

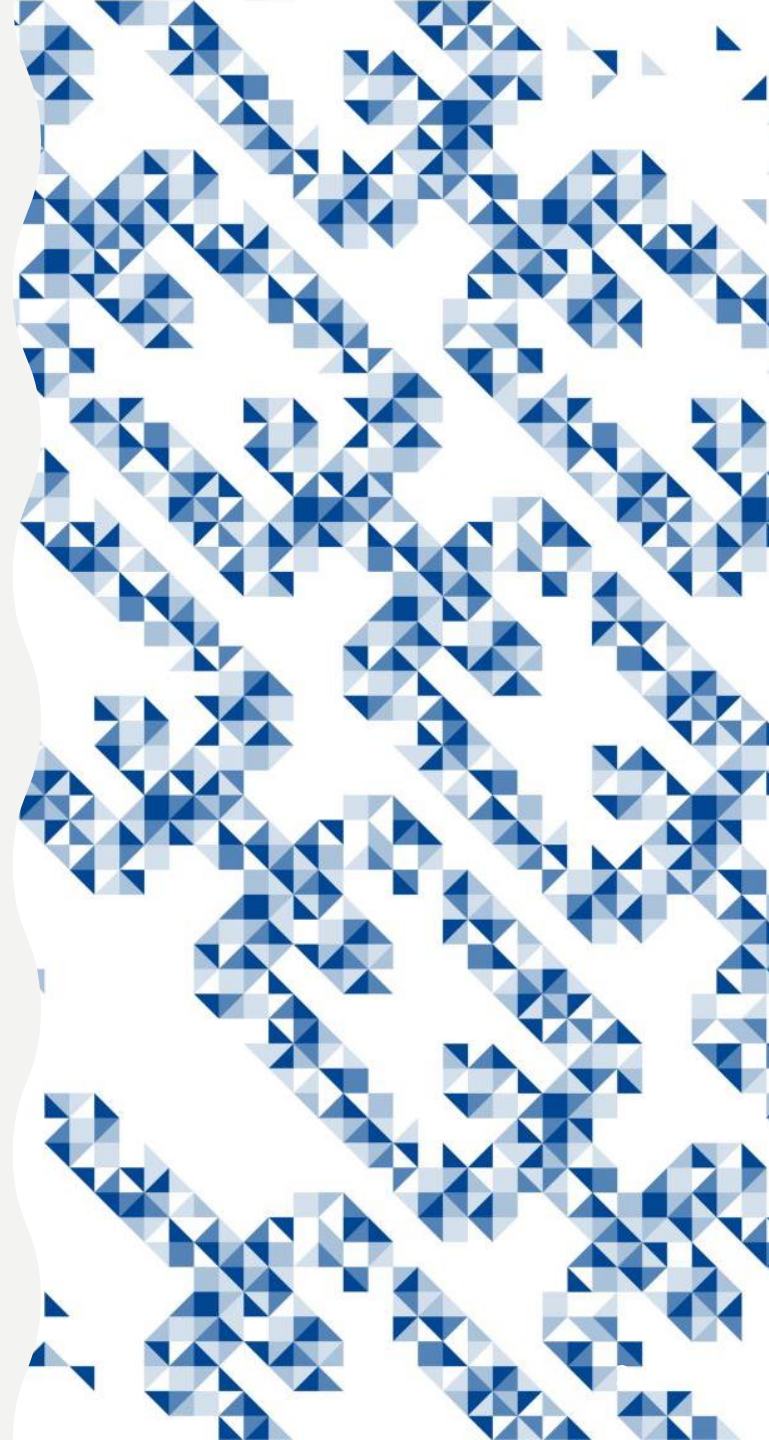
LENGUAJES DE MARCAS Y SISTEMAS DE INFORMACIÓN

**UI. INTRODUCCIÓN A LOS
LENGUAJES DE MARCAS**

2024-2025

INDICE

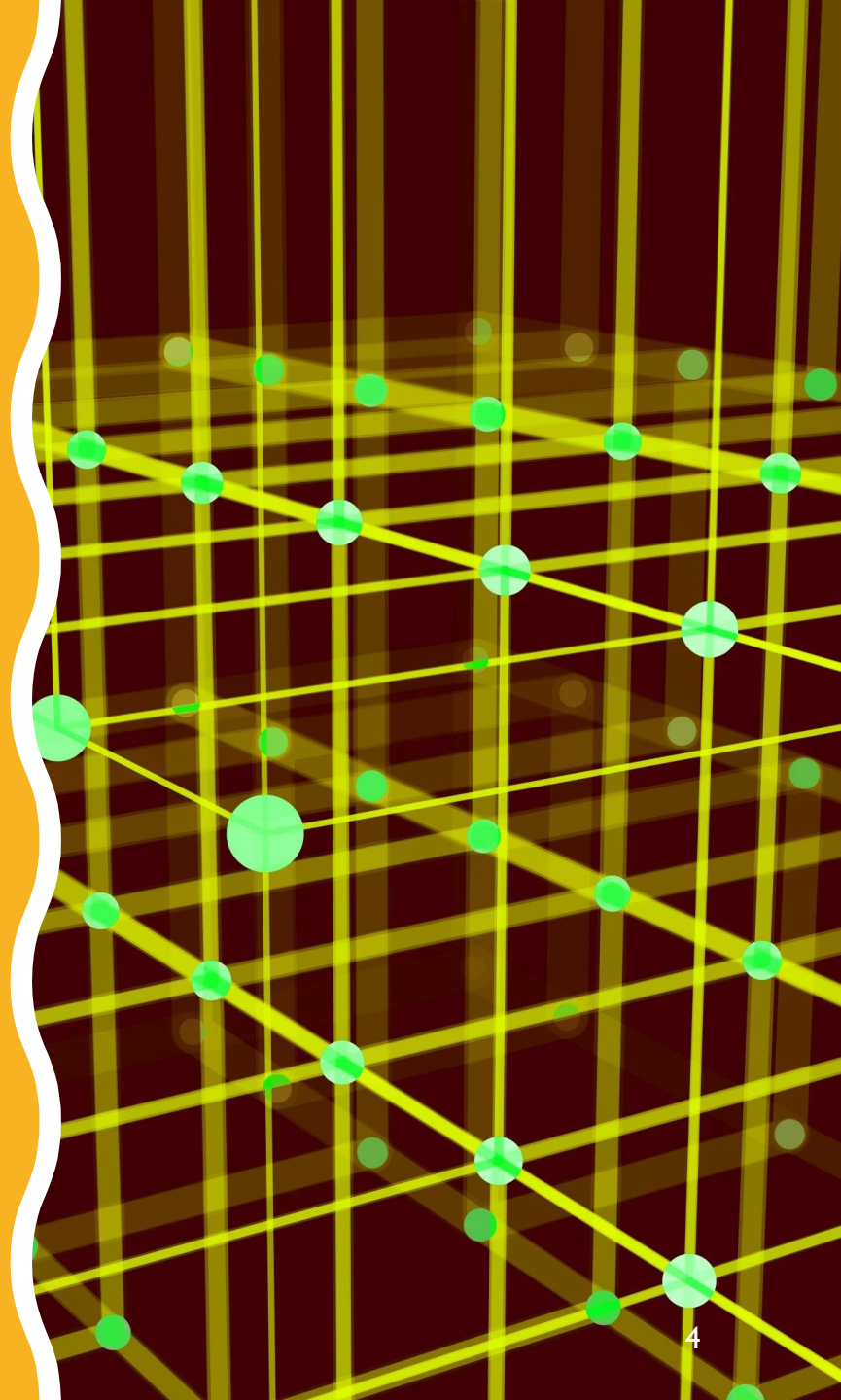
- INTRO
 1. Un poco de historia
 2. Características de los lenguajes de marcas
 3. XML. Estructura y Sintaxis
 4. Documentos XML bien formados
 5. XML Válido
 6. Herramientas
 7. Otros lenguajes



INTRO

- ¿Dónde está internet?
- Si pongo la URL de Instagram en mi navegador, ¿qué está pasando internamente? ¿Cómo me llega la información?

INTRO



¿QUÉ ES UN LENGUAJE DE MARCAS?

MARCA Y LENGUAJE DE MARCADO

```
<párrafo>Este es un ejemplo de  
texto estructurado mediante  
marcas.</párrafo>
```

- Mark-up Language
- Una “**marca**” **históricamente** es una anotación que intenta señalar a un tipógrafo cómo debe imprimirse o prepararse un texto determinado
- “**Marca**” **en la actualidad** es toda clase de códigos insertados en textos electrónicos que determinan el formato, el modo de impresión o cualquier otro proceso
- **Un lenguaje de marcado** en los documentos electrónicos es un tipo de lenguaje que combina texto con información extra acerca del texto. Esa información extra se entremezcla con el texto primario. El lenguaje de marcas más conocido en la actualidad es el HTML, que se utiliza en las páginas Web.

***mark-up:** to correct or write notes or instructions on a piece of writing, especially before it is published.*

***tag:** a small piece of paper or other material that is fixed to something to give information about it, or is fixed to someone to show who they are.*

RECONOCIMIENTO DE LAS CARACTERÍSTICAS DE LOS LENGUAJES DE MARCAS

`<párrafo>Este es un ejemplo
de texto estructurado
mediante
marcas.</párrafo>`

MARCA Y LENGUAJE DE MARCADO

- El marcado define una serie de códigos llamados **etiquetas**
- `<etiqueta>` Texto sobre el que se actúa `</etiqueta>`
 - Definen la estructura
 - Los elementos con que se organiza un documento (título, temas, capítulos...)
 - Ejemplo de etiqueta de estructura: `<head>` Esta es la cabecera de mi pagina `</head>`
 - Definen el formato en el que tienen que aparecer.
 - Se refiere a la presentación (tipo de fuente, negrita, cursiva, etc.)
 - Ejemplo de etiqueta de formato: ``Esto se muestra en negrita ``

RECONOCIMIENTO DE LAS CARACTERÍSTICAS DE LOS LENGUAJES DE MARCAS

CLASES DE LENGUAJES DE MARCAS

Marcas de presentación

- Indican el formato del texto.
- Este tipo de marcado es útil para maquetar la presentación de un documento para su lectura, pero resulta insuficiente para el procesamiento automático de la información. `
` (salto de línea)

Marcas de procedimientos

- Se utilizan para la presentación del texto
- Por ejemplo, para que se vea en negrita: `` **Esto está en negrita** ``

Marcas descriptivas

- También llamadas marcado descriptivo, o semántico.
- Aquí se utilizan las marcas para describir fragmentos de texto sin especificar cómo deben representarse. Algunos lenguajes diseñados para esto son el SGML y el XML. (Ejemplo XML: `<libro>`El Quijote`</libro>`)

1. UN POCO DE HISTORIA



UN POCO DE HISTORIA

- El concepto de lenguaje de marcas fue expuesto por vez primera por W.Tunnicliffe en 1967.
- Separación entre la presentación y la estructura del texto.
- Desarrollo de un estándar al que bautizaría como GenCode, destinado a la industria editorial.
- El editor Stanley Fish también expuso ideas similares a finales de los años 1960.
- Quien es considerado el padre de los lenguajes de marcas es **Charles Goldfarb**, investigador para la compañía IBM -->GML.



UN POCO DE HISTORIA



UN POCO DE HISTORIA

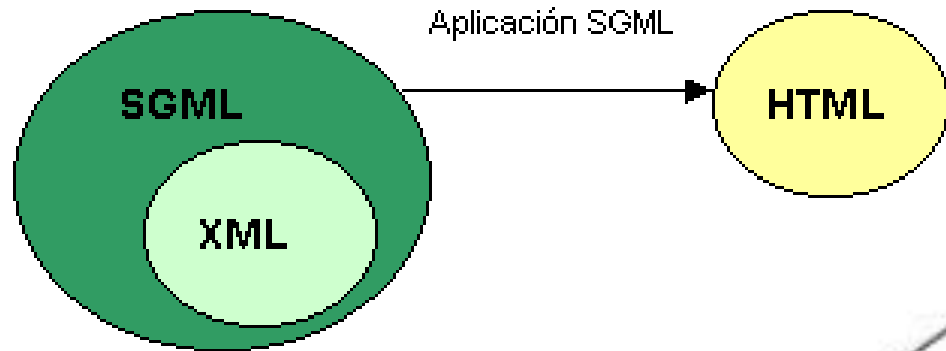
```
nt: Bungler OED      At: "<
try>
hwsec>
(hwgp>
(hwlem>bungler</hwlem>
(pron>b<I>a</I>ngla</pron>
(vfl>Also <vd>b</vd> <vf>bongler
</vfl>
(etym>f. as. nra. + (xra)<xlam>=
ten>On
(quot
<qd
<au
<uk
<at
```

LA ISO creó SGML en
1986 (Standard
Generalized Markup
Language)

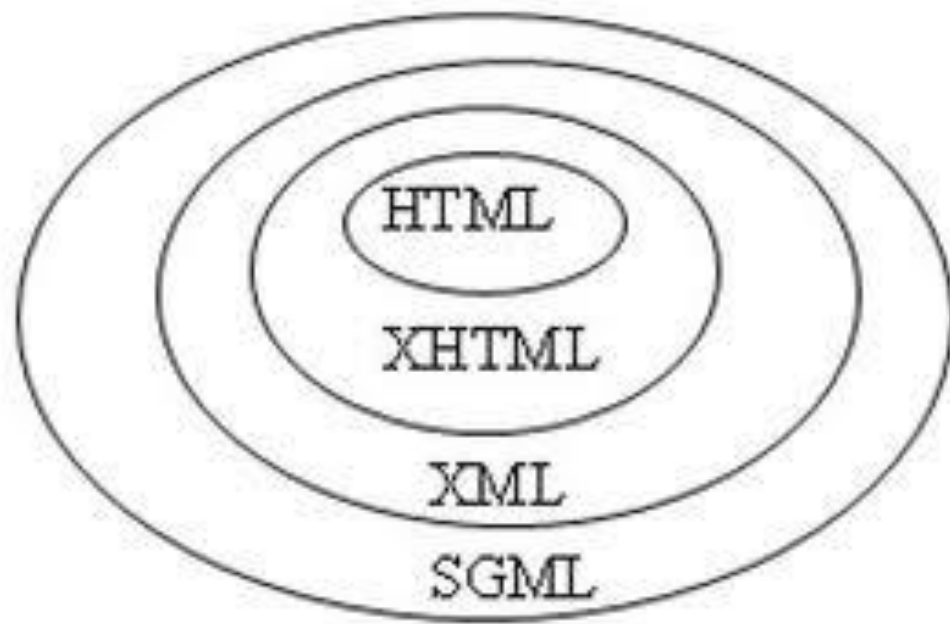
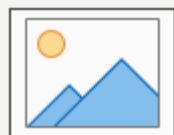


W3C

W3C creó XML en
1998, subconjunto
más fácil de utilizar
(eXtensible Markup
Language)

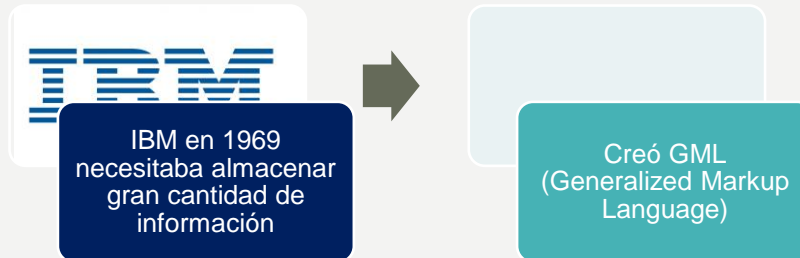


LENGUAJES DE MARCA



<https://www.youtube.com/watch?v=E EttUcYhv30&list=PLrFIr4pKV7bjxLjcyBACsqrliIBHuA31v&index=3>

UN POCO DE HISTORIA



- EL ABUELO: **GML**
 - IBM propuso un sistema de documentos que identificase a cada uno de sus elementos lógicos (títulos y subtítulos, direcciones, páginas, capítulos, párrafos, listas, etc.) con algún tipo de etiqueta dentro del propio documento.
 - La visualización e impresión de dichos documentos podría ser independiente del hardware en particular
 - IBM desarrolló una especie de pseudolenguaje de computadora que **combina sólo texto e instrucciones de formateado**.
 - Dicho lenguaje se llamó "lenguaje de marcas" e IBM lo bautizó como Lenguaje de marcas generalizado o GML (Generalized Markup Language).



UN POCO DE HISTORIA



- Ejemplo de GML

```
:h1.Chapter 1:  Introduction
:p.GML supported hierarchical containers, such as
:ol
:li.Ordered lists (like this one),
:li.Unordered lists, and
:li.Definition lists
:eol.
```

as well as simple structures.

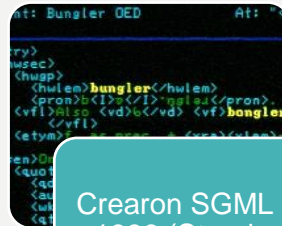
:p.Markup minimization (later generalized and formalized in SGML), allowed the end-tags to be omitted for the "h1" and "p" elements.

Esta foto de Autor desconocido se concede bajo licencia de [CC BY-SA-NC](#).

UN POCO DE HISTORIA



A la ISO le gustó



Crearon SGML en 1986 (Standard Generalized Markup Language)

EL PADRE: SGML

- GML fue un gran éxito, siendo adoptado por el gobierno de Estados Unidos, → necesidad de estandarizarlo
- ISO (Organización Internacional de Estándares) creó su lenguaje a partir del GML de IBM y lo llamó Lenguaje de marcas generalizado estándar o SGML (Standard Generalized Markup Language).
- SGML no almacena el diseño, sino la estructura lógica de los documentos. Lo hace a través de etiquetas o "tags" que se incluyen entre los signos < y > e indican cuando comienza y termina un título, una lista, etc.
- El objetivo era poder asegurar electrónicamente que los documentos importantes fuesen independientes de los formatos de archivo binario en constante cambio o de los sistemas operativos
- Era muy complicado.



UN POCO DE HISTORIA

- Ejemplo de SGML



```
<!doctype linuxdoc system>

<!-- Here's an SGML example file. Format it and print out the source, and
      use it as a model for your own SGML files. As you can see this is a
      comment.
-->

<article>

<!-- Title information -->

<title>Quick SGML Example
<author>Matt Welsh, <tt/mdlw@cs.cornell.edu/
<date>v1.0, 28 March 1994
<abstract>
This document is a brief example using the Linuxdoc-SGML DTD.
</abstract>

<!-- Table of contents -->
<toc>

<!-- Begin the document -->

<sect>Introduction

<p>
This is an SGML example file using the Linuxdoc-SGML DTD. You can format it
using the command
<tsscreen><verb>
% sgm12txt example.sgm1
</verb></tsscreen>
this will produce plain ASCII. You can also produce LaTeX, and HTML
and GNU info.

<sect>The source
```

UN POCO DE HISTORIA

EL PRIMOGÉNITO: **HTML**

- HTML: Lenguaje de marcas hipertextual o HyperText Markup Language), base de la World Wide Web.
- En 1990 Tim Berners-Lee, del Laboratorio Europeo de Física de Partículas tomó el **SGML + enlaces = HTML**.
- Era fácil de aprender.
- Fue tal el crecimiento que decidió crear el World Wide Web Consortium o W3C, encargado del desarrollo de lenguajes y estándares para la Web.
- SGML usa unas reglas, DTD's, (Document Type Definition) que explican qué etiquetas puede contener un archivo SGML.
- HTML es un SGML con un DTD concreto, estandarizado y usado por la mayoría.



UN POCO DE HISTORIA

- Ejemplo de HTML 4.01: aplicación de SGML

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>Universitat d'Alacant - Universidad de Alicante - University of Alicante</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>

<body bgcolor="#FFFFFF">
<table width="490" border="0" cellspacing="0" cellpadding="0" align="center">
<tr>
<td>
</td>
<td>
</td>
</tr>
</table>
<map name="Map">
<area shape="rect" coords="0,104,145,134" href="va/index.html" alt="Universitat d'Alacant">
<area shape="rect" coords="160,104,325,134" href="es/index.html" alt="Universidad de Alicante">
<area shape="rect" coords="342,104,490,134" href="en/index.html" alt="University of Alicante">
</map>
</body>
</html>
```

UN POCO DE HISTORIA

LOS HERMANOS: **XML** (EXTENSIBLE MARKUP LANGUAGE) Y XHTML

- XML
 - HTML tiene limitaciones y causa algunos problemas
 - No está pensado para impresión o tareas de diseño.
 - Las etiquetas son limitadas y poco flexibles.
 - Tampoco puede mostrar contenidos dinámicos.
 - Se decidió volver a la fuente SGML, pero SGML seguía siendo complicado
 - En 1998, el Consorcio W3 y empresas como Microsoft decidieron implementar el lenguaje XML (Lenguaje de marcas extensible o Extensible Markup Language) como sucesor del HTML.



UN POCO DE HISTORIA

LOS HERMANOS: XML (EXTENSIBLE MARKUP LANGUAGE) Y XHTML

- XML
 - Una diferencia fundamental de XML con su predecesor SGML es que en este lenguaje **no hay elementos predefinidos**.
 - Cada usuario puede **crear su propio lenguaje** para el formato de datos y documentos, su propio vocabulario, según sus necesidades, siguiendo ciertas reglas.
 - Elementos como <titulo>, <autor>, <sexo>, <fecha_nacimiento>, <pregunta>, <respuesta>, <nombre>, <apellido> o <precio> son perfectamente válidos en XML.
 - XML es como SGML un **Metalenguaje**.
 - No incluye ninguna información relativa al diseño. Sólo almacena datos y estructura. Para dar formato a esos datos para cada finalidad (web, impresión, proyección, Braille, etc.) se debe usar un lenguaje de diseño, como CSS o XSL.



UN POCO DE HISTORIA

- Ejemplo de XML. Formato RSS que se usa para difundir información actualizada frecuentemente a usuarios que se han suscrito a una fuente de contenidos, como periódicos o blogs

```
<?xml version="1.0" encoding="iso-8859-1"?>
<rss version="2.0">
  <channel>
    <title><![CDATA[INFORMACION - Alicante]]></title>
    <link><![CDATA[http://www.diarioinformacion.com]]></link>
    <description><![CDATA[INFORMACION - Alicante]]></description>
    <language>es-es</language>
    <copyright><![CDATA[Copyright INFORMACION]]></copyright>
    <ttl>60</ttl>
    <pubDate>2009-11-09T06:06:57Z</pubDate>
    <lastBuildDate>2009-11-09T05:06:57Z</lastBuildDate>
    <category>News</category>
    <image>
      <title><![CDATA[INFORMACION]]></title>
      <url><![CDATA[http://www.diarioinformacion.com/favicon.ico]]></url>
      <link><![CDATA[http://www.diarioinformacion.com]]></link>
      <description><![CDATA[INFORMACION - Alicante]]></description>
    </image>
    <item>
      <title><![CDATA[Nick Nolte rodará en Ciudad de la Luz]]></title>
      <link><![CDATA[http://www.diarioinformacion.com/cine/2009/11/10/cine-nick-nolte-rodara-ciudad/9]]></link>
      <description><![CDATA[El actor norteamericano participa en el rodaje de "Mi querido desconocido]]></description>
      <enclosure url='http://media.epi.es/www.diarioinformacion.com/media/fotos/noticias/150x200/2009-11-10-nick-nolte-rodara-ciudad-de-la-luz.jpg' type='image/jpeg' />
      <author><![CDATA[INFORMACION]]></author>
      <guid isPermaLink="true"><![CDATA[http://www.diarioinformacion.com/cine/2009/11/10/cine-nick-nolte-rodara-ciudad-de-la-luz/9]]></guid>
      <pubDate><![CDATA[2009-11-10T15:47:40Z]]></pubDate>
    </item>
    <item>
      <title><![CDATA[Un hombre armado toma como rehén al director de una escuela en EEUU]]></title>
      <link><![CDATA[http://www.diarioinformacion.com/internacional/2009/11/10/internacional-hombre-armado-toma-como-rehen-al-director-de-una-escuela-en-eeuu]]></link>
      <description><![CDATA[Las autoridades están negociando con el secuestrador, que podría ser el p]]></description>
      <author><![CDATA[INFORMACION]]></author>
      <guid isPermaLink="true"><![CDATA[http://www.diarioinformacion.com/internacional/2009/11/10/internacional-hombre-armado-toma-como-rehen-al-director-de-una-escuela-en-eeuu]]></guid>
      <pubDate><![CDATA[2009-11-10T15:22:12Z]]></pubDate>
    </item>
  </channel>
</rss>
```

UN POCO DE HISTORIA

- Ejemplo de XML con vocabulario propio

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/css" href="boe.css"?>
<!DOCTYPE BOE SYSTEM "boe.dtd">
<BOE>
  <TITULO>BOLETIN OFICIAL DEL ESTADO</TITULO>
  <FECHA><AAAA>2001</AAAA><MM>03</MM><DD>27</DD></FECHA>
  <SUMARIO>
    <SECCION>
      <TIT-SECCION>I. Disposiciones Generales</TIT-SECCION>
      <APARTADO>
        <ORGANISMO>MINISTERIO DE ASUNTOS EXTERIORES</ORGANISMO>
        <PARRAFO>
          <TEXT0>
            <LEY>Ley 1/2001 de 5 de marzo</LEY>, de la cesión gratuita a las Comunidades
              Autónomas el Impuesto sobre Animales y Cosas establecido por el
            <RD>REAL DECRETO 13/1991</RD>.
          </TEXT0>
          <PAGINA>11290</PAGINA>
        </PARRAFO>
        <PARRAFO>
          <TEXT0>
            <ORDEN>ORDEN de 28 de febrero de 2001</ORDEN> por la ...
          </TEXT0>
          <PAGINA>11290</PAGINA>
          <PAGINA>11291</PAGINA>
        </PARRAFO>
      </APARTADO>
      <APARTADO>
        <ORGANISMO>MINISTERIO DE HACIENDA</ORGANISMO>
        <PARRAFO>
          <TEXT0>
            <RD>REAL DECRETO 285/2001</RD>, de 19 de marzo, sobre ...
          </TEXT0>
          <PAGINA>11291</PAGINA>
          <PAGINA>11292</PAGINA>
          <PAGINA>11293</PAGINA>
        </PARRAFO>
      </APARTADO>
    </SECCION>
  </SUMARIO>
</BOE>
```


UN POCO DE HISTORIA

LOS HERMANOS: XML (EXTENSIBLE MARKUP LANGUAGE) Y XHTML

- **XHTML**
 - XML también resultaba complicado para los usuarios comunes
 - Muchos navegadores tenían problemas de compatibilidad con XML
 - Se decidió crear un "lenguaje de transición" :**XHTML**
 - Así nació el XHTML, una reformularización del HTML basada en XML.



UN POCO DE HISTORIA

- ¿En qué se parecen?

- GML

```
:ol
:li.Ordered lists (like this one),
:li.Unordered lists, and
:li.Definition lists
:eol.
```

- SGML

```
<title>Quick SGML Example
<author>Matt Welsh, <tt/mdw@cs.cornell.edu/
<date>v1.0, 28 March 1994
<abstract>
This document is a brief example using the Linuxdoc-SGML DTD.
</abstract>
```

- HTML

```
<html>
<head>
<title>Universitat d'Alacant - Universidad de Alicante - University of Alicante.
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
```

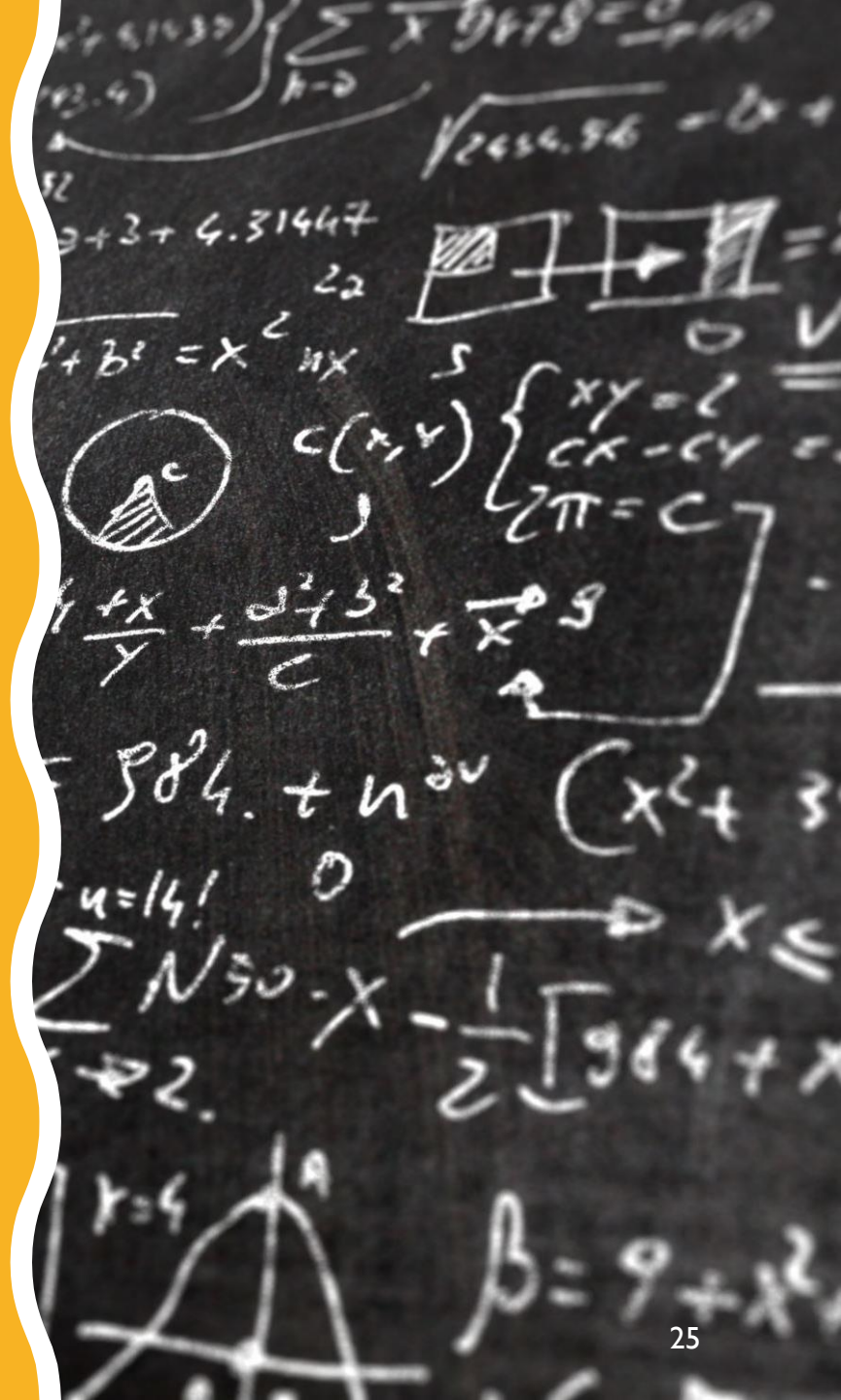
- XML

```
<rss version="2.0">
<channel>
<title><![CDATA[INFORMACION - Alicante]]></title>
<link><![CDATA[http://www.diarioinformacion.com]]></link>
<description><![CDATA[INFORMACION - Alicante]]></description>
<language>es-es</language>
```

- XML

```
<TITULO>BOLETIN OFICIAL DEL ESTADO</TITULO>
<FECHA><AAAA>2001</AAAA><MM>03</MM><DD>27</DD></FECHA>
<SUMARIO>
```

2. CARACTERÍSTICAS



CARACTERÍSTICAS

¿Un lenguaje de marcas es un lenguaje de programación?

```
press(void
```

```
    *bt = (s
```

```
    m_sleep =
```

```
    time_cnt =
```

```
    e(1)
```

```
    time cnt =
```

CARACTERÍSTICAS

- Elige una web.
- Visualiza el código en el navegador
- ¿Qué observas?



CARACTERÍSTICAS

Texto plano.

- Este tipo de codificación es que puede ser interpretada directamente, dado que son archivos de texto plano. Ej. Notepad

Compacidad.

- Las instrucciones de marcado se **entremezclan** con el propio contenido en un único archivo o flujo de datos.

Facilidad de procesamiento.

- Las organizaciones de estándares han venido desarrollando lenguajes especializados para los tipos de documentos de comunidades o industrias concretas

CARACTERÍSTICAS

TEXTO PLANO.VENTAJAS:

Legibilidad y Edición Fácil

Portabilidad

Compatibilidad con Versiones Futuras

Facilidad de Colaboración

Facilidad de Integración

Mantenimiento y Depuración Eficientes

Interoperabilidad

Documentación Clara y Autoexplicativa

Resistencia a la Obsolescencia

Adecuación para el Almacenamiento a Largo Plazo

CARACTERÍSTICAS

- **Compacidad.**

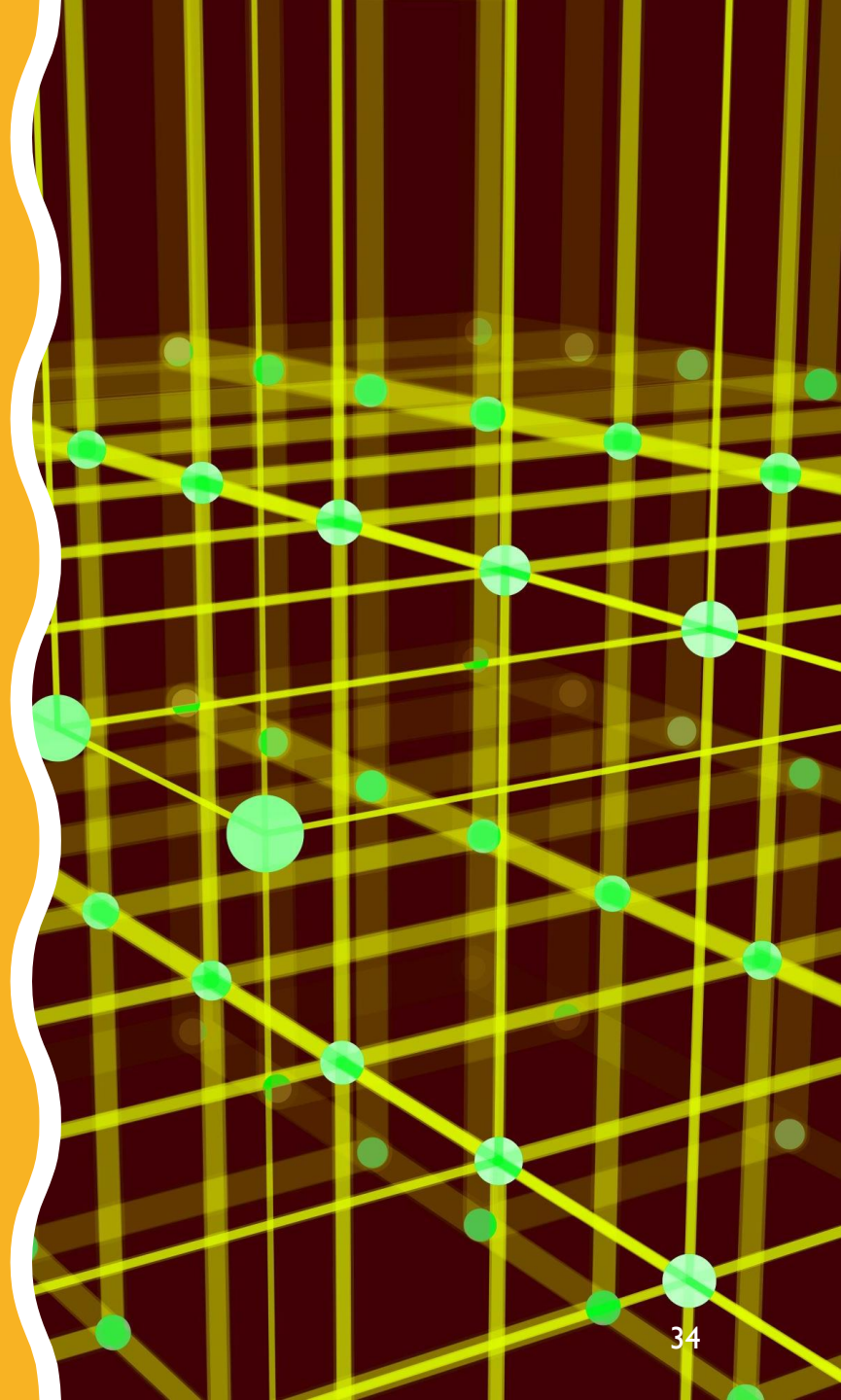
- Las instrucciones de marcado se **entremezclan** con el propio contenido en un único archivo o flujo de datos.

Ejemplos	HTML	LaTeX	Wikitexto
Título	<code><h1>Título</h1></code>	<code>\section{Título}</code>	<code>== Título ==</code>
Lista	<code> Punto 1 Punto 2 Punto 3 </code>	<code>\begin{itemize} \item Punto 1 \item Punto 2 \item Punto 3 \end{itemize}</code>	<code>* Punto 1 * Punto 2 * Punto 3</code>
texto en negrita	<code>texto</code>	<code>\bf{texto}</code>	<code>''' texto '''</code>
texto en <i>cursiva</i>	<code><i>texto</i></code>	<code>\it{texto}</code>	<code>'' texto ''</code>

CARACTERÍSTICAS

- **Facilidad de procesamiento.**
 - Las organizaciones de estándares han venido desarrollando lenguajes especializados para los tipos de documentos de comunidades o industrias concretas.
 - Uno de los primeros fue el CALS, utilizado por las fuerzas armadas de EE.UU. para sus manuales técnicos.
 - Otras industrias con necesidad de gran cantidad de documentación han elaborado lenguajes adaptados a sus necesidades.
 - Un ejemplo notable fue el caso de Sun Microsystems, empresa que optó por escribir la documentación de sus productos en SGML, ahorrando costes considerables. El responsable de aquella decisión fue Jon Bosak, que más tarde fundaría el comité del XML.

3. XML. ESTRUCTURA Y SINTAXIS



XML. ESTRUCTURA Y SINTAXIS



XML, eXtensible Markup Language, es un lenguaje de marcado “especial” →

metalenguaje: que nos permite definir nuestros propios lenguajes de marcado.

Es el estándar internacional para la definición de la estructura y el contenido de diferentes tipos de documentos electrónicos.

XML. ESTRUCTURA Y SINTAXIS

ORIGEN: SGML etiqueta la información, consiguiendo así tener identificada cada parte de la misma.

Este etiquetado sigue unas reglas, unos DTD's (Document Type Definition),
que explican qué tipos de etiquetas puede contener el archivo SGML, con qué atributos y la forma de relacionarse entre sí.

Pues bien, podríamos decir que HTML viene a ser un tipo de archivo SGML con un DTD concreto, que simplemente está estandarizado y, por tanto, admitido y usado por la mayoría.

Se puede decir que XML es una reducción de SGML, cuando menos un subconjunto del mismo, de modo que conserva casi todas sus características, pero es más sencillo de manejar y de aplicar, orientándose de este modo a su utilización en Internet.

XML. ESTRUCTURA Y SINTAXIS

- Aquí podemos ver un ejemplo muy sencillo:
¿Qué características tiene?

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<biblioteca>
  <libro>
    <titulo>El Gran Gatsby</titulo>
    <autor>F. Scott Fitzgerald</autor>
    <editorial/>
  </libro>
</biblioteca>
```

XML. ESTRUCTURA Y SINTAXIS

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<biblioteca>
  <libro>
    <titulo>El Gran Gatsby</titulo>
    <autor>F. Scott Fitzgerald</autor>
    <editorial/>
  </libro>
</biblioteca>
```

- XML es un lenguaje **descriptivo** lo que significa que las etiquetas identifican el tipo de contenido que contienen, es decir crea una estructura de datos contenidos en etiquetas semánticas, etiquetas que tienen un significado para la lectura humana.
- Los datos en XML se engloban en **etiquetas**, delimitadas por los caracteres <>
- **El nombre de las etiquetas será descriptivo**, tendrá un valor semántico para facilitar la lectura humana y para dotar de un significado a los datos que contenga. Ejemplo: <titulo>El Gran Gatsby</titulo>

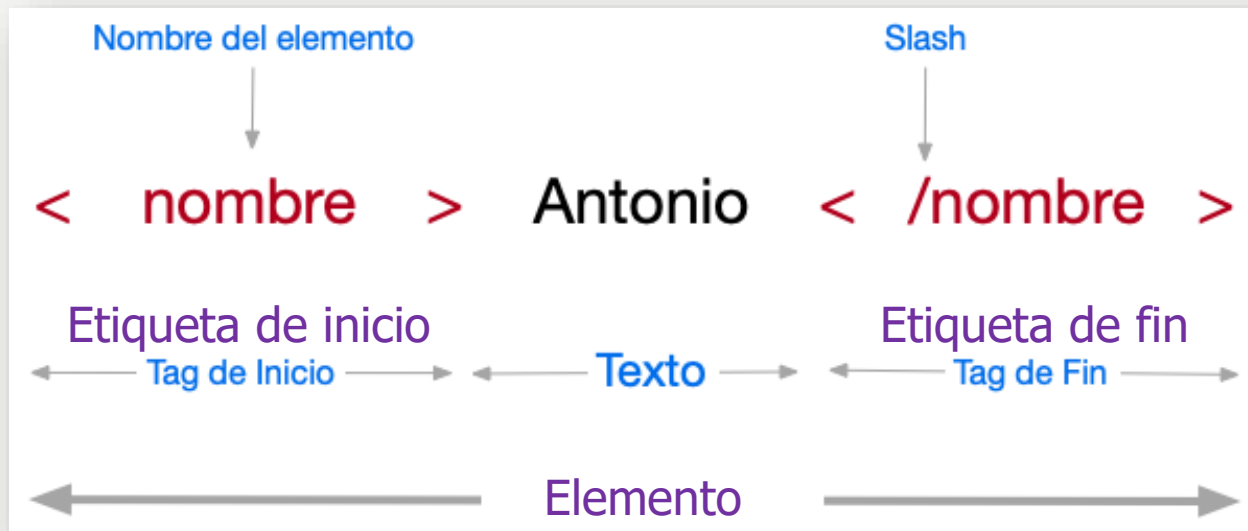
ELEMENTOS. ETIQUETAS

3. XML. ESTRUCTURA Y SINTAXIS.

XML. ESTRUCTURA Y SINTAXIS

Elemento

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<biblioteca>
  <libro>
    <titulo>El Gran Gatsby</titulo>
    <autor>F. Scott Fitzgerald</autor>
    <editorial/>
  </libro>
</biblioteca>
```



XML. ESTRUCTURA Y SINTAXIS

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<biblioteca>
  <libro>
    <titulo>El Gran Gatsby</titulo>
    <autor>F. Scott Fitzgerald</autor>
    <editorial/>
  </libro>
</biblioteca>
```

- En este caso la etiqueta contiene datos, por tanto, se deben escribir dos marcas, la de inicio de etiqueta `<animal>` y la de fin de etiqueta `</animal>`. **La de fin de etiqueta** siempre comenzará por “`</`”.
Ejemplo: `<animal/>`
- En este caso, la etiqueta no contiene datos, es **una etiqueta vacía**. Las etiquetas vacías siempre hay que “cerrarlas”, lo cual se hace finalizándolas con “`/>`”.

XML. ESTRUCTURA Y SINTAXIS

EJERCICIO

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<biblioteca>
  <libro>
    <titulo>El Gran Gatsby</titulo>
    <autor>F. Scott Fitzgerald</autor>
    <editorial/>
  </libro>
</biblioteca>
```

- Añade otros dos libros a la biblioteca
- ¿Cómo lo harías?

ATRIBUTOS

3. XML. ESTRUCTURA Y SINTAXIS.

XML. ESTRUCTURA Y SINTAXIS

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<biblioteca>
  <libro genero="novela de ficción">
    <titulo>El Gran Gatsby</titulo>
    <autor>F. Scott Fitzgerald</autor>
    <editorial/>
  </libro>
</biblioteca>
```

- **Atributos:**
- Las etiquetas pueden contener **atributos** que almacenan características del elemento.
- En XML: **<animal altura="25">Perro</animal>** con el atributo altura estamos almacenando el valor 25; puede significar que el animal mide 25 centímetros de altura o puede significar que al darle un formato al archivo XML utilicemos ese valor para pintar una celda de 25 píxeles de altura. Simplemente hemos almacenado la información y luego ya le daremos uso, pero no implica la aplicación de esa cualidad, de forma inmediata al elemento, es meramente un almacenaje de información.

XML. ESTRUCTURA Y SINTAXIS

EJERCICIO

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<biblioteca>
  <libro genero="novela de ficción">
    <titulo>El Gran Gatsby</titulo>
    <autor>F. Scott Fitzgerald</autor>
    <editorial/>
  </libro>
</biblioteca>
```

- Vamos a añadir atributos:
 - añade el atributo *idioma* a la etiqueta libro
 - Añade el atributo *nacionalidad* a la etiqueta autor

COMENTARIOS

3. XML. ESTRUCTURA Y SINTAXIS.

XML. ESTRUCTURA Y SINTAXIS

COMENTARIOS:

<!-- Este es un comentario en XML. Puede contener información adicional o notas. -->

Un comentario en XML es un tipo de anotación o nota que se incluye en un documento XML:

- Proporciona información adicional o aclaraciones destinadas a los humanos que leen el documento.
- No afectan al procesamiento del contenido por parte de las aplicaciones o sistemas informáticos.

Son útiles para:

- **Documentación:** Proporcionar notas, explicaciones o descripciones que ayuden a otros a entender el propósito o el contexto de un elemento o sección del documento XML.
- **Desactivación temporal:** Pueden utilizarse para "desactivar" temporalmente partes del XML sin eliminarlas, lo que puede ser útil durante el desarrollo o la depuración.

XML. ESTRUCTURA Y SINTAXIS

COMENTARIOS:

<!-- Este es un comentario en XML. Puede contener información adicional o notas. -->

Un comentario en XML se inicia con `<!--` y se cierra con `-->`, y todo lo que se encuentra entre estos delimitadores se trata como comentario y se ignora durante el procesamiento XML.

Es importante destacar que los comentarios en XML son para beneficio de los humanos que leen y mantienen el documento XML, y no tienen impacto en el contenido o la estructura del documento en sí ni en su procesamiento por parte de las aplicaciones XML.

Añade un comentario a tu archivo XML y válíalo

ENTIDADES PREDEFINIDAS

3. XML. ESTRUCTURA Y SINTAXIS.



XML. ESTRUCTURA Y SINTAXIS EJERCICIO

- ¿Qué ocurre si quiero añadir un título de un libro que se llame `<xml>` : “*El lenguaje*”?
- Inténtalo y válídalos
- ¿Qué ocurre?
- ¿Cómo lo soluciono?

XML. ESTRUCTURA Y SINTAXIS

- Existen otros tipos de datos como las **entidades predefinidas** que permiten introducir caracteres especiales; estas entidades se pueden definir en el **DTD** o **Schema**

Entidad	Carácter
&	&
<	<
>	>
'	'
"	"

XML. ESTRUCTURA Y SINTAXIS

EJERCICIO

- Añade un libro que se llame `<xml>` : “*El lenguaje*”.
- *Indica si está bien formado a través de una web de validación*

SECCIONES CDATA

3. XML. ESTRUCTURA Y SINTAXIS.

XML. ESTRUCTURA Y SINTAXIS

EJERCICIO

- Crea un XML que contenga un elemento raíz llamado documentos, que contenga elementos llamados archivos. Cada archivo tendrá dos elementos: nombre y contenido.
- Añade un archivo:
 - Nombre: pruebas.xml
 - Contenido:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<biblioteca>
  <libro genero="novela de ficción">
    <titulo>El Gran Gatsby</titulo>
    <autor>F. Scott Fitzgerald</autor>
    <editorial/>
  </libro>
</biblioteca>
```
 - Valídalo.

XML. ESTRUCTURA Y SINTAXIS EJERCICIO

- ¿Hay alguna forma más rápida de hacerlo y que sea más legible?

XML. ESTRUCTURA Y SINTAXIS

- Existen las secciones **CDATA** que permiten especificar datos, utilizando cualquier carácter, especial o no, sin que se interprete como marcado.
- Un ejemplo sería (primero usando entidades predefinidas y luego con un bloque CDATA):

- **Con entidades predefinidas:**

```
<ejemplo>
  &lt;html>
    &lt;head>&lt;title>Rock &amp; Roll&lt;/title>&lt;/head>
    &lt;/html>
</ejemplo>
```

- **Con un bloque CDATA**

```
<ejemplo>
  <![CDATA[ <!--lo de dentro no es marcado sino texto
  <html>
    <head>
      <title>Rock & Roll</title>
    </head>
  </html>
]]>
</ejemplo>
```

- El único “carácter” no permitido en una cadena CDATA es “]]” que son el cierre de la sección CDATA. Todos los demás están permitidos.

XML. ESTRUCTURA Y SINTAXIS

EJERCICIO

- Añade a tu archivo xml anterior el mismo libro pero con CDATA. Valídalo
- Añade un archivo:

- Nombre: pruebas.xml

- Contenido:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<biblioteca>
  <libro genero="novela de ficción">
    <titulo>El Gran Gatsby</titulo>
    <autor>F. Scott Fitzgerald</autor>
    <editorial/>
  </libro>
</biblioteca>
```


DECLARACIONES

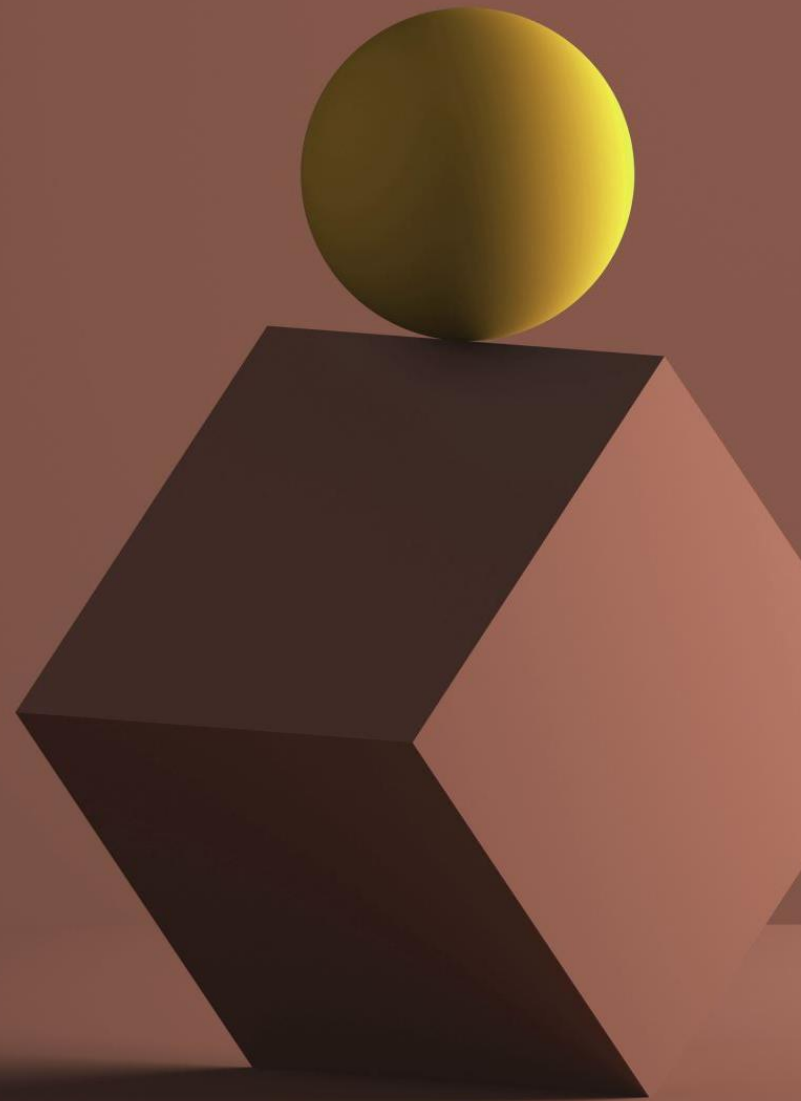
3. XML. ESTRUCTURA Y SINTAXIS.

XML. ESTRUCTURA Y SINTAXIS

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<biblioteca>
  <libro>
    <titulo>El Gran Gatsby</titulo>
    <autor>F. Scott Fitzgerald</autor>
    <editorial/>
  </libro>
</biblioteca>
```

- **DECLARACIONES: LA PRIMERA LÍNEA**
- Atributos, algunos obligatorios y otros no:
 - **version:** Indica la versión de XML usada en el documento. Es obligatorio ponerlo, a no ser que sea un documento externo a otro que ya lo incluía.
 - **encoding:** La forma en que se ha codificado el documento. Se puede poner cualquiera, y depende del parser (analizador) el entender o no la codificación. Por defecto es UTF-8, aunque podrían ponerse otras, como UTF-16, US-ASCII, ISO-8859-1, etc. No es obligatorio salvo que sea un documento externo a otro principal.
 - **standalone:** Indica si el documento va acompañado de un DTD ("no"), o no lo necesita ("yes"); en principio no hay porqué ponerlo, porque luego se indica el DTD si se necesita.

4. DOCUMENTOS XML BIEN FORMADOS



XML BIEN FORMADOS

Dos estados de validación:

Está **bien formado** si sus documentos respetan la gramática básica especificada por el W3C.

Es válido si el documento respeta una gramática descrita previamente llamada DTD (Document Type Definition) o un esquema XML (XSD)

Ejemplo: en mi DTD he especificado que sólo son válidos los elementos biblioteca, libro, título, autor. Y que autor y título deben estar jerárquicamente debajo de libro. Y que libro puede tener un atributo que se llama género.

XML BIEN FORMADOS

- **Bien formados:**
- **W3C Word Wide Web Consorcio :**
 - Es un consorcio internacional donde las organizaciones miembros, trabajan conjuntamente para desarrollar estándares Web.
 - La misión del W3C es:
 - Guiar la Web hacia su máximo potencial a través del desarrollo de protocolos y pautas que aseguren el crecimiento futuro de la Web.
 - El W3C trata de alcanzar su objetivo principalmente a través de la creación de Estándares Web y Pautas. Desde 1994, el W3C ha publicado más de ciento diez estándares, denominados Recomendaciones del W3C



XML BIEN FORMADOS



- Un documento XML está bien formado según las especificaciones W3C si: <https://www.w3.org/TR/xml11/#sec-notation>

Estructura jerárquica de etiquetas

Etiquetas vacías cerradas

Un solo elemento raíz

Valores de atributo: con comillas dobles " o simples '

Tipo de letra y espacios en blanco

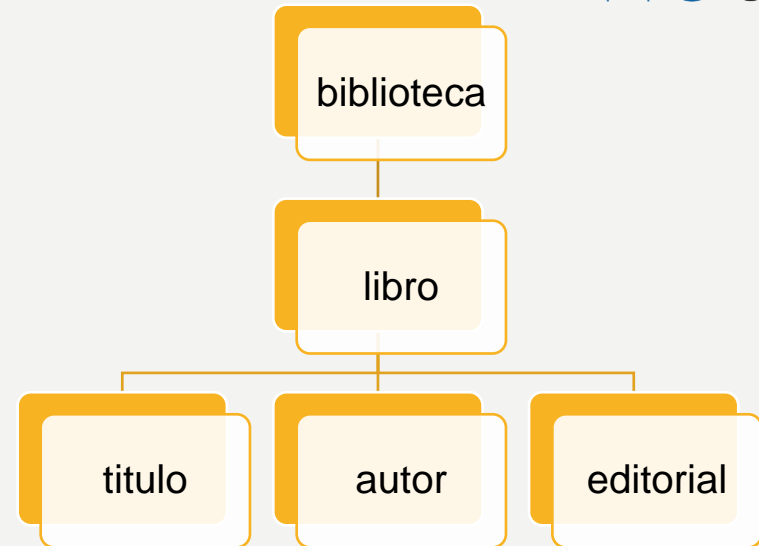
Nomenclatura

Marcado y datos

XML BIEN FORMADOS



- **Estructura jerárquica de etiquetas:**
 - Los documentos XML deben seguir una **estructura estrictamente jerárquica** con lo que respecta a las etiquetas que delimitan sus elementos. Una etiqueta debe estar correctamente "incluida" en otra.



```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<biblioteca>
  <libro>
    <titulo>El Gran Gatsby</titulo>
    <autor>F. Scott Fitzgerald</autor>
    <editorial/>
  </libro>
</biblioteca>
```


XML BIEN FORMADOS



- Estructura jerárquica de etiquetas:



```
<td>HTML <b>permite  
<i>esto</b></i></td>
```




```
<td>En XML la <b>estructura  
<i>es</i>  
jerárquica</b></td>
```

XML BIEN FORMADOS



Etiquetas vacías:

- XML permite elementos sin contenido, vacíos, pero la etiqueta debe cerrarse siempre de una de las siguientes maneras:
- `<elemento-sin-contenido/>`  **Recomendado**
- `<elemento-sin-contenido></elemento-sin-contenido>`

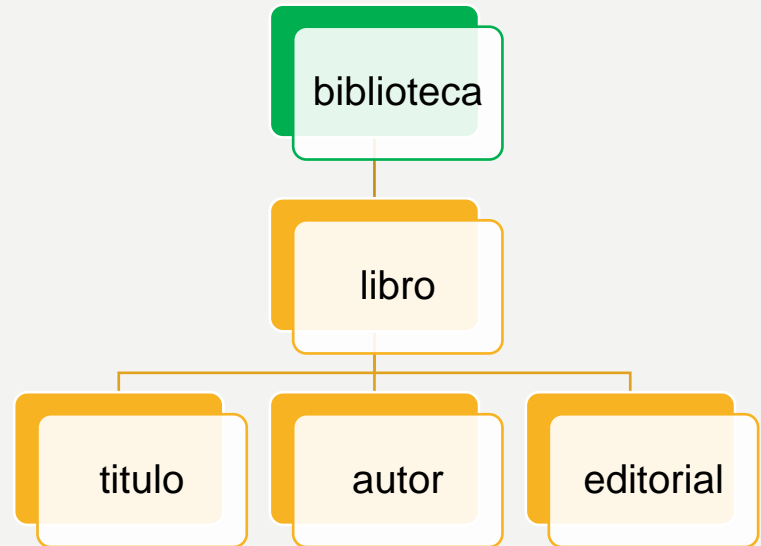
```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<biblioteca>
  <libro>
    <titulo>El Gran Gatsby</titulo>
    <autor>F. Scott Fitzgerald</autor>
    <editorial/>
  </libro>
</biblioteca>
```

XML BIEN FORMADOS



Elemento raíz: uno y solo uno

- Solo permiten un elemento raíz, del que todos los demás sean parte



```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<biblioteca>
  <libro>
    <titulo>El Gran Gatsby</titulo>
    <autor>F. Scott Fitzgerald</autor>
    <editorial/>
  </libro>
</biblioteca>
```

XML BIEN FORMADOS



- Estructura jerárquica de etiquetas:



```
<animal>  
  <dueño></dueño>  
</animal>  
<especialidad>  
  <titulo/>  
</especialidad>
```



```
<veterinario>  
  <animal>  
    <dueño>pepe perez</dueño>  
  </animal>  
  <especialidad>  
    <titulo/>  
  </especialidad>  
</veterinario>
```

XML BIEN FORMADOS



Valores de atributos:

- Los valores de atributos en XML, al contrario de HTML, siempre deben estar encerradas en comillas simples (') o dobles (").
- Así tendríamos:
- En HTML `` (esto sería válido en HTML)
- En XML `<enlace ruta="http://www.developer.com" />`
- Si dentro del atributo fuera necesario el uso de comillas, entonces éstas no podrán coincidir en el tipo con las que delimitan el contenido del atributo:
- En XML `<enlace ruta="http://www.developer.com?id= 'php' " />`

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<biblioteca>
  <libro genero="novela">
    <titulo>El Gran Gatsby</titulo>
    <autor>F. Scott Fitzgerald</autor>
    <editorial/>
  </libro>
</biblioteca>
```

XML BIEN FORMADOS



Tipo de letra y espacios en blanco:

Tipo de letra: Distingue entre mayúsculas y minúsculas. No es lo mismo `<elemento>` que `<Elemento>` que `<ELEMENTO>`.

Espacios en blanco: espacios, tabuladores y retornos de carro.

- **Espacios en blanco fuera de los elementos:** Los espacios en blanco al principio y al final de un documento XML son ignorados y no tienen ningún efecto en la interpretación del documento.
- **Espacios en blanco dentro de los elementos:** Los espacios en blanco dentro de los elementos XML generalmente se **conservan**, a menos que se especifique lo contrario.
- **Preservación de espacios en blanco:** Puedes utilizar la entidad ` ` para representar un espacio en blanco literal en XML. Además, puedes usar elementos como `<pre>` o atributos como `xml:space` para indicar que los espacios en blanco deben preservarse en ciertas partes del documento.
- **Atributo `xml:space`:** XML 1.1 incluye el atributo `xml:space`, que se puede agregar a un elemento para especificar si se deben preservar o colapsar los espacios en blanco dentro de ese elemento y sus descendientes. Los valores posibles son "default" (colapsado) y "preserve" (preservado).
- **Los nombres de atributos o etiquetas** no deben tener espacios en blanco. Normalmente se usa camelCase. Ejemplo: `nombreAutor`.

```
<parrafo xml:space="preserve">Este es un      ejemplo con múltiples  
espacios    en blanco.</parrafo>
```

XML BIEN FORMADOS



Nomenclatura

Según la especificación XML 1.1:

Un nombre empieza con una letra o guion bajo, y continúa con letras, dígitos, guiones, rayas, dos puntos o puntos, denominados de forma global como caracteres de nombre.

NO se deben crear nombres que empiecen con la cadena "xml", "xMI", "XML" o cualquier otra variante. Las letras y rayas se pueden usar en cualquier parte del nombre. También se pueden incluir dígitos, guiones y caracteres de punto, pero no se puede empezar por ninguno de ellos. El resto de caracteres, como algunos símbolos, y espacios en blanco, no se pueden usar.

```
<parrafo xml:space="preserve">Este es un      ejemplo con múltiples  
espacios    en blanco.</parrafo>
```

XML BIEN FORMADOS



Marcado y datos

Las construcciones como etiquetas, referencias de entidad y declaraciones se denominan "marcas". Éstas son las partes del documento que el procesador XML espera entender. El resto del documento que se encuentra entre las marcas, son los datos que resultan entendibles por las personas.

Es sencillo reconocer las marcas en un documento XML. Son aquellas porciones que empiezan con "<" y acaban con ">", o bien, en el caso de las referencias de entidad, empiezan por "&" y acaban con ";".

```
<parrafo xml:space="preserve">Este es un      ejemplo con múltiples  
espacios    en blanco.</parrafo>
```


= "xs:string" minOccurs="1" maxOccurs="unbounded"> <xs:attribute name="sexo" type="xs:string" use="optional"/> </xs:element> <xs:element name="apellido" type="xs:string" maxOccurs="unbounded">

XML 1.1 BLUEBERRY

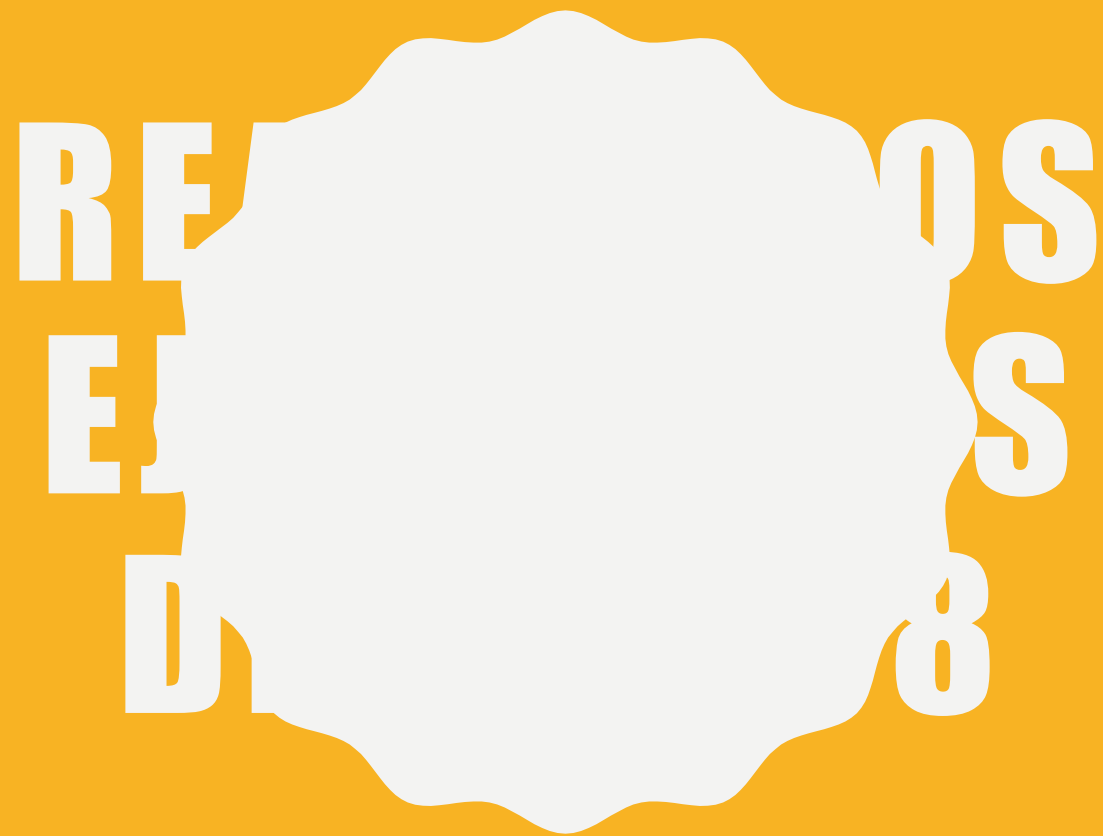
XML 1.1

- XML 1.1 es una revisión de XML 1.0 que introduce algunas características adicionales y cambios con respecto a la versión anterior. A continuación, se presentan las principales diferencias entre XML 1.0 y XML 1.1:
 - **Caracteres permitidos:**
 - XML 1.0: XML 1.0 se basa en la especificación Unicode 2.0 y tiene restricciones en la gama de caracteres permitidos.
 - XML 1.1: XML 1.1 se basa en la especificación Unicode 3.1, lo que amplía la gama de caracteres permitidos.
 - **Nombres de etiquetas y atributos:**
 - XML 1.1: XML 1.1 permite un conjunto más amplio de caracteres en los nombres de etiquetas y atributos, lo que hace que sea más flexible en términos de nomenclatura.
 - **Entidades predefinidas:**
 - XML 1.1: XML 1.1 mantiene las entidades predefinidas de XML 1.0 y agrega algunas más, como ' y ".
 - **Espacios en blanco en nombres de etiquetas:**
 - XML 1.0: No se permiten espacios en blanco en nombres de etiquetas en XML 1.0.
 - XML 1.1: XML 1.1 permite espacios en blanco en nombres de etiquetas, pero esto es poco común y no se recomienda por razones de legibilidad y mantenimiento.
 - **Compatibilidad hacia atrás:**
 - XML 1.0: XML 1.0 es más ampliamente compatible hacia atrás y se usa ampliamente en sistemas y aplicaciones.
 - XML 1.1: XML 1.1 introduce cambios que pueden no ser compatibles con algunas aplicaciones XML 1.0 existentes.

ESTILO RECOMENDADO

RECOMENDACIONES DE ESTILO

- **Código indentado:** usar sangría, tabular
- **Usar minúsculas en etiquetas y atributos:** Se recomienda utilizar letras minúsculas para nombres de etiquetas y atributos. Esto ayuda a distinguirlos de las entidades XML y mejora la legibilidad.
- **Nombrar de manera descriptiva:** Elija nombres de etiquetas, atributos y elementos que sean descriptivos y significativos para su contenido.
- **Escapar caracteres especiales:** Si necesita incluir caracteres especiales como &, <, >, use entidades predefinidas o CDATA para asegurarse de que no se interpreten como etiquetas XML.
- **Documentar:** Incluya comentarios que expliquen la estructura o el propósito de partes importantes del documento.
- **Siga convenciones de nomenclatura:** Mantenga consistencia en la forma en que nombra elementos y atributos. Elija un estilo de nomenclatura (como guiones bajos o notación **camelCase**) y sígalo en todo el documento.
- No espacios en blanco dentro de las etiquetas: <xml> sí, <xml > no





ANEXO: CODIFICACIÓN DE CARACTERES

CODIFICACIÓN DE CARACTERES

- Es el método que permite convertir un carácter de un lenguaje natural en un símbolo de otro sistema de representación como un número o una secuencia de pulsos eléctricos en un sistema electrónico, aplicando normas o reglas de codificación
- Normas de codificación:

ASCII

ASCII
Extendido

ISO 8859-1 o
ISO Latin I

Unicode

UTF-8
UTF-16
UTF-32

CODIFICACIÓN DE CARACTERES

ASCII

ASCII
Extendido

Unicode

- ASCII
- Codifica con 7 bits: $2^7=128$ símbolos.
- El octavo dígito se usa para detectar errores de transmisión (paridad)
- No incluye caracteres acentuados, ni ¿ ni muchos símbolos matemáticos, letras griegas...
- Busca la tabla ASCII y encuentra la ñ

CODIFICACIÓN DE CARACTERES

ASCII

ASCII
Extendido

Unicode

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[END OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

CODIFICACIÓN DE CARACTERES

ASCII

ASCII
Extendido

Unicode

- ASCII Extendido
- Debido a las limitaciones de ASCII se definieron códigos usando los 8 bits. Hay diferentes normas según lenguas
 - Una de las codificaciones de caracteres más comunes que se basa en ASCII extendido es **ISO 8859-1**, también conocida como **Latin-1**.
 - Latin-1 utiliza el octavo bit para representar caracteres utilizados en idiomas europeos occidentales, como letras acentuadas, letras con diéresis y otros caracteres específicos de estos idiomas., como Á, É, Í, Ó, Ú (letras acentuadas), ä, ö, ü, ñ (letras con diéresis y ñ), ÿ, ÿ, ç, ß (caracteres especiales).
 - Otros ejemplos so ISO 8859-5 (Latin/Cyrillic): Se enfoca en la escritura cirílica utilizada en idiomas como ruso, búlgaro y serbio o ISO 8859-6 (Latin/Arabic): Diseñado para idiomas escritos en árabe, como árabe clásico y persa.
- 8 bits no son suficientes para codificar todos los alfabetos y escrituras del mundo

CODIFICACIÓN DE CARACTERES

ASCII

ASCII
Extendido

ISO 8859-1 o
ISO Latin I

Unicode

Caracteres de control ASCII

DEC	HEX	Símbolo ASCII	
00	00h	NULL	(carácter nulo)
01	01h	SOH	(inicio encabezado)
02	02h	STX	(inicio texto)
03	03h	ETX	(fin de texto)
04	04h	EOT	(fin transmisión)
05	05h	ENQ	(enquiry)
06	06h	ACK	(acknowledgement)
07	07h	BEL	(timbre)
08	08h	BS	(retroceso)
09	09h	HT	(tab horizontal)
10	0Ah	LF	(salto de línea)
11	0Bh	VT	(tab vertical)
12	0Ch	FF	(form feed)
13	0Dh	CR	(retorno de carro)
14	0Eh	SO	(shift Out)
15	0Fh	SI	(shift In)
16	10h	DLE	(data link escape)
17	11h	DC1	(device control 1)
18	12h	DC2	(device control 2)
19	13h	DC3	(device control 3)
20	14h	DC4	(device control 4)
21	15h	NAK	(negative acknowle.)
22	16h	SYN	(synchronous idle)
23	17h	ETB	(end of trans. block)
24	18h	CAN	(cancel)
25	19h	EM	(end of medium)
26	1Ah	SUB	(substitute)
27	1Bh	ESC	(escape)
28	1Ch	FS	(file separator)
29	1Dh	GS	(group separator)
30	1Eh	RS	(record separator)
31	1Fh	US	(unit separator)
127	20h	DEL	(delete)

Caracteres ASCII imprimibles

DEC	HEX	Símbolo	DEC	HEX	Símbolo	DEC	HEX	Símbolo
32	20h	espacio	64	40h	@	96	60h	`
33	21h	!	65	41h	A	97	61h	a
34	22h	"	66	42h	B	98	62h	b
35	23h	#	67	43h	C	99	63h	c
36	24h	\$	68	44h	D	100	64h	d
37	25h	%	69	45h	E	101	65h	e
38	26h	&	70	46h	F	102	66h	f
39	27h	'	71	47h	G	103	67h	g
40	28h	(72	48h	H	104	68h	h
41	29h)	73	49h	I	105	69h	i
42	2Ah	*	74	4Ah	J	106	6Ah	j
43	2Bh	+	75	4Bh	K	107	6Bh	k
44	2Ch	,	76	4Ch	L	108	6Ch	l
45	2Dh	-	77	4Dh	M	109	6Dh	m
46	2Eh	.	78	4Eh	N	110	6Eh	n
47	2Fh	/	79	4Fh	O	111	6Fh	o
48	30h	0	80	50h	P	112	70h	p
49	31h	1	81	51h	Q	113	71h	q
50	32h	2	82	52h	R	114	72h	r
51	33h	3	83	53h	S	115	73h	s
52	34h	4	84	54h	T	116	74h	t
53	35h	5	85	55h	U	117	75h	u
54	36h	6	86	56h	V	118	76h	v
55	37h	7	87	57h	W	119	77h	w
56	38h	8	88	58h	X	120	78h	x
57	39h	9	89	59h	Y	121	79h	y
58	3Ah	:	90	5Ah	Z	122	7Ah	z
59	3Bh	;	91	5Bh	[123	7Bh	{
60	3Ch	<	92	5Ch	\	124	7Ch	
61	3Dh	=	93	5Dh]	125	7Dh	}
62	3Eh	>	94	5Eh	^	126	7Eh	~
63	3Fh	?	95	5Fh	_			

elCodigoASCII.com.ar

ASCII extendido

DEC	HEX	Símbolo	DEC	HEX	Símbolo	DEC	HEX	Símbolo	DEC	HEX	Símbolo
128	80h	Ç	160	A0h	á	192	C0h	Ł	224	E0h	Ó
129	81h	ü	161	A1h	í	193	C1h	ł	225	E1h	ô
130	82h	é	162	A2h	ó	194	C2h	Ł	226	E2h	Ô
131	83h	â	163	A3h	û	195	C3h	ł	227	E3h	Õ
132	84h	ä	164	A4h	ñ	196	C4h	Ł	228	E4h	ö
133	85h	à	165	A5h	Ñ	197	C5h	ł	229	E5h	Ö
134	86h	á	166	A6h	ª	198	C6h	Ł	230	E6h	μ
135	87h	ç	167	A7h	º	199	C7h	ł	231	E7h	µ
136	88h	ê	168	A8h	¿	200	C8h	Ł	232	E8h	þ
137	89h	è	169	A9h	®	201	C9h	ł	233	E9h	Û
138	8Ah	ë	170	AAh	¬	202	CAh	Ł	234	EAh	Ü
139	8Bh	ï	171	ABh	½	203	CBh	ł	235	EBh	Ù
140	8Ch	î	172	ACH	¼	204	CCh	Ł	236	ECh	Ý
141	8Dh	ì	173	ADh	¡	205	CDh	ł	237	EDh	Ÿ
142	8Eh	Ā	174	Aeh	«	206	CEh	Ł	238	Eeh	ˆ
143	8Fh	Ă	175	Afh	»	207	CFh	ł	239	Efh	˙
144	90h	É	176	B0h	⋮	208	D0h	Ł	240	F0h	±
145	91h	æ	177	B1h	⋮	209	D1h	ł	241	F1h	±
146	92h	Æ	178	B2h	⋮	210	D2h	Ł	242	F2h	ˉ
147	93h	ô	179	B3h	⋮	211	D3h	ł	243	F3h	¾
148	94h	ò	180	B4h	⋮	212	D4h	Ł	244	F4h	¶
149	95h	õ	181	B5h	⋮	213	D5h	ł	245	F5h	§
150	96h	ù	182	B6h	⋮	214	D6h	Ł	246	F6h	÷
151	97h	û	183	B7h	⋮	215	D7h	ł	247	F7h	ˆ
152	98h	ÿ	184	B8h	⋮	216	D8h	Ł	248	F8h	ˆ
153	99h	Œ	185	B9h	⋮	217	D9h	ł	249	F9h	ˆ
154	9Ah	Ů	186	BAh	⋮	218	DAh	Ł	250	FAh	ˆ
155	9Bh	ø	187	BBh	⋮	219	DBh	ł	251	FBh	ˆ
156	9Ch	£	188	BCb	⋮	220	DCb	Ł	252	FBh	ˆ
157	9Dh	Ø	189	BDh	⋮	221	DDh	ł	253	FDh	ˆ
158	9Eh	×	190	BEh	⋮	222	DEh	Ł	254	FEh	ˆ
159	9Fh	f	191	BFh	⋮	223	DFh	ł	255	FFh	ˆ

CODIFICACIÓN DE CARACTERES

ASCII

ASCII
Extendido

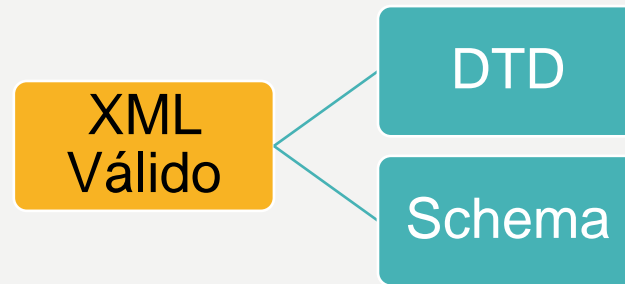
Unicode

UTF-8
UTF-16
UTF-32

- **UNICODE:** Como no son suficientes 8 bits se usan 8 o 16 o 32
- **UTF-8** (Unicode Transformation Format - 8 bits):
 - Codificación de longitud variable: UTF-8 utiliza una codificación de longitud variable, lo que significa que puede representar caracteres con diferentes longitudes de bytes. Los caracteres comunes se representan con 1 byte, mientras que caracteres menos comunes o más allá del plano básico de Unicode requieren más bytes.
 - Es eficiente en términos de almacenamiento.
 - Compatibilidad: UTF-8 es compatible con ASCII, lo que facilita la transición desde codificaciones más antiguas.
- **UTF-16** (Unicode Transformation Format - 16 bits):
 - Es común scripts asiáticos y caracteres matemáticos. Se utiliza en tecnologías como Java y el sistema de archivos NTFS de Windows.
- **UTF-32** (Unicode Transformation Format - 32 bits):
 - Uso en procesamiento interno: UTF-32 es útil en aplicaciones donde el procesamiento interno del texto necesita ser rápido y eficiente. Se utiliza en ciertas bibliotecas y sistemas de procesamiento de texto.

5. XML VÁLIDO

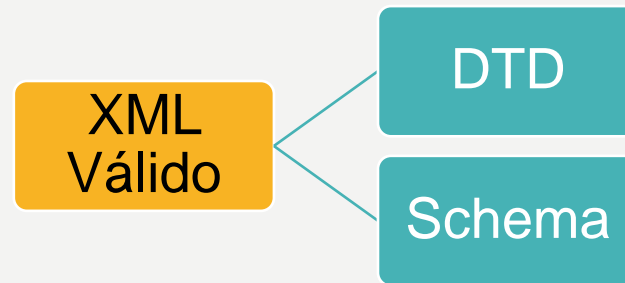
XML VÁLIDO



- Debe estar **bien formado** y seguir las reglas de un **DTD** o **Schema XML**
- Los **DTD** explican qué tipos de etiquetas puede contener el archivo XML, qué hijos puede tener cuántos de cada tipo, qué tipo de contenido pueden tomar (texto, numérico, etc), qué atributos puede tener cada etiqueta, qué valores pueden tomar esos atributos y, también, la forma de relacionarse entre sí las etiquetas, es decir, su jerarquía

```
<!DOCTYPE ficha [  
  <!ELEMENT ficha (nombre+, apellido+, direccion+, foto?)>  
    <!ELEMENT nombre (#PCDATA)>  
      <!ATTLIST nombre sexo (masculino|femenino) #IMPLIED>  
    <!ELEMENT apellido (#PCDATA)>  
    <!ELEMENT direccion (#PCDATA)>  
    <!ELEMENT foto EMPTY>  
>
```

XML VÁLIDO



- Debe estar **bien formado y seguir las reglas de un DTD o Schema XML**
- Los **Schemas** tienen ventajas sobre los **DTDs**:
 - Usan sintaxis de XML, al contrario que los DTD.
 - Permiten especificar los tipos de datos.
 - Son extensibles.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="ficha">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="nombre" type="xs:string" maxOccurs="unbounded">
          <xs:attribute name="sexo" type="xs:string" use="optional"/>
        </xs:element>
        <xs:element name="apellido" type="xs:string" maxOccurs="unbounded"/>
        <xs:element name="direccion" type="xs:string" maxOccurs="unbounded"/>
        <xs:element name="foto" type="xs:empty" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

5. ESPACIOS DE NOMBRES



ESPACIOS DE NOMBRES

- Supongamos que estás trabajando en un proyecto y tenemos lo siguiente:

```
<cliente>
```

```
  <nombre>Juan Pérez</nombre>
```

```
  <mascota>
```

```
    <nombre>Scooby</nombre>
```

```
  </mascota>
```

```
</cliente>
```

Hay un conflicto de nombres. ¿Cómo lo solucionamos?

ESPACIOS DE NOMBRES

```
<cliente>  
  <nombre>Juan Pérez</nombre>  
  <mascota>  
    <nombre>Scooby</nombre>  
  </mascota>  
</cliente>
```



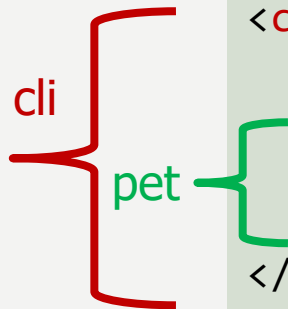
```
<cli:cliente xmlns:cli='http://www.prueba.com/cliente'>  
  <cli:nombre>Juan Pérez</cli:nombre>  
  <pet:mascota xmlns:pet='http://www.prueba.com/pet'>  
    <pet:nombre>Scooby</pet:nombre>  
  </pet:mascota>  
</cli:cliente>
```

ESPACIOS DE NOMBRES

```
<cli:cliente xmlns:cli='http://www.prueba.com/cliente'>
  <cli:nombre>Juan Pérez</cli:nombre>
  <pet:mascota xmlns:pet='http://www.prueba.com/pet'>
    <pet:nombre>Scooby</pet:nombre>
  </pet:mascota>
</cli:cliente>
```

- Un **espacio de nombres XML** es una **recomendación W3C** para proporcionar elementos y atributos **con nombre único** en una instancia XML.
- Una instancia 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, se resuelve la ambigüedad existente entre elementos o atributos que se llamen igual.
- Los **nombres de elementos dentro de un espacio de nombres deben ser únicos**.
- Se puede definir un espacio de nombres XML (namespace XML) como un conjunto de nombres, los cuales se identifican por una referencia **URI** (UniformResourceIdentifiers) que se usan en los documentos XML como tipos de elemento y nombres de atributos.
- El concepto de espacios de nombres (namespaces) permite particionar el conjunto de todos los nombres posibles, de forma que se pueda definir a qué zona de ese espacio corresponde una etiqueta.

ESPACIOS DE NOMBRES



```
<cli:cliente xmlns:cli='http://www.prueba.com/cliente'>
  <cli:nombre>Juan Pérez</cli:nombre>
  <pet:mascota xmlns:pet='http://www.prueba.com/pet'>
    <pet:nombre>Scooby</pet:nombre>
  </pet:mascota>
</cli:cliente>
```

- Para declarar un espacio de nombres que afecte a una etiqueta (elemento) se usará el atributo xmlns.
- Dicho atributo xmlns puede ir seguido de un identificador para el espacio de nombres y separados por “:”.
- Usa prefijos significativos.
- **Alcance de los espacios 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.

ESPACIOS DE NOMBRES

```
<cli:cliente xmlns:cli='http://www.prueba.com/cliente'>
  <cli:nombre>Juan Pérez</cli:nombre>
  <pet:mascota xmlns:pet='http://www.prueba.com/pet'>
    <pet:nombre>Scooby</pet:nombre>
  </pet:mascota>
</cli:cliente>
```

- Declaración de los espacios de nombres:
 - Hay que destacar que el **URI** (Uniform Resource Identifier) **no se lee realmente como una dirección**; se trata como una cadena de texto por el Parser XML.
 - El hecho de usar una **URL** (Uniform Resource Locator) (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.
 - Se recomienda que sea una URI pero no es obligatorio: <elemento xmlns:pre="prueba" /> es correcto.

ESPACIOS DE NOMBRES

- **Espacios de nombres por defecto:**
- También se puede declarar el espacio de nombres sin asignarle un nombre, con lo cual el atributo xmlns se escribirá de la misma forma que otros atributos:


```
<elemento xmlns ="http://www.uriprueba.com/ns">
```

- Esto se hace para evitar sobrecargar el código con prefijos.
- Los namespace por defecto no se aplican a los atributos, por tanto cualquier namespace aplicado a un atributo tiene que estar asociado a un prefijo.

```
<elemento xmlns:pre="http://www.prueba.com"  
pre:atributo="valor" />
```

```
<?xml version="1.0"?>  
<cliente xmlns='http://www.Espacio_de_nombres_XML/cliente'  
xmlns:ped='http://www.Espacio_de_nombres_XML/pedido'>  
  <numero_ID>1232654</numero_ID>  
  <nombre>Fulanito de Tal</nombre>  
  <telefono>99999999</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>  
</cliente>
```

**REFORMAS
E
DEL**



**OS
S
10**



A top-down view of various school and office supplies arranged on a light blue background. The items include a large metal compass, several colorful markers (green, blue, yellow, black), a teal stapler, a black magnifying glass, a roll of white tape, a grey stapler, a black ruler, a pair of tortoiseshell glasses, and several teal paper clips. The supplies are scattered across the frame, with some overlapping. A white and yellow wavy border is visible on the left side.

HERRAMIENTAS DE EDICION

- XML Copy Editor GPL 2.0. Básico
- Eclipse IDE for Enterprise java and Web Developers
- Altova XML Spy
- Adobe Dreamweaver : para front-end

7. OTROS LENGUAJES



OTROS LENGUAJES

- **Markdown**

- <https://es.wikipedia.org/wiki/Markdown>
- <https://markdown.es/>

- **LaTeX**: LaTeX es un sistema de composición de documentos ampliamente utilizado para la creación de documentos técnicos y científicos. Permite la creación de documentos de alta calidad con fórmulas matemáticas y estructura compleja. Utiliza comandos y marcas para formatear documentos técnicos y científicos. Aunque no es un lenguaje de marcas en el sentido estricto, se utiliza para formatear documentos.
- **JSON** (JavaScript Object Notation): Aunque no es un lenguaje de marcas en el sentido tradicional, JSON se utiliza ampliamente para representar datos estructurados en forma de objetos y matrices. Es comúnmente utilizado en aplicaciones web y servicios API. Es un formato de serialización de datos que no utiliza etiquetas o marcas para estructurar datos, sino que se basa en objetos y matrices.
- **YAML** (YAML Ain't Markup Language): YAML es un formato de serialización de datos legible por humanos que se utiliza para configuración y representación de datos estructurados. Es comúnmente utilizado en configuración de software y archivos de configuración. Se basa en una estructura de indentación.
- **RDF** (Resource Description Framework): RDF es un lenguaje de marcas utilizado para representar metadatos y descripciones de recursos en la web semántica. Se utiliza para vincular y describir datos en la web de una manera que las máquinas puedan comprender.

IMÁGENES

- Todas las imágenes son proporcionadas por Office, de autor desconocido está bajo licencia [CC BY-SA-NC](#)