Modificar el DOM

Existen dos técnicas diferentes para modificar el DOM, ambas son capaces de producir los mismos resultados, pero según el contexto una puede resultar más cómoda que la otra:

- **Utilizar código HTML**. Esta técnica se basa en el uso de cadenas de caracteres que contienen código HTML.
- **Utilizar objetos de tipo elemento**. Esta técnica se basa en el uso de objetos de tipo nodo.

Al utilizar código HTML el navegador tiene que interpretar ese código traduciéndolo en nodos DOM y luego actualizar esos nodos en la presentación de la página. Utilizando objetos de tipo nodo nos ahorramos el primer paso, pero irónicamente esto no siempre supone una mejor eficiencia (depende de la implementación de cada navegador).

En la siguiente tabla se describen las propiedades y el método con los que podemos manipular el **DOM** mediante código **HTML**.

Propiedad/método	**Descripción**
innerHTML	Propiedad asociada al código HTML que contiene el elemento.
outerHTML	Propiedad asociada al código HTML del elemento, incluido su contenido.
insertAdjacentHTML(*posición*, *codigo*) 	Este método sirve para insertar código HTML en una posición relativa al elemento actual. Los valores posibles de *posición* son las cadenas 'beforebegin' (inserta el código como hermano mayor del elemento actual), 'afterbegin' (inserta el código como hijo primogénito del elemento actual), 'beforeend' (inserta el código como hijo benjamín del elemento), y 'afterend' (inserta el código como hermano menor del elemento).

En la siguiente tabla se describen los métodos que nos permiten manipular el **DOM** mediante objetos de tipo **elemento**; los dos primeros pertenecen al objeto document, y el resto puede aplicarse sobre objetos de tipo elemento.

Método	**Descripción**
var nuevoNodo= document.createElement(*etiqueta*)	Devuelve un objeto de tipo nodo de elemento y, más concretamente, del tipo de elemento correspondiente a la *etiqueta* **HTML**. Este método crea el nodo, pero no lo inserta en ninguna posición del **DOM**, por lo que no será visible hasta que lo ubiquemos.
var nodoTexto=document.createTextNode(*texto*)	Devuelve un objeto de tipo nodo de texto con el *texto* indicado. Este método crea el nodo, pero no lo inserta en ninguna posición del **DOM**, por lo que no será visible hasta que lo ubiquemos.
nombreNodoPadre.appendChild(*nodoHijo*)	Inserta el *nodo* en el **DOM** como hijo menor del elemento actual.
nodoQueSeraPadreDelNuevoNodo.insertBefore(*nodo1*,*nodo2*)	Inserta el *nodo1* como hermano inmediatamente mayor de *nodo2* dentro del elemento actual.
nodoPadre.removeChild(*nodo*)	Elimina el nodo del **DOM** y lo devuelve como una referencia que podemos recoger en una variable para rescatarlo posteriormente.

nodoPadre.replaceChild(*nuevo*,*viejo*)	Sustituye el nodo *viejo* por el *nuevo* dentro del elemento actual.
var nodoNuevaCopia=nodoACopiar.cloneNode(*incluirDescencientes*)	Devuelve un elemento que es la copia exacta del actual, salvo porque no incluye sus detectores de eventos. *incluirDescendientes* es un valor booleano; si es true duplica el elemento y todos sus nodos descendientes; si es false duplica el nodo de elemento sin ninguno de sus descendientes (ni siquiera los nodos de tipo texto). Hay que tener en cuenta que este argumento es obligatorio en algunos navegadores, mientras que en otros se considera por defecto false.

El proceso de creación de nuevos nodos puede llegar a ser tedioso, ya que implica la utilización de tres funciones DOM:

createElement(etiqueta): crea un nodo de tipo Element que representa al elemento XHTML cuya etiqueta se pasa como parámetro.

createTextNode(contenido): crea un nodo de tipo Text que almacena el contenido textual de los elementos XHTML.

nodoPadre.appendChild(nodoHijo): añade un nodo como hijo de otro nodo. Se debe utilizar al menos dos veces con los nodos habituales: en primer lugar se añade el nodo Text como hijo del nodo Element y a continuación se añade el nodo Element como hijo de algún nodo de la página.

Por ejemplo:

```
001  <!DOCTYPE html>
002  <html>
003          <head>
004                  <title>DOM</title>
005                  <script>
006                          function insertar(){
007                                  var codigoFila = '<td>' + document.getElementById('texto').value +'</td>';
008                                  codigoFila += "<td><button type='button' onclick='clonarEncima(this);'>Clonar
     encima</button></td>";
009                                  codigoFila += "<td><button type='button'
     onclick='eliminar(this);'>Eliminar</button></td>";
010                                  var fila = document.createElement('tr');
011                                  fila.innerHTML = codigoFila;
012                                  document.getElementById('tabla').appendChild(fila);
013                          }
014                          function eliminar(boton){
015                                  var fila = boton.parentNode.parentNode;
016                                  document.getElementById('tabla').removeChild(fila);
017                          }
018                          function clonarEncima(boton){
019                                  var fila = boton.parentNode.parentNode;
020                                  document.getElementById('tabla').insertBefore(fila.cloneNode(true),fila);
021                          }
022                  </script>
023          </head>
024          <body>
025                  <label for='texto'>Texto:</label>
026                  <input id ='texto' type='text'/>
027                  <button type='button' onclick='insertar();'>Insertar en tabla</button>
028                  <table id='tabla' style='border: 3px solid black;'></table>
```

```
029             </body>
030     </html>
```

Observe que en este ejemplo se ha optado por crear el contenido de cada fila mediante código HTML; piense por un momento todo el trabajo adicional que hubiera supuesto definir cada celda, texto y botón como objetos de nodo con los métodos createElement() y createTextNode().

Nota: No siempre es posible crear nodos mediante código HTML pues en algunos elementos y navegadores la propiedad innerHTML es de sólo lectura (por ejemplo, los elementos tbody y tr en Internet Explorer). Ésa es precisamente la causa de que el código anterior no funcione correctamente en Internet Explorer

---