HTML5 Reference

The Syntax, Vocabulary and APIs of HTML5

W3C Editor's Draft 9 August 2010

This version:
http://www.w3.org/TR/2010/ED-html5-author-20100809
Latest version:
http://www.w3.org/TR/
Previous version:
No previous versions.
Editors:
Lachlan Hunt (Opera Software ASA) lachlan.hunt@lachy.id.au

Abstract

This document explains the syntax, vocabulary and the available APIs for HTML5 documents, focussing on simplicity and practical applications for beginners while also providing in depth information for more advanced web developers. This document is complimentary to the HTML5 Guide.

Status of this document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current W3C publications and the latest revision of this technical report can be found in the W3C technical reports index at http://www.w3.org/TR/.

This document is an Editors Draft of the "HTML5 Reference" produced by the HTML Working Group, part of the HTML Activity. The working group is working on HTML5 (see the HTML5 Editor's draft). The appropriate forum for comments on this document is public-html-comments@w3.org (public archive) or public-html@w3.org (public archive).

Publication as a Working Group Note does not imply endorsement by the W3C Membership. This is a draft document and may be updated, replaced or obsoleted by other documents at any time. It is inappropriate to cite this document as other than work in progress.

This document was produced by a group operating under the 5 February 2004 W3C Patent Policy. W3C maintains a public list of any patent disclosures made in connection with the deliverables of the group; that page also includes instructions for disclosing a patent. An individual who has actual knowledge of a patent which the individual believes contains Essential Claim(s) must disclose the information in accordance with section 6 of the W3C Patent Policy.

Table of contents

1 Introduction

This document serves as a reference for the HTML syntax, vocabulary and its associated DOM APIs and is intended for web site and application developers, publishers, tutorial writers and teachers and their students. That is, people who write documents using HTML, or who teach others to do so.

This guide is structured into three major sections. The first provides a set of basic templates for authors

to get started with.

The second section provides an in depth look at the syntax of HTML and XHTML documents. This will investigate both the similarities and differences between the two alternatives and provides guidance on choosing which to use for your own projects, depending on your needs. Additionally, this will also provide details about creating polyglot documents—that is, documents that conform to both HTML and XHTML simultaneiously—including issues related to ensuring stylesheets and scripts work correctly under both conditions.

The third and final section provides a reference for the HTML vocabulary. Each element is described, providing details about its its meaning, allowed attributes, content models and DOM APIs. Each is accompanied by clear examples illustrating how the element is designed to be used for a range of different use cases.

2 Basic Templates

The following basic templates are designed to be used as starting points for the documents you create. Several alternatives are provided, allowing individuals to choose the template that is most appropriate for their needs.

Simple: HTML:

The most basic template recommended for most authors.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>HTML Template</title>
  </head>
  <body>
    <p>Insert content here.</p>
  </body>
</html>
```

Intermediate: HTML and XHTML Polyglot:

Template suitable for intermediate authors who are producing documents adhering to the polyglot markup conventions. While this template is safe for beginners to use, it is only recommended for authors familiar with the techniques for creating polyglot documents.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta charset="UTF-8"/>
    <title>Polyglot (X)HTML Template</title>
  </head>
  <body>
    <p>Insert content here.</p>
  </body>
</html>
```

Advanced: XHTML:

Template suitable for advanced authors who are producing XHTML documents. This template is not suitable for use as HTML because it lacks the DOCTYPE and does not declare the character encoding. Authors using this template should be using UTF-8 or UTF-16, or should at least be aware of how to declare the correct encoding for XML.

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>XHTML Template</title>
  </head>
  <body>
    <p>Insert content here.</p>
  </body>
</html>
```

Advanced: Minimal HTML:

Template suitable for advanced authors who are producing HTML documents. This template takes advantage of the ability to omit certain tags for some elements. Authors who are not familiar with, or are uncomfortable with, these features should avoid this template.

```
<!DOCTYPE html>
<meta charset="UTF-8">
<title>Minimal HTML Template</title>

<p>Insert content here.</p>
```

3 The HTML and XHTML Syntax

It is useful to make a distinction between the vocabulary of an HTML document—the elements and attributes, and their meanings—and the syntax in which it is written.

HTML has a defined set of elements and attributes which can be used in a document; each designed for a specific purpose with their own meaning. Consider this set of elements to be analogous to the list of words in a dictionary. This includes elements for headings, paragraphs, lists, tables, links, form controls and

many other features. This is the vocabulary of HTML. Similarly, just as natural languages have grammatical rules for how different words can be used, HTML has rules for where and how each element and attribute can be used.

The basic structure of elements in an HTML document is a tree structure. Most elements have at most one parent element, (except for the root element), and may have any number of child elements. This structure needs to be reflected in the syntax used to write the document.

3.1 Syntactic Overview

There are two syntaxes that can be used: the traditional HTML syntax, and the XHTML syntax. While these are similar, each is optimised for different needs and authoring habits. The former is more lenient in its design and handling requirements, and has a number of convenient shorthands for authors to use. The latter is based on XML and has much stricter syntactic requirements, designed to discourage the proliferation of syntactic errors.

The HTML syntax is loosely based upon the older, though very widely used syntax from HTML 4.01. Although it is inspired by its SGML origins, in practice, it really only shares minor syntactic similarities. This features a range of shorthand syntaxes, designed to make hand coding more convenient, such as allowing the omission of some optional tags and attribute values. Authors are free to choose whether or not they wish to take advantage of these shorthand features based upon their own personal preferences.

HTML Example:

A basic HTML document, demonstrating some shorthand syntax.

```
<!DOCTYPE html>
<html>
 <head>
   <title>An HTML Document</title>
 </head>
 <body class=example>
   <h1>Example</h1>
   <p>This is an example HTML document.
 </body>
</html>
```

XHTML, however, is based on the much more strict XML syntax. While this too is inspired by SGML, this syntax requires documents to be well-formed, which some people prefer because of its stricter error handling, forcing authors to maintain cleaner markup.

XHTML Example:

A basic XHTML document, demonstrating the stricter XML syntax.

```
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
   <title>An HTML Document</title>
 </head>
 <body class="example">
   <h1>Example</h1>
   <p>This is an example HTML document.</p>
 </body>
</html>
```

Note: The XHTML document does not need to include the DOCTYPE because XHTML documents that are delivered correctly using an XML MIME type and are processed as XML by browsers, are always rendered in no quirks mode. However, the DOCTYPE may optionally be included, and should be included if the document uses the compatible subset of markup that is conforming in both HTML and XHTML, and is ever expected to be used in `text/html` environments.

Due to the similarities of both the HTML and XHTML syntaxes, it is possible to mark up documents using a common subset of the syntax that is the same in both, while avoiding the syntactic features that are unique to each. This type of document is said to use polyglot markup, often simply referred to as a polyglot document, because it simultaneously conforms to both syntaxes and may be treated as either. There are a number of issues involved with creating such documents and authors wishing to do so should familiarise themselves with the similarities and differences between HTML and XHTML.

3.2 The Syntax

There are a number of basic components make up the syntax of HTML, that are used throughout any document. These include the DOCTYPE declaration, elements, attributes, comments, text and CDATA sections.

3.2.1 DOCTYPE Declaration

The Document Type Declaration needs to be present at the beginning of a document that uses the HTML syntax. It may optionally be used within the XHTML syntax, but it is not required. The following DOCTYPE is strongly recommended for most HTML documents:

Example:

The HTML5 DOCTYPE declaration.

```
<!DOCTYPE html>
```

For compatibility with legacy producers of HTML — that is, software that outputs HTML documents — an alternative legacy compatibility DOCTYPE is available. This is recommended for use by systems that are unable to output the DOCTYPE given above. This limitation occurs in software that expects a DOCTYPE to include either a PUBLIC or SYSTEM identifier, and is unable to omit them. The canonical form of this DOCTYPE is as follows:

Example:

The HTML5 legacy compatibility DOCTYPE declaration.

```
<!DOCTYPE html SYSTEM "about:legacy-compat">
```

Note: The term "legacy-compat" refers to compatibility with legacy producers only. In particular, it does not refer to compatibility with legacy browsers, which, in practice, ignore SYSTEM identifiers and DTDs.

In HTML, the DOCTYPE is case insensitive, except for the quoted string "about:legacy-compat", which must be written in lower case. This quoted string, however, may also be quoted with single quotes, rather than double quotes. The highlighted fragments below illustrate which parts are case insensitive in HTML.

Example:

The three conforming variants of the HTML5 DOCTYPE declarations, indicating case insensitivity in the HTML syntax.

```
<!DOCTYPE html>

<!DOCTYPE html SYSTEM "about:legacy-compat">

<!DOCTYPE html SYSTEM 'about:legacy-compat'>
```

For XHTML, the DOCTYPE may be omitted because it is unnecessary. If you choose to use a DOCTYPE, then the canonical case-sensitive versions of the above DOCTYPEs are recommended. But there are no restrictions placed on the use of alternative DOCTYPEs. You may, if you wish, use a custom DOCTYPE referring to a custom DTD, typically for validation purposes. Although, be advised that DTDs have a number of limitations compared with other alternative schema languages and validation techniques.

3.2.1.1 Obsolete But Permitted DOCTYPEs

In order to ease the the transitional process from previous editions of HTML, selected legacy DOCTYPEs are grandfathered into the HTML syntax as conforming, but they are considered obsolete. This does not apply to XHTML.

Warning: When using these DOCTYPEs, some validators may apply conformance requirements from older specifications, instead of the requirements of HTML5. HTML5 conformance checkers are required to issue a warning if one is present. Authors are strongly discouraged from using these DOCTYPEs.

For the permitted DOCTYPEs using only a public identifier, the syntax is as follows. The highlighted fragments indicate case insensitivity in the HTML syntax.

HTML Example:

The obsolete but permitted HTML 4.01 Strict DOCTYPE with only the public identifier.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD  HTML  4.01//EN">
```

For the permitted DOCTYPEs that also have a system identifier, the syntax is:

HTML Example:

The obsolete but permitted HTML 4.01 Strict DOCTYPE with both the public and system identifiers.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD  HTML  4.01//EN"
    SYSTEM "http://www.w3.org/TR/html4/strict.dtd">
```

In addition to those, the following public and system identifiers in the table below may be substituted accordingly.

Allowed values for public and system identifiers in an obsolete permitted DOCTYPE string.

Public Identifier	System Identifier
-//W3C//DTD HTML 4.0//EN	
-//W3C//DTD HTML 4.0//EN	http://www.w3.org/TR/REC-html40/strict.dtd
-//W3C//DTD HTML 4.01//EN	
-//W3C//DTD HTML 4.01//EN	http://www.w3.org/TR/html4/strict.dtd
-//W3C//DTD XHTML 1.0 Strict//EN	http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd
-//W3C//DTD XHTML 1.1//EN	http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd

3.2.1.2 HTML and XHTML Comparison

Comparison of the DOCTYPE syntax requirements between HTML and XHTML

	HTML	XHTML
DOCTYPE	Required	Optional
DOCTYPE keyword	Case-insensitive	Case-sensitive
html keyword	Case-insensitive	Case-sensitive
PUBLIC keyword	Case-insensitive	Case-sensitive
SYSTEM keyword	Case-insensitive	Case-sensitive
Public Identifier	Optional	Optional
System Identifier	Optional	Optional, but required if public identifier is present
Legacy Compatibility DOCTYPE	Allowed	Allowed
Other DOCTYPEs	Obsolete but permitted DOCTYPE list	Any XML compliant DOCTYPE (excludes HTML 4.01 and earlier DOCTYPEs)

3.2.1.3 Historical Notes

This section needs revising and may be moved to an external document and simply referred to.

The DOCTYPE originates from HTML's SGML lineage and, in previous levels of HTML, was originally used to refer to a Document Type Definition (DTD) — a formal declaration of the elements, attributes and syntactic features that could be used within the document. Those who are familiar with previous levels of HTML will notice that there is no PUBLIC identifier present in this DOCTYPE, which were used to refer to the DTD. Also, note that the about: URI scheme in the SYSTEM identifier of the latter DOCTYPE is used specifically because it cannot be resolved to any specific DTD.

As HTML5 is no longer formally based upon SGML, the DOCTYPE no longer serves this purpose, and thus no longer needs to refer to a DTD. However, due to legacy constraints, it has gained another very important purpose: triggering no-quirks mode in browsers.

HTML5 defines three modes: quirks mode, limited quirks mode and no quirks mode, of which only the latter is considered conforming to use. The reason for this is due to backwards compatibility. The important thing to understand is that there are some differences in the way documents are visually rendered in each of the modes; and to ensure the most standards compliant rendering, it is important to ensure no-quirks mode is used.

3.2.2 Elements

There are five different kinds of elements: normal, void, raw text, RCDATA and foreign elements. Each is represented syntactically by a tag. While the types have some similarities to each other, they are distinguished by their syntactic requirements for their content models and the types of tags that may be used.

3.2.2.1 Tags

All elements are identified by their tag name and are marked up using either start tags and end tags or self-closing tags. A start tag marks the beginning of an element, while an end tag marks the end. Start tags are delimited using angle brackets with the tag name and any attributes in between. End tags are delimited by angle brackets with a slash before the tag name.

> Example:
>
> The markup for the start and end tags of the p element.
>
> ```
> The quick brown fox jumps over the lazy dog.
> ```

A Self-closing tag is a special form of start tag with a slash immediately before the closing right angle bracket. These indicate that the element is to be closed immediately, and has no content. Where this syntax is permitted and used, the end tag must be omitted. In HTML, the use of this syntax is restricted to void elements and foreign elements. If it is used for other elements, it is treated as a start tag. In XHTML, it is possible for any element to use this syntax. But note that it is only conforming for elements with content models that permit them to be empty.

> Example:
>
> A br element using the self-closing tag syntax.
>
> ```
> The quick brown fox<br/>
> jumps over the lazy dog.
> ```

In HTML, tag names are case insensitive. It is conventional to use their canonical case, but this is not required. For example, they could be written in all uppercase or mixed case, depending on your own preferences.

HTML Example:

Uppercase tag names are permitted in the HTML syntax.

```
<DIV>...</DIV>
```

In XHTML, tag names are case sensitive and must be written in their canonical case. In general, the canonical case is lowercase for HTML and MathML elements, and camel case for SVG. Refer to the definition of each element if in doubt.

In both HTML and XHTML, within each tag, whitespace is permitted after the tag name, but it is not permitted before the tag name. Some authors choose to include a space before the slash in the self-closing tag. This practice is based upon a convention that originated within the compatibility guidelines in XHTML 1.0, Appendix C. However, adherence to this convention is unnecessary.

The permitted syntax of each element and its content model varies based on the type of element. The term empty element is used to describe an element that contains no content. Elements of any type can be empty, depending on their content model. Depending on the type, these are represented either by a start tag that is immediately closed, either implicitly or by the presence of an end tag, or by using the self-closing tag syntax.

Example:

An empty span element.

```
<span></span>
```

3.2.2.2 Void Elements

The term void elements is used to designate elements that must be empty. These requirements only apply to the HTML syntax. In XHTML, all such elements are treated as normal elements, but must be marked up as empty elements.

These elements are forbidden from containing any content at all. In HTML, these elements have a start tag only. The self-closing tag syntax may be used. The end tag must be omitted because the element is automatically closed by the parser.

HTML Example:

A void element in the HTML syntax. This is not permitted in the XHTML syntax.

```
<hr>
```

Example:

A void element using the HTML- and XHTML-compatible self-closing tag syntax.

```
<hr/>
```

XHTML Example:

A void element using the XHTML-only syntax with an explicit end tag. This is not permitted for void elements in the HTML syntax.

```
<hr></hr>
```

Such elements include, among others, br, hr, link and meta.

3.2.2.3 Raw Text Elements

The term raw text elements refers to elements within which the content is treated as raw text instead of markup. These requirements only apply to the HTML syntax. In XHTML, all such elements are treated as normal elements.

These require a start tag and an end tag. Neither the self-closing tag syntax, nor optional tags are supported.

These elements can only contain raw text. This means that other content like comments, character references and other elements cannot be represented in the HTML syntax. That is, the markup for such constructs is treated as text instead of markup. All occurrences of special characters do not need to be escaped as character references, as they otherwise would within normal elements.

HTML Example:

A script element containing some functions. Special characters do not need to be escaped using character references or a CDATA section.

```
<script>
```

```
function isLeap(year) {
        return year % 4 == 0 && year % 100 != 0 || year % 400 == 0;
}

function compare(a, b) {
        if (a < b)
                return -1;
        else if (a > b)
                return 1;
        return 0;
}
</script>
```

There is, however, one additional restriction that the text cannot contain a string that looks like the element's end tag. Specifically, a string that matches the pattern of "</" followed by a case insensitive match for the element's tag name, followed by either "/", ">", or whitespace.

Example Error:

A script element cannot contain a string that is indistinguishable from the end tag.

```
<script>
if (condition) {
        document.write("<script src=\"example.js\"></script>");
}
</script>
```

In the previous example, an HTML conformance checker would report the error as relating to the second end tag, believing that the first is in fact the end tag. Care must be taken to ensure such strings do not occur within the script.

HTML Example:

The javascript can be modified by escaping the slash, to workaround the HTML syntax error.

```
<script>
if (condition) {
        document.write("<script src=\"example.js\"><\/script>");
}
</script>
```

In XHTML, because such elements are treated as normal elements, markup like comments, character references and other elements can be represented. It is therefore necessary to escape all special characters using either character references or by enclosing the content within a CDATA section.

XHTML Example:

Special characters within a script element may be escaped using character references in the XHTML syntax.

```
<script>
function isLeap(year) {
        return year % 4 == 0 &amp;amp; year % 100 != 0 || year % 400 == 0;
}

function compare(a, b) {
        if (a &lt; b)
                return -1;
        else if (a &gt; b)
                return 1;
        return 0;
}
</script>
```

XHTML Example:

Special characters within a script element may be escaped using a CDATA section in the XHTML syntax.

```
<script><![CDATA[
function isLeap(year) {
        return year % 4 == 0 && year % 100 != 0 || year % 400 == 0;
}

function compare(a, b) {
        if (a < b)
                return -1;
        else if (a > b)
                return 1;
        return 0;
}
]]></script>
```

The above does not work in HTML because the strings "<![CDATA[" and "]]>" are treated as text instead of markup. However, by taking advantage of the scripting language's comment syntax, those strings can be commented so that they are ignored by the script in HTML, but still processed as markup in XHTML. This provides us with an an HTML- and XHTML- compatible syntax for scripts.

Example:

The HTML- and XHTML-compatible syntax for including javascript.

```
<script>//<![CDATA[
function isLeap(year) {
        return year % 4 == 0 && year % 100 != 0 || year % 400 == 0;
}

function compare(a, b) {
        if (a < b)
                return -1;
        else if (a > b)
                return 1;
        return 0;
}
//]]></script>
```

3.2.2.4 RCDATA Elements

The term RCDATA elements refers to elements within which character references are supported, but all other content is treated as raw text instead of markup. These requirements only apply to the HTML syntax. In XHTML, all such elements are treated as normal elements.

These require a start tag and an end tag. Neither the self-closing tag syntax, nor optional tags are supported.

These elements can only contain text or character references. This means that only character references are treated as markup, whereas all other content like comments and other elements cannot be represented in the HTML syntax. That is, the markup for such constructs is treated as text.

Occurrences of special characters may, but generally do not need to be escaped as character references, as they otherwise would within normal elements. But this is subject to the additional restriction that the text cannot include an ambiguous ampersand. That is, an ampersand followed a sequence of characters that looks like a named character reference, but which is not valid.

HTML Example:

A textarea element can contain text and character references.

```
<textarea>
This can contain character references like &amp;, &lt; and &gt;,
but such characters and also be written directly as &, < and >.
Strings that look like <!-- comments --> or other elements <span>
are treated as plain text, instead of markup.
</textarea>
```

Example Error:

An ambiguous ampersand cannot occur within a textarea element.

```
<textarea>
An unknown named character reference like &this
gives an ambiguous ampersand error.
</textarea>
```

Another additional restriction that the text cannot contain a string that looks like the element's end tag. Specifically, a string that matches the pattern of "</" followed by a case insensitive match for the element's tag name, followed by either "/", ">", or whitespace.

Example Error:

A textarea element cannot contain a string that is indistinguishable from the end tag.

```
<textarea>
This text cannot contain a string like </textarea because
it looks like an end tag for the element.
</textarea>
```

In XHTML, because such elements are treated as normal elements, markup like comments and other elements can also be represented. It is therefore necessary to escape all special characters using either character references or by enclosing the content within a CDATA section.

XHTML Example:

A textarea element can contain text and character references and comments in the XHTML syntax.

```
<textarea>
This can contain character references like &amp;, &lt; and &gt;.
Unlike the HTML syntax, this can also contain comments.
<!-- just like other normal elements -->

If the string is intended to be rendered, the special characters must be escaped
&lt;!-- like this --&gt;.
</textarea>
```

XHTML Example:

The content of a textarea element can also be contained within a CDATA section.

```
<textarea><![CDATA[This can contain special characters like &, < and >.
Strings that look like markup, like <!-- comments -->,
character references &amp;, &lt; and &gt;,
or other elements <span> are treated as plain text, instead of markup.
]]></textarea>
```

3.2.2.5 Foreign Elements

The term foreign elements refers to elements that are in the SVG and MathML namespaces.

These require a start tag and an end tag. The self-closing tag syntax may be used, where the element's content model permits. Optional tags are not supported.

These elements can contain text, character references, CDATA sections, other elements, and comments, but the text must not contain the less-than sign character (<) or an ambiguous ampersand.

Need examples.

3.2.2.6 Normal Elements

Normal Elements refers to all other HTML elements that are not categorised in any of the other types.

These have a start tag and an end tag. These elements do not support the self-closing tag syntax. Certain elements have optional tags, meaning that one or both tags may be omitted in specific circumstances where the presence of the tag will be implied by surrounding markup. Consult the element definitions to see which tags may be omitted.

Normal elements can contain text, character references, other elements, and comments, but the text must not contain the less-than sign character (<) or an ambiguous ampersand. Some additional restrictions apply on a per-element basis to some specific elements. Consult the element definitions for details.

Need examples.

3.2.2.7 Element Type Comparision

Comparison of the element type syntax requirements in HTML. (This does not apply to XHTML)

Element Type	Start Tag	End Tag	Self-Closing	Allowed Content Syntax
Normal Element	Yes (sometimes optional)	Yes (sometimes optional)	No	Text, elements, comments, character references
Void Element	Yes	No	Yes (optional)	Empty
Raw Text Element	Yes (Required)	Yes (Required)	No	Text
RCDATA Element	Yes (Required)	Yes (Required)	No	Text, character references
Foreign Element	Yes	Yes	Yes	Text, elements, CDATA sections, comments, character references

3.2.2.8 HTML and XHTML Comparison

Comparison of the element and tag syntax requirements between HTML and XHTML

	HTML	XHTML
Start Tag	Optional for selected elements; required for others, unless an empty element tag is used	Required for all elements
End Tag	Optional for selected elements; forbidden for void elements or if empty element tag is used; required for others	Required for all elements with matching start tag, unnecessary and forbidden if empty element tag is used
Self-Closing Tag	Allowed for void elements and foreign content elements (if content model permits)	Allowed for all elements (if content model permits)
Tag Name	Case-insensitive	Case-sensitive
Namespace Prefixes for Tag Name	Forbidden	Allowed
Normal Elements	Yes	Yes

Void Elements	Yes, use start tag with no end tag, or empty element tag	Yes, use start tag immediately closed by an end tag, or empty element tag
Raw Text Elements	Yes	No (treated as normal elements)
RCDATA Elements	Yes	No (treated as normal elements)
Foreign Elements	SVG and MathML only	Any
Line feed immediately after start tag	Normalised	Allowed

3.2.3 Attributes

Elements may have attributes that are used to specify additional information about them. Some attributes are defined globally and can be used on any HTML element, while others are defined for specific elements only. Every attribute must have an attribute name that is used to identify it. Every attribute also has an associated attribute value, which, depending on the attribute's definition, may represent one of several different types. The permitted syntax for each attribute depends on the given value.

A typical attribute in HTML has a name and a value separated by an equals sign (=). Attributes are placed within a start tag and are separated from the tag name and from each other by whitespace. They must not be specified within an end tag.

Example:

The syntax of two typical attributes.

```
<div class="example" lang="en">...</div>
```

Example Error:

Attributes cannot be placed within end tags.

```
<section id="example">...</section id="example">
```

In HTML, attribute names are case insensitive. It is conventional to use their canonical case, but this is not required. For example, they could be written in all uppercase or mixed case, depending on your own preferences. For custom attributes, attribute names must consist of one or more characters other than the space characters, control characters, NULL, one of the characters: double quote ("), single quote ('), greater-than sign (>), solidus (/), equals sign (=), nor any characters that are not defined by Unicode. It is, however, recommended that authors use the lowercase letters in the ranges a - z and 0 - 9.

HTML Example:

Attribute names in HTML are case-insensitive, and can be written in uppercase or mixed case.

```
<div CLASS="example" Lang="en">...</div>
```

In XHTML, attribute names are case sensitive and must be written in their canonical case. In general, the canonical case is lowercase for most attributes on HTML and MathML elements, and camel case for SVG. Refer to the definition of each element if in doubt.

Attribute values may contain text and character references, subject to the restriction that the text cannot contain an ambiguous ampersand, and to any additional restrictions imposed by the syntax being used. There are four slightly different syntaxes that may be used for attributes in HTML: empty, unquoted, single-quoted and double-quoted. All four syntaxes may be used in the HTML syntax, depending on what is needed for each specific attribute. However, in the XHTML syntax, attribute values must always be quoted using either single or double quotes.

3.2.3.1 Empty Attribute Syntax

An empty attribute is one where the value has been omitted. This is a syntactic shorthand for specifying the attribute with an empty value, and is commonly used for boolean attributes. This syntax may be used in the HTML syntax, but not in the XHTML syntax.

Note: In previous editions of HTML, which were formally based on SGML, it was technically an attribute's name that could be omitted where the value was a unique enumerated value specified in the DTD. However, due to legacy constraints, this has been changed in HTML5 to reflect the way implementations really work.

HTML Example:

Empty attributes may have their value omitted in the HTML syntax.

```
<input disabled>...</div>
```

Example:

Specifying the value as as an empty string is equivalent to the empty attribute syntax. This is an XHTML-compatible syntax.

```
<input disabled="">...</div>
```

Note: The previous examples are semantically equivalent to specifying the attribute with the value "disabled", but are not exactly the same.

This syntax is permitted only for boolean attributes.

3.2.3.2 Unquoted Attribute Value Syntax

An unquoted attribute value is one where the value is supplied, but is not surrounded by quotation marks. This syntax may be used in the HTML syntax, but not in the XHTML syntax.

The attribute value must not contain any literal space characters, any of the characters: double quote ("), apostrophe ('), equals sign (=), less-than sign (<), greater-than sign (>), or grave accent (`), and the value must not be the empty string. To represent those characters in an attribute value, they either need to be escaped using character references, or you need to use either the single- or double-quoted attribute values.

HTML Example:

Some attribute values may be left unquoted in the HTML syntax.

```
<img src=example.png height=200 width=300/>
```

In the previous example, each attribute is separated from the last by a space. The slash at the end, which despite not being separated from the last value by any space characters, is not considered part of the attribute's value. Instead, the the slash indicates the use of the self-closing tag syntax.

There may be space characters between the attribute name and the equals sign (=), and between that and the attribute value.

HTML Example:

Whitespace is permitted before and after the the equals sign, separating the name and value.

```
<img src  =  image.png>
```

3.2.3.3 Double-Quoted Attribute Value Syntax

An double-quoted attribute value is one where the supplied value is surrounded by double quotation marks ("). This syntax may be used in both HTML and XHTML.

The attribute value must delimited by double-quote characters (") before and after the value, and must not contain any double-quote characters or an ambiguous ampersands in between. All other text and character references are permitted.

The advantage of quoting attributes, the value may contain the additional characters that can't be used in unquoted attribute values. To include a double-quote character within the value, either use a character reference ("), or use a single-quoted attribute value instead.

Example:

Attribute values may be quoted with double quote marks.

```
<img src="example.jpg" title="An example photograph">
```

There may be space characters between the attribute name and the equals sign (=), and between that and the attribute value.

HTML Example:

Whitespace is permitted before and after the the equals sign, separating the name and value.

```
<img src  =  "example.png">
```

3.2.3.4 Single-Quoted Attribute Value Syntax

An single-quoted attribute value is one where the supplied value is surrounded by single quotation marks ('). This syntax may be used in both HTML and XHTML.

The attribute value must delimited by single-quote characters (') before and after the value, and must not contain any single-quote characters or an ambiguous ampersands in between. All other text and character references are permitted.

The advantage of quoting attributes, the value may contain the additional characters that can't be used in unquoted attribute values. To include a single-quote (or apostrophe) character within the value, either use a character reference ('), or use a double-quoted attribute value instead.

Example:

Attribute values may be quoted with single quote marks.

```
<img src='example.jpg' title='An example photograph'>
```

There may be space characters between the attribute name and the equals sign (=), and between that and the attribute value.

HTML Example:

Whitespace is permitted before and after the the equals sign, separating the name and value.

```
<img src  =  'example.png'>
```

3.2.3.5 HTML and XHTML Comparison

Comparison of the attribute syntax requirements between HTML and XHTML

	HTML	XHTML
Attribute Names	Case-insensitive	Case-sensitive
Namespace Prefixes for Attribute Names	Limited, specified prefixes only	Allowed
Empty attributes	Allowed	Forbidden
Unquoted attributes	Allowed	Forbidden
Double quoted attributes	Allowed	Allowed
Single quoted attributes	Allowed	Allowed
Line feeds in attribute values	Allowed	Normalised to spaces

3.2.4 Comments

...

3.2.5 Text

...

3.2.6 CDATA Sections

...

3.2.7 Character References

Discuss numeric and named character reference syntax. May link to the list of entity references in a separate document, rather than trying to list them all in here.

3.3 Understanding MIME Types

Discuss text/html, application/xhtml+xml, etc.

3.4 Character Encoding

Overview of Unicode, character repertoires, encodings, etc. Declaring the encoding with the Content-Type header, BOM, meta, etc.

3.5 Choosing HTML or XHTML

The choice of HTML or XHTML syntax is largely dependent upon a number of factors the, including needs of a given project, the skill set of the developers involved, level of support in browsers used by the site's target audience, or it may simply be a matter of personal preference.

The important thing to understand is that there are valid reasons to choose both, and that authors are encouraged to make an informed decision.

Need to develop guidelines to help authors make this choice.

3.6 Polyglot Documents

A polyglot HTML document is a document that conforms to both the HTML and XHTML syntactic requirements, and which can be processed as either by browsers, depending on the MIME type used. This works by using a common subset of the syntax that is shared by both HTML and XHTML.

Polyglot documents are useful to create for situations where a document is intended to be served as either HTML or XHTML, depending on the support in particular browsers, or when it is not known at the time of

creation, which MIME type the document will ultimately be served as.

In order to successfully create and maintain polyglot documents, authors need to be familiar with both the similarities and differences between the two syntaxes. This includes not only syntactic differences, but also differences in the way stylesheets, and scripts are handled, and the way in which character encodings are detected.

This section will provide the details about each of these similarities and differences, and provide guidelines on the creation of polyglot documents.

Base this on the HTML vs. XHTML article.

4 The HTML Vocabulary and APIs

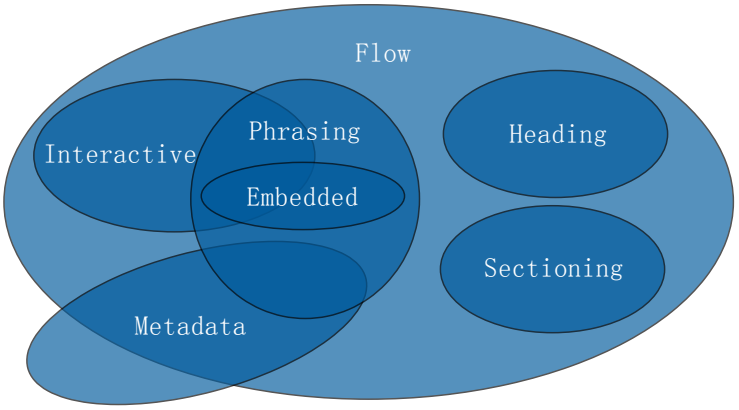
4.1 Categories

Each element in HTML falls into zero or more categories that group elements with similar characteristics together. The following categories are used in this guide:

- Metadata content
- Flow content
- Sectioning root
- Sectioning content
- Heading content
- Phrasing content
- Embedded content
- Interactive content
- Transparent

Some elements have unique requirements and do not fit into any particular category.

These categories are related as follows:



[Create and link to some sort of index of elements that lists each element in each category.]

4.1.1 Metadata Content

Metadata content includes elements for marking up document metadata; marking up or linking to resources that describe the behaviour or presentation of the document; or indicate relationships with other documents.

Metadata elements appear within the head of a document. Some common examples of metadata elements include: title, meta, link, script and style.

4.1.2 Flow content

Most elements that are used in the body of documents and applications are categorised as flow content. Most of the elements used to mark up the main content in the body of a page are considered to be flow content. In general, this includes elements that are presented visually as either block level or inline level.

Some common flow content includes elements like div, p, em and strong.

Elements categorised as heading content, phrasing content or embedded content are also considered to be flow content.

4.1.3 Sectioning root

[This description needs improving.]

These elements can have their own outlines, but the sections and headers inside these elements do not contribute to the outlines of their ancestors.

Some common sectioning root elements include, among others, body, blockquote and figure.

4.1.4 Sectioning content

Sectioning content is used for structuring a document into sections, each of which generally has its own heading. These elements provide a scope within which associated headers, footers and contact information apply.

Some common sectioning elements include, among others, section, article and nav.

Most sectioning elements, with the exception of the body element, are also classified as flow content.

4.1.5 Heading content

Heading content includes the elements for marking up headers. Headings, in conjunction with the sectioning elements, are used to describe the the structure of the document.

Heading content includes the header element and the h1 to h6 elements.

Elements categorised as heading content are considered to be flow content.

4.1.6 Phrasing content

Phrasing content includes text and text-level markup. This is similar to the concept of inline level elements in HTML 4.01. Most elements that are categorised as phrasing content can only contain other phrasing content.

Some common examples of phrasing content elements include abbr, em, strong and span

Elements categorised as phrasing content are considered to be flow content.

4.1.7 Embedded content

Embedded content includes elements that load external resources into the document. Such external resources include, for example, images, videos and Flash-based content. Some embedded content elements include img, object, embed and video.

Elements categorised as embedded content are considered to be phrasing content, and thus also considered to be flow content.

4.1.8 Interactive content

Interactive elements are those that allow the user to interact with or activate in some way. Depending on the user's browser and device, this could be performed using any kind of input device, such as, for example, a mouse, keyboard, touch screen or voice input.

Some common examples of interactive content include a, audio and video when used with the controls attribute, and most form controls using input.

4.1.9 Transparent Content Models

Some elements have transparent content models, meaning that their allowed content depends upon the parent element. They may contain any content that their parent element may contain, in addition to any other allowances or exceptions described for the element.

When the element has no parent, then the content model defaults to flow content.

4.2 Global Attributes

To be completed.

4.3 The Elements

Expect major changes to this section. Each of these needs longer descriptions and the elements should be divided into categories. The IDL for the DOM Interfaces is likely to be replaced by something a lot more reader-friendly in the future; consider it a placeholder for now. Attributes will likely be accompanied by brief descriptions within the summary box, in addition to fuller descriptions and examples afterwards.

4.3.1 The Root Element

4.3.1.1 The html element

Start tag: optional End tag: optional

The html element represents the root of an HTML document.

Categories:	None.
Contained By:	As the root element of a document. Wherever a subdocument fragment is allowed in a compound document.
Content Model:	A head element followed by a body element.

Attributes	DOM Interface
Global attributes manifest	Uses HTMLElement.

The html element is the root element of a document. Every document must begin with this element, and it must contain both the head and body elements.

It is considered good practice to specify the primary language of the document on this element using the lang attribute.

HTML Example:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    ...
  </head>
  <body>
    ...
  </body>
</html>
```

In the HTML syntax only, both the start and end tags are optional, and so for convenience either may be omitted, unless you wish to specify attributes on this element, in which case, at least the start tag needs to be included.

HTML Example:

```
<!DOCTYPE html>
<head>
  ...
</head>
<body>
  ...
</body>
```

In the XHTML syntax, the xmlns attribute needs to be specified on this element to declare that it is in the HTML namespace. You may use either the lang or xml:lang attribute to specify the language.

XHTML Example:

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    ...
  </head>
  <body>
    ...
  </body>
</html>
```

manifest
: The manifest attribute gives the address of the document's application cache manifest, if there is one. If the attribute is present, the attribute's value must be a valid URL.

Need to describe application cache manifests.

4.3.2 Document Metadata

4.3.2.1 The head element — Start tag: optional End tag: optional

The head element collects the document's metadata.

Categories:	None.
Contained By:	As the first element in an html element.
Content Model:	One or more elements of metadata content, of which exactly one is a title element.

Attributes	DOM Interface

Global attributes	Uses HTMLElement.

The head element is the container for the document's metadata. Metadata is information about the document itself, such as it's title, author. Scripts and stylesheets may also be included within the head element. Every document must have a head element.

The following examples illustrate the typical usage of the head element in HTML and XHTML.

HTML Example

```
<!DOCTYPE html>
<html>
<head>
  <title>Example</title>
</head>
<body>
  <h1>Document</h1>
</body>
</html>
```

XHTML Example

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Example</title>
</head>
<body>
  <h1>Document</h1>
</body>
</html>
```

4.3.2.2 The title element

Start tag: required End tag: required

The title element represents the document's title or name, and should be meaningful even when read out of context.

Categories: Metadata content.

Contained By: In a head element containing no other title elements.

Content Model: Text.

Attributes	DOM Interface
Global attributes	Uses HTMLElement.

4.3.2.3 The base element

Start tag: required End tag: empty

The base element is for specifying a base URL against which relative links will be resolved, and the name of the default target for opening links and form submissions.

Categories: Metadata content.

Contained By: In a head element containing no other base elements.

Content Model: Empty.

Attributes	DOM Interface

Global attributes
href
target
	interface HTMLBaseElement : HTMLElement {
	attribute DOMString href;
	attribute DOMString target;
	};

4.3.2.4 The link element

Start tag: required End tag: empty

The link is for linking to other resources, such as stylesheets, favicons and syndication feeds.

Categories: Metadata content.

Contained By: Where metadata content is expected.
In a noscript element that is a child of a head element.

Content Model: Empty.

Attributes	DOM Interface

Global attributes
href
rel
media
hreflang
type
sizes
Also, the title attribute has special semantics on this element.

```
interface HTMLLinkElement : HTMLElement {
           attribute boolean disabled;
           attribute DOMString href;
           attribute DOMString rel;
  readonly attribute DOMTokenList relList;
           attribute DOMString media;
           attribute DOMString hreflang;
           attribute DOMString type;
           attribute DOMString sizes;
};
```

The LinkStyle interface must also be implemented by this element, the styling processing model defines how. [CSSOM]

4.3.2.5 The meta element

Start tag: required End tag: empty

The meta element is for providing various types of metadata, such as the application-name or specifying the documents character encoding.

Categories: Metadata content.

Contained By:
If the charset attribute is present, or if the element is in the Encoding declaration state: in a head element.
If the http-equiv attribute is present, and the element is not in the Encoding declaration state: in a head element.
If the http-equiv attribute is present, and the element is not in the Encoding declaration state: in a noscript element that is a child of a head element.
If the name attribute is present: where metadata content is expected.

Content Model: Empty.

ATTRIBUTES

Global attributes
name
http-equiv
content
charset

DOM INTERFACE

```
interface HTMLMetaElement : HTMLElement {
           attribute DOMString content;
           attribute DOMString name;
           attribute DOMString httpEquiv;
};
```

4.3.2.6 The style element

Start tag: required End tag: required

The style element allows authors to embed stylesheets, typically CSS, within their documents.

Categories: Metadata content.
If the scoped attribute is present: flow content.

Contained By:
If the scoped attribute is absent: where metadata content is expected.
If the scoped attribute is absent: in a noscript element that is a child of a head element.
If the scoped attribute is present: where flow content is expected, but before any other flow content other than other style elements and inter-element whitespace.

Content Model: Depends on the value of the type attribute.

ATTRIBUTES

Global attributes
media
type
scoped
Also, the title attribute has special semantics on this element.

DOM INTERFACE

```
interface HTMLStyleElement : HTMLElement {
           attribute boolean disabled;
           attribute DOMString media;
           attribute DOMString type;
           attribute boolean scoped;
};
```

The LinkStyle interface must also be implemented by this element, the styling processing model defines how. [CSSOM]

4.3.3 Scripting

4.3.3.1 The script element

Start tag: required End tag: required

The script element allows authors to include scripts, typically javaScript, and data blocks in their documents.

Categories:	Metadata content. Flow content. Phrasing content.
Contained By:	Where metadata content is expected. Where phrasing content is expected.
Content Model:	If there is no src attribute, depends on the value of the type attribute. If there is a src attribute, the element must be either empty or contain only script documentation.

ATTRIBUTES

Global attributes
src
async
defer
type
charset

DOM INTERFACE

```
interface HTMLScriptElement : HTMLElement {
           attribute DOMString src;
           attribute boolean async;
           attribute boolean defer;
           attribute DOMString type;
           attribute DOMString charset;
           attribute DOMString text;
};
```

4.3.3.2 The noscript element

Start tag: required End tag: required

The noscript element is used to provide alternative content for users using browsers that do not support scripting or have it disabled.

Categories:	Metadata content. Flow content. Phrasing content.
Contained By:	In a head element of an HTML document, if there are no ancestor noscript elements. Where phrasing content is expected in HTML documents, if there are no ancestor noscript elements.
Content Model:	When scripting is disabled, in a head element: in any order, zero or more link elements, zero or more style elements, and zero or more meta elements. When scripting is disabled, not in a head element: transparent, but there must be no noscript element descendants. Otherwise: text that conforms to the requirements given in the prose.

ATTRIBUTES

Global attributes

DOM INTERFACE

Uses HTMLElement.

4.3.4 Sections

4.3.4.1 The body element

Start tag: optional End tag: optional

The body element represents the main content of the document.

Categories:	Sectioning root.
Contained By:	As the second element in an html element.
Content Model:	Flow content.

ATTRIBUTES

Global attributes
onbeforeunload
onerror
onhashchange
onload
onmessage
onoffline
ononline

DOM INTERFACE

```
interface HTMLBodyElement : HTMLElement {
           attribute Function onbeforeunload;
           attribute Function onerror;
           attribute Function onhashchange;
           attribute Function onload;
           attribute Function onmessage;
           attribute Function onoffline;
           attribute Function ononline;
           attribute Function onpopstate;
           attribute Function onresize;
           attribute Function onstorage;
```

onpopstate onresize onstorage onunload	attribute Function onunload; };

4.3.4.2 The section element

Start tag: required End tag: required

The section element represents a generic document or application section. A section, in this context, is a thematic grouping of content, typically with a header and possibly a footer.

Categories: Flow content. Sectioning content.

Contained By: Where flow content is expected.

Content Model: Flow content.

Attributes	DOM Interface
Global attributes	Uses HTMLElement.

4.3.4.3 The nav element

Start tag: required End tag: required

The nav element represents a section of a page containing primary navigation links to other pages or to parts within the page.

Categories: Flow content. Sectioning content.

Contained By: Where flow content is expected.

Content Model: Flow content.

Attributes	DOM Interface
Global attributes	Uses HTMLElement.

4.3.4.4 The article element

Start tag: required End tag: required

The article element represents an independent section of a document, page, or site. This could be a forum post, a magazine or newspaper article, a blog entry, a user-submitted comment, or any other independent item of content.

Categories: Flow content. Sectioning content.

Contained By: Where flow content is expected.

Content Model: Flow content.

Attributes	DOM Interface
Global attributes	Uses HTMLElement.

4.3.4.5 The aside element

Start tag: required End tag: required

The aside element represents a section of a page that consists of content that is tangentially related to the content around the aside element, and which could be considered separate from that content. Such sections are often represented as sidebars in printed typography.

Categories: Flow content. Sectioning content.

Contained By: Where flow content is expected.

Content Model: Flow content.

Attributes	DOM Interface

Global attributes	Uses HTMLElement.

4.3.4.6 The h1, h2, h3, h4, h5, and h6 elements

Start tag: required End tag: required

These elements define headers for their sections.

Categories:	Flow content. Heading content.
Contained By:	Where flow content is expected.
Content Model:	Phrasing content.

Attributes	DOM Interface
Global attributes	Uses HTMLElement.

4.3.4.7 The header element

Start tag: required End tag: required

The header element represents the header of a section, typically containing headings and subheadings, and other metadata about the section.

Categories:	Flow content. Heading content.
Contained By:	Where flow content is expected.
Content Model:	Flow content, including at least one descendant that is heading content, but no sectioning content descendants, no header element descendants, and no footer element descendants.

Attributes	DOM Interface
Global attributes	Uses HTMLElement.

4.3.4.8 The footer element

Start tag: required End tag: required

The footer element represents a footer of a section, typically containing information such as who wrote it, links to related documents, and copyright notices.

Categories:	Flow content.
Contained By:	Where flow content is expected.
Content Model:	Flow content, but with no heading content descendants, no sectioning content descendants, and no footer element descendants.

Attributes	DOM Interface
Global attributes	Uses HTMLElement.

4.3.4.9 The address element

Start tag: required End tag: required

The address element represents the contact information for the section it applies to. If it applies to the body element, then it instead applies to the document as a whole.

Categories:	Flow content.
Contained By:	Where flow content is expected.
Content Model:	Flow content, but with no heading content descendants, no sectioning content descendants, no footer element descendants, and no address element descendants.

Attributes	DOM Interface
Global attributes	Uses HTMLElement.

4.3.4.10 Headings and Sections

A typical document is often structured into different sections and subsections, with each potentially having its own heading and possibly a subheading. These heading and sectioning elements provide a way for this structure to be conveyed to the reader.

Consider a simple page like a blog, which is structured into a few main sections.

4.3.5 Grouping Content

4.3.5.1 The p element — Start tag: required End tag: optional

The p element represents a paragraph.

Categories:	Flow content.
Contained By:	Where flow content is expected.
Content Model:	Phrasing content.

Attributes	DOM Interface
Global attributes	Uses HTMLElement.

4.3.5.2 The hr element — Start tag: required End tag: empty

The hr element represents a paragraph-level thematic break, e.g. a scene change in a story, or a transition to another topic within a section of a reference book.

Categories:	Flow content.
Contained By:	Where flow content is expected.
Content Model:	Empty.

Attributes	DOM Interface
Global attributes	Uses HTMLElement.

4.3.5.3 The br element — Start tag: required End tag: empty

The br element represents a line break.

Categories:	Flow content. Phrasing content.
Contained By:	Where phrasing content is expected.
Content Model:	Empty.

Attributes	DOM Interface
Global attributes	Uses HTMLElement.

4.3.5.4 The pre element — Start tag: required End tag: required

The pre element represents a block of preformatted text, in which structure is represented by typographic conventions rather than by elements.

Categories:	Flow content.
Contained By:	Where flow content is expected.
Content Model:	Phrasing content.

Attributes	DOM Interface
Global attributes	Uses HTMLElement.

4.3.5.5 The dialog element

Start tag: required End tag: required

The dialog element represents a conversation.

Categories: Flow content.

Contained By: Where flow content is expected.

Content Model: Zero or more pairs of one dt element followed by one dd element.

ATTRIBUTES

Global attributes

DOM INTERFACE

Uses HTMLElement.

4.3.5.6 The blockquote element

Start tag: required End tag: required

The blockquote element represents a section that is quoted from another source.

Categories: Flow content.
Sectioning root.

Contained By: Where flow content is expected.

Content Model: Flow content.

ATTRIBUTES

Global attributes
cite

DOM INTERFACE

```
interface HTMLQuoteElement : HTMLElement {
           attribute DOMString cite;
};
```

The HTMLQuoteElement interface is also used by the q element.

4.3.5.7 The ol element

Start tag: required End tag: required

The ol element represents an ordered list.

Categories: Flow content.

Contained By: Where flow content is expected.

Content Model: Zero or more li elements.

ATTRIBUTES

Global attributes
reversed
start

DOM INTERFACE

```
interface HTMLOListElement : HTMLElement {
           attribute boolean reversed;
           attribute long start;
};
```

4.3.5.8 The ul element

Start tag: required End tag: required

The ul element represents an unordered list.

Categories: Flow content.

Contained By: Where flow content is expected.

Content Model: Zero or more li elements.

ATTRIBUTES

Global attributes

DOM INTERFACE

Uses HTMLElement.

4.3.5.9 The li element

Start tag: required End tag: optional

The li element represents a list item.

Categories: None.

Contained By: Inside ol elements.
Inside ul elements.
Inside menu elements.

Content Model: Flow content.

Attributes

Global attributes
If the element is a child of an ol element: value

DOM Interface

```
interface HTMLLIElement : HTMLElement {
          attribute long value;
};
```

4.3.5.10 The dl element

Start tag: required End tag: required

The dl element introduces an association list containing groups of terms and associated descriptions. (a description list).

Categories: Flow content.

Contained By: Where flow content is expected.

Content Model: Zero or more groups each consisting of one or more dt elements followed by one or more dd elements.

Attributes

Global attributes

DOM Interface

Uses HTMLElement.

4.3.5.11 The dt element

Start tag: required End tag: optional

The dt element represents the term, or name, part of a term-description group in a description list (dl element), and the talker, or speaker, part of a talker-discourse pair in a conversation (dialog element).

Categories: None.

Contained By: Before dd or dt elements inside dl elements.
Before a dd element inside a dialog element.

Content Model: Phrasing content.

Attributes

Global attributes

DOM Interface

Uses HTMLElement.

4.3.5.12 The dd element

Start tag: required End tag: optional

The dd element represents the description, definition, or value, part of a term-description group in a description list (dl element), and the discourse, or quote, part in a conversation (dialog element).

Categories: None.

Contained By: After dt or dd elements inside dl elements.
After a dt element inside a dialog element.

Content Model: Flow content.

Attributes

Global attributes

DOM Interface

Uses HTMLElement.

4.3.6 Text-Level Semantics

4.3.6.1 The a element

Start tag: required End tag: required

If the a element has an href attribute, then it represents a hyperlink.

Categories: Interactive content.
Flow content.
When the element only contains phrasing content: phrasing content.

Contained By: Where phrasing content is expected.

Content Model: Transparent, but there must be no interactive content descendant.

ATTRIBUTES

Global attributes
href
target
ping
rel
media
hreflang
type

DOM INTERFACE

```
[Stringifies=href] interface HTMLAnchorElement : HTMLElement {
           attribute DOMString href;
           attribute DOMString target;
           attribute DOMString ping;
           attribute DOMString rel;
  readonly attribute DOMTokenList relList;
           attribute DOMString media;
           attribute DOMString hreflang;
           attribute DOMString type;
};
```

The Command interface must also be implemented by this element.

4.3.6.2 The q element

Start tag: required End tag: required

The q element represents a phrase quoted from another source.

Categories: Flow content. Phrasing content.

Contained By: Where phrasing content is expected.

Content Model: Phrasing content.

ATTRIBUTES

Global attributes
cite

DOM INTERFACE

The q element uses the HTMLQuoteElement interface.

4.3.6.3 The cite element

Start tag: required End tag: required

The cite element represents the title of a work, such as an article, a book, a poem, a song, a film, or any other creative work.

Categories: Flow content. Phrasing content.

Contained By: Where phrasing content is expected.

Content Model: Phrasing content.

ATTRIBUTES

Global attributes

DOM INTERFACE

Uses HTMLElement.

4.3.6.4 The em element

Start tag: required End tag: required

The em element represents stress emphasis of its contents.

Categories: Flow content. Phrasing content.

Contained By: Where phrasing content is expected.

Content Model: Phrasing content.

ATTRIBUTES

Global attributes

DOM INTERFACE

Uses HTMLElement.

4.3.6.5 The strong element

Start tag: required End tag: required

The strong element represents strong importance for its contents.

Categories:	Flow content. Phrasing content.
Contained By:	Where phrasing content is expected.
Content Model:	Phrasing content.

Attributes	DOM Interface
Global attributes	Uses HTMLElement.

4.3.6.6 The small element

Start tag: required End tag: required

The small element represents small print (part of a document often describing legal restrictions, such as copyrights or other disadvantages), or other side comments.

Categories:	Flow content. Phrasing content.
Contained By:	Where phrasing content is expected.
Content Model:	Phrasing content.

Attributes	DOM Interface
Global attributes	Uses HTMLElement.

4.3.6.7 The mark element

Start tag: required End tag: required

The mark element represents a run of text in one document marked or highlighted for reference purposes, due to its relevance in another context.

Categories:	Flow content. Phrasing content.
Contained By:	Where phrasing content is expected.
Content Model:	Phrasing content.

Attributes	DOM Interface
Global attributes	Uses HTMLElement.

4.3.6.8 The dfn element

Start tag: required End tag: required

The dfn element represents the defining instance of a term, where its definition is provided nearby.

Categories:	Flow content. Phrasing content.
Contained By:	Where phrasing content is expected.
Content Model:	Phrasing content, but there must be no descendant dfn elements.

Attributes	DOM Interface
Global attributes Also, the title attribute has special semantics on this element.	Uses HTMLElement.

4.3.6.9 The abbr element

Start tag: required End tag: required

The abbr element represents an abbreviation or acronym, optionally with its expansion.

Categories:	Flow content. Phrasing content.
Contained By:	Where phrasing content is expected.

Content Model: Phrasing content.

Attributes

Global attributes
Also, the title attribute has special semantics on this element.

DOM Interface

Uses HTMLElement.

4.3.6.10 The time element

Start tag: required End tag: required

The time element represents a date and/or a time.

Categories: Flow content.
Phrasing content.

Contained By: Where phrasing content is expected.

Content Model: Phrasing content.

Attributes

Global attributes
datetime

DOM Interface

```
interface HTMLTimeElement : HTMLElement {
           attribute DOMString dateTime;
  readonly attribute Date date;
  readonly attribute Date time;
  readonly attribute Date timezone;
};
```

4.3.6.11 The progress element

Start tag: required End tag: required

The progress element represents the completion progress of a task.

Categories: Flow content.
Phrasing content.

Contained By: Where phrasing content is expected.

Content Model: Phrasing content.

Attributes

Global attributes
value
max

DOM Interface

```
interface HTMLProgressElement : HTMLElement {
           attribute float value;
           attribute float max;
  readonly attribute float position;
};
```

4.3.6.12 The meter element

Start tag: required End tag: required

The meter element represents a scalar measurement within a known range, or a fractional value.

Categories: Flow content.
Phrasing content.

Contained By: Where phrasing content is expected.

Content Model: Phrasing content.

Attributes

Global attributes
value
min
low
high
max
optimum

DOM Interface

```
interface HTMLMeterElement : HTMLElement {
           attribute float value;
           attribute float min;
           attribute float max;
           attribute float low;
           attribute float high;
           attribute float optimum;
};
```

4.3.6.13 The code element

Start tag: required End tag: required

The code element represents a fragment of computer code.

Categories:	Flow content. Phrasing content.
Contained By:	Where phrasing content is expected.
Content Model:	Phrasing content.

Attributes	DOM Interface
Global attributes	Uses HTMLElement.

4.3.6.14 The var element

Start tag: required End tag: required

The var element represents a variable, such as in a mathematical expression or programming context, or it could just be a term used as a placeholder in prose.

Categories:	Flow content. Phrasing content.
Contained By:	Where phrasing content is expected.
Content Model:	Phrasing content.

Attributes	DOM Interface
Global attributes	Uses HTMLElement.

4.3.6.15 The samp element

Start tag: required End tag: required

The samp element represents (sample) output from a program or computing system.

Categories:	Flow content. Phrasing content.
Contained By:	Where phrasing content is expected.
Content Model:	Phrasing content.

Attributes	DOM Interface
Global attributes	Uses HTMLElement.

4.3.6.16 The kbd element

Start tag: required End tag: required

The kbd element represents user input (typically keyboard input, although it may also be used to represent other input, such as voice commands).

Categories:	Flow content. Phrasing content.
Contained By:	Where phrasing content is expected.
Content Model:	Phrasing content.

Attributes	DOM Interface
Global attributes	Uses HTMLElement.

4.3.6.17 The sub and sup elements

Start tag: required End tag: required

The sup element represents a superscript and the sub element represents a subscript.

Categories:	Flow content. Phrasing content.
Contained By:	Where phrasing content is expected.

Content Model: Phrasing content.

Attributes	DOM Interface
Global attributes	Uses HTMLElement.

4.3.6.18 The span element

Start tag: required End tag: required

The span element doesn't mean anything on its own, but can be useful when used together with other attributes, e.g. class, lang, or dir.

Categories: Flow content. Phrasing content.

Contained By: Where phrasing content is expected.

Content Model: Phrasing content.

Attributes	DOM Interface
Global attributes	Uses HTMLElement.

4.3.6.19 The i element

Start tag: required End tag: required

The i element represents a span of text in an alternate voice or mood, or otherwise offset from the normal prose, such as a taxonomic designation, a technical term, an idiomatic phrase from another language, a thought, a ship name, or some other prose whose typical typographic presentation is italicized.

Categories: Flow content. Phrasing content.

Contained By: Where phrasing content is expected.

Content Model: Phrasing content.

Attributes	DOM Interface
Global attributes	Uses HTMLElement.

4.3.6.20 The b element

Start tag: required End tag: required

The b element represents a span of text to be stylistically offset from the normal prose without conveying any extra importance, such as key words in a document abstract, product names in a review, or other spans of text whose typical typographic presentation is boldened.

Categories: Flow content. Phrasing content.

Contained By: Where phrasing content is expected.

Content Model: Phrasing content.

Attributes	DOM Interface
Global attributes	Uses HTMLElement.

4.3.6.21 The bdo element

Start tag: required End tag: required

The bdo element allows authors to override the Unicode bidi algorithm by explicitly specifying a direction override. [BIDI]

Categories: Flow content. Phrasing content.

Contained By: Where phrasing content is expected.

Content Model: Phrasing content.

Attributes	DOM Interface

Global attributes Also, the dir global attribute has special semantics on this element.	Uses HTMLElement.

4.3.6.22 The ruby element

Start tag: required End tag: required

The ruby element allows one or more spans of phrasing content to be marked with ruby annotations. Ruby annotations are short runs of text presented alongside base text, primarily used in East Asian typography as a guide for pronounciation or to include other annotations. In Japanese, this form of typography is also known as furigana.

Categories:	Flow content. Phrasing content.
Contained By:	Where phrasing content is expected.
Content Model:	One or more groups of: phrasing content followed either by a single rt element, or an rp element, an rt element, and another rp element.

Attributes	DOM Interface
Global attributes	Uses HTMLElement.

4.3.6.23 The rt element

Start tag: required End tag: required

The rt element marks the ruby text component of a ruby annotation.

Categories:	None.
Contained By:	As a child of a ruby element.
Content Model:	Phrasing content.

Attributes	DOM Interface
Global attributes	Uses HTMLElement.

4.3.6.24 The rp element

Start tag: required End tag: required

The rp element can be used to provide parentheses around a ruby text component of a ruby annotation, to be shown by user agents that don't support ruby annotations.

Categories:	None.
Contained By:	As a child of a ruby element, either immediately before or immediately after an rt element.
Content Model:	If the rp element is immediately after an rt element that is immediately preceded by another rp element: a single character from Unicode character class Pe. Otherwise: a single character from Unicode character class Ps.

Attributes	DOM Interface
Global attributes	Uses HTMLElement.

4.3.7 Edits

4.3.7.1 The ins element

Start tag: required End tag: required

The ins element represents an addition to the document.

Categories:	Flow content. When the element only contains phrasing content: phrasing content.
Contained By:	Where phrasing content is expected.
Content Model:	Transparent.

Attributes	DOM Interface

Global attributes cite datetime	Uses the HTMLModElement interface.

4.3.7.2 The del element

Start tag: required End tag: required

The del element represents a removal from the document.

Categories: Flow content.
When the element only contains phrasing content: phrasing content.

Contained By: Where phrasing content is expected.

Content Model: Transparent.

Attributes	DOM Interface
Global attributes cite datetime	Uses the HTMLModElement interface.

4.3.8 Embedded Content

4.3.8.1 The figure element

Start tag: required End tag: required

The figure element represents some flow content, optionally with a caption, which can be moved away from the main flow of the document without affecting the document's meaning.

Categories: Flow content.
Sectioning root.

Contained By: Where flow content is expected.

Content Model: Either: one legend element followed by flow content.
Or: Flow content followed by one legend element.
Or: Flow content.

Attributes	DOM Interface
Global attributes	Uses HTMLElement.

4.3.8.2 The img element

Start tag: required End tag: empty

An img element represents an image.

Categories: Flow content.
Phrasing content.
Embedded content.
If the element has an usemap attribute: Interactive content.

Contained By: Where embedded content is expected.

Content Model: Empty.

Attributes	DOM Interface

Global attributes
alt
src
usemap
ismap
width
height
	[NamedConstructor=Image(),
	NamedConstructor=Image(in unsigned long width),
	NamedConstructor=Image(in unsigned long width, in unsigned long height)]
	interface HTMLImageElement : HTMLElement {
	attribute DOMString alt;
	attribute DOMString src;
	attribute DOMString useMap;
	attribute boolean isMap;
	attribute unsigned long width;
	attribute unsigned long height;
	readonly attribute boolean complete;
	};

4.3.8.3 The iframe element

Start tag: required End tag: required

The iframe element introduces a new nested browsing context.

Categories: Flow content.
Phrasing content.
Embedded content.

Contained By: Where embedded content is expected.

Content Model: Text that conforms to the requirements given in the prose.

ATTRIBUTES

Global attributes
src
name
sandbox
seamless
width
height

DOM INTERFACE

```
interface HTMLIFrameElement : HTMLElement {
           attribute DOMString src;
           attribute DOMString name;
           attribute DOMString sandbox;
           attribute boolean seamless;
           attribute DOMString width;
           attribute DOMString height;
};
```

Objects implementing the HTMLIFrameElement interface must also implement the EmbeddingElement interface defined in the Window Object specification. [WINDOW]

4.3.8.4 The embed element

Start tag: required End tag: empty

The embed element represents an integration point for an external (typically non-HTML) application or interactive content.

Categories: Flow content.
Phrasing content.
Embedded content.

Contained By: Where embedded content is expected.

Content Model: Empty.

ATTRIBUTES

Global attributes
src
type
width
height
Any other attribute that has no namespace (see prose).

DOM INTERFACE

```
interface HTMLEmbedElement : HTMLElement {
           attribute DOMString src;
           attribute DOMString type;
           attribute DOMString width;
           attribute DOMString height;
};
```

Depending on the type of content instantiated by the embed element, the node may also support other interfaces.

4.3.8.5 The object element

Start tag: required End tag: required

The object element can represent an external resource, which, depending on the type of the resource, will either be treated as an image, as a nested browsing context, or as an external resource to be processed by a plugin.

Categories: Flow content.
Phrasing content.
Embedded content.
Listed, submittable, form-associated element.

Contained By: Where embedded content is expected.

Content Model: Zero or more param elements, then, transparent.

ATTRIBUTES

Global attributes
data
type

DOM INTERFACE

```
interface HTMLObjectElement : HTMLElement {
           attribute DOMString data;
           attribute DOMString type;
           attribute DOMString name;
```

name
usemap
form
width
height

```
            attribute DOMString useMap;
  readonly  attribute HTMLFormElement form;
            attribute DOMString width;
            attribute DOMString height;
};
```

Objects implementing the HTMLObjectElement interface must also implement the EmbeddingElement interface defined in the Window Object specification. [WINDOW]

Depending on the type of content instantiated by the object element, the node may also support other interfaces.

4.3.8.6 The param element

Start tag: required End tag: empty

The param element defines parameters for plugins invoked by object elements.

Categories: None.

Contained By: As a child of an object element, before any flow content.

Content Model: Empty.

ATTRIBUTES

Global attributes
name
value

DOM INTERFACE

```
interface HTMLParamElement : HTMLElement {
            attribute DOMString name;
            attribute DOMString value;
};
```

4.3.8.7 The video element

Start tag: required End tag: required

A video element represents a video or movie.

Categories:
Flow content.
Phrasing content.
Embedded content.
If the element has a controls attribute: Interactive content.

Contained By: Where embedded content is expected.

Content Model:
If the element has a src attribute: transparent.
If the element does not have a src attribute: one or more source elements, then, transparent.

ATTRIBUTES

Global attributes
src
poster
autobuffer
autoplay
loop
controls
width
height

DOM INTERFACE

```
interface HTMLVideoElement : HTMLMediaElement {
            attribute DOMString width;
            attribute DOMString height;
  readonly  attribute unsigned long videoWidth;
  readonly  attribute unsigned long videoHeight;
            attribute DOMString poster;
};
```

4.3.8.8 The audio element

Start tag: required End tag: required

An audio element represents a sound or audio stream.

Categories:
Flow content.
Phrasing content.
Embedded content.
If the element has a controls attribute: Interactive content.

Contained By: Where embedded content is expected.

Content Model: If the element has a src attribute: transparent.
If the element does not have a src attribute: one or more source elements, then, transparent.

ATTRIBUTES

Global attributes
src
autobuffer
autoplay
loop
controls

DOM INTERFACE

```
[NamedConstructor=Audio(),
 NamedConstructor=Audio(in DOMString src)]
interface HTMLAudioElement : HTMLMediaElement {
  // no members
};
```

4.3.8.9 The source element

Start tag: required End tag: empty

The source element allows authors to specify multiple media resources for media elements.

Categories: None.

Contained By: As a child of a media element, before any flow content.

Content Model: Empty.

ATTRIBUTES

Global attributes
src
type
media

DOM INTERFACE

```
interface HTMLSourceElement : HTMLElement {
           attribute DOMString src;
           attribute DOMString type;
           attribute DOMString media;
};
```

4.3.8.10 The canvas element

Start tag: required End tag: required

The canvas element represents a resolution-dependent bitmap canvas, which can be used for rendering graphs, game graphics, or other visual images on the fly.

Categories: Flow content.
Phrasing content.
Embedded content.

Contained By: Where embedded content is expected.

Content Model: Transparent.

ATTRIBUTES

Global attributes
width
height

DOM INTERFACE

```
interface HTMLCanvasElement : HTMLElement {
           attribute unsigned long width;
           attribute unsigned long height;

  DOMString toDataURL([Optional] in DOMString type, [Variadic] in any args);

  Object getContext(in DOMString contextId);
};
```

4.3.8.11 The map element

Start tag: required End tag: required

The map element, in conjunction with any area element descendants, defines an image map.

Categories: Flow content.

Contained By: Where flow content is expected.

Content Model: Flow content.

ATTRIBUTES

Global attributes
name

DOM INTERFACE

```
interface HTMLMapElement : HTMLElement {
           attribute DOMString name;
  readonly attribute HTMLCollection areas;
  readonly attribute HTMLCollection images;
};
```

4.3.8.12 The area element

Start tag: required End tag: empty

The area element represents either a hyperlink with some text and a corresponding area on an image map, or a dead area on an image map.

Categories: Flow content. Phrasing content.

Contained By: Where phrasing content is expected, but only if there is a map element ancestor.

Content Model: Empty.

Attributes

Global attributes
alt
coords
shape
href
target
ping
rel
media
hreflang
type

DOM Interface

```
interface HTMLAreaElement : HTMLElement {
           attribute DOMString alt;
           attribute DOMString coords;
           attribute DOMString shape;
           attribute DOMString href;
           attribute DOMString target;
           attribute DOMString ping;
           attribute DOMString rel;
  readonly attribute DOMTokenList relList;
           attribute DOMString media;
           attribute DOMString hreflang;
           attribute DOMString type;
};
```

4.3.9 Tabular Data

4.3.9.1 The table element

Start tag: required End tag: required

The table element represents data with more than one dimension (a table).

Categories: Flow content.

Contained By: Where flow content is expected.

Content Model: In this order: optionally a caption element, followed by either zero or more colgroup elements, followed optionally by a thead element, followed optionally by a tfoot element, followed by either zero or more tbody elements or one or more tr elements, followed optionally by a tfoot element (but there can only be one tfoot element child in total).

Attributes

Global attributes

DOM Interface

```
interface HTMLTableElement : HTMLElement {
           attribute HTMLTableCaptionElement caption;
  HTMLElement createCaption();
  void deleteCaption();
           attribute HTMLTableSectionElement tHead;
  HTMLElement createTHead();
  void deleteTHead();
           attribute HTMLTableSectionElement tFoot;
  HTMLElement createTFoot();
  void deleteTFoot();
  readonly attribute HTMLCollection tBodies;
  HTMLElement createTBody();
  readonly attribute HTMLCollection rows;
  HTMLElement insertRow([Optional] in long index);
  void deleteRow(in long index);
};
```

4.3.9.2 The caption element

Start tag: required End tag: required

The caption element represents the title of the table that is its parent, if it has a parent and that is a table element.

Categories: None.

Contained By: As the first element child of a table element.

Content Model: Phrasing content.

Attributes	DOM Interface
Global attributes	Uses HTMLElement.

4.3.9.3 The colgroup element

Start tag: optional End tag: optional

The colgroup element represents a group of one or more columns in the table that is its parent, if it has a parent and that is a table element.

Categories: None.

Contained By: As a child of a table element, after any caption elements and before any thead, tbody, tfoot, and tr elements.

Content Model: Zero or more col elements.

Attributes	DOM Interface

Global attributes
span
	interface HTMLTableColElement : HTMLElement {
	attribute unsigned long span;
	};

4.3.9.4 The col element

Start tag: required End tag: empty

If a col element has a parent and that is a colgroup element that itself has a parent that is a table element, then the col element represents one or more columns in the column group represented by that colgroup.

Categories: None.

Contained By: As a child of a colgroup element that doesn't have a span attribute.

Content Model: Empty.

Attributes	DOM Interface
Global attributes span	HTMLTableColElement, same as for colgroup elements. This interface defines one member, span.

4.3.9.5 The tbody element

Start tag: optional End tag: optional

The tbody element represents a block of rows that consist of a body of data for the parent table element, if the tbody element has a parent and it is a table.

Categories: None.

Contained By: As a child of a table element, after any caption, colgroup, and thead elements, but only if there are no tr elements that are children of the table element.

Content Model: Zero or more tr elements

Attributes	DOM Interface

Global attributes	interface HTMLTableSectionElement : HTMLElement {
	readonly attribute HTMLCollection rows;
	HTMLElement insertRow([Optional] in long index);
	void deleteRow(in long index);
	};
	
	The HTMLTableSectionElement interface is also used for thead and tfoot elements.

4.3.9.6 The thead element

Start tag: optional End tag: optional

The thead element represents the block of rows that consist of the column labels (headers) for the parent table element, if the thead element has a parent and it is a table.

Categories: None.

Contained By:

As a child of a table element, after any caption, and colgroup elements and before any tbody, tfoot, and tr elements, but only if there are no other thead elements that are children of the table element.

Content Model: Zero or more tr elements

ATTRIBUTES

Global attributes

DOM INTERFACE

HTMLTableSectionElement, as defined for tbody elements.

4.3.9.7 The tfoot element

Start tag: optional End tag: optional

The tfoot element represents the block of rows that consist of the column summaries (footers) for the parent table element, if the tfoot element has a parent and it is a table.

Categories: None.

Contained By:
As a child of a table element, after any caption, colgroup, and thead elements and before any tbody and tr elements, but only if there are no other tfoot elements that are children of the table element.
As a child of a table element, after any caption, colgroup, thead, tbody, and tr elements, but only if there are no other tfoot elements that are children of the table element.

Content Model: Zero or more tr elements

ATTRIBUTES

Global attributes

DOM INTERFACE

HTMLTableSectionElement, as defined for tbody elements.

4.3.9.8 The tr element

Start tag: required End tag: optional

The tr element represents a row of cells in a table.

Categories: None.

Contained By:
As a child of a thead element.
As a child of a tbody element.
As a child of a tfoot element.
As a child of a table element, after any caption, colgroup, and thead elements, but only if there are no tbody elements that are children of the table element.

Content Model: Zero or more td or th elements

ATTRIBUTES

Global attributes

DOM INTERFACE

```
interface HTMLTableRowElement : HTMLElement {
  readonly attribute long rowIndex;
  readonly attribute long sectionRowIndex;
  readonly attribute HTMLCollection cells;
  HTMLElement insertCell([Optional] in long index);
  void deleteCell(in long index);
};
```

4.3.9.9 The td element

Start tag: required End tag: optional

The td element represents a data cell in a table.

Categories: Sectioning root.

Contained By: As a child of a tr element.

Content Model: Flow content.

ATTRIBUTES

Global attributes
colspan
rowspan
headers

DOM INTERFACE

```
interface HTMLTableDataCellElement : HTMLTableCellElement {};
```

4.3.9.10 The th element

Start tag: required End tag: optional

The th element represents a header cell in a table.

Categories: None.

Contained By: As a child of a tr element.

Content Model: Phrasing content.

ATTRIBUTES

Global attributes
colspan
rowspan
headers
scope

DOM INTERFACE

```
interface HTMLTableHeaderCellElement : HTMLTableCellElement {
          attribute DOMString scope;
};
```

4.3.10 Forms

4.3.10.1 The form element

Start tag: required End tag: required

The form element represents a collection of form-associated elements, some of which can represent editable values that can be submitted to a server for processing.

Categories: Flow content.

Contained By: Where flow content is expected.

Content Model: Flow content, but with no form element descendants.

ATTRIBUTES

Global attributes
accept-charset
action
autocomplete
enctype
method
name
novalidate
target

DOM INTERFACE

```
[Callable=namedItem]
interface HTMLFormElement : HTMLElement {
           attribute DOMString acceptCharset;
           attribute DOMString action;
           attribute boolean autocomplete;
           attribute DOMString enctype;
           attribute DOMString method;
           attribute DOMString name;
           attribute boolean novalidate;
           attribute DOMString target;

  readonly attribute HTMLFormControlsCollection elements;
  readonly attribute long length;
  [IndexGetter] any item(in DOMString name);
  [NameGetter=OverrideBuiltins] any namedItem(in DOMString name);

  void submit();
  void reset();
  boolean checkValidity();

  void dispatchFormInput();
  void dispatchFormChange();
};
```

4.3.10.2 The fieldset element

Start tag: required End tag: required

The fieldset element represents a set of form controls grouped under a common name.

Categories: Flow content.
Listed form-associated element.

Contained By: Where flow content is expected.

Content Model: One legend element follwed by flow content.

ATTRIBUTES

Global attributes
disabled
form
name

DOM INTERFACE

```
interface HTMLFieldSetElement : HTMLElement {
           attribute boolean disabled;
  readonly attribute HTMLFormElement form;
           attribute DOMString name;
```

```
    readonly attribute DOMString type;

    readonly attribute HTMLFormControlsCollection elements;

    readonly attribute boolean willValidate;
    readonly attribute ValidityState validity;
    readonly attribute DOMString validationMessage;
    boolean checkValidity();
    void setCustomValidity(in DOMString error);
  };
```

4.3.10.3 The label element

Start tag: required End tag: required

The label represents a caption in a user interface. The caption can be associated with a specific form control, known as the label element's labeled control.

Categories: Flow content. Phrasing content. Interactive content. Form-associated element.

Contained By: Where phrasing content is expected.

Content Model: Phrasing content, but with no descendant labelable form-associated elements unless it is the element's labeled control, and no descendant label elements.

Attributes

Global attributes
form
for

DOM Interface

```
interface HTMLLabelElement : HTMLElement {
  readonly attribute HTMLFormElement form;
           attribute DOMString htmlFor;
  readonly attribute HTMLElement control;
};
```

4.3.10.4 The input element

Start tag: required End tag: empty

The input element represents a typed data field, usually with a form control to allow the user to edit the data.

Categories: Flow content. Phrasing content. If the type attribute is not in the Hidden state: Interactive content. Listed, labelable, submittable, and resettable form-associated element.

Contained By: Where phrasing content is expected.

Content Model: Empty.

Attributes

Global attributes
accept
action
alt
autocomplete
autofocus
checked
disabled
enctype
form
height
list
max
maxlength
method
min
multiple
name
novalidate
pattern
placeholder

DOM Interface

```
interface HTMLInputElement : HTMLElement {
           attribute DOMString accept;
           attribute DOMString action;
           attribute DOMString alt;
           attribute boolean autocomplete;
           attribute boolean autofocus;
           attribute boolean defaultChecked;
           attribute boolean checked;
           attribute boolean disabled;
           attribute DOMString enctype;
  readonly attribute HTMLFormElement form;
           attribute DOMString height;
           attribute boolean indeterminate;
  readonly attribute HTMLElement list;
           attribute DOMString max;
           attribute long maxLength;
           attribute DOMString method;
           attribute DOMString min;
           attribute boolean multiple;
           attribute DOMString name;
           attribute boolean noValidate;
           attribute DOMString pattern;
           attribute DOMString placeholder;
           attribute boolean readOnly;
           attribute boolean required;
           attribute unsigned long size;
           attribute DOMString src;
           attribute DOMString step;
           attribute DOMString target;
```

readonly
required
size
src
step
target
type
value
width

```
            attribute DOMString type;
            attribute DOMString defaultValue;
            attribute DOMString value;
            attribute Date valueAsDate;
            attribute float valueAsNumber;
  readonly attribute HTMLOptionElement selectedOption;
            attribute DOMString width;

  void stepUp(in long n);
  void stepDown(in long n);

  readonly attribute boolean willValidate;
  readonly attribute ValidityState validity;
  readonly attribute DOMString validationMessage;
  boolean checkValidity();
  void setCustomValidity(in DOMString error);

  readonly attribute NodeList labels;

  void select();
            attribute unsigned long selectionStart;
            attribute unsigned long selectionEnd;
  void setSelectionRange(in unsigned long start, in unsigned long end);
};
```

4.3.10.5 The button element

Start tag: required End tag: required

The button element represents a button. If the element is not disabled, then the user agent should allow the user to activate the button.

Categories: Flow content.
Phrasing content.
Interactive content.
Listed, labelable, and submittable form-associated element.

Contained By: Where phrasing content is expected.

Content Model: Phrasing content, but there must be no interactive content descendant.

ATTRIBUTES

Global attributes
action
autofocus
disabled
enctype
form
method
name
novalidate
target
type
value

DOM INTERFACE

```
interface HTMLButtonElement : HTMLElement {
           attribute DOMString action;
           attribute boolean autofocus;
           attribute boolean disabled;
           attribute DOMString enctype;
  readonly attribute HTMLFormElement form;
           attribute DOMString method;
           attribute DOMString name;
           attribute DOMString noValidate;
           attribute DOMString target;
           attribute DOMString type;
           attribute DOMString value;

  readonly attribute boolean willValidate;
  readonly attribute ValidityState validity;
  readonly attribute DOMString validationMessage;
  boolean checkValidity();
  void setCustomValidity(in DOMString error);

  readonly attribute NodeList labels;
};
```

4.3.10.6 The select element

Start tag: required End tag: required

The select element represents a control for selecting amongst a set of options.

Categories: Flow content.
Phrasing content.
Interactive content.
Listed, labelable, submittable, and resettable form-associated element.

Contained By: Where phrasing content is expected.

Content Model: Zero or more option or optgroup elements.

ATTRIBUTES

Global attributes

DOM INTERFACE

```
[Callable=namedItem]
interface HTMLSelectElement : HTMLElement {
```

autofocus
disabled
form
multiple
name
size

```
          attribute boolean autofocus;
          attribute boolean disabled;
  readonly attribute HTMLFormElement form;
          attribute boolean multiple;
          attribute DOMString name;
          attribute boolean size;

  readonly attribute DOMString type;

  readonly attribute HTMLOptionsCollection options;
          attribute unsigned long length;
  [IndexGetter] any item(in DOMString name);
  [NameGetter] any namedItem(in DOMString name);
  void add(in HTMLElement element, in HTMLElement before);
  void add(in HTMLElement element, in long before);
  void remove(in long index);

  readonly attribute HTMLCollection selectedOptions;
          attribute long selectedIndex;
          attribute DOMString value;

  readonly attribute boolean willValidate;
  readonly attribute ValidityState validity;
  readonly attribute DOMString validationMessage;
  boolean checkValidity();
  void setCustomValidity(in DOMString error);

  readonly attribute NodeList labels;
};
```

4.3.10.7 The datalist element

Start tag: required End tag: required

The datalist element represents a set of option elements that represent predefined options for other controls. The contents of the element represents fallback content for legacy user agents, intermixed with option elements that represent the predefined options. In the rendering, the datalist element represents nothing and it, along with its children, should be hidden.

Categories: Flow content. Phrasing content.

Contained By: Where phrasing content is expected.

Content Model: Either: phrasing content. Or: Zero or more option elements.

ATTRIBUTES

Global attributes

DOM INTERFACE

```
interface HTMLDataListElement : HTMLElement {
  readonly attribute HTMLCollection options;
};
```

4.3.10.8 The optgroup element

Start tag: required End tag: optional

The optgroup element represents a group of option elements with a common label.

Categories: None.

Contained By: As a child of a select element.

Content Model: Zero or more option elements.

ATTRIBUTES

Global attributes
disabled
label

DOM INTERFACE

```
interface HTMLOptGroupElement : HTMLElement {
          attribute boolean disabled;
          attribute DOMString label;
};
```

4.3.10.9 The option element

Start tag: required End tag: optional

The option element represents an option in a select element or as part of a list of suggestions in a datalist element.

Categories: None.

Contained By: As a child of a select element.

As a child of a datalist element.
As a child of an optgroup element.

Content Model: Text.

ATTRIBUTES

Global attributes
disabled
label
selected
value

DOM INTERFACE

```
[NamedConstructor=Option(),
 NamedConstructor=Option(in DOMString text),
 NamedConstructor=Option(in DOMString text, in DOMString value),
 NamedConstructor=Option(in DOMString text, in DOMString value, in boolean defaultSelected),
 NamedConstructor=Option(in DOMString text, in DOMString value, in boolean defaultSelected, in boolean selected)]
interface HTMLOptionElement : HTMLElement {
           attribute boolean disabled;
  readonly attribute HTMLFormElement form;
           attribute DOMString label;
           attribute boolean defaultSelected;
           attribute boolean selected;
           attribute DOMString value;

  readonly attribute DOMString text;
  readonly attribute long index;
};
```

4.3.10.10 The textarea element

Start tag: required End tag: required

The textarea element represents a multiline plain text edit control for the element's raw value. The contents of the control represent the control's default value.

Categories: Flow content.
Phrasing content.
Interactive content.
Listed, labelable, submittable, and resettable form-associated element.

Contained By: Where phrasing content is expected.

Content Model: Text.

ATTRIBUTES

Global attributes
autofocus
cols
disabled
form
maxlength
name
readonly
required
rows
wrap

DOM INTERFACE

```
interface HTMLTextAreaElement : HTMLElement {
           attribute boolean autofocus;
           attribute unsigned long cols;
           attribute boolean disabled;
  readonly attribute HTMLFormElement form;
           attribute long maxLength;
           attribute DOMString name;
           attribute boolean readOnly;
           attribute boolean required;
           attribute unsigned long rows;
           attribute DOMString wrap;

  readonly attribute DOMString type;
           attribute DOMString defaultValue;
           attribute DOMString value;

  readonly attribute boolean willValidate;
  readonly attribute ValidityState validity;
  readonly attribute DOMString validationMessage;
  boolean checkValidity();
  void setCustomValidity(in DOMString error);

  readonly attribute NodeList labels;

  void select();
           attribute unsigned long selectionStart;
           attribute unsigned long selectionEnd;
  void setSelectionRange(in unsigned long start, in unsigned long end);
};
```

4.3.10.11 The output element

Start tag: required End tag: required

The output element represents the result of a calculation.

Categories: Flow content.
Phrasing content.
Listed and resettable form-associated element.

Contained By: Where phrasing content is expected.

Content Model: Phrasing content.

Attributes

Global attributes
for
form
name

DOM Interface

```
interface HTMLOutputElement : HTMLElement {
           attribute DOMString htmlFor;
  readonly attribute HTMLFormElement form;
           attribute DOMString name;

  readonly attribute DOMString type;
           attribute DOMString defaultValue;
           attribute DOMString value;

  readonly attribute boolean willValidate;
  readonly attribute ValidityState validity;
  readonly attribute DOMString validationMessage;
  boolean checkValidity();
  void setCustomValidity(in DOMString error);
};
```

4.3.11 Interactive Elements

4.3.11.1 The details element

Start tag: required End tag: required

The details element represents additional information or controls which the user can obtain on demand.

Categories: Flow content.
Interactive content.

Contained By: Where flow content is expected.

Content Model: One legend element followed by flow content.

Attributes

Global attributes
open

DOM Interface

```
interface HTMLDetailsElement : HTMLElement {
           attribute boolean open;
};
```

4.3.11.2 The command element

Start tag: required End tag: empty

The command element represents a command that the user can invoke.

Categories: Metadata content.
Flow content.
Phrasing content.

Contained By: Where metadata content is expected.
Where phrasing content is expected.

Content Model: Empty.

Attributes

Global attributes
type
label
icon
disabled
checked
radiogroup
default

Also, the title attribute has special semantics on this element.

DOM Interface

```
interface HTMLCommandElement : HTMLElement {
           attribute DOMString type;
           attribute DOMString label;
           attribute DOMString icon;
           attribute boolean disabled;
           attribute boolean checked;
           attribute DOMString radiogroup;
           attribute boolean default;
 void click(); // shadows HTMLElement.click()
};
```

The Command interface must also be implemented by this element.

4.3.11.3 The bb element

Start tag: required End tag: required

The bb element represents a user agent command that the user can invoke.

Categories: Flow content.

Phrasing content.
Interactive content.

Contained By: Where phrasing content is expected.

Content Model: Phrasing content, but there must be no interactive content descendant.

ATTRIBUTES

Global attributes
type

DOM INTERFACE

```
interface HTMLBrowserButtonElement : HTMLElement {
          attribute DOMString type;
  readonly attribute boolean supported;
  readonly attribute boolean disabled;
};
```

The Command interface must also be implemented by this element.

4.3.11.4 The menu element

Start tag: required End tag: required

The menu element represents a list of commands.

Categories: Flow content.
If the element's type attribute is in the tool bar state: Interactive content.

Contained By: Where flow content is expected.

Content Model: Either: Zero or more li elements.
Or: Flow content.

ATTRIBUTES

Global attributes
type
label

DOM INTERFACE

```
interface HTMLMenuElement : HTMLElement {
          attribute DOMString type;
          attribute DOMString label;
};
```

4.3.12 Miscellaneous Elements

4.3.12.1 The legend element

Start tag: required End tag: required

The legend element represents a title or explanatory caption for the rest of the contents of the legend element's parent element.

Categories: None.

Contained By: As the first child of a fieldset element.
As the first child of a details element.
As a child of a figure element, if there are no other legend element children of that element.

Content Model: Phrasing content.

ATTRIBUTES

Global attributes

DOM INTERFACE

```
interface HTMLLegendElement : HTMLElement {
  readonly attribute HTMLFormElement form;
};
```

4.3.12.2 The div element

Start tag: required End tag: required

The div element represents nothing at all. It can be used with the class, lang/xml:lang, and title attributes to mark up semantics common to a group of consecutive elements.

Categories: Flow content.

Contained By: Where flow content is expected.

Content Model: Flow content.

ATTRIBUTES

DOM INTERFACE

Global attributes	Uses HTMLElement.

4.4 Microdata

...

5 Index of Elements

5.1 Conforming Elements

Element	Start Tag	End Tag	Short Description	Notes
a	required	required	Hyperlink	
abbr	required	required	Abbreviation	
address	required	required	Contact information	
area	required	empty	Image map region	
article	required	required	Independent section	
aside	required	required	Auxiliary section	
audio	required	required	Audio stream	
b	required	required	Bold text	
base	required	empty	Document base URI	
bb	required	required	Browser button	
bdo	required	required	Bi-directional text override	
blockquote	required	required	Long quotation	
body	optional	optional	Main content	
br	required	empty	Line break	
button	required	required	Push button control	
canvas	required	required	Bitmap canvas	
caption	required	required	Table caption	
cite	required	required	Citation	
code	required	required	Code fragment	
col	required	empty	Table column	
colgroup	required	optional	Table column group	
command	required	empty	Command that a user can invoke	
datagrid	required	required	Interactive tree, list or tabular data	
datalist	required	required	Predefined control values	
dd	required	optional	Description description	
del	required	required	Deletion	
details	required	required	Additional information	
dfn	required	required	Defining instance of a term	
dialog	required	required	Conversation	
div	required	required	Generic division	
dl	required	required	Description list	
dt	required	optional	Description term	
em	required	required	Stress emphasis	
embed	required	empty	Embedded application	
fieldset	required	required	Form control group	
figure	required	required	A figure with a caption.	
footer	required	required	Section footer	
form	required	required	Form	
h1	required	required	Heading level 1	The heading level is also affected by sectioning elements
h2	required	required	Heading level 2	
h3	required	required	Heading level 3	
h4	required	required	Heading level 4	
h5	required	required	Heading level 5	
h6	required	required	Heading level 6	
head	optional	optional	Document head	
header	required	required	Section header	

hr	required	empty	Separator
html	optional	optional	Document root
i	required	required	Italic text
iframe	required	required	Inline frame
img	required	empty	Image
input	required	empty	Form control
ins	required	required	Insertion
kbd	required	required	User input
label	required	required	Form control label
legend	required	required	Explanatory title or caption
li	required	optional	List item
link	required	empty	Link to resources
map	required	required	Client-side image map
mark	required	required	Marked or highlighted text
menu	required	required	Command menu
meta	required	empty	Metadata
meter	required	required	Scalar measurement
nav	required	required	Navigation
noscript	required	required	Alternative content for no script support
object	required	required	Generic embedded resource
ol	required	required	Ordered list
optgroup	required	optional	Option group
option	required	optional	Selection choice
output	required	required	Output control
p	required	optional	Paragraph
param	required	empty	Plugin parameter
pre	required	required	Preformatted text
progress	required	required	Progress of a task
q	required	required	Inline quotation
rp	required	required	Ruby parenthesis
rt	required	required	Ruby text
ruby	required	required	Ruby annotation
samp	required	required	Sample output
script	required	required	Linked or embedded script
section	required	required	Document section
select	required	required	Selection control
small	required	required	Small print
source	required	empty	Media resource
span	required	required	Generic inline container
strong	required	required	Strong importance
style	required	required	Embedded stylesheet
sub	required	required	Subscript
sup	required	required	Superscript
table	required	required	Table
tbody	optional	optional	Table body
td	required	optional	Table cell
textarea	required	required	Multi-line text control
tfoot	optional	optional	Table footer
th	required	optional	Table header cell
thead	optional	optional	Table head
time	required	required	Date and/or time
title	required	required	Document title
tr	required	optional	Table row
ul	required	required	Unordered list
var	required	required	Variable
video	required	required	Video or movie

5.2 Obsolete Elements

These elements are obsolete and should not be used by authors. However, they are documented here because they are supported by browsers, along with notes about conforming alternatives that may be used instead.

This list may be incomplete. Please report any missing elements.

Element	Start Tag	End Tag	Short Description	Notes
acronym	required	required	Acronym	Use the abbr element
applet	required	required	Java applet	Use the object element.
basefont	required	empty	Base font style	This has limited support in browsers. Use CSS instead.
bgsound	required	empty		Use the audio element.
big	required	required		Use a semantically appropriate element with CSS for style.
blink	required	required		CSS provides an alternative with limited browser support, but note that blinking text is annoying.
center	required	required		Use a semantically appropriate element with CSS for style.
dir	required	required		Use the ul element.
font	required	required	Font style	Use a semantically appropriate element with CSS for style.
frame	required	required		Consider using CSS layouts or the iframe element.
frameset	required	required		Consider using CSS layouts or the iframe element.
isindex	required	required		Use a form with a text input and submit button.
listing	required	required	Preformatted text	Use the pre element.
marquee	required	required		Scripting or CSS animations can be used to simulate scrolling text.
nobr	required	required		Use a semantically appropriate element with CSS for style.
noembed	required	required		
noframes	required	required		
plaintext	required	required	Preformatted text	Use the pre element.
s	required	required		Consider using the del element, if appropriate, or another semantically appropriate element with CSS for style.
spacer	required	required		Use CSS layout techniques.
strike	required	required		Consider using the del element, if appropriate, or another semantically appropriate element with CSS for style.
tt	required	required	Teletype	Consider using the code element, if appropriate, or another semantically appropriate element with CSS for style.
u	required	required		Use a semantically appropriate element with CSS for style.
wbr	required	empty		
xmp	required	required	Preformatted text	Use the pre element.

5.3 Comparison of HTML 4.01 and HTML5 Elements

Element	HTML 4.01/XHTML 1.0	HTML5	Short Description
a	strict	yes	Hyperlink
abbr	strict	yes	Abbreviation
acronym	strict	-	Acronym
address	strict	yes	Contact information
applet	transitional	-	Java applet
area	strict	yes	Image map region
article	-	yes	Independent section
aside	-	yes	Auxiliary section
audio	-	yes	Audio stream
b	strict	yes	Bold text
base	strict	yes	Document base URI
basefont	transitional	-	Base font style
bb	-	yes	Browser button
bdo	strict	yes	Bi-directional text override
bgsound	-	-	
big	strict	-	

blink	-	-	
blockquote	strict	yes	Long quotation
body	strict	yes	Main content
br	strict	yes	Line break
button	strict	yes	Push button control
canvas	-	yes	Bitmap canvas
caption	strict	yes	Table caption
center	transitional	-	
cite	strict	yes	Citation
code	strict	yes	Code fragment
col	strict	yes	Table column
colgroup	strict	yes	Table column group
command	-	yes	Command that a user can invoke
datagrid	-	yes	Interactive tree, list or tabular data
datalist	-	yes	Predefined control values
dd	strict	yes	Description description
del	strict	yes	Deletion
details	-	yes	Additional information
dfn	strict	yes	Defining instance of a term
dialog	-	yes	Conversation
dir	transitional	-	
div	strict	yes	Generic division
dl	strict	yes	Description list
dt	strict	yes	Description term
em	strict	yes	Stress emphasis
embed	-	yes	Embedded application
fieldset	strict	yes	Form control group
figure	-	yes	A figure with a caption.
font	transitional	-	Font style
footer	-	yes	Section footer
form	strict	yes	Form
frame	frameset	-	
frameset	frameset	-	
h1	strict	yes	Heading level 1
h2	strict	yes	Heading level 2
h3	strict	yes	Heading level 3
h4	strict	yes	Heading level 4
h5	strict	yes	Heading level 5
h6	strict	yes	Heading level 6
head	strict	yes	Document head
header	-	yes	Section header
hr	strict	yes	Separator
html	strict	yes	Document root
i	strict	yes	Italic text
iframe	transitional	yes	Inline frame
img	strict	yes	Image
input	strict	yes	Form control
ins	strict	yes	Insertion
isindex	transitional	-	
kbd	strict	yes	User input
label	strict	yes	Form control label
legend	strict	yes	Explanatory title or caption
li	strict	yes	List item
link	strict	yes	Link to resources
listing	-	-	Preformatted text
map	strict	yes	Client-side image map
mark	-	yes	Marked or highlighted text
[illegible]	-	-	

marquee	–	–	
menu	transitional	yes	Command menu
meta	strict	yes	Metadata
meter	–	yes	Scalar measurement
nav	–	yes	Navigation
nobr	–	–	
noembed	–	–	
noframes	frameset	–	
noscript	strict	yes	Alternative content for no script support
object	strict	yes	Generic embedded resource
ol	strict	yes	Ordered list
optgroup	strict	yes	Option group
option	strict	yes	Selection choice
output	–	yes	Output control
p	strict	yes	Paragraph
param	strict	yes	Plugin parameter
plaintext	–	–	Preformatted text
pre	strict	yes	Preformatted text
progress	–	yes	Progress of a task
q	strict	yes	Inline quotation
rp	–	yes	Ruby parenthesis
rt	–	yes	Ruby text
ruby	–	yes	Ruby annotation
s	transitional	–	
samp	strict	yes	Sample output
script	strict	yes	Linked or embedded script
section	–	yes	Document section
select	strict	yes	Selection control
small	strict	yes	Small print
source	–	yes	Media resource
spacer	–	–	
span	strict	yes	Generic inline container
strike	transitional	–	
strong	strict	yes	Strong importance
style	strict	yes	Embedded stylesheet
sub	strict	yes	Subscript
sup	strict	yes	Superscript
table	strict	yes	Table
tbody	strict	yes	Table body
td	strict	yes	Table cell
textarea	strict	yes	Multi-line text control
tfoot	strict	yes	Table footer
th	strict	yes	Table header cell
thead	strict	yes	Table head
time	–	yes	Date and/or time
title	strict	yes	Document title
tr	strict	yes	Table row
u	transitional	–	
ul	strict	yes	Unordered list
var	strict	yes	Variable
video	–	yes	Video or movie
wbr	–	–	
xmp	–	–	Preformatted text

6 How to Read This Guide

This section needs major revision and may be dropped.

6.1 Conventions

To ease readability and improve understanding, this document uses a number of conventions.

6.1.1 Notes, Tips and Warnings

Notes are used throughout this document to provide additional information. Tips are used to provide useful hints and suggestions. Warnings are used to point out common authoring errors and highlight important issues to be aware of.

[Need to provide examples of these]

6.1.2 Example Markup

Example markup is provided for both HTML and XHTML. In some cases, the markup is the same and thus only one example is needed, but in others there may be differences syntactic differences. Where HTML and XHTML differ, separate examples are given with each one clearly labelled.

HTML Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>HTML Example</title>
</head>
<body>
  <p>This is a sample HTML document.
</body>
</html>
```

XHTML Example:

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
  <title>XHTML Example</title>
</head>
<body>
  <p>This is a sample XHTML document.</p>
</body>
</html>
```

Sometimes, erroneous examples are included. This is usually done to illustrate common authoring errors, bad practices and other issues to be cautious of.

Erroneous Example:

```
<p>This markup contains a <em><strong>mistake</em></strong></p>
```

6.1.2.1 Attributes

Unless explicitly stated otherwise for a specific purpose, all attribute values in examples are quoted using double quotes. In HTML examples, boolean attributes are written in their minimised form and in XHTML examples, they are written in expanded form.

HTML Example:

```
<input type="checkbox" checked>
```

XHTML Example:

```
<input type="checkbox" checked="checked"/>
```

6.1.2.2 Void Elements

In XHTML examples, due to the XML Well-Formedness requirements, void elements are always marked up using the trailing slash.

XHTML Example:

```
<img src="image.png" alt="example"/>
```

In HTML, however, the trailing slash is optional and, unless explicitly stated otherwise, is always omitted.

HTML Example:

```
<img src="image.png" alt="example">
```

6.1.2.3 Namespaces

Some XHTML examples make use of XML namespaces. In such cases, the following prefixes are assumed to be defined even if there is no `xmlns` attributes in the fragment of code.

xml
http://www.w3.org/XML/1998/namespace
html
http://www.w3.org/1999/xhtml
math
http://www.w3.org/1998/Math/MathML
svg
http://www.w3.org/2000/svg

XHTML Example:

```
<html xml:lang="en">
...
</html>
```

XHTML Example:

```
<div>
<svg:svg><svg:circle r="50" cx="50" cy="50" fill="green"/></svg:svg>
</div>
```