ceviche.js

Apéndice A: Referencia DOM

Interfaz `Node`

Todos los elementos HTML en el DOM heredan de la interfaz `Node`, incluyendo el objeto `document`. Esta interfaz contiene métodos utilizados para realizar operaciones con otros elementos HTML, como agregar, reemplazar, eliminar o verificar si algún elemento está anidado dentro de otro.

`appendChild`

Agrega un nodo al final de la lista de sus nodos hijos.

`cloneNode`

Clona un nodo, con o sin nodos hijos, dependiendo del valor booleano que se le pase como parámetro.

`compareDocumentPosition`

Compara la posición de un nodo con respecto a otro. El valor devuelto por `compareDocumentPosition` es un bitmask, el cual puede tener uno o más de los siguientes valores:

Nombre	Valor
`DOCUMENT_POSITION_DISCONNECTED`	1
`DOCUMENT_POSITION_PRECEDING`	2
`DOCUMENT_POSITION_FOLLOWING`	4
`DOCUMENT_POSITION_CONTAINS`	8
`DOCUMENT_POSITION_CONTAINED_BY`	16

Nombre	Valor
`DOCUMENT_POSITION_IMPLEMENTATION_SPECIFIC`	32

Un bitmask es una forma de representar múltiples valores con un único valor, mediante operaciones de bits. En este caso, los valores posibles, y sus posibles combinaciones, devuelven valores irrepetibles. Por ejemplo:

```
document.compareDocumentPosition(document.body);
// 20

document.body.compareDocumentPosition(document);
// 10
```

En el primer caso, el valor devuelto (20), resulta de la suma de `DOCUMENT_POSITION_CONTAINED_BY` (16) y `DOCUMENT_POSITION_FOLLOWING` (4). Esto significa, respectivamente, que el nodo comparado (`document.body`) está contenido por `document` y que el nodo comparado le sigue a `document`.

De igual forma, en el segundo caso el valor devuelto es 10, resultado de la suma de `DOCUMENT_POSITION_CONTAINS` y `DOCUMENT_POSITION_PRECEDING`, lo cual implica que el nodo comparado (`document`) contiene a `document.body` y, a su vez, el nodo comparado es anterior en posición a `document.body`.

`contains`

Devuelve un valor booleano de si un nodo está contenido dentro de otro. Siguiendo el ejemplo anterior:

```
document.contains(document.body);
// true

document.body.contains(document);
// false
```

```
document.body.contains(document.head);
// false

document.body.contains(document.body);
// true
```

Este método tiene la particularidad de devolver `true` en caso ambos nodos sean el mismo. Para el caso de nodos hermanos, devuelve `false`.

hasChildNodes

Verifica si un nodo tiene nodos hijos.

insertBefore

Agrega un elemento a la lista de nodos hijos de un nodo, con la diferencia que toma un nodo de referencia para insertarlo antes de él (a diferencia de `appendChild`, que siempre lo agrega al final de la lista).

isEqualNode

Verifica si un nodo es igual a otro. Para que sean iguales deben cumplir ciertas condiciones:

- Ambos nodos son del mismo tipo
- Los siguientes atributos tienen el mismo valor en ambos nodos: `nodeName`, `localName`, `namespaceURI`, `prefix`, `nodeValue`.
- El resto de atributos (accesibles mediante la propiedad `attributes`) deben ser iguales.
- Sus nodos hijos deben ser iguales.

```
var body = document.getElementsByTagName('body').item(0);
document.body.isEqualNode(body);
// true

var div1 = document.createElement('div');
```

```
var div2 = document.createElement('div');
div1.isEqualNode(div2);
// true
```

isSameNode

Verifica si tanto un nodo como el otro referencian al mismo objeto en el árbol del DOM. Dos nodos que referencien a un mismo objeto son iguales (`isEqualNode` devolverá `true`), pero no sucede de forma inversa.

```
var body = document.getElementsByTagName('body').item(0);
document.body.isSameNode(body);
// true

var div1 = document.createElement('div');
var div2 = document.createElement('div');
div1.isEqualNode(div2);
// true

div1.isSameNode(div2);
// false
```

normalize

Normaliza un nodo y sus nodos hijos. En una forma normalizada, un nodo no tiene nodos de texto vacíos ni existen dos o más nodos de texto adyacentes.

removeChild

Elimina un nodo de la lista de sus nodos hijos. El método devuelve una referencia al nodo eliminado, el cual sigue existiendo en memoria pero ya no es parte del árbol del DOM y que puede ser reutilizado luego.

replaceChild

Reemplaza un nodo hijo con otro. El método devuelve una referencia al nodo que

ha sido reemplazado, al igual que `removeChild`.

`hasAttributes`

Verifica si un nodo tiene atributos o no.

Así mismo, todas las instancias de `Node` tienen las siguientes propiedades:

`childNodes`

Devuelve una lista "viva" instancia de `NodeList` con todos sus nodos hijos.

`firstChild`

Devuelve el primer nodo hijo.

`lastChild`

Devuelve el último nodo hijo.

`parentNode`

Devuelve el nodo padre.

`parentElement`

Similar a `parentNode`, excepto que solo devuelve un nodo si el nodo padre es una instancia de `Element`. En caso contrario, devuelve `null`.

`nextSibling`

Devuelve el nodo hermano siguiente. Dos nodos son hermanos cuando están al mismo nivel en un árbol y comparten el mismo nodo padre.

`previousSibling`

Devuelve el nodo hermano anterior.

`nodeName`

Devuelve el nombre de la etiqueta.

`localName`

En HTML, el valor de `localName` es similar a `nodeName` pero en minúsculas.

`nodeValue`

Si el nodo es texto, comentario o atributo, devuelven sus correspondientes valores. Para elementos, o para `document`, devuelve `null`.

`nodeType`

Devuelve un número, correspondiente al tipo de nodo que es:

Nombre	Valor
`ELEMENT_NODE`	1
`ATTRIBUTE_NODE`	2
`TEXT_NODE`	3
`CDATA_SECTION_NODE`	4
`ENTITY_REFERENCE_NODE`	5
`ENTITY_NODE`	6
`PROCESSING_INSTRUCTION_NODE`	7
`COMMENT_NODE`	8
`DOCUMENT_NODE`	9
`DOCUMENT_TYPE_NODE`	10
`DOCUMENT_FRAGMENT_NODE`	11
`NOTATION_NODE`	12

`textContent`

Devuelve una cadena con el contenido de un nodo, resultado de unir todos los valores de `textContent` de sus nodos hijos. Si un nodo solo tiene un nodo de

texto como nodo hijo, el valor de `textContent` es una cadena con el contenido del texto.

Interfaz `Element`

Todos los elementos HTML en el DOM heredan de la interfaz `Element`, la misma que hereda de la interfaz `Node`. Esta interfaz contiene métodos utilizados para realizar operaciones dentro de un elemento, como manipular atributos u obtener elementos anidados según criterios como clase o selector CSS, o nombre de etiqueta.

`getAttribute`

Devuelve una cadena con el valor de un atributo definido como parámetro.

`setAttribute`

Define o reemplaza un atributo, con su respectivo valor.

`removeAttribute`

Elimina un atributo definido como parámetro.

`hasAttribute`

Verifica si un atributo está definido en un elemento.

`getAttributeNode`

Similar a `getAttribute`, excepto que en vez de devolver una cadena devuelve una instancia de `Attr`.

`setAttributeNode`

Similar a `setAttribute`, excepto que en vez de asignar un nombre y valor, se debe asignar una instancia de `Attr`.

`removeAttributeNode`

Similar a `removeAttribute`, excepto que debe pasarse como parámetro una instancia de `Attr`.

`getElementsByTagName`

Devuelve una instancia de `NodeList`* con todos los nodos hijos cuyo nombre de etiqueta sea el pasado como parámetro.

*En Firefox y otros navegadores basados en Gecko, `getElementsByTagName` devuelve una instancia de `HTMLCollection`. Aunque son interfaces diferentes, `NodeList` y `HTMLCollection` tienen implementados el método `item`, que permite acceder a un elemento de la lista mediante índices, de igual forma que un arreglo. Así mismo, ambos permiten acceder a los elementos de su lista mediante corchetes.

`HTMLCollection` tiene un método `namedItem`, que permite obtener un elemento HTML de la lista a través de su nombre o id, y no de su índice.

En el caso de `getElementsByTagName`, la lista que devuelve (ya sea instancia de `NodeList` o `HTMLCollection`) es una lista "viva".

`scrollIntoView`

Desplaza la vista del navegador hasta el elemento que llama a este método. Acepta un parámetro, el cual si es `true`, alínea el elemento a la parte superior de la vista del navegador, mientras que si es `false`, lo alinea a su parte inferior.

`getElementsByClassName`

Devuelve una instancia de `NodeList` con todos los nodos hijos que tengan todas las clases pasadas como parámetro. Estas clases deben estar en una cadena y separadas por un espacio.

Al igual que con `getElementsByTagName`, Firefox y otros navegadores basados en Gecko devuelven una instancia de `HTMLCollection`. Así mismo, la lista devuelta por `getElementsByClassName` es una lista "viva".

`insertAdjacentHTML`

Convierte una cadena con HTML en un nodo o conjunto de nodos, según el contenido de la cadena, e inserta el resultado en una posición dada con respecto al nodo. El valor de la posición está definida por los siguientes valores:

- `beforebegin`: Antes del elemento
- `afterbegin`: Antes del primer nodo hijo del elemento
- `beforeend`: Después del último nodo hijo del elemento
- `afterend`: Después del elemento

`querySelector`

Devuelve el primer elemento hijo que concuerde con un selector CSS dado.

`querySelectorAll`

Devuelve una lista instancia de `NodeList` con todos los elementos hijos que concuerden con el selector CSS que se le pase como parámetro. En este caso, el resultado de `querySelectorAll` no es una lista "viva".

`matchesSelector`

Verifica si un elemento concuerda con un selector CSS dado.

Al ser un método relativamente nuevo y en modo experimental, los navegadores le agregan un prefijo según cada implementación: `webkitMatchesSelector`, `mozMatchesSelector`, `oMatchesSelector` o `msMatchesSelector`

`getClientRects`

Devuelve una lista instancia de `ClientRectList` con objetos `TextRectangle`, los cuales indican tanto el tamaño del elemento como la posición con respecto al *viewport* (la zona visible del documento en el navegador). Si el elemento es *inline* (propiedad `display: inline` en CSS), `getClientRects` devolverá un objeto `ClientRect` por cada línea de texto dentro del elemento.

`getBoundingClientRect`

Devuelve una instancia de `TextRectangle` (`ClientRect` en Webkit y `DOMRect` en Gecko) con el tamaño del elemento y su posición con respecto al

viewport.

`remove`

Elimina un elemento del árbol del DOM. Este método no devuelve el elemento eliminado, por lo que se si desea tenerlo guardado en memoria primero se debe guardar una referencia del nodo a eliminar en una variable.

Así mismo, tiene las siguientes propiedades:

`attributes`

Devuelve una instancia de `NamedNodeMap` con todos los atributos de un elemento.

`children`

Devuelve una lista "viva" instancia de `HTMLCollection` con los nodos hijos del elemento que son elementos.

`childElementCount`

Devuelve el número de nodos hijos que son elementos.

`firstElementChild`

Devuelve el primer nodo hijo que es un elemento.

`lastElementChild`

Devuelve el último nodo hijo que es un elemento.

`nextElementSibling`

Devuelve el siguiente nodo hermano que sea un elemento.

`previousElementSibling`

Devuelve el anterior nodo hermano que sea un elemento.

`classList`

Devuelve una lista instancia de `DOMTokenList` con todas las clases asignadas al elemento.

`className`

Devuelve una cadena con el valor del atributo `class`.

`clientHeight`

Devuelve el alto de un elemento, incluyendo el *padding* superior e inferior (definido por la propiedad `padding` en CSS), pero sin contar el espacio de la barra de desplazamiento (si la hubiera), ni el margen (`margin` en CSS) o tamaño de borde (`border-width` en CSS).

`clientLeft`

Devuelve el ancho del borde izquierdo de un elemento, incluyendo el ancho de la barra de desplazamiento vertical, si la hubiera.

`clientTop`

Devuelve el alto del borde superior de un elemento. No incluye ni margen ni *padding* superiores.

`clientWidth`

Devuelve el ancho de un elemento, incluyendo el *padding* a la izquierda y derecha, pero sin contar el espacio de la barra de desplazamiento (si la hubiera), bordes o márgenes.

`id`

Devuelve una cadena con el valor del atributo `id`.

`innerHTML`

Devuelve una cadena con el contenido de un elemento en forma de HTML.

`outerHTML`

Similar a `innerHTML`, pero incluye las etiquetas HTML del elemento.

`scrollHeight`

Devuelve el alto de un elemento, incluyendo su *padding* superior e inferior, pero no sus márgenes. Si el contenido del elemento sobrepasa un alto definido previamente y aparecen barras de desplazamiento, `scrollHeight` devolverá el alto del contenido total, incluyendo el que está visible y el que no.

`scrollLeft`

Devuelve la posición del contenido del elemento con respecto al borde izquierdo del elemento.

`scrollTop`

Devuelve la posición del contenido del elemento con respecto al borde superior del elemento.

`scrollWidth`

Devuelve el ancho de un elemento, incluyendo su *padding* derecho e izquierdo, pero no sus márgenes. Si el contenido del elemento sobrepasa un ancho definido previamente y aparecen barras de desplazamiento, `scrollWidth` devolverá el ancho del contenido total, incluyendo el que está visible y el que no.

`tagName`

Devuelve el nombre de la etiqueta, en mayúsculas. Similar a `nodeName`

Objeto `document`

Entre los atributos que tiene un objeto `document` se encuentran:

`documentElement`

Devuelve un elemento que representa a la etiqueta `html`.

`title`

Devuelve el contenido de la etiqueta `<title></title>`, o uno asignado manualmente.

`head`

Devuelve un elemento que representa a la etiqueta `head`.

`body`

Devuelve un elemento que representa a la etiqueta `body`.

`styleSheets`

Devuelve una lista instancia de `StyleSheetList` con objetos `CSSStyleSheet`, uno por cada hoja de estilos definida (con `<style></style>`) o enlazada al documento (con `<link />`).

`cookie`

Devuelve una cadena con todas las cookies asignadas al documento actual.

`location`

Devuelve un objeto `Location`, el cual contiene información sobre la URL actual del documento. Esta propiedad puede ser cambiada, asignando una cadena con la nueva URL a la cual el navegador direccionará.

`document` también tiene algunas propiedades de la interfaz `Element`, como `children`, `firstElementChild`, `lastElementChild` y `childElementCount`.

Así mismo, tiene métodos que sirven para crear elementos, y acceder a cualquier elemento dentro del documento:

`createAttribute`

Crea un nodo atributo (instancia de `Attr`) con un nombre pasado como parámetro.

`createComment`

Crea un nodo comentario con el contenido pasado como parámetro.

`createElement`

Crea un nodo elemento con el nombre de la etiqueta pasado como parámetro.

`createTextNode`

Crea un nodo texto con el contenido pasado como parámetro.

2. Apéndice B: Bibliotecas de terceros →

Acerca de | Disponible en Amazon.com