

Lenguajes de Marcas y Sistemas de Gestión de Información (LMSGI)



{ Carlos Pes }

Libro del Tutorial de LMSGI
www.abrirllave.com/lmsgi/



Primera edición, febrero 2017.

Todos los contenidos de este documento forman parte del [Tutorial de LMSGI](#) de [Abrirllave](#) y están bajo la Licencia Creative Commons Reconocimiento 4.0 Internacional ([CC BY 4.0](#)).

- Véase: creativecommons.org/licenses/by/4.0/deed.es ES

Por tanto:

Usted es libre de:

- **Compartir** — copiar y redistribuir el material en cualquier medio o formato
- **Adaptar** — remezclar, transformar y crear a partir del material

para cualquier finalidad, incluso comercial.

El licenciador no puede revocar estas libertades mientras cumpla con los términos de la licencia.

Bajo las siguientes condiciones:

- **Reconocimiento** — Debe reconocer adecuadamente la autoría, proporcionar un enlace a la licencia e indicar si se han realizado cambios. Puede hacerlo de cualquier manera razonable, pero no de una manera que sugiera que tiene el apoyo del licenciador o lo recibe por el uso que hace.

No hay restricciones adicionales — No puede aplicar términos legales o medidas tecnológicas que legalmente restrinjan realizar aquello que la licencia permite.

Dedicado a todos los profesores y profesoras de informática.

ÍNDICE DEL CONTENIDO

Prefacio	8
-----------------------	----------

Capítulo 1. Reconocimiento de las características de los lenguajes de marcas	10
---	-----------

1.1. XML	11
1.1.1. Qué es XML.....	12
1.1.2. Elementos	12
1.1.3 Normas de sintaxis básicas.....	14
1.1.4. Atributos	16
1.1.5. Declaración XML.....	17
1.1.6. Instrucciones de procesamiento	19
1.1.7. Referencias a entidades	20
1.1.8. Referencias de caracteres	23
1.1.9. Comentarios	24
1.1.10. Secciones CDATA	25
1.1.11. Espacios de nombres.....	26
1.1.12. Espacios en blanco	30
1.1.13. Documentos XML bien formados y válidos	34
1.1.14. Recursos de XML	34

Capítulo 2. Utilización de lenguajes de marcas en entornos web ...	35
---	-----------

2.1. HTML.....	36
2.1.1. Qué es HTML	38
2.1.2. Primer documento HTML.....	38
2.1.3. Comentarios	44
2.1.4. Estilos.....	45

2.1.5. Párrafos	49
2.1.6. Formato de texto.....	55
2.1.7. Cabeceras	61
2.1.8. Enlaces.....	63
2.1.9. Listas	65
2.1.10. Imágenes	68
2.1.11. Tablas.....	70
2.1.12. Divisiones.....	82
2.1.13. Formularios	86
2.1.14. Recursos de HTML.....	95

Capítulo 3. Aplicación de los lenguajes de marcas a la sindicación de contenidos 96

Capítulo 4. Definición de esquemas y vocabularios en XML 97

4.1. DTD	98
4.1.1. Qué es DTD	100
4.1.2. Declaración de tipo de documento	100
4.1.2.1. Documento XML asociado a una DTD interna.....	100
4.1.2.2. Documento XML asociado a una DTD externa	102
4.1.2.3. Cuándo utilizar una DTD interna o una DTD externa	104
4.1.2.4. Uso combinado de DTD interna y externa en un documento XML.....	104
4.1.3. Estructura de un documento XML	105
4.1.4. Declaración de elementos.....	105
4.1.4.1. Elementos vacíos - EMPTY	107
4.1.4.2. Elementos con cualquier contenido - ANY	108
4.1.4.3. Elementos con contenido de tipo texto - (#PCDATA)	110
4.1.4.4. Secuencias de elementos	111
4.1.4.5. Cardinalidad de los elementos	112

4.1.4.6. Elementos opcionales	114
4.1.5. Declaración de atributos	117
4.1.6. Tipos de declaración de atributos	119
4.1.6.1. Atributo obligatorio - #REQUIRED	120
4.1.6.2. Atributo opcional - #IMPLIED	121
4.1.6.3. Atributo con valor fijo - #FIXED valor	121
4.1.7. Tipos de atributos.....	122
4.1.7.1. Atributos de tipo CDATA	123
4.1.7.2. Atributos de tipo enumerado	123
4.1.7.3. Atributos de tipo ID	124
4.1.7.4. Atributos de tipo IDREF	125
4.1.7.5. Atributos de tipo IDREFS	126
4.1.7.6. Atributos de tipo NMTOKEN	127
4.1.7.7. Atributos de tipo NMTOKENS	127
4.1.7.8. Atributos de tipo NOTATION	128
4.1.7.9. Atributos de tipo ENTITY	129
4.1.7.10. Atributos de tipo ENTITIES	130
4.1.7.11. Atributos especiales.....	130
4.1.8. Declaración de entidades	133
4.1.8.1. Entidades generales internas analizables.....	133
4.1.8.2. Entidades generales externas analizables	134
4.1.8.3. Entidades generales externas no analizables	136
4.1.8.4. Entidades paramétricas internas analizables	137
4.1.8.5. Entidades paramétricas externas analizables.....	139
4.1.8.6. Uso de una entidad dentro de otra	141
4.1.9. Declaración de notaciones	142
4.1.10. Secciones condicionales	145

4.1.11. Espacios de nombres.....	146
4.1.12. Comentarios	148
4.1.13. Recursos de DTD.....	148
4.2. XSD (XML Schema).....	149
4.2.1. Qué es XSD	150
4.2.2. Validación de un documento XML con XSD	150
4.2.3. Elementos simples.....	156
4.2.4. Atributos.....	157
4.2.5. Restricciones (facetas).....	158
4.2.6. Extensiones.....	161
4.2.7. Elementos complejos	164
4.2.8. Indicadores	165
4.2.9. Recursos de XSD	169
Capítulo 5. Conversión y adaptación de documentos XML.....	170
Capítulo 6. Almacenamiento de información	171
Capítulo 7. Sistemas de gestión empresarial	172
Epílogo.....	173

PREFACIO

Objetivo del libro

El objetivo principal de este libro es servir de guía –y proporcionar material de apoyo– a los contenidos del módulo **“Lenguajes de Marcas y Sistemas de Gestión de Información” (LMSGI)** de los Ciclos Formativos de Grado Superior de Informática:

- Administración de Sistemas Informáticos en Red (ASIR).
- Desarrollo de Aplicaciones Multiplataforma (DAM).
- Desarrollo de Aplicaciones Web (DAW).

Contenidos del libro

Los contenidos de los capítulos de este libro están basados en algunos de los tutoriales de Abrirllave (XML, HTML, XSD...) referenciados desde los apartados del tutorial de LMSGI. Seguidamente, se indica a qué apartados del tutorial de LMSGI corresponden los contenidos de cada capítulo del libro:

- Capítulo 1: [Reconocimiento de las características de los lenguajes de marcas](#)
- Capítulo 2: [Utilización de lenguajes de marcas en entornos web](#)
- Capítulo 3: [Aplicación de los lenguajes de marcas a la sindicación de contenidos](#)
- Capítulo 4: [Definición de esquemas y vocabularios en XML](#)
- Capítulo 5: [Conversión y adaptación de documentos XML](#)
- Capítulo 6: [Almacenamiento de información](#)
- Capítulo 7: [Sistemas de gestión empresarial](#)

Ahora bien, en esta primera edición del libro, solamente se han publicado los contenidos de los capítulos 1, 2 y 4. El resto de capítulos están en desarrollo y se añadirán al libro en una próxima edición.

Material extra

En la web del tutorial de LMSGI “www.abrirllave.com/lmsgi/” se proporcionan más recursos: actividades prácticas, ejercicios resueltos, etc.

Erratas

Para comunicar cualquier comentario, sugerencia o error detectado en el texto, puede hacerlo escribiendo un correo electrónico a:

correo@abrirllave.com

Todas las sugerencias serán atendidas lo antes posible. Gracias de antemano por colaborar en la mejora del contenido de este libro.

Agradecimientos

Este es el cuarto libro que publico, y quiero dar las gracias a todos aquellos que, en algún momento de la vida, me han inspirado o motivado –en mayor o menor medida– para escribir libros o contenidos educativos de informática en la Web; desde 2006 en www.carlospes.com y desde 2014 en www.abrirllave.com.

Gran parte de mi entusiasmo es alimentado por esas personas –familiares, amigos e incluso desconocidos– que tanto me inspiran y motivan. ¡Gracias a todos!

Carlos Pes
Pamplona, febrero de 2017.

Twitter: [@CarlosPes](https://twitter.com/CarlosPes)
Blog: <http://carlospes.blogspot.com>

Capítulo 1

Reconocimiento de las características de los lenguajes de marcas

Teoría y ejemplos

- *Apuntes de XML*
<http://www.abrirllave.com/xml/apuntes-de-xml.php>
- *Presentación PDF*
<http://www.abrirllave.com/xml/presentacion.php>

Ejercicios

- *Cómo comprobar si un documento XML está bien formado (con XML Copy Editor)*
<http://www.abrirllave.com/xml/como-comprobar-si-un-documento-xml-esta-bien-formado.php>
 - *Ejercicios resueltos de XML*
<http://www.abrirllave.com/xml/ejercicios-resueltos.php>
-

1.1. XML

Contenidos del tutorial de XML www.abrirllave.com/xml/

[1.1.1. Qué es XML](#)

[1.1.2. Elementos](#)

- Elementos vacíos
- Relaciones padre-hijo entre elementos
- Elemento raíz de un documento XML
- Elementos con contenido mixto

[1.1.3 Normas de sintaxis básicas](#)

[1.1.4. Atributos](#)

- Normas de sintaxis

[1.1.5. Declaración XML](#)

- Atributos **version** y **encoding**
- Cómo crear un documento XML
- Atributo **standalone**

[1.1.6. Instrucciones de procesamiento](#)

- Cómo asociar un archivo CSS a un documento XML

[1.1.7. Referencias a entidades](#)

- Caracteres problemáticos en XML: menor que (<) y ampersand (&)
- Uso de la comilla doble (") y de la comilla simple (') en atributos

[1.1.8. Referencias de caracteres](#)

- Representación del carácter Euro (€) en XML

[1.1.9. Comentarios](#)

[1.1.10. Secciones CDATA](#)

[1.1.11. Espacios de nombres](#)

- Uso de espacios de nombres
- Sintaxis para definir un espacio de nombres
- Definición de espacios de nombres en elementos distintos al raíz
- Definición de un espacio de nombres por defecto
- Cómo indicar que un elemento no pertenece a ningún espacio de nombres

[1.1.12. Espacios en blanco](#)

- Espacios en blanco en el contenido (texto) de un elemento
- Espacios en blanco en atributos
- Espacios en blanco entre elementos
- Uso del atributo **xml:space**

[1.1.13. Documentos XML bien formados y válidos](#)

[1.1.14. Recursos de XML](#)

1.1.1. Qué es XML

XML (*eXtensible Markup Language*, Lenguaje de Marcado eXtensible) es un lenguaje desarrollado por **W3C** (*World Wide Web Consortium*) que está basado en **SGML** (*Standard Generalized Markup Language*, Lenguaje de Marcado Generalizado Estándar).

XML es un lenguaje utilizado para el almacenamiento e intercambio de datos estructurados entre distintas plataformas.

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

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

1.1.2. Elementos

Los documentos XML están formados por **texto plano (sin formato)** y contienen **marcas (etiquetas)** definidas por el desarrollador. Dichas marcas, es recomendable que sean lo más descriptivas posible y, para escribirlas, se utilizan los caracteres *menor que* "<", *mayor que* ">" y *barra inclinada* "/".

EJEMPLO Si en un documento XML se quiere guardar el nombre **Elsa**, se puede escribir:

```
<nombre>Elsa</nombre>
```

La sintaxis utilizada en el ejemplo es la básica para escribir un elemento en XML:

```
<etiqueta>valor</etiqueta>
```

Obsérvese que, entre la etiqueta de inicio (<nombre>) y la etiqueta de fin (</nombre>) se ha escrito el dato (**valor**) que se quiere almacenar. En este caso **Elsa**.

Elementos vacíos

En un documento XML, **un elemento puede no contener ningún valor**. En tal caso hay que escribir:

```
<etiqueta></etiqueta>
```

Se puede expresar lo mismo escribiendo:

```
<etiqueta/>
```

EJEMPLO Para escribir el elemento “nombre” vacío, se puede escribir:

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

O también:

```
<nombre/>
```

Relaciones padre-hijo entre elementos

EJEMPLO Por otra parte, un elemento (*padre*) puede contener a otro u otros elementos (*hijos*):

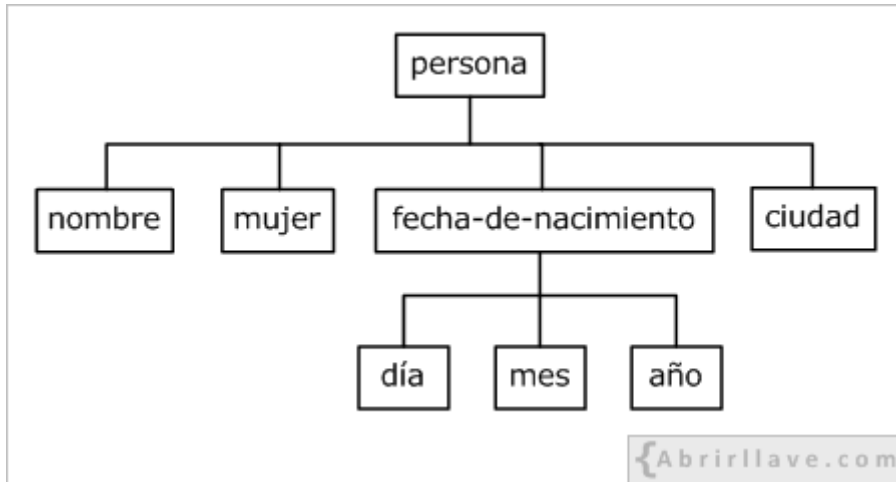
```
<persona>
  <nombre>Elsa</nombre>
  <mujer/>
  <fecha-de-nacimiento>
    <día>18</día>
    <mes>6</mes>
    <año>1996</año>
  </fecha-de-nacimiento>
  <ciudad>Pamplona</ciudad>
</persona>
```

En este ejemplo, el elemento “persona” contiene cuatro elementos (*hijos*): “nombre”, “mujer”, “fecha de nacimiento” y “ciudad”. A su vez, el elemento “fecha de nacimiento” contiene otros tres elementos (*hijos*): “día”, “mes” y “año”.

Véase que, de todos los elementos que aparecen en este ejemplo, sólo el elemento “mujer” está vacío.

Elemento raíz de un documento XML

Todo documento XML tiene que tener un único elemento raíz (*padre*) del que desciendan todos los demás. En este caso, el elemento raíz es “persona”. Gráficamente, la estructura de elementos de este documento se puede representar como se muestra a continuación:



De esta forma, la estructura de cualquier documento XML se puede representar como un **árbol invertido de elementos**. Se dice que los elementos son los que dan estructura semántica al documento.

Elementos con contenido mixto

EJEMPLO Un elemento puede contener contenido mixto, es decir, texto y otros elementos:

```

<persona>
  <nombre>Elsa</nombre> vive en <ciudad>Pamplona</ciudad>.
</persona>
  
```

En este ejemplo, el elemento "persona" contiene los elementos "nombre" y "ciudad", además de los textos " **vive en** " y " **.**".

1.1.3. Normas de sintaxis básicas

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

- Pueden contener letras minúsculas, letras mayúsculas, números, *puntos* ".", *guiones medios* "-" y *guiones bajos* "_".
- Asimismo, pueden contener el carácter *dos puntos* ":". No obstante, su uso se reserva para cuando se definan espacios de nombres.
- El primer carácter tiene que ser una letra o un *guion bajo* "_".

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

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

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

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

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

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

En su lugar, sería correcto escribir:

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

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

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

1.1.4. Atributos

Los elementos de un documento XML pueden tener atributos definidos en la etiqueta de inicio. Un atributo sirve para proporcionar información extra sobre el elemento que lo contiene.

EJEMPLO Dados los siguientes datos de un producto:

- Código: **G45**
- Nombre: **Gorro de lana**
- Color: **negro**
- Precio: **12.56**

Su representación en un documento XML podría ser, por ejemplo:

```
<producto codigo="G45">
  <nombre color="negro" precio="12.56">Gorro de lana</nombre>
</producto>
```

En este ejemplo se han escrito tres atributos: **codigo**, **color** y **precio**. Obsérvese que, sus valores ("G45", "negro" y "12.56") se han escrito entre *comillas dobles* ("). No obstante, también pueden ir entre *comillas simples* (').

Si, por ejemplo, el atributo **codigo** se quisiera representar como un elemento, se podría escribir:

```
<producto>
  <codigo>G45</codigo>
  <nombre color="negro" precio="12.56">Gorro de lana</nombre>
</producto>
```

Como se puede apreciar, ahora el valor del código no se ha escrito entre comillas dobles.

Normas de sintaxis

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

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

Sin embargo, sí es correcto escribir:

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

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

1.1.5. Declaración XML

La declaración XML que se puede escribir al principio de un documento XML, empieza con los caracteres "<?" y termina con "?>" al igual que las instrucciones de procesamiento. Sin embargo, **la declaración XML no es una instrucción de procesamiento (o proceso)**.

Atributos **version** y **encoding**

EJEMPLO Un documento XML podría contener la siguiente declaración XML:

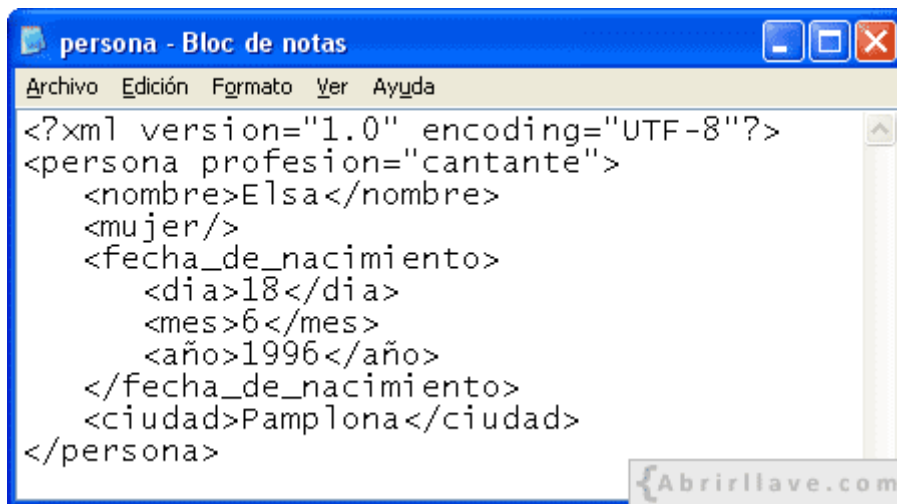
```
<?xml version="1.0" encoding="UTF-8"?>
```

En esta declaración XML, se está indicando que **1.0** es la versión de XML utilizada en el documento y **UTF-8** (*8-bit Unicode Transformation Format*, Formato de Transformación Unicode de 8 bits) es la codificación de caracteres empleada.

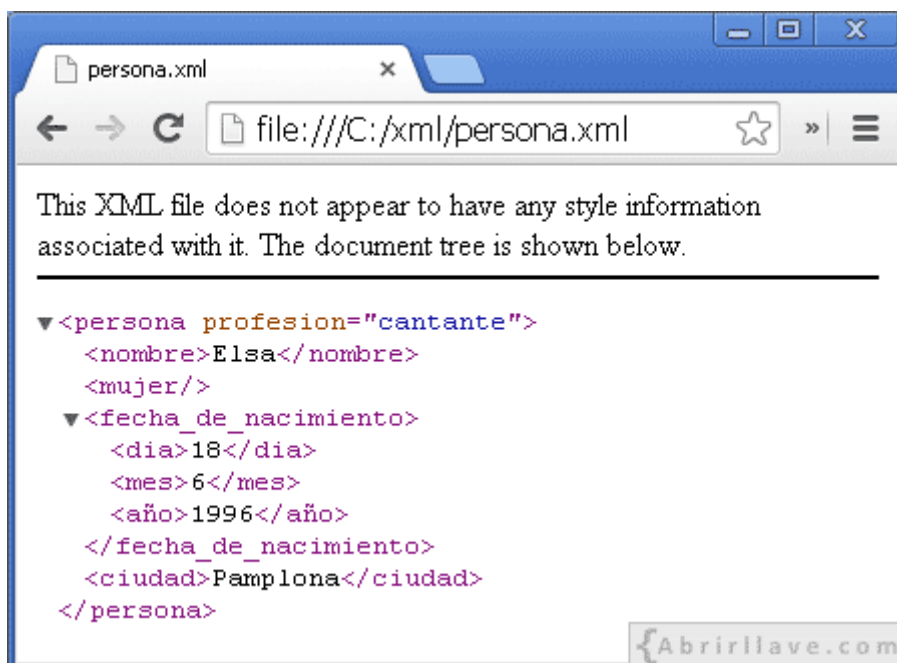
En un documento XML no es obligatorio que aparezca la declaración XML. Ahora bien, si se incluye, tiene que aparecer en la primera línea del documento, y el carácter "<" debe ser el primero de dicha línea, es decir, antes no pueden aparecer espacios en blanco.

Cómo crear un documento XML

EJEMPLO Si, por ejemplo, en el *Bloc de notas* de *Microsoft Windows* escribimos y guardamos (codificado en UTF-8) un archivo llamado **"persona.xml"** como se muestra en la siguiente imagen:



Al visualizar dicho archivo en un navegador web, como por ejemplo *Google Chrome*, se podrá ver algo parecido a:



Como se puede ver, a la izquierda de los elementos que tienen hijos, en este caso **persona** y **fecha_de_nacimiento**, aparece un pequeño triángulo. Por otra parte, el elemento **persona** es el único que tiene un atributo.

Atributo **standalone**

EJEMPLO En una declaración XML, además de los atributos **version** y **encoding**, también se puede escribir el atributo **standalone**, que puede tomar dos valores ("**yes**" o "**no**");

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
```

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

En un documento XML, escribir la declaración XML es opcional. Pero, si se escribe, el atributo **version** es obligatorio indicarlo. Sin embargo, los atributos **encoding** y **standalone** son opcionales y, por defecto, sus valores son **"UTF-8"** y **"no"**, respectivamente.

Por otra parte, cuando se escriba el atributo **encoding**, siempre deberá aparecer después de **version**. Y, respecto al atributo **standalone**, siempre que se escriba, deberá ser en último lugar.

Ejercicios resueltos

- [Ciudades de países](#)
- [Hechos históricos](#)

1.1.6. Instrucciones de procesamiento

En un documento XML, una **instrucción de procesamiento** (*processing instruction*) sirve para indicar cierta información al programa que procese dicho documento. Las instrucciones de proceso se escriben empezando con la pareja de caracteres "**<?**" y finalizando con **">"**.

Cómo asociar un archivo CSS a un documento XML

EJEMPLO En un documento XML podría escribirse, por ejemplo, la siguiente instrucción de procesamiento:

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

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

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

De forma que, dado por ejemplo el archivo **"animales.xml"**:

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

En un navegador web se verá algo parecido a:



En un documento XML, no es obligatorio que aparezcan instrucciones de procesamiento.

Ejercicios resueltos

- [Artículos](#)
- [Fechas de un año](#)
- [Mezclas de colores](#)

1.1.7. Referencias a entidades

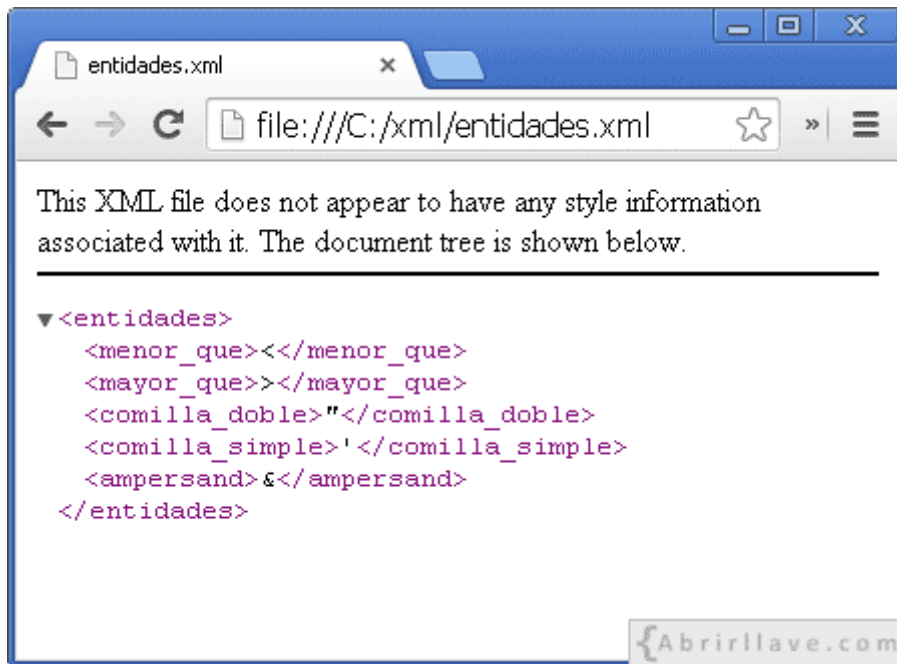
En XML existen algunos caracteres que son especiales por su significado y, para escribirlos en un documento XML, se pueden utilizar las referencias a entidades mostradas en la siguiente tabla:

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

EJEMPLO Dado el archivo "*entidades.xml*":

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

Al abrirlo en *Google Chrome* se podrá visualizar:



En el navegador web, se puede ver que donde se han escrito las referencias a entidades en el documento XML (por ejemplo `<`), se muestran los caracteres correspondientes (por ejemplo `<`).

Caracteres problemáticos en XML: menor que (<) y ampersand (&)

EJEMPLO En un documento XML, el carácter "<" es problemático porque indica el comienzo de una etiqueta. Por tanto, en vez de escribir, por ejemplo:

```
<condicion>a<b</condicion>
```

Habría que utilizar la referencia a entidad `<`; escribiendo:

```
<condicion>a&lt;b</condicion>
```

EJEMPLO El carácter ">" sí puede utilizarse en el texto contenido en un elemento, no siendo incorrecto escribir, por ejemplo:

```
<condicion>a>b</condicion>
```

Ahora bien, se recomienda hacer uso de su referencia a entidad (>).

EJEMPLO En un documento XML, el carácter ampersand "&" también es problemático, ya que se utiliza para indicar el comienzo de una referencia a entidad. Por ejemplo, no es correcto escribir:

```
<condicion>a=1 && b=2</condicion>
```

En su lugar se debe escribir lo siguiente:

```
<condicion>a=1 &amp;&amp; b=2</condicion>
```

Uso de la comilla doble (") y de la comilla simple (') en atributos

EJEMPLO Si el valor de un atributo se escribe entre comillas dobles ("), dicho valor no podrá contener dicho carácter. Por ejemplo, no es correcto escribir:

```
<dato caracter="comilla doble(")"/>
```

Para ello, hay que utilizar la referencia a entidad " como se muestra a continuación:

```
<dato caracter="comilla doble(&quot;)" />
```

De igual modo ocurre con la comilla simple ('), siendo incorrecto escribir, por ejemplo:

```
<dato caracter='comilla simple(')'/>
```

Por lo que, en este caso, habría que usar ' como se muestra seguidamente:

```
<dato caracter='comilla simple(&apos;)' />
```

Por otro lado, los valores de atributos escritos entre comillas dobles (") sí pueden contener al carácter comilla simple (') y a la inversa. Por ejemplo, es correcto escribir:

```
<dato caracter="comilla simple(')"/>
<dato caracter='comilla doble(")'/>
```

En estos casos, no es obligatorio usar las referencias a entidades, pero sí recomendable.

1.1.8. Referencias de caracteres

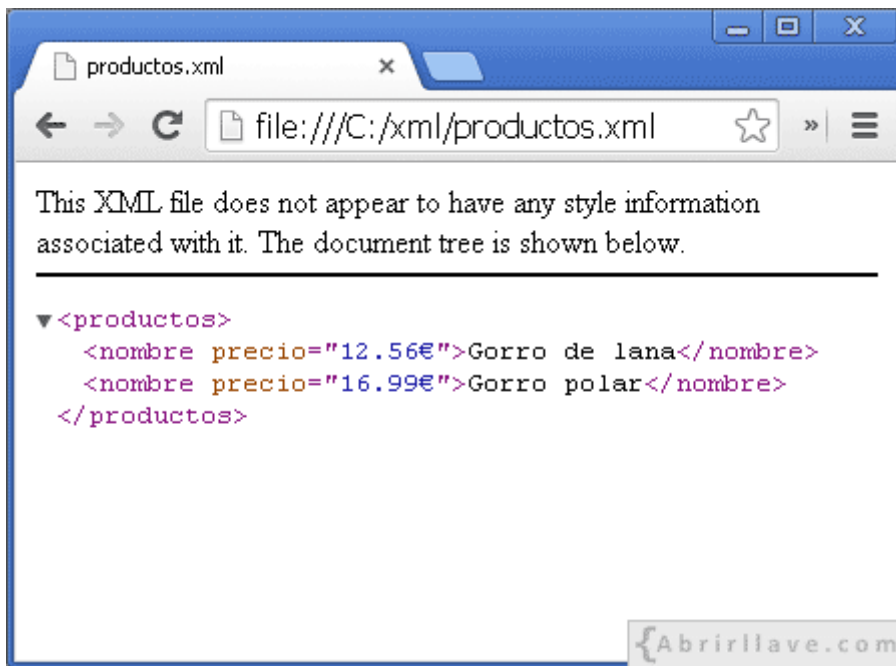
En un documento XML se pueden escribir referencias de caracteres Unicode con los símbolos **&#**, seguidos del valor decimal o hexadecimal del carácter Unicode que se quiera representar y, finalmente, añadiendo el carácter *punto y coma* " ; ".

Representación del carácter Euro (€) en XML

EJEMPLO Dado el documento XML "*productos.xml*":

```
<?xml version="1.0" encoding="UTF-8"?>
<productos>
  <nombre precio="12.56&#8364;">Gorro de lana</nombre>
  <nombre precio="16.99&#x20AC;">Gorro polar</nombre>
</productos>
```

Al visualizarlo en un navegador web, se podrá ver lo siguiente:



Obsérvese que, en este caso, para representar al símbolo del Euro (€), la primera vez se ha utilizado su valor decimal (**€**) en Unicode y, la segunda vez, su valor hexadecimal (**€**).

1.1.9. Comentarios

Para escribir comentarios en un documento XML, estos deben escribirse entre los caracteres "`<!--`" y "`-->`". Por ejemplo:

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

EJEMPLO Dado el archivo XML "*letras.xml*":

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Ejemplo uso de comentarios.-->
<a>
  <b>
    <c cantidad="4">cccc</c>
    <d cantidad="2">dd</d>
  </b>
  <e>
    <f cantidad="8">ffffffff</f>
    <!--g puede aparecer varias veces.-->
    <g cantidad="5">ggggg</g>
    <g cantidad="2">gg</g>
  </e>
</a>
```

En un navegador web se verá:



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

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

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

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

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

1.1.10. Secciones CDATA

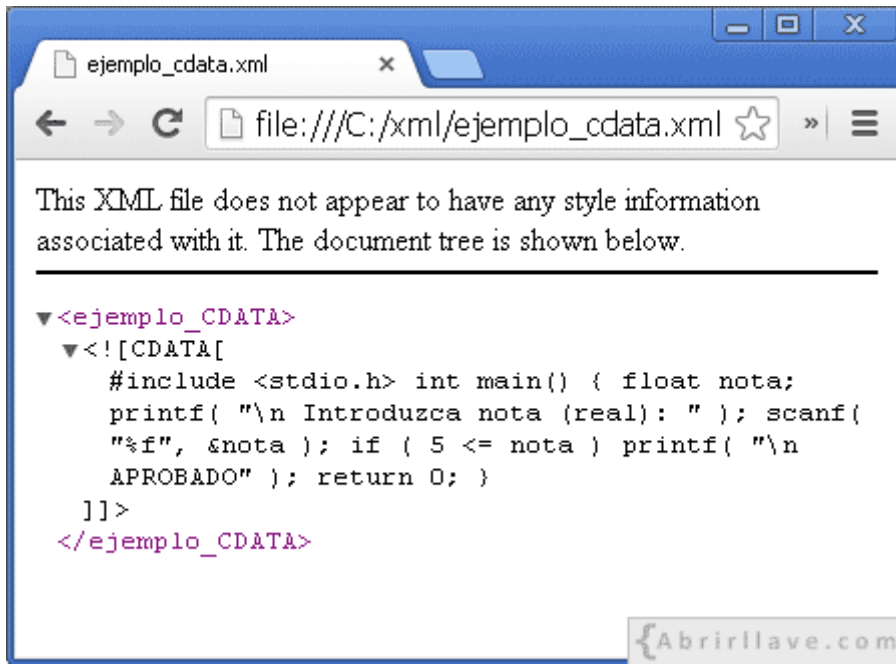
Un documento XML puede contener secciones **CDATA** (*Character DATA*) para escribir texto que no se desea que sea analizado. Por ejemplo, esto puede ser útil cuando se quiere escribir texto que contenga alguno de los caracteres problemáticos: *menor que* "<" o *ampersand* "&".

En un documento XML, para incluir una sección CDATA, esta se escribe comenzando con la cadena de caracteres "<![CDATA[" y terminando con los caracteres "]]>".

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

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

En un navegador web se visualizará algo parecido a:



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

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

1.1.11. Espacios de nombres

Varios documentos XML se pueden combinar entre sí, pudiendo en estos casos coincidir el nombre de algunos elementos.

EJEMPLO Dos documentos XML podrían contener un elemento llamado “carta”, pero con significados distintos.

```
<carta>
  <palo>Corazones</palo>
  <numero>7</numero>
</carta>
```

```
<carta>
  <carnes>
    <filete_de_tenera precio="12.95"/>
    <solomillo_a_la_pimienta precio="13.60"/>
  </carnes>
  <pescados>
    <lenguado_al_horno precio="16.20"/>
    <merluza_en_salsa_verde precio="15.85"/>
  </pescados>
</carta>
```

Uso de espacios de nombres

De forma que, si se incluyen ambos elementos **<carta>** en un documento XML, se origina un conflicto de nombres. Para resolverlo, se pueden utilizar espacios de nombres (*XML Namespaces*). Por ejemplo, escribiendo:

```
<?xml version="1.0" encoding="UTF-8"?>
<e1:ejemplo xmlns:e1="http://www.abrirllave.com/ejemplo1"
  xmlns:e2="http://www.abrirllave.com/ejemplo2">

  <e1:carta>
    <e1:palo>Corazones</e1:palo>
    <e1:numero>7</e1:numero>
  </e1:carta>

  <e2:carta>
    <e2:carnes>
      <e2:filete_de_tenera precio="12.95"/>
      <e2:solomillo_a_la_pimienta precio="13.60"/>
    </e2:carnes>
    <e2:pescados>
      <e2:lenguado_al_horno precio="16.20"/>
      <e2:merluza_en_salsa_verde precio="15.85"/>
    </e2:pescados>
  </e2:carta>
</e1:ejemplo>
```

Sintaxis para definir un espacio de nombres

Para definir un espacio de nombres se utiliza la siguiente sintaxis:

```
xmlns:prefijo="URI"
```

En el ejemplo, obsérvese que, **xmlns** es un atributo que se ha utilizado en la etiqueta de inicio del elemento **<ejemplo>** y, en este caso, se han definido dos espacios de nombres que hacen referencia a los siguientes **URI** (*Uniform Resource Identifier*, Identificador Uniforme de Recurso):

- **http://www.abrirllave.com/ejemplo1**
- **http://www.abrirllave.com/ejemplo2**

Los prefijos definidos son **e1** y **e2**, respectivamente. Véase que, se han añadido dichos prefijos a las etiquetas que aparecen en el documento: **<e1:carta>**, **<e2:carta>**, **<e1:palo>**, etc.

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

- <http://www.w3.org/1999/xhtml/>
- <http://www.w3.org/1999/XSL/Transform>
- <http://www.w3.org/2000/svg>

Definición de espacios de nombres en elementos distintos al raíz

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

```
<?xml version="1.0" encoding="UTF-8"?>
<el:ejemplo xmlns:el="http://www.abrirllave.com/ejemplo1">

  <el:carta>
    <el:palo>Corazones</el:palo>
    <el:numero>7</el:numero>
  </el:carta>

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

</el:ejemplo>
```

En un documento XML es posible definir todos los espacios de nombres que se necesiten, pudiéndose mezclar –si fuese necesario– los elementos de dichos espacios de nombres.

Definición de un espacio de nombres por defecto

EJEMPLO Por otra parte, se puede definir un espacio de nombres por defecto mediante la siguiente sintaxis:

```
xmlns="URI"
```

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

```
<?xml version="1.0" encoding="UTF-8"?>
<ejemplo xmlns="http://www.abrirllave.com/ejemplo1">

  <carta>
    <palo>Corazones</palo>
    <numero>7</numero>
  </carta>

</ejemplo>
```

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

- **<carnes>**
- **<pescados>**
- **<filete_de_tenera>**
- ...

```
<?xml version="1.0" encoding="UTF-8"?>
<ejemplo xmlns="http://www.abrirllave.com/ejemplo1">

  <carta>
    <palo>Corazones</palo>
    <numero>7</numero>
  </carta>

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

</ejemplo>
```

Cómo indicar que un elemento no pertenece a ningún espacio de nombres

EJEMPLO En un documento XML, para indicar que determinados elementos –o todos– no pertenecen a ningún espacio de nombres, se escribe el atributo **xmlns** vacío, es decir, **xmlns=""**.

```
<?xml version="1.0" encoding="UTF-8"?>
<ejemplo xmlns="http://www.abrirllave.com/ejemplo1">

  <carta>
    <palo>Corazones</palo>
    <numero>7</numero>
  </carta>

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

</ejemplo>
```

En este caso, el elemento **<pescados>** y sus hijos, no pertenecen a ningún espacio de nombres.

1.1.12. Espacios en blanco

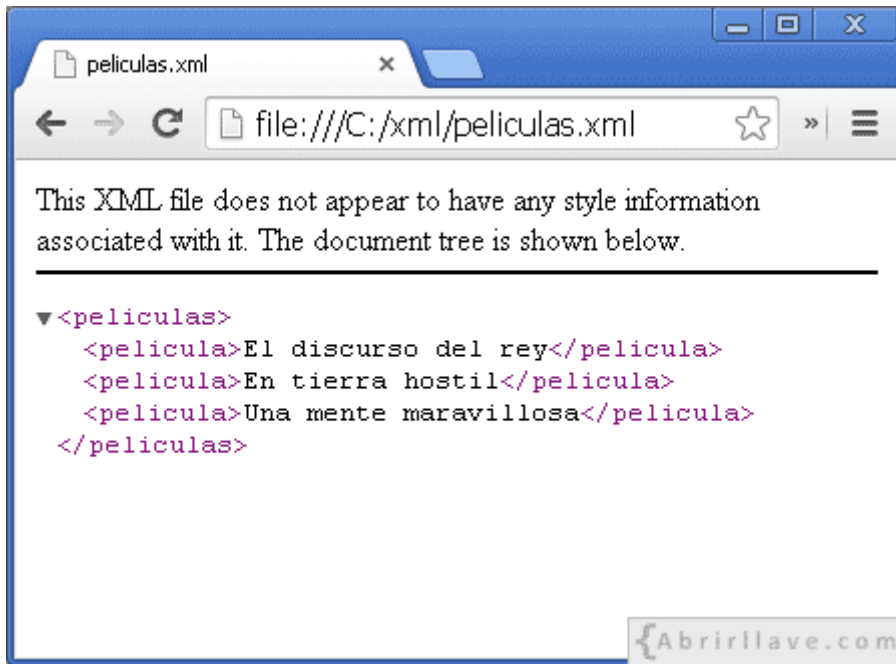
En un documento XML, los espacios en blanco, las tabulaciones y los retornos de carro pueden ser tratados de un modo especial.

Espacios en blanco en el contenido (texto) de un elemento

EJEMPLO Dado el archivo XML *"películas.xml"*:

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

Al visualizar dicho archivo en *Google Chrome*, se verá algo parecido a:



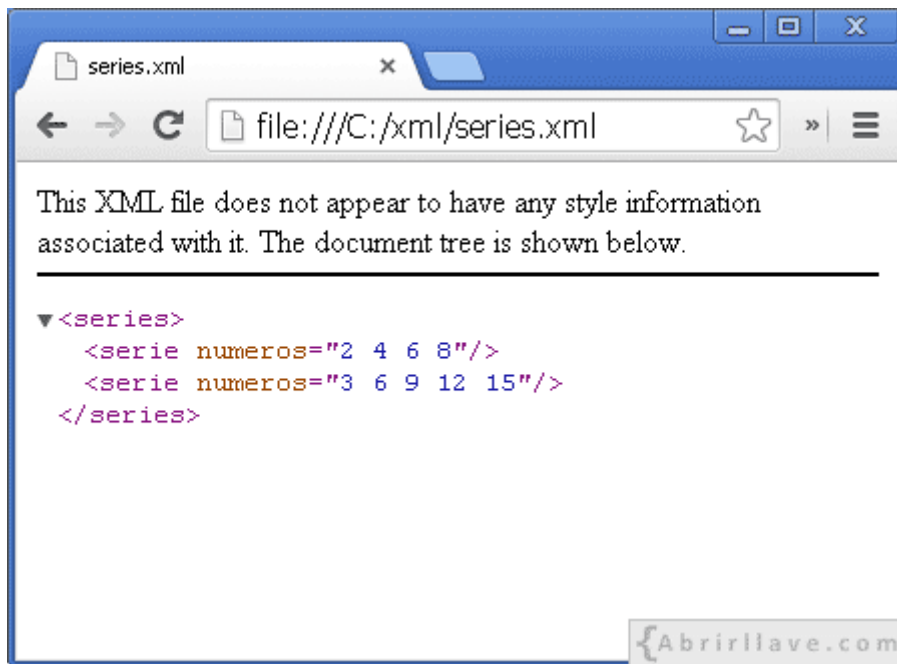
Esto es debido a que, las tabulaciones, los retornos de carro y varios espacios en blanco contenidos en el texto de los elementos del documento, han sido representados como un único espacio en blanco.

Espacios en blanco en atributos

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

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

En un navegador web se podrá visualizar:



Espacios en blanco entre elementos

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

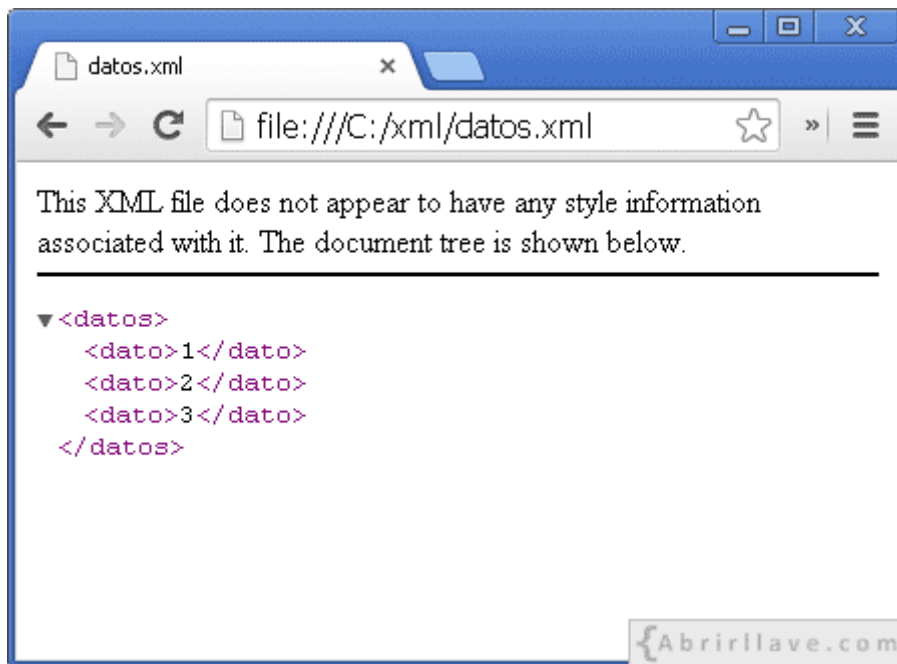
```
<?xml version="1.0" encoding="UTF-8"?>
<datos>
  <dato>1</dato>
  <dato>2</dato>
  <dato>3</dato>
</datos>
```

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

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

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

En todos los casos, en *Google Chrome* veremos:



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

Uso del atributo **xml:space**

EJEMPLO Por otra parte, los elementos de un documento XML pueden contener el atributo predefinido **xml:space** con el valor "**preserve**" para indicar que los espacios que aparecen en el contenido (texto) de dicho elemento, y sus sucesores, deben ser preservados:

```
<clasificacion xml:space="preserve">
1      Fernando Alonso      1:55.341
2      Lewis Hamilton       1:55.729
3      Sebastian Vettel     1:56.122
</clasificacion>
```

Los únicos valores que admite el atributo **xml:space** son "**preserve**" y "**default**", siendo este último su valor por defecto cuando no se escribe dicho atributo.

El valor "**default**" indica que la aplicación que haga uso del documento XML es la encargada de decidir cómo tratar los espacios en blanco. Ahora bien, aún indicando el valor "**preserve**", hay que tener en cuenta que no todos los programas que hacen uso de documentos XML reconocen este atributo.

1.1.13. Documentos XML bien formados y válidos

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

- Los nombres de los elementos y sus atributos deben estar escritos correctamente.
- Los valores de los atributos deben estar escritos entre comillas dobles o simples.
- Los atributos de un elemento deben separarse con espacios en blanco.
- Se tienen que utilizar referencias a entidades donde sea necesario.
- Tiene que existir un único elemento raíz.
- Todo elemento debe tener un elemento padre, excepto el elemento raíz.
- Todos los elementos deben tener una etiqueta de apertura y otra de cierre.
- Las etiquetas deben estar correctamente anidadas.
- Las instrucciones de proceso deben estar escritas de forma correcta.
- La declaración XML debe estar en la primera línea escrita correctamente.
- Las secciones CDATA y los comentarios deben estar correctamente escritos.

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

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

Ejercicios resueltos
<ul style="list-style-type: none"> • Lista de marcadores de páginas web • Equipos de fútbol

1.1.14. Recursos de XML

- **Tutorial de XML**
<http://www.abrirllave.com/xml/>
- **Chuleta de XML**
<http://www.abrirllave.com/xml/chuleta-de-xml.php>
- **Ejercicios resueltos de XML**
<http://www.abrirllave.com/xml/ejercicios-resueltos.php>
- **Test de autoevaluación de XML**
<http://www.abrirllave.com/xml/test-de-autoevaluacion.php>

Capítulo 2

Utilización de lenguajes de marcas en entornos web

Teoría y ejemplos

- *Apuntes de HTML*
<http://www.abrirllave.com/html/apuntes-de-html.php>
- *Ejemplos de HTML*
<http://www.abrirllave.com/html/ejemplos.php>

Ejercicios

- *Ejercicios resueltos de HTML*
<http://www.abrirllave.com/html/ejercicios-resueltos.php>
 - *Enunciados de los ejercicios resueltos*
<http://www.abrirllave.com/html/enunciados-de-los-ejercicios-resueltos.php>
-

2.1. HTML

Contenidos del tutorial de HTML www.abrirllave.com/html/

[2.1.1. Qué es HTML](#)

[2.1.2. Primer documento HTML](#)

Elemento "html"

Elemento "head"

Elemento "title"

Elemento "body"

Elemento "p"

Estructura básica de un documento HTML

Cómo crear un documento HTML

Cómo ver el código fuente de un documento HTML en un navegador web

[2.1.3. Comentarios](#)

[2.1.4. Estilos](#)

Estilo en línea - Atributo **style**

Estilo interno - Elemento "style"

Estilo externo - Elemento "link"

Elemento "span" y atributo **class**

[2.1.5. Párrafos](#)

Elemento "pre"

Elemento "br"

Elemento "hr"

[2.1.6. Formato de texto](#)

Elemento "b"

Elemento "i"

Elementos "strong" y "em"

Elemento "small"

Elementos "sub" y "sup"

Elemento "mark"

[2.1.7. Cabeceras](#)

[2.1.8. Enlaces](#)

Elemento "a"

Atributo **target**

Enlace a otra página del mismo sitio web

[2.1.9. Listas](#)

Lista desordenada - Elementos "ul" y "li"

Lista ordenada - Elementos "ol" y "li"

Lista de descripción de términos - Elementos "dl", "dt" y "dd"

2.1.10. Imágenes

Elemento `"img"`

Atributo **src**

Atributos **width** y **height**

Atributo **alt**

2.1.11. Tablas

Tabla básica - Elementos `"table"`, `"tr"` y `"td"`

Tabla con bordes - Propiedad CSS **border**

Tabla con celdas separadas - Propiedad CSS **border-spacing**

Tabla con borde junto - Propiedad CSS **border-collapse**

Tabla con relleno de celdas - Propiedad CSS **padding**

Ancho de una tabla - Propiedad CSS **width**

Título de una tabla - Elemento `"caption"`

Una celda puede abarcar varias columnas de una tabla - Atributo **colspan**

Cabeceras de una tabla - Elemento `"th"`

Estructura de una tabla - Elementos `"thead"`, `"tbody"` y `"tfoot"`

2.1.12. Divisiones

Elemento `"div"`

Elemento `"div"` y atributo **class**

Elemento `"div"` y atributo **id**

2.1.13. Formularios

Formulario básico - Elementos `"form"` e `"input"`

Atributo **action**

Atributo **method**

Controles de un formulario - Elementos `"button"`, `"input"`, `"select"` y `"textarea"`

Agrupaciones de elementos - Elementos `"fieldset"` y `"legend"`

Elemento `"label"`

Otros elementos: `"datalist"`, `"meter"`, `"optgroup"`, `"output"`, `"progress"`

2.1.14. Recursos de HTML

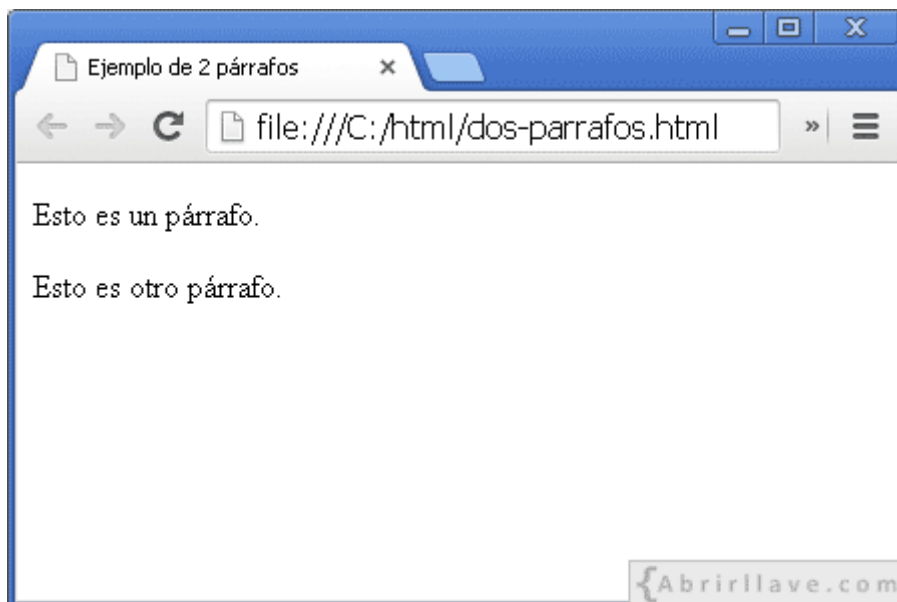
2.1.1. Qué es HTML

HTML (*HyperText Markup Language*, Lenguaje de Marcado de HiperTexto) es un lenguaje utilizado para la creación de páginas web.

HTML es un estándar desarrollado por W3C (*World Wide Web Consortium*). En este enlace "<http://www.w3.org/TR/html/>" se puede consultar su especificación actual, en la cual se basa este tutorial.

2.1.2. Primer documento HTML

En este apartado del tutorial se explica cuál es la estructura básica de un documento HTML a través de un ejemplo sencillo ("**dos-parrafos.html**") donde se visualizan dos párrafos. Por ejemplo, en *Google Chrome*, el resultado que se espera ver en pantalla será algo parecido a:



Elemento "html"

Un documento HTML contiene marcas (etiquetas), las cuales se escriben empleando los caracteres *menor que* "<", *mayor que* ">" y *barra inclinada* "/".

Por ejemplo, las etiquetas de inicio (<html>) y fin (</html>) representan al elemento raíz ("html") que en todo documento HTML hay que escribir.

```
<html>  
</html>
```

Elemento “head”

Dentro del elemento “html”, es decir, entre `<html>` y `</html>`, se debe escribir el elemento “head” que, como iremos viendo a lo largo del tutorial, puede contener diversa información sobre el documento:

```
<html>
  <head>
  </head>
</html>
```

Elemento “title”

Entre `<head>` y `</head>` se pueden escribir otros elementos. Por ejemplo, el elemento “title”, el cual sirve para indicar el título del documento.

```
<html>
  <head>
    <title>Ejemplo de 2 párrafos</title>
  </head>
</html>
```

Véase en el ejemplo *“dos-parrafos.html”* que, el texto escrito entre `<title>` y `</title>`, es decir *“Ejemplo de 2 párrafos”*, se visualiza en la parte superior de la pestaña donde se muestra la página web en el navegador.

Elemento “body”

En un documento HTML, después del “head”, hay que escribir el elemento “body”:

```
<html>
  <head>
    <title>Ejemplo de 2 párrafos</title>
  </head>
  <body>
  </body>
</html>
```

El elemento “body” alberga todo el contenido (párrafos, imágenes, vídeos...) del documento, los cuales se mostrarán en el navegador.

Elemento “p”

En este caso, entre `<body>` y `</body>` se incluyen dos elementos “p” delimitados por la etiqueta de inicio `<p>` y la de cierre `</p>`:

```
<html>
  <head>
    <title>Ejemplo de 2 párrafos</title>
  </head>
  <body>
    <p>Esto es un párrafo.</p>
    <p>Esto es otro párrafo.</p>
  </body>
</html>
```

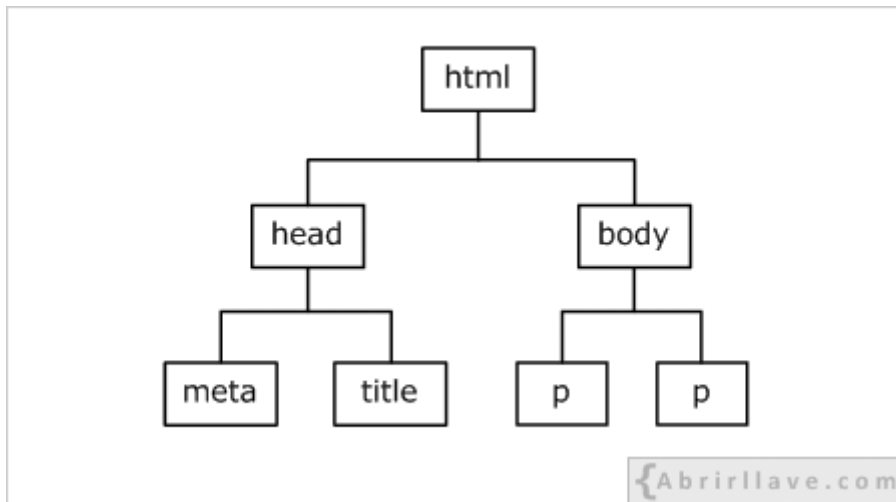
Estructura básica de un documento HTML

Finalmente, completaremos el código de este primer ejemplo añadiendo:

```
<!DOCTYPE html>
<html lang="es-ES">
  <head>
    <meta charset="utf-8">
    <title>Ejemplo de 2 párrafos</title>
  </head>
  <body>
    <p>Esto es un párrafo.</p>
    <p>Esto es otro párrafo.</p>
  </body>
</html>
```

- **<!DOCTYPE html>** en la primera línea. Esta es la forma de indicar al navegador donde se visualice el documento que este está escrito en la versión HTML5.
- **lang="es-ES"** en la etiqueta de inicio **<html>**. Esto sirve para especificar, mediante el atributo **lang**, el idioma del contenido del elemento "html". En este caso español (España). Otros posibles valores del atributo **lang** son: **"es-AR"** para español (Argentina), **"es-CO"** para español (Colombia), **"es-MX"** para español (México), etc.
- **<meta charset="utf-8">** dentro del elemento "head". Así, se indica la codificación de caracteres del documento, en esta ocasión UTF-8 (*8-bit Unicode Transformation Format*, Formato de Transformación Unicode de 8 bits). Obsérvese que, el elemento "meta" está vacío (no tiene contenido) y, en consecuencia, no necesita ser cerrado escribiendo **</meta>**.

De esta forma, hemos escrito la estructura básica de un documento HTML que muestra por pantalla dos párrafos. Gráficamente, los elementos utilizados se pueden representar del siguiente modo:



Nota: los documentos HTML están formados por elementos cuya gran mayoría deben escribirse delimitados por una etiqueta de inicio **<etiqueta>** y otra de fin **</etiqueta>**, tales como: "html", "head", "body", "title" o "p". No obstante, existen otros elementos que no necesitan ser cerrados con la etiqueta de fin, como por ejemplo: "meta".

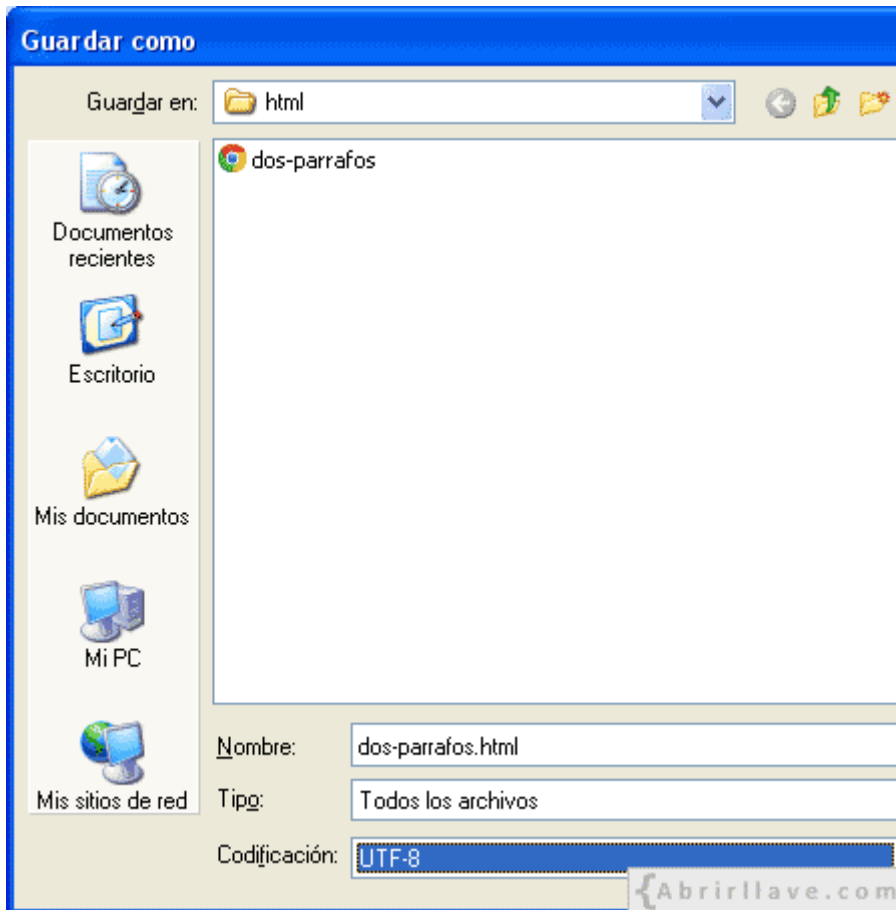
Cómo crear un documento HTML

Por ejemplo, con el *Bloc de notas* de *Microsoft Windows*, podemos crear el siguiente archivo **"dos-parrafos.html"** (la extensión del archivo puede ser ".html" o ".htm"):

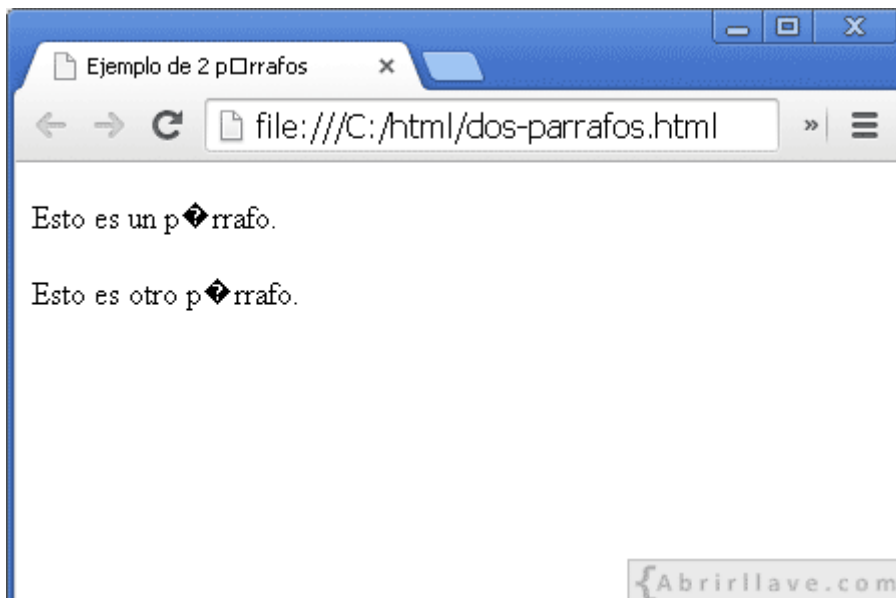
```

<!DOCTYPE html>
<html lang="es-ES">
  <head>
    <meta charset="utf-8">
    <title>Ejemplo de 2 párrafos</title>
  </head>
  <body>
    <p>Esto es un párrafo.</p>
    <p>Esto es otro párrafo.</p>
  </body>
</html>
  
```

A la hora de guardar el archivo, pinchando en **"Archivo" > "Guardar como..."**, es importante hacerlo codificado en UTF-8, como se muestra en la siguiente imagen:



El resultado sería el esperado. Ahora bien, si se guardase codificado en ANSI en vez de UTF-8, en pantalla se visualizaría algo parecido a:

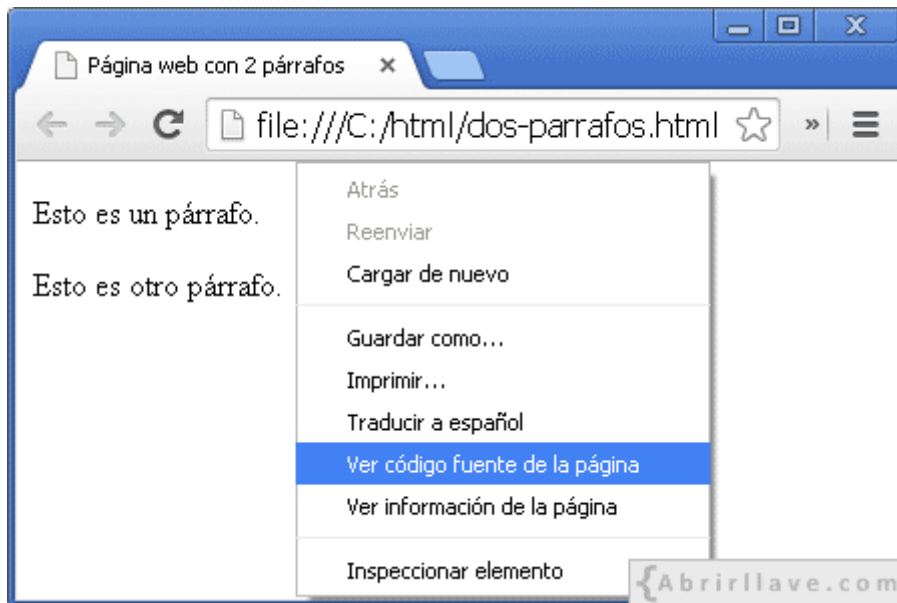


Ejercicio resuelto

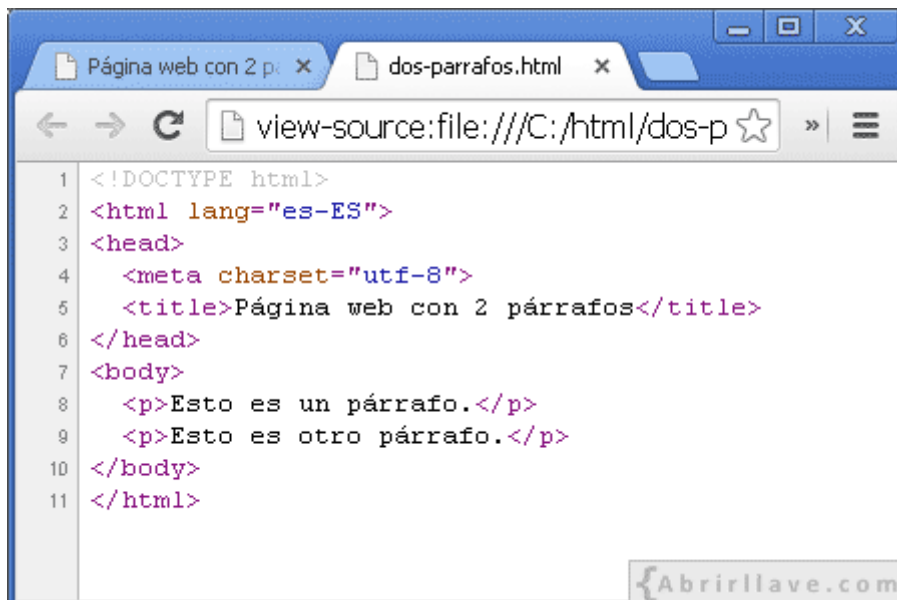
- [Crear primer documento HTML](#)

Cómo ver el código fuente de un documento HTML en un navegador web

En un navegador es posible ver el código fuente de un documento HTML. Por ejemplo, en *Google Chrome* se puede ver pulsando el botón derecho del ratón sobre la pantalla y seleccionando "Ver código fuente de la página":



En este caso se verá:



2.1.3. Comentarios

En un documento HTML, los comentarios pueden escribirse entre los caracteres "`<!--`" y "`-->`". Por ejemplo:

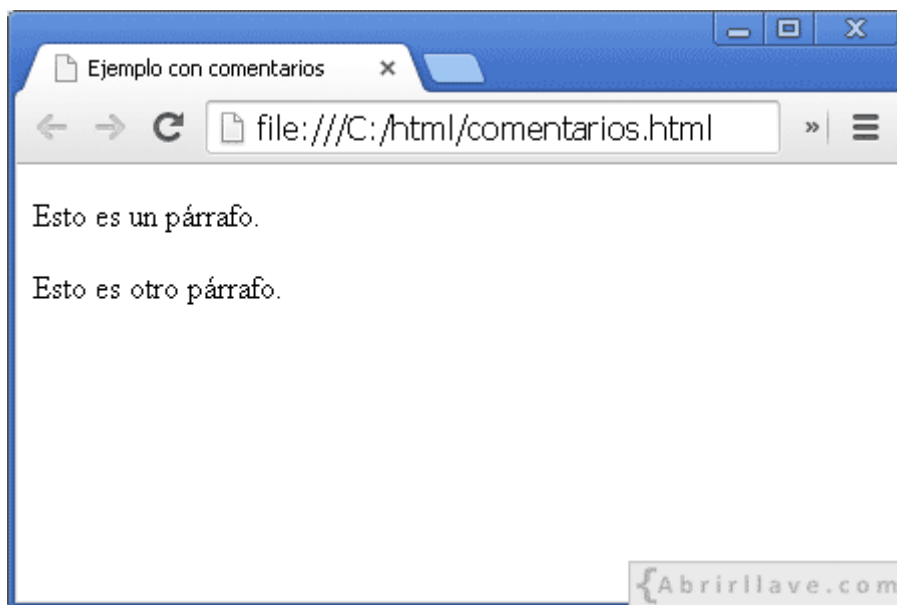
```
<!-- Esto es un comentario escrito en un documento HTML -->
```

Los comentarios no se mostrarán en el navegador donde se visualice el documento HTML.

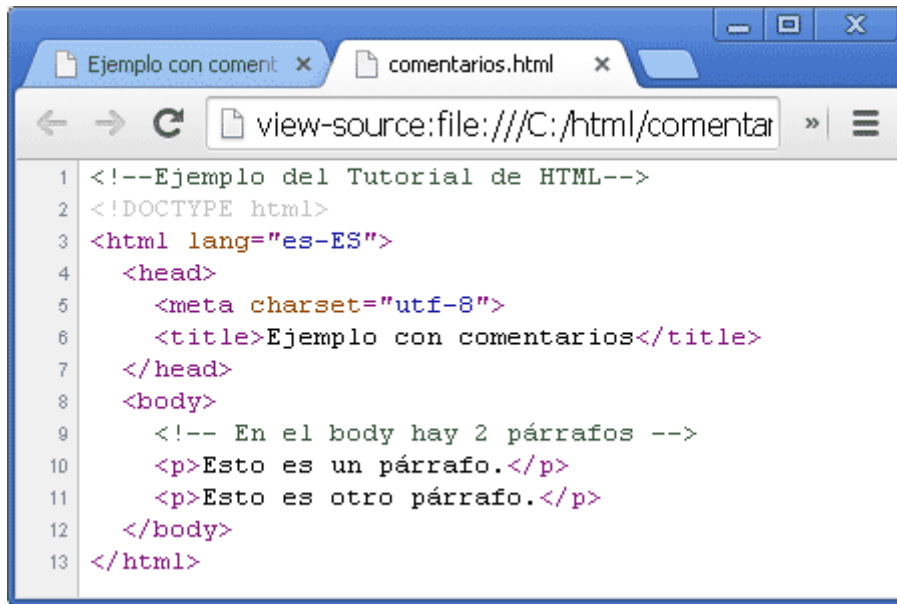
EJEMPLO Dado el archivo "*comentarios.html*", con dos comentarios:

```
<!--Ejemplo del Tutorial de HTML-->
<!DOCTYPE html>
<html lang="es-ES">
  <head>
    <meta charset="utf-8">
    <title>Ejemplo con comentarios</title>
  </head>
  <body>
    <!-- En el body hay 2 párrafos -->
    <p>Esto es un párrafo.</p>
    <p>Esto es otro párrafo.</p>
  </body>
</html>
```

En un navegador se mostrarán solamente los dos párrafos escritos en el "body":



No obstante, al ver el código fuente del documento en un navegador, sí se mostrarán los comentarios:



Ejercicio resuelto

- [Crear documento HTML con comentarios](#)

2.1.4. Estilos

Para cambiar el estilo en el que por defecto se visualizan los elementos de un documento HTML en un navegador web, existen varios métodos:

- **Estilo en línea (*Inline*):** utilizando el atributo **style**.
- **Estilo interno (*Internal*):** usando un elemento "style".
- **Estilo externo (*External*):** empleando un archivo CSS (*Cascading Style Sheets*) externo al documento HTML.

EJEMPLO A continuación se explica cómo hacer uso de estos métodos para mostrar un párrafo de color rojo.

Estilo en línea - Atributo **style**

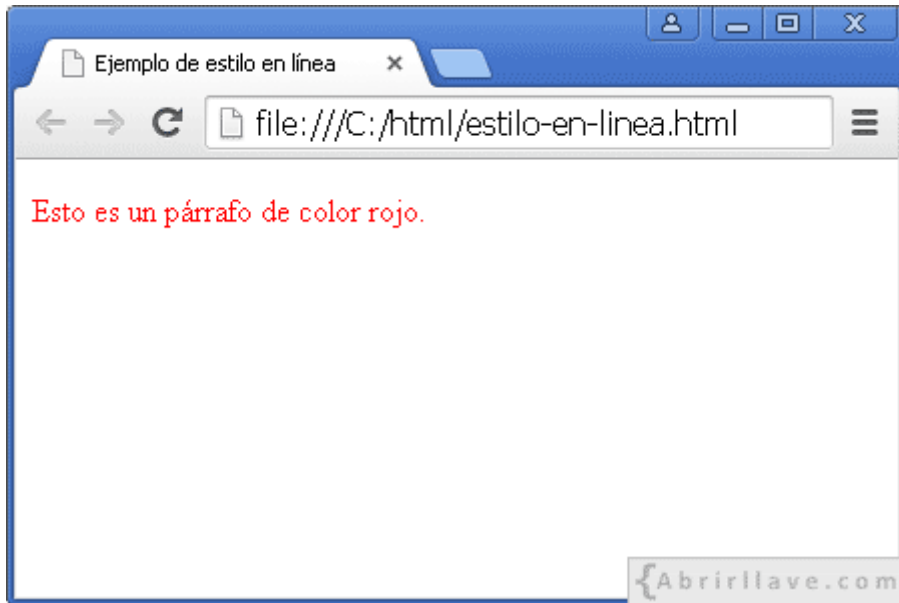
Para cambiar el color de un párrafo, se puede escribir el siguiente código ("**estilo-en-línea.html**"):

```

<!DOCTYPE html>
<html lang="es-ES">
  <head>
    <meta charset="utf-8">
    <title>Ejemplo de estilo en línea</title>
  </head>
  <body>
    <p style="color:red">Esto es un párrafo de color rojo.</p>
  </body>
</html>

```

Obsérvese que, se ha utilizado el atributo **style** en el elemento "p". En un navegador se verá algo similar a:



Estilo interno - Elemento "style"

En este caso, se usa el elemento "style" dentro del "head" ("**estilo-interno.html**"):

```
<!DOCTYPE html>
<html lang="es-ES">
  <head>
    <meta charset="utf-8">
    <title>Ejemplo de estilo interno</title>
    <style>
      p {color:red;}
    </style>
  </head>
  <body>
    <p>Esto es un párrafo de color rojo.</p>
  </body>
</html>
```

Estilo externo - Elemento "link"

Para aplicar un estilo externo, hay que hacer uso del elemento "link", el cual permite enlazar el documento HTML con un archivo externo. El código HTML puede ser ("**estilo-externo.html**"):

```
<!DOCTYPE html>
<html lang="es-ES">
  <head>
    <meta charset="utf-8">
    <title>Ejemplo de estilo externo</title>
    <link rel="stylesheet" href="styles.css">
  </head>
  <body>
    <p>Esto es un párrafo de color rojo.</p>
  </body>
</html>
```

Véase que, en el atributo **rel** se indica la relación que existe entre el archivo HTML y el archivo CSS. Por otro lado, por medio del atributo **href** se ha enlazado el documento HTML con el archivo **"styles.css"**, que es donde está definido el estilo del párrafo:

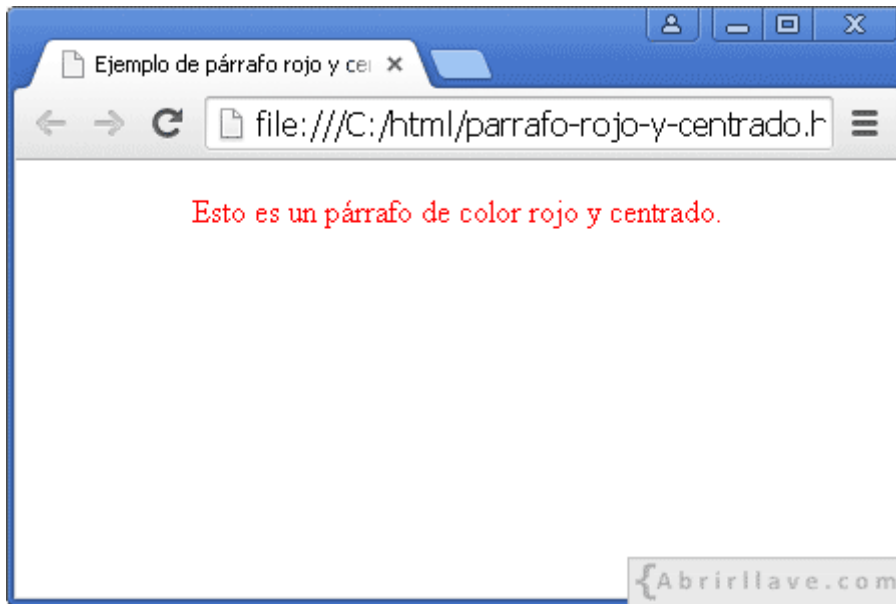
```
p {color:red;}
```

Por norma general, utilizar archivos CSS externos es la forma más habitual y adecuada de aplicar estilos a documentos HTML.

EJEMPLO Además de **color**, existen otras muchas propiedades CSS que pueden utilizarse. Por ejemplo, si además de rojo, también se quiere indicar que el párrafo se muestre centrado en la pantalla del navegador, esto se puede hacer usando la propiedad **text-align** (**"parrafo-rojo-y-centrado.html"**):

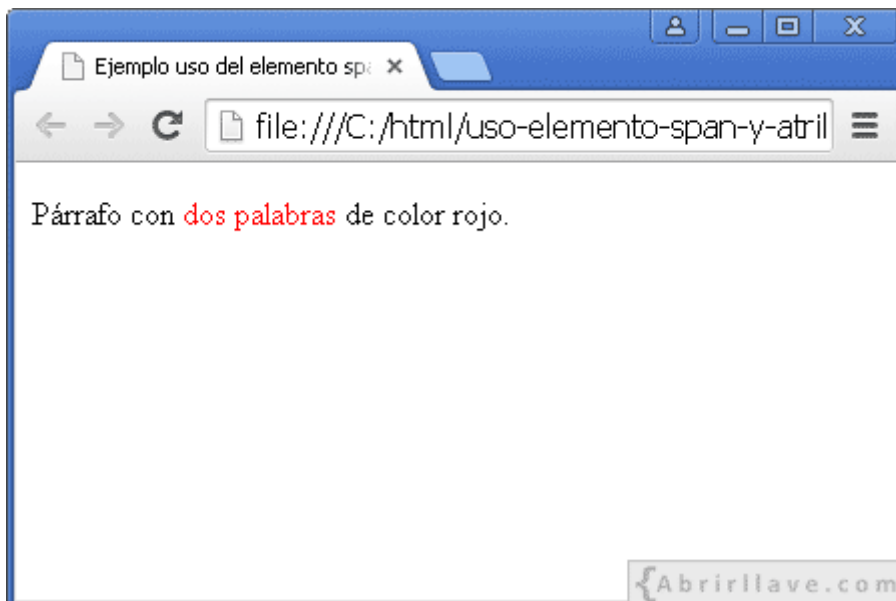
```
<!DOCTYPE html>
<html lang="es-ES">
  <head>
    <meta charset="utf-8">
    <title>Ejemplo de párrafo rojo y centrado</title>
    <style>
      p {
        color:red;
        text-align:center;
      }
    </style>
  </head>
  <body>
    <p>Esto es un párrafo de color rojo y centrado.</p>
  </body>
</html>
```

En pantalla se visualizará:



Elemento "span" y atributo **class**

EJEMPLO Para ver parte del texto de un párrafo de color rojo:



Se puede escribir ("*uso-elemento-span-y-atributo-class.html*"):


```

<!DOCTYPE html>
<html lang="es-ES">
  <head>
    <meta charset="utf-8">
    <title>Ejemplo uso del elemento span y del atributo class</title>
    <style>
      .rojo {color:red;}
    </style>
  </head>
  <body>
    <p>Párrafo con <span class="rojo">dos palabras</span> de color
    rojo.</p>
  </body>
</html>

```

Véase que, asignando el valor "**rojo**" al atributo **class**, se ha indicado el estilo (en este caso **color:red;**) con el que el texto contenido dentro del elemento "span" (en este caso "dos palabras") debe mostrarse en pantalla.

.rojo tiene que llevar el carácter punto, ya que, ha sido definido por el programador.

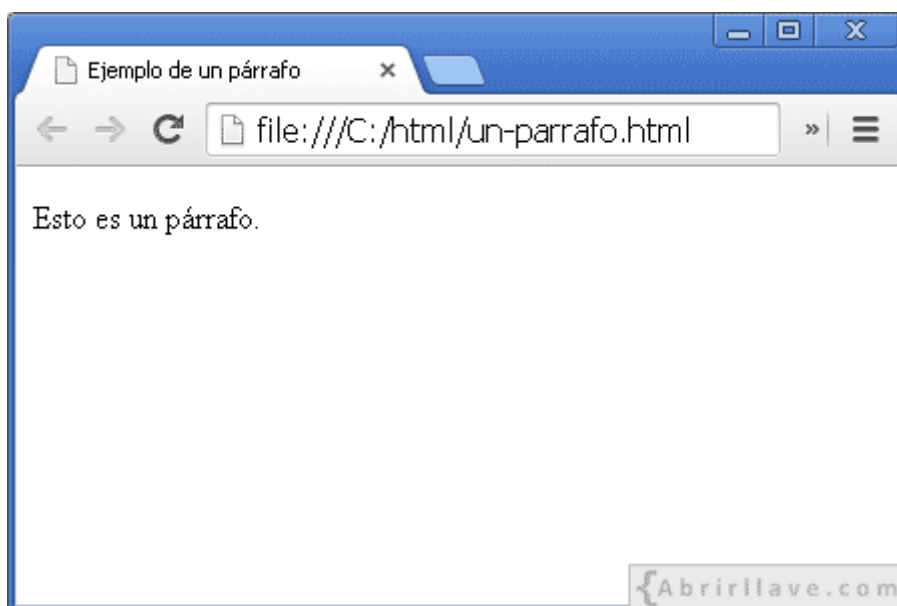
Ejercicio resuelto

- [Cuatro párrafos con estilos](#)

2.1.5. Párrafos

En un documento HTML, **para escribir un párrafo se usa el elemento "p"**, cuyas etiquetas de inicio y fin son, respectivamente: **<p>** y **</p>**.

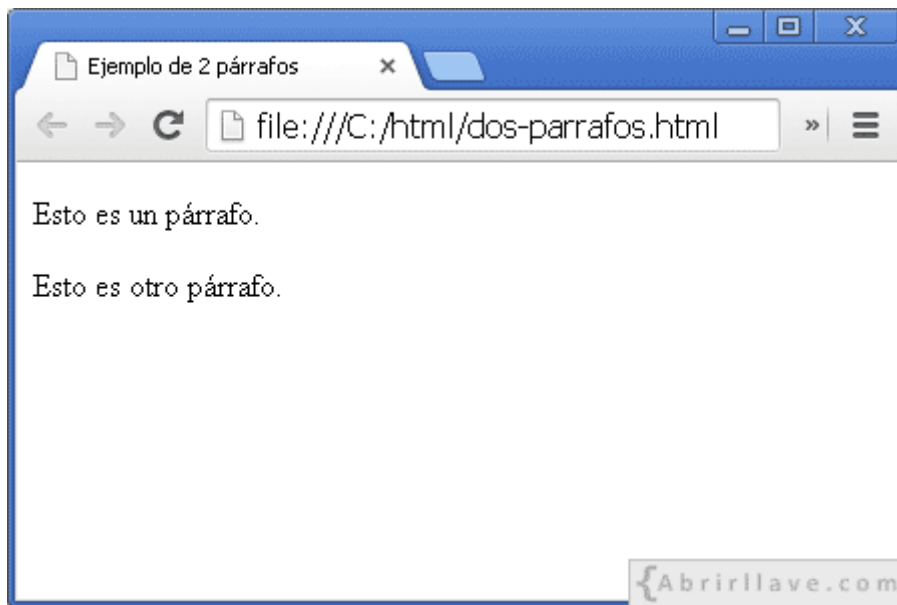
EJEMPLO Para que en un navegador web se visualice:



Se puede escribir el siguiente código ("**un-parrafo.html**"):

```
<!DOCTYPE html>
<html lang="es-ES">
  <head>
    <meta charset="utf-8">
    <title>Ejemplo de un párrafo</title>
  </head>
  <body>
    <p>Esto es un párrafo.</p>
  </body>
</html>
```

EJEMPLO Para ver 2 párrafos:



Se puede escribir ("*dos-parrafos.html*"):

```
<!DOCTYPE html>
<html lang="es-ES">
  <head>
    <meta charset="utf-8">
    <title>Ejemplo de 2 párrafos</title>
  </head>
  <body>
    <p>Esto es un párrafo.</p>
    <p>Esto es otro párrafo.</p>
  </body>
</html>
```

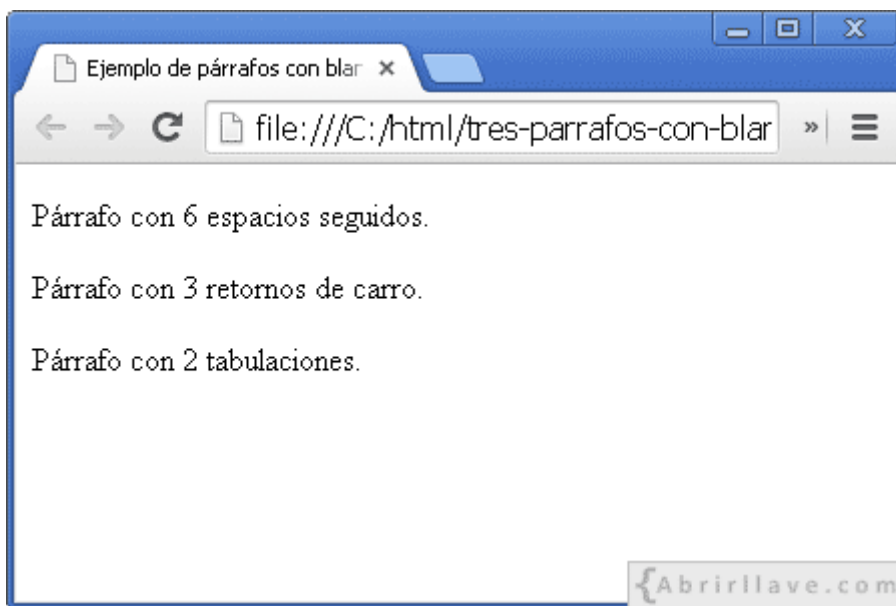
EJEMPLO Varias tabulaciones, retornos de carro y/o espacios en blanco escritos –seguidos– en el texto de un documento HTML, se visualizarán como un único espacio en blanco en un navegador web. Por ejemplo, dado el archivo "*tres-parrafos-con-blancos.html*":

```

<!DOCTYPE html>
<html lang="es-ES">
  <head>
    <meta charset="utf-8">
    <title>Ejemplo de párrafos con blancos</title>
  </head>
  <body>
    <p>Párrafo con          6 espacios seguidos.</p>
    <p>Párrafo con
3 retornos de carro.</p>
    <p>Párrafo con          2 tabulaciones.</p>
  </body>
</html>

```

En un navegador web se verá:



Elemento “pre”

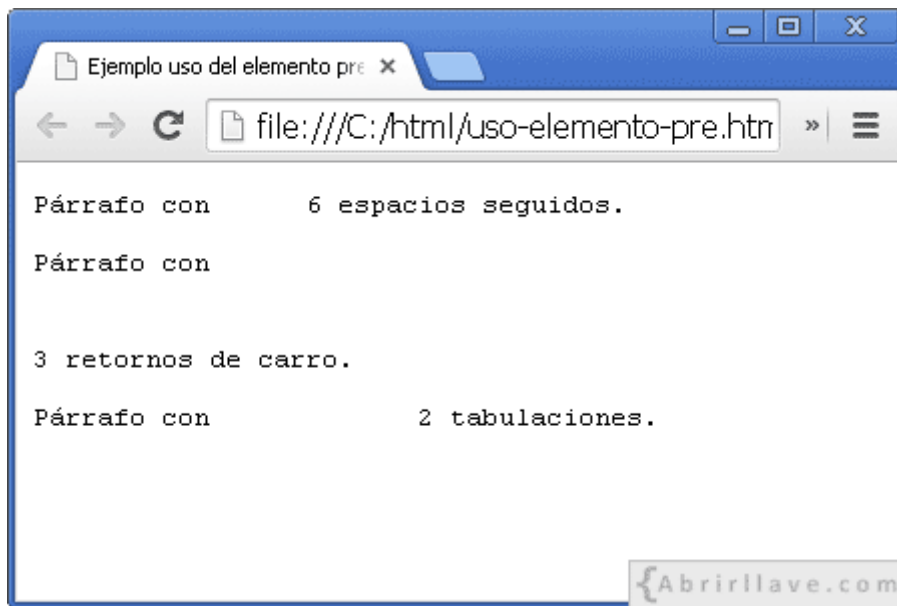
EJEMPLO Si se desea mostrar todos los espacios en blanco, tabuladores y retornos de carro escritos en un documento HTML, se puede hacer utilizando el elemento “pre”. Como ejemplo, véase *“uso-elemento-pre.html”*:

```

<!DOCTYPE html>
<html lang="es-ES">
  <head>
    <meta charset="utf-8">
    <title>Ejemplo uso del elemento pre</title>
  </head>
  <body>
    <pre>Párrafo con          6 espacios seguidos.</pre>
    <pre>Párrafo con
3 retornos de carro.</pre>
    <pre>Párrafo con          2 tabulaciones.</pre>
  </body>
</html>

```

Ahora, en pantalla se verá:



Se obtiene un resultado similar escribiendo:

```
<!DOCTYPE html>
<html lang="es-ES">
  <head>
    <meta charset="utf-8">
    <title>Ejemplo uso del elemento pre</title>
  </head>
  <body>
    <pre>Párrafo con      6 espacios seguidos.
Párrafo con

3 retornos de carro.
Párrafo con          2 tabulaciones.</pre>
  </body>
</html>
```

Elemento "br"

EJEMPLO Para ver por pantalla:



Se puede escribir el siguiente código ("**tres-textos.html**"):

```
<!DOCTYPE html>
<html lang="es-ES">
  <head>
    <meta charset="utf-8">
    <title>Ejemplo de 3 textos</title>
  </head>
  <body>
    <p>Texto uno.</p>
    <p>Texto dos.</p>
    <p>Texto tres.</p>
  </body>
</html>
```

Ahora bien, en vez de:

```
<p>Texto uno.</p>
<p>Texto dos.</p>
<p>Texto tres.</p>
```

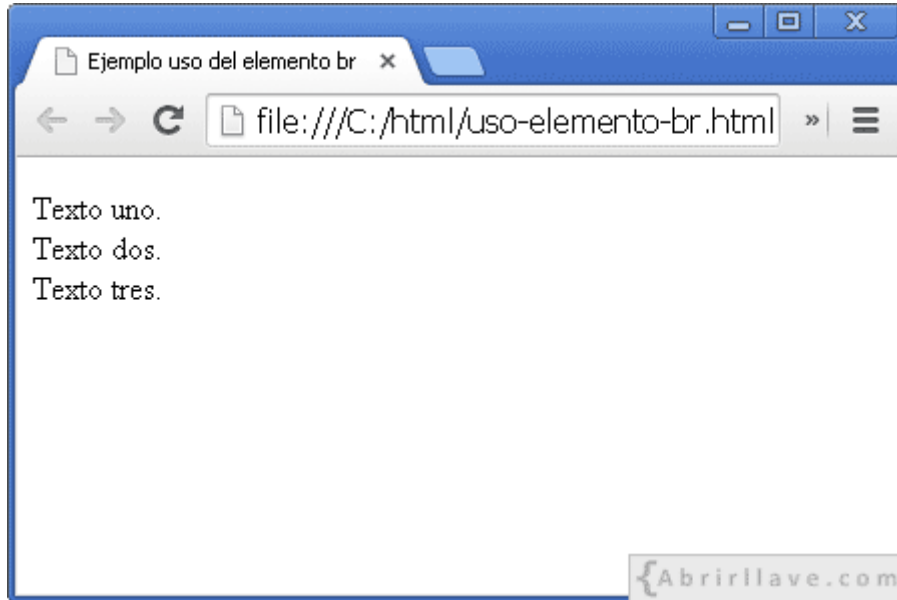
Un resultado parecido puede conseguirse utilizando un único elemento "p" y dos elementos "br", escribiendo por ejemplo ("**uso-elemento-br.html**"):

```
<p>Texto uno.<br>Texto dos.<br>Texto tres.</p>
```

```
<p>Texto uno.<br>
Texto dos.<br>
Texto tres.</p>
```

```
<p>Texto uno.<br>
  Texto dos.<br>
  Texto tres.</p>
```

En todos los casos anteriores, en pantalla se verá algo similar a:



Como se puede apreciar, **el elemento "br" sirve para hacer un salto de línea**, y no necesita la etiqueta de cierre (**</br>**) debido a que no tiene contenido (está vacío).

Se puede obtener un resultado parecido haciendo uso del elemento "pre" escribiendo:

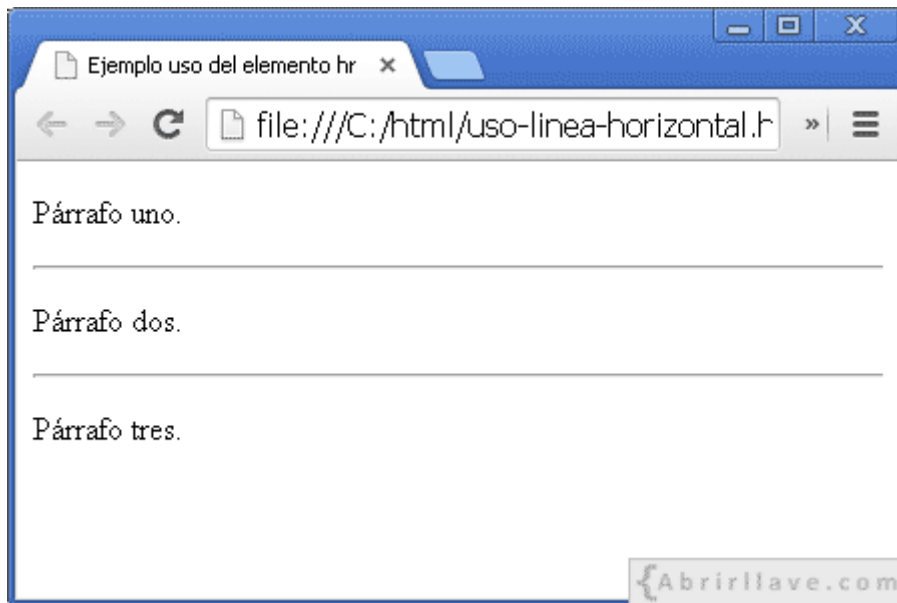
```
<pre>Párrafo uno.
Párrafo dos.
Párrafo tres.</pre>
```

Elemento "hr"

EJEMPLO Para mostrar una línea horizontal en la pantalla, se puede usar el elemento "hr". Por ejemplo, dado el archivo **"uso-linea-horizontal.html"**:

```
<!DOCTYPE html>
<html lang="es-ES">
  <head>
    <meta charset="utf-8">
    <title>Ejemplo uso del elemento hr</title>
  </head>
  <body>
    <p>Párrafo uno.</p>
    <hr>
    <p>Párrafo dos.</p>
    <hr>
    <p>Párrafo tres.</p>
  </body>
</html>
```

Se verá:



Obsérvese que, el elemento "hr" no tiene contenido y, por tanto, no es necesario escribir la etiqueta de fin (`</hr>`).

Ejercicio resuelto

- [Datos y números](#)

2.1.6. Formato de texto

HTML proporciona algunos elementos para indicar en qué formato se va a mostrar en un navegador web un determinado texto.

Elemento "b"

EJEMPLO El elemento "b" sirve para indicar que un texto se muestre en negrita. Así, para visualizar:

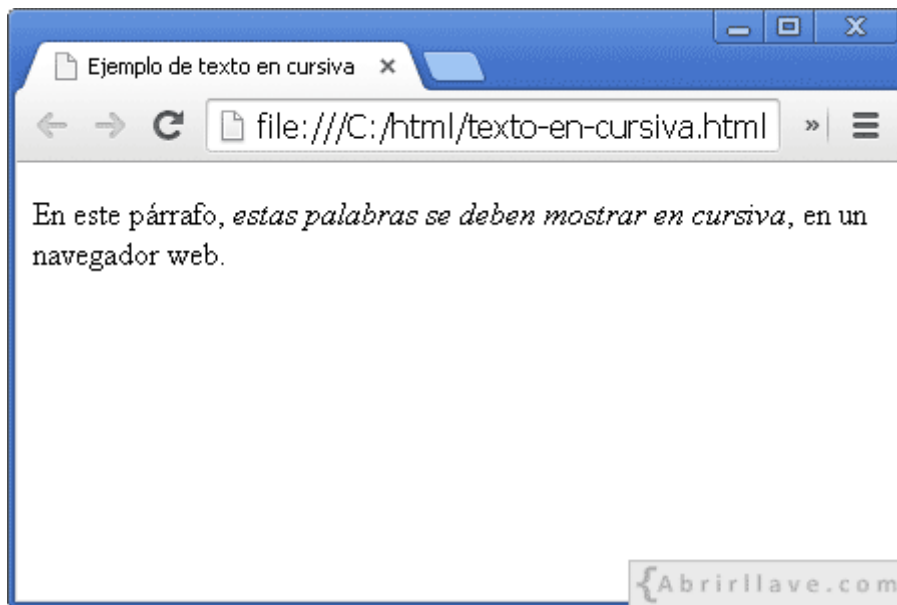


Se puede escribir ("**texto-en-negrita.html**"):

```
<!DOCTYPE html>
<html lang="es-ES">
  <head>
    <meta charset="utf-8">
    <title>Ejemplo de texto en negrita</title>
  </head>
  <body>
    <p>En este párrafo, estas palabras se deben mostrar en negrita, en un navegador web.</p>
  </body>
</html>
```

Elemento "i"

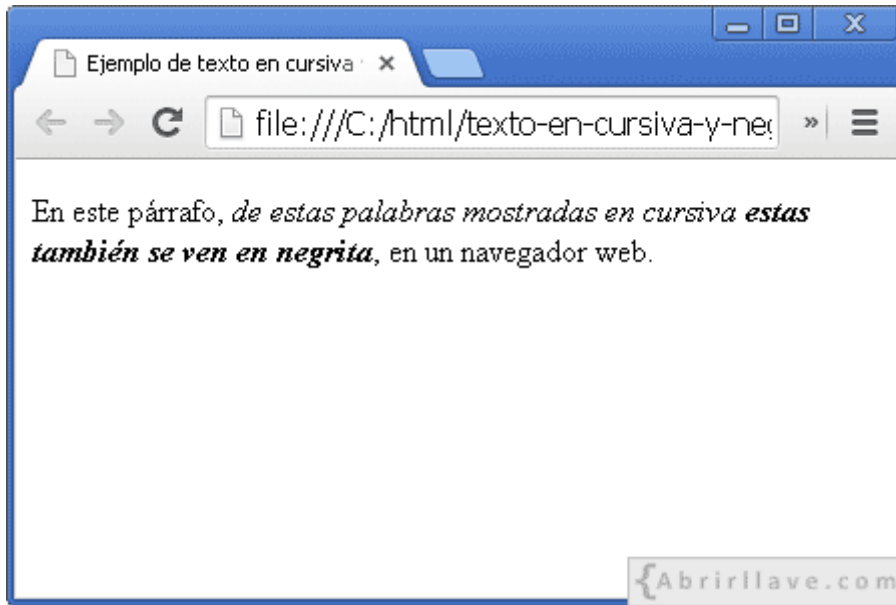
EJEMPLO El elemento "i" sirve para indicar que un texto se muestre en itálica (cursiva). En consecuencia, para ver:



El código puede ser ("**texto-en-cursiva.html**"):

```
<!DOCTYPE html>
<html lang="es-ES">
  <head>
    <meta charset="utf-8">
    <title>Ejemplo de texto en cursiva</title>
  </head>
  <body>
    <p>En este párrafo, estas palabras se deben mostrar en cursiva, en un navegador web.</p>
  </body>
</html>
```


EJEMPLO A un mismo texto se puede indicar más de un formato. Por ejemplo, para ver:



Podemos escribir el siguiente documento HTML ("**texto-en-cursiva-y-negrita.html**"):

```
<!DOCTYPE html>
<html lang="es-ES">
  <head>
    <meta charset="utf-8">
    <title>Ejemplo de texto en cursiva y negrita</title>
  </head>
  <body>
    <p>En este párrafo, <i>de estas palabras en cursiva <b>estas
también se ven en negrita</b></i>, en un navegador web.</p>
  </body>
</html>
```

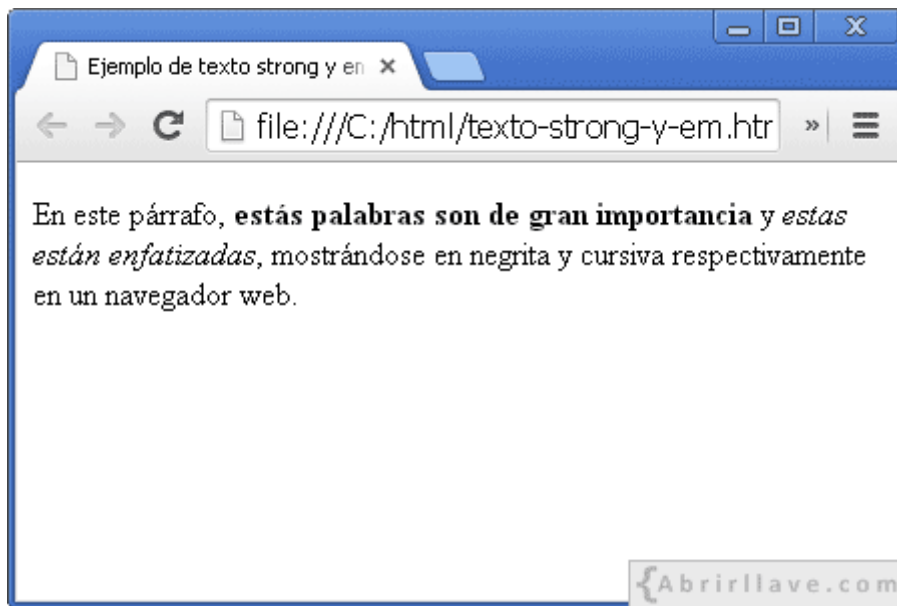
Obsérvese en el código, que las etiquetas **** y **** están anidadas dentro de las etiquetas **<i>** e **</i>**.

Elementos "strong" y "em"

EJEMPLO Por otra parte, el elemento "strong" (que alberga texto de gran importancia) y el elemento "em" (que contiene texto enfatizado), también muestran texto en negrita y cursiva, respectivamente. Véase que, al visualizar el archivo ("**texto-strong-y-em.html**"):

```
<!DOCTYPE html>
<html lang="es-ES">
  <head>
    <meta charset="utf-8">
    <title>Ejemplo de texto en strong y em</title>
  </head>
  <body>
    <p>En este párrafo, <strong>estás palabras son de gran
importancia</strong> y <em>estas están enfatizadas</em>, mostrándose
en negrita y cursiva respectivamente en un navegador web.</p>
  </body>
</html>
```

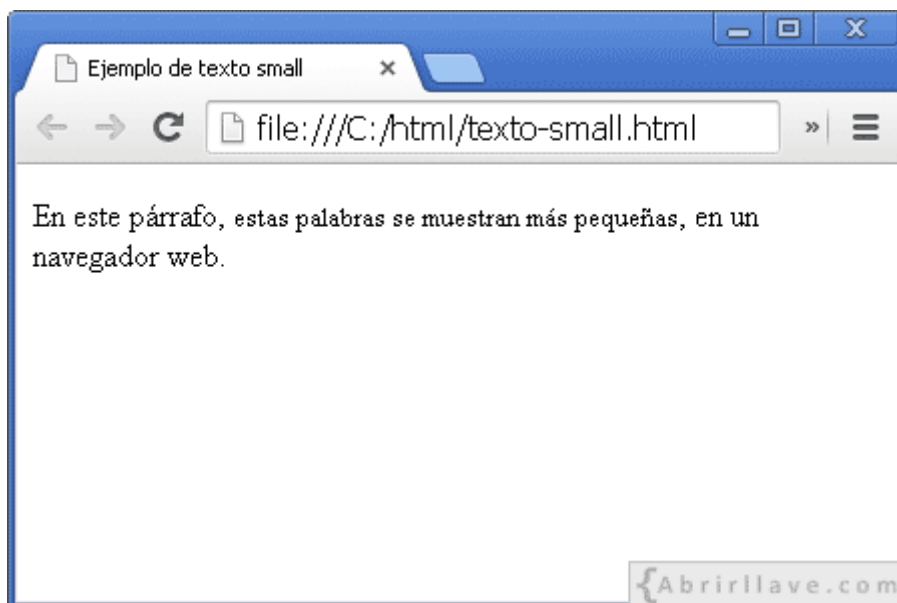
En un navegador se visualizará:



Nota: los elementos "b" e "i", no proporcionan la importancia semántica que sí tienen "strong" y "em".

Elemento "small"

EJEMPLO Con el elemento "small" se indica que un texto se muestre más pequeño en un navegador. Por ejemplo, para ver:

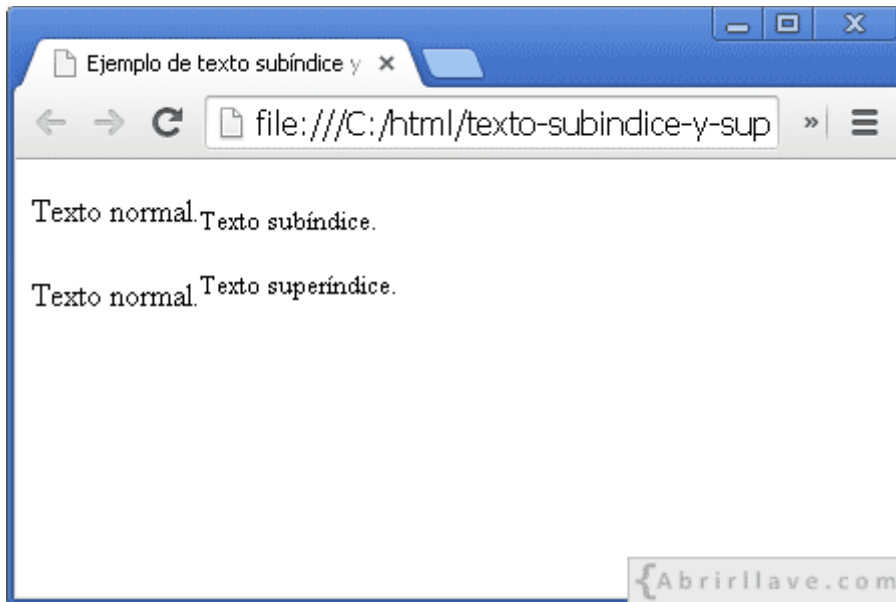


Se puede escribir ("**texto-small.html**"):

```
<!DOCTYPE html>
<html lang="es-ES">
  <head>
    <meta charset="utf-8">
    <title>Ejemplo de texto small</title>
  </head>
  <body>
    <p>En este párrafo, <small>estas palabras se muestran más
    pequeñas</small>, en un navegador web.</p>
  </body>
</html>
```

Elementos “sub” y “sup”

EJEMPLO Los elementos “sub” y “sup” permiten indicar que un texto se muestre como un subíndice o como un superíndice, respectivamente. Por ejemplo, para ver en un navegador:

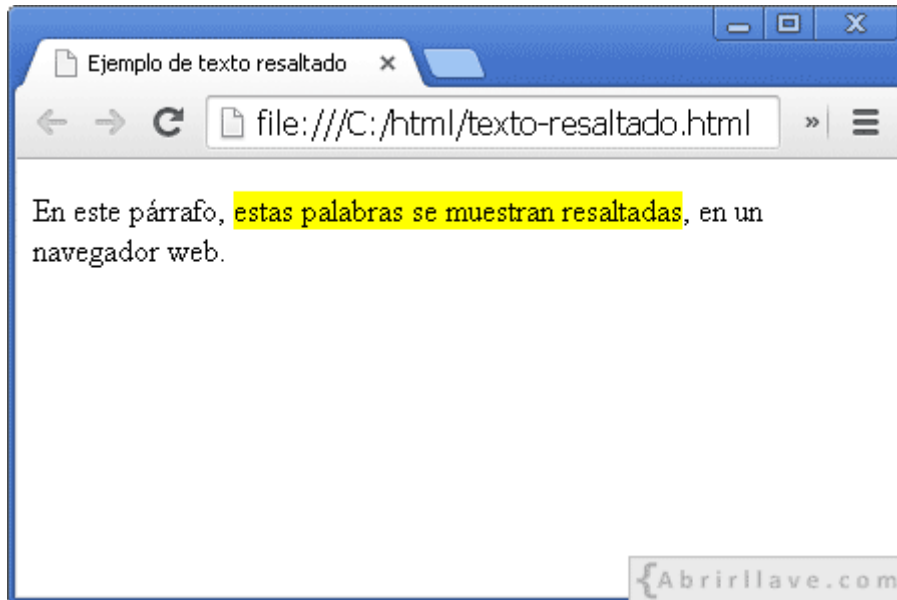


Se puede escribir el siguiente documento HTML (“*texto-subindice-y-superindice.html*”):

```
<!DOCTYPE html>
<html lang="es-ES">
  <head>
    <meta charset="utf-8">
    <title>Ejemplo de texto subíndice y superíndice</title>
  </head>
  <body>
    <p>Texto normal.<sub>Texto subíndice.</sub></p>
    <p>Texto normal.<sup>Texto superíndice.</sup></p>
  </body>
</html>
```

Elemento "mark"

EJEMPLO El elemento "mark" permite indicar que un texto se muestre resaltado en un navegador, como por ejemplo:



Para ello, se puede escribir el siguiente código ("**texto-resaltado.html**"):

```
<!DOCTYPE html>
<html lang="es-ES">
  <head>
    <meta charset="utf-8">
    <title>Ejemplo de texto resaltado</title>
  </head>
  <body>
    <p>En este párrafo, <mark>estas palabras se muestran
resaltadas</mark>, en un navegador web.</p>
  </body>
</html>
```

Ejercicio resuelto

- [Formatos de texto](#)

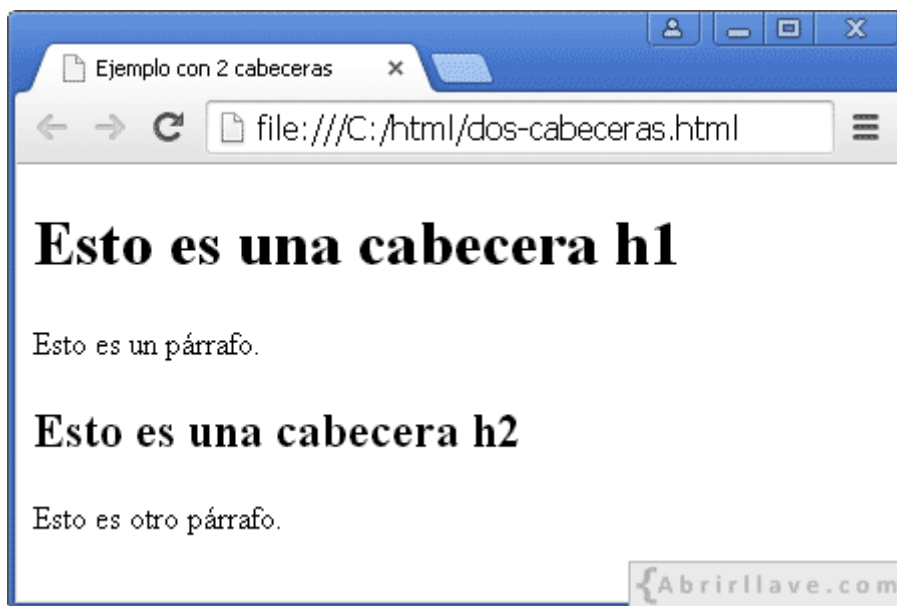
2.1.7. Cabeceras

En HTML existen seis elementos que pueden ser utilizados para escribir cabeceras o encabezados: "h1", "h2", "h3", "h4", "h5" y "h6".

EJEMPLO En el siguiente documento HTML se han escrito dos cabeceras:

```
<!DOCTYPE html>
<html lang="es-ES">
  <head>
    <meta charset="utf-8">
    <title>Ejemplo con 2 cabeceras</title>
  </head>
  <body>
    <h1>Esto es una cabecera h1</h1>
    <p>Esto es un párrafo.</p>
    <h2>Esto es una cabecera h2</h2>
    <p>Esto es otro párrafo.</p>
  </body>
</html>
```

Al abrir el archivo **"dos-cabeceras.html"** en *Google Chrome* se verá:

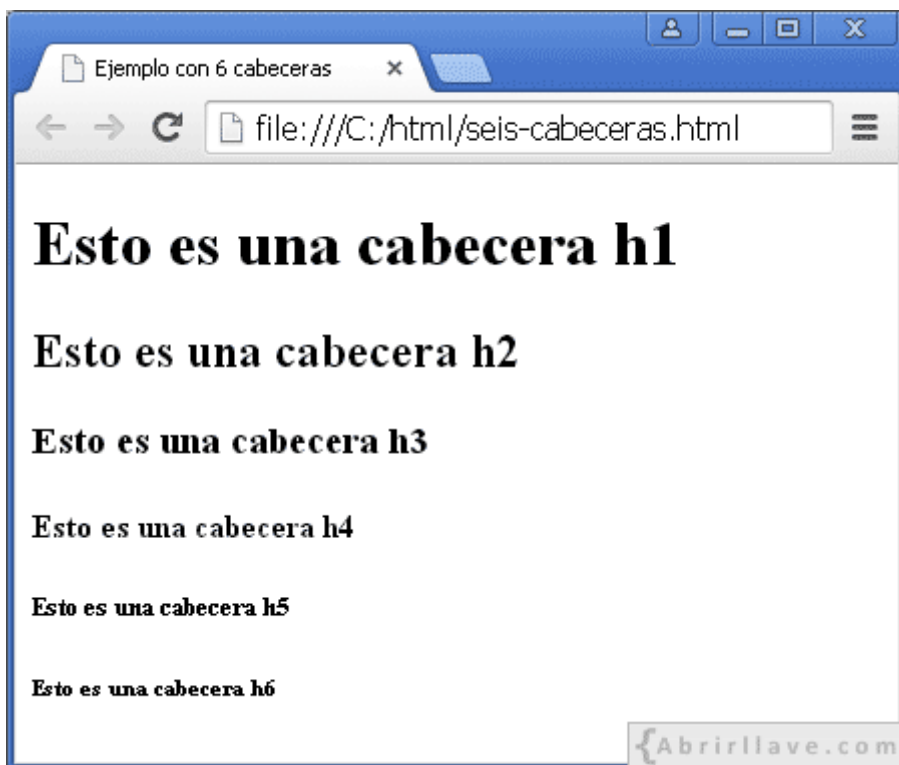


Las cabeceras se utilizan para estructurar el contenido de una página web y, a la hora de utilizarlas, hay que tener en cuenta que, la cabecera "h1" es la de mayor importancia, después "h2", "h3", etc. Siendo "h6" la menos importante.

EJEMPLO Nótese en el siguiente archivo **"seis-cabeceras.html"**:

```
<!DOCTYPE html>
<html lang="es-ES">
  <head>
    <meta charset="utf-8">
    <title>Ejemplo con 6 cabeceras</title>
  </head>
  <body>
    <h1>Esto es una cabecera h1</h1>
    <h2>Esto es una cabecera h2</h2>
    <h3>Esto es una cabecera h3</h3>
    <h4>Esto es una cabecera h4</h4>
    <h5>Esto es una cabecera h5</h5>
    <h6>Esto es una cabecera h6</h6>
  </body>
</html>
```

que en pantalla las cabeceras se muestran de mayor a menor tamaño, según su importancia:



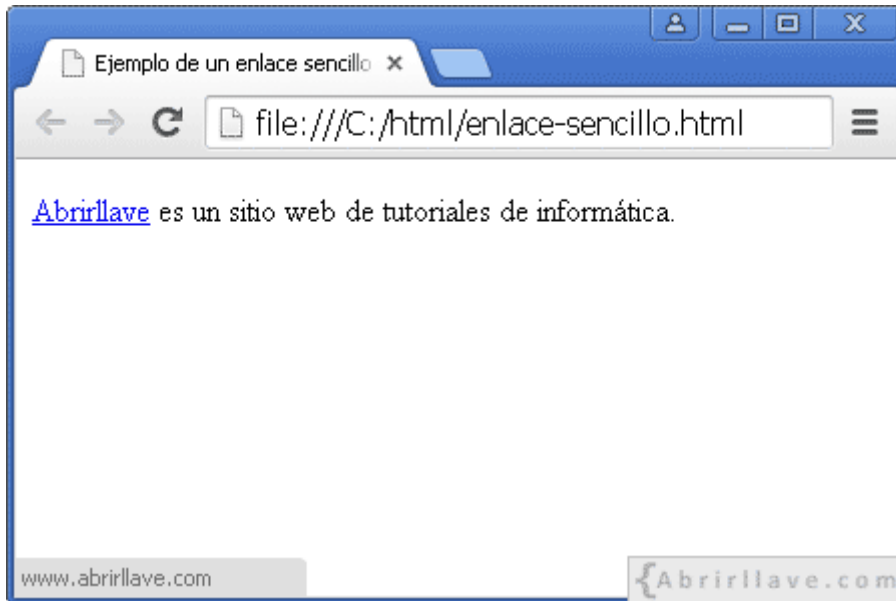
En muchas páginas web es habitual que exista una única cabecera "h1" y varias cabeceras "h2", varias más "h3", etc.

2.1.8. Enlaces

En un documento HTML los enlaces se pueden definir utilizando el elemento "a".

Elemento "a"

EJEMPLO Si en pantalla se quiere ver el siguiente párrafo, donde hay un enlace a www.abrirllave.com:



El código puede ser ("**enlace-sencillo.html**"):

```
<!DOCTYPE html>
<html lang="es-ES">
  <head>
    <meta charset="utf-8">
    <title>Ejemplo de un enlace sencillo</title>
  </head>
  <body>
    <p><a href="http://www.abrirllave.com/">Abrirllave</a> es
un sitio web de tutoriales de informática.</p>
  </body>
</html>
```

Obsérvese que, entre las etiquetas `<a>` y `` se ha escrito el texto del enlace mostrado, en este caso **Abrirllave**. Por otro lado, con el atributo **href**, se ha indicado la dirección web a la que se accederá al hacer clic en el enlace, es decir, **http://www.abrirllave.com/**.

Atributo target

EJEMPLO Compruébese en *"enlace-sencillo.html"* que al hacer clic en el enlace hacia *www.abrirllave.com*, no se abre una nueva pestaña en el navegador. Para que esto suceda, se puede utilizar el atributo **target** con el valor **"_blank"** como se muestra a continuación (*"enlace-a-otra-pestana.html"*):

```
<!DOCTYPE html>
<html lang="es-ES">
  <head>
    <meta charset="utf-8">
    <title>Ejemplo de enlace que abre otra pestaña</title>
  </head>
  <body>
    <p><a href="http://www.abrirllave.com/"
target="_blank">Abrirllave</a> es un sitio web de tutoriales de
informática.</p>
  </body>
</html>
```

Enlace a otra página del mismo sitio web

EJEMPLO Si en un documento HTML se quiere escribir un enlace a otro documento ubicado en el mismo sitio web, por ejemplo a *"otra-pagina.html"*, visualizándose en pantalla:



Se podría escribir (*"pagina-con-un-enlace-a-otra-pagina.html"*):


```
<!DOCTYPE html>
<html lang="es-ES">
  <head>
    <meta charset="utf-8">
    <title>Ejemplo de enlace a otra página</title>
  </head>
  <body>
    <p>Enlace a <a href="otra-pagina.html">otra página</a>
    ubicada en el mismo sitio web.</p>
  </body>
</html>
```

Ejercicio resuelto

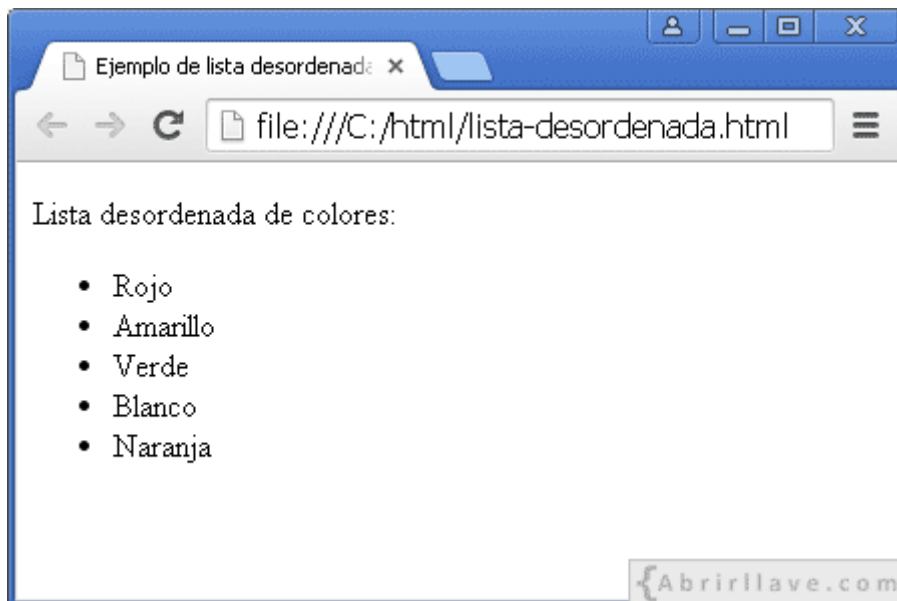
- [Enlaces favoritos](#)

2.1.9. Listas

En HTML, los elementos "ul", "ol", "li", "dl", "dt" y "dd", permiten representar listas ordenadas (*ordered lists*), listas desordenadas (*unordered lists*) y listas de descripción de términos (*description lists*).

Lista desordenada - Elementos "ul" y "li"

EJEMPLO Para que en un navegador se muestre la siguiente lista desordenada de colores:



Se puede hacer uso de los elementos "ul" y "li" del siguiente modo ("*lista-desordenada.html*"):

```

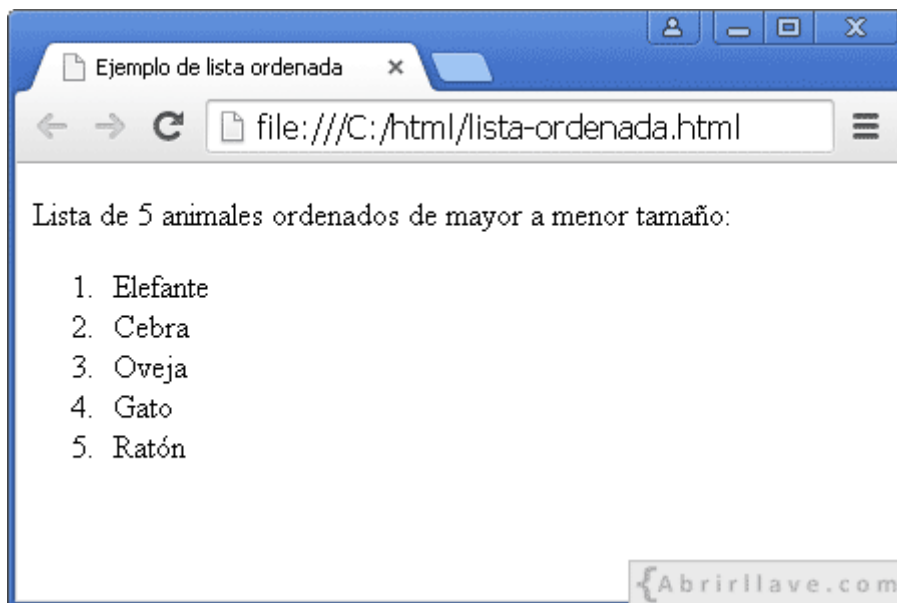
<!DOCTYPE html>
<html lang="es-ES">
  <head>
    <meta charset="utf-8">
    <title>Ejemplo de lista desordenada</title>
  </head>
  <body>
    <p>Lista desordenada de colores:</p>
    <ul>
      <li>Rojo</li>
      <li>Amarillo</li>
      <li>Verde</li>
      <li>Blanco</li>
      <li>Naranja</li>
    </ul>
  </body>
</html>

```

Obsérvese que, cada color está contenido en un elemento "li", y todos ellos se escriben dentro del elemento "ul".

Lista ordenada - Elementos "ol" y "li"

EJEMPLO Si se quiere visualizar una lista ordenada como la siguiente:



Para ello, se pueden utilizar los elementos "ol" y "li" ("**lista-ordenada.html**"):

```

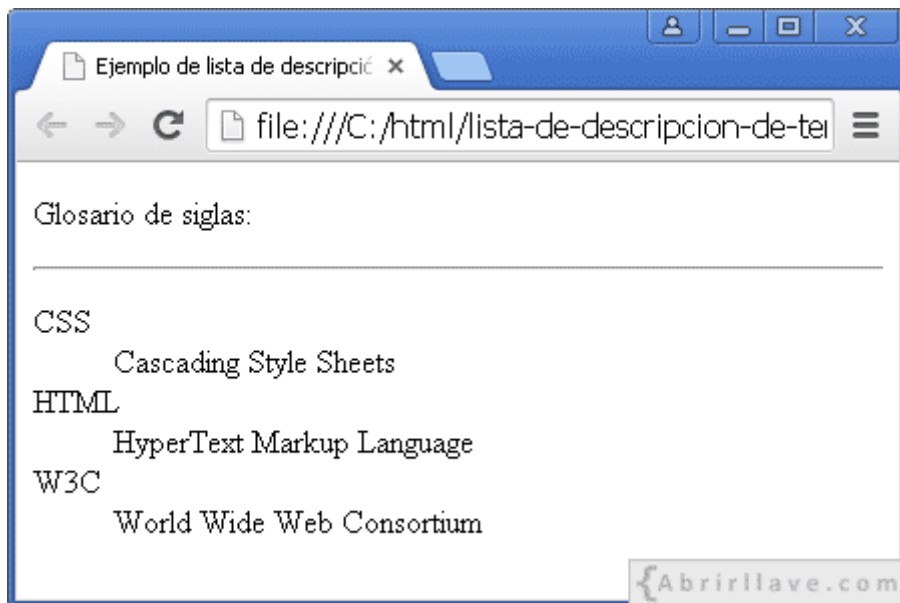
<!DOCTYPE html>
<html lang="es-ES">
  <head>
    <meta charset="utf-8">
    <title>Ejemplo de lista ordenada</title>
  </head>
  <body>
    <p>Lista de 5 animales ordenados de mayor a menor tamaño:</p>
    <ol>
      <li>Elefante</li>
      <li>Cebra</li>
      <li>Oveja</li>
      <li>Gato</li>
      <li>Ratón</li>
    </ol>
  </body>
</html>

```

En una lista ordenada, los elementos están ubicados de forma que, si se cambiasen de lugar, cambiaría también el significado de la lista.

Lista de descripción de términos - Elementos "dl", "dt" y "dd"

EJEMPLO Para mostrar un glosario de siglas:



Se puede escribir el siguiente código ("*lista-de-descripcion-de-terminos.html*"):

```

<!DOCTYPE html>
<html lang="es-ES">
  <head>
    <meta charset="utf-8">
    <title>Ejemplo de lista de descripción de términos</title>
  </head>
  <body>
    <p>Glosario de siglas:</p>
    <hr>
    <dl>
      <dt>CSS</dt>
      <dd>Cascading Style Sheets</dd>
      <dt>HTML</dt>
      <dd>HyperText Markup Language</dd>
      <dt>W3C</dt>
      <dd>World Wide Web Consortium</dd>
    </dl>
  </body>
</html>

```

Fíjese que, el elemento “dl” alberga a los grupos “término-descripción” (elemento “dt” - elemento “dd”).

Ejercicio resuelto

- [Listas de enlaces](#)

2.1.10. Imágenes

Para visualizar una imagen en un documento HTML se puede utilizar el elemento “img”.

Elemento “img”

EJEMPLO Si en pantalla se quiere ver, por ejemplo, la siguiente imagen con el logo de *Abrirllave*:



Se puede escribir (*“una-imagen.html”*):

```
<!DOCTYPE html>
<html lang="es-ES">
  <head>
    <meta charset="utf-8">
    <title>Ejemplo de una imagen</title>
  </head>
  <body>
    
  </body>
</html>
```

Como se puede observar, el elemento “img” no necesita la etiqueta de cierre () ya que no tiene contenido. Ahora bien, véase que en “img” se han escrito varios atributos.

Atributo **src**

El atributo **src** sirve para indicar la ubicación de la imagen que se quiere mostrar en el navegador web. En este caso, la imagen debe estar en la misma carpeta que el archivo *“una-imagen.html”*.

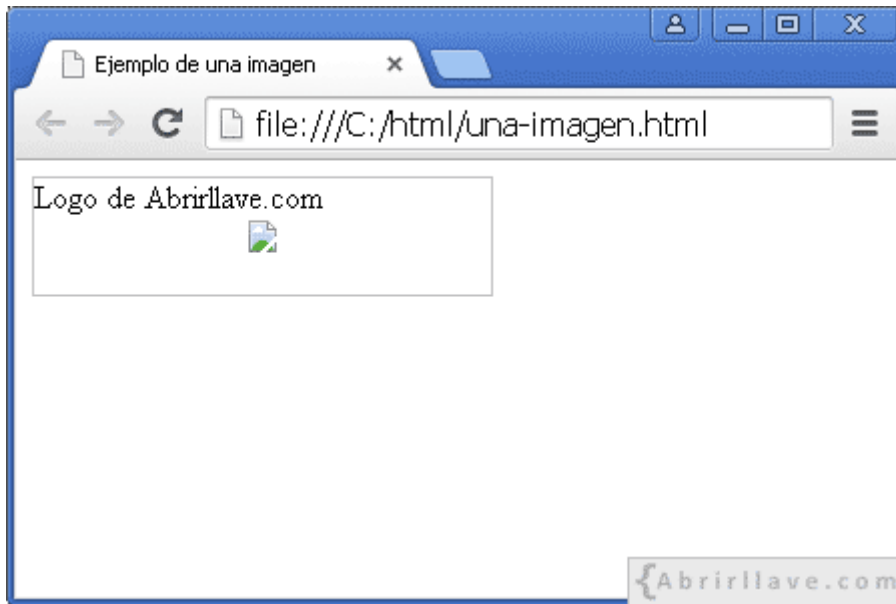
Atributos **width** y **height**

width y **height** permiten especificar, respectivamente, la anchura y la altura de la imagen en píxeles.

Atributo **alt**

El atributo **alt** sirve para indicar un texto que se mostrará en pantalla en el caso de que la imagen no se encontrara.

EJEMPLO Si en *“una-imagen.html”*, en vez de **src="logo-abrirllave.gif"**, por error se hubiese escrito **src="logoabrirllave.gif"**, al visualizar el documento HTML se vería algo parecido a:



Ejercicio resuelto

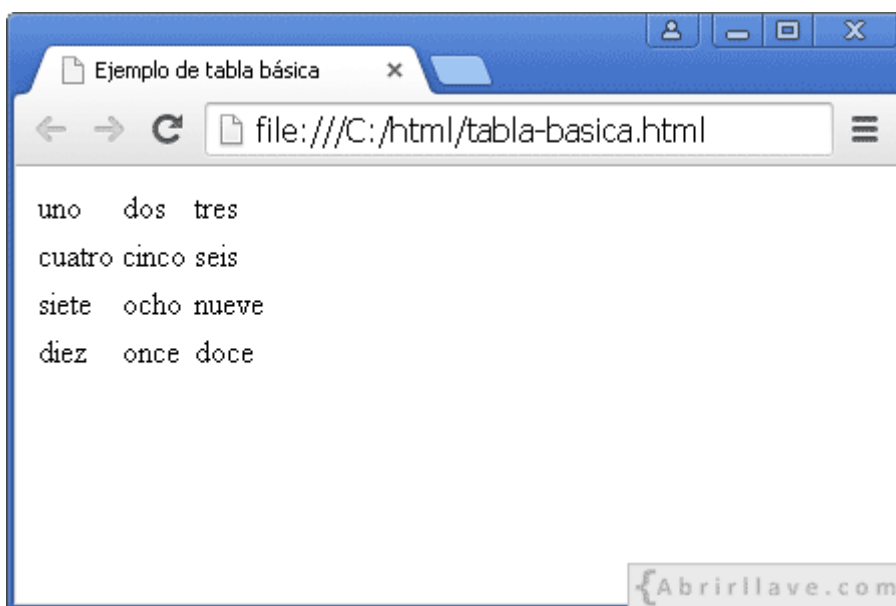
- [Receta de patatas fritas](#)

2.1.11. Tablas

En HTML se pueden representar datos en tablas.

Tabla básica - Elementos "table", "tr" y "td"

EJEMPLO En la siguiente tabla se muestran 12 datos:



Para ello, el código puede ser ("**tabla-basica.html**"):

```

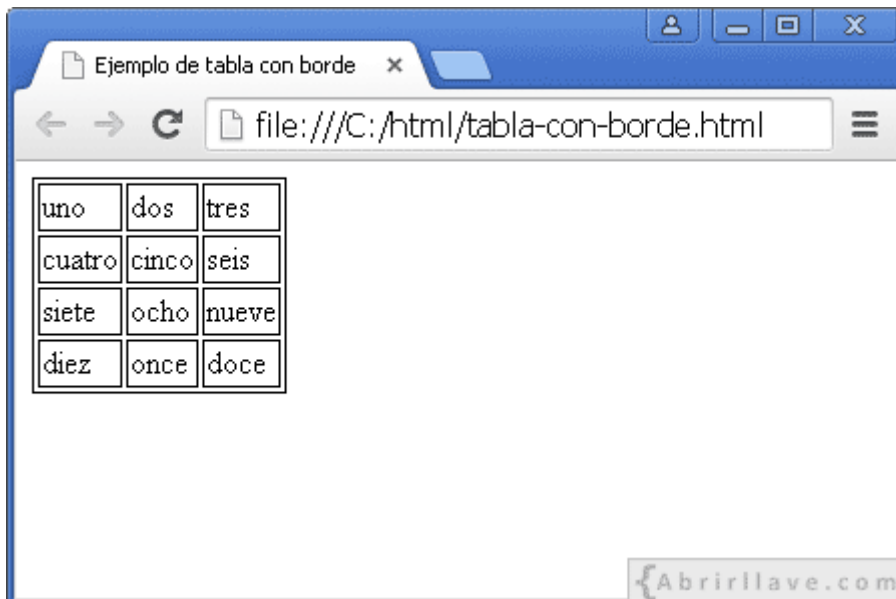
<!DOCTYPE html>
<html lang="es-ES">
  <head>
    <meta charset="utf-8">
    <title>Ejemplo de tabla básica</title>
  </head>
  <body>
    <table>
      <tr>
        <td>uno</td>
        <td>dos</td>
        <td>tres</td>
      </tr>
      <tr>
        <td>cuatro</td>
        <td>cinco</td>
        <td>seis</td>
      </tr>
      <tr>
        <td>siete</td>
        <td>ocho</td>
        <td>nueve</td>
      </tr>
      <tr>
        <td>diez</td>
        <td>once</td>
        <td>doce</td>
      </tr>
    </table>
  </body>
</html>

```

Obsérvese que, cada fila de la tabla se representa por medio de un elemento “tr” y, dentro de cada fila, se ha escrito cada uno de los datos dentro de un “td”.

Tabla con bordes - Propiedad CSS **border**

EJEMPLO Por defecto, las tablas se muestran sin bordes. No obstante, para ver por ejemplo:

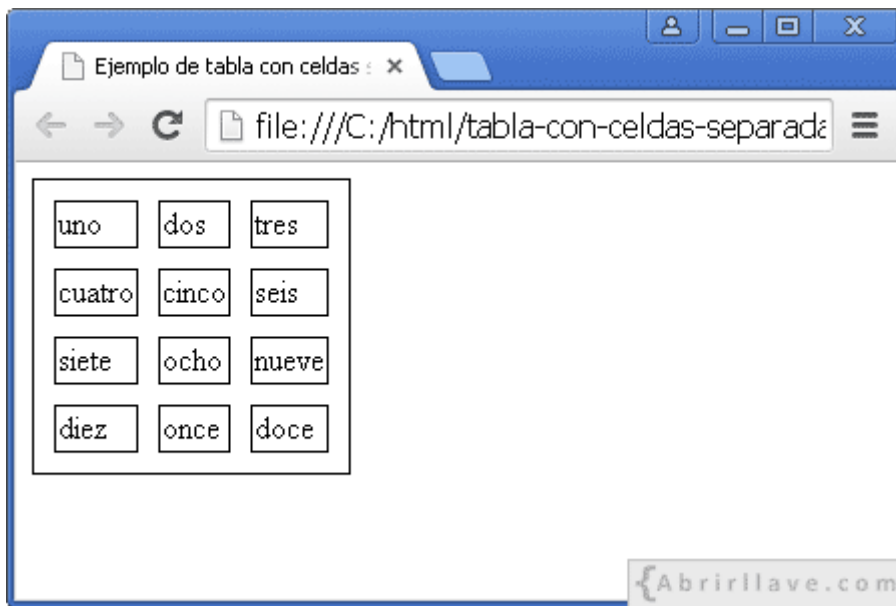


En "**tabla-con-bordes.html**" se ha definido la siguiente regla de estilo:

```
table, td {
    border:1px solid black;
}
```

Tabla con celdas separadas - Propiedad CSS **border-spacing**

EJEMPLO Si se quiere separar el espacio que existe entre celdas, por ejemplo 10 píxeles:

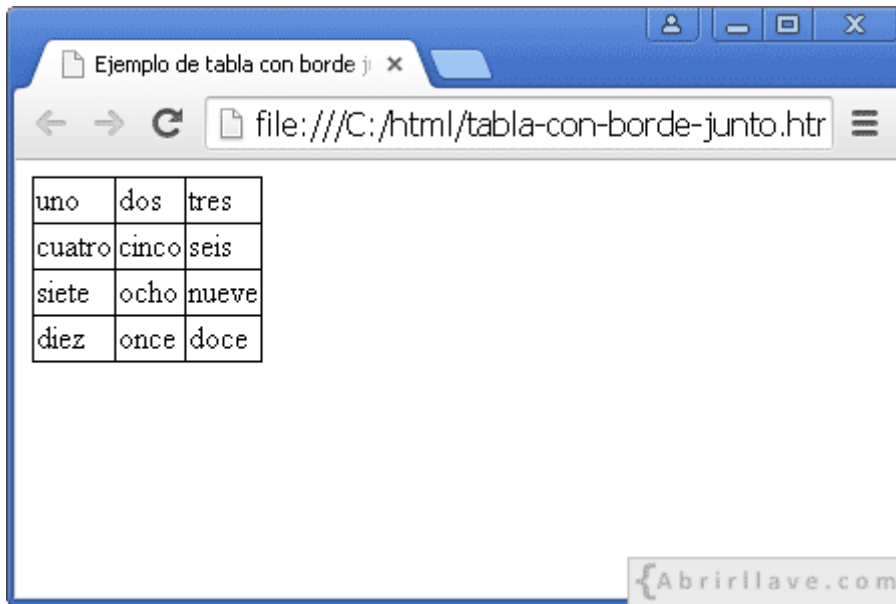


Se puede utilizar la propiedad CSS **border-spacing** ("**tabla-con-celdas-separadas.html**"):

```
table, td {
    border:1px solid black;
}
table {
    border-spacing:10px;
}
```

Tabla con borde junto - Propiedad CSS **border-collapse**

EJEMPLO Para juntar las líneas del borde de una tabla, viéndose en pantalla:

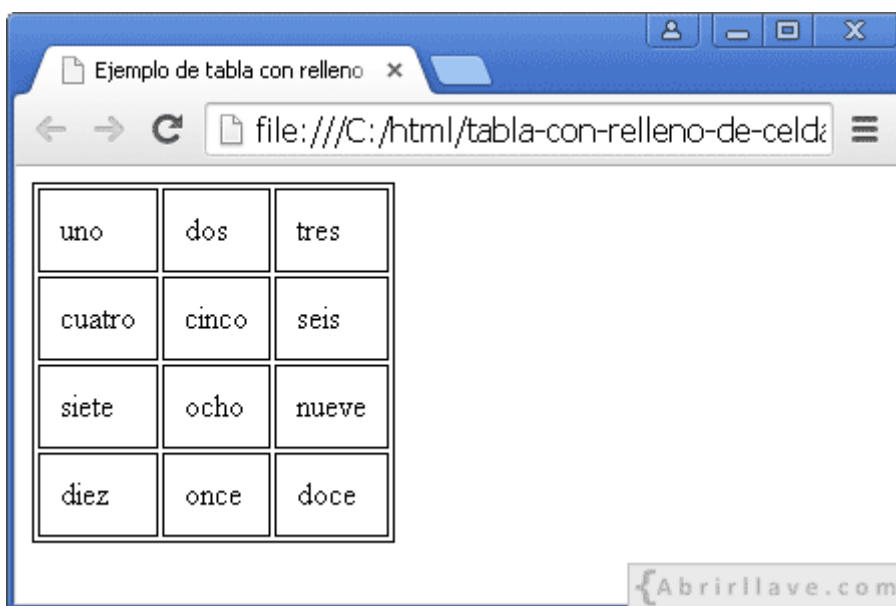


Se puede escribir ("**tabla-con-borde-junto.html**"):

```
table, td {
    border:1px solid black;
}
table {
    border-collapse:collapse;
}
```

Tabla con relleno de celdas - Propiedad CSS **padding**

EJEMPLO Si se desea separar el contenido de las celdas de una tabla con sus respectivos bordes, por ejemplo 10 píxeles:

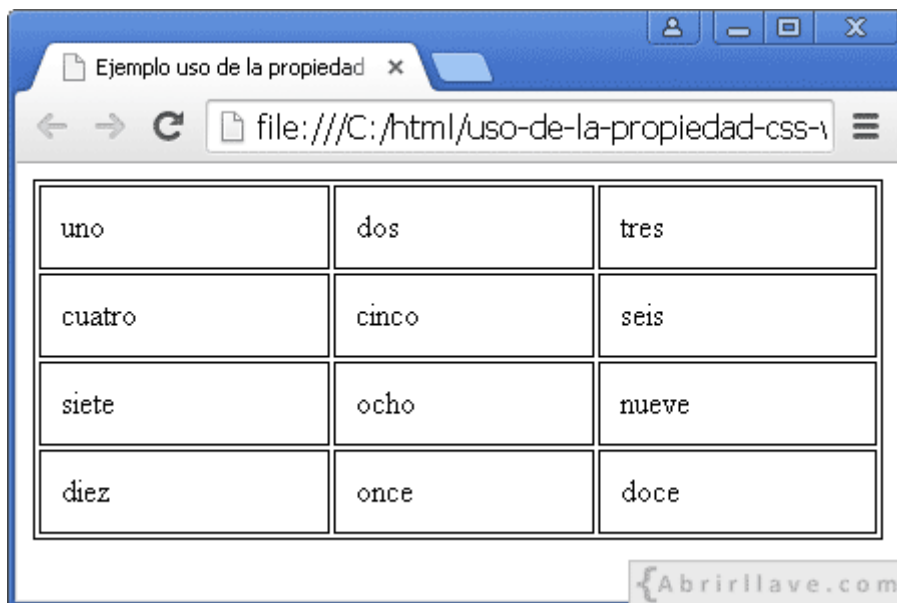


Se puede utilizar la propiedad CSS **padding** ("**tabla-con-relleno-de-celdas.html**"):

```
table, td {
    border:1px solid black;
}
td {
    padding:10px;
}
```

Ancho de una tabla - Propiedad **width**

EJEMPLO La propiedad CSS **width** permite definir el ancho de una tabla. Por ejemplo, para ver:

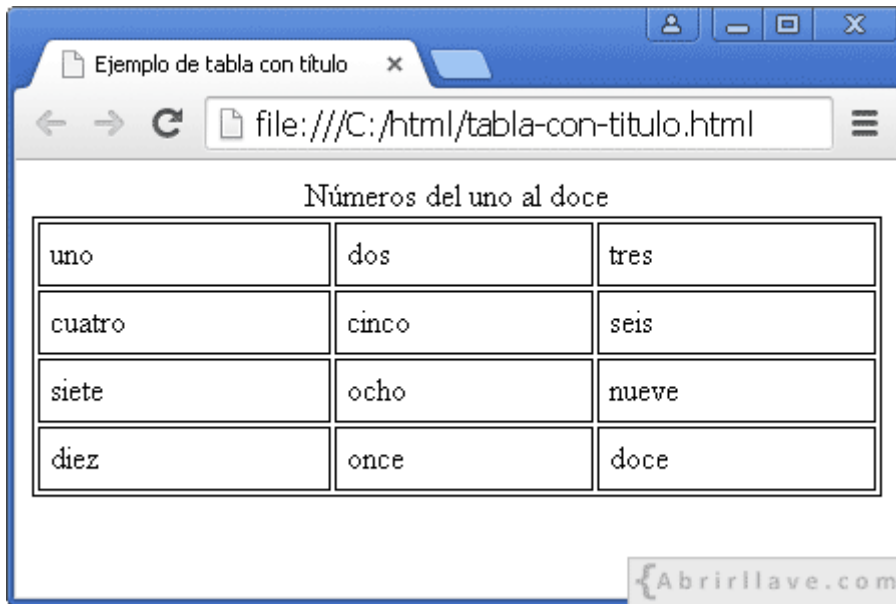


Se puede escribir ("*uso-de-la-propiedad-css-width-en-una-tabla.html*"):

```
table, td {
    border:1px solid black;
}
table {
    width:100%;
}
td {
    padding:10px;
}
```

Título de una tabla - Elemento "caption"

EJEMPLO A una tabla se le puede indicar un título (o leyenda). Por ejemplo, la siguiente tabla lleva por título "*Números del uno al doce*":



Para ello, se ha hecho uso del elemento "caption". Véase "[tabla-con-titulo.html](#)".

```
<!DOCTYPE html>
<html lang="es-ES">
  <head>
    <meta charset="utf-8">
    <title>Ejemplo de tabla con título</title>
    <style>
      table, td {
        border:1px solid black;
      }
      table {
        width:100%;
      }
      td {
        padding:5px;
      }
    </style>
  </head>
  <body>
    <table>
      <caption>Números del uno al doce</caption>
      <tr>
        <td>uno</td>
        <td>dos</td>
        <td>tres</td>
      </tr>
      <tr>
        <td>cuatro</td>
        <td>cinco</td>
        <td>seis</td>
      </tr>
      <tr>
        <td>siete</td>
        <td>ocho</td>
        <td>nueve</td>
      </tr>
    </table>
```

```

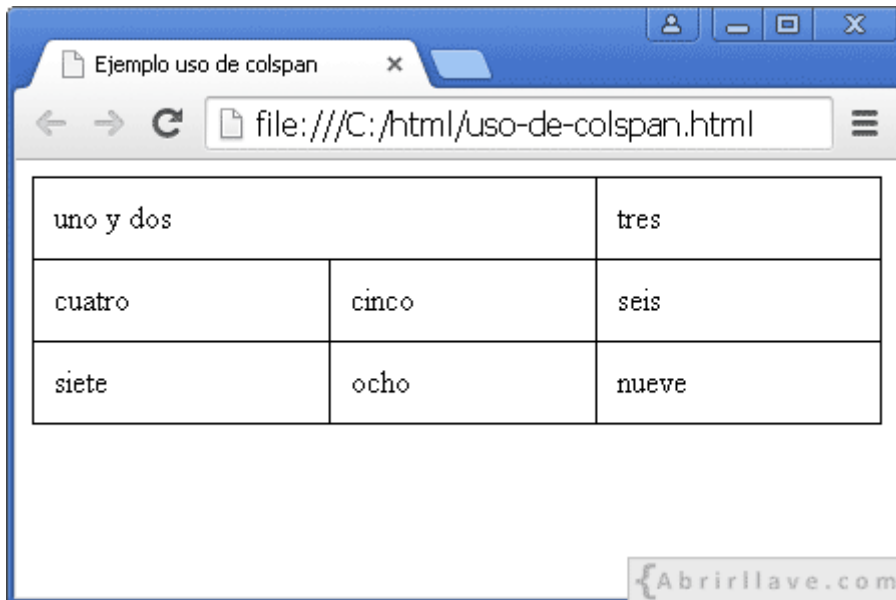
        <tr>
            <td>diez</td>
            <td>once</td>
            <td>doce</td>
        </tr>
    </table>
</body>
</html>

```

El elemento "caption" tiene que escribirse justo después de la etiqueta **<table>**.

Una celda puede abarcar varias columnas de una tabla - Atributo **colspan**

EJEMPLO Con el atributo **colspan** se puede indicar que una celda abarque más de una columna:



Véase en el código del archivo **"uso-de-colspan.html"** que, al primer elemento "td" de la primera fila, se le ha indicado mediante **colspan="2"** que debe abarcar dos columnas de la tabla:

```

<!DOCTYPE html>
<html lang="es-ES">
    <head>
        <meta charset="utf-8">
        <title>Ejemplo uso de colspan</title>
        <style>
            table, td {
                border:1px solid black;
            }
            table {
                border-collapse:collapse;
                width:100%;
            }
            td {
                padding:10px;
            }
        </style>
    </head>
    <body>
        <table>
            <tr>
                <td colspan="2">uno y dos</td>
                <td>tres</td>
            </tr>
            <tr>
                <td>cuatro</td>
                <td>cinco</td>
                <td>seis</td>
            </tr>
            <tr>
                <td>siete</td>
                <td>ocho</td>
                <td>nueve</td>
            </tr>
        </table>
    </body>
</html>

```

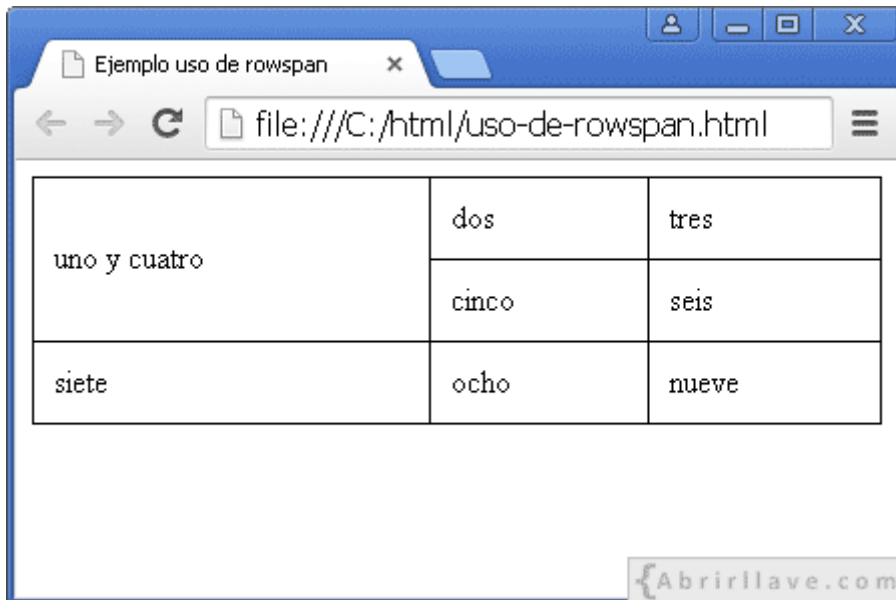
```

    }
</style>
</head>
<body>
  <table>
    <tr>
      <td colspan="2">uno y dos</td>
      <td>tres</td>
    </tr>
    <tr>
      <td>cuatro</td>
      <td>cinco</td>
      <td>seis</td>
    </tr>
    <tr>
      <td>siete</td>
      <td>ocho</td>
      <td>nueve</td>
    </tr>
  </table>
</body>
</html>

```

Una celda puede abarcar varias filas de una tabla - Atributo **rowspan**

EJEMPLO Con el atributo **rowspan** se puede indicar que una celda abarque más de una fila:



uno y cuatro	dos	tres
uno y cuatro	cinco	seis
siete	ocho	nueve

Fíjese en el archivo **“uso-de-rowspan.html”** que, al primer elemento **“td”** de la primera fila, se le ha indicado mediante **rowspan="2"** que debe abarcar dos filas de la tabla:

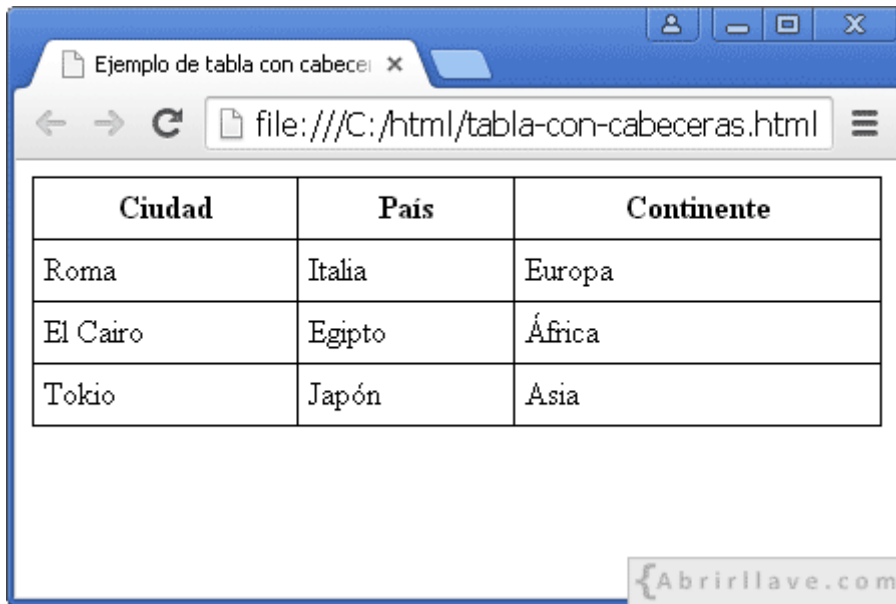
```

<!DOCTYPE html>
<html lang="es-ES">
  <head>
    <meta charset="utf-8">
    <title>Ejemplo uso de rowspan</title>
    <style>
      table, td {
        border:1px solid black;
      }
      table {
        border-collapse:collapse;
        width:100%;
      }
      td {
        padding:10px;
      }
    </style>
  </head>
  <body>
    <table>
      <tr>
        <td rowspan="2">uno y cuatro</td>
        <td>dos</td>
        <td>tres</td>
      </tr>
      <tr>
        <td>cinco</td>
        <td>seis</td>
      </tr>
      <tr>
        <td>siete</td>
        <td>ocho</td>
        <td>nueve</td>
      </tr>
    </table>
  </body>
</html>

```

Cabeceras de una tabla - Elemento "th"

EJEMPLO La siguiente tabla tiene tres celdas cabecera, las cuales han sido definidas con el elemento "th":



Ciudad	País	Continente
Roma	Italia	Europa
El Cairo	Egipto	África
Tokio	Japón	Asia

Obsérvese que, por defecto, el contenido de un elemento "th" se visualiza en negrita y centrado ("**tabla-con-cabeceras.html**").

```
<!DOCTYPE html>
<html lang="es-ES">
  <head>
    <meta charset="utf-8">
    <title>Ejemplo de tabla con cabeceras</title>
    <style>
      table, td, th {
        border:1px solid black;
      }
      table {
        border-collapse:collapse;
        width:100%;
      }
      td, th {
        padding:5px;
      }
    </style>
  </head>
  <body>
    <table>
      <tr>
        <th>Ciudad</th>
        <th>País</th>
        <th>Continente</th>
      </tr>
      <tr>
        <td>Roma</td>
        <td>Italia</td>
        <td>Europa</td>
      </tr>
      <tr>
        <td>El Cairo</td>
        <td>Egipto</td>
```

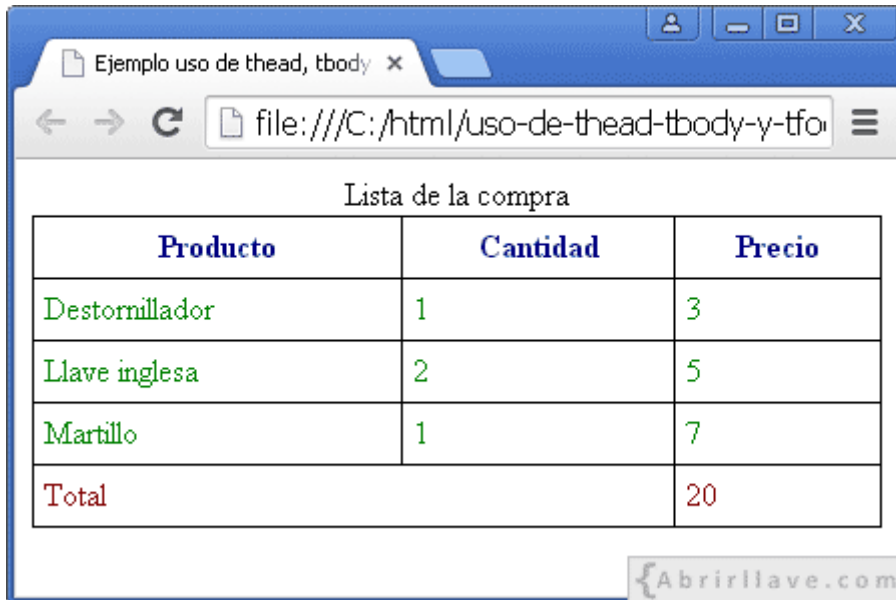
```

        <td>África</td>
    </tr>
    <tr>
        <td>Tokio</td>
        <td>Japón</td>
        <td>Asia</td>
    </tr>
</table>
</body>
</html>

```

Estructura de una tabla - Elementos "thead", "tbody" y "tfoot"

EJEMPLO Los elementos "thead", "tbody" y "tfoot" permiten estructurar los datos de una tabla y que sobre ellos se apliquen estilos distintos. Véase por ejemplo:



The screenshot shows a web browser window with the title 'Ejemplo uso de thead, tbody y tfoot'. The address bar shows the file path 'file:///C:/html/uso-de-thead-tbody-y-tfo'. The main content area displays a table titled 'Lista de la compra'. The table has three columns: 'Producto', 'Cantidad', and 'Precio'. The rows are: 'Destornillador' (1, 3), 'Llave inglesa' (2, 5), 'Martillo' (1, 7), and a 'Total' row (20). The 'Total' row is highlighted in red. The browser's status bar at the bottom shows 'Abrirllave.com'.

Producto	Cantidad	Precio
Destornillador	1	3
Llave inglesa	2	5
Martillo	1	7
Total		20

Para ello, se ha escrito ("*uso-de-thead-tbody-y-tfoot.html*"):

```

<!DOCTYPE html>
<html lang="es-ES">
    <head>
        <meta charset="utf-8">
        <title>Ejemplo uso de thead, tbody y tfoot</title>
        <style>
            table, td, th {
                border:1px solid black;
            }
            table {
                border-collapse:collapse;
                width:100%;
            }
            td, th {
                padding:5px;
            }
        </style>
    </head>
    <body>
        <table>
            <thead>
                <tr>
                    <th>Producto</th>
                    <th>Cantidad</th>
                    <th>Precio</th>
                </tr>
            </thead>
            <tbody>
                <tr>
                    <td>Destornillador</td>
                    <td>1</td>
                    <td>3</td>
                </tr>
                <tr>
                    <td>Llave inglesa</td>
                    <td>2</td>
                    <td>5</td>
                </tr>
                <tr>
                    <td>Martillo</td>
                    <td>1</td>
                    <td>7</td>
                </tr>
                <tr>
                    <td>Total</td>
                    <td></td>
                    <td>20</td>
                </tr>
            </tbody>
        </table>
    </body>
</html>

```



```

    thead {
        color:navy;
    }
    tbody {
        color:green;
    }
    tfoot {
        color:maroon;
    }
</style>
</head>
<body>
<table>
    <caption>Lista de la compra</caption>
    <thead>
        <tr>
            <th>Producto</th>
            <th>Cantidad</th>
            <th>Precio</th>
        </tr>
    </thead>
    <tfoot>
        <tr>
            <td colspan="2">Total</td>
            <td>20</td>
        </tr>
    </tfoot>
    <tbody>
        <tr>
            <td>Destornillador</td>
            <td>1</td>
            <td>3</td>
        </tr>
        <tr>
            <td>Llave inglesa</td>
            <td>2</td>
            <td>5</td>
        </tr>
        <tr>
            <td>Martillo</td>
            <td>1</td>
            <td>7</td>
        </tr>
    </tbody>
</table>
</body>
</html>

```

Los elementos “thead”, “tbody” y “tfoot” pueden agrupar, respectivamente:

- una o más filas de cabeceras,
- una o más filas de datos del cuerpo de la tabla y
- una o más filas de datos del pie de la tabla.

Por otro lado, hay que saber que, en una tabla puede aparecer más de un elemento “tbody”.

Además, hay que tener en cuenta que, el elemento "tfoot" debe escribirse después del "thead" y antes de todos los "tbody" presentes en la tabla.

Ejercicio resuelto

- [Grupo de música \(Queen\)](#)

2.1.12. Divisiones

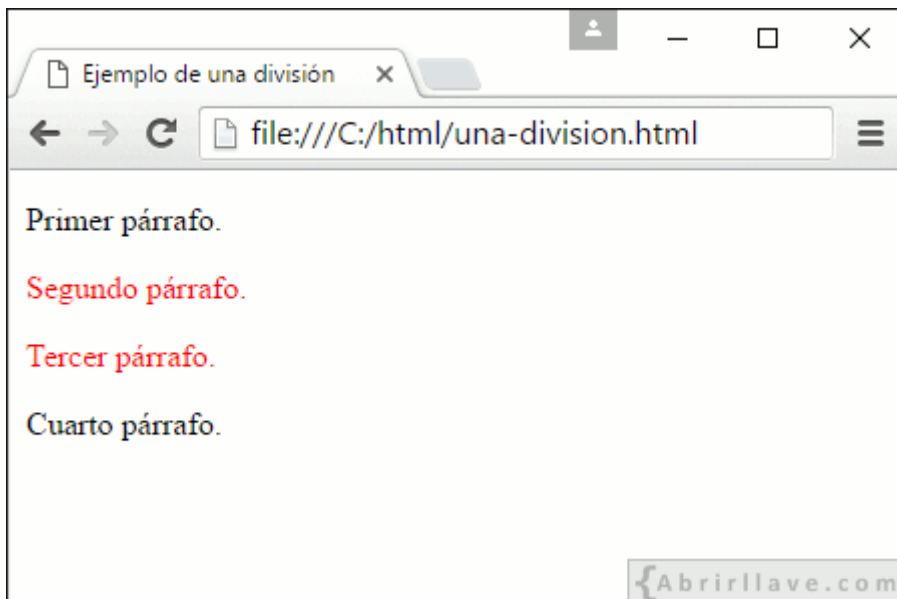
En un documento HTML el elemento "div" permite crear divisiones, también llamadas secciones o zonas. Las divisiones se utilizan para agrupar elementos y aplicarles estilos.

Elemento "div"

EJEMPLO En el siguiente documento ("*una-division.html*") el segundo y tercer párrafo están agrupados dentro de un elemento "div" donde se ha establecido que dichos párrafos deben mostrarse de color rojo:

```
<!DOCTYPE html>
<html lang="es-ES">
  <head>
    <meta charset="utf-8">
    <title>Ejemplo de una división</title>
  </head>
  <body>
    <p>Primer párrafo.</p>
    <div style="color:red">
      <p>Segundo párrafo.</p>
      <p>Tercer párrafo.</p>
    </div>
    <p>Cuarto párrafo.</p>
  </body>
</html>
```

Al visualizarlo en pantalla, se verá algo similar a:

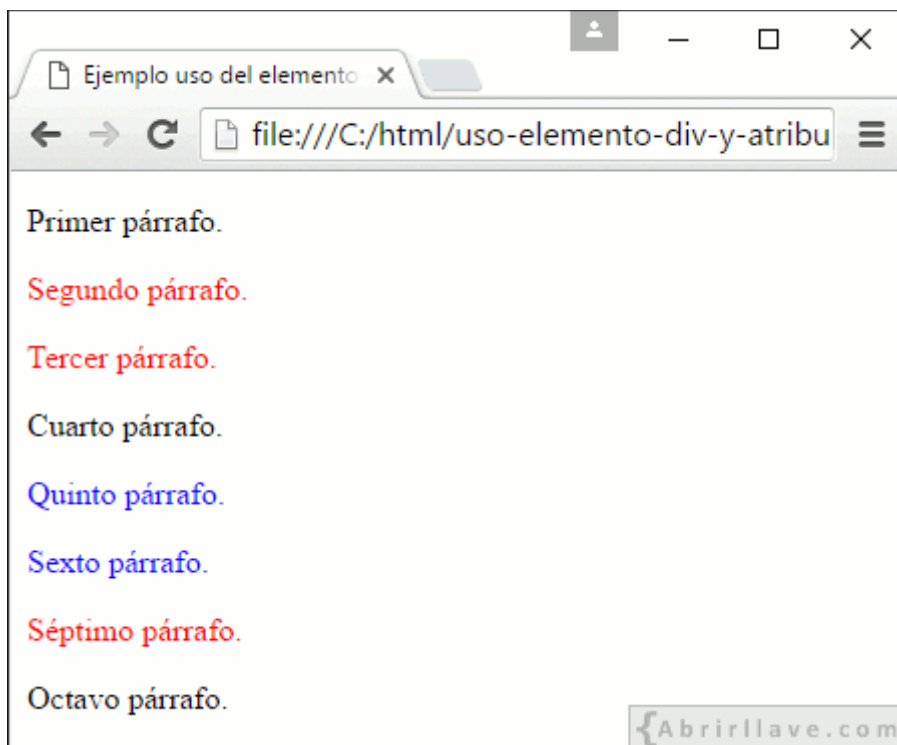


Elemento "div" y atributo **class**

EJEMPLO En el siguiente documento HTML ("*uso-elemento-div-y-atributo-class.html*") se han escrito tres divisiones haciendo uso del atributo **class**. Obsérvese que, en dos de dichas divisiones el valor del atributo **class** es "**rojo**" y en la otra "**azul**".

```
<!DOCTYPE html>
<html lang="es-ES">
  <head>
    <meta charset="utf-8">
    <title>Ejemplo uso del elemento div y del atributo class</title>
    <style>
      .rojo {color:red;}
      .azul {color:blue;}
    </style>
  </head>
  <body>
    <p>Primer párrafo.</p>
    <div class="rojo">
      <p>Segundo párrafo.</p>
      <p>Tercer párrafo.</p>
    </div>
    <p>Cuarto párrafo.</p>
    <div class="azul">
      <p>Quinto párrafo.</p>
      <p>Sexto párrafo.</p>
    </div>
    <div class="rojo">
      <p>Séptimo párrafo.</p>
    </div>
    <p>Octavo párrafo.</p>
  </body>
</html>
```

En un navegador se visualizará:



Elemento "div" y atributo id

EJEMPLO En el código del siguiente documento ("*uso-elemento-div-y-atributo-id.html*") se han especificado cuatro divisiones (**cabecera**, **contenido**, **menu** y **pie**) contenidas dentro de otra división llamada **contenedor**.

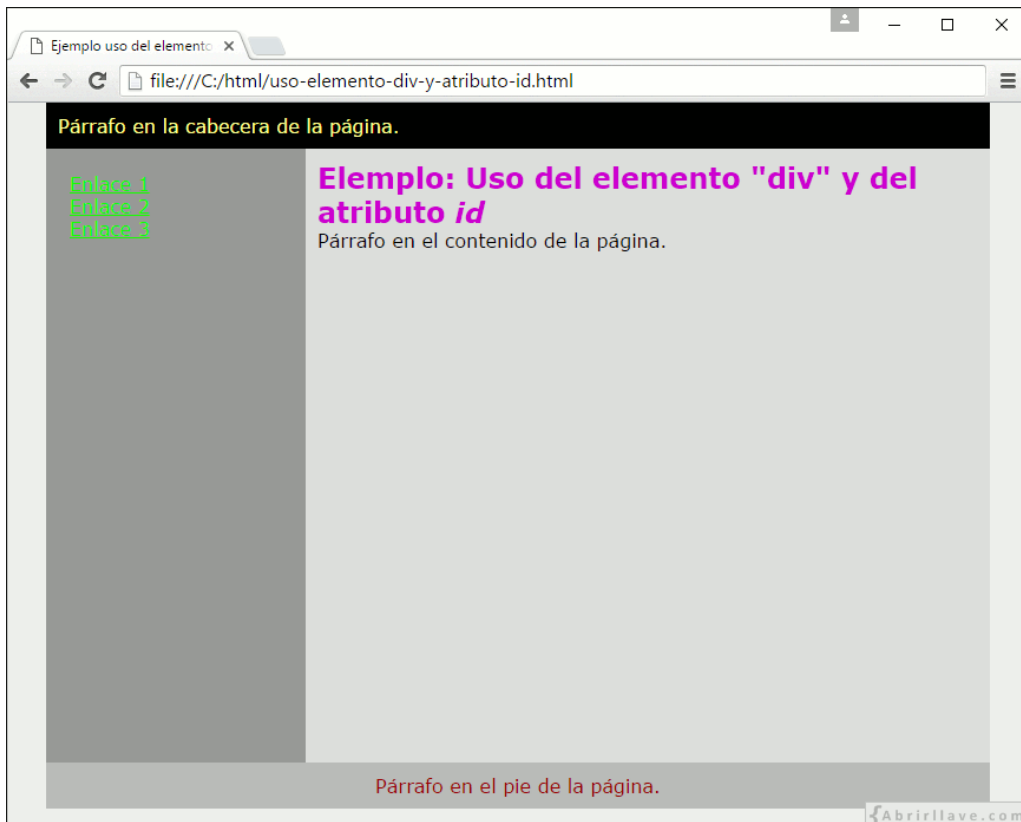
```
<!DOCTYPE html>
<html lang="es-ES">
  <head>
    <meta charset="utf-8">
    <title>Ejemplo uso del elemento div y del atributo id</title>
    <style>
      * {
        margin:0;
        padding:0;
      }
      a:link, a:visited, a:hover, a:active {
        color:#0f0;
        font-size:16px;
      }
      body {
        background:#eee;
        font-family:verdana;
      }
      h1 {
        color:#c0c;
        font-size:24px;
      }
      p {
        font-size:16px;
      }
      ul {
        list-style-type:none;
      }
      #cabecera {
        color:#ff9;
        background-color:#000;
        padding:10px;
      }
      #contenedor {
        margin:0 auto;
        width:800px;
      }
      #contenido {
        background-color:#ddd;
        float:left;
        height:500px;
        padding:10px;
        width:560px;
      }
      #menu {
        background-color:#999;
        float:left;
        height:480px;
        padding:20px;
        width:180px;
      }
    </style>
  </head>
  <body>
    <div id="cabecera">
      <h1>Ejemplo uso del elemento div y del atributo id</h1>
    </div>
    <div id="contenedor">
      <div id="contenido">
        <p>Contenido principal</p>
      </div>
      <div id="menu">
        <ul>
          <li>Menu item 1</li>
          <li>Menu item 2</li>
          <li>Menu item 3</li>
          <li>Menu item 4</li>
          <li>Menu item 5</li>
        </ul>
      </div>
    </div>
  </body>
</html>
```

```

#pie {
    background-color:#bbb;
    clear:both;
    color:#900;
    padding:10px;
    text-align:center;
}
</style>
</head>
<body>
  <div id="contenedor">
    <div id="cabecera">
      <p>Párrafo en la cabecera de la página.</p>
    </div>
    <div id="menu">
      <ul>
        <li><a href="enlace-1.html">Enlace 1</a></li>
        <li><a href="enlace-2.html">Enlace 2</a></li>
        <li><a href="enlace-3.html">Enlace 3</a></li>
      </ul>
    </div>
    <div id="contenido">
      <h1>Ejemplo: Uso del elemento &quot;div&quot; y del atributo <i>id</i></h1>
      <p>Párrafo en el contenido de la página.</p>
    </div>
    <div id="pie">
      <p>Párrafo en el pie de la página.</p>
    </div>
  </div>
</body>
</html>

```

En pantalla se mostrará algo parecido a:



Ejercicio resuelto

- [Diseño fluido con dos columnas](#)

2.1.13. Formularios

En un documento HTML el elemento "form" permite crear formularios. En ellos, los usuarios pueden introducir datos (información) para ser enviados y procesados en un servidor.

Formulario básico - Elementos "form" e "input"

EJEMPLO Para crear un formulario sencillo:

The screenshot shows a web browser window with a single tab titled 'Ejemplo de formulario básico'. The address bar displays 'file:///C:/html/formulario-basico.html'. The page content includes a form with the following elements: a label 'Nombre:' followed by a text input field, a label 'Edad:' followed by a number input field, and a submit button labeled 'Enviar datos'. A watermark 'Abrirllave.com' is visible in the bottom right corner of the browser window.

Se puede escribir ("*formulario-basico.html*"):

```
<!DOCTYPE html>
<html lang="es-ES">
  <head>
    <meta charset="utf-8">
    <title>Ejemplo de formulario básico</title>
  </head>
  <body>
    <form>
      Nombre:
      <br>
      <input type="text">
      <br>
      Edad:
      <br>
      <input type="number">
      <br><br>
      <input type="submit" value="Enviar datos">
    </form>
  </body>
</html>
```

Como se puede observar, las etiquetas `<form>` y `</form>` delimitan el formulario, el cual contiene en este ejemplo tres elementos "input" para que el usuario pueda:

- Introducir su nombre (dato de tipo texto, `type="text"`).
- Introducir su edad (dato de tipo número, `type="number"`).
- Enviar dichos datos, `type="submit"`.

Por otra parte, véase que al atributo `value` se le ha asignado el texto que se muestra en el botón de envío, "Enviar datos".

Atributo `action`

Cuando alguien hace clic en el botón de envío de datos de un formulario, habitualmente estos son enviados a otra página web para su procesamiento en un servidor. Para indicar la URL de dicha página, en el elemento "form" se utiliza el atributo `action`.

EJEMPLO Si el archivo "*procesar-datos.php*" fuese el encargado de procesar los datos de un formulario, se podría escribir:

```
<form action="procesar-datos.php">
```

Nota: para procesar los datos en un servidor, se utilizan lenguajes que pueden ejecutarse en un servidor, tales como: ASP, PHP, etc.

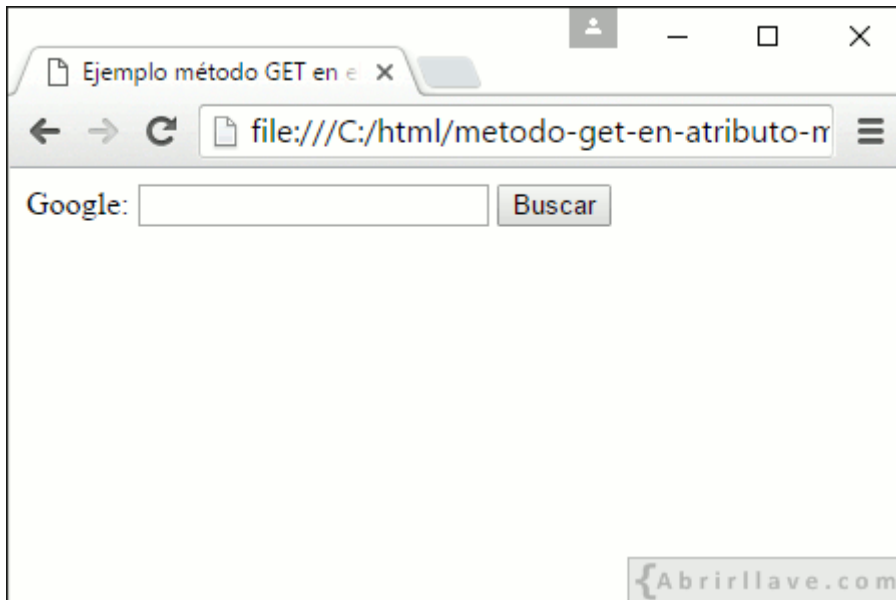
Atributo `method`

El atributo `method` del elemento "form" permite especificar el método HTTP que se va a utilizar para enviar los datos de un formulario. Pudiendo ser el método GET o el método POST.

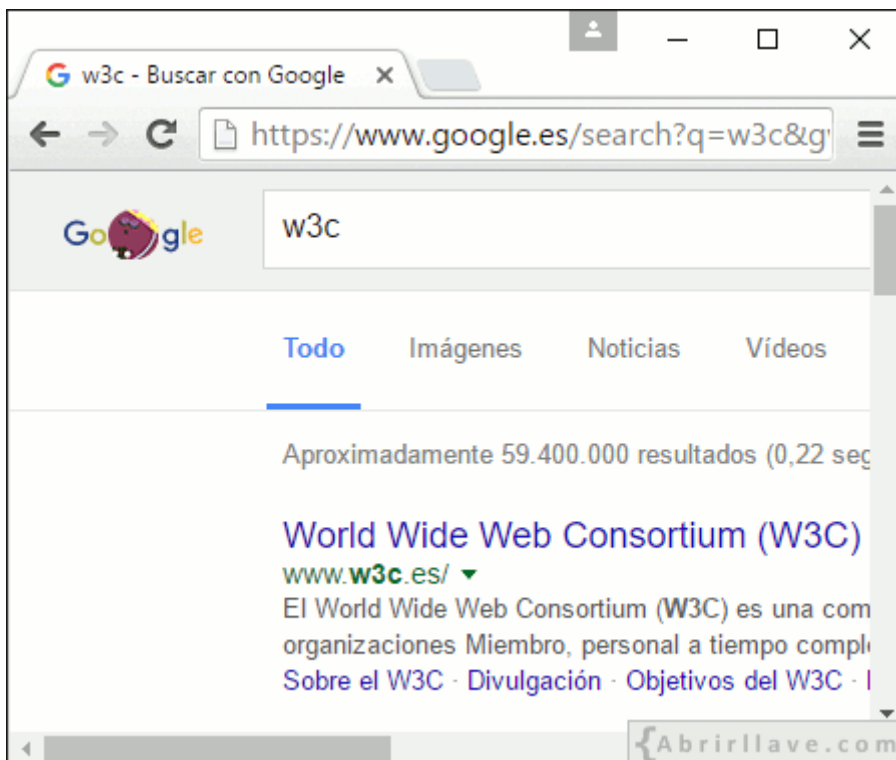
EJEMPLO El método GET se suele utilizar para enviar información (datos) que no sea sensible (contraseñas, teléfonos, direcciones de correo electrónico...). Por ejemplo, el formulario del siguiente documento ("*metodo-get-en-atributo-method.html*"), se puede utilizar para realizar una búsqueda en *Google*:

```
<!DOCTYPE html>
<html lang="es-ES">
  <head>
    <meta charset="utf-8">
    <title>Ejemplo método GET en el atributo method</title>
  </head>
  <body>
    <form action="http://www.google.com/search" method="get">
      Google:
      <input type="search" name="q">
      <input type="submit" value="Buscar">
    </form>
  </body>
</html>
```

Fíjese que, con el atributo **name** del elemento "input" se ha indicado el nombre (identificador) del dato que se va a enviar. Esto es necesario hacerlo con todos los datos que se envíen —en este caso sólo **q**— para que la página de destino que reciba dichos datos pueda procesarlos. En pantalla se verá algo parecido a:



Al hacer clic en el botón de "Buscar" del formulario, en la URL de la página de destino se mostrará el dato que se hubiese introducido. Por ejemplo, si dicho dato fuese **w3c**, se visualizaría algo similar a:



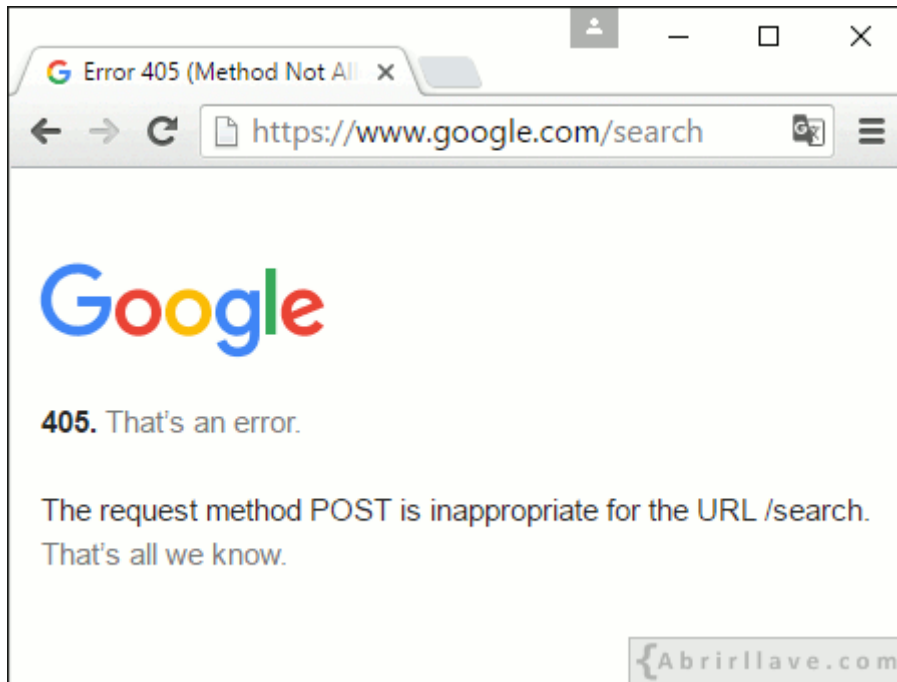
Por defecto se aplica el método GET. Por tanto, el mismo resultado se habría obtenido escribiendo:


```
<form action="http://www.google.com/search">
```

Ahora bien, si se hubiese indicado el método POST:

```
<form action="http://www.google.com/search" method="post">
```

En pantalla se habría obtenido un mensaje de error:



EJEMPLO En el formulario del siguiente documento HTML (*"metodo-post-en-atributo-method.html"*) se ha indicado utilizar el método POST para enviar los datos (**usuario** y **clave**) a la página *"procesar-datos.php"*:

```
<!DOCTYPE html>
<html lang="es-ES">
  <head>
    <meta charset="utf-8">
    <title>Ejemplo método POST en el atributo method</title>
  </head>
  <body>
    <form action="procesar-datos.php" method="post">
      Nombre de usuario:
      <br>
      <input type="text" name="usuario">
      <br>
      Contraseña:
      <br>
      <input type="password" name="clave">
      <br><br>
      <input type="submit" value="Enviar datos">
    </form>
  </body>
</html>
```

Si, por ejemplo, el usuario “iker” escribiese la contraseña “TdidA822”:

Se puede comprobar que, al utilizar el método POST –a diferencia del método GET– en la URL de la página de destino no se mostrarán los datos enviados.

Controles de un formulario - Elementos “button”, “input”, “select” y “textarea”

En un formulario, a los elementos que permiten la interacción del usuario (botones, campos de texto, etc.) se les denomina controles.

EJEMPLO El formulario del siguiente documento HTML contiene algunos de los controles más utilizados habitualmente (*“controles-mas-utilizados-en-formularios.html”*):

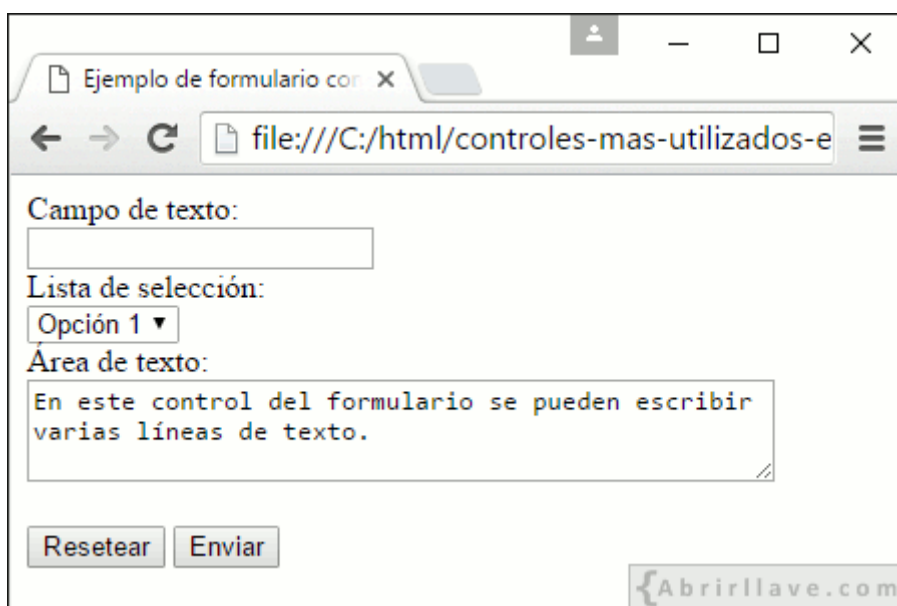
- “button”: representa a un botón.
- “input”: puede representar diferentes tipos de control (texto, contraseña, etc.) en función del valor que se asigne a su atributo **type**.
- “select”: define una lista de opciones, donde cada una de las opciones se representa con un elemento “option”.
- “textarea”: permite introducir varias líneas de texto.

```

<!DOCTYPE html>
<html lang="es-ES">
  <head>
    <meta charset="utf-8">
    <title>Ejemplo de formulario con algunos de los controles
    más utilizados</title>
  </head>
  <body>
    <form action="procesar-datos.php" method="post">
      Campo de texto:
      <br>
      <input type="text" name="campo_de_texto">
      <br>
      Lista de selección:
      <br>
      <select name="lista_de_seleccion">
        <option value="opcion1">Opción 1</option>
        <option value="opcion2">Opción 2</option>
        <option value="opcion3">Opción 3</option>
      </select>
      <br>
      Área de texto:
      <br>
      <textarea name="area_de_texto" rows="3" cols="50">En este
      control del formulario se pueden escribir varias líneas de
      texto.</textarea>
      <br><br>
      <button type="reset">Resetear</button>
      <button type="submit">Enviar</button>
    </form>
  </body>
</html>

```

En un navegador web se mostrará:



En lugar de los elementos "button", se pueden utilizar elementos "input" escribiendo:

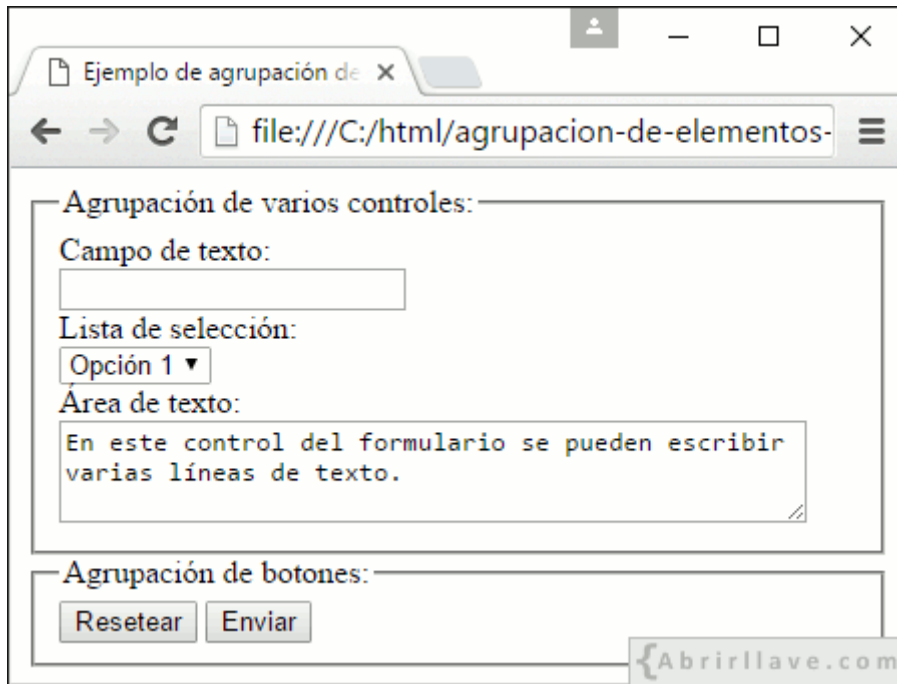
```
<input type="reset" value="Resetear">
<input type="submit" value="Enviar">
```

Agrupaciones de elementos "fieldset" y "legend"

EJEMPLO El elemento "fieldset" permite agrupar varios elementos de un formulario. En el siguiente documento se han definido dos grupos ("*agrupacion-de-elementos-en-un-formulario.html*"):

```
<!DOCTYPE html>
<html lang="es-ES">
  <head>
    <meta charset="utf-8">
    <title>Ejemplo de agrupación de controles en un
formulario</title>
  </head>
  <body>
    <form action="procesar-datos.php" method="post">
      <fieldset>
        <legend>Agrupación de varios controles:</legend>
        Campo de texto:
        <br>
        <input type="text" name="campo_de_texto">
        <br>
        Lista de selección:
        <br>
        <select name="lista_de_seleccion">
          <option value="opcion1">Opción 1</option>
          <option value="opcion2">Opción 2</option>
          <option value="opcion3">Opción 3</option>
        </select>
        <br>
        Área de texto:
        <br>
        <textarea name="area_de_texto" rows="3" cols="50">En
este control del formulario se pueden escribir varias líneas de
texto.</textarea>
      </fieldset>
      <fieldset>
        <legend>Agrupación de botones:</legend>
        <button type="reset">Resetear</button>
        <button type="submit">Enviar</button>
      </fieldset>
    </form>
  </body>
</html>
```

En pantalla se vería:



Obsérvese que, el elemento “legend” permite especificar una leyenda –o título– en cada una de las agrupaciones.

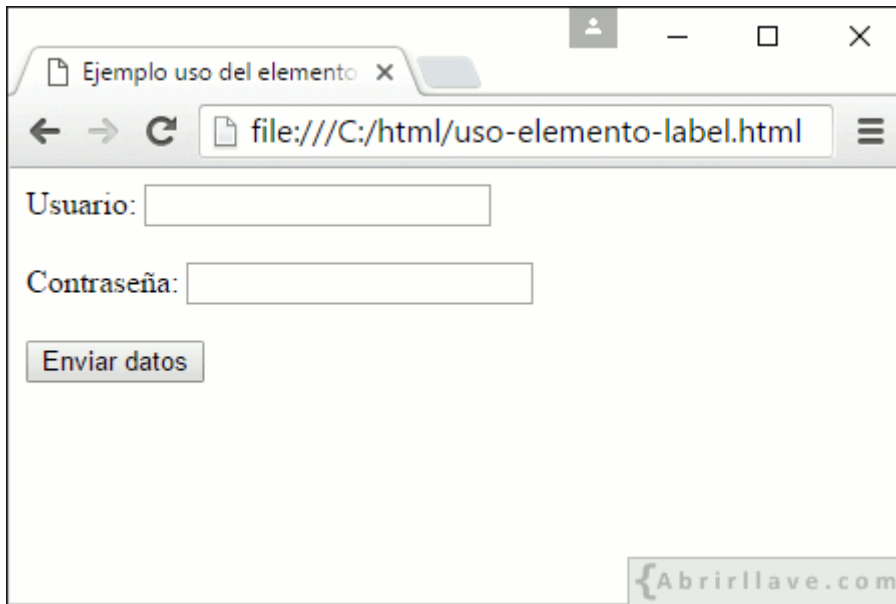
Elemento “label”

El elemento “label” permite definir etiquetas –o títulos– a los controles de un formulario, incrementando de esta forma su accesibilidad.

EJEMPLO En el siguiente documento HTML han sido etiquetados dos controles (“*uso-elemento-label.html*”):

```
<!DOCTYPE html>
<html lang="es-ES">
  <head>
    <meta charset="utf-8">
    <title>Ejemplo uso del elemento label</title>
  </head>
  <body>
    <form action="procesar-datos.php" method="post">
      <label>Usuario: <input type="text" name="usuario"></label>
      <br><br>
      <label>Contraseña: <input type="password" name="clave"></label>
      <br><br>
      <input type="submit" value="Enviar datos">
    </form>
  </body>
</html>
```

En pantalla se mostraría:



Fíjese que, ambos controles etiquetados se han escrito dentro de sendos elementos "label". No obstante, también se pueden etiquetar controles utilizando atributos **for** en los elementos "label" y asociándolos con atributos **id** de los controles (*"uso-elemento-label-y-atributo-for.html"*):

```
<!DOCTYPE html>
<html lang="es-ES">
  <head>
    <meta charset="utf-8">
    <title>Ejemplo uso del elemento label y del atributo for</title>
  </head>
  <body>
    <form action="procesar-datos.php" method="post">
      <label for="nombre_usuario">Usuario: </label>
      <input type="text" name="usuario" id="nombre_usuario">
      <br><br>
      <label for="clave_usuario">Contraseña: </label>
      <input type="password" name="clave" id="clave_usuario">
      <br><br>
      <input type="submit" value="Enviar datos">
    </form>
  </body>
</html>
```

En un navegador web, se puede comprobar que, al hacer clic en el título de un control, se activa dicho control. Por ejemplo, si en *Google Chrome* se visualiza el documento *"uso-elemento-label-y-atributo-for.html"*, al hacer clic sobre el texto "Contraseña: " el cursor se posicionará en el control cuyo **id** es **"clave_usuario"**:



Otros elementos: “datalist”, “meter”, “optgroup”, “output”, “progress”

Además de los elementos vistos en este apartado del tutorial, en un formulario también se puede hacer uso de otros elementos: “datalist”, “meter”, “optgroup”, “output”, “progress”.

Ejercicio resuelto

- [Notificar una incidencia](#)

2.1.14. Recursos de HTML

- **Tutorial de HTML**
<http://www.abrirllave.com/html/>
- **Ejemplos de HTML**
<http://www.abrirllave.com/html/ejemplos.php>
- **Ejercicios resueltos de HTML**
<http://www.abrirllave.com/html/ejercicios-resueltos.php>

Capítulo 3

Aplicación de los lenguajes de marcas a la sindicación de contenidos

Los contenidos de este capítulo están en desarrollo y serán incorporados al libro en una próxima edición.

Capítulo 4

Definición de esquemas y vocabularios en XML

Teoría y ejemplos

- *Apuntes de DTD*
<http://www.abrirllave.com/dtd/apuntes-de-dtd.php>
- *Apuntes de XSD (XML Schema)*
<http://www.abrirllave.com/xsd/apuntes-de-xsd.php>

Ejercicios

- *Ejercicios resueltos de DTD*
<http://www.abrirllave.com/dtd/ejercicios-resueltos.php>
 - *Ejercicios resueltos de XSD (XML Schema)*
<http://www.abrirllave.com/xsd/ejercicios-resueltos.php>
-

4.1. DTD

Contenidos del tutorial de DTD www.abrirllave.com/dtd/

[4.1.1. Qué es DTD](#)

[4.1.2. Declaración de tipo de documento](#)

[4.1.2.1. Documento XML asociado a una DTD interna](#)

[4.1.2.2. Documento XML asociado a una DTD externa](#)

DTD externa privada - **SYSTEM**

DTD externa pública - **PUBLIC**

[4.1.2.3. Cuándo utilizar una DTD interna o una DTD externa](#)

[4.1.2.4. Uso combinado de DTD interna y externa en un documento XML](#)

[4.1.3. Estructura de un documento XML](#)

[4.1.4. Declaración de elementos](#)

El contenido de un elemento puede ser texto - (**#PCDATA**)

Un elemento puede contener a otros elementos

Un elemento puede no contener contenido (estar vacío) - **EMPTY**

Un elemento puede definirse para contener contenido mixto - **ANY**

[4.1.4.1. Elementos vacíos - **EMPTY**](#)

Un elemento vacío puede tener atributos

[4.1.4.2. Elementos con cualquier contenido - **ANY**](#)

[4.1.4.3. Elementos con contenido de tipo texto - \(**#PCDATA**\)](#)

[4.1.4.4. Secuencias de elementos](#)

Elemento con varios hijos

Orden de los hijos de un elemento

[4.1.4.5. Cardinalidad de los elementos](#)

Operador de cardinalidad "+" (signo más)

Operador de cardinalidad "*" (asterisco)

Operador de cardinalidad "?" (interrogación)

[4.1.4.6. Elementos opcionales](#)

Operador de elección "|" (barra vertical)

Operador de elección "|" y operador "*"

Operador de elección "|" en una secuencia de elementos

Secuencia de elementos en una lista de opciones

#PCDATA en una lista de opciones permite contenido mixto

[4.1.5. Declaración de atributos](#)

Declaración de un atributo indicando un valor por defecto

Declaración de varios atributos en un elemento

[4.1.6. Tipos de declaración de atributos](#)

[4.1.6.1. Atributo obligatorio - **#REQUIRED**](#)

[4.1.6.2. Atributo opcional - **#IMPLIED**](#)

[4.1.6.3. Atributo con valor fijo - **#FIXED** valor](#)

[4.1.7. Tipos de atributos](#)

[4.1.7.1. Atributos de tipo CDATA](#)

[4.1.7.2. Atributos de tipo enumerado](#)

[4.1.7.3. Atributos de tipo ID](#)

[4.1.7.4. Atributos de tipo IDREF](#)

[4.1.7.5. Atributos de tipo IDREFS](#)

[4.1.7.6. Atributos de tipo NMTOKEN](#)

[4.1.7.7. Atributos de tipo NMTOKENS](#)

[4.1.7.8. Atributos de tipo NOTATION](#)

[4.1.7.9. Atributos de tipo ENTITY](#)

Uso de **ENTITY** y **NOTATION**

[4.1.7.10. Atributos de tipo ENTITIES](#)

Uso de **ENTITIES** y **NOTATION**

[4.1.7.11. Atributos especiales](#)

Uso del atributo **xml:lang**

Uso del atributo **xml:space**

[4.1.8. Declaración de entidades](#)

[4.1.8.1. Entidades generales internas analizables](#)

[4.1.8.2. Entidades generales externas analizables](#)

Entidades generales externas analizables privadas - **SYSTEM**

Entidades generales externas analizables públicas - **PUBLIC**

[4.1.8.3. Entidades generales externas no analizables](#)

Entidades generales externas no analizables privadas - **SYSTEM**

Entidades generales externas no analizables públicas - **PUBLIC**

[4.1.8.4. Entidades paramétricas internas analizables](#)

Las entidades de parámetro se declaran antes de referenciarlas

A una entidad paramétrica interna no se le puede referenciar en una DTD interna

Declaración de una entidad paramétrica en la DTD interna de un documento XML y referenciada en la DTD externa

[4.1.8.5. Entidades paramétricas externas analizables](#)

Entidades paramétricas externas analizables privadas - **SYSTEM**

Entidades paramétricas externas analizables públicas - **PUBLIC**

[4.1.8.6. Uso de una entidad dentro de otra](#)

Referencia circular o recursiva de entidades

[4.1.9. Declaración de notaciones](#)

Notaciones para indicar el formato de entidades externas - Uso de **SYSTEM**

Notación pública - **PUBLIC**

Atributos cuyo valor es el nombre de una notación

[4.1.10. Secciones condicionales](#)

[4.1.11. Espacios de nombres](#)

[4.1.12. Comentarios](#)

[4.1.13. Recursos de DTD](#)

4.1.1. Qué es DTD

DTD (*Document Type Definition*, Definición de Tipo de Documento) sirve para definir la estructura de un documento SGML o XML, permitiendo su validación.

- **SGML** (*Standard Generalized Markup Language*, Lenguaje de Marcado Generalizado Estándar). Véase: <http://www.w3.org/MarkUp/SGML/>.
- **XML** (*eXtensible Markup Language*, Lenguaje de Marcado eXtensible) es un lenguaje desarrollado por **W3C** (*World Wide Web Consortium*) que está basado en SGML.

En "<http://www.w3.org/TR/xml/>" se puede consultar la *W3C Recommendation* de XML, en la cual se fundamenta este tutorial, donde se explica –de forma introductoria a través de ejemplos– **cómo escribir y utilizar DTD para validar documentos XML**.

Un documento XML es válido (*valid*) cuando, además de estar bien formado, no incumple ninguna de las normas establecidas en su estructura.

Existen otros métodos que también permiten validar documentos XML, como por ejemplo *XML Schema* o *RELAX NG*.

4.1.2. Declaración de tipo de documento

Una DTD se puede escribir tanto interna como externamente a un archivo XML. Ahora bien, en ambos casos hay que escribir una definición **DOCTYPE** (*Document Type Declaration*, Declaración de Tipo de Documento) para asociar el documento XML a la DTD. Asimismo, un archivo XML se puede asociar simultáneamente a una DTD interna y externa.

4.1.2.1. Documento XML asociado a una DTD interna

La sintaxis para escribir una DTD interna es:

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

EJEMPLO En un documento XML se quiere guardar una lista de marcadores de páginas web, almacenando de cada uno de ellos su nombre, una descripción y su URL. Para ello, se puede escribir, por ejemplo, el archivo "*marcadores-con-dtd-interna.xml*" siguiente, que contiene una DTD interna:

```

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

<marcadores>
  <pagina>
    <nombre>Abrirllave</nombre>
    <descripcion>Tutoriales de informática.</descripcion>
    <url>http://www.abrirllave.com/</url>
  </pagina>
  <pagina>
    <nombre>Wikipedia</nombre>
    <descripcion>La enciclopedia libre.</descripcion>
    <url>http://www.wikipedia.org/</url>
  </pagina>
  <pagina>
    <nombre>W3C</nombre>
    <descripcion>World Wide Web Consortium.</descripcion>
    <url>http://www.w3.org/</url>
  </pagina>
</marcadores>

```

- Obsérvese que, en la DTD se ha indicado que **marcadores** es el elemento raíz del documento XML, el cual puede contener cero o más páginas. Para indicar esto último, se ha escrito: **(pagina)***
- Escribiendo **pagina (nombre, descripcion, url)** se especifica que, cada elemento "pagina" tiene que contener tres elementos (hijos): "nombre", "descripcion" y "url".
- Con **#PCDATA (Parsed Character Data)** escrito entre paréntesis "()" se indica que los elementos "nombre", "descripcion" y "url" pueden contener texto (cadenas de caracteres) analizable por un procesador XML.

Nota: al nombrar a los elementos "pagina" y "descripcion" no se han utilizado los caracteres (á) y (ó), respectivamente, para evitar posibles incompatibilidades con programas que puedan no reconocerlos.

Al ver el archivo "**marcadores-con-dtd-interna.xml**" en un navegador web, como por ejemplo *Google Chrome*, se visualizará algo similar a:



Como se puede observar, la DTD no se muestra en el navegador.

Por otro lado, para comprobar que el documento XML escrito en este ejemplo es válido se pueden utilizar distintos programas. Por ejemplo, véase [cómo validar un documento XML asociado a una DTD con XML Copy Editor](#).

4.1.2.2. Documento XML asociado a una DTD externa

Existen dos tipos de DTD externa: privada y pública. Para las privadas se utiliza **SYSTEM** y para las públicas **PUBLIC**. La sintaxis en cada caso es:

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

```
<!DOCTYPE elemento-raíz PUBLIC "identificador-público" "URI">
```

DTD externa privada - SYSTEM

EJEMPLO Si en un archivo llamado *"marcadores.dtd"* se escribiese la siguiente DTD:

```
<!ELEMENT marcadores (pagina)*>
<!ELEMENT pagina (nombre, descripcion, url)>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT descripcion (#PCDATA)>
<!ELEMENT url (#PCDATA)>
```

El siguiente documento XML llamado *"marcadores-con-dtd-externa.xml"*, sería válido:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE marcadores SYSTEM "marcadores.dtd">
<marcadores>
  <pagina>
    <nombre>Abrirllave</nombre>
    <descripcion>Tutoriales de informática.</descripcion>
    <url>http://www.abrirllave.com/</url>
  </pagina>
  <pagina>
    <nombre>Wikipedia</nombre>
    <descripcion>La enciclopedia libre.</descripcion>
    <url>http://www.wikipedia.org/</url>
  </pagina>
  <pagina>
    <nombre>W3C</nombre>
    <descripcion>World Wide Web Consortium.</descripcion>
    <url>http://www.w3.org/</url>
  </pagina>
</marcadores>
```

- En este documento XML, haciendo uso de una DTD externa privada, se han escrito una lista de marcadores de páginas web, guardando de cada uno de ellos su nombre, una descripción y su URL.

DTD externa pública - PUBLIC

EJEMPLO El siguiente documento XML está asociado a una DTD externa pública:

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
  <head>
    <title>Título</title>
  </head>
  <body>
    <p>Párrafo</p>
  </body>
</html>
```

- `-//W3C//DTD XHTML 1.0 Strict//EN` es un FPI (*Formal Public Identifier*, Identificador Público Formal).

4.1.2.3. Cuándo utilizar una DTD interna o una DTD externa

Para validar más de un documento XML con la misma DTD, escribir esta en un archivo externo proporciona la ventaja de no tener que repetir la DTD internamente en cada documento XML.

En el caso de que la DTD solo se utilice para validar un único documento XML, la DTD es habitual escribirla internamente.

4.1.2.4. Uso combinado de DTD interna y externa en un documento XML

Para asociar un documento XML a una DTD interna y externa simultáneamente, se pueden utilizar las siguientes sintaxis:

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

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

EJEMPLO Si en un documento XML llamado *"marcadores-con-dtd-interna-y-externa.xml"* se quiere almacenar una lista de marcadores de páginas web, guardando de cada uno de ellos su nombre, una descripción y su URL. En dicho documento se podría escribir:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE marcadores SYSTEM "marcadores.dtd" [
  <!ELEMENT marcadores (pagina)*>
  <!ELEMENT pagina (nombre, descripcion, url)>
]>

<marcadores>
  <pagina>
    <nombre>Abrirllave</nombre>
    <descripcion>Tutoriales de informática.</descripcion>
    <url>http://www.abrirllave.com/</url>
  </pagina>
  <pagina>
    <nombre>Wikipedia</nombre>
    <descripcion>La enciclopedia libre.</descripcion>
    <url>http://www.wikipedia.org/</url>
  </pagina>
  <pagina>
    <nombre>W3C</nombre>
    <descripcion>World Wide Web Consortium.</descripcion>
    <url>http://www.w3.org/</url>
  </pagina>
</marcadores>
```


De tal forma que, el contenido del archivo "*marcadores.dtd*" podría ser:

```
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT descripcion (#PCDATA)>
<!ELEMENT url (#PCDATA)>
```

4.1.3. Estructura de un documento XML

En una DTD se pueden declarar:

- Elementos
- Atributos
- Entidades
- Notaciones

Por tanto, **un documento XML será válido si** –además de no tener errores de sintaxis– **cumple lo indicado en las declaraciones de elementos, atributos, entidades y notaciones, de la DTD a la que esté asociado.**

4.1.4. Declaración de elementos

Para declarar un elemento en una DTD se utiliza la siguiente sintaxis:

```
<!ELEMENT nombre-del-elemento tipo-de-contenido>
```

En el **tipo-de-contenido** se especifica el contenido permitido en el elemento, pudiendo ser:

- Texto, (**#PCDATA**).
- Otros elementos (hijos).
- Estar vacío, **EMPTY**.
- Mixto (texto y otros elementos), **ANY**.

El contenido de un elemento puede ser texto - (**#PCDATA**)

EJEMPLO En el siguiente documento XML, el elemento "ciudad" puede contener cualquier texto (cadena de caracteres):

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ciudad [
  <!ELEMENT ciudad (#PCDATA)>
]>
<ciudad>Roma</ciudad>
```

- Escribiendo **#PCDATA** (*Parsed Character Data*) entre paréntesis "()", se ha indicado que el elemento "ciudad" puede contener una cadena de caracteres analizable.

Un elemento puede contener a otros elementos

EJEMPLO En el siguiente ejemplo, el elemento "ciudad" contiene a los elementos "nombre" y "país":

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ciudad [
    <!ELEMENT ciudad (nombre, pais)>
    <!ELEMENT nombre (#PCDATA)>
    <!ELEMENT pais (#PCDATA)>
]>

<ciudad>
    <nombre>Roma</nombre>
    <pais>Italia</pais>
</ciudad>
```

Un elemento puede no contener contenido (estar vacío) - **EMPTY**

EJEMPLO En la DTD interna del siguiente documento XML, se ha declarado el elemento "mayor_de_edad" como vacío, **EMPTY**. Por tanto, debe escribirse **<mayor_de_edad/>**:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE persona [
    <!ELEMENT persona (nombre, mayor_de_edad, ciudad)>
    <!ELEMENT nombre (#PCDATA)>
    <!ELEMENT mayor_de_edad EMPTY>
    <!ELEMENT ciudad (#PCDATA)>
]>

<persona>
    <nombre>Elsa</nombre>
    <mayor_de_edad/>
    <ciudad>Pamplona</ciudad>
</persona>
```

Nota: los elementos vacíos no pueden tener contenido, pero sí pueden tener atributos.

Un elemento puede definirse para contener contenido mixto - **ANY**

EJEMPLO En la DTD interna del siguiente documento XML, se ha indicado que el elemento "persona" puede contener texto y otros elementos, es decir, contenido mixto, **ANY**:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE persona [
    <!ELEMENT persona ANY>
    <!ELEMENT nombre (#PCDATA)>
    <!ELEMENT ciudad (#PCDATA)>
]>
```

```
<persona>
  <nombre>Elsa</nombre> vive en <ciudad>Pamplona</ciudad>.
</persona>
```

Obsérvese que, por ejemplo, también sería válido el siguiente documento XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE persona [
  <!ELEMENT persona ANY>
  <!ELEMENT nombre (#PCDATA)>
  <!ELEMENT ciudad (#PCDATA)>
]>

<persona>
  <nombre>Elsa</nombre> vive en Pamplona.
</persona>
```

O el siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE persona [
  <!ELEMENT persona ANY>
  <!ELEMENT nombre (#PCDATA)>
  <!ELEMENT ciudad (#PCDATA)>
]>

<persona>
  <nombre>Elsa</nombre>
</persona>
```

Incluso, si el elemento “persona” estuviese vacío, el documento también sería válido:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE persona [
  <!ELEMENT persona ANY>
  <!ELEMENT nombre (#PCDATA)>
  <!ELEMENT ciudad (#PCDATA)>
]>

<persona/>
```

4.1.4.1. Elementos vacíos - EMPTY

Para declarar un elemento vacío en una DTD, hay que indicar que su contenido es **EMPTY**. Un ejemplo de ello podría ser el elemento “br” del HTML, el cual sirve para hacer un salto de línea y no tiene contenido:

```
<!ELEMENT br EMPTY>
```

Dada la declaración anterior, en un documento XML el elemento "br" podría escribirse como:

```
<br/>
```

O también:

```
<br></br>
```

Por ejemplo, el siguiente documento XML sería válido:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE etiquetas_html [
  <!ELEMENT etiquetas_html (br)>
  <!ELEMENT br EMPTY>
]>

<etiquetas_html>
  <br/>
</etiquetas_html>
```

Un elemento vacío puede tener atributos

EJEMPLO Aunque un elemento se declare vacío, no pudiendo contener texto ni otros elementos, sí puede tener atributos:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE etiquetas_html [
  <!ELEMENT etiquetas_html (br)>
  <!ELEMENT br EMPTY>
  <!!ATTLIST br descripcion CDATA #REQUIRED>
]>

<etiquetas_html>
  <br descripcion="Salto de línea"/>
</etiquetas_html>
```

- En este ejemplo, para el elemento "br" se ha declarado el atributo **descripcion** de tipo **CDATA** (*Character DATA*), es decir, su valor puede ser una cadena de caracteres. Además, se ha indicado que el atributo es obligatorio escribirlo, **#REQUIRED**.

4.1.4.2. Elementos con cualquier contenido - ANY

Cuando en una DTD se quiere declarar un elemento que pueda contener cualquier contenido – bien sea texto, otros elementos o una mezcla de ambos– esto se puede hacer indicando que su contenido es de tipo **ANY**:

```
<!ELEMENT cualquier_contenido ANY>
```

EJEMPLO En el siguiente documento XML, el elemento “cualquier_contenido” contiene tres elementos “texto”:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ejemplo [
  <!ELEMENT ejemplo (cualquier_contenido)>
  <!--ELEMENT cualquier_contenido ANY-->
  <!ELEMENT texto (#PCDATA)>
]>

<ejemplo>
  <cualquier_contenido>
    <texto>Texto1</texto>
    <texto>Texto2</texto>
    <texto>Texto3</texto>
  </cualquier_contenido>
</ejemplo>
```

Fíjese que, definiendo la misma DTD, también sería válido el siguiente documento XML donde el elemento “cualquier_contenido” solo contiene texto:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ejemplo [
  <!ELEMENT ejemplo (cualquier_contenido)>
  <!--ELEMENT cualquier_contenido ANY-->
  <!ELEMENT texto (#PCDATA)>
]>

<ejemplo>
  <cualquier_contenido>Texto1. Texto2. Texto3</cualquier_contenido>
</ejemplo>
```

Asimismo, el elemento “cualquier_contenido” podría contener una mezcla de texto y uno o más elementos.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ejemplo [
  <!ELEMENT ejemplo (cualquier_contenido)>
  <!--ELEMENT cualquier_contenido ANY-->
  <!ELEMENT texto (#PCDATA)>
]>

<ejemplo>
  <cualquier_contenido>Texto1<texto>Texto2</texto>Texto3</cualquier_contenido>
</ejemplo>
```

Por otra parte, si el elemento “cualquier_contenido” estuviese vacío, el documento XML seguiría siendo válido:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ejemplo [
  <!ELEMENT ejemplo (cualquier_contenido)>
  <!!ELEMENT cualquier_contenido ANY>
  <!ELEMENT texto (#PCDATA)>
]>

<ejemplo>
  <cualquier_contenido></cualquier_contenido>
</ejemplo>
```

- En vez de `<cualquier_contenido></cualquier_contenido>`, también se puede escribir `<cualquier_contenido/>`.

4.1.4.3. Elementos con contenido de tipo texto - (#PCDATA)

Para declarar en una DTD un elemento que pueda contener texto analizable, se tiene que indicar que su contenido es **(#PCDATA)**, (*Parsed Character Data*):

```
<!ELEMENT texto (#PCDATA)>
```

EJEMPLO En el siguiente documento XML, el elemento “texto” contiene caracteres:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ejemplo [
  <!ELEMENT ejemplo (texto)>
  <!!ELEMENT texto (#PCDATA)>
]>

<ejemplo>
  <texto>Este elemento solo contiene caracteres.</texto>
</ejemplo>
```

Ahora bien, el elemento “texto” podría estar vacío y el documento XML seguiría siendo válido:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ejemplo [
  <!ELEMENT ejemplo (texto)>
  <!!ELEMENT texto (#PCDATA)>
]>

<ejemplo>
  <texto></texto>
</ejemplo>
```

En vez de `<texto></texto>`, también se puede escribir `<texto/>`.

4.1.4.4. Secuencias de elementos

En una DTD, un elemento (padre) puede ser declarado para contener a otro u otros elementos (hijos). En la sintaxis, los hijos –también llamados sucesores– tienen que escribirse entre paréntesis “()” y separados por comas “,”.

Elemento con varios hijos

EJEMPLO Para declarar un elemento (padre) que contenga tres elementos (hijos), se puede escribir:

```
<!ELEMENT padre (hijo1, hijo2, hijo3)>
```

EJEMPLO En el siguiente documento XML, el elemento “persona” contiene a los elementos “nombre”, “fecha_de_nacimiento” y “ciudad”:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE persona [
  <!ELEMENT persona (nombre, fecha_de_nacimiento, ciudad)>
  <!ELEMENT nombre (#PCDATA)>
  <!ELEMENT fecha_de_nacimiento (#PCDATA)>
  <!ELEMENT ciudad (#PCDATA)>
]>

<persona>
  <nombre>Iker</nombre>
  <fecha_de_nacimiento>26-12-1997</fecha_de_nacimiento>
  <ciudad>Valencia</ciudad>
</persona>
```

A su vez, los hijos también pueden tener sus propios hijos. Así, el elemento “fecha_de_nacimiento” puede contener, por ejemplo, a los elementos “día”, “mes” y “año”:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE persona [
  <!ELEMENT persona (nombre, fecha_de_nacimiento, ciudad)>
  <!ELEMENT nombre (#PCDATA)>
  <!ELEMENT fecha_de_nacimiento (día, mes, año)>
  <!ELEMENT día (#PCDATA)>
  <!ELEMENT mes (#PCDATA)>
  <!ELEMENT año (#PCDATA)>
  <!ELEMENT ciudad (#PCDATA)>
]>

<persona>
  <nombre>Iker</nombre>
  <fecha_de_nacimiento>
    <día>26</día>
    <mes>12</mes>
    <año>1997</año>
  </fecha_de_nacimiento>
  <ciudad>Valencia</ciudad>
</persona>
```

Orden de los hijos de un elemento

En un documento XML, los elementos (hijos) de un elemento (padre), deben escribirse en el mismo orden en el que han sido declarados en la DTD.

EJEMPLO El siguiente documento XML no es válido:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE persona [
  <!ELEMENT persona (nombre, fecha_de_nacimiento, ciudad)>
  <!ELEMENT nombre (#PCDATA)>
  <!ELEMENT fecha_de_nacimiento (dia, mes, anio)>
  <!ELEMENT dia (#PCDATA)>
  <!ELEMENT mes (#PCDATA)>
  <!ELEMENT anio (#PCDATA)>
  <!ELEMENT ciudad (#PCDATA)>
]>

<persona>
  <nombre>Iker</nombre>
  <fecha_de_nacimiento>
    <anio>1997</anio>
    <mes>12</mes>
    <dia>26</dia>
  </fecha_de_nacimiento>
  <ciudad>Valencia</ciudad>
</persona>
```

- El documento no es válido porque los elementos sucesores (hijos) del elemento "fecha_de_nacimiento" no se han escrito en el mismo orden que en la DTD.

4.1.4.5. Cardinalidad de los elementos

En una DTD, para definir el número de veces que pueden aparecer los elementos de un documento XML, se pueden utilizar los *operadores de cardinalidad* mostrados en la siguiente tabla:

Operadores de cardinalidad en DTD		
Operador	Cardinalidad	Significado
? (interrogación)	0-1	El elemento es opcional, pudiendo aparecer una sola vez o ninguna.
* (asterisco)	0-n	El elemento puede aparecer cero, una o más veces.
+ (signo más)	1-n	El elemento tiene que aparecer, obligatoriamente, una o más veces.

Nota: los elementos declarados en una DTD sobre los que no actúe ningún operador de cardinalidad, tendrán que aparecer obligatoriamente una única vez, en el o los documentos XML a los que se asocie.

Operador de cardinalidad "+" (signo más)

EJEMPLO En el siguiente documento XML, el elemento "nombre" tiene que aparecer una o más veces. En este caso, aparece tres veces:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE personas [
  <!ELEMENT personas (nombre+)>
  <!ELEMENT nombre (#PCDATA)>
]>

<personas>
  <nombre>Ana</nombre>
  <nombre>Iker</nombre>
  <nombre>Elsa</nombre>
</personas>
```

- Si sobre **nombre** no actuase el operador (+) el documento no sería válido, ya que, el elemento "personas" solo tendría que contener un elemento "nombre".
- En vez de **(nombre+)**, también se puede escribir **(nombre)+**.

Operador de cardinalidad "*" (asterisco)

EJEMPLO En la DTD interna del siguiente documento XML, se ha indicado que el elemento "nombre" tiene que aparecer una única vez. Ahora bien, el elemento "ingrediente" tiene cardinalidad (0-n), por tanto, puede aparecer cero, una o más veces:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE receta_de_cocina [
  <!ELEMENT receta_de_cocina (nombre, ingrediente*)>
  <!ELEMENT nombre (#PCDATA)>
  <!ELEMENT ingrediente (#PCDATA)>
]>

<receta_de_cocina>
  <nombre>Tortilla de patatas</nombre>
  <ingrediente>Huevo</ingrediente>
  <ingrediente>Patata</ingrediente>
  <ingrediente>Aceite</ingrediente>
  <ingrediente>Sal</ingrediente>
</receta_de_cocina>
```

Operador de cardinalidad "?" (interrogación)

EJEMPLO En la DTD del siguiente documento XML, la cardinalidad del elemento "mayor_de_edad" es (0-1), siendo opcional su aparición:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE persona [
  <!ELEMENT persona (nombre, mayor_de_edad?)>
  <!ELEMENT nombre (#PCDATA)>
  <!ELEMENT mayor_de_edad EMPTY>
]>

<persona>
  <nombre>Iker</nombre>
  <mayor_de_edad/>
</persona>
```

Así pues, el siguiente documento también es válido:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE persona [
  <!ELEMENT persona (nombre, mayor_de_edad?)>
  <!ELEMENT nombre (#PCDATA)>
  <!ELEMENT mayor_de_edad EMPTY>
]>

<persona>
  <nombre>Pedro</nombre>
</persona>
```

4.1.4.6. Elementos opcionales

En la DTD asociada a un documento XML, se pueden declarar elementos que contengan elementos opcionales. Para ello, se utiliza el *operador de elección*, representado por una barra vertical (|).

Operador de elección “|” (barra vertical)

EJEMPLO En el siguiente documento XML el elemento “articulo” puede contener un elemento “codigo” o un elemento “id”; obligatoriamente uno de ellos, pero no ambos:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE articulo [
  <!ELEMENT articulo (codigo | id)>
  <!ELEMENT codigo (#PCDATA)>
  <!ELEMENT id (#PCDATA)>
]>

<articulo>
  <codigo>AF-33</codigo>
</articulo>
```

Operador de elección “|” y operador “*”

EJEMPLO En la DTD del siguiente documento XML se indica que el elemento “articulos” puede contener varios elementos “codigo” e “id”:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE articulos [
    <!ELEMENT articulos (codigo | id)*>
    <!ELEMENT codigo (#PCDATA)>
    <!ELEMENT id (#PCDATA)>
]>

<articulos>
    <codigo>AF-32</codigo>
    <id>3891</id>
    <codigo>AF-50</codigo>
    <codigo>AF-89</codigo>
</articulos>
```

Obsérvese que, con el operador “*”, en este ejemplo se ha indicado que el contenido del elemento “articulos” tiene cardinalidad (0-n). Por tanto, el elemento “articulos” puede:

- Estar vacío.
- Contener un elemento “codigo”.
- Contener un elemento “id”.
- Contener un elemento “codigo” y un elemento “id”.
- Contener un elemento “codigo” y varios elementos “id”.
- Contener un elemento “id” y varios elementos “codigo”.
- Contener varios elementos “codigo” y varios elementos “id”.

Nótese también que, dentro del elemento “articulos” pueden aparecer elementos “codigo” e “id” en cualquier orden.

Operador de elección “|” en una secuencia de elementos

EJEMPLO En el siguiente documento XML, pueden aparecer cero o más elementos “articulo” que contengan un elemento “codigo” o un elemento “id”, y obligatoriamente un elemento “nombre”:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE articulos [
    <!ELEMENT articulos (articulo)*>
    <!ELEMENT articulo ((codigo | id), nombre)>
    <!ELEMENT codigo (#PCDATA)>
    <!ELEMENT id (#PCDATA)>
    <!ELEMENT nombre (#PCDATA)>
]>
```

```

<articulos>
  <articulo>
    <codigo>AF-47</codigo>
    <nombre>Martillo</nombre>
  </articulo>
  <articulo>
    <id>2056</id>
    <nombre>Destornillador</nombre>
  </articulo>
</articulos>

```

Secuencia de elementos en una lista de opciones

EJEMPLO En la DTD del siguiente documento XML se ha indicado que pueden aparecer cero o más elementos "localidad". En el caso de aparecer, cada uno de ellos contendrá los elementos "pais" y "ciudad", o alternativamente un elemento "codigo_postal":

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE localidades [
  <!ELEMENT localidades (localidad)*>
  <!ELEMENT localidad ((pais, ciudad) | codigo_postal)>
  <!ELEMENT pais (#PCDATA)>
  <!ELEMENT ciudad (#PCDATA)>
  <!ELEMENT codigo_postal (#PCDATA)>
]>

<localidades>
  <localidad>
    <pais>España</pais>
    <ciudad>Valencia</ciudad>
  </localidad>
  <localidad>
    <codigo_postal>31015</codigo_postal>
  </localidad>
</localidades>

```

#PCDATA en una lista de opciones permite contenido mixto

EJEMPLO Al utilizar el *operador de elección* (|) en una DTD, si una de las opciones es **#PCDATA**, esta debe escribirse en primer lugar:

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE articulos [
  <!ELEMENT articulos (#PCDATA | codigo | id)*>
  <!ELEMENT codigo (#PCDATA)>
  <!ELEMENT id (#PCDATA)>
]>

```

```
<articulos>
  <id>8608</id>
  Teclado
  <codigo>AF-18</codigo>
  <codigo>AF-45</codigo>
  Disquetera
  <id>7552</id>
  <id>4602</id>
</articulos>
```

- Fíjese que, el elemento “articulos” de este documento, puede contener contenido mixto, es decir, texto y otros elementos.

EJEMPLO Véase, en este último ejemplo, que el elemento “provincia” puede aparecer cero o más veces, pudiendo contener contenido mixto:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE localidades [
  <!ELEMENT localidades (provincia*)>
  <!ELEMENT provincia (#PCDATA | ciudad | codigo_postal)*>
  <!ELEMENT ciudad (#PCDATA)>
  <!ELEMENT codigo_postal (#PCDATA)>
]>

<localidades>
  <provincia>
    Navarra
    <ciudad>Estella</ciudad>
    <codigo_postal>31015</codigo_postal>
    <ciudad>Tafalla</ciudad>
  </provincia>
  <provincia>
    Valencia
    <codigo_postal>46520</codigo_postal>
  </provincia>
</localidades>
```

4.1.5. Declaración de atributos

La sintaxis básica para declarar un atributo en una DTD es:

```
<!ATTLIST nombre-del-elemento nombre-del-atributo tipo-de-atributo
valor-del-atributo>
```


Declaración de varios atributos en un elemento

EJEMPLO En la DTD del siguiente documento XML se ha indicado que el elemento "f1" puede tener tres atributos (**pais**, **fecha_de_nacimiento** y **equipo**):

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE deportistas [
  <!ELEMENT deportistas (futbol | f1 | tenis)*>
  <!ELEMENT futbol (#PCDATA)>
  <!ELEMENT f1 (#PCDATA)>
    <!ATTLIST f1 pais CDATA "España">
    <!ATTLIST f1 fecha_de_nacimiento CDATA #IMPLIED>
    <!ATTLIST f1 equipo CDATA #REQUIRED>
  <!ELEMENT tenis (#PCDATA)>
]>

<deportistas>
  <f1 pais="Alemania" fecha_de_nacimiento="03/07/1987"
    equipo="Ferrari">Sebastian Vettel</f1>
  <f1 equipo="McLaren">Fernando Alonso</f1>
  <tenis>Rafael Nadal</tenis>
</deportistas>
```

- Obsérvese que, en este ejemplo, el atributo **equipo** es obligatorio escribirlo, **#REQUIRED**. Mientras que, el atributo **fecha_de_nacimiento** es opcional, **#IMPLIED**.

En una DTD, cuando se declara más de un atributo para un elemento –como se ha hecho en este caso– no es necesario escribir varias veces **<!ATTLIST**, pudiéndose escribir, por ejemplo:

```
<!ATTLIST f1 pais CDATA "España"
           fecha_de_nacimiento CDATA #IMPLIED
           equipo CDATA #REQUIRED>
```

4.1.6. Tipos de declaración de atributos

En DTD, existen los siguientes tipos de declaración de atributos:

Tipos de declaración de atributos en DTD	
Valor	Significado
valor entre comillas dobles (") o simples (').	El atributo tiene un valor por defecto.
#REQUIRED	El atributo es obligatorio escribirlo.
#IMPLIED	El atributo es opcional escribirlo.
#FIXED valor entre comillas dobles (") o simples (').	El valor del atributo es fijo.

En el apartado anterior de este tutorial "5. Declaración de atributos", se muestra un ejemplo declaración de atributo con un valor por defecto.

Respecto a los otros tipos de declaración de atributos (**#REQUIRED**, **#IMPLIED** y **#FIXED valor**), en los siguientes apartados se muestran ejemplos:

- 6.1. Atributo obligatorio - **#REQUIRED**
- 6.2. Atributo opcional - **#IMPLIED**
- 6.3. Atributo con valor fijo - **#FIXED valor**

4.1.6.1. Atributo obligatorio - **#REQUIRED**

EJEMPLO En la DTD interna del siguiente documento XML se ha declarado un atributo indicando que es obligatorio, es decir, **#REQUIRED**:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE deportistas [
  <!ELEMENT deportistas (futbol | f1 | tenis)*>
  <!ELEMENT futbol (#PCDATA)>
  <!ELEMENT f1 (#PCDATA)>
    <!ATTLIST f1 pais CDATA #REQUIRED>
  <!ELEMENT tenis (#PCDATA)>
]>

<deportistas>
  <f1 pais="Alemania">Sebastian Vettel</f1>
  <f1>Fernando Alonso</f1>
  <tenis>Rafael Nadal</tenis>
</deportistas>
```

- En este ejemplo, es obligatorio escribir el atributo **pais** en los elementos "f1". Por tanto, aunque el documento XML está bien formado, habría que indicar el **pais** de *Fernando Alonso* para que fuese válido.

```
<f1 pais="España">Fernando Alonso</f1>
```

- Por otra parte, fíjese que, de *Rafael Nadal* no es obligatorio indicar su país, ni se puede hacer.

4.1.6.2. Atributo opcional - **#IMPLIED**

EJEMPLO En una DTD, para especificar que un atributo es opcional escribirlo o no, hay que indicarlo mediante **#IMPLIED**:


```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE deportistas [
  <!ELEMENT deportistas (futbol | f1 | tenis)*>
  <!ELEMENT futbol (#PCDATA)>
  <!ELEMENT f1 (#PCDATA)>
    <!ATTLIST f1 pais CDATA #IMPLIED>
  <!ELEMENT tenis (#PCDATA)>
]>

<deportistas>
  <f1 pais="Alemania">Sebastian Vettel</f1>
  <f1>Fernando Alonso</f1>
  <tenis>Rafael Nadal</tenis>
</deportistas>
```

- En este caso, el atributo **pais** es opcional para los elementos "f1" que aparezcan en el documento XML. Así pues, obsérvese que, aunque no se ha indicado el país de *Fernando Alonso*, el documento es válido.

4.1.6.3. Atributo con valor fijo - #FIXED valor

EJEMPLO Cuando en una DTD, se quiere declarar un atributo que tome un valor fijo, esto se puede hacer con **#FIXED valor**:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE deportistas [
  <!ELEMENT deportistas (futbol | f1 | tenis)*>
  <!ELEMENT futbol (#PCDATA)>
  <!ELEMENT f1 (#PCDATA)>
    <!ATTLIST f1 pais CDATA #FIXED "España">
  <!ELEMENT tenis (#PCDATA)>
]>

<deportistas>
  <f1 pais="España">Carlos Sainz</f1>
  <f1>Fernando Alonso</f1>
  <tenis>Rafael Nadal</tenis>
</deportistas>
```

- Según la DTD de este documento XML, todos los elementos "f1" que aparezcan tendrán el atributo **pais** con el valor "**España**". Por tanto, no es necesario haberlo escrito para *Carlos Sainz*. De hecho, si se hubiese escrito otro valor, el documento no sería válido.

De modo que, para este caso, al visualizar el documento XML en un navegador web, se mostrará algo parecido a:



4.1.7. Tipos de atributos

En DTD, existen los siguientes tipos de atributos:

Tipos de atributos en DTD	
Tipo	Descripción
CDATA	(<i>Character DATA</i>) El valor son datos de tipo carácter, es decir, texto.
Enumerado	El valor puede ser uno de los pertenecientes a una lista de valores escritos entre <i>paréntesis</i> “ () ” y separados por el carácter “ ”.
ID	El valor es un identificador único.
IDREF	El valor es un identificador que tiene que existir en otro atributo ID del documento XML.
IDREFS	El valor es una lista de valores que existan en otros atributos ID del documento XML, separados por espacios en blanco.
NMTOKEN	El valor es una cadena de caracteres, pudiendo contener letras minúsculas, letras mayúsculas, números, puntos “ . ”, guiones medios “ - ”, guiones bajos “ _ ” o el carácter dos puntos “ : ”.
NMTOKENS	El valor puede contener uno o varios valores de tipo NMTOKEN separados por espacios en blanco.
NOTATION	El valor es el nombre de una notación.
ENTITY	El valor es el nombre de una entidad.
ENTITIES	El valor puede contener uno o varios valores de tipo ENTITY separados por espacios en blanco.
Especiales	Existen dos atributos especiales: xml : lang y xml : space .

- Véase que, en este caso, se ha especificado "**ESP**" como valor por defecto, siendo obligatorio que esté en la lista de valores escritos entre *paréntesis* " ()".

Al visualizar este documento en un navegador web, en pantalla se verá:



Si se quiere definir el atributo **pais** obligatorio, habría que escribir:

```
<!ATTLIST f1 pais (ESP | FRA | ITA | ALE) #REQUIRED>
```

Por tanto, para Fernando Alonso se tendría que escribir:

```
<f1 pais="ESP">Fernando Alonso</f1>
```

4.1.7.3. Atributos de tipo ID

En una DTD, los atributos declarados **ID** son aquellos que solo pueden tomar un valor único (identificador) para cada elemento.

EJEMPLO En la DTD del siguiente documento XML, el atributo **codigo** del elemento "f1" ha sido declarado de tipo **ID**:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE deportistas [
  <!ELEMENT deportistas (futbol | f1)*>
  <!ELEMENT futbol (#PCDATA)>
  <!ELEMENT f1 (#PCDATA)>
    <!ATTLIST f1 codigo ID #REQUIRED>
]>
```

```
<deportistas>
  <f1 codigo="ALO">Fernando Alonso</f1>
  <f1 codigo="VET">Sebastian Vettel</f1>
</deportistas>
```

Hay que tener en cuenta que:

- Los valores de atributos **ID**, tienen que cumplir las mismas normas de sintaxis utilizadas para escribir nombres en XML.
- Cada elemento escrito en un documento XML, solo puede tener un atributo **ID**.
- En un documento XML, no pueden escribirse dos elementos que tengan el mismo valor en un atributo **ID**, aunque dicho atributo sea distinto.
- Todo atributo declarado de tipo **ID** tiene que ser **#IMPLIED** (opcional) o **#REQUIRED** (obligatorio).

4.1.7.4. Atributos de tipo IDREF

En una DTD, los atributos declarados **IDREF** son aquellos cuyo valor tiene que existir en otro atributo **ID** del documento XML.

EJEMPLO En la DTD del siguiente documento XML, se indica que los elementos “película” que se escriban, deben incluir el atributo **direccion**, cuyo valor estará asignado a un atributo **ID** de otro elemento del documento. En este caso, el valor estará asignado a un atributo **coddir** de un elemento “director”:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cine [
  <!ELEMENT cine (directores, peliculas)>
  <!ELEMENT directores (director)*>
    <!ELEMENT director (#PCDATA)>
      <!ATTLIST director coddir ID #REQUIRED>
  <!ELEMENT peliculas (película)*>
    <!ELEMENT película (#PCDATA)>
      <!ATTLIST película direccion IDREF #REQUIRED>
]>

<cine>
  <directores>
    <director coddir="CE">Clint Eastwood</director>
    <director coddir="JC">James Cameron</director>
  </directores>
  <peliculas>
    <película direccion="JC">Avatar</película>
    <película direccion="CE">Mystic River</película>
    <película direccion="JC">Titanic</película>
  </peliculas>
</cine>
```

- Obsérvese que, por ejemplo, para la película *Titanic* se ha indicado en su atributo **direccion** el valor "JC", que es el valor del atributo **coddir** del director *James Cameron*.
- En este documento XML, el atributo de tipo **IDREF** se ha definido obligatorio, **#REQUIRED**. Pero, a un atributo **IDREF** también se le puede especificar un valor por defecto, un valor fijo o que sea opcional escribirlo, **#IMPLIED**.

4.1.7.5. Atributos de tipo IDREFS

En una DTD, los atributos declarados **IDREFS** son aquellos cuyo valor puede ser una lista de valores que existan en otros atributos **ID** del documento XML.

EJEMPLO En la DTD del siguiente documento XML, se indica que el valor del atributo **filmografia** de un elemento "director", puede ser una lista de valores de atributos **ID**. En este caso, una lista de valores escritos en el atributo **codpel** de los elementos "película" que aparezcan en el documento XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cine [
  <!ELEMENT cine (peliculas, directores)>
  <!ELEMENT peliculas (pelicula)*>
    <!ELEMENT pelicula (#PCDATA)>
      <!ATTLIST pelicula codpel ID #REQUIRED>
  <!ELEMENT directores (director)*>
    <!ELEMENT director (#PCDATA)>
      <!ATTLIST director filmografia IDREFS #REQUIRED>
]>

<cine>
  <peliculas>
    <pelicula codpel="P1">Avatar</pelicula>
    <pelicula codpel="P2">Mystic River</pelicula>
    <pelicula codpel="P3">The Terminator</pelicula>
    <pelicula codpel="P4">Titanic</pelicula>
  </peliculas>
  <directores>
    <director filmografia="P2">Clint Eastwood</director>
    <director filmografia="P1 P3 P4">James Cameron</director>
  </directores>
</cine>
```

- Obsérvese que, los valores de la lista de valores de un atributo **IDREFS**, se escriben separados por un espacio en blanco.

4.1.7.6. Atributos de tipo NMTOKEN

En una DTD, los atributos declarados **NMTOKEN** son aquellos cuyo valor será una cadena de caracteres, pudiendo contener letras minúsculas, letras mayúsculas, números, puntos ".", guiones medios "-", guiones bajos "_" o el carácter dos puntos ":".

EJEMPLO En la DTD del siguiente documento XML, el atributo **clave** del elemento "usuario" ha sido declarado de tipo **NMTOKEN**:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE usuarios [
  <!ELEMENT usuarios (usuario)*>
  <!ELEMENT usuario (#PCDATA)>
  <!ATTLIST usuario clave NMTOKEN #REQUIRED>
]>

<usuarios>
  <usuario clave="123456789">Ana</usuario>
  <usuario clave="ab-c-d-fg">Iker</usuario>
  <usuario clave="A1_B2..C3">Elsa</usuario>
</usuarios>
```

- En el valor de un atributo **NMTOKEN** no se pueden escribir *espacios en blanco* ni caracteres especiales, tales como: *, \$, %, &, ?, @...

4.1.7.7. Atributos de tipo NMTOKENS

En una DTD, los atributos declarados **NMTOKENS** son aquellos cuyo valor puede contener uno o varios valores de tipo **NMTOKEN** separados por espacios en blanco.

EJEMPLO En la DTD del siguiente documento XML, el atributo **codigos** del elemento "usuario" ha sido declarado de tipo **NMTOKENS**:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE usuarios [
  <!ELEMENT usuarios (usuario)*>
  <!ELEMENT usuario (#PCDATA)>
  <!ATTLIST usuario codigos NMTOKENS #REQUIRED>
]>

<usuarios>
  <usuario codigos="1234 567 89">Ana</usuario>
  <usuario codigos="ab c-d fg">Iker</usuario>
  <usuario codigos="A1:B2">Elsa</usuario>
</usuarios>
```

- Obsérvese que, los valores escritos en el atributo **codigos**, se escriben separados por espacios en blanco.

4.1.7.8. Atributos de tipo NOTATION

En una DTD, los atributos declarados **NOTATION** son aquellos cuyo valor puede ser el nombre de una notación.

EJEMPLO En la DTD del siguiente documento XML, se indica que los elementos "animal" que se escriban, deben incluir opcionalmente el atributo **tipo_de_imagen**, cuyo valor será una notación (**gif**, **jpg** o **png**):

```
<?xml version = "1.0" encoding="UTF-8"?>
<!DOCTYPE animales [
  <!ELEMENT animales (animal)*>
  <!ELEMENT animal (nombre)>
  <!ELEMENT nombre (#PCDATA)>
  <!ATTLIST animal
    imagen CDATA #IMPLIED
    tipo_de_imagen NOTATION (jpg | gif | png) #IMPLIED>

  <!NOTATION gif SYSTEM "image/gif">
  <!NOTATION jpg SYSTEM "image/jpeg">
  <!NOTATION png SYSTEM "image/png">
]>

<animales>
  <animal imagen="ballena-azul.gif" tipo_de_imagen="gif">
    <nombre>Ballena</nombre>
  </animal>
  <animal imagen="leon-dormido.png" tipo_de_imagen="png">
    <nombre>Leon</nombre>
  </animal>
</animales>
```

- En este ejemplo, las notaciones **gif**, **jpg** y **png** son declaraciones de los siguientes tipos MIME (*Multipurpose Internet Mail Extensions*, Extensiones Multipropósito de Correo de Internet): *image/gif*, *image/jpeg* e *image/png*.

EJEMPLO En la DTD del siguiente documento XML, se indica que los elementos "programa" que se escriban, deben incluir obligatoriamente el atributo **lenguaje**, cuyo valor será una notación (**csharp** o **java**):

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE programas [
  <!ELEMENT programas (programa)*>
  <!ELEMENT programa (#PCDATA)>
  <!ATTLIST programa lenguaje NOTATION (csharp|java) #REQUIRED>

  <!NOTATION csharp PUBLIC "CSharp 5.0">
  <!NOTATION java PUBLIC "Java 8.0">
]>
```



```
<programas>
  <programa lenguaje="java"><!-- Código fuente 1. --></programa>
  <programa lenguaje="java"><!-- Código fuente 2. --></programa>
  <programa lenguaje="csharp"><!-- Código fuente 3. --></programa>
</programas>
```

- *CSharp 5.0* y *Java 8.0* son identificadores públicos.

4.1.7.9. Atributos de tipo ENTITY

En una DTD, los atributos declarados **ENTITY** son aquellos cuyo valor puede ser el nombre de una entidad.

Uso de ENTITY y NOTATION

EJEMPLO En la DTD del siguiente documento XML, se indica que los elementos “animal” que se escriban, tiene que incluir obligatoriamente el atributo **imagen**, cuyo valor será una entidad:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE animales [
  <!ELEMENT animales (animal)*>
  <!ELEMENT animal EMPTY>
  <!ATTLIST animal imagen ENTITY #REQUIRED>

  <!ENTITY ballena SYSTEM "ballena.gif" NDATA gif>
  <!ENTITY delfin SYSTEM "delfin.gif" NDATA gif>

  <!NOTATION gif SYSTEM "image/gif">
]>

<animales>
  <animal imagen="ballena"/>
  <animal imagen="delfin"/>
</animales>
```

- En la DTD de este ejemplo se está indicando que los valores –datos– de las entidades (**ballena** y **delfin**) van a ser cargados desde una URI (*Uniform Resource Identifier*, Identificador Uniforme de Recurso). En este caso, se hace referencia a los archivos externos “*ballena.gif*” y “*delfin.gif*”.
- Con **NDATA** (*Notation Data*) se ha asociado a las entidades **ballena** y **delfin** con la notación **gif**.
- La notación **gif** es una declaración del tipo *MIME image/gif*.

4.1.7.10. Atributos de tipo ENTITIES

En una DTD, los atributos declarados **ENTITIES** son aquellos cuyo valor puede contener uno o varios valores de tipo **ENTITY** separados por espacios en blanco.

Uso de ENTITIES y NOTATION

EJEMPLO En la DTD del siguiente documento XML, el atributo **imagenes** del elemento "grupos" ha sido declarado de tipo **ENTITIES**:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE animales [
  <!ELEMENT animales (grupos)*>
  <!ELEMENT grupos EMPTY>
  <!ATTLIST grupos imagenes ENTITIES #REQUIRED>

  <!ENTITY ballena SYSTEM "ballena.gif" NDATA gif>
  <!ENTITY delfin SYSTEM "delfin.gif" NDATA gif>
  <!ENTITY elefante SYSTEM "elefante.gif" NDATA gif>
  <!ENTITY leon SYSTEM "leon.gif" NDATA gif>
  <!ENTITY oso SYSTEM "oso.gif" NDATA gif>

  <!NOTATION gif SYSTEM "image/gif">
]>

<animales>
  <grupos imagenes="ballena"/>
  <grupos imagenes="ballena delfin"/>
  <grupos imagenes="elefante leon oso"/>
  <grupos imagenes="ballena elefante"/>
</animales>
```

- En la DTD de este ejemplo se está indicando que los valores –datos– de las entidades (**ballena**, **delfin**, **elefante**, **leon** y **oso**) van a ser cargados desde una URI (*Uniform Resource Identifier*, Identificador Uniforme de Recurso). En este caso, se hace referencia a los archivos externos "*ballena.gif*", "*delfin.gif*", "*elefante.gif*", "*leon.gif*" y "*oso.gif*".
- Con **NDATA** (*Notation Data*) se ha asociado a las entidades **ballena**, **delfin**, **elefante**, **leon** y **oso** con la notación **gif**.
- La notación **gif** es una declaración del tipo *MIME image/gif*.

4.1.7.11. Atributos especiales

En DTD existen dos tipos de atributos especiales (predefinidos), llamados: **xml:lang** y **xml:space**.

Uso del atributo **xml:lang**

En una DTD, el atributo **xml:lang** permite indicar el idioma del contenido y de los valores de los atributos de un elemento declarado. De forma que, cuando se utiliza **xml:lang** en un elemento, el idioma especificado afecta a todos los valores de sus posibles atributos y a todo su contenido, incluyendo a sus posibles sucesores a menos que se indique lo contrario con otra instancia de **xml:lang**.

EJEMPLO En la DTD del siguiente documento XML, con el atributo **xml:lang** se ha indicado el idioma de los elementos "sigla" y "traduccion":

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE siglas [
  <!ELEMENT siglas (sigla)*>
  <!ELEMENT sigla (significado, traduccion)>
  <!ATTLIST sigla letras CDATA #REQUIRED>
  <!ATTLIST sigla xml:lang CDATA "en">
  <!ELEMENT significado (#PCDATA)>
  <!ELEMENT traduccion (#PCDATA)>
  <!ATTLIST traduccion xml:lang CDATA #FIXED "es">
]>

<siglas>
  <sigla letras="ANSI">
    <significado>American National Standards Institute</significado>
    <traduccion>Instituto Nacional Estadounidense de Estándares</traduccion>
  </sigla>
  <sigla letras="ISO">
    <significado>International Organization for Standardization</significado>
    <traduccion>Organización Internacional de Normalización</traduccion>
  </sigla>
  <sigla letras="CERN" xml:lang="fr">
    <significado>Conseil Européen pour la Recherche Nucléaire</significado>
    <traduccion>Organización Europea para la Investigación Nuclear</traduccion>
  </sigla>
</siglas>
```

- Inicialmente, para el elemento "sigla" se ha indicado el idioma inglés, "en", por defecto.
- No obstante, después se ha fijado el valor "es", del español, para el atributo **xml:lang** del elemento "traduccion".
- Por otra parte, para el CERN se ha especificado que el idioma es el francés, "fr".

Uso del atributo **xml:space**

En una DTD, el atributo **xml:space** permite indicar que los espacios en blanco, las tabulaciones y los retornos de carro que aparezcan en el contenido (texto) de un elemento –y sus sucesores a menos que se indique lo contrario con otra instancia de **xml:space**– tienen que ser preservados. Este atributo siempre tiene que ser declarado de tipo enumerado, siendo "**default**", "**preserve**" o ambos, los posibles valores pertenecientes a la lista de valores que se indiquen entre paréntesis " () ".

EJEMPLO En la DTD del siguiente documento XML, con el atributo **xml:space** se ha indicado que, por defecto, los espacios que se escriban en el contenido de los elementos "programa" del documento, deben preservarse. Ahora bien, en la declaración de **xml:space** se ha indicado que su valor podría ser también "**default**":

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE programas [
  <!ELEMENT programas (programa)*>
  <!ELEMENT programa (#PCDATA)>
  <!ATTLIST programa xml:space (default|preserve) "preserve">
]>
<programas>
  <programa>/* Programa: Hola mundo */

#include <conio.h>;
#include <stdio.h>;

int main()
{
    printf( "Hola mundo." );

    getch(); /* Pausa */

    return 0;
}</programa>
  <programa>/* Programa: Calificación según nota */

#include <conio.h>;
#include <stdio.h>;

int main()
{
    float nota;

    printf( "\n  Introduzca nota (real): " );
    scanf( "%f", &nota );

    if ( nota &gt;= 5 )
        printf( "\n  APROBADO" );
    else
        printf( "\n  SUSPENDIDO" );

    getch(); /* Pausa */

    return 0;
}</programa>
</programas>
```

- En este ejemplo, los espacios en blanco, las tabulaciones y los retornos de carro de los dos programas escritos tienen que preservarse.
- No obstante, tal y como está declarado el atributo **xml:space** del elemento "programa", se podría asignar el valor **"default"** a **xml:space** en cualquier **programa**. En tal caso, sería el programa que procese el documento, el que decidiese qué tratamiento hacer a los espacios en blanco, las tabulaciones y los retornos de carro.

4.1.8. Declaración de entidades

En una DTD se pueden declarar entidades generales y paramétricas (de parámetro). Las entidades generales pueden ser:

- Entidades generales internas analizables (*parsed*).
- Entidades generales externas analizables (*parsed*).
- Entidades generales externas no analizables (*unparsed*).

Por otro lado, las entidades paramétricas pueden ser:

- Entidades paramétricas internas analizables (*parsed*).
- Entidades paramétricas externas analizables (*parsed*).

Las entidades generales pueden utilizarse en el cuerpo de un documento XML y en su DTD. Sin embargo, las entidades paramétricas solo pueden utilizarse dentro de la DTD.

4.1.8.1. Entidades generales internas analizables

Para declarar una entidad general interna analizable (*parsed*) en una DTD, se utiliza la siguiente sintaxis:

```
<!ENTITY nombre-de-la-entidad "valor-de-la-entidad">
```

EJEMPLO En la DTD del siguiente documento XML, se han declarado tres entidades (**escritor**, **obra** y **fecha**):

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE textos [
  <!ELEMENT textos (texto)+>
  <!ELEMENT texto (#PCDATA)>

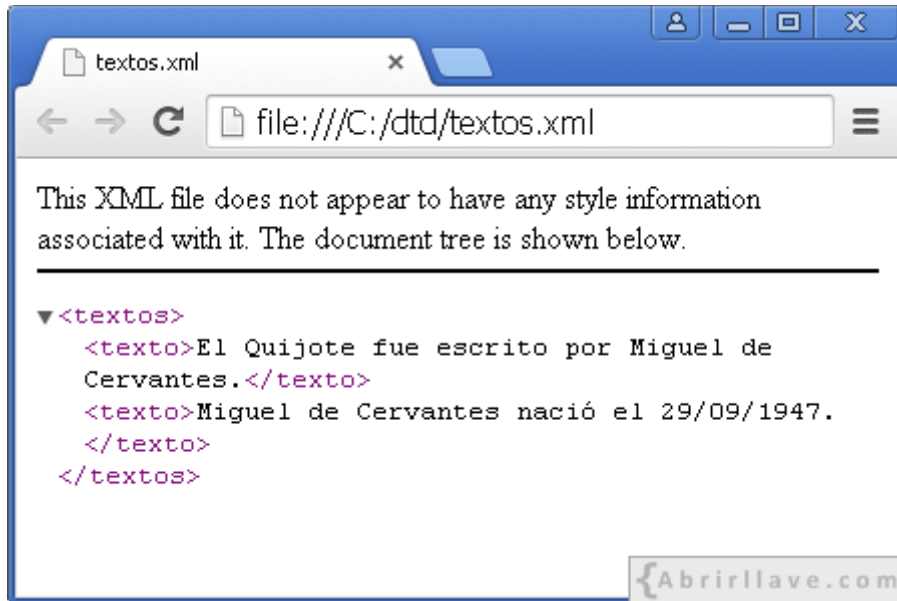
  <!ENTITY escritor "Miguel de Cervantes">
  <!ENTITY obra "El Quijote">
  <!ENTITY fecha "29/09/1947">
]>
```

```
<textos>
  <texto>&obra; fue escrito por &escritor;.</texto>
  <texto>&escritor; nació el &fecha;.</texto>
</textos>
```

- Obsérvese que, para referenciar a las entidades, se ha utilizado la sintaxis:

```
&nombre-de-la-entidad;
```

Si este documento XML se visualizase en un navegador web, se vería algo parecido a:



4.1.8.2. Entidades generales externas analizables

En una DTD se pueden declarar dos tipos de entidades generales externas analizables (*parsed*): privadas y públicas. Para las privadas se utiliza **SYSTEM**, y para las públicas **PUBLIC**. La sintaxis en cada caso es:

```
<!ENTITY nombre-de-la-entidad SYSTEM "URI">
```

```
<!ENTITY nombre-de-la-entidad PUBLIC "identificador-público" "URI">
```

Entidades generales externas analizables privadas - **SYSTEM**

EJEMPLO En la DTD del siguiente documento XML, se ha declarado la entidad **escritor**:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE textos [
  <!ELEMENT textos (texto)+>
  <!ELEMENT texto (#PCDATA)>

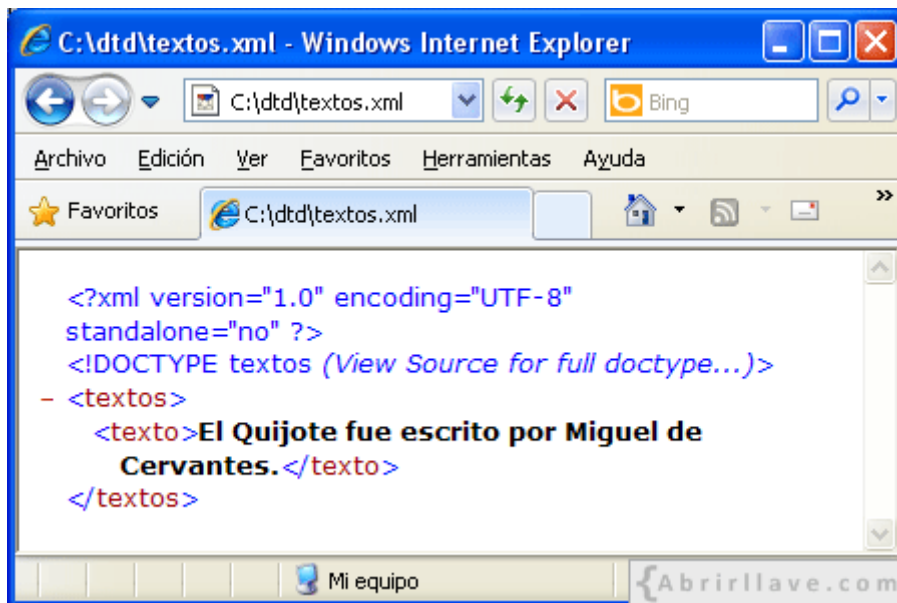
  <!ENTITY escritor SYSTEM "escritor.txt">
]>

<textos>
  <texto>El Quijote fue escrito por &escritor;.</texto>
</textos>
```

Suponiendo que el archivo **"escritor.txt"** contenga:

Miguel de Cervantes

En un navegador web (por ejemplo en *Internet Explorer 8*) se podrá ver:



Entidades generales externas analizables públicas - **PUBLIC**

EJEMPLO Para declarar **escritor** como entidad general externa analizable pública, se puede escribir:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<!DOCTYPE textos [
  <!ELEMENT textos (texto)+>
  <!ELEMENT texto (#PCDATA)>

  <!ENTITY escritor PUBLIC "-//W3C//TEXT escritor//EN"
    "http://www.abrirllave.com/dtd/escritor.txt">
]>
```

```
<textos>
  <texto>El Quijote fue escrito por &escritor;.</texto>
</textos>
```

4.1.8.3. Entidades generales externas no analizables

En una DTD, al igual que ocurre con las entidades generales externas analizables, se pueden declarar dos tipos de entidades generales externas no analizables (*unparsed*): privadas y públicas. Para las privadas se utiliza **SYSTEM**, y para las públicas **PUBLIC**. La sintaxis en cada caso es:

```
<!ENTITY nombre-de-la-entidad SYSTEM "URI" NDATA notación>
```

```
<!ENTITY nombre-de-la-entidad PUBLIC "identificador-público" "URI"
NDATA notación>
```

Las entidades no analizables pueden contener cualquier tipo de datos (no XML). Por tanto, pueden hacer referencia a datos que un procesador XML no tiene porqué analizar, como por ejemplo una imagen.

Entidades generales externas no analizables privadas - **SYSTEM**

EJEMPLO En la DTD del siguiente documento XML, se indica que el elemento “imagen” que se escriba, tiene que incluir obligatoriamente el atributo **fuelle**, cuyo valor será una entidad:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE imagen [
  <!ELEMENT imagen EMPTY>
  <!ATTLIST imagen fuele ENTITY #REQUIRED>

  <!ENTITY logo SYSTEM "logo.gif" NDATA gif>

  <!NOTATION gif SYSTEM "image/gif">
]>

<imagen fuelle="logo"/>
```

- En la DTD de este ejemplo se está indicando que el valor –datos– de la entidad **logo** va a ser cargado desde una URI. En este caso, se hace referencia al archivo “logo.gif”.
- Con **NDATA** (*Notation Data*) se indica que la entidad no es analizable y, en este caso, se ha asociado a la entidad **logo** con la notación **gif**.
- La notación **gif** es una declaración del tipo *MIME image/gif*.

Entidades generales externas no analizables públicas - PUBLIC

EJEMPLO Para declarar **logo** como entidad pública, se puede escribir:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE imagen [
  <!ELEMENT imagen EMPTY>
  <!--ATTLIST imagen fuente ENTITY #REQUIRED-->

  <!--ENTITY logo PUBLIC "-//W3C//GIF logo//EN"
  "http://www.abrirllave.com/dtd/logo.gif" NDATA gif-->

  <!--NOTATION gif SYSTEM "image/gif"-->
]>

<imagen fuente="logo"/>
```

- Véase que, se referencia al archivo "<http://www.abrirllave.com/dtd/logo.gif>".

4.1.8.4. Entidades paramétricas internas analizables

Para declarar una entidad paramétrica (de parámetro) interna analizable (*parsed*) en una DTD, se utiliza la siguiente sintaxis:

```
<!--ENTITY % nombre-de-la-entidad "valor-de-la-entidad"-->
```

EJEMPLO La DTD del siguiente documento XML es externa, habiéndose escrito esta en el archivo "**persona.dtd**":

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE persona SYSTEM "persona.dtd">
<persona>
  <nombre>Iker</nombre>
  <mayor_de_edad/>
  <ciudad>Pamplona</ciudad>
</persona>
```

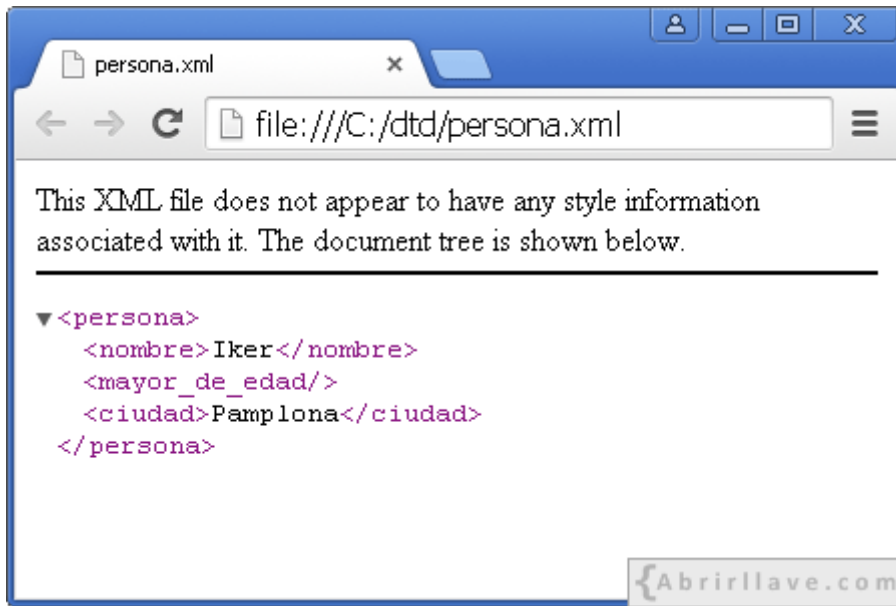
El contenido del archivo "**persona.dtd**" podría ser:

```
<!--ENTITY % p "(#PCDATA)"-->
<!--ELEMENT persona (nombre, mayor_de_edad?, ciudad)-->
<!--ELEMENT nombre %p;-->
<!--ELEMENT mayor_de_edad EMPTY-->
<!--ELEMENT ciudad %p;-->
```

- Obsérvese que, en la DTD se ha declarado la entidad paramétrica **p** y, para referenciarla, se utiliza la sintaxis:

```
%nombre-de-la-entidad;
```

Si este documento XML se visualizase en un navegador web, se vería algo parecido a:



Las entidades de parámetro se declaran antes de referenciarlas

En una DTD las entidades paramétricas tienen que declararse antes de ser referenciadas. Por tanto, no sería correcto haber escrito, por ejemplo:

```
<!ELEMENT persona (nombre, mayor_de_edad?, ciudad)>
<!ELEMENT nombre %p;>
<!ELEMENT mayor_de_edad EMPTY>
<!ELEMENT ciudad %p;>

<!ENTITY % p "(#PCDATA)">
```

A una entidad paramétrica interna no se le puede referenciar en una DTD interna

Las entidades paramétricas internas pueden declararse en DTD internas o externas. Sin embargo, no pueden referenciarse desde una DTD interna. En consecuencia, el siguiente documento no sería válido:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE persona [
  <!ENTITY % p "(#PCDATA)">

  <!ELEMENT persona (nombre, mayor_de_edad?, ciudad)>
  <!ELEMENT nombre %p;>
  <!ELEMENT mayor_de_edad EMPTY>
  <!ELEMENT ciudad %p;>
]>
```

```
<persona>
  <nombre>Iker</nombre>
  <mayor_de_edad/>
  <ciudad>Pamplona</ciudad>
</persona>
```

Declaración de una entidad paramétrica en la DTD interna de un documento XML y referenciada en la DTD externa

Ahora bien, sí sería válido el siguiente documento XML, donde internamente se declara la entidad paramétrica **p**:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE persona SYSTEM "persona.dtd" [
  <!ENTITY % p "(#PCDATA)">
]>

<persona>
  <nombre>Iker</nombre>
  <mayor_de_edad/>
  <ciudad>Pamplona</ciudad>
</persona>
```

En este caso, el contenido del archivo "**persona.dtd**" podría ser:

```
<!ELEMENT persona (nombre, mayor_de_edad?, ciudad)>
<!ELEMENT nombre %p;>
<!ELEMENT mayor_de_edad EMPTY>
<!ELEMENT ciudad %p;>
```

4.1.8.5. Entidades paramétricas externas analizables

En una DTD se pueden declarar dos tipos de entidades paramétricas externas analizables (*parsed*): privadas y públicas. Para las privadas se utiliza **SYSTEM**, y para las públicas **PUBLIC**. La sintaxis en cada caso es:

```
<!ENTITY % nombre-de-la-entidad SYSTEM "URI">
%nombre-de-la-entidad;
```

```
<!ENTITY % nombre-de-la-entidad PUBLIC "identificador-público" "URI">
%nombre-de-la-entidad;
```

Entidades paramétricas externas analizables privadas - **SYSTEM**

EJEMPLO En la DTD del siguiente documento XML, se ha declarado la entidad **persona**:

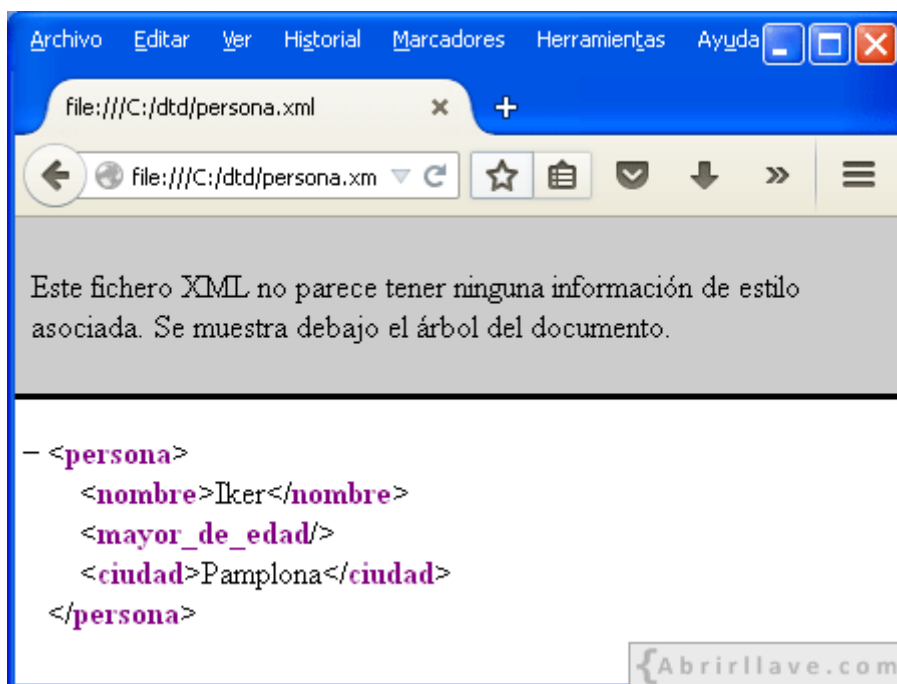
```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE persona [
  <!ENTITY % persona SYSTEM "persona.dtd">
  %persona;
]>

<persona>
  <nombre>Iker</nombre>
  <mayor_de_edad/>
  <ciudad>Pamplona</ciudad>
</persona>
```

Suponiendo que el archivo "**persona.dtd**" contenga:

```
<!ELEMENT persona (nombre, mayor_de_edad?, ciudad)>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT mayor_de_edad EMPTY>
<!ELEMENT ciudad (#PCDATA)>
```

En un navegador web (por ejemplo en *Mozilla Firefox*) se podrá ver:



Entidades paramétricas externas analizables públicas - **PUBLIC**

EJEMPLO Para declarar **persona** como entidad paramétrica externa analizable pública, se puede escribir:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE persona [
  <!ENTITY % persona PUBLIC "-//W3C//TEXT persona//EN"
    "http://www.abrirllave.com/dtd/persona.dtd">
    %persona;
]>

<persona>
  <nombre>Iker</nombre>
  <mayor_de_edad/>
  <ciudad>Pamplona</ciudad>
</persona>
```

4.1.8.6. Uso de una entidad dentro de otra

EJEMPLO En la DTD del siguiente documento XML, se han declarado dos entidades generales internas analizables (**color** y **frase**):

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE frase [
  <!ELEMENT frase (#PCDATA)>

  <!ENTITY color "azul">
  <!ENTITY frase "El cielo es &color;.">
]>

<frase>&frase;</frase>
```

- Obsérvese que, la entidad **color** ha sido referenciada en el valor de la entidad **frase**. De forma que, si este documento XML se visualizase en un navegador web, se vería:



Referencia circular o recursiva de entidades

EJEMPLO La DTD del siguiente documento XML no es correcta, ya que, la entidad **frase1** ha sido referenciada en el valor de la entidad **frase2**, y al revés también:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE frase [
  <!ELEMENT frase (#PCDATA)>

  <!ENTITY frase1 "Esta frase incluye a la &frase2;.">
  <!ENTITY frase2 "Esta frase incluye a la &frase1;.">
]>

<frase>&frase1;</frase>
```

Para que dicha DTD fuese correcta, habría que quitar una de las dos referencias a entidades. Por ejemplo escribiendo:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE frase [
  <!ELEMENT frase (#PCDATA)>

  <!ENTITY frase1 "Esta frase incluye a la &frase2;.">
  <!ENTITY frase2 "segunda frase">
]>

<frase>&frase1;</frase>
```

4.1.9. Declaración de notaciones

En una DTD se pueden declarar dos tipos de notaciones: privadas y públicas. Para las privadas se utiliza **SYSTEM**, y para las públicas **PUBLIC**, pudiéndose utilizar las siguientes sintaxis:

```
<!NOTATION nombre-de-la-notación SYSTEM "identificador-del-sistema">
```

```
<!NOTATION nombre-de-la-notación PUBLIC "identificador-público">
```

```
<!NOTATION nombre-de-la-notación PUBLIC "identificador-público"
"identificador-del-sistema">
```

Notaciones para indicar el formato de entidades externas - Uso de **SYSTEM**

En la DTD de un documento XML, las notaciones se pueden utilizar para especificar el formato de entidades externas (datos no XML), como por ejemplo un archivo que contenga una imagen. Dichas entidades externas no las analizará un procesador XML, sino que serán tratadas por el programa que procese el documento.

EJEMPLO En la DTD del siguiente documento XML, se indica que los elementos “fruta” que se escriban, tienen que incluir obligatoriamente el atributo **foto**, cuyo valor será una entidad y, para indicar el formato de dicha entidad, se usa la notación **gif**:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE frutas [
  <!ELEMENT frutas (fruta)*>
  <!ELEMENT fruta EMPTY>
  <!ATTLIST fruta foto ENTITY #REQUIRED>

  <!ENTITY manzana SYSTEM "manzana.gif" NDATA gif>
  <!ENTITY naranja SYSTEM "naranja.gif" NDATA gif>

  <!NOTATION gif SYSTEM "image/gif">
]>

<frutas>
  <fruta foto="manzana"/>
  <fruta foto="naranja"/>
</frutas>
```

- En la DTD de este ejemplo se está indicando que los valores –datos– de las entidades (**manzana** y **naranja**) van a ser cargados desde una URI (*Uniform Resource Identifier*, Identificador Uniforme de Recurso). En este caso, se hace referencia a los archivos externos “manzana.gif” y “naranja.gif”.
- Con **NDATA** (*Notation Data*) se ha asociado a las entidades **manzana** y **naranja** con la notación **gif**.
- La notación **gif** es una declaración del tipo *MIME image/gif*.

EJEMPLO Si en el sistema existe, por ejemplo, un programa llamado “procesadorGIF.exe” en la carpeta “aplicaciones” capaz de procesar imágenes GIF (*Graphics Interchange Format*, Formato de Intercambio de Gráficos), también se podría escribir:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE frutas [
  <!ELEMENT frutas (fruta)*>
  <!ELEMENT fruta EMPTY>
  <!ATTLIST fruta foto ENTITY #REQUIRED>

  <!ENTITY manzana SYSTEM "manzana.gif" NDATA gif>
  <!ENTITY naranja SYSTEM "naranja.gif" NDATA gif>

  <!NOTATION gif SYSTEM "aplicaciones/procesadorGIF.exe">
]>
```

```
<frutas>
  <fruta foto="manzana"/>
  <fruta foto="naranja"/>
</frutas>
```

Notación pública - **PUBLIC**

EJEMPLO En la declaración de una notación se puede indicar un identificador público estándar, como por ejemplo, *GIF 1.0*:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE frutas [
  <!ELEMENT frutas (fruta)*>
  <!ELEMENT fruta EMPTY>
  <!ATTLIST fruta foto ENTITY #REQUIRED>

  <!ENTITY manzana SYSTEM "manzana.gif" NDATA gif>
  <!ENTITY naranja SYSTEM "naranja.gif" NDATA gif>

  <!NOTATION gif PUBLIC "GIF 1.0">
]>

<frutas>
  <fruta foto="manzana"/>
  <fruta foto="naranja"/>
</frutas>
```

EJEMPLO En la notación escrita en la DTD del siguiente documento XML, se ha declarado el tipo *MIME imagen/gif* e indicado el identificador público estándar *GIF 1.0*:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE frutas [
  <!ELEMENT frutas (fruta)*>
  <!ELEMENT fruta EMPTY>
  <!ATTLIST fruta foto ENTITY #REQUIRED>

  <!ENTITY manzana SYSTEM "manzana.gif" NDATA gif>
  <!ENTITY naranja SYSTEM "naranja.gif" NDATA gif>

  <!NOTATION gif PUBLIC "GIF 1.0" "image/gif">
]>

<frutas>
  <fruta foto="manzana"/>
  <fruta foto="naranja"/>
</frutas>
```


Atributos cuyo valor es el nombre de una notación

En una DTD, pueden existir elementos con atributos cuyo valor sea el nombre de una notación.

EJEMPLO En la DTD del siguiente documento XML, se indica que los elementos "documento" que se escriban, tienen que incluir obligatoriamente el atributo **version**, cuyo valor será una notación (**h4** o **h5**):

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE documentos [
  <!ELEMENT documentos (documento)*>
  <!ELEMENT documento (#PCDATA)>
  <!ATTLIST documento version NOTATION (h4|h5) #REQUIRED>

  <!NOTATION h5 PUBLIC "HTML 5">
  <!NOTATION h4 PUBLIC "HTML 4.01">
]>

<documentos>
  <documento version="h4"><!-- Código del documento 1. --></documento>
  <documento version="h5"><!-- Código del documento 2. --></documento>
  <documento version="h5"><!-- Código del documento 3. --></documento>
  <documento version="h4"><!-- Código del documento 4. --></documento>
</documentos>
```

- *HTML 5* y *HTML 4.01* son identificadores públicos.

4.1.10. Secciones condicionales

En DTD externas se pueden definir las secciones **IGNORE** e **INCLUDE**, para ignorar o incluir declaraciones. Las sintaxis empleadas para ello son:

```
<![ IGNORE [ declaraciones ] ]>
```

```
<![ INCLUDE [ declaraciones ] ]>
```

El uso de las secciones condicionales suele estar ligado a entidades paramétricas.

EJEMPLO Si en un archivo llamado "*persona.dtd*" se ha escrito:

```
<![ %datos_basicos; [
  <!ELEMENT persona (nombre, edad)>
]]>

<![ %datos_ampliados; [
  <!ELEMENT persona (nombre, apellidos, edad, ciudad)>
]]>

<!ELEMENT nombre (#PCDATA)>
<!ELEMENT apellidos (#PCDATA)>
<!ELEMENT edad (#PCDATA)>
<!ELEMENT ciudad (#PCDATA)>
```

El siguiente documento XML sería válido:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE persona SYSTEM "persona.dtd" [
  <!ENTITY % datos_basicos "INCLUDE">
  <!ENTITY % datos_ampliados "IGNORE">
]>

<persona>
  <nombre>Elsa</nombre>
  <edad>23</edad>
</persona>
```

También sería válido el documento:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE persona SYSTEM "persona.dtd" [
  <!ENTITY % datos_basicos "IGNORE">
  <!ENTITY % datos_ampliados "INCLUDE">
]>

<persona>
  <nombre>Ana</nombre>
  <apellidos>Sanz Tin</apellidos>
  <edad>19</edad>
  <ciudad>Pamplona</ciudad>
</persona>
```

- Obsérvese que, en este ejemplo, en los dos documentos XML asociados a la DTD externa escrita en el archivo ***persona.dtd***, se ha indicado –por medio de **IGNORE** e **INCLUDE**– si el elemento “persona” tiene que contener 2 ó 4 hijos, es decir, (“nombre” y “edad”) o (“nombre”, “apellidos”, “edad” y “ciudad”).

4.1.11. Espacios de nombres en DTD

EJEMPLO Dado el siguiente documento XML (visto en el apartado [espacios de nombres](#) del tutorial de XML de Abrirllave) bien formado, pero no validado, donde se utilizan dos espacios de nombres (*XML Namespaces*):

```
<?xml version="1.0" encoding="UTF-8"?>
<el:ejemplo xmlns:el="http://www.abrirllave.com/ejemplo1">

  <el:carta>
    <el:palo>Corazones</el:palo>
    <el:numero>7</el:numero>
  </el:carta>
```

```

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

```

Se podría escribir dicho documento XML con una DTD interna como se muestra a continuación:

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE e1:ejemplo [
  <!ELEMENT e1:ejemplo (e1:carta, e2:carta)>
  <!ATTLIST e1:ejemplo xmlns:e1 CDATA #FIXED "http://www.abrirllave.com/ejemplo1">
  <!ELEMENT e1:carta (e1:palo, e1:numero)>
  <!ELEMENT e1:palo (#PCDATA)>
  <!ELEMENT e1:numero (#PCDATA)>

  <!ELEMENT e2:carta (e2:carnes, e2:pescados)>
  <!ATTLIST e2:carta xmlns:e2 CDATA #FIXED "http://www.abrirllave.com/ejemplo2">
  <!ELEMENT e2:carnes (e2:filete_de_tenera, e2:solomillo_a_la_pimienta)>
  <!ELEMENT e2:pescados (e2:lenguado_al_horno, e2:merluza_en_salsa_verde)>
  <!ELEMENT e2:filete_de_tenera EMPTY>
  <!ATTLIST e2:filete_de_tenera precio CDATA #REQUIRED>
  <!ELEMENT e2:solomillo_a_la_pimienta EMPTY>
  <!ATTLIST e2:solomillo_a_la_pimienta precio CDATA #REQUIRED>
  <!ELEMENT e2:lenguado_al_horno EMPTY>
  <!ATTLIST e2:lenguado_al_horno precio CDATA #REQUIRED>
  <!ELEMENT e2:merluza_en_salsa_verde EMPTY>
  <!ATTLIST e2:merluza_en_salsa_verde precio CDATA #REQUIRED>
]>

<e1:ejemplo xmlns:e1="http://www.abrirllave.com/ejemplo1">

  <e1:carta>
    <e1:palo>Corazones</e1:palo>
    <e1:numero>7</e1:numero>
  </e1:carta>

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

```

4.1.12. Comentarios

En una DTD asociada a un documento XML, se pueden escribir comentarios entre los caracteres "`<!--`" y "`-->`". Por ejemplo:

```
<!-- Esto es un comentario escrito en una DTD -->
```

EJEMPLO En la DTD interna del siguiente documento se han escrito dos comentarios:

```
<!-- Ejemplo de documento XML con comentarios en su DTD
interna, del Tutorial de DTD de Abrirllave.com -->
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ciudades [
    <!ELEMENT ciudades (ciudad*)>
    <!ELEMENT ciudad (#PCDATA)>
    <!-- pais es atributo del elemento ciudad -->
    <!ATTLIST ciudad pais CDATA #REQUIRED>
]>

<ciudades>
    <ciudad pais="Italia">Roma</ciudad>
    <ciudad pais="Francia">París</ciudad>
    <ciudad pais="Alemania">Berlín</ciudad>
    <ciudad pais="">Viena</ciudad>
</ciudades>
```

4.1.13. Recursos de DTD

- **Tutorial de DTD**
<http://www.abrirllave.com/dtd/>
- **Chuleta de DTD**
<http://www.abrirllave.com/dtd/chuleta-de-dtd.php>
- **Cómo validar con XML Copy Editor un documento XML asociado a una DTD**
<http://www.abrirllave.com/dtd/como-validar-con-xml-copy-editor-un-documento-xml-asociado-a-una-dtd.php>
- **Ejercicios resueltos de DTD**
<http://www.abrirllave.com/dtd/ejercicios-resueltos.php>
- **Presentación PDF**
<http://www.abrirllave.com/dtd/presentacion.php>

4.2. XSD (XML Schema)

Contenidos del tutorial de XSD www.abrirllave.com/xsd/

[4.2.1. Qué es XSD](#)

[4.2.2. Validación de un documento XML con XSD](#)

Definición de un espacio de nombres

Elementos globales y locales en XSD

`elementFormDefault="unqualified"`

`elementFormDefault="qualified"`

Validación de un sitemap XML

[4.2.3. Elementos simples](#)

Tipos de declaración de elementos simples (`fixed`, `default`)

[4.2.4. Atributos](#)

Tipos de declaración de atributos (`fixed`, `default`, `optional`, `required`)

[4.2.5. Restricciones \(facetas\)](#)

`xs:minExclusive` y `xs:maxInclusive`

`xs:enumeration`

`xs:pattern`

`xs:length`

`xs:whiteSpace`

[4.2.6. Extensiones](#)

`xs:extension (complexContent)`

`xs:extension (simpleContent)`

[4.2.7. Elementos complejos](#)

Elemento vacío

Contenido mixto

[4.2.8. Indicadores](#)

Indicadores de orden (`xs:sequence`, `xs:all`, `xs:choice`)

Indicadores de ocurrencia (`maxOccurs`, `minOccurs`)

Indicadores de grupo (`xs:group`, `xs:attributeGroup`)

[4.2.9. Recursos de XSD](#)

4.2.1. Qué es XSD

XSD (*XML Schema Definition*) es un lenguaje, también llamado simplemente **XML Schema**, que sirve para definir la estructura de un documento XML, permitiendo su validación.

En los siguientes enlaces se puede consultar la *W3C Recommendation* de XSD, en la cual están basados estos apuntes de introducción a XML Schema:

- **XML Schema Part 0: Primer**
www.w3.org/TR/xmlschema-0/
- **XML Schema Part 1: Structures**
www.w3.org/TR/xmlschema-1/
- **XML Schema Part 2: Datatypes**
www.w3.org/TR/xmlschema-2/

Los contenidos –teoría y ejemplos– de estos apuntes, están redactados asumiendo que el lector ya posee conocimientos previos de XML. De no ser así, se recomienda consultar el tutorial de XML en www.abrirllave.com/xml/.

4.2.2. Validación de un documento XML con XSD

EJEMPLO Se quiere almacenar una lista de marcadores de páginas web, guardando de cada uno de ellos su nombre, una descripción y su URL. Para ello, se ha escrito el siguiente documento XML ("**marcadores.xml**") asociado al archivo "**marcadores.xsd**":

```
<?xml version="1.0" encoding="UTF-8"?>
<marcadores xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="marcadores.xsd">
  <pagina>
    <nombre>Abrirllave</nombre>
    <descripcion>Tutoriales de informática.</descripcion>
    <url>http://www.abrirllave.com/</url>
  </pagina>
  <pagina>
    <nombre>Wikipedia</nombre>
    <descripcion>La enciclopedia libre.</descripcion>
    <url>http://www.wikipedia.org/</url>
  </pagina>
  <pagina>
    <nombre>W3C</nombre>
    <descripcion>World Wide Web Consortium.</descripcion>
    <url>http://www.w3.org/</url>
  </pagina>
</marcadores>
```

- Para vincular un esquema a un documento XML, es obligatorio que este último haga referencia al espacio de nombres **http://www.w3.org/2001/XMLSchema-instance**. Para ello, habitualmente se utiliza el prefijo **xsi**.
- El atributo **noNameSchemaLocation** permite referenciar a un archivo con la definición de un esquema que no tiene ningún espacio de nombres asociado. En este caso, dicho archivo es **"marcadores.xsd"**.

El esquema XML guardado en **"marcadores.xsd"** y que permita validar el documento XML **"marcadores.xml"** podría ser:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="marcadores">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="pagina" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="nombre" type="xs:string"/>
              <xs:element name="descripcion" type="xs:string"/>
              <xs:element name="url" type="xs:string"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Para estar bien formado, un esquema XML tiene que cumplir las mismas reglas de sintaxis que cualquier otro documento XML.

Por otra parte, hay que tener en cuenta que, en todos los esquemas XML, el elemento raíz es **"schema"**. Ahora bien, para escribirlo, es muy común utilizar el prefijo **xsd** o **xs**.

Con **xmlns:xs="http://www.w3.org/2001/XMLSchema"** se ha indicado que:

- Los elementos y tipos de datos utilizados en el esquema pertenecen al espacio de nombres **http://www.w3.org/2001/XMLSchema**.
- Dichos elementos y tipos de datos deben llevar el prefijo **xs** (**xs:schema**, **xs:element**, **xs:complexType**, **xs:string**...).

Fíjese también que:

- Los elementos **"marcadores"** y **"página"** son de tipo complejo (**complexType**), ya que, contienen a otros elementos.
- **sequence** indica que los elementos hijo deben aparecer, en el documento XML, en el mismo orden en el que sean declarados en el esquema.
- Los elementos **"nombre"**, **"descripción"** y **"url"** son de tipo simple (**string** en este caso) y no pueden contener a otros elementos.
- Mediante **maxOccurs="unbounded"** se ha indicado que pueden aparecer ilimitados elementos **"página"** en el documento XML.

Ejercicio

- [Validar un documento XML](#)

Definición de un espacio de nombres

EJEMPLO En el siguiente documento XML se ha definido un espacio de nombres escribiendo **xmlns:mar="http://www.abrirllave.com/marcadores"**:

```
<?xml version="1.0" encoding="UTF-8"?>
<mar:marcadores xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.abrirllave.com/marcadores marcadores.xsd"
xmlns:mar="http://www.abrirllave.com/marcadores">
  <mar:pagina>
    <mar:nombre>Abrirllave</mar:nombre>
    <mar:descripcion>Tutoriales de informática.</mar:descripcion>
    <mar:url>http://www.abrirllave.com/</mar:url>
  </mar:pagina>
  <mar:pagina>
    <mar:nombre>Wikipedia</mar:nombre>
    <mar:descripcion>La enciclopedia libre.</mar:descripcion>
    <mar:url>http://www.wikipedia.org/</mar:url>
  </mar:pagina>
  <mar:pagina>
    <mar:nombre>W3C</mar:nombre>
    <mar:descripcion>World Wide Web Consortium.</mar:descripcion>
    <mar:url>http://www.w3.org/</mar:url>
  </mar:pagina>
</mar:marcadores>
```

En el atributo **schemaLocation** se pueden escribir parejas de valores:

- En el primer valor de cada pareja, hay que hacer referencia a un espacio de nombres.
- En el segundo valor, se tiene que indicar la ubicación de un archivo donde hay un esquema de ese espacio de nombres.

En cuanto al archivo **"marcadores.xsd"**, ahora su código podría ser:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.abrirllave.com/marcadores"
xmlns="http://www.abrirllave.com/marcadores"
elementFormDefault="qualified">
  <xs:element name="marcadores">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="pagina" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="nombre" type="xs:string"/>
              <xs:element name="descripcion" type="xs:string"/>
              <xs:element name="url" type="xs:string"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```


- En el atributo **targetNamespace** se está indicando que los elementos definidos en este esquema ("marcadores", "página", "nombre", "descripción" y "url"), provienen del espacio de nombres **http://www.abrirllave.com/marcadores**.
- **xmlns="http://www.abrirllave.com/marcadores"** especifica que este es el espacio de nombres por defecto.
- El atributo **elementFormDefault="qualified"** indica que todos los elementos declarados localmente en el esquema tienen que estar calificados, es decir, tienen que pertenecer a un espacio de nombres. Por esta razón, en **"marcadores.xml"** se han escrito con el prefijo **mar**.

Elementos globales y locales en XSD

Los elementos pueden ser globales o locales:

- Los elementos globales son hijos directos del elemento raíz, **<xs:schema>** en este caso. En el ejemplo que estamos tratando, solamente existe un elemento global: **<xs:element name="marcadores">**.
- Los elementos locales son el resto de elementos.

Cuando se define un espacio de nombres, los elementos globales tienen que estar calificados, obligatoriamente.

elementFormDefault="unqualified"

EJEMPLO En el supuesto de que el valor del atributo **elementFormDefault** fuese **"unqualified"**, para que **"marcadores.xml"** fuese válido, se podría escribir algo similar a:

```
<?xml version="1.0" encoding="UTF-8"?>
<mar:marcadores xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.abrirllave.com/marcadores marcadores.xsd"
xmlns:mar="http://www.abrirllave.com/marcadores">
  <pagina>
    <nombre>Abrirllave</nombre>
    <descripcion>Tutoriales de informática.</descripcion>
    <url>http://www.abrirllave.com/</url>
  </pagina>
  <pagina>
    <nombre>Wikipedia</nombre>
    <descripcion>La enciclopedia libre.</descripcion>
    <url>http://www.wikipedia.org/</url>
  </pagina>
  <pagina>
    <nombre>W3C</nombre>
    <descripcion>World Wide Web Consortium.</descripcion>
    <url>http://www.w3.org/</url>
  </pagina>
</mar:marcadores>
```

- **"unqualified"** es el valor por defecto del atributo **elementFormDefault**.

elementFormDefault="qualified"

EJEMPLO Si el atributo `elementFormDefault` se definiese `"qualified"`, también sería válido el siguiente documento XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<marcadores xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.abrirllave.com/marcadores marcadores.xsd"
xmlns="http://www.abrirllave.com/marcadores">
  <pagina>
    <nombre>Abrirllave</nombre>
    <descripcion>Tutoriales de informática.</descripcion>
    <url>http://www.abrirllave.com/</url>
  </pagina>
  <pagina>
    <nombre>Wikipedia</nombre>
    <descripcion>La enciclopedia libre.</descripcion>
    <url>http://www.wikipedia.org/</url>
  </pagina>
  <pagina>
    <nombre>W3C</nombre>
    <descripcion>World Wide Web Consortium.</descripcion>
    <url>http://www.w3.org/</url>
  </pagina>
</marcadores>
```

Ahora, ningún elemento lleva prefijo, al igual que el espacio de nombres al que pertenecen: `xmlns="http://www.abrirllave.com/marcadores"`

Validación de un sitemap XML

EJEMPLO Para validar el archivo *"sitemap.xml"* que da solución al ejercicio de XML propuesto en el siguiente enlace:

- <http://www.abrirllave.com/xml/ejercicio-crear-un-sitemap-xml.php>

Obsérvese que, es necesario añadir el texto resaltado que se muestra a continuación:

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.sitemaps.org/schemas/sitemap/0.9
http://www.sitemaps.org/schemas/sitemap/0.9/sitemap.xsd"
xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
  <url>
    <loc>http://www.ejemplos-de-abrirllave.com/</loc>
    <lastmod>2016-09-30</lastmod>
    <priority>0.8</priority>
  </url>
  <url>
    <loc>http://www.ejemplos-de-abrirllave.com/contactar.html</loc>
    <lastmod>2016-09-30</lastmod>
    <priority>0.3</priority>
  </url>
```

```

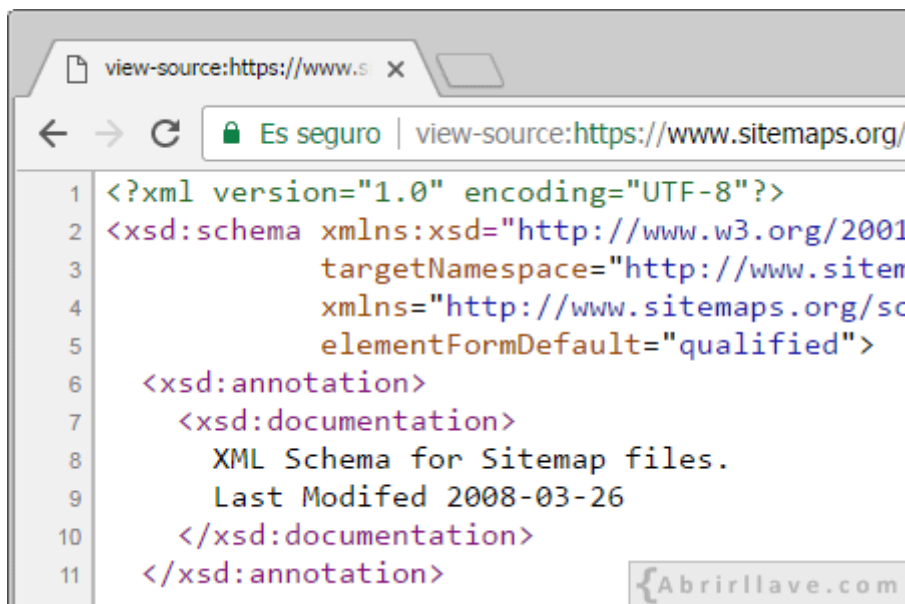
<url>
  <loc>http://www.ejemplos-de-abrirllave.com/productos/impresora.html</loc>
  <lastmod>2016-09-30</lastmod>
  <priority>0.5</priority>
</url>
<url>
  <loc>http://www.ejemplos-de-abrirllave.com/productos/monitor.html</loc>
  <lastmod>2016-09-30</lastmod>
  <priority>0.5</priority>
</url>
<url>
  <loc>http://www.ejemplos-de-abrirllave.com/productos/teclado.html</loc>
  <lastmod>2016-09-30</lastmod>
  <priority>0.5</priority>
</url>
</urlset>

```

Al visualizar en un navegador web el código fuente del archivo **"sitemap.xsd"** ubicado en:

- <http://www.sitemaps.org/schemas/sitemap/0.9/sitemap.xsd>

En pantalla se verá algo parecido a:



Nótese que, en el archivo **"sitemap.xsd"**, la etiqueta **<xsd:schema>** contiene los atributos **xmlns:xsd**, **targetNamespace**, **xmlns** y **elementFormDefault**.

```

<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.sitemaps.org/schemas/sitemap/0.9"
  xmlns="http://www.sitemaps.org/schemas/sitemap/0.9"
  elementFormDefault="qualified">

```

Ejercicios

- [Validar un sitemap XML](#)
- [Mensaje entre personas](#)

4.2.3. Elementos simples

Los elementos simples solamente pueden contener texto (caracteres). Dicho de otro modo, los elementos simples no pueden contener a otro u otros elementos (hijos), ni tampoco pueden tener atributos. Ahora bien, el texto contenido en un elemento simple, puede ser de diferentes tipos de datos predefinidos en **W3C XML Schema** o definidos por el usuario (programador).

Los tipos de datos predefinidos pueden ser primitivos (**string**, **boolean**, **decimal**...) o derivados de estos (**integer**, **ID**, **IDREF**...). Véase en el siguiente enlace, una imagen donde se puede ver la relación que existe entre todos ellos:

- <https://goo.gl/1K2l2I>: este enlace lleva a una carpeta compartida de Google Drive –pública en la Web– donde se puede acceder a la imagen "**w3c-xml-schema-type-hierarchy.gif**".

Para definir un elemento simple se puede utilizar la siguiente sintaxis:

```
<xs:element name="nombre_del_elemento" type="tipo_de_dato"/>
```

EJEMPLO Para los siguientes elementos XML:

```
<nombre>Elsa</nombre>
<edad>23</edad>
```

Sus definiciones pueden ser:

```
<xs:element name="nombre" type="xs:string"/>
<xs:element name="edad" type="xs:integer"/>
```

Tipos de declaración de elementos simples (**fixed**, **default**)

Si se quiere indicar que un valor es fijo (**fixed**), se puede escribir, por ejemplo:

```
<xs:element name="mes" type="xs:string" fixed="agosto"/>
```

También, se puede especificar un valor por defecto (**default**), por ejemplo, tecleando:

```
<xs:element name="mes" type="xs:string" default="agosto"/>
```

Ejercicios resueltos

- [Definición de elementos simples](#)
- [Puerta cerrada y ventana abierta](#)

4.2.4. Atributos

Para definir un atributo se puede emplear la siguiente sintaxis:

```
<xs:attribute name="nombre_del_atributo" type="tipo_de_dato"/>
```

EJEMPLO Para el elemento “curso” siguiente, donde aparece el atributo “grupo”:

```
<curso grupo="B">2</curso>
```

Sus definiciones pueden ser:

```
<xs:element name="curso" type="xs:integer"/>
<xs:attribute name="grupo" type="xs:string"/>
```

- Todos los atributos pueden tomar por valor tipos simples.
- Por otra parte, cuando un elemento tiene al menos un atributo –como es el caso del elemento “curso” en este ejemplo– dicho elemento se dice que es complejo.

Tipos de declaración de atributos (**fixed**, **default**, **optional**, **required**)

Para indicar que el valor de un atributo es fijo (**fixed**), es posible escribir, por ejemplo:

```
<xs:attribute name="grupo" type="xs:string" fixed="B"/>
```

Para especificar el valor por defecto (**default**) de un atributo, se puede escribir:

```
<xs:attribute name="grupo" type="xs:string" default="B"/>
```

Para indicar que un atributo es obligatorio (**required**) escribirlo, se puede teclear:

```
<xs:attribute name="grupo" type="xs:string" use="required"/>
```

Por defecto, si no se indica nada, el atributo será opcional (**optional**).

Ejercicio resuelto

- [Fichas de personas](#)

4.2.5. Restricciones (facetas)

XML Schema permite definir restricciones a los posibles valores de los tipos de datos. Dichas restricciones se pueden establecer en diferentes aspectos, llamados facetas.

Dicho de otro modo, **las facetas permiten definir restricciones sobre los posibles valores de atributos o elementos**. Las facetas que pueden utilizarse son:

Facetas de XSD	
Faceta	Descripción
xs:length	Especifica una longitud fija.
xs:minLength	Especifica una longitud mínima.
xs:maxLength	Especifica una longitud máxima.
xs:pattern	Especifica un patrón de caracteres admitidos.
xs:enumeration	Especifica una lista de valores admitidos.
xs:whiteSpace	Especifica cómo se debe tratar a los posibles espacios en blanco, las tabulaciones, los saltos de línea y los retornos de carro que puedan aparecer.
xs:maxInclusive	Especifica que el valor debe ser menor o igual que el indicado.
xs:maxExclusive	Especifica que el valor debe ser menor que el indicado.
xs:minExclusive	Especifica que el valor debe ser mayor que el indicado.
xs:minInclusive	Especifica que el valor debe ser mayor o igual que el indicado.
xs:totalDigits	Especifica el número máximo de dígitos que puede tener un número.
xs:fractionDigits	Especifica el número máximo de decimales que puede tener un número.

Seguidamente, se muestran algunos ejemplos de restricciones definidas con una o más facetas:

xs:minExclusive y xs:maxInclusive

EJEMPLO En el siguiente código se define un elemento llamado "mes" con la restricción de que el valor que tome no pueda ser menor que 1 ni mayor que 12:

```
<xs:element name="mes">
  <xs:simpleType>
    <xs:restriction base="xs:integer">
      <xs:minInclusive value="1"/>
      <xs:maxInclusive value="12"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

- **xs:simpleType** permite definir un tipo simple y especificar sus restricciones.
- **xs:restriction** sirve para definir restricciones de un **xs:simpleType** (como se ha hecho en este ejemplo). También sirve para definir restricciones de un **xs:simpleContent** o de un **xs:complexContent**. Estos elementos se estudiarán más adelante.
- En el atributo **base** se indica el tipo de dato a partir del cual se define la restricción.
- **xs:minInclusive** sirve para especificar que el valor debe ser mayor o igual que el indicado en su atributo **value**, (en este caso, mayor o igual que 1).
- **xs:maxInclusive** sirve para especificar que el valor debe ser menor o igual que el indicado en su atributo **value**, (en este caso, menor o igual que 12).

También se podría haber escrito:

```
<xs:element name="mes" type="numeroMes"/>
<xs:simpleType name="numeroMes">
  <xs:restriction base="xs:integer">
    <xs:minInclusive value="1"/>
    <xs:maxInclusive value="12"/>
  </xs:restriction>
</xs:simpleType>
```

Haciendo esto, el tipo **numeroMes** definido, podría ser utilizado por otros elementos, ya que, no está contenido en el elemento "mes".

Ejercicios resueltos

- [Edad entre 0 y 130 años](#)
- [Precios de tres dígitos](#)

xs:enumeration

EJEMPLO En el siguiente ejemplo se define un elemento llamado "color" con la restricción de que los únicos valores admitidos son: "**verde**", "**amarillo**" y "**rojo**".

```
<xs:element name="color">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="verde"/>
      <xs:enumeration value="amarillo"/>
      <xs:enumeration value="rojo"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

- **xs:enumeration** sirve para definir una lista de valores admitidos.

Ejercicio resuelto

- [Tipo de vehículo](#)

xs:pattern

EJEMPLO En el siguiente ejemplo se define un elemento llamado "letra" con la restricción de que el único valor admitido es una de las letras minúsculas de la "a" a la "z":

```
<xs:element name="letra">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="[a-z]" />
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

- **xs:pattern** sirve para definir un patrón de caracteres admitidos (en este caso se admite una única letra minúscula de la "a" a la "z"). El valor del patrón tiene que ser una expresión regular.

Ejercicios resueltos

- [Iniciales de personas famosas](#)
- [Iniciales al revés](#)
- [Respuestas admitidas](#)
- [Números y letras](#)
- [Escribir expresiones regulares](#)
- [Letras admitidas](#)

xs:length

EJEMPLO En el siguiente ejemplo se define un elemento llamado "clave" con la restricción de que su valor tiene que ser una cadena de, exactamente, doce caracteres:

```
<xs:element name="clave">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:length value="12" />
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

- **xs:length** sirve para especificar una longitud fija.

Ejercicios resueltos

- [Longitud fija de una clave](#)
- [Longitud mínima y máxima de una clave](#)

xs:whiteSpace

EJEMPLO En el siguiente ejemplo se define un elemento llamado "dirección" con la restricción de que los espacios en blanco, las tabulaciones, los saltos de línea y los retornos de carro que aparezcan en él, se deben mantener (**preserve**):

```
<xs:element name="direccion">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:whiteSpace value="preserve"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

- **xs:whiteSpace** sirve para especificar cómo se debe tratar a los posibles espacios en blanco, las tabulaciones, los saltos de línea y los retornos de carro que puedan aparecer.

En vez de **preserve** también se puede utilizar:

- **replace** para sustituir todas las tabulaciones, los saltos de línea y los retornos de carro por espacios en blanco.
- **collapse** para, después de reemplazar todas las tabulaciones, los saltos de línea y los retornos de carro por espacios en blanco, eliminar todos los espacios en blanco únicos y sustituir varios espacios en blanco seguidos por un único espacio en blanco.

4.2.6. Extensiones

xs:extension sirve para extender un elemento **simpleType** o **complexType**.

xs:extension (complexContent)

EJEMPLO Dado el siguiente documento XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<fichas xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="fichas.xsd">
  <ficha numero="1">
    <nombre>Eva</nombre>
    <edad>25</edad>
    <ciudad>París</ciudad>
    <pais>Francia</pais>
  </ficha>
  <ficha numero="2">
    <nombre>Giovanni</nombre>
    <edad>26</edad>
    <ciudad>Florenzia</ciudad>
    <pais>Italia</pais>
  </ficha>
</fichas>
```

Y el archivo **"fichas.xsd"** que permite validarlo:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="fichas">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="ficha" type="infoPersonaAmpliada"
          maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:complexType name="infoPersonaAmpliada">
    <xs:complexContent>
      <xs:extension base="infoPersona">
        <xs:sequence>
          <xs:element name="ciudad" type="xs:string"/>
          <xs:element name="pais" type="xs:string"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="infoPersona">
    <xs:sequence>
      <xs:element name="nombre" type="xs:string"/>
      <xs:element name="edad" type="edadPersona"/>
    </xs:sequence>
    <xs:attribute name="numero" type="xs:integer"/>
  </xs:complexType>

  <xs:simpleType name="edadPersona">
    <xs:restriction base="xs:integer">
      <xs:minExclusive value="-1"/>
      <xs:maxExclusive value="131"/>
    </xs:restriction>
  </xs:simpleType>

</xs:schema>
```

- Obsérvese que, **infoPersonaAmpliada** se basa en **infoPersona**, añadiéndole dos elementos: "ciudad" y "país".
- En cuanto a **xs:complexContent**, sirve para definir restricciones o extensiones a un tipo complejo (**complexType**).

Ejercicio resuelto

- [Información de persona ampliada](#)

xs:extension (simpleContent)

xs:simpleContent permite definir restricciones o extensiones a elementos que solo contienen datos, es decir, no contienen a otros elementos.

EJEMPLO El siguiente archivo "*precios.xsd*":

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="precios">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="precio" maxOccurs="unbounded">
          <xs:complexType>
            <xs:simpleContent>
              <xs:extension base="xs:decimal">
                <xs:attribute name="moneda" type="xs:string"/>
              </xs:extension>
            </xs:simpleContent>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Permite validar el siguiente documento XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<precios xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="precios.xsd">
  <precio moneda="Euro">5</precio>
  <precio moneda="Dólar">6.2</precio>
  <precio moneda="Libra esterlina">4.3</precio>
</precios>
```

- Nótese que, utilizando **xs:extension**, al elemento "precio" se le ha incorporado el atributo **moneda**.

Ejercicios resueltos

- [Precios de artículos](#)
- [Información de ubicaciones](#)
- [Colores de muebles](#)

4.2.7. Elementos complejos

Un elemento es complejo (**complexType**) cuando contiene uno o más elementos y/o atributos. De entre las posibles combinaciones de elementos y/o atributos que puede contener un elemento complejo (1 elemento y 0 atributos, 1 elemento y 1 atributo, 1 elemento y varios atributos, 0 elementos y 1 atributo...) cabe destacar las siguientes:

- **Un elemento complejo puede estar vacío**, es decir, no contener elementos ni texto, pero sí tener al menos un atributo.
- **Un elemento complejo puede contener contenido mixto**, es decir, contener uno o más elementos, además de texto. Por otra parte, podría tener atributos, o no.

Elemento vacío

EJEMPLO En el siguiente código se ha definido vacío el elemento "bola", no pudiendo contener ni otros elementos ni texto. Ahora bien, véase que sí tiene un atributo, llamado **numero**:

```
<xs:element name="bola">
  <xs:complexType>
    <xs:attribute name="numero" type="numeroDeBola"/>
  </xs:complexType>
</xs:element>

<xs:simpleType name="numeroDeBola">
  <xs:restriction base="xs:positiveInteger">
    <xs:minInclusive value="1"/>
    <xs:maxExclusive value="90"/>
  </xs:restriction>
</xs:simpleType>
```

- **xs:positiveInteger** indica que el valor del atributo **numero** debe ser un número entero mayor que cero.

Ejercicio resuelto

- [Números del bingo](#)

Contenido mixto

Fíjese que, en el siguiente código se ha definido el elemento "persona" de tipo complejo mixto (**mixed="true"**):

```
<xs:element name="persona">
  <xs:complexType mixed="true">
    <xs:sequence>
      <xs:element name="nombre" type="xs:string"/>
      <xs:element name="ciudad" type="xs:string"/>
      <xs:element name="edad" type="xs:positiveInteger"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Ejercicio resuelto

- [Información de personas en contenido mixto](#)

4.2.8. Indicadores

Los indicadores permiten establecer cómo se van a escribir –o utilizar– los elementos en un documento XML. Hay siete tipos de indicadores que se pueden clasificar en:

- **Indicadores de orden:** secuencia (**sequence**), todo (**all**) y elección (**choice**).
- **Indicadores de ocurrencia:** **maxOccurs** y **minOccurs**.
- **Indicadores de grupo:** de elementos (**group**) y de atributos (**attributeGroup**).

Indicadores de orden (**xs:sequence**, **xs:all**, **xs:choice**)

Mientras que **xs:sequence** sirve para especificar el orden en el que obligatoriamente deben aparecer los elementos hijo de un elemento, **xs:all** sirve para indicar que dichos elementos pueden aparecer en cualquier orden.

EJEMPLO El siguiente archivo "*lugar.xsd*":

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="lugar">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="ciudad">
          <xs:complexType>
            <xs:all>
              <xs:element name="nombre" type="xs:string"/>
              <xs:element name="pais" type="xs:string"/>
            </xs:all>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Permite validar el siguiente documento XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<lugar xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="lugar.xsd">
  <ciudad>
    <pais>Italia</pais>
    <nombre>Florenzia</nombre>
  </ciudad>
</lugar>
```

Por otra parte, **xs:choice** sirve para especificar que solamente se permite escribir uno de los elementos hijo. Por ejemplo, en este caso, se podría utilizar para indicar que habría que elegir entre escribir el "nombre" o escribir el "país" de la "ciudad", pero no ambos.

Indicadores de ocurrencia (**maxOccurs**, **minOccurs**)

maxOccurs y **minOccurs** permiten establecer, respectivamente, el número máximo y mínimo de veces que puede aparecer un determinado elemento. El valor por defecto para **maxOccurs** y **minOccurs** es 1.

EJEMPLO Dado el siguiente documento XML "*países.xml*":

```
<?xml version="1.0" encoding="UTF-8"?>
<países xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="países.xsd">
  <país>
    <nombre>Argentina</nombre>
    <ciudad>Buenos Aires</ciudad>
    <ciudad>Rosario</ciudad>
  </país>
  <país>
    <nombre>México</nombre>
    <ciudad>Guadalajara</ciudad>
    <ciudad>Monterrey</ciudad>
    <ciudad>Cancún</ciudad>
    <ciudad>Mérida</ciudad>
    <ciudad>Ciudad de México</ciudad>
  </país>
  <país>
    <nombre>Colombia</nombre>
  </país>
</países>
```

Considerando que se quiere especificar que:

- "país" pueda aparecer una o ilimitadas veces.
- "nombre" tenga que escribirse obligatoriamente, y solo una vez, dentro de "país".
- De cada "país" puedan escribirse de cero a cinco "ciudades".

El código del archivo "*países.xsd*" que permita validar "*países.xml*", podría ser:

```

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="países">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="pais" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="nombre" type="xs:string"/>
              <xs:element name="ciudad" type="xs:string"
                minOccurs="0" maxOccurs="5"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>

```

Indicadores de grupo (**xs:group**, **xs:attributeGroup**)

xs:group sirve para agrupar un conjunto de declaraciones de elementos relacionados.

EJEMPLO Dado el siguiente documento XML *"personas.xml"*:

```

<?xml version="1.0" encoding="UTF-8"?>
<personas xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="personas.xsd">
  <persona>
    <nombre>Eva</nombre>
    <edad>25</edad>
    <pais>Francia</pais>
    <telefono>999888777</telefono>
  </persona>
  <persona>
    <nombre>Giovanni</nombre>
    <edad>26</edad>
    <pais>Italia</pais>
    <telefono>111222333</telefono>
  </persona>
</personas>

```

Y el archivo *"personas.xsd"* que permite validarlo:

```

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="personas">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="persona" type="datosDePersona"
          maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:complexType name="datosDePersona">
    <xs:sequence>
      <xs:group ref="datosBasicos"/>
      <xs:element name="telefono" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>

  <xs:group name="datosBasicos">
    <xs:sequence>
      <xs:element name="nombre" type="xs:string"/>
      <xs:element name="edad" type="xs:positiveInteger"/>
      <xs:element name="pais" type="xs:string"/>
    </xs:sequence>
  </xs:group>

</xs:schema>

```

- Obsérvese que, se ha definido el grupo **datosBasicos**, el cual ha sido incorporado a la definición del tipo complejo **datosDePersona**.

Del mismo modo, **attributeGroup** sirve para definir un grupo de atributos. Por ejemplo, para validar el siguiente documento XML:

```

<?xml version="1.0" encoding="UTF-8"?>
<personas xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="personas.xsd">
  <persona nombre="Eva" edad="25" pais="Francia"/>
  <persona nombre="Giovanni" edad="26" pais="Italia"/>
</personas>

```

Se puede escribir el siguiente código en el archivo **"personas.xsd"**:


```

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="personas">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="persona" maxOccurs="unbounded">
          <xs:complexType>
            <xs:attributeGroup ref="datosDePersona"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:attributeGroup name="datosDePersona">
    <xs:attribute name="nombre" type="xs:string"/>
    <xs:attribute name="edad" type="xs:positiveInteger"/>
    <xs:attribute name="pais" type="xs:string"/>
  </xs:attributeGroup>

</xs:schema>

```

- En este caso, se ha definido el grupo de atributos **datosDePersona**, el cual ha sido incorporado a la definición del elemento **persona**.

Ejercicios resueltos

- [Panel de vuelos](#)
- [Factura](#)
- [Registro de conexiones](#)
- [Personal de departamentos](#)

4.2.9. Recursos de XSD

- **Tutorial de XSD**
<http://www.abrirllave.com/xsd/>
- **Ejercicios resueltos de XSD**
<http://www.abrirllave.com/xsd/ejercicios-resueltos.php>
- **Enunciados de los ejercicios resueltos**
<http://www.abrirllave.com/xsd/enunciados-de-los-ejercicios-resueltos.php>

Capítulo 5

Conversión y adaptación de documentos XML

Los contenidos de este capítulo están en desarrollo y serán incorporados al libro en una próxima edición.

Capítulo 6

Almacenamiento de información

Los contenidos de este capítulo están en desarrollo y serán incorporados al libro en una próxima edición.

Capítulo 7

Sistemas de gestión empresarial

Los contenidos de este capítulo están en desarrollo y serán incorporados al libro en una próxima edición.

EPÍLOGO

"Abrirllave.com" es un sitio web de tutoriales de informática y otros recursos relacionados.

Tutoriales de informática

En Abrirllave puede consultar, entre otros, los siguientes tutoriales:

- **Tutorial de CMD**
<http://www.abrirllave.com/cmd/>
- **Tutorial de desarrollo web**
<http://www.abrirllave.com/desarrollo-web/>
- **Tutorial de Google Sites**
<http://www.abrirllave.com/google-sites/>
- **Tutorial de lenguaje C**
<http://www.abrirllave.com/c/>
- **Tutorial de SEO (Search Engine Optimization)**
<http://www.abrirllave.com/seo/>

En la dirección web www.abrirllave.com/tutoriales.php puede consultar el listado de todos los tutoriales publicados en Abrirllave a día de hoy.

Abrirllave en redes sociales

Para mantenerse informado de la incorporación de nuevos contenidos a "Abrirllave.com", puede hacerlo a través de:

- **Facebook**
<https://www.facebook.com/Abrirllave>
- **Google+**
<https://plus.google.com/+Abrirllavecom>
- **Slideshare**
<https://www.slideshare.net/abrirllave>
- **Twitter**
<https://twitter.com/Abrirllave>

Publicar en Abrirllave

Si lo desea, en la página www.abrirllave.com/publicar.php se explica cómo puede compartir sus conocimientos de informática en "Abrirllave.com".