padding-before	
7.8.36 padding-bottom	[CSS2.1] padding-bottom	
7.8.34 padding-end	-ah-padding-end	
7.8.37 padding-left	[CSS2.1] padding-left	
7.8.38 padding-right	[CSS2.1] padding-right	
7.8.33 padding-start	-ah-padding-start	
7.8.35 padding-top	[CSS2.1] padding-top	
	[CSS3-GCPM] (-ah-)page	
7.31.16 page-break-after	[CSS2.1] page-break-after	
7.31.17 page-break-before	[CSS2.1] page-break-before	
7.31.18 page-break-inside	[CSS2.1] page-break-inside	
7.30.10 page-citation-strategy		
7.27.13 page-height		

XSL	CSS	Description
axf:page-number-prefix		Sets the prefix of page number.
7.24.3 page-number-treatment		
7.27.14 page-position		
7.27.15 page-width		
axf:pdftag	-ah-pdftag	Specifies the tag name of Tagged PDF.
axf:physical-page-number		Gets physical page number.
7.31.20 position	[CSS2.1] position	
axf:poster-content-type	-ah-poster-content-type	Specifies the content type of the poster image for embedded multimedia.
axf:poster-image	-ah-poster-image	Specifies the poster image for embedded multimedia.
7.27.16 precedence		
axf:prefix	[CSS3-CounterStyle] (-ah-)prefix	☞ <axf:counter-style>
axf:printer-bin-selection	-ah-printer-bin-selection	Selects the printer tray.
axf:printer-duplex	-ah-printer-duplex	Specifies to print in duplex mode.
axf:printer-marks	[CSS3-GCPM] (-ah-)marks -ah-printer-marks	Specifies the Printing marks, such as a crop mark. Specifies the action of external link.

XSL	CSS	Description
axf:printer-marks-line-color	-ah-printer-marks-line-color	Specifies the line color of printer marks.
axf:printer-marks-line-length	-ah-printer-marks-line-length	Specifies the line length of printer marks.
axf:printer-marks-line-width	-ah-printer-marks-line-width	Specifies the line width of printer marks.
axf:printer-marks-spine-width	-ah-printer-marks-spine-width	Specifies the spine width of the facing page.
axf:printer-marks-zero-margin	-ah-printer-marks-zero-margin	Specifies the margin between the page and the printer marks when bleed is 0.
7.30.12 provisional-distance-between-starts		
7.30.11 provisional-label-separation		
axf:punctuation-spacing	-ah-punctuation-spacing	Specifies the trimming spacing between a full width punctuation and a full width character in Japanese.
axf:punctuation-trim	[CSS3-Text] (-ah-)punctuation-trim	The axf:punctuation-trim specifies whether to treat full width punctuation marks as half width in Japanese.
	[CSS2.1] quotes	
axf:quotetype	-ah-quotetype	Specifies the direction of the quotes.
axf:range	[CSS3-CounterStyle] (-ah-)range	☞ <axf:counter-style>

XSL	CSS	Description
7.30.13 ref-id		
7.24.7 ref-index-key		
7.21.3 reference-orientation	-ah-reference-orientation	
7.27.17 region-name		
7.27.21 region-name-reference		
7.14.6 relative-align	[CSS2.1] vertical-align	
7.13.5 relative-position	[CSS2.1] position	
axf:repeat-cell-content-at-break	-ah-repeat-cell-content-at-break	Specifies whether to copy the contents of a cell when a cell breaks.
axf:repeat-footnote-in-table-footer		Specifies whether to repeat the <fo:footnote> in the <fo:table-footer> that is repeated by table-omit-footer-at-break="false".
axf:repeat-footnote-in-table-header		Specifies whether to repeat the <fo:footnote> in the <fo:table-header> that is repeated by table-omit-header-at-break="false".
axf:repeat-page-sequence-master		Specifies the repetition of the page sequence.
7.25.5 retrieve-boundary		
7.25.2 retrieve-boundary-within-table		

XSL	CSS	Description
7.25.3 retrieve-class-name		
7.25.4 retrieve-position		
7.25.6 retrieve-position-within-table		
axf:retrieve-table-rows		Specifies the maximum number of fo:table-row in fo:marker referenced from fo:retrieve-table-marker.
axf:reverse-diagonal-border-color	-ah-reverse-diagonal-border-color	Specifies the color of the reverse diagonal border.
axf:reverse-diagonal-border-style	-ah-reverse-diagonal-border-style	Specifies the style of the reverse diagonal border.
axf:reverse-diagonal-border-width	-ah-reverse-diagonal-border-width	Specifies the width of the reverse diagonal border.
axf:reverse-page	-ah-reverse-page	Outputs pages in reverse order.
axf:reverse-page-number		Places page numbers in reverse order.
axf:revision-bar-color	-ah-revision-bar-color	Specifies the color of the revision bar.
axf:revision-bar-offset	-ah-revision-bar-offset	Specifies the offset of the revision bar.
axf:revision-bar-position	-ah-revision-bar-position	Specifies the position of the revision bar.
axf:revision-bar-style	-ah-revision-bar-style	Specifies the style of the revision bar.
axf:revision-bar-width	-ah-revision-bar-width	Specifies the width of the revision bar.
7.6.3 right	[CSS2.1] right	

XSL	CSS	Description
axf:right-page-master-reference		Master name of the page master for the right page of a two-page spread.
7.5.2 role		
axf:ruby-align	[CSS3-Ruby] (-ah-)ruby-align	Specifies the alignment of ruby.
axf:ruby-color	-ah-ruby-color	Specifies the color of ruby text.
axf:ruby-condense	-ah-ruby-condense	Specifies the font condense when the ruby text is longer than its base.
axf:ruby-font-family	-ah-ruby-font-family	Specifies the font family of ruby text.
axf:ruby-font-size	-ah-ruby-font-size	Specifies the font size of ruby text.
axf:ruby-font-stretch	-ah-ruby-font-stretch	Specifies the font stretching of ruby text.
axf:ruby-font-style	-ah-ruby-font-style	Specifies the font style of ruby text.
axf:ruby-font-weight	-ah-ruby-font-weight	Specifies the font weight of ruby text.
axf:ruby-limit-overhang	-ah-ruby-limit-overhang	Specifies the limit of the amount that ruby overhangs the adjacent base character when ruby is longer than its own base character.
axf:ruby-limit-space	-ah-ruby-limit-space	Specifies the limit of the amount of spaces leading and following the ruby text when the ruby text is shorter than its base characters.

XSL	CSS	Description
axf:ruby-minimum-font-size	-ah-ruby-minimum-font-size	Specifies the minimum font size of ruby text.
axf:ruby-offset	-ah-ruby-offset	Specifies the spacing between the ruby text and its base characters.
axf:ruby-overhang	[CSS3-Ruby] (-ah-)ruby-overhang	Specifies how ruby overhangs the adjacent base character.
axf:ruby-position	[CSS3-Ruby] (-ah-)ruby-position	Specifies on which side of the base characters the ruby text appears.
axf:ruby-small-kana	-ah-ruby-small-kana	Specifies whether to allow using small kana for ruby text.
7.22.5 rule-style		
7.22.6 rule-thickness		
7.30.14 scale-option		
7.15.12 scaling	-ah-scaling	
axf:scope		Specifies the range of cells with which this table header cell is associated.
7.30.15 score-spaces	-ah-score-spaces	
7.10.3 script	-ah-script	

XSL	CSS	Description
axf:show-controls	-ah-show-controls	Specifies whether to show the player control bar for multimedia.
7.23.9 show-destination		
7.31.21 size	[CSS3-Page] (-ah-)size	
axf:soft-hyphen-treatment	-ah-soft-hyphen-treatment	Specifies how to treat SOFT HYPEN.
7.5.1 source-document		
7.11.6 space-after	-ah-margin-after	Specifies the margin of the after side.
7.11.5 space-before	-ah-margin-before	Specifies the margin of the before side.
7.12.5 space-end	-ah-margin-end	Specifies the margin of the end side.
7.12.6 space-start	-ah-margin-start	Specifies the margin of the start side.
7.21.4 span	[CSS3-Multicol] (-ah-)column-span	
7.30.16 src	[HTML] src	
7.11.7 start-indent		
7.28.15 starts-row		
	[CSS3-GCPM] (-ah-)string-set	
axf:suffix	[CSS3-CounterStyle] (-ah-)suffix	☞ <axf:counter-style>

XSL	CSS	Description
axf:suppress-duplicate-footnote	-ah-suppress-duplicate-footnote	Specifies whether to delete footnotes duplicated in the same page.
axf:suppress-duplicate-marker-contents		Specifies the removal of duplicate marker references.
axf:suppress-duplicate-page-number	-ah-suppress-duplicate-page-number	Specifies to delete the duplicated page numbers.
axf:suppress-folio-prefix		Invalidates the prefix of page numbers.
axf:suppress-folio-suffix		Invalidates the suffix of page numbers.
axf:suppress-if-first-on-page	-ah-suppress-if-first-on-page	Specifies whether to suppress the block at the beginning of a page or column.
axf:symbols	[CSS3-CounterStyle] (-ah-)symbols	☞ <axf:counter-style>
axf:system	[CSS3-CounterStyle] (-ah-)system	☞ <axf:counter-style>
axf:tab-align		Specifies the tab alignment at the tab stop position.
axf:tab-overlap-treatment		Specifies a behavior when tab alignment makes letters overlapped.
	[CSS3-Text] (-ah-)tab-size	
axf:tab-stops		Specifies the tab stop.
axf:tab-treatment		Specifies the method to treat the tab character (U+0009).

XSL	CSS	Description
axf:table-auto-layout-limit	-ah-table-auto-layout-limit	Specifies the number of rows of fo:table to read ahead to determine the width of column when table-layout="auto" is specified.
7.28.16 table-layout	[CSS2.1] table-layout	
7.28.17 table-omit-footer-at-break	-ah-table-omit-footer-at-break	
7.28.18 table-omit-header-at-break	-ah-table-omit-header-at-break	
axf:table-row-orphans	-ah-table-row-orphans	Specifies the number of table-rows that must remain at the bottom of the page (column).
axf:table-row-widows	-ah-table-row-widows	Specifies the number of table-rows that must remain at the top of the page (column).
axf:table-summary	-ah-table-summary	Describes the table summary.
7.16.9 text-align	[CSS2.1] text-align [CSS3-Text] (-ah-)text-align	
axf:text-align-first	-ah-text-align-first	Specifies the text alignment of the first line.
7.16.10 text-align-last	[CSS3-Text] (-ah-)text-align-last	
axf:text-align-string	-ah-text-align-string	Specifies the text alignment when text-align="<string>".
7.29.4 text-altitude		

XSL	CSS	Description
axf:text-autospace	[CSS3-Text] (-ah-)text-autospace	Specifies whether to add space surrounding ideographic glyphs or not.
axf:text-autospace-width	-ah-text-autospace-width	Specifies the width for axf:text-autospace.
	[CSS3-WritingModes] (-ah-)text-combine	
axf:text-combine-horizontal	[CSS3-WritingModes] (-ah-)text-combine-horizontal	Sets horizontal-in-vertical composition in vertical writing mode automatically.
7.17.4 text-decoration	[CSS2.1] text-decoration [CSS3-TextDecor] (-ah-)text-decoration	
	[CSS3-TextDecor] (-ah-)text-decoration-color	
	[CSS3-TextDecor] (-ah-)text-decoration-line	
	[CSS3-TextDecor] (-ah-)text-decoration-style	
7.29.5 text-depth		
	[CSS3-TextDecor] (-ah-)text-emphasis	
axf:text-emphasis-color	[CSS3-TextDecor] (-ah-)text-emphasis-color	Specifies the color of emphasis marks.
axf:text-emphasis-font-family	-ah-text-emphasis-font-family	Specifies the font family of emphasis marks.
axf:text-emphasis-font-size	-ah-text-emphasis-font-size	Specifies the font size of emphasis marks.
axf:text-emphasis-font-stretch	-ah-text-emphasis-font-stretch	Specifies the font stretching of emphasis marks.

XSL	CSS	Description
axf:text-emphasis-font-style	-ah-text-emphasis-font-style	Specifies whether emphasis marks are made Italic.
axf:text-emphasis-font-weight	-ah-text-emphasis-font-weight	Specifies the font weight of emphasis marks.
axf:text-emphasis-offset	-ah-text-emphasis-offset	Specifies the space between emphasis marks and the base characters.
axf:text-emphasis-position	[CSS3-TextDecor] (-ah-)text-emphasis-position	Specifies on which side of base characters emphasis marks are put.
axf:text-emphasis-skip	-ah-text-emphasis-skip	Specifies the character to which emphasis marks are not applied.
axf:text-emphasis-style	[CSS3-TextDecor] (-ah-)text-emphasis-style	Specifies the style of emphasis marks.
7.16.11 text-indent	[CSS2.1] text-indent	
axf:text-indent-if-first-on-page	-ah-text-indent-if-first-on-page	Specifies the text-indent of a block at the top of the page or the top of the column.
axf:text-justify	[CSS3-Text] (-ah-)text-justify	Specifies how to justify text.
axf:text-justify-trim	[CSS3-Text] (-ah-)text-justify-trim	Specifies the way to trim in text justification.
axf:text-kashida-space	-ah-text-kashida-space	Specifies the percentage of Kashida in Arabic justification.
axf:text-line-color	-ah-text-line-color	Specifies the color of underline, strikethrough, and overline.

XSL	CSS	Description
axf:text-line-style	-ah-text-line-style	Specifies the style of underline, strikethrough, and overline.
axf:text-line-width	-ah-text-line-width	Specifies the width of underline, strikethrough, and overline.
axf:text-orientation	[CSS3-WritingModes] (-ah-)text-orientation	Specifies the orientation of text in vertical writing mode.
axf:text-overflow	[CSS3-UI] (-ah-)text-overflow	Specifies the display method at the end of the content when overflowing in the inline progression direction.
axf:text-replace	[CSS3-GCPM] (-ah-)text-replace	Replaces the character strings.
7.17.5 text-shadow	[CSS3-TextDecor] (-ah-)text-shadow	Specifies the text shadow.
axf:text-stroke	-ah-text-stroke	Specifies the stroke of the character.
axf:text-stroke-color	-ah-text-stroke-color	Specifies the stroke color of the character.
axf:text-stroke-width	-ah-text-stroke-width	Specifies the stroke width of the character.
7.17.6 text-transform	[CSS2.1] text-transform [CSS3-Text] (-ah-)text-transform	
axf:text-underline-position	[CSS3-Text] (-ah-)text-underline-position	Specifies the position of underline.
7.6.2 top	[CSS2.1] top	

XSL	CSS	Description
axf:transform	[CSS3-Transforms] (-ah-)transform	Specifies the block transformation.
axf:transform-origin	[CSS3-Transforms] (-ah-)transform-origin	Specifies the origin of the block transformation.
7.29.6 unicode-bidi	[CSS2.1] unicode-bidi	
axf:value		☞ <axf:document-info>
7.31.22 vertical-align	[CSS2.1] vertical-align [CSS3-Line] (-ah-)vertical-align	
axf:vertical-underline-side	-ah-vertical-underline-side	Specifies on which side of the text to put underline in vertical writing-mode.
7.30.17 visibility	[CSS2.1] visibility	
7.31.23 white-space	[CSS2.1] white-space	
7.16.12 white-space-collapse	[CSS2.1] white-space	
7.16.8 white-space-treatment	[CSS2.1] white-space -ah-white-space-treatment	
7.20.7 widows	[CSS2.1] widows	
7.15.14 width	[CSS2.1] width	
axf:word-break	[CSS3-Text] (-ah-)word-break	Specifies whether to enable line breaking even inside a word.

XSL	CSS	Description
7.17.8 word-spacing	[CSS2.1] word-spacing	
axf:word-wrap	[CSS3-Text] (-ah-)word-wrap	Specifies whether to break word forcibly when line break cannot be performed.
7.16.13 wrap-option	[CSS2.1] white-space	
7.29.7 writing-mode	[CSS3-WritingModes] (-ah-)writing-mode	
7.30.18 z-index	[CSS2.1] z-index	