

Document Markup Language (DML) Specification 1.0

Abstract

This specification defines the Document Markup Language (DML), a markup language for books, articles, notes and other types of document. DML is normatively available as a [RELAX NG](#) (Appendix A, pg. 25) schema with additional [Schematron](#) (Appendix A, pg. 25) assertions.

Conventions

The keywords *must*, *must not*, *required*, *shall*, *shall not*, *should*, *should not*, *recommended*, *may*, and *optional*, when emphasized, are to be interpreted as described in [IETF RFC 2119](#) (Appendix A, pg. 25).

- A `monospaced` font is used for code, elements, attributes, tags and value literals.
- An *italic monospaced* font is used for variables.

Element:

(Review) When an element (node with type "`element`") is mentioned in the text with an associated [attribute](#) (pg. 1) it is always showed as a predicate. [Element EBNF definition](#) (pg. 3).

Notation for the `section` element

```
section
section[@role]
```

Attribute:

When an attribute (node with type "`attribute`") is mentioned in the text, it is always preceded by an at-sign (@) and it optionally has an associated value. [Attribute EBNF definition](#) (pg. 3).

Notation for the @role attribute

```
@role  
@role="chapter"
```

Value:

When a value is mentioned in the text, it is always preceded and followed by an quote ("). [Value EBNF definition](#) (pg. 3).

Notation for the "chapter" value

```
"chapter"
```

Tag:

When a tag is mentioned in the text, it is always preceded by a less-than symbol (<) and it is followed by a greater-than symbol (>). [Tag EBNF definition](#) (pg. 3).

When a tag is mentioned with some omitted attributes it has an ellipsis symbol (...) preceding greater-than symbol (>).

Notation for the start tag <section ...>

```
<section role="chapter" ...>
```

Any element or attribute can be modified by a quantifier modifier as follows:

?

Zero or one time.

+

One or more times.

*

Zero or more times.

(Review) Therefore, to indicate that an “status” attribute is optional the expression will be `@status?`. Or, if a “section” element is repeatable the expression will be `section+`.

For brevity, throughout this document, assume that the following namespace prefixes have been defined:

"dct"

<http://purl.org/dc/terms/>

"dml"

<http://purl.oclc.org/NET/dml/1.0/>

"rdf"

<http://www.w3.org/1999/02/22-rdf-syntax-ns#>

"xi"

<http://www.w3.org/2001/XInclude>

"xs"

<http://www.w3.org/2001/XMLSchema>

EBNF^[1] definitions

(Draft) TODO: define xpath syntax used in children, attribute and parent definitions.

- Element ::= Name ('[' Attribute '] ') *
- Attribute ::= '@' Name ('=' Value) ?
- Tag ::= '<' Name (S Name '=' Value) * S ? '...' ? '/' ? '>'
- Name ::= ([A-Za-z] + ':') ? [A-Za-z_] [A-Za-z0-9_-.] *
- Value ::= '"' [^<> "] + '"'
- S ::= (#x20 | #x9 | #xD | #xA) +

Status of this document

This is a *draft* and it may change at any time based on comments and on its development process.

^[1] [W3C notation](http://www.w3.org/TR/REC-xml/#sec-notation) (<http://www.w3.org/TR/REC-xml/#sec-notation>)

Table of Contents

1. Elements.....	4
1.1. The <code>abbr</code> element.....	5
1.2. The <code>cell</code> element.....	6
1.3. The <code>citation</code> element.....	6
1.4. The <code>dml</code> element.....	6
1.5. The <code>em</code> element.....	7
1.6. The <code>example</code> element.....	8
1.7. The <code>figure</code> element.....	9
1.8. The <code>group</code> element.....	9
1.9. The <code>item</code> element.....	10
1.10. The <code>list</code> element.....	10
1.11. The <code>metadata</code> element.....	12
1.12. The <code>note</code> element.....	13
1.13. The <code>object</code> element.....	14
1.14. The <code>p</code> element.....	15
1.15. The <code>quote</code> element.....	16
1.16. The <code>section</code> element.....	16
1.17. The <code>span</code> element.....	18
1.18. The <code>sub</code> element.....	18
1.19. The <code>summary</code> element.....	18
1.20. The <code>sup</code> element.....	19
1.21. The <code>table</code> element.....	19
1.22. The <code>title</code> element.....	20
2. Core attributes.....	20
2.1. The <code>@class</code> attribute.....	21
2.2. The <code>@dir</code> attribute.....	21
2.3. The <code>@href</code> attribute.....	21
2.4. The <code>@status</code> attribute.....	21
2.5. The <code>@xml:base</code> attribute.....	22
2.6. The <code>@xml:id</code> attribute.....	22
2.7. The <code>@xml:lang</code> attribute.....	22
3. Metadata attributes.....	22
3.1. The <code>@about</code> attribute.....	22
3.2. The <code>@content</code> attribute.....	23
3.3. The <code>@datatype</code> attribute.....	23
3.4. The <code>@property</code> attribute.....	23
3.5. The <code>@rel</code> attribute.....	23
3.6. The <code>@resource</code> attribute.....	23
3.7. The <code>@rev</code> attribute.....	23
3.8. The <code>@typeof</code> attribute.....	24
4. Flow.....	24
4.1. Block.....	24
4.2. Table.....	24
4.3. Inline.....	24
5. Namespace.....	24
6. Relationship with RDFa.....	24

7. Schema.....	24
Appendix A — Resources.....	25

1. Elements

(Draft) Add /listing for program listing? in cdm1?

1.1. The `abbr` element

The `abbr` element represents an abbreviation or acronym.

Flow

`Inline` (Section 4.3, pg. 24)

Children

`($inline[not(abbr)] | text())+`

Attributes

`($core.attrs* | $meta.attrs*)`

Parents

`($block | $inline[not(abbr)])`

The `@content` attribute (Section 3.2, pg. 23) *may* be used to provide an expansion of the abbreviation.

The `@about` attribute (Section 3.1, pg. 22) *may* be used to provide a resource which contains the expanded form.

`@content` and `@about` attributes are mutually exclusive.

Example 1.1-1: `abbr` element with inline expansion

```
<p>Example of <abbr content="Document Markup Language">DML</abbr>'s abbr
element.</p>
```

Example 1.1-2: `abbr` element with remote expansion

```
<p>Example of <abbr about="http://example.org/glossary#dml">DML</abbr>'s abbr
element.</p>
```

1.2. The `cell` element

The `cell` element represents a table data container.

Flow

[Table](#) (Section 4.2, pg. 24)

Children

```
( ( example | figure | list | note | p | quote )+ | ( $inline | text() )+ )
```

Attributes

```
( $core.attrs* | $meta.attrs* )
```

Parents

```
( group )
```

1.3. The `citation` element

The `citation` element represents a citation reference of a quotation block.

Flow

[Block](#) (Section 4.1, pg. 24)

Children

```
( $inline | text() )+
```

Attributes

```
( $core.attrs* | $meta.attrs* )
```

Parents

```
( quote )
```

1.4. The `dml` element

The `dml` element is the root element for a DML document.

Flow

[Block](#) (Section 4.1, pg. 24)

Children

```
( title, $block[not( title | citation )]+ )
(: this expression is more accurated but necessary? :)
(
  title,
  $block[not( title | citation | preceding-sibling::section )]+,
  section*
)
```

Attributes

```
( $score.attrs* )
```

Example 1.4-1: Simple DML document

```
<dml xmlns="http://purl.oclc.org/NET/dml/1.0/">
  <title>Simple DML document</title>
  <p>Lorem ipsum dolor sit amet...</p>
</example>
```

Example 1.4-2: DML document with metadata

```
<dml xmlns="http://purl.oclc.org/NET/dml/1.0/"
xmlns:dct="http://purl.org/dc/terms/">
  <title>DML document</title>
  <metadata about="">
    <list>
      <item property="dct:creator">Arnau Siches</item>
      <item property="dct:created">2009-01-02</item>
    </list>
  </metadata>
  <p>Lorem ipsum dolor sit amet...</p>
</example>
```

1.5. The **em** element

The **em** element represents an emphasized text.

Flow

[Inline](#) (Section 4.3, pg. 24)

Children

(*\$inline* | `text()`)+

Attributes

(*\$core.attrs** | *\$meta.attrs** | `@role?`)

Parents

(*\$block* | *\$inline*)

The `@role` attribute *may* be used to provide strong emphasized text with "`strong`" value.

Example 1.5-1: Usage of `em` element

```
<p>
  <em>Lorem ipsum</em> dolor sit amet, consectetur adipisicing elit, sed do <em
    role="strong">eiusmod tempor incididunt ut labore</em> et dolore magna aliqua.
</p>
```

1.6. The `example` element

The `example` element represents an example.

Flow

[Block](#) (Section 4.1, pg. 24)

Children

(`title?`, *\$block*[`not(example | citation)`]+)

Attributes

(*\$core.attrs** | *\$meta.attrs**)

Parents

(`dml` | `note` | `section`)

Example 1.6-1: Usage of `example` element

```
<example xml:id="example-identifier">
  <title>Title of the Lorem Ipsum example</title>
  <p>Lorem ipsum dolor sit amet...</p>
</example>
```

1.7. The `figure` element

The `figure` element is a figure container; it usually contains an illustration or something to be shown graphically.

Flow

[Block](#) (Section 4.1, pg. 24)

Children

```
( title?, $block[not( example | figure | citation | quote )]+ )
```

Attributes

```
( $core.attrs* | $meta.attrs* )
```

Parents

```
( dml | example | note | section )
```

Example 1.7-1: Usage of `figure` element

```
<figure xml:id="figure-identifier">
  <title>It shown an illustration through a figure element</title>
  <object src="path/to/illustration"/>
</figure>
```

1.8. The `group` element

The `group` element represents a generic table cell container.

Flow

[Table](#) (Section 4.2, pg. 24)

Children

```
( group+ | title+ | ( title?, cell+ ) )
```

Attributes

```
( $core.attrs* | $meta.attrs* | @role? )
```

Parents

```
( group | table )
```

The `@role` attribute *may* be used to provide a form to refine the `group` element meaning. Allowed values are:

`"header"`

A header table group. Table header *must* be the first child of a `table` element.

`"footer"`

A footer table group. Table footer *must* be child of a `table` element.

1.9. The `item` element

The `item` element represents a list item container.

Flow

[Block](#) (Section 4.1, pg. 24)

Children

```
(
  ( title*, $block[not( item | title | citation )]+ ) |
  ( $inline | text() )+
)
```

Attributes

```
( $core.attrs* | $meta.attrs* )
```

Parents

```
( list )
```

1.10. The `list` element

The `list` element represents a list of items.

Flow

[Block](#) (Section 4.1, pg. 24)

Children

```
( title?, item+ )
```

Attributes

```
( $core.attrs* | $meta.attrs* | @role? )
```

Parents

```
( dml | $block[$block[not( self::list )]] )
```

The `@role` attribute *may* be used to define an ordered list with "ordered" value.

Example 1.10-1: Simple list

```
<list>
  <item>sugar</item>
  <item>salt</item>
  <item>pepper</item>
</list>
```

Example 1.10-2: Ordered list

```
<list role="ordered">
  <item>first</item>
  <item>second</item>
  <item>third</item>
</list>
```

Example 1.10-3: List with title

```
<list>
  <title>List title</title>
  <item>first</item>
  <item>second</item>
  <item>third</item>
</list>
```

Example 1.10-4: Definition list

```
<list>
  <item>
    <title>Dweeb</title>
    <p>Young excitable person who may mature into a Nerd or Geek.</p>
  </item>
  <item>
    <title>Hacker</title>
    <p>A clever programmer.</p>
  </item>
  <item>
    <title>Nerd</title>
    <p>Technically bright but socially inept person.</p>
  </item>
</list>
```

Example 1.10-5: Definition list with multiple terms and definitions

```
<list>
  <item>
    <title>Center</title>
    <title>Centre</title>
    <list>
      <item>A point equidistant from all points on the surface of a
        sphere.</item>
      <item>In some field sports, the player who holds the middle position on
        the field, court, or forward line.</item>
    </list>
  </item>
  <item>
    <title>Color</title>
    <title>Colour</title>
    <p>The property possessed by an object of producing different sensations on
      the eye.</p>
  </item>
</list>
```

1.11. The `metadata` element

The `metadata` element represents a metadata container.

Flow

`Block` (Section 4.1, pg. 24)

Children

(`$block+` | `$inline+`)

Attributes

```
( $core.attrs* | $meta.attrs* )
```

Parents

```
( dml | $block | $inline )
```

(Draft) TODO: examples

1.12. The `note` element

The `note` element represents a generic document note or annotation. It *may* be used as a root element in *(Review) DML islands* in non-DML documents.

Flow

[Block](#) (Section 4.1, pg. 24)

Children

```
(  
  ( title?, $block[not( title | note | citation )]+ ) |  
  ( $inline | text() )+  
)
```

Attributes

```
( $core.attrs* | $meta.attrs* | @role? )
```

Parents

```
( dml | $block[$block[not( self::note )]] )
```

The `@role` attribute may be used to provide a form to refine the `note` element meaning. Allowed values are:

`"tip"`

A suggestion, tip or trick.

`"warning"`

An admonition note.

`"sidebar"`

A note that is isolated from the main narrative flow.

(Draft) `section[@role="aside"]` or `note[@role="aside"]` or `@role="sidebar" ...?`

"footnote"

A footnote. Footnotes in paged medias usually occur at the end of the page which cite it.

(Draft) TODO: examples

1.13. The `object` element

The `object` element represents a generic embedded media object like images, videos, audio and other types of multimedia files.

Flow

When its parent is an inline element or a block element that only allows inline elements its flow is `inline` (Section 4.3, pg. 24), otherwise its flow is `block` (Section 4.1, pg. 24).

Children

```
( $block* | ( $inline | text() )* )
```

Attributes

```
( $core.attrs* | $meta.attrs* | @src | @type? )
```

Parents

```
( dml | $block | $inline )
```

The `@src` attribute *must* be used to provide the URI (`xs:anyURI`) of the resource. It also specifies a *resource object* in RDF triple, as its described in [RDFa Recommendation](#) (Appendix A, pg. 25) of W3C.

The `@type` attribute *may* be used to provide the mime type of the resource.

The children of the `object` element *must* be used to provide an alternative content if the resource provided by `@src` fails to load.

The alternative content *must* be *inline* or *block* in accordance of the flow of its `object` parent.

Example 1.13-1: Usage of block flow `object` element.

```
<figure xml:id="fig-markup-trends">
  <title>Usage of markup language in %</title>
  <object src="markup-trends.svg" type="application/svg+xml">
    <list>
      <item>
        <title>HTML</title>
        <p>98%</p>
      </item>
      <item>
        <title>DocBook</title>
        <p>1%</p>
      </item>
      <item>
        <title>Other</title>
        <p>1%</p>
      </item>
    </list>
  </object>
</figure>
```

Example 1.13-2: Usage of inline flow `object` element.

```
<p>
  Press the <object src="accept-call-button-icon.svg"/><em>accept
  call</em></object> button to allow an incoming call.
</p>
```

1.14. The `p` element

The `p` element represents a generic block of text usually a paragraph.

Flow

[Block](#) (Section 4.1, pg. 24)

Children

(*\$inline* | `text()`)+

Attributes

(*\$core.attrs** | *\$meta.attrs**)

Parents

(`dml` | *\$block[\$block]*)

1.15. The `quote` element

The `quote` element represents a generic quotation container.

Flow

When its parent is an inline element or a block element that only allows inline elements its flow is `inline` (Section 4.3, pg. 24), otherwise its flow is `block` (Section 4.1, pg. 24).

Children

```
( $block[not( quote | citation )]+ citation | ( $inline | text() )+ )
```

Attributes

```
( $core.attrs* | $meta.attrs* | @citation? )
```

Parents

```
( dml | $block[not( quote | citation )] | $inline[not( quote )] )
```

The `@citation` attribute *must* be used to provide the URI (`xs:anyURI`) of the resource cited when the flow of `quote` element is *inline*, otherwise *must not* be used.

(Draft)

Example 1.15-1: Usage of block flow `quote` element.

```
<section>
  ( ... )
  <quote>
    <p>Lorem ipsum</p>
    <citation>??? <span href="http://some.resource">???</span> ??? </citation>
  </quote>
  ( ... )
</section>
```

(Draft)

Example 1.15-2: Usage of inline flow `quote` element.

```
<p>
  ??? <quote citation="http://some.resource">cite</quote> ???
</p>
```

1.16. The `section` element

The `section` element represents a generic document section.

Flow

[Block](#) (Section 4.1, pg. 24)

Children

```
( title, $block[not( title | citation )]+ )
```

Attributes

```
( $core.attrs* | $meta.attrs* | @role? )
```

Parents

```
( dml | note | object[parent::$block] | quote[parent::$block] | section )
```

The `@role` attribute *may* be used to provide a form to refine the `section` element meaning. Allowed values are:

`"abstract"`

A summary or statement of the contents of a document.

`"part"`

A part of a book. Parts usually group related chapters in a book.

`"chapter"`

(Review) A main division of a book.

`"appendix"`

An appendix in a document. Appendixes usually occur at the end of a document.

(Draft) `"header"`

(Draft) description ...?

(Draft) `"footer"`

(Draft) description ...?

(Draft) `"toc"`

(Draft) description ...?

`"license"`

(Draft) description ...?

(Draft) TODO: examples

1.17. The `span` element

The `span` element has no specific semantic. It is provided as a container of inline content.

Flow

[Inline](#) (Section 4.3, pg. 24)

Children

```
( $inline | text() )+
```

Attributes

```
( $core.attrs* | $meta.attrs* )
```

Parents

```
( $block | $inline )
```

1.18. The `sub` element

The `sub` element represents a subscript.

Flow

[Inline](#) (Section 4.3, pg. 24)

Children

```
( $inline | text() )+
```

Attributes

```
( $core.attrs* | $meta.attrs* )
```

Parents

```
( $block | $inline )
```

1.19. The `summary` element

The `summary` element is a tabular data summary.

Flow

[Table](#) (Section 4.2, pg. 24)

Children

```
( $inline | text() )+
```

Attributes

```
( $core.attrs* | $meta.attrs* )
```

Parents

```
( table )
```

1.20. The *sup* element

The *sup* element represents a superscript.

Flow

[Inline](#) (Section 4.3, pg. 24)

Children

```
( $inline | text() )+
```

Attributes

```
( $core.attrs* | $meta.attrs* )
```

Parents

```
( $block | $inline )
```

1.21. The *table* element

The *table* element represents a table container.

Flow

[Block](#) (Section 4.1, pg. 24)

Children

```
( title?, summary, group+ )
```

Attributes

```
( $core.attrs* | $meta.attrs* | @scope )
```

Parents

```
( dml | $block[$block] )
```

The `@scope` attribute *must* be used to provide the primary scope of groups. Allowed values are: "row" and "column".

(Draft) TODO: examples

1.22. The `title` element

The `title` element represents a header container.

Flow

[Block](#) (Section 4.1, pg. 24)

Children

```
( $inline | text() )+
```

Attributes

```
( $core.attrs* | $meta.attrs* )
```

Parents

```
( dml | $block[$block] )
```

(Draft) TODO: examples

2. Core attributes

```
$core.attrs ::= (  
  @class | @dir | @href | @status | @xml:base | @xml:id | @xml:lang  
)
```

These attributes *must not* be repeated in same element.

2.1. The `@class` attribute

The `@class` attribute provides additional user-specified classification for an element. Value type is `xs:NMTOKENS`.

Any number of elements *may* be assigned the same class name.

2.2. The `@dir` attribute

The `@dir` attribute specifies the direction of the element and its descendants. Allowed values are:

`"ltr"`

Left to right text.

`"rtl"`

Right to left text.

2.3. The `@href` attribute

The `@href` attribute specifies the location of a resource through an URI (`xs:anyURI`). It also specifies a *resource object* in RDF triple, as its described in [RDFa Recommendation](#) (Appendix A, pg. 25) of W3C.

2.4. The `@status` attribute

The `@status` attribute specifies the status of content in the element. Allowed values are:

`"added"`

Added text since last revision.

`"deleted"`

Deleted text since last revision.

`"draft"`

Text work in progress.

`"review"`

Text to evaluate or reevaluate but publishable.

user-value

Specific status defined by the users according they publishing process. This value *must* be an `xs:NMTOKEN`.

2.5. The `@xml:base` attribute

The `@xml:base` attribute specifies the base URI (`xs:anyURI`) of the element and its descendants. Its value *must* be interpreted according [xml:base W3C recomendation](#) (Appendix A, pg. 25).

2.6. The `@xml:id` attribute

The `@xml:id` attribute identifies the unique ID (`xs:ID`) value of the element. Its value *must* be interpreted according [xml:id W3C recomendation](#) (Appendix A, pg. 25).

2.7. The `@xml:lang` attribute

The `@xml:lang` attribute identifies the language of the element and its descendants. Its value *must* be interpreted according [XML 1.0](#) (Appendix A, pg. 25).

3. Metadata attributes

```
$meta.attrs ::= (  
  @about | @content | @datatype | @property | @rel | @resource | @rev | @typeof  
)
```

These attributes *must not* be repeated in same element.

3.1. The `@about` attribute

The `@about` attribute provides a *subject* for an RDF triple through an [URI or Safe CURIE](#) (Appendix A, pg. 25).

This attribute is part of [RDFa Recomendation](#) (Appendix A, pg. 25) of W3C.

3.2. The `@content` attribute

The `@content` attribute provides a machine-readable content for a literal in an RDF triple.

This attribute is part of [RDFa Recommendation](#) (Appendix A, pg. 25) of W3C.

3.3. The `@datatype` attribute

The `@datatype` attribute provides a datatype of a literal through a [CURIE](#) (Appendix A, pg. 25).

This attribute is part of [RDFa Recommendation](#) (Appendix A, pg. 25) of W3C.

3.4. The `@property` attribute

The `@property` attribute provides a predicate for an RDF triple through a whitespace separated list of [CURIEs](#) (Appendix A, pg. 25).

This attribute is part of [RDFa Recommendation](#) (Appendix A, pg. 25) of W3C.

3.5. The `@rel` attribute

The `@rel` attribute provides a predicate for an RDF triple through a whitespace separated list of [CURIEs](#) (Appendix A, pg. 25).

This attribute is part of [RDFa Recommendation](#) (Appendix A, pg. 25) of W3C.

3.6. The `@resource` attribute

The `@resource` attribute provides an object for an RDF triple through a [URIsafeCURIE](#) (Appendix A, pg. 25).

This attribute is part of [RDFa Recommendation](#) (Appendix A, pg. 25) of W3C.

3.7. The `@rev` attribute

The `@rev` attribute provides a reverse predicate for an RDF triple through a whitespace separated list of [CURIEs](#) (Appendix A, pg. 25).

This attribute is part of [RDFa Recommendation](#) (Appendix A, pg. 25) of W3C.

3.8. The @typeof attribute

The @typeof attribute provides the type(s) associated with a subject for an RDF triple through a whitespace separated list of [CURIEs](#) (Appendix A, pg. 25).

This attribute is part of [RDFa Recommendation](#) (Appendix A, pg. 25) of W3C.

(Draft)

4. Flow

4.1. Block

4.2. Table

<http://www.w3.org/TR/CSS21/tables.html>

4.3. Inline

5. Namespace

The DML namespace has de URI "<http://purl.oclc.org/NET/dml/1.0/>". Usually it is used associated with "dml" prefix.

(Draft)

6. Relationship with RDFa

...?

(Draft)

7. Schema

RELAX NG and Schematron references

Appendix A — Resources

RELAX NG

- ISO/IEC 19757-2:2008: [Information technology — Document Schema Definition Language \(DSDL\) — Part 2: Regular-grammar-based validation — RELAX NG](http://standards.iso.org/ittf/PubliclyAvailableStandards/c052348_ISO_IEC_19757-2_2008(E).zip) ([http://standards.iso.org/ittf/PubliclyAvailableStandards/c052348_ISO_IEC_19757-2_2008\(E\).zip](http://standards.iso.org/ittf/PubliclyAvailableStandards/c052348_ISO_IEC_19757-2_2008(E).zip)). ISO/IEC. 2008.
- [RELAX NG Home page](http://www.relaxng.org/) (<http://www.relaxng.org/>)

Schematron

- ISO/IEC 19757-3:2006: [Information technology — Document Schema Definition Language \(DSDL\) — Part 3: Rule-based validation — Schematron](http://standards.iso.org/ittf/PubliclyAvailableStandards/c040833_ISO_IEC_19757-3_2006(E).zip) ([http://standards.iso.org/ittf/PubliclyAvailableStandards/c040833_ISO_IEC_19757-3_2006\(E\).zip](http://standards.iso.org/ittf/PubliclyAvailableStandards/c040833_ISO_IEC_19757-3_2006(E).zip)). ISO/IEC. 2006.
- [Schematron Home page](http://www.schematron.com) (<http://www.schematron.com>)

IETF (Internet Engineering Task Force)

- [RFC 2119: Key words for use in RFCs to Indicate Requirement Levels](http://www.apps.ietf.org/rfc/rfc2119.html) (<http://www.apps.ietf.org/rfc/rfc2119.html>). S. Bradner. 1997.
- [RFC 4646: Tags for the Identification of Languages](http://www.apps.ietf.org/rfc/rfc4646.html) (<http://www.apps.ietf.org/rfc/rfc4646.html>). A. Phillips, Ed., M. Davis. 2006.

xml namespace

- [xml:id Version 1.0](http://www.w3.org/TR/2005/REC-xml-id-20050909/) (<http://www.w3.org/TR/2005/REC-xml-id-20050909/>). N. Walsh, D. Veillard, J. Marsh. 2005.
- [Extensible Markup Language \(XML\) 1.0 \(Fifth Edition\), 2.12 Language Identification](http://www.w3.org/TR/REC-xml/#sec-lang-tag) (<http://www.w3.org/TR/REC-xml/#sec-lang-tag>). T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, F. Yergeau. 2008.
- [XML Base](http://www.w3.org/TR/2001/REC-xmlbase-20010627/) (<http://www.w3.org/TR/2001/REC-xmlbase-20010627/>). J. Marsh. 2001.

RDFa

- [RDFa in XHTML: Syntax and Processing](http://www.w3.org/TR/2008/REC-rdfa-syntax-20081014/) (<http://www.w3.org/TR/2008/REC-rdfa-syntax-20081014/>). B. Adida, M. Birbeck, S. McCarron, S. Pemberton. 2008.
- [RDFa Primer](http://www.w3.org/TR/2008/NOTE-xhtml-rdfa-primer-20081014/) (<http://www.w3.org/TR/2008/NOTE-xhtml-rdfa-primer-20081014/>). B. Adida, M. Birbeck. 2008.
- [RDFa in XHTML: Syntax and Processing, CURIE definition](http://www.w3.org/TR/rdfa-syntax/#dt_curie) (http://www.w3.org/TR/rdfa-syntax/#dt_curie). B. Adida, M. Birbeck, S. McCarron, S. Pemberton. 2008.
- [RDFa in XHTML: Syntax and Processing, URIorSafeCURIE definition](http://www.w3.org/TR/rdfa-syntax/#dt_urisafecurie) (http://www.w3.org/TR/rdfa-syntax/#dt_urisafecurie). B. Adida, M. Birbeck, S. McCarron, S. Pemberton. 2008.

Dublin Core Metadata Initiative

- [Dublin Core Metadata Initiative Home page](http://dublincore.org/). (<http://dublincore.org/>)
- [Expressing Dublin Core metadata using HTML/XHTML meta and link elements](http://dublincore.org/documents/2008/08/04/dc-html/) (<http://dublincore.org/documents/2008/08/04/dc-html/>). P. Jhonston, A. Powell. 2008.