Análisis Sintáctico de XML con DOM

JUAN CARLOS CONDE RAMÍREZ

WEB-TECHNOLOGIES

Objetivos

- Conocer las características básicas de los principales *Analizadores Sintácticos* de XML.
- Recordar la importancia del uso de DOM para la generación de páginas HTML Dinámicas (DHTML).
- Conocer las característica e importancia del DOM con soporte para XML.
- Entender los fundamentos para programar usando DOM-XML y su utilidad.

Analizadores sintácticos

- *Parsers* hay muchos, variados, y con funcionamientos muy diferentes.
- Pueden incluir validación o no.
- Pueden realizar transformaciones o no.
- Pueden exponer la información de diferentes formas (DOM, SAX, etc.).
- Existen para la mayoría de los lenguajes y plataformas de desarrollo (VB, ASP, C, VC++, Perl, Python, PHP, JAVA, ...).

Modelo de Objetos de Documento, I

- Hasta ahora DOM ha sido un modelo que convierte a un HTML estático en dinámico.

- Podemos entenderlo como la forma en la que los exploradores interpretan una página que, por su naturaleza es estática (o desprovista de comportamientos programables), transforma sus elementos en objetos, y como tales poseen:
  - Propiedades
  - Métodos
  - Eventos

- Por lo tanto, gracias a DOM estos elementos se convierten en *entidades programables*.

Modelo de Objetos de Documento, II

- A los lenguajes de programación que nos permiten programar estos objetos DHTML (Dynamic HTML), se les denomina Lenguajes Script (JavaScript, TypeScript, VBScript, etc).

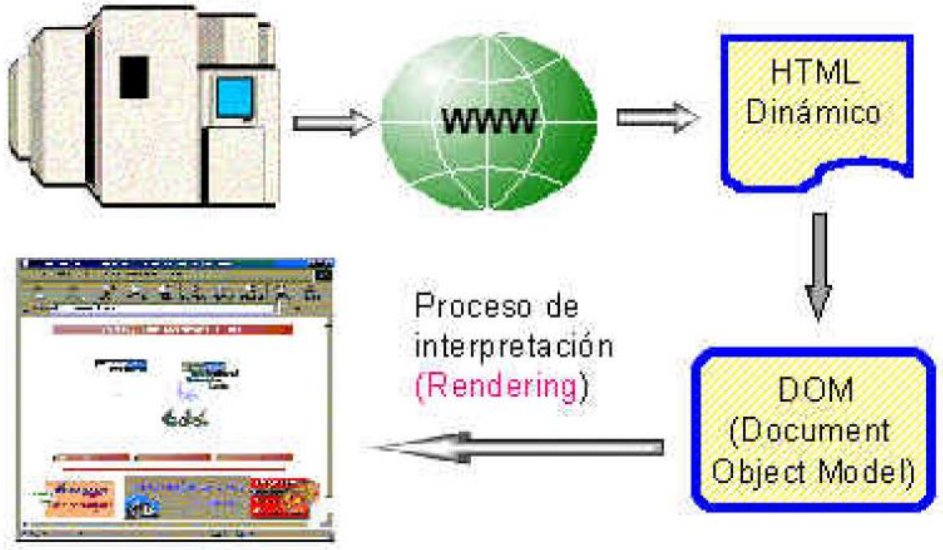
- Un objeto DHTML se programa:

1. Asignándole un identificador (un valor para su atributo ID, lo que lo convierte en objeto programable)

2. Estableciendo una acción escrita en uno de estos lenguajes y asociada con uno cualquiera de los eventos de que el objeto disponga.

Modelo de Objetos de Documento, III

- El esquema de la siguiente figura ilustra la forma en la que viaja y se transforma la información de estática a dinámica.



Funcionamiento del DOM, I

- Cuando un usuario solicita una página web (por ejemplo, https://www.amazon.com/).

1. El servidor Web busca dicha página
2. La página es enviada al cliente
3. Allí sufre un proceso de TRANSFORMACIÓN:
   a. Primero se lee todo el contenido,
   b. se construyen tantos objetos en la memoria como elementos de la página HTML tengan un identificador (ID),
   c. finalmente se da un formato gráfico de salida al documento.

   Al mismo tiempo el motor del navegador permanece a la escucha de los eventos que el usuario genere al navegar por la página.

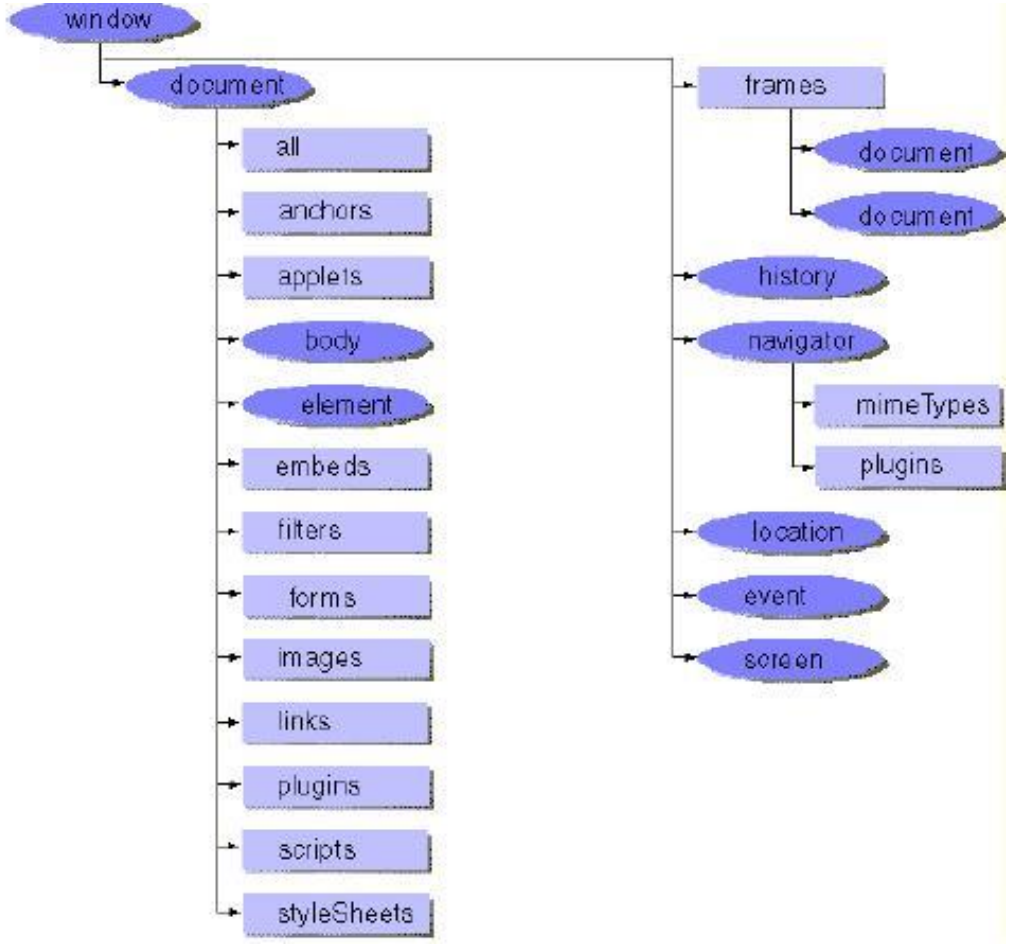
Funcionamiento del DOM, II

- Cuando se produce un **evento** (como pasar el cursor por encima de un ítem o de un gráfico), el *parser* llama al intérprete del lenguaje de *script* que corresponda, y la acción se ejecuta.

- Esta es la forma en que los menús cambian de color o de tamaño, y también la forma en la que se producen un sinfín de efectos especiales a los que ya estamos acostumbrados.

- En la siguiente figura se muestra una jerarquía de objetos del DOM en su versión inicial (DHTML Object Model), es decir, antes de la aparición de XML.

DHTML Object Model, I



DHTML Object Model, II

- Observa que la jerarquía se basa sobre todo en dos objetos fundamentales:
  - El objeto `window`, que refleja la estructura y propiedades del navegador.
  - El objeto `document` (uno sólo, y no una colección, ya que se trata de una interfaz SDI), que contiene todo lo referente a la página web que se visualiza en un momento dado.
  - Como podemos ver, algunos objetos pasan a pertenecer a colecciones concretas, como la colección de imágenes (`images`) o la colección de hojas de estilo (`styleSheets`).
  - Otros, por su importancia se transforman en objetos individuales, como `body` o `element`, y además existe una colección donde van parar todos los objetos programables: `all`.

La llegada de XML

- La llegada de XML supone una expansión de este concepto.

- Podemos trabajar con grupos de objetos (p.e. registros) dotándolos de cierto *comportamiento*, en lugar de tener que hacer un tratamiento individualizado de cada objeto.

- Una vez más es tarea del navegador, o del API de *rendering*, construir objetos dinámicos a partir de lo que sólo es un documento, evitando así, el re-envío de componentes a través de la Web.

Uso de DOM con XML

- Pasemos ahora a la manipulación de los datos desde un doble enfoque:

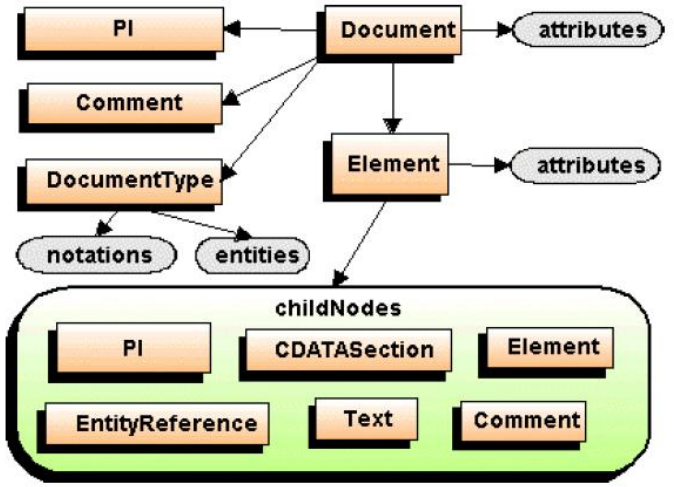
1. El manejo de DOM desde el propio navegador.

2. El uso de documentos XML (como formato de intercambio de datos) tratados con una tecnología de desarrollo, como: PHP. Java, C#, C++, etc. y DOM.

DOM con soporte XML, I

- Todo elemento en un XML puede ser accedido a través de un DOM para XML.

- El DOM con soporte XML es:
  - Un modelo de objetos estándar para XML.
  - Una interfaz de programación estándar para XML.
  - Una plataforma y lenguaje independiente.
  - Un estándar de la W3C.



- En otras palabras:

El DOM-XML es un estándar que permite obtener, modificar, agregar o eliminar elementos XML.

DOM con soporte XML, II

- Se debe tener presente que todo el contenido de un documento XML se ve como un conjunto de **nodos** ordenados jerárquicamente.

- ¿Por qué nodos en lugar de elementos?
  - Porque una de las diferencias principales con la jerarquía DHTML es que allí encontrábamos *colecciones* que debían existir siempre, independientemente de que tuvieran o no elementos.
  - Por ejemplo, siempre encontramos una colección imágenes, aunque la página no contuviera una sola imagen.

DOM con soporte XML, III

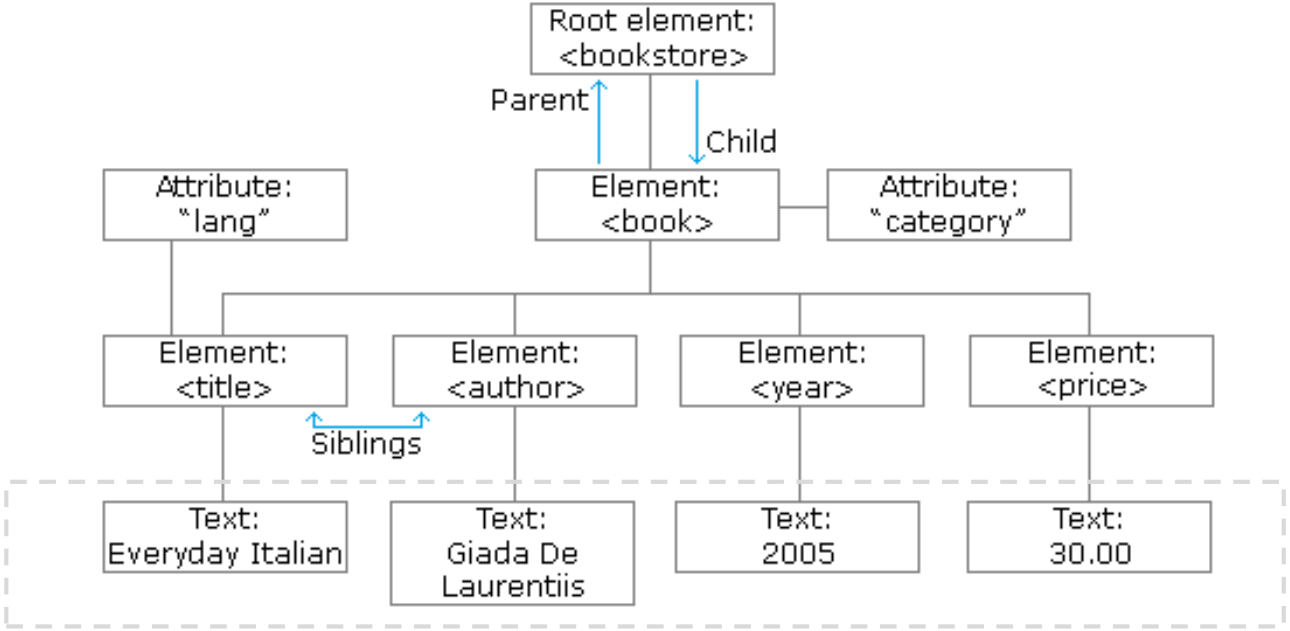
- En XML no existen una serie de *colecciones* predefinidas hasta que no ha concluido el proceso de transformación a un modelo.

- La transformación considera como nodos individuales a:
  - Elementos contenedores (las marcas mismas).
  - Los contenidos (lo que hay entre las marcas).

- La única cosa que se sabe con seguridad, es que habrá un objeto `element`, que se corresponderá con la raíz. Todo lo demás dependerá del contenido del documento.

DOM con soporte XML, IV

- Debido a que no existe una *colección* predefinida, a excepción del elemento raíz, es mejor imaginarse el contenido como un estructura de árbol y cada ítem como un nodo genérico.



Tomado de https://www.w3schools.com/xml/dom_intro.asp

DOM con soporte XML, V

- Algunos nodos equivaldrán a las hojas, que no contendrán más sub-elementos.
- Otros serán equivalentes a los troncos y su misión será contener una serie de nodos hijos (*child nodes*).
- Cada nodo dispone de un conjunto de métodos que permiten la manipulación programática de elementos y contenidos.

Programación con DOM-XML, I

- El usuario del API puede, dinámicamente, crear nuevos nodos y cambiar características de los ya existentes.

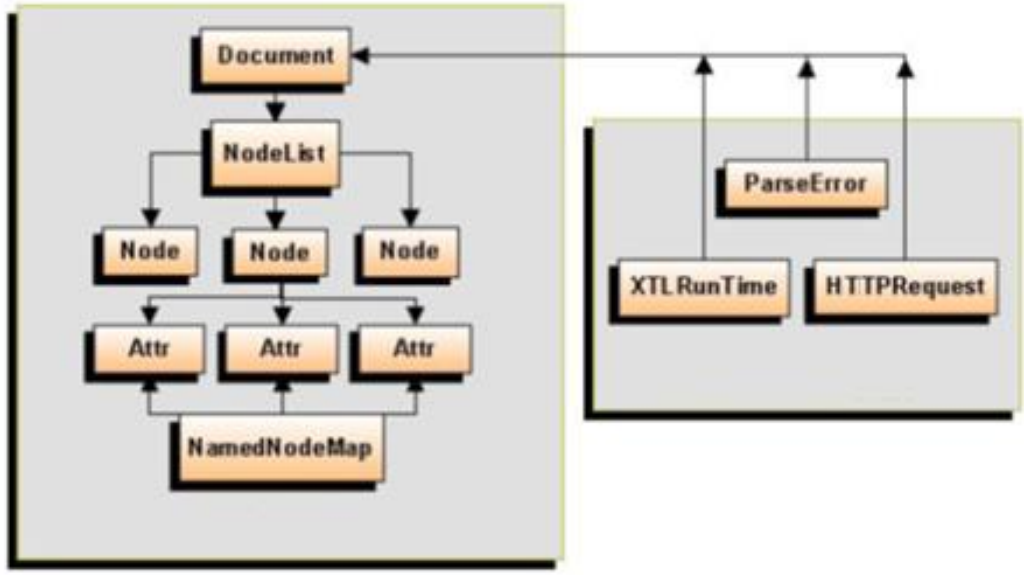
- Cada nodo dispone de los **métodos** necesarios para acceder a aquellos nodos que se encuentran vinculados con él mediante relaciones jerárquicas.

- Además se dispone de algunas **propiedades** específicas para consultar valores como el *namespace*, el prefijo (*prefix*), o la definición de un elemento.

Programación con DOM-XML, II

- La lista completa de objetos, métodos y atributos de DOM-XML es larguísima.

- Sin embargo, haciendo una abstracción de cara a la programación, podemos hacernos una idea de los objetos más importantes mediante el siguiente diagrama.



Programación con DOM-XML, III

- El diagrama anterior muestra cómo, además de componentes estándar, añade ciertas funcionalidades para facilitar el manejo de los documentos XML.
  - `ParseError` informa acerca de los problemas que pudiera encontrar el propio intérprete a la hora de transformar el documento XML.
  - `HTTPRequest` (ó `XMLHttpRequest`) permite la gestión de un protocolo de comunicaciones HTTP entre el cliente y el servidor.
  - Y otras funcionalidades asociadas con XML para validar documentos y manipular hojas de estilo XSL.

Programación con DOM-XML, IV

- Cuando se carga un documento XML existe un objeto especial denominado `Document` del cual dependen todos los objetos de la jerarquía. Podemos interpretarlo como el **nodo raíz**.

- De él se derivan objetos `Node` y `NamedNodeMap` que permiten el acceso a los valores, además de incluir soporte para los espacios de nombres (*namespaces*).

- Por su parte el objeto `NodeList` permite acceder a los nodos de forma global, ya que contiene la lista de los nodos obtenidos.

Programación con DOM-XML, V

- Cada nodo tiene una propiedad `childNodes` que hace referencia a sus nodos hijos y a una propiedad `attributes`.

  - Si el nodo contiene **nodos hijos** la propiedad `childNodes` devuelve una referencia al objeto `NodeList` que contiene los nodos hijos. En caso contrario devuelve `null`.

  - Si un nodo contiene algún **atributo**, la propiedad `attributes` devuelve una referencia al objeto `NamedNodeMap`, que contiene los objetos nodo de tipo `attribute`. En caso contrario devuelve `null`.

Más del DOM-XML

- Para mayor información sobre toda la jerarquía de objetos del modelo XML DOM, se puede encontrar en su totalidad en la dirección Internet:

http://www.w3.org/TR/REC-DOM-Level-1/

Ejemplo, I

```
<Clientes>
    <Ciudad>Salamanca</Ciudad>
</Clientes>
```

- Para el *parser* existirá un nodo `Ciudad` que será de tipo `element` y la cadena `Salamanca`, que está contenida en él, también será un nodo pero de tipo `Text`.

- Por lo tanto, propiedad `value` del nodo `Ciudad` valdrá `null`, pero dispondrá de un nodo hijo cuya propiedad `value` será `Salamanca`.

Ejemplo, II

- Un nodo sólo puede tener un nodo padre, pero podrá contener una colección de nodos hijos (con la sola excepción del nodo raíz, que no tiene nodo).

- En el ejemplo anterior hemos llamado `Clientes`, por tanto el nodo raíz es como si hiciera referencia al documento mismo.

- Finalmente, los elementos pueden tener atributos, y estos atributos son a su vez objetos nodo, sólo que de un subtipo diferente al del resto de nodos.