Desarrollo Web en Entorno Cliente  
UD 05. Document Object Model (DOM)

short line

Actualizado Octubre 2020

Licencia

**Reconocimiento – NoComercial - CompartirIgual (BY-NC-SA)**: No se permite un uso comercial de la obra original ni de las posibles obras derivadas, la distribución de las cuales se debe hacer con una licencia igual a la que regula la obra original.

Nomenclatura

A lo largo de este tema se utilizarán distintos símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:

**📖 Importante**

**❕ Atención**

**💬 Interesante**

**Índice de contenido**

**1.** [**Introducción**](#_ge43nvu7ywtj) **3**

**2.** [**Modificando atributos**](#_ufc7q4t6mm5g) **3**

**3.** [**Funciones de Javascript para localizar elementos en el DOM**](#_8dceuqivbfx4) **3**

**3.1.** [**getElementById(identificador)**](#_1v21wd2509k7) **3**

**3.2.** [**getElementsByTagName(etiqueta)**](#_f1e6phdy1r) **4**

**3.3.** [**getElementsByName(nombre)**](#_ei6twop484i1) **4**

**4.** [**Funciones para crear/eliminar nodos**](#_naet01avjrg2) **4**

**4.1.** [**removeChild(nodo)**](#_caqv3stdwtvt) **4**

**4.2.** [**appendChild(nodo)**](#_bvr07ehj0m58) **5**

**5.** [**Bibliografía**](#_ech63litfpca) **5**

**6.** [**Autores (en orden alfabético)**](#_g1qlmy5ta6mu) **5**

UD05. Document Object Model (DOM)

# Introducción

El llamado DOM (Document Object Model) es un modelo que permite tratar un documento Web XHTML como si fuera XML, navegando por los nodos existentes que forman la página, pudiendo manipular sus atributos e incluso crear nuevos elementos.

Para más información general de DOM:

* <https://es.wikipedia.org/wiki/Document_Object_Model>
* <http://www.w3schools.com/js/js_htmldom.asp>

Usando Javascript para navegar en el DOM podemos acceder a todos los elementos XHTML de una página. Esto nos permite cambiar dinámicamente el aspecto de nuestras páginas Web.

En este tema vamos a estudiar las principales funciones de Javascript para modificar el DOM.

# Modificando atributos

Cuando obtengamos algún elemento con las funciones que estudiaremos más adelante, podemos manipular los atributos de dicho elemento de la siguiente forma.

|  |
| --- |
| **let** elemento=document.getElementById("miElemento"); elemento.innerHTML="El html interno a cambiar de ese elemento"; |

Los atributos disponibles a modificar pueden depender de cada elemento.

# Funciones de Javascript para localizar elementos en el DOM

Las funciones aquí estudiadas normalmente se usan sobre el elemento “document”, ya que así se aplican a todo el documento.

Aun así, pueden usarse en cualquier nodo XHTML, entonces la búsqueda se realizaría no en todo en el documento, sino en el sub-árbol formado por el elemento en sí y sus hijos.

## getElementById(identificador)

Esta función devuelve un elemento DOM del subárbol cuya identificador sea el indicado en la cadena “identificador”.

<http://www.w3schools.com/jsref/met_document_getelementbyid.asp>

**Ejemplo:**

|  |
| --- |
| **let** myDiv = document.getElementById("miDiv"); console.log("El html de miDiv es "+myDiv.innerHTML); |

## getElementsByTagName(etiqueta)

Esta función devuelve una array con todos los elementos DOM del subárbol cuya etiqueta XHTML sea la indicada en la cadena “etiqueta”.

<http://www.w3schools.com/jsref/met_document_getelementsbytagname.asp>

**Ejemplo:**

|  |
| --- |
| **let** myDiv = document.getElementById("miDiv") **let** losP = myDiv.getElementsByTagName("p"); **let** num = losP.length; console.log("Hay " + num + " <p> elementos en el elemento miDiv"); console.log("En el primer P el HTML asociado es "+losP[0].innerHTML); |

## getElementsByName(nombre)

Esta función devuelve una array con todos los elementos DOM del subárbol cuya atributo name sea el indicado en la cadena “nombre”.

<http://www.w3schools.com/jsref/met_doc_getelementsbyname.asp>

**Ejemplo**:

|  |
| --- |
| **let** elementos = document.getElementsByName("name"); **let** i; *// Todos los textbox que tengan de name alumnos, los marcamos* **for** (i = 0; i < elementos.length; i++) {  **if** (elementos[i].type === "checkbox") {  elementos[i].checked = true;   } } |

# Funciones para crear/eliminar nodos

En esta parte veremos las funciones básicas para crear y eliminar nodos XHTML.

## removeChild(nodo)

Esta función se aplica a un nodo padre. La función recibe un nodo hijo suyo y lo borra. Es útil usarlo con el atributo “parentnode”, que devuelve el nodo padre del elemento que estamos manejando.

<http://www.w3schools.com/jsref/met_node_removechild.asp>

**Ejemplo**:

|  |
| --- |
| **let** parrafo=document.getElementById("miParrafo"); *// Obtiene la referencia del padre, y al padre le aplica la función removeChild*  parrafo.parentnode.removeChild(parrafo); |

## appendChild(nodo)

Esta función se aplica a un nodo padre. La función recibe un nodo y lo incluye como nodo hijo del padre. Se puede combinar con funciones como “createElement”, que permiten crear elementos XHTML.

<http://www.w3schools.com/jsref/met_node_appendchild.asp>

**Ejemplo**:

|  |
| --- |
| *// Creo un nodo de tipo LI* **let** nuevoNodo = document.createElement("LI"); *// Al nodo LI le asocio un texto (también podría asociarse XHTML con innerHTML)* **let** nodoTexto = document.createTextNode("Agua"); nuevoNodo.appendChild(nodoTexto); *// A miLista, lista ya existente, le añado el elemento creado* document.getElementById("miLista").appendChild(nuevoNodo); |

# Bibliografía

[1] Javascript Mozilla Developer <https://developer.mozilla.org/es/docs/Web/JavaScript>

[2] Javascript ES6 W3C <https://www.w3schools.com/js/js_es6.asp>

[3] Referencia DOM

<https://developer.mozilla.org/es/docs/Referencia_DOM_de_Gecko/Introducci%C3%B3n>

# Autores (en orden alfabético)

A continuación ofrecemos en orden alfabético el listado de autores que han hecho aportaciones a este documento:

* García Barea, Sergi