Document Object Model

Document Object Model o **DOM** ('Modelo de Objetos del Documento' o 'Modelo en Objetos para la Representación de Documentos') es esencialmente una interfaz de plataforma, o API de programación para documentos HTML, que proporciona un conjunto estándar de objetos para representar documentos HTML, XHTML, XML y SVG,[nota 1] un modelo estándar sobre cómo pueden combinarse dichos objetos, y una interfaz estándar para acceder a ellos y manipularlos.[1] A través del DOM, los programas pueden acceder y modificar el contenido, estructura y estilo de los documentos mencionados, que es para lo que se diseñó principalmente.

Jerarquía de DOM.

De esta manera el DOM permite acceso dinámico a través de la programación para acceder, añadir y cambiar dinámicamente contenido estructurado en documentos con lenguajes como ECMAScript (JavaScript). De hecho DOM como interfaz de plataforma multilingüe (cross-language API) para acceder y modificar documentos XML es independiente del lenguaje, de esta manera una variedad de lenguajes de programación utilizan DOM cotidianamente —como C++ y Python entre otros— a medida que interactúan con XML para diversas tareas.[2] [3]

La estandarización principal del DOM corre a cargo del World Wide Web Consortium (W3C).

Historia

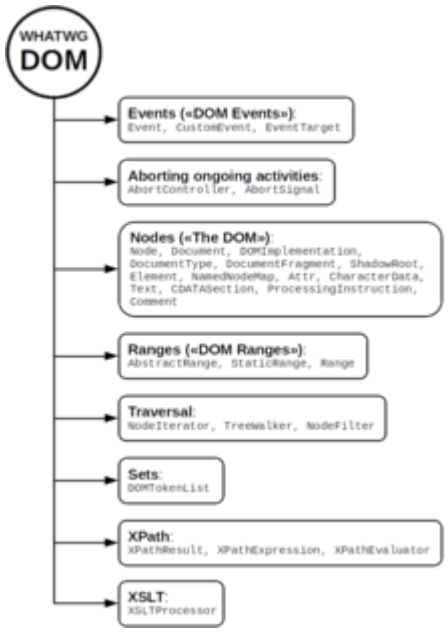
El DOM nació como un conjunto de objetos que representan a un documento HTML, y que actúan como una interfaz entre JavaScript y el propio documento, su origen es Netscape Navigator de la compañía Netscape Communications. Pensado para detectar eventos generados por el usuario y modificar al documento HTML, a su primera generación se le conoce como DOM Nivel 0 o legado (DOM Level 0 o Legacy DOM).

Surgió originalmente bajo la influencia de al menos dos desarrollos que han dado forma significativa al mundo de la informática en el pasado reciente. Ambos se basan en la necesidad de poder acceder de manera fácil y uniforme a los datos estructurados en los documentos HTML y XML.[4]

A mediados de la década de 1990, a medida que aumentaba la popularidad de la World Wide Web, se inventó el lenguaje de secuencias de comandos JavaScript y, desde entonces, los navegadores web incluyen intérpretes que ejecutan dichas secuencias de comandos.

Internet Explorer de Microsoft le siguió a Netscape Navigator, estos dos navegadores incluían soporte para HTML dinámico (DHTML)[5] , que requería extensiones que entonces ofrecía el DOM rudimentario de la época, permitiendo a los desarrolladores web crear páginas con interactividad

del lado del cliente, así el documento ahora podría manipularse a través del DOM; no obstante, el mismo documento no fue representado de la misma manera por los dos navegadores.



Esquema de DOM hecho por WHATWG (Web Hypertext Application Technology Working Group)

Los intérpretes de JavaScript o motor para JavaScript, fueron ampliamente implementados por ambos navegadores. Microsoft en aquel momento decidió implementar, para el entonces Internet Explorer, JScript y VBScript, lo cual dio origen a la llamada "guerras de navegadores" de finales de la década de 1990 entre Netscape Navigator e Internet Explorer.

Entonces JavaScript definió formas rudimentarias de acceder al documento HTML y manejar eventos. Más tarde, diferentes fabricantes de navegadores inventaron diferentes modelos de HTML dinámico (DHTML) que permitían una modificación más extensa de la estructura y apariencia del documento mientras se mostraba en el navegador. Sin embargo, estas diferencias hicieron que el trabajo de los desarrolladores web que querían usar HTML dinámico fuera extremadamente tedioso, ya que a menudo se veían prácticamente obligados a escribir una versión separada para cada navegador. Los primeros estándares DOM del W3C (Consorcio WWW) son, por lo tanto, intentos de fusionar, estandarizar y finalmente reemplazar las diversas técnicas patentadas de JavaScript y DHTML que surgieron durante la guerra de los navegadores. Esto ha tenido tanto éxito que el DOM ahora juega un papel central en la programación de JavaScript.[6]

Para septiembre del 2008, nuevos navegadores aparecen gradualmente en escena bajo el proyecto Chromium, cuyo código base es ampliamente utilizado, tanto por el navegador Chrome como por el más reciente Microsoft Edge. Netscape Navigator desaparece como navegador comercial [7] dejando como una especie de sucesor a Mozilla Firefox.

Estos últimos acontecimientos y la intervención del Consorcio WWW (W3C), marcaron una nueva tendencia en continua evolución, desapareciendo gradualmente los problemas de incompatibilidad con la adopción del DOM estandarizado por el W3C. El DOM (nivel 4) ahora está incorporado en el estándar HTML 5, fue lanzado en diciembre de 2015 y su última actualización fue en diciembre de 2020.[6]

En la actualidad DOM ha permeado todos los lenguajes de programación, independiente del lenguaje, una variedad de lenguajes de programación utilizan DOM cotidianamente, como C++ y Python entre otros, a medida que interactúan con aplicaciones de XML para diversas tareas.[2] [3]

Establecer referencias a objetos

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

Puede utilizarse cualquier lenguaje de programación adecuado para el diseño web. En el caso de JavaScript, cada objeto tiene un nombre, el cual es exclusivo y único. Cuando existe 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 usarlo para hacer referencia a este, por ejemplo:

```
<!- documento html -->
<div id="IdiomasWiki">
    <li>Deutsch</li>
    <li>English</li>
    <li>Français</li>
</div>
```

Para hacer referencia a este elemento se puede usar la función *getElementById*

```
// JavaScript
document.getElementById("IdiomasWiki")
```

Y realizar alguna operación sobre el mismo, en este caso agregamos un nuevo elemento:

```
document.getElementById("IdiomasWiki").innerHTML += "<li>Português</li>"
```

Manipular 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 en la mayoría de 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 in Texto){
    //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 cambia alguna situación 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 del usuario con la computadora.

Cuando alguno de estos eventos ocurre, 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:miFuncion('bar');">
Aquí va otro texto
</div>
```

Siendo el indicador de pseudo-protocolo *javascript:* un agregado opcional, y solo usado por Microsoft Internet Explorer, donde podía configurase por defecto su lenguaje alternativo VBScript.[8]

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

```
objecto.evento = funcion;
//Por ejemplo:
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 pendientes 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 qué 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);
```

Véase también

- HyperText Markup Language (HTML)
- Extensible Markup Language (XML)

- JavaScript (JS)

Notas

1. Tanto XHTML como SVG son aplicaciones de XML —un formato de lenguaje de marcado extensible— así XHTML es esencialmente HTML expresado como XML válido, mientras que SVG es un formato de imagen vectorial basado en XML para definir gráficos bidimensionales.

Referencias

1. «Document Object Model (DOM)» (http://www.w3.org/DOM/#what). http://www.w3.org/: W3C. «*Document Object Model* es una plataforma, y una interfaz independiente del lenguaje, que permite a programas y scripts, acceder y actualizar el contenido, la estructura y el estilo de documentos en forma dinámica. »
2. «The Document Object Model API» (https://docs.python.org/3/library/xml.dom.html). Python Software Foundation. «The Python Standard Library: Structured Markup Processing Tools. »
3. «Program with DOM in C-C++» (https://docs.microsoft.com/en-us/previous-versions/windows/desktop/ms759192(v=vs.85)). Documentación técnica de Microsoft. «This tutorial is intended for C/C++ developers interested in writing XML applications using the DOM. »
4. Weiss-Engelsberger, Roman (2016). *Diseño web universal compatible con W3C*. AV Akademikerverlag. p. 521. ISBN 9783330506824.
5. «Working with DHTML and the DHTML Object Model» (https://docs.microsoft.com/en-us/previous-versions/aa651135(v=vs.71)). Documentación técnica de Microsoft. «*DHTML* uses standard HTML tags to render and manipulate content on a page. »
6. Wang, Paul S. (2012). *Dynamic Web Programming and HTML5* (https://www.taylorfrancis.com/books/mono/10.1201/b13928/dynamic-web-programming-html5-paul-wang). Chapman & Hall CRC. p. 664. ISBN 9780429169625.
7. «Netscape desaparece y pasa el testigo a su «heredero», el navegador Firefox» (https://www.abc.es/tecnologia/redes/abci-netscape-desaparece-y-pasa-testigo-heredero-navegador-firefox-200803030300-1641691782143_noticia.html). Diario ABC, S.L. «*Netscape*, el programa gracias al que millones de personas tuvieron sus primeras experiencias en internet está condenado sin remedio a desaparecer. »
8. «The useless javascript: pseudo-protocol - Crisp's blog - Tweakblogs - Tweakers» (https://crisp.tweakblogs.net/blog/the-useless-javascript-pseudo-protocol.html). *crisp.tweakblogs.net*. Consultado el 28 de julio de 2021.

Bibliografía

- Wang, Paul S (2013). *«Dynamic Web Programming and HTML5»*. New York: Chapman and Hall/CRC. ISBN 9780429169625.
- Weiss-Engelsberger, Roman (2016). *«Diseño web universal compatible con W3C»*. Berlín: AV Akademikerverlag. ISBN 9783330506824.

Enlaces externos

- DOM según el W3C (https://www.w3.org/DOM/)
- DOM según Mozilla (https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model)
- Especificación de DOM Level 1 (http://html.conclase.net/w3c/dom1-es/cover.html) (en español)