MANIPULACIÓN DEL DOM

1. Modificación de las propiedades de los elementos HTML

Cuando tenemos una referencia a un elemento HTML (recuerda que se trata de un objeto) podremos acceder a sus propiedades para obtener información del mismo o para modificarlo. Por ejemplo, si tenemos una referencia a un párrafo podremos realizar alguna de estas operaciones:

- Leer el contenido del párrafo
- Modificar el contenido del párrafo
- Acceder a los valores de sus atributos (id, name, style, etc.)

Supongamos que tenemos el siguiente código HTML:

¡Hola!, soy un párrafo

Posteriormente, desde JavaScript referenciamos el párrafo con el siguiente código:

```
let miParrafo = document.getElementById("parrafo1");
```

Este objeto expone una serie de propiedades que se corresponden con los respectivos atributos HTML de manera que:

- `miParrafo.innerHTML` ⇒ `Permite leer o modificar el contenido del párrafo (el texto mostrado)`
- `miParrafo.id` ⇒ Permitirá leer o modificar el atributo id del párrafo
- `miParrafo.style` ⇒ Permite acceder al atributo style, por lo que podremos modificar el estilo del párrafo, para por ejemplo cambiar la fuente, color, ocultarlo, etc.

Veamos algunos ejemplos.

```
// Modificar el contenido del párrafo con innerHTML
miParrafo.innerHTML = "Soy un párrafo con distinto texto";
// Mostrar el id del párrafo
alert(miParrafo.id); // Se mostrará "parrafo1"
// Cambiar el color del párrafo a rojo
miParrafo.style.color = "red";
// Cambiar el tamaño del texto a 12 píxeles
miParrafo.style.fontSize = "12px";
// Ocultar el párrafo
miParrafo.style.visibility = "hidden";
// Mostrar un párrafo oculto
miParrafo.style.visibility = "visible";
```

1.1. Modificar el estilo de un elemento HTML

En el apartado anterior hemos visto cómo modificar algunos de los estilos de un párrafo, como el color (mediante miParrafo.style.color) o el tamaño de la fuente (mediante miParrafo.style.fontSize).

Existe una relación entre los nombres de las propiedades usadas en CSS y la correspondiente propiedad en JavaScript. En CSS se usa una notación basada en palabras en minúsculas separadas por guiones (se llama notación Kebab Case), mientras que en JavaScript se usa la notación lowerCamelCase en la que la primera palabra empieza en minúsculas y las siguientes palabras empiezan en mayúscula. Según esto, en CSS tendríamos la propiedad **font-size**, mientras que la respectiva propiedad en JavaScript sería **fontSize**.

Veamos una tabla con más ejemplos:

Propiedad CSS	Sintaxis JavaScript	Ejemplos de uso
`background-color`	**`backgroundColor`**	`elto.style.backgroundColor = "blue";`
`color`	**`color`**	`elto.style.color = "##ff0000"; //rojo`
`font-style`	**`fontStyle`**	`elto.style.fontStyle = "Italic";`
`font-family`	**`fontFamily`**	`elto.style.fontFamily = "Courier";`
`font-size`	**`fontSize`**	`elto.style.fontSize = "12px";`
`height`	**`height`**	`elto.style.heigth = "50px";`
`width`	**`width`**	`elto.style.width = "50px";`
`text-align`	**`textAlign`**	`elto.style.textAlign = "center";`
`text-decoration`	**`textDecoration`**	`elto.style.textDecoration = "underline";`
`display`	**`display`**	`elto.style.display = "none";` `elto.style.display = "inline";`
`visibility`	**`visibility`**	`elto.style.visibility = "hidden";` `elto.style.visibility = "visible";`

2. Acceso y modificación de los atributos de un elemento HTML

Los objetos de tipo element disponen de los siguientes métodos:

- getAttribute(<atributo>) ⇒ Para obtener el valor de un atributo
- setAttribute(<atributo>, <valor>) ⇒ Para modificar el valor de un atributo
- removeAttribute(<atributo>) => Para eliminar un atributo

Veamos algunos ejemplos de uso. Para el siguiente código HTML:

```
<a id="miancla" href="https://www.coit.es">Web COIT</a>
```

Podríamos rescatar el contenido del atributo href mediante el siguiente código:

```
document.querySelector("#miancla").getAttribute("href");
```

Ahora vamos a modificar el contenido del atributo (si no existiera ese atributo este comando lo crearía):

```
document.querySelector("#miancla").setAttribute("href","https://nasa.gov");
```

Para eliminar el atributo usaremos:

```
document.querySelector("#miancla").removeAttribute("href");
```

3. Modos de manipulación del DOM

Podemos manipular el DOM de dos maneras:

- Usando código HTML ⇒ Usaremos datos de tipo string, similar a “ Hola ”
- Usando objetos de tipo Elements ⇒ Mediante la creación de objetos usando instrucciones del tipo document.createElement (“p”)

Se muestra una tabla resumen con las distintas propiedades y métodos disponibles en función del método que elijamos para manipular el DOM:

Usando código HTML	**Usando objetos de tipo Element**
innerHTML	document.createElement(etiqueta)
outerHTML	document.createTextNodo(texto)
insertAdjacentHTML()	appendChild(nodo)
	insertBefore(nodo1,nodo2)
	removeChild(nodo)
	replaceChild(nuevo,viejo)
	cloneNode(incluirDescendientes)

A continuación se muestra un ejemplo que añade nuevos elementos a una lista ordenada usando código HTML y mediante la creación de objetos Elements:

```
1. <!DOCTYPE html>
2. <html lang="en">
3. <head>
4.     <meta charset="UTF-8">
5.     <meta name="viewport" content="width=device-width,
   initial-scale=1.0">
6.     <title>Document</title>
7. </head>
8. <body>
9.     <script>
10.         let contador=1;
11.
12.         function insertarUsandoInnerHTML(){
13.             let parrafo=document.createElement('p');
14.             parrafo.innerHTML=`Párrafo ${contador} creado usando
   innerHTML`;
15.             let refDiv=document.getElementById('contenedor');
16.             refDiv.appendChild(parrafo);
17.
18.             contador++;
19.         }
20.
21.         function insertarUsandoCreateElement(){
22.             let parrafo=document.createElement('p');
```

```
            let texto=document.createTextNode(`Párrafo ${contador}
   creado usando createTextNode`);
            parrafo.appendChild(texto);
            let refDiv=document.getElementById('contenedor');
            refDiv.appendChild(parrafo);

            contador++;
        }
    </script>
    <h1>Ejemplo insertar elementos html de dos maneras</h1>
    <div style="background:yellow" id="contenedor"></div>
    <input type="button" onclick="insertarUsandoInnerHTML();"
   value="Crear usando innerHTML">
    <input type="button" onclick="insertarUsandoCreateElement();"
   value="Crear usando createElement">
</body>
</html>
```

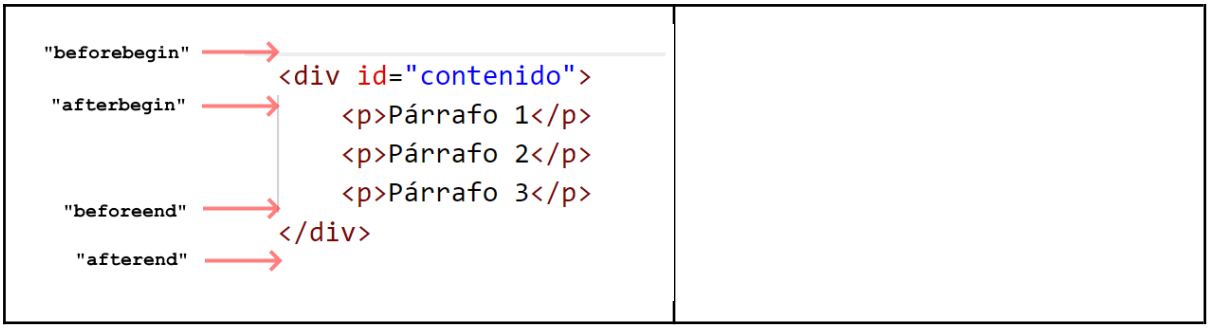
(Código en [ej_manipulacion_dom_codigohtml-elements.html](#)l)

3.1. Manipulación del DOM mediante código HTML

Esta técnica se basa en el uso de cadenas de caracteres que contienen código HTML (dato de tipo string).

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 incluyendo al propio elemento
insertAdjacentHTML(*posición*, *código*) *código*: código HTML (de tipo string) *posición*: • **“beforebegin”** ⇒ Hermano Mayor • **“afterbegin”** ⇒ Hijo primogénito • **“beforeend”** ⇒ Hijo benjamín • **“afterend”** ⇒ Hermano menor	Este método sirve para insertar código HTML en una posición relativa al elemento actual.



En el siguiente ejemplo podemos apreciar cómo usar la propiedad innerHTML para modificar el contenido de un elemento párrafo. También muestra la diferencia entre innerHTML y outerHTML:

```
1. <body>
2.         <script>
3.                 function mostrarInnerHTML() {
4.                         var parrafos = document.getElementsByTagName('p');
5.                         alert("innerHTML: " + parrafos[0].innerHTML);
6.                 }
7.                 function mostrarOuterHTML() {
8.                         var parrafos = document.getElementsByTagName('p');
9.                         alert("outerHTML: " + parrafos[0].outerHTML);
10.                }
11.                function modificarInnerHTML() {
12.                        var parrafos = document.getElementsByTagName('p');
13.                        parrafos[0].innerHTML = "¡Párrafo con nuevo contenido!";
14.                }
15.        </script>
16.        <div id="elementosTag">
17.                <h3>Diferencias entre innerHTML y outerHTML</h3>
18.                <p>Lorem ipsum dolor sit amet</p>
19.                <p>Praesent cursus sollicitudin nunc sit amet pharetra.</p>
20.                <input type="button" onclick="mostrarInnerHTML();"
    value="Mostrar innerHTML">
21.                <input type="button" onclick="mostrarOuterHTML();"
    value="Mostrar outerHTML">
22.                <input type="button" onclick="modificarInnerHTML();"
    value="Modificar innerHTML">
23.        </div>
24. </body>
```

(Código en [ej_innerhtml-outerhtml.html])

Veamos un ejemplo del uso de insertAdjacentHTML:

```
1.  <!DOCTYPE html>
2.  <html lang="en">
3.  <head>
4.      <meta charset="UTF-8">
5.      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6.      <meta http-equiv="X-UA-Compatible" content="ie=edge">
7.      <title>Document</title>
8.  </head>
9.  <body>
10.     <style>
11.     div {background-color: gray;}
12.
13.     </style>
14.     <script>
15.         var contador=1;
16.         function insertarCodigoHTML(){
17.             var posicion=document.getElementById('selectposicion').value;
18.             if(posicion=="beforebegin")  alert("Se va a insertar un hermano mayor a la sección
    div");
19.             else if(posicion=="afterend")  alert("Se va a insertar un hermano menor a la sección
    div");
20.             else if(posicion=="afterbegin")  alert("Se va a insertar un hijo primogénito en la
    sección div");
21.             else if(posicion=="beforeend")  alert("Se va a insertar un hijo benjamín en la sección
    div");
22.
23.             var refdiv=document.getElementById('bloque');
24.             refdiv.insertAdjacentHTML(posicion,"<p> Párrafo " + contador + "</p>");
25.             contador++;
26.         }
27.     </script>
28.     <h1>Ejemplo uso insertAdjacentHTML</h1>
29.     <div id="bloque">
30.         <p>Sección div</p>
31.     </div>
32.     <label for="selectposicion">Posición:</label>
33.     <select  id="selectposicion">
34.         <option value="beforebegin">beforebegin: (hermano mayor)</option>
35.         <option value="afterend">afterend: (hermano menor)</option>
36.         <option value="afterbegin">afterbegin: (hijo primogénito)</option>
37.         <option value="beforeend">beforeend: (hijo benjamín)</option>
38.     </select>
39.     <input type="button" onclick="insertarCodigoHTML();" value="Insertar">
40. </body>
41. </html>
```

(Código en fichero ej_insertadjacentthml.html)

3.2. Manipulación del DOM mediante objetos de tipo Element

La otra manera de manipular el DOM es mediante la creación de objetos de tipo element y su posterior agregación a la página web.

Mediante el método createElement() de objeto DOM podemos crear elementos.
Posteriormente le podemos añadir todos los atributos que necesitemos (lo haremos mediante propiedades que exponen dicho objetos) y finalmente lo agregaremos a la página web.

En la siguiente tabla se describen los métodos que nos permiten manipular el DOM mediante objetos de tipo Element:

Método	**Descripción**

`document.createElement(etiqueta)`
 	
`Ej:`
 	
`document.createElement("p")`
	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.

`document.createTextNode(texto)`
 	
`Ej: document.createTextNode("contenido del párrafo");`
	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.

`elto.appendChild(nodo)`
 	
`elto``: elto. contenedor (p.e. un div)`
`Ej:`
`miDiv.appendChild(miParrafo);`
	Inserta el nodo en el DOM como hijo menor del elemento actual.

`elto.insertBefore(nodo1, nodo2)` **elto**: elto. contenedor (p.e. un div) **nodo1**: el nuevo elto **nodo2**: existente y sobre el que insertamos el nuevo	Inserta el nodo1 como hermano inmediatamente mayor de nodo2 dentro del elemento actual.
`elto.removeChild(nodo)` **elto**: elto. contenedor (p.e. un div) **nodo**: p.e. un párrafo dentro del div	Elimina el nodo del DOM y lo devuelve como una referencia que podemos recoger en una variable para rescatarlo posteriormente.
`elto.replaceChild(nuevo, viejo)` **elto**: elto. contenedor (p.e. un div) **nuevo**: nuevo elto HTML **viejo**: p.e. un párrafo existente en el div	Sustituye el nodo viejo por el nuevo dentro del elemento actual.

`elto.cloneNode(incluirDescendientes)`
 	
incluirDescendientes: **true**/false
 	
elto: cualquier elto HTML (p.e. un párrafo)
 	
Ej:
 	
`miParrafo.cloneNode(true);`
	Devuelve un elemento que es la copia exacta del actual, salvo porque no incluye sus detectores de eventos (los eventos se tratarán posteriormente en este mismo tema).
	`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).

El siguiente ejemplo muestra cómo usar los métodos anteriores para añadir, insertar, reemplazar, clonar y eliminar elementos HTML:

```
1. <!DOCTYPE html>
2. <html lang="es">
3.
4. <head>
5.   <meta charset="UTF-8">
```

```
6.    <title>Ejemplo Interactivo de Manipulación del DOM</title>
7.  </head>
8.
9.  <body>
10.   <h1>Lista de Tareas</h1>
11.   <ul id="lista-tareas">
12.     <!-- Aquí se agregarán las tareas -->
13.   </ul>
14.
15.   <button onclick="agregarTarea()">Agregar Tarea</button>
16.   <button onclick="insertarTarea()">Insertar Tarea al Inicio</button>
17.   <button onclick="reemplazarTarea()">Reemplazar Primera Tarea</button>
18.   <button onclick="clonarTarea()">Clonar Segunda Tarea</button>
19.   <button onclick="eliminarTarea()">Eliminar Última Tarea</button>
20.
21.   <script>
22.
23.     let contador = 1;
24.
25.     // Funciones para cada acción
26.
27.     // Agregar una nueva tarea al final de la lista
28.     function agregarTarea() {
29.       // Obtener el elemento de la lista donde se van a añadir las tareas
30.       const lista = document.getElementById("lista-tareas");
31.       const nuevaTarea = document.createElement("li");
32.       const textoTarea = document.createTextNode(`Nueva Tarea ${contador}`);
33.       nuevaTarea.appendChild(textoTarea);
34.       lista.appendChild(nuevaTarea);
35.       contador++;
36.     }
37.
38.     // Insertar una nueva tarea al inicio de la lista
39.     function insertarTarea() {
40.       // Obtener el elemento de la lista donde se van a añadir las tareas
41.       const lista = document.getElementById("lista-tareas");
42.       const nuevaTarea = document.createElement("li");
43.       const textoTarea = document.createTextNode(`Tarea ${contador}
    insertada al Inicio`);
44.       nuevaTarea.appendChild(textoTarea);
45.       if (lista.firstElementChild) {
46.         lista.insertBefore(nuevaTarea, lista.firstElementChild);
47.       } else {
48.         lista.appendChild(nuevaTarea);
49.       }
50.       contador++;
51.     }
```

```
52.
53.     // Reemplazar la primera tarea de la lista
54.     function reemplazarTarea() {
55.       // Obtener el elemento de la lista donde se van a añadir las tareas
56.       const lista = document.getElementById("lista-tareas");
57.       if (lista.firstElementChild) {
58.         const nuevaTarea = document.createElement("li");
59.         const textoTarea = document.createTextNode(`Tarea Reemplazada
   ${contador}`);
60.         nuevaTarea.appendChild(textoTarea);
61.         lista.replaceChild(nuevaTarea, lista.firstElementChild);
62.       } else {
63.         alert("No hay tareas para reemplazar.");
64.       }
65.       contador++;
66.     }
67.
68.     // Clonar la segunda tarea y agregarla al final de la lista
69.     function clonarTarea() {
70.       // Obtener el elemento de la lista donde se van a añadir las tareas
71.       const lista = document.getElementById("lista-tareas");
72.       if (lista.children.length >= 2) {
73.         const segundaTarea = lista.children[1];
74.         const tareaClonada = segundaTarea.cloneNode(true);
75.         lista.appendChild(tareaClonada);
76.       } else {
77.         alert("No hay suficientes tareas para clonar.");
78.       }
79.     }
80.
81.     // Eliminar la última tarea de la lista
82.     function eliminarTarea() {
83.       // Obtener el elemento de la lista donde se van a añadir las tareas
84.       const lista = document.getElementById("lista-tareas");
85.       if (lista.lastElementChild) {
86.         lista.removeChild(lista.lastElementChild);
87.       } else {
88.         alert("No hay tareas para eliminar.");
89.       }
90.     }
91.   </script>
92. </body>
93.
94. </html>
```

(Código en fichero ej_manipular_dom_elements.html)

Veamos otro ejemplo más, en el que se mezcla el acceso que permite añadir, borrar y copiar filas de manera dinámica en una tabla. En este ejemplo se han utilizado las siguientes técnicas:

- Creación de nuevas filas usando métodos HTML
- Navegación por el DOM para referenciar ciertos elementos HTML a partir de otros
- Clonado de nodos
- Mover nodos mediante el método insertBefore()

```
1. <!DOCTYPE html>
2. <html>
3. <head>
4.     <meta charset="uft-8" />
5.     <title>Acceso DOM</title>
6. </head>
7. <body>
8. <span>Texto</span>
9. <input type="text" id="txtCaja">
10. <input type="button" onclick="insertar();" value="Insertar texto nuevo">
11. <table id="tabla" style="border:1px dashed greed;"></table>
12. <script>
13. function insertar(){
14.     var texto=document.getElementById('txtCaja').value;
15.     var codigoFila="<td>" +  texto + "</td>";
16.     codigoFila += "<td><input type='button' onclick='copiarEncima(this);'
    value='Copiar encima'></td>";
17.     codigoFila += "<td><input type='button' onclick='borrar(this);'
    value='Borrar'></td>";
18.
19.     var fila=document.createElement('tr');
20.     fila.innerHTML=codigoFila;
21.     document.getElementById('tabla').appendChild(fila);
22.
23. }
24. function borrar(boton){
25.     var filaABorrar=boton.parentElement.parentElement; // Será un elto tr
26.
27.     // Manera 1 de eliminar el elto tr
28.     // filaABorrar.remove();
29.
30.     // Manera 2 de eliminar el elto tr
31.     document.getElementById('tabla').removeChild(filaABorrar);
32. }
33. function copiarEncima(boton){
34.     var filaActual=boton.parentElement.parentElement;
35.     var filaACopiar=boton.parentElement.parentElement.cloneNode(true);
36.
    document.getElementById('tabla').insertBefore(filaACopiar,filaActual);
37. }
38. </script>
39. </body>
40. </html>
```

(Código en fichero ej_tabla-dinamica.html)