Document Object Model

De Wikipedia, la enciclopedia libre

El ***Document Object Model*** o **DOM** ('Modelo de Objetos del Documento' o 'Modelo en Objetos para la Representación de Documentos') es esencialmente una interfaz de programación de aplicaciones (API) que proporciona un conjunto estándar de objetos para representar documentos HTML y XML, un modelo estándar sobre cómo pueden combinarse dichos objetos, y una interfaz estándar para acceder a ellos y manipularlos. A través del DOM, los programas pueden acceder y modificar el contenido, estructura y estilo de los documentos HTML y XML, que es para lo que se diseñó principalmente.

El responsable del DOM es el World Wide Web Consortium (W3C).

El DOM es una interfaz de programación de aplicaciones para acceder, añadir y cambiar dinámicamente contenido estructurado en documentos con lenguajes como ECMAScript (JavaScript).

Jerarquía de DOM.

Índice

Desarrollo del DOM

La primera vez que el DOM se utilizó, fue con el navegador Netscape Navigator en su versión 2.0. Este DOM se conoce también como «modelo básico», o «DOM nivel 0». Internet Explorer 3.0 fue el primer navegador de Microsoft que utilizó este nivel. Netscape 3.0 empezó a utilizar *rollovers*. Microsoft empezó a usar *rollovers* en Internet Explorer 4.0. Netscape 4.0 agregó la capacidad de detectar eventos ocurridos en el ratón y el teclado. Una característica de este navegador fue el uso de capas. Sin embargo, esta capacidad se ha eliminado en los navegadores creados posteriormente.

En Internet Explorer 4.0 todos los *elementos* de una página web se empezaron a considerar *objetos* computacionales con la capacidad de ser modificados. Debido a las diferencias en estos navegadores, el World Wide Web Consortium emitió una especificación denominada «DOM nivel 1» en el mes de

octubre de 1998 en la cual se consideraron las características y manipulación de todos los elementos existentes en los archivos HTML y XML. En noviembre del año 2000 se emitió la especificación del «DOM nivel 2». En esta especificación se incluyó la manipulación de eventos en el navegador, la capacidad de interacción con CSS, y la manipulación de partes del texto en las páginas de la web. «DOM nivel 3» se emitió en abril de 2004; utiliza la definición de tipo de documento (DTD) y la validación de documentos.

Problemas de compatibilidad

La guerra entre navegadores que existió entre el Netscape Navigator y el Internet Explorer de Microsoft y otras compañías sigue creando graves problemas para los programadores de páginas web, ya que, aunque todos los navegadores utilizan Javascript como uno de los lenguajes de programación, los objetos no se comportan de la misma forma, lo que obliga con frecuencia a programar las páginas en más de una versión, una para el Netscape, o Mozilla Firefox, Safari, Opera y Chrome, etc, y otra para Internet Explorer. En suma, no todas las versiones de un mismo navegador se comportan igual.

El World Wide Web Consortium (W3C), el consorcio encargado de definir los estándares de la web, decidió crear un modelo de objetos único, el DOM, para que todos los fabricantes pudieran adoptarlo, facilitando la compatibilidad plena entre ellos.

No obstante, Microsoft ha añadido su propia extensión al DOM, creando problemas de interoperabilidad para los navegadores web.

Como el navegador de Microsoft, Internet Explorer, hasta el año 2002 fue el navegador web más empleado, esto llevó a un problema real a los desarrolladores de navegadores más comprometidos con los estándares, como Mozilla. Si adoptan las extensiones de Microsoft al DOM, se arriesgan a perder credibilidad en sus llamadas a que los sitios web respeten el estándar, y si no lo hacen, se arriesgan a alienar a sus usuarios por la pérdida de compatibilidad con casi todos los sitios web que utilizan las extensiones de Microsoft. Los críticos han observado esta actitud como otro caso de aplicación de la táctica de Microsoft de «adoptar, extender y extinguir». Esto puede ser considerado irónico, debido a que tanto Microsoft como Netscape fueron responsables de proporcionar extensiones no estándares en la «carrera armamentística» por el control del estándar, y Mozilla surgió como una iniciativa de Netscape.

La opinión general parece indicar que esto cambiará solo si nuevos navegadores que respeten los estándares ganan una cuota de mercado significativa en la Web, de forma que el uso de extensiones no estándares se convierta en un problema comercial para los autores de los sitios web que las usen. Esto ya está pasando con Mozilla Firefox, el cual desde hace varios años, viene incrementando su utilización en PC junto a la novedad de Google Chrome.

Actualmente la cuota de mercado del navegador IE se va reduciendo, actualmente representa 1/3 de la cuota de mercado, perdiendo un 10% en este último año 2011-12.

Estableciendo referencias a objetos

El DOM define la manera en que objetos y elementos se relacionan entre sí en el navegador y en el documento.

Cualquier lenguaje de programación adecuado para el diseño web puede ser utilizado. En el caso de JavaScript, cada objeto tiene un nombre, el cual es exclusivo y único. Cuando existen más de un objeto del mismo tipo en un documento web, estos se organizan en un vector.

Es posible asignarle una identificación a un objeto, y luego usarla para hacer referencia a éste, por ejemplo:

```
<div id="Juan">....</div>
```

Para hacer referencia a elementos del mismo tipo, los cuales, como se ha dicho, están organizados en un vector, se pueden utilizar puntos de la siguiente manera.

```
document.div[0]
document.div["Juan"]
document.div.Juan
```

Donde el elemento «Juan» es el primer elemento del vector de elementos del tipo `<div>`.

También se puede usar la función *getElementById*.

```
document.getElementById("Juan")
```

Manipulando las propiedades y funciones de objetos

Los objetos computacionales de la misma forma que cualquier objeto de la vida real, tienen propiedades. Algunos ejemplos de propiedades de objetos de la vida real son dimensiones, color y peso.

En la mayoría de los objetos computacionales algunas propiedades se pueden determinar de la siguiente manera:

```
Objeto.propiedad = valor;

//por ejemplo para el objeto «Vaso»

Vaso.color = rojo;
```

La manipulación de objetos sigue los mismos principios que en el lenguaje de programación que se esté utilizando. Una de las características de estos objetos es la *función* para la cual están diseñados, de hecho la mayoría de las ocasiones tienen más de una función. En JavaScript, muchas funciones para cada uno de los objetos, incluyendo el navegador y la ventana que lo contiene, han sido definidas previamente; adicionalmente, el usuario puede definir funciones de acuerdo a sus necesidades, por ejemplo el código:

```
function comeLaLetraA(Texto){
  var TextoNuevo = "";
  while(letras en el Texto recibido){
    //lee la siguiente letra
    //si esta letra no es «a» añádela al nuevo texto
  }
  return TextoNuevo;
}
```

Añade una nueva función al documento utilizado para crear una página web.

Eventos

Un evento desde el punto de vista computacional ocurre cuando alguna situación cambia en la computadora, como por ejemplo, la posición del ratón, la pulsación de alguna tecla, los contenidos de alguna de las memorias, la condición de la pantalla, etc. En la creación de páginas web estos eventos representan la interacción de la computadora con el usuario.

Cuando algunos de estos eventos ocurren, como por ejemplo la presión de algún botón del ratón, es deseable que la computadora responda de alguna manera. Esta es la razón por la que existen *event handlers* ('encargados de manejar eventos') los cuales son objetos que responden a eventos. Una manera de añadir eventos en el DOM utilizando javascript es:

```
<element onevent="script">....</element>
```

Por ejemplo:

```
<div id="midivision" onClick="javascript:comeLaLetraA('bar');">
Aquí va otro texto
</div>
```

Otra forma de manipular eventos en JavaScript al crear páginas web es tratándolos como propiedades de los elementos que forman la página, por ejemplo:

```
object.event = funcion;
//como puede ser:
document.midivision.onclick = hazAlgo;
// también:
document.getElementById("midivision").onclick = hazAlgo;
```

En DOM se considera que un evento se origina en el exterior de la página web y se propaga de alguna manera hasta los elementos internos de la página. Un posible ejemplo de esta propagación es:

```
EVENTO → Ventana → Document → HTML → BODY → DIV → DESTINO
RESPUESTA → DIV → BODY → HTML → Document → Ventana → EVENTO
```

Siguiendo esta idea, se establecen tres etapas: *captura*, la cual se da cuando el evento se está trasladando a su destino. *Blanco*, que ocurre cuando llega al blanco, o sea que llega a su destino. Este destino es el objeto en el cual se va a crear una reacción a este evento. Finalmente la etapa de *burbujeo* que ocurre cuando el evento «regresa» a su posición original.

Ciertos objetos pueden estar al pendiente de ciertos eventos. Para hacer esto el objeto añade un «oyente de eventos» con la función addEventListener. Cuando el evento ocurra, alguna función determinada se lleva a cabo, en este proceso se indica en que momento la función se lleva a cabo, ya sea en la etapa de *captura* o en la etapa de *burbujeo*. Este momento se indica con la palabra *true* si debe ocurrir en la etapa de captura o *false* si debe ocurrir en la etapa de burbujeo. En JavaScript se escribe de la siguiente manera:

```
objeto.addEventListener(evento, funcion, momento);

por ejemplo:

document.getElementById("mydivision").addEventListener("click", hazAlgo, false);
```

Enlaces externos

- Especificación de DOM Level 1 (http://html.conclase.net/w3c/dom1-es/cover.html) (en español)
- DOM según el W3C (http://www.w3.org/DOM)
- DOM según Mozilla (https://developer.mozilla.org/en/Gecko_DOM_Reference)
- DOM según Microsoft (http://msdn.microsoft.com/en-us/library/yek4tbz0(VS.85).aspx)
- Frameworks de JavaScript que trabajan a través de distintos DOM
  - MooTools (http://mootools.net) (MIT Style License)
  - JQuery (http://jquery.com) (licencia: MIT o GPL Versión 2)
  - Prototype (http://www.prototypejs.org) (licencia: MIT para código fuente y CC BY-SA para documentación)
  - Dojo (http://www.dojotoolkit.org) (licencia: AFL o BSD)
  - Ext JS (http://www.sencha.com/products/js) (Licencia: GPL (http://www.sencha.com/products/license.php))
- Interfaces para...
  - C++ (http://xml.apache.org/xerces-c/program-dom.html)
  - Java (http://www.w3.org/2003/01/dom2-javadoc/index.html) — W3C Document Object Model Level 2
  - Lisp (http://interaction.in-progress.com/developer/dom/)
  - Pascal (http://www.philo.de/xml/downloads.shtml) (Kylix)
  - Perl (http://search.cpan.org/~tjmather/XML-DOM-1.44/lib/XML/DOM.pm)
  - PHP (http://mx1.php.net/manual/es/book.dom.php)
  - Python (http://www.python.org/doc/current/lib/module-xml.dom.html)
  - Ruby (http://libgdome-ruby.berlios.de/)
  - TCL (http://tclxml.sourceforge.net/tcldom.html)

Categorías: Navegadores web | XML | HTML | Interfaces de programación de aplicaciones | Desarrollo web

---