JS - Code - Document Object Model (DOM)

1. Introducción a DOM (Document Object Model)

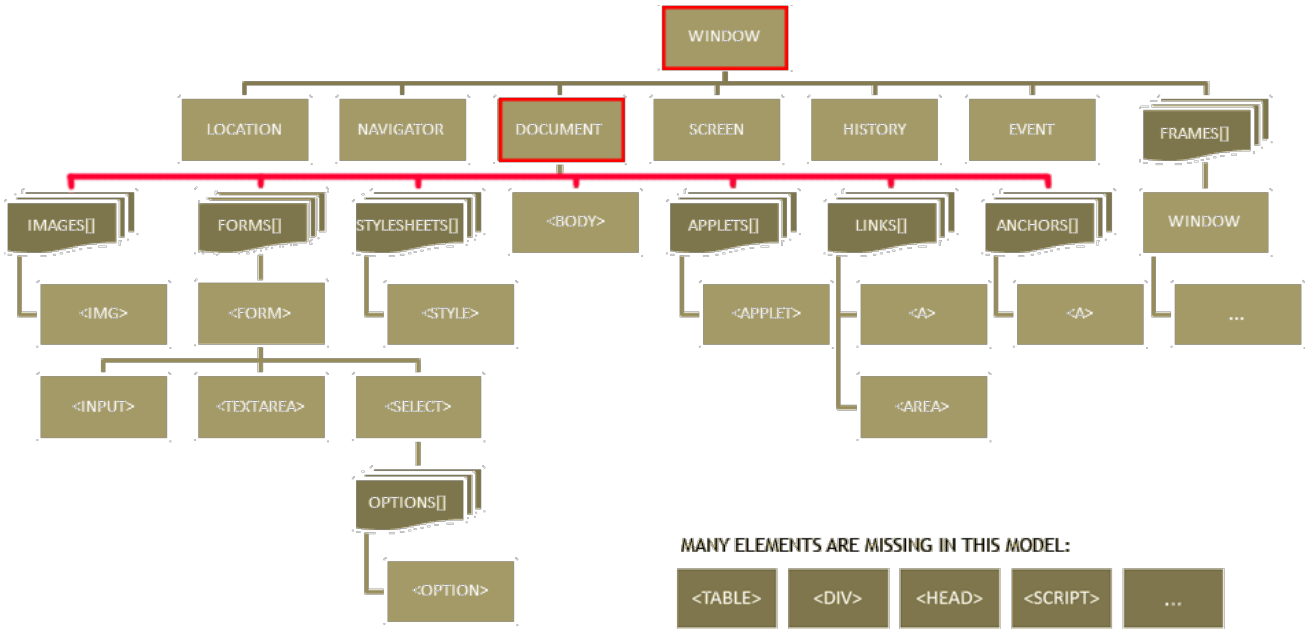
La creación del Document Object Model o DOM es una de las innovaciones que más ha influido en el desarrollo de las páginas web dinámicas y de las aplicaciones web más complejas.

DOM permite a los programadores web acceder y manipular las páginas HTML. A pesar de sus orígenes, 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.

2. Arbol 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 o eliminar un elemento (p, div, img, etc.) de forma dinámica, manipular (poner/quitar/cambiar) una clase de un elemento, etc. Todas estas tareas habituales son muy sencillas de realizar gracias a DOM, pero para ello es necesario "cambiar" la página original.

Los navegadores web transforman automáticamente todas las páginas web en una estructura más eficiente de manipular, el DOM. Esta transformación la realizan todos los navegadores de forma automática y nos permite utilizar las herramientas de DOM de forma muy sencilla. 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 raíz del árbol de nodos de cualquier página HTML siempre es la misma: un nodo de tipo especial denominado document. A partir de ese nodo raíz, cada etiqueta HTML se transforma en un nodo de tipo "Elemento". Como se puede suponer, las páginas HTML habituales producen árboles con miles de nodos. Aun así, el proceso de transformación es rápido y automático.

3. Tipos de nodos

La especificación completa de DOM define 12 tipos de nodos, aunque en las páginas HTML se manipulan habitualmente cuatro tipos de nodos:

Document: nodo raíz 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.

4. Acceso a nodos

Una vez que se ha accedido a un nodo, el siguiente paso 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 XHTML detrás del nombre del nodo.

```
<a href="https://es.wikipedia.org">Wikipedia</a>

var enlace = document.getElementsByTagName('a');
alert(enlace[0].href);   // mostrando la URL  https://es.w...
```

En el ejemplo anterior, se obtiene el nodo DOM que representa el enlace mediante el método document.getElementsByTagName() que devuelve un array con todos los elementos a de la página. A continuación, se obtiene el atributo href del enlace mediante enlace[0].href. Para obtener por ejemplo el atributo target, se utilizaría enlace[0].target.

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

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.

Ejemplo de todo lo dicho:

Respuesta:

```
La url es: https://es.wikipedia.org/
Target es: _blank
class del boton: boton azul
text-align del párrafo: center
src de la imagen: http://localhost:7879/img/ico128.png
```

Como hemos dicho, tenemos varias maneras de acceder a un determinado nodo:

```
//  getElementsByTagName('etiquetaHTML')
//  getElementsByName('nameHTML')
//  getElementById('id')
//  querySelector('...')
```

5. Creación/eliminación de nodos

Cuatro pasos para crear elementos HTML simples:

1) Creación de un nodo de tipo Element que represente al elemento.
   createElement('etiqueta')

2) Creación de un nodo de tipo Text que represente el contenido del elemento.
   createTextNode('contenido')

3) Añadir el nodo Text como nodo hijo del nodo Element.
   nodoPadre.appendChild(nodoHijo)

4) Añadir el nodo Element a la página, en forma de nodo hijo
   nodoPadre.appendChild(nodoHijo) del nodo correspondiente al lugar en el que se quiere insertar el elemento.

Ejemplos

1) Crear nodo de tipo Element
```
var parrafo = document.createElement("p");
```

2) Crear nodo de tipo Text
```
var contenido = document.createTextNode("Hola Mundo!");
```

3) Añadir el nodo Text como hijo del nodo Element
```
parrafo.appendChild(contenido);
```

4) Añadir el nodo Element como hijo de la pagina
```
document.body.appendChild(parrafo);
```

Con este ejemplo creamos 4 párrafos dentro de div id="main"

6. Acceso a los atributos

Añadir y eliminar clases es una de las tareas más comunes que llevamos a cabo al trabajar con JavaScript y el DOM.

Los elementos del DOM tienen una propiedad llamada classList la cual es una matriz que contiene todas las clases que tiene el elemento.
Además, tiene varios métodos adicionales:

- add - Añade una clase.
- remove - Elimina una clase.
- toggle - Añade una clase si está presente y si no, la elimina.
- contains - Revisa si la clase está presente en el elemento.

Añadiendo y eliminando clases

Añadir y eliminar una clase es muy sencillo, simplemente pasa la clase que quieres añadir/eliminar como argumento al método.
Ejemplos: