# Fundamentos de la Interacción Persona-Ordenador

# 5. XHTML

Luis Rodríguez Baena (luis.rodriguez@upsam.es)

Universidad Pontificia de Salamanca
Escuela Superior de Ingeniería y Arquitectura

# ¿Qué es XML?

- XHTML: *Extensible Hypertext Markup Language* (lenguaje de marcas hipertexto ampliable).
  - Es una versión de HTML compatible con XML.
- XML: *Extensible Markup Language* (Lenguaje de marcas ampliable).
  - Estándar del W3C que define una sintaxis genérica para marcar documentos con etiquetas de forma que sean legibles por humanos.
- El término *marca* de utiliza para indicar el significado de un carácter o un grupo de caracteres de un documento.
  - Permiten codificar texto electrónico para indicar el significado de una porción de texto mediante caracteres especiales: los caracteres de marcado.
    - ✓ Por ejemplo, un texto en negrita se colocará entre las marcas de inicio y fin de negrita.
      - Esto es lo que hacen formatos como RTF o HTML.
- Algunos lenguajes de marcas ya tienen caracteres de marcado predefinidos.
  - XML es ampliable: permite crear marcas propias con un significado específico dentro de un documento.
  - Proporciona la sintaxis necesaria para crear lenguajes de marcas a partir de la definición de sus propias etiquetas.
  - La flexibilidad le permite definir documentos de dominios muy distintos (datos bancarios, gráficos, vectoriales, fórmulas matemáticas, datos de localización geográfica, noticias, etc.).
- La sintaxis estándar permite que estos documentos sean explotados por programas que interactúen, traten y manipulen la información contenida.
  - Muchos lenguajes (como Java o .NET Framework) incluyen bibliotecas de clases para trabajar con datos XML

# ¿Qué es XML?
## Ventajas y utilidades

- Ventajas de XML.
  - Permite definir cualquier cosa, desde datos estructurados, documentos, gráficos, etc.
    - ✓ Sólo define la sintaxis del lenguaje.
  - Es fácil de leer e interpretar.
  - Ha sido diseñado para los protocolos de Internet.
    - ✓ Aunque también se utiliza en otros dominios (por ejemplo, en archivos de configuración).
  - Es independiente de la aplicación y portable.
    - ✓ Se puede utilizar para traspasar datos de un lenguaje a otro, de una plataforma a otra, de una aplicación a otra.
  - Utiliza caracteres Unicode, lo que permite su adaptación a múltiples lenguas.
  - La especificaciones son de libre uso.
- Utilidades.
  - Definición de formatos estándar de documentos.
    - ✓ Por ejemplo el formato OpenDocument (ODF), que se está adoptando por muchas administraciones o el Open XML de Microsoft.
  - Aplicaciones que necesitan almacenar datos de forma estructurada.
  - Gestores de bases de datos.
  - Persistencia de objetos para servicios Web.
  - Transferencia de datos mediante HTTP en comercio electrónico, B2B, B2C.
  - Migración entre distintas plataformas.
  - …

# Lenguajes de marcas

- Los lenguajes de marcas están formados por etiquetas (marcas, *tags*, *tokens*) que especifican la apariencia o la estructura de un documento.
- La unión de las etiquetas y el texto forman el código fuente del documento.
- Ese código fuente puede ser código abierto o propietario.
  - El formato de Word 2007 es código cerrado.
    - ✓ No se puede interpretar ni modificar si no es con aplicaciones específicas.
  - El formato RTF, HTML (o XML) es código abierto.
    - ✓ Puede entenderse y modificarse con herramientas estándar de edición.

# Lenguajes de marcas (II)

- XML (y XHTML) están basados en SGML (*Standard Generalized Markup Language*).
  - Lenguaje desarrollado a finales de los años 70 para definir documentos de texto.
  - Se utilizó en sectores que necesitaban administrar grandes volúmenes de información (militar, aeroespacial, gubernamental).
  - Fue el lenguaje elegido para aplicaciones que utilizaban los protocolos de Internet.
    - ✓ HTML está basado en SGML.
  - Problema: excesivamente complicado.
- En 1996 se planteó hacer una versión ligera de SGML que resolviera el mismo tipo de problemas: definir documentos.
  - En 1998 nace la especificación de XML 1.0.
- XML es un lenguaje de "meta-marcas".
  - No tiene un conjunto fijo de etiquetas.
  - Aporta la manera de definir etiquetas adaptadas a un dominio específico.
- XML es un "meta-lenguaje".
- Se utiliza para definir otros lenguajes (como XHTML).

# Introducción

- XHTML es una recomendación oficial del W3C que define una versión de HTML compatible con XML.
  - Redefine HTML como una aplicación XML.
    - ✓ HTML sería una versión SGML.
- Ventajas que aporta.
  - Al tratarse de documentos XML se pueden incorporar elementos de distintos espacios de nombre.
    - ✓ Permite mezclar el documento con otros vocabularios XML, por ejemplo, incluir gráficos vectoriales con SVG o expresiones matemáticas con MathML.
  - Al tratarse de documentos bien formados los analizadores sintácticos se pueden simplificar.
    - ✓ La libertad de HTML hace que su renderización en un navegador pueda ser más lenta.
  - Al tratarse de documentos XML se pueden utilizar las mismas herramientas para el tratamiento de los datos.
  - Cómo XML, se trata de un lenguaje ampliable.
    - ✓ Es relativamente fácil añadir elementos al lenguaje a través de módulos.
    - ✓ Permite la interoperabilidad entre distintas plataformas y aplicaciones de usuario.
- XHTML utiliza un conjunto de etiquetas similares a HTML.
  - Pone algunas limitaciones al uso de etiquetas y atributos HTML que el W3C considera obsoletos.

# Introducción
## Documentos bien formados

- Obliga a adaptar la escritura de las etiquetas y atributos del documento a las restricciones del lenguaje XML.
- El primer requisito de un documento XML es que debe tratarse de un documento XML bien formado:
  - Debe cumplir las especificaciones del lenguaje respecto a las reglas sintácticas y tener una estructura jerárquica estricta.
    1. Toda etiqueta de inicio debe tener una de cierre.
       - Si una etiqueta no tiene contenido debe tener también una etiqueta de cierre:
         ```
         <etiqueta></etiqueta>
         ```
       - Una etiqueta sin etiqueta de cierre deberá acabar con la secuencia °.
         ```
         <etiqueta />
         ```
    2. Se pueden anidar elementos, pero no superponer ni mezclar.
       - Esta combinación no es posible en XHTML (aunque si en HTML).
         ```
         <strong><em>Contenido</strong></em>
         ```
       - Se debería escribir así:
         ```
         <strong><em>Contenido</em></strong>
         ```

# Introducción
## Documentos bien formados (II)

- Características de los documentos bien formados (*continuación)*.
  3. Debe tener un único elemento raíz.
     - Un documento XML tiene una estructura de árbol, por lo que deberá tener un único elemento raíz: el elemento `html`.
  4. Los valores de los atributos deben estar entrecomillados.
     - Un elemento XML puede tener atributos (características especiales del elemento, como el URL de una imagen). El contenido de esos atributos deberá ir entrecomillado.

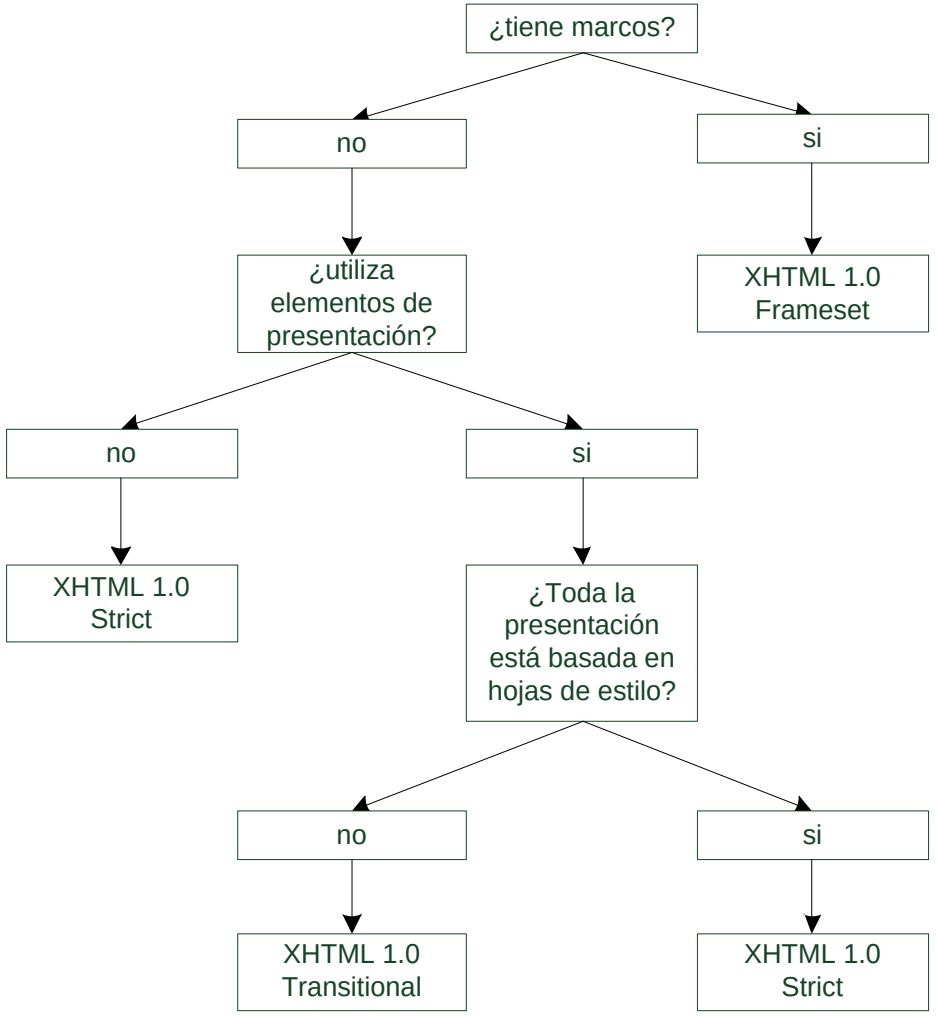
       ```
       <img src="imagen01.png" alt="casa" />
       ```
  5. Un elemento no puede tener dos atributos con el mismo nombre.
  6. No pueden aparecer signos `<` y `&` individuales dentro del contenido de un elemento o atributo.
     - El signo `<` se utiliza para indicar el comienzo de una etiqueta, por lo que no puede aparecer en el texto. En su lugar se utiliza la entidad `<` (*less than*).
     - Por esa razón, tampoco se puede utilizar el carácter `&`, ya que sería el comienzo de una entidad. En su lugar se utilizar `&`.

# Introducción
## Documentos validados

- Un documento validado es un documento bien formado en el que sus elementos cumplen una serie de reglas.
  - Esas normas pueden regular el número, nombre u orden de los elementos o atributos o el contenido de los mismos.
  - XML presenta dos mecanismos para asegurar esas reglas:
    - ✓ Definición del tipo de documento (DTD).
    - ✓ Esquemas XML.
- Una DTD hace una descripción formal de un vocabulario XML.
  - Determina:
    - ✓ Qué elementos (etiquetas) puede tener un documento.
    - ✓ Qué tipo de datos puede tener el contenido de una etiqueta.
    - ✓ Qué atributos y de qué tipo puede contener una etiqueta.
    - ✓ Qué jerarquía deben tener los elementos de un documento.
  - Las DTD pueden ser compartidas por varios documentos.
  - Permiten validar si esos documentos utilizan bien el lenguaje.
  - La declaración del tipo de documento (DOCTYPE) en la cabecera de un documento XHTML especifica que DTD utilizará ese documento.

# Introducción
## Tipos de documentos XHTML

- Los documentos XHTML deben ser documentos válidos.
  - Deben tener una declaración de tipo de documento.
- XHTML permite utilizar algunas de las siguientes DTD.
  - XHTML 1.0 Strict.
  - XHTML 1.0 Transitional.
  - XHTML 1.0 Frameset.
- La elección de una u otra DTD para el documento depende de:
  - ¿El documento utilizará marcos?
  - ¿Se van a utilizar elementos de presentación dentro del propio documento?
  - ¿Se va utilizar CSS o algún tipo de hoja de estilo para la presentación de la información?



# Introducción
## Tipos de documentos XHTML (II)

- XHTML 1.0 Strict.
  - No utiliza elementos o atributos de presentación (por ejemplo `<basefont>` o `<center>`) u otros elementos obsoletos (`<applet>`) .
    - ✓ Toda la presentación se realiza a través de hojas de estilo.
      - Esto incluye el color, la alineación, la fuente y su tamaño o aspectos visuales de tablas (grosor de la líneas, color del fondo, etc.).
- XHTML 1.0 Transitional.
  - Apareció como advertencia a los diseñadores de HTML 4.0 para indicar que los elementos y atributos de presentación estaban obsoletos.
    - ✓ Estos elementos están oficialmente en desuso y sus efectos se pueden conseguir de otras maneras.
  - La DTD de XHTML Transitional permite que los documentos que incluyan estos elementos obsoletos sean validos.
- XHTML 1.0 Frameset.
  - Se utiliza cuando se utilizan marcos como parte de la estructura del documento.
- Además existen dos DTD más para XHTML:
  - XHTML 1.1.
    - ✓ Elimina todos los elementos obsoletos de XHTML 1.0, manteniendo sólo una versión Strict.
    - ✓ Permite utilizar en un documento sólo los módulos necesarios para una aplicación concreta.
  - XHTML 2.
    - ✓ En fase de borrador, se ha abandonado el desarrollo favoreciendo HTML 5.

# Introducción
## Tipos de documentos XHTML (III)

- Declaraciones de tipo de documento:
  - XHTML 1.0 Strict.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

  - XHTML 1.0 Transitional.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

  - XHTML 1.0 Frameset.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

  - XHTML 1.1.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

# Diferencias con HTML

- Al ser una aplicación XML, algunas prácticas de HTML basadas en SGML deben cambiar.
  - Las etiquetas se pueden anidar, pero no pueden estar solapadas.
  - XML es sensible a minúsculas, por lo que los nombres de los elementos y atributos deben escribirse en minúsculas.
  - Todos los elementos no vacíos requieren elementos de cierre.
  - Los valores de los atributos deben encerrarse entre comillas.
  - No soporta minimización de atributos.
    - ✓ En HTML algunos atributos tenían un valor booleano según estuvieran o no presentes (`<dl compact>`).
    - ✓ XHTML requiere que todos los atributos tengan un valor (`<dl compact="compact">`).
  - Los elementos vacíos deben tener una etiqueta de cierre o bien acabar con la secuencia `/>`.
    - ✓ Por ejemplo, el elemento `br` (elemento vacío) puede aparecer como `<br />` o como `<br></br>`
  - Manejo de los espacios en blanco.
    - ✓ Las aplicaciones de usuario eliminarán los espacios en blanco al comienzo o final de los atributos.
      - Por ejemplo, el valor de un atributo "  menú" se transformará en "menu".
    - ✓ También sustituirán las secuencias de más de un espacio en blanco por un único espacio en blanco entre palabras.

# Diferencias con HTML (II)

- Los elementos `script` y `style` son de contenido `#PCDATA`.
  - ✓ Los caracteres `<` y `&` serán tratados como inicio de una etiqueta y entidad.
  - ✓ El uso de secciones CDATA evitará la necesidad de utilizar las entidades `<` y `&`.
    - En XML las secciones CDATA hacen que la aplicación no procese el texto que incluye como XML.

```
<![CDATA[
    … Este código no será tratado como código XML.
    … Puede incluir símbolos < y &
]]>
```

- No se deben anidar los siguientes elementos.
  - ✓ `a` no puede contener otros elementos `a` (esto ya ocurría en HTML 4.0).
  - ✓ `pre` no puede contener los elementos `img`, `object`, `big`, `small`, `sub` o `sup`.
  - ✓ `button` no puede contener los elementos `input`, `select`, `textarea`, `label`, `button`, `form`, `fieldset`, `iframe` o `isindex`.
  - ✓ `label` no puede contener otros elementos `label`.
  - ✓ `form` no puede contener otros elementos `form`.

# Diferencias con HTML (III)

- Elementos con atributos `id` y `name`.
  - ✓ HTML 4.0 define el atributo `name` para algunos elementos (`a`, `applet`, `frame`, `iframe`, `img`, y `maps`).
    - También introduce el elemento `id` para identificar de forma unívoca fragmentos de información.
  - ✓ XML identifica los fragmentos con datos de tipo ID que deben ser únicos.
  - ✓ En XHTML el atributo `id` contiene datos de tipo ID que deben ser únicos.
    - Se deben utilizar atributos el atributo `id` para identificar fragmentos a todos los elementos, incluso a aquellos que antes se identificaban mediante `name`, que está prohibido.

# Elementos XHTML
## Etiquetas

- Etiquetas.
  - HTML (y XHTML) definen 91 etiquetas para sus documentos:
    - ✓ `a`, `abbr`, `acronym`, `address`, `applet`, `area`, `b`, `base`, `basefont`, `bdo`, `big`, `blockquote`, `body`, `br`, `button`, `caption`, `center`, `cite`, `code`, `col`, `colgroup`, `dd`, `del`, `dfn`, `dir`, `div`, `dl`, `dt`, `em`, `fieldset`, `font`, `form`, `frame`, `frameset`, `h1`, `h2`, `h3`, `h4`, `h5`, `h6`, `head`, `hr`, `html`, `i`, `iframe`, `img`, `input`, `ins`, `isindex`, `kbd`, `label`, `legend`, `li`, `link`, `map`, `menu`, `meta`, `noframes`, `noscript`, `object`, `ol`, `optgroup`, `option`, `p`, `param`, `pre`, `q`, `s`, `samp`, `script`, `select`, `small`, `span`, `strike`, `strong`, `style`, `sub`, `sup`, `table`, `tbody`, `td`, `textarea`, `tfoot`, `th`, `thead`, `title`, `tr`, `tt`, `u`, `ul` y `var`
    - ✓ `applet`, `basefont`, `center`, `dir`, `font`, `isindex`, `menu`, `s`, `strike` y `u` se consideran obsoletos en XHTML Strict.
    - ✓ Todas las etiquetas referentes a marcos (`frame`, `frameset`, `iframe`, `noframes`) no están permitidas en XHTML Strict.

# Elementos XHTML
## Atributos

- Además, las etiquetas pueden contener algunos atributos.
  - En algunos casos (por ejemplo en las etiquetas vacías como `img` o `a` es obligatorio el uso de atributos).
  - Algunas etiquetas tienen atributos propios, pero también existen atributos comunes a todas las etiquetas.
- Atributos básicos.

| Atributo | Descripción |
|---|---|
| `id="texto"` | Establece un identificador único a un fragmento |
| `class="texto"` | Establece la clase CSS que se aplica como estilo al elemento a través de una hoja de estilos |
| `style="texto"` | Establece un estilo al elemento de forma directa |
| `title="texto"` | Establece un nombre significativo al elemento. Dependiendo del agente de usuario, este nombre aparecerá al pasar el cursor por encima del elemento o se utilizará como información. |

# Elementos XHTML
## Atributos (II)

- Atributos de internacionalización.

| Atributo | Descripción |
|---|---|
| `lang="código de idioma"` | Especifica el idioma en que aparece el elemento. Puede resultar útil cuando se utiliza un navegador basado en voz. Se trata de códigos normalizados según la norma ISO 639 (se puede encontrar una relación de esos códigos en es.wikipedia.org/wiki/ISO_639-1) |
| `xml:lang="código de idioma"` | Especifica el idioma en el que aparece el documento. Tiene prioridad sobre el atributo `lang` y debe aparecer siempre que aparece el atributo `lang`. |
| `dir="dirección del texto"` | Establece la dirección del texto. Es útil si se utilizan idiomas que se escriben de derecha a izquierda. La dirección puede tomar los valores `rtl` (de derecha a izquierda) o `ltr` (de izquierda a derecha, opción por omisión). |

# Elementos XHTML
## Atributos (III)

- Atributos de eventos (para aquellos elementos que pueden recibir eventos).

| Atributo | Descripción |
|---|---|
| `onclick`, `ondblclick`, `onmousedown`, `onmouseup`, `onmouseover`, `onmousemove`, `onmouseout`, `onkeypress`, `onkeydown`, `onkeyup` | Permiten controlar los eventos que se producen sobre los elementos de la página cuando se utiliza JavaScript |

# Elementos XHTML
## Atributos (III)

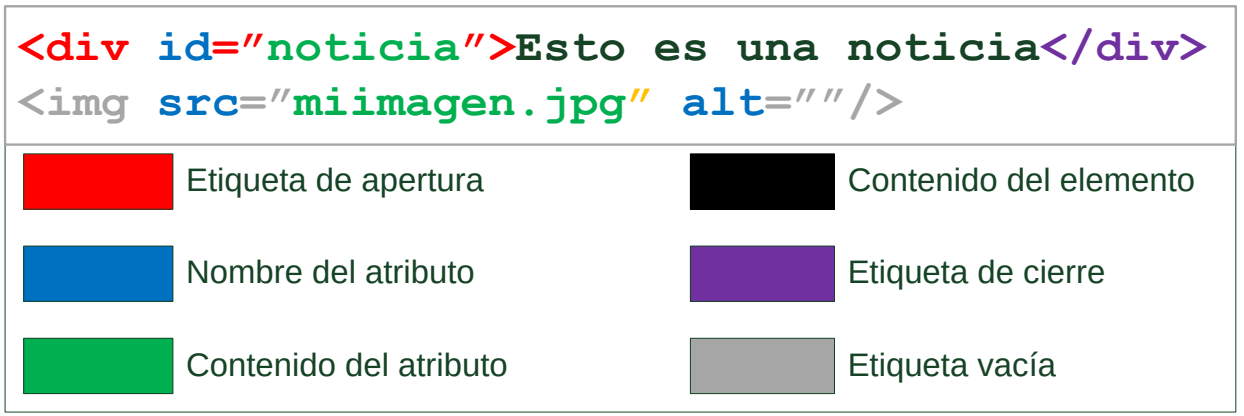
- Atributos de control del foco (para aquellos elementos que pueden recibir el foco).

| Atributo | Descripción |
|---|---|
| `accesskey="letra"` | Permite establecer una tecla de acceso rápido para acceder al elemento. Se utiliza sobre todo en enlaces, botones o campos de formularios. Se utiliza para mejorar la accesibilidad. La forma de acceder a ellos varía según el navegador (alt+letra en Internet Explorer, alt+shift+letra en Firefox o shift+esc+letra en Opera) |
| `tabindex="número"` | Permite establecer el orden de tabulación entre los elementos que permiten recibir el foco. Se utiliza para mejorar la accesibilidad. |
| `onfocus, onblur` | Permite controlar los eventos que se producen cuando el elemento entra o pierde el foco. |

# Elementos XHTML
## Elementos

- Un elemento XHTML, como en cualquier documento XML, se compone de:
  - Etiqueta de inicio.
    - ✓ La etiqueta de inicio puede tener atributos.
  - Contenido del elemento.
  - Etiqueta de cierre.
    - ✓ En el caso de elementos vacíos la etiqueta de cierre puede no existir.
      - En ese caso no tendrá contenido y la etiqueta de inicio acabará con la secuencia `/>`.

```
<div id="noticia">Esto es una noticia</div>
<img src="miimagen.jpg" alt=""/>
```



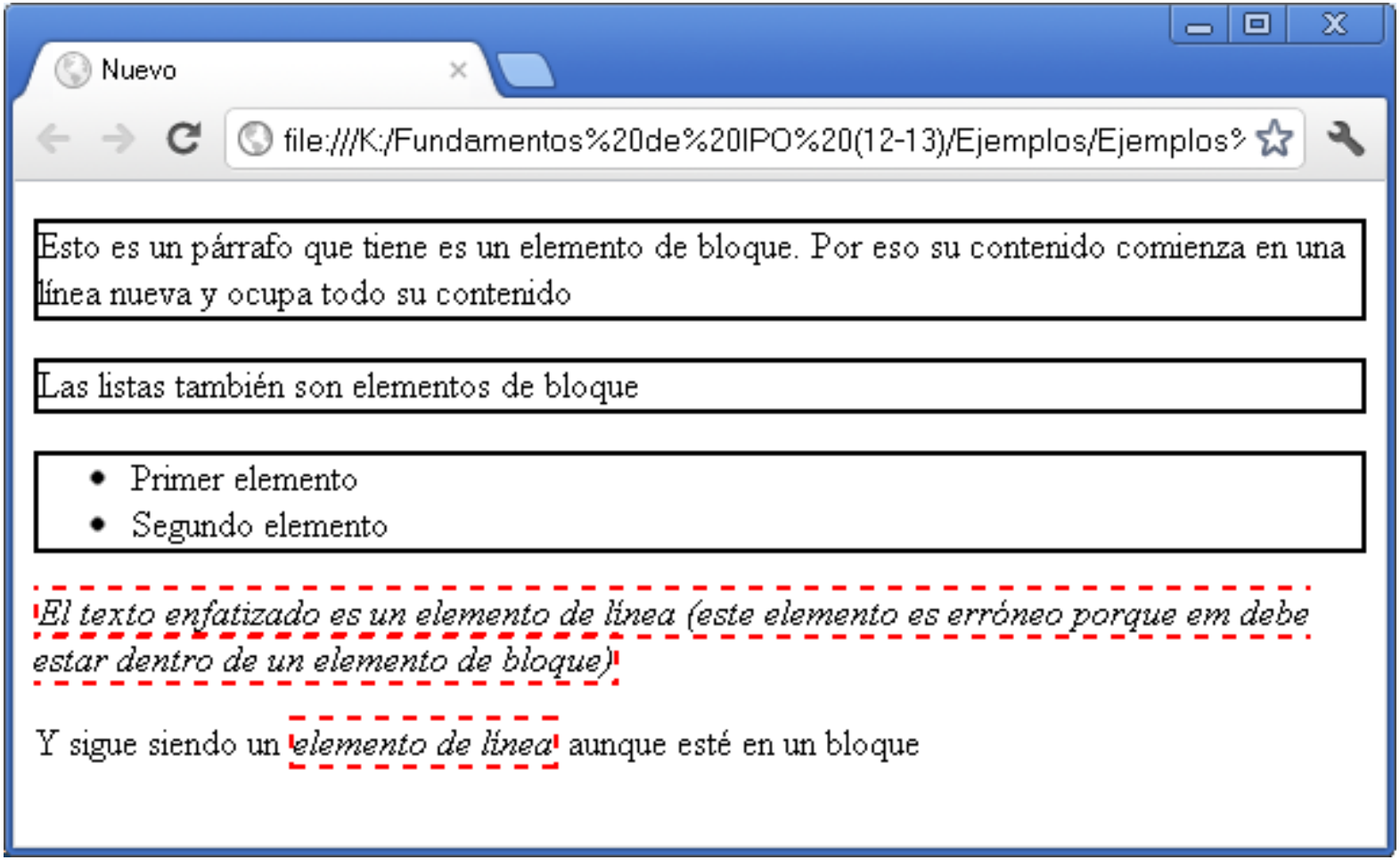
# Elementos XHTML
## Elementos de bloque

- Los elementos de un documento XHTML se pueden dividir en dos categorías:
  - Elementos de bloque.
  - Elementos en línea.
- Los elementos de bloque proporcionan la estructura principal del documento.
  - Podrían equivaler a los párrafos o secciones de un documento.
  - Siempre empiezan en una línea nueva y ocupan todo el espacio disponible.
- Pueden incluir contenido y otros elementos en línea o de bloque.
- Se consideran elementos de bloque:
  - `address`, `blockquote`, `center`, `dd`, `dir`, `div`, `dl`, `dt`, `fieldset`, `form`, `frameset`, `h1`, `h2`, `h3`, `h4`, `h5`, `h6`, `hr`, `isindex`, `li`, `menu`, `noframes`, `noscript`, `ol`, `p`, `pre`, `table`, `tbody`, `td`, `tfoot`, `th`, `thead`, `tr`, `ul`.

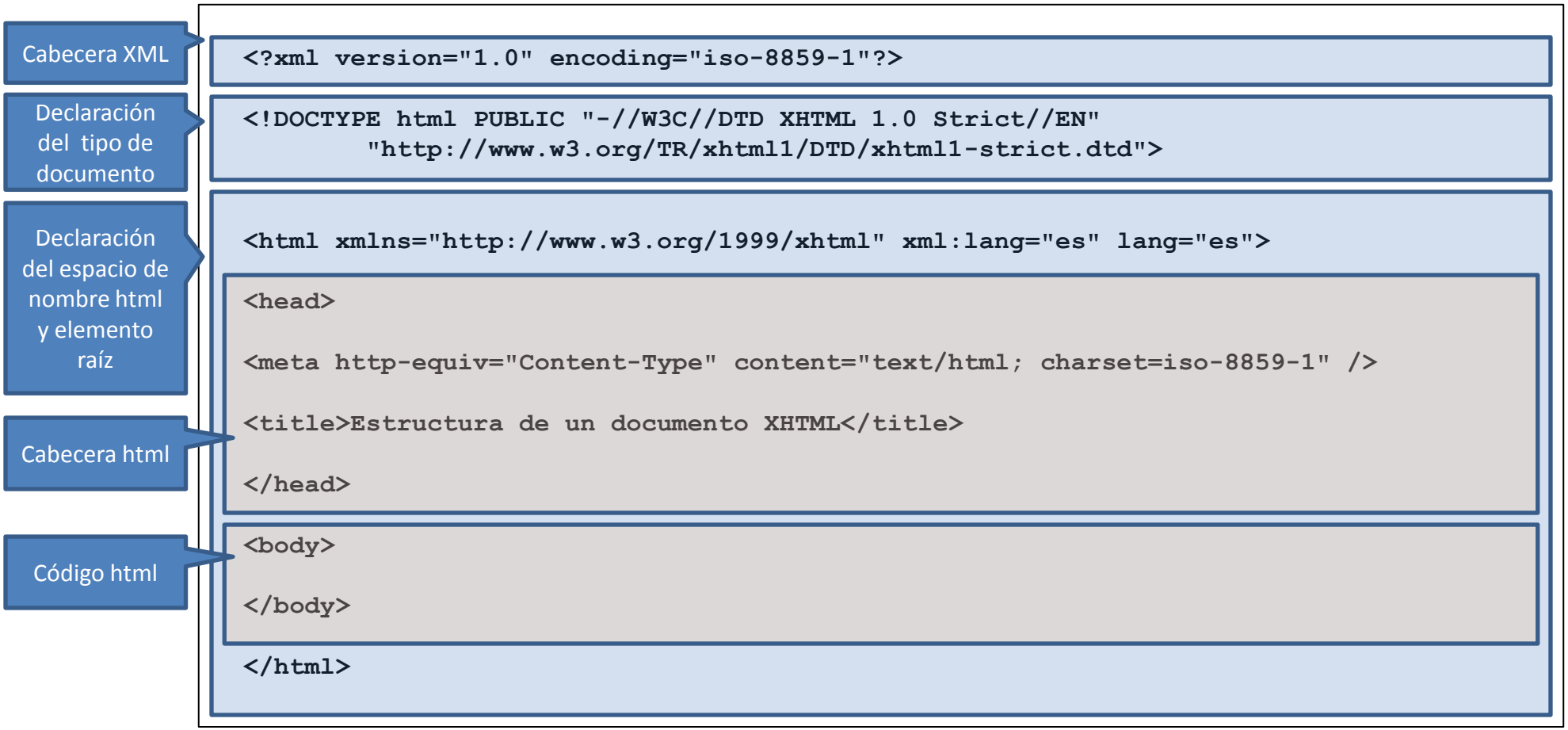
# Elementos XHTML
## Elementos en línea

- Sólo pueden contener otros elementos en línea o contenido.
  - No pueden contener elementos de bloque.
- El elemento no comienza en una línea nueva, y sólo ocupa lo que ocupe su contenido.
- Se consideran elementos en línea:
  - `a`, `abbr`, `acronym`, `b`, `basefont`, `bdo`, `big`, `br`, `cite`, `code`, `dfn`, `em`, `font`, `i`, `img`, `input`, `kbd`, `label`, `q`, `s`, `samp`, `select`, `small`, `span`, `strike`, `strong`, `sub`, `sup`, `textarea`, `tt`, `u`, `var`.
- Los siguientes elementos pueden funcionar como elementos de bloque o en línea:
  - `button`, `del`, `iframe`, `ins`, `map`, `object`, `script`

# Elementos XHTML:
## Elementos de bloque y en línea



# Estructura de un documento XHTML



# Estructura de un documento XHTML (II)

- XHTML es una aplicación XML, por lo que debería llevar una declaración XML (no es obligatorio).
  - Se consideraría obligatoria si se utilizara una codificación de caracteres distinta de `utf-8`.
  - Si se utiliza, deberían ser los primeros caracteres del archivo de texto.
- Declaración del tipo de documento.
  - Algunas de las DTD XHTML Strict, XHTML Trasitional o XHTML Frameset.
- Declaración del elemento raíz y del espacio de nombres xhtml.
  - El elemento raíz es el elemento `html` e incluye tanto la cabecera (elemento `head`) como el cuerpo del documento (elemento `body`).
  - Debe incluir el atributo `xml:lang` y el atributo de html `lang`.
    - ✓ Si se utiliza XHTML 1.1 sólo debe incluir el atributo `xml:lang`.
- Declaración de la cabecera del documento: elemento `head`.
  - Incluye de forma obligatoria el elemento `title`.
  - Es muy común incluir metainformación `http-equiv` que indica el tipo de información que debe suministrar el servidor.
- Elemento `body` con los elementos del código html.

# Cabecera de un documento XHTML

- El elemento raíz.
  - Debe incluir una declaración de espacios de nombre (*namespace*) para el elemento html.
  - Incluye también los atributos `xml:lang` y/o `lang`.
    - ✓ El valor es un código de idioma recogido en la norma ISO 639.
    - ✓ `xml:lang` sería el atributo básico de xml para indicar el idioma principal del documento.
    - ✓ `lang` se utiliza por compatibilidad con versiones antiguas.
    - ✓ No se trata de atributos obligatorios, aunque si convenientes para la interoperabilidad entre aplicaciones.
      - En las pautas de accesibilidad indica su obligatoriedad.
- El elemento `head`.
  - Incluye todos los elementos de la cabecera.
- El elemento `title`.
  - Se trata de un elemento obligatorio de la cabecera.
  - Su valor aparece en la barra de títulos del navegador y en las listas de resultados de los buscadores.

# Cabecera de un documento XHTML
## Metainformación

- La cabecera puede incluir información sobre el documento a partir de la etiqueta `meta`.
  - Puede incluir información acerca de cómo debe proporcionar el servidor la información o información acerca del documento que los motores de búsqueda pueden utilizar para indexar el documento.
- El elemento meta puede incluir los siguientes atributos:
  - `name`, especifica el nombre del atributo que se define.
  - `content`, especifica el valor del atributo que se define.
  - `http-equiv`, puede sustituir a `name`, para especificar información a los servidores.
  - También puede incluir información sobre el idioma de la información.
- HTML no define los metadatos que puede incluir.

# Cabecera de un documento XHTML
## Metainformación

- Un metadato que es interesante poner es el que define el tipo de información que se envía al servidor y que éste puede utilizar en la cabecea http.
  - `<meta http-equiv="content-type" content="text/html ; charset=iso-8859-1" />`
    - ✓ Indica al servidor que el contenido de la página es texto html y que utiliza la codificación de caracteres latin-1.
- Algunos metadatos que se han convertido en estándar por su utilización:
  - `<meta name="author" content="…" />`
  - `<meta name="generator" content="…" />`
    - ✓ Nombre de la aplicación que ha generado el contenido.
  - `<meta name="description" content="…" />`
  - `<meta name="copyright" content="…" />`
  - `<meta name="keywords" content="…" />`

# Cabecera de un documento XHTML
## Enlaces, scripts, estilos

- La etiqueta `link` de la cabecera permite en enlazar el documento de forma automática con otros recursos.
  - Una utilidad muy normal es enlazar con hojas de estilo…

```
<link href="CSS/colimbo.css" rel="stylesheet" type="text/css" />
```

    - `href`, indica el URL del enlace.
    - `rel`, indica la relación del enlace con el documento actual.
    - `type`, indica el tipo de contenido del recurso.
  - Para enlazar con fuentes de noticias RSS…

```
<link rel="alternate" type="application/rss+xml"
  href="http://www.colimbo.net/rss/rss_colimbo.xml" title="Noticias
  colimbo.net" />
```

    - `rel="alternate"`, indica que se trata de contenido alternativo a la página actual.
    - `type` indica que el contenido está en formato de rss.
    - `title`, indica el nombre del recurso.
  - Para indicar en documentos tipo libro, dónde se encuentra el índice, el capítulo anterior y el posterior…

```
<link rel="Index" href="../indice.html" />
<link rel="Next" href="Capitulo3.html" />
<link rel="Prev" href="Capitulo1.html" />
<link rel="Contents" href="contenidos.html" />
```

# Cabecera de un documento XHTML
## Enlaces, scripts, estilos (II)

- Scripts.
  - La cabecera puede incluir información sobre los fragmentos de código (normalmente en Javascript) del documento.
    - ✓ El elemento `script` puede utilizarse también en el cuerpo del documento.
  - Admite dos modos de funcionamiento.
    - ✓ Incluir directamente el código del script.

```
<script type="text/javascript">
<![CDATA[
    …Código javascript
]]>
</script>
```

    - ✓ Enlazar con un archivo que contenga el código de los script.

```
<script type="text/javascript" src="url dónde se almacena el
  código javascript" />
```

- Estilos.
  - El elemento `style` permite indicar los estilos CSS en línea que utilizará la página.

# Cabecera de un documento XHTML
## Ejemplo

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>P&aacute;gina personal de Luis Rodr&iacute;guez Baena</title>
<!-- Indica un url para incluir en los agregadores de noticicias RSS -->
<link rel="alternate" type="application/rss+xml"
       ref="http://www.colimbo.net/rss/rss_colimbo.xml"
       title="Noticias colimbo.net" />
<!-- Estilos generales -->
<link href="CSS/colimbo.css" rel="stylesheet" type="text/css" />
<!-- Estilos incrustados -->
<style>
   * {font-family: helvetica,arial,sans-serif}
</style>
```

# Cabecera de un documento XHTML
## Ejemplo (II)

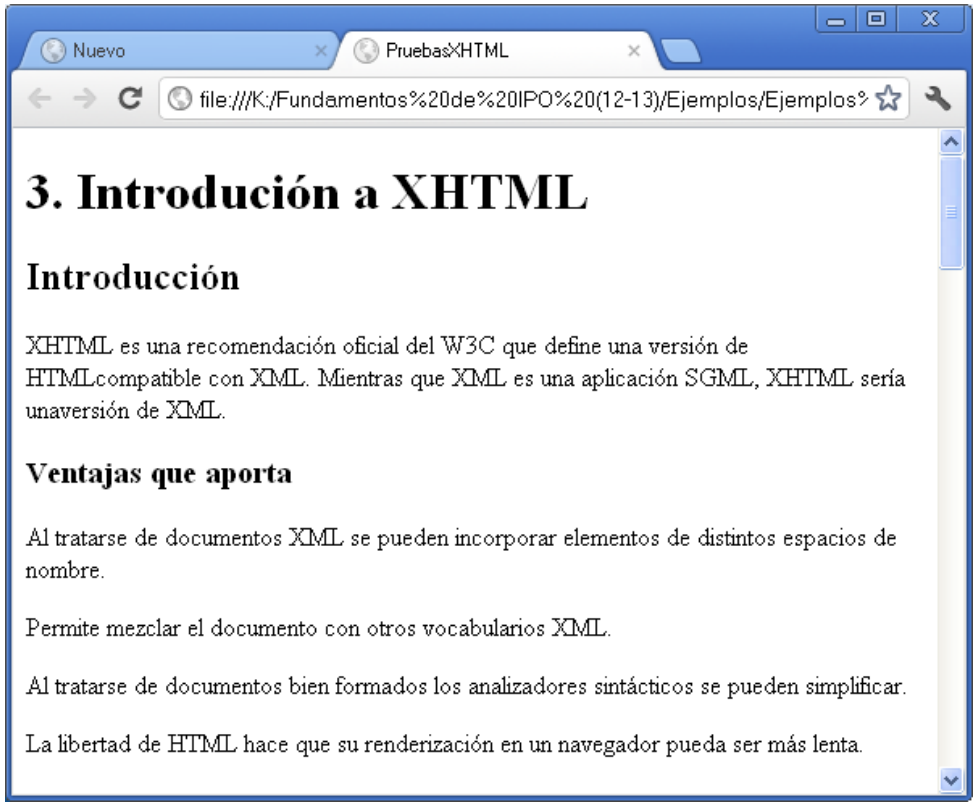
```
<!-- Funcion en JavaScript para la búsqueda) -->
<script type="text/javascript">
<![CDATA[
   function MiGoogle( Topicos )
   {
      // Verificamos que hayan ingresado alguna palabra..
        if(!Topicos)
        { alert( "Por favor introduzca los términos de la búsqeuda" ); return; }
          var MiSitio = "www.colimbo.net";
          window.open
        (
          "http://www.google.com/search?q=site:" + MiSitio + "+" + Topicos,
                "_self"
        );

   }
]]>
</script>
<!-- Fin del Script-->
</head>
```

# Elementos de texto
## Estructura de la página

- La función inicial de html es especificar la estructura y la semántica del texto.
  - Es necesario identificar los distintos elementos del contenido y su relación con el resto de elementos.
- En su forma más básica, el texto estará compuesto de:
  - Párrafos (elemento `p`).
    - ✓ Un elemento de bloque.
    - ✓ En la mayoría de los agentes, se provocará un salto de línea y dejará un espacio entre dos párrafos.
  - Encabezados (elementos `h1..h6`).
    - ✓ También son elementos de bloque.
    - ✓ Establece los títulos de las distintas secciones de la página.
    - ✓ Los navegadores los suelen mostrar con distintos tamaños según su importancia jerárquica.
      - No se deben utilizar para modificar el cuerpo de la letra, sino para indicar la relación jerárquica entre las secciones.
      - Aunque los *parsers* xhtml no lo requieren, las normas de accesibilidad especifican que deben tener un orden lógico.
        - La jerarquía de las cabeceras debe indicar la jerarquía de llas secciones que marcan.

# Elementos de texto
## Estructura de la página (II)

```
<h1>3. Introdución a XHTML</h1>
<h2>Introducción</h2>
<p>XHTML es una recomendación oficial del W3C que
define una versión de HTMLcompatible con XML.
Mientras que XML es una aplicación SGML, XHTML sería
unaversión de XML.</p>
<h3>Ventajas que aporta</h3>
<p>Al tratarse de documentos XML se pueden incorporar
elementos de distintos espacios de nombre.</p>
<p>Permite mezclar el documento con otros
vocabularios XML.</p>
<p>Al tratarse de documentos bien formados los
analizadores sintácticos se pueden simplificar.</p>
<p>La libertad de HTML hace que su renderización en
un navegador pueda ser más lenta.</p>
```

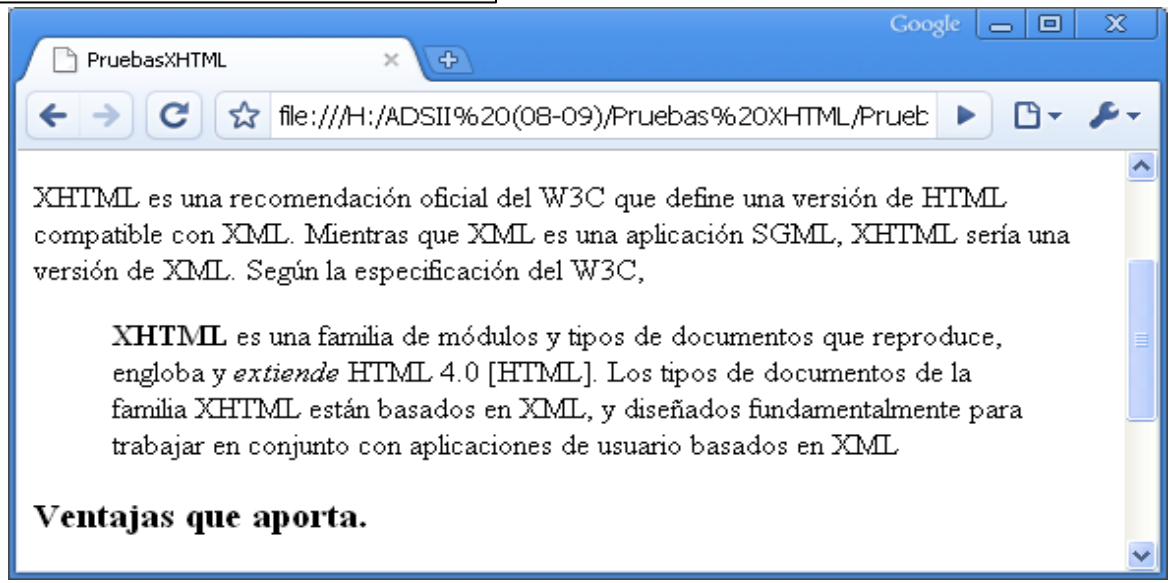


# Elementos de texto
## Enfatizar el contenido

- Énfasis en línea.
  - Los dos métodos más comunes para destacar porciones de un texto en los documentos impresos es utilizar las negritas y las cursivas.
    - ✓ HTML define cuatro elementos (`b`, `strong`, `i` y `em`) para ello.
    - ✓ Es preferible utilizar los elementos `em` para enfatizar el texto y `strong` para destacarlo.
      - Mientras que `b` e `i` indican el modo de la presentación (lo que indican es el estilo de la fuente, como el subrayado o el tachado), `em` y `strong` marcan el texto desde el punto de vista estructural y marcan texto para enfatizar o para remarcar.
      - En la mayoría de los agentes de usuario (navegadores) eso se traduce en cursiva y negrita respectivamente.
        - En otros agentes (por ejemplo, navegadores de voz) permitirían un aspecto distinto.
- Énfasis en bloque.
  - El elemento `blockquote` permite destacar un bloque de texto.
    - ✓ El equivalente impreso sería un texto tratado como cita o encerrado entre comillas.
    - ✓ Los navegadores más habituales sangrarán el texto.
    - ✓ Además de los atributos normales puede incluir el atributo `cite` que incluiría un URL del lugar dónde se ha sacado la cita.

# Elementos de texto
## Enfatizar el contenido (II)

```
<p>XHTML es una recomendación oficial del W3C que define una versión de
HTMLcompatible con XML. Mientras que XML es una aplicación SGML, XHTML
sería unaversión de XML. Según la especificación del W3C, </p>
<blockquote
cite="http://www.sidar.org/recur/desdi/traduc/es/xhtml/xhtml11.htm#diff
s">
<p><strong>XHTML</strong> es una familia de módulos y tipos de
documentos que  reproduce, engloba y <em>extiende</em> HTML 4.0 [HTML].
Los tipos de  documentos de la familia XHTML están basados en XML, y
diseñados  fundamentalmente para trabajar en conjunto con aplicaciones
de usuario  basados en XML</p></blockquote>
<h3>Ventajas que aporta.</h3>
```



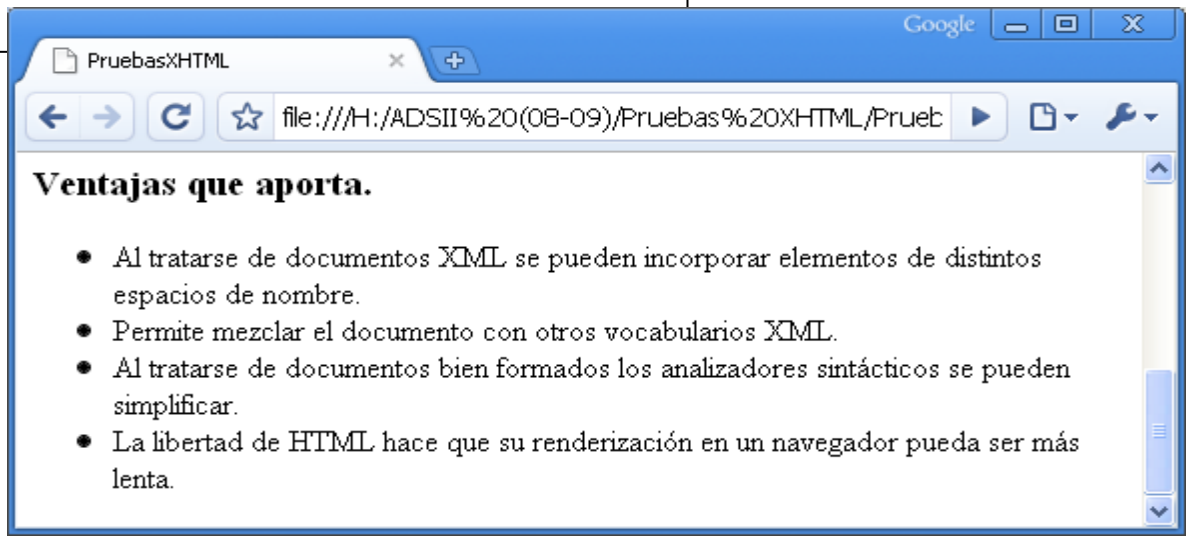
# Elementos de texto
## Listas

- La estructura de un documento puede requerir la enumeración de elementos en forma de listas.
- XHTML presenta tres tipos de listas:
  - Listas no ordenadas.
    - ✓ Representa una enumeración de elementos sin ninguna relación jerárquica entre ellos.
  - Listas ordenadas.
    - ✓ Representa una secuencia de elementos establecida según un orden determinado.
  - Listas de definición.
    - ✓ Representan una relación de términos (por ejemplo de un glosario) y la definición de cada uno de ellos.
- Los tres tipos de listas presentan atributos para su presentación
  - Tipos de viñetas en listas no ordenadas o numeración inicial o estilo de numeración (números arábigos, romanos, letras, etc.).
  - Es recomendable dejar esa tarea a los estilos.

# Elementos de texto
## Listas no ordenadas

- Los elementos se incluyen dentro de la etiqueta `ul`.
  - Cada ítem de la lista representa un elemento `li`.

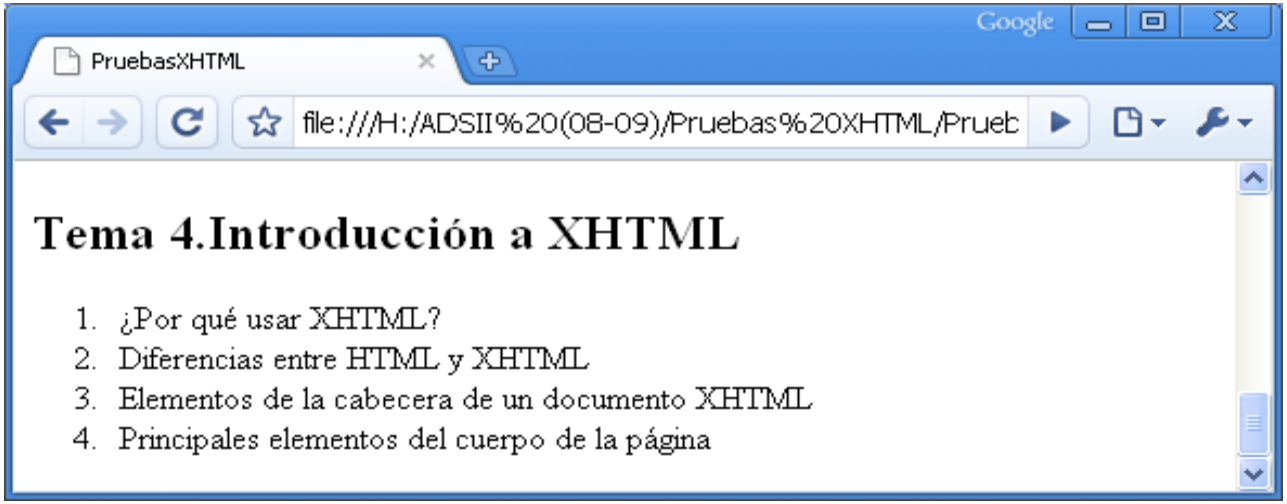
```
<h3>Ventajas que aporta.</h3>
<ul>
  <li>Al tratarse de documentos XML se pueden incorporar elementos de
      distintos  espacios de nombre.</li>
  <li>Permite mezclar el documento con otros vocabularios XML.</li>
  <li>Al tratarse de documentos bien formados los analizadores sintácticos
      se pueden simplificar.</li>
   <li>La libertad de HTML hace que su renderización en un navegador pueda
       ser más lenta.</li>
</ul>
```



# Elementos de texto
## Listas ordenadas

- Los elementos se incluyen dentro de la etiqueta `ol`.
  - Cada ítem de la lista representa un elemento `li`.

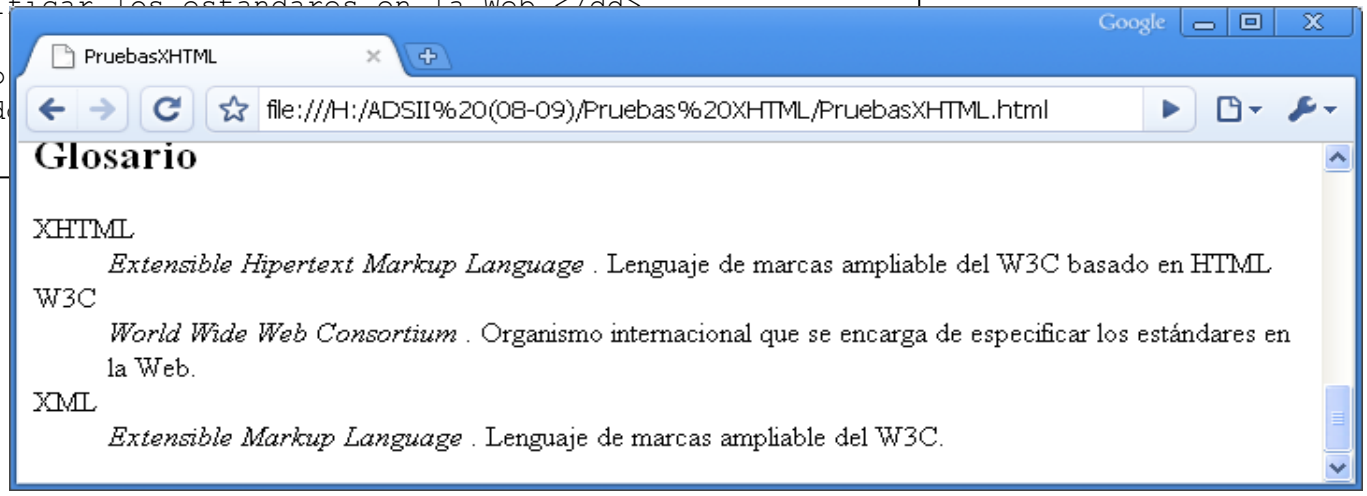
```
<h2>Tema 4.Introducción a XHTML</h2>
<ol>
   <li>¿Por qué usar XHTML?</li>
   <li>Diferencias entre HTML y XHTML</li>
   <li>Elementos de la cabecera de un documento XHTML</li>
   <li>Principales elementos del cuerpo de la página</li>
</ol>
```



# Elementos de texto
## Listas de definición

- Los elementos se incluyen dentro de la etiqueta `dl`.
  - Cada término a definir se incluye en el elemento `dt`.
  - La definición se incluye en el elemento `dd`.
  - Puede haber varias definiciones por cada término.

```
<h2>Glosario</h2>
<dl>
   <dt>XHTML</dt>
     <dd><em>Extensible Hipertext Markup Language</em> .
          Lenguaje de marcas ampliable del W3C basado en HTML</dd>
   <dt>W3C</dt>
     <dd><em>World Wide Web Consortium</em> . Organismo internacional que
          se encarga de especificar los estándares en la Web.</dd>
   <dt>XML</dt>
     <dd><em>Extensible Markup
         ampliable del W3C.</d
</dl>
```

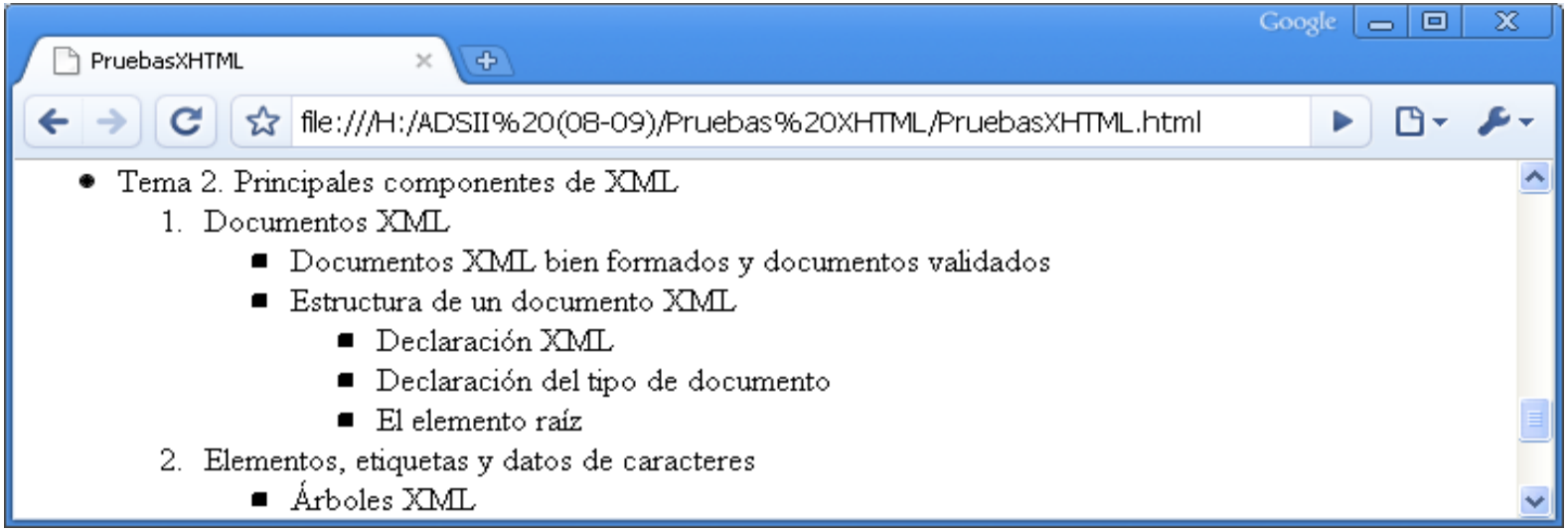


# Elementos de texto
## Anidamiento de listas

- Los elementos *listitems* (`li`) pueden contener otras listas  lo que daría lugar a listas anidadas.

```
<ul>
   <li>Tema 1. Introducción a XML</li>
   <li>Tema 2. Principales componentes de XML
      <ol>
        <li>Documentos XML
           <ul>
             <li>Documentos XML bien formados y documentos validados</li>
             <li>Estructura de un documento XML
                <ul>
                   <li>Declaración XML </li>
                   <li>Declaración del tipo de documento </li>
                  <li>El elemento raíz</li>
               </ul>
             </li>
           </ul>
         </li>
         <li>Elementos, etiquetas y datos de caracteres
           <ul>
              <li>Árboles XML</li>
          </ul>
       </li>
     </ol>
   </li>
</ul>
```

# Elementos de texto
## Anidamiento de listas (II)

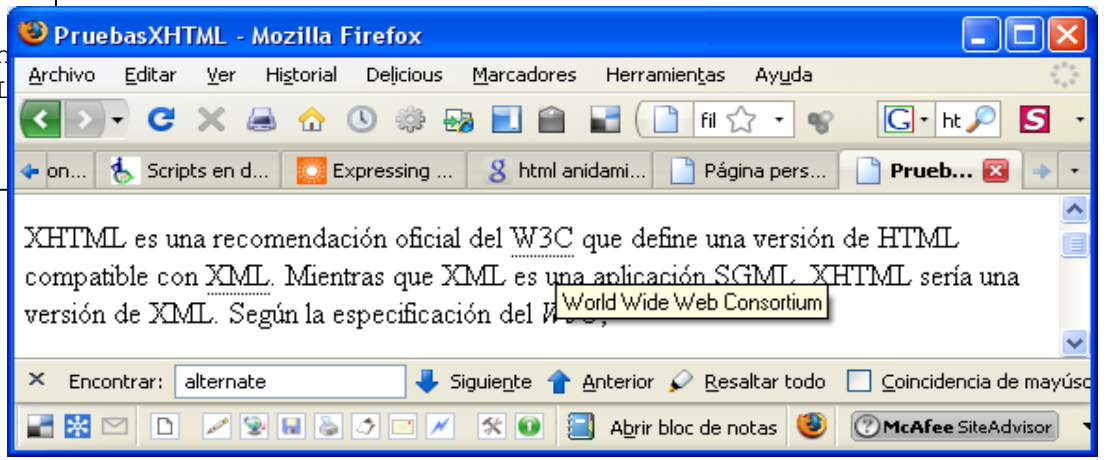


# Elementos de texto
## Otros elementos

### Abreviaturas y acrónimos.

- El elemento `abbr` se utiliza para marcar una abreviatura.
- El atributo `title` incluiría el texto completo de la abreviatura.
- El elemento `acronym` se utiliza para marcar un acrónimo.
  - ✓ El atributo `title` se utilizaría para indicar la explicación del acrónimo.
- Normalmente los navegadores muestran el elemento subrayado.
  - ✓ Al pasar el cursor sobre el elemento aparecería el texto de la abreviatura.

```
<p>XHTML es una recomendación oficial del <abbr
title="World Wide Web Consortium">W3C</abbr> que
define una versión de HTML compatible con
<acronym title="Extended Markup
Language">XML</acronym>. Mientras que XML es un
aplicación SGML, XHTML sería una versión de XML
Según la especificación del <cite>W3C</cite>,
</p>
```



# Elementos de texto
## Otros elementos (II)

- Definiciones.
  - El elemento `dfn` marca un elemento para su definición en el mismo párrafo.
    - ✓ Se utiliza cuando se quiere explicar algún término complicado (por ejemplo algún término médico o técnico).
    - ✓ Algunos navegadores muestran el elemento como cursiva.
- Citas.
  - El elemento `cite` marca un fragmento de texto como una referencia a otras fuentes.
  - El elemento `q` (*quote* , comillas) se utiliza para incluir citas textuales en línea.
    - ✓ Los navegadores suelen encerrar la cita entre comillas.
    - ✓ El atributo `cite` se puede utilizar para incluir la referencia de la cita. A diferencia de `blockquote`, se trata de un elemento en línea.
    - ✓ Los agentes de usuario suelen mostrarlo en cursiva.
- Direcciones.
  - El elemento `address` indica que el texto contenido es una dirección.
  - A diferencia de los elementos anteriores, se trata de un elemento de bloque.
  - Los navegadores suelen poner el contenido en cursiva.

# Elementos de texto
## Otros elementos (III)

- Subíndices y superíndices.
  - Se marcan, respectivamente, con los elementos `sub` y `sup`.
- Código.
  - Un fragmento de código de programación se puede marcar con el elemento `code`.
- Entrada de usuario.
  - Si en algún caso se quiere marcar un texto como una entrada que el usuario debe introducir se puede utilizar el elemento `kbd`.
- Variables.
  - Las variables de un programa se marcan con el elemento `var`.
- Salida de programas, scripts, etc.
  - El elemento `samp` se utiliza para marcar la salida por pantalla de un programa, script, etc.

# Elementos de texto
## Marcado genérico

- xhtml permite marcar fragmentos de texto de forma genérica.
  - Esto permite marcar contenido que no pertenezca a ninguna de las categorías anteriores.
    - ✓ Por ejemplo, permitiría marcar un fragmento de contenido como una dirección de e-mail, un teléfono o una profesión.
  - Los elementos `div` y `span` permiten marcar fragmentos de texto.
    - ✓ `div` es un elemento de bloque.
    - ✓ `span` es un elemento de texto.
  - Aunque permitiría marcar un elemento con un significado no previsto (por ejemplo `<span class="e-mail">yo@miservidor.com </span>`, se suele utilizar para indicar un estilo concreto al fragmento de texto.

# Elementos de texto
## Saltos de línea, espacios en blanco, tabulaciones

- Los saltos de línea, tabulaciones o los espacios en blanco del código de un documento se comprimen a un único espacio en blanco en su visualización.
  - El elemento `br` inserta una nueva línea en el lugar de su aparición.
    - ✓ Se trata de un elemento vacío, por lo que se debe codificar como `<br />`.
  - Para insertar espacios en blanco se debe utilizar la entidad de carácter ` `.
  - El elemento de bloque `pre` (texto preformateado) respeta todos los espacios en blanco, tabulaciones y saltos de línea de su contenido.
    - ✓ Los navegadores, además de respetar el formato utilizan un tipo de letra monospaced (tipo courier).

# Elementos de texto
## Codificación de caracteres

- xhtml (y html) no puede utilizar todos los caracteres. Además otros caracteres no se mostrarán correctamente en todas las ocasiones.
  - Caracteres especiales como los símbolos de mayor y menor, el ampersand, el espacio en blanco, comillas, caracteres regionales, etc. no se mostrarán bien en todas las ocasiones.
    - ✓ Es necesario sustituirlas por entidades de carácter.
    - ✓ En en.wikipedia.org/wiki/List_of_XML_and_HTML_character_entity_references aparece una lista completa de las entidades.

| Entidad | | Entidad | Carácter | Entidad | Carácter | Entidad | Carácter |
|---|---|---|---|---|---|---|---|
| &quot; | " | &ntilde; | ñ | &iacute; | í | &uuml; | ü |
| &amp; | & | &Ntilde; | Ñ | &Iacute; | Í | &Uuml; | Ú |
| &apos; | ' | &aacute; | á | &oacute; | ó | &ordm; | º |
| &lt; | < | &Aacute; | Á | &Oacute; | Ó | &ordf; | ª |
| &gt; | > | &eacute; | é | &uacute; | ú | &ccedil; | ç |
|   | *blanco* | &Eacute; | É | &Uacute; | Ú | &Ccedil; | Ç |

# Enlaces

- El elemento `a` permite incluir un enlace a un recurso o indicar el destino de un enlace dentro de un recurso.
- Para incluir el destino (ancla) de un enlace dentro de un documento se puede utilizar el atributo `name` (en XHTML 1.0).

```
<h2><a name="Capitulo1">Capítulo 1</a></h2>
<p>bla, bla, bla, …. </p>
```

  - Se puede conseguir el mismo efecto utilizando el atributo `id` del elemento que se desea utilizar como ancla (recomendable y obligatorio en XHTML 1.1).

```
<h2 id="Capitulo1">Capítulo 1</h2>
<p>bla, bla, bla, …. </p>
```

  - El URL de ese destino se conseguiría utilizando el carácter #.
    - ✓ Suponiendo que se encuentre en el documento `MiDocumento.html` alojado en la carpeta `docs` del servidor `MiServidor.com`…

      `http://www.miservidor.com/docs/MiDocumento.html#Capitulo1`
  - El texto marcado como destino no aparece señalado de ninguna forma en el navegador.

# Enlaces (II)

- La forma más habitual de utilizar el atributo `a` es como enlace a otro documento.
  - El contenido del elemento aparecerá marcado como enlace.
    - ✓ Habitualmente aparecerá en azul y subrayado.
  - Para indicar el destino del enlace se utiliza el atributo `href`.
    - ✓ El contenido del atributo será el URI del recurso.
      - Normalmente se tratará de un URL completo que consta de:
        - Protocolo (por ejemplo, `http://`)
        - Servidor (por ejemplo, `www.miservidor.com`)
        - Ruta del recurso (por ejemplo `/documentos/`)
        - Nombre del recurso (por ejemplo, `midocumento.html`)
        - Parámetros de la llamada (por ejemplo `?id=1`)
        - Destino dentro del recurso (por ejemplo `#Capitulo1`).
      - Dependiendo de la ubicación del enlace, se puede omitir algunas de estas partes.
        - Se pueden utilizar URI absolutos o relativos.
          - Las absolutas indican el camino a partir del servidor (el URI de la diapositiva anterior sería absoluto).
          - Las relativas indican el camino a partir de la posición actual de la página. Por ejemplo, si la página de la diapositiva anterior estuviera situada en la carpeta raíz de `MiServidor.com`, podría ser `docs/MiDocumento.html#Capitulo1`

# Enlaces
## Otros atributos

- `hreflang`.
  - Indica el idioma del recurso de destino.
- `type`.
  - Indica el tipo de contenido del recurso.
    - ✓ Algunos ejemplos:
      - `text/html`
      - `image/png`
      - `application/pdf`
      - …
    - ✓ Se puede encontrar una lista de tipos de contenido en www.iana.org/assignments/media-types/.
- `charset`.
  - Indica la codificación de caracteres del recurso enlazado.

# Enlaces
## Otros atributos (II)

- `rel` y `rev`.
  - Describen la relación del elemento enlazado con el actual (`rel`) y la del actual con el enlazado (`rev`).
  - Puede contener los valores:
    - ✓ `alternate`, indica que se trata de una versión alternativa al documento actual (por ejemplo, una versión para imprimir).
    - ✓ `stylesheet` , indica que se trata de una hoja de estilos.
    - ✓ `start`, indica que se trata del primer documento de una colección de documentos relacionados.
    - ✓ `next`, indica que se trata del siguiente documento de una colección de documentos relacionados.
    - ✓ `prev`, indica que se trata del documento anterior de una colección de documentos relacionados.
    - ✓ `contents`, indica que el recurso enlazado contiene la tabla de contenidos de una colección de documentos relacionados.
  - En www.w3.org/TR/1999/REC-html401-19991224/types.html#type-links se puede encontrar una lista de las relaciones posibles.

# Imágenes

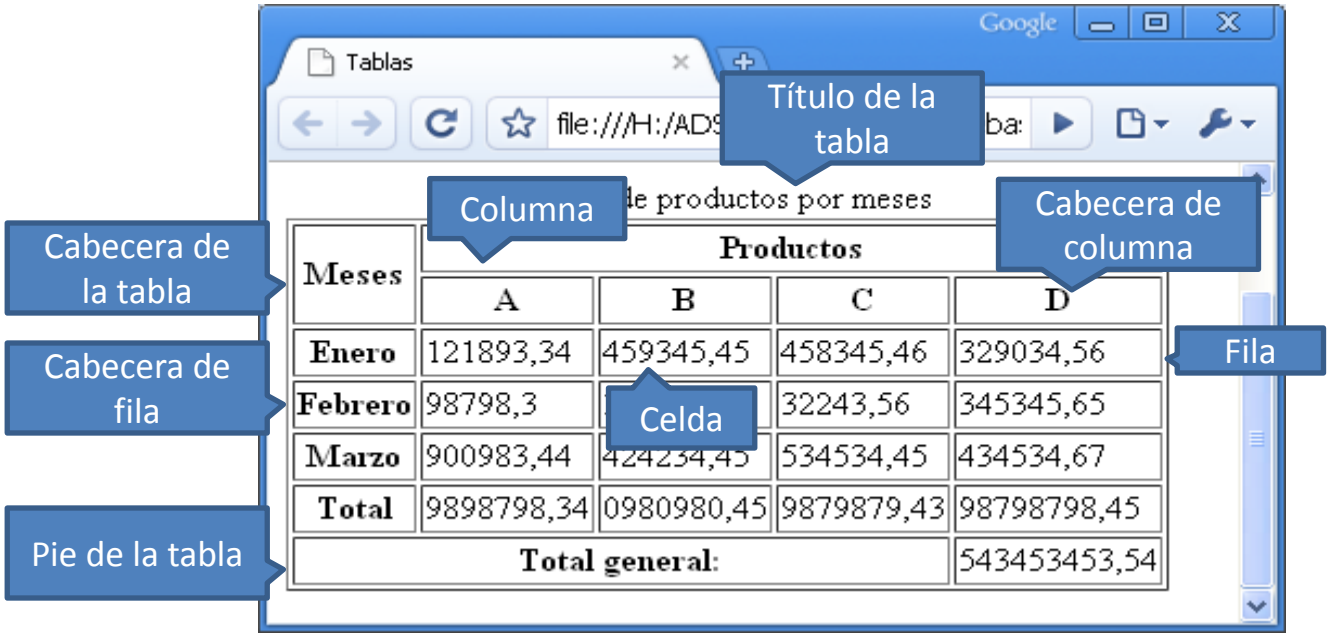
- El elemento `img` se utiliza para insertar imágenes en un documento.
  - Se trata de un etiqueta vacía.
    - ✓ La información de la etiqueta se establece por medio de atributos.
    - ✓ Debería acabar con la secuencia `/>` o incluir el elemento de cierr `</img>`.
- En xhtml son obligatorios los atributos `src` y `alt`.
  - `src` establece un URL a la imagen que se va a mostrar.
    - ✓ En principio el recurso al que apunta el URL podrá ser cualquier archivo que contenga información gráfica.
      - Normalmente los navegadores sólo admiten imágenes `.gif`, `.jpg` o `.png`.
  - `alt` establece un texto alternativo a la imagen.
    - ✓ Todas las imágenes deberían ir acompañadas de un texto alternativo que explique el contenido de la imagen para aquellas personas que no puedan ver la imagen o para aquellos navegadores que no sean capaces de mostrar imágenes.
      - Este es uno de los requisitos prioritarios de las pautas de accesibilidad.
      - El texto alternativo debería poder explicar la imagen, incluso fuera de contexto.

# Imágenes (II)

- Atributo `longdesc`.
  - El atributo `alt`, sólo permite incluir texto de hasta 1024 caracteres.
    - ✓ En imágenes complejas (por ejemplo, un gráfico de distribución de frecuencias en un aplicación estadística), `longdesc` permite apuntar a un URL con una descripción completa de la imagen.
- Atributo `title`.
  - Permite dar un nombre explicativo a la imagen.
    - ✓ Normalmente, los navegadores sacarán el texto en forma de tooltip.
    - ✓ Algunos navegadores (IE) sacan el contenido de `alt` como tooltip. Para que esto ocurra en otros como Firefox, es necesario incluir este atributo.
- Atributos `height` y `width`.
  - Permiten establece el alto y ancho (normalmente en pixels) de la imagen.
    - ✓ Para evitar imágenes pixeladas, lo ideal sería establecerlos al ancho y alto real o no incluir ninguno.
      - Si se desea utilizar una imagen más grande o más pequeña, se debería conseguir en el archivo original mediante algún programa de retoque fotográfico.

# Tablas

- Se deben utilizar para mostrar **exclusivamente** información tabular.
  - El uso de tablas como método para crear la estructura de la página está desaconsejado.
- Partes básicas de una tabla



# Tablas
## Elementos básicos

- En XHTML, una tabla contiene tres elementos básicos:
  - Elemento `table`.
    - ✓ Engloba a toda la tabla.
    - ✓ Atributos:
      - `summary`. Permite especificar un resumen explicativo de la tabla.
        - Se utiliza por cuestiones de accesibilidad, para que agentes no visuales tengan información sobre el contenido de la tabla.
  - Elemento `tr`.
    - ✓ Engloba todas las celdas de una fila.
  - Elemento `td`.
    - ✓ Define cada una de las celdas de la tabla.
    - ✓ Atributos:
      - `abbr`. Proporciona una forma abreviada del contenido de la tabla.
        - Se utiliza por motivos de accesibilidad.
      - `scope`. Si se trata de una cabecera de fila o columna, indica el ámbito de las celdas sobre la que es cabecera.
        - Puede tomar los valores `col`, `row`, `rowgroup` o `colgroup`.
        - Se utiliza por motivos de accesibilidad.
      - `headers`. Permite explicitar las celdas que actúan de cabecera de la tabla.
        - El valor será una serie de valores de atributos `id` de las celdas que actúan como cabecera.
        - Se utiliza por motivos de accesibilidad.
      - `colspan` y `rowspan`.

# Tablas
## Elementos básicos (II)

- Elemento `caption`.
  - Permite añadir un título explicativo a la tabla.
  - Sólo puede haber un elemento por tabla.
  - Debe ser el primer elemento de la tabla.
  - Las normas de accesibilidad indican que toda tabla de datos debería tener un elemento `caption`.
    - ✓ Un agente no visual relacionará la tabla con este elemento.
- Elemento `th`.
  - En una tabla de datos se deben marcar todas las celdas que sean encabezados.
  - Indica que se trata de una celda de encabezado.
  - Tiene los mismos atributo que el elemento `td`.
  - En un navegador visual, puede que se señale el elemento de alguna forma especial (por ejemplo en negrita).
  - En un navegador de voz, u otro agente no visual se utiliza para identificar las celdas.

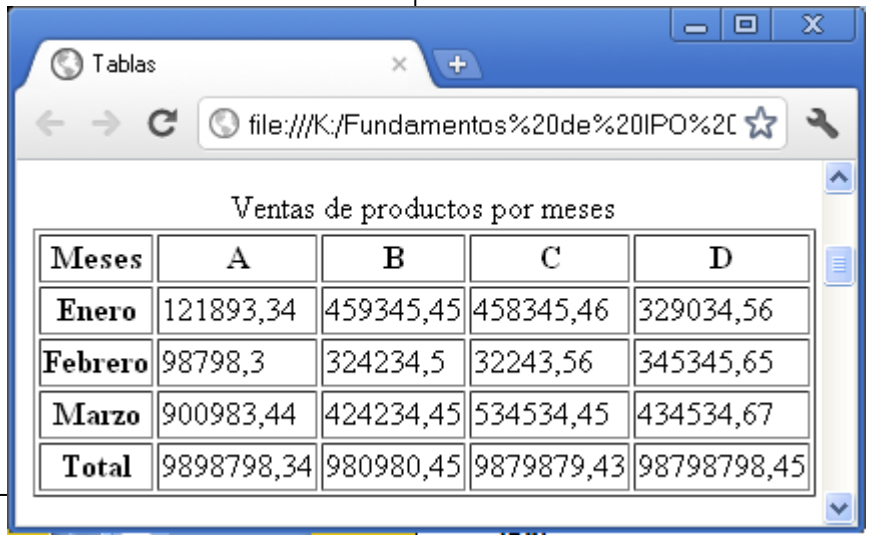
# Tablas
## Elementos básicos (III)

- Atributos `headers` y `scope`.
  - Algunos agentes no visuales pueden indicar la cabecera de un elemento de la tabla.
  - Para tablas simples, basta marcarlas con `th`.
  - Para tablas más complejas se pueden utilizar los atributos `scope` y `headers`.
  - Para ello se puede utilizar cualquiera de los dos atributos.
    - ✓ En una celda de cabecera, `scope` indica el conjunto de celdas sobre la que actuará como cabecera.
      - Para indicar que el ámbito de una celda de cabecera es una fila o una columna se utilizan los valores para `scope` de `row` o `col`.
      - Si se han agrupado filas o columnas (ver más adelante), se pueden utilizar los valores de `scope` `colgroup` o `rowgroup`
    - ✓ Cuando no está claro el ámbito de una celda, se puede utilizar el atributo `headers`.
      - Algunos agentes de usuario no visuales no manejan bien el atributo `scope`, por lo que puede ser conveniente incluir también el atributo `headers`.
      - En este atributo se indicarán los valores de los `id` de las celdas que actúan como cabecera.
        - Las cedas que actúan como cabecera deben llevar el atributo `id` que las identifica.
      - Generalmente, es necesario utilizarlo cuando se colocan los encabezados en posiciones irregulares con respecto a los datos a los que se aplican.

# Tablas
## Elementos básicos (IV)

- Ejemplo de marcado de cabeceras con `scope`.
  - Nota: el atributo `border` esta desaconsejado en XHTML Strict. En los ejemplos se utiliza sólo para ver los límites de la tabla.

```
<table border="1" summary="Esta tabla representa las ventas de los
  productos A, B, C y D por meses. Al final aparece el total
  de ventas de cada producto">
    <caption>Ventas de productos por meses</caption>
    <tr>
      <th>Meses</th>
      <th scope="col">A</th>
      <th scope="col">B</th>
      <th scope="col">C</th>
      <th scope="col">D</th>
    </tr>
    <tr>
      <th scope="row" abbr="Ene">Enero</th>
      <td>121893,34</td>
      <td>459345,45</td>
      <td>458345,46</td>
      <td>329034,56</td>
    </tr>
    ...
<table>
```



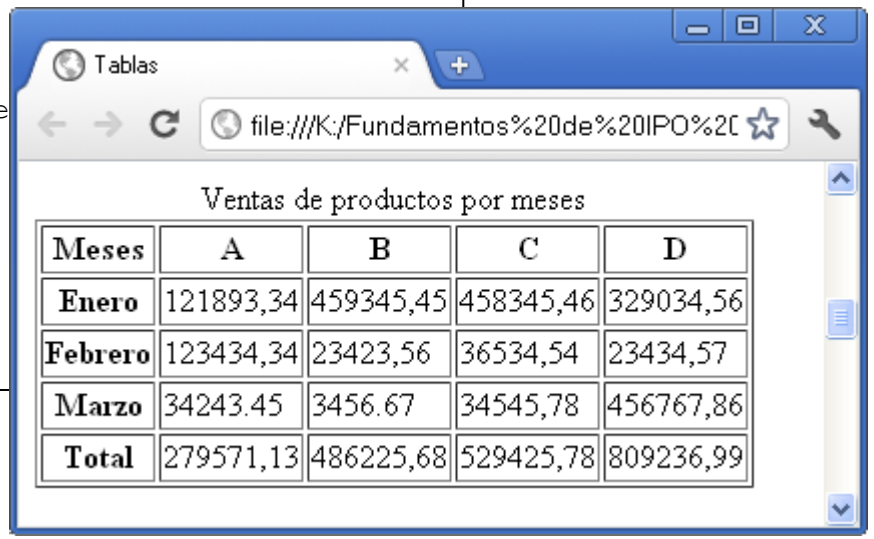

Ventas de productos por meses

| Meses | A | B | C | D |
|---|---|---|---|---|
| Enero | 121893,34 | 459345,45 | 458345,46 | 329034,56 |
| Febrero | 98798,3 | 324234,5 | 32243,56 | 345345,65 |
| Marzo | 900983,44 | 424234,45 | 534534,45 | 434534,67 |
| Total | 9898798,34 | 980980,45 | 9879879,43 | 98798798,45 |

# Tablas
## Elementos básicos (V)

- Ejemplo de marcado de cabeceras con `headers`.

```
<table border="1" summary="Esta tabla representa las ventas de los
  productos A, B, C y D por meses. Al final aparece el total
  de ventas de cada producto">
          <caption>Ventas de productos por meses</caption>
    <tr>
      <th id="meses>Meses</th>
      <th scope="col" id="prodA">A</th>
      <th scope="col" id="prodB">B</th>
      <th scope="col" id="prodC">C</th>
      <th scope="col" id="prodD">D</th>
    </tr>
    <tr>
      <th scope="row" headers="meses" id="ene" abbr="Ene
      <td headers="ene prodA">121893,34</td>
      <td headers="ene prodB">459345,45</td>
      <td headers="ene prodC">458345,46</td>
      <td headers="ene prodD">329034,56</td>
    </tr>
    ...
<table>
```




Ventas de productos por meses

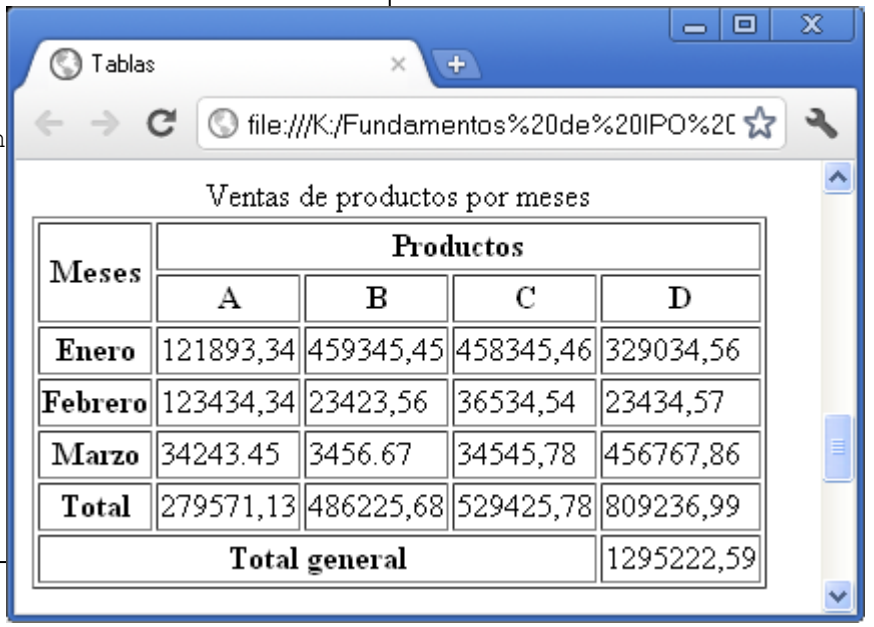
| Meses | A | B | C | D |
|---|---|---|---|---|
| Enero | 121893,34 | 459345,45 | 458345,46 | 329034,56 |
| Febrero | 123434,34 | 23423,56 | 36534,54 | 23434,57 |
| Marzo | 34243.45 | 3456.67 | 34545,78 | 456767,86 |
| Total | 279571,13 | 486225,68 | 529425,78 | 809236,99 |

# Tablas
## Celdas que abarcan varias filas o columnas

- Las celdas pueden abarcar varias filas o columnas.
- Los atributos `colspan` y `rowspan` de los elementos `td` y `th` permiten ampliar la celda a lo largo de varias filas o columnas.
- El valor de estos atributos representaría el número de filas o columnas que abarcan.

# Tablas
## Celdas que abarcan varias filas o columnas (II)

```
<table border="1" summary="Esta tabla representa las ventas de los
  productos A, B, C y D por meses. Al final aparece el total
  de ventas de cada producto">
          <caption>Ventas de productos por meses</caption>
    <tr>
      <th id="meses" rowspan="2">Meses</th>
      <th scope="col" id="productos" colspan="4">Productos</th>
    </tr>
    <tr>
      <th scope="col" id="prodA">A</th>
      <th scope="col" id="prodB">B</th>
      <th scope="col" id="prodC">C</th>
      <th scope="col" id="prodD">D</th>
    </tr>
    <tr>
      <th scope="row" headers="meses" id="ene" abbr="Ene">En
      <td headers="ene productos prodA">121893,34</td>
      <td headers="ene productos prodB">459345,45</td>
      <td headers="ene productos prodC">458345,46</td>
      <td headers="ene productos prodD">329034,56</td>
    </tr>
    ...
    <tr>
      <th colspan="4" id="totalGeneral">Total general</th>
      <td headers="totalGeneral">1295222,59</td>
    </tr>
</table>
```




Ventas de productos por meses

| Meses | Productos | | | |
|---|---|---|---|---|
| | A | B | C | D |
| Enero | 121893,34 | 459345,45 | 458345,46 | 329034,56 |
| Febrero | 123434,34 | 23423,56 | 36534,54 | 23434,57 |
| Marzo | 34243.45 | 3456.67 | 34545,78 | 456767,86 |
| Total | 279571,13 | 486225,68 | 529425,78 | 809236,99 |
| Total general | | | | 1295222,59 |

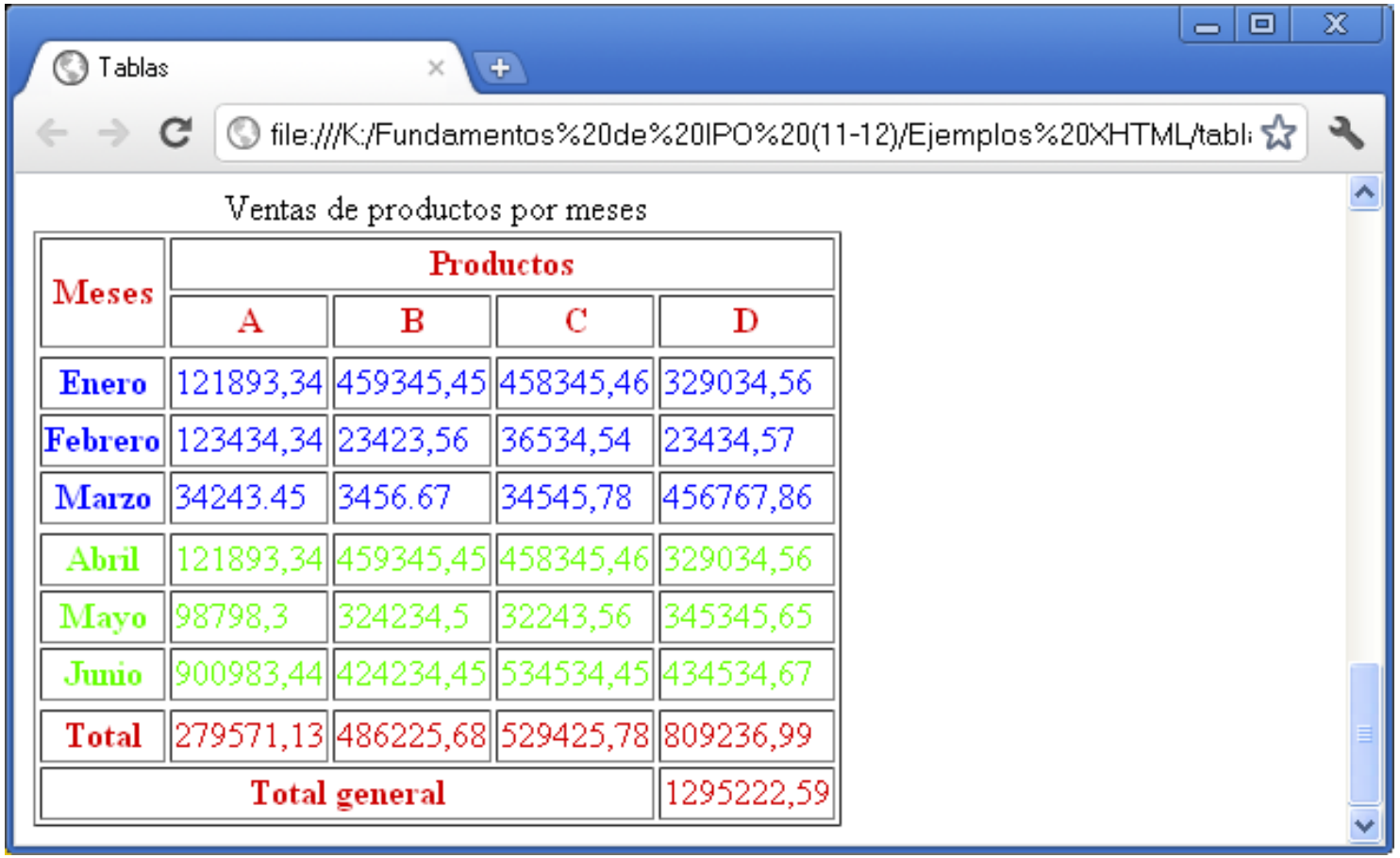
# Tablas
## Grupos de filas

- Las filas de una tabla pueden agruparse en:
  - Una cabecera de tabla.
    - ✓ Elemento `thead`.
  - Un pie de tabla.
    - ✓ Elemento `tfoot`.
  - Una o más secciones del cuerpo de la tabla.
    - ✓ Elementos `tbody`.
- Esta división permite a los agentes de usuario la posibilidad de especificar las secciones de la tabla para tratar la presentación de forma diferente.
  - Al imprimir tablas largas se podría repetir la información de la cabecera a lo largo de toda la tabla.
- Los elementos deben aparecer en el siguiente orden:
  - `thead`.
  - `tfoot`.
  - Lista de elementos `tbody`.

# Tablas
## Grupos de filas (II)

```
<table border="1" summary="Esta tabla representa las ventas de los
  productos A, B, C y D por meses. Al final aparece el total
  de ventas de cada producto">
    <caption>Ventas de productos por meses</caption>
    <thead style="color:#CC0000">
    <tr>
      <th id="meses" rowspan="2">Meses</th>
      <th scope="col" id="productos" colspan="4">Productos</th>
     </tr>
     <tr>
     <th scope="col" id="prodA">A</th>
       ...
    </tr>
    </thead>
    <tfoot style="color:#CC0000">
    <tr>
      <th scope="row" id="total">Total</th>
      <td headers="total prodA">279571,13</td>
      ...
    </tr>
    <tr>
       <th colspan="4" id="totalGeneral">Total general</th>
       <td headers="totalGeneral">1295222,59</td>
    </tr>
    </tfoot>
    <tbody id="primertrimestre" style="color:#0000FF">
    ...
    </tbody>
    <tbody id="segundotrimestre" style="color:#66FF00">
    ...
    </tbody>
</table>
```

# Tablas
## Grupos de filas (III)




Ventas de productos por meses

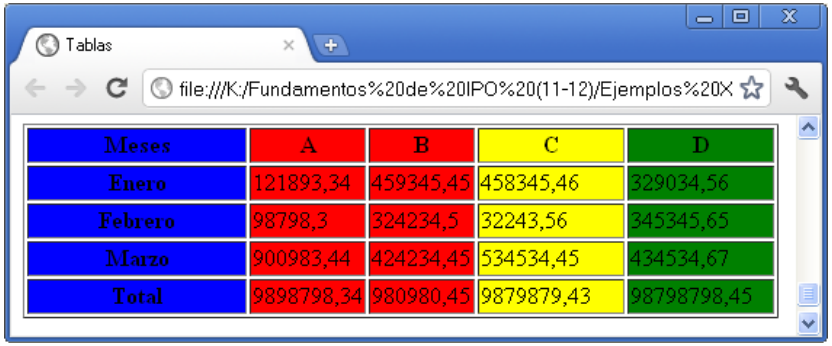
| Meses | Productos | | | |
|---|---|---|---|---|
| | A | B | C | D |
| Enero | 121893,34 | 459345,45 | 458345,46 | 329034,56 |
| Febrero | 123434,34 | 23423,56 | 36534,54 | 23434,57 |
| Marzo | 34243.45 | 3456.67 | 34545,78 | 456767,86 |
| Abril | 121893,34 | 459345,45 | 458345,46 | 329034,56 |
| Mayo | 98798,3 | 324234,5 | 32243,56 | 345345,65 |
| Junio | 900983,44 | 424234,45 | 534534,45 | 434534,67 |
| Total | 279571,13 | 486225,68 | 529425,78 | 809236,99 |
| Total general | | | | 1295222,59 |

# Tablas
## Grupos de columnas

- Es posible agrupar estructuralmente varias columnas.
  - Esta agrupación permite asignar ciertos atributos a columnas determinadas.
  - Desde el punto de vista de la accesibilidad se utiliza para agrupar columnas (por ejemplo, una columna con totales o un grupo de columnas con los productos).
  - El elemento `col` permite marcar una o varias columnas consecutivas.
    - ✓ Es una etiqueta sin contenido (`<col … />`)
    - ✓ Aparece después del elemento `table`.
    - ✓ El atributo `span` permite dar el mismo atributo a tantas columnas consecutivas como indique su valor.
    - ✓ El atributo `width` permite establecer el ancho de las columnas.
    - ✓ El atributo `style` permite dar un estilo determinado a las columnas.
      - También es posible establecer el estilo visual de una serie de columnas mediante los atributos `id` y `class` y hojas de estilo.
  - El elemento `colgroup` permite agrupar varias columnas consecutivas.
    - ✓ Puede tener o no contenido. Si tiene contenido tendrá una etiqueta de cierre (`</colgroup>`).
    - ✓ También tiene los atributos `span`, `width` y `style`.

# Tablas
## Grupos de columnas (II)

```
<table border="1" summary="...">
  <caption>...</caption>
  <col width="30%" style="background-color:blue" />
  <colgroup span="2" style="background-color:red"/>
  <colgroup width="20%">
     <col style="background-color:yellow" />
      <col style="background-color:green" />
  </colgroup>
  <tr>
  ...
```



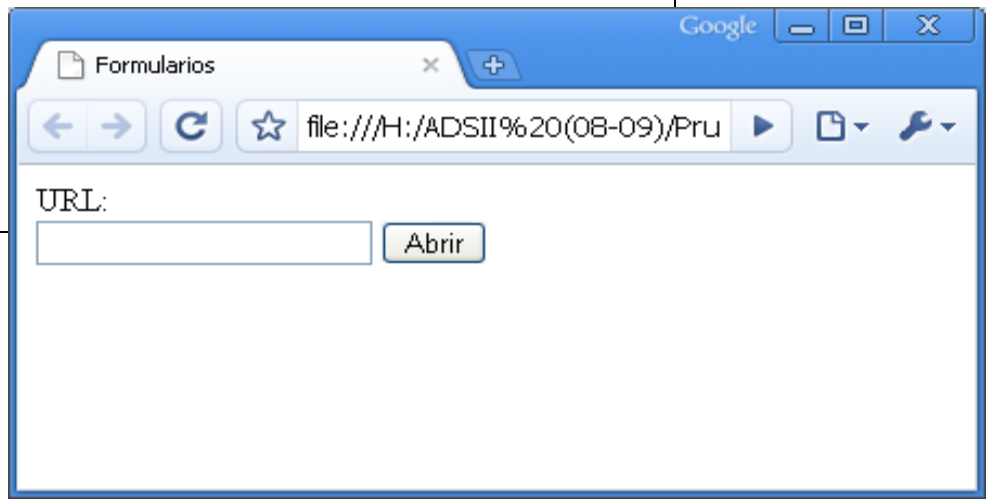

| Meses | A | B | C | D |
|---|---|---|---|---|
| Enero | 121893,34 | 459345,45 | 458345,46 | 329034,56 |
| Febrero | 98798,3 | 324234,5 | 32243,56 | 345345,65 |
| Marzo | 900983,44 | 424234,45 | 534534,45 | 434534,67 |
| Total | 9898798,34 | 980980,45 | 9879879,43 | 98798798,45 |

- Asigna a la primera columna un ancho del 30% y el color de fondo azul.
- Agrupa las siguientes dos columnas y les da un color de fondo rojo.
- Agrupa las siguientes columnas y les da un ancho del 20%.
- Dentro del grupo anterior, la primera columna tiene un color de fondo amarillo y la segunda verde.

# Formularios

- Se utilizan cuando se desea que el usuario introduzca información en un documento xhtml para ser utilizada por una aplicación Web.
- Para la creación de formularios básicos se utilizan los elementos `form` e `input`.

```
<form action="document.forms['abrirVentana'].submit()"
   id="abrirVentana"
   onsubmit="window.open(document.forms['abrirVentana'].url.value);
   return(false);">
   <div>
      <label for="url">URL:</label> <br/>
      <input name="url" id="url" type="text" />
      <input name="abrir"
          type="submit" value="Abrir" />
   </div>
</form>
```



# Formularios
## Elemento form

- Encierra todos los elementos del formulario.
  - Además de los controles del formulario (`input`, `label`, `select`, `textarea`, `button`) puede contener otros elementos html de bloque (`p`, `list`, `div`, `etc`.).
    - ✓ Los controles de formulario son elementos de línea, por lo que deben ir dentro de otro elemento de bloque.
- Atributo `action`.
  - Especifica el url de la aplicación que procesará los datos del formulario.
  - También puede incluir url o instrucciones de javascript.
    - ✓ `action="document.forms['abrirVentana'].submit()"`
      - Indica que realice la llamada a la función `submit` (que se produce al enviar el formulario) del formulario actual.
- Atributo `id`.
  - Permite identificar de forma unívoca el formulario de entre todos los formularios del documento.
- Atributos de eventos.
  - Puede incluir instrucciones de javascript o una llamada a una función javascript.
    - ✓ `onsubmit="window.open(document.forms['abrirVentana'].url.value);`
      - Al enviar el formulario, abre una ventana con el valor del cuadro de texto `url`.

# Formularios
## Elemento form (II)

- Atributo `method`.
  - Indica la forma en la que se envían los datos del formulario mediante http a la aplicación que los procesa.
  - Puede tomar los valores `post` o `get` (opción por omisión).
    - ✓ `get`, el conjunto de datos del formulario se agrega al URI especificado por el atributo `action` (con un signo de interrogación ("?") como separador) y este nuevo URI se envía al agente procesador.
      - Sólo admite unos 500 bytes de información.
      - Sólo admite caracteres ASCII en los datos enviados.
      - Los datos aparecen en la llamada al agente procesador.
      - Se utiliza para datos no sensibles, es decir cuando la aplicación no modifica los datos (por ejemplo, búsquedas en bases de datos).
    - ✓ `post`, el conjunto de datos del formulario se incluye en el cuerpo del formulario y se envía al agente procesador.
      - Oculta los datos en la llamada a la aplicación.
      - Permite enviar cualquier carácter del conjunto de caracteres.
        - Para ellos es necesario establecer el atributo `enctype` del elemento `form` a `"multipart/form-data"`.
      - Se utiliza para datos sensibles, es decir cuando el agente procesador causa efectos secundarios (por ejemplo cuando se modifica una base de datos).

# Formularios
## Elemento input

- Representa a la mayoría de los controles del formulario.
  - Mediante el atributo `type`, se especifica el tipo de control.
  - En todos es obligatorio el atributo `name` que identifica el control para que se pueda procesar el formulario.
    - ✓ Se enviará a la aplicación parejas de valores `nombreControl=valorContol`.
      - El nombre del control será el asignado en el atributo `name`.
      - Cada pareja de valores se separa con el carácter *ampersand* (&).
  - En todos los controles:
    - ✓ `alt`, texto con la descripción del control.
    - ✓ `readonly="readonly"`, indica que el control es de sólo lectura.
    - ✓ `id`, identificador único que permite identificar el control. A menudo se utiliza para relacionar el control con una etiqueta.
- Cuadros de texto.
  - `type="text"` (valor por omisión).
    - ✓ Permite utilizar los atributos…
      - `value`, indica el valor inicial.
      - `size`, tamaño del formulario en número de caracteres.
      - `maxlength`, tamaño máximo en número de caracteres.

# Formularios
## Elemento input (II)

- Cuadros de contraseña.
  - `type="password"`.
    - ✓ Funciona igual que los cuadros de texto.
    - ✓ Los caracteres tecleados no aparecen en el control.
- Casillas de verificación.
  - `type="checkbox"`.
    - ✓ Muestra una casilla de verificación sin texto.
      - Es necesario añadir el texto sobre el significado del control.
    - ✓ Permite utilizar los atributos…
      - `checked="checked"`, el control aparece como marcado.
      - `value`, establece el valor que se enviará como información para procesar.

```
<input name="acepto" type="checkbox" value="si"  />
```

        – Enviará al formulario la pareja `acepto=si`.

# Formularios
## Elemento input (III)

- Botones de radio.
  - `type="radio"`.
    - ✓ Funciona igual que las casillas de verificación.
    - ✓ Los elemento marcados son excluyentes entre los botones de radio que tengan el mismo valor en el atributo `name`.
      - Permite seleccionar o la tarjeta Visa o la MasterCard (sólo una de ellas).

```
<input name="tarjeta" type="radio" value="V" id="visa"
  checked="checked" />
<input name="tarjeta" type="radio" value="M" id="mastercard"
  />
```

        - Enviará al formulario la pareja `tarjeta=V` o `tarjeta=M`.
- Campos ocultos.
  - `type="hidden"`.
    - ✓ Permite enviar información adicional a la aplicación.
    - ✓ Envía valores fijos que el usuario no puede ver ni modificar.
    - ✓ Los atributos `name` y `value` especifican la pareja de valores que se envían a la aplicación.

# Formularios
# Elemento input (IV)

- Botón de envío.
  - `type="submit"`.
    - ✓ Se utiliza para enviar los datos a la aplicación.
    - ✓ Los atributos `name` y `value` especifican la pareja de valores que se envían a la aplicación.
      - `value`, especifica también el valor que aparece como etiqueta del botón.
      - Si no aparece el atributo `value`, tomará el valor "Enviar consulta".
- Botón para restablecer la información.
  - `type="reset"`.
    - ✓ Se utiliza para borrar los datos del formulario o para restablecerlos a su valor original.
      - Actualmente, su utilización ha caído en desuso.
      - El atributo `value` permite especificar la etiqueta del botón.
        - Si no aparece, el valor por omisión será "Restablecer".
- Envío de archivos adjuntos.
  - `type="file"`.
    - ✓ Muestra un cuadro de texto con un botón "Examinar…".
    - ✓ Al pulsar el botón aparece el cuadro de diálogo "Abrir archivo" que permite seleccionar un archivo.
    - ✓ Al formulario se envía el nombre del control acompañado del nombre del archivo seleccionado.
    - ✓ Si se utiliza, entre los atributos del formulario debería estar `enctype` con el valor`"multipart/form-data"`.

# Formularios
# Elemento input (V)

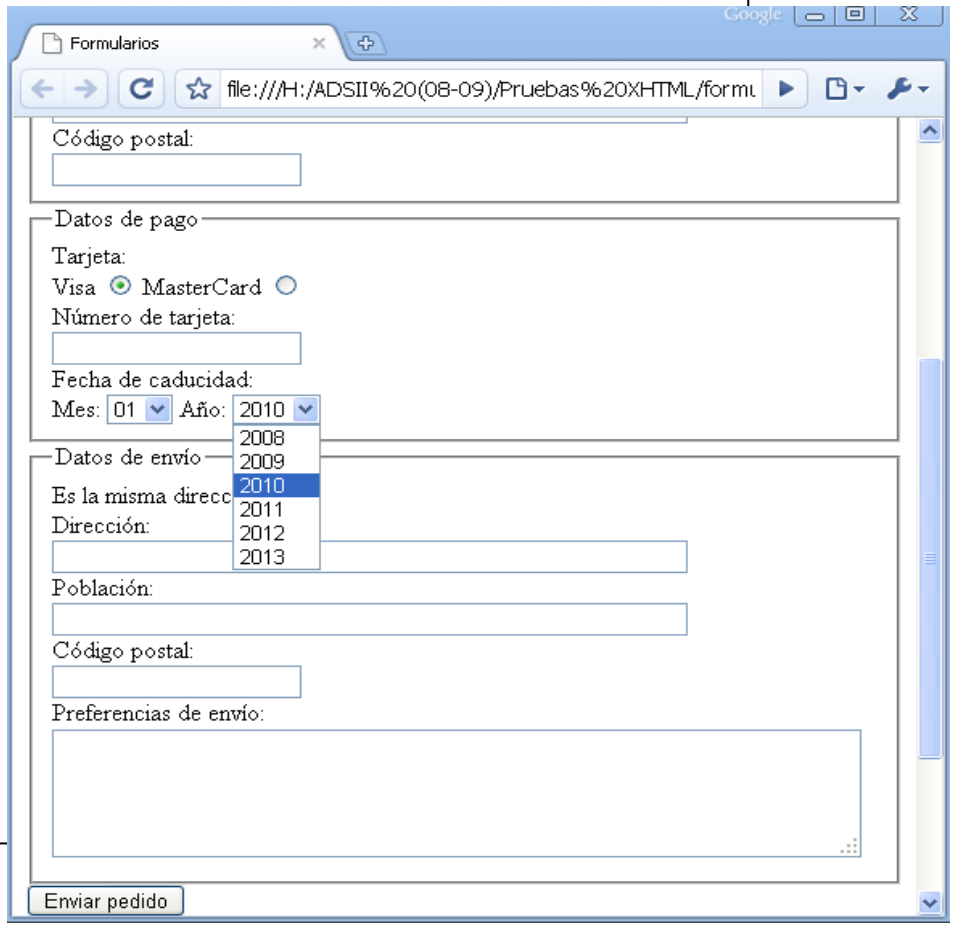
- Botón.
  - `type="button"`.
    - ✓ Permite utilizar otros botones, además de los de enviar y restaurar.
    - ✓ Los atributos `name` y `value` permiten identificar al botón y dotarle de una etiqueta respectivamente.
    - ✓ Por si solos estos botones no hacen nada.
      - Mediante javascript es posible asociarles acciones.
- Botones de imagen.
  - `type="image"`.
    - ✓ Se trata de un botón al que se le asocia una imagen.
      - El atributo `src` permite indicarle la imagen.
      - Por accesibilidad, deberían llevar también el atributo `alt`.
    - ✓ Al pulsarlo, además de enviar al formulario las parejas de valores con `name` y `value`, envía los valores `nombreControl.x` y `nombreControl.y` con las coordenadas x e y de dónde se ha pulsado.

# Formularios
## Otros elementos

- Áreas de texto
  - Elemento `textarea`.
    - ✓ Permite describir cuadros de texto multilínea.
      - Los atributos `cols` y `rows` permiten establecer el ancho y largo del control.
- Cuadros de lista y listas desplegables.
  - Elemento `select`.
    - ✓ Define tanto las listas deplegables como los cuadros de lista.
      - El atributo `size` indica el número de elementos que se visualizarán.
        - Por omisión sólo se muestra una (se tratará de una lista desplegable).
        - Para los cuadros de lista se utilizará un valor mayor que 1.
      - El atributo `multiple="multiple"` indica que se permite la selección múltiple.
  - Cada opción de la lista sería un elemento `option`.
    - ✓ El atributo `selected="selected"` indica si el elemento estará seleccionado.
    - ✓ El atributo `value` se utiliza para indicar el valor que se enviará a la aplicación que procesa el formulario.
      - Los datos se enviarán en la forma:
        - `nombreSelect=valueElementoSeleccionado`.
  - El elemento `optgroup` permite hacer agrupaciones lógicas con los elementos de una lista.
    - ✓ El atributo `label` permite dar un nombre al grupo de opciones.

# Formularios
## Otros elementos (II)

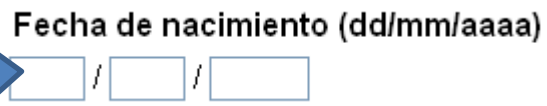
```
...
<label for="mes">Mes:</label>
<select  id="mes" name="mes">
   <option value="01">01</option>
   <option value="02">02</option>
   <option value="03">03</option>
   <option value="04">04</option>
   <option value="05">05</option>
   <option value="06">06</option>
   <option value="07">07</option>
   <option value="08">08</option>
   <option value="09">09</option>
   <option value="10">10</option>
   <option value="11">11</option>
   <option value="12">12</option>
</select>
<label for="anno">A&ntilde;o: </label>
<select id="anno" name="anno">
   <option value="2008">2008</option>
   <option value="2009">2009</option>
   <option value="2010">2010</option>
   <option value="2011">2011</option>
   <option value="2012">2012</option>
   <option value="2013">2013</option>
</select>
...
Preferencias de env&iacute;o: <br />
<textarea rows="5" cols="60" name="preferencias"></textarea>
...
```



# Formularios
## Etiquetar los elementos

- Es conveniente etiquetar los elementos de forma que los agentes de usuario puedan identificar el cometido de los controles:
  - Agrupar los controles relacionados mediante el elemento `fieldset`.
  - Etiquetar los controles individuales con el elemento `label`.
  - Si por alguna razón no se pudieran etiquetar los controles, también se puede utilizar el atributo `title` del control para identificarlo.

Por cuestiones de maquetación es preferible identificar estos cuadros de texto con `title`




- Elemento `fieldset`.
  - Permite hacer una agrupación lógica de controles relacionados.
  - Debería incluir un elemento `legend` para identificar el nombre del grupo de controles.
    - ✓ En los agentes visuales, los controles relacionados estarán agrupados visualmente mediante una línea.
    - ✓ En la línea parecerá el contenido del elemento `legend`.

# Formularios
## Etiquetar los elementos (II)

- Elemento `label`.
  - Permite añadir una etiqueta a un control.
  - Se deben usar para los elementos `textarea`, `select` e `input` de tipo `text`, `checkbox`, `radio`, `file` y `password`.
  - No se usa para los siguientes elementos:
    - ✓ `button`. Se usa su contenido como etiqueta.
    - ✓ `input` de tipo `image`. Se usa el atributo `alt` como etiqueta.
    - ✓ `input` de tipo `submit` y `reset`. Se usa el atributo `value` como etiqueta.
    - ✓ `input` de tipo `hidden`. No se usa etiqueta.
  - Debe identificar adecuadamente el control de formulario y debe situarse:
    - ✓ Antes del control para elementos `input` de tipo `text`, `file`, `password` y para los elementos `textarea` y `select`
    - ✓ Después del control para elementos `input` de tipo `checkbox` y `radio`

# Formularios
## Etiquetar los elementos (III)

- Elemento `label` (*continuación*).
  - Asociación de la etiqueta al control.
    - ✓ Se puede asociar de forma implícita o explícita.
      - Para asociarlo de forma implícita el control se debe encerrar dentro del elemento.

```
<label >Apellidos y nombre:<br />
   <input name="nombre" type="text" id="apellidosynombre" size="60" /><br />
</label>
```

      - Para asociarlo de forma explícita se utiliza el atributo `for`.
        - Indica el identificador del control al que está asociado. El control debería llevar el atributo `id` con el mismo valor que aparece en el `for`.
    - ✓ Las normas de accesibilidad recomiendan hacer una asociación explícita.

```
<label for="apellidosynombre">Apellidos y nombre:</label><br />
<input name="nombre" type="text" id="apellidosynombre" size="60" /><br />
```

# Formularios
## Navegación por teclado

- Para navegar por los controles mediante el teclado se utiliza la tecla de tabulación.
  - El orden de tabulación es el orden en que aparecen los controles dentro del código HTML.
  - El atributo `tabindex` permite variar el orden de tabulación.
    - ✓ Está presente en los elementos `a`, `area`, `button`, `input`, `select` y `textarea`.
    - ✓ Puede tomar un valor entre 0 y 32767.
    - ✓ Si existen elementos con el atributo `tabindex` el orden de tabulación será:
      - Primero los elementos con atributo `tabindex` mayor que 0, según el valor del atributo.
      - Después los elemento sin atributo `tabindex` o con un valor 0 según el orden en el código.

# Formularios
## Navegación por teclado (II)

- Teclas de acceso.
  - Es posible asignar una tecla de acceso a un control (el control tomará el foco cuando se pulse una combinación de teclas) mediante el atributo `accesskey`.
  - El atributo está presente en los elementos `a`, `area`, `button`, `input`, `label`, `legend` y `textarea`.
  - El valor del atributo será un carácter.
  - La llamada a las teclas de acceso depende del sistema operativo y del navegador.
    - ✓ En Windows…
      - En Firefox, IE, Chrome y Safari: alt+tecla de acceso (hay que tener en cuenta que en Firefox, si la tecla es mayúscula hay que pulsar también shift).
      - En Opera ctrl+esc+tecla de acceso.
- Controles deshabilitados.
  - Los controles `button`, `input`, `optgroup`, `option`, `select` y `textarea` tienen el atributo booleano `disabled` para deshabilitar el control.
  - El control no será accesible por el usuario y no entrará en el orden de tabulación.
    - ✓ El contenido del atributo en XHTML será `disabled="disabled"`.
      - En HTML sólo habría que poner `disabled`.

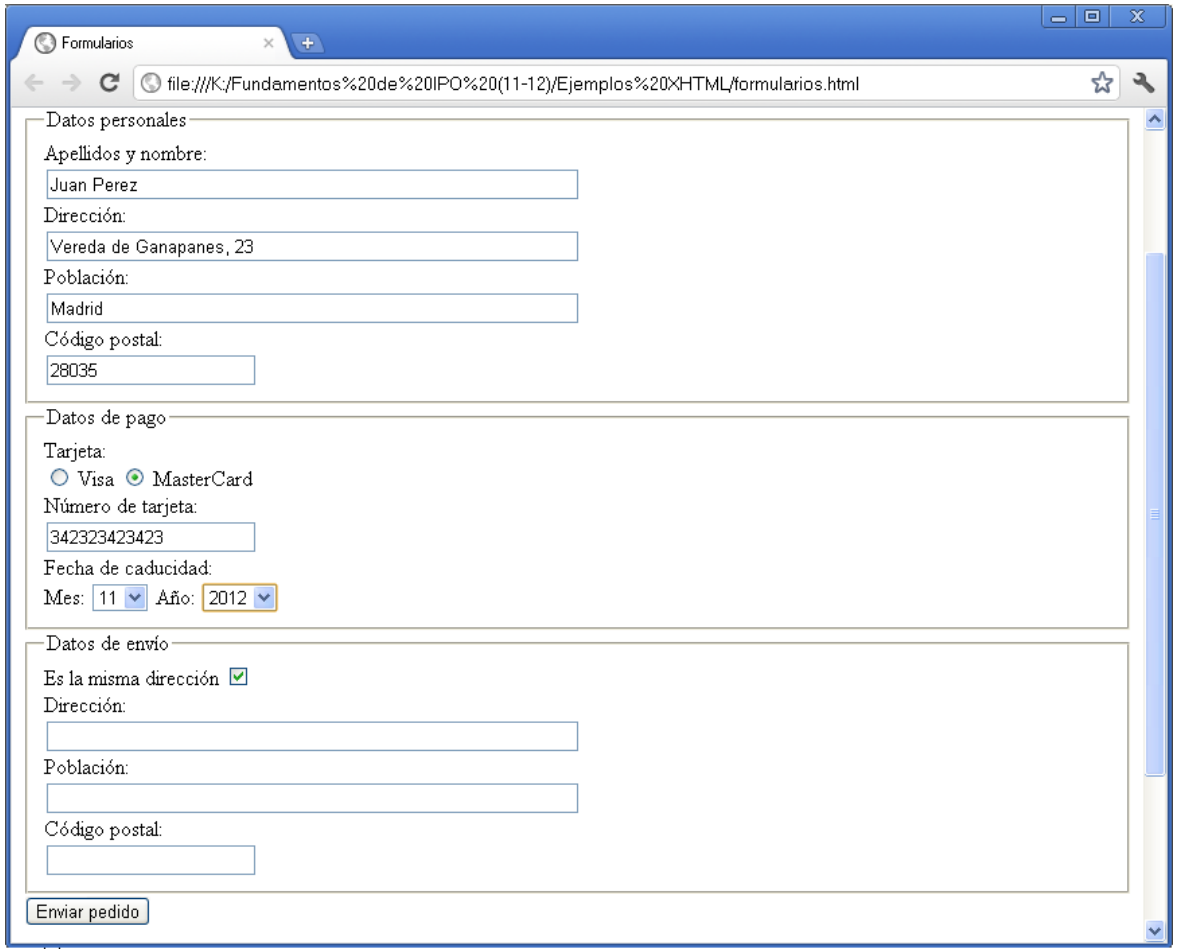
# Formularios
## Ejemplo

```
<form id="frmPrueba" method="get" action="http://miservidor.com/bin/aplicacion.pl">
   <fieldset>
      <legend>Datos personales</legend>
      <label accesskey="A" for="apellidosynombre">Apellidos y nombre:</label><br />
      <input name="nombre" type="text" id="apellidosynombre" tabindex="1" size="60" /><br />
              <label for="direccion">Direcci&oacute;n:</label><br />
              <input name="direccion" id="direccion" type="text" size="60" tabindex="2"/><br />
              <label for="poblacion">Poblaci&oacute;n:</label> <br />
              <input name="poblacion" id="poblacion" type="text" size="60" tabindex="3"/><br />
              <label for="cp">C&oacute;digo postal:</label><br />
              <input name="cp" id="cp"type="text" size="20" maxlength="5" tabindex="4"/>
   </fieldset>
   <fieldset>
   <legend>Datos de pago</legend>
   Tarjeta:<br />
   <input name="tarjeta" type="radio" value="V" id="visa" checked="checked" tabindex="5" />
   <label for="visa">Visa</label>
   <input name="tarjeta" type="radio" value="M" id="mastercard" tabindex="6"/>
   <label for="mastercard">MasterCard</label><br />
   <label for="numerotarjeta">N&uacute;mero de tarjeta:</label> <br />
   <input name="numerotarjeta" id="numerotarjeta" type="text" size="20" tabindex="7"/><br />
   Fecha de caducidad: <br />
   <label for="mes">Mes: </label>
   <select id="mes" name="mes">
      <option value="01">01</option>
      ...
      <option value="12">12</option>
   </select>
```

# Formularios
## Ejemplo (II)

```
<label for="anno">A&ntilde;o:</label>
   <select id="anno" name="anno">
      <option value="2008">2008</option>
      ...
      <option value="2013">2013</option>
   </select>
   </fieldset>
   <fieldset>
      <legend>Datos de env&iacute;o</legend>
             <label for="misma" accesskey="M">Es la misma direcci&oacute;n </label>
             <input name="mismadireccion" id="misma"
                    type="checkbox" value="lamisma" checked="checked" /><br />
             <label for="direccionenvio">Direcci&oacute;n:</label><br />
             <input name="direccionenvio" id="direccionenvio" type="text" size="60" /><br />
             <label for="poblacionenvio">Poblaci&oacute;n: </label><br />
             <input name="poblacionenvio" id="poblacionenvio" type="text" size="60" /><br />
             <label for="cpenvio">C&oacute;digo postal:</label><br />
             <input name="cpenvio" id="cpenvio" type="text" size="20" maxlength="5"/><br />
   </fieldset>
   <div><input type="submit" name="enviar" value="Enviar pedido" /></div>
</form>
```

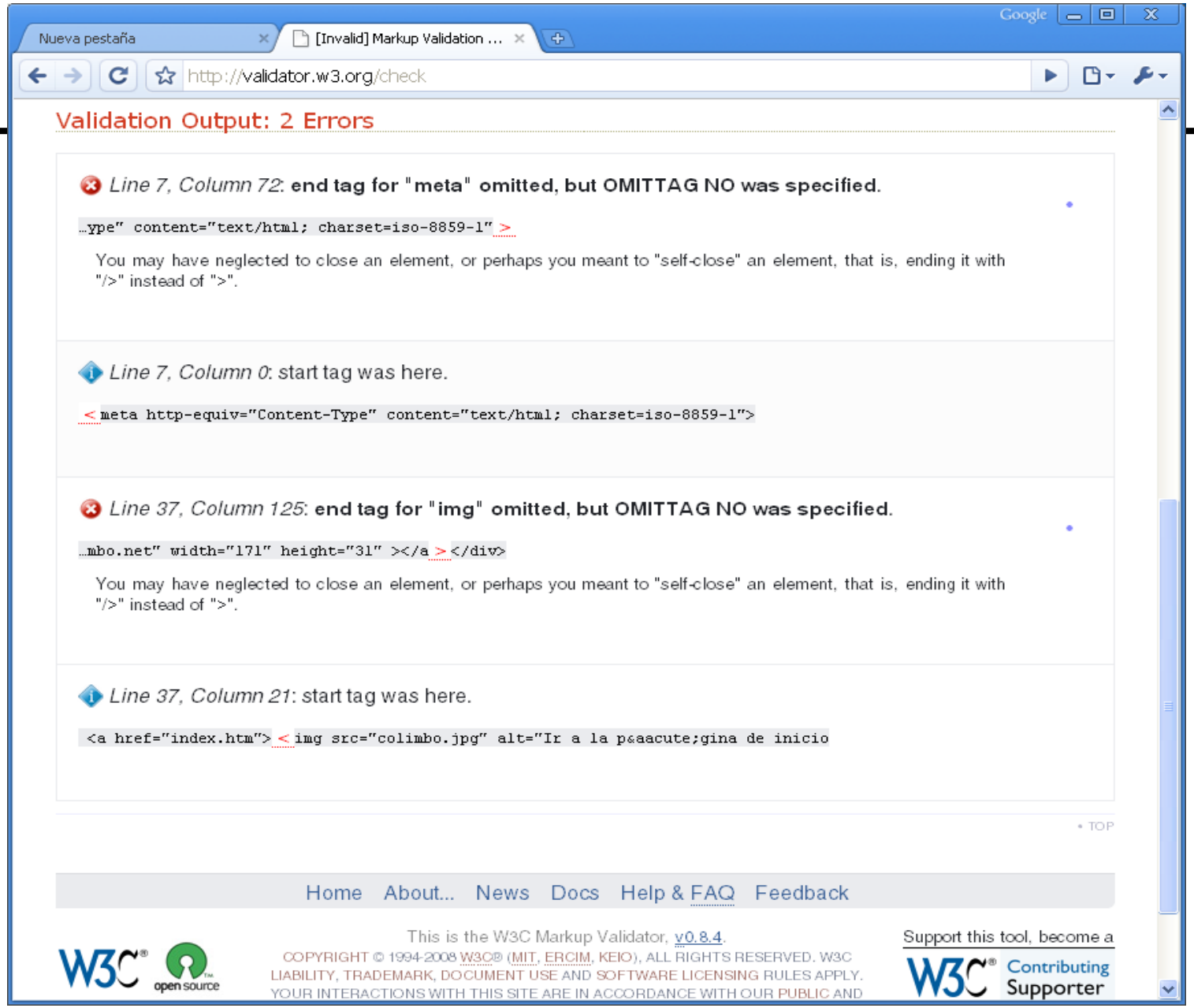
# Formularios
## Ejemplo (III)




La información que se enviaría sería:
http://miservidor.com/bin/aplicacion.pl?nombre=Juan+Perez&direccion=Vereda+de+Ganapanes%2C+23&poblacion=Madrid&cp=28035&tarjeta=M&numerotarjeta=342323423423&mes=11&anno=2012&mismadireccion=lamisma&direccionenvio=&poblacionenvio=&cpenvio=&enviar=Enviar+pedido

# Validación de código

- Es importante comprobar si el código xhtml cumple los estándares.
- Los validadores comprobarán si el documento está bien formado y si se ajustan a las especificaciones del tipo de documento o esquema indicado en el preámbulo del documento.
- Algunos validadores:
  - W3C (validator.w3.org/).
    - ✓ Permite validar contra un URI, contra un archivo local o introduciendo el código directamente.


Nueva pestaña
[Invalid] Markup Validation ...
http://validator.w3.org/check
Validation Output: 2 Errors
Line 7, Column 72: end tag for "meta" omitted, but OMITTAG NO was specified.
…ype" content="text/html; charset=iso-8859-1" >
You may have neglected to close an element, or perhaps you meant to "self-close" an element, that is, ending it with "/>" instead of ">".
Line 7, Column 0: start tag was here.
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
Line 37, Column 125: end tag for "img" omitted, but OMITTAG NO was specified.
…mbo.net" width="171" height="31" ></a >
You may have neglected to close an element, or perhaps you meant to "self-close" an element, that is, ending it with "/>" instead of ">".
Line 37, Column 21: start tag was here.
<a href="index.htm"><img src="colimbo.jpg" alt="Ir a la página de inicio
• TOP
Home About... News Docs Help & FAQ Feedback
This is the W3C Markup Validator, v0.8.4.
COPYRIGHT © 1994-2008 W3C® (MIT, ERCIM, KEIO), ALL RIGHTS RESERVED. W3C LIABILITY, TRADEMARK, DOCUMENT USE AND SOFTWARE LICENSING RULES APPLY. YOUR INTERACTIONS WITH THIS SITE ARE IN ACCORDANCE WITH OUR PUBLIC AND
Support this tool, become a
Contributing Supporter

# Validación de código (III)

- Complemento HTML Validator para Firefox (addons.mozilla.org/es-ES/firefox/addon/249).