Web Semántica

Jose Emilio Labra Gayo

Departamento de Informática
Universidad de Oviedo

Tecnologías de Base

Internet
World Wide Web
HTTP
Unicode
URI

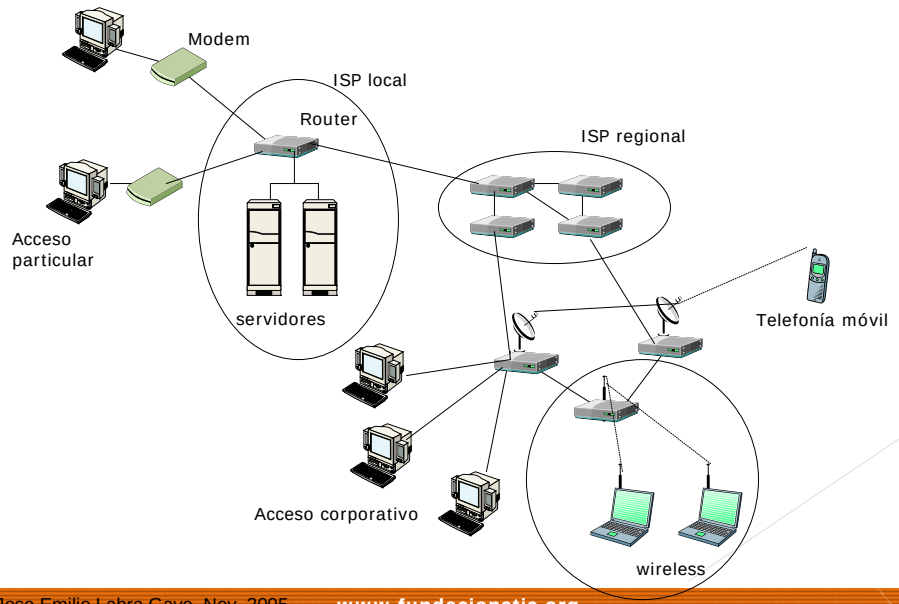
(60-80) Origen militar
- Protocolos de comunicación (TCP/IP)
- Seguridad ante ataques (múltiples servidores)

(80 – 90) Implantación académica
- Protocolos de intercambio de información (FTP, SMTP, ...)

(90-95) World Wide Web
- HTTP, HTML, etc.
- Enorme biblioteca con material hipermedia

(95 – 00) Acceso comercial
- Posibilidad de negocio ⇒ Dinero!!
- Boom comercial

(00-) Crisis de las punto com
- Historias de fracasos ⇒ Lecciones aprendidas
- Nuevas posibilidades: Computación ubicua, Web semántica, etc.



IP *(Internet Protocol)* protocolo de intercambio de paquetes
- Asigna direcciones globales únicas (32 bits)
- IPv6: nueva versión propuesta, con direcciones de 128 bits
- **Host***:* es un nodo de la red (con dirección IP)
  - Cliente: Ordenador que solicita servicios
  - Servidor: Ordenador que propicia respuestas a los clientes
  - Además de la **dirección IP**, el **puerto** (16bits) identifica el servicio
  - Varios puertos predefinidos (80 = HTTP, 25 = SMTP, 20/21 = FTP, etc.)

Protocolo TCP: Comunicación entre nodos manteniendo una conexión.
- Incluye servicio de transporte y de control de congestiones
- No garantiza tiempos ni retardos mínimos
- Se utiliza para transmitir correos, ficheros, etc.

Protocolo UDP (Datagrama) no se realiza una conexión
- Pueden producirse pérdidas de algunos paquetes
- Utilizado para transmisión de voz

HTTP: Protocolo de transferencia de hipertexto
- Puerto 80
- Utiliza TCP/IP como protocolo subyacente
- No incluye estado (no se almacena información del cliente)
- HTTP/1.1 proporciona conexiones persistentes

FTP: Transferencia de ficheros
- Puertos 20/21
- Mantiene estado

SMTP (simple mail transfer protocol)
- Formato de mensajes mediante MIME
- Protocolos de acceso (POP3, IMAP, etc.)

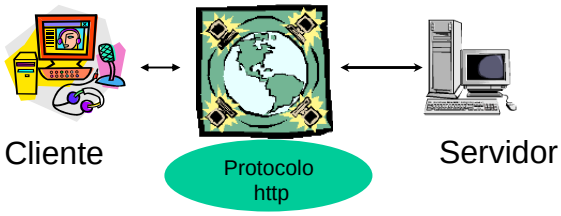
DNS: Asignación de nombres de dominio

Otros: LDAP , NNTP, WebDAV, etc.

Hipertexto (**Ted Nelson, 1965**)
- Texto no secuencial, con enlaces

WWW (**Tim Berners-Lee, 1989**):
- Protocolo HTTP (**Hypertext transfer protocol**)
  - Arquitectura cliente/servidor
- Lenguaje HTML (**HyperText Markup Language**)
  - Lenguaje de Marcado descriptivo para hipermedia
- URIs (**Identificadores universales de recursos**)
  - Identificación única de recursos (nombres globales)
  - Describen protocolo de acceso (http, ftp, etc.)

Arquitectura cliente/servidor



Visualizador

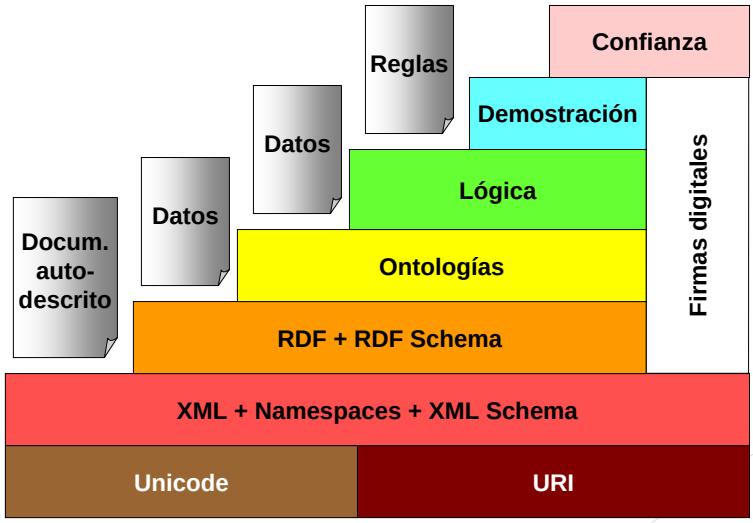
```
GET http://servidor.com/hola.html
```


```
http:/1.0 200 OK
<html>
<body>
Enlace a
<a href="http://www.di.uniovi.es/p.html">
   Otro</a>
</body>
</html>
```

Filosofía de la WWW

- Universalidad
  - Sistema de información a nivel mundial
  - Enlaces únicos (URI)
- Accesibilidad
  - Facilitar el acceso desde cualquier punto
  - Evitar discriminación por razón de ...
- Estandarización
  - Gran número de plataformas y entornos computacionales
    - Linux / Windows / Macintosh /...
    - Mainframes / PCs / PDA / ...
  - Sistema basado en recomendaciones
  - Consorcio formado por
    - Organismos internacionales
    - Industrias
    - Instituciones educativas

Arquitectura de la Web



Los ordenadores manejan código binario: 0s y 1s

Bytes: Grupos de 8 bits

Números: Sistemas de codificación binaria, octal, hexadecimal...

Caracteres: Código que asocia a cada carácter un nº

ASCII: 7 bits ⇒ (0 – 127)

(A)merican (S)tandard (C)ode for (I)nformation (I)nterchange

Extensiones de ASCII

ISO-8859-1 (iso-latin-1)

(8 bits) ASCII (0-127) + otros caracteres típicos de Europa occidental

Familia ISO-8859-X = Otros alfabetos europeos

ISO-8859-15 (iso-latin-9) Igual que iso-8859-1 + símbolo de €

¡CUIDADO! ...hay muchos idiomas y muchos caracteres...
ℜ € き せ サイクル ㊎ ∈ ∀ ℵ £ ウ ぼ

ISO-10646 (31 bits) Define un repertorio universal de caracteres (UCS)

En continua revisión: ISO-10646-2:2001 contiene más de 70.000 caracteres

UNICODE = Consorcio de empresas que define restricciones sobre la implementación de ISO-10646

Varias codificaciones (UTF = Unicode Transformation Format)

- UTF-8: Los primeros 127 códigos se presentan igual (compatible con ASCII)

  El resto se codifican en longitud variable

  Relativamente Eficiente

- UTF-16: Usa 16bits para los caracteres más comunes, el resto con pares de 16 bits

- UTF-32: Codificación directa en 32 bits (desperdicio de espacio)

NOTA: Conviene distinguir:
Carácter: Entidad abstracta (Letra A)
Glifo (*Glyph*): Representación del carácter A A A A A A
Fuente (*Font*): Conjunto de glyphs, ejemplo: Times Roman, Arial, etc.

Imágenes: En pantalla = matriz de puntos de colores (pixels)

Formatos Raster (Raw): Se enumeran todos los puntos con sus colores

Ejemplo: Bitmap, TIFF

Compresión: diversos algoritmos de compresión

GIF: Utiliza 8 bits (hasta 256 colores)

Byte de color = Indice en la paleta de colores

Bueno para dibujos o texto, poco recomendable para fotografías

JPEG: utiliza 24 bits (hasta 16 millones de colores)

Recomendable para forografías

Vectorial: Se enumeran las instrucciones de dibujo

Ejemplos: DXF, SVG (estándar de Internet)

Sonido: Formatos raster (WAV) y comprimidos (MP3)

Vídeo: Formatos comprimidos (MPEG)

Realidad Virtual: Lenguaje de Modelado (VRML, X3D)

URI: (Uniform resource Indentifier) Identifica un recurso de forma global

Puede sub-clasificarse en:

URL (Uniform resource locator)

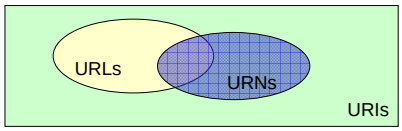
Además de identificar el recurso, indica cómo llegar hasta él

esquema://servidor:puerto/ruta?datosGET

http://www.uniovi.es:8080/prueba/carrito?action=print

URN (Uniform resource name): Nombre de recurso

Ejemplo: urn:xmlorg:objects:schema:xmlschema:xcatalog



IRIs (Internationalized Resource Identifiers)
permiten utilizar caracteres Unicode en los identificadores

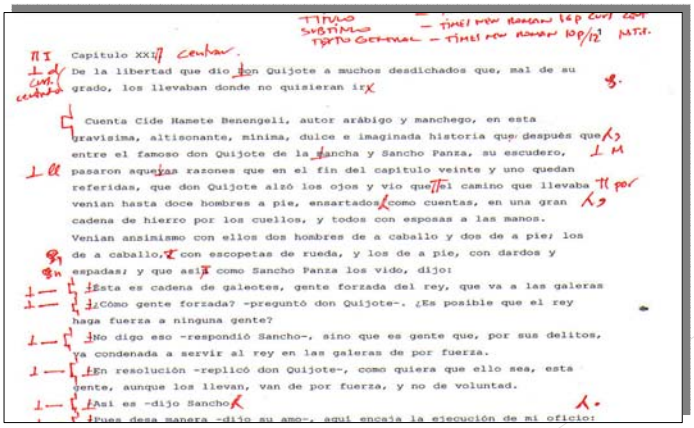
Sesión 2
Lenguaje XML

Departamento de Informática
Universidad de Oviedo

Marcado de documentos

Orígenes: Industria de la Publicación

Se usaban marcas para indicar cómo componer el documento para la impresión



Ejemplo de corrección tipográfica
Fuente: http://recursos.cnice.mec.es/bancoimagenes

Sólo texto

```
ALBA Abril de 1915 Granada Mi corazón oprimido
siente junto a la alborada el dolor de sus
amores y el sueño de las distancias.
```

Texto marcado

```
]ALBA[  ← Título, negrita, centrado, 14pt
]Abril de 1915[← SubTítulo, negrita, centrado
]Granada[← SubTítulo cursiva, centrado
]Mi corazón oprimido [← Verso, 10pt
]siente junto a la alborada [← Verso
]el dolor de sus amores [← Verso
]y el sueño de las distancias. [← Verso
```

Resultado

ALBA
Abril de 1915
Granada

Mi corazón oprimido
siente junto a la alborada
el dolor de sus amores
y el sueño de las distancias.

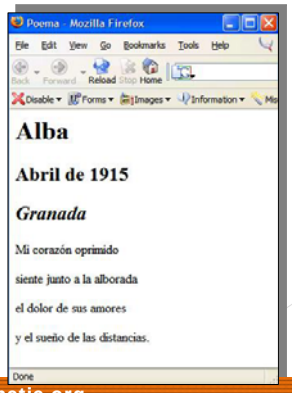
El marcado **no** es la información que contiene el documento

Marcado = **información acerca del documento** = **meta-información**

Lenguajes de Marcado descriptivo: Incluyen marcas que describen cómo procesar el documento

Ejemplo: HTML

```
<html>
<head><title>Poema</title></head>
<body>
<h1>Alba</h1>
<h2>Abril de 1915 </h2>
<h2><i>Granada</i></h2>
<p>Mi corazón oprimido</p>
<p>siente junto a la alborada</p>
<p>el dolor de sus amores</p>
<p>y el sueño de las distancias. </p>
</body>
</html>
```



Marcado generalizado = Sintaxis común que facilita la creación de lenguajes descriptivos

HTML

```
<html>
<head><title>Poema</title></head>
<body>
<h1>Alba</h1>
<h2>Abril de 1915 </h2>
<h2><i>Granada</i></h2>
<p>Mi corazón oprimido</p>
<p>siente junto a la alborada</p>
<p>el dolor de sus amores</p>
<p>y el sueño de las distancias.</p>
</body>
</html>
```

Otras marcas...(misma sintaxis)

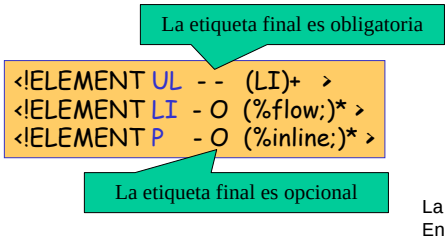
```
<poema fecha="Abril de 1915"
       lugar="Granada">

<titulo>Alba</titulo>

<verso>Mi corazón oprimido</verso>
<verso>siente junto a la alborada</verso>
<verso>el dolor de sus amores</verso>
<verso>y el sueño de las distancias. </verso>
</poema>
```

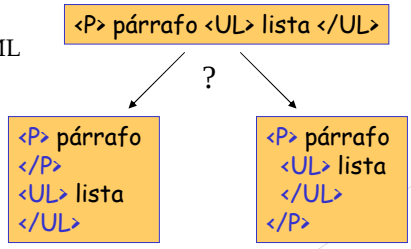

(70-) GML desarrollado en IBM – Generalized Markup Language (Goldfarb, Mosher, Lorie)

(86) SGML *Standard Generalized Markup Language (Estándar ISO)*

- *Utilizado para el intercambio de documentos*
- *Principio: Separar contenido de la forma de representarlo*
- *Muy flexible (permite definir vocabularios específicos para cada aplicación)*
- *HTML era un vocabulario de SGML*

Ejemplo de la especificación de HTML en SGML



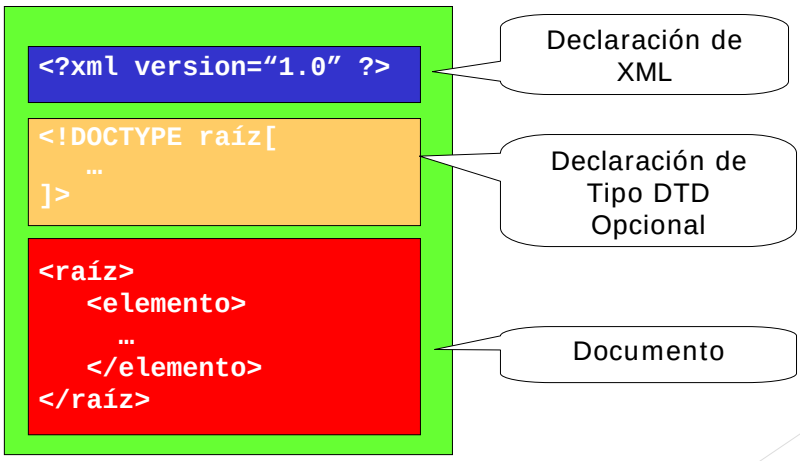


La ambigüedad se deshace mediante el DTD
En Internet no siempre podremos acceder al DTD

T. Bray, J. Paoli, C. M. Sperberg-McQueen (1995)
Objetivos de diseño (según la especificación)

1. Utilizable en Internet
2. Soporte a gran variedad de aplicaciones
3. Compatible con SGML
4. Debe ser fácil escribir programas que procesen XML
5. Número de características opcionales = Mínimo
6. Documentos legibles por personas
7. El diseño de XML debe poder hacerse rápidamente
8. El diseño de XML debe ser formal y conciso
9. La creación de documentos XML debe ser fácil
10. La concisión de las marcas XML no tiene importancia (es preferible la claridad a la brevedad)

20% de características de SGML ⇔ 80% de funcionalidad de SGML
Detalle (Especificación de XML = 26 páginas, de SGML > 500



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

version: Actual = 1.0
Borrador de versión 1.1
Mayor compatibilidad con Unicode
Identificadores: Permite cualquier carácter Unicode
encoding: UTF-8, UTF-16, iso-8859-1, etc.
standalone: Indica si el documento no hace referencias
a entidades externas

Los documentos consisten en una serie de datos marcados mediante etiquetas
Las etiquetas describen la estructura del documento
Un elemento = grupo formado por etiqueta inicial, etiqueta final y contenido entre ambas. La etiqueta inicial puede incluir atributos.

```
<etiqueta attributo="valor">……</etiqueta>
```

Distinción minúsculas/mayúsculas

Elemento vacío: Entre la etiqueta inicial y final no hay información:

```
<etiqueta attributo="valor"></etiqueta>
```

⇓

```
<etiqueta atributo="valor"/>
```

Se pueden anidar elementos

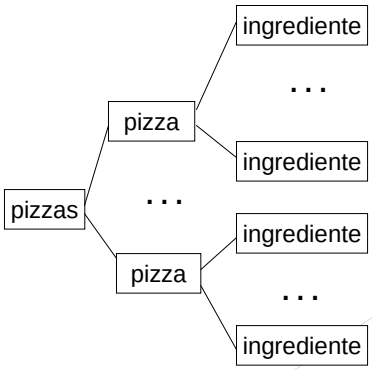
```
<externo>
  <interno>texto</interno>
</externo>
```

...pero no se pueden entrelazar:

```
<externo>
  <interno>texto</externo>
</interno>
```

Sólo puede haber un único elemento raíz
Cada documento XML equivale a un árbol

```
<pizzas>
  <pizza nombre="Barbacoa" precio="8">
    <ingrediente nombre="Salsa Barbacoa" />
    <ingrediente nombre="Mozzarella" />
    <ingrediente nombre="Pollo" />
    <ingrediente nombre="Bacon" />
    <ingrediente nombre="Ternera" />
  </pizza>
  . . .
  <pizza nombre="Margarita" precio="6">
    <ingrediente nombre="Tomate" />
    <ingrediente nombre="Jamón" />
    <ingrediente nombre="Queso" />
  </pizza>
</pizzas>
```



Cada elemento puede contener atributos en la etiqueta inicial

```
<pizza nombre="Margarita" precio="6">
  . . .
</pizza>
```

El orden de los atributos no es significativo
No puede haber 2 atributos con el mismo nombre

Atributos predefinidos:

xml:lang: Especifica el idioma.
Por ejemplo: en (inglés), sp (español)

xml:space: Especifica cómo tratar el espacio en blanco.
Valores: preserve = Mantenerlo
default = Permitir a la aplicación que lo trate como quiera.

Comentarios

```
<!-- el texto de un comentario
       no es analizado -->
```

Caracteres especiales: No pueden incluirse directamente

```
<código>
  if x &lt; 4 then x:=x + 1;
</código>
```

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

Secciones CDATA

Si se desea introducir código sin analizar

```
<código>
  if x < 3 && x > 4 then
    print "Hola"
</código>
```

```
<código>
  if x &lt; 3
    &amp;amp;&amp;amp; x &gt; 4 then
      print &amp;quot;Hola&amp;quot;
</código>
```

☹

```
<código>
<![CDATA[
if x < 3 && x > 4 then
    print "Hola"
]]>
</código>
```

Es posible incluir instrucciones que indican al procesador alguna acción a realizar

Sintaxis: `<?aplicación datos ?>`

Pueden utilizarse para asociar una hoja de estilos al documento:

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

...o para otros propósitos especiales

En realidad la declaración de documento es una instrucción de procesamiento

```
<?xml version="1.0" ?>
```


Documento bien formado

Sigue las reglas sintácticas

Importante:

- Contiene un único elemento raíz
- Todas las etiquetas están correctamente anidadas

```
<pizzas>
 <pizza nombre="Margarita" precio="6">
  <ingrediente nombre="Tomate" />
  <ingrediente nombre="Queso" />
 </pizza>
</pizzas>
```

```
<pizzas>
 <pizza nombre="Margarita" precio="6">
  <ingrediente nombre="Tomate" >
</pizzas>
```

Se puede incluir una declaración del tipo de documento

```
<?xml version="1.0"?>
<!DOCTYPE pizzas SYSTEM "pizzas.dtd">
<pizzas>
<pizza nombre="Margarita" precio="6">
<ingrediente nombre="Tomate" />
</pizza>
</pizzas>
```

pizzas.dtd

```
<!ELEMENT pizzas (pizza*)>
<!ELEMENT pizza (ingrediente*)>
<!ELEMENT ingrediente (#PCDATA)>
<!ATTLIST pizza nombre CDATA #REQUIRED>
<!ATTLIST pizza precio CDATA #REQUIRED>
<!ATTLIST ingrediente nombre CDATA #REQUIRED>
```

Documento válido

Está bien formado y

La estructura encaja con la declaración del tipo de documento

DTD interno

```
<?xml version="1.0"?>
<!DOCTYPE pizzas [
 <!ELEMENT pizzas (pizza*)>
. . .
]>
<pizzas>. . . </pizzas>
```

También es posible especificar un DTD externo y añadir definiciones locales

DTD externo

SYSTEM (DTDs de ámbito local)

```
<?xml version="1.0"?>
<!DOCTYPE pizzas SYSTEM "http://www.mafia.it/pizzas.dtd" >
<pizzas>
. . .
</pizzas>
```

PUBLIC (DTDs compartidos por diversas organizaciones)

```
<?xml version="1.0"?>
<!DOCTYPE HTML PUBLIC
        "-//W3C//DTD HTML 4.0//EN"
        "http://www.w3c.org/TR/REC-html/strict.dtd">
```

ELEMENT
- Elementos del documento XML

ATTLIST
- Lista de atributos de un elemento

ENTITY
- Entidades (≈variables o macros)

NOTATION
- Definen tipos de contenidos
- Facilitan la inclusión de formatos binarios (imágenes, vídeos, sonidos, ...)

(?)	= 0, 1 elemento
(*)	= 0 ó más elementos
(+)	= 1 ó más elementos
(|)	= alternativa
(,)	= secuencia
EMPTY	= vacío
ANY	= cualquier estructura de subelementos
#PCDATA	= cadena de caracteres analizados

```
<!ELEMENT  pizza (ingrediente*, inventor?)>
<!ELEMENT servicio (domicilio | restaurante) >
<!ELEMENT ingrediente EMPTY>
<!ELEMENT inventor (#PCDATA)>
<!ELEMENT sección (título, (contenido | sección+))>
<!ELEMENT p (#PCDATA | a | ul | em )* >
```

Recursividad

PCDATA = Parsed Character Data
Indica que los datos son analizados buscando etiquetas

DTD
Atributos

Tipos de datos
CDATA = Cadena de caracteres
NMTOKEN = Palabra (sin espacios)
NMTOKENS = Lista de palabras
Enumeración separada por |
ID = Nombre único (sin duplicados)
IDREF = Su valor debe apuntar a un ID

Valor de los Atributos
#REQUIRED Obligatorio
#IMPLIED Opcional
#FIXED Constante
Valor Valor por defecto

```
<!ATTLIST pizza nombre CDATA #REQUIRED>

<!ATTLIST ingrediente nombre CDATA #REQUIRED
                      calorías CDATA #IMPLIED>

<!ATTLIST precio moneda (euros|dólares) #REQUIRED
                valor CDATA #REQUIRED>

<!ATTLIST persona código ID #REQUIRED>

<!ATTLIST dueño código IDREF #REQUIRED>

<!ATTLIST conOrégano (sí|no) "sí" >

<!ATTLIST impuesto tipo CDATA #FIXED "IVA">
```

```
<pizza nombre="4 estaciones" >
  <ingrediente nombre="Jamón" />
  <precio moneda="euros" valor="7" />
</pizza>

<persona código="23" nombre ="Juan" />
<persona código="35" nombre ="Pepe" />
<persona código="37" nombre ="Luis" />

<dueño código="35" />

<impuesto tipo="IVA" />
```


DTD
Entidades Generales

Entidades: Asignan nombres a ciertos elementos (similar a variables)
Se denotan por &entidad;
No se admite recursividad

```
<!ENTITY marg "Pizza Margarita">
<!ENTITY queso  "<ingrediente nombre='queso' />" >
```

```
<pizza nombre="&marg;" precio="7">
&queso;
</pizza>
```

```
<pizza nombre="Pizza Margarita" precio="7">
<ingrediente nombre='queso' />
</pizza>
```

Entidades numéricas: Código numérico del carácter

∀ ∀ ∃ ∃

Entidades predefinidas: Permiten incluir etiquetas sin analizar

< < " " ' '
> > & &

Permiten usar archivos externos (Incluir otros documentos XML)

pizzas.xml

```
<pizzas>
<pizza nombre="4 Quesos" precio="7">
<ingrediente nombre="Jamón" />
 <ingrediente nombre="Queso" />
</pizza>
. . .
</pizzas>
```

personal.xml

```
<personal>
<trabajador
  nombre=" Benito Alcaparra" >
  . . .
 </trabajador>
. . .
</personal>
```

establecimiento.dtd

```
<!ELEMENT establecimiento ANY>
<!ENTITY personal SYSTEM "personal.xml">
<!ENTITY pizzas SYSTEM "pizzas.xml">
```

establecimiento.xml

```
<establecimiento
      nombre="Pizzería Al Capone">
 &personal;
 &pizzas;
</establecimiento>
```

También se pueden incluir archivos externos de formatos binarios

```
<!NOTATION gif SYSTEM "gifEditor.exe">
<!ENTITY dibujo SYSTEM "logotipo.gif"
NDATA gif>
```


Permiten dar nombres a partes de un DTD

Se denotan por %entidad;

```
<!ENTITY establecimiento (nombre,dueño?,calle,número?,ciudad,país,códigoPostal)>
<!ENTITY persona (dni, nombre,calle,número?,ciudad,país,códigoPostal)>
```

```
<!ENTITY %localización "calle,número?,ciudad,país,códigoPostal" >
<!ENTITY establecimiento (nombre,dueño?,%localización;)>
<!ENTITY persona (dni, nombre, %localización;>
```

Entidades externas: Permiten incluir elementos externos en una DTD

Aplicación: Dividir la definición de una DTD en varios documentos

```
<!ENTITY %persona SYSTEM "persona.dtd">
<!ENTITY %establecimiento SYSTEM
"establecimiento.dtd">

%persona;
%establecimiento;
```

Ejercicios

Creación de ficheros XML y validación

Procesadores de XML

- Chequean que está bien formado
- Validan

Productos

- Visuales: XML Writer, XML Spy, ...
- Modo texto: xmllint, msxml, ...

xmllint forma parte de la librería libxml de GNOME

```
xmllint --valid --noout fichero.xml
```

Validar
Si no se pone nada,
Chequea que está bien formado

No muestra resultado
Si no hay mensajes ⇒ OK

Valoración de XML

Discusión sobre XML: Ventajas

Es un formato estructurado
- Contiene información y meta-información

Ha sido diseñado específicamente para Internet
- Soportado por visualizadores y servidores

Numerosas herramientas de procesamiento

Legible por personas humanas

Admite la definición de vocabularios específicos

Separa contenido del procesamiento y visualización

Aumenta la seguridad mediante la validación de documentos

Formato abierto, respaldado por numerosas organizaciones

Una vez definido un DTD común, facilita intercambio de información

Discusión sobre XML: Inconvenientes

Puede requerir demasiado espacio, ancho de banda y tiempo de procesamiento
Documentos largos con mucha información redundante
Es una sintaxis de documentos, no un lenguaje de programación

```
int main(void) {
 printf("Hola");
 return 0;
}
```

```
<function name="main" type="int">
<arg type="void" />
<call function="printf">
  <param>Hola</param>
</call>
<return value="0"/>
</function>
```

Es posible crear formatos y vocabularios propietarios
Puede fomentar la proliferación de vocabularios específicos
Bueno para texto, malo para datos binarios

```
<?xml version="1.0">
<imagen formato="base64">
DS34JSCDF029876D76523981DFNDF3F2134F5FD019A
FGF23DAND345CD2135911943DCBKAPFGDAJJK32A10
....
</imagen>
```

Poco eficiente como lenguaje de almacenamiento de bases de datos

Espacios de Nombres

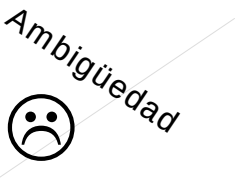
Homonimia: Mismo nombre con diferentes propósitos

```
<país nombre="Francia">
<capital>París</capital>
</país>
```

```
<inversión>
  <capital>7000€</capital>
</inversión>
```

¿Cómo combinar en el mismo documento estos vocabularios?

```
<inversiones>
  <país nombre="Francia">
    <capital>París</capital>
    <capital>1200€</capital>
  </país>
  . . .
</inversiones>
```



Asignar un nombre único a cada etiqueta...

Posibles soluciones:

- Crear una autoridad mundial que asigne nombres...
- ... o usar un mecanismo ya existente: URIs
- Una URI es un identificador global único
  - Ejemplo: **http://www.aulanet.uniovi.es**

SOLUCIÓN:

Asociar a cada etiqueta una URI que indica a qué espacio de nombres pertenece...

[http:www.bolsa.com]:capital
[http:www.geog.es]:capital

Solución fácil...
Asociar a cada etiqueta una URI

```
<[http://www.bolsa.com]:inversiones>
  <[http://www.geog.es]:país
      [http://www.geog.es]:nombre="Francia">
    <[http://www.geog.es]:capital>París
    </[http://www.geog.es]:capital>
    <[http://www.bolsa.com]:capital>1200€
    </[http://www.bolsa.com]:capital>
  </[http://www.bolsa.com]:país>
  . . .
</[http://www.bolsa.com]:inversiones>
```

Legibilidad...

Solución: Asociar un alias a los elementos de un espacio de nombres dentro de un ámbito

xmlns:*alias* define *alias* en el ámbito de un elemento

```
<bolsa:inversiones
          xmlns:bolsa="http://www.bolsa.com"
          xmlns:geog="http://www.geog.es">
  <geog:país geog:nombre="Francia">
    <geog:capital>París</geog:capital>
    <bolsa:capital>1200€</bolsa:capital>
  </geog:país>
  . . .
</bolsa:inversiones>
```

NOTA: Las URIs sólo se utilizan para que el nombre sea único, no son enlaces, ni tienen que contener información

Es posible ir asociando espacios de nombres a los elementos según van apareciendo

```
<bolsa:inversiones
        xmlns:bolsa="http://www.bolsa.com">
  <geog:país
          xmlns:geog="http://www.geog.es"
          geog:nombre="Francia">
    <geog:capital>París</geog:capital>
    <bolsa:capital>1200€</bolsa:capital>
  </geog:país>
  . . .
</bolsa:inversiones>
```


Mediante **xmlns="..."** se define un espacio de nombres por defecto (sin alias)

```
<inversiones
        xmlns="http://www.bolsa.com">
  <geog:país
          xmlns:geog="http://www.geog.es"
          geog:nombre="Francia">
    <geog:capital>París</geog:capital>
    <capital>1200€</capital>
  </geog:país>
  . . .
</inversiones>
```

Se refiere a
http://www.bolsa.com

Posteriores a los DTDs, por tanto, los DTDs no dan soporte a Espacios de Nombres

- Hay que definir los espacios de nombre usados

```
<!DOCTYPE inversiones [
<!ELEMENT inversiones (geog:país*)>
<!ELEMENT geog:país (geog:capital,capital) >
<!ELEMENT geog:capital (#PCDATA)>
<!ELEMENT capital (#PCDATA)>
<!ATTLIST inversiones
        xmlns CDATA #FIXED "http://www.bolsa.com">
<!ATTLIST geog:país
        geog:nombre CDATA #REQUIRED
        xmlns:geog CDATA #FIXED "http://www.geog.es">
]>
```


Ampliamente utilizados para combinar vocabularios

Facilitan la incorporación de elementos no previstos inicialmente

Sintaxis *extraña* al principio

- Uso de prefijos
- URIs como elemento diferenciador...pero las URLS también sirven para acceder a recursos

Difícil combinación con DTDs

XML Schema

Esquema = define estructura de un conjunto de documentos XML

Validar = Chequear que un documento sigue un esquema
- Principal Ventaja: Protección de errores
- Otras aplicaciones: Edición, compresión, etc.

Originalmente se utilizaron los DTDs

Posteriormente se ha desarrollado XML Schema

Existen Otros:
- RELAX-NG, Schematron, etc.

Especifican estructura del documento:
Elementos, atributos, anidamientos, etc.

Integridad referencial mínima (ID, IDREF)

Mecanismo sencillo de abstracción
Entidades Macros

Inclusión de documentos externos

Integrados en XML (Parte de la especificación)

Sencillos de comprender (Expresiones regulares)

La Sintaxis ***no es XML*** (difíciles de manipular)

No soportan ***Espacios de nombres***

No permiten especificar ***tipos de datos*** (por ejemplo: enteros, flotantes, fechas, etc.

No permiten especificar ***secuencias no ordenadas***
((e1,e2,e3)|(e1,e3,e2)|(e2,e1,e3)|...(e3,e2,e1))

No hay soporte para declaraciones ***sensibles al contexto***: Los elementos se definen todos a nivel de documento, ejemplo, contenido con el mismo nombre cuya estructura cambia en diferentes contextos

Soporte limitado para ***Referencias cruzadas***, no es posible formar claves a partir de varios atributos o de elementos

No son extensibles (una vez definido, no es posible añadir nuevos vocabularios a un DTD)

- Sintaxis XML
- Soporte para Espacios de Nombres
- Mayor expresividad
  - Restricciones numéricas
  - Integridad dependientes del contexto
- Tipos de datos
  - Gran cantidad de tipos de datos predefinidos
  - Creación de tipos de datos por el usuario
- Extensibilidad
  - Inclusión/Redefinición de esquemas
  - Herencia de tipos de datos
- Soporte a Documentación

alumnos.xsd

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           targetNamespace="http://www.uniovi.es/alumnos"
           xmlns="http://www.uniovi.es/alumnos">
<xs:element name="alumnos">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="alumno" minOccurs="1" maxOccurs="200"
               type="TipoAlumno"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
<xs:complexType name="TipoAlumno">
  <xs:sequence>
    <xs:element name="nombre" type="xs:string"/>
    <xs:element name="apellidos" type="xs:string"/>
    <xs:element name="nacim" type="xs:gYear"/>
  </xs:sequence>
  <xs:attribute name="dni" type="xs:integer"/>
 </xs:complexType>
</xs:schema>
```

Elemento raíz schema y espacio de nombres determinado

Permite especificar rangos de inclusión

Permite especificar tipos

alumnos.xsd

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           targetNamespace="http://www.uniovi.es/alumnos"
           xmlns="http://www.uniovi.es/alumnos">
<xs:element name="alumnos">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="alumno" minOccurs="1" maxOccurs="200"
               type="TipoAlumno"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
<xs:complexTy
  <xs:sequenc
    <xs:eleme
    <xs:eleme
    <xs:eleme
  </xs:sequen
  <xs:attribu
 </xs:complex
</xs:schema>
```

Los espacios de nombres deben coincidir.
También puede usarse:
xsi:noNameSpaceLocation

alumnos.xml

```
<alumnos
  xmlns="http://www.uniovi.es/alumnos"
  xsi:SchemaLocation="http://www.uniovi.es/alumnos
                      alumnos.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
. . .
</alumnos>
```



```
<xs:element name="alumno">
<xs:sequence>
    <xs:element name="nombre" type="xs:string"/>
    <xs:element name="apellidos" type="xs:string"/>
  </xs:sequence>
</xs:element>
```

+ legible

```
…
<xs:element name="alumno" type="TipoAlumno"/>
…
<xs:ComplexType name="TipoAlumno">
<xs:sequence>
    <xs:element name="nombre" type="xs:string"/>
    <xs:element name="apellidos" type="xs:string"/>
  </xs:sequence>
</xs:ComplexType>
```

+ Reutilizable

```
<xs:element name="alumno">
<xs:sequence>
    <xs:element name="nombre" type="xs:string"/>
    <xs:element name="apellidos" type="xs:string"/>
  </xs:sequence>
</xs:element>
```

```
<xs:element name="alumnos">
<xs:sequence>
  <xs:element ref="alumno" />
</xs:sequence>
</xs:element>
```


Es posible nombrar agrupaciones de elementos y de atributos para hacer referencias a ellas

```
<xs:group name="nombApell">
  <xs:sequence>
    <xs:element name="nombre" type="xs:string"/>
    <xs:element name="apellidos" type="xs:string"/>
  </xs:sequence>
</xs:group>
```

```
<xs:complexType name="TipoAlumno">
 <xs:group ref="nombApell" />
 <xs:element name="carrera" type="xs:string"/>
</xs:complexType>
```

Tipos Complejos: Son tipos que pueden contener elementos o atributos
Construcción básica mediante enumeración de elementos

```
<xs:complexType name="TipoAlumno">
  <xs:sequence>
    <xs:element name="nombre"    type="xs:string"/>
    <xs:element name="apellidos" type="xs:string"/>
    <xs:element name="nacim" type="xs:gYear"
                minOccurs="0" maxOccurs="1"/>
  </xs:sequence>
  <xs:attribute name="dni" type="xs:integer"/>
 </xs:complexType>
```

```
<alumno dni="9399390">
 <nombre>Juan</nombre>
 <apellidos>García García</apellidos>
 <nacim>1985</nacim>
</alumno>
```


choice: Representa alternativas
OJO: Es una o-exclusiva

```
 <xs:complexType name="Transporte">
    <xs:choice>
     <xs:element name="coche" type="xs:string"/>
      <xs:element name="tren" type="xs:string"/>
      <xs:element name="avión" type="xs:string"/>
    </xs:choice>
  </xs:complexType>
```

```
<transporte>
<coche>Renault R23</coche>
</transporte>
```

El contenido Mixto permite mezclar texto con elementos

```
 <xs:complexType name="TCom" mixed="true">
  <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element name="emph" type="xs:string"/>
  </xs:choice>
</xs:complexType>

<xs:element name="comentarios" type="TCom" />
```

```
<comentarios>
 Es un poco <emph>listillo</emph>
</comentarios>
```


all = Todos los elementos en cualquier orden
En DTDs requería enumerar las combinaciones:
(A,B,C)|(A,C,B)|...|(C,B,A)

```
<xs:complexType name="TipoLibro">
  <xs:all>
      <xs:element name="autor" type="xs:string"/>
      <xs:element name="título" type="xs:string"/>
  </xs:all>
</xs:complexType>
<xs:element name="libro" type="TipoLibro" />
```

```
<libro>
 <autor>Juanita la Loca</autor>
 <título>No estoy loca</título>
</libro>
```

```
<libro>
  <título>El kigote</título>
  <autor>Cerbantes</autor>
</libro>
```

No pueden contener elementos o atributos

Pueden ser:

- Predefinidos o *built-in* (Definidos en la especificación)
  - Primitivos
  - Derivados
- Definidos por el usuario (a partir de tipos predefinidos)

string

boolean

number, float, double

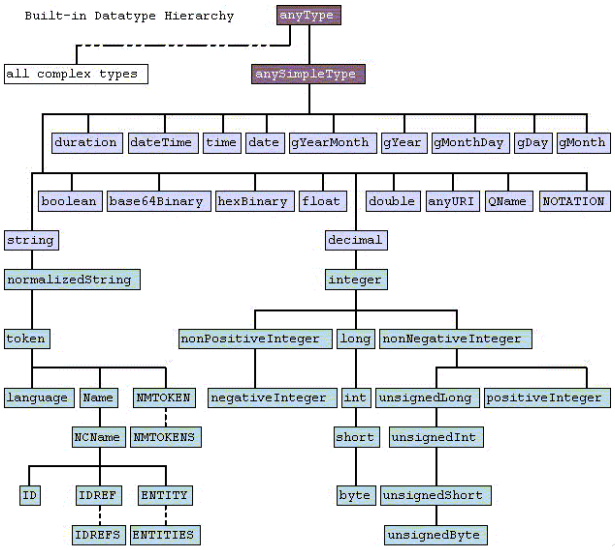
duration, dateTime, time, date, gYearMonth, gYear, gMonthDay, gDay, gMonth

hexBinary, base64Binary

anyURI

QName = Nombre cualificado con espacio de nombres

NOTATION = Notación binaria (similar a DTD)



Facetas fundamentales:

- *equal:* Igualdad entre valores de un tipo de datos
- *ordered:* Relaciones de orden entre valores
- *bounded:* Límites inferiores y superiores para valores
- *cardinality:* Define si es finito o infinito (contable, no contable)
- *numeric:* Define si es numérico o no

Facetas de restricción

- *length*, *minlength*, *maxlength:* Longitud del tipo de datos
- *pattern:* Restricciones sobre valores mediante expresiones regulares
- *enumeration:* Restringe a una determinada enumeración de valores
- *whitespace:* Define política de tratamiento de espacios (preserve/replace, collapse)
- *(max/min)(in/ex)clusive:* Límites superiores/inferiores del tipo de datos
- *totaldigits, fractionDigits:* número de dígitos totales y decimales

Enumeración

```
<xs:simpleType name="TipoCarrera">
<xs:restriction base="xs:token">
 <xs:enumeration value="Gestión"/>
 <xs:enumeration value="Sistemas"/>
</xs:restriction>
</xs:simpleType>
```

Restricciones sobre valores

```
<xs:simpleType name="mes">
<xs:restriction base="xs:integer">
  <xs:minInclusive value="1" />
  <xs:maxInclusive value="31" />
</xs:restriction>
</xs:simpleType>
```



```
<xs:simpleType name="ComponentesRGB">
 <xs:list itemType="ComponenteRGB"/>
</xs:simpleType>

<xs:simpleType name="ComponenteRGB">
<xs:restriction base="xs:nonNegativeInteger">
   <xs:maxInclusive value="255" />
</xs:restriction>
</xs:simpleType>
```

Se pueden aplicar las facetas: length, maxLength, minLength, enumeration

```
<xs:simpleType name="ColorRGB">
   <xs:restriction base="ComponentesRGB">
     <xs:length value="3" />
   </xs:restriction>
 </xs:simpleType>
```

```
<color>255 255 0</color>
```

```
<xs:simpleType name="TipoNota">
<xs:union>
  <xs:simpleType>
    <xs:restriction base="xs:float">
      <xs:maxInclusive value="10" />
      <xs:minInclusive value="0" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="No presentado" />
    </xs:restriction>
  </xs:simpleType>
</xs:union>
</xs:simpleType>

<xs:element name="nota" type="TipoNota" />
```

```
<nota> 5.75 </nota>
```

```
<nota> No presentado </nota>
```



```
<xs:simpleType name="NIF">
 <xs:restriction base="xs:token">
   <xs:pattern value="\d{7,8}[A-Z]" />
 </xs:restriction>
</xs:simpleType>

<xs:element name="nif" type="NIF" />
```

```
<nif>9394173J</nif>
```

```
<nif>11079845M</nif>
```

Expresión	Posibles valores
Elemento \d	Elemento 2
a*b	b, ab, aab, aaab, …
[xyz]b	xb, yb, zb
a?b	b, ab
a+b	ab, aab, aaab, …
[a-c]x	ax, bx, cx

[a-c]x	ax, bx, cx
[^0-9]x	Carácter ≠ dígito seguido de x
\Dx	Carácter ≠ dígito seguido de x
(pa){2}rucha	paparucha
.abc	Cualquier carácter (1) seguido de abc
(a|b)+x	ax, bx, aax, bbx, abx, bax,...
a{1,3}x	ax, aax, aaax
\n	Salto de línea
\p{Lu}	Letra mayúscula
\p{Sc}	Símbolo de moneda

Similar a las subclases de POO. Consiste en añadir elementos a un tipo base

```
<xs:complexType name="Figura" >
  <xs:attribute name="color" type="Color"/>
 </xs:complexType>

<xs:complexType name="Rectángulo">
 <xs:complexContent>
 <xs:extension base="Figura">
 <xs:attribute name="base" type="xs:float" />
 <xs:attribute name="altura" type="xs:float" />
 </xs:extension>
 </xs:complexContent>
</xs:complexType>

<xs:complexType name="Círculo">
...similar pero incluyendo el radio
</xs:complexType>
```

Los tipos derivados pueden utilizarse en los mismos sitios que la clase base

```
<xs:element name="figuras">
   <xs:complexType>
     <xs:sequence>
      <xs:element name="figura" type="Figura"
                  maxOccurs="unbounded" />
     </xs:sequence>
   </xs:complexType>
  </xs:element>
```

```
<figuras>
<figura base="23" altura="3" xsi:type="Rectángulo" />
<figura radio="3" xsi:type="Círculo" />
</figuras>
```

Es necesario especificar el tipo mediante **xsi:type**

Mediante abstract="true" se declara un tipo como abstracto.

Ese tipo no puede usarse directamente

```
<xs:complexType name="Figura" abstract="true">
  <xs:attribute name="color" type="Color"/>
 </xs:complexType>
```

También es posible limitar la derivación de tipos

final="restriction"

```
<xs:complexType name="Círculo">
 <xs:attribute name="radio"
               type="xs:float"
               use="required" />

 <xs:attribute name="color"
               type="Color"
               default="255 0 0"/>

 <xs:attribute name="tipo"
               type="xs:string"
               fixed="jpeg" />
</xs:complexType>
```

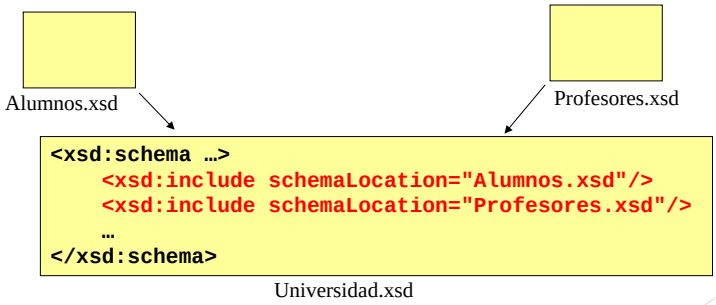
Por defecto los atributos son opcionales. Indicar que son obligatorios: **use="required"**

Valor por defecto de un atributo. Podría definirse otro valor.

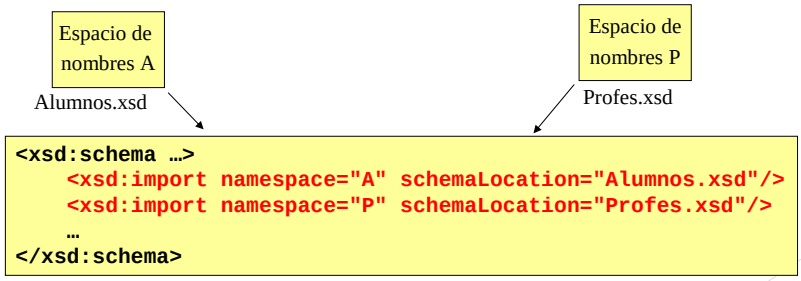
Valor fijo de un atributo. Si no se define, se utiliza ése. Si se define, debe coincidir.

include permite incluir elementos de otros esquemas

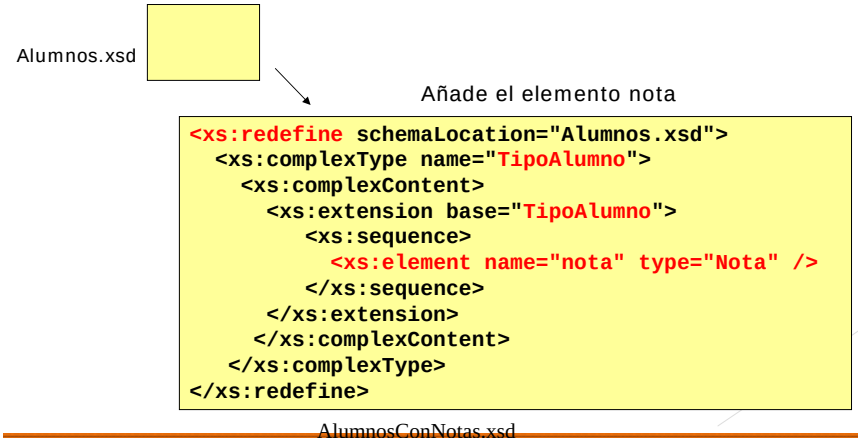
- Los elementos deben estar en el mismo espacio de nombres
- Es como si se hubiesen tecleado todos en un mismo fichero



import permite incluir elementos de otros esquemas con distintos espacios de nombres



redefine es similar a *include* pero permite modificar los elementos incluidos.



Los DTDs proporcionaban el atributo ID para marcar la unicidad (un valor ID era único en todo el documento)

XML Schema tiene más posibilidades:

- Indicar que un elemento es único (**unique**)
- Definir atributos únicos
- Definir combinaciones de elementos y atributos como únicos
- Distinción entre unicidad y claves (**key)**
  - Clave = además de ser único, debe existir y no puede ser nulo.
- Declarar el rango de un documento en el que algo es único


```
<xs:complexType name="Alumnos">
 <xs:sequence>
  <xs:element name="Alumno" type="TipoAlumno"/>
 </xs:sequence>
  <xs:key name="DNI">
    <xs:selector xpath="a:alumno"/>
    <xs:field xpath="a:dni"/>
  </xs:key>
</xs:complexType>
```

Es necesario incluir el espacio de nombres (XPath)

La clave puede formarse para atributos y elementos

```
<xs:key name="DNI">
   <xs:selector xpath="a:alumno"/>
   <xs:field xpath="a:nombre"/>
   <xs:field xpath="a:nombre"/>
</xs:key>
```

Una clave puede estar formada por varios elementos

```
<xs:complexType name="Alumnos">
 <xs:sequence>
  <xs:element name="Alumno" type="TipoAlumno"/>
 </xs:sequence>
  <xs:unique name="DNI">
    <xs:selector xpath="a:alumno"/>
    <xs:field xpath="a:dni"/>
  </xs:unique>
</xs:complexType>
```

Unique especifica que debe ser único, pero podría no existir

keyref especifica que debe hacer referencia a una clave (Claves Externas)

```
<xs:element name="clase">
 <xs:sequence>
  <xs:element name="alumnos" ...
  <xs:element name="delegado" ...
 </xs:sequence>

<xs:key name="DNI">
 <xs:selector xpath="a:alumnos/a:alumno"/>
 <xs:field xpath="a:dni"/>
</xs:key>

<xs:keyref name="Delegado" refer="DNI">
  <xs:selector xpath="a:delegado"/>
  <xs:field xpath="a:dni"/>
</xs:keyref>
```

Indicar que un elemento puede ser nulo sin estar vacío.

- Vacío (Empty): Un elemento sin contenido
- Nulo (Nil): Un elemento que indica que no hay valor

```
<xsd:element name="Persona">
  <xsd:complexType>
  <xsd:element name="nombre" type="xsd:NMTOKEN"/>
  <xsd:element name="primerApell" type="xsd:NMTOKEN"/>
  <xsd:element name="segundoApell" type="xsd:NMTOKEN"
                                   nillable="true"/>
 </xsd:complexType>
</xsd:element>
```

```
<persona>
      <nombre>John</nombre>
      <primerApell>Smith</primerApell>
      <segundoApell xsi:nil="true"/>
</persona>
```

El segundo apellido puede ser un NMTOKEN o estar indefinido

any indica cualquier contenido de un determinado espacio de nombres
anyAttribute cualquier atributo de un espacio de nombres

```
<xs:complexType name="Comentario">
 <xs:sequence>
  <xs:any namespace="http://www.w3.org/1999/xhtml"
          minOccurs="1"
          processContents="skip" />
 </xs:sequence>
 <xs:anyAttribute
      namespace="http://www.w3.org/1999/xhtml"/>
</xs:complexType>
```

```
<comentarios>
  <html:p>Es un
     <html:emph>Listillo</html:emph>
  </html:p>
</comentarios>
```

También puede declararse
##any, ##local, ##other

Otros valores
strict = obliga a validar
lax = valida si es posible

No soporta entidades. Mecanismo para crear macros

<!ENTITY &texto; "Esto texto se repite muchas veces" >

Es necesario seguir usando los DTDs ☹

Lenguaje de Restricciones limitado

Ejemplo: ¿Verificar valor total = suma de valores parciales?

Sensibilidad al contexto limitada

Por ejemplo: Especificar que el contenido depende del valor de un atributo

```
<transporte tipo="coche"> ...</transporte>
<transporte tipo="avión"> ...</transporte>
```

Tamaño de archivos XML Schema puede ser excesivo

Legibilidad de las especificaciones...XML no siempre es legible

Complejidad de la especificación:

Muchas situaciones/combinaciones excepcionales

Relax NG. Desarrollado por OASIS a partir de TREX y RELAX

Soporta mayor número de restricciones y gramáticas ambigüas

Incluye una sintaxis abreviada (no XML)

Schematron

Utiliza un modelo basado en reglas (en vez de gramáticas)

Asocia reglas de validación a expresiones XPath

Puede expresar restricciones arbitrarias

Lenguajes para XML (Encaje de patrones con expresiones Regulares)

XDuce, CDuce, HydroJ

Creación ficheros XML y validación mediante Esquemas
Herramientas:

- xsv (http://www.ltg.ed.ac.uk/~ht/xsv-status.html)
  - Herramienta desarrollada en Python
  - Funcionamiento a través de Web o en línea de comandos
  - Salida en formato XML (difícil de leer al principio)

- Xerces (Apache)
  - Librerías XML en Java y C++
  - Contiene diversas utilidades de prueba
  - Ejemplo: SAXCount cuenta el número de elementos pero también valida el Schema:

```
SAXCount -v=always -s  -n fichero.xml
```


Diseño de Vocabularios XML

Separación tradicional de dos mundos

- Sistemas orientados a Datos
  - Información uniforme y fuertemente estructurada (ej. Tablas)
  - Mucha cantidad de información repetida
  - Objetivo: Procesamiento eficiente (Almacenes de datos)
- Sistemas orientados a Documentación
  - Información poco uniforme y entrelazada (ej. Libros)
  - No existe un patrón uniforme
  - Objetivo: Comunicación, Presentación (Mensajes)

Se podría añadir un tercer mundo:

- Programación Orientada a Objetos
- Propuestas para añadir capacidad de programación a documentos

XML: Información semi-estructurada (Lugar intermedio)

- Estructuras jerárquicas entrelazadas

Características a tener en cuenta

- Tamaño de documentos
- Facilidad de escritura
- Facilidad de procesamiento
- Flexibilidad (ej. HTML es muy flexible, Bases de Datos = menos)
- Consistencia: Evitar características incoherentes
- Nivel de abstracción: Buscar término medio en nivel de detalle

```
<fecha>10 Marzo 2003</fecha>
<fecha><día>10</dia><mes>Marzo</mes><año>2003</año></fecha>
```

Patrones de diseño:

- **www.xmlpatterns.com**

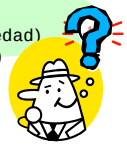
Representación de propiedades

```
<pizza
   nombre="Margarita"
   precio="6" />
```

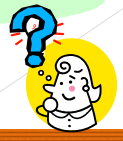
¿Atributos o Elementos?

```
<pizza>
   <nombre>Margarita </nombre>
   <precio>6</precio>
</pizza>
```

Razones filosóficas:
Atributos: valores asociados con objetos sin identidad propia (edad)
Subelementos: valores con identidad propia (fecha-nacimiento)



Orígenes (SGML):
Atributos: meta-información (información sobre el contenido)
Subelementos: Contenido



Diseño de Vocabularios XML
Ejemplo de Discusión

Representación de propiedades

```
<pizza
   nombre="Margarita"
   precio="6" />
```

¿Atributos o Elementos?

```
<pizza>
   <nombre>Margarita </nombre>
   <precio>6</precio>
</pizza>
```

En los DTDs
Pueden incluirse restricciones sobre su valor
Ej. valor "si" o "no"
Pueden definirse valores por defecto
Pueden validarse los valores ID e IDREF
Pueden definirse restricciones sobre espacios
en blanco (NMTOKENS)
Ocupan menos espacio
Más fáciles de procesar (SAX y DOM)
Acceso a entidades externas (datos binarios)

Soportan valores arbitrariamente complejos y repetidos
Establecen un orden
Soportan *atributos de atributos*
Mayor flexibilidad ante modificaciones

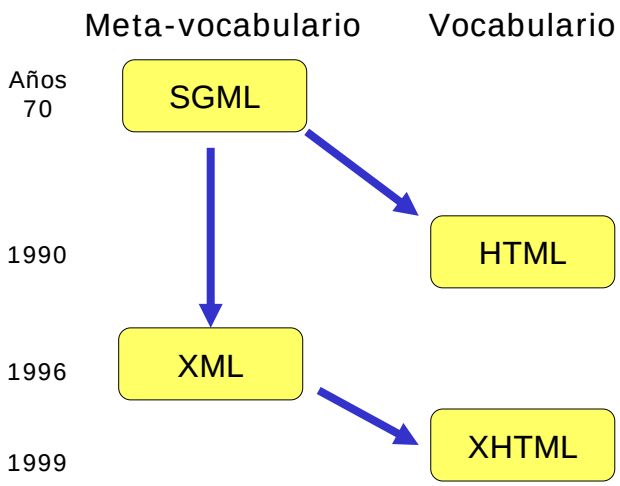
Jose Emilio Labra Gayo, Nov. 2005 www.fundacionctic.org

...Aparición de una nueva torre de Babel...

Algunos Consejos:

- Estudiar dominio de la Aplicación (ver estándares ya definidos!!!)
- Considerar futuras ampliaciones (extensibilidad)
- Validar antes de que sea tarde
- Usar espacios de nombres
- etc. etc.

Ejemplos de Vocabularios XML



(1999) XHTML 1.0 consiste en adaptar HTML para que sea un vocabulario XML

- Principales diferencias:
  - Todas las etiquetas deben cerrarse (XML bien formado)
  - Los nombres de elementos deben ser minúsculas

Nuevas versiones:

- (2000)XHTML Basic = Subconjunto para pequeños dispositivos
- (2000) Modularización de XHTML: XHTML 1.1
- (2003) XML Events. Gestión de eventos

En desarrollo:

- XHTML 2.0, XFrames, XML Print, HLink...

Structure: body, head, html, title

Text: abbr, acronym, address, blockquote, br, cite, code, dfn, div, em, h1, h2, h3, h4, h5, h6, kbd, p, pre, q, samp, span, strong, var

Hypertext: a

List: dl, dt, dd, ol, ul, li

Object: object, param

Presentation: b, big, hr, i, small, sub, sup, tt

Edit: del, ins

Bidirectional Text: bdo

Forms: button, fieldset, form, input, label, legend, select, optgroup, option, textarea

Table: caption, col, colgroup, table, tbody, td, tfoot, th, thead, tr

Image: img

Client-side Image Map: area, map

Server-side Image Map: Atributo *ismap* de *img*

Intrinsic Events: Atributos de sucesos

Metainformation: meta

Scripting: noscript, script

Stylesheet : elemento style

Style Attribute : atributo style

Link : link

Base: base

(1994) Primer borrador de CSS

Objetivo: Permitir combinar preferencias visuales del autor y del usuario (*en cascada)*

(1996) CSS nivel 1: Propiedades de fuentes, márgenes, colores, etc.

(1998) CSS nivel 2: Añade posiciones absolutas, páginas, numeración automática, etc.

En desarrollo CSS 3, añadirá selectores, texto vertical, interacción, etc.

Otros perfiles de CSS para móviles, TV e impresión

CSS no tiene sintaxis XML

XHTML+CSS: Ejemplo

AlCapone.html

```
<html>
<head>
<title>Pizzeria Al Capone</title>
<link rel="stylesheet" href="pizzeria.css">
</head>
<body>
<h1>Pizzería Al Capone</h1>
<p>Lista de enlaces</p>
<ul>
<li><a href="Pizzas.html">
        Tipos de Pizzas</a></li>
<li><a href="http://www.mafia.it">
        Patrocinadores</a></li>
<li><a href="#Contacto">
        Contacto</a></li>
<ul>

<h2 ><a name="Contacto">Contacto</a></h2>
<p><span class="item">Dirección:</span>
        C/ Génova Nº 3, Oviedo, España</p>
<p><span class="item">Teléfono:</span>
        985203040</p>
</body>
</html>
```

Enlace a hoja de estilo

Sin aspectos visuales

Identificación elementos

pizzeria.css

```
body { color : yellow;
       background: blue
     }
a:link { color: red }
a:visited { color: white }
span.item { color : red }
```

Página visualizada

XHTML+CSS: Ejemplo

pizzas.html

```
<html>
<head>
<title>Tipos Pizzas</title>
<link rel="stylesheet" href="pizzeria.css">
</head>
<body>
<h1>Pizzas del Restaurante Al Capone</h1>
<table><caption>Tipos de Pizzas</caption>
<th>
<td>Pizza</td><td>Ingredientes</td><td>Precio</td></th>
<tr><td>Barbacoa</td>
    <td>Salsa barbacoa, Mozzarella, Pollo, Bacon, Ternera</td>
    <td>8 &euro;</td></tr>
. . .
<tr><td>Margarita</td>
    <td>Tomate, Jamón, Queso</td>
    <td>6 &euro;</td>
</tr>
</table>
</body>
</html>
```

Referencia a la misma hoja de estilos

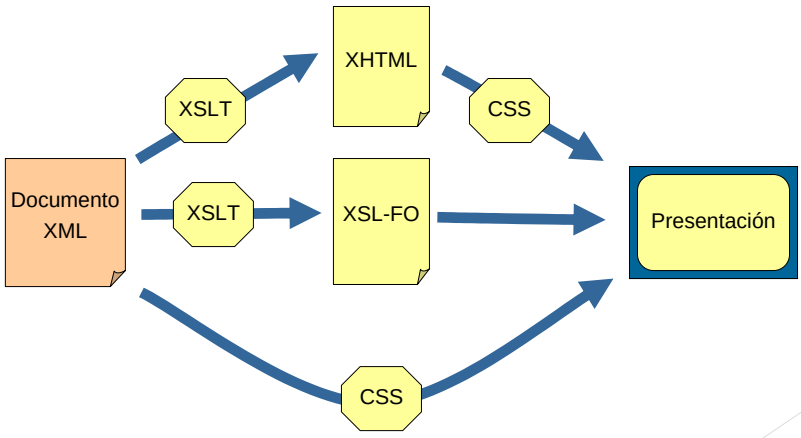
Misma apariencia

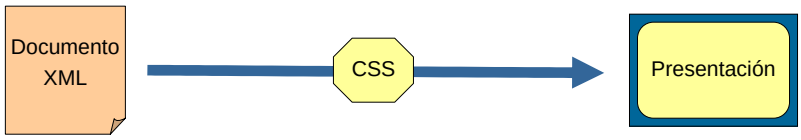
SGML tenía DSSSL (Document Style Semantics and Specification Language)

Para XML se optó por crear XSL (XML Stylesheet Language)

- Posteriormente se dividió en 3 partes:
- XSLT: Transformación de documentos XML
- XPath: Especificar caminos y expresiones XML
- XSL-FO: Objetos de formateo

Además, CSS también puede usarse con XML





mensaje.xml

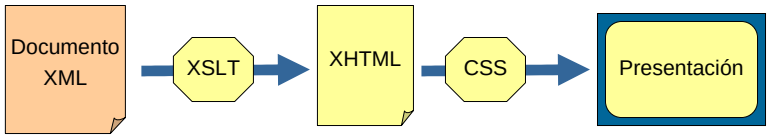
```
<?xml version="1.0" encoding="iso-8859-1" ?>
<?xml-stylesheet type="text/css"
                 href="mensaje.css" ?>
<mensaje>
<destino>Juanito el Loco</destino>
<remitente>María la Impaciente</remitente>
<texto>
Recuerda que nos tenemos que ver en la alacena...
</texto>
</mensaje>
```

mensaje.css

```
mensaje { border: 5px solid; ...}
destino { color: blue; ... }
remitente { color: red; ... }
texto { margin: 5em; ... }
```


Facilidad de uso


Posibilidades limitadas



Es la técnica más popular
Permite añadir características de hipertexto e interactivas
Menor calidad para medios impresos

Originalmente, XSL = XSLT + XPath + XSL-FO
Posteriormente, XPath y XSLT toman identidad propia
XSL-FO = Objetos de formateo con propiedades
Muchas propiedades, son comunes con CSS

(2001) XSL-1.0 (Incluía XPath y XSLT)

Incluye modelos de páginas, soporte para internacionalización e hiper-enlaces.

(2003) XSL-1.1

Mayor soporte para marcadores, índices y múltiples flujos

En desarrollo:

XPath 2.0: Añade soporte para tipos de datos y Schemas
XSLT 2.0: Tipos de datos de XPath 2.0 y gestión de errores

XSLT es un lenguaje declarativo (transforma un árbol en otro árbol)
El programador incluye una serie de reglas de transformación
El procesador es el que se encarga de obtener el árbol y de escribir el resultado
Las reglas se basan en la definición de plantillas (*templates)*

Las plantillas utilizan sintaxis de XPath

```
<xsl:template match="valor a encajar">
   código de salida
</xsl:template>
```

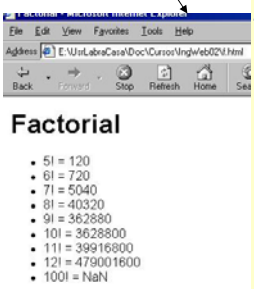
```
<xsl:stylesheet version="1.0"
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" />

<xsl:template match="/">
  <html><body><h1>Pizzas del Restaurante Al Capone</h1>
   <xsl:apply-templates />
  </body></html>
</xsl:template>

<xsl:template match="pizzas">
<table><caption>Tipos de Pizzas</caption><tr>
<xsl:apply-templates />
</table>
</xsl:template>

<xsl:template match="pizza">
<tr><td><xsl:value-of select="@nombre"/></td>
<td><xsl:apply-templates /></td>
<td><xsl:value-of select="@precio" /></td></tr>
</xsl:template>

<xsl:template match="ingrediente"><xsl:value-of select="@nombre" />
</xsl:template>
</xsl:stylesheet>
```

Valores que se incluyen en resultado

Patrón de encaje

Referencia a valor de atributo


```
<ns>
<num>5</num>
<num>6</num>
<num>7</num>
<num>8</num>
<num>9</num>
<num>10</num>
<num>11</num>
<num>12</num>
<num>100</num>
</ns>
```

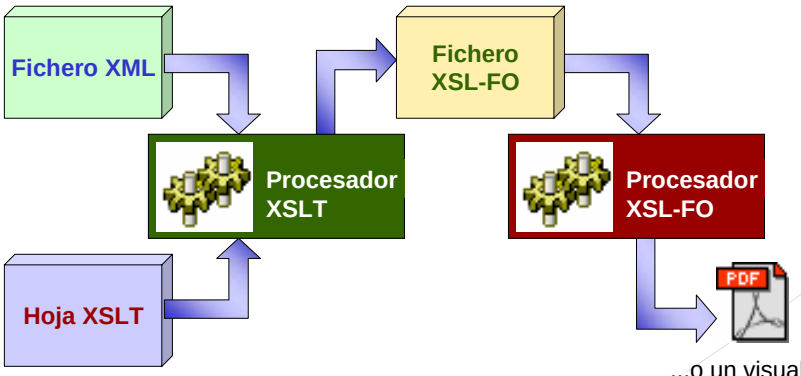



```
<xsl:template match="num">
 <li>
 <xsl:value-of select="."/>! =
 <xsl:call-template name="fact">
   <xsl:with-param name="x"><xsl:value-of select="." />
   </xsl:with-param>
 </xsl:call-template>
 </li>
</xsl:template>

<xsl:template name="fact">
 <xsl:param name="x" />
  <xsl:choose> <xsl:when test="$x = 0">1</xsl:when>
  <xsl:otherwise>
    <xsl:variable name="llamada">
     <xsl:call-template name="fact">
      <xsl:with-param name="x"><xsl:value-of select="$x - 1" />
       </xsl:with-param>
     </xsl:call-template>
    </xsl:variable>
    <xsl:value-of select="$llamada * $x"/>
  </xsl:otherwise>
  </xsl:choose>
</xsl:template>
```

fact x = if x = 0 then 1
else x * fact (x – 1)

XSL-Formatting Objects

Describe documento en un formato "imprimible" (presentación)



...o un visualizador...

Sintaxis XML

Muchas propiedades compatibles con CSS

Otras posibilidades no contempladas en CSS

- Texto de derecha a izquierda o de arriba a abajo
- Notas al pie
- Notas al margen
- Números de página y referencias cruzadas
- ...

```
<?xml version="1.0" encoding="iso-8859-1"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <fo:layout-master-set>
    <fo:simple-page-master master-name="PáginaEjemplo">
      <fo:region-body margin="2cm"/>
    </fo:simple-page-master>
  </fo:layout-master-set>

  <fo:page-sequence
                  master-reference="PáginaEjemplo">
    <fo:flow flow-name="xsl-region-body">
      <fo:block>Hola en FO</fo:block>
    </fo:flow>
  </fo:page-sequence>
</fo:root>
```

- HTML carece de facilidades para incorporar fórmulas matemáticas
  - Se recurría a incluirlas como imágenes
  - Múltiples problemas:
    - Fórmula como algo indivisible
    - No es posible adaptar a diferentes formatos visuales
    - Procesamiento de fórmulas: buscadores, índices, reutilización, etc
- (1999) MathML 1.0
- (2001) MathML 2.0: nuevos elementos y DOM

Material matemático a todos los niveles
Codificar tanto notaciones como significados
Facilitar conversión con otros formatos
Facilitar la visualización de expresiones complejas
Permitir la extensibilidad
Legible por personas...por ser XML pero...
NO está pensado para edición manual de fórmulas

2 estilos
- Presentación: Estructura visual en 2 dimensiones
- Contenido: Significado de las fórmulas

Presentación

```
<math>
<mrow>
  <msup>
    <mfenced>
      <mrow>
        <mi>a</mi>
        <mo>+</mo>
        <mi>b</mi>
      </mrow>
    </mfenced>
    <mn>2</mn>
  </msup>
 </mrow>
</math>
```

$(a+b)^2$

Contenido

```
<math>
 <apply>
   <power/>
    <apply>
      <plus/>
      <ci>a</ci>
      <ci>b</ci>
    </apply>
   <cn>2</cn>
 </apply>
</math>
```

```
<math xmlns="http://www.w3.org/1998/Math/MathML">
 <mrow>
   <mi>x</mi><mo>=</mo>
   <mfrac>
     <mrow><mrow><mo>-</mo><mi>b</mi></mrow>
           <mo>±</mo>
      <msqrt><mrow><msup><mi>b</mi><mn>2</mn></msup>
                 <mo>-</mo>
           <mrow><mn>4</mn><mo>&InvisibleTimes;</mo>
                 <mi>a</mi><mo>&InvisibleTimes;</mo><mi>c</mi></mrow>
           </mrow>
       </msqrt></mrow>
     <mrow><mn>2</mn><mo>&InvisibleTimes;</mo><mi>a</mi></mrow>
   </mfrac>
  </mrow>
 </math>
```

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

(2001) SVG 1.0 - Scalable Vector Graphics
(2003) SVG 1.1 Modularización
...actualmente: SVG 1.2 en desarrollo

Objetivos
- Gráficos vectoriales: Precisión, escalabilidad, etc.
- Compatibilidad con XML y vocabularios de la Web: CSS, Espacios de nombres, XLink, SMIL, ECMAScript, etc.
- También permite incluir texto, imágenes *raster* e hiper-enlaces
- Formato de texto (no binario): Facilita indexación, búsquedas, etc.

Buena acogida
- Soportado en principales navegadores: IE, Mozilla, Amaya, etc.
- Planes para incorporación en pequeños dispositivos

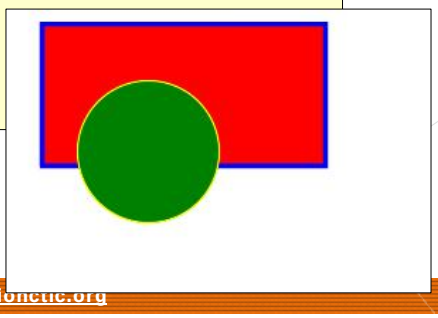
Formato *raster*

Arrays de pixels. Al hacer zoom se pierde calidad

SVG = Formato vectorial

Al hacer zoom no se pierde calidad


```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20010904//EN"
 "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">

<svg width="300" height="200"
     xmlns="http://www.w3.org/2000/svg">

<rect x="25" y="10" width="200" height="100"
    fill="red" stroke="blue" stroke-width="3" />

<circle cx="100" cy="100" r="50"
    fill="green" stroke="yellow"/>

</svg>
```



SMIL = Synchronized Multimedia Integration Language

SMIL es a multimedia lo que HTML es a hipertexto

Objetivo: Integrar/sincronizar elementos de diferentes medios: vídeos, imágenes, sonidos, etc.

(1998) SMIL 1.0

(2001) SMIL 2.0 Creación de diferentes módulos

Combinación en otras aplicaciones:

XHTML+SMIL

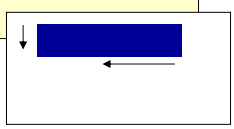
SVG+SMIL: Las animaciones de SVG forman parte de SMIL

etc...

SMIL puede combinarse con SVG para realizar animaciones

Ejemplo, modificar valores de atributos

```
<rect x="20" y="10" width="120" height="40" fill="blue">
  <animate attributeName="width"
           from="120" to="20" begin="0s" dur="4s" fill="freeze"/>
  <animate attributeName="height"
           from="40" to="100" begin="2s" dur="6s" fill="freeze"/>
</rect>
```



SMIL también puede combinarse con XHTML o utilizarse en documentos independientes

Definir escenas de realidad virtual en Internet

Adaptación de VRML (Virtual Reality Modeling Language) a XML

Evolución:

(1994) Posibilidad de desarrollar un estándar para realidad virtual en Internet

- Aparecen VRML 1.0 y VRML 2.0

(1997) VRML 97 = Estándar ISO Internacional

- Objetivos: Independencia de plataforma, extensibilidad, bajo ancho de banda

(1999) Se cambia el nombre de Consorcio VRML a consorcio Web3D

(2003) Desarrollo de X3D

- Conversión a sintaxis XML
- Modularización
- Características: Gráficos en 2D y 3D, Animación, Audio/Vídeo, Interacción con el usuario, *scripting,* simulaciones físicas: comportamientos humanos, espacios geográficos, etc.


```
<?xml version="1.0" encoding="UTF-8"?>
<X3D profile="Immersive">
<Scene><Group>
<Viewpoint description="Hola" orientation="0 1 0 1.57"
        position="6 -1 0"/>
<NavigationInfo type="&quot;EXAMINE&quot; &quot;ANY&quot;"/>
<Shape>
  <Sphere/>
     <Appearance><ImageTexture url="&quot;tierra.png&quot;"/>
        </Appearance>
  </Shape>
  <Transform rotation="0 1 0 1.57"
             translation="0 -2 1.25">
   <Shape>
     <Text
       string="&quot;Hola&quot;
               &quot;en X3D!&quot;"/>
     <Appearance>
       <Material diffuseColor="0.1 0.5 1"/>
     </Appearance>
   </Shape>
  </Transform>
 </Group>
</Scene></X3D>
```

Lenguaje de marcas para representar información en dispositivos con pocos recursos (teléfonos móviles)

WML forma parte de WAP

(1995) Ericsson inicia ITTP (Intelligent Terminal Transfer Protocol)

(1996) Openwave desarrolla HDML (Handhelp Device Markup Language) subconjunto de HTML

(1997) Ericsson, Motorola, Nokia y Openwave fundan WapForum

(1998) WAP 1.0. Es un protocolo que permite acceso a Internet desde dispositivos móviles

(2002) Se crea Open Mobile Alliance


```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
      <card id="Tarjeta1" title="Ejemplo ">
      <p> Hola</p>
      <p>Qué tal?</p>
      <anchor>Siguiente
         <go href="#siguiente">
      </anchor>
      </card>

      <card id="siguiente" title="siguiente">
      <p>Final</p>
      </card>
</wml>
```

Portales basados en voz

- Ejemplos: Contestadores automáticos de empresas
  - Artefactos empotrados (coches)

Objetivos:

- Fácil creación de contenido hablado
- Reconocimiento/generación de voz
- Interacción con el usuario

(1995) Phone Markup Language de AT&T

(1998) Se crea el VoiceXML Forum

(2000) VoiceXML 1.0

(2004) VoiceXML 2.0


```
<vxml version="2.0">
<form id="start">
  <field name="answer">
    <noinput> Hey, don't sleep! </noinput>
    <nomatch> say 'yes' or 'no' </nomatch>
    <prompt> Are you sleepy? </prompt>
    <grammar root="main">
      <rule id="main" scope="public">
        <one-of>
          <item><ruleref uri="#yes" tag="yes"/></item>
          <item><ruleref uri="#no" tag="no"/></item>
        </one-of>
      </rule>
      <rule id="yes">
        <one-of><item>yes</item><item>yeah</item></one-of>
      </rule>
      <rule id="no">
        <one-of><item>no</item><item>not</item></one-of>
      </rule>
    </grammar>
    <filled>
      <if cond="answer=='yes'">So you are sleepy. Me too.
      <else/>So you are not sleepy. But I am.
      </if>
    </filled>
  </field>
</form>
</vxml>
```

Otros vocabularios XML

Existe una gran cantidad de vocabularios XML para diferentes dominios

Los vocabularios suelen denominarse “Aplicaciones XML”

Ejemplos:

- XBRL (eXtensible Buisiness Reporting Language) Información financiera)
- DocBook (Proceso de documentos)
- CML (Chemical Markup Language)
- UPnP (Universal Plug and Play)

...

Otros:

http://xml.coverpages.org/xmlApplications.html

Selección de Enlaces

- Página del consorcio: **http://www.w3c.org**
- En español: **http://www.it.uc3m.es/~xml/enlaces.html**
- Especificación anotada:
  http://www.xml.com/axml/testaxml.htm
- XML en industria: **http://www.xml.org**
- Diseño de vocabularios XML: **http://www.xmlpatterns.com**
- Tutoriales: **http://www.w3schools.com**
  http://www.brics.dk/~amoeller/XML/
- Artículos de XML:
  http://www.topxml.com
  http://www.xmlpatterns.com
- Software de XML
  http://www.xmlsoftware.com
  http://www.xmlhack.com
  http://www.garshol.priv.no/download/xmltools/