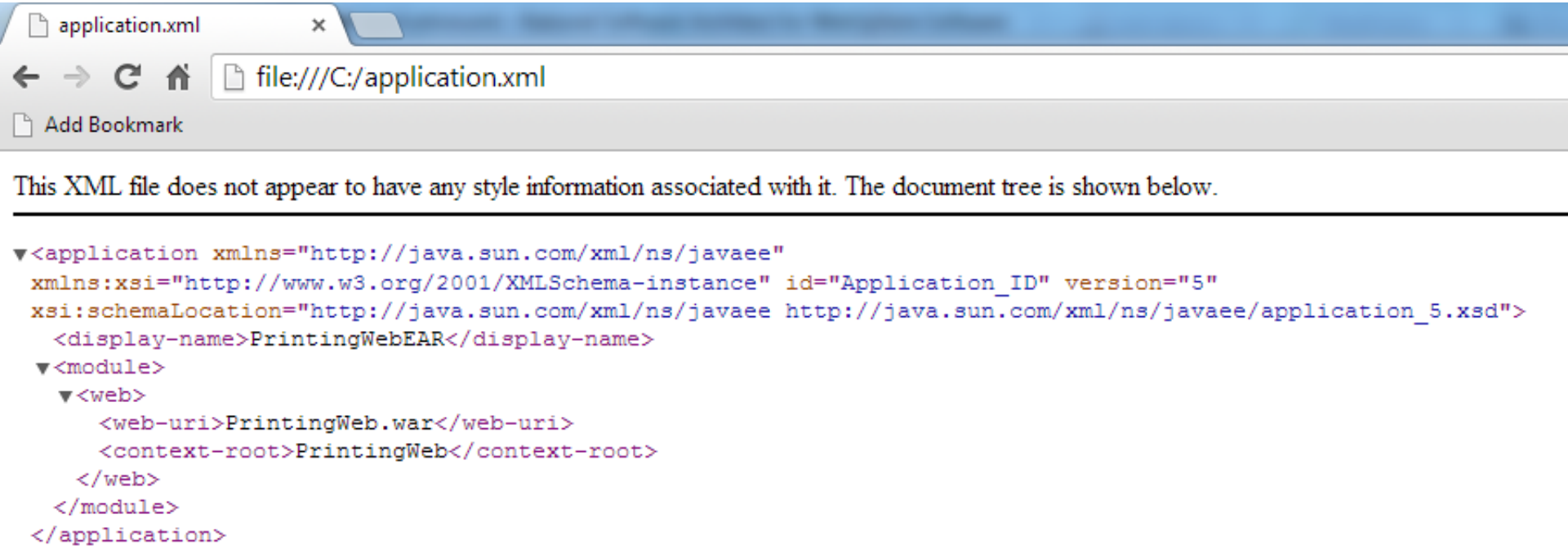
Visualización y Transformaciones en XML

Visualización

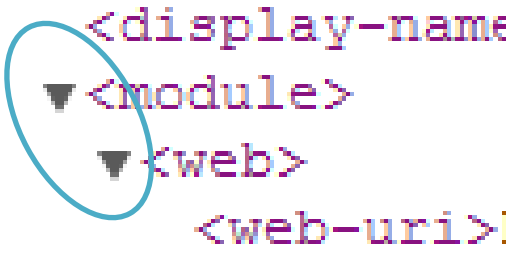
- Los archivos XLM pueden ser vistos prácticamente en cualquier browser



Visualización

- Los XML en los web browsers no se despliegan como páginas HTML

- Se muestran con símbolos de colapsar/descolapsar para contraer secciones del árbol



Visualización

- Para ver el XML plano, se debe dar botón derecho y luego “view source”
- Si el XML no es valido, el browser reportará un error

This page contains the following errors:

```
error on line 14 at column 16: Opening and ending tag mismatch: display-name line 0 and application
```

Below is a rendering of the page up to the first error.

Visualización

- Por qué se ven así?
  - XML no llevan información de visualización
    - Llevan datos
  - Como los tags son inventados, un browser no puede saber si <table> representa una tabla HTML o una tabla para apoyar cosas
- Solución?
  - CSS
  - XSLT
  - JavaScript

XML CSS

- CSS agrega información de visualización que es entendible por los web browsers

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="catalogo.css"?>
<catalogo>
   <cd>
      <titulo>Empire Burlesque</titulo>
      <artista>Bob Dylan</artista>
      <pais>USA</pais>
      <fecha>1985</fecha>
   </cd>
   <cd>
      <titulo>Hide your heart</titulo>
      <artista>Bonnie Tyler</artista>
      <pais>UK</pais>
      <fecha>1988</fecha>
   </cd>
</catalogo>
```

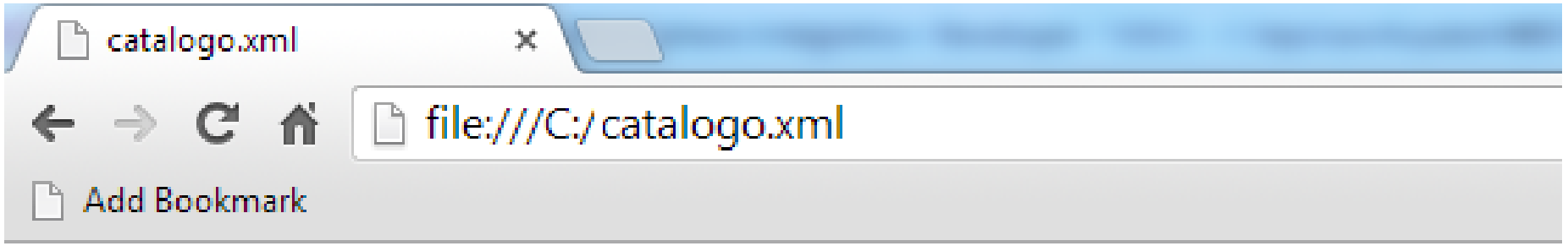
XML CSS

- catalogo .css

```
titulo {
    color: green;
}
pais {
    color: blue;
}
```

XML CSS

- En un browser



XSLT

- Formatear XML con CSS no es lo mas común
  - W3C recomienda XSLT

- XSLT? e**X**tensible **S**tylesheet **L**anguage **T**ransformations
  - Parte de XSL
    - Incluye vocabulario para especificar formato
  - Especifica como se transforma un XML en otro
  - Independiente de XSL

XSLT

- Con XSLT se puede transformar un XML en un HTML antes de ser desplegado en un web browser
  - La transformación XSLT es hecha por el web browser al leer el XML
  - Diferentes browsers pueden producir diferentes resultados
    - Para evitar este problema, la transformación debería ser hecha en el servidor

XSLT

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:template match="/">
      <html><body>
         <h2>Mi colección de cds</h2>
         <table border="1">
            <tr bgcolor="#9acd32">
               <th>Titulo</th>
               <th>Artista</th>
            </tr>
            <xsl:for-each select="catalogo/cd">
               <tr>
                  <td><xsl:value-of select="titulo"/></td>
                  <td><xsl:value-of select="artista"/></td>
               </tr>
            </xsl:for-each>
         </table>
      </body></html>
   </xsl:template>
</xsl:stylesheet>
```

XSLT

- Resultado

Mi colección de cds

Titulo	Artista
Empire Burlesque	Bob Dylan
Hide your heart	Bonnie Tyler
Greatest Hits	Dolly Parton
Still got the blues	Gary Moore
Eros	Eros Ramazzotti
One night only	Bee Gees
Sylvias Mother	Dr.Hook
Maggie May	Rod Stewart
Romanza	Andrea Bocelli
When a man loves a woman	Percy Sledge
Black angel	Savage Rose
1999 Grammy Nominees	Many

XSLT

- Otras características de XSLT
  - Directivas
    - <template>
    - <for-each>
    - <sort>
    - <if>
    - <choose>
  - Ejecución en cliente
  - Ejecución en servidor
  - Funciones XSLT

Consultas sobre XML

XML DOM

- **D**ocument **O**bject **M**odel

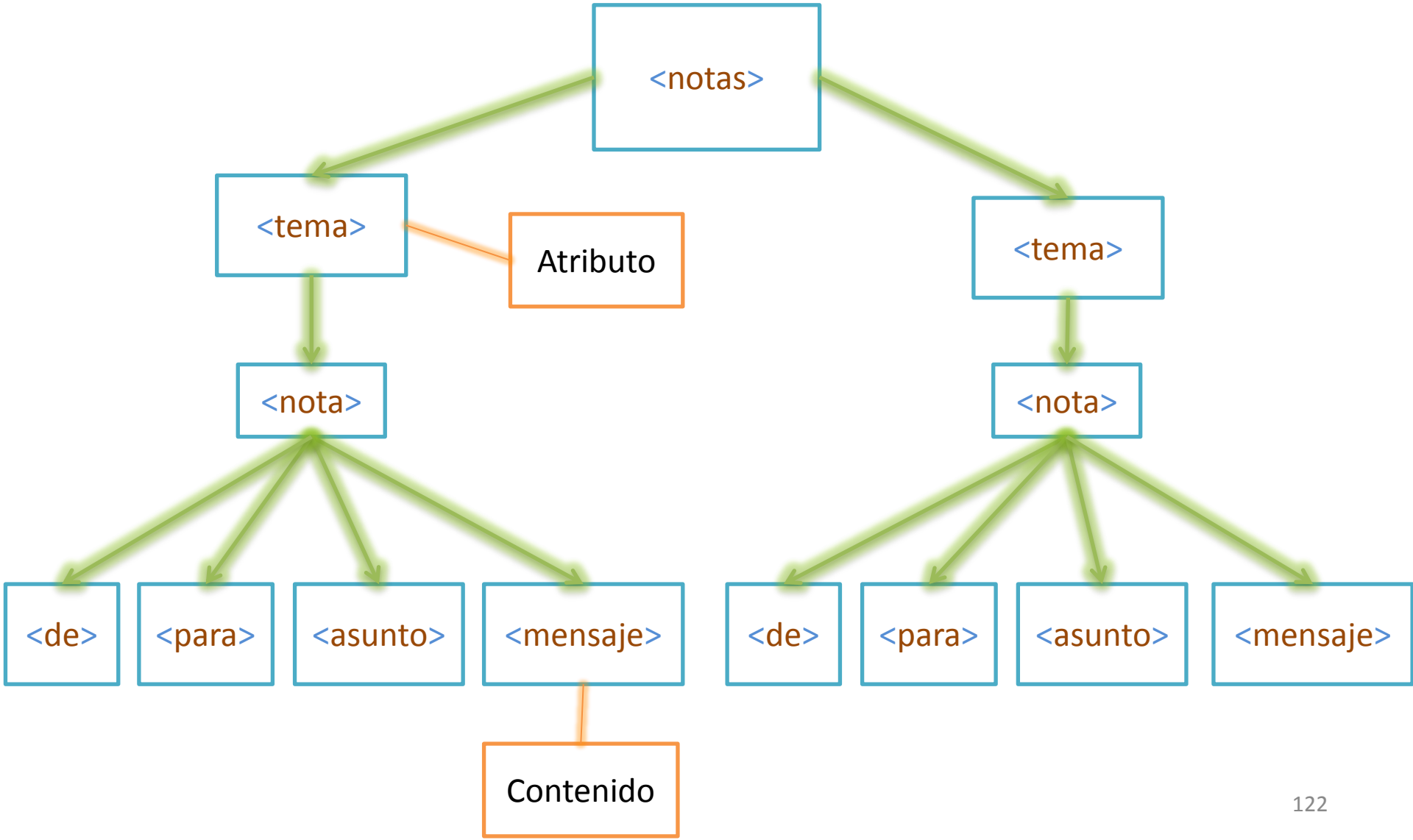
- Un DOM define una forma estándar para acceder y manipular documentos
  - XML DOM

- XML DOM ve un documento XML como una estructura de árbol

XML DOM

```
<?xml version="1.0"?>
<notas>
  <tema valor="Tecnólogo">
    <nota>
      <de>Juan</de>
      <para>Pedro</para>
      <asunto>Créditos asignatura XML</asunto>
      <mensaje>Son 4 créditos</mensaje>
    </nota>
  </tema>
  <tema valor="Personales">
    <nota>
      <de>Juan</de>
      <para>Maria</para>
      <asunto>Sábado</asunto>
      <mensaje>¿Salimos a tomar algo?</mensaje>
    </nota>
  </tema>
</notas>
```

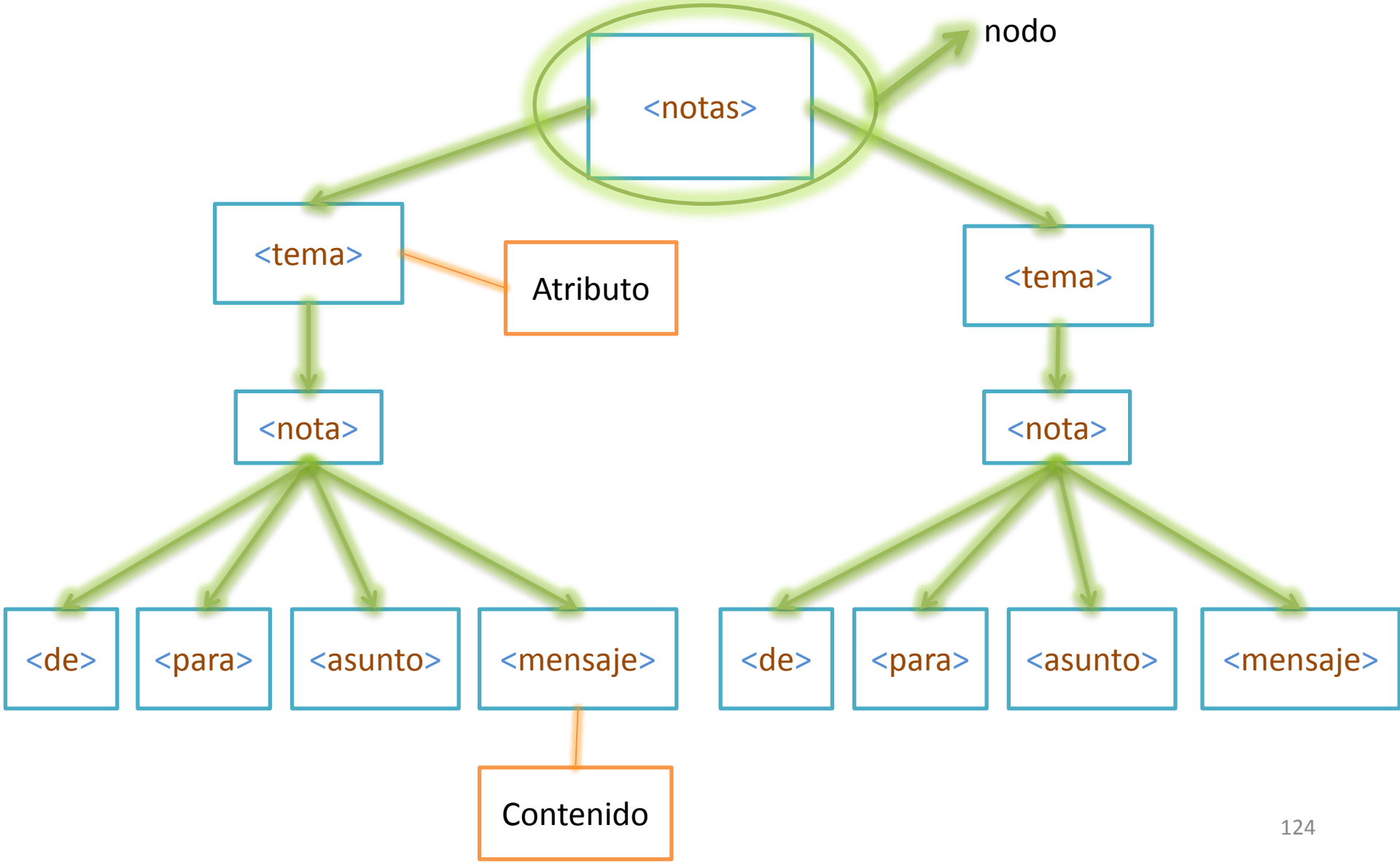
XML DOM


<notas>
<tema>
Atributo
<tema>
<nota>
<nota>
<de>
<para>
<asunto>
<mensaje>
<de>
<para>
<asunto>
<mensaje>
Contenido

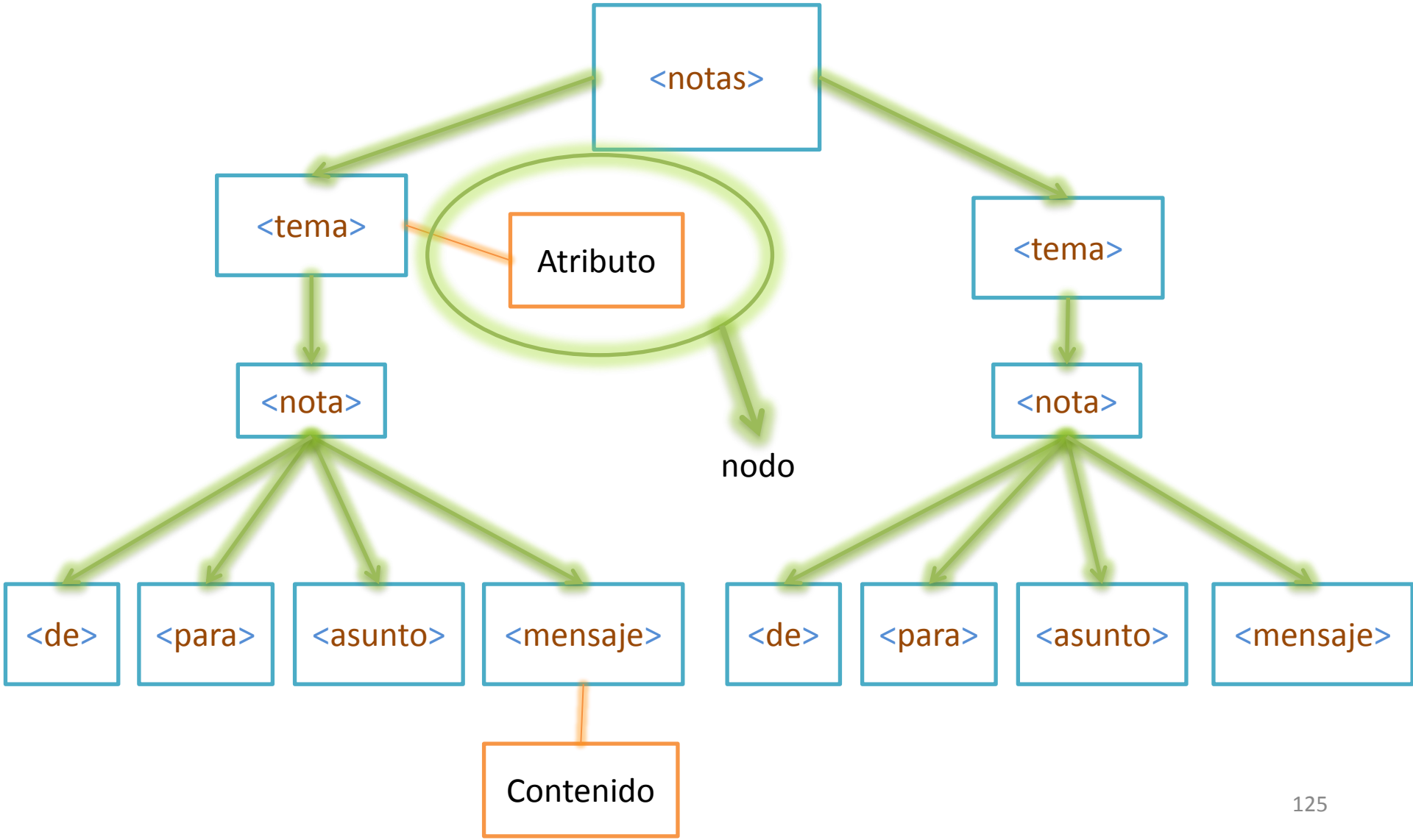
XML DOM

- Todos los elementos del documento XML pueden ser accedidos a través del árbol DOM
- Mediante XML DOM se pueden hacer consultas al XML, modificaciones, borrados y agregados
- En XML DOM, atributos, elementos, contenidos son todos vistos como nodos

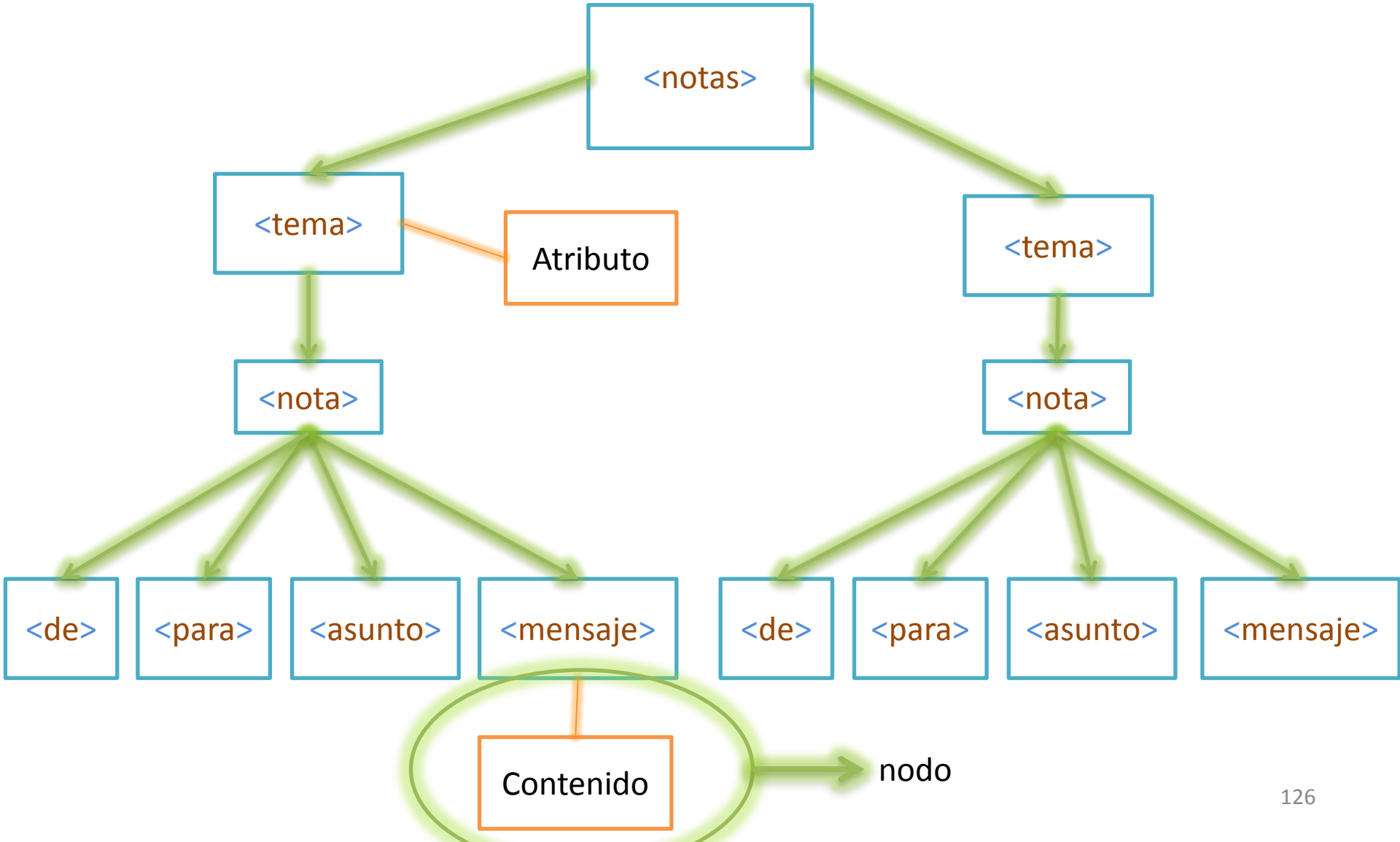
XML DOM


nodo
<notas>
<tema>
Atributo
<tema>
<nota>
<nota>
<de>
<para>
<asunto>
<mensaje>
<de>
<para>
<asunto>
<mensaje>
Contenido

XML DOM


<notas>
<tema>
Atributo
<tema>
<nota>
nodo
<nota>
<de>
<para>
<asunto>
<mensaje>
<de>
<para>
<asunto>
<mensaje>
Contenido

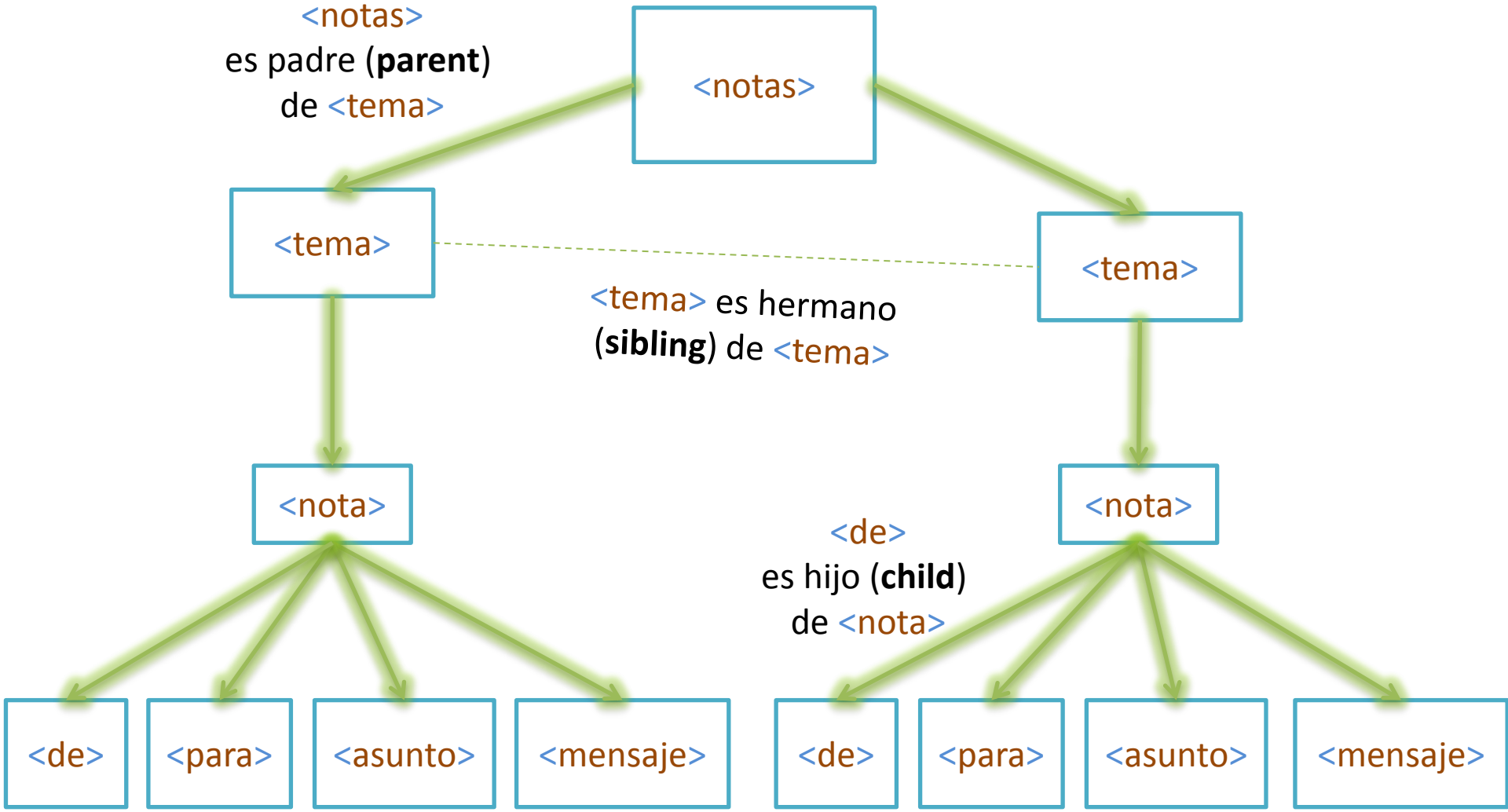
XML DOM


<notas>
<tema>
Atributo
<tema>
<nota>
<nota>
<de>
<para>
<asunto>
<mensaje>
<de>
<para>
<asunto>
<mensaje>
Contenido
nodo

XML DOM

- XML DOM ve a los nodos con una relación jerárquica entre ellos (padres, hermanos e hijos)

XML DOM y el árbol XML



XML DOM API

- La API XML DOM está formada por propiedades y métodos
- Dependen del lenguaje pero tienen cosas en común
  - String *node*.nodeName
    - Devuelve el nombre el nodo
    - Es el mismo nombre del tag, del nombre del atributo, #text o #document
    - Read-only

XML DOM API

- String *node*.nodeValue
  - Devuelve el valor del nodo
  - Cuando es un elemento -> undefined
  - Cuando es texto -> el contenido
  - Cuando es un atributo -> el valor del atributo
- String nodeType
  - Devuelve el tipo de nodo
  - Read-only

Node type	NodeType
Element	1
Attribute	2
Text	3
Comment	8
Document	9

XML DOM API

- Node *node*.parentNode
  - Devuelve el nodo padre del nodo actual
- NodeList *node*.childNodes
  - Devuelve el conjunto de nodos hijos del nodo actual
- NamedNodeMap *node*.attributes
  - Devuelve el conjunto de atributos del nodo actual
- NodeList *node*.getElementsByTagName(*name*)
  - Devuelve el conjunto de nodos de nombre especificado
- *node*.appendChild(*node2*)
  - Inserta un nuevo nodo hijo (*node2*) al nodo actual
- *node*.removeChild(*node2*)
  - Remueve un nodo hijo (*node2*) al nodo actual

XML DOM API

- NodeList
  - Es un array de nodos
  - Propiedad length
  - Se accede posicionalmente por indice
    - Comienza en 0
- NamedNodeMap
  - Lista de nodos que son atributos
  - Similar a NodeList pero con algunas diferencias
    - Se actualiza directamente con los cambios en el nodo

XML DOM API

- Como navegar un DOM
  - parentNode
  - childNodes
  - firstChild
  - lastChild
  - nextSibling
  - previousSibling

XML Parsers DOM y SAX

- SAX (**S**imple **A**PI for **X**ML) es una alternativa a DOM
- DOM opera sobre el documento XML como un todo y sigue una especificación formal
  - Estado-independiente
- SAX operan sobre cada nodo de forma secuencial y no existe una especificación formal
  - Estado-dependiente

XML Parsers DOM y SAX

- Beneficios de SAX
  - SAX parsers reportan eventos a medida que procesan elementos
  - SAX orientado a eventos
    - Mas performante que DOM
  - Memoria consumida por un parser SAX es menor a la de un parser DOM
    - Por que?

XML Parsers DOM y SAX

- Desventajas
  - DOM, al parsear el documento en una única instancia y en su totalidad, crea una representación completa
    - La memoria crece con el tamaño del documento XML
    - Permite acceso a cualquier elemento una vez parseado el XML
  - En SAX la cantidad de memoria consumida es proporcional a la profundidad del árbol XML
    - SAX deja migajas de pan sobre los elementos que comenzó a procesar, y las elimina cuando encuentra el tag de cierre
  - No es sencillo hacer validaciones XML con SAX
    - Se requiere acceso al documento en su totalidad

SAX Parsers

- Como funcionan?
  - El usuario define métodos de callBack
    - Cuando sucede evento X llamar a método M
  - Los eventos definidos son
    - XLM Text nodes
    - XML Element
      - Start
      - End
    - XML Comment
    - XML Processing Instructions
      - **<?xml** version="1.0" encoding="UTF-8" **?>**
      - **<?xml-stylesheet** type="text/xsl" href="style.xsl"**?>**

Conceptos Avanzados sobre XML

Namespaces

- Mecanismo para evitar los conflictos de nombre entre 2 elementos
- Cuando se quieren mezclar 2 XML de diferentes aplicaciones puede suceder que existan elementos con igual nombre
  - Los nombres de los elementos lo define el programador

Namespaces

- Tabla HTML

```
<table>
  <tr>
    <td>Apples</td>
    <td>Bananas</td>
  </tr>
</table>
```

- Tabla (mesa)

```
<table>
  <name>African Coffee Table</name>
  <width>80</width>
  <length>120</length>
</table>
```

Namespaces

- Si los XML anteriores se quisieran combinar, existiría un conflicto de nombres
  - Dos elementos de igual nombre que hacen referencia a conceptos distintos
    - Diferente semántica
- Como se resuelve con conflicto de nombres?
  - Utilizando un prefijio

Namespaces

- Tabla HTML

```
<h:table>
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>
```

- Tabla (mesa)

```
<f:table>
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>
```

Namespaces

- Los prefijos forman parte del nombre
  - No existiría entonces conflicto de nombres
- Cuando se utilizan prefijos se debe definir un namespace por cada prefijo
- Como se define un namespace?
  - Utilizando el atributo **xmlns** en el tag de comienzo
  - Debe cumplir con la siguiente sintaxis
    - **xmlns:*prefijo*="*URI*"**
  - Por lo tanto el ejemplo anterior no esta completo…

Namespaces

- Tabla HTML

```
<h:table xmlns:h="http://www.utu.edu.uy/html/table">
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>
```

- Tabla (mesa)

```
<f:table xmlns:f="http://www.utu.edu.uy/mesa/table">
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>
```

Namespaces

- Si se quisieran combinar en un único XML

```
<?xml version="1.0" encoding="UTF-8"?>
<combinacion xmlns:h="http://www.utu.edu.uy/html/table"
             xmlns:f="http://www.utu.edu.uy/mesa/table">
  <h:table>
    <h:tr>
      <h:td>Apples</h:td>
      <h:td>Bananas</h:td>
    </h:tr>
  </h:table>
  <f:table>
    <f:name>African Coffee Table</f:name>
    <f:width>80</f:width>
    <f:length>120</f:length>
  </f:table>
</combinacion>
```

Namespaces

- URI = **U**niform **R**esource **I**dentifier
  - Cadena de caracteres que identifica a un recurso en la web
  - La URI mas común es una URL
    - Identifica una una dirección de un dominio de internet
- Default namespace
  - Si se define un namespace por defecto, no es necesario prefijar todos los elementos del XML
  - Como se define?
    - **xmlns="*namespaceURI*"**

Namespaces

- Ejemplo de default namespace

```
<table xmlns="http://www.utu.edu.uy/html/table">
  <tr>
    <td>Apples</td>
    <td>Bananas</td>
  </tr>
</table>
```

- Esto es equivalente a

```
<h:table xmlns:h="http://www.utu.edu.uy/html/table">
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>
```

CDATA

- Todo el contenido de un documento XML será intentado leer por un parser
  - Excepto la sección CDATA
- PCDATA = **P**arsed **C**haracter **DATA**
  - Los parser si toman en cuenta la PCDATA
  - Cualquier contenido de texto que será parseado
  - No es necesario especificarlo
    - Por defecto se asume que todo es PCDATA

CDATA

- CDATA = Unparsed **C**haracter **DATA**
  - Cualquier contenido que no debe ser leído por un parser
  - Caracteres inválidos
    - <
    - &
  - Algunos textos (código JavaScript por ejemplo) contienen caracteres inválidos
    - Como evitar estos errores?

CDATA

- Cómo se evitan los errores?
  - Se lo engloba en una sección CDATA

```
<?xml version="1.0" encoding="UTF-8"?>
<cdata>
  <codigo_javascript>
    <![CDATA[
      function menor(a,b){
         if (a<b) return a;
         return b;
      }
    ]]>
  </codigo_javascript>
</cdata>
```

CDATA

- Cómo se evitan los errores?
  - Se lo engloba en una sección CDATA

```
<?xml version="1.0" encoding="UTF-8"?>
<cdata>
  <codigo_javascript>
    <![CDATA[
      function menor(a,b){
         if (a<b) return a;
         return b;
      }
    ]]>
  </codigo_javascript>
</cdata>
```

Indica el inicio de una sección CDATA

Indica el fin de la sección CDATA. No pueden haber espacios o saltos de línea.

  - No se admiten secciones CDATA anidadas

Encoding

- Los documentos XML pueden contener caracteres no ASCII
  - Frances ê è é

- Para evitar errores se debe especificar un encoding o guardar el archivo XML como Unicode

Encoding

- Como se especifica el encoding en un documento XML?
  - Ejemplos de encoding

```
<?xml version="1.0" encoding="us-ascii"?>
<?xml version="1.0" encoding="windows-1252"?>
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml version="1.0" encoding="UTF-8"?>
<?xml version="1.0" encoding="UTF-16"?>
```

Problemas con el encoding

- An invalid character was found in text content
  - Cuando aparece un carácter no ASCII en un XML no Unicode o en un XML sin encoding
- Switch from current encoding to specified encoding not supported.
  - Cuando un documento XML se guarda como UTF-16 (Unicode de 2ble byte) y se especifico un encoding de 8 bytes (UTF-8) o se guardo el archivo XML en ASCII

Encoding y mejores prácticas

- Siempre usar el atributo encoding
- Utilizar editores de XML que den soporte a diferentes tipos de encoding
- Conocer el encoding del editor y especificar el mismo en el documento XML

Editores XML

- XML es un lenguaje de marcas basado en texto plano
  - Se podría usar Notepad como editor XML
- Por que usar un editor para escribir XML?
  - El principal objetivo es ser asistido en la creación del documento para evitar errores

Editores XML

- Que funcionalidades debería tener un editor XML?
  - Cerrar tags de forma automática
  - Validar en runtime el XML
    - De acuerdo a las reglas generales de XML
    - De acuerdo al lenguaje de validación establecido
      - DTD
      - XSD
  - Colorear e indentar el texto para facilitar su lectura y entendimiento