4.4.3. Atributos en DOM

Además del tipo de etiqueta HTML y su contenido de texto, DOM permite el acceso directo a todos los atributos de cada etiqueta. Para ello, los nodos de tipo `Element` contienen la propiedad `attributes`, que permite acceder a todos los atributos de cada elemento. Aunque técnicamente la propiedad `attributes` es de tipo `NamedNodeMap`, sus elementos se pueden acceder como si fuera un array. DOM proporciona diversos métodos para tratar con los atributos:

- `getNamedItem(nombre)`, devuelve el nodo cuya propiedad `nodeName` contenga el valor `nombre`.
- `removeNamedItem(nombre)`, elimina el nodo cuya propiedad `nodeName` coincida con el valor `nombre`.
- `setNamedItem(nodo)`, añade el nodo a la lista `attributes`, indexándolo según su propiedad `nodeName`.
- `item(posicion)`, devuelve el nodo que se encuentra en la posición indicada por el valor numérico `posicion`.

Los métodos anteriores devuelven un nodo de tipo `Attr`, por lo que no devuelven directamente el valor del atributo. Utilizando estos métodos, es posible procesar y modificar fácilmente los atributos de los elementosHTML:

```
<p id="introduccion" style="color: blue">Párrafo de prueba</p>

var p = document.getElementById("introduccion");
var elId = p.attributes.getNamedItem("id").nodeValue; // elId = "introduccion"
var elId = p.attributes.item(0).nodeValue;            // elId = "introduccion"
p.attributes.getNamedItem("id").nodeValue = "preintroduccion";

var atributo = document.createAttribute("lang");
atributo.nodeValue = "es";
p.attributes.setNamedItem(atributo);
```

Afortunadamente, DOM proporciona otros métodos que permiten el acceso y la modificación de los atributos de forma más directa:

- `getAttribute(nombre)`, es equivalente a `attributes.getNamedItem(nombre)`.
- `setAttribute(nombre, valor)` equivalente a `attributes.getNamedItem(nombre).value = valor`.
- `removeAttribute(nombre)`, equivalente a `attributes.removeNamedItem(nombre)`.

De esta forma, el ejemplo anterior se puede reescribir utilizando los nuevos métodos:

```
<p id="introduccion" style="color: blue">Párrafo de prueba</p>

var p = document.getElementById("introduccion");
var elId = p.getAttribute("id"); // elId = "introduccion"
p.setAttribute("id", "preintroduccion");
```

4.4.6. Atributos HTML y propiedades CSS en DOM

Los métodos presentados anteriormente para el acceso a los atributos de los elementos, son genéricos de XML. La versión de DOM específica para HTML incluye algunas propiedades y métodos aún más directos y sencillos para el acceso a los atributos de los elementos HTML y a sus propiedades CSS.

La principal ventaja del DOM para HTML es que todos los atributos de todos los elementos HTML se transforman en propiedades de los nodos. De esta forma, es posible acceder de forma directa a cualquier atributo de HTML. Si se considera el siguiente elemento `<img>` de HTML con sus tres atributos:

```
<img id="logo" src="logo.gif" border="0" />
```

Empleando los métodos tradicionales de DOM, se puede acceder y manipular cada atributo:

```
var laImagen = document.getElementById("logo");

// acceder a los atributos
var archivo = laImagen.getAttribute("src");
var borde = laImagen.getAttribute("border");

// modificar los atributos
laImagen.setAttribute("src", "nuevo_logo.gif");
laImagen.setAttribute("border", "1");
```

La ventaja de la especificación de DOM para HTML es que permite acceder y modificar todos los atributos de los elementos de forma directa:

```
var laImagen = document.getElementById("logo");

// acceder a los atributos
var archivo = laImagen.src;
var borde = laImagen.border;

// modificar los atributos
laImagen.src = "nuevo_logo.gif";
laImagen.border = "1";
```

Las ventajas de utilizar esta forma de acceder y modificar los atributos de los elementos es que el código resultante es más sencillo y conciso. Por otra parte, algunas versiones de Internet Explorer no implementan correctamente el método `setAttribute()`, lo que provoca que, en ocasiones, los cambios realizados no se reflejan en la página HTML.
La única excepción que existe en esta forma de obtener el valor de los atributos HTML es el atributo `class`. Como la palabra `class` está reservada por JavaScript para su uso futuro, no es posible utilizarla para acceder al valor del atributo `class` de HTML. La solución consiste en acceder a ese atributo mediante el nombre alternativo `className`:

```
<p id="parrafo" class="normal">...</p>

var parrafo = document.getElementById("parrafo");
alert(parrafo.class);     // muestra "undefined"
alert(parrafo.className); // muestra "normal"
```

El acceso a las propiedades CSS no es tan directo y sencillo como el acceso a los atributos HTML. En primer lugar, los estilos CSS se pueden aplicar de varias formas diferentes sobre un mismo elemento HTML. Si se establecen las propiedades CSS mediante el atributo `style` de HTML:

```
<p id="parrafo" style="color: #C00">...</p>
```

El acceso a las propiedades CSS establecidas mediante el atributo `style` se realiza a través de la propiedad `style`del nodo que representa a ese elemento:

```
var parrafo = document.getElementById("parrafo");
var color = parrafo.style.color;
```

Para acceder al valor de una propiedad CSS, se obtiene la referencia del nodo, se accede a su propiedad `style` y a continuación se indica el nombre de la propiedad CSS cuyo valor se quiere obtener. Aunque parece lógico que en el ejemplo anterior el valor obtenido sea `#C00`, en realidad cada navegador obtiene el mismo valor de formas diferentes:

- Firefox y Safari muestran el valor `rgb(204, 0, 0)`
- Internet Explorer muestra el valor `#c00`
- Opera muestra el valor `#cc0000`

En el caso de las propiedades CSS con nombre compuesto (`font-weight`, `border-top-style`, `list-style-type`, etc.), para acceder a su valor es necesario modificar su nombre original eliminando los guiones medios y escribiendo en mayúsculas la primera letra de cada palabra que no sea la primera:

```
var parrafo = document.getElementById("parrafo");
var negrita = parrafo.style.fontWeight;
```

El nombre original de la propiedad CSS es `font-weight`. Para obtener su valor mediante JavaScript, se elimina el guión medio (`fontweight`) y se pasa a mayúsculas la primera letra de cada palabra que no sea la primera (`fontWeight`).
Si el nombre de la propiedad CSS está formado por tres palabras, se realiza la misma transformación. De esta forma, la propiedad `border-top-style` se accede en DOM mediante el nombre `borderTopStyle`.
Además de obtener el valor de las propiedades CSS, también es posible modificar su valor mediante JavaScript. Una vez obtenida la referencia del nodo, se puede modificar el valor de cualquier propiedad CSS accediendo a ella mediante la propiedad `style`:

```
<p id="parrafo">...</p>

var parrafo = document.getElementById("parrafo");
parrafo.style.margin = "10px";   // añade un margen de 10px al párrafo
parrafo.style.color = "#CCC";    // modifica el color de la letra del párrafo
```

Todos los ejemplos anteriores hacen uso de la propiedad `style` para acceder o establecer el valor de las propiedadesCSS de los elementos. Sin embargo, esta propiedad sólo permite acceder al valor de las propiedades CSSestablecidas directamente sobre el elemento HTML. En otras palabras, la propiedad `style` del nodo sólo contiene el valor de las propiedades CSS establecidas mediante el atributo `style` de HTML.
Por otra parte, los estilos CSS normalmente se aplican mediante reglas CSS incluidas en archivos externos. Si se utiliza la

propiedad `style` de DOM para acceder al valor de una propiedad CSS establecida mediante una regla externa, el navegador no obtiene el valor correcto:

```
// Código HTML
<p id="parrafo">...</p>

// Regla CSS
#parrafo { color: #008000; }

// Código JavaScript
var parrafo = document.getElementById("parrafo");
var color = parrafo.style.color;  // color no almacena ningún valor
```

Para obtener el valor de las propiedades CSS independientemente de cómo se hayan aplicado, es necesario utilizar otras propiedades de JavaScript. Si se utiliza un navegador de la familia Internet Explorer, se hace uso de la propiedad `currentStyle`. Si se utiliza cualquier otro navegador, se puede emplear la función `getComputedStyle()`.

```
// Código HTML
<p id="parrafo">...</p>

// Regla CSS
#parrafo { color: #008000; }

// Código JavaScript para Internet Explorer
var parrafo = document.getElementById("parrafo");
var color = parrafo.currentStyle['color'];

// Código JavaScript para otros navegadores
var parrafo = document.getElementById("parrafo");
var color = document.defaultView.getComputedStyle(parrafo,
'').getPropertyValue('color');
```

La propiedad `currentStyle` requiere el nombre de las propiedades CSS según el formato de JavaScript (sin guiones medios), mientras que la función `getPropertyValue()` exige el uso del nombre original de la propiedad CSS. Por este motivo, la creación de aplicaciones compatibles con todos los navegadores se puede complicar en exceso.

A continuación se muestra una función compatible con todos los navegadores, creada por el programador Robert Nyman y publicada en su blog personal:

```
function getStyle(elemento, propiedadCss) {
  var valor = "";
  if(document.defaultView && document.defaultView.getComputedStyle){
    valor = document.defaultView.getComputedStyle(elemento,
'').getPropertyValue(propiedadCss);
  }
  else if(elemento.currentStyle) {
    propiedadCss = propiedadCss.replace(/\-(\w)/g, function (strMatch, p1) {
      return p1.toUpperCase();
    });
    valor = elemento.currentStyle[propiedadCss];
  }
  return valor;
}
```

Utilizando la función anterior, es posible rehacer el ejemplo para que funcione correctamente en cualquier navegador:

```
// Código HTML
<p id="parrafo">...</p>

// Regla CSS
#parrafo { color: #008000; }

// Código JavaScript para cualquier navegador
var parrafo = document.getElementById("parrafo");
var color = getStyle(parrafo, 'color');
```

* Extraido de la página librosweb.es