HTML: The Markup Language

« documents elements by function »

4. HTML syntax

This section describes the HTML syntax in detail. In places, it also notes differences between the the HTML syntax and the XML syntax, but it does not describe the XML syntax in detail (the XML syntax is instead defined by rules in the [XML] specification and in the [Namespaces in XML] specification).

This section is divided into the following parts:

4.1. The doctype

A **doctype** (sometimes capitalized as “DOCTYPE”) is an special instruction which, for legacy reasons that have to do with processing modes in browsers, is a required part of any document in the HTML syntax; it must match the characteristics of one of the following three formats:

- normal doctype
- deprecated doctype
- legacy-tool-compatible doctype

A **normal doctype** consists of the following parts, in exactly the following order:

1. Any case-insensitive match for the string "`<!DOCTYPE`".
2. One or more space characters.
3. Any case-insensitive match for the string "`HTML`".
4. Optionally, one or more space characters.
5. A "`>`" character.

The following is an example of a conformant normal doctype.

```
<!DOCTYPE html>
```

A **deprecated doctype** consists of the following parts, in exactly the following order:

1. Any case-insensitive match for the string "`<!DOCTYPE`".
2. One or more space characters.
3. Any case-insensitive match for the string "`HTML`".
4. One or more space characters.
5. Any case-insensitive match for the string "`PUBLIC`".
6. One or more space characters.
7. A *quote mark (public ID)*, consisting of either a "`"`" character or a "`'`" character.
8. A permitted public ID
9. A matching *quote mark (public ID)*, identical to the *quote mark (public ID)* used earlier (either a "`"`" character or a "`'`" character).
10. Conditionally, depending on whether it is part of a permitted-public-ID-system-ID-combination, the following parts, in exactly the following order:
    1. One or more space characters.
    2. A *quote mark (system ID)*, consisting of either a "`"`" character or a "`'`" character.
    3. A permitted system ID
    4. A matching *quote mark (system ID)*, identical to the *quote mark (system ID)* used earlier (either a "`"`"

jump

character or a "'" character).
11. Optionally, one or more space characters.
12. A ">" character.

A **permitted-public-ID-system-ID-combination** is any combination of a **public ID** (the first quoted string in the doctype) and **system ID** (the second quoted string, if any, in the doctype) such that the combination corresponds to one of the six deprecated doctypes in the following list of deprecated doctypes:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0//EN">
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0//EN" "http://www.w3.org/TR/REC-html40/strict.dtd">
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN">
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

A **legacy-tool-compatible doctype** consists of the following parts, in exactly the following order:

1. Any case-insensitive match for the string "`<!DOCTYPE`".
2. One or more space characters.
3. Any case-insensitive match for the string "`HTML`".
4. One or more space characters.
5. Any case-insensitive match for the string "`SYSTEM`".
6. One or more space characters
7. A *quote mark*, consisting of either a """ character or a "'" character.
8. The literal string "`about:legacy-compat`".
9. A matching *quote mark*, identical to the *quote mark* used earlier (either a """ character or a "'" character).
10. Optionally, one or more space characters.
11. A ">" character.

The following is examples of a conformant legacy-tool-compatible doctype.

```
<!doctype HTML system "about:legacy-compat">
```

4.2. Character encoding declaration

A **character encoding declaration** is a mechanism for specifying the character encoding used to store or transmit a document.

The following restrictions apply to character encoding declarations:

- The character encoding name given must be the name of the character encoding used to serialize the file.
- The value must be a valid character encoding name, and must be the preferred name for that encoding, as specified in the IANA [Character Sets] registry.
- The character encoding declaration must be serialized without the use of character references or character escapes of any kind.
- The element containing the character encoding declaration must be serialized completely within the first 512 bytes of the document.

If the document does not start with a U+FEFF BYTE ORDER MARK (BOM) character, and if its encoding is not explicitly given by a `Content-Type` HTTP header, then the character encoding used must be an ASCII-compatible character encoding, and, in addition, if that encoding isn't US-ASCII itself, then the encoding must be specified using a meta element with a charset attribute or a meta element in the encoding declaration state.

If the document contains a meta element with a charset attribute or a meta element in the encoding declaration state, then the character encoding used must be an ASCII-compatible character encoding.

An **ASCII-compatible character encoding** is one that is a superset of US-ASCII (specifically, ANSI_X3.4-1968) for bytes in the set 0x09, 0x0A, 0x0C, 0x0D, 0x20 - 0x22, 0x26, 0x27, 0x2C - 0x3F, 0x41 - 0x5A, and 0x61 - 0x7A.

Documents should not use UTF-32, JIS_C6226-1983, JIS_X0212-1990, HZ-GB-2312, JOHAB (Windows code page 1361), encodings based on ISO-2022, or encodings based on EBCDIC.

Documents must not use CESU-8, UTF-7, BOCU-1, or SCSU encodings.

jump

In a document in the XML syntax, the **XML declaration**, as defined in the XML specification [XML] should be used to provide character-encoding information, if necessary.

4.3. Elements

An element's **content model** defines the element's structure: What contents (if any) the element can contain, as well as what attributes (if any) the element can have. The HTML elements section of this reference describes the content models for all of elements that are part of the HTML language. An element must not contain contents or attributes that are not part of its content model.

The **contents** of an element are any elements, character data, and comments that it contains. Attributes and their values are not considered to be the "contents" of an element.

The **text content** of an element is the value of the `textContent` IDL attribute of the element, as defined in [DOM4].

A **void element** is an element whose content model never allows it to have contents under any circumstances. Void elements can have attributes.

The following is a complete list of the **void elements in HTML**:

- area, base, br, col, command, embed, hr, img, input, keygen, link, meta, param, source, track, wbr

The following list describes syntax rules for the the HTML syntax. Rules for the the XML syntax are defined in the XML specification [XML].

- **Tags** are used to delimit the start and end of elements in markup. Elements have a start tag to indicate where they begin. Non-void elements have an end tag to indicate where they end.
- **Tag names** are used within element start tags and end tags to give the element's name. HTML elements all have names that only use characters in the range `0–9`, `a–z`, and `A–Z`.
- **Start tags** consist of the following parts, in exactly the following order:
  1. A "`<`" character.
  2. The element's tag name.
  3. Optionally, one or more attributes, each of which must be preceded by one or more space characters.
  4. Optionally, one or more space characters.
  5. Optionally, a "`/`" character, which may be present only if the element is a void element.
  6. A "`>`" character.
- **End tags** consist of the following parts, in exactly the following order:
  1. A "`<`" character.
  2. A "`/`" character
  3. The element's tag name.
  4. Optionally, one or more space characters.
  5. A "`>`" character.
- Void elements only have a start tag; end tags must not be specified for void elements.
- The start and end tags of certain elements can be **omitted**. The subsection for each element in the HTML elements section of this reference provides information about which tags (if any) can be omitted for that particular element.
- A non-void element must have an end tag, unless the subsection for that element in the HTML elements section of this reference indicates that its end tag can be omitted.
- The contents of an element must be placed between just after its start tag (which might be implied, in certain cases) and just before its end tag (which might be implied in certain cases).

4.3.1. Misnested tags

If an element has both a start tag and an end tag, its end tag must be contained within the contents of the same element in which its start tag is contained. An end tag that is not contained within the same contents as its start tag is said to be a **misnested tag**.

In the following example, the "`</i>`" end tag is a misnested tag, because it is not contained within the contents of the b element that contains its corresponding "`<i>`" start tag.

```
<b>foo <i>bar</b> baz</i>
```

jump

4.4. Attributes

Attributes for an element are expressed inside the element's start tag. Attributes have a name and a value.

There must never be two or more attributes on the same start tag whose names are a case-insensitive match for each other.

The following list describes syntax rules for attributes in documents in the HTML syntax. Syntax rules for attributes in documents in the XML syntax. are defined in the XML specification [XML].

- **Attribute names** must consist of one or more characters other than the space characters, U+0000 NULL, `"`"`", "`'`", "`>`", "`/`", "`=`", the control characters, and any characters that are not defined by Unicode.
- **XML-compatible** attribute names are those that match the `Name` production defined in the XML specification [XML] and that contain no "`:`" characters, and whose first three characters are not a case-insensitive match for the string "`xml`".
- **Attribute values** can contain text and character references, with additional restrictions depending on whether they are unquoted attribute values, single-quoted attribute values, or double-quoted attribute values. Also, the HTML elements section of this reference describes further restrictions on the allowed values of particular attributes, and attributes must have values that conform to those restrictions.

In the the HTML syntax, attributes can be specified in four different ways:

1. empty attribute syntax
2. unquoted attribute-value syntax
3. single-quoted attribute-value syntax
4. double-quoted attribute-value syntax

Empty attribute syntax

Certain attributes may be specified by providing just the attribute name, with no value.

In the following example, the `disabled` attribute is given with the empty attribute syntax:

```
<input disabled>
```

Note that empty attribute syntax is exactly equivalent to specifying the empty string as the value for the attribute, as in the following example.

```
<input disabled="">
```

Unquoted attribute-value syntax

An **unquoted attribute value** is specified by providing the following parts in exactly the following order:

1. an attribute name
2. zero or more space characters
3. a single "`=`" character
4. zero or more space characters
5. an attribute value

In addition to the general requirements for attribute values, an unquoted attribute value has the following restrictions:

- must not contain any literal space characters
- must not contain any "`"`", "`'`", "`=`", "`>`", "`<`", or "`` ` ``", characters
- must not be the empty string

In the following example, the `value` attribute is given with the unquoted attribute value syntax:

```
<input value=yes>
```

If the value of an attribute using the unquoted attribute syntax is followed by a "`/`" character, then there must be at least one space character after the value and before the "`/`" character.

jump

Single-quoted attribute-value syntax

A **single-quoted attribute value** is specified by providing the following parts in exactly the following order:

1. an attribute name
2. zero or more space characters
3. a "=" character
4. zero or more space characters
5. a single "'" character
6. an attribute value
7. a "'" character.

In addition to the general requirements for attribute values, a single-quoted attribute value has the following restriction:

- must not contain any literal "'" characters

In the following example, the `type` attribute is given with the single-quoted attribute value syntax:

```
<input type='checkbox'>
```

Double-quoted attribute-value syntax

A **double-quoted attribute value** is specified by providing the following parts in exactly the following order:

1. an attribute name
2. zero or more space characters
3. a single "=" character
4. zero or more space characters
5. a single """ character
6. an attribute value
7. a """ character

In addition to the general requirements for attribute values, a double-quoted attribute value has the following restriction:

- must not contain any literal """ characters

In the following example, the `title` attribute is given with the double-quoted attribute value syntax:

```
<code title="U+003C LESS-THAN SIGN">&lt;</code>
```

4.5. Text and character data

Text in element contents (including in comments) and attribute values must consist of Unicode characters, with the following restrictions:

- must not contain U+0000 characters
- must not contain permanently undefined Unicode characters
- must not contain control characters other than space characters

Character data contains text, in some cases in combination with character references, along with certain additional restrictions. There are three types of character data that can occur in documents:

1. normal character data
2. replaceable character data
3. non-replaceable character data

Normal character data

Certain elements contain normal character data. Normal character data can contain the

jump

following:

- text
- character references

Normal character data has the following restrictions:

- must not contain any "`<`" characters

Replaceable character data

In documents in the HTML syntax, the title and textarea elements can contain replaceable character data. Replaceable character data can contain the following:

- text, optionally including "`<`" characters
- character references

Replaceable character data has the following restrictions:

- must not contain any ambiguous ampersands
- must not contain any occurrences of the string "`</`" followed by characters that are a case-insensitive match for the tag name of the element containing the replaceable character data (for example, "`</title`" or "`</textarea`"), followed by a space character, "`>`", or "`/`".

Note: *Replaceable character data, as described in this reference, is a feature of the HTML syntax that is not available in the XML syntax. Documents in the XML syntax must not contain replaceable character data as described in this reference; instead they must conform to all syntax constraints described in the XML specification [XML].*

Non-replaceable character data

In documents in the HTML syntax, the script, and style elements can contain non-replaceable character data. Non-replaceable character data can contain the following:

- text, optionally including "`<`" characters
- ambiguous ampersands

Non-replaceable character data has the following restrictions:

- must not contain character references
- must not contain any occurrences of the string "`</`", followed by characters that are a case-insensitive match for the tag name of the element containing the replaceable character data (for example, "`</script`" or "`</style`", followed by a space character, "`>`", or "`/`".

Note: *Non-replaceable character data, as described in this reference, is a feature of the HTML syntax that is not available in the XML syntax. Documents in the XML syntax must not contain non-replaceable character data as described in this reference; instead they must conform to all syntax constraints defined in the XML specification [XML].*

4.6. Character references

Character references are a form of markup for representing single individual characters. There are three types of character references:

- named character references
- decimal numeric character references
- hexadecimal numeric character references

Named character reference

Named character references consist of the following parts in exactly the following order:

1. An "`&`" character.

jump

2. One of the entity names listed in the “Named character references” section of the HTML5 specification [HTML5], using the same case.
3. A ";" character.

***Note:** For further information about named character references, see [XML Entities].*

The following is an example of a named character reference for the character "†" (U+2020 DAGGER).

```
&dagger;
```

Decimal numeric character reference

Decimal numerical character references consist of the following parts, in exactly the following order.

1. An "&" character.
2. A "#" character.
3. One or more digits in the range 0–9, representing a base-ten integer that itself is a Unicode code point that is not U+0000, U+000D, in the range U+0080–U+009F, or in the range 0xD8000–0xDFFF (surrogates).
4. A ";" character.

The following is an example of a decimal numeric character reference for the character "†" (U+2020 DAGGER).

```
&#8224;
```

Hexadecimal numeric character reference

Hexadecimal numeric character references consist of the following parts, in exactly the following order.

1. An "&" character.
2. A "#" character.
3. Either a "x" character or a "X" character.
4. One or more digits in the range 0–9, a–f, and A–F, representing a base-sixteen integer that itself is a Unicode code point that is not U+0000, U+000D, in the range U+0080–U+009F, or in the range 0xD800–0xDFFF (surrogates).
5. A ";" character.

The following is an example of a hexadecimal numeric character reference for the character "†" (U+2020 DAGGER).

```
&#x2020;
```

***Note:** Character references are not themselves text, and no part of a character reference is text.*

An **ambiguous ampersand** is an "&" character that is followed by some text other than a space character, a "<", character, or another "&" character.

4.7. Comments

Comments consist of the following parts, in exactly the following order:

1. the **comment start delimiter** "<!--"
2. text
3. the **comment end delimiter** "-->"

The text part of comments has the following restrictions:

- must not start with a ">" character

jump

- must not start with the string "->"
- must not contain the string "--"
- must not end with a "-" character

The following is an example of a comment.

```
<!-- main content starts here -->
```

4.8. SVG and MathML elements in HTML documents

SVG and MathML elements are elements from the SVG and MathML namespaces. SVG and MathML elements can be used both in documents in the HTML syntax and in documents in the XML syntax. Syntax rules for SVG and MathML elements in documents in the XML syntax are defined in the XML specification [XML]. The following list describes additional syntax rules that specifically apply to SVG and MathML elements in documents in the HTML syntax.

- SVG and MathML elements whose start tags have a single "/" character before the closing ">" character are said to be **marked as self-closing**.
- SVG and MathML elements must either have a start tag and an end tag, or a start tag that is marked as self-closing, in which case they must not have an end tag.
- SVG and MathML elements whose start tag is marked as self-closing, can’t have any contents.
- The contents of an SVG or MathML element whose start tag is not marked as self-closing are any elements, character data, comments, and CDATA sections that it contains, with the restriction that any character data it contains must be normal character data.

4.9. CDATA sections in SVG and MathML contents

CDATA sections in SVG and MathML contents in documents in the HTML syntax consist of the following parts, in exactly the following order:

1. the **CDATA start delimiter** "<![CDATA["
2. text, with the additional restriction that the text must not contain the string "]]>“
3. the **CDATA end delimiter** "]]>"

CDATA sections are allowed only in the contents of elements from the SVG and MathML namespaces.

The following shows an example of a CDATA section.

```
<annotation encoding="text/latex">
<![CDATA[\documentclass{article}
\usepackage{amsmath}
\begin{document}
The absolute value of $x$:
\[
\left|x\right|= \begin{cases}-x& \text{if $x<0$}\\
                              x& \text{otherwise}\end{cases}
\]
\end{document}]]>
</annotation>
```

« documents elements by function »

jump