



Documentación recogida de la traducción realizada por Sidar.org de la especificación XHTML del W3C que se encuentra en:

<http://www.sidar.org/recur/desdi/traduc/es/xhtml/xhtml11.htm>

Utilizada por Carixma para la realización del siguiente tutorial:

<http://www.carixma.com/tutoriales/xhtml.htm>

XHTML™ 1.0: El Lenguaje de Etiquetado Hipertextual Extensible

Una Reformulación de HTML 4.0 en XML 1.0

Traducción al castellano de la Recomendación del W3C con fecha 26 de enero de 2000

La única versión normativa oficial de este documento es la versión original (en inglés):

<http://www.w3.org/TR/2000/REC-xhtml1-20000126>

Versión anterior (en inglés):

<http://www.w3.org/TR/1999/PR-xhtml1-19991210>

Última versión:

<http://www.w3.org/TR/xhtml1>

([Versión Postscript](#), [Versión PDF](#), [Archivo comprimido ZIP](#), o [Archivo TAR GZIP](#))

Autores:

Ver los [agradecimientos](#).

Traductores:

[Juan Luis Lara](#) (miembro del Grupo G3 de Seguimiento de Diseño de [SID@R](#)).

Información sobre los derechos de autor se pueden encontrar en [Copyright](#) © 2000.

Resumen

Esta especificación define XHTML 1.0, una reformulación de HTML 4.0 como aplicación XML 1.0, y tres definiciones de tipo de documentos (DTD) correspondientes a las usadas en HTML 4.0. El significado de los elementos y sus atributos se encuentran en la Recomendación del W3C para HTML 4.0 y conforman la base para una futura extensión de XHTML. La compatibilidad con aplicaciones de usuario HTML existentes es posible siguiendo un pequeño conjunto de reglas.



Estado de este documento

Esta sección describe el estado de este documento en el momento de su publicación. Otros documentos pueden dejarlo obsoleto. La última versión de esta serie de documentos se encuentra en la W3C.

Este documento ha sido revisado por los miembros del W3C y otras partes interesadas y ha sido avalado por el Director como una Recomendación del W3C. Es un documento estable y puede ser citado como material de referencia o referencia normativa en otro documento. El objetivo del W3C al hacer la recomendación es llamar la atención sobre la especificación y promover su máxima difusión. Esto potencia la funcionalidad e interoperabilidad en la web.

Este documento ha sido producido como parte de la [Actividad HTML del W3C](#). Las metas del [Grupo de Trabajo de HTML \(de acceso exclusivo a sus miembros\)](#) se discuten en la [carta del Grupo de Trabajo de HTML \(de acceso exclusivo a sus miembros\)](#).

Una lista actualizada de las Recomendaciones del W3C y otros documentos técnicos pueden encontrarse en <http://www.w3.org/TR>.

La discusión pública sobre características del HTML tiene lugar en la lista de correo www-html@w3.org ([también disponible está disponible una recopilación](#)).

Por favor, notifiquen los errores que encuentren en este documento a www-html-editor@w3.org.

Una lista de los errores de esta especificación puede encontrarse en <http://www.w3.org/2000/01/REC-xhtml1-20000126-errata> (en inglés).



Contenido

- 1. [¿Qué es XHTML?](#)
 - 1.1 [¿Qué es HTML 4.0?](#)
 - 1.2 [¿Qué es XML?](#)
 - 1.3 [¿Por qué es necesario XHTML?](#)
- 2. [Definiciones](#)
 - 2.1 [Terminología](#)
 - 2.2 [Conceptos Generales](#)
- 3. [Definición Normativa de XHTML 1.0](#)
 - 3.1 [Requisitos de Conformidad para Documentos](#)
 - 3.2 [Requisitos de Conformidad para Aplicaciones de Usuario](#)
- 4. [Diferencias con HTML 4.0](#)
- 5. [Compatibilidad](#)
 - 5.1 [Especificación del Tipo de Soporte de Internet](#)
- 6. [Futuras Líneas de Actuación](#)
 - 6.1 [Modularizar HTML](#)
 - 6.2 [Subconjuntos y Extensibilidad](#)
 - 6.3 [Perfiles de Documento](#)
- [Apéndice A. DTD](#)
- [Apéndice B. Incompatibilidades entre elementos](#)
- [Apéndice C. Directrices de compatibilidad con HTML](#)
- [Apéndice D. Agradecimientos](#)
- [Apéndice E. Referencias](#)

1. ¿Qué es XHTML?

XHTML es una familia de módulos y tipos de documentos que reproduce, engloba y extiende HTML 4.0 [\[HTML\]](#). Los tipos de documentos de la familia XHTML están basados en XML, y diseñados fundamentalmente para trabajar en conjunto con aplicaciones de usuario basados en XML. Los detalles de esta familia y su evolución se discuten en más profundidad en la sección de [Futuras Directrices](#).

XHTML 1.0 (esta especificación) es el primer tipo de documento de la familia XHTML. Es una reformulación de las tres definiciones de tipo de documento HTML 4.0 como aplicaciones de XML 1.0 [\[XML\]](#). Su finalidad es ser usado como lenguaje de contenidos que es a la vez conforme a XML y, si se siguen algunas sencillas [directrices](#), funciona en aplicaciones de usuario conformes con HTML 4.0. Los desarrolladores que migren aplicaciones hacia XHTML 1.0 apreciarán las siguientes mejoras:

- Los documentos XHTML son conformes a XML. Como tales, son fácilmente visualizados, editados y validados con herramientas XML estándar.
- Los documentos XHTML pueden escribirse para que funcionen igual o mejor que lo hacían antes tanto en las aplicaciones de usuario conformes a HTML 4.0 como en las nuevas aplicaciones conformes a XHTML 1.0.
- Los documentos XHTML pueden usar aplicaciones (p.e. scripts y applets) que se basen ya sea en el Modelo del Objeto Documento de HTML o XML [\[DOM\]](#).
- A medida que la familia XHTML evolucione, los documentos conformes a XHTML 1.0 estarán más preparados para interactuar dentro de y entre distintos entornos XHTML.

La familia XHTML es el siguiente paso en la evolución de Internet. Al migrar en este momento hacia XHTML, los desarrolladores de contenidos web entran en el mundo de XML con todos los beneficios que se esperan de él a la vez que se aseguran la compatibilidad con aplicaciones de usuario pasadas y futuras.



1.1 ¿Qué es HTML 4.0?

HTML 4.0 [\[HTML\]](#) es una aplicación SGML (Language de Etiquetado Generalizado Estándar) conforme al estándar internacional ISO 8879, y está ampliamente considerado como el lenguaje de publicación estándar de la World Wide Web.

SGML es un lenguaje para la descripción de lenguajes de etiquetado, particularmente aquellos usados en el intercambio electrónico, manejo y publicación de documentos. HTML es un ejemplo de un lenguaje definido en SGML.

SGML es utilizado desde mitad de los 80 y ha permanecido bastante estable. Gran parte de su estabilidad se la debe al hecho de que el lenguaje es a la vez flexible y rico en posibilidades. Esta flexibilidad tiene sin embargo su coste, el nivel de complejidad que ha inhibido su uso en diversos ámbitos como la World Wide Web.

HTML, tal y como fue concebido, era un lenguaje para el intercambio de documentos científicos y técnicos adaptado para su uso por no especialistas en tratamiento de documentos. HTML resolvió el problema de la complejidad de SGML sirviéndose de un reducido conjunto de etiquetas estructurales y semánticas apropiadas para la realización de documentos relativamente simples. Además de simplificar la estructura de los documentos, HTML soportaba el hipertexto. Las posibilidades de usar elementos multimedia fueron añadidas con posterioridad.

En un corto período de tiempo, HTML se hizo muy popular y rápidamente superó los propósitos para los que había sido creado. Desde los albores de HTML, ha habido una constante invención de nuevos elementos para ser usados dentro de HTML (como estándar) y para adaptar HTML a mercados verticales, altamente especializados. Esta plétora de nuevos elementos ha llevado a problemas de compatibilidad de los documentos en las distintas plataformas.

Dada la creciente heterogeneidad de programas y plataformas, está claro que la idoneidad del HTML 4.0 'clásico' para ser usado en dichas plataformas es más que limitado.

1.2 ¿Qué es XML?

XML[™] son las siglas de Lenguaje de Etiquetado Extensible, formándose la palabra como acrónimo de la expresión inglesa eXtensible Markup Language' [\[XML\]](#).

XML fue concebido como un medio para recobrar la potencia y flexibilidad de SGML sin que adquiriese su gran complejidad. A pesar de ser una forma restringida de SGML, XML conserva casi toda la potencia y riqueza de las características de SGML.

Aún manteniendo estas características, XML elimina las más complejas de SGML que hacían la creación y diseño de los programas apropiados difícil y costosa.

1.3 ¿Por qué es necesario XHTML?

Los beneficios de migrar hacia XHTML 1.0 se han descrito más arriba. Algunos de esos beneficios generales son:

- Los desarrolladores de aplicaciones de usuario y documentos descubren constantemente nuevas formas de expresar sus ideas usando nuevas etiquetas. En XML, es relativamente fácil añadir nuevos elementos así como atributos adicionales a dichos elementos o a los ya existentes. La familia XHTML está concebida para acomodar estas extensiones a través de módulos XHTML y técnicas para desarrollar nuevos módulos conformes a XHTML (tal y cómo se describirá en la futura especificación sobre Modularización XHTML). Estos módulos permitirán la combinación de las características



existentes con las nuevas al crear contenidos para la web así como al desarrollar nuevas aplicaciones de usuario.

- Constantemente se desarrollan nuevas formas de acceder a Internet. Algunas estimaciones indican que en el año 2002, un 75% de las peticiones de documentos que se visualicen en Internet se realizarán desde esas plataformas alternativas. La familia XHTML está concebida teniendo en mente la interoperabilidad con aplicaciones de usuario generales. A través de un nuevo mecanismo de especificación de documentos y aplicaciones de usuarios, los servidores, proxies, y aplicaciones de usuario finales podrán realizar una mejor transformación del contenido. Como objetivo final, será posible desarrollar contenido conforme a XHTML que sea utilizable por cualquier aplicación de usuario conforme a XHTML.

2. Definiciones

2.1 Terminología

Los siguientes términos se utilizan en esta especificación. Estos términos extienden las definiciones de [\[RFC2119\]](#) basándose en definiciones similares que aparecen en ISO/IEC 9945-1:1990 [\[POSIX.1\]](#):

Definido por la aplicación

Un valor o comportamiento se considera definido por la aplicación cuando esta última es la encargada de definir [y documentar] los requisitos correspondientes para la correcta construcción del documento.

Puede

Con respecto a las aplicaciones, la palabra "puede" debe entenderse como introductoria a una característica opcional que no es obligatoria para hacer conforme el documento a esta especificación pero que puede indicarse. Con respecto a los [Requisitos de Conformidad para Documentos](#), la palabra "puede" significa que la característica opcional que introduce no debe utilizarse. El término "opcional" tiene la misma definición que "puede".

Debe

En esta especificación la palabra "debe" denota un requisito obligatorio ya sea de la aplicación o de los Documentos XHTML Estrictamente Conformes con la especificación, dependiendo del contexto. El término "deberá" tiene el mismo sentido que "debe".

Reservado

Denota un valor o comportamiento que no está especificado y cuyo uso tampoco está permitido en Documentos Conformes ni debe ser aceptado por Aplicaciones de Usuario Conformes.

Debería

Con respecto a las aplicaciones, la palabra "debería" ha de interpretarse como una recomendación para la aplicación, pero no como un requisito. Con respecto a los documentos, este término ha de entenderse como una práctica de programación deseable en general y un requisito para Documentos XHTML Estrictamente Conformes en particular.

Admitido

Algunos recursos recogidos en esta especificación son opcionales. Si uno de ellos es admitido por alguna aplicación, se comporta como se indica en esta especificación.

No determinado

Cuando un valor o comportamiento no quedan determinados, la especificación no define requisitos de portabilidad para un recurso en una aplicación ni siquiera cuando deba analizar un documento que use dicho recurso. Un documento tal que requiera un comportamiento específico en dicha situación en vez de tolerar cualquier comportamiento al usar el recurso, no será un Documento XHTML Estrictamente Conforme.



2.2 Conceptos Generales

Atributo

Un atributo es un parámetro de un elemento declarado en la DTD. El tipo de un atributo y su rango de valores, incluyendo la posibilidad de un valor por defecto, se definen en la DTD.

DTD

Una DTD, o definición del tipo de documento, es una colección de declaraciones XML que, como colección, define la estructura reglamentaria, los elementos y atributos que están disponibles para su uso en documentos que cumplan con la DTD.

Documento

Un documento es una cadena de datos que, tras ser combinado con cualquier otra cadena a la que referencie, queda estructurado de tal manera que porta información contenida en elementos que se organizan tal y como está especificado en la correspondiente DTD. Para más información, ver los [Requisitos de Conformidad para Documentos](#).

Elemento

Un elemento es una unidad estructural de un documento que ha sido declarada en la DTD. El modelo de contenidos del elemento está definido en la DTD, y la explicación adicional puede especificarse en la descripción comentada del elemento.

Recursos

Los componentes disponibles incluyen elementos, atributos y las explicaciones asociadas a dichos elementos y atributos. Una aplicación que admita dichos componentes debe facilitar los recursos necesarios para procesarlos.

Aplicación

Una aplicación es un sistema que posee una colección de recursos y servicios que admite esta especificación. Ver el apartado de [Conformidad de Aplicaciones de Usuario](#) para más información.

Análisis

El análisis es el proceso por el cual un documento es leído y la información en él contenida se traduce en el contexto de elementos en que esta información está estructurada.

Presentación

La presentación es el proceso por el cual la información contenida en un documento se muestra al usuario. Esto se lleva a cabo de la forma más apropiada al entorno que utilice el usuario (e.j. de forma auditiva, visual, impresa).

Aplicación de Usuario

Una aplicación de usuario es una aplicación que lee y procesa documentos XHTML. Ver [Conformidad de Aplicaciones de Usuario](#) para más información.

Convalidación

La convalidación es un proceso por el cual los documentos son contrastados con la DTD asociada, asegurándose de que la estructura, el uso de elementos y el uso de atributos son consistentes con las definiciones de la DTD.

Gramaticalidad (Documento "bien formado")

Un documento se dice "bien formado" o "gramaticalmente correcto" cuando está estructurado de acuerdo a las reglas definidas en la [Sección 2.1](#) de la Recomendación de XML 1.0 [\[XML\]](#). Básicamente, en dicha definición implica que los elementos, delimitados por sus etiquetas de inicio y fin, estén convenientemente anidados.



3. Definición Normativa de XHTML 1.0

3.1 Requisitos de Conformidad para Documentos

Esta versión de XHTML suministra una definición de documentos XHTML estrictamente conformes que se restringe a las etiquetas y atributos del espacio nominal de XHTML. Ver la [Sección 3.1.2](#) para información concerniente al uso de XHTML con otros espacios nominales, por ejemplo la inclusión de metadatos expresados en RDF dentro de documentos XHTML.

3.1.1 Documentos Estrictamente Conformes

Un documento XHTML estrictamente conforme es un documento que para ser procesado requiere tan sólo los recursos descritos como obligatorios en esta especificación. Tales documentos deben ajustarse a los siguientes puntos:

1. Deben poder validarse con alguna de las tres DTD que se encuentran en el [Apéndice A](#).
2. El elemento raíz del documento debe ser <html>.
3. El elemento raíz del documento debe indicar el espacio nominal XHTML usando el atributo xmlns [\[XMLNAMES\]](#). El espacio nominal para XHTML es <http://www.w3.org/1999/xhtml>
4. Debe haber una declaración DOCTYPE en el documento antes del elemento raíz. El identificador público incluido en la declaración DOCTYPE debe hacer referencia a alguna de las tres DTD que se hallan en el [Apéndice A](#) usando el Identificador Formal Público correspondiente. El identificador del sistema puede ser modificado apropiadamente para reflejar convenciones de rango local.
 - 5.
 6. <!DOCTYPE html
 7. PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 8. "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
 - 9.
 - 10.
 - 11.
 12. <!DOCTYPE html
 13. PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 14. "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
 - 15.
 - 16.
 17. <!DOCTYPE html
 18. PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
 19. "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
 - 20.

He aquí un ejemplo de un pequeño documento XHTML.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html
  PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="sp" lang="sp">
<head>
<title>Biblioteca Virtual</title>
</head>
<body>
<p>Trasladada a <a href="http://vlib.org">vlib.org</a>.</p>
</body>
</html>
```



Nótese que en este ejemplo, se incluye la declaración XML. Una declaración XML como la que se hace arriba no es necesaria en todos los documentos XML. Aún así, se recomienda encarecidamente a los autores de documentos XHTML que incluyan declaraciones XML en todos sus documentos. Tal declaración es necesaria cuando la codificación de los caracteres que se usa en el documento no es UTF-8 o UTF-16, los tipos usados por defecto en este tipo de documentos.

3.1.2 Usando XHTML con otros espacios nominales

El espacio nominal XHTML 1.0 puede usarse conjuntamente a otros espacios nominales XML como se indica en [XMLNAMES](#), aunque los documentos así producidos no serán documentos XHTML 1.0 estrictamente conformes. Futuros trabajos del W3C darán directrices que especifiquen la conformidad de documentos que usen varios espacios nominales.

El siguiente ejemplo muestra cómo XHTML 1.0 podría usarse en conjunción con la Recomendación MathML:

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="sp" lang="sp">
<head>
  <title>Un ejemplo matemático</title>
</head>
<body>
  <p>Lo que viene a continuación es etiquetado MathML:</p>
  <math xmlns="http://www.w3.org/1998/Math/MathML">
    <apply> <log/>
      <logbase>
        <cn> 3 </cn>
      </logbase>
      <ci> x </ci>
    </apply>
  </math>
</body>
</html>
```

El siguiente ejemplo muestra cómo el etiquetado XHTML 1.0 podría usarse en otro espacio nominal XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- inicialmente, el espacio nominal por defecto es "books" -->
<book xmlns='urn:loc.gov:books'
  xmlns:isbn='urn:ISBN:0-395-36341-6' xml:lang="en" lang="en">
  <title>Cheaper by the Dozen</title>
  <isbn:number>1568491379</isbn:number>
  <notes>
    <!-- hacemos HTML el espacio nominal por efecto para un comentario
      hipertextual -->
    <p xmlns='http://www.w3.org/1999/xhtml' xml:lang="sp" lang="sp">
      También está disponible <a href="http://www.w3.org/">en la red</a>.
    </p>
  </notes>
</book>
```



3.2 Requisitos de Conformidad para Aplicaciones de Usuario

Una aplicación de usuario debe cumplir todos y cada uno de los siguientes criterios de conformidad:

1. Para ser consistente con la Recomendación XML 1.0 [\[XML\]](#), la aplicación de usuario debe analizar y evaluar si un documento XHTML es "gramaticalmente correcto". Si la aplicación de usuario se dice convalidante debe contrastar los documentos con las DTD de acuerdo con [\[XML\]](#).
2. Cuando una aplicación de usuario dice soportar [recursos](#) definidos en esta especificación o requeridos por esta especificación a través de referencia normativa debe hacerlo de manera consistente con la definición del recurso.
3. Cuando una aplicación de usuario procesa un documento XHTML como XML genérico, tan sólo debería reconocer atributos del tipo ID (e.j. el atributo id de la mayoría de los elementos XHTML) como identificadores de fragmentos.
4. Si una aplicación de usuario encuentra un elemento que no reconoce, debe presentar el contenido de dicho elemento.
5. Si una aplicación de usuario encuentra un atributo que no reconoce, debe ignorar completamente la directriz que marque el atributo (i.e., el atributo y su valor).
6. Si una aplicación de usuario encuentra un valor de un atributo que no reconoce, debe usar en su lugar el valor por defecto del atributo.
7. Si encuentra una referencia a una entidad (distinta a las entidades predefinidas) para la que la aplicación de usuario no ha procesado ninguna declaración (lo que podría suceder si la declaración se encuentra en un subconjunto externo al que la aplicación de usuario no ha accedido), la entidad debe presentarse como los caracteres (comenzando con & y terminando con punto y coma) que componen la referencia a la entidad.
8. Cuando se presente el contenido, las aplicaciones de usuario que encuentren caracteres o referencias a entidades de tipo carácter que reconozcan pero que no sean capaces de mostrar, deberían mostrar el documento de tal manera que el usuario aprecie claramente que no ha sido posible una presentación correcta.
9. Los siguientes caracteres se definen en [\[XML\]](#) como caracteres de espacios en blanco:
 - o Espacio ()
 - o Tabulación ()
 - o Retorno de carro ()
 - o Avance de línea (
)

El procesador XML normaliza varios sistemas de códigos de fin de línea en un único carácter de avance de línea que se pasa a la aplicación. La aplicación de usuario XHTML debe, además, tratar los siguientes caracteres como espacios en blanco:

- o Salto de página ()
- o Espacio de anchura nula (​)

En elementos donde el atributo 'xml:space' tenga el valor 'preserve', la aplicación de usuario debe conservar intactos todos los espacios en blanco (con excepción de los caracteres de espacio en blanco de encabezamiento y terminación, que deberían ser suprimidos). En otros casos un espacio en blanco se manipula de acuerdo con las siguientes reglas:

- o Todos los espacios en blanco que rodean a un elemento en bloque deberían ser suprimidos.
- o Los comentarios se suprimen por completo y no afectan a la manipulación de espacios en blanco. Un carácter de espacio en blanco a ambos lados de un comentario se trata como dos espacios en blanco.
- o Los espacios en blanco de encabezamiento y terminación dentro de un elemento en bloque deben suprimirse.



- Los caracteres de avance de línea dentro de un elemento en bloque deben convertirse en un espacio (excepto cuando el atributo 'xml:space' tenga el valor 'preserve').
- Una secuencia de espacios en blanco debe reducirse a un solo carácter de espacio en blanco (excepto cuando el atributo 'xml:space' tenga el valor 'preserve').
- Con respecto a la presentación, la aplicación de usuario debería presentar el contenido de manera adecuada al idioma en el que el contenido está escrito. En idiomas cuya base de escritura sea de caracteres latinos, el carácter ASCII de espacio se usa normalmente bien como límite gramatical entre palabras bien como espacio tipográfico en blanco; en idiomas cuya base esté relacionada con Nagari (e.j., Sanskrit, Thai, etc.), los límites gramaticales pueden codificarse con el carácter de 'espacio' ZW, pero no serán normalmente representados por caracteres tipográficos de espacio en blanco al ser mostrados; los idiomas que usen una base de escritura árabe pueden codificar el carácter tipográfico espacio en blanco usando un carácter de espacio en blanco, pero también pueden usar el carácter de 'espacio' ZW para delimitar límites gramaticales 'internos' (lo que parecen palabras en árabe para un lector inglés frecuentemente engloba varias palabras, e.g. 'kitAbuhum' = 'kitAbu-hum' = 'libro de ellos' == su libro); e idiomas provenientes del chino normalmente no codifican tales delimitadores ni usan el carácter tipográfico de espacio en blanco de este modo.

El espacio en blanco en los valores de atributos se procesa de acuerdo con [\[XML\]](#).

4. Diferencias con HTML 4.0

Debido al hecho de que XHTML es una aplicación XML, ciertas prácticas que eran perfectamente válidas en HTML 4.0 [\[HTML\]](#), basado en SGML, deben cambiar.

4.1 Los documentos deben ser "gramaticalmente correctos"

La [gramaticalidad](#) de los documentos es un nuevo concepto introducido por [\[XML\]](#). Esencialmente significa que todos los elementos bien deben tener etiquetas de cierre bien deben ser escritos de manera especial (tal y como se describe abajo), y que todos los elementos deben estar anidados.

Aunque el solapamiento de elementos no está permitido en SGML, era tolerado en los navegadores existentes.

CORRECTO: elementos anidados

```
<p>he aquí un <em>párrafo</em> enfatizado.</p>
```

INCORRECTO: elementos solapados

```
<p>he aquí un <em>párrafo</p> enfatizado</em>
```



4.2 Los nombres de elementos y atributos deben escribirse en minúscula

Los documentos XHTML deben usar minúsculas para los nombres de todos los elementos y atributos HTML. Esta diferencia es necesaria porque XML es sensible a minúsculas y mayúsculas e.j. and son etiquetas diferentes.

4.3 Los elementos no vacíos requieren etiquetas de cierre

Con HTML 4.0, basado en SGML, en algunos elementos podía omitirse la etiqueta de cierre, de tal manera que la apertura de los elementos que les sucedían implicaba dicho cierre. Esta omisión no está permitida en XHTML, basado en XML. Todos los elementos que no estén declarados en la DTD como EMPTY deben tener una etiqueta de cierre.

CORRECTO: elementos cerrados

```
<p>he aquí un párrafo.</p><p>aquí hay otro párrafo.</p>
```

INCORRECTO: elementos no cerrados

```
<p>he aquí un párrafo.<p>aquí hay un párrafo.
```

4.4 Los valores de los atributos deben ir entre comillas

Todos los valores de atributos deben ir entrecomillados, incluso aquellos que son numéricos.

CORRECTO: valores de atributo entrecomillados

```
<table rows="3">
```

INCORRECTO: valores de atributo no entrecomillados

```
<table rows=3>
```

4.5 Minimización de atributos

XML no soporta la minimización de atributos. Los pares atributo-valor deben escribirse en toda su extensión. Los nombres de atributos como compact y checked no pueden aparecer en elementos sin que sea especificado su valor.

CORRECTO: atributos no minimizados

```
<dl compact="compact">
```

INCORRECTO: atributos minimizados

```
<dl compact>
```



4.6 Elementos vacíos

Los elementos vacíos deben bien tener una etiqueta de cierre bien terminar su etiqueta de apertura con />. Por ejemplo,
 o <hr></hr>. Ver las [directrices de compatibilidad con HTML](#) para recabar información sobre cómo asegurar la compatibilidad retroactiva con aplicaciones de usuario HTML 4.0.

CORRECTO: etiquetas vacías cerradas

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

INCORRECTO: etiquetas vacías no cerradas

```
<br><hr>
```

4.7 Manipulación de espacios en blanco dentro de los valores de atributos

En los valores de atributos, las aplicaciones de usuario eliminarán los espacios en blanco de encabezamiento y terminación y sustituirán las secuencias de uno o más espacios en blanco (incluyendo los saltos de línea) por un único espacio en blanco entre palabras (un carácter ASCII de espacio en blanco para escrituras occidentales). Ver la [sección 3.3.3](#) de [\[XML\]](#).

4.8 Elementos `script` y `style`

En XHTML, los elementos `script` y `style` se declaran como elementos con contenido `#PCDATA`. Como resultado, `<` y `&` serán tratados como comienzos de etiquetado, y entidades como `<` y `&` serán reconocidas como referencias a las entidades `<` y `&` respectivamente por el procesador XML. Englobar el contenido del elemento `script` o `style` dentro de una sección marcada como `CDATA` evita el procesamiento de estas entidades.

```
<script>
<![CDATA[
... contenido no procesado del script ...
]]>
</script>
```

Las secciones `CDATA` son reconocidas por el procesador XML y aparecen como nodos en el Modelo del Objeto Documento (DOM), ver la [sección 1.3](#) de la Recomendación DOM Level 1 [\[DOM\]](#).

Una alternativa es usar documentos externos de estilo y escritura de código.

4.9 Las exclusiones de SGML

SGML da al escritor de una DTD la posibilidad de impedir que elementos específicos estén anidados en otros elementos. Tales prohibiciones (denominadas "exclusiones") no son posibles de realizar en XML.

Por ejemplo, la DTD de HTML 4.0 Strict prohíbe el anidamiento de un elemento 'a' dentro de otro elemento 'a' en cualquier profundidad de anidamiento. No es posible dictar tal prohibición en XML. Aunque tales prohibiciones no puedan definirse en la DTD, algunos elementos no deberían anidarse. Un resumen de dichos elementos y los elementos que no deberían anidarse en ellos se encuentra en la normativa [Apéndice B](#).



4.10 Elementos con atributos id y name

HTML 4.0 definía el atributo name para los elementos a, applet, frame, iframe, img, y map. HTML 4.0 también introducía el atributo id. Ambos atributos están diseñados para ser usados como identificadores de fragmentos de información.

En XML, los identificadores de fragmentos son del tipo ID, y tan sólo puede haber un único atributo de tipo ID por elemento. Por tanto, en XHTML 1.0 el atributo id se define con tipo ID. Con objeto de asegurar que los documentos XHTML 1.0 sean documentos XML bien estructurados, los documentos XHTML 1.0 DEBEN usar el atributo id para definir un identificador de fragmento, incluso en elementos que históricamente también hayan usado el atributo name. Ver las [directrices de compatibilidad con HTML](#) para recabar información sobre cómo asegurar la compatibilidad retroactiva de los anclajes cuando se sirvan documentos XHTML cuyo tipo de soporte de internet (MIME) sea text/html.

Notar que en XHTML 1.0, el atributo name de dichos elementos está formalmente prohibido y desaparecerá en la siguiente versión de XHTML.

5. Compatibilidad

Aunque no hay ninguna obligación de que los documentos XHTML 1.0 sean compatibles con las aplicaciones de usuario existentes, en la práctica es algo fácil de conseguir. Las directrices para crear documentos compatibles pueden encontrarse en el [Apéndice C](#).

5.1 Especificación del Tipo de Soporte de Internet

En el momento de la publicación de esta recomendación, el etiquetado MIME general recomendado para aplicaciones basadas en XML aún no ha sido decidido.

Sin embargo, los documentos XHTML que sigan las directrices indicadas en el [Apéndice C](#), "Directrices de compatibilidad con HTML " pueden ser etiquetados con el tipo de soporte de internet "text/html", dado que son compatibles con la mayoría de los navegadores HTML. Este documento no hace ninguna recomendación sobre el etiquetado MIME de otros documentos XHTML.

6. Futuras Líneas de Actuación

XHTML 1.0 sienta la base para una familia de tipos de documentos que extenderán y acotarán XHTML con objeto de soportar un amplio rango de nuevos dispositivos y aplicaciones, definiendo módulos que especifiquen un mecanismo para combinar dichos módulos. Dicho mecanismo permitirá la extensión y el acotamiento de XHTML 1.0 de una manera uniforme a través de la definición de nuevos módulos.

6.1 Modularizar HTML

A la vez que el uso de XHTML vaya pasando de las aplicaciones de usuario del ordenador de sobremesa tradicional a otras plataformas, está claro que no todos los elementos de XHTML serán necesarios en todas las plataformas. Por ejemplo un dispositivo de mano o un teléfono móvil pueden soportar sólo un subconjunto de elementos de XHTML.

El proceso de modularización rompe XHTML en una serie de pequeños conjuntos de elementos. Dichos elementos pueden ser recombinados para cumplir las necesidades de diferentes comunidades.

Estos módulos se definirán en un documento posterior de la W3C.



6.2 Subconjuntos y Extensibilidad

La modularización conlleva diversas ventajas:

- Provee un mecanismo formal para acotar XHTML.
- Provee un mecanismo formal para extender XHTML.
- Simplifica la transformación entre tipos de documento.
- Promueve la reutilización de módulos en nuevos tipos de documento.

6.3 Perfiles de Documento

Un perfil de documento especifica la sintaxis y la semántica de un conjunto de documentos. La conformidad con un perfil de documento provee una base para la garantía de interoperabilidad. El perfil de documento especifica los recursos necesarios para procesar los documentos de dicho tipo, e.g. qué formatos de imagen pueden usarse, niveles de escritura de código, soporte de hojas de estilo, etc.

Para diseñadores de productos, esto permite a distintos grupos la definición de su propio perfil estándar.

Para autores, esto permitirá obviar la necesidad de escribir diferentes versiones de documentos para diferentes clientes.

Para grupos especiales tales como químicos, médicos o matemáticos esto permitirá la construcción de un perfil especial usando elementos HTML estándar más un grupo de elementos específicamente diseñados para cubrir las necesidades de los especialistas.

Apéndice A. DTD

Este apéndice es normativo.

Estas DTD y conjuntos de entidades forman una parte normativa de esta especificación. El conjunto completo de archivos DTD conjuntamente con una declaración XML y el Catálogo Abierto SGML se incluye en el [archivo zip](#) disponible para esta especificación.

A.1 Definiciones del Tipo de Documento

Estas DTD se aproximan a las DTD de HTML 4.0. Se trata de que cuando, en un futuro, estas DTD se modularicen, se emplee un método de construcción de DTD que se corresponda más claramente con HTML 4.

- [XHTML-1.0-Strict](#)
- [XHTML-1.0-Transitional](#)
- [XHTML-1.0-Frameset](#)

A.2 Conjunto de Entidades

Los conjuntos de entidades XHTML predefinidas son los mismos que en HTML 4.0, pero han sido modificados para ser declaraciones de entidades válidas en XML 1.0. Fijémonos en que la entidad para el signo del Euro (€ o € o €) se define como una parte de los caracteres especiales.

- [Caracteres Latin-1](#)
- [Caracteres Especiales](#)
- [Símbolos](#)



Apéndice B. Incompatibilidades entre elementos

Este apéndice es normativo.

A continuación se detallan las incompatibilidades en el anidamiento de elementos (ver la [sección 4.9](#)). Esta prohibición se aplica a todas las profundidades de anidamiento, i.e. afecta a todos los elementos descendientes de aquel para el que se especifica la restricción.

a no puede contener otros elementos a.

pre no puede contener los elementos `img`, `object`, `big`, `small`, `sub` o `sup`.

button no puede contener los elementos `input`, `select`, `textarea`, `label`, `button`, `form`, `fieldset`, `iframe` o `isindex`.

label no puede contener otros elementos `label`.

form no puede contener otros elementos `form`.

Apéndice C. Directrices de Compatibilidad con HTML

Este apéndice es informativo.

Este apéndice resume las directrices de diseño para autores que quieren que sus documentos XHTML puedan ser presentados en aplicaciones de usuario HTML ya existentes.

C.1 Instrucciones de Proceso

Hay que ser consciente de que las instrucciones de proceso se ejecutan en algunas aplicaciones de usuario. Sin embargo, hay que notar que cuando la declaración XML no se incluye en un documento, éste sólo puede usar las codificaciones de caracteres por defecto UTF-8 o UTF-16.

C.2 Elementos Vacíos

Incluir un espacio en blanco antes de la barra y ángulo de cierre / y > de los elementos vacíos, e.g. `
`, `<hr />` y ``. También, usar la sintaxis minimizada de etiquetas para los elementos vacíos, e.g. `
`, dado que la sintaxis alternativa a `
</br>` permitida por XML da resultados no previsibles en muchos de las aplicaciones de usuario ya existentes.

C.3 Minimización de Elementos y Contenido de Elementos Vacíos

Dada una instancia vacía de un elemento cuyo modelo de contenido no es EMPTY (por ejemplo, un título o párrafo vacíos) no usar la forma minimizada (e.g. usar `<p> </p>` y no `<p />`).

C.4 Hojas de Estilo y Archivos de Código Incrustados

Usar hojas de estilo externas si la hoja en cuestión utiliza los caracteres `< o & o]]>` o `--`. Usar archivos externos de código si el código utiliza los caracteres `< o & o]]>` o `--`. Notar que los analizadores XML tienen permitido suprimir el contenido de los comentarios. De esta manera, la práctica común hasta ahora de "esconder" los fragmentos de código (`script`) y hojas de estilo (`style`) entre comentarios, para hacerlos invisibles a antiguos navegadores, normalmente no funcionará en aplicaciones basadas en XML.



C.5 Saltos de Línea dentro de Valores de Atributos

Evitar saltos de línea y múltiples espacios en blanco dentro de los valores de los atributos. Estos son manipulados de manera inconsistente por las aplicaciones de usuario.

C.6 `isindex`

No incluir más de un elemento `isindex` en el head del documento. El elemento `isindex` se tiende a descartar en favor del elemento `input`.

C.7 Los atributos `lang` y `xml:lang`

Úsense ambos atributos, `lang` y `xml:lang`, cuando se quiera especificar el idioma de un elemento. El valor del atributo `xml:lang` tiene preferencia.

C.8 Identificadores de Fragmentos

En XML, los URI [\[RFC2396\]](#) que terminan con identificadores de fragmentos de la forma "#foo" no se refieren a elementos con un atributo `name="foo"`; por el contrario se refieren a elementos con un atributo del tipo ID, e.g., el atributo `id` de HTML 4.0. Muchos clientes de HTML actuales no soportan este uso de atributos de tipo ID, de tal manera que se puede dar valores idénticos a ambos atributos para asegurar la máxima compatibilidad futura y retroactiva (e.g., `...`).

Más aún, dado que el conjunto de valores permitidos para atributos del tipo ID es mucho menor que los permitidos para atributos del tipo CDATA, el tipo del atributo `name` ha sido cambiado a NMTOKEN. Este atributo está limitado de tal manera que sólo puede tomar los mismos valores que los de tipo ID o los de la producción `Name` de XML 1.0, sección 2.5, producción 5. Desafortunadamente esta limitación no puede expresarse en las DTD de XHTML 1.0. Debido a este cambio, debe tenerse cuidado cuando se conviertan documentos HTML ya existentes a XHTML 1.0. Los valores de estos atributos deben ser únicos en todo el documento, válidos, y tales que cualquier referencia a estos identificadores de fragmentos (tanto interna como externa) deben actualizarse durante la conversión.

Finalmente, notar que XHTML 1.0 tiende a desechar el atributo `name` de los elementos `a`, `applet`, `frame`, `iframe`, `img`, y `map`, y será eliminado en versiones posteriores de XHTML.

C.9 Codificación de caracteres

Para especificar una codificación de caracteres en el documento, usar tanto la especificación del atributo de codificación en la declaración `xml` (e.g. `<?xml version="1.0" encoding="EUC-JP"?>`) como una sentencia meta `http-equiv` (e.g. `<meta http-equiv="Content-type" content="text/html; charset="EUC-JP" />`). El valor del atributo de codificación de la instrucción de proceso `xml` tiene preferencia.

C.10 Atributos booleanos

Algunas aplicaciones de usuario no son capaces de interpretar atributos booleanos cuando estos aparecen en su forma extendida (no minimizada), tal y como requiere XML 1.0. Notar que este problema no afecta a aplicaciones de usuario conformes a la especificación HTML 4.0. Los siguientes atributos se encuentran afectados: `compact`, `nowrap`, `ismap`, `declare`, `noshade`, `checked`, `disabled`, `readonly`, `multiple`, `selected`, `noresize`, `defer`.



C.11 EL Modelo del Objeto Documento y XHTML

La Recomendación de nivel 1 del Modelo del Objeto Documento [DOM] define interfaces del modelo del objeto documento para XML y HTML 4.0. El modelo del objeto documento de HTML 4.0 especifica que los nombres de los elementos y atributos HTML se devuelven en mayúsculas. El modelo del objeto documento de XML especifica que los nombres de los elementos y atributos se devuelven con el tipo en que se hayan escrito en el propio documento. En XHTML 1.0, los elementos y atributos se escriben en minúsculas. Esta diferencia aparente puede ser resuelta de dos modos:

1. Las aplicaciones que accedan a documentos XHTML servidos como tipo de soporte Internet text/html vía el DOM pueden usar el DOM HTML, y asegurarse así de que los nombres de los elementos y atributos serán devueltos en mayúsculas por dichas interfaces.
2. Las aplicaciones que accedan a documentos XHTML servidos como tipo de soporte Internet text/xml o application/xml pueden usar también el DOM XML. Los elementos y atributos serán devueltos en minúsculas. Además, algunos elementos XML pueden o no aparecer en el árbol de objetos porque son opcionales en el modelo de contenidos (e.g. el elemento tbody dentro de table). Esto ocurre porque en HTML 4.0 a algunos elementos se les permitía ser minimizados de tal manera que tanto la etiqueta de apertura como la de cierre se omitían (una característica de SGML). Esto no es posible en XML. En vez de hacer obligatorios estos elementos que no solían usarse, XHTML ha optado por hacerlos opcionales. Las aplicaciones necesitan adaptarse a ello.

C.12 Uso del carácter & en Valores de Atributos

Cuando el valor de un atributo contenga un carácter &, debe expresarse como una referencia a la entidad de tipo carácter (e.j. "&"). Por ejemplo, cuando el atributo href de un elemento apunta a un código CGI que tome parámetros, debe expresarse como `http://my.site.dom/cgi-bin/myscript.pl?class=guest&name=user` en vez de `http://my.site.dom/cgi-bin/myscript.pl?class=guest&name=user`.

C.13 Hojas de Estilo en Cascada (CSS) y XHTML

La Recomendación de nivel 2 de las Hojas de Estilo en Cascada [CSS2] define propiedades de estilo que se aplican al árbol de análisis del documento HTML o XML. Las diferencias en el análisis producirán diversos resultados visibles o auditivos, dependiendo de los selectores usados. Las siguientes pistas reducirán este efecto en los documentos que se sirvan sin modificación como cualquiera de estos tipos de soporte:

1. Las hojas de estilo CSS para XHTML deberían usar nombres de elementos y atributos en minúsculas.
2. En las tablas, el elemento tbody será inferido por el analizador de una aplicación de usuario HTML, pero no por el analizador de una aplicación de usuario XML. Por tanto se debería añadir siempre explícitamente un elemento tbody si se hace referencia a él en un selector CSS.
3. Dentro del espacio nominal XHTML, se espera que las aplicaciones de usuario reconozcan el atributo "id" como un atributo de tipo ID. Por tanto, las hojas de estilo deberían ser capaces de continuar usando la sintaxis taquigráfica de selectores "#" incluso si la aplicación de usuario no es capaz de leer la DTD.
4. Dentro del espacio nominal XHTML, se espera que las aplicaciones de usuario reconozcan el atributo "class". Por tanto, las hojas de estilo deberían ser capaces de continuar usando la sintaxis taquigráfica de selectores ".".
5. CSS define diferentes reglas de conformidad para los documentos HTML y XML; téngase en cuenta que las reglas HTML se aplican a los documentos XHTML suministrados como HTML y las reglas XML se aplican a los documentos XHTML suministrados como XML.



Apéndice D. Agradecimientos

Este apéndice es informativo.

Esta especificación ha sido escrita con la participación de los miembros del grupo de trabajo de HTML del W3C:

Steven Pemberton, CWI (HTML Working Group Chair)
Murray Altheim, Sun Microsystems
Daniel Austin, AskJeeves (CNET: The Computer Network through July 1999)
Frank Boumphrey, HTML Writers Guild
John Burger, Mitre
Andrew W. Donoho, IBM
Sam Dooley, IBM
Klaus Hofrichter, GMD
Philipp Hoschka, W3C
Masayasu Ishikawa, W3C
Warner ten Kate, Philips Electronics
Peter King, Phone.com
Paula Klante, JetForm
Shin'ichi Matsui, Panasonic (W3C visiting engineer through September 1999)
Shane McCarron, Applied Testing and Technology (The Open Group through August 1999)
Ann Navarro, HTML Writers Guild
Zach Nies, Quark
Dave Raggett, W3C/HP (W3C lead for HTML)
Patrick Schmitz, Microsoft
Sebastian Schnitzenbaumer, Stack Overflow
Peter Stark, Phone.com
Chris Wilson, Microsoft
Ted Wugofski, Gateway 2000
Dan Zigmond, WebTV Networks

Apéndice E. Referencias

Este apéndice es informativo.

[CSS2]

["Cascading Style Sheets, level 2 \(CSS2\) Specification"](#), B. Bos, H. W. Lie, C. Lilley, I. Jacobs, 12 May 1998.

Disponible en: <http://www.w3.org/TR/REC-CSS2>

[DOM]

["Document Object Model \(DOM\) Level 1 Specification"](#), Lauren Wood *et al.*, 1 October 1998.

Disponible en: <http://www.w3.org/TR/REC-DOM-Level-1>

[HTML]

["HTML 4.01 Specification"](#), D. Raggett, A. Le Hors, I. Jacobs, 24 August 1999.

Disponible en: <http://www.w3.org/TR/1999/PR-html40-19990824>

[POSIX.1]

"ISO/IEC 9945-1:1990 Information Technology - Portable Operating System Interface (POSIX) - Part 1: System Application Program Interface (API) [C Language]", Institute of Electrical and Electronics Engineers, Inc, 1990.

[RFC2046]

["RFC2046: Multipurpose Internet Mail Extensions \(MIME\) Part Two: Media Types"](#), N. Freed and N. Borenstein, November 1996.

Disponible en <http://www.ietf.org/rfc/rfc2046.txt>. Notar que este RFC deja obsoletos los RFC1521, RFC1522, y RFC1590.



[RFC2119]

"[RFC2119: Key words for use in RFCs to Indicate Requirement Levels](#)", S. Bradner, March 1997.
Disponible en: <http://www.ietf.org/rfc/rfc2119.txt>

[RFC2376]

"[RFC2376: XML Media Types](#)", E. Whitehead, M. Murata, July 1998.
Disponible en: <http://www.ietf.org/rfc/rfc2376.txt>

[RFC2396]

"[RFC2396: Uniform Resource Identifiers \(URI\): Generic Syntax](#)", T. Berners-Lee, R. Fielding, L. Masinter, August 1998.
Este documento actualiza los RFC1738 y RFC1808.
Disponible en: <http://www.ietf.org/rfc/rfc2396.txt>

[XML]

"[Extensible Markup Language \(XML\) 1.0 Specification](#)", T. Bray, J. Paoli, C. M. Sperberg-McQueen, 10 February 1998.
Disponible en: <http://www.w3.org/TR/REC-xml>

[XMLNAMES]

"[Namespaces in XML](#)", T. Bray, D. Hollander, A. Layman, 14 January 1999.
Los espacios nominales XML proveen un método simple para calificar los nombres usados en documentos XML asociándolos con espacios nominales identificados con una URI. by URI.
Disponible en: <http://www.w3.org/TR/REC-xml-names>