JavaScript

Acceso a los NODOS

Acceso a los nodos

Una vez construido automáticamente el árbol completo de nodos **DOM**, ya es posible utilizar las funciones **DOM** para acceder de forma directa a cualquier nodo del árbol.

Como acceder a un nodo del árbol es equivalente a acceder a *"un trozo"* de la página, una vez construido el árbol, ya es posible manipular de forma sencilla la página: acceder al valor de un elemento, establecer el valor de un elemento, mover un elemento de la página, crear y añadir nuevos elementos, etc.

DOM proporciona dos métodos alternativos para acceder a un nodo específico: acceso a través de sus nodos padre y acceso directo.

Acceso a los nodos

Las funciones que proporciona **DOM** para acceder a un nodo a través de sus nodos padre consisten en acceder al nodo raíz de la página y después a sus nodos hijos y a los nodos hijos de esos hijos y así sucesivamente hasta el último nodo de la rama terminada por el nodo buscado.

Sin embargo, cuando se quiere acceder a un nodo específico, es mucho más rápido acceder directamente a ese nodo y no llegar hasta él descendiendo a través de todos sus nodos padre.

Por ese motivo, no se van a presentar las funciones necesarias para el acceso jerárquico de nodos y se muestran solamente las que permiten acceder de forma directa a los nodos.

Acceso a los nodos

Por último, es importante recordar que el acceso a los nodos, su modificación y su eliminación solamente es posible cuando el árbol **DOM** ha sido construido completamente, es decir, después de que la página XHTML se cargue por completo.

Más adelante se verá cómo asegurar que un código **JavaScript** solamente se ejecute cuando el navegador ha cargado entera la página XHTML.

getElementsByTagName()

Como sucede con todas las funciones que proporciona **DOM**, la función:

getElementsByTagName()

tiene un nombre muy largo, pero que lo hace autoexplicativo.

La función:

getElementsByTagName(nombreEtiqueta)

obtiene todos los elementos de la página XHTML cuya etiqueta sea igual que el parámetro que se le pasa a la función.

getElementsByTagName()

El siguiente ejemplo muestra cómo obtener todos los párrafos de una página XHTML:

```
var parrafos = document.getElementsByTagName("p");
```

El valor que se indica delante del nombre de la función (en este caso, **document**) es el nodo a partir del cual se realiza la búsqueda de los elementos.

En este caso, como se quieren obtener todos los párrafos de la página, se utiliza el valor **document** como punto de partida de la búsqueda.

getElementsByTagName()

El valor que devuelve la función es un array con todos los nodos que cumplen la condición de que su etiqueta coincide con el parámetro proporcionado.

El valor devuelto es un array de nodos **DOM**, no un array de cadenas de texto o un array de objetos normales. Por lo tanto, se debe procesar cada valor del array de la forma que se muestra en las siguientes secciones.

getElementsByTagName()

De este modo, se puede obtener el primer párrafo de la página de la siguiente manera:

```
var primerParrafo = parrafos[0];
```

De la misma forma, se podrían recorrer todos los párrafos de la página con el siguiente código:

```
for(var i=0; i<parrafos.length; i++) {
var parrafo = parrafos[i];
}
```

getElementsByTagName()

La función **getElementsByTagName()** se puede aplicar de forma recursiva sobre cada uno de los nodos devueltos por la función.

En el siguiente ejemplo, se obtienen todos los enlaces del primer párrafo de la página:

```
var parrafos = document.getElementsByTagName("p");
var primerParrafo = parrafos[0];
var enlaces = primerParrafo.getElementsByTagName("a");
```

getElementsByName()

La función **getElementsByName()** es similar a la anterior, pero en este caso se buscan los elementos cuyo atributo name sea igual al parámetro proporcionado. En el siguiente ejemplo, se obtiene directamente el único párrafo con el nombre indicado:

```
var parrafoEspecial = document.getElementsByName("especial");
<p name="prueba">...</p>
<p name="especial">...</p>
<p>...</p>
```

getElementsByName()

Normalmente el atributo **name** es único para los elementos HTML que lo definen, por lo que es un método muy práctico para acceder directamente al nodo deseado.

getElementById()

La función **getElementById()** es la más utilizada cuando se desarrollan aplicaciones web dinámicas. Se trata de la función preferida para acceder directamente a un nodo y poder leer o modificar sus propiedades.

La función **getElementById()** devuelve el elemento XHTML cuyo atributo id coincide con el parámetro indicado en la función. Como el atributo **id** debe ser único para cada elemento de una misma página, la función devuelve únicamente el nodo deseado.

getElementById()

```
var cabecera = document.getElementById("cabecera");
<div id="cabecera">
<a href="/" id="logo">...</a>
</div>
```

La función **getElementById()** es muy importante y muy utilizada en todas las aplicaciones web.

Creación de Nodos

Como se ha visto, un elemento XHTML sencillo, como por ejemplo un párrafo, genera dos nodos: el primer nodo es de tipo **Element** y representa la etiqueta **<p>** y el segundo nodo es de tipo **Text** y representa el contenido textual de la etiqueta **<p>**.

Por este motivo, crear y añadir a la página un nuevo elemento XHTML sencillo consta de cuatro pasos diferentes:

1. Creación de un nodo de tipo **Element** que represente al elemento.
2. Creación de un nodo de tipo **Text** que represente el contenido del elemento.
3. Añadir el nodo **Text** como nodo hijo del nodo **Element**.
4. Añadir el nodo **Element** a la página, en forma de nodo hijo del nodo correspondiente al lugar en el que se quiere insertar el elemento

Creación de Nodos

De este modo, si se quiere añadir un párrafo simple al final de una página XHTML, es necesario incluir el siguiente código JavaScript:

```
// Crear nodo de tipo Element
var parrafo = document.createElement("p");
// Crear nodo de tipo Text
var contenido = document.createTextNode("Hola Mundo!");
// Añadir el nodo Text como hijo del nodo Element
parrafo.appendChild(contenido);
// Añadir el nodo Element como hijo de la pagina
document.body.appendChild(parrafo);
```

Creación de Nodos

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

1. **createElement(etiqueta)**: crea un nodo de tipo Element que representa al elemento XHTML cuya etiqueta se pasa como parámetro.
2. **createTextNode(contenido)**: crea un nodo de tipo Text que almacena el contenido textual de los elementos XHTML.
3. **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.

Borrado de Nodos

Afortunadamente, eliminar un nodo del árbol **DOM** de la página es mucho más sencillo que añadirlo. En este caso, solamente es necesario utilizar la función **removeChild()**:

```
var parrafo = document.getElementById("provisional");
parrafo.parentNode.removeChild(parrafo);
<p id="provisional">...</p>
```

Borrado de Nodos

La función **removeChild()** requiere como parámetro el nodo que se va a eliminar. Además, esta función debe ser invocada desde el elemento padre de ese nodo que se quiere eliminar. La forma más segura y rápida de acceder al nodo padre de un elemento es mediante la propiedad **nodoHijo.parentNode**.

Así, para eliminar un nodo de una página XHTML se invoca a la función **removeChild()** desde el valor **parentNode** del nodo que se quiere eliminar.

Cuando se elimina un nodo, también se eliminan automáticamente todos los nodos hijos que tenga, por lo que no es necesario borrar manualmente cada nodo hijo.

Acceso a los atributos XHTML

Una vez que se ha accedido a un nodo, el siguiente paso natural consiste en acceder y/o modificar sus atributos y propiedades. Mediante DOM, es posible acceder de forma sencilla a todos los atributos XHTML y todas las propiedades CSS de cualquier elemento de la página.

Los atributos XHTML de los elementos de la página se transforman automáticamente en propiedades de los nodos. Para acceder a su valor, simplemente se indica el nombre del atributo XHTML detrás del nombre del nodo.

Acceso a los atributos XHTML

El siguiente ejemplo obtiene de forma directa la dirección a la que enlaza el enlace:

```
var enlace = document.getElementById("enlace");
 alert(enlace.href); // muestra http://www...com
<a id="enlace" href="http://www...com">Enlace</a>
```

En el ejemplo anterior, se obtiene el nodo DOM que representa el enlace mediante la función **document.getElementById().**

A continuación, se obtiene el atributo href del enlace mediante **enlace.href**. Para obtener por ejemplo el atributo id, se utilizaría enlace.id.

Acceso a los atributos XHTML

Las propiedades CSS no son tan fáciles de obtener como los atributos XHTML. Para obtener el valor de cualquier propiedad CSS del nodo, se debe utilizar el atributo **style**.

El siguiente ejemplo obtiene el valor de la propiedad **margin** de la imagen:

```
var imagen = document.getElementById("imagen");
alert(imagen.style.margin);
<img id="imagen" style="margin:0; border:0;" src="logo.png" />
```

Acceso a los atributos XHTML

Si el nombre de una propiedad CSS es compuesto, se accede a su valor modificando ligeramente su nombre:

```
var parrafo = document.getElementById("parrafo");
alert(parrafo.style.fontWeight); // muestra "bold"
<p id="parrafo" style="font-weight: bold;">...</p>
```

Acceso a los atributos XHTML

La transformación del nombre de las propiedades CSS compuestas consiste en eliminar todos los guiones medios (-) y escribir en mayúscula la letra siguiente a cada guión medio. A continuación se muestran algunos ejemplos:

- **font-weight** se transforma en **fontWeight**
- **line-height** se transforma en **lineHeight**
- **border-top-style** se transforma en **borderTopStyle**
- **list-style-image** se transforma en **listStyleImage**

Acceso a los atributos XHTML

El único atributo XHTML que no tiene el mismo nombre en XHTML y en las propiedades DOM es el atributo class. Como la palabra class está reservada por JavaScript, no es posible utilizarla para acceder al atributo class del elemento XHTML.

En su lugar, DOM utiliza el nombre className para acceder al atributo class de XHTML:

```
var parrafo = document.getElementById("parrafo");
alert(parrafo.class); // muestra "undefined“
alert(parrafo.className); // muestra "normal"
<p id="parrafo" class="normal">...</p>
```