DAM ONLINE

DIGI TECH

Centro Oficial FP
Digital & Tech

Árbol de nodos

Una de las tareas habituales en la programación de aplicaciones web con JavaScript consiste en la manipulación de las páginas web. De esta forma, es habitual obtener el valor almacenado por algunos elementos (por ejemplo, los elementos de un formulario), crear un elemento (párrafos, `<div>`, etc.) de forma dinámica y añadirlo a la página, aplicar una animación a un elemento (que aparezca/desaparezca, que se desplace, etc.).

Todas estas tareas habituales son muy sencillas de realizar gracias a DOM. Sin embargo, para poder utilizar las utilidades de DOM, es necesario "*transformar*" la página original. Una página HTML normal no es más que una sucesión de caracteres, por lo que es un formato muy difícil de manipular. Por ello, los navegadores web transforman automáticamente todas las páginas web en una estructura más eficiente de manipular.

Árbol de nodos

DOM transforma todos los documentos HTML en un conjunto de elementos llamados *nodos*, que están interconectados y que representan los contenidos de las páginas web y las relaciones entre ellos. Por su aspecto, la unión de todos los nodos se llama "*árbol de nodos*".

La siguiente página HTML sencilla:

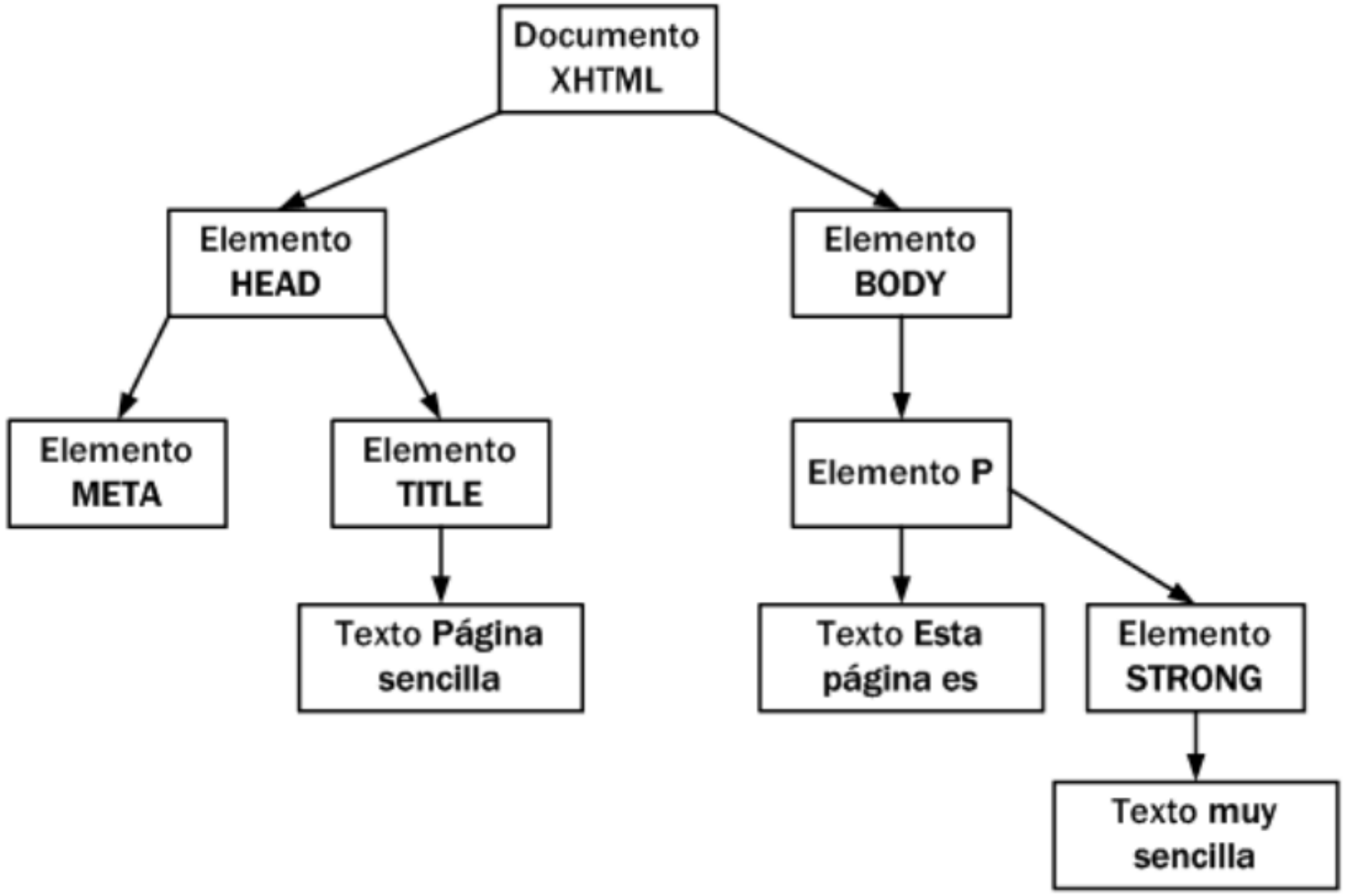
```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Página sencilla</title>
</head>

<body>
    <p>Esta página es <strong>muy sencilla</strong></p>
</body>
```

Árbol de nodos

Se transforma en el siguiente árbol de nodos:



Tipos de nodos

La especificación completa de DOM define 12 tipos de nodos, aunque las páginas HTML habituales se pueden manipular manejando solamente cuatro o cinco tipos de nodos:

- `Document`, nodo raiz del que derivan todos los demás nodos del árbol.
- `Element`, representa cada una de las etiquetas HTML. Se trata del único nodo que puede contener atributos y el único del que pueden derivar otros nodos.
- `Attr`, se define un nodo de este tipo para representar cada uno de los atributos de las etiquetas HTML, es decir, uno por cada par `atributo=valor`.
- `Text`, nodo que contiene el texto encerrado por una etiqueta HTML.
- `Comment`, representa los comentarios incluidos en la página HTML.

Los otros tipos de nodos existentes que no se van a considerar son `DocumentType`, `CDataSection`, `DocumentFragment`, `Entity`, `EntityReference`, `ProcessingInstruction` y `Notation`.

Acceso a nodos

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

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.

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 HTML cuya etiqueta sea igual que el parámetro que se le pasa a la función.

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

```
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.

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 HTML 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.

```
var cabecera = document.getElementById("cabecera");
```

```
<div id="cabecera">
```

querySelector()

El método querySelector () devuelve el primer elemento que coincide con un *selector o selectores CSS* especificados en el documento.

Nota: El método querySelector () solo devuelve el primer elemento que coincide con los selectores especificados. Para devolver todas las coincidencias, utilice el método querySelectorAll () en su lugar.

Si el selector coincide con un ID en el documento que se usa varias veces (tenga en cuenta que un "id" debe ser único dentro de una página y no debe usarse más de una vez), devuelve el primer elemento coincidente.

Ejemplo

Obtenga el primer elemento <p> en el documento:

```
document.querySelector("p");
```

querySelector()

Ejemplo

Obtenga el primer elemento <p> en el documento donde el padre es un elemento .

```
document.querySelector("div > p");
```

El método querySelectorAll () devuelve todos los elementos del documento que coinciden con un selector o selectores CSS especificados, como un objeto NodeList estático.

querySelectorAll()

El objeto NodeList representa una colección de nodos. Se puede acceder a los nodos mediante números de índice. El índice comienza en 0.

Consejo: Puede utilizar el longitud propiedad del objeto NodeList para determinar el número de elementos que coincide con el selector especificado, entonces se puede recorrer todos los elementos y extraer la información que desee.

```
document.querySelectorAll(CSS selectors)
```

Ejemplo

Obtenga todos los elementos <p> en el documento y establezca el color de fondo del primer elemento <p> (índice 0):

```
// Get all <p> elements in the document
var x = document.querySelectorAll("p");

// Set the background color of the first <p> element
x[0].style.backgroundColor = "red";
```

Creación de nodos

Como se ha visto, un elemento HTML 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 HTML 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 HTML, 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

```
// Añadir el nodo Element como hijo de la pagina

document.body.appendChild(parrafo);
```

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 HTML cuya etiqueta se pasa como parámetro.
- `createTextNode(contenido)`: crea un nodo de tipo `Text` que almacena el contenido textual de los elementos HTML.
- `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.

Eliminación 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>
```

La función `removeChild()` requiere como parámetro el nodo que se va a eliminar. Además, esta función debe ser invocada desde el elemento padre de

Eliminación de nodos

Así, para eliminar un nodo de una página HTML 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 directo a nodos

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 HTML y todas las propiedades CSS de cualquier elemento de la página.

Los atributos HTML 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 HTML detrás del nombre del nodo.

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>
```

DIGI TECH

DIGITALENT
GROUP

Centro Oficial FP
Digital & Tech