W3C Recommendation

07 April 2004

¿Qué es el Modelo de Objetos del Documento?

Redactores:
Philippe Le Hégaret, W3C
Lauren Wood, SoftQuad Software Inc. (for DOM Level 2)
Jonathan Robie, Texcel (for DOM Level 1)

Introducción

El Modelo de Objetos del Documento (DOM) es una interfaz de programación de aplicaciones (API) para documentos validos HTML y bien construidos XML. Define la estructura lógica de los documentos y el modo en que se accede y manipula. En la especificación DOM, el término "documento" es utilizado en un sentido amplio - the term "document" is used in the broad sense - cada vez más XML es utilizado como un medio de representar muchas clases diferentes de información que puede ser almacenada en sistemas diversos, y mucha de esta información se vería, en términos tradicionales, más como datos que como documentos. Sin embargo, XML presenta estos datos como documentos, y se puede utilizar DOM para manejar estos datos.

Con el Modelo de Objetos del Documento, los programadores pueden construir documentos, navegar por su estructura, y añadir, modificar, o eliminiar elementos y contenido. Se puede acceder a cualquier cosa que se encuentre en un documento HTML o XML, modificando, borrando o añadiendo utilizando el Modelo de Objetos del Documento, con algunas excepciones - en particular, aún no se han especificado aplicaciones DOM para los subconjuntos interneto y externos de XML.

Como una especificación de W3C, un objetivo importante para el Modelo de Objetos del Documento es proporcionar un interfaz estándar de programación que puede ser utilizado en una amplia variedad de entornos y aplicaciones. El DOM se diseña para ser utilizado en cualquier lenguaje de programación. Para proporcionar una especificación de las aplicaciones DOM precisa e independiente del lenguaje, hemos decidido definir las especificaciones en Grupo de Dirección de Objeto (GDO) (OMG -- Object Management Group) IDL [*OMG IDL*], según se define en la especificación CORBA 2.3.1 [*CORBA*]. Ademas de la especificación GDO (OMG) IDL, proporcionamos correspondencias con los lenguajes Java [*Java*] y ECMAScript [*ECMAScript*] (un lenguaje de scripts industrial basado en JavaScript [*JavaScript*] y JScript [*JScript*]). Por restricciones de correspondencias del lenguaje, se ha de aplicar un mapeado entre GDO (OMG) IDL y el lenguaje de programación utilizado. Por ejemplo, mientras que DOM utiliza atributos IDL en la definición de la interfaz, Java no permite que las aplicaciones contengan atributos:

```
// Ejemplo 1: Quitar el primer hijo de un elemento utilizando ECMAScript
mySecondTrElement.removeChild(mySecondTrElement.firstChild);

// Ejemplo 2: Quitar el primer hijo de un elemento utilizando Java
mySecondTrElement.removeChild(mySecondTrElement.getFirstChild());
```

Nota: GDO (OMG) IDL se utiliza únicamente como un medio de especificar las aplicaciones independiente de la plataforma y del lenguaje. Se podría hacer utilizando otros IDLs ([*COM*], [*Java IDL*], [*MIDL*], ...). En general, los IDLs se diseñan para entornos de computación específicos. El Modelo de Objetos del Documento puede implentarse en cualquier entorno de computación, y no requiere las librerías de enlazado de objetos (object binding runtimes) generalmente asociadas con tales IDLs.

Lo que el Modelo de Objetos es

DOM es un API de programación para documentos. Está basado en una estructura de objeto muy parecida a la estructura del documento que modela. Por ejemplo, considere esta tabla, tomada de un documento XHTML:

```
<table>
  <tbody>
    <tr>
      <td>Shady Grove</td>
      <td>Aeolian</td>
    </tr>
    <tr>
      <td>Sobre el río, Charlie</td>
      <td>Dorian</td>
    </tr>
  </tbody>
</table>
```

Una representación gráfica DOM de la tabla del ejemplo, con espacios en blanco en el contenido del elemento (amenudo abusivamente llamado "espacios en blanco ignorables") eliminados, es:

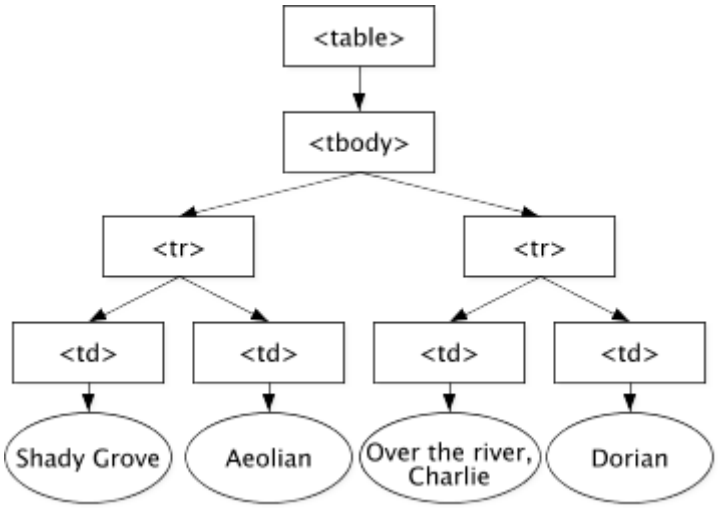

Figura: representación gráfica DOM de la tabla del ejemplo [SVG 1.0 versión]

Un ejemplo de manipulación de DOM utilizando ECMAScript podría ser:

```
// Tener acceso sobre el elemento tbody del elemento tabla
var myTbodyElement = myTableElement.firstChild;

// Tener acceso al segundo elemento tr
// La lista de hijos empieza en 0 (no en 1).
var mySecondTrElement = myTbodyElement.childNodes[1];
```

```
// Quitar su primir elemento td
mySecondTrElement.removeChild(mySecondTrElement.firstChild);

// Cambiar el contenido del texto del elemento restante td
mySecondTrElement.firstChild.firstChild.data = "Peter";
```

En DOM, los documentos tienen una estructura lógica que es muy parecida a un árbol; para ser más preciso, es más bien como un "bosque" o una "arboleda", que puede contener más de un árbol. Cada documento contiene cero o un nodo doctype, uno nodo de elemento de documento, y cero o más comentarios o instrucciones de tratamiento; el elemento del documento sirve como la raíz del árbol para el documento. Sin embargo, DOM no especifica que los documentos deban ser *implementados* como un árbol o un bosque, ni tampoco especifica como deben implementarse las relaciones entre los objetos. DOM es un modelo lógico que puede implementarse de cualquier manera que sea conveniente. En esta especificación, usamos el término *modelo de estructura* para describir la representación en forma de árbol de un documento. También utilizamos el termino "árbol" cuando refiriendose al arreglo de aquellos artículos de información que pueden ser alcanzados utilizando métodos "exploració por árbol"; (esto no incluye atributos). Una propiedad importante de los modelos de estructura de DOM es su *isomorfismo estructural*: si dos implementaciones cualesquiera del Modelo de Objetos del Documento se usan para crear una representacio´n del mismo documento, ambas crearán el mismo modelo de estructura, en concordancia con la Información XML Establecida [*Información XML Establecida*].

Nota: Puede haber variaciones según el programa de análisis utilizado para contruir el DOM. Por ejemplo, DOM puede no contener espacios en blanco en el elemento contenido si el análisis los descarta.

El nombre "Modelo de Objetos del Documento" se eligio porque es un "modelo de objeto" en el sentido tradicional del diseño orientado a objetos: los documentos se modelizan utilizando objetos, y el modelo comprende no solamente la estructura de un documento, sino también el comportamiento de un documento y los objetos de los cuales se compone. En otras palabras, los nodos del diagrama anterior no representan una estructura de datos, representan objetos, los cuales pueden tener funciones e indentidad. Como modelo de objeto, DOM identifica:

- las aplicaciones y objetos utilizados para representar y manipular el documento
- la semantica de estas aplicaciones y objetos - incluyendo comportamientos y atributos
- las relaciones y colaboraciones entre estas aplicaciones y objetos

Tradicionalmente, la estructura de documentos SGML se ha representado mediante un modelo de datos abstractos, no por un modelo de objeto. en un modelo de datos abstracto, el modelo se centra en los datos. En los lenguajes de programación orientados a objetos, los datos se encapsulan en objetos que ocultan los datos, protegiéndolos de su manipulación directa desde el exterior. Las funciones asociadas con estos objetos determinan como pueden manipularse los objetos, y son parte del modelo de objeto.

Lo que el Modelo de Objetos del Documento no es

Esta secció es diseñada para dar un entendimiento más preciso del DOM distinguiendolo de otros sistemas que aparentemente pueden resultar similares a él.

- El Modelo de Objetos del Documento no es una especificación binaria. Los programas DOM escritos en el mismo lenguaje serán compatibles entre

plataformas a nivel de código fuente, pero DOM no define ninguna forma de interoperatibilidad binaria.
- El Modelo de Objetos del Documento no es una manera de ofrecer objetos persistentes para XML o HTML, DOM especifica los documentos como XML y HTML son representados como objetos, de modo que pueden ser utilizados por programas orientados a objetos.
- El Modelo de Objetos del Documento no es un conjunto de estructuras de datos, es un modelo de objeto que especifica aplicaciones. Aunque este documento contiene diagramas que muestran relaciones padres/hijos, estas son relaciones lógicas definidas por las aplicacione de programación, no representaciones de ninguna estructura interna de datos particular.
- El Modelo de Objetos del Documento no define que información en u ndocumento es relevante o que información en un documento es estructurado. Para XML, esta es especificacicada por el Conjunto de Información XML [*Conjunto de Información XML*]. DOM es simplemente un API de este conjunto de información.
- El Modelo de Objetos del Documento, a pesar de su nombre, no es un competidor del Modelo de Objetos de Componentes (MOC (Component Object Model -- COM) [*MOC (COM)*]. MOC (COM), al igual que CORBA, es una manera independiente de lenguaje de especificar aplicaciones y objetos diseñado para manipular documentos HTML y XML. DOM puede implementarse utilizando sistemas independientes del lenguaje como MOC (COM) o CORBA; también se puede implementar usando enlaces especín con lenguajes, como los especificados en este documento para Java o ECMAScript.

De donde viene el Modelo de Objetos del Documento

El DOM se originón como una especificación para permitir que los scrips de JavaScript y los programas Java fueran portables entre los navegadores Web. El "HTML Dinámico" fue el antesesor inmediato del Modelo de Objetos del Documento, y originalmente se pensaba en él principalmente en términos de navegadores. Sin embargo, cuando se formó el Grupo de Trabajo DOM en W3C, también se unieron a él compañías de otros ámbitos, incluyendo redactores y depositos de documentos HTML o XML. Varios de estos redactores han trabajdo con SGML antes de ser desarrollado XML; como resultado de ello, DOM ha recibido influencias de los "Bosques" y del estándar HyTime. Algunas de estas compañias también habían desarrollado sus propios modelos de objetos para documentos a fin de proporcionar un API para los editores o los archivos de documentos SGML/XML, y estos modelos de objetos también han influido en el DOM.

Las entidades y el núcleo del DOM

En las aplicaciones fundamentales del DOM, no hay objetos que representen entidades. Las referencias numéricas de caracteres y las referencias a entidades predefinidas en HTML y en XML, son reemplazadas por el carácter individutal que constituye la sustitución de la entiedad. Por ejemplo, en:

```
<p>Esto es un perro &amp;amp; un gato</p>
```

El "&" será reemplazado por el carácter "&", y el texto del elemento P formará una única secuencia continua de caracteres. Debido a que las referencias numéricas de caracteres y las entidades predefinidas no son reconocidas como tales en las secciones CDATA, o en los elementos SCRIPT y STYLE de HTML, no son reemplazadas por el carácter individual al que aparentemente se refieren. Si el ejemplo anterior estuviera contenido en una sección; CDATA, el "&" no sería reemplazado por "&"; y el <p> tampco sería reemplazado como etiqueta inicial. La representación de entidades

generales, tanto internas como externas, está definida dentro de las aplicaciones extendidas (XML) del Núcleo del Modelo de Objetos del Documento.

Nota: Cuando una representación DOM de un documento es serializada como texto XML o HTML, las aplicaciones necesitarán comprobar cada carácter de los datos del texto para ver si necesita ser convertido en una secuencia de escape utilizando una entidad numérica o predefinida. De lo contrario se podría obtener un HTML o XML no válido. Además, las implementaciones deberían ser conscientes del hecho de que la serialización en una codificación de caracteres ("charset") que no cubra completamente la ISO 10646 puede fallar si hay caracteres en el código o en las secciones CDATA que no estén presentees en esa codificación.

Arquitectura del DOM

La especificación del DOM proporciona un conjunto de APIs que forman la API del DOM. Cada especificación del DOM define uno o mas módulos y cada módulo esta asociado con un nombre de funcionalidad. Por ejemplo, la especificación del Núcleo de DOM (esta especificación) define dos módulos:

- El módulo Núcleo, el cual contiene las aplicaciones fundamentales que deben ser implementadas por todas las aplicaciones DOM conformadas, está asociado con el funcionalidad "Core";
- El módulo XML, el cual contiene las aplicaciones que deben ser implementadas por todos las aplicaciones XML 1.0 [*XML 1.0*] (y superiores) DOM conformadas, está asociado con el funcionalidad "XML".

La siguiente representación contiene todos los módulos de DOM, representado utilizando sus nombres de funcionalidades, definidos a lo largo de las especificaciones DOM:

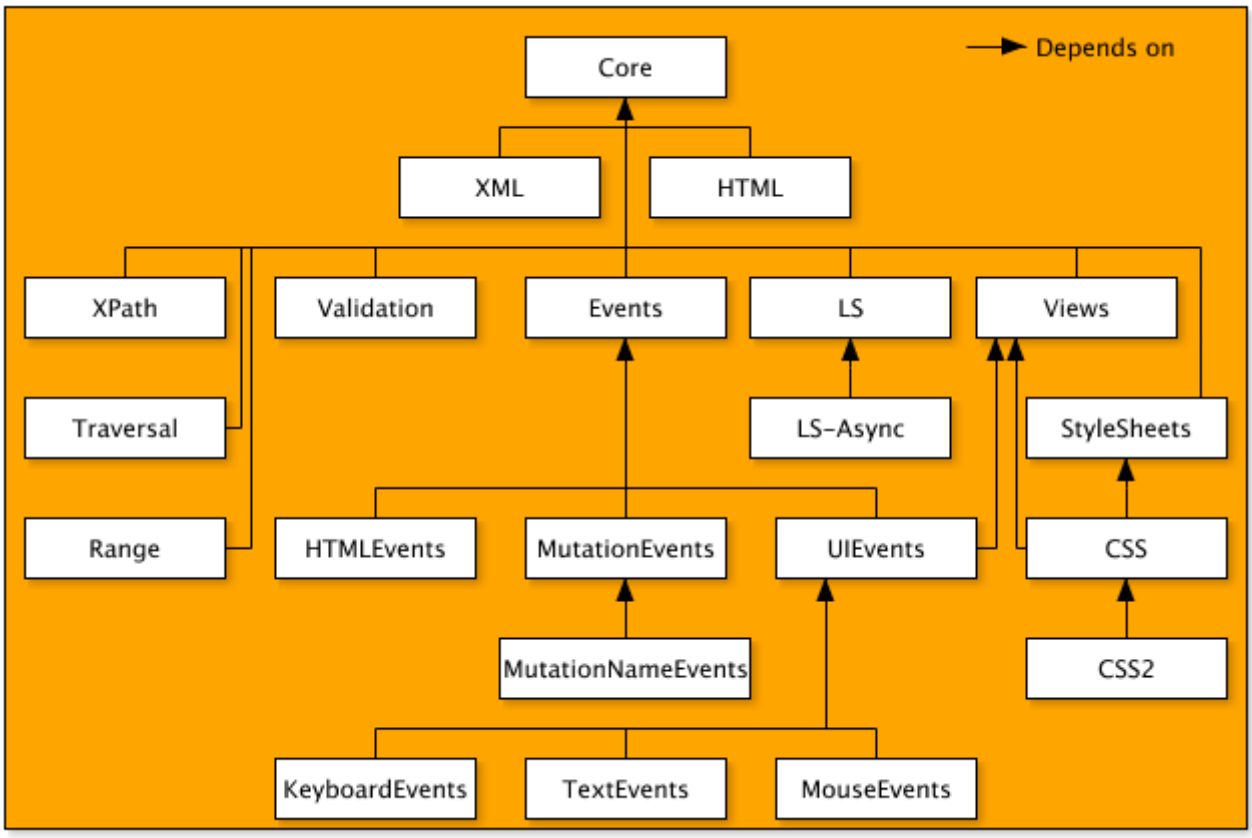

Figura: Una visión de la arquitectura de DOM [SVG 1.0 versión]

Una implementación DOM puede entonces implementar un (es decir, solo el módulo Núcleo) o más módulos dependiendo del uso de la aplicación. Una aplicación de usuario para Web es muy probable que implemente el módulo "MouseEvents" (eventos del ratón), mientras que una aplicación del lado del servidor no tendría que utilizar este módulo y probablemente no lo implementaría.

Conformidad

Esta sección explica los diferentes niveldes de conformidad del Nivel 3 de DOM. DOM Nivel 3 consiste en 16 módulos. Es posible ser conformes de DOM Nivel 3, o al módulo de DOM Nivel 3.

Una aplicación es conforme a DOM Nivel 3 si soporta el módulo Núcleo definido en este documentoconformant if it supports the Core (vea Aplicaciones Fundamentales: Módulo Núcleo). Una aplicación es conforme al módulo DOM Nivel 3 si soporta todas las aplicaciones para ese módulo y la semántica asociada.

Aquí está la lista completa de los módulos de DOM Nivel 3.0 y las funcionalidades utilizadas por ellos. Los nombres de las funcionalidades son insensibles a mayúsculas y minúculas (case-insensitive).

Módulo Núcleo
: Define la funcionalidad *"Core"*.

Módulo XML
: Define la funcionalidad *"XML"*.

Módulo Eventos
: Define la funcionalidad *"Events"* en [*Eventos de DOM Nivel 3*].

Módulo de Eventos de la Aplicación del Usuario
: Define la funcionalidad *"UIEvents"* en [*Eventos de DOM Nivel 3*].

Módulo de Eventos del Ratón
: Define la funcionalidad *"MouseEvents"* en [*Eventos de DOM Nivel 3*].

Módulo de Eventos del Texto
: Define la funcionalidad *"TextEvents"* en [*Eventos de DOM Nivel 3*].

Módulo de Eventos del Teclado
: Define la funcionalidad *"KeyboardEvents"* en [*Eventos de DOM Nivel 3*].

Módulo de Eventos de Mutación
: Define la funcionalidad *"MutationEvents"* en [*Eventos de DOM Nivel 3*].

Módulo de Eventos de nombre Mutación
: Define la funcionalidad *"MutationNameEvents"* en [*Eventos de DOM Nivel 3*].

Módulo de Eventos HTML
: Define la funcionalidad *"HTMLEvents"* en [*Eventos de DOM Level 3*].

Módulo Cargar y Guardar
: Define la funcionalidad *"LS"* en [*Cargar y Guardar de DOM Nivel 3*].

Módulo de carga Asincronica
: Define la funcionalidad *"LS-Async"* en [*Cargar y Guardar de DOM Nivel 3*].

Módulo de Validación
: Define la funcionalidad *"Validation"* en [*Validación de DOM Nivel 3*].

Módulo XPath
: Define la funcionalidad *"XPath"* en [*XPath de DOM Nivel 3*].

Una implementación no debe devolver `verdadero` al `DOMImplementation.hasFeature(funcioinalidad, versión)` método de una aplicación `DOMImplementation` para ese funcionalidad a menos que la implementación tenga conformidad con ese módulo. EL número de `versión` para todos los de las funcionalidades utilizados en DOM Nivel 3.0 es `"3.0"`.

Aplicaciones DOM e Implementaciones DOM

DOM especifica aplicaciones que pueden utilizarse para manipular documento XML o HTML. Es importante darse cuenta de que estas aplicaciones son una abstracción - comparables a las "clases de base abstractas" en C++, constituyen un medio para especificar una forma de acceder y manipular la representación interna que una aplicación hace de un documento. Las aplicaciones no implican una implementación particular concreta. Cada aplicación DOM es libre de mantener los documentos según una repsresentación cualquiera, siempre y cuando soporte las aplicaciones mostradas en esta especificación. Algunas implementaciones del DOM serán programas existentes que usen las interfaces del DOM para acceder a programas escritos mucho antes de que existiera la especificación DOM. Por tanto, DOM se ha diseñado para evitar dependencias de implementación; en particular,

1. Los atributos definidos en el IDL no implican objetos concretos que deban tener miembros de datos especificos - en las correspondencias con cada lenguaje, se transforman en pares de funciones get()/set(), no en un miembro de datos. Los atributos de solo lectura tendrán una función get() en la correspondencia con el lenguaje.
2. Las aplicaciones DOM pueden proporcionar interfaces adicionales y objetos que no se encuentren en esta especificación y seguir siendo consideradas como conformes con el DOM.
3. Debido a que especificamos interfaces y no los objetos reales que deben ser creados, DOM no puede saber a qué constructores llamará una implementación. En general, los usuarios de DOM llamar´n a métodos createX() de la clase Document para crear estructuras del documento, y las implementaciones DOM crearán sus propias representaciones internas de dichas estructuras en sus implementaciones de las funciones createX().

Las interface de Nivel 2 fueron apliadas para proveer funcionalidad tanto al Nivel 2 como Nivel 3.

Las implementaciones DOM en otros lenguajes como Java o ECMAScript pueden elegir los enlaces que sean apropiados y naturales para sus lenguajes y las librerias necesarias (run time environment). Por ejemplo, algún sistema puede necesitar crear una clase Document3 que hereda de una clase Document y contiene los nuevos métodos y atributos.

DOM Nivel 3 no especifica mecanismos de multiinserciones.
