CDATA

From Wikipedia, the free encyclopedia

The term **CDATA**, meaning **character data**, is used for distinct, but related, purposes in the markup languages SGML and XML. The term indicates that a certain portion of the document is general *character data*, rather than non-character data or character data with a more specific, limited structure.

Contents

- 1 CDATA sections in XML
  - 1.1 Syntax and interpretation
  - 1.2 Uses of CDATA sections
    - 1.2.1 Nesting
    - 1.2.2 Issues with encoding
- 2 Use of CDATA in program output
- 3 CDATA in DTDs
  - 3.1 CDATA-type attribute value
  - 3.2 CDATA-type entity
  - 3.3 CDATA-type element content
- 4 See also
- 5 External links

CDATA sections in XML

In an XML document or external parsed entity, a **CDATA section** is a section of element content that is marked for the parser to interpret purely as textual data, not as markup. A CDATA section is merely an alternative syntax for expressing character data; there is no semantic difference between character data that manifests as a CDATA section and character data that manifests as in the usual syntax in which, for example, "`<`" and "`&`" would be represented by "`&lt;`" and "`&amp;`", respectively.

Syntax and interpretation

A CDATA section starts with the following sequence:

```
<![CDATA[
```

and ends with the first occurrence of the sequence:

```
]]>
```

All characters enclosed between these two sequences are interpreted as characters, not markup or entity references

```
<sender>John Smith</sender>
```

the start and end "sender" tags are interpreted as markup. However, if written like this:

```
<![CDATA[<sender>John Smith</sender>]]>
```

then the code is interpreted the same as if it had been written like this:

```
&lt;sender&gt;John Smith&lt;/sender&gt;
```

That is, the "sender" tags will have exactly the same status as the "John Smith" — they will be treated as text.

Similarly, if the numeric character reference `&#240;` appears in element content, it will be interpreted as the single Unicode character 00F0 (small letter eth). But if the same appears in a CDATA section, it will be parsed as six characters: ampersand, hash mark, digit 2, digit 4, digit 0, semicolon.

Uses of CDATA sections

New authors of XML documents often misunderstand the purpose of a CDATA section, mistakenly believing that its purpose is to "protect" data from being treated as ordinary character data during processing. Some APIs for working with XML documents do offer options for independent access to CDATA sections, but such options exist above and beyond the normal requirements of XML processing systems, and still do not change the implicit meaning of the data. Character data is character data, regardless of whether it is expressed via a CDATA section or ordinary markup. CDATA sections are useful for writing XML code as text data within an XML document. For example, if one wishes to typeset a book with XSL explaining the use of an XML application, the XML markup to appear in the book itself will be written in the source file in a CDATA section.

Nesting

A CDATA section cannot contain the string "`]]>`" and therefore it is not possible for a CDATA section to contain nested CDATA sections. The preferred approach to using CDATA sections for encoding text that contains the triad "`]]>`" is to use multiple CDATA sections by splitting each occurrence of the triad just before the "`>`". For example, to encode "`]]>`" one would write:

```
<![CDATA[]]]]><![CDATA[>]]>
```

This means that to encode "`]]>`" in the middle of a CDATA section, replace all occurrences of "`]]>`" with the following:

```
]]]]><![CDATA[>
```

This effectively stops and restarts the CDATA section.

Issues with encoding

In text data, any Unicode character not available in the encoding declared in the `<?xml ...?>` header can be represented using a `&#nnn;` numerical character reference. But the text within a CDATA section is strictly limited to the characters available in the encoding.

Because of this, using a CDATA section programmatically to quote data that could potentially contain '`&`' or '`<`' characters can cause problems when the data happens to contain characters that cannot be represented in the encoding. Depending on the implementation of the encoder, these characters can get lost, can get converted to the characters of the `&#nnn;` character reference, or can cause the encoding to fail. But they will not be maintained.

Another issue is that an XML document can be transcoded from one encoding to another during transport. When the XML document is converted to a more limited character set, such as ASCII, characters that cannot be represented any more are converted to `&#nnn;` character references for a lossless conversion. But within a CDATA section, these characters can not be represented at all, and have to be removed or converted to some equivalent, altering the content of the CDATA section.

That is why CDATA sections should be used only for XML documents that are keyed in manually, where they contain code or XML as data. Enclosing these in a CDATA section greatly improves readability. But when XML is generated programmatically, CDATA sections should be avoided.

Use of CDATA in program output

CDATA sections in XHTML documents are liable to be parsed differently by web browsers if they render the document as HTML, since HTML parsers do not recognise the CDATA start and end markers, nor do they recognise HTML entity references such as &lt; within <script> tags. This can cause rendering problems in web browsers and can lead to cross-site scripting vulnerabilities if used to display data from untrusted sources, since the two kinds of parser will disagree on where the CDATA section ends.

Since it is useful to be able to use less-than signs (<) and ampersands (&) in web page scripts, and to a lesser extent styles, without having to remember to escape them, it is common to use CDATA markers around the text of inline <script> and <style> elements in XHTML documents. But so that the document can also be parsed by HTML parsers, which do not recognise the CDATA markers, the CDATA markers are usually commented-out, as in this JavaScript example:

```
<script type="text/javascript">
//<![CDATA[
document.write("<");
//]]>
</script>
```

or this CSS example:

```
<style type="text/css">
/*<![CDATA[*/
body { background-image: url("marble.png?width=300&height=300") }
/*]]>*/
</style>
```

This technique is only necessary when using inline scripts and stylesheets, and is language-specific. CSS stylesheets, for example, only support the second style of commenting-out (/* ... */), but CSS also has less need for the < and & characters than JavaScript and so less need for explicit CDATA markers.

CDATA in DTDs

CDATA-type attribute value

In Document Type Definition (DTD) files for SGML and XML, an attribute value may be designated as being of type CDATA: arbitrary character data. Within a CDATA-type attribute, character and entity reference markup is allowed and will be processed when the document is read.

For example, if an XML DTD contains

```
<!ATTLIST foo a CDATA #IMPLIED>
```

it means that elements named foo may optionally have an attribute named "*a*" which is of type CDATA. In an XML document that is valid according to this DTD, an element like this might appear:

```
<foo a="1 &amp; 2 are &lt; &#51; &#x0A;" />
```

and an XML parser would interpret the "*a*" attribute's value as being the character data "*1 & 2 are < 3*".

CDATA-type entity

An SGML or XML DTD may also include entity declarations in which the token CDATA is used to indicate that entity consists of character data. The character data may appear within the declaration itself or may be available externally, referenced by a URI. In either case, character reference and parameter entity reference markup is allowed in the entity, and will be processed as such when it is read.

<DISPLAY_NAME Attribute="Y"><![CDATA[PFTEST0__COUNTER_6__:4:199:, PFTEST0__COUNTER_7__:4:199:]]></DISPLAY_NAME>

CDATA-type element content

An SGML DTD may declare an element's content as being of type //<![CDATA[. Within a CDATA-type element, no markup will be processed. It is similar to a CDATA section in XML, but has no special boundary markup, as it applies to the entire element. <DISPLAY_NAME Attribute="Y"><![CDATA[PFTEST0__COUNTER_6__:4:199:, PFTEST0__COUNTER_7__:4:199:]]></DISPLAY_NAME>

See also

- PCDATA

External links

- CDATA Confusion (http://www.flightlab.com/~joe/sgml/cdata.html)
- Character Data and Markup (in XML) (http://www.w3.org/TR/REC-xml/#syntax)

Retrieved from "https://en.wikipedia.org/w/index.php?title=CDATA&oldid=732786062"

Categories: XML