6. DOM

- *Document Object Model* o DOM es una de las innovaciones que más ha influido en el desarrollo de las páginas web dinámicas y de las aplicaciones web más complejas.
- DOM permite a los programadores web acceder y manipular las páginas HTML como si fueran documentos XML. DOM se diseñó originalmente para manipular de forma sencilla los documentos XML.
- DOM se ha convertido en una utilidad disponible para la mayoría de lenguajes de programación (Java, PHP, JavaScript) y cuyas únicas diferencias se encuentran en la forma de implementarlo.

6.1. Árbol de nodos

- Una página HTML normal no es más que una sucesión de caracteres. No tiene estructura. Es un formato muy difícil de manipular.
- Para poder utilizar las utilidades de DOM es necesario "transformar"la página original en una estructura jerárquica.
- Los navegadores web transforman todas las páginas web en una estructura más eficiente de manipular.
- Esta transformación la realizan todos los navegadores de forma automática y nos permite utilizar las herramientas de DOM de forma muy sencilla.
- Conocer cómo se realiza esta transformación interna (saber cómo es la estructura que se genera) es crucial para poder manipular las páginas HTML.
- 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. El DOM transforma la sucesión de caracteres en un *árbol de nodos*.
- Consideremos la siguiente página:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>Página sencilla</title>
</head>
<body>
<p>Esta página es <strong>muy sencilla</strong></p>
</body>
</html>
```

  se transforma el el árbol de nodos de la figura 1.

- La raíz del árbol de nodos de cualquier página HTML siempre es la misma: un nodo de tipo especial denominado *Documento*.
- A partir de ese nodo raíz, cada etiqueta HTML se transforma en un nodo de tipo *Elemento*. La conversión de etiquetas en nodos se realiza de forma jerárquica. De esta forma, del nodo raíz solamente pueden derivar los nodos `HEAD` y `BODY`. A partir de esta derivación inicial, cada etiqueta HTML se transforma en un nodo que deriva del nodo correspondiente a su "etiqueta padre".
- La transformación automática de la página en un árbol de nodos siempre sigue las mismas reglas:
  - Las etiquetas HTML se transforman en dos nodos: el primero es la propia etiqueta y el segundo nodo es hijo del primero y consiste en el contenido textual de la etiqueta.

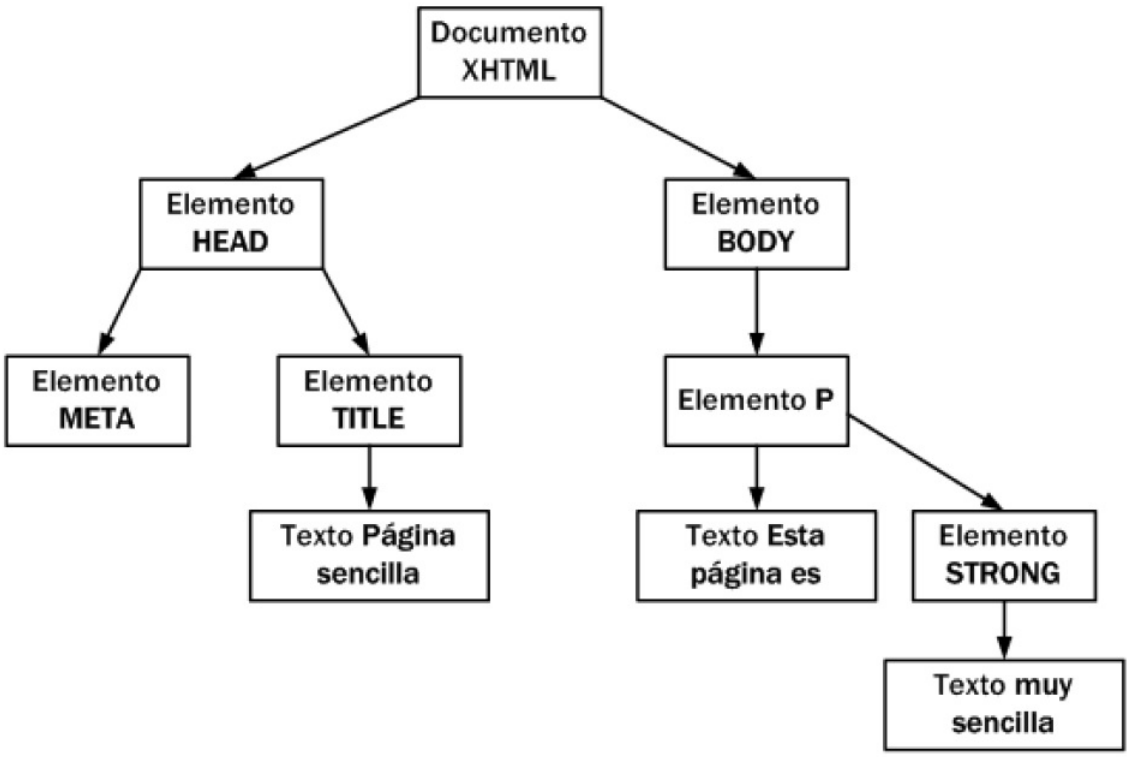

Figura 1: Árbol de nodos generado automáticamente por DOM a partir del código HTML de la página

    - Si una etiqueta HTML se encuentra dentro de otra, se sigue el mismo procedimiento anterior, pero los nodos generados serán nodos hijo de su etiqueta padre.

- Las páginas HTML habituales producen árboles con miles de nodos. Aun así, el proceso de transformación es rápido y automático, siendo las funciones proporcionadas por DOM las únicas que permiten acceder a cualquier nodo de la página de forma sencilla e inmediata.

6.2. 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.
- 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 permiten 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. Poco práctico.
- Cuando se quiere acceder a un nodo específico es mucho más rápido acceder directamente a ese nodo y no llegar a el descendiendo a través de todos sus nodos padre.
- 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 HTML se cargue por completo.

6.2.1. getElementsByTagName()

- 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.
- La instrucción `var parrafos = document.getElementsByTagName("p");` obtiene todos los párrafos de una página.
- 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.
- El valor que devuelve la función es un array con todos los nodos que cumplen la condición (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.
- La instrucción `var primerParrafo = parrafos[0];` obtiene el primer párrafo seleccionado.
- 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");
```

6.2.2. getElementsByName()

- 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>
```

- En el caso de los elementos HTML botón de radio, el atributo name es común a todos los botón de radio que están relacionados, por lo que la función devuelve una colección de elementos.

6.2.3. 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.
- 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.
- Ejemplo:

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

6.2.4. Acceso directo a los atributos HTML

- 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>
```

- 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`.
- 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" />
```

- 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>
```

- 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`.
- El único atributo HTML que no tiene el mismo nombre en HTML 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 HTML. En su lugar, DOM utiliza el nombre `className` para acceder al atributo `class` de HTML:

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

6.3. Manipulando nodos

- La propiedad `innerHTML` permite leer o cambiar el contenido HTML de un elemento.
- Sintaxis:
  - Leer: acceso a la variable `HTMLElementObject.innerHTML`
  - Modificar: `HTMLElementObject.innerHTML=text`

Ejemplos:

```
<!DOCTYPE html>
<html>
<script type="text/javascript">
function myFunction() {
    document.getElementById("demo").innerHTML = "Paragraph changed!";
}
</script>
<body>
<p id="demo" onclick="myFunction()">
    Click me to change my HTML content (innerHTML).</p>
</body>
</html>
```

```
<!DOCTYPE html>
<html>
<script type="text/javascript">
function myFunction() {
```

```
    var x = document.getElementById("myP").innerHTML;
    document.getElementById("demo").innerHTML = x;
}
</script>
<body>
<p id="myP">I am a paragraph.</p>
<p>Click the button get the HTML content of the p element.</p>
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
</body>
</html>
```

```
<!DOCTYPE html>
<html>
<script type="text/javascript">
function myFunction() {
    var x = document.getElementById("myList").innerHTML;
    document.getElementById("demo").innerHTML = x;
}
</script>
<body>
<ul id="myList">
  <li>Coffee</li>
  <li>Tea</li>
</ul>
<p>Click the button get the HTML content of the ul element.</p>
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
</body>
</html>
```

```
<!DOCTYPE html>
<html>
<script type="text/javascript">
document.getElementById("myP").innerHTML = "Hello Dolly.";
document.getElementById("myDIV").innerHTML = "How are you?";
</script>
<body>
<h1>My Web Page</h1>
<p id="myP">This is a p element.</p>
<div id="myDIV">This is a div element.</div>
</body>
</html>
```

```
<!DOCTYPE html>
<html>
<script type="text/javascript">
function myFunction() {
    alert(document.getElementById("demo").innerHTML);
}
</script>
<body>
<p id="demo">Click the button to alert the text of this paragraph.</p>
<button onclick="myFunction()">Try it</button>
</body>
```

```
</html>
```