Tutorial de XML.

Mario A. Valdez-Ramírez,
Interactive Bureau México.
Editor de MSDN Latinoamérica.

¿Qué es el XML?

Repasemos lo conocido...

Comparando con...

- HTML (*HyperText Markup Language*).
- SGML (*Standard Generalized Markup Language*).
- PDF (*Portable Document Format*).

HTML: Lo bueno.

- El hipertexto funciona.
- Es multiplataforma.
- Tiene una curva de aprendizaje muy plana.
- Barato (muchos editores, visores, verificadores, etc., gratuitos).
- Base de información grande.
- Los navegadores son baratos, sencillos de construir y de usar y poderosos.

HTML: Lo malo.

- Pobre herramienta de presentación.
  - » Poco control de espaciado.
  - » Problemas con el control de guiones, *kerning*, justificación y otras manipulaciones de texto.
  - » EL uso de columnas es problemático.
- Pobre herramienta de marcaje (*markup*).
  - » No se pueden agregar etiquetas nuevas.
  - » No es modular, poca oportunidad de reciclar.
  - » Hay demasiado código inválido publicado actualmente.

HTML: Lo peor.

- No puede ser extendido elegantemente.
  - » Las etiquetas son fijas.
  - » Las compañías y personas involucradas en hacer extensiones no saben de composición (*typesetting*) ni edición estructurada.
  - » Es campo de batalla comercial (Mozilla vs IE).

HTML: Los problemas.

- La principal queja es:

  “No puedo hacer [x cosa]”.

- Las razones:
  - » Es solo un conjunto de etiquetas.
  - » Pocos usan el DTD como guía.
  - » Los navegadores siempre tienen que lidiar con código mal escrito.

HTML: Lo nuevo.

- Las hojas de estilo en cascada (*cascading style sheets*, CSS).
  - » Netscape 4.0 y superior.
  - » Internet Explorer 3.0 y superior.
  - » Opera y otros navegadores en sus últimas versiones.
- La versión 1, (CSS1) emitida como recomendación del W3C en 1996.
- Separa la estructura del formato.
- Mayor control sobre la apariencia y posición.

SGML: Lo bueno.

- Es multiplataforma.
- Es un estándar ISO estable.
- Hay disponibles muchas herramientas gratuitas para edición y conversión.
- Es un conjunto de reglas, no un conjunto de etiquetas fijas.
- Separa completamente la estructura del formato.

SGML: Lo malo.

- Es complicado.
- Es costoso.
  - » El diseño de documentos es costoso.
  - » La mano de obra es cara.
  - » El entrenamiento es caro.
  - » Aunque hay herramientas gratuitas, las que no lo son son muy costosas.

SGML: Los problemas.

- La principal queja es:

  ¡El SGML es demasiado complicado y muy costoso!

- Las razones:
  - » El análisis de requerimientos es caro.
  - » Los consultores son caros.
  - » Las herramientas no son tan diversas ni tan flexibles.

PDF: Lo bueno.

- Rápido y barato.
- Preserva perfectamente la composición (*layout*) del documento.
- Excelente para imprimir en cualquier dispositivo.
- Multiplataforma.

PDF: Lo malo.

- Archivos muy grandes.
- Poca flexibilidad.
- Pobres capacidades de búsqueda y navegación.
- Pobre capacidad para reconvertir en otros formatos.
- Pobre accesibilidad.

PDF: Los problemas.

- La queja principal es:

  ¡Solo se puede imprimir!

- Razones:
  - » Difícil de manipular.
  - » La capacidades de los dispositivos diferentes.

Se requiere algo nuevo...

- Barato, veloz y sencillo:
  - » Para crear documentos.
  - » Para procesar documentos.
  - » Para presentar documentos.
- Extensible:
  - » Un conjunto de reglas, no un conjunto de etiquetas.
- Compatible con el HTML:
  - » Debe tener una manera sencilla de convertir de HTML.
- Compatible con el SGML:
  - » Debe de conservar su potencia sin contener complejidades no necesarias.

Necesitamos XML.

Metas de diseño.

- XML debe ser utilizable a través de Internet.
- XML debe soportar muchos **escenarios** de aplicación.
- XML debe ser compatible con el SGML.
- Los programas que procesen documentos XML deben ser fáciles de crear.
- Las características opcionales deben ser idealmente cero.
- Lso documentos en XML deben de ser legibles por humanos y razonablemente claros.

Metas de diseño.

- El diseño con XML debe ser rápido.
- El diseño de documentos XML debe de ser formal y conciso.
- Los documentos XML deben de ser fáciles de crear.
- El laconismo en el uso de etiquetas no es importante.

Ahora... un ejemplo.

```
<Cliente ID="HVet950283">
   <Nombre>Hospital Veterinario Kermit</Nombre>
   <Direccion verificada="si">
        <Calle>Padre Mier 1528</Calle>
        <Ciudad>Monterrey</Ciudad>
        <Estado>NL</Estado>
        <CodigoPostal>64000</CodigoPostal>
   </Direccion>
</Cliente>
```

Sintaxis simple

Legible por personas

Muy parecido al HTML

El XML es...

- El Lenguaje de Marcaje Extensible (*Extensible Markup Languaje*, XML).
  - » Un metalenguaje de marcaje.
  - » Una sintaxis utilizada para crear lenguajes declarativos.
- Una recomendación técnica del W3C.
  - » Es un estándar del W3C, no de alguna compañía.
- Multiplataforma, simple, fácil de aprender.
  - » Es fácil construir herramientas para XML.
  - » Optimizado para usarse en Internet.
- Libre (y gratuito).

El XML **no** es...

- Un lenguaje de marcaje (*markup*).
  - » No. Es un estándar que especifica una sintaxis para crear lenguajes de marcaje.
- Solo para Web.
  - » No. Puede ser usado para describir y comunicar cualquier información estructurada.
- Un superconjunto del HTML.
  - » No. Aunque el HTML puede ser definido con sintaxis de XML.
- Un invento de [x compañía].
  - » No. XML es un estándar creado por el W3C y soportado por compañías e instituciones de todo el mundo.

El XML sirve para...

- Hacer publicación electrónica independiente del medio.
- Crear protocolos para el intercambio de datos entre miembros de una industria.
- Facilitar el procesamiento de datos usando software barato.
- Permite a las personas visualizar la información de la manera que quieran.
- Proporcionar metadatos que mejoran la calidad de la búsqueda de información.

Dos versiones.

- XML bien formado.
  - » Las etiquetas de inicio y final coinciden.
  - » Los elementos vacíos tienen una forma especial.
  - » No hay elementos traslapados.
  - » Los atributos van en comillas.
- XML válido.
  - » Es código bien formado con funciones adicionales.
  - » Se adhiere a una estructura predefinida dictada por un esquema,
    - ° DTD, DCD, SOX, etc.

Sintaxis del XML.

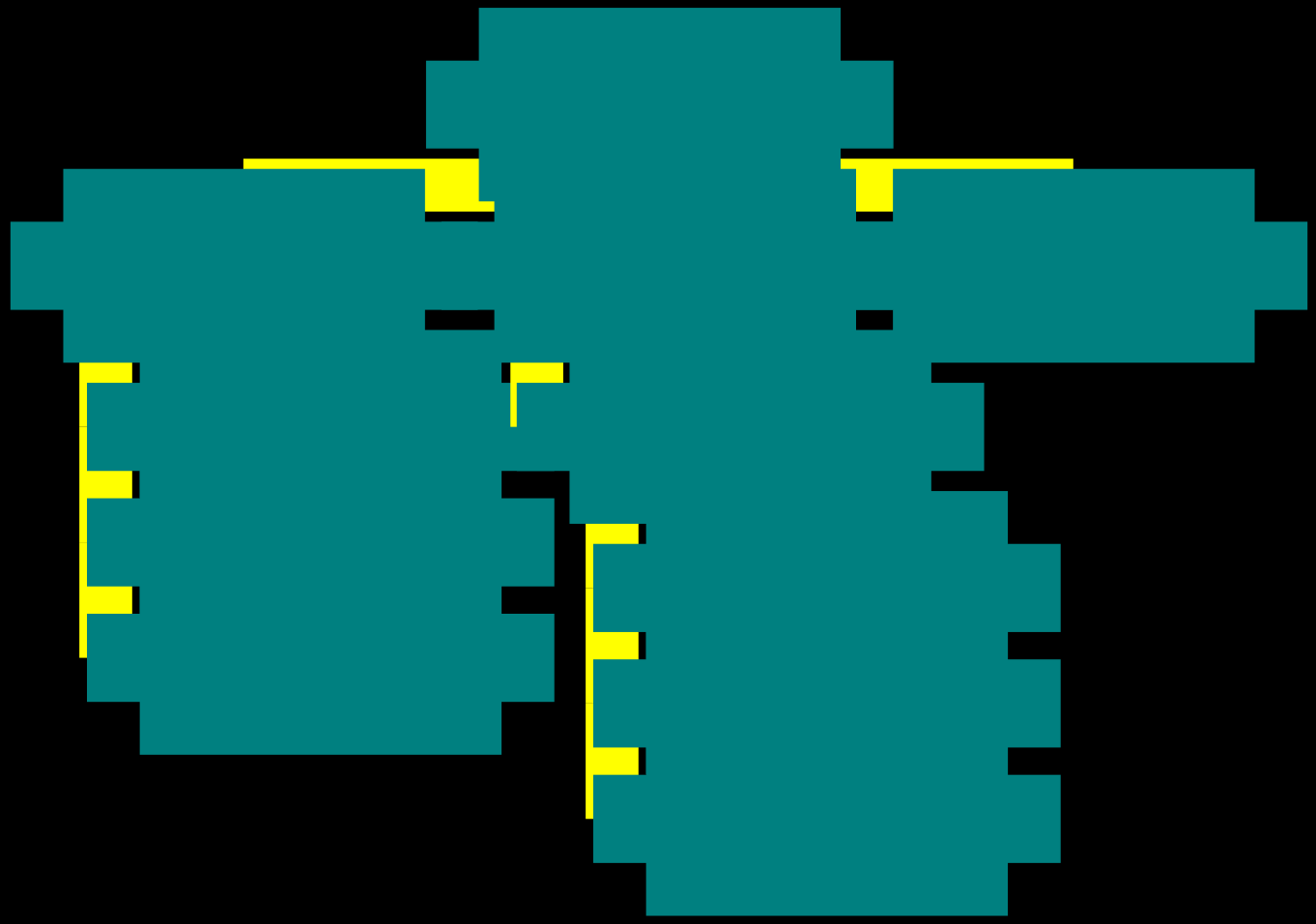
El documento XML bien formado.

Un documento XML es...

- Una colección de piezas llamadas “entidades”.
- Texto y etiquetas en Unicode.
- Válido, o por lo menos bien formado.

- Representa una jerarquía de datos.

Jerarquía de datos.



Contenedor = elemento.

- Declarativo (sustantivo).
- Lo que está encerrado entre las etiquetas.
- De lo que habla la sintaxis.
- Cinco cosas necesarias:
  - » Cómo se llama el elemento.
  - » Dónde inicia el elemento.
  - » Dónde termina el elemento.
  - » Qué contiene el elemento.
  - » Qué relación tiene el elemento con otros elementos.

Creando documentos bien formados.

- Un único elemento raíz.
- Los elementos en la raíz aparecen secuencialmente o anidados.
- Los elementos no se deben traslapar.
- Todo elemento tiene una etiqueta de inicio y una de final.
  - » Inicia con **<Nombre_elemento>**
  - » Termina con **</Nombre_elemento>**
  - » Los elementos vacios inician y terminan con **<Nombre_elemento/>**

Etiquetas.

- El XML diferencía entre mayúsculas y minúsculas.
  - » `<Libro>`, `<libro>`, `<LIBRO>` y `<LiBrO>` son etiquetas que se refieren a diferentes elementos.
- Los nombres de elementos:
  - » Deben de iniciar con una letra, subrayado o dos puntos (:).
  - » Los caracteres siguientes pueden ser letras, números, puntos, guiones, subrayados o dos puntos.
  - » El nombre “XML” y sus variaciones están reservadas.

La declaración XML.

- Dice “¡Soy un documento XML!”.
- Tiene partes específicas:

<?xml	apertura
version=“1.0”	versión
encoding=“”	codificación de caracteres
standalone=“”	doc. independ. (yes/no)
?>	fin

- Cada entidad XML que no esté en UTF-8 o UTF-16 debe contener la declaración de codificación (*encoding*).

Ejemplos de declaraciones XML.

(ninguna)

<?xml version="1.0"?>

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

<?xml version="1.0" standalone="yes"?>

<?xml version="1.0" encoding="ASCII"
standalone="no"?>

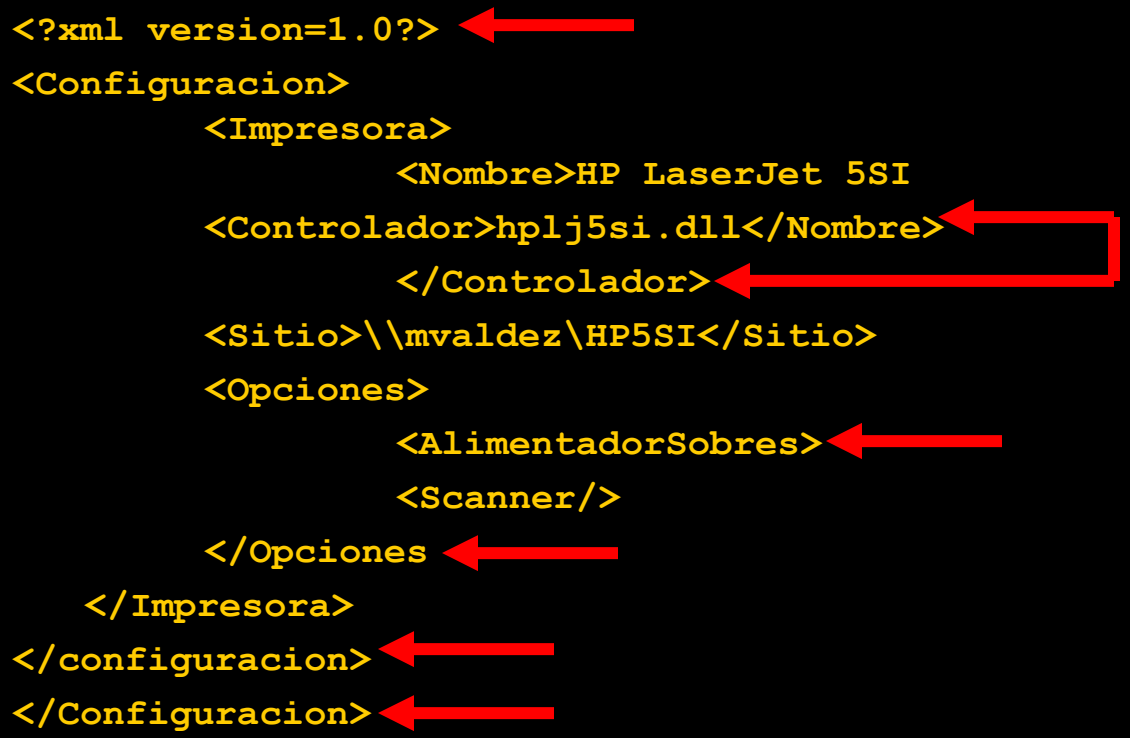
Ejemplo bien formado.

<Bienvenida>¡Hola mundo!</Bienvenida>

Ejemplo bien formado.

```
<?xml version="1.0"?>
<Configuracion>
        <Impresora>
                <Nombre>HP LaserJet 5SI</Nombre>
        <Controlador>hplj5si.dll</Controlador>
        <Sitio>\\mvaldez\HP5SI</Sitio>
        <Opciones>
                <AlimentadorSobres/>
                <Scanner/>
        </Opciones>
   </Impresora>
</configuracion>
```

Documento mal formado.



Entidades carácter.

- Para documentos bien formados:

>	`>`	(greater than)
<	`<`	(less than)
&	`&`	(ampersand)
’	`&apos;`	(apóstrofe)
”	`&quot;`	(double quote)

- Los documentos válidos deben de definir estas entidades antes de usarlas.
- Ejemplos:
  - `AT&T`
  - `Nombre=“Mario Moreno &apos;Cantinflas&apos;”`

Atributos.

- Propiedades (adjetivos).
- Contienen información acerca del elemento.
  - » Información sobre gráficos.
  - » Fechas, nombres, colores, etc.
- Aparecen en la etiqueta de inicio:

```
<Nombre_elemento Nombre_atributo="valor">
```

ó

```
<Nombre_elemento Nombre_atributo='valor'>
```

Ejemplo con atributos.

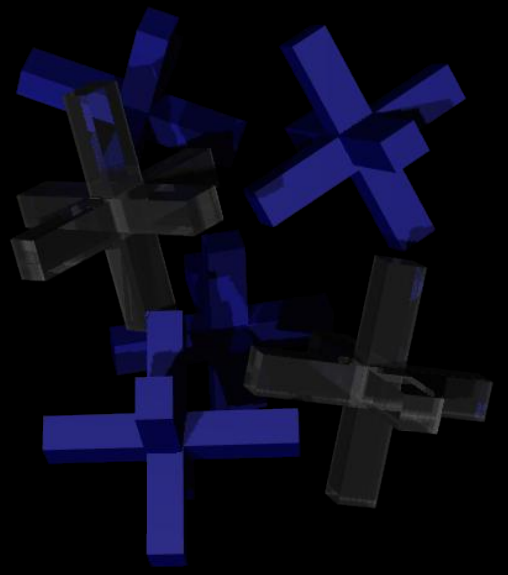
```
<?xml version="1.0"?>
<Configuracion>
        <Impresora local="si">
                <Nombre>HP LaserJet 5SI</Nombre>
        <Controlador Instalado="si">hplj5si.dll</Controlador>
        <Sitio>\\mvaldez\HP5SI</Sitio>
        <Opciones>
                <AlimentadorSobres/>
                <Scanner/>
                <Color Colores="256"/>
        </Opciones>
   </Impresora>
</Configuracion>
```

¿Qué editor puedo usar?

- Requisitos mínimos:
  - No debe generar caracteres EOF (Ctrl-Z) al final del archivo.
  - No debe generar tabulaciones (si se usan deben de expandirse a espacios al grabar).
- Sugerencias:
  - Cualquier editor de texto o procesador de palabras.
  - Editores especiales para XML.
    - Variantes de editores de SGML.
    - Editores de XML.
  - Editores de SGML.
    - Requiere hacer algunos ajustes, no recomendable.

Ejercicio: Construir un documento bien formado.

- Cargar el editor.
  - » Architag Real-time XML Editor.
    - ° http://www.architag.com/editor/
  - » Microsoft XML Notepad.
    - ° http://msdn.microsoft.com/xml/notepad/
- Crear el documento.
- Visualizar en el navegador.



La necesidad de guías de uso y estilo.

- Si queremos que todos en nuestro proceso (personas y máquinas) usen las mismas etiquetas de la misma manera, entonces requerimos...
- Guías de uso y estilo para las estructuras de datos.
  - » Listas de etiquetas válidas.
  - » Relaciones entre etiquetas.
  - » Valores predeterminados para los atributos.
  - » Tipos de datos que no sean XML estandarizados.

Guías de uso y estilo.

- Nada de esto es nuevo.
  - » Las guías de uso y estilo se han usado por décadas para intercambiar información entre personas.
- Pero el XML no es solo para personas.
  - » Se requieren guías legibles por máquinas.

XML válido.

Reglas: La DTD del XML.

- Una Definición de Tipo de Documento (Document Type Definition, DTD) permite:
  - » Definir un conjunto específico de etiquetas con relaciones específicas.
  - » Definir valores predeterminados para los atributos.
  - » Definir entidades de texto y binarias adicionales junto con sus notaciones.
  - » Indicar el elemento raíz.

El control de la DTD.

- La DTD proporciona:
  - » Una sintaxis formal que sirva de guía a un intérprete/analizador (*parser*).
  - » La habilidad de definir valores predeterminados para los atributos.
  - » Especificaciones para la estructura.
- Una DTD es una buena manera (pero no la única) de controlar la creación de datos.

Creando DTDs.

Declaración de elemento **ELEMENT**.

`<!ELEMENT`	Apertura y palabra clave.
`Nombre_elemento`	Nombre del elemento.
`( . . . )`	Modelo de contenido o
`PALABRACLAVE`	contenido declarado.
`>`	Cierre.

Palabras clave para el contenido declarado:

`EMPTY`	Sin elemento o contenido.
`ANY`	Cualquier combinación de elementos descendientes y datos caracter.

Modelo de contenido.

- Elementos o #PCDATA.
- Conectores.

,	seguido de	**(a,b)**
|	uno u otro	**(a|b)**

- Indicadores de ocurrencia.

	Uno y solo uno	**configuracion**
?	Cero o uno	**Nombre?**
+	Uno o más	**Controlador+**
*	Cero o más	**Opciones***

Ejemplos de modelos de contenido.

```
(Titulo, Seccion+)

(Titulo, (Parrafo+ | Seccion+))

(Titulo, (Parrafo | Seccion)+)

(Nombre, Numero, (Articulo, (Cantidad | Lote),
   Descripcion, precio)+, Descuento*)


<!ELEMENT Capitulo (Titulo, Seccion+)>
```

Contenido mixto.

- Caracteres (#PCDATA) que aparecen solos o en combinación con elementos descendientes en un modelo de contenido.
- Pueden ser expresados en combinaciones como un grupo o un contenido repetible:

```
(#PCDATA | grafico | tabla | lista)
```

- El mismo elemento descendiente no puede aparecer más de una vez en el grupo.

```
<!ELEMENT parrafo (#PCDATA | lista)*>
```

Comentarios XML.

- Los comentarios pueden aparecer en cualquier parte del documento fuera de otros marcajes.
- Pueden aparecer dentro de la declaración de tipo de documento.
- Un procesador XML puede, pero no requiere, ser capaz de leer y recuperar los comentarios.

```
<!-- Articulos secundarios para BD,
                revisado el 2000/I/29 -->
```

Ejemplos de declaraciones de elementos.

```
<!--    Nombre          Modelo contenido                -->
<!ELEMENT       clima   (ciudad+)                       >
<!ELEMENT       ciudad  (nombre, reporte)               >
<!ELEMENT       nombre  (#PCDATA)                       >
<!ELEMENT       reporte (alta, baja, precip?)           >
<!ELEMENT       alta    (#PCDATA)                       >
<!ELEMENT       baja    (#PCDATA)                       >
<!ELEMENT       precip  EMPTY                           >
```

Declaración de atributos **ATTLIST**.

<!ATTLIST	Apertura y palabra clave.
Nombre_elemento	Nombre del elemento.
Nombre_atributo	Nombre del atributo.
(. . .)	Lista de valores o
PALABRACLAVE	valor declarado.
“ . . . ”	Valor predeterminado o
#PALABRACLAVE	palabra clave de valor predeterminado.
>	Cierre.

Ejemplos de declaracions de atributos.

```
<!ELEMENT Novela       (titulo, parrafo+)>
<!ATTLIST Novela
   Copyright           CDATA          #REQUIRED
   PalabraClave        CDATA          #IMPLED
        type    (original|revisada|adaptada) "original"
   Estante             CDATA          #REQUIRED>
...
<Novela Copyright="1998 Ed. Diana" Estante="i1022">
...
</Novela>
```

Ejemplos de elementos con atributos.

```
<!--    Nombre          Modelo contenido                -->
<!ELEMENT       clima   (ciudad+)                        >
<!ELEMENT       ciudad  (nombre, reporte)               >
<!ELEMENT       nombre  (#PCDATA)                       >
<!ELEMENT       reporte (alta, baja, precip?)           >
<!ELEMENT       alta    (#PCDATA)                       >
<!ELEMENT       baja    (#PCDATA)                       >
<!ELEMENT       precip  EMPTY                           >
<!ATTLIST       precip  total_dia       CDATA   #REQUIRED
   tipo         (lluvia | nieve)        "lluvia"
   fuerza       (ligera | fuerte)       #IMPLIED        >
```

Declaracion de documento **DOCTYPE**.

<!DOCTYPE	Apertura y palabra clave.
Elemento_raiz	Nombre del elemento raíz.
PALABRACLAVE	SYSTEM o PUBLIC y
“dtd.dtd”	una DTD XML externa o
[. . .]	declaraciones internas.
>	Cierre.

Ejemplos de declaraciones de documento.

(ninguna)

```
<!DOCTYPE novela [
<!ELEMENT      novela  (titulo, parrafo+)          >
<!ELEMENT      titulo  (#PCDATA)                   >
<!ELEMENT      parrafo (#PCDATA)                   >
]>
```

```
<!DOCTYPE novela      SYSTEM  "novela.dtd"   >
```

Mitos de las DTDs de XML.

- El DTD clarifica el significado del documento.
  - » No necesariamente. La DTD solo especifica el orden de los elementos de un documento, no su significado.
- Es posible intercambiar información ciegamente usando una DTD.
  - » No. La DTD sirve para asegurarse de que todos los involucrados usan la misma estructura.

Ejemplo completo (XML+DTD).

```
<?xml version="1.0"?>
<!DOCTYPE clima [
<!--       Nombre     Modelo contenido     -->
<!ELEMENT clima      (ciudad+)                  >
<!ELEMENT ciudad     (nombre, reporte)          >
<!ELEMENT nombre     (#PCDATA)                  >
<!ELEMENT reporte    (alta, baja, precip?)      >
<!ELEMENT alta       (#PCDATA)                  >
<!ELEMENT baja       (#PCDATA)                  >
<!ELEMENT precip     EMPTY                      >
<!ATTLIST precip     total_dia      CDATA       #REQUIRED
     tipo       (lluvia | nieve)    "lluvia"
     fuerza     (ligera | fuerte)   #IMPLIED   >
]>
```

DTD incluida con el XML.

```
<clima>
   <ciudad>
      <nombre>Mexico DF</nombre>
      <reporte>
         <alta>27</alta>
         <baja>18</baja>
         <precip total_dia="0" tipo="lluvia" fuerza="ligera"/>
      </reporte>
   </ciudad>
   <ciudad>
      <nombre>Monterrey</nombre>
      <reporte>
         <alta>42</alta>
         <baja>36</baja>
         <precip total_dia="0" tipo="lluvia" fuerza="fuerte"/>
      </reporte>
   </ciudad>
</clima>
```

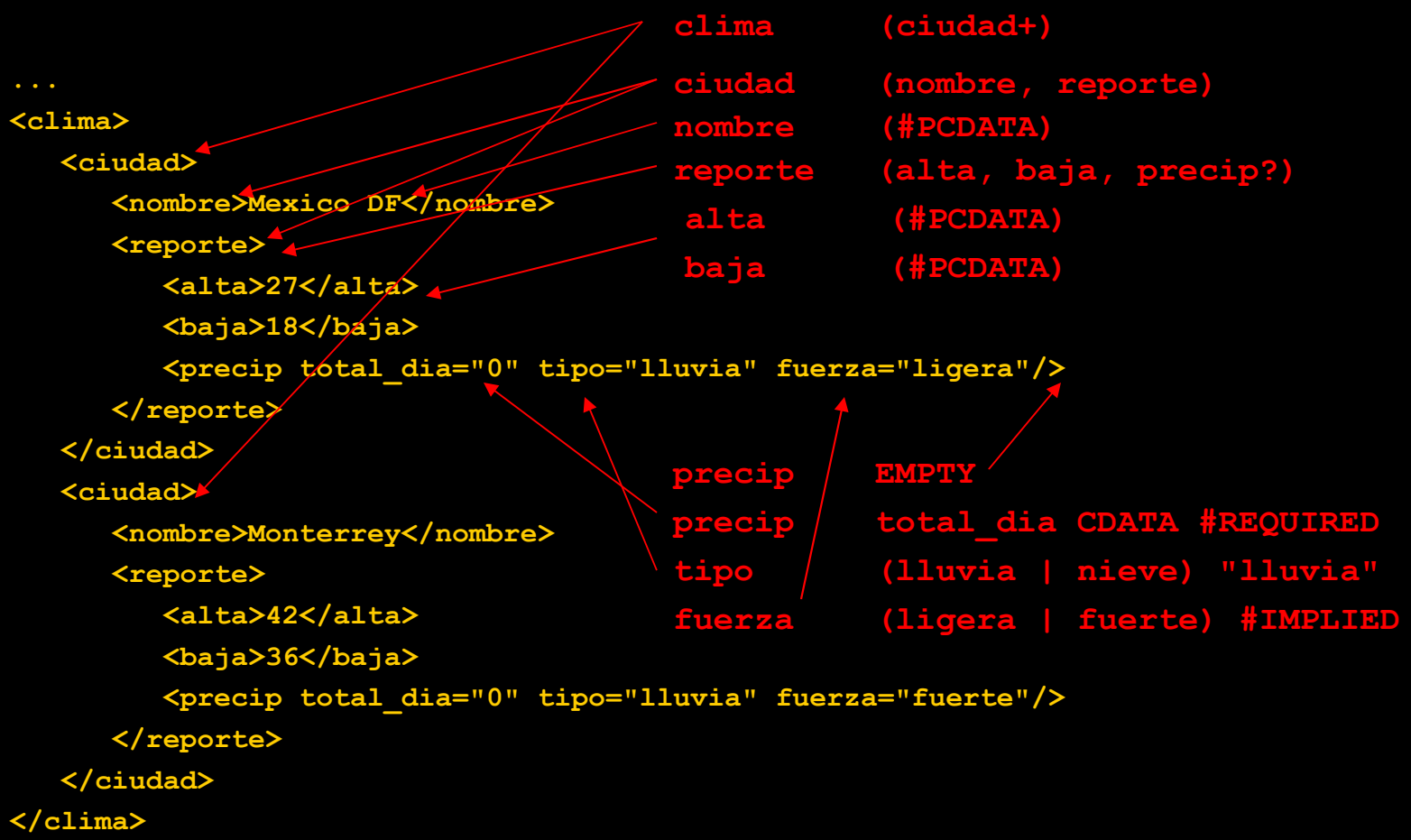
Código XML.

2 elementos de segundo nivel de ejemplo.

Ejemplo completo (XML+DTD).

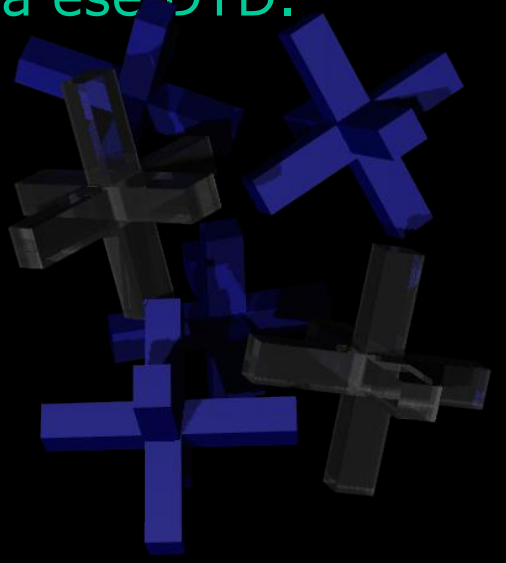
```
...
<!DOCTYPE clima [
<!--          Nombre    Modelo contenido    -->
<!ELEMENT clima     (ciudad+)                  >
<!ELEMENT ciudad    (nombre, reporte)          >
<!ELEMENT nombre    (#PCDATA)                  >
<!ELEMENT reporte   (alta, baja, precip?)      >
<!ELEMENT alta      (#PCDATA)                  >
<!ELEMENT baja      (#PCDATA)                  >
<!ELEMENT precip    EMPTY                      >
<!ATTLIST precip    total_dia      CDATA       #REQUIRED
     tipo       (lluvia | nieve)    "lluvia"
     fuerza     (ligera | fuerte)   #IMPLIED   >
]>
...
```

Ejemplo completo (XML+DTD).



Ejercicio: Construir una DTD.

- Analizar documento estructurado.
- Disecar los elementos de información.
- Definir la estructura subyacente.
- Crear una representación de la estructura del documento en forma de una DTD.
- Crear un documento XML que siga ese DTD.



Alternativas a las DTDs.

- Para datos generados automáticamente:
  - » Scripts.
  - » Programas.
- Para datos generados por personas:
  - » Formularios.
  - » Scripts de conversión.
  - » Editores restringidos.
  - » “Guías de uso y estilo”.
- Esquemas W3C.
  - » Nueva especificación del W3C.

El procesador XML (*parser*).

- Software que reconoce e interpreta las reglas del XML.
  - » También se le llama analizador o intérprete XML.
- Con XML bien formado:
  - » Revisa que el documento siga las reglas del XML para considerarse bien formado.
- Con XML válido:
  - » Revisa una DTD XML, luego
  - » revisa el documento XML con las reglas XML, luego
  - » revisa el documento XML con las reglas del DTD.

Semántica XML.

Semántica XML.

- El papel de la semántica.
  - » Agrega procesamiento (verbos) al documento XML (sustantivos y adjetivos).
- Semántica de visualización.
  - » Indica cómo debe de formatearse un elemento.
- Semántica de procesamiento.
  - » Indica cómo debe procesarse cada elemento.

El XSL.

Lenguaje extensible de hojas de estilo
(*extensible stylesheets languaje*).

XSL.

- HTML: Formato sin estructura.
  - » Lenguaje de composición (*typesetting*).
  - » No extensible.
- CSS: Formato mejorado, sin estructura.
  - » Lenguaje de hojas de estilo.
  - » Ignora el formato predeterminado del HML, pero no puede modificar la estructura básica del documento.
- XML: Estructura sin formato.
  - » Define los elementos.
  - » Crea una estructura jerárquica de un conjunto de información.

Agregando formato a la estructura.

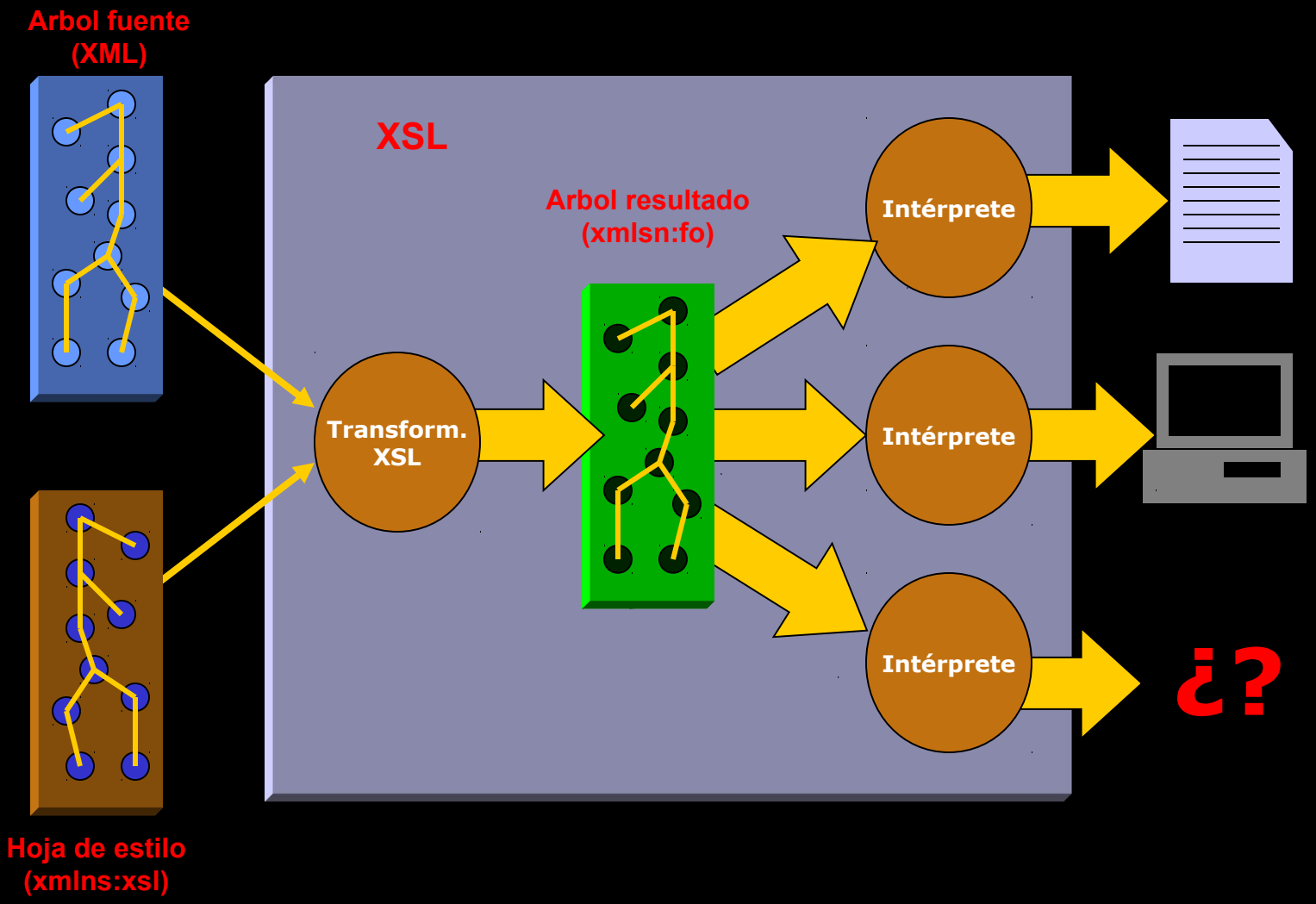
- Lenguaje de hojas de estilo.
- Proporciona definiciones semánticas (verbos o acciones) para los elementos.

- Consiste de dos partes:
  - » Lenguaje para transformar XML.
  - » Vocabulario para especificar semántica de formato.

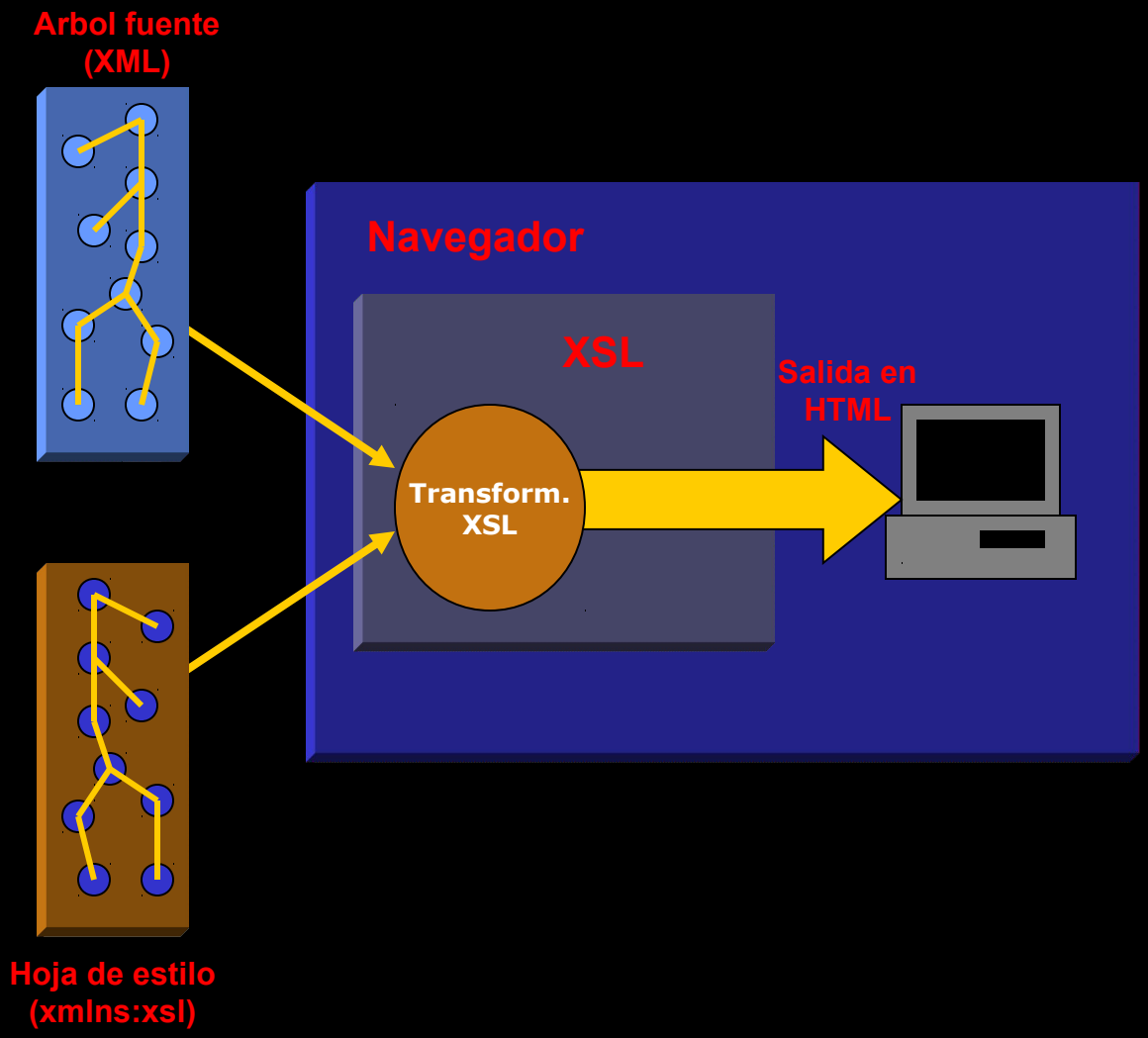
XSL.

- Un lenguaje para expresar hojas de estilo.
- Proporciona semántica de visualización para el XML.
  - » Relaciona elementos XML con HTML o con otros lenguajes de formato (PDF, LaTeX, PostScript, etc).
- Soporte funcional para CSS.
  - » Simple, sintaxis conocida.
  - » Los principiantes pueden aprender rápido.

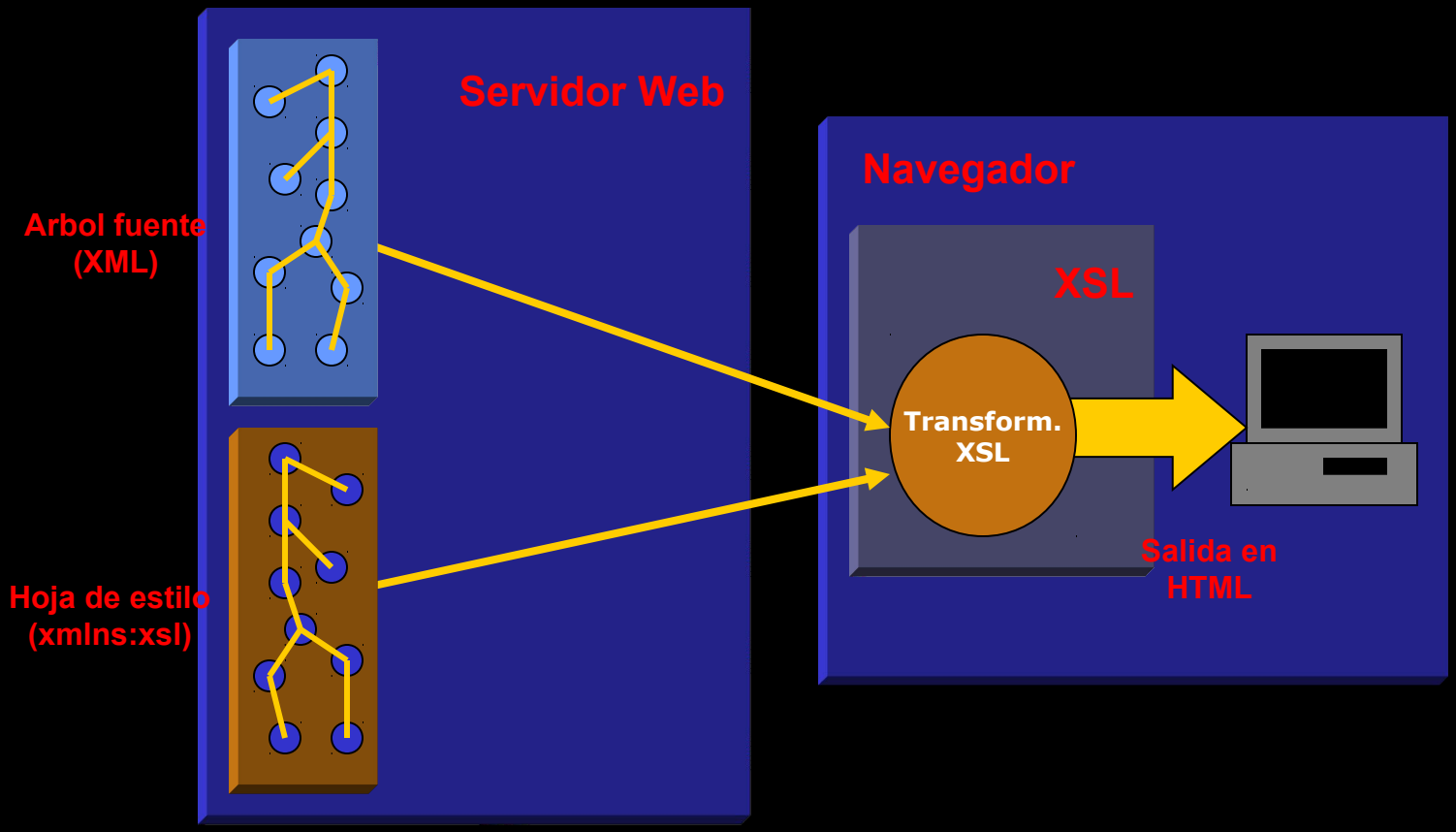
Cómo funciona el XSL.



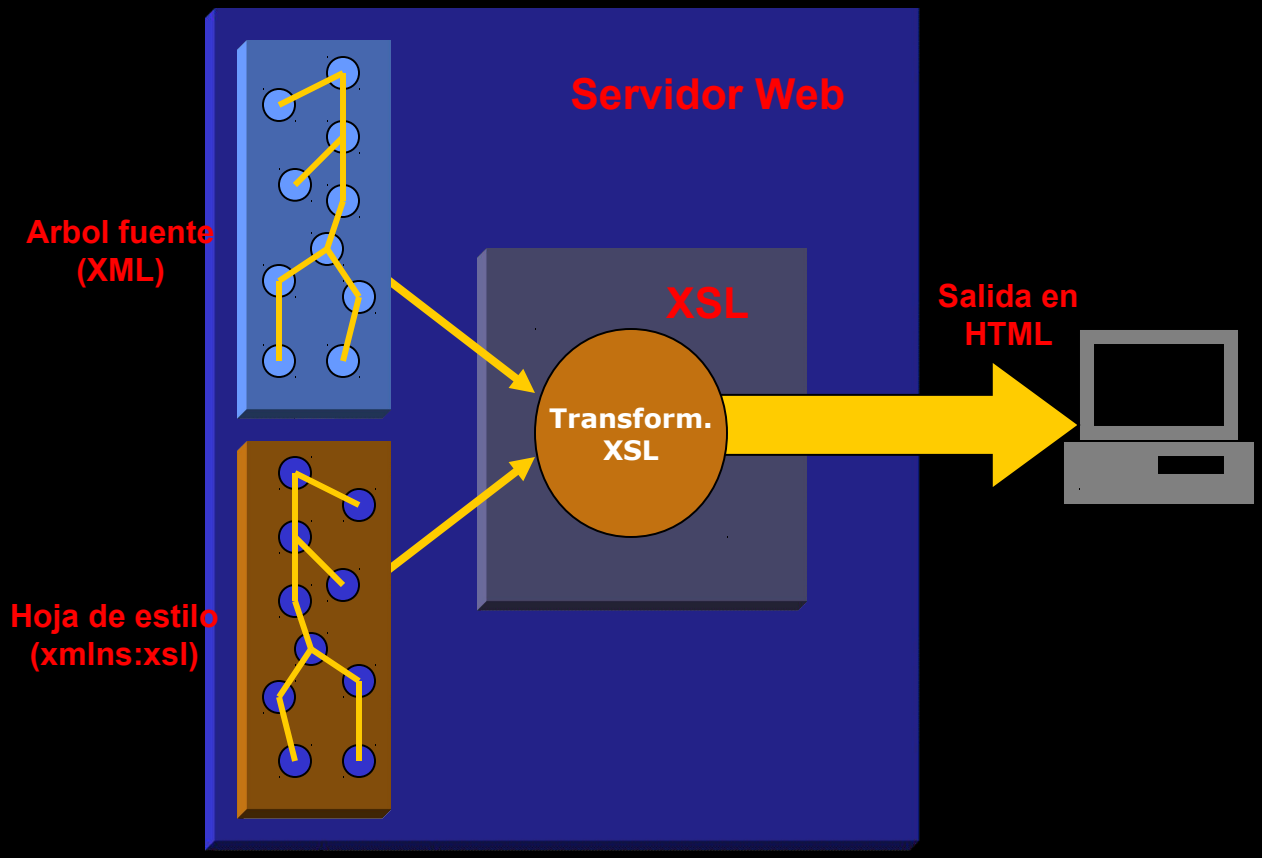
Cómo funciona el XSL en el navegador.



Cómo funciona el XSL en el servidor sin conversion HTML.



Cómo funciona el XSL en el servidor.



XML + XSL

- Un documento XML referencía a un documento XSL por medio de un fragmento de código como este:

```
<?xml-stylesheet type="text/xsl" href="clima.xsl"?>
```

Inicio típico de código XSL.

```
<?xml version="1.0"?>
 <xsl:stylesheet
     xmlns:xsl="http://www.w3.org/TR/WD-xsl"
     xmlns:html="http://www.w3.org/TR/REC-html40"
     result-ns=""
     language="JScript">
 <xsl:template match="/">
```

Plantillas XSL.

- Un documento XSL aplica una o varias plantillas (*templates*) al código fuente XML.
- Un archivo XSL es una secuencia de plantillas que se aplican a una o más etiquetas XML de acuerdo a un patrón.

```
<xsl:template match="/">
  . . .
</xsl:template>
<xsl:template match="clima/ciudad">
  . . .
</xsl:template>
```

coincide con el elemento raíz.

coincide con ciudad, descendiente de clima.

Ejemplo de plantillas XSL.

```
<xsl:template match="/">
  <HTML>
    <HEAD>
      <TITLE>El clima.</TITLE>
    </HEAD>
    <BODY BGCOLOR="White">
      <h1>El clima.</h1>
      <TABLE width="60%" border="1" cellspacing="0"
   cellpadding="5">
        <xsl:apply-templates select="clima/ciudad" order-
   by="+nombre"/>
      </TABLE>
    </BODY>
  </HTML>
</xsl:template>
```

Ejemplo de plantillas XSL.

```
<xsl:template match="clima/ciudad">
  <TR>
    <TD style="font-weight: bold; color: Black; font-family:
    sans-serif;">
      <xsl:apply-templates select="nombre"/>
    </TD>
    <TD style="font-weight: bold; color: Red; font-family: sans-
    serif;">
      <xsl:apply-templates select="reporte"/>
    </TD>
    <TD style="font-weight: bold; color: Blue; font-family:
    sans-serif;">
      <xsl:apply-templates select="reporte/precip"/>
    </TD>
  </TR>
</xsl:template>
```

Plantillas XSL.

<xsl:template match="etiqueta">

- Define el código HTML asociado con una etiqueta XML dada.

<xsl:value-of select="nombre_nodo">

<xsl:value-of select="@nombre_atributo">

- Regresa el texto asociado con el atributo o nodo.

<xsl:for-each select="nombre_nodo">

. . .

</xsl:for-each>

- Repite un proceso para cada elemento con la etiqueta especificada.

Plantillas XSL.

<xsl:apply-templates match="Nombre">

<xsl:apply-templates match="@Atributo">

- Aplica todas las plantillas posibles a todos los elementos que coincidan.

Patrones XSL.

`ciudad`	Elemento.
`clima/ciudad`	Elemento de un ancestro dado.
`precip[@tipo]`	Filtro para atributo.
`precip[@tipo="lluvia"]`	Filtro para atributo.
`.[@total_dia > 0]`	Filtro para nodo actual.

Hay muchas variantes de patrones XSL.

XSL condicional.

- Cuando la generación de HTML depende del valor de algún atributo o elemento hay dos opciones:
  - Estatutos XSL condicionales.
  - Scripts.
    - Extensión de IE, no estándar.

Estatutos condicion XSL **if**.

```
<xsl:if test="condicion">
  . . .
</xsl:eval>
```

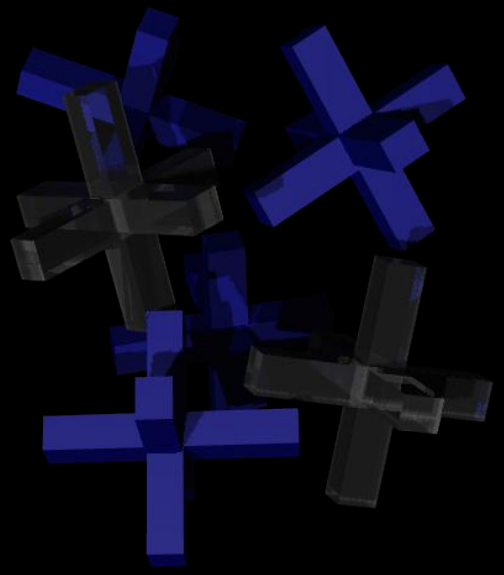
- » Evalua una condición, si el nodo actual retorna un valor, entonces se considera verdadera la condición.

- Ejemplo:

```
<xsl:template match="precip">
  <xsl:if test=".[@total_dia > 0]">
       <xsl:value-of select="@total_dia"/> mm
</xsl:if>
</xsl:template>
```

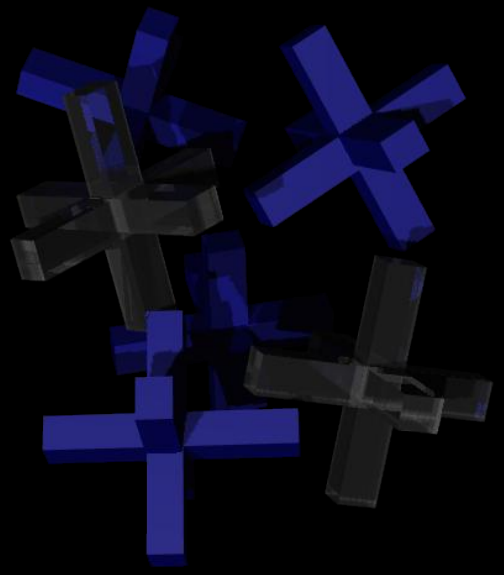
Ejercicio: XSL en el navegador.

- Crear un documento XSL para procesar el XML anterior.
- Visualizar en el navegador.
- Modificarlo para crear salida condicional.



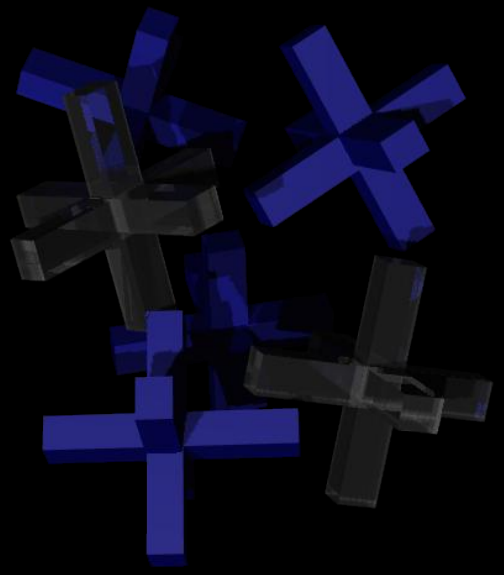
Ejercicio: XSL en el servidor sin conversión HTML.

- Convertir el documento XML en ASP.
- Visualizar en el navegador.



Ejercicio: XSL en el servidor.

- Crear una página ASP para convertir el documento XML en HTML.
- Visualizar en el navegador.



Islas de datos XML.

Islas XML.

- XML dentro de una página HTML.
- Invoca una instancia del procesador XML del cliente.
- Puede ser identificada por un ID.
- Puede controlarse y modificarse con scripts en el cliente.

Ejemplos de islas XML.

```
<html>
<head>
   <title>Islas XML</title>
   <XML id="info-clima">
        <clima><ciudad>
              <nombre>Mexico DF</nombre>
              <reporte>
                 <alta>27</alta><baja>18</baja>
                 <precip total_dia="0" tipo="lluvia"
                                    fuerza="ligera"/>
               </reporte>
        </ciudad></clima>
   </XML>
</head>
<body>
...
```

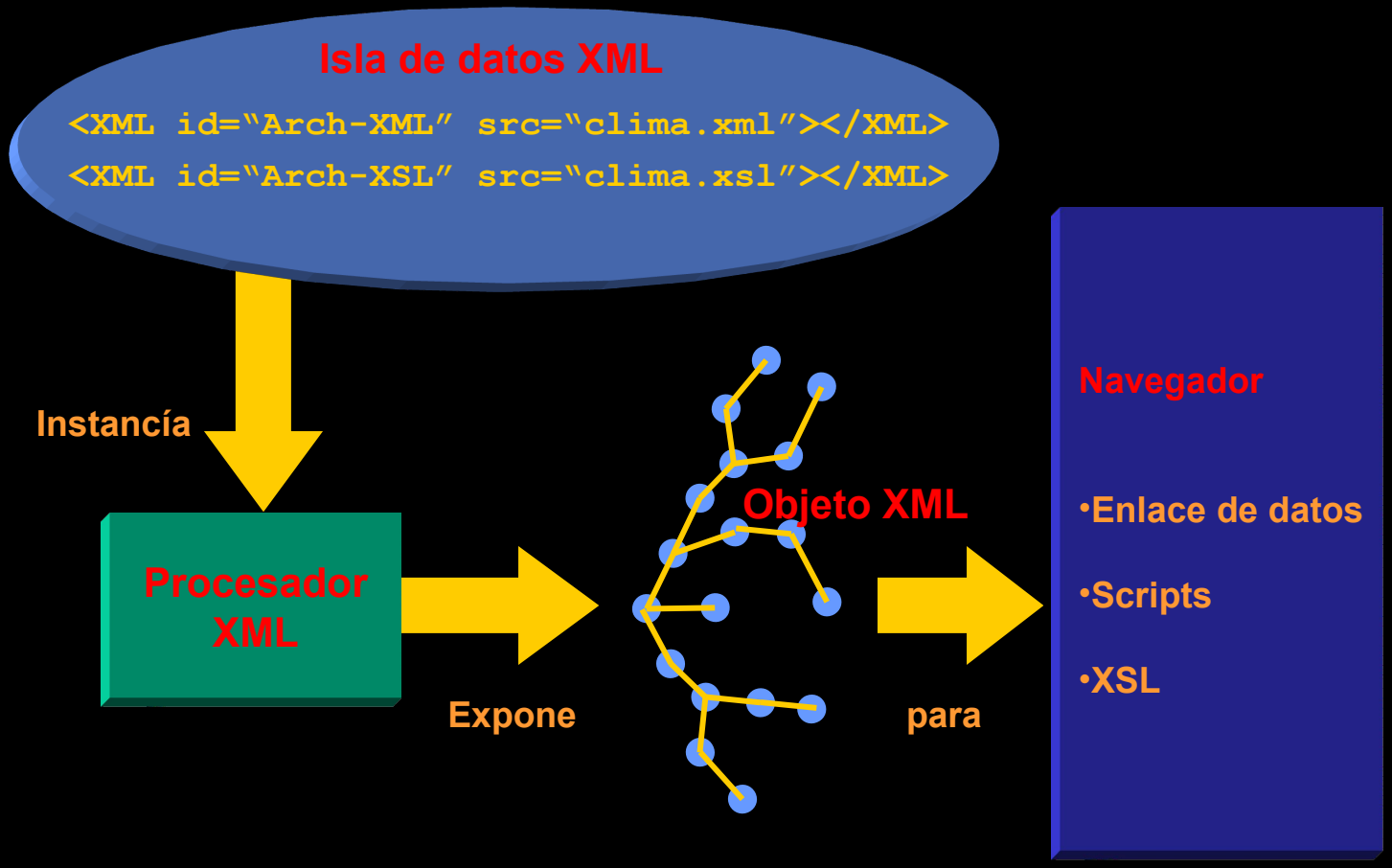
Isla XML

Ejemplos de islas XML.

```
<html>
<head>
   <title>Islas XML</title>
   <XML id="Arch-XML" src="clima.xml"></XML>
   <XML id="Arch-XSL" src="clima.xsl"></XML>
</head>
<body> ...
```

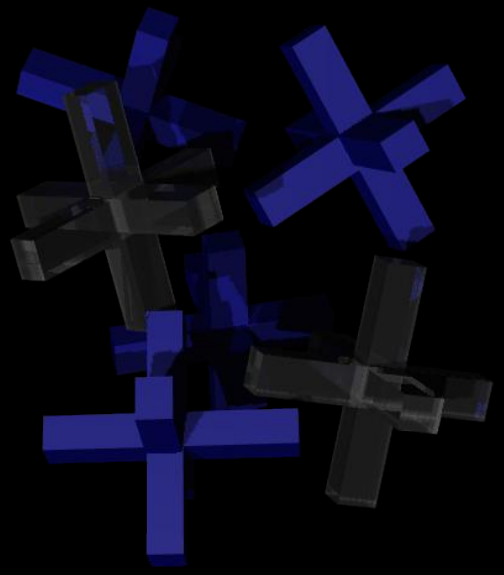
Isla XML

XML en el DOM.


Isla de datos XML
<XML id="Arch-XML" src="clima.xml"></XML>
<XML id="Arch-XSL" src="clima.xsl"></XML>
Instancía
Procesador XML
Expone
Objeto XML
para
Navegador
•Enlace de datos
•Scripts
•XSL

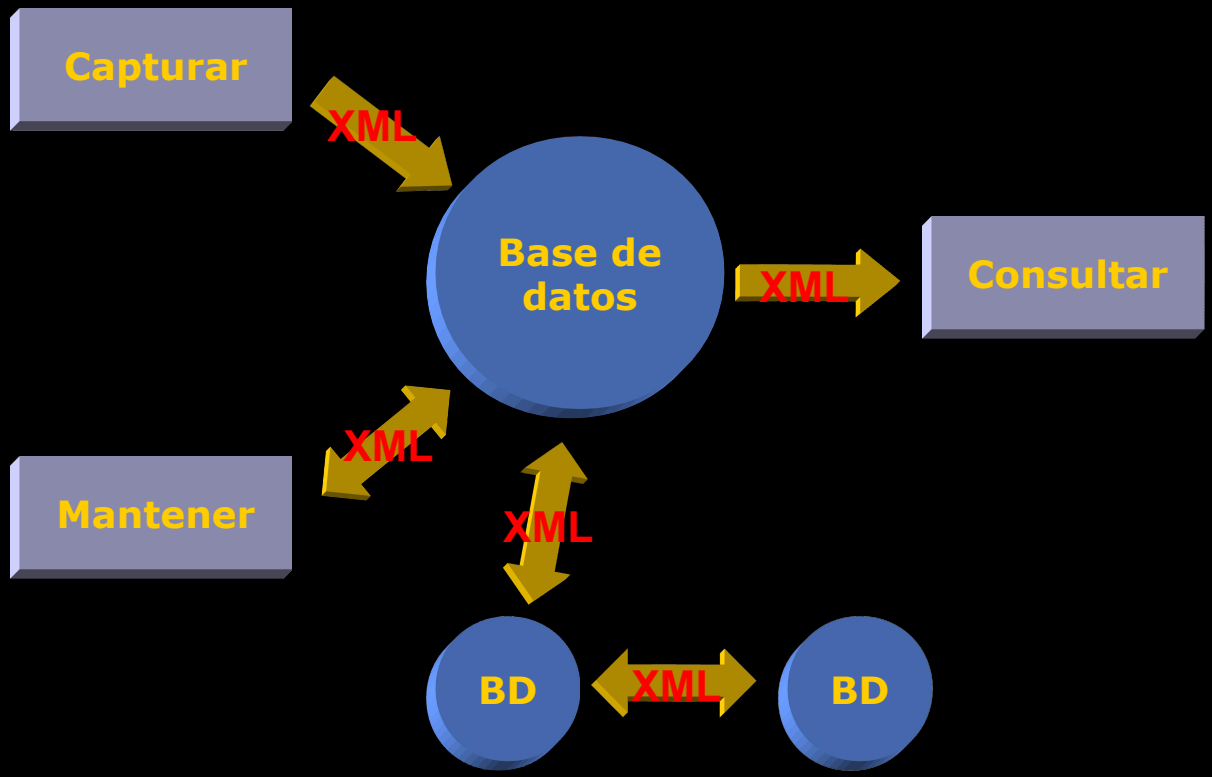
Ejercicio: Islas XML.

- Crear una página HTML con una isla XML.
- Incluir documento XSL.
- Intercambiar entre varios documentos XSL para alterar el estilo de la visualización.



El XML y las bases de datos.

¿Dónde coinciden el XML y las BD?



XML y las BDs.

- Capturar en XML.
  - » Publicar de una fuente XML hacia la BD.
- Consultar en XML.
  - » Crear salidas desde la BD a un formato de presentación (como el HTML).
- Exportar en XML.
  - » Crear vistas lógicas de la base de datos.
- XML como protocolo entre BDs.
  - » Operaciones entre bases de datos usando XML.

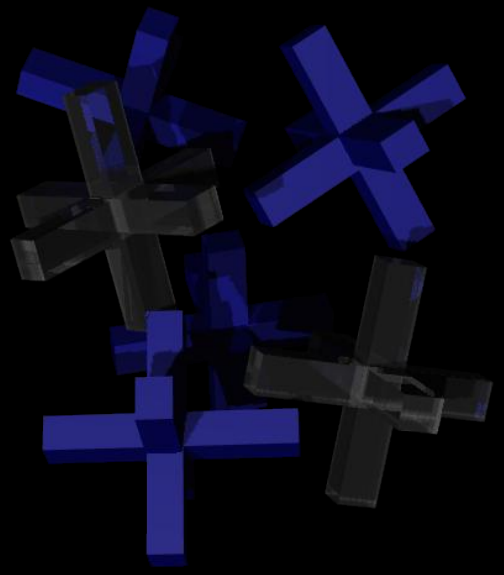
Publicación Web de XML a HTML.

- El XML funciona como BLOB persistente en el sistema de archivos.
  - » Requiere asistencia para encontrar cada documento.
- Almacenar y consultar los documentos XML desde una base de datos.
- Conversión a HTML usando XSL en el servidor.
- Enviar al navegador.

- Ventajas:
  - » XML para manejo de documentos.
  - » HTML para máxima compatibilidad con los navegadores.

Ejercicio: Publicación de XML a HTML.

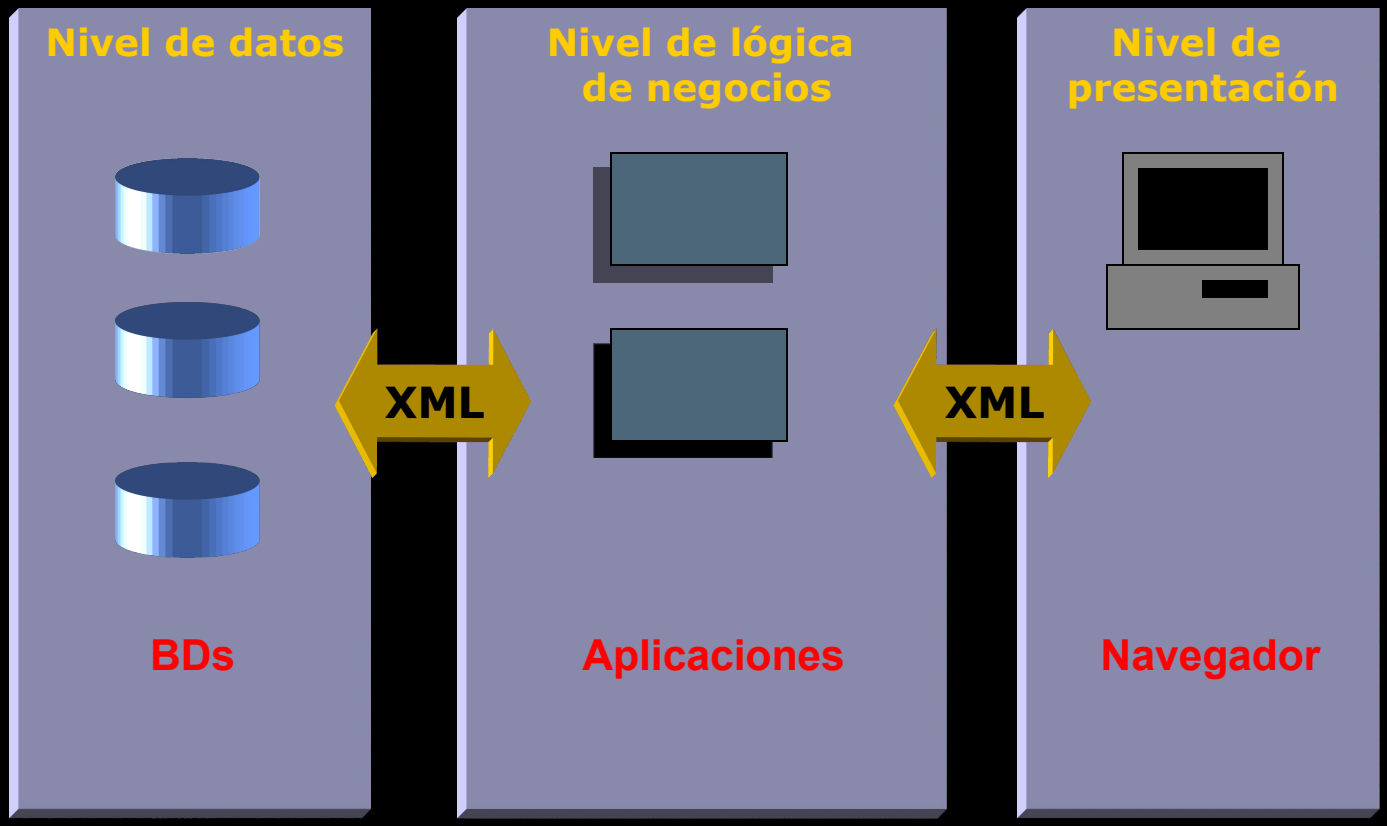
- Crear documentos en XML en archivos independientes y un XSL.
- Crear páginas ASP para mostrar contenido.
- Visualizar en el navegador.
- Crear página ASP para generar índices automáticamente.
- Visualizar en el navegador.



Aplicaciones Web con XML.

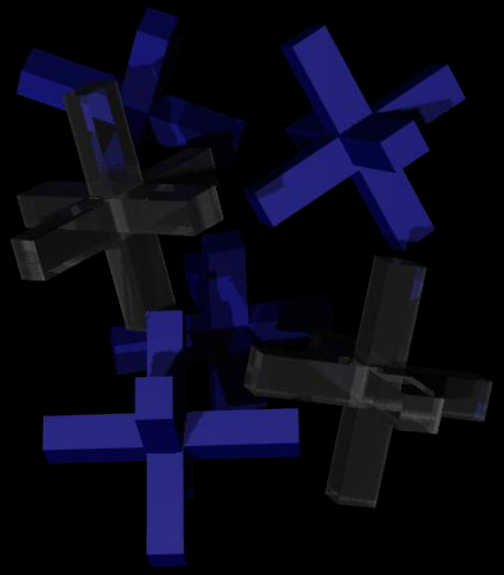
- Modelo de 3 niveles (3-tier).
  - » Nivel de presentación o de usuario.
    - ° Código para presentación.
  - » Nivel de lógica de negocios.
    - ° Código donde se ejecutan las decisiones de la aplicación, se aplican políticas y casi toda la lógica que rige a la aplicación.
  - » Nivel de datos.
    - ° Código para accesar y transformar básicamente el contenido de bases de datos.
- Mejor escalabilidad y flexibilidad.

XML y el modelo de 3 niveles.



Ejercicio: Análisis de “Buscador de Computadoras”.

- Versión modificada del original de Dave Cohen,
  - “Construya su nivel de negocios para comercio electrónico de la manera sencilla con XML, ASP y Scripts”, MIND, enero 2000.



Más información.

- MSDN Latinoamérica,
  - » http://www.microsoft.com/latam/msdn/
- MSDN Web Workshop,
  - » http://msdn.microsoft.com/workshop/
- W3C,
  - » http://www.w3.org/xml/
- Architag,
  - » http://architag.com/xmlu/
- XML en DevX,
  - » http://www.xml-zone.com/

Tutorial de XML.

Mario A. Valdez-Ramírez,
Interactive Bureau México.
Editor de MSDN Latinoamérica.

Partes de esta presentación basadas en la presentación de “Día XML de TechEd 1999” (a su vez con secciones basadas en contenido del X Simposium de XML por Architag.

¿Qué es el XML?

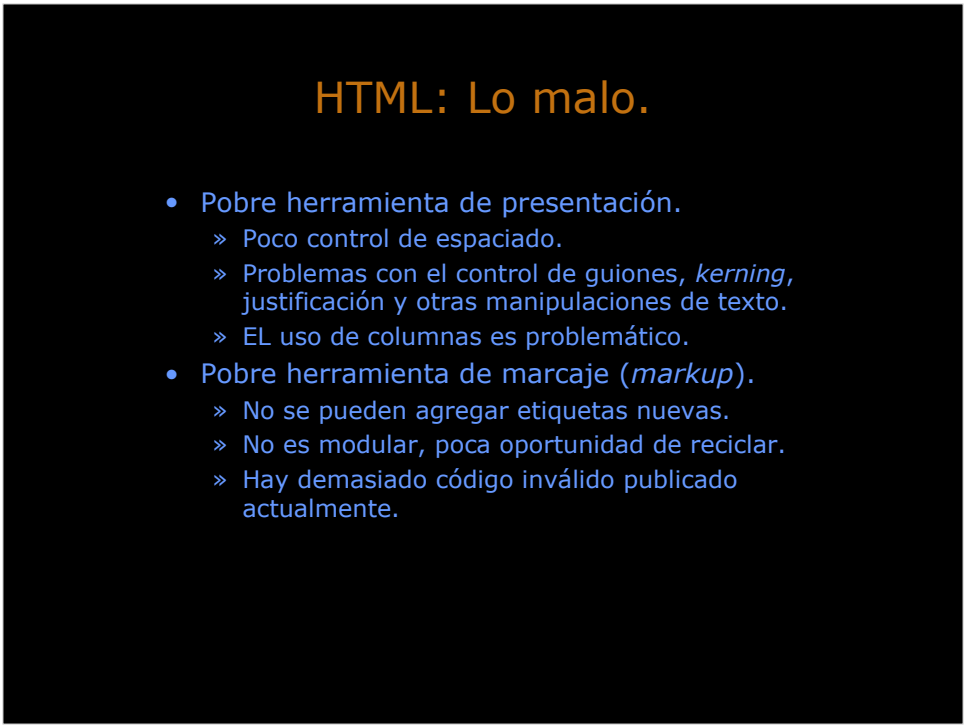
Repasemos lo conocido...

Comparando con...

- HTML (*HyperText Markup Language*).
- SGML (*Standard Generalized Markup Language*).
- PDF (*Portable Document Format*).

HTML: Lo bueno.

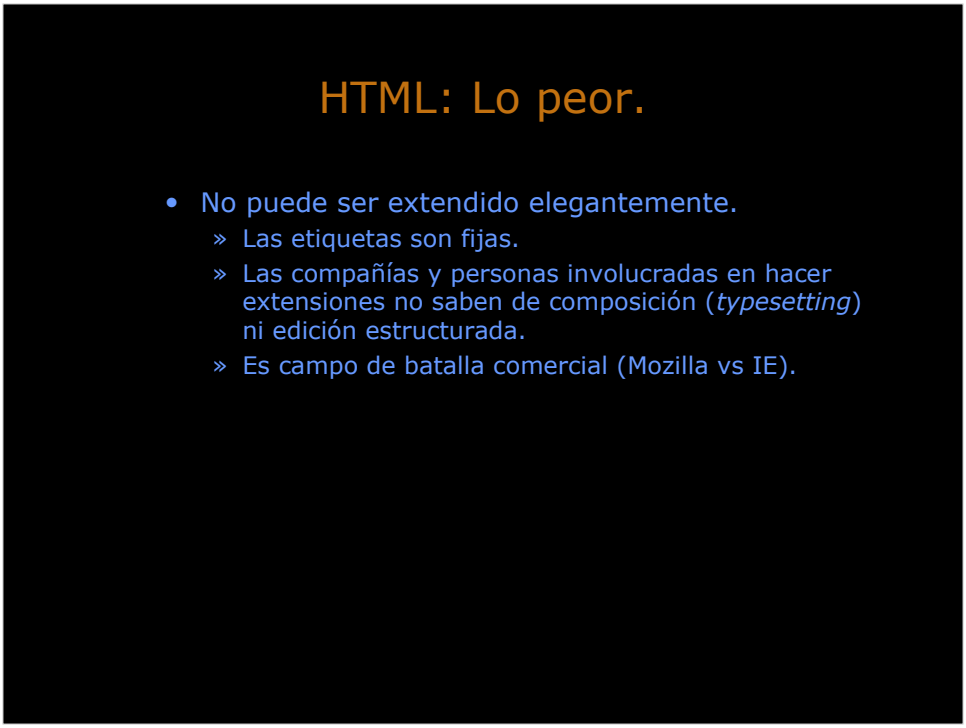
- El hipertexto funciona.
- Es multiplataforma.
- Tiene una curva de aprendizaje muy plana.
- Barato (muchos editores, visores, verificadores, etc., gratuitos).
- Base de información grande.
- Los navegadores son baratos, sencillos de construir y de usar y poderosos.



El código inválido no es culpa del HTML sino de la falta de verificación formal durante su creación y lectura. Veremos como el XML resuelve este problema. Es como cuando usan Basic vs Pascal, no es culpa de Basic que la gente escriba código mal con él, simplemente que favorece la creación de código incorrecto.

Escribir HTML, igual que cualquier trabajo de programación, requiere que se pueda reciclar código de manera limpia y lo más automatizadamente posible. Es posible usar herramientas externas para lograrlo con HTML, pero el XML (como veremos más adelante) lo facilita.

Es importante recordar que cuando se creo el HTML no se hizo como un mecanismo de diseño gráfico, sino como una herramienta de presentación de hipertexto, es decir solo describía el formato básico del texto y qué elementos del texto estaban relacionados (por hiperenlaces) con otros texto. Su propósito era describir el hipertexto de manera independiente de la plataforma.



No es un conjunto de reglas.

El HTML ya no es adecuado para los editores, ni los diseñadores ni para nadie. Es una mezcla bizarra que se está tratando de limpiar con el HTML 4.0 y las CSS.

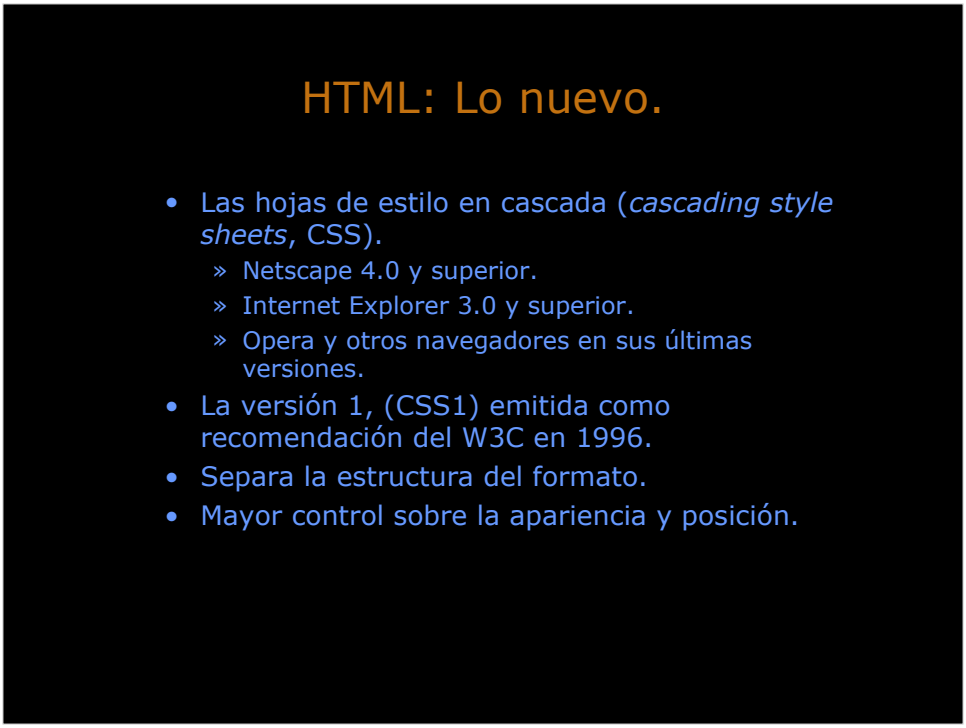
El estándar es usado como un club.

HTML: Los problemas.

- La principal queja es:

  "No puedo hacer [x cosa]".

- Las razones:
  - » Es solo un conjunto de etiquetas.
  - » Pocos usan el DTD como guía.
  - » Los navegadores siempre tienen que lidiar con código mal escrito.



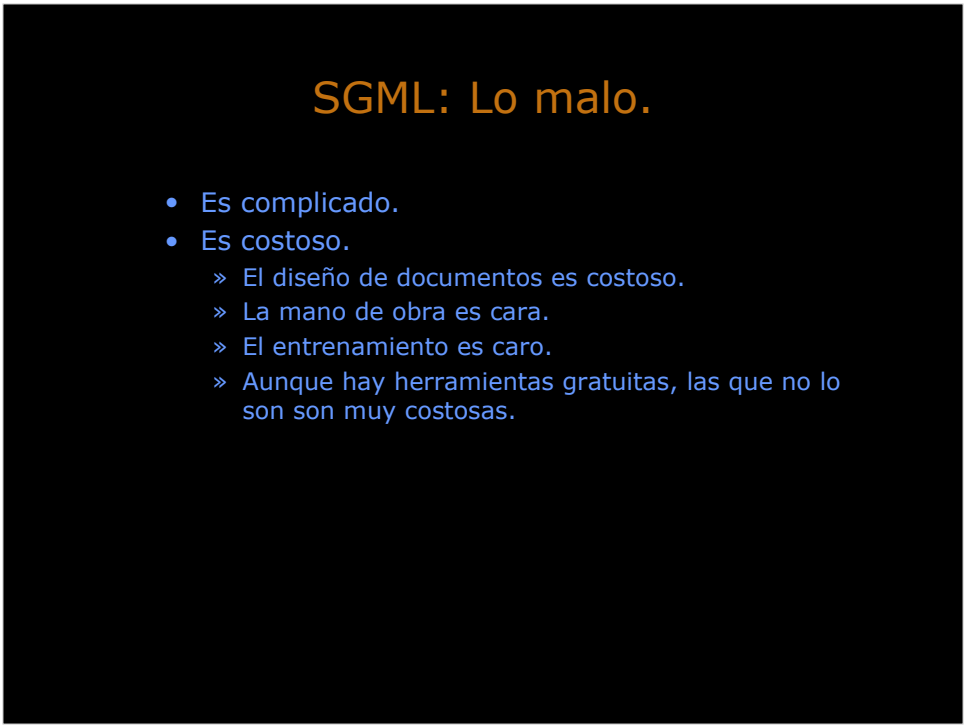
Las CSS permiten hacer manipulaciones de elementos HTML que no se podían hacer antes.

Además, permite centralizar el formato y apariencia de un sitio completo, por ejemplo, al modificar una sola CSS se altera la apariencia de todo el sitio, sin tener que editar cada pagina individualmente.

Requiere soporte de parte del navegador.

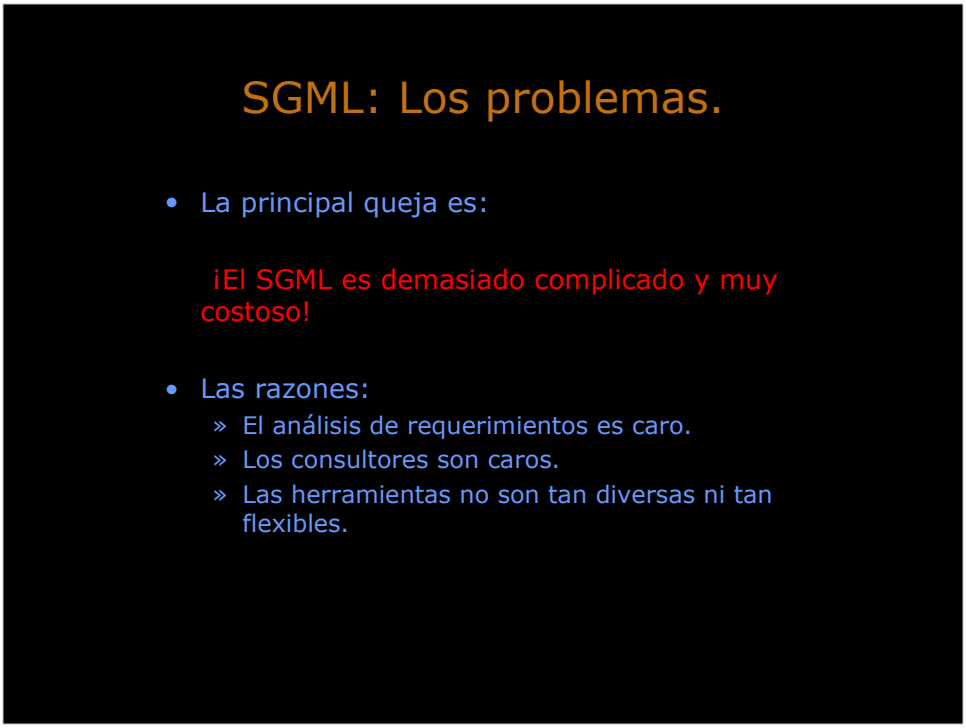
SGML: Lo bueno.

- Es multiplataforma.
- Es un estándar ISO estable.
- Hay disponibles muchas herramientas gratuitas para edición y conversión.
- Es un conjunto de reglas, no un conjunto de etiquetas fijas.
- Separa completamente la estructura del formato.



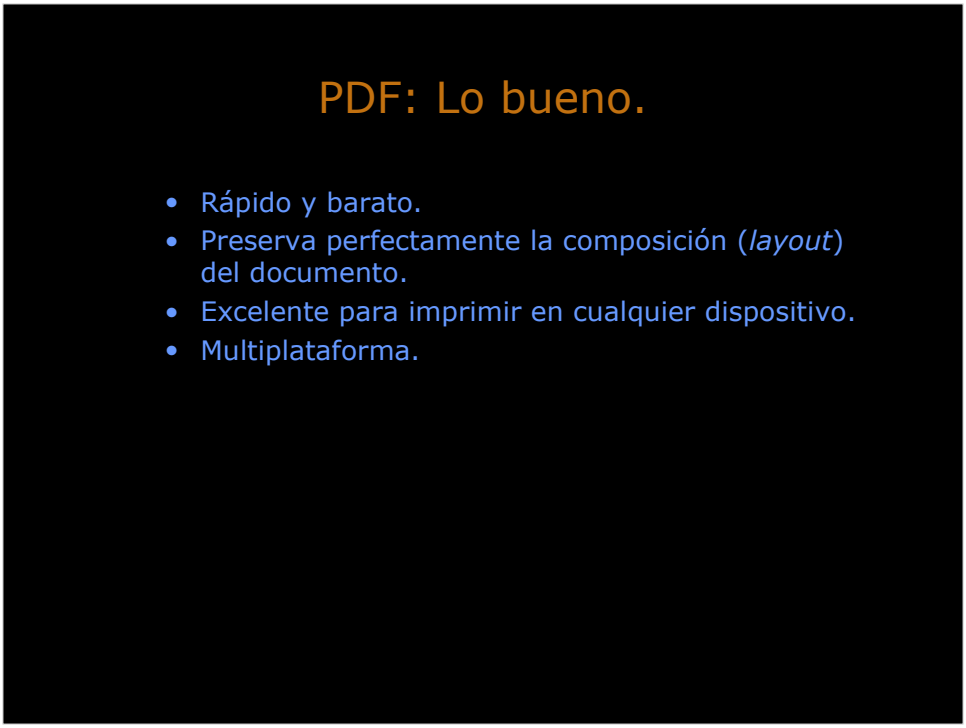
Lo complicado es relativamente con el HTML.

EL entrenamiento es caro en términos de tiempo y/o dinero.



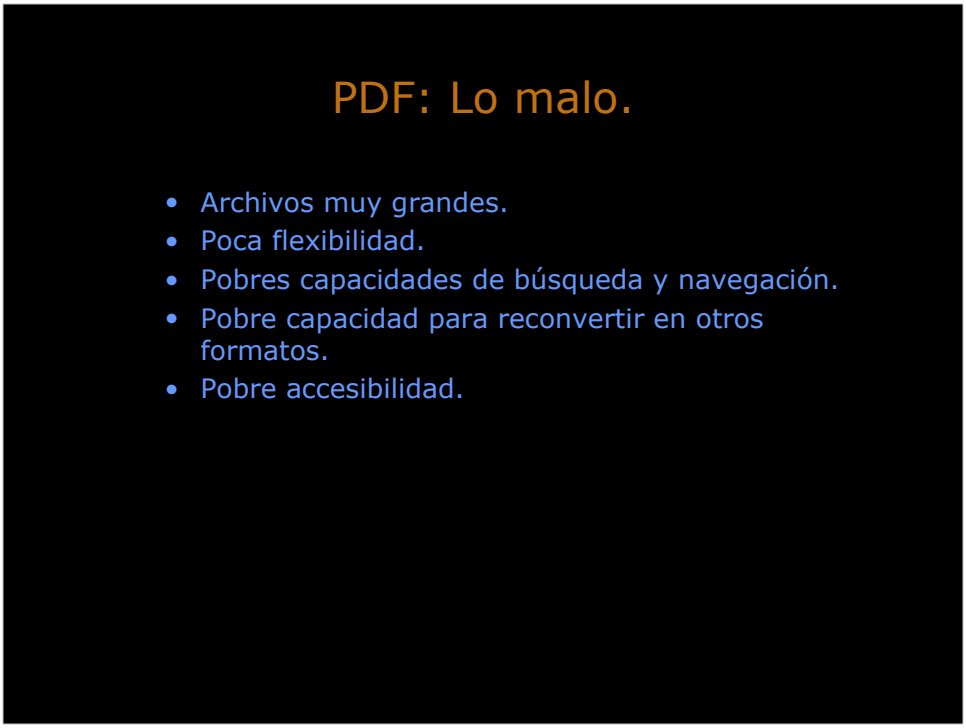
De nuevo, no cualquiera aprende SGML ni lo hace en un tiempo muy corto.

Si una persona o compañía desea convertir sus documentos usando SGML tardará mucho más y le constará mucho más que usando HTML.



Adobe ha hecho una gran trabajo mejorando la especificación PDF, hijo de PS, y haciéndola abierta. Es decir, ya no es una especificación propietaria.

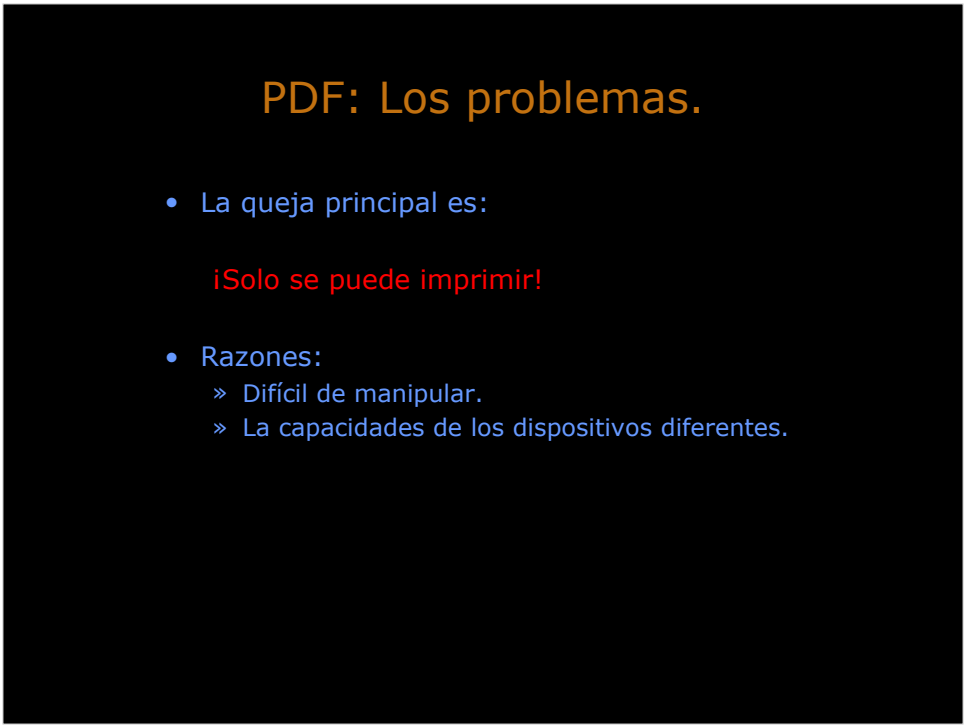
Actualmente hay muchas herramientas para convertir texto en PDF, en varias plataformas.



El precio que pagan los PDF por su excelente conservación de la composición es la falta de flexibilidad.

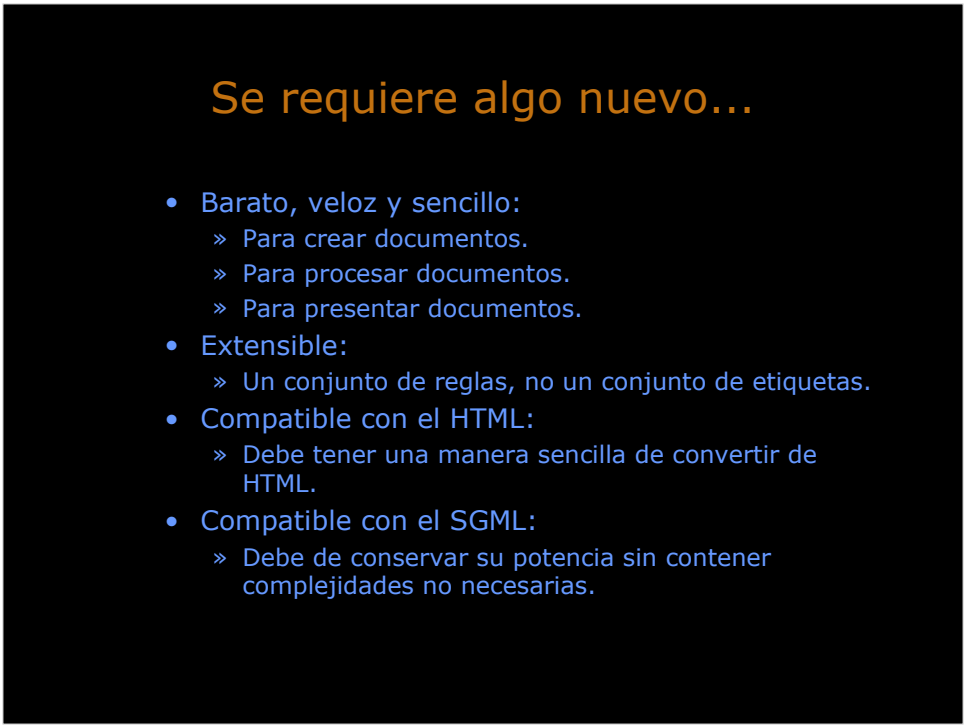
No siempre es posible presentar el mismo documento en diferentes dispositivos con características muy distintas.

Sobre la accesibilidad, me refiero a la capacidad de extrar el texto para convertirlo a otras formas como Braille o voz, para personas discapacitadas.



Sabemos que imprimir no es lo único que se puede hacer, pero generalmente los visores están optimizados para eso, no para “navegar”.

No es lo mismo querer imprimir en una impresora de 1200x1200 dpi a color que un la pantalla LCD de una Palm Pilot.



Extensible de una manera estándar, es decir, sin oportunidad de que una modificación lo convierta en incompatible. La solución es crear reglas, no etiquetas.

¿Acaso el SGML tiene complejidades no necesarias? No, el SGML es complejo porque es el “padre” de todos los lenguajes de marcaje (markup) incluyendo al XML, pero nuestra nueva alternativa no tiene que ser tan extenso.


Necesitamos XML.

Metas de diseño.

- XML debe ser utilizable a través de Internet.
- XML debe soportar muchos **escenarios** de aplicación.
- XML debe ser compatible con el SGML.
- Los programas que procesen documentos XML deben ser fáciles de crear.
- Las características opcionales deben ser idealmente cero.
- Lso documentos en XML deben de ser legibles por humanos y razonablemente claros.

Metas de diseño.

- El diseño con XML debe ser rápido.
- El diseño de documentos XML debe de ser formal y conciso.
- Los documentos XML deben de ser fáciles de crear.
- El laconismo en el uso de etiquetas no es importante.

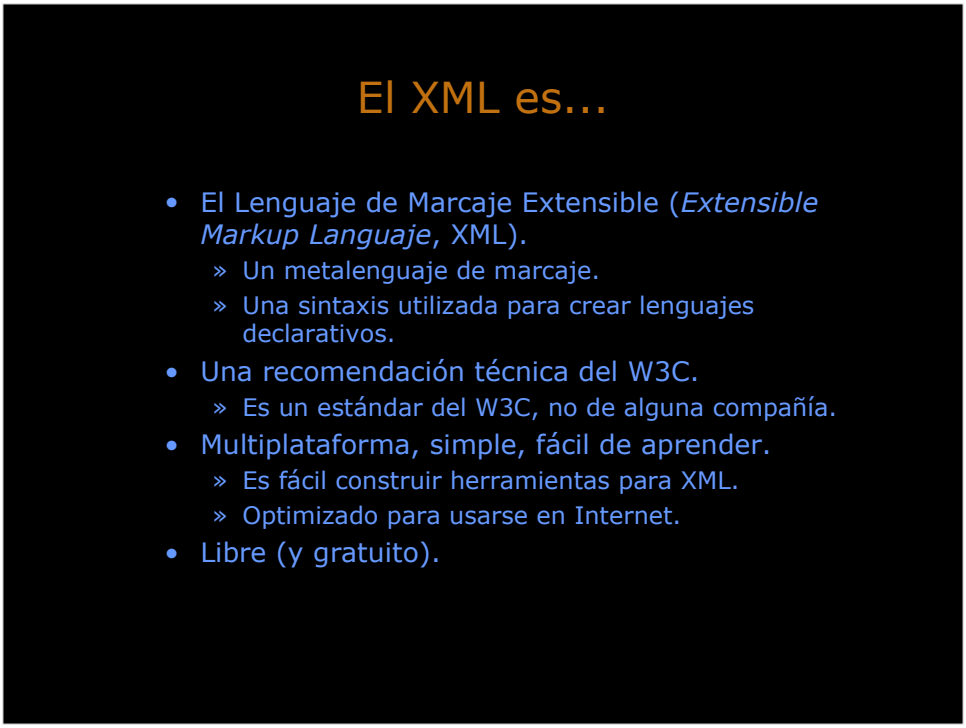
Ahora... un ejemplo.

```
<Cliente ID="HVet950283">
   <Nombre>Hospital Veterinario Kermit</Nombre>
   <Direccion verificada="si">
        <Calle>Padre Mier 1528</Calle>
        <Ciudad>Monterrey</Ciudad>
        <Estado>NL</Estado>
        <CodigoPostal>64000</CodigoPostal>
   </Direccion>
</Cliente>
```

Sintaxis simple

Legible por personas

Muy parecido al HTML



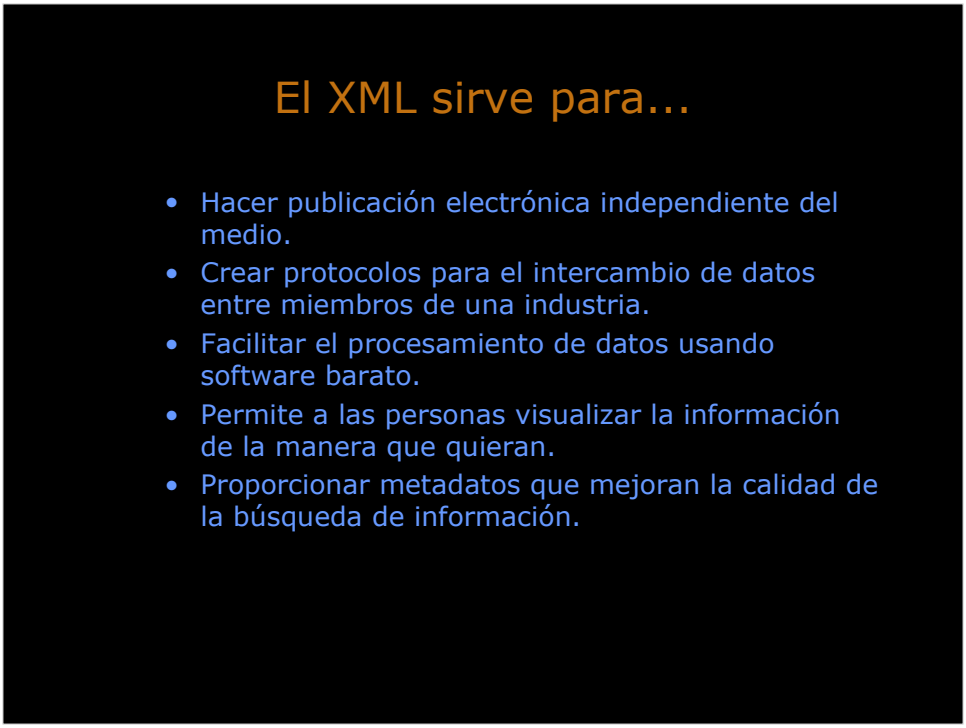
Es un metalenguaje, no especifica el marcaje (las etiquetas) sino las reglas que gobiernan a las etiquetas.

No define lenguajes imperativos (como la mayoría de los lenguajes de programación) sino declarativos, trata de lenguajes que dicen “cómo” son las cosas, no qué hacen.

Si es fácil construir herrmientas para procesar XML entonces las herramientas serán muy baratas. Algunas serán libres.

El XML **no** es...

- Un lenguaje de marcaje (*markup*).
  - » No. Es un estándar que especifica una sintaxis para crear lenguajes de marcaje.
- Solo para Web.
  - » No. Puede ser usado para describir y comunicar cualquier información estructurada.
- Un superconjunto del HTML.
  - » No. Aunque el HTML puede ser definido con sintaxis de XML.
- Un invento de [x compañía].
  - » No. XML es un estándar creado por el W3C y soportado por compañías e instituciones de todo el mundo.



Quizás no sea tan obvio, pero además de hacer posible la publicación sin importar le medio (periódico, revista, página Web, generación de voz, etc.), el XML permite a miembors de una industria hacer intercambio de datos usando DTDs o esquemas comunes. También se puede usar para mejorar la calidad de las búsquedas de informació, porque ahora los documentos dan **significado** a cada una de sus partes; es decir proporciona información de su propia información (metdatos), ya no son más un depósito de texto e imágenes. Otro efecto colateral de que el XML sea tan simple es que el software de procesamiento es muy simple de crear y por tanto barato.

Dos versiones.

- XML bien formado.
  - » Las etiquetas de inicio y final coinciden.
  - » Los elementos vacíos tienen una forma especial.
  - » No hay elementos traslapados.
  - » Los atributos van en comillas.
- XML válido.
  - » Es código bien formado con funciones adicionales.
  - » Se adhiere a una estructura predefinida dictada por un esquema,
    - ° DTD, DCD, SOX, etc.

DTD =

DCD =

SOX =

Sintaxis del XML.

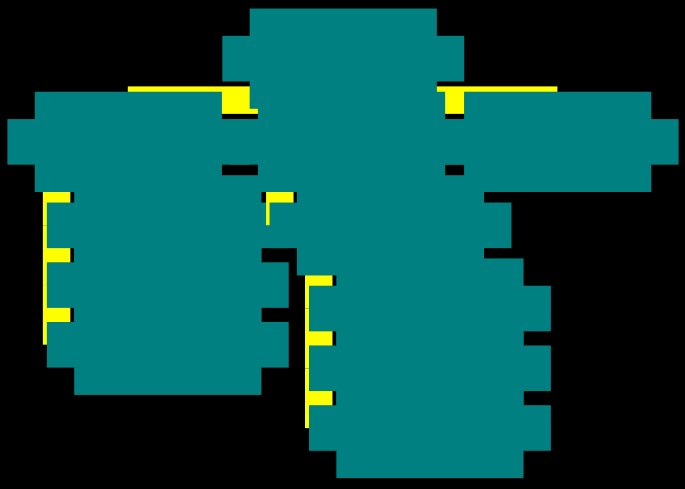

El documento XML
bien formado.

Un documento XML es...

- Una colección de piezas llamadas “entidades”.
- Texto y etiquetas en Unicode.
- Válido, o por lo menos bien formado.

- Representa una jerarquía de datos.

Jerarquía de datos.



Contenedor = elemento.

- Declarativo (sustantivo).
- Lo que está encerrado entre las etiquetas.
- De lo que habla la sintaxis.
- Cinco cosas necesarias:
  - » Cómo se llama el elemento.
  - » Dónde inicia el elemento.
  - » Dónde termina el elemento.
  - » Qué contiene el elemento.
  - » Qué relación tiene el elemento con otros elementos.

Creando documentos bien formados.

- Un único elemento raíz.
- Los elementos en la raíz aparecen secuencialmente o anidados.
- Los elementos no se deben traslapar.
- Todo elemento tiene una etiqueta de inicio y una de final.
  - » Inicia con `<Nombre_elemento>`
  - » Termina con `</Nombre_elemento>`
  - » Los elementos vacios inician y terminan con `<Nombre_elemento/>`

Etiquetas.

- El XML diferencía entre mayúsculas y minúsculas.
  - » `<Libro>`, `<libro>`, `<LIBRO>` y `<LiBrO>` son etiquetas que se refieren a diferentes elementos.
- Los nombres de elementos:
  - » Deben de iniciar con una letra, subrayado o dos puntos (:).
  - » Los caracteres siguientes pueden ser letras, números, puntos, guiones, subrayados o dos puntos.
  - » El nombre "XML" y sus variaciones están reservadas.

La declaración XML.

- Dice "¡Soy un documento XML!".
- Tiene partes específicas:

<?xml	apertura
version="1.0"	versión
encoding=""	codificación de caracteres
standalone=""	doc. independ. (yes/no)
?>	fin

- Cada entidad XML que no esté en UTF-8 o UTF-16 debe contener la declaración de codificación (*encoding*).

Ejemplos de declaraciones XML.

(ninguna)

<?xml version="1.0"?>

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

<?xml version="1.0" standalone="yes"?>

<?xml version="1.0" encoding="ASCII"
standalone="no"?>

Ejemplo bien formado.

<Bienvenida>¡Hola mundo!</Bienvenida>

Ejemplo bien formado.

```
<?xml version="1.0"?>
<Configuracion>
        <Impresora>
                <Nombre>HP LaserJet 5SI</Nombre>
        <Controlador>hplj5si.dll</Controlador>
        <Sitio>\\mvaldez\HP5SI</Sitio>
        <Opciones>
                <AlimentadorSobres/>
                <Scanner/>
        </Opciones>
   </Impresora>
</configuracion>
```

Documento mal formado.

```
<?xml version=1.0?>
<Configuracion>
        <Impresora>
                <Nombre>HP LaserJet 5SI
        <Controlador>hplj5si.dll</Nombre>
                </Controlador>
        <Sitio>\\mvaldez\HP5SI</Sitio>
        <Opciones>
                <AlimentadorSobres>
                <Scanner/>
        </Opciones
   </Impresora>
</configuracion>
</Configuracion>
```

Entidades carácter.

- Para documentos bien formados:
  - > `>` (greater than)
  - < `<` (less than)
  - & `&` (ampersand)
  - ’ `&apos;` (apóstrofe)
  - ” `&quot;` (double quote)
- Los documentos válidos deben de definir estas entidades antes de usarlas.
- Ejemplos:
  - `AT&T`
  - `Nombre=“Mario Moreno &apos;Cantinflas&apos;”`

Atributos.

- Propiedades (adjetivos).
- Contienen información acerca del elemento.
  - » Información sobre gráficos.
  - » Fechas, nombres, colores, etc.
- Aparecen en la etiqueta de inicio:

```
<Nombre_elemento Nombre_atributo="valor">
```

ó

```
<Nombre_elemento Nombre_atributo='valor'>
```

Ejemplo con atributos.

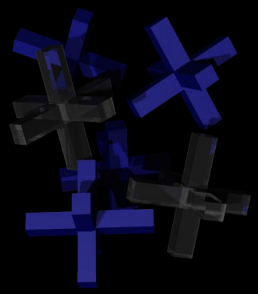
```
<?xml version="1.0"?>
<Configuracion>
        <Impresora local="si">
                <Nombre>HP LaserJet 5SI</Nombre>
        <Controlador Instalado="si">hplj5si.dll</Controlador>
        <Sitio>\\mvaldez\HP5SI</Sitio>
        <Opciones>
                <AlimentadorSobres/>
                <Scanner/>
                <Color Colores="256"/>
        </Opciones>
   </Impresora>
</Configuracion>
```

¿Qué editor puedo usar?

- Requisitos mínimos:
  - » No debe generar caracteres EOF (Ctrl-Z) al final del archivo.
  - » No debe generar tabulaciones (si se usan deben de expandirse a espacios al grabar).
- Sugerencias:
  - » Cualquier editor de texto o procesador de palabras.
  - » Editores especiales para XML.
    - ° Variantes de editores de SGML.
    - ° Editores de XML.
  - » Editores de SGML.
    - ° Requiere hacer algunos ajustes, no recomendable.

Ejercicio: Construir un documento bien formado.

- Cargar el editor.
  - » Architag Real-time XML Editor.
    - ° http://www.architag.com/editor/
  - » Microsoft XML Notepad.
    - ° http://msdn.microsoft.com/xml/notepad/
- Crear el documento.
- Visualizar en el navegador.



La necesidad de guías de uso y estilo.

- Si queremos que todos en nuestro proceso (personas y máquinas) usen las mismas etiquetas de la misma manera, entonces requerimos...
- Guías de uso y estilo para las estructuras de datos.
  - » Listas de etiquetas válidas.
  - » Relaciones entre etiquetas.
  - » Valores predeterminados para los atributos.
  - » Tipos de datos que no sean XML estandarizados.

Guías de uso y estilo.

- Nada de esto es nuevo.
  - » Las guías de uso y estilo se han usado por décadas para intercambiar información entre personas.
- Pero el XML no es solo para personas.
  - » Se requieren guías legibles por máquinas.

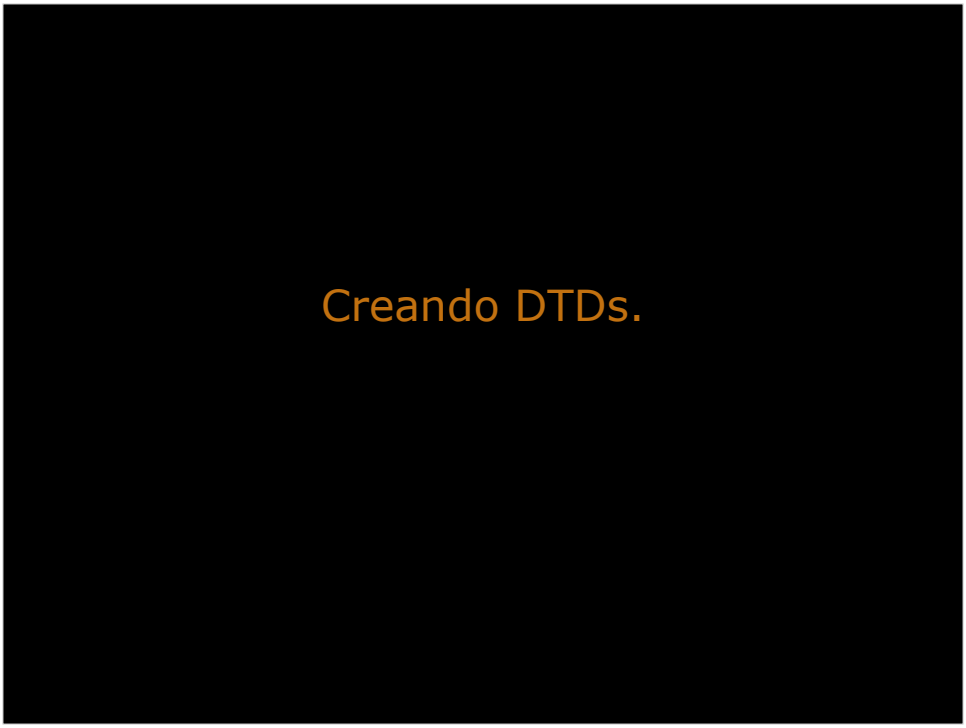

XML válido.

Reglas: La DTD del XML.

- Una Definición de Tipo de Documento (Document Type Definition, DTD) permite:
  - » Definir un conjunto específico de etiquetas con relaciones específicas.
  - » Definir valores predeterminados para los atributos.
  - » Definir entidades de texto y binarias adicionales junto con sus notaciones.
  - » Indicar el elemento raíz.

El control de la DTD.

- La DTD proporciona:
  - » Una sintaxis formal que sirva de guía a un intérprete/analizador (*parser*).
  - » La habilidad de definir valores predeterminados para los atributos.
  - » Especificaciones para la estructura.
- Una DTD es una buena manera (pero no la única) de controlar la creación de datos.


Creando DTDs.

Declaración de elemento **ELEMENT**.

`<!ELEMENT`	Apertura y palabra clave.
`Nombre_elemento`	Nombre del elemento.
`( . . . )`	Modelo de contenido o
`PALABRACLAVE`	contenido declarado.
`>`	Cierre.

Palabras clave para el contenido declarado:

`EMPTY`	Sin elemento o contenido.
`ANY`	Cualquier combinación de elementos descendientes y datos caracter.

Modelo de contenido.

- Elementos o #PCDATA.
- Conectores.

,	seguido de	`(a,b)`
|	uno u otro	`(a|b)`

- Indicadores de ocurrencia.

	Uno y solo uno	`configuracion`
?	Cero o uno	`Nombre?`
+	Uno o más	`Controlador+`
*	Cero o más	`Opciones*`

Ejemplos de modelos de contenido.

```
(Titulo, Seccion+)

(Titulo, (Parrafo+ | Seccion+))

(Titulo, (Parrafo | Seccion)+)

(Nombre, Numero, (Articulo, (Cantidad | Lote),
   Descripcion, precio)+, Descuento*)


<!ELEMENT Capitulo (Titulo, Seccion+)>
```

Contenido mixto.

- Caracteres (#PCDATA) que aparecen solos o en combinación con elementos descendientes en un modelo de contenido.
- Pueden ser expresados en combinaciones como un grupo o un contenido repetible:

```
(#PCDATA | grafico | tabla | lista)
```

- El mismo elemento descendiente no puede aparecer más de una vez en el grupo.

```
<!ELEMENT parrafo (#PCDATA | lista)*>
```

Comentarios XML.

- Los comentarios pueden aparecer en cualquier parte del documento fuera de otros marcajes.
- Pueden aparecer dentro de la declaración de tipo de documento.
- Un procesador XML puede, pero no requiere, ser capaz de leer y recuperar los comentarios.

```
<!-- Articulos secundarios para BD,
                revisado el 2000/I/29 -->
```

Ejemplos de declaraciones de elementos.

```
<!--    Nombre          Modelo contenido              -->
<!ELEMENT       clima   (ciudad+)                     >
<!ELEMENT       ciudad  (nombre, reporte)             >
<!ELEMENT       nombre  (#PCDATA)                     >
<!ELEMENT       reporte (alta, baja, precip?)         >
<!ELEMENT       alta    (#PCDATA)                     >
<!ELEMENT       baja    (#PCDATA)                     >
<!ELEMENT       precip  EMPTY                         >
```

Declaración de atributos **ATTLIST**.

`<!ATTLIST`	Apertura y palabra clave.
`Nombre_elemento`	Nombre del elemento.
`Nombre_atributo`	Nombre del atributo.
`( . . . )`	Lista de valores o
`PALABRACLAVE`	valor declarado.
`" . . . "`	Valor predeterminado o
`#PALABRACLAVE`	palabra clave de valor predeterminado.
`>`	Cierre.

Ejemplos de declaracions de atributos.

```
<!ELEMENT Novela       (titulo, parrafo+)>
<!ATTLIST Novela
   Copyright           CDATA          #REQUIRED
   PalabraClave        CDATA          #IMPLED
        type    (original|revisada|adaptada) "original"
   Estante             CDATA          #REQUIRED>
...
<Novela Copyright="1998 Ed. Diana" Estante="i1022">
...
</Novela>
```

Ejemplos de elementos con atributos.

```
<!--    Nombre          Modelo contenido                --->
<!ELEMENT       clima   (ciudad+)                        >
<!ELEMENT       ciudad  (nombre, reporte)                >
<!ELEMENT       nombre  (#PCDATA)                        >
<!ELEMENT       reporte (alta, baja, precip?)            >
<!ELEMENT       alta    (#PCDATA)                        >
<!ELEMENT       baja    (#PCDATA)                        >
<!ELEMENT       precip  EMPTY                            >
<!ATTLIST       precip  total_dia       CDATA   #REQUIRED
   tipo         (lluvia | nieve)        "lluvia"
   fuerza       (ligera | fuerte)       #IMPLIED        >
```

Declaracion de documento **DOCTYPE**.

<!DOCTYPE	Apertura y palabra clave.
Elemento_raiz	Nombre del elemento raíz.
PALABRACLAVE	SYSTEM o PUBLIC y
“dtd.dtd”	una DTD XML externa o
[. . .]	declaraciones internas.
>	Cierre.

Ejemplos de declaraciones de documento.

(ninguna)

```
<!DOCTYPE novela [
<!ELEMENT      novela  (titulo, parrafo+)          >
<!ELEMENT      titulo  (#PCDATA)                   >
<!ELEMENT      parrafo (#PCDATA)                   >
]>
```

```
<!DOCTYPE novela      SYSTEM  "novela.dtd"   >
```

Mitos de las DTDs de XML.

- El DTD clarifica el significado del documento.
  - » No necesariamente. La DTD solo especifica el orden de los elementos de un documento, no su significado.
- Es posible intercambiar información ciegamente usando una DTD.
  - » No. La DTD sirve para asegurarse de que todos los involucrados usan la misma estructura.

Ejemplo completo (XML+DTD).

```
<?xml version="1.0"?>
<!DOCTYPE clima [
<!--       Nombre     Modelo contenido     -->
<!ELEMENT clima      (ciudad+)                >
<!ELEMENT ciudad     (nombre, reporte)        >
<!ELEMENT nombre     (#PCDATA)                >
<!ELEMENT reporte    (alta, baja, precip?)    >
<!ELEMENT alta       (#PCDATA)                >
<!ELEMENT baja       (#PCDATA)                >
<!ELEMENT precip     EMPTY                    >
<!ATTLIST precip     total_dia      CDATA     #REQUIRED
     tipo       (lluvia | nieve)    "lluvia"
     fuerza     (ligera | fuerte)   #IMPLIED  >
]>
<clima>
   <ciudad>
      <nombre>Mexico DF</nombre>
      <reporte>
         <alta>27</alta>
         <baja>18</baja>
         <precip total_dia="0" tipo="lluvia" fuerza="ligera"/>
      </reporte>
   </ciudad>
   <ciudad>
      <nombre>Monterrey</nombre>
      <reporte>
         <alta>42</alta>
         <baja>36</baja>
         <precip total_dia="0" tipo="lluvia" fuerza="fuerte"/>
      </reporte>
   </ciudad>
</clima>
```

DTD incluida con el XML.

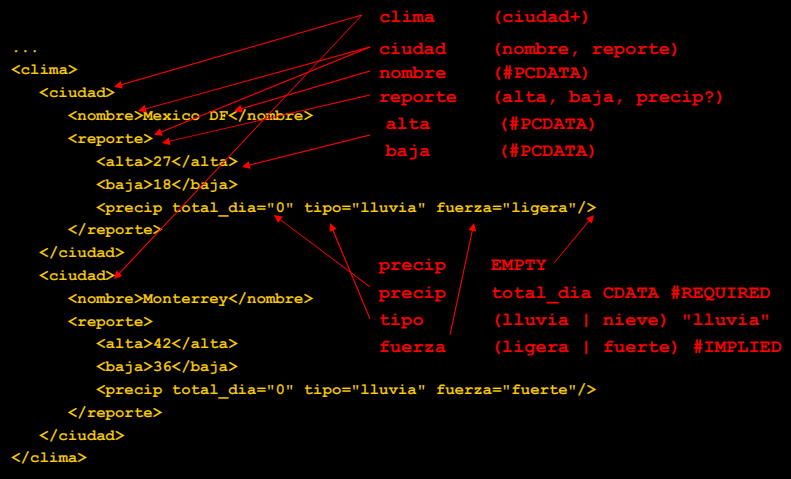
Código XML.

2 elementos de segundo nivel de ejemplo.

Ejemplo completo (XML+DTD).

```
...
<!DOCTYPE clima [
<!--          Nombre    Modelo contenido    -->
<!ELEMENT clima     (ciudad+)                 >
<!ELEMENT ciudad    (nombre, reporte)         >
<!ELEMENT nombre    (#PCDATA)                 >
<!ELEMENT reporte   (alta, baja, precip?)     >
<!ELEMENT alta      (#PCDATA)                 >
<!ELEMENT baja      (#PCDATA)                 >
<!ELEMENT precip    EMPTY                     >
<!ATTLIST precip    total_dia      CDATA      #REQUIRED
     tipo      (lluvia | nieve)    "lluvia"
     fuerza    (ligera | fuerte)   #IMPLIED  >
]>
...
```

Ejemplo completo (XML+DTD).



Ejercicio: Construir una DTD.

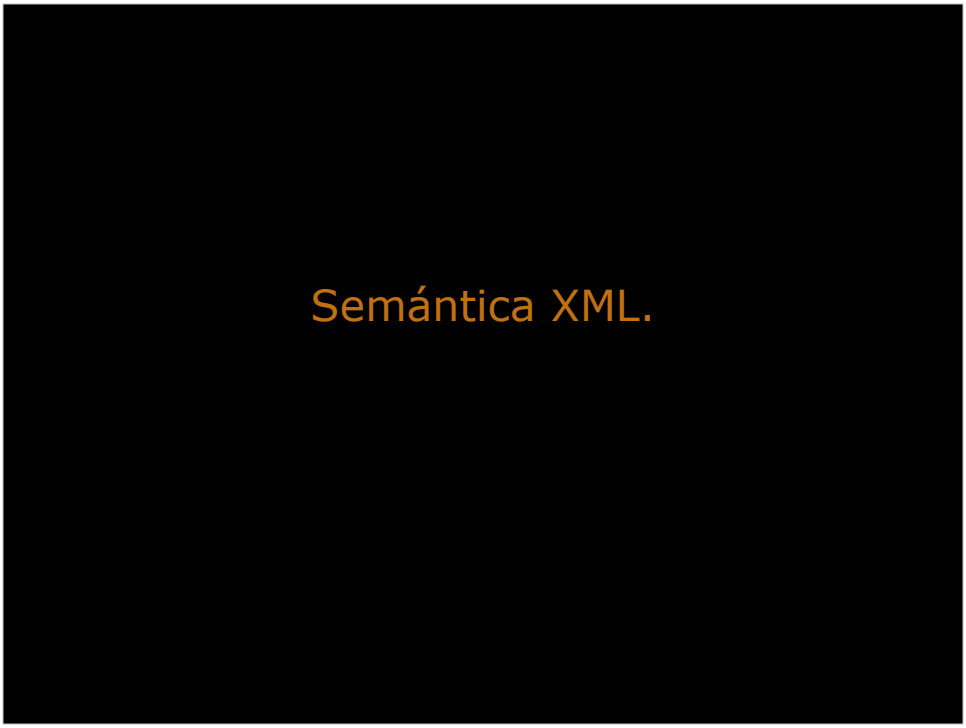
- Analizar documento estructurado.
- Disecar los elementos de información.
- Definir la estructura subyacente.
- Crear una representación de la estructura del documento en forma de una DTD.
- Crear un documento XML que siga ese DTD.

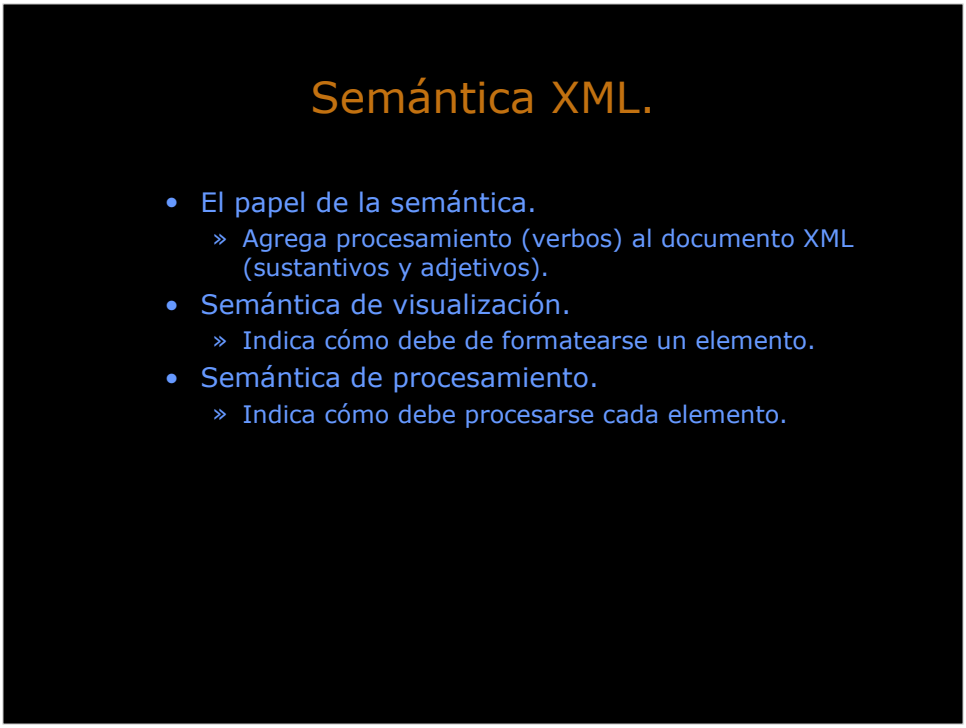
Alternativas a las DTDs.

- Para datos generados automáticamente:
  - » Scripts.
  - » Programas.
- Para datos generados por personas:
  - » Formularios.
  - » Scripts de conversión.
  - » Editores restringidos.
  - » “Guías de uso y estilo”.
- Esquemas W3C.
  - » Nueva especificación del W3C.

El procesador XML (*parser*).

- Software que reconoce e interpreta las reglas del XML.
  - » También se le llama analizador o intérprete XML.
- Con XML bien formado:
  - » Revisa que el documento siga las reglas del XML para considerarse bien formado.
- Con XML válido:
  - » Revisa una DTD XML, luego
  - » revisa el documento XML con las reglas XML, luego
  - » revisa el documento XML con las reglas del DTD.


Semántica XML.



La semántica de visualización es muy sencilla porque hay un número finito de maneras de formatear un documento.

La semántica de procesamiento es complicada porque hay una cantidad infinita de maneras de procesar un documento.

El XSL.

Lenguaje extensible de hojas de estilo
(*extensible stylesheets languaje*).

XSL.

- HTML: Formato sin estructura.
  - » Lenguaje de composición (*typesetting*).
  - » No extensible.
- CSS: Formato mejorado, sin estructura.
  - » Lenguaje de hojas de estilo.
  - » Ignora el formato predeterminado del HML, pero no puede modificar la estructura básica del documento.
- XML: Estructura sin formato.
  - » Define los elementos.
  - » Crea una estructura jerárquica de un conjunto de información.

Agregando formato a la estructura.

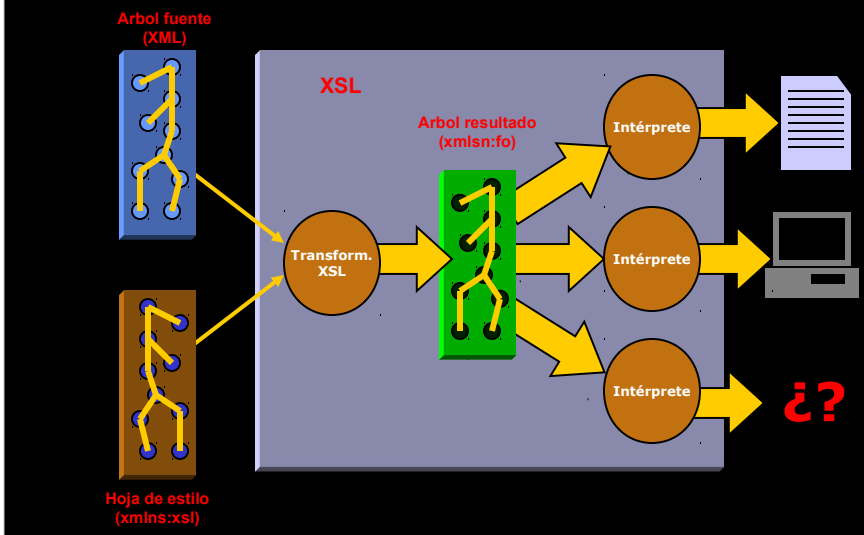
- Lenguaje de hojas de estilo.
- Proporciona definiciones semánticas (verbos o acciones) para los elementos.

- Consiste de dos partes:
  - » Lenguaje para transformar XML.
  - » Vocabulario para especificar semántica de formato.

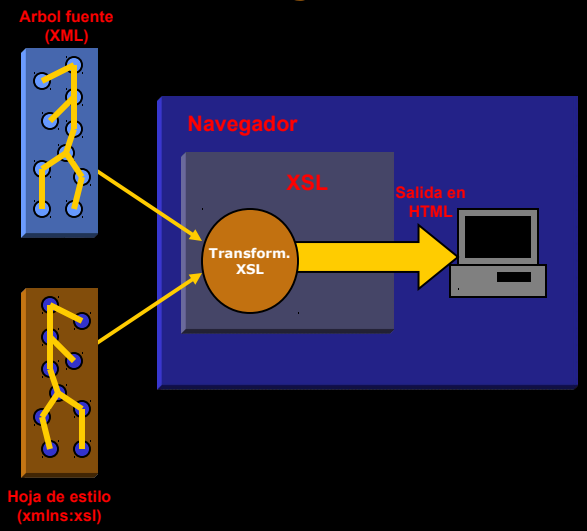
XSL.

- Un lenguaje para expresar hojas de estilo.
- Proporciona semántica de visualización para el XML.
  - » Relaciona elementos XML con HTML o con otros lenguajes de formato (PDF, LaTeX, PostScript, etc).
- Soporte funcional para CSS.
  - » Simple, sintaxis conocida.
  - » Los principiantes pueden aprender rápido.

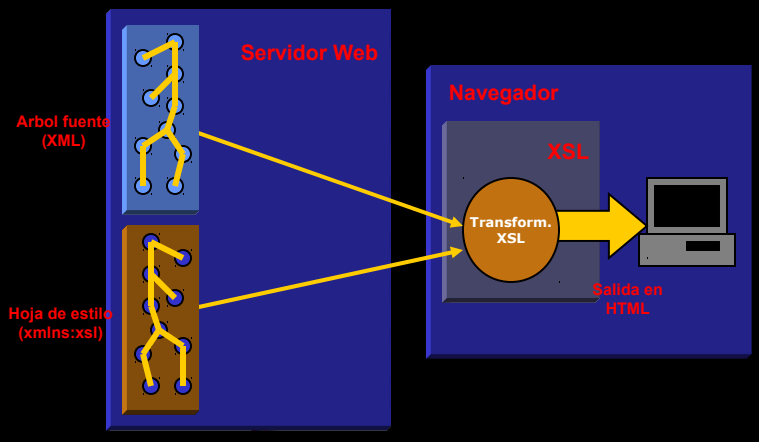
Cómo funciona el XSL.



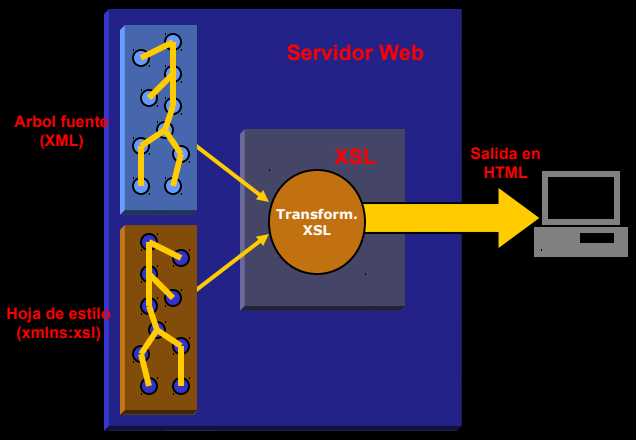
Cómo funciona el XSL en el navegador.



Cómo funciona el XSL en el servidor sin conversion HTML.



Cómo funciona el XSL en el servidor.



XML + XSL

- Un documento XML referencía a un documento XSL por medio de un fragmento de código como este:

```
<?xml-stylesheet type="text/xsl" href="clima.xsl"?>
```

Inicio típico de código XSL.

```
<?xml version="1.0"?>
 <xsl:stylesheet
     xmlns:xsl="http://www.w3.org/TR/WD-xsl"
     xmlns:html="http://www.w3.org/TR/REC-html40"
     result-ns=""
     language="JScript">
 <xsl:template match="/">
```

La primera línea es la declaración de que es un doc. XML. Recordemos que la sintaxis del XSL está definida en XML.

Lo siguiente describe qué espacio de nombres (namespace) se debe de usar para identificar las etiquetas de este documento. Definimos que hay algunas que inician su identificación con “XSL” y otra con “html”. Definimos finalmente que el espacio de nombres por defecto es el rferenciado con “html”. Esto significa que una etiqueta <P> es igual que <html:P>.

Finalmente, el último atributo especifica que JScript es el lenguaje de script por defecto para este documento.

Plantillas XSL.

- Un documento XSL aplica una o varias plantillas (*templates*) al código fuente XML.
- Un archivo XSL es una secuencia de plantillas que se aplican a una o más etiquetas XML de acuerdo a un patrón.

```
<xsl:template match="/">
 . . .
</xsl:template>
<xsl:template match="clima/ciudad">
 . . .
</xsl:template>
```

coincide con el elemento raíz.

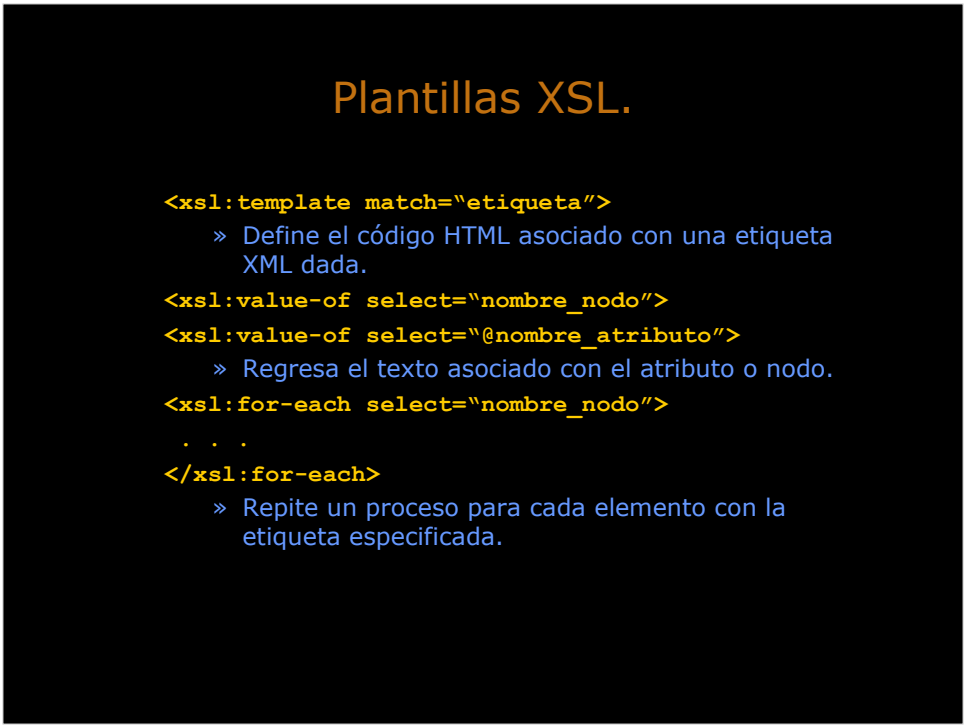
coincide con ciudad, descendiente de clima.

Ejemplo de plantillas XSL.

```
<xsl:template match="/">
  <HTML>
    <HEAD>
      <TITLE>El clima.</TITLE>
    </HEAD>
    <BODY BGCOLOR="White">
      <h1>El clima.</h1>
      <TABLE width="60%" border="1" cellspacing="0"
   cellpadding="5">
        <xsl:apply-templates select="clima/ciudad" order-
   by="+nombre"/>
      </TABLE>
    </BODY>
  </HTML>
</xsl:template>
```

Ejemplo de plantillas XSL.

```
<xsl:template match="clima/ciudad">
  <TR>
    <TD style="font-weight: bold; color: Black; font-family:
   sans-serif;">
      <xsl:apply-templates select="nombre"/>
    </TD>
    <TD style="font-weight: bold; color: Red; font-family: sans-
   serif;">
      <xsl:apply-templates select="reporte"/>
    </TD>
    <TD style="font-weight: bold; color: Blue; font-family:
   sans-serif;">
      <xsl:apply-templates select="reporte/precip"/>
    </TD>
  </TR>
</xsl:template>
```



Si se omite “select=“ entonce se regresa el contenido del elemento actual.

Plantillas XSL.

```
<xsl:apply-templates match="Nombre">
<xsl:apply-templates match="@Atributo">
```

- Aplica todas las plantillas posibles a todos los elementos que coincidan.

Patrones XSL.

`ciudad`	Elemento.
`clima/ciudad`	Elemento de un ancestro dado.
`precip[@tipo]`	Filtro para atributo.
`precip[@tipo="lluvia"]`	Filtro para atributo.
`.[@total_dia > 0]`	Filtro para nodo actual.

Hay muchas variantes de patrones XSL.

XSL condicional.

- Cuando la generación de HTML depende del valor de algún atributo o elemento hay dos opciones:
  - » Estatutos XSL condicionales.
  - » Scripts.
    - ° Extensión de IE, no estándar.

Estatutos condicion XSL **if**.

```
<xsl:if test="condicion">
 . . .
</xsl:eval>
```

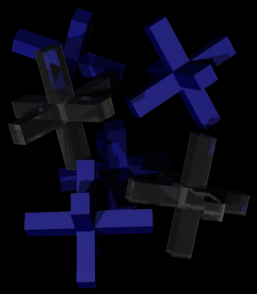
» Evalua una condición, si el nodo actual retorna un valor, entonces se considera verdadera la condición.

- Ejemplo:

```
<xsl:template match="precip">
  <xsl:if test=".[@total_dia > 0]">
       <xsl:value-of select="@total_dia"/> mm
</xsl:if>
</xsl:template>
```

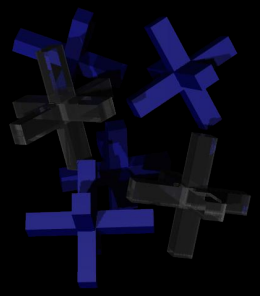
Ejercicio: XSL en el navegador.

- Crear un documento XSL para procesar el XML anterior.
- Visualizar en el navegador.
- Modificarlo para crear salida condicional.



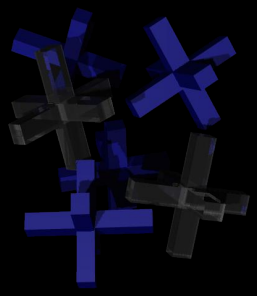
Ejercicio: XSL en el servidor sin conversión HTML.

- Convertir el documento XML en ASP.
- Visualizar en el navegador.



Ejercicio: XSL en el servidor.

- Crear una página ASP para convertir el documento XML en HTML.
- Visualizar en el navegador.




Islas de datos XML.

Islas XML.

- XML dentro de una página HTML.
- Invoca una instancia del procesador XML del cliente.
- Puede ser identificada por un ID.
- Puede controlarse y modificarse con scripts en el cliente.

Ejemplos de islas XML.

```
<html>
<head>
   <title>Islas XML</title>
   <XML id="info-clima">
        <clima><ciudad>
              <nombre>Mexico DF</nombre>
              <reporte>
                 <alta>27</alta><baja>18</baja>
                 <precip total_dia="0" tipo="lluvia"
                                  fuerza="ligera"/>
               </reporte>
        </ciudad></clima>
   </XML>
</head>
<body>
...
```

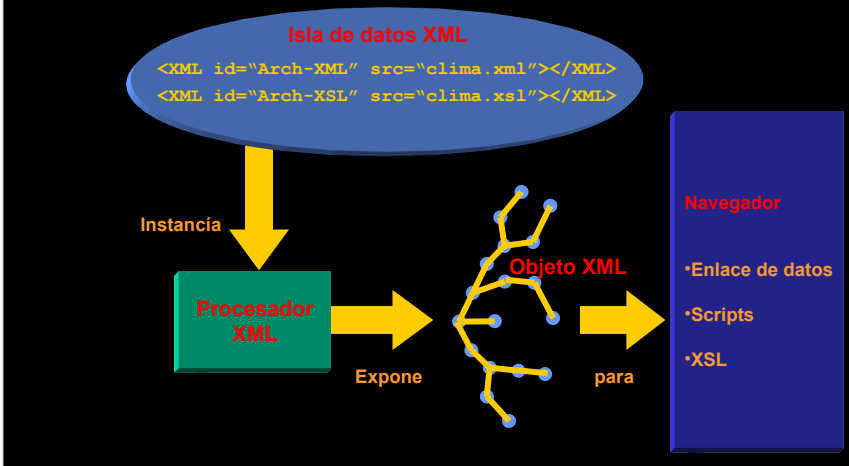
Isla XML

Ejemplos de islas XML.

```
<html>
<head>
   <title>Islas XML</title>
   <XML id="Arch-XML" src="clima.xml"></XML>
   <XML id="Arch-XSL" src="clima.xsl"></XML>
</head>
<body> ...
```

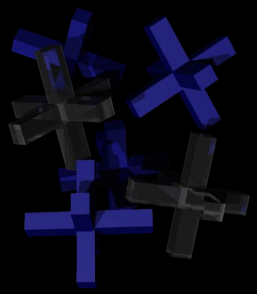
Isla XML

XML en el DOM.

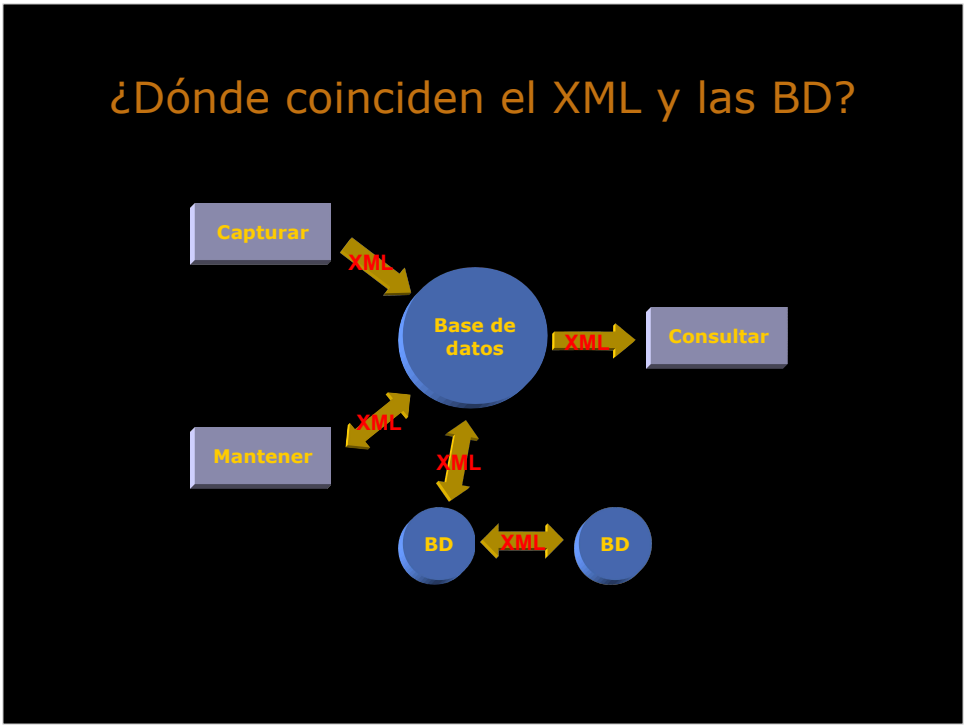

Isla de datos XML
<XML id="Arch-XML" src="clima.xml"></XML>
<XML id="Arch-XSL" src="clima.xsl"></XML>
Instancia
Procesador XML
Expone
Objeto XML
para
Navegador
•Enlace de datos
•Scripts
•XSL

Ejercicio: Islas XML.

- Crear una página HTML con una isla XML.
- Incluir documento XSL.
- Intercambiar entre varios documentos XSL para alterar el estilo de la visualización.



El XML y las bases de datos.



En la comunicación, recordemos que XML no descrime mecanismos para accesar bases de datos; solo describe estructuras para intercambiar información.

XML y las BDs.

- Capturar en XML.
  - » Publicar de una fuente XML hacia la BD.
- Consultar en XML.
  - » Crear salidas desde la BD a un formato de presentación (como el HTML).
- Exportar en XML.
  - » Crear vistas lógicas de la base de datos.
- XML como protocolo entre BDs.
  - » Operaciones entre bases de datos usando XML.

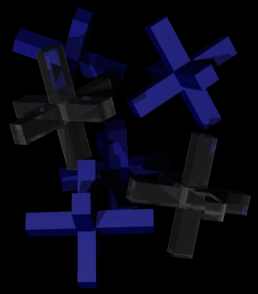
Publicación Web de XML a HTML.

- El XML funciona como BLOB persistente en el sistema de archivos.
  - » Requiere asistencia para encontrar cada documento.
- Almacenar y consultar los documentos XML desde una base de datos.
- Conversión a HTML usando XSL en el servidor.
- Enviar al navegador.

- Ventajas:
  - » XML para manejo de documentos.
  - » HTML para máxima compatibilidad con los navegadores.

Ejercicio: Publicación de XML a HTML.

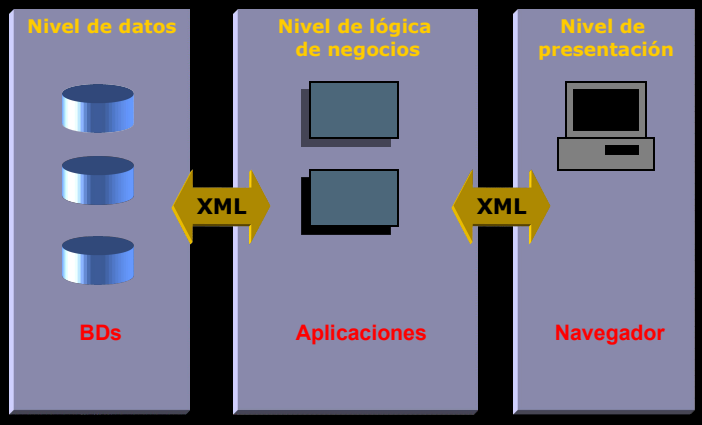
- Crear documentos en XML en archivos independientes y un XSL.
- Crear páginas ASP para mostrar contenido.
- Visualizar en el navegador.
- Crear página ASP para generar índices automáticamente.
- Visualizar en el navegador.



Aplicaciones Web con XML.

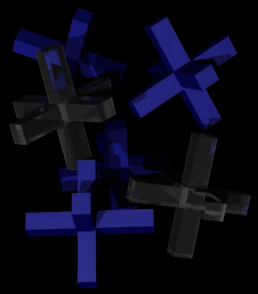
- Modelo de 3 niveles (3-tier).
  - » Nivel de presentación o de usuario.
    - ° Código para presentación.
  - » Nivel de lógica de negocios.
    - ° Código donde se ejecutan las decisiones de la aplicación, se aplican políticas y casi toda la lógica que rige a la aplicación.
  - » Nivel de datos.
    - ° Código para accesar y transformar básicamente el contenido de bases de datos.
- Mejor escalabilidad y flexibilidad.

XML y el modelo de 3 niveles.



Ejercicio: Análisis de “Buscador de Computadoras”.

- Versión modificada del original de Dave Cohen,
  - “Construya su nivel de negocios para comercio electrónico de la manera sencilla con XML, ASP y Scripts”, MIND, enero 2000.



Más información.

- MSDN Latinoamérica,
  - » http://www.microsoft.com/latam/msdn/
- MSDN Web Workshop,
  - » http://msdn.microsoft.com/workshop/
- W3C,
  - » http://www.w3.org/xml/
- Architag,
  - » http://architag.com/xmlu/
- XML en DevX,
  - » http://www.xml-zone.com/