indice

I possibili errori presenti in questo documento, dovuti alla traduzione, sono di responsabilità del traduttore e non sono in alcun modo imputabili al W3C. Per qualsiasi commento riguardo la traduzione rivolgersi a Patrizia Andronico.
L'unica versione ufficiale di questo documento è la versione originale in inglese: http://www.w3.org/TR/2000/REC-xhtml1-20000126)

XHTML™ 1.0: Extensible HyperText Markup Language

Una riformulazione di HTML 4 in XML 1.0

W3C Recommendation 26 January 2000

Questa versione (in inglese):
http://www.w3.org/TR/2000/REC-xhtml1-20000126
(Postscript version, PDF version, ZIP archive, or Gzip'd TAR archive)
Ultima versione (in inglese):
http://www.w3.org/TR/xhtml1
Precedente versione (in inglese):
http://www.w3.org/TR/1999/PR-xhtml1-19991210
Autori:
Vedi acknowledgements.

Riassunto

Questa specifica definisce XHTML 1.0, una riformulazione di HTML 4 come applicazione XML 1.0, e le tre DTD corrispondenti a quelle definite in HTML 4. La semantica degli elementi e dei loro attributi è definita nella Raccomandazione del W3C per HTML 4. Tale semantica fornisce la base per future estensioni di XHTML. La compatibilità con gli user agent esistenti è possibile seguendo un piccolo insieme di regole.

Stato di questo documento

Questa sezione descrive lo stato di questo documento al momento della sua

pubblicazione. Altri documenti possono sostituire questo documento. L'ultima versione di questa serie di documenti si trova al W3C.

Questo documento è stato rivisto dai Membri del W3C e da altre parti interessate ed è stato approvato dal Direttore come una Raccomandazione del W3C. E' un documento stabile e può essere usato come materiale di riferimento o citato da un altro documento come una normativa di riferimento. L'obiettivo del W3C nel fare le Raccomandazioni è quello di richiamare l'attenzione alle specifiche e di promuovere la loro più ampia diffusione. Questo aumenta la funzionalità e l'interoperabilità del Web.

Questo documento è stato prodotto come parte del W3C HTML Activity. Gli scopi del HTML Working Group *(members only)* sono discussi nel HTML Working Group charter *(members only)*.

Un elenco delle attuali Raccomandazioni del W3C e di altri documenti tecnici è reperibile all'indirizzo http://www.w3.org/TR.

La discussione pubblica sulle caratteristiche dell'HTML si ha attraverso la mailing list www-html@w3.org (archivio).

Per favore, riportate gli errori presenti in questo documento a www-html-editor@w3.org.

Una lista degli errori noti di questa specifica si trovano a http://www.w3.org/2000/01/REC-xhtml1-20000126-errata (in inglese).

Contenuto

1. Cosa è XHTML?

XHTML è una famiglia di attuali e futuri tipi di documenti che riproduce, ingloba ed estende l'HTML 4 [HTML]. I tipi di documenti della famiglia XHTML si basano su XML, e sono disegnati fondamentalmente per poter lavorare insieme agli user agent basati su XML. I dettagli di questa famiglia e della sua evoluzione sono discussi nella sezione Direzioni future.

XHTML 1.0 (questa specifica) è il primo tipo di documento della famiglia XHTML. E' una riformulazione dei tre tipi di documento HTML 4 come applicazioni di XML 1.0 [XML]. Il suo scopo è quello di essere usato come linguaggio di contenuto che sia contemporaneamente conforme a XML e che operi con gli user agent in conformità a HTML 4, nel caso in cui vengano seguite alcune semplici linee guida. Gli sviluppatori che migrano le loro applicazioni verso XHTML 1.0 avranno i seguenti vantaggi:

- i documenti XHTML sono conformi a XML. In quanto tali sono facilmente visualizzati, editati e validati con i tool standard di XML.
- i documenti XHTML possono essere scritti per funzionare in modo uguale o migliore di quanto facessero prima con gli user agent conformi a HTML 4, così come nei nuovi user agent conformi a XHTML 1.0.
- i documenti XHTML possono utilizzare applicazioni (per esempio script e applet) che si basano sia sul Document Object Model di HTML sia su quello di XML [DOM]
- poichè si ha una evoluzione della famiglia XHTML, i documenti conformi a XHTML 1.0 saranno sempre più pronti per interagire all'interno e con diversi ambienti XHTML.

La famiglia XHTML è il prossimo passo nell'evoluzione di Internet. Passando oggi a XHTML, gli sviluppatori possono entrare nel mondo XML con tutti i benefici che si aspettano, assicurandosi la compatibilità con gli user agent attuali e futuri.

1.1 Cosa è HTML 4?

HTML 4.0 [HTML] è una applicazione SGML (Standard Generalized Markup Language) conforme allo Standard Internazionale ISO 8879, e viene considerato da tutti il linguaggio standard per le pubblicazioni del World Wide Web.

SGML è un linguaggio per descrivere linguaggi di markup, in particolare per quelli usati nello scambio di documenti elettronici, sviluppo e pubblicazione di documenti. HTML è un esempio di linguaggio definito in SGML.

SGML viene usato dalla metà degli anni '80 ed è rimasto abbastanza stabile. La maggior parte della sua stabilità si deve al fatto che il linguaggio è ricco di caratteristiche e flessibile. La flessibilità, comunque, ha il suo prezzo, e questo prezzo è l'alto livello di complessità che ha inibito l'uso di questo linguaggio in diversi ambienti, incluso il World Wide Web.

L'HTML, così come fu concepito originariamente, era un linguaggio per lo scambio di documenti scientifici e tecnici, adatto per essere usato da persone non specializzate nel trattamento di documenti. L'HTML ha risolto il problema della complessità dell'SGML specificando un piccolo insieme di tag strutturali e semantici adatti alla realizzazione di semplici documenti. Inoltre, per semplificare la struttura del documento, l'HTML ha aggiunto un supporto per l'ipertestualità. In seguito sono state aggiunte le capacità multimediali.

In pochissimo tempo l'HTML è diventato largamente popolare e rapidamente ha superato il suo scopo originale. Fin dagli inizi dell'HTML c'è stata una continua invenzione di nuovi elementi da usare all'interno dell'HTML (come uno standard) e per adattarlo ad un mercato verticale altamente specializzato. Questa pletora di nuovi elementi ha portato a problemi di compatibilità per i

documenti nelle diverse piattaforme.

Dato il rapido proliferarsi dell'eterogenità sia del software che delle piattaforme, è chiaro che l'adeguatezza dell'HTML 'classico' per l'uso su queste piattaforme è in qualche modo limitato.

1.2 Cosa è XML?

XML è l'abbreviazione di Linguaggio Estensibile di Markup ed è l'acronimo dell'espressione inglese Extensible Markup Language [XML].

XML è stato concepito come un mezzo per riguadagnare la potenzialità e la flessibilità di SGML eliminando la maggior parte della sua complessità. Nonostante sia la forma ristretta di SGML, XML conserva gran parte della potenza e della ricchezza di SGML, mantenendo ancora tutte le caratteristiche comunemente usate da SGML.

Conservando tali caratteristiche, XML rimuove le caratteristiche più complesse di SGML che rendevano difficile e costosa la creazione e il disegno di software appropriato.

1.3 Perchè la necessità di XHTML?

I benefici del passaggio a XHTML 1.0 vengono descritti di seguito. In generale alcuni di questi sono:

- Gli sviluppatori di documenti e i disegnatori di user agent sono costantemente alla scoperta di modi nuovi per esprimere le loro idee attraverso nuovi marcatori. In XML è relativamente semplice intodurre nuovi elementi o aggiungere attributi agli elementi. La famiglia XHTML è stata concepita per accogliere queste estensioni attraverso moduli XHTML e tecniche per lo sviluppo di nuovi moduli conformi ad XHTML (descritti nella futura specifica di Modularizzazione di XHTML). Questi moduli permetteranno la combinazione di insiemi di caratteristiche nuove ed esitenti durante la creazione del contenuto così come il disegno di nuovi user agent.
- Costantemente vengono prodotte nuove forme di accesso ad Internet. Alcune stime indicano che nell'anno 2002 il 75% dei documenti visibili su Internet saranno sviluppati da queste piattaforme alternative. La famiglia XHTML è stata concepita tenendo conto della interoperabilità con gli user agent generali. Attraverso un nuovo user agent e un nuovo meccanismo di specifica del documento, i server, i proxies e gli user agent potranno realizzare una migliore trasformazione del contenuto. In ultimo, sarà possibile sviluppare contenuto conforme a XHTML che sia utilizzabili da tutti gli user agent conformi a XHTML.

2. Definizioni

2.1 Terminologia

In questa specifica vengono usati i termini che seguono. Tali termini estendono le definizioni presenti in [RFC2119] basandosi su definizioni simili in ISO/IEC 9945-1:1990 [POSIX.1]:

Definito per l'implementazione (Implementation-defined)
: Un valore o un comportamento si dice definito per l'implementazione quando viene lasciata alla stessa implementazione il compito di definire (e documentare) i requisiti corrispondenti per una costruzione corretta del documento.

Può (May)
: Rispettando l'implementazione, la parola "may" deve essere interpretata come una caratteristica futura che non viene richiesta in queste specifiche ma che può essere fornita.

Rispettando la Conformità del Documento, la parola "may" significa che le caratteristiche opzionali non devono essere usate. Il termine "optional" ha la stessa definizione di "may".

Deve (Must)
: In queste specifiche, la parola "must" deve essere interpretata come un requisito obbligatorio dell'implementazione o della Stretta Conformità dei Documenti XHTML, a seconda del contesto. Il termine "shall" ha la stessa definizione di "must".

Riservato (Reserved)
: Si riferisce ad una valore o ad un comportamento che non è specificato e il cui uso non è neppure ammesso dalla Conformità dei Documenti nè deve essere supportato dagli User Agent Conformi.

Dovrebbe (Should)
: Rispettando l'implementazione, la parola "should" deve essere interpretata come una raccomandazione per l'implementazione, ma non come un requisito. Rispettando i documenti, la parola "should" deve essere interpretata come una raccomandazione pratica per la programmazione dei documenti e un requisito per la Stretta Conformità dei Documenti XHTML.

Ammesso (Supported)
: Alcune strutture in queste specifiche sono opzionali. Se alcune di queste vengono supportate, il loro comportamento sarà quello indicato in queste specifiche.

Non specificato (Unspecified)
: Quando un valore o un comportamento non è determinato, le specifiche non definiscono requisiti di portabilità sull'implementazione neanche quando si deve analizzare un documento che usi tale struttura. Un documento che richiede uno specifico comportamento in questa situazione, invece di tollerare qualsiasi comportamento nell'uso di questa struttura, non sarà un Documento Strettamente Conforme a XHTML.

2.2 Termini generali

Attributo
: Un attributo è un parametro ad un elemento dichiarato nella DTD. Il tipo di un attributo e il suo insieme di valori, compreso un possibile valore di default, vengono definiti nella DTD.

DTD
: Una DTD, o definizione di tipo di documento, è una collezione di dichiarazioni XML che. come collezione, definisce la struttura legale, gli *elementi*, e gli *attributi* che sono disponibili per l'uso in un documento che soddisfa la DTD.

Documento
: Un documento è un flusso di dati che, dopo essere stato combinato con altri flussi di dati a cui fa riferimento, viene strutturato in modo che porta le informazioni contenute all'interno degli *elementi* organizzati come definito nella *DTD* associata. Vedere I Requisiti di Conformità dei Documenti per maggiori informazioni.

Elemento
: Un elemento è una unità strutturale di un documento dichiarato nella *DTD*. Il modello del contenuto dell'elemento viene definito nella *DTD* e la semantica addizionale può essere definita nella descrizione commentata dell'elemento.

Funzionalità
: Le funzionalità comprendono gli *elementi*, gli *attributi*, e la semantica associata con questi *elementi* e *attributi*. Una implementazione che supporti questa funzionalità deve fornire le strutture necessarie.

Implementazione
: Una implementazione è un sistema che fornisce un insieme di *strutture* e servizi che supporti queste specifiche. Vedere User Agent Conformance per maggiori informazioni.

Analisi (Parsing)
: Il parsing è l'azione secondo cui un *documento* viene analizzato, e l'informazione contenuta

all'interno del *documento* viene filtrata nel contesto degli elementi in cui l'informazione è strutturata.

Presentazione (Rendering)
: Rendering è l'azione secondo cui l'informazione viene presentata all'interno di un *documento*. Questa presentazione viene fatta nella forma più appropriata rispetto all'ambiente (esempio: uditiva, visiva, per la stampa).

User Agent
: Uno user agent è una implementazione che ricerca e processa i documenti XHTML. Vedi Conformità degli User Agent (User Agent Conformance) per maggiori informazioni.

Validazione
: La validazione è un processo secondo cui i *documenti* vengono verificati rispetto alla *DTD* associata, assicurando che la loro struttura, l'uso degli *elementi*, e l'uso degli *attributi* sia consistente con le definizioni nella *DTD*.

Ben-formato
: Un *documento* è ben formato quando la sua struttura è in accordo con le regole definite nella Sezione 2.1 della Raccomandazione XML 1.0 [XML]. Principalmente questa definizione indica che gli elementi, delimitati dai loro tag iniziale e finale, sono annidati in maniera corretta uno dentro l'altro.

3. Definizione Normativa di XHTML 1.0

3.1 Conformità del Documento

Questa versione di XHTML fornisce una definizione di conformità stretta dei documenti XHTML, che viene ristretta ai tag e agli attributi dello spazio dei nomi di XHTML. Vedi Sezione 3.1.2 per informazioni sull'uso di XHTML con altri spazi di nomi, per esempio, l'inclusione di metadati espressi in RDF all'interno di documenti XHTML.

3.1.1 Documenti Strettamente Conformi

Un documento XHTML strettamente conforme è un documento che richiede solo le strutture descritte come obbligatorie in queste specifiche. Questi documenti devono rispettare tutti i seguenti punti:

1. Deve essere validato rispetto ad una delle tre DTD presenti in Appendice A.
2. L'elemento radice del documento deve essere **`<html>`**.
3. L'elemento radice del documento deve indicare lo spazio dei nomi di XHTML usando l'attributo **`xmlns`** [XMLNAMES]. Lo spazio dei nomi per XHTML è definito in **`http://www.w3.org/1999/xhtml`**.
4. All'interno del documento ci deve essere una dichiarazione di DOCTYPE prima dell'elemento radice. L'identificatore pubblico incluso nella dichiarazione di DOCTYPE si deve riferire ad una delle tre DTD presenti in Appendice A usando il corrispondente Identificatore Formale Pubblico. L'identificatore di sistema può essere cambiato per riflettere le convenzioni del sistema locale.

```
<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
     "DTD/xhtml1-strict.dtd">

<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
     "DTD/xhtml1-transitional.dtd">

<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
```

```
          "DTD/xhtml1-frameset.dtd">
```

Qui viene riportato un piccolo esempio di un documento XHTML.

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

Nota che in questo esempio viene inclusa la dichiarazione di XML. Una dichiarazione XML come quella sopra non viene richiesta da tutti i documenti XML. Si raccomanda fermamente agli autori di documenti XHTML di usare le dichiarazioni XML in tutti i loro documenti. Questo tipo di dichiarazione viene richiesta quando la codifica dei caratteri del documento è un'altra rispetto a quella usata per default UTF-8 o UTF-16.

3.1.2 Uso di XHTML con altri spazi di nomi

Lo spazio dei nomi XHTML può essere usato con altri spazi dei nomi XML come viene indicato in [XMLNAMES], sebbene questi documenti non siano strettamente conformi ai documenti XHTML 1.0 come definito sopra. Il lavoro futuro del W3C sarà nella direzione di specificare la conformità per quei documenti che comprendono diversi spazi dei nomi.

L'esempio seguente mostra il modo in cui XHTML 1.0 può essere usato insieme alle Raccomandazioni MathML:

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <title>A Math Example</title>
  </head>
  <body>
    <p>The following is MathML markup:</p>
    <math xmlns="http://www.w3.org/1998/Math/MathML">
      <apply> <log/>
        <logbase>
          <cn> 3 </cn>
        </logbase>
        <ci> x </ci>
      </apply>
    </math>
  </body>
</html>
```

L'esempio seguente mostra il modo in cui i marcatori di XHTML possono essere incorporati in un altro spazio dei nomi XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- initially, the default namespace is "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>
    <!-- make HTML the default namespace for a hypertext commentary -->
    <p xmlns='http://www.w3.org/1999/xhtml'>
        This is also available <a href="http://www.w3.org/">online</a>.
    </p>
  </notes>
</book>
```

3.2 Conformità degli User Agent

Uno user agent conforme deve rispettare tutti i seguenti punti:

1. Per essere consistente con le Raccomandazioni di XML 1.0 [XML], lo user agent deve analizzare e valutare un documento XHTML come "ben formato". Se lo user agent si dice validante, allora deve validare anche i documenti rispetto alla loro DTD in accordo con [XML].
2. Quando uno user agent afferma di supportare strutture definite all'interno di queste specifiche o richieste da queste specifiche attraverso le normative di riferimento, lo deve fare in modo consistente rispetto alla definizione di stuttura.
3. Quando uno user agent processa un documento XHTML come un generico XML, dovrà riconoscere solo gli attributi di tipo `ID` come identificatori di frammento (per esempio l'attributo `id` nella maggior parte degli elementi XHTML).
4. Se uno user agent incontra un elemento che non riconosce, deve presentare il contenuto di tale elemento.
5. Se lo user agent incontra un attributo che non riconosce, deve completamente ignorare le specifiche dell'attributo (per esempio, l'attributo e i suoi valori).
6. Se lo user agent incontra un valore di attributo che non riconosce, deve usare il valore di default dell'attributo.
7. Se lo user agent incontra un riferimento ad una entità (diversa da quelle predefinite) per la quale lo user agent non ha processato nessuna dichiarazione (può succedere se la dichiarazione si trova in un sottoinsieme esterno che lo user agent non ha letto), l'entità deve essere presentata come i caratteri (iniziando con un ampersand, &, e finendo con un punto e virgola) che formano il riferimento alla entità.
8. Quando viene presentato il contenuto, lo user agent che incontra caratteri o entità di tipo carattere che riconosce ma che non sa come presentare, dovrebbe visualizza il documento in modo tale che l'utente si accorga chiaramente che non è stata possibile una corretta presentazione.
9. I seguenti caratteri vengono definiti in [XML] come caratteri di spazi bianchi:
    - Spazio ( )
    - Tabulazione (	)
    - Ritorno carrello (
)
    - Line feed (
)

    Il processore XML normalizza diversi sistemi di codice di fine linea in un singolo carattere di line-feed che viene passato all'applicazione. In aggiunta lo user agent XHTML deve trattare come spazi bianchi anche i seguenti caratteri:
    - Form feed ()
    - Zero-width space ()

Negli elementi dove l'attributo 'xml:space' ha il valore 'preserve', lo user agent deve conservare intatti tutti i caratteri di spazi bianchi (fatta eccezione per i caratteri di spazi bianchi iniziali e finali che dovrebbero essere rimossi). In altri casi uno spazio bianco viene considerato secondo le seguenti regole:

- Tutti gli spazi bianchi che circondano elementi in blocco dovrebbero essere rimossi.
- I commenti vengono rimossi interamente e non hanno conseguenza sulla manipolazione di spazi bianchi. Un carattere di spazio bianco su entrambi i lati di un commento viene trattato come due caratteri di spazio bianco.
- Gli spazi bianchi iniziale e finale dentro un elemento in blocco devono essere rimossi.
- I caratteri di line feed all'interno di elementi in blocco devono essere convertiti in uno spazio (tranne quando l'attributo di 'xml:space' viene posto a 'preserve').
- Una sequenza di caratteri di spazi bianchi deve essere ridotta ad un singolo carattere di spazio bianco (tranne quando l'attributo di 'xml:space' viene posto a 'preserve').
- Rispetto alla presentazione, lo user agent dovrebbe visualizzare il contenuto in maniera appropriata al linguaggio secondo cui è scritto il contenuto stesso. Nelle lingue in cui la base di scrittura è il carattere latino, il carattere di spazio ASCII viene usato tipicamente sia come spazio grammaticale tra le parole che come spazio tipografico bianco; nelle lingue la cui scrittura è in relazione al Nagari (esempio Sanscrito, Thai, ecc.) i limiti tra le parole possono essere codificati usando il carattere 'space' ZW, ma non verranno visualizzati come caratteri tipografici di spazi bianchi; lingue con scritture di base araba possono codificare il carattere tipografico di spazio bianco con un carattere di spazio, ma possono anche usare il carattere spazio ZW per delimitare i limiti grammaticali 'interni' (quelle che sembrano parole arabe per un lettore inglese normalmente comprendono più parole, per esempio 'kitAbuhum'= 'kitAbu-hum' = 'libri loro' == i loro libri); e lingue di scrittura cinese normalmente non codificano questi identificatori nè usano spazi bianchi in questo modo.

Gli spazi bianchi nel valore degli attributi vengono considerati in accordo a [XML].

4. Differenze con HTML 4

Poichè XHTML è un'applicazione XML, certi usi che erano perfettamente legali in HTML 4.0 basato su SGML [HTML] devono essere cambiati.

4.1 I documenti devono essere ben formati

Ben-formato è un concetto introdotto da [XML]. Sostanzialmente questo significa che tutti gli elementi devono avere il tag di chiusura o devono essere scritti in una forma speciale (come descritto sotto), e che tutti gli elementi devono essere annidati.

Sebbene la sovrapposizione sia illegale in SGML, è stata ampiamente tollerata dai browser esistenti.

CORRETTO: elementi annidati.

```
<p>here is an emphasized <em>paragraph</em>.</p>
```

SBAGLIATO: elementi sovrapposti

```
<p>here is an emphasized <em>paragraph.</p></em>
```

4.2 Gli elementi e i nomi degli attributi devono essere in lettere minuscole

I documenti XHTML devono usare lettere minuscole per tutti gli elementi HTML e per i nomi degli attributi. Questa differenza è necessaria perchè XML è sensibile alle minuscole e alle maiuscole, per esempio <li> e <LI> sono tag diversi.

4.3 Gli elementi non vuoti richiedono il tag di chiusura

In HTML 4.0 basato su SGML alcuni elementi potevano omettere il tag di chiusura, in modo tale che gli elementi che seguivano implicavano tale chiusura. Questa omissione non è permessa in XHTML basato su XML. Tutti gli elementi, ad eccezione di quelli dichiarati come `EMPTY` nella DTD devono avere un tag di chiusura.

CORRETTO: elementi chiusi

** here is a paragraph. here is another paragraph. **

SBAGLIATO: elementi non chiusi

here is a paragraph. here is another paragraph.

4.4 I valori degli attributi devono sempre essere compresi fra doppi apici

Tutti i valori degli attributi devono essere compresi fra doppi apici, inclusi i valori numerici.

CORRETTO: valori di attributo tra doppi apici

<table rows="3">

SBAGLIATO: valori di attributo senza doppi apici

<table rows=3>

4.5 Minimizzazione degli attributi

XML non supporta la minimizzazione degli attributi. I valori degli attributi accoppiati devono essere scritti completamente. I nomi degli attributi come `compact` e `checked` non possono essere presenti negli elementi se non viene specificato il loro valore.

CORRETTO: attributi non minimizzati

<dl compact="compact">

SBAGLIATO: attributi minimizzati

<dl compact>

4.6 Elementi vuoti

Gli elementi vuoti devono avere un tag di chiusura o il tag iniziale deve terminare con `/>`. Per esempio `<br/>` o `<hr></hr>`. Vedere HTML Compatibility Guidelines per informazioni sul modo con cui poter assicurare la compatibilità retroattiva con gli user agent di HTML 4.

CORRETTO: tag vuoto chiuso

**
<hr/>**

SBAGLIATO: tag vuoto non chiuso

<hr>

4.7 La manipolazione di spazi bianchi nei valori degli attributi

Nei valori degli attributi, gli user agent elimineranno gli spazi bianchi iniziali e finali dai valori degli attributi e sostituiranno la sequenza di uno o più spazi bianchi (compreso il salto di linea) con un singolo spazio tra le parole (un carattere di spazio ASCII per le scritture di tipo occidentale). Vedere la Sezione 3.3.3 di [XML].

4.8 Gli elementi Script e Style

In XHTML gli elementi script e style vengono dichiarati come se avessere un contenuto di tipo `#PCDATA`. Come risultato `<` e `&` verranno trattati come inizio del marcatore, e entità quali `<` e `&` saranno riconosciute come entità di riferimento dal processore XML rispettivamente come `<` e `&`. Racchiudere il contenuto dell'elemento script o style dentro una sezione marcata `CDATA` evita il processamento di queste entità.

```
<script>
 <![CDATA[
 ... unescaped script content ...
 ]]>
 </script>
```

Le sezioni `CDATA` vengono riconosciute dal processore XML e appaiono come nodi nel Modello dell'Oggetto di Documento, vedi Section 1.3 delle Raccomandazioni DOM Livello 1 [DOM].Una alternativa è quella di usare documenti esterni per gli script e gli style.

4.9 Esclusioni dell'SGML

L'SGML dà allo scrittore di una DTD la possibilità di impedire che specifici elementi siano annidati in altri elementi. Queste proibizioni (chiamate "esclusioni") non sono possibili in XML.

Per esempio, la DTD Stretta di HTML 4 proibisce l'annidamento di un elemento '`a`' dentro un altro elemento '`a`' a qualsiasi profondità. Non è possibile dettagliare tale proibizione in XML. Anche se queste proibizioni non possono essere definite in una DTD, certi elementi non dovrebbero essere annidati. Un elenco di questi elementi e degli elementi che non dovrebbero essere annidati con loro si trova nella normativa in Appendix B.

4.10 Gli elementi con attributi 'id' e 'name'

HTML 4 definisce l'attributo `name` per gli elementi `a`, `applet`, `form`, `frame`, `iframe`, `img`, e `map`. HTML 4 introduce anche l'attributo `id`. Entrambi questi attributi sono disegnati per essere usati come identificatori di frammenti di informazioni.

In XML gli identificatori di frammenti sono di tipo `ID`, e ci può essere un solo attributo di tipo `ID` per elemento. Quindi, in XHTML 1.0 l'attributo `id` viene definito di tipo `ID`. Con l'obiettivo di assicurare che i documenti XHTML 1.0 siano documenti XML ben-formati, i documenti XHTML 1.0 DEVONO usare l'attributo `id` quando definiscono gli identificatori di frammento, compreso con elementi che storicamente usavano anche un attributo `name`. Vedere HTML Compatibility Guidelines per informazioni su come assicurare la compatibilità retroattiva delle ancore quando si servono di documenti XHTML come media di tipo `text/html`.

Nota che in XHTML 1.0 l'attributo `name` di questi elementi è formalmente proibito e verrà rimosso nella futura versione di XHTML.

5. Compatibilità: punti di attenzione

Sebbene non vi sia nessun obbligo per i documenti XHTML 1.0 per quanto riguarda la compatibilità con gli user agent esistenti, in pratica questo è facile da ottenere. Le linee guide per creare la compatibilità dei documenti si trovano in Appendix C.

5.1 Tipi di Media di Internet

Al momento della pubblicazione di questa raccomandazione, il MIME generale raccomandato per le applicazioni basate su XML è stato già deciso.

Comunque, i documenti XHTML che seguono le linee guida indicate in Appendix C, "Linee guida sulla compatibilità di HTML" possono essere etichettati con il tipo di Media per Internet "text/html", in quanto sono compatibili con la maggior parte dei browser HTML. Questo documento non presenta nessuna raccomandazione sull'etichetta MIME di altri documenti XHTML.

6. Direzioni future

XHTML 1.0 fornisce le basi per una famiglia di tipi di documento che estenderà e delimiterà XHTML per supportare un ampio insieme di nuovi dispositivi e applicazioni, definendo moduli e specificando un meccanismo per la combinazione di questi modelli. Questo meccanismo permetterà l'estensione e la delimitazione di XHTML 1.0 in maniera uniforme attraverso la definizione di nuovi moduli.

6.1 Modularizzare HTML

Nel momento in cui XHTML si sposterà dallo user agent del tradizionale desktop ad altre piattaforme, è chiaro che non tutti gli elementi di XHTML saranno necessari in tutte le piattaforme. Per esempio un dispositivo manuale o un telefono cellulare possono supportare solo un sottoinsieme di elementi XHTML.

Il processo di modularizzazione spezza XHTML in una serie di piccoli insiemi di elementi. Questi

elementi possono essere ricombinati per raggiungere i bisogni delle diverse comunità.

Questi moduli saranno definiti in un futuro documento del W3C.

6.2 Sottoinsiemi e estensibilità

La modularizzazione si porta dietro molti vantaggi:

- Fornisce un meccanismo formale per il sottoinsieme di XHTML.
- Fornisce un meccanismo formale per l'estensione di XHTML
  .
- Semplifica la trasformazione tra i tipi di documenti.
- Promuove il riutilizzo di moduli in nuovi tipi di documenti.

6.3 Profili del documento

Il profilo di un documento specifica la sintassi e la semantica di un insieme di documenti. La conformità ad un profilo di documento fornisce la base per garantire l'interoperabilità. Il profilo del documento specifica le caratteristiche richieste per processare i documenti di questo tipo, per esempio quale formato di immagine può essere usato, i livelli di scripting, il supporto degli style sheet, e così via.

Per i designer dei prodotti questo permette di definire il proprio profilo standard dei vari gruppi.

Per gli autori, questo permette di evitare di scrivere versioni differenti dei documenti per diversi clienti.

Per gruppi speciali come i chimici, i medici o i matematici, questo permette di costruire un profilo speciale usando elementi standard di HTML insieme ad un gruppo di elementi appositamente disegnati per le specifiche necessità.

Appendice A. DTD

Questa appendice è normativa.

Queste DTD e gli insiemi di entità formano una parte normativa di questa specifica. L'insieme completo dei file delle DTD insieme con la dichiarazione XML e il Catalogo Aperto SGML è incluso nel file zip di questa specifica.

A.1 Definizione del Tipo di Documento

Queste DTD si avvicinano alle DTD di HTML 4. Verosimilmente quando queste DTD verranno modularizzate, verrà adottato un modo di costruzione delle DTD che corrisponda più strettamente a HTML 4.0.

XHTML-1.0-Strict

XHTML-1.0-Transitional

XHTML-1.0-Frameset

A.2 Insieme delle entità

Gli insiemi di entità di XHTML sono gli stessi di HTML 4, ma sono stati modificati in modo da essere dichiarazioni di entità valide in XML 1.0. Nota che l'entità per il segno dell'Euro (`&euro;` or `&#8364;` or `&#x20AC;`) è stata definita come parte di caratteri speciali.

Latin-1 characters

Special characters

Symbols

Appendix B. Limitazioni degli elementi

Questa appendice è normativa.

I seguenti elementi hanno limitazioni sugli elementi che possono essere annidati (vedere la Sezione 4.9). Questa proibizione si applica a tutte le profondità di annidamento, cioè riguarda tutti gli elementi discendenti.

`a`
: non può contenere altri elementi `a`.

`pre`
: non può contenere gli elementi `img`, `object`, `big`, `small`, `sub`, or `sup`

`button`
: non può contenere gli elementi `input`, `select`, `textarea`, `label`, `button`, `form`, `fieldset`, `iframe` or `isindex`

`label`
: non può contenere altri elementi `label`

`form`
: non può contenere altri elementi `form`

Appendix C. Linee guida per la compatibilità con HTML

Questa appendice è normativa.

Questa appendice riassume le linee guida per gli autori che desiderano che i loro documenti XHTML sia visualizzabili con gli user agent HTML esistenti.

C.1 Istruzioni per il processo

E' necessario sapere che le istruzioni di processo sono visualizzabili con alcuni user agent. Nota comunque che quando la dichiarazione XML non viene inclusa in un documento, il documento può usare solo il carattere di default codificato da UTF-8 or UTF-16.

C.2 Elementi vuoti

Includere uno spazio bianco prima della barra e della parentesi angolare chiusa `/` e `>` di elementi vuoti, per esempio, `<br />`, `<hr />` e `<img src="karen.jpg" alt="Karen" />`. Inoltre, usare la

sintassi minimizzata per i tag degli elementi vuoti, per esempio, `<br />`, dato che la sintassi alternativa `<br></br>` permessa da XML dà risultati imprevisti con molti user agent esistenti.

C.3 Minimizzazione degli elementi e contenuto degli elementi vuoti

Data una istanza vuota di un elemento il cui modello di contenuto non è `EMPTY` (per esempio, un titolo vuoto o un paragrafo) non usare la forma mimimizzata (per esempio usa `<p> </p>` e non `<p />`).

C.4 Style sheet e script incorporati

Usare style sheet esterni se questi usano i caratteri `<` or `&` or `]]>` or `--`. Usare script esterni se questi usano i caratteri `<` or `&` or `]]>` or `--`. Nota che i parser XML possono rimuovere il contenuto dei commenti. Comunque, la pratica comune di "nascondere" gli script e gli style sheet all'interno di commenti rende il documento compatibile con le vecchie versioni di browser, ma non lo rende funzionante con implementazioni basate su XML.

C.5 Salto di linea all'interno dei valori dell'attributo

Evitare i salti di linea e i caratteri di spazi multipli all'interno dei valori dell'attributo. Questi sono manipolati in maniera inconsistente dagli user agent.

C.6 Isindex

Non comprendere più di un elemento `isindex` nel documento `head`. L'elemento `isindex` è sconsigliato in favore dell'elemento `input`.

C.7 Gli attributi *lang* e *xml:lang*

Usare entrambi gli attributi `lang` and `xml:lang` quando si vuole specificare il linguaggio di un elemento. Il valore dell'attributo `xml:lang` ha la precedenza.

C.8 Identificatori di frammenti

In XML gli URI [RFC2396] che terminano con gli identificatori di frammenti della forma `"#foo"` non si riferiscono a elementi con un attributo `name="foo"`; piuttosto si riferiscono a elementi con un attributo definito del tipo `ID`, per esempio l'attributo `id` in HTML 4. Molti client HTML esistenti non supportano l'uso degli attributi di tipo `ID` in questo modo, quindi devono essere sostituiti valori identici per entrambi questi attributi in modo da assicurare una compatibilità futura e retroattiva (e.g., `<a id="foo" name="foo">...</a>`).

Inoltre, poichè l'insieme dei valori permessi per gli attributi di tipo `ID` è molto più piccolo di quelli per il tipo `CDATA`, il tipo dell'attributo `name` è stato cambiato in `NMTOKEN`. Questo attributo è limitato in modo tale che possa avere solo gli stessi valori del tipo `ID`, o come la produzione `Name` in XML 1.0 Sezione 2.5, produzione 5. Sfortunatamente questa limitazione non può essere espressa nella DTD di XHTML 1.0. A causa di questo cambiamento, si deve avere molta attenzione quando si convertono i documenti HTML esistenti. I valori di questi attributi devono essere unici all'interno del documento, validi, ed ogni riferimento a questi identificatori di frammenti (sia interni che

esterni) dovrebbe essere aggiornato in modo che i valori vengano cambiati durante la conversione.

Notare, in fine, che XHTML 1.0 ha disapprovato l'uso dell'attributo `name` degli elementi `a`, `applet`, `form`, `frame`, `iframe`, `img`, e `map`, e sarà eliminato da XHTML nella versioni successive.

C.9 Codifica dei caratteri

Per specificare la codifica dei caratteri nel documento, usare sia la specifica dell'attributo di codifica nella dichiarazione xml (per esempio, `<?xml version="1.0" encoding="EUC-JP"?>`) sia una frase meta http-equiv (per esempio, `<meta http-equiv="Content-type" content='text/html; charset="EUC-JP"' />`). Ha la precedenza il valore dell'attributo di codifica dell'istruzione di processo xml.

C.10 Attributi booleani

Alcuni user agent HTML sono capaci di interpretare gli attributi booleani quando questi appaiono nella loro forma completa (non-minimizzata), come richiesto da XML 1.0. Notare che questo problema non ha effetto sugli user agent conformi a HTML 4.0. Sono coinvolti i seguenti attributi: `compact`, `nowrap`, `ismap`, `declare`, `noshade`, `checked`, `disabled`, `readonly`, `multiple`, `selected`, `noresize`, `defer`.

C.11 Il Document Object Model e XHTML

La Raccomandazione del Livello 1 del Document Object Model [DOM] definisce le interfacce del modello dell'oggetto documento per XML e HTML 4. Il modello di oggetto del documento di HTML 4 specifica che gli elementi e i nomi degli attributi di HTML vengono restituiti in lettere maiuscole. Il modello di oggetto del documento XML specifica che gli elementi e i nomi degli attributi vengono restituiti in maiuscolo o minuscolo a seconda del caso in cui erano specificati. In XHTML 1.0, gli elementi e gli attributi sono specificarti in lettere minuscole. Questa apparente differenza può spiegata in due modi:

1. Le applicazioni che permettono l'accesso a documenti XHTML come tipo di media su Internet `text/html` via DOM possono usare il DOM di HTML e possono così assicurarsi che gli elementi e i nomi degli attributi vengano ritornati in lettere maiuscole da queste interfacce.
2. Le applicazioni che permettono l'accesso a documenti XHTML come tipo di media su Internet `text/xml` o `application/xml`, possono usare anche il DOM di XML. Gli elementi e gli attributi saranno ritornati in lettere minuscole. Inoltre, alcuni elementi XHTML possono o no apparire nell'albero degli oggetti in quanto sono opzionali nel modello di contenuto (per esempio l'elemento `tbody` all'interno di `table`). Questo accade perchè in HTML 4.0 alcuni elementi possono essere minimizzati in modo tale che il tag di apertura e di chiusura sia entrambi omessi (una caratteristica di SGML). Questo non è possibile in XML. Piuttosto che richiedere agli autori di documenti di inserire elementi estranei, XHTML ha posto tali elementi come opzionali. Le applicazioni si devono adattare a questo.

C.12 L'uso del carattere ampersand (&) nei valori degli attributi

Quando un valore di attributo contiene un ampersand (&), questo può essere espresso come un riferimento ad una entità di tipo carattere (per esempio, "`&amp;`"). Per esempio, quando l'attributo `href` dell'elemento `a` si riferisce ad uno script CGI con dei parametri, deve essere espresso come `http://my.site.dom/cgi-bin/myscript.pl?class=guest&name=user` invece che

`http://my.site.dom/cgi-bin/myscript.pl?class=guest&name=user`.

C.13 Cascading Style Sheets (CSS) e XHTML

Le Raccomandazioni di livello 2 dei Cascading Style Sheet [CSS2] definiscono le proprietà di stile che vengono applicate all'albero di analisi di un documento HTML o XML. Le differenze nell'analisi produrranno risultati visivi o auditivi diversi a seconda del selector usato. La traccia seguente ridurrà questo effetto per i documenti che sono serviti senza modifiche in entrambi i tipi di media:

1. Gli style sheet CSS per XHTML dovrebbero usare i nomi degli elementi e degli attributi in lettere minuscole.
2. Nelle tabelle, l'elemento tbody verrà dedotto dall'analizzatore dello user agent HTML, ma non dall'analizzatore dello user agent XML. Quindi si dovrebbe sempre aggiungere esplicitamente l'elemento tbody se si vuole riferirsi a un selector CSS.
3. Dentro lo spazio dei nomi XHTML gli user agent si aspettano di riconoscere l'attributo "`id`" come un attributo di tipo `ID`. Quindi gli style sheet dovrebbero essere capaci di continuare usando la sintassi abbreviata del selector "#" anche se lo user agent non legge la DTD.
4. Dentro lo spazio dei nomi XHTML, gli user agent si aspettano di riconoscere l'attributo "class". Quindi gli style sheet dovrebbero essere capaci di continuare usando la sintassi abbreviata del selector ".".
5. I CSS definiscono diverse regole di conformità per i documenti HTML e XML; tenere in considerazione che le regole HTML si applicano ai documenti XHTML distribuiti come HTML e le regole XML si applicano ai documenti XHTML distribuiti come XML.

Appendix D. Ringraziamenti

Questa appendice è informativa.

Queste specifiche sono state scritte con la partecipazione dei membri del gruppo di lavoro 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

Appendix E. Riferimenti

Questa appendice è informativa.

[CSS2]
"Cascading Style Sheets, level 2 (CSS2) Specification", B. Bos, H. W. Lie, C. Lilley, I. Jacobs, 12 May 1998.
Ultima versione disponibile a: http://www.w3.org/TR/REC-CSS2

[DOM]
"Document Object Model (DOM) Level 1 Specification", Lauren Wood *et al.*, 1 October 1998.
Ultima versione disponibile a: http://www.w3.org/TR/REC-DOM-Level-1

[HTML]
"HTML 4.01 Specification", D. Raggett, A. Le Hors, I. Jacobs, 24 December 1999.
Ultima versione disponibile a: http://www.w3.org/TR/html401

[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.
Disponibile all'indirizzo: http://www.ietf.org/rfc/rfc2046.txt. Note that this RFC obsoletes RFC1521, RFC1522, and RFC1590.

[RFC2119]
"RFC2119: Key words for use in RFCs to Indicate Requirement Levels", S. Bradner, March 1997.
Disponibile all'indirizzo: http://www.ietf.org/rfc/rfc2119.txt

[RFC2376]
"RFC2376: XML Media Types", E. Whitehead, M. Murata, July 1998.
Disponibile all'indirizzo: http://www.ietf.org/rfc/rfc2376.txt

[RFC2396]
"RFC2396: Uniform Resource Identifiers (URI): Generic Syntax", T. Berners-Lee, R. Fielding, L. Masinter, August 1998.
Questo documento aggiorna gli RFC1738 e RFC1808.
Disponibile all'indirizzo: 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.
Ultima versione disponibile a: http://www.w3.org/TR/REC-xml

[XMLNAMES]
"Namespaces in XML", T. Bray, D. Hollander, A. Layman, 14 January 1999.
Lo spazio dei nomi XML fornisce un metodo semplice per qualificare i nomi usati nei documenti XML associandoli con lo spazio dei nomi identificati da una URI.
Ultima versione disponibile a: http://www.w3.org/TR/REC-xml-names

indice