

HTML/XHTML FAQ

The Flask documentation and example applications are using HTML5. You may notice that in many situations, when end tags are optional they are not used, so that the HTML is cleaner and faster to load. Because there is much confusion about HTML and XHTML among developers, this document tries to answer some of the major questions.

History of XHTML

For a while, it appeared that HTML was about to be replaced by XHTML. However, barely any websites on the Internet are actual XHTML (which is HTML processed using XML rules). There are a couple of major reasons why this is the case. One of them is Internet Explorer's lack of proper XHTML support. The XHTML spec states that XHTML must be served with the MIME type *application/xhtml+xml*, but Internet Explorer refuses to read files with that MIME type. While it is relatively easy to configure Web servers to serve XHTML properly, few people do. This is likely because properly using XHTML can be quite painful.

One of the most important causes of pain is XML's draconian (strict and ruthless) error handling. When an XML parsing error is encountered, the browser is supposed to show the user an ugly error message, instead of attempting to recover from the error and display what it can. Most of the (X)HTML generation on the web is based on non-XML template engines (such as Jinja, the one used in Flask) which do not protect you from accidentally creating invalid XHTML. There are XML based template engines, such as Kid and the popular Genshi, but they often come with a larger runtime overhead and are not as straightforward to use because they have to obey XML rules.

The majority of users, however, assumed they were properly using XHTML. They wrote an XHTML doctype at the top of the document and self-closed all the necessary tags (`
` becomes `
` or `
</br>` in XHTML). However, even if the document properly validates as XHTML, what really determines XHTML/HTML processing in browsers is the MIME type, which as said before is often not set properly. So the valid XHTML was being treated as invalid HTML.

XHTML also changed the way JavaScript is used. To properly work with XHTML, programmers have to use the namespaced DOM interface with the XHTML namespace to query for HTML elements.

History of HTML5

Development of the HTML5 specification was started in 2004 under the name “Web Applications 1.0” by the Web Hypertext Application Technology Working Group, or WHATWG (which was formed by the major browser vendors Apple, Mozilla, and Opera) with the goal of writing a new and improved HTML specification, based on existing browser behavior instead of unrealistic and backwards-incompatible specifications.


For example, in HTML4 `<title/Hello/` theoretically parses exactly the same as `<title>Hello</title>`. However, since people were using XHTML-like tags along the lines of `<link />`, browser vendors implemented the XHTML syntax over the syntax defined by the specification.

In 2007, the specification was adopted as the basis of a new HTML specification under the umbrella of the W3C, known as HTML5. Currently, it appears that XHTML is losing traction, as the XHTML 2 working group has been disbanded and HTML5 is being implemented by all major browser vendors.

HTML versus XHTML

The following table gives you a quick overview of features available in HTML 4.01, XHTML 1.1 and HTML5. (XHTML 1.0 is not included, as it was superseded by XHTML 1.1 and the barely-used XHTML5.)

	HTM-L4.01	XHTM-L1.1	HTML5
<code><tag/value/ == <tag>value</tag></code>	✓ [1]	✗	✗
<code>
</code> supported	✗	✓	✓ [2]
<code><script/></code> supported	✗	✓	✗
should be served as <i>text/html</i>	✓	✗ [3]	✓
should be served as <i>application/xhtml+xml</i>	✗	✓	✗
strict error handling	✗	✓	✗
inline SVG	✗	✓	✓
inline MathML	✗	✓	✓
<code><video></code> tag	✗	✗	✓
<code><audio></code> tag	✗	✗	✓
New semantic tags like <code><article></code>	✗	✗	✓

 v: 1.1.x ▼

[\[1\]](#) This is an obscure feature inherited from SGML. It is usually not supported by browsers, for reasons detailed above.

[2] This is for compatibility with server code that generates XHTML for tags such as `
`. It should not be used in new code.

[3] XHTML 1.0 is the last XHTML standard that allows to be served as *text/html* for backwards compatibility reasons.

What does “strict” mean?

HTML5 has strictly defined parsing rules, but it also specifies exactly how a browser should react to parsing errors - unlike XHTML, which simply states parsing should abort. Some people are confused by apparently invalid syntax that still generates the expected results (for example, missing end tags or unquoted attribute values).

Some of these work because of the lenient error handling most browsers use when they encounter a markup error, others are actually specified. The following constructs are optional in HTML5 by standard, but have to be supported by browsers:

- Wrapping the document in an `<html>` tag
- Wrapping header elements in `<head>` or the body elements in `<body>`
- Closing the `<p>`, ``, `<dt>`, `<dd>`, `<tr>`, `<td>`, `<th>`, `<tbody>`, `<thead>`, or `<tfoot>` tags.
- Quoting attributes, so long as they contain no whitespace or special characters (like `<`, `>`, `'`, or `"`).
- Requiring boolean attributes to have a value.

This means the following page in HTML5 is perfectly valid:

```
<!doctype html>
<title>Hello HTML5</title>
<div class=header>
  <h1>Hello HTML5</h1>
  <p class=tagline>HTML5 is awesome
</div>
<ul class=nav>
  <li><a href=/index>Index</a>
  <li><a href=/downloads>Downloads</a>
  <li><a href=/about>About</a>
</ul>
<div class=body>
  <h2>HTML5 is probably the future</h2>
  <p>
    There might be some other things around but in terms of
    browser vendor support, HTML5 is hard to beat.
  <dl>
    <dt>Key 1
```

```
<dd>Value 1
<dt>Key 2
<dd>Value 2
</dl>
</div>
```

New technologies in HTML5

HTML5 adds many new features that make Web applications easier to write and to use.

- The `<audio>` and `<video>` tags provide a way to embed audio and video without complicated add-ons like QuickTime or Flash.
- Semantic elements like `<article>`, `<header>`, `<nav>`, and `<time>` that make content easier to understand.
- The `<canvas>` tag, which supports a powerful drawing API, reducing the need for server-generated images to present data graphically.
- New form control types like `<input type="date">` that allow user agents to make entering and validating values easier.
- Advanced JavaScript APIs like Web Storage, Web Workers, Web Sockets, geolocation, and offline applications.

Many other features have been added, as well. A good guide to new features in HTML5 is Mark Pilgrim's soon-to-be-published book, [Dive Into HTML5](#). Not all of them are supported in browsers yet, however, so use caution.

What should be used?

Currently, the answer is HTML5. There are very few reasons to use XHTML considering the latest developments in Web browsers. To summarize the reasons given above:

- Internet Explorer (which, sadly, currently leads in market share) has poor support for XHTML.
- Many JavaScript libraries also do not support XHTML, due to the more complicated namespacing API it requires.
- HTML5 adds several new features, including semantic tags and the long-awaited `<audio>` and `<video>` tags.
- It has the support of most browser vendors behind it.
- It is much easier to write, and more compact.

For most applications, it is undoubtedly better to use HTML5 than XHTML.