Curso HTML, CSS y JavaScript

Junio de 2017

#### Autor: Fran Linde Blázquez

# Tema 7: Trabajando en el DOM con JS

ÍNDICE

[Tema 7: Trabajando en el DOM con JS 1](#_Toc485978956)

[1. Introducción 3](#_Toc485978957)

[2. Buscando en el DOM 4](#_Toc485978958)

[3. Manipulando el DOM 5](#_Toc485978959)

[4. Propiedades de los nodos 6](#_Toc485978960)

## 1. Introducción

El DOM es un conjunto de utilidades específicamente diseñadas para manipular documentos XML, y por tanto documentos HTML.

El DOM transforma el archivo HTML en un árbol de nodos jerárquico, fácilmente manipulable.

Los nodos más importantes son:

* **Document**: Representa el nodo raíz. Todo el documento HTML.
* **Element**: Representa el contenido definido por un par de etiquetas de apertura y cierre, lo que se conoce como un tag HTML. Puede tener como hijos más nodos de tipo Element y también atributos.
* **Attr**: Representa el atributo de un elemento.
* **Text**: Almacena el contenido del texto que se encuentra entre una etiqueta HTML de apertura y cierre.

## 2. Buscando en el DOM

Para poder recorrer el DOM, lo podemos hacer a través de un API JavaScript con métodos para acceder y manipular los nodos. Algunas de estas funciones son:

* **getElementById(id) :** Devuelve el elemento con un id especifico.
* **getElementsByName(name)** Devuelve los elementos que un name (nombre) específico.
* **getElementsByTagName(tagname)** : Devuelve los elementos con un nombre de tag específico.
* **getElementsByClassName(classname) :** Devuelve los elementos con un nombre de clase específico.
* **getAttribute(attributeName) :** Devuelve el valor del atributo con nombre attributeName
* **querySelector(selector) :** Devuelve un único elemento que corresponda con el selector , ya sea por tag, id, o clase.
* **querySelectorAll(selector) :** Devuelve un array con los elementos que correspondan con la query introducida en selector

## 3. Manipulando el DOM

De igual manera podemos manipular el DOM con las siguientes funciones

* **createElement(name):** Crea un elemento HTML con el nombre que le pasemos en el parámetro name.
* **createTextNode(text):** Crea un nodo de texto que puede ser añadido a un elemento HTML.
* **createTextAttribute(attribute):** Crea un atributo que puede ser añadido posteriormente a un elemento HTML.
* **appendChild(node):** Nos permite hacer hijo un elemento a otro.
* **insertBefore(new, target):** Permite insertar un elemento o nodo new antes del indicado en target .
* **removeAttribute(attribute):** Elimina el atributo de nombre attribute del nodo desde el que se le llama.
* **removeChild(child):** Elimina el nodo hijo que se indica con child
* **replaceChild(new, old):** Reemplaza el nodo old por el que se indica en el parámetro new.

## 4. Propiedades de los nodos

Todos los nodos tienen algunas propiedades que pueden ser muy útiles para las necesidades de nuestros desarrollos:

* attributes: Nos devuelve un objeto con todos los atributos que posee un nodo.
* className: Permite setear o devolver el nombre de la clase (para CSS) que tenga el nodo si la tiene.
* id: Igual que className pero para el atributo id
* innerHTML: Devuelve o permite insertar código HTML (incluyendo tags y texto) dentro de un nodo.
* nodeName: Devuelve o nombre del nodo, si es un <div> devolverá DIV .
* nodeValue: Devuelve el valor del nodo. Si es de tipo element devolverá null . Pero por ejemplo si es un nodo de tipo texto, devolverá ese valor.
* style: Permite insertar código CSS para editar el estilo.
* tagName: Devuelve el nombre de la etiqueta HTML correspondiente al nodo. Similar a nodeName, pero solo en nodos de tipo tag HTML.
* title: Devuelve o permite modificar el valor del atributo title de un nodo.
* childNodes : Devuelve un array con los nodos hijos del nodo desde el que se llama.
* firstChild : Devuelve el primer hijo.
* lastChild : Devuelve el último hijo.
* previousSibling : Devuelve el anterior "hermano" o nodo al mismo nivel.
* nextSibling : Devuelve el siguiente "hermano" o nodo al mismo nivel.
* ownerDocument : Devuelve el nodo raíz donde se encuentra el nodo desde el que se llama.
* parentNode : Devuelve el nodo padre del nodo que se llama.

Vemos a continuación un ejemplo de código que utilice varias cosas de las que hemos visto en teoría:

var newElem = document.createElement('div');

newElem.id = 'nuevoElemento';

newElem.className = 'bloque';

newElem.style = 'background:red; width:200px; height:200px';

var body = document.querySelector('body');

body.appendChild(newElem);

Veamos el resultado sobre nuestro HTML:

...

<body>

<div id="nuevoElemento" class="bloque" style="background: red; width: 200px; height: 200px;"></div>

</body>

...

Este ejemplo lo que hace es crear un elemento div con un id de nombre nuevoElemento, una clase bloque y un estilo CSS que define un color de fondo red (rojo) y un ancho y alto de 200px.

Todo el API del DOM nos permite cualquier modificación y edición de elementos HTML, de forma que dinámicamente, por ejemplo por medio de eventos, podemos modificar el aspecto y funcionalidad del documento HTML que estamos visualizando.