

1.- Lenguajes de marcas.

Un "lenguaje de marcas" es un modo de codificar un documento donde, junto con el texto, se incorporan etiquetas, marcas o anotaciones con información adicional relativa a la estructura del texto o su formato de presentación. Permiten hacer explícita la estructura de un documento, su contenido semántico o cualquier otra información lingüística o extralingüística que se quiera hacer patente.

Todo lenguaje de marcas está definido en un documento denominado DTD (Document Type Definition). En él se establecen las marcas, los elementos utilizados por dicho lenguaje y sus correspondientes etiquetas y atributos, su sintaxis y normas de uso.

```
<carta>
  <fecha>22/11/2006</fecha>
  <presentacion>Estimado cliente:</presentacion>
  <contenido>bla bla bla bla ...</contenido>
  <firma>Don José Gutiérrez González</firma>
</carta>
```

Los lenguajes de marcas se utilizan para:

- Dar formato a los documentos de texto.
- Definir la estructura de los datos de un documento.
- Permitir el intercambio de ficheros entre diferentes aplicaciones y plataformas.

2.- XML (eXtensible Markup Language).

XML es un estándar, no una implementación concreta. Es un metalenguaje caracterizado por:

- Permitir definir etiquetas propias.
- Permitir asignar atributos a las etiquetas.
- Utilizar un esquema para definir de forma exacta las etiquetas y los atributos.
- La estructura y el diseño son independientes.

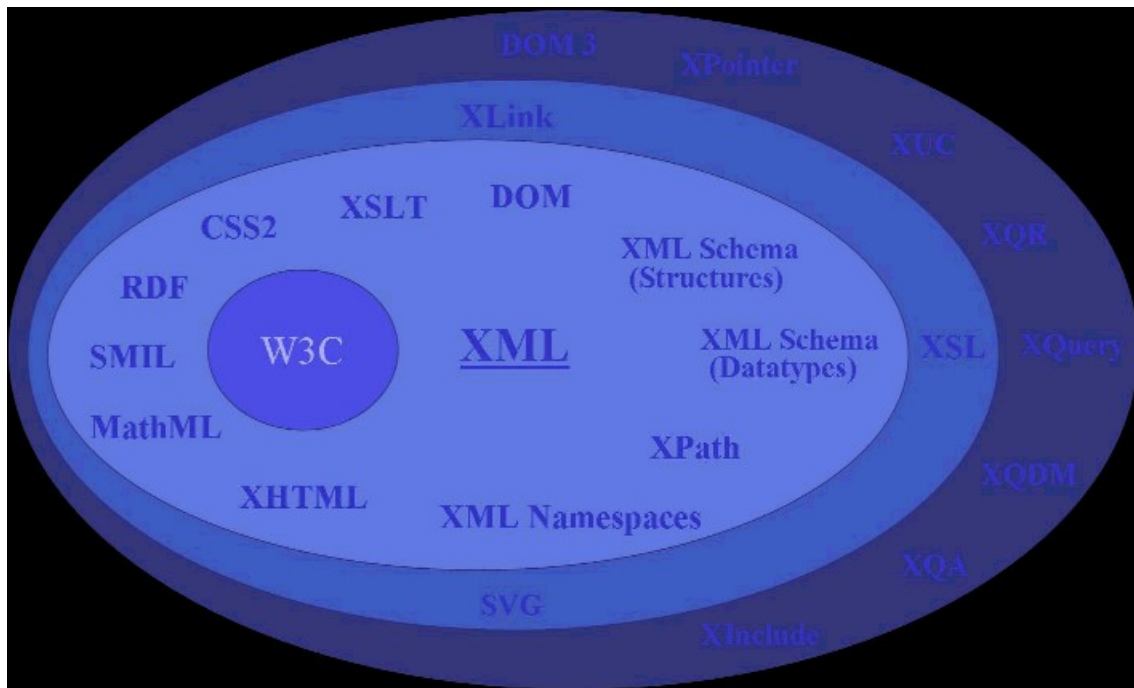
XML es un metalenguaje, es decir, puede ser empleado para definir otros lenguajes, llamados dialectos XML. Por ejemplo, algunos lenguajes basados en XML son:

- GML (Geography Markup Language, Lenguaje de Mercado Geográfico).
- MathML (Mathematical Markup Language, Lenguaje de Mercado Matemático).
- RSS (Really Simple Syndication, Sindicación Realmente Simple).
- SVG (Scalable Vector Graphics, Gráficos Vectoriales Escalables).
- XHTML (eXtensible HyperText Markup Language, Lenguaje de Mercado de Hipertexto eXtensible).
- etc.

XML no es:

- No es un lenguaje de programación, de manera que no existen compiladores de XML que generen ejecutables a partir de documentos XML.
- No es un protocolo de comunicación (como HTTP, FTP, etc).
- No es un sistema gestor de bases de datos.
- No es propietario, es decir, no pertenece a ninguna compañía.

En realidad, XML es un conjunto de estándares relacionados entre sí. La siguiente imagen nos muestra la estructura de componentes/recomendaciones que presenta XML:



El conjunto interior contiene las recomendaciones actuales, el intermedio las tecnologías candidatas a recomendación y el exterior son los documentos de trabajo actuales (working draft). Algunas de las tecnologías más importantes son:

- **XML (versión 1.0):** Identifica los requisitos de un documento XML bien formado, así como el origen y los objetivos del XML.

[<http://www.w3.org/XML>]

- **DTD:** Una definición de tipo de documento (*Document Type Definition*) contiene las reglas por las que es posible validar la información de un documento XML.

XML Linking Language, incluye Xpath, Xlink y Xpointer. Determinan aspectos sobre los enlaces entre documentos XML:

- **XLink** (*XML Linking Language*): conocido anteriormente conocido como **XLL**, define la forma en que los documentos XML deben enlazarse entre sí.

[<http://www.w3.org/TR/xlink/>]

- **XPointer**: describe cómo debe apuntarse a un lugar específico dentro de un documento.

[<http://www.w3.org/TR/xptr/>]

- **XPath**: proporciona un medio estándar de hacer referencia a direcciones y ubicaciones dentro de documentos XML así como de manipulación y comparación de cadenas, números y valores booleanos.

[<http://www.w3.org/TR/xpath/>]

- **CSS** (*Cascading Style Sheets*): define un modelo de formato visual con controles avanzados para medios paginados e impresos.

[<http://www.w3.org/Style/CSS/>]

- **XSL** (*eXtensible Stylesheet Language*): es un lenguaje diseñado específicamente para la creación de hojas de estilo con páginas XML. Se compone de: Un conjunto de propiedades y controles de formato para presentar la información de los documentos XML. Y **XSLT** (*XSL Transformations*) que define la sintaxis y la semántica que se utiliza para convertir documentos XML.

[<http://www.w3.org/Style/XSL/>]

- **XML Namespaces**: Proporciona las reglas para asociar nombres de elementos y atributos de XML a conjuntos de vocabularios identificados previamente.

[<http://www.w3.org/TR/1999/REC-xml-names-19990114/>]

- **XML Schemas**: permiten incorporar más funcionalidad en los documentos XML de la que proporcionan actualmente las DTD.

[<http://www.w3.org/TR/xmlschema-0/>
[<http://www.w3.org/TR/xmlschema-1/>
[<http://www.w3.org/TR/xmlschema-2/>]

- **XQL** (*XML Query Language*): proporciona un modelo de datos para su utilización con documentos XML, así como un lenguaje de consulta y un conjunto de operadores completos para el modelo de datos.

[<http://www.w3.org/TR/xmlquery-req/>]

- **Canonical XML:** el XML canónico crea un subconjunto del documento XML original que se puede comparar con otro documento que utilice una sintaxis igual o similar.

[<http://www.w3.org/TR/xml-c14n/>]

- **XHTML** (*eXtensible Hypertext Markup Language*): es una reformulación de la especificación 4.0 de HTML existente en forma de módulos que siguen las reglas impuestas por la especificación XML.

[<http://www.w3.org/TR/XHTML1/>]

[<http://www.w3.org/TR/XHTML-basic/>]

- **XMLDS** (*XML Digital Signatures*): describe las reglas de procesado y la sintaxis para la presentación de firmas digitales dentro de documentos XML.
[<http://www.w3.org/TR/xmlsig-core/>]

2.1.- Comparación de XML con HTML.

XML	HTML
✓ Es un perfil de SGML.	✓ Es una aplicación de SGML.
✓ Especifica cómo deben definirse conjuntos de etiquetas aplicables a un tipo de documento.	✓ Aplica un conjunto limitado de etiquetas sobre un único tipo de documento.
✓ Modelo de hiperenlaces complejo.	✓ Modelo de hiperenlaces simple.
✓ El navegador es una plataforma para el desarrollo de aplicaciones.	✓ El navegador es un visor de páginas.
✓ Fin de la guerra de los navegadores y etiquetas propietarias.	✓ El problema de la 'no compatibilidad' y las diferencias entre navegadores ha alcanzado un punto en el que la solución es difícil.

- HTML tiene una **orientación hacia la presentación**, descuidando otros factores importantes en un documento como son la estructura y la semántica. Las etiquetas de las que dispone HTML para mostrar la estructura del documento

son escasas, (por ejemplo: H1... H6) y no añaden información sobre el contenido del documento.

- HTML, a pesar de ser una aplicación de SGML y como tal disponer de medios para comprobar si la estructura de sus documentos es correcta, **no comprueba la buena formación** de los documentos y sus etiquetas. Los navegadores de HTML se diseñan para admitir casi cualquier cosa.
- HTML **no es internacional**, está basado en ASCII y la inclusión de caracteres de algunos lenguajes no es tan sencilla como cabría esperar.
- HTML **no es extensible**, al menos oficialmente, es decir, no tiene capacidad para adaptarse a un gran número de situaciones. No se diseñó para ello.
- HTML **no añade información sobre el contenido** de un documento, ni se marcan sintácticamente elementos lógicos de los documentos
- HTML **no es reutilizable**. Ésta es una consecuencia también de la ausencia de marcas que añadan semántica, no se pueden diferenciar unos elementos de otros, con lo que no se pueden utilizar en otros contextos.
- HTML **no es válido para intercambio de datos**, hace poco para identificar la información que un documento contiene.

En los siguientes códigos podemos observar algunas de las diferencias entre HTML y XML:

<pre><h1>Factura</h1> <p>De : Pepe</p> <p>a : Juan</p> <p>fecha : 1/02/1999</p> <p>cantidad:1000 ptas</p> <p>i.v.a.:16%</p> <p>total : 1160 ptas</p></pre>	<pre><?xml version="1.0"?> <factura> <de>Pepe</de> <a>Juan <fecha dd="1" mm="02" yy="1999"/> <cant. moneda="ptas">1000</cant.> <iva>16</iva> <total moneda="ptas">1160</total> </factura></pre>
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Ejemplo fichero XML:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE libro>
<libro>
  <titulo>XML practico </titulo>
  <autor>SebastienLecomte</autor>
  <autor>Thierry Boulanger</autor>
  <editorial>Ediciones Eni</editorial>
  <isbn>978-2-7460-4958-1</isbn>
  <edicion>1</edicion>
  <paginas>347</paginas>
</libro>
```

Al interpretar este fichero con un navegador se obtiene:

El presente archivo XML no parece tener ninguna información de estilo asociada. A continuación se muestra su árbol.

```
- <libro>
  <titulo>XML practico </titulo>
  <autor>SebastienLecomte</autor>
  <autor>Thierry Boulanger</autor>
  <editorial>Ediciones Eni</editorial>
  <isbn>978-2-7460-4958-1</isbn>
  <edicion>1</edicion>
  <paginas>347</paginas>
</libro>
```

3.- Herramientas de edición.

Para trabajar en XML es necesario editar los documentos y luego procesarlos, por tanto, tenemos dos tipos de herramientas:

1. Editores XML

Una característica de los lenguajes de marcas es que se basan en la utilización de ficheros de **texto plano** por lo que basta utilizar un procesador de texto normal y corriente para construir un documento XML.

Para crear documentos XML complejos e ir añadiendo datos es conveniente usar algún editor XML. Estos nos ayudan a crear estructuras y etiquetas de los elementos usados en los documentos, además algunos incluyen ayuda para la creación de otros elementos como DTD, hojas de estilo CSS o XSL, ... El W3C ha desarrollado un editor de HTML, XHTML, CSS y XML gratuito cuyo nombre es Amaya. Otras herramientas son XMLSpy (<http://www.altova.com/es>), <Oxygen/> (<http://www.oxygenxml.com>), XML Copy Editor (<http://xml-copy-editor.sourceforge.net>), XMLPad Pro Edition (<http://www.wmhelp.com/download.html>) ...

2. Procesadores XML

Para interpretar el código XML se puede utilizar cualquier navegador. Los procesadores de XML permiten leer los documentos XML y acceder a su contenido y estructura. Un procesador es un conjunto de módulos de software entre los que se encuentra un parser o analizador de XML que comprueba que el documento cumple las normas establecidas para que pueda abrirse. Estas normas pueden corresponderse con las necesarias para trabajar sólo con documentos de tipo válido o sólo exigir que el documento esté bien formado, primeros se conocen como validadores y los segundos como no validadores. El modo en que los procesadores deben leer los datos XML está descrito en la recomendación de XML establecida por W3C.

Si el analizador comprueba las reglas de buena formación y además valida el documento contra un DTD o esquema, se dice que se trata de un **analizador validador**.

Existen analizadores XML en línea, como XML Validation (<http://www.xmlvalidation.com>).

Para publicar un documento XML en Internet se utilizan los procesadores XSLT, que permiten generar archivos HTML a partir de documentos XML.

Puesto que XML se puede utilizar para el intercambio de datos entre aplicaciones, hay que recurrir a motores independientes que se ejecutan sin que nos demos cuenta. Entre estos destacan "XML para Java" de IBM, JAXP de Sun, etc.

4.- XML: sintaxis

En un documento XML, todos los nombres de los elementos son case sensitive, es decir, sensibles a letras minúsculas y mayúsculas, teniendo que cumplir las siguientes normas:

- Pueden contener letras (con o sin tilde) minúsculas, letras mayúsculas, números, puntos ".", guiones medios "-" y guiones bajos "_".
- Asimismo, pueden contener el carácter dos puntos ":". No obstante, su uso se reserva para cuando se definan espacios de nombres.
- El primer carácter tiene que ser una letra o un guion bajo "_".
- Los nombres que empiezan por XML, en cualquier combinación de mayúsculas y minúsculas, se reserva para estandarización.

No pueden contener:

- Ningún carácter de espaciado.
- Ningún otro carácter de puntuación de los ya citados. Esto incluye: comillas simples o dobles, signo de dólar, acento circunflejo, signo de porcentaje, punto y coma.

Por otra parte, hay que tener en cuenta que, **detrás del nombre de una etiqueta se permite escribir un espacio en blanco o un salto de línea**. Por ejemplo, sintácticamente es correcto escribir:

```
<ciudad >Pamplona</ciudad >
```

Ahora bien, **no puede haber un salto de línea o un espacio en blanco antes del nombre de una etiqueta**:

```
< ciudad>Pamplona</ ciudad>
```

EJEMPLO: Los siguientes elementos no están escritos correctamente por incumplir alguna regla de sintaxis:

```
<Ciudad>Pamplona</ciudad>
<día>18</día>
<mes>6<mes/>
<ciudad>Pamplona</finciudad>
<_rojo>
<2colores>Rojo y Naranja</2colores>
< Aficiones >Cine, Bailar, Nadar</ Aficiones >
<persona><nombre>Elsa</persona></nombre>
<color favorito>azul</color favorito>
```

En su lugar, sería correcto escribir:

```
<Ciudad>Pamplona</Ciudad>
<día>18</día>
<mes>6</mes>
<ciudad>Pamplona</ciudad>
<_rojo/>
<colores2>Rojo y Naranja</colores2>
<Aficiones >Cine, Bailar, Nadar</Aficiones >
<persona><nombre>Elsa</nombre></persona>
<color.favorito>azul</color.favorito>
<color-favorito>azul</color-favorito>
<color_favorito>azul</color_favorito>
```

Las letras no inglesas (á, Á, ñ, Ñ ...) están permitidas. Sin embargo, es recomendable no utilizarlas para reducir posibles incompatibilidades con programas que puedan no reconocerlas.

En cuanto al carácter guion medio “-” y al punto “.”, aunque también están permitidos para nombrar etiquetas, igualmente se aconseja evitar su uso; el guion medio porque podría confundirse con el signo menos, y el punto porque, por ejemplo al escribir *color.favorito*, podría interpretarse que *favorito* es una propiedad del objeto *color*.

5.- XML: estructura.

El XML, o Lenguaje de Etiquetas Extendido, es lenguaje de etiquetas, creadas por el programador, que estructuran y guardan de forma ordenada la información. No representa datos por sí mismo, solamente organiza la estructura.

El XML ahorra tiempos de desarrollo y proporciona ventajas, dotando a webs y a aplicaciones de una forma realmente potente de guardar la información. Además, se ha convertido en un formato universal que ha sido asimilado por todo tipo de sistemas operativos y dispositivos móviles.

Al igual que en HTML un documento XML es un documento de texto, en este caso con extensión ".xml", compuesto de parejas de etiquetas, estructuradas en árbol, que describen una función en la organización del documento, que puede editarse con cualquier editor de texto y que es interpretado por los navegadores Web.

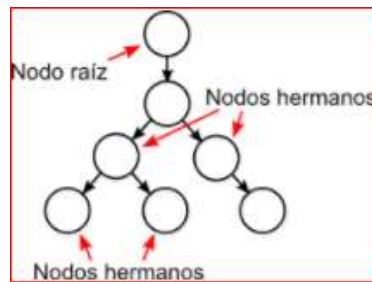
Las características básicas de XML son:

- Dado que XML se concibió para trabajar en la Web, es directamente compatible con protocolos que ya funcionan, como HTTP y los URL.
- Todo documento que verifique las reglas de XML está conforme con SGML.
- No se requieren conocimientos de programación para realizar tareas sencillas en XML.
- Los documentos XML son fáciles de crear.

- La difusión de los documentos XML está asegurada ya que cualquier procesador de XML puede leer un documento de XML.
- El marcado de XML es legible para los humanos.
- El diseño XML es formal y conciso.
- XML es extensible, adaptable y aplicable a una gran variedad de situaciones.
- XML es orientado a objetos.
- Todo documento XML se compone exclusivamente de datos de marcado y datos carácter entremezclados.

En un documento XML la información se organiza de forma jerárquica, de manera que los elementos del documento se relacionan entre sí mediante relaciones de padres, hijos, hermanos, descendientes...

Por encima de cualquier elemento se ubica el nodo raíz, que se designa como “/”. No es un componente que tenga representación dentro del documento XML, pero se utilizará más adelante como punto de partida para recorrer el árbol XML y ubicar el resto de nodos.



Relaciones entre nodos

A esta estructura jerárquica se la llama árbol del documento XML. A las partes del árbol que tienen hijos se las denomina nodos intermedios o ramas, mientras que las que no los tienen se conocen como nodos finales u hojas.

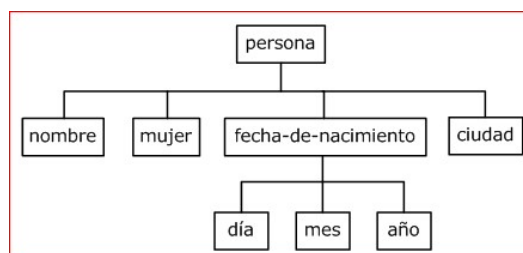


Diagrama de árbol

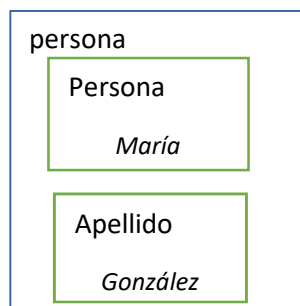
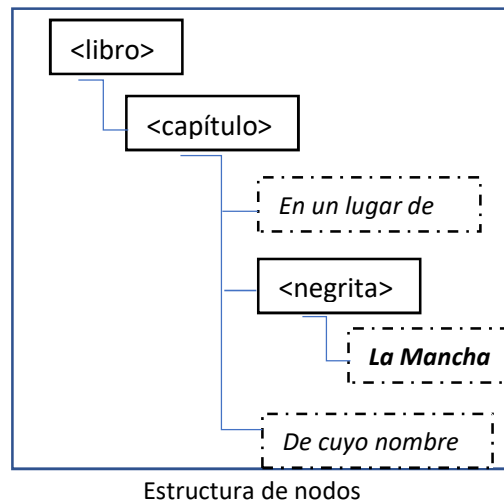


Diagrama de cajas



El proceso de creación de un documento XML pasa por varias etapas en las que el éxito de cada una de ellas se basa en la calidad de la anterior. Estas etapas son:

- Especificación de requisitos.
- Diseño de etiquetas.
- Marcado de los documentos.
- El marcado en XML son etiquetas que se añaden a un texto para estructurar el contenido del documento. Esta información extra permite a los ordenadores "interpretar" los textos. El marcado es todo lo que se sitúa entre los caracteres "<" y ">" o "&" y ";"

Los datos carácter son los que forman la verdadera información del documento XML. El marcado puede ser tan rico como se quiera. Puede ser interesante detectar necesidades futuras y crear documentos con una estructura fácilmente actualizables.

Los documentos XML pueden tener **comentarios**, que no son interpretados por el intérprete XML. Estos se incluyen entre las cadenas "<!--" y "-->", pueden estar en cualquier posición en el documento salvo:

- Antes del prólogo.
- Dentro de una etiqueta.

```
<!-- Esto es un comentario escrito en un documento XML -->
```

En un documento XML, no se pueden escribir comentarios dentro de las etiquetas. Por ejemplo, no es correcto escribir:

```
<mujer <!-- elemento vacío --> />
```

Por otro lado, hay que tener en cuenta que en los comentarios de un documento XML no está permitido usar dos guiones seguidos:

```
<!-- Dos guiones seguidos -- en un comentario da error -->
```

De forma que, **no es posible anidar comentarios en un documento XML.**

Los documentos XML pueden estar formados por una parte opcional llamada prólogo y otra parte obligatoria llamada ejemplar.

5.1.- El prólogo.

Si se incluye, el prólogo **debe preceder al ejemplar del documento**. Informa al intérprete encargado de procesar el documento de todos aquellos datos que necesita para realizar su trabajo. Su inclusión facilita el procesado de la información del ejemplar. El prólogo está dividido en dos partes:

- **La declaración XML:** En un documento XML no es obligatorio que aparezca la **declaración XML**. Ahora bien, si se incluye, tiene que aparecer en la primera línea del documento, y el carácter "<" debe ser el primero de dicha línea, es decir, antes no pueden aparecer espacios en blanco. De no ser así se genera un error que impide que el documento sea procesado.

El hecho de que sea opcional permite el procesamiento de documentos HTML y SGML como si fueran XML, si fuera obligatoria éstos deberían incluir una declaración de versión XML que no tienen.

El prólogo puede tener tres funciones:

- *Declaración la versión de XML usada para elaborar el documento.* Para ello se utiliza la etiqueta:

```
<?xml versión= "1.0" ?>
```

En este caso indica que el documento fue creado para la versión 1.0 de XML.

- *Declaración de la codificación empleada para representar los caracteres.* Determina el conjunto de caracteres que se utiliza en el documento. Para ello se escribe:

```
<?xml versión= "1.0" encoding="iso-8859-1" ?>
```

En este caso se usa el código iso-8859-1 (Latin-1) que permite el uso de acentos o caracteres como la ñ. Los códigos más importantes son:

Estándar ISO	Código de país
UTF-8 (Unicode)	Conjunto de caracteres universal
ISO -8859-1 (Latin-1)	Europa occidental, Latinoamérica
ISO -8859-2 (Latin-2)	Europa central y oriental
ISO -8859-3 (Latin-3)	Sudoeste de Europa
ISO -8859-4 (Latin-4)	Países Escandinavos, Bálticos
ISO -8859-5	Cirílico
ISO -8859-6	Árabe
ISO -8859-7	Griego
ISO -8859-8	Hebreo
ISO -8859-9	Turco
ISO-8859-10	Lapón. Nórdico, esquimal
EUC-JP oder Shift_JIS	Japonés

- *Declaración de la autonomía del documento.* Informa de si el documento necesita de otro para su interpretación. Para declararlo hay que definir el prólogo completo:

```
<?xml versión= "1.0" encoding="iso-8859-1" standalone="yes" ?>
```

Al escribir **standalone="yes"** se está indicando que el documento es independiente de otros, como por ejemplo de una **DTD** (*Document Type Definition*, Definición de Tipo de Documento) externa. En caso contrario, significará que el documento no es independiente.

<http://developerji.com/Post/El-elemento-DOCTYPE-de-html5/4169>

En un documento XML, escribir la declaración XML es opcional. Pero, si se escribe, el atributo **version** es obligatorio indicarlo. Sin embargo, los atributos **encoding** y **standalone** son opcionales y, por defecto, sus valores son **"UTF-8"** y **"no"**, respectivamente. Por otra parte, cuando se escriba el atributo **encoding**, siempre deberá aparecer después de **version**. Y, respecto al atributo **standalone**, siempre que se escriba, deberá ser en último lugar.

La **declaración del tipo de documento**, define qué tipo de documento estamos creando para ser procesado correctamente. El nombre del tipo ha de ser idéntico al del ejemplar del documento XML en el que se está trabajando. Toda declaración de tipo de documento comienza por la cadena:

```
<!DOCTYPE Nombre_tipo ...>
```

La definición del tipo de documento. Permite asociar al documento una definición de tipo DTD, la cual se encarga de definir las cualidades del tipo. Es decir, define los tipos de los elementos, atributos y notaciones que se pueden utilizar en el documento, así como las restricciones del documento, valores por defecto, etc. Para formalizar todo esto, XML está provisto de ciertas estructuras llamadas **declaraciones de marcado**, las cuales pueden ser internas o externas.

Normalmente un documento XML se compone de una mezcla de declaraciones de marcado internas y externas. En este último caso debe expresarse en el documento dónde encontrar las declaraciones, así como indicar en la declaración de XML que el documento no es autónomo. Las diferencias entre estos tipos de declaraciones de marcado dan lugar a dos subconjuntos, el interno y el externo, conviene saber que primero se procesa el subconjunto interno y después el externo, lo que permite sobrescribir declaraciones externas compartidas entre varios documentos y ajustar el DTD a un documento específico.

En función del tipo de DTD la sintaxis varía. Las características que definen el tipo son:

- **UBICACIÓN:** dónde se localizan las reglas del DTD.
 - **Interno:** las reglas aparecen en el propio documento DTD.
 - **Externo:** las reglas aparecen en un archivo independiente.
 - **Mixto:** mezcla de los anteriores, las reglas aparecen en ambos lugares. Las reglas internas tienen prioridad sobre las externas.
- **CARÁCTER:** si es DTD para uso privado o público.
 - **Privado:** se identifica con la palabra **SYSTEM**.
 - **Público:** se identifica con la palabra **PUBLIC**. Debe ir acompañado del **FPI** (Formal Public Identifier – Identificador Público Formal), una etiqueta que identifica al DTD de manera “universal”.

Las **distintas** combinaciones son (veremos ejemplos más adelante):

*** Sintaxis DTD interna (privada)**

```
<!DOCTYPE elemento-raíz [ declaraciones ]>
```

*** Sintaxis DTD externa privada:**

```
<!DOCTYPE elemento-raíz SYSTEM "URI">
```

*** Sintaxis DTD externa pública:**

```
<!DOCTYPE elemento-raíz PUBLIC "FPI" "URI">
```

*** Sintaxis DTD mixta y privada:**

```
<!DOCTYPE elemento-raíz SYSTEM "URI" [ declaraciones ]>
```

*** Sintaxis DTD mixta y pública:**

```
<!DOCTYPE elemento-raíz PUBLIC "FPI" "URI" [ declaraciones ]>
```

Estructura del FPI

Está compuesto de 4 campos separados por //.

- **Primer campo: norma formal o no formal:**
 - - → Si el DTD no ha sido aprobado por una norma formal, como por ejemplo un DTD escrito por uno mismo.
 - + → Si ha sido aprobado por un organismo no oficial.
 - **Referencia al estándar** → Si ha sido aprobado por un organismo oficial.
- **Segundo campo: nombre del organismo responsable del estándar:**
- **Tercer campo: tipo del documento que se describe. Suele incluir la versión.**
- **Cuarto campo: idioma del DTD.**

1 2 3 4
"-//W3C//DTD HTML 4.01//ES"

URI

Solo existe cuando el DTD (en parte o en totalidad) se encuentra declarado en un documento externo, del que se da su ubicación.

Subconjunto interno: Contiene las **declaraciones que pertenecen exclusivamente a un documento** y no es posible compartirlas. Se localizan dentro de unos corchetes que siguen a la declaración de tipo del documento.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE marcadores [
    <!ELEMENT marcadores (pagina)*>
    <!ELEMENT pagina (nombre, descripcion, url)>
    <!ELEMENT nombre (#PCDATA)>
    <!ELEMENT descripcion (#PCDATA)>
    <!ELEMENT url (#PCDATA)>
]>
<marcadores>
...
</marcadores>
```

Subconjunto externo: Están localizadas en un documento con extensión **dtd** que puede situarse en el mismo directorio que el documento XML. Habitualmente son declaraciones que pueden ser compartidas entre múltiples documentos XML que pertenecen al mismo tipo. En este caso la declaración de documento autónomo ha de ser negativa, ya que es necesario el fichero del subconjunto externo para la correcta interpretación del documento. Con ello el procesado del documento será más lento, ya que antes de procesar el documento el procesador ha de obtener todas las entidades.

<!DOCTYPE nombre_ejemplar SYSTEM "URI"

En este caso, se especifica un URI donde pueden localizarse las declaraciones.

<!DOCTYPE nombre_ejemplar PUBLIC "id_publico" "URI"

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

En este caso también se especifica un identificador, que puede ser utilizado por el procesador XML para intentar generar un URI alternativo, posiblemente basado en alguna tabla. **Como se puede observar también es necesario incluir algún URI.** Ahora los corchetes pierden sentido.

En este ejemplo, tenemos un DTD mixto y privado, donde charlas.dtd se encuentra en el mismo directorio que el archivo xml.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE charlas SYSTEM "charlas.dtd" [
    <!ELEMENT charlas (charla)+>
    <!ELEMENT charla (nombre, lugar, despedida)>
    <!ENTITY despedidaInglesa "Thank you, good bye!">
    <!ENTITY despedidaFrancesa "Merci, au revoir!">
]>
<charlas>
...
</charlas>
```

El contenido del archivo charlas.dtd será:

```
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT lugar (#PCDATA)>
<!ELEMENT despedida (#PCDATA)>
```

Observar que el archivo externo no incluye la declaración del tipo de documento (DOCTYPE), que solo aparece en la cabecera del documento XML. Resaltar que el atributo **standalone** tiene el valor **no**, lo que significa que para el correcto procesamiento del documento necesitamos el uso de otros documentos externos (en este caso, charlas.dtd).

Más adelante veremos que hay cuatro tipos posibles de componentes que se pueden declarar en un DTD:

- Elemento
- Atributo
- Entidad
- Notación

5.2.- El ejemplar. Los elementos.

Es la parte más importante de un documento XML, ya que contiene los **datos reales del documento**. **Es el elemento raíz del documento y ha de ser único**. Está formado por elementos anidados según una estructura de árbol en la que el elemento raíz es el ejemplar y las hojas los elementos terminales, es decir, aquellos que no contienen elementos. Los elementos pueden estar a su vez formados por atributos.

Los elementos son los distintos bloques de información que permiten definir la estructura de un documento XML. Están delimitados por una etiqueta de apertura y

una etiqueta de cierre. A su vez los elementos pueden estar formados por otros elementos y/o por atributos. Por ejemplo:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE libro>
<libro>
  <titulo>XML practico </titulo>
  <autor>Sebastien Lecomte</autor>
  <autor>Thierry Boulanger</autor>
  <editorial>Ediciones Eni</editorial>
  <isbn>978-2-7460-4958-1</isbn>
  <edicion>1</edicion>
  <paginas>347</paginas>
</libro>
```

El ejemplar es el elemento <libro>, que a su vez está compuesto de los elementos <autor>, <editorial>, <isbn>, <edicion> y <paginas>.

En realidad, el ejemplar es el elemento raíz de un documento XML. Todos los datos de un documento XML han de pertenecer a un elemento del mismo.

Los nombres de las etiquetas han de ser autodescriptivos, lo que facilita el trabajo que se hace con ellas. La formación de elementos ha de cumplir ciertas normas para que queden perfectamente definidos y que el documento XML al que pertenecen pueda ser interpretado por los procesadores XML sin generar ningún error fatal. Dichas reglas son:

- En todo documento XML debe existir un elemento raíz, y sólo uno.
- Todos los elementos tienen una etiqueta de inicio y otra de cierre. En el caso de que en el documento existan elementos vacíos, se pueden sustituir las etiquetas de inicio y cierre por una de elemento vacío. Ésta se construye como la etiqueta de inicio, pero sustituyendo el carácter ">" por "/>". Es decir, <elemento> </elemento> puede sustituirse por: <elemento/>
- Al anidar elementos hay que tener en cuenta que no puede cerrarse un elemento que contenga algún otro elemento que aún no se haya cerrado. Los nombres de las etiquetas de inicio y de cierre de un mismo elemento han de ser idénticos, respetando las mayúsculas y minúsculas. Pueden ser cualquier cadena alfanumérica que no contenga espacios y no comience ni por el carácter dos puntos, ":", ni por la cadena "xml" ni ninguna de sus versiones en que se cambien mayúsculas y minúsculas ("XML", "XmL", "xML", ...).
- El contenido de los elementos no puede contener la cadena "]]>" por compatibilidad con SGML. Además, no se pueden utilizar directamente los caracteres: mayor que, >, menor que, <, ampersand, &, dobles comillas, ", y apostrofe, '. En el caso de tener que utilizar estos caracteres se sustituyen por las siguientes cadenas:

Carácter	Cadena	Carácter	Cadena	Carácter	Cadena
>	>	&	&	'	'
<	<	"	"		

- Para utilizar caracteres especiales, como £, ©, ®, ... hay que usar las expresiones &#D; o &#H; donde D y H se corresponden respectivamente con el número decimal o hexadecimal correspondiente al carácter que se quiere representar en el código UNICODE. Por ejemplo, para incluir el carácter de Euro, €, se usarían las cadenas € o €

5.3.- Atributos.

Un atributo sirve para proporcionar información extra sobre el elemento que lo contiene. Permiten añadir propiedades a los elementos de un documento. Los atributos no pueden organizarse en ninguna jerarquía, no pueden contener ningún otro elemento o atributo y no reflejan ninguna estructura lógica.

No se debe utilizar un atributo para contener información susceptible de ser dividido.

Ejemplo:

```
<?xml version="1.0" encoding="iso-8859-1" standalone="yes" ?>
<!DOCTYPE biblioteca>
<biblioteca>
  <ejemplar tipo Ejem="libro" titulo="XML práctico" editorial="Ediciones Eni">
    <tipo>
      <libro isbn="978-2-7460-4958-1" edicion="1" paginas="347"></libro>
    </tipo>
    <autor nombre="Sebastien Lecomte"></autor>
    <autor nombre="Thierry Boulanger"></autor>
    <autor nombre="Angel Belinchon Calleja" funcion="traductor"></autor>
    <prestado lector="Pepito Grillo">
      <fecha_pres dia="13" mes="mar" año="2009"></fecha_pres>
      <fecha_devol dia="21" mes="jun" año="2009"></fecha_devol>
    </prestado>
  </ejemplar>
</biblioteca>
```

Al abrir el documento anterior con un navegador obtenemos:

```
-<biblioteca>
- <ejemplar tipo Ejem="libro" titulo="XML práctico" editorial="Ediciones Eni">
- <tipo>
  <libro isbn="978-2-7460-4958-1" edicion="1" paginas="347"/>
</tipo>
<autor nombre="Sebastien Lecomte"/>
<autor nombre="Thierry Boulanger"/>
<autor nombre="Angel Belinchon Calleja" funcion="traductor"/>
- <prestado lector="Pepito Grillo">
  <fecha_pres dia="13" mes="mar" año="2009"/>
  <fecha_devol dia="21" mes="jun" año="2009"/>
</prestado>
</ejemplar>
</biblioteca>
```

Vemos que los elementos aparecen coloreados en ciruela, los nombres de los atributos en negro y sus valores en azul.

Como se observa en el ejemplo, los atributos se definen y dan valor dentro de una etiqueta de inicio o de elemento vacío, a continuación del nombre del elemento o de la definición de otro atributo siempre separado de ellos por un espacio. Los valores del atributo van precedidos de un igual que sigue al nombre del mismo y tienen que definirse entre comillas simples o dobles.

Los nombres de los atributos han de cumplir las mismas reglas que los de los elementos, y no pueden contener el carácter menor que, <.

Sabemos que los atributos no pueden tener nodos que dependan de ellos, por tanto, solo pueden corresponder con hojas de la estructura de árbol que jerarquiza los datos. ¿Significa esto que todas las hojas van a ser atributos? Pues no, es cierto que los atributos son hojas, pero las hojas pueden ser atributos o elementos.

En ese caso, ¿qué criterios podemos utilizar para decidir si un dato del documento que se pretende estructurar ha de representarse mediante un elemento o un atributo? Aunque no siempre se respetan, podemos usar los siguientes criterios:

El dato será un elemento si cumple alguna de las siguientes condiciones:

- Se emplean para representar jerarquías o contenido de unos dentro de otros.
- Se pueden extender con otros elementos en su interior.
- Es de un tamaño considerable.
- Su valor cambia frecuentemente.
- Su valor va a ser mostrado a un usuario o aplicación.
- El orden en el que aparecen es representativo.
- Pueden tener atributos.
- Puede haber múltiples ocurrencias de un elemento.

Los casos en los que el dato será un atributo son:

- Van asociados a los elementos.
- Son modificadores de la información.
- Se suelen usar para registrar metadatos.
- El orden en el que aparecen dentro del elemento al que van asociados no es representativo.
- No se pueden extender con otros elementos contenidos en su interior.
- No puede haber múltiples ocurrencias de un atributo dentro de un mismo elemento.
- El dato es de pequeño tamaño y su valor raramente cambia, aunque hay situaciones en las que este caso puede ser un elemento.
- El dato solo puede tener unos cuantos valores fijos.
- El dato guía el procesamiento XML pero no se va a mostrar.

Normas de sintaxis: Los nombres de los atributos deben cumplir las mismas normas de sintaxis que los nombres de los elementos. Además, **todos los atributos de un elemento tienen que ser únicos**. Por ejemplo, es incorrecto escribir:

```
<datos x="3" x="4" y="5"/>
```

Sin embargo, sí es correcto escribir:

```
<datos x="3" X="4" y="5"/>
```

Los atributos contenidos en un elemento, como en este caso x, X e y, deben separarse con espacios en blanco, no siendo significativo su orden.

6. Referencias a entidades

En XML existen algunos caracteres que son especiales por su significado y, para escribirlos en un documento XML, se pueden utilizar las referencias a entidades mostradas en la siguiente tabla:

Referencias a entidades en XML		
Carácter	Entidad	Referencia a entidad
< (menor que)	lt (less than)	<
> (mayor que)	gt (greater than)	>
" (comilla doble)	quot (quotation mark)	"
' (comilla simple)	apos (apostrophe)	'
& (ampersand)	amp (ampersand)	&

EJEMPLO: Dado el archivo “*entidades.xml*”:

```
<?xml version="1.0" encoding="UTF-8"?>
<entidades>
  <menor_que>&lt;</menor_que>
  <mayor_que>&gt;</mayor_que>
  <comilla_doble>&quot;</comilla_doble>
  <comilla_simple>&apos;</comilla_simple>
  <ampersand>&amp;</ampersand>
</entidades>
```

Al abrirlo en *Google Chrome* se podrá visualizar:

```
▼ <entidades>
  <menor_que><</menor_que>
  <mayor_que>></mayor_que>
  <comilla_doble>"</comilla_doble>
  <comilla_simple>'</comilla_simple>
  <ampersand>&</ampersand>
</entidades>
```

7. Referencias a caracteres

En un documento XML se pueden escribir referencias de caracteres Unicode con los símbolos **&#**, seguidos del valor decimal o hexadecimal del carácter Unicode que se quiera representar y, finalmente, añadiendo el carácter *punto y coma* “;”.

Por ejemplo, para representar al símbolo del Euro (€), puede usarse su valor decimal (**€**) en Unicode y su valor hexadecimal (**€**).

Los espacios en blanco tienen los siguientes tratamientos:

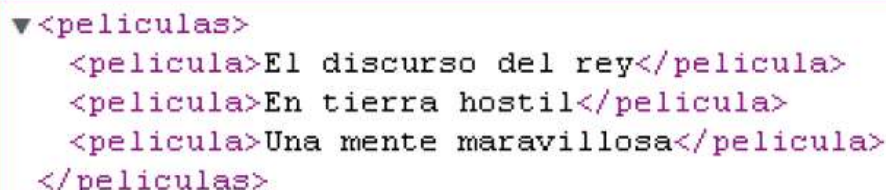
- Tabulador: **	**
- Nueva línea: **
**
- Retorno de carro: ****
- Espacio: ** **

Dentro del contenido textual de un elemento se mantendrán como están y serán tratados por el procesador. Como valor de un atributo, los espacios en blanco adyacentes se condensarán en uno solo. Los espacios en blanco entre elementos serán ignorados.

Dado el archivo XML *“películas.xml”*:

```
<?xml version="1.0" encoding="UTF-8"?>
<películas>
  <película>El discurso del rey</película>
  <película>En      tierra          hostile</película>
  <película>Una
    mente
    maravillosa</película>
</películas>
```

Se verá así en un navegador:



```
▼<películas>
  <película>El discurso del rey</película>
  <película>En tierra hostile</película>
  <película>Una mente maravillosa</película>
</películas>
```

Esto es debido a que, las tabulaciones, los retornos de carro y varios espacios en blanco contenidos en el texto de los elementos del documento, han sido representados como un único espacio en blanco.

De igual modo ocurre con los valores de los atributos. Por ejemplo, dado el archivo **"series.xml"**:

```
<?xml version="1.0" encoding="UTF-8"?>
<series>
  <serie numeros="2 4 6 8"/>
  <serie numeros="3
6
9
12 15"/>
</series>
```

```
▼<series>
  <serie numeros="2 4 6 8"/>
  <serie numeros="3 6 9 12 15"/>
</series>
```

En un navegador web se podrá visualizar:

Obsérvese que, los siguientes documentos XML contienen la misma información, pero, escrita de distinta forma:

```
<?xml version="1.0" encoding="UTF-8"?>
<datos>
  <dato>1</dato>
  <dato>2</dato>
  <dato>3</dato>
</datos>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<datos><dato>1</dato><dato>2</dato><dato>3</dato></datos>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<datos><dato>1</dato> <dato>2</dato>

<dato>3</dato></datos>
```

En todos los casos, en el navegador veremos:

```
▼<datos>
  <dato>1</dato>
  <dato>2</dato>
  <dato>3</dato>
</datos>
```

Las aplicaciones que hacen uso de documentos XML suelen hacer este tratamiento de las tabulaciones, retornos de carro y espacios en blanco.

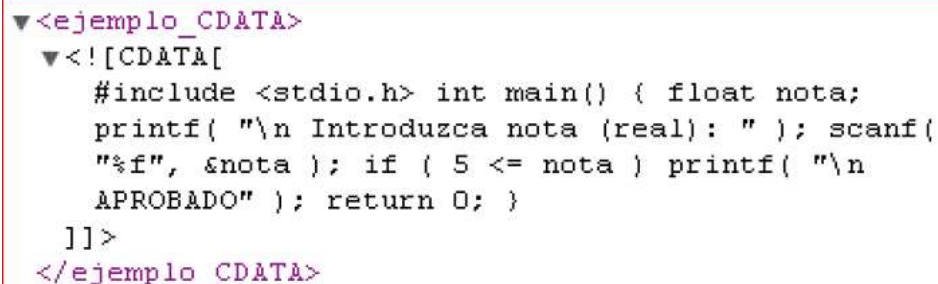
8. Secciones CDATA

Un documento XML puede contener secciones **CDATA** (*Character DATA*) para escribir texto que no se desea que sea analizado (parecidos a un comentario). Por ejemplo, esto puede ser útil cuando se quiere escribir texto que contenga alguno de los caracteres problemáticos: *menor que* “<” o *ampersand* “&”. En un documento XML, para incluir una sección CDATA, esta se escribe comenzando con la cadena de caracteres “<![CDATA[” y terminando con los caracteres “]]>”. No pueden aparecer antes del elemento raíz ni después de su cierre.

Una sección CDATA puede contener, por ejemplo, el código fuente de un programa escrito en lenguaje C:

```
<?xml version="1.0" encoding="UTF-8"?>
<ejemplo_CDATA>
<![CDATA[ #include <stdio.h>
int main() {
    float nota; printf( "\n Introduzca nota (real): " );
    scanf( "%f", &nota );
    if ( 5 <= nota )
        printf( "\n APROBADO" );
    return 0;
}
]]>
</ejemplo_CDATA>
```

En un navegador web se visualizará algo parecido a:

A screenshot of a web browser window showing the content of the CDATA section. The text is displayed in a monospaced font, with the XML tags <ejemplo_CDATA> and </ejemplo_CDATA> highlighted in purple. The code inside the CDATA section is shown as a C program snippet, including a header, a main function, and some printf/scanf statements. The browser's interface, including a scrollbar, is visible on the right side of the window.

```
▼<ejemplo_CDATA>
  ▼<![CDATA[
    #include <stdio.h> int main() { float nota;
    printf( "\n Introduzca nota (real): " ); scanf(
    "%f", &nota ); if ( 5 <= nota ) printf( "\n
    APROBADO" ); return 0; }
  ]]>
  </ejemplo_CDATA>
```

Dentro de una sección CDATA no se puede escribir la cadena “]]>”. En consecuencia, **no se pueden anidar secciones CDATA**.

Por otra parte, no está permitido escribir espacios en blanco o saltos de línea en las cadenas de inicio “<![CDATA[” o fin “]]>” de una sección CDATA.

9.- Visualización de un documento XML.

No dispone de una visualización concreta en el navegador, puesto que el documento no refleja una apariencia, sino unos datos.

Existen varias formas de representar visualmente los datos de un documento XML:

- Mediante una **hoja de estilo CSS** que indique al navegador cómo convertir cada elemento del documento XML en un elemento visual. Se lograría con la instrucción de procesamiento que se verá más adelante:

```
<?xml-stylesheet type="text/css" href="estilo-animales.css"?>
```

Esta instrucción sirve para asociar el archivo **CSS** (*Cascading Style Sheets*, Hojas de Estilo en Cascada) **"estilo-animales.css"** al documento XML. Dicho archivo podría contener, por ejemplo, el siguiente código:

```
nombre{color:blue;font-size:40px}
patas{color:red;font-size:22px}

<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="estilo-animales.css"?>
<animales>
  <animal>
    <nombre>perro</nombre>
    <patas>4</patas>
  </animal>
  <animal>
    <nombre>pato</nombre>
    <patas>2</patas>
  </animal>
  <animal>
    <nombre>ballena</nombre>
    <patas>0</patas>
  </animal>
</animales>
```

En un navegador web se verá algo parecido a:



- Mediante el uso de una **hoja de transformaciones XSLT**. La instrucción equivalente para asociar a un documento XML una hoja de transformaciones XSLT es:

```
<?xml-stylesheet type="text/xsl" href="transforma.xsl"?>
```

- También podría hacerse mediante el uso de un **lenguaje de programación**, como Java o JavaScript, que procese el documento XML.

Recordar cómo se inserta un archivo **CSS** en un documento HTML:

```
<link rel="stylesheet" type="text/css" href="/css/estilos.css" media="screen" />
```

10.- Documentos XML bien formados.

Todos los documentos XML deben verificar las reglas sintácticas que define la recomendación del W3C para el estándar XML. Esas normas básicas son:

- El documento ha de tener definido un prólogo con la declaración xml completa.
- Existe un único elemento raíz para cada documento: es un solo elemento en el que todos los demás elementos y contenidos se encuentran anidados.
- Cómo se delimitan los elementos con etiquetas.
- Qué formato puede tener una etiqueta.
- Qué nombres son aceptables para los elementos.
- Dónde se colocan los atributos.

Se dice que **un documento XML está bien formado (*well-formed document*) cuando no tiene errores de sintaxis**. Esto incluye los siguientes aspectos:

- El documento puede (el W3C lo recomienda) empezar por una instrucción de procesamiento *xml*, que indica la versión del XML y, opcionalmente, la codificación de caracteres (**encoding**) que por defecto es *UTF-8*, y si está listo para procesarse independientemente o requiere de otros archivos externos para dicha tarea (**standalone**), que por defecto vale *no*.
- Los nombres de los elementos y sus atributos deben estar escritos correctamente.
- Los valores de los atributos deben estar escritos entre comillas dobles o simples.
- Los atributos de un elemento deben separarse con espacios en blanco.
- Se tienen que utilizar referencias a entidades donde sea necesario.
- Tiene que existir un único elemento raíz.
- Los elementos son sensibles a mayúsculas y minúsculas.
- No puede haber dos atributos con el mismo nombre asociados al mismo elemento.
- Todo elemento debe tener un elemento padre, excepto el elemento raíz.
- Todos los elementos deben tener una etiqueta de apertura y otra de cierre.
- Las etiquetas deben estar correctamente anidadas.
- Las instrucciones de proceso deben estar escritas de forma correcta.
- La declaración XML debe estar en la primera línea escrita correctamente.
- Las secciones CDATA y los comentarios deben estar correctamente escritos.

Una manera de verificar que un documento XML está bien formado es abriéndolo con un navegador web. Si muestra el árbol de nodos, significa que está bien formado.

Por otro lado, se dice que **un documento XML es válido (*valid*) cuando, además de no tener errores de sintaxis, no incumple ninguna de las normas establecidas en su estructura**. Dicha estructura se puede definir utilizando distintos métodos, tales como:

- **DTD** (*Document Type Definition*, Definición de Tipo de Documento).
- **XML Schema**.
- **RELAX NG** (REgular LAnguage for XML Next Generation).

Todo documento XML válido está bien formado, pero no a la inversa.

11.- Utilización de espacios de nombres en XML.

Un **espacio de nombres XML** es una recomendación **W3C** para proporcionar elementos y atributos con nombre único en un archivo XML. Un archivo XML puede contener nombres de elementos o atributos procedentes de más de un vocabulario XML. Si a cada uno de estos vocabularios se le da un espacio de nombres, un ámbito semántico propio, referenciado a una URI donde se listen los términos que incluye, se resuelve la ambigüedad existente entre elementos o atributos que se llamen igual, la **homonimia**. Los nombres de elementos dentro de cada espacio de nombres deben ser únicos.

Un ejemplo sería una instancia XML que contuviera referencias a un cliente y a un producto solicitado por éste. Tanto el elemento que representa el cliente como el que representa el producto pueden tener un elemento hijo llamado "numero_ID". Las referencias al elemento "numero_ID" podrían ser ambiguas, salvo que los elementos, con el mismo nombre, pero significado distinto, se llevaran a espacios de nombres distintos que los diferenciaran.

```
<?xml version="1.0"?>
<cli:cliente xmlns:cli='http://es.wikipedia.org/wiki/Espacio_de_nombres_XML/cliente'
  xmlns:ped='http://es.wikipedia.org/wiki/Espacio_de_nombres_XML/pedido'>
  <cli:numero_ID>1232654</cli:numero_ID>
  <cli:nombre>Fulanito de Tal</cli:nombre>
  <cli:telefono>99999999</cli:telefono>
  <ped:pedido>
    <ped:numero_ID>6523213</ped:numero_ID>
    <ped:articulo>Caja de herramientas</ped:articulo>
    <ped:precio>187,90</ped:precio>
  </ped:pedido>
</cli:cliente>
```

11.1.- Declaración de espacios de nombres

Un espacio de nombres se declara usando el atributo XML reservado `xmlns`, cuyo valor debe ser un **identificador uniforme de recurso (URI)** - del inglés *uniform resource identifier*). Sin embargo, hay que destacar que el URI no se lee realmente como una dirección; se trata como una cadena de texto por el intérprete XML. Por ejemplo, el propio <http://www.w3.org/1999/xhtml> no contiene código alguno, simplemente describe el

espacio de nombres [XHTML](http://www.w3.org/1999/xhtml) a lectores humanos. El hecho de usar una URL (tal como "http://www.w3.org/1999/xhtml") para identificar un espacio de nombres, en lugar de una simple cadena (como "xhtml"), reduce la posibilidad de que diferentes espacios de nombres usen identificadores iguales. Los identificadores de los espacios de nombres no necesitan seguir las convenciones de las direcciones de internet, aunque a menudo lo hagan.

La declaración puede incluir también un prefijo corto con el que los elementos y atributos pueden identificarse, por ejemplo:

xmlns:xhtml="http://www.w3.org/1999/xhtml"

Un espacio de nombres XML no necesita que su vocabulario sea definido, aunque es una buena práctica utilizar un **DTD** o un **esquema XML** para definir la estructura de datos en la ubicación URI del espacio de nombres.

Otro ejemplo:


Problema de la Homonimia

Homonimia: Mismo nombre con diferentes propósitos

<pre><país nombre="Francia"> <capital>París</capital> </país></pre>	<pre><inversión> <capital>7000€</capital> </inversión></pre>
-----------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------

¿Cómo combinar en el mismo documento estos vocabularios?

```
<inversiones>
  <país nombre="Francia">
    <capital>París</capital>
    <capital>1200€</capital>
  </país>
  . . .
</inversiones>
```

Ambigüedad


Posibles soluciones

Solución fácil...

Asociar a cada etiqueta una URI

```
<[http://www.bolsa.com/]:inversiones>
  <[http://www.geog.es/]:país
    [http://www.geog.es/]:nombre="Francia">
      <[http://www.geog.es/]:capital>Paris
    </[http://www.geog.es/]:capital>
    <[http://www.bolsa.com/]:capital>1200€
  </[http://www.bolsa.com/]:capital>
</[http://www.bolsa.com/]:país>
.
.
.
</[http://www.bolsa.com/]:inversiones>
```

Legibilidad...



Solución: Asociar un alias a los elementos de un espacio de nombres dentro de un ámbito

`xmlns:alias` define *alias* en el ámbito de un elemento

```
<b:inversiones
  xmlns:b="http://www.bolsa.com/"
  xmlns:g="http://www.geog.es/"
  <g:país g:nombre="Francia">
    <g:capital>Paris</g:capital>
    <b:capital>1200€</b:capital>
  </g:país>
.
.
.
</b:inversiones>
```

NOTA: Las URIs sólo se utilizan para que el nombre sea único, no son enlaces, ni tienen que contener información

Asignación Dinámica

Es posible ir asociando espacios de nombres a los elementos según van apareciendo

```
<b:inversiones
  xmlns:b="http://www.bolsa.com/"
  <g:país
    xmlns:g="http://www.geog.es/"
    g:nombre="Francia">
      <g:capital>Paris</g:capital>
      <b:capital>1200€</b:capital>
    </g:país>
  .
  .
  .
</b:inversiones>
```

Son un mecanismo para evitar conflictos de nombres, de manera que se puedan diferenciar elementos o atributos dentro de un mismo documento XML que tengan idénticos nombres, pero diferentes definiciones.

Permiten definir la pertenencia de los elementos y los atributos de un documento XML al contexto de un vocabulario XML. De este modo se resuelven las ambigüedades que se pueden producir al juntar dos documentos distintos, de dos autores diferentes, que han utilizado el mismo nombre de etiqueta para representar cosas distintas.

Los espacios de nombres también conocidos como name spaces, permiten dar un nombre único a cada elemento, indexándolos según el nombre del vocabulario adecuado además están asociados a un **URI** (*Uniform Resource Identifier*, Identificador Uniforme de Recurso) que los identifica de forma única.

Los URI especificados en un documento XML no tienen porqué contener nada, su función es ser únicos. No obstante, en un URI se puede mostrar información si se considera oportuno.

Antes de poder utilizar un prefijo de un espacio de nombres, para resolver la ambigüedad de dos o más etiquetas, es necesario declarar el espacio de nombres, es decir, asociar un índice con el URI asignado al espacio de nombres, mediante un atributo especial xmlns. Esto se hace entre el prólogo y el ejemplar de un documento XML y su sintaxis es la siguiente:

```
<nombre_elemento xmlns:prefijo="URI_del_espacio_de_nombres">
```

Y se usan anteponiendo a elementos y atributos el prefijo asociado al espacio de nombres, además del carácter ":". En el documento, las etiquetas ambiguas se sustituyen por otras en las que el nombre del elemento está precedido de un prefijo, que determina el contexto al que pertenece la etiqueta, seguido de dos puntos, ":". Esto es:

```
<prefijo:nombre_etiqueta></prefijo:nombre_etiqueta>
```

Esta etiqueta se denomina "nombre cualificado". Al definir el prefijo hay que tener en cuenta que no se pueden utilizar espacios ni caracteres espaciales y que no puede comenzar por un dígito.

Ejemplo: Sean los documentos XML que organizan la información sobre los profesores y los alumnos del DAW respectivamente:

```
<?xml version="1.0" encoding="iso-8859-1" standalone="yes"
<!DOCTYPE alumnos>
<alumnos>
  <nombre>Fernando Fernández González</nombre>
  <nombre>Isabel González Fernández</nombre>
  <nombre>Ricardo Martínez López</nombre>
</alumnos>

<?xml version="1.0" encoding="iso-8859-1" standalone="yes" ?>
<!DOCTYPE profesores>
```

```
<profesores>
  <nombre>Pilar Ruiz Pérez</nombre>
  <nombre>Tomás Rodríguez Hernández</nombre>
</profesores>
```

Al hacer un documento sobre los miembros del curso DAW no se distinguirían los profesores de los alumnos, para resolverlo definiremos un espacio de nombres para cada contexto:

```
<?xml version="1.0" encoding="iso-8859-1" standalone="yes" ?>
<alumnos xmlns:alumnos="http://DAW/alumnos">
<profesores xmlns:profesores="http://DAW/profesores">
<asistentes>
  <alumnos:nombre>Fernando Fernández González</alumnos:nombre>
  <alumnos:nombre>Isabel González Fernández</alumnos:nombre>
  <alumnos:nombre>Ricardo Martínez López</alumnos:nombre>
  <profesores:nombre>Pilar Ruiz Pérez</profesores:nombre>
  <profesores:nombre>Tomás Rodríguez Hernández</profesores:nombre>
</asistentes>
```

Cualquier cadena de texto puede ser usada como prefijo del espacio de nombres. El URI del espacio de nombres sí debe ser único, aunque realmente no se comprueba mediante conexión alguna. El URI no es más que un nombre lógico del espacio de nombres.

El alcance de la declaración de un prefijo de espacio de nombres comprende desde la etiqueta de inicio de un elemento XML, en la que se declara, hasta la etiqueta final de dicho elemento XML. Cubre el elemento donde se ha declarado y sus elementos descendientes. En las etiquetas vacías, correspondientes a elementos sin "hijos", el alcance es la propia etiqueta. A todos estos elementos se les puede anteponer el prefijo del espacio de nombres. Se puede declarar un espacio de nombres diferente para un elemento descendiente de otro, en el cual ya se ha declarado otro espacio de nombres.

```
<?xml version="1.0" encoding="iso-8859-1" standalone="yes" ?>
<dept:departamentos xmlns:dept="empresa:espacios:dept"
  xmlns:emp="empresa:espacios:emp">
  <dept:departamento dept:deptno="10">
    <emp:empleado emp:empno="7654">
      <emp:nombre>Isabel</emp:nombre>
      <emp:apellido>González Fernández</emp:apellido>
    </emp:empleado>
    <emp:empleado emp:empno="7654">
      <emp:nombre>Pilar</emp:nombre>
      <emp:apellido>Ruiz Pérez</emp:apellido>
    </emp:empleado>
  </dept:departamento>
</dept:departamentos>
```

En un documento XML, los espacios de nombres pueden definirse en el elemento raíz –como acabamos de ver– o, directamente, en los elementos que los vayan a utilizar. Por ejemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
  <e1:ejemplo xmlns:e1="http://www.abrirllave.com/ejemplo1">
    <e1:carta>
      <e1:palo>Corazones</e1:palo>
      <e1:numero>7</e1:numero>
```

```

</e1:carta>
<e2:carta xmlns:e2="http://www.abrirllave.com/ejemplo2">
  <e2:carnes>
    <e2:filete_de_ternera precio="12.95"/>
    <e2:solomillo_a_la_pimienta precio="13.60"/>
  </e2:carnes>
  <e2:pescados>
    <e2:lenguado_al_horno precio="16.20"/>
    <e2:merluza_en_salsa_verde precio="15.85"/>
  </e2:pescados>
</e2:carta>
</e1:ejemplo>

```

11.1. - Espacio de nombres por defecto

Mediante `xmlns="..."` se define un espacio de nombres por defecto (sin alias)

```

<inversiones
  xmlns="http://www.bolsa.com/">
  <g:país
    xmlns:g="http://www.geog.es/"
    g:nombre="Francia">
    <g:capital>París</g:capital>
    <capital>1200€</capital>
  </g:país>
  . . .
</inversiones>

```

Se refiere a <http://www.bolsa.com/>

Es aquel en el que no se define un prefijo (`xmlns="URI"`). El ámbito de aplicación es el del elemento en el que se ha declarado y sus elementos descendientes, **pero no a sus atributos**.

De esta forma, tanto el elemento donde se ha definido el espacio de nombres, como todos sus sucesores (hijos, hijos de hijos, etc.), pertenecerán a dicho espacio de nombres. Por ejemplo:

```

<?xml version="1.0" encoding="UTF-8"?>
<ejemplo xmlns="http://www.avellaneda.com/ejemplo1">
  <carta>
    <palo>Corazones</palo>
    <numero>7</numero>
  </carta>
</ejemplo>

```

En el siguiente ejemplo, inicialmente se define un espacio de nombres por defecto para el elemento `<ejemplo>` y los contenidos en él. Ahora bien, posteriormente, se define un segundo espacio de nombres, que por defecto afecta al segundo elemento `<carta>` que aparece en el documento y a sus sucesores:

- <carnes>
- <pescados>

- <filete_de_ternera>
- ...

```
<?xml version="1.0" encoding="UTF-8"?>
<ejemplo xmlns="http://www.abrirllave.com/ejemplo1">
  <carta>
    <palo>Corazones</palo>
    <numero>7</numero>
  </carta>
  <carta xmlns="http://www.abrirllave.com/ejemplo2">
    <carnes>
      <filete_de_ternera precio="12.95"/>
      <solomillo_a_la_pimienta precio="13.60"/>
    </carnes>
    <pescados>
      <lenguado_al_horno precio="16.20"/>
      <merluza_en_salsa_verde precio="15.85"/>
    </pescados>
  </carta>
</ejemplo>
```

Se puede desasignar un a un elemento de un espacio de nombres por defecto con la orden:

```
<nombre_elemento xmlns="">
```

Modificando el ejemplo anterior, el elemento *<pescados>* y sus hijos, no pertenecen a ningún espacio de nombres.

```
<pescados xmlns="">
  <lenguado_al_horno precio="16.20"/>
  <merluza_en_salsa_verde precio="15.85"/>
</pescados>
```