Especialidad: Técnico en Programación

Módulo III. Submódulo2: Desarrolla aplicaciones que se ejecutan en el cliente.

Prof: Hilda Lucía Rodríguez Gómez

Competencia Profesional: Usa JavaScript para manejar eventos.

DOM (DOCUMENT OBJECT MODEL)

El modelo de documento en objeto (DOM) es una interfaz de programación para los documentos HTML y XML. Facilita una representación estructurada del documento y define de qué manera los programas pueden acceder, al fin de modificar, tanto su estructura, estilo y contenido. El DOM da una representación del documento como un grupo de nodos y objetos estructurados que tienen propiedades y métodos.

REPRESENTACIÓN DOM DE UNA PÁGINA WEB. Una página web es un documento HTML que puede representarse de diferentes maneras:

a) Representación web: como una página web en un navegador donde vemos imágenes, texto, colores, etc.

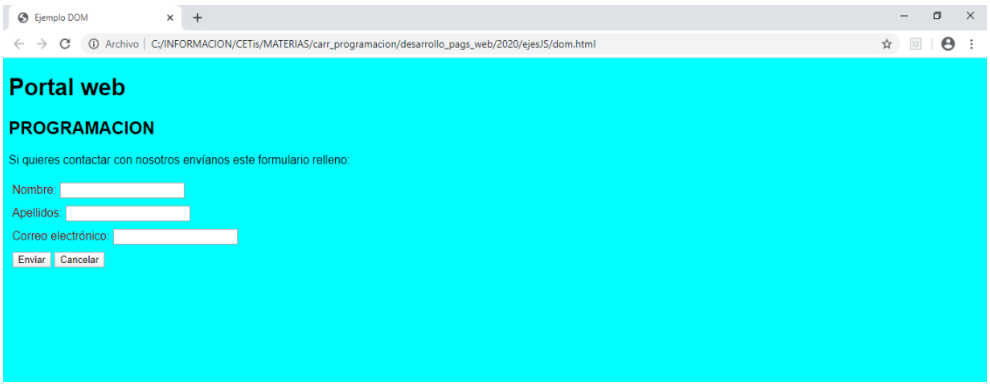
b) Representación texto: como un texto plano (código HTML) que podemos visualizar en cualquier editor de textos como el bloc de notas de Windows ó Notepad++ ó cualquier otro.

c) Representación DOM: como un árbol donde los elementos de la página web están organizados jerárquicamente, con nodos superiores (nodos padre o parent) y nodos que derivan de los nodos padre (nodos hijo o child). El DOM es una representación completamente orientada objetos de la página web y puede ser modificado con un lenguaje de script como JavaScript.

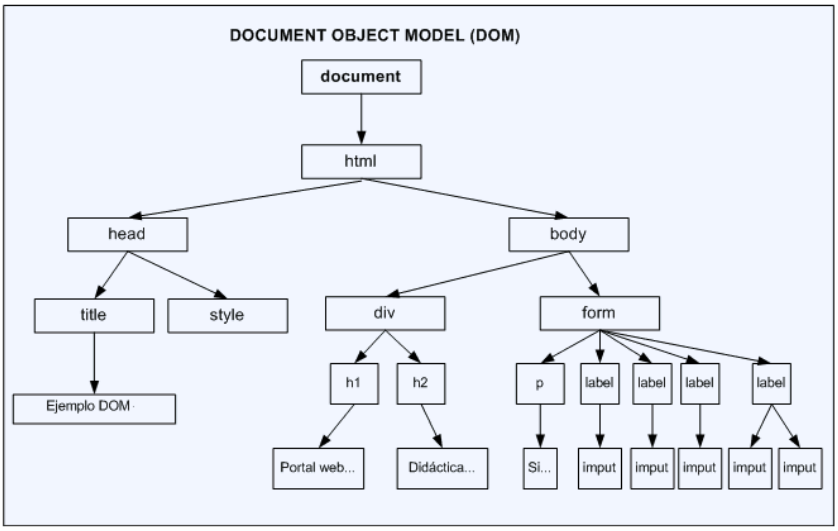
Aquí tenemos un documento en html

```
<!DOCTYPE html>
<html>
<head>
<title>Ejemplo DOM </title><meta charset="utf-8">
<style type="text/css">
body {background-color:aqua; font-family: sans-serif;}
label {color: maroon; display:block; padding:5px;}
</style>
</head>
<body>
<div id="cabecera">
<h1>Portal web </h1>
<h2>PROGRAMACION</h2>
</div>
<!-- Formulario de contacto -->
<form name ="formularioContacto" class="formularioTipo1" method="get" action="accion.html">
<p>Si quieres contactar con nosotros envíanos este formulario relleno:</p>
<label for="nombre"><span>Nombre:</span> <input id="nombre" type="text" name="nombre" /></label>
<label for="apellidos"><span>Apellidos:</span> <input id="apellidos" type="text" name="apellidos" /></label>
<label for="email"><span>Correo electrónico:</span> <input id="email" type="text" name="email" /></label>
<label>
<input type="submit" value="Enviar">
<input type="reset" value="Cancelar">
</label>
</form>
</body>
</html>
```

Si lo visualizamos en el navegador, tendrá la siguiente representación:



La representación del documento conforme al estándar del DOM sería (de forma aproximada) esta:



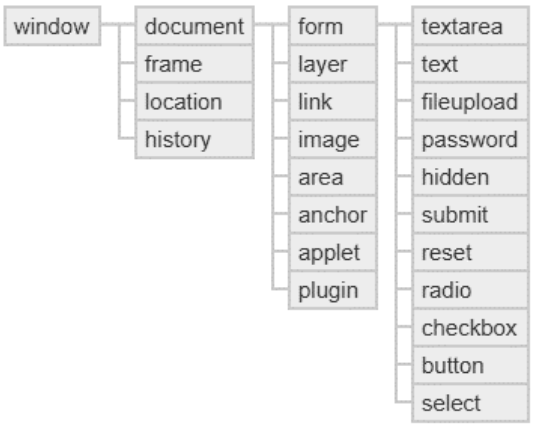
La representación anterior es solo aproximada, lo que nos interesa es conocer cómo se estructura una página web conforme al DOM para saber cómo podemos acceder a sus elementos y manipularlos usando JavaScript (u otro lenguaje). Conforme al DOM la página web se representa como un árbol de objetos o nodos, interconectados y relacionados de acuerdo con una jerarquía.

Los principales tipos de objetos o nodos en el DOM son: (Existen más tipos de nodos en el DOM, pero de uso más infrecuente.)

a) Document: el nodo document es el nodo raíz, a partir del cual derivan el resto de nodos.
b) Element: son los nodos definidos por etiquetas html. Por ejemplo, una etiqueta div genera un nodo. Si dentro de ese div tenemos tres etiquetas p, dichas etiquetas definen nodos hijos de la etiqueta div.
c) Text: el texto dentro de un nodo element se considera un nuevo nodo hijo de tipo text (texto). Los navegadores también crean nodos tipo texto sin contenido para representar elementos como saltos de línea o espacios vacíos.
d) Attribute: los atributos de las etiquetas definen nodos, aunque trabajando con JavaScript no los veremos cómo nodos, sino que lo consideramos información asociada al nodo de tipo element.
e) Comentarios y otros: los comentarios y otros elementos como las declaraciones doctype en cabecera de los documentos HTML generan nodos.

DOM y JavaScript

Cuando nuestro navegador abre una página web, su motor JavaScript lo analiza, y crea una estructura de objetos, ordenados jerárquicamente de la siguiente forma:



Todos los objetos tienen propiedades, a las que podemos acceder como nombre_objeto.nombre_propiedad. Todos los objetos tienen métodos, accesibles como nombre_objeto.nombre_método(parámetro). Algunos objetos o propiedades son arrays. Por ejemplo, el objeto image de document es un array con todas las imágenes. Para cada una, tenemos sus propiedades, como el *src* con el nombre del archivo.

Así, podemos acceder a objetos con instrucciones del tipo **document.getElementById("menu");** (acceder a un elemento por su id), pero también podremos crear objetos (por ejemplo document.createElement("h1"); ó document.body.appendChild(heading);).

Algunos de los objetos con sus propiedades y métodos más utilizados son:

window: se refiere a la ventana actual del navegador, por lo que es el más alto en la jerarquía.	
Propiedades más usadas:	
location	URL de la dirección actual
history	Array con las páginas visitadas en la ventana actual.
Métodos más usados:	
close()	Cierra la ventana
moveBy(x, y)	Mueve la ventana x pixeles en horizontal y y pixeles en vertical
moveTo(x, y)	Mueve la ventana a las coordenadas dadas.
open (URL, nombre, características)	Abre una ventana con la URL indicada, el nombre que le demos y las características por ejemplo si muestra barra de menús, barra de herramientas, alto y ancho, etc. Los parámetros son opcionales.
back(), forward()	Navega hacia atrás o hacia adelante en el historial.
document : Es el que contiene todos los elementos de nuestra página.	
Propiedades más usadas:	
bgColor	Color de fondo.
fgColor	Color del texto.
forms	Array con los formularios
images	Array con las imágenes
links	Array con los enlaces externos
title	Título de la página
Métodos más usados:	
clear()	Limpia el documento
write('texto')	Escribe el texto que le pasemos como parámetro en el documento
Writeln('texto)	Escribe el texto, añadiéndole un salto de línea al final.
getElementById()	Permite encontrar un elemento utilizando su ID.
open()	Abre la escritura del documento
close()	Cierra la escritura del documento.

EJEMPLO DE USO:

```
<HTML>
<HEAD>
      <TITLE>Prueba bgColor</TITLE>
</HEAD>
<BODY bgcolor=white>
    <p> Hola a todos </p>

<script>
   document.bgColor = "blue"
   document.write("Hola a todos en JS")
</script>
</BODY>
</HTML>
```

En HTML, así podemos definir el color de fondo del documento, y colocar el mensaje: Hola a todos

En JavaScript, utilizaríamos el objeto document, con su propiedad bgColor, para cambiar el color de fondo del documento, y la propiedad write para escribir un texto.

Cuando una página se carga, el navegador construye en memoria la jerarquía de objetos. De manera adicional, construye también estos arrays de objetos. Por ejemplo, en el caso de las imágenes, va creando el array colocando en la posición 0 la primera imagen, en la posición 1 la segunda imagen y así hasta que las introduce todas.

Vamos a ver un bucle que recorre todas las imágenes de la página e imprime su propiedad src, que contiene la URL donde está situada la imagen->

```
<script>
for (i=0; i<document.images.length; i++){
  document.write(document.images[i].src)
  document.write("<br>")
 }
</script>
```

Funciones de selección de elementos.

Los objetos DOM pueden buscarse por atributos ("id", "class",…) de HTML, utilizando las siguientes funciones:

getElementById("my_id") //devuelve el objeto DOM con el identificador buscado, o null si no lo encuentra

Devuelven una matriz de objetos:

getElementsByName("my_name")
getElementsByClassName("my_class")
querySelectorAll("CSS selector")

Ejemplo de uso:

(*) La propiedad innerHTML permite extraer/insertar HTML en el elemento del documento; proporciona una forma sencilla de cambiar completamente los contenidos de un elemento por contenido nuevo. Por ejemplo, los contenidos completos del cuerpo del documento se pueden borrar así:

```
document.body.innerHTML = "";  // Reemplaza el contenido de <body> con una cadena vacía
```