

# Reconocimiento de las características de lenguajes de marcas.

## Caso práctico

María y Félix son los fundadores y propietarios de una asesoría legal y empresarial, que tiene su sede en Cantabria, con oficinas en los municipios más importantes de la región.

María, licenciada en Derecho, ejercía como abogada especializada en derecho laboral y representaba a alguna empresa, además de particulares en su propio despacho situado en Torrelavega. Tenía una red informática cliente-servidor sobre un sistema operativo Windows y trabajaba con una base de datos de documentos jurídicos.

Félix, diplomado en Ciencias Empresariales, había creado una asesoría empresarial, ubicada en Santander, que básicamente, se encargaba de la contabilidad de varias PYMES. También tenía una red cliente-servidor, pero ésta bajo un sistema Linux con software libre de contabilidad.

Ambos eran amigos y un día en que habían estado hablando de sus respectivos trabajos, decidieron que sus ingresos podían aumentar sustancialmente si, además de mantener sus respectivas carteras de clientes, se unían y formaban una sociedad que ofreciese a las empresas asesoría legal y empresarial de forma conjunta.

Desde el principio, la idea de asociarse fue un éxito. Al cabo de dos años el volumen del negocio se había extendido y se hizo imprescindible el intercambio de comunicación entre ambos.

Dado que trabajaban con sistemas informáticos diferentes se planteaba el problema de cómo podían compartir información sobre los clientes comunes manteniendo la infraestructura informática con la que trabaja cada uno.

Consultaron el problema a Juan, un técnico en administración de sistemas informáticos en red, y éste les dijo que no había ningún problema de interconexión si los ficheros que manejaban se ajustaban a un formato estándar conocido como XML. Según lo que Juan les dijo, generar documentos con dicho estándar apenas requiere conocimientos previos de informática, por tanto era una solución que parecía perfecta para su problema.

# Orientaciones para el alumnado

## Descripción Unidad UT1

En esta unidad de trabajo estudiaras la evolución de los lenguajes de marcas y las características elementales de los mismos. Además serás capaz de generar documentos XML básicos.

Con los conocimientos que aquí se aportarán podrás comenzar a trabajar y elaborar documentos XML que tengan cierta funcionalidad. Todas las herramientas y posibilidades de XML que veremos más adelante se construyen sobre conceptos aquí presentados.



[Ministerio de Educación y Formación Profesional](#). (Dominio público)

**Materiales formativos de FP Online propiedad del Ministerio de Educación y Formación Profesional.**

[Aviso Legal](#)

# 1.- Lenguajes de Marcas.

---

## Introducción

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

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

## Ejemplo

Aspecto de un documento realizado en un lenguaje de marcas:

```
1 <carta>
2   <fecha>22/11/2006</fecha>
3   <presentacion>Estimado cliente:</presentacion>
4   <contenido>bla bla bla bla ...</contenido>
5   <firma>Don José Gutiérrez González</firma>
6 </carta>
```

En la práctica, en un mismo documento pueden combinarse varios tipos diferentes de lenguajes de marcas.

Los lenguajes de marcas se pueden clasificar como sigue:

- ✓ De presentación: Define el formato del texto.
- ✓ De procedimientos: Orientado también a la presentación pero, en este caso, el programa que representa el documento debe interpretar el código en el mismo orden en que aparece.
- ✓ Descriptivo o semántico: Describen las diferentes partes en las que se estructura el documento pero sin especificar cómo deben representarse.

Algunos ejemplos de lenguajes de marcado agrupados por su ámbito de utilización son:

Documentación electrónica:

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

Tecnologías de Internet:

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

Otros lenguajes especializados:

- ✓ **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** (Music Extended Markup Language): Permite el intercambio de partituras entre distintos editores de partituras.

## Autoevaluación

### Los lenguajes de marcas se utilizan para:

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

Respuesta errónea. Te recomiendo volver a leer el apartado.

Respuesta incorrecta. Te recomiendo volver a leer el apartado.

No es la respuesta correcta. Te recomiendo volver a leer el apartado.

¡Correcta!

## Solución

1. Incorrecto
2. Incorrecto
3. Incorrecto
4. Opción correcta



## 2.- Evolución de los lenguajes de marcas.

---

### Introducción

En los años 70 surgieron unos lenguajes informáticos, distintos de los lenguajes de programación, orientados a la gestión de información. Con el desarrollo de los editores y procesadores de texto aparecieron los primeros lenguajes informáticos especializados en tareas de descripción y estructuración de información: los lenguajes de marcas. Paralelamente, también, emergieron otros lenguajes informáticos, orientados a la representación, almacenamiento y consulta eficiente de grandes cantidades de datos: lenguajes y sistemas de bases de datos.

Los lenguajes de marcas surgieron, inicialmente, como lenguajes formados por el conjunto de códigos de formato que los procesadores de texto introducen en los documentos para dirigir el proceso de presentación (impresión) mediante una impresora. Como en el caso de los lenguajes de programación, inicialmente estos códigos de formato estaban ligados a las características de una máquina, programa o procesador de textos concreto y, en ellos, inicialmente no había nada que permitiese al programador ("formateador" de documentos en este caso) abstraerse de las características del procesador de textos y expresar de forma independiente a éste la estructura y la lógica interna del documento.

### Ejemplo

Código de marcas anterior a GML. Las etiquetas son de invención propia.

Dado el siguiente documento:

```
1 <times 14><color verde><centrado> Este texto es un ejemplo para mostrar la utilización primitiva de las marcas</centrado><,  
2 <color granate><times 10><cursiva>Para realiza este ejemplo se utilizan etiquetas de nuestra invención. </cursiva>  
3 Las partes importantes del texto pueden resaltarse usando la  
4 <negrita>negrita</negrita>, o el <subrayar>subrayado</subrayar></times 10></color>
```

Mostrar retroalimentación

Al imprimirlo se obtendría:

Este texto es un ejemplo para mostrar la utilización primitiva de las marcas

*Para realiza este ejemplo se utilizan etiquetas de nuestra invención. Las partes importantes del texto pueden resaltarse usando la **negrita** , o el subrayado*

Posteriormente, se añadieron este tipo de características, como medio de presentación a la pantalla. Los códigos de estilo de visualización anteriores ya no aparecían, y se empleaban otros medios para marcados, distintos de la inclusión a mano de cadenas formateadoras. Ese proceso se automatizó y bastaba pulsar una combinación de teclas, o un botón, para lograr los resultados requeridos. Aunque esto era sólo una abstracción, para su uso interno, las aplicaciones seguían utilizando marcas, para delimitar aquellas partes del texto que tenían un formato especial.

Este marcado estaba exclusivamente orientado a la presentación de la información, aunque pronto se percataron de las posibilidades del marcado y le dieron nuevos usos que resolvían una gran variedad de necesidades. De este modo apareció el formato generalizado.



## 2.1.- GML (Generalized Markup Language).

---

Uno de los problemas que se conocen desde hace décadas en la informática es la falta de estandarización en los formatos de información usados por los distintos programas.

Para resolver este problema, en los años sesenta IBM encargó a Charles F. Goldfab 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, cuyo objetivo era describir los documentos de tal modo que el resultado fuese independiente de la plataforma y la aplicación utilizada.

## 2.2.- SGML (Standard Generalized Markup Language).

---

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

### Ejemplo

Documento SGML sencillo:

```
1 <email>
2   <remitente>
3     <persona>
4       <nombre> Pepito </nombre>
5       <apellido> Grillo </apellido>
6     </persona>
7   </remitente>
8   <destinatario>
9     <direccion> pinocho@hotmail.com </direccion>
10  </destinatario>
11  <asunto>¿quedamos?</asunto>
12  <mensaje> Hola, he visto que ponen esta noche la película que querías ver. ¿Te apetece ir?</mensaje>
13 </email>
```

## 2.3.- HTML (HyperText Markup Language).

---

En 1989/90 Tim Berners-Lee creó el **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**, que, en realidad, era una combinación de dos estándares ya existentes:

- ✓ **ASCII**: Es el formato que cualquier procesador de textos sencillo puede reconocer y almacenar. Por tanto es un formato que permite la transferencia 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, desde su creación, tanto las herramientas de software como los navegadores, que permiten visualizar páginas **HTML**, no han parado de mejorar.

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.

### Ejemplo

Ejemplo: Documento HTML

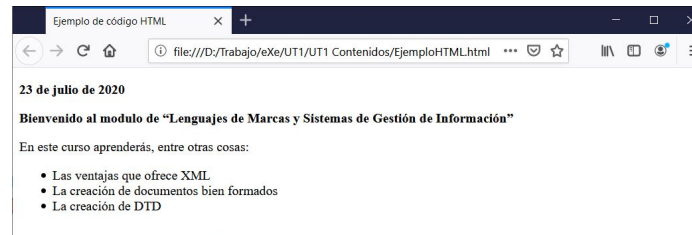
```
1 <html>
2   <head>
3     <title> Ejemplo de código HTML</title>
```

```

4      </head>
5      <body>
6          <p></p>
7          <p>
8              <b>23 de julio de 2020</b>
9          </p>
10         <p><b> Bienvenido al modulo de “Lenguajes de Marcas y Sistemas de Gestión de Información” </b></p>
11         <p> En este curso aprenderás, entre otras cosas:<br/>
12         <ul>
13             <li>Las ventajas que ofrece XML </li>
14             <li>La creación de documentos bien formados </li>
15             <li>La creación de DTD</li>
16         </ul>
17         </p>
18     </body>
19 </html>

```

En el navegador se visualizaría así:



Rubén Carrasco Peña - A partir de materiales del Ministerio de Educación ([CC BY-NC-SA](#))

## 2.4.- XML (eXtensible Markup Language).

Para resolver estos problemas de **HTML** el **W3C** establece, en 1998, el estándar internacional **XML**, un lenguaje de marcas puramente estructural que **no incluye ninguna información relativa al diseño**. Está convirtiéndose 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.

**XML** es un metalenguaje caracterizado por:

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

En realidad **XML** es un conjunto de estándares relacionados entre sí y que son:

- ✓ **XSL**, eXtensible Style Language. Permite definir hojas de estilo para los documentos XML e incluye capacidad para la transformación de documentos.
- ✓ **XML Linking Language**, incluye Xpath, Xlink y Xpointer. Determinan aspectos sobre los enlaces entre documentos XML.
- ✓ **XML Namespaces**. Proveen un contexto al que se aplican las marcas de un documento de XML y que sirve para diferenciarlas de otras con idéntico nombre válidas en otros contextos.
- ✓ **XML Schemas**. Permiten definir restricciones que se aplicarán a un documento XML. Actualmente los más usados son las **DTD**.

### Ejemplo

Documento XML

```
1 <?xml version="1.0" encoding="iso-8859-1"?>
2 <!DOCTYPE biblioteca">
3 <biblioteca>
```

```
4      <ejemplar tipo Ejem="libro" titulo="XML practico" editorial="Ediciones Eni">
5      <tipo> <libro isbn="978-2-7460-4958-1" edicion="1" paginas="347"></libro> </tipo>
6      <autor nombre="Sebastien Lecomte"></autor>
7      <autor nombre="Thierry Boulanger"></autor>
8      <autor nombre="Ángel Belinchon Calleja" funcion="traductor"></autor>
9      <prestado lector="Pepito Grillo">
10         <fecha_pres dia="13" mes="mar" año="2009"></fecha_pres>
11         <fecha_devol dia="21" mes="jun" año="2009"></fecha_devol>
12     </prestado>
13 </ejemplar>
14 <ejemplar tipo Ejem="revista" titulo="Todo Linux 101. Virtualización en GNU/Linux" editorial="Studio Press">
15     <tipo>
16         <revista>
17             <fecha_publicacion mes="abr" año="2009"></fecha_publicacion>
18         </revista>
19     </tipo>
20     <autor nombre="Varios"></autor>
21     <prestado lector="Pedro Picapiedra">
22         <fecha_pres dia="12" mes="ene" año="2010"></fecha_pres>
23     </prestado>
24 </ejemplar>
25 </biblioteca>
```

## 2.5.- XML vs HTML.

---

A continuación encontrarás una tabla comparativa de ambos lenguajes.

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

### Ejemplo XML vs. HTML

## Ejemplo fichero con código XML

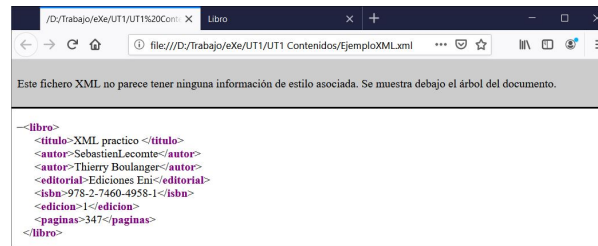
```
1 <?xml version="1.0" encoding="iso-8859-1"?>
2 <!DOCTYPE libro>
3 <libro>
4   <titulo>XML practico </titulo>
5   <autor>SebastienLecomte</autor>
6   <autor>Thierry Boulanger</autor>
7   <editorial>Ediciones Eni</editorial>
8   <isbn>978-2-7460-4958-1</isbn>
9   <edicion>1</edicion>
10  <paginas>347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

## Ejemplo de fichero con código HTML

```
1 <html>
2   <head>
3     <title>Libro</title>
4   </head>
5   <body>
6     <h3>XML practico</h3><br>
7     <p>autores: Sebastien Lecomte,
8     Thierry Boulanger</p>
9     <ul>
10      <li>editorial: Ediciones Eni</li>
11      <li>isbn:978-2-7460-4958-1</li>
12      <li>edicion: 1 </li>
13      <li>paginas: 347</li>
14    </ul>
15  </body>
16 </html>
```

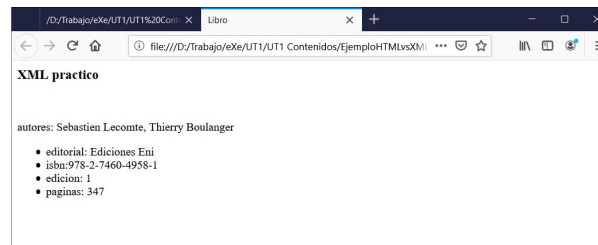


## Visualización en Navegador fichero código XML



Rubén Carrasco a partir de material del Ministerio ([CC BY-NC-SA](#))

## Visualización en Navegador fichero código HTML



Rubén Carrasco Peña a partir de material del Ministerio ([CC BY-NC-SA](#))

## 2.6.- Comparación de XML con SGML.

<u>XML</u>	<u>SGML</u>
Uso sencillo	Uso complejo
Trabaja con ..... documentos bien formados. No exige que estén validados	Solo trabaja con ..... documentos válidos
Facilita el desarrollo de aplicaciones de bajo coste	Su complejidad hace que las aplicaciones informáticas para procesar SGML sean muy costosas
Es muy utilizado en informática y en más áreas de aplicación	Solo se utiliza en sectores muy específicos
Compatibilidad e integración con <u>HTML</u>	No hay una compatibilidad con HTML definida
Formato y estilos fáciles de aplicar	Formateo y estilos relativamente complejos
No usa etiquetas opcionales	

### Autoevaluación

¿Cuáles son las características comunes de XML y SGML?

- ☐ Guardan el formato de un documento.
- ☐ Guardan la estructura lógica de los documentos.
- ☐ Guardan los documentos en el formato universal txt.

- ☐ Guardan el formato de los documentos independientemente de la plataforma.

Respuesta errónea. Te recomiendo volver a leer el apartado.

¡Correcto!

Respuesta incorrecta. Te recomiendo volver a leer el apartado.

No es la respuesta correcta. Te recomiendo volver a leer el apartado.

## Solución

1. Incorrecto
2. Opción correcta
3. Incorrecto
4. Incorrecto

**Para saber más**

La recomendación de **XML** publicada por el W3C es pública y accesible en:

<https://www.w3.org/TR/xml/>

## 2.7.- 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.

<etiqueta>texto que sufrirá las consecuencias de la etiqueta</etiqueta>

### Ejemplo: Etiqueta HTML de subrayado (Underline)

1 | `<u>Esto está subrayado</u>`

En el navegador el texto se verá:

Esto está subrayado

Las últimas especificaciones emitidas por el [W3C](#) indican la necesidad de que vayan escritas **siempre en minúsculas** para considerar que el documento está correctamente creado.

# Autoevaluación

¿Cuál de las siguientes líneas es correcta?

- ☐ <i>Texto en cursiva
- ☐ <i>Texto en cursiva<i>
- ☐ <i>Texto en cursiva</i>
- ☐ <l>Texto en cursiva<l>

Respuesta errónea. Te recomiendo volver a leer el apartado.

Respuesta incorrecta. Te recomiendo volver a leer el apartado.

¡Correcta!

No es la respuesta correcta. Te recomiendo volver a leer el apartado.

## Solución

1. Incorrecto
2. Incorrecto
3. Opción correcta
4. Incorrecto

## 2.8.- Herramientas de edición.

---

### Caso práctico

**Félix** tiene la curiosidad de saber si para codificar lenguajes de marcas hay que utilizar algún software característico o basta con usar un editor de texto plano, como en el caso de XML.

**Juan** le cuenta que puede bastar el bloc de notas, pero que existen varios editores que facilitan la tarea.

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

#### ✔ Editores XML

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

Para crear documentos **XML** complejos e ir añadiendo datos es conveniente usar algún editor **XML**. Estos nos ayudan a crear estructuras y etiquetas de los elementos usados en los documentos, además algunos incluyen ayuda para la creación de otros elementos como **DTD**, hojas de estilo **CSS** o **XSL**, ... El **W3C** ha desarrollado un editor de **HTML**, **XHTML**, **CSS** y **XML** gratuito cuyo nombre es [Amaya](#).

#### ✔ Procesadores XML

Los procesadores **XML** permiten leer los documentos **XML** y acceder a su contenido y estructura. Un procesador es un conjunto de módulos de software, entre los que se encuentra un **parser** o analizador de **XML**, que comprueba que el documento cumple las normas establecidas para que pueda abrirse.

Los procesadores **XML** pueden obliguen a trabajar sólo con documentos de tipo válido (entonces se denominan "validadores") o pueden sólo exigir que el documento esté bien formado ("no validadores").



El modo en que los procesadores deben leer los datos **XML** está descrito en la recomendación de **XML** establecida por **W3C**.

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

Para interpretar el código **XML** se puede utilizar cualquier navegador.

**XML** también se puede utilizar para el intercambio de datos entre aplicaciones. En este caso, hay que recurrir a motores independientes, que se ejecutan sin que nos demos cuenta. Por ejemplo [JAXP de Oracle](#).

## Autoevaluación

**Para crear documentos XML es necesario:**

- ☐ Software especializado para la tecnología XML.
- ☐ Herramientas de validación de XML.
- ☐ Un block de notas y un navegador.
- ☐ Al menos, un editor XML.

Respuesta errónea. Te recomiendo volver a leer el apartado.

Respuesta incorrecta. Te recomiendo volver a leer el apartado.

¡Correcta!

No es la respuesta correcta. Te recomiendo volver a leer el apartado.

## Solución

1. Incorrecto
2. Incorrecto
3. Opción correcta
4. Incorrecto

## Para saber más

Información sobre analizadores **XML** - <http://xml.coverpages.org/index.html>

expat - XML Parser Toolkit - <http://www.jclark.com/xml/expat.html>

## 3.- XML, estructura y sintaxis.

---

### Introducción

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

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

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

Las características básicas de **XML** son:

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

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

- ✓ Especificación de requisitos.
- ✓ Diseño de etiquetas.
- ✓ Marcado de los documentos.

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

Los datos carácter son los que forman la verdadera información del documento **XML**.

El marcado puede ser tan rico como se quiera. Puede ser interesante detectar necesidades futuras y crear documentos con una estructura fácilmente actualizables.

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

- ✔ Antes del prólogo.
- ✔ Dentro de una etiqueta.

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

## 3.1.- El prólogo.

---

Si se incluye, el prólogo **debe preceder al ejemplar del documento**. Su inclusión facilita el procesado de la información del ejemplar. El prólogo está dividido en dos partes:

- ✓ **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.

El prólogo puede tener tres funciones:

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

```
<?xml versión= "1.0" ?>
```

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

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

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

En este caso se usa el código **iso-8859-1 (Latin-1)** que permite el uso de acentos o caracteres como la ñ.

Los códigos más importantes son:

Estándar ISO	Código de país
--------------	----------------

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

➡ *Declaración de la autonomía del documento.*

Informa de si el documento necesita de otro para su interpretación. Para declararlo hay que definir el prólogo completo:

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

En este caso, el documento es independiente, de no ser así el atributo standalone hubiese tomado el valor "no".

- ✔ **La declaración del tipo de documento**, define qué tipo de documento estamos creando para ser procesado correctamente. Toda declaración de tipo de documento comienza por la cadena:

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

## 3.2.- El ejemplar. Los elementos.

---

Es la parte más importante de un documento **XML**, ya que **contiene los datos reales del documento**. Está formado por elementos anidados.

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

### Ejemplo

Dado el siguiente código XML...

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

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



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

Los nombres de las etiquetas han de ser autodescriptivos, lo que facilita el trabajo que se hace con ellas.

La formación de elementos ha de cumplir ciertas normas para que queden perfectamente definidos y que el documento XML al que pertenecen pueda ser interpretado por los procesadores **XML** sin generar ningún error fatal. Dichas reglas son:

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

Carácter	Cadena
>	&gt;
<	&lt;
&	&amp;
"	&quot;
'	&apos;

- ➡ 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 &#8364; o &#x20AC;

## Debes conocer

En el siguiente enlace encontrarás una tabla con los caracteres ASCII, el nombre HTML, y el número HTML de cada uno de ellos que te será imprescindible a la hora de realizar documentos en HTML y XML.

<http://ascii.cl/es/codigos-html.htm>

## 3.2.1.- 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.

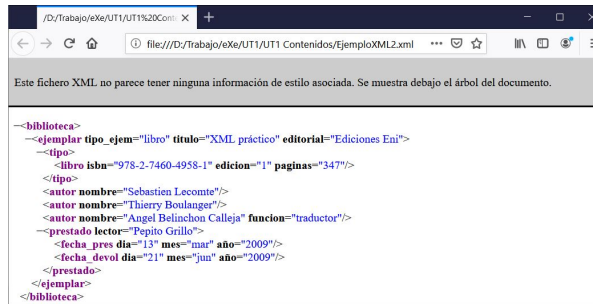
No se debe utilizar un atributo para contener información susceptible de ser dividida.

### Ejemplo

Dado el siguiente código XML

```
1 <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2 <!DOCTYPE biblioteca >
3 <biblioteca>
4     <ejemplar tipo_ejem="libro" titulo="XML práctico" editorial="Ediciones Eni">
5         <tipo> <libro isbn="978-2-7460-4958-1" edicion="1" paginas="347"></libro> </tipo>
6         <autor nombre="Sebastien Lecomte"></autor>
7         <autor nombre="Thierry Boulanger"></autor>
8         <autor nombre="Angel Belinchon Calleja" funcion="traductor"></autor>
9         <prestado lector="Pepito Grillo">
10             <fecha_pres dia="13" mes="mar" año="2009"></fecha_pres>
11             <fecha_devol dia="21" mes="jun" año="2009"></fecha_devol>
12         </prestado>
13     </ejemplar>
14 </biblioteca>
```

Al abrir el documento anterior con el navegador obtendremos:



```
<?xml version='1.0' encoding='UTF-8' />
<biblioteca>
  <ejemplar tipo='libro' titulo='XML práctico' editorial='Ediciones Eni'>
    <tipo>
      <libro isbn='978-2-7460-4958-1' edicion='1' paginas='347'>
      </tipo>
      <autor nombre='Sebastien Leconte' />
      <autor nombre='Thierry Boulanger' />
      <autor nombre='Angel Belinchon Calleja' funcion='traductor' />
    <prestado lector='Pepito Grillo'>
      <fecha_pres dia='13' mes='mar' año='2009' />
      <fecha_devol dia='21' mes='jun' año='2009' />
    </prestado>
  </ejemplar>
</biblioteca>
```

Rubén Carrasco Peña a partir de material del Ministerio ([CC BY-NC-SA](#))

Mostrar retroalimentación

Los nombres de los elementos aparecen en color morado, los atributos en negro y sus valores en azul.

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

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

## 3.2.2.- Ejercicio Detectar errores.

### Autoevaluación

#### 3.2.2 Ejercicio: Detectar errores

Indica cuál de los errores indicados aparecen en el siguiente documento XML.

```
1 <?XML version="1.0" encoding="UTF-8" standalone="yes" ?>
2 <!DOCTYPE biblioteca >
3 <biblioteca>
4   <ejemplar tipo_ejem='libro' titulo='XML práctico' editorial='Ediciones Eni'>
5     <tipo> <libro isbn='978-2-7460-4958-1' edicion= paginas='347'></libro> </tipo>
6     <autor nombre='Sebastien Lecomte'></autor>
7     <autor nombre='Thierry Boulanger'></autor>
8     <autor nombre='Angel Belinchon Calleja' funcion='traductor'></autor>
9     <prestado lector='Pepito Grillo'>
10       <fecha_pres dia='13' mes='mar' año='2009'></fecha_pres>
11       <fecha_devol/>
12     </prestado>
13   </ejemplar>
14 </biblioteca>
```

- ☐ Utiliza mayúsculas en la definición de la versión XML.

☐ No utiliza el código de caracteres adecuado.

☐ Los valores de los atributos no están entre comillas dobles.

☐ Hay algún atributo vacío.

☐ La etiqueta <fecha\_devol/> no se cierra.

☐ Se usan mayúsculas en los datos del documento.

Mostrar retroalimentación

## Solución

1. Incorrecto
2. Incorrecto

- 3. Correcto
- 4. Correcto
- 5. Incorrecto
- 6. Incorrecto

## 4.- Documentos XML bien formados.

### Introducción

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

### Autoevaluación

¿Está bien formado el siguiente documento XML?

```
1 <?xml version="1.0"?>
2 <mensaje>
3   <destinatario>Tomas</ destinatario>
4   <remitente>Juan</ remitente>
5   <asunto>
6   <contenido> No olvides ir a recogerme al aeropuerto mañana por la mañana!</contenido>
7 </mensaje>
```

☐ Verdadero ☐ Falso



**Falso**

No, la etiqueta <asunto> sigue abierta y el prólogo no tiene una declaración XML completa.

## 5.- Utilización de espacios de nombres en XML.

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

Los espacios de nombres, también conocidos como "name spaces", permiten dar un nombre único a cada elemento, indexándolos según el nombre del vocabulario adecuado. Además están asociados a un URI que los identifica de forma única.

En el documento, las etiquetas ambiguas se sustituyen por otras en las que el nombre del elemento está precedido de un prefijo, que determina el contexto al que pertenece la etiqueta, seguido de dos puntos, ":". Esto es:

**<prefijo:nombre\_etiqueta></prefijo:nombre\_etiqueta>**

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

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

**<conexion>://<direccionserver/><apartado1>/<apartado2>/...**

### Autoevaluación

#### Los espacios de nombres permiten...

- ☐ Utilizar etiquetas idénticas para estructurar distintos tipos de información de texto.
- ☐ Estructurar la información de un documento XML cuando proviene de varios documentos.
- ☐ Asignar varias etiquetas a una misma información.
- ☐ Definir etiquetas en otros documentos.

¡Correcta!

Respuesta errónea. Te recomiendo volver a leer el apartado.

Respuesta incorrecta. Te recomiendo volver a leer el apartado.

No es la respuesta correcta. Te recomiendo volver a leer el apartado.

## Solución

1. Opción correcta
2. Incorrecto
3. Incorrecto
4. Incorrecto

**Para saber más**

Los espacios de nombres tienen una recomendación en XML - <http://www.w3.org/TR/REC-xml-names/>

## 5.1.- Ejemplo: Utilización de espacios de nombres.

### Ejercicio Resuelto

Supongamos dos documentos que organizan la información sobre los profesores y los alumnos del Ciclo Formativo.

#### XML de alumnos:

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

#### XML de profesores

```
1 <?xml version="1.0" encoding="iso-8859-1" standalone="yes" ?>
2 <!DOCTYPE profesores>
3 <profesores>
4   <nombre>Pilar Ruiz Pérez</nombre>
5   <nombre>Tomás Rodríguez Hernández</nombre>
6 </profesores>
```

Si uniéramos los dos documentos en uno único, sin usar espacios de nombres, no se distinguirían los profesores de los alumnos ya que en los dos casos la etiqueta <nombre> se llama igual.

Para resolverlo necesitamos definir un espacio de nombres para cada contexto.

Mostrar retroalimentación

```
1 <?xml version="1.0" encoding="iso-8859-1" standalone="yes" ?>
2 <!DOCTYPE miembros>
3 <alumnos xmlns:alumnos="http://DAM/alumnos">
4 <profesores xmlns:profesores="http://DAM/profesores">
5 <asistentes>
6   <alumnos:nombre>Fernando Fernández González</alumnos:nombre>
7   <alumnos:nombre>Isabel González Fernández</alumnos:nombre>
8   <alumnos:nombre>Ricardo Martínez López</alumnos:nombre>
9   <profesores:nombre>Pilar Ruiz Pérez</profesores:nombre>
10  <profesores:nombre>Tomás Rodríguez Hernández</profesores:nombre>
11 </asistentes>
```

## 6.- Para saber más.

---

### Para saber más

#### Tutoriales del w3shools.com

- ✓ [Tutorial XML](#)
- ✓ [Validador en línea de documentos bien formados](#)
- ✓ [Documentación Web de Mozilla Developer Network](#)