UT1 - Introducción a los lenguajes de marcas. XML

1. Historia de los Lenguajes de Marcas

1.1. ¿Porqué surgen los lenguajes de marcas?

Los problemas relacionados con el intercambio de información entre aplicaciones y máquinas informáticas es tan viejo como la propia informática. El problema parte del hecho de haber realizado un determinado trabajo con un software en un ordenador concreto y después querer pasar dicho trabajo a otro software en ese u otro ordenador, al **no existir una estandarización en los formatos de información** usados por los distintos programas el proceso se hace muy complejo e incluso inviable.

Los **archivos binarios** tienen la complicación de que para hacer ese proceso, el origen y el destino de los datos deben comprender **cómo codificar y descodificar la información**. Eso, en muchos casos, ha sido un gran problema que ha obligado a que todos los trabajadores y trabajadoras hayan tenido que adaptarse al software de la empresa y por supuesto en toda la empresa utilizar dicho software. En la informática actual eso es aún más problema al tener una necesidad de disponibilidad global del trabajo y además la posibilidad de ver dicho trabajo en dispositivos de todo tipo como portátiles, tablet, móviles, etc.

Por ello poco a poco **han aparecido formatos binarios de archivo que han sido estándares de facto** (aunque no han sido reconocidos por ningún organismo de estándares) como por ejemplo el formato documental PDF, el formato de imagen JPEG, la música MP3 o el formato MPEG de vídeo. Pero **siguen habiendo empresas que utilizan formato propio** por la idea de que sus formatos de archivo están directamente relacionados con la calidad de su software es decir razonan que el software que fabrican es muy potente y necesitan un formato binario propio compatible con esa potencia. De ahí que **muchas veces la opción para exportar e importar datos sea utilizar conversores**, capaces de convertir los datos de un formato a otro (por ejemplo de Word a Open Office; de MP3 a MOV de Apple, etc.).

Pero **existe un formato de archivo que cualquier dispositivo es capaz de entender, el texto**, por lo que se intenta que el propio texto sirva para almacenar otros datos. Evidentemente no es posible usar texto para almacenar por ejemplo imágenes, pero sí otras cosas. Para ello **dentro del archivo habrá contenido que no se interpretará como texto** sin más que simplemente se debe mostrar, **sino que hay texto en el archivo que se marca de manera especial haciendo que signifique otra cosa**.

Los **procesadores de texto fueron el primer software en encontrarse con este dilema**. Puesto que son programas que sirven para escribir texto parecía que lo lógico era que sus datos se almacenaran como texto. Pero necesitan guardar datos referidos al formato del texto, tamaño de la página, márgenes, etc. **La solución clásica ha sido guardar la información de formato de forma binaria**, lo que provoca los ya comentados problemas.

Algunos procesadores de texto optaron por guardar toda la información como texto, haciendo que las indicaciones de formato no se almacenen de forma binaria sino textual. Dichas indicaciones son **caracteres marcados de manera especial para que así un programa adecuado pueda traducir dichos caracteres no como texto sino como operaciones** que finalmente producirán mostrar el texto del documento de forma adecuada.

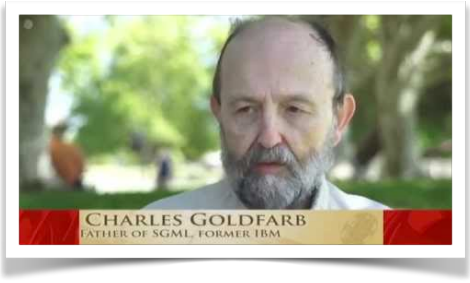
La idea del marcado procede del inglés marking up término con el que se referían a la técnica de marcar manuscritos con lápiz de color para hacer anotaciones como, por ejemplo, la tipografía a emplear en las imprentas. Este mismo término se ha utilizado para los documentos de texto que contienen comandos u anotaciones.

Las posibles anotaciones o indicaciones incluidos en los documentos de texto han dado lugar a lenguajes (entendiendo que en realidad son formatos de documento y no lenguajes en el sentido de los lenguajes de programación de aplicaciones) **llamados lenguajes de marcas, lenguajes de marcado o lenguajes de etiquetas**.

Hay un problema con el texto, puesto que al ser formato tan universal, y ser **su contenido siempre visible,** es peligroso como fuente para almacenar datos confidenciales, ya que quedaría expuesto a cualquier persona. Los datos binarios no son del todo seguros, pero como requieren del software que entienda el formato binario concreto hacen que su contenido quede menos expuesto.

1.2. Desde el GML hasta el XML

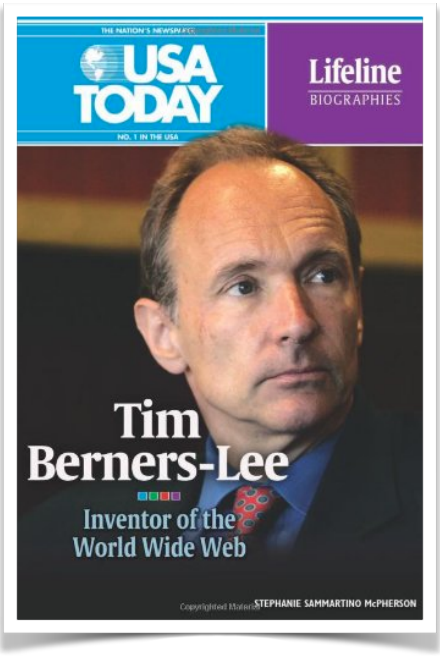
Para resolver el problema relacionados con el intercambio de información entre aplicaciones y máquinas informáticas, en los años sesenta **IBM** encargó a **Charles F. Goldfarb** la construcción de un sistema de edición, almacenamiento y búsqueda de documentos legales. Tras analizar el funcionamiento de la empresa llegaron a la conclusión de que para realizar un buen procesado informático de los documentos **había que establecer un formato estándar para todos los documentos** que se manejaban en la empresa. Con ello se lograba gestionar cualquier documento en cualquier departamento y con cualquier aplicación, sin tener en cuenta dónde ni con qué se generó el documento. Dicho formato tenía que ser válido para los distintos tipos de documentos legales que utilizaba la empresa, por tanto, debía ser flexible para que se pudiera ajustar a las distintas situaciones.



El formato de documentos que se creó como resultado de este trabajo fue **GML (Generalized Markup Language)**, cuyo objetivo era describir los documentos de tal modo que el resultado fuese independiente de la plataforma y la aplicación utilizada.

El formato GML evolucionó hasta que **en 1986 dio lugar al estándar ISO-8879** que se denominó **SGML (Standard Generalized Markup Language)**. Éste era **un lenguaje muy complejo** y requería de unas herramientas de software caras. Por ello su uso ha quedo relegado a grandes aplicaciones industriales.

En 1989/90 **Tim Berners-Lee** creó el **WWW** (World Wide Web) y se encontró con la necesidad **de organizar, enlazar y compatibilizar gran cantidad de información procedente de diversos sistemas**. Para resolverlo creó un lenguaje de descripción de documentos llamado **HTML** (HyperText Markup Language), que, en realidad, era una combinación de dos estándares ya existentes:



- **ASCII**: Código Estándar Estadounidense para el Intercambio de Información. Es el formato que cualquier procesador de textos sencillo puede reconocer y almacenar. Por tanto es un formato que permite la trasferencia de datos entre diferentes ordenadores.
- **SGML**: Lenguaje que permite dar estructura al texto, resaltando los títulos o aplicando diversos formatos al texto

HTML es una versión simplificada de SGML, ya que sólo se utilizaban las instrucciones absolutamente imprescindibles. Era tan fácil de comprender que rápidamente tuvo gran aceptación logrando lo que no pudo SGML, **HTML se convirtió en un estándar general para la creación de páginas web**. Además, tanto las herramientas de software como los navegadores que permiten visualizar páginas HTML son cada vez mejores. Ejemplo sencillo de un código HTML:

```
<html>
    <head>
        <meta charset="utf-8"/>
        <title>Esto es un ejemplo</title>
    </head>
    <body>
        <h1>Este es el título 1</h1>
        <h2>Este es el título 2</h2>
        <h3>Este es el título 3</h3>
    </body>
</html>
```

Este es el título 1

Este es el título 2

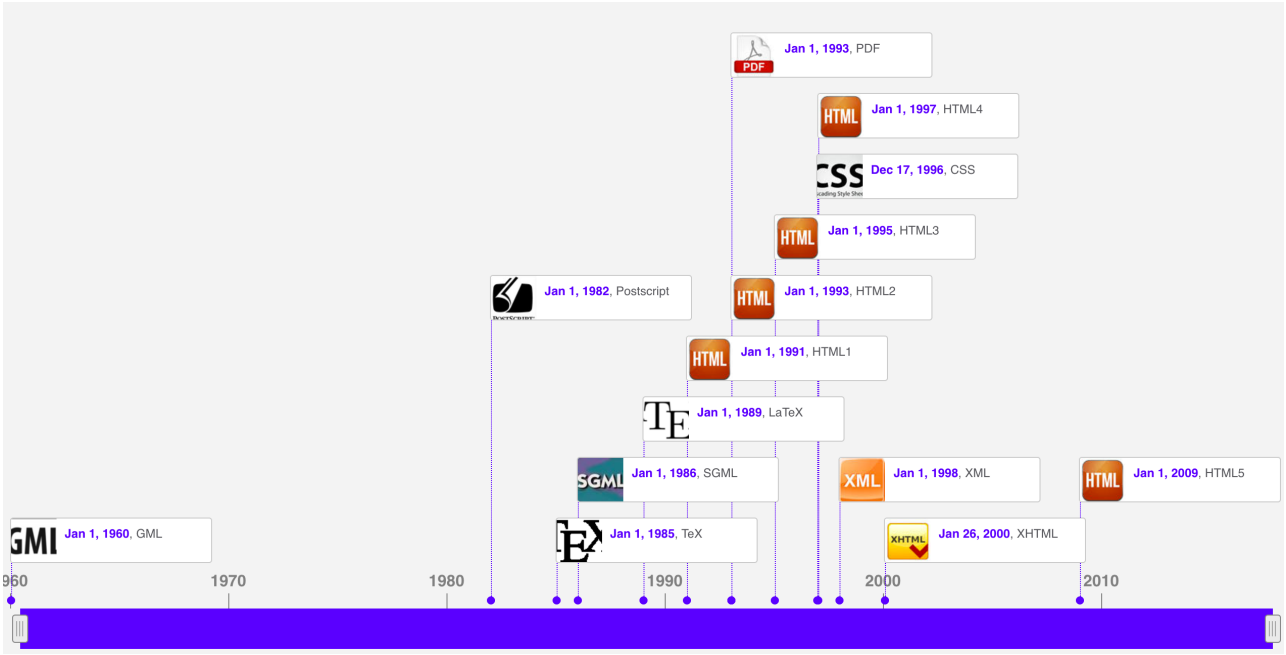
Este es el título 3

A pesar de todas estas ventajas HTML no es un lenguaje perfecto, sus **principales desventajas** son:

- No soporta tareas de impresión y diseño.
- El lenguaje **no es flexible**, ya que las etiquetas son limitadas.
- No permite mostrar contenido dinámico.
- La estructura y el diseño están mezclados en el documento.

Para resolver estos problemas de HTML el **W3C** establece, en 1998, el estándar internacional **XML(eXtensible Markup Language)**, un lenguaje de marcas puramente estructural que **no incluye ninguna información relativa al diseño**. Se convirtió con rapidez en **estándar para el intercambio de datos** en la Web. A diferencia de HTML las **etiquetas indican el significado de los datos** en lugar del formato con el que se van a visualizar los datos.

Imagen representativa de la evolución de los principales lenguajes de marcas.



2. Definición de lenguaje de marcas.

Un lenguajes de marcas es una forma de codificar un documento que, junto con el **texto** (que es la **información**), incorpora **etiquetas o marcas** que contienen información adicional acerca de la estructura del texto, su presentación o contenido.

3. Características de los lenguajes de marcas

3.1. Uso de texto plano.

Los archivos de texto plano son aquellos que están compuestos únicamente por texto sin formato, sólo caracteres. Estos caracteres se pueden codificar de distintos modos dependiendo de la lengua usada. Algunos de los sistemas de codificación de caracteres más usados son: ASCII, ISO-8859-1, Latín-1, Unicode, etc...

Los lenguajes de marcas se escriben en archivos de texto plano, y como principal ventaja es que estos archivos son abiertos directamente con editor sencillo y pueden ser interpretados directamente. Esto es una ventaja evidente respecto a los sistemas de archivos binarios, que requieren siempre de un programa intermediario para trabajar con ellos que lo interprete.

Al tratarse solamente de texto, los documentos son independientes de la plataforma, sistema operativo o programa con el que fueron creados.

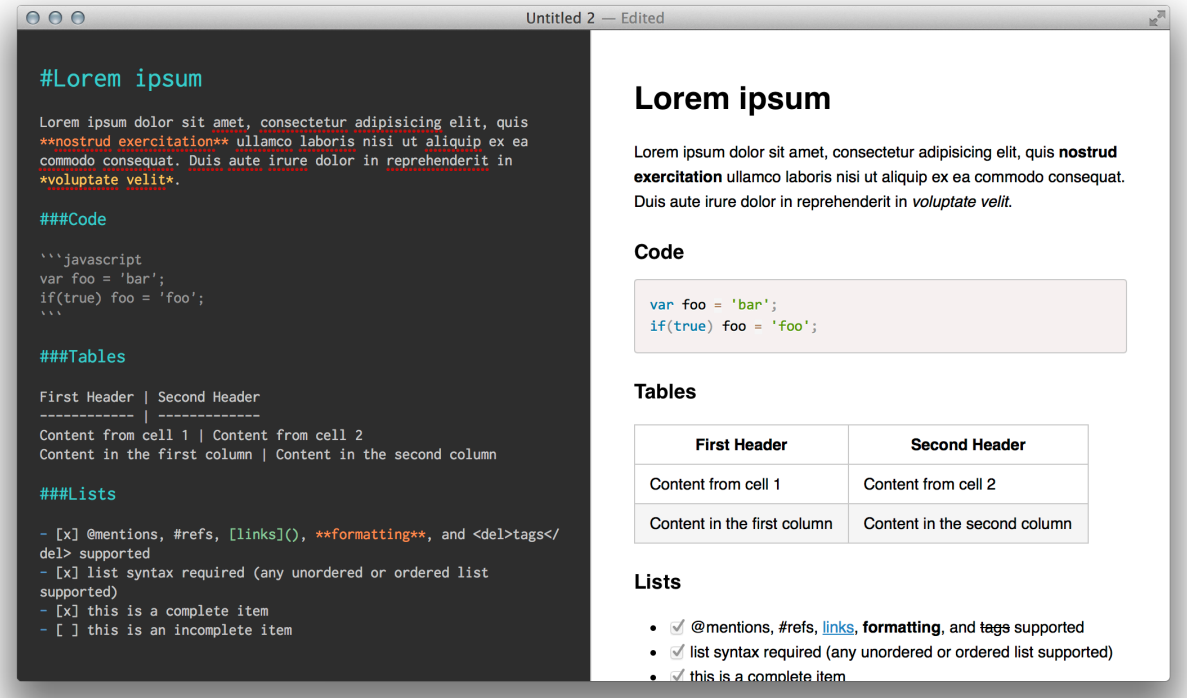
3.2. Compactos.

Las etiquetas o marcas se entremezclan con el propio contenido en un único archivo. Veamos un ejemplo en diferentes lenguajes de marcas:

Ejemplos	HTML	LaTeX	Wikitexto
Título	Título	\section{Título}	== Título ==
Lista	 - Punto 1 - Punto 2 - Punto 3 	\begin{itemize} \item Punto 1 \item Punto 2 \item Punto 3 \end{itemize}	* Punto 1 * Punto 2 * Punto 3
texto en **negrita**	texto	\bf{texto}	''' texto '''
texto en *cursiva*	texto	\it{texto}	'' texto ''

En el ejemplo vemos como diferentes lenguajes de marcas usan sus propios conjuntos de marcas. Por ejemplo si queremos poner un texto en negrita, en HTML lo ponemos entre las etiquetas y , pero el LaTex se debe poner la marca \bf y en WikiTexto encerrando el texto entre comillas.

Ejemplo usando Markdown



3.3. Flexibles

Aunque originalmente los lenguajes de marcas se idearon para documentos de texto, se han empezado a utilizar en áreas como gráficos vectoriales (**SVG**), servicios web XML, sindicación web (**RSS**) o interfaces de usuario (**Android**). Estas nuevas aplicaciones aprovechan la sencillez y potencia del lenguaje XML.

4. Clasificación de los lenguajes de marcas

4.1. Según la naturaleza de las marcas.

4.1.1. Lenguajes de presentación.

Son aquellos que definen **el formato del texto**, es decir, **la forma o apariencia que adquirirá el texto** (la presentación). En ellos al texto común se añaden palabras encerradas en símbolos especiales que contienen indicaciones de formato que permiten a los traductores de este tipo de documentos generar un documento final en el que el texto aparece con el formato indicado. Es el caso de **HTML** en el que se indica cómo debe presentarse el texto (y no por ejemplo lo que significa el mismo) también se considera así los archivos generados por los procesadores de texto tradicionales en los que al texto del documento se le acompaña de indicaciones de formato (como negrita, cursiva, etc.), por ejemplo **RTF**.

4.1.2. Lenguajes de procedimiento.

Se trata de documentos en los que hay texto marcado especialmente que en realidad **se interpreta como órdenes a seguir y así el archivo en realidad contiene instrucciones a realizar con el texto**. Es el caso de **LaTeX, PostScript** o **Markdown** donde por ejemplo se puede indicar una fórmula matemática. Es el caso LaTeX o PostScript donde por ejemplo se puede indicar una fórmula matemática.

4.1.3. Lenguajes semánticos.

En ellos las marcas especiales permiten **dar significado al texto pero no indican cómo se debe presentar en pantalla el mismo**. Sería el caso de **XML** (o de SGML) y **JSON** en el que la presentación nunca se indica en el documento; simplemente se indica una semántica de contenido que lo hace ideal para almacenar datos (por ejemplo si el texto es un nombre de persona o un número de identificación fiscal).

```
<persona>
        <nombre>Javier</nombre>
        <apellido>González</apellido>
        <apellido>Pérez</apellido>
        <edad>23</edad>
</persona>
```

El marcado descriptivo está evolucionando hacia el marcado genérico. Los nuevos sistemas de marcado descriptivo estructuran los documentos en árbol, con la posibilidad de añadir referencias cruzadas. Esto permite tratarlos como bases de datos, en las que el propio almacenamiento tiene en cuenta la estructura.

A continuación se muestra un ejemplo de la filosofía de XML, en el que podemos ver como se estructura una persona, pero no tenemos información de cómo mostrar sus datos en la pantalla.

4.2. Según el uso que se le da al lenguaje

4.2.1. Elaboración de documentos electrónicos.

- **RTF** (Rich Text Format): Formato de Texto Enriquecido, fue desarrollado por Microsoft en 1987. Permite el intercambio de documentos de texto ente distintos procesadores de texto.

- **TeX**: Su objetivo es la creación de ecuaciones matemáticas complejas.
- **Wikitexto**: Permite la creación de páginas wiki en servidores preparados para soportar este lenguaje.
- **DocBook:** Permite generar documentos separando la estructura lógica del documento de su formato. De este modo, dichos documentos, pueden publicarse en diferentes formatos sin necesidad de realizar modificaciones en el documento original.

4.2.2. Uso de Internet (páginas web, canales de noticias, etc).

- **HTML, XHTML**: (Hypertext Markup Language, eXtensible Hypertext Markup Language): Su objetivo es la creación de páginas web.
- **RSS:** Permite la difusión de contenidos web. Sindicación de contenidos.

4.2.3. Especializados en ámbitos.

- **MathML** (Mathematical Markup Language), su objetivo es expresar el formalismo matemático de tal modo que pueda ser entendido por distintos sistemas y aplicaciones.
- **VoiceXML** (Voice Extended Markup Language) tiene como objetivo el intercambio de información entre un usuario y una aplicación con capacidad de reconocimiento de habla.
- **MusicXML**: Permite el intercambio de partituras entre distintos editores de partituras.
- **SGV** (Scalable Vector Graphics), es un formato de gráficos vectoriales bidimensionales, tanto estáticos como animados. Se convirtió en una recomendación del W3C en Septiembre del 2001, por lo que en la actualidad los principales navegadores (Google Chrome, Safari, Mozilla Firefox, Opera, Internet Explorer a partir de la versión 9) son capaces de mostrar imágenes en formato SGV sin necesidad de complementos externos.

5. Introducción a XML

XML es un lenguaje de marcas que **ofrece un formato para la descripción de datos estructurados**, el cual conserva todas las propiedades importantes de SGML. Es decir, XML **es un metalenguaje**, dado que con el podemos **definir nuestro propio lenguaje**, que se centra en la información en sí misma.

La particularidad más importante del XML es que **no posee etiquetas prefijadas** con anterioridad, ya que es el propio diseñador el que las crea, dependiendo del contenido del documento.

El XML o Lenguaje de Etiquetas Extendido, es un **metalenguaje** caracterizado por:

- **Permitir definir etiquetas propias**, creadas por el programador.
- Estas etiquetas estructuran y guardan de forma ordenada la información.
- 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.

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.

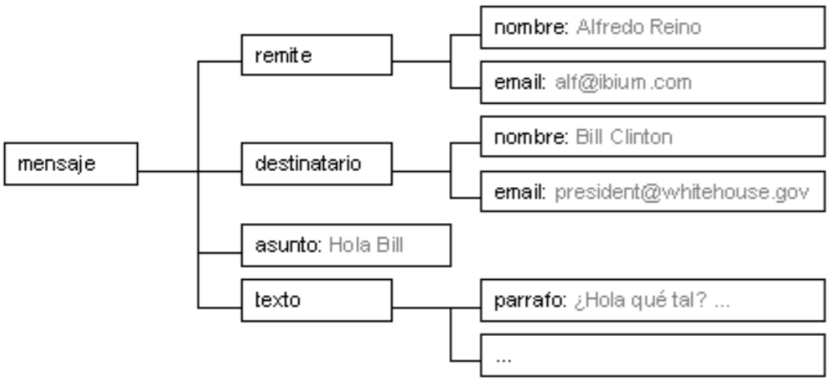
Al igual que en HTML un documento XML es un **documento de texto**, en este caso con extensión **".xml"**, compuesto de parejas de etiquetas, estructuradas en árbol, que describen una función en la organización del documento, que puede editarse con cualquier editor de texto y que es interpretado por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

Las **características básicas** de XML son:

- Los documentos XML ***son fáciles de crear***.
- Dado que XML se concibió para trabajar en la Web, es directamente ***compatible con protocolos*** que ya funcionan, como HTTP y los URL.
- El marcado de XML ***es legible para los humanos y por las máquinas***.
- XML ***es extensible, adaptable y aplicable a una gran variedad de situaciones***.
- Todo documento que verifique las reglas de XML está conforme con SGML.
- No se requieren conocimientos de programación para realizar tareas sencillas en XML.
- La difusión de los documentos XML está asegurada ya que cualquier procesador de XML puede leer un documento de XML.
- El diseño XML es formal y conciso.
- XML es orientado a objetos.
- Todo documento XML se compone exclusivamente de datos de marcado y datos carácter entremezclados.

Vamos a tener el primer contacto con un documento XML y su estructura de forma gráfica para su mayor comprensión.

```
<?xml version="1.0"?>
<!DOCTYPE MENSAJE SYSTEM "mensaje.dtd">
<mensaje>
 <remite>
  <nombre>Alfredo Reino</nombre>
  <email>alf@ibium.com</email>
 </remite>
 <destinatario>
  <nombre>Bill Clinton</nombre>
  <email>president@whitehouse.gov</email>
 </destinatario>
 <asunto>Hola Bill</asunto>
 <texto>
  <parrafo>¿Hola qué tal? Hace <enfasis>mucho</enfasis>
  no escribes. A ver si llamas y quedamos para tomar al
 </texto>
</mensaje>
```



Es **muy importante** utilizar nombre de etiquetas que **describan la información que encierran**. Si nos fijamos en el ejemplo anterior, los nombre de etiquetas que utilizan el diseñador del XML, describen perfectamente lo que almacenan, por ejemplo la etiqueta <remite>, contiene datos del usuario que envía el mensaje.

```
<?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>
```

6. Ejemplos de aplicación

En XML se denomina **aplicación XML** a cualquier lenguaje de marcado basado en XML. O sea en XML, **aplicación** significa el uso de XML para un dominio específico, como los que veremos a continuación. En XML se habla de **aplicación XML** o **Lenguaje de Marcado basado en XML** o **Vocabulario**.

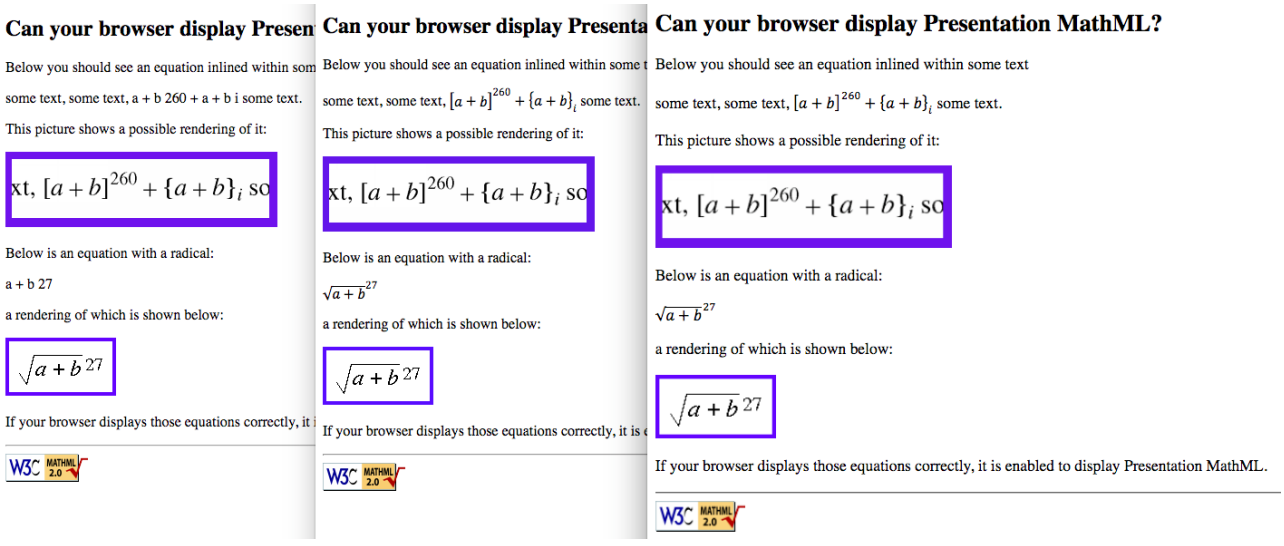
MathML - Mathematical Markup Language

Se utiliza para representar fórmulas y ecuaciones matemáticas de tal forma que se puedan entender tanto por los humanos como por las máquinas. Se puede usar en combinación con XHTML, para poder mostrar estas formulas matemáticas en la web. También se emplea para el intercambio de información entre programas específicos de matemáticas como Maple o Matlab.

```
<!DOCTYPE html PUBLIC
        "-//W3C//DTD XHTML 1.1 plus MathML 2.0//EN"
        "http://www.w3.org/TR/MathML2/dtd/xhtml-math11-f.dtd">

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

  <head>
    <title>MathML's Hello Square</title>
    <meta http-equiv ="content-type" content="mathml" />
  </head>

  <body>

    <p> This is a perfect square:</p>

    <math xmlns="http://www.w3.org/1998/Math/MathML">
     <mrow>
       <msup>
         <mfenced>
           <mrow>
             <mi>a</mi>
             <mo>+</mo>
             <mi>b</mi>
           </mrow>
         </mfenced>
         <mn>2</mn>
       </msup>
     </mrow>
    </math>

  </body>
</html>
```

This is a perfect square:

$$(a+b)^2$$



Pregunta: ¿Qué navegador no soporta MathML?

Más información en: https://www.w3.org/Math/

KML - Keyhole Markup Language

KML es un lenguaje de marcado, creado por Google, que se utiliza para mostrar datos geográficos en un navegador terrestre, como Google Earth, Google Maps y Google Maps para móviles. KML utiliza una estructura basada en etiquetas con atributos y elementos anidados y está basado en el estándar XML.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
    <Placemark>
        <name>New York City</name>
        <description>New York City</description>
        <Point>
            <coordinates>-74.006393,40.714172,0</coordinates>
        </Point>
    </Placemark>
</kml>
```

Más información en: https://developers.google.com/kml/

OpenDocument

Es un formato de archivo abierto y estándar para el almacenamiento de documentos ofimáticos como pueden ser documentos de texto, hojas de calculo, etc. Si creamos un documento de texto con OpenOffice y lo almacenamos con extensión **.odt**, a continuación cambiamos su extensión a **.zip** y lo abrimos con una herramienta de descompresión, podemos observar como está formado por varios ficheros **.xml**. Entre ellos veremos a **content.xml**, que contiene la información que hemos almacenado, entre otras cosas.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<office:document-content office:version="1.2" xmlns:chart="urn:oasis:names:tc:opendocument:xmlns:chart:1.0" xmlns:dc="http://purl.org/dc/elements/1.1/
" xmlns:dom="http://www.w3.org/2001/xml-events" xmlns:dr3d="urn:oasis:names:tc:opendocument:xmlns:dr3d:1.0" xmlns:draw="
urn:oasis:names:tc:opendocument:xmlns:drawing:1.0" xmlns:field="urn:openoffice:names:experimental:ooo-ms-interop:xmlns:field:1.0" xmlns:fo="
urn:oasis:names:tc:opendocument:xmlns:xsl-fo-compatible:1.0" xmlns:form="urn:oasis:names:tc:opendocument:xmlns:form:1.0" xmlns:grddl="http://www.w3.
org/2003/g/data-view#" xmlns:math="http://www.w3.org/1998/Math/MathML" xmlns:meta="urn:oasis:names:tc:opendocument:xmlns:meta:1.0" xmlns:number="
urn:oasis:names:tc:opendocument:xmlns:datastyle:1.0" xmlns:of="urn:oasis:names:tc:opendocument:xmlns:of:1.2" xmlns:office="
urn:oasis:names:tc:opendocument:xmlns:office:1.0" xmlns:ooo="http://openoffice.org/2004/office" xmlns:oooc="http://openoffice.org/2004/calc" xmlns:
ooow="http://openoffice.org/2004/writer" xmlns:rpt="http://openoffice.org/2005/report" xmlns:script="urn:oasis:names:tc:opendocument:xmlns:script:1.0"
 xmlns:style="urn:oasis:names:tc:opendocument:xmlns:style:1.0" xmlns:svg="urn:oasis:names:tc:opendocument:xmlns:svg-compatible:1.0" xmlns:table="
urn:oasis:names:tc:opendocument:xmlns:table:1.0" xmlns:tableooo="http://openoffice.org/2009/table" xmlns:text="
urn:oasis:names:tc:opendocument:xmlns:text:1.0" xmlns:textooo="http://openoffice.org/2013/office" xmlns:xforms="http://www.w3.org/2002/xforms" xmlns:
xhtml="http://www.w3.org/1999/xhtml" xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.
org/2001/XMLSchema-instance">
    <office:scripts/>
    <office:font-face-decls>
        <style:font-face style:font-family-generic="swiss" style:name="Lucida Sans1" svg:font-family="'Lucida Sans'"/>
        <style:font-face style:font-family-generic="roman" style:font-pitch="variable" style:name="Times New Roman" svg:font-family="'Times New Roman'
        "/>
        <style:font-face style:font-family-generic="swiss" style:font-pitch="variable" style:name="Arial" svg:font-family="Arial"/>
        <style:font-face style:font-family-generic="system" style:font-pitch="variable" style:name="Lucida Sans" svg:font-family="'Lucida Sans'"/>
        <style:font-face style:font-family-generic="system" style:font-pitch="variable" style:name="SimSun" svg:font-family="SimSun"/>
    </office:font-face-decls>
    <office:automatic-styles/>
    <office:body>
        <office:text>
            <text:sequence-decls>
                <text:sequence-decl text:display-outline-level="0" text:name="Illustration"/>
                <text:sequence-decl text:display-outline-level="0" text:name="Table"/>
                <text:sequence-decl text:display-outline-level="0" text:name="Text"/>
                <text:sequence-decl text:display-outline-level="0" text:name="Drawing"/>
            </text:sequence-decls>
            <text:p text:style-name="Standard">Hola mundo XML</text:p>
        </office:text>
    </office:body>
</office:document-content>
```

Office Open XML

Formato de almacenamiento de documentos que compite directamente con Opendocument, fue desarrollado originalmente por **Microsoft**, este lo entrego a **ECMA** para su estandarización. Posteriormente **ISO** publico la estandarización internacional.

Si creamos un documento en word y lo almacenamos con la extensión **.docx**, y repetimos el proceso realizado con el .odt, nos encontraremos con varios ficheros .xml si abrimos el **document.xml** veremos el contenido de nuestro fichero.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<w:document mc:Ignorable="w14 wp14" xmlns:m="http://schemas.openxmlformats.org/officeDocument/2006/math" xmlns:mc="http://schemas.
openxmlformats.org/markup-compatibility/2006" xmlns:mo="http://schemas.microsoft.com/office/mac/office/2008/main" xmlns:mv="urn:schemas-
microsoft-com:mac:vml" xmlns:o="urn:schemas-microsoft-com:office:office" xmlns:r="http://schemas.openxmlformats.
org/officeDocument/2006/relationships" xmlns:v="urn:schemas-microsoft-com:vml" xmlns:w="http://schemas.openxmlformats.
org/wordprocessingml/2006/main" xmlns:w10="urn:schemas-microsoft-com:office:word" xmlns:w14="http://schemas.microsoft.
com/office/word/2010/wordml" xmlns:wne="http://schemas.microsoft.com/office/word/2006/wordml" xmlns:wp="http://schemas.openxmlformats.
org/drawingml/2006/wordprocessingDrawing" xmlns:wp14="http://schemas.microsoft.com/office/word/2010/wordprocessingDrawing" xmlns:wpc="
http://schemas.microsoft.com/office/word/2010/wordprocessingCanvas" xmlns:wpg="http://schemas.microsoft.
com/office/word/2010/wordprocessingGroup" xmlns:wpi="http://schemas.microsoft.com/office/word/2010/wordprocessingInk" xmlns:wps="
http://schemas.microsoft.com/office/word/2010/wordprocessingShape">
    <w:body>
        <w:p w:rsidR="00740CBA" w:rsidRDefault="00FD2AAC">
            <w:r>
                <w:t>Hola mundo XML</w:t>
            </w:r>
            <w:bookmarkStart w:id="0" w:name="_GoBack"/>
            <w:bookmarkEnd w:id="0"/>
        </w:p>
        <w:sectPr w:rsidR="00740CBA" w:rsidRDefault="00740CBA">
            <w:pgSz w:h="16840" w:w="11900"/>
            <w:pgMar w:bottom="1417" w:footer="708" w:gutter="0" w:header="708" w:left="1701" w:right="1701" w:top="1417"/>
            <w:cols w:space="708"/>
            <w:docGrid w:linePitch="360"/>
        </w:sectPr>
    </w:body>
</w:document>
```

SVG - Scalable Vector Graphics

Formato de **gráficos vectoriales** desarrollado por la W3C. La podemos usar para representar gráficos vectoriales en la Web. Como por ejemplo el siguiente código.

```html
<!DOCTYPE html>
<html>
    <body>
        <h1>My first SVG</h1>
        <svg height="100" width="100">
            <circle cx="50" cy="50" fill="yellow" r="40" stroke="green" stroke-width="4"/>
            Sorry, your browser does not support inline SVG.
        </svg>

        <img src="imagenes-svg.svg" alt="Prueba de imagen SVG">
    </body>
</html>
```

Más información en: https://www.w3.org/TR/SVG/

7. Estructura de un documento XML

Un documento XML está formado por una mezcla de datos e información de etiquetado sobre los mismos. La información de etiquetado se corresponde con el texto que aparece encerrado entre los caracteres '<' y '>'. Es importante resaltar que XML **distingue entre mayúsculas y minúsculas**, tanto en los datos como en el etiquetado. Por ejemplo, las etiquetas <dni> y <DNI> son completamente diferentes.

Un documento XML está formado por el **prólogo** y por el **cuerpo** así como texto de etiquetas que contiene una gran variedad de efectos positivos o negativos en la referencia opcional a la que se refiere el documento, hay que tener mucho cuidado de esa parte de la gramática léxica para que se componga de manera uniforme.

Prólogo

Aunque **no es obligatorio**, es muy importante ponerlo y debe preceder al ejemplar ó cuerpo del documento. Su inclusión **facilita el procesado** de la información del ejemplar. El prólogo de un documento XML contiene:

I. **Una declaración XML.**
II. **Una declaración de tipo de documento.**

I. **La declaración XML:** En el caso de incluirse, **ha de ser la primera línea** del documento, 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. La declaración tiene la función de indicar la versión de XML, la codificación empleada para representar los caracteres y la autonomía del documento.

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 Shitf_JIS	Japonés

Versión de XML básica	**<?xml version= "1.0" ?>**
Versión de XML y codificación empleada	**<?xml version= "1.0" encoding="UTF-8" ?>**

II. **La declaración del tipo de documento (DTD)**, nos sirve para validar los documentos XML. También es opcional, pero veremos que los necesitamos para garantizar una determinada estructura para todos nuestros documentos XML. **Enlaza el documento XML con su DTD** (definición de tipo de documento). El DTD también puede estar incluido en la propia declaración o ambas cosas al mismo tiempo. Toda declaración de tipo de documento comienza por la cadena:

<!DOCTYPE Nombre_tipo ...>	**<!DOCTYPE html>**

En el siguiente enlace podéis encontrar los DOCTYPE que podemos encontrar en distintos documentos web, HTML, XHTML, MathML, SVG, etc.

https://www.w3.org/QA/2002/04/valid-dtd-list.html

Cuerpo. El ejemplar, los elementos y sus atributos.

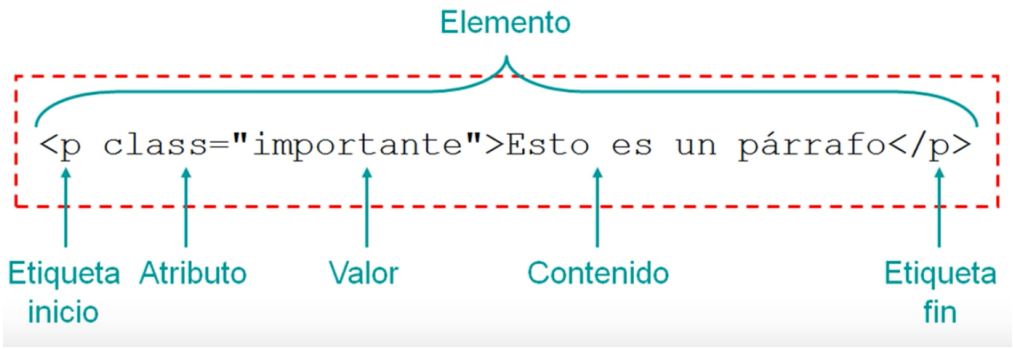
A diferencia del prólogo, el **cuerpo no es opcional** en un documento XML. Es la **parte más importante de un documento XML**, ya que contiene los datos reales del documento. Está formado por una jerarquía de elementos a partir de un único elemento raíz.

Los **elementos** son los distintos bloques de información que permiten definir la estructura de un documento XML. Están delimitados por una etiqueta de apertura, un contenido y una etiqueta de cierre. A su vez los elementos pueden estar formados por otros elementos y/o por atributos.

```
<?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 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**, 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. Debe tener una **estructura estrictamente jerárquica**.
- Los **nombres de los elementos** se forma siguiendo la siguiente sintaxis:
  - Primer carácter puede ser: [A-Z] | "_" | [a-z]
  - El resto de caracteres pueden ser los mismos que los anteriores más: "-" | "." | [0-9]
- El XML es **sensible a mayúsculas y minúsculas**. 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 **>**, **<**, **&**, **"** y apóstrofe **'**. 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 **&#x20AC**;

Etiquetas

Los lenguajes de marcas utilizan una serie de etiquetas especiales intercaladas en un documento de texto sin formato. Dichas etiquetas serán posteriormente interpretadas por los intérpretes del lenguaje y ayudan al procesado del documento.

Las **etiquetas se escriben** encerradas entre ángulos, es decir **< y >**. Normalmente, se utilizan **dos etiquetas: una de inicio y otra de fin** para indicar que ha terminado el efecto que queríamos presentar. La única diferencia entre ambas es que la de cierre lleva una barra inclinada "/" antes del código.

Atributos

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.

```
<?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> </tip
        <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>
```

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 la de los elementos**, y no pueden contener el carácter menor que "<".

Comentarios

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

- Antes del prólogo.
- Dentro de la apertura o cierre de una etiqueta.

Secciones CDATA

```
<![CDATA [
    …
]]>
```

Las secciones CDATA (Character DATA) permiten que el analizador ignore ciertas secciones del documento. Dentro de la secciones CDATA se pueden utilizar todos los caracteres problemáticos, como "**<**" o "**&**", sin el riesgo que el analizador muestre errores de sintaxis. Su sintaxis es:

Un ejemplo de uso CDATA podría ser como el siguiente en el que se muestra 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>
```

NOTA IMPORTANTE - En los nombres de etiquetas y atributos vamos a **evitar** la utilización de **mayúsculas**, **acentos**, la **ñ** y demás caracteres propios del idioma español y no tienen su correspondiente en inglés. Aunque lo permite utilizar, definiendo correctamente el encoding utilizado, puede ser una **fuente de problemas** que nos puede hacer perder mucho tiempo en encontrar porque algo está fallando.

8. Documentos XML bien formados y válidos

Los documentos XML pueden ser correctos a 2 niveles **bien formados** y **válidos**.

Documentos XML bien formados

Los documentos denominados como «bien formados» (del inglés well formed) son aquellos que cumplen con todas las definiciones básicas de formato y pueden, por lo tanto, analizarse correctamente por cualquier analizador sintáctico (parser) que cumpla con la norma.

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.
- Hay que **cumplir las reglas sintácticas** del lenguaje XML para definir los distintos elementos y atributos del documento, como se comentó en la sección anterior.

Documentos XML válido

Para que un documento sea válido lo primero es que debe de estar bien formado y a continuación debe de **cumplir la definición concreta del lenguaje** que se está empleando, como puede ser que se use los nombres de etiquetas correctos. Por ejemplo cuando usamos las etiquetas de XHTML <body>, <p>, <ul>, etc. Para especificar un lenguaje se utilizan DTD o XML Schema.

Un documento XML bien formado puede ser válido. Para ser válido, un documento XML debe:

- Incluir una **referencia a una gramática**,
- Incluir **únicamente elementos y atributos** definidos en la gramática,
- **Cumplir las reglas gramaticales definidas** en la gramática.

Existen varios formas de **definir una gramática** para documentos XML, las más empleadas son :

- **DTD** (Document Type Definition = Definición de Tipo de Documento). Es el modelo más antiguo, heredado del SGML.
- **XML Schema**. Es un modelo creado por el W3C como sucesor de las DTDs.
- Relax NG. Es un modelo creado por OASIS, más sencillo que XML Schema.

9. Actividades. Documentos bien formados.

AC1.1 - Los siguientes documentos no están bien formados porque cada uno contiene dos errores (dos errores del mismo tipo cuentan cómo uno sólo). Corrija los errores y compruebe con un validador de XML que son documentos bien formados (si para corregir algún error hay que inventarse una etiqueta o atributo, utilice un nombre que tenga relación con la información contenida en el documento). Además debes de tabular bien el código XML resultante.

```
<?xml version="1.0" encoding="UTF-8"?>
<deportistas>
  <deportista>
    <deporte Atletismo />
    <nombre>Jesse Owens</nombre>
  <deportista>
    <deporte Natación />
    <nombre>Mark Spitz</nombre>
  </deportista>
</deportistas>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<pelicula>
  <titulo>Con faldas y a lo loco</titulo>
  <director>Billy Wilder</director>
</pelicula>
<pelicula>
  <director>Leo McCarey</director>
  <titulo>Sopa de ganso</titulo>
</pelicula>
<autor />barto</autor>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<texto>
  <Titulo>XML explicado a los niños</titulo>
  <párrafo>El <abreviatura>XML</abreviatura>define cómo crear
  lenguajes de marcas.</párrafo>
  <párrafo>Las marcas se añaden a un documento de texto
  para añadir información.</párrafo>
  <http://>www.example.org</http://>
</texto>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<mundiales-de-futbol>
  <mundial>
    <pais="España" />
    <1982 />
  </mundial>
</mundiales-de-futbol>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<geografia mundial>
  <pais>
    <pais>España</pais>
    <continente>Europa</continente>
    <capital></capital nombre="Madrid">
  </pais>
</geografia mundial>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<mediosDeTransporte>
  <bicicleta velocidad="v<100km/h" />
  <patinete velocidad maxima="50 km/h"
</mediosDeTransporte>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<programas>
  <programa nombre="Firefox" licencia="GPL" licencia="MPL" />
  <programa nombre="LibreOffice" licencia="LGPL" />
  <programa nombre="Inkscape" licencia=GPL />
</programas>
```

10.

11. A tener en cuenta cuando diseñamos un XML

Al diseñar un documento XML nos surge la duda de cuándo usar elementos y cuándo atributos. Aunque no se puede dar un criterio válido para todos los casos si podemos dar algunas pautas que ayuden a tomar una decisión:

1. Si la información tiene una **estructura interna** debe se un **elemento**.
2. Si contiene una **gran cantidad de información** es más adecuado un **elemento**.
3. Podemos hacer el símil de que un **elemento** es un **sustantivo** y un **atributo** es un **adjetivo**.
4. Los mecanismos de procesamiento y presentación de documentos, permiten tener un mejor control sobre los elementos. Por tanto aquella información que tenga un procesamiento o presentación complejos debe de ser un elemento.
5. Como consejo final, **en caso de duda**, se suele utilizar un **elemento**.

Video explicando las pautas básicas que debemos de seguir para el diseño de un XML.

6. Actividades de Diseño

AC1.2 - Crea un documentos XML que almacene la siguiente información:

CIUDADES		
Nombre	***País***	***Continente***
Nueva Delhi	India	Ásia
Lisboa	Portugal	Europa
El Cairo	Egipto	África

Nota: el ***continente*** al que pertenecen un ***país*** hay que representarlo mediante un atributo, el resto de información no.

AC1.3 - Crea un documentos XML que almacene la siguiente información:

HECHOS HISTÓRICOS			
Descripción de cada hecho	***Fecha***		
	Día	***Mes***	***Año***
IBM da a conocer el PC.	12	8	1981
Se funda Google.	4	9	1998
Se funda Facebook.	4	2	2004

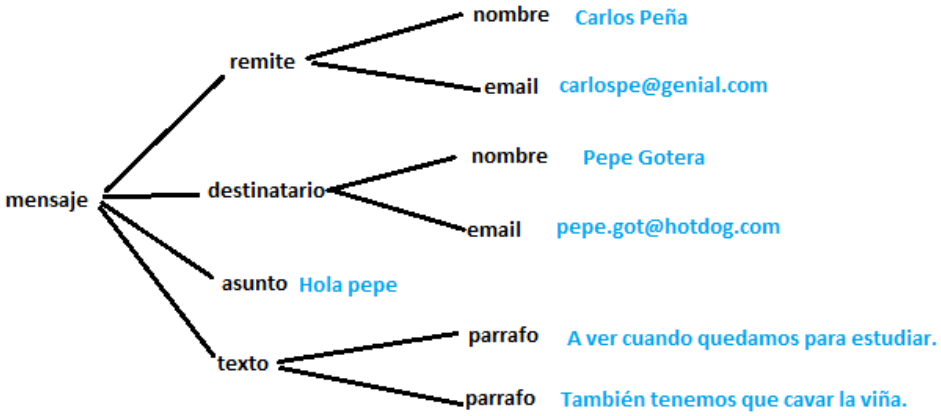
Nota: la ***descripción*** de cada ***hecho*** hay que representarla mediante un atributo, el resto de información no.

AC1.4 - Sin utilizar atributos, crear un documento XML bien formado que describa una lista de marcadores de páginas web, sabiendo que se desea que la información de cada página sea el nombre, una descripción breve y su URL. Los datos de los marcadores son los descritos en la siguiente tabla:

Wikipedia La enciclopedia libre http://www.wikipedia.org
W3C World Wide Web Consortium http://www.w3.org
w3school Tutoriales Web Online https://www.w3schools.com/

AC1.5 - Escribir un documento XML bien formado que guarde información de dos equipos de fútbol (nombre, ciudad y entrenador) con dos jugadores (nombre, posición y nacionalidad) cada uno. La posición (portero, defensa, medio, delantero) deberá representarse mediante un atributo del jugador. Utilizar datos reales para los equipos y jugadores. No obstante, no deberá indicarse el nombre del entrenador.

AC1.6 - Crear un documento xml a partir del siguiente árbol.



AC1.7 - Representa un diagrama de árbol y un documento XML para describir la siguiente información relativa a la carta de una cafetería.

AC1.8 - **Coches**. Imaginemos un concesionario de coches que periódicamente deben enviar información sobre los vehículos que tiene en oferta a un portal publicitario de compra - venta de coches. La información que debe enviar es, la marca, el modelo, motor, matricula, kilómetros, precio original, precio oferta, extras, fotos. Crea un documento XML con la información pedida.

AC1.9 - Ejemplo de biblioteca

Diseñar un documento Xml bien formado que permita estructurar la información de una biblioteca, almacenando los datos de los distintos ejemplares como son, el tipo de ejemplar (libro, revista, etc.), datos del ejemplar (isbn, edición, páginas, autores) la información sobre los prestamos realizados (nombre del lector, fecha de préstamo y de devolución).

AC1.10 - Alumnos del ciclo formativo

Diseñar un documento Xml bien formado que permita estructurar la información de los ciclos formativos que se imparten en un centro de formación profesional. Se debe almacenar el nombre de los ciclos, los módulos que lo componen con nombre, horas semanales, si es obligatoria o opcional, fecha de inicio y fin del modulo, y el listado de alumnos matriculados en cada modulo. De los alumnos necesito almacenar su nombre, apellidos, nif, teléfono, email, dirección (calle, numero, ciudad y código postal), las faltas de asistencia y la nota que será de apto y no apto.

Validadores XML Online:

- Validador w3schools.com: https://www.w3schools.com/xml/xml_validator.asp
- Otro validador: https://www.xmlvalidation.com

7. Enlaces de interés

- https://es.wikipedia.org/wiki/Extensible_Markup_Language
- https://www.timetoast.com/timelines/evolucion-de-los-lenguajes-de-marcas-7241eb2c-5e19-46b0-8d84-333745d0f729
- http://www.xmlvalidation.com/
- http://www.mclibre.org/consultar/xml/index.html
- http://www.abrirllave.com/xml/