

**MP\_0373.**  
**Lenguajes de marcas y sistemas de**  
**gestión de información**

**UF2. Lenguajes para la visualización**  
**de la información**

**2.1. Modelo de objetos DOM**

# Índice

---

☰	Objetivos	3
☰	Modelo de Objetos del Documento (DOM)	4
☰	Definición del árbol	6
☰	Tipos de nodos	8
☰	Jerarquía de nodos	11
☰	Clasificación de objetos en HTML	13
☰	Resumen	14

# Objetivos

---

**En esta unidad perseguimos los siguientes objetivos:**

- 1 Conocer qué es Modelo de Objetos del Documento (DOM)
- 2 Conocer de dónde viene y para qué sirve el Modelo de Objetos del Documento.
- 3 Conocer cuáles son sus entidades.
- 4 Conocer la arquitectura del DOM.

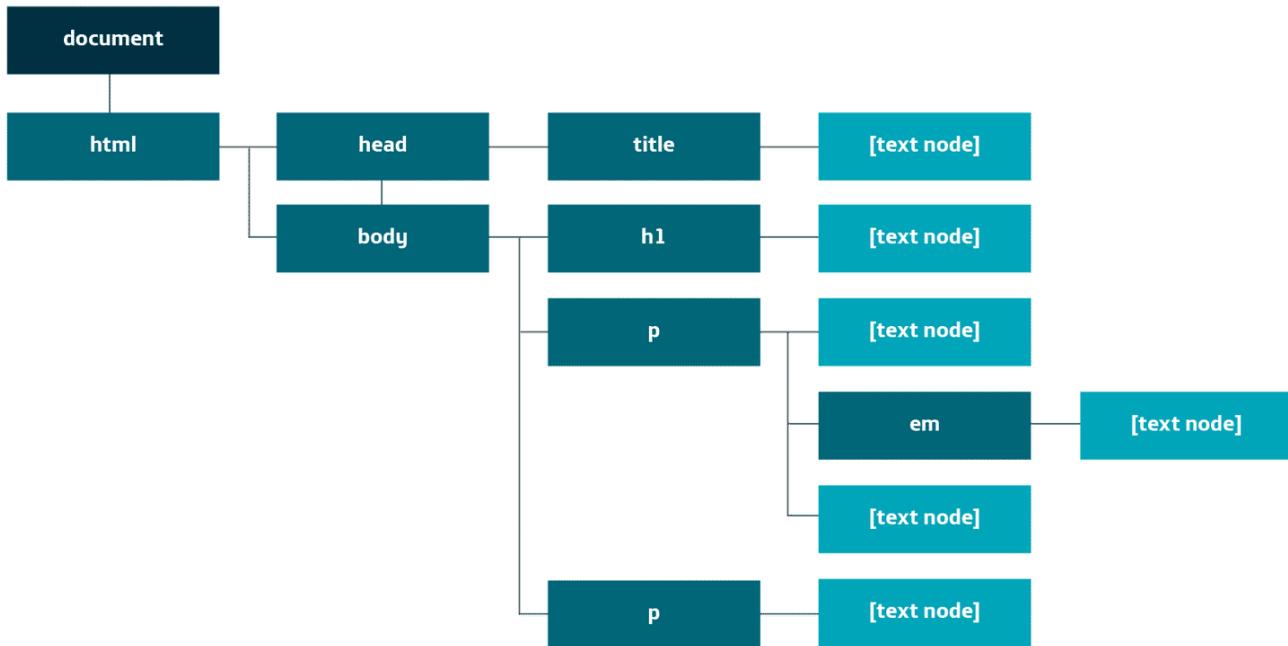
# Modelo de Objetos del Documento (DOM)

---

El Document Object Model (DOM) es el árbol creado por las aplicaciones para representar la estructura de los documentos de tipo HTML y XML.

Proporciona una representación estructurada del documento, define su estructura creando diferentes nodos de forma jerarquizada. De este modo el programador es capaz de acceder a ellos pudiendo modificarlos, añadir nuevos, etc., dando dinamismo a la visualización del documento.

Cuando creamos un documento XML o HTML cada etiqueta es un nodo o un objeto del DOM, permitiendo a otros lenguajes acceder a ellos, identificando su tipo y asignando diferentes métodos o atributos. El lenguaje más extendido para trabajar con el DOM es JavaScript.



El DOM se originó como una especificación para permitir que los programas Java y los *scripts* de JavaScript fueran portables entre los navegadores web.

El "HTML Dinámico" fue el ascendiente inmediato del Modelo de Objetos del Documento; originalmente se pensaba en él principalmente en términos de navegadores.

Cuando se formó el grupo de trabajo DOM en el W3C también se unieron a él compañías de otros ámbitos, incluyendo los de la edición y archivo de documentos HTML y XML. Varias de estas compañías habían trabajado con SGML antes de que se hubiera desarrollado el XML.

Como resultado, el DOM ha recibido influencias SGML y del estándar HyTime. Algunas de estas compañías también habían desarrollado sus propios modelos de objetos para documentos, a fin de proporcionar un API para los editores o los archivos de documentos SGML/XML. Estos modelos de objetos también han influido en el DOM.

# Definición del árbol

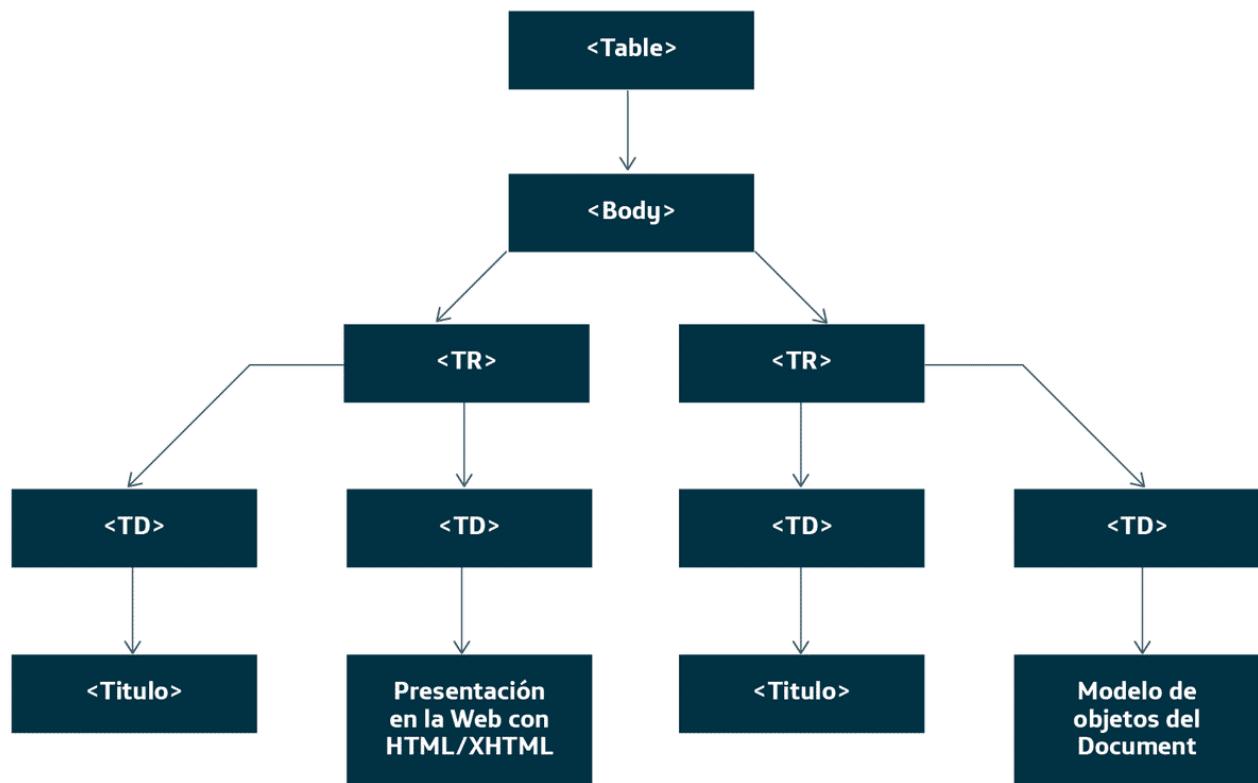
La interfaz del **Modelo de Objetos del Documento** (DOM) tiene una gran similitud con la estructura del documento al que modeliza.

Veamos un ejemplo de cómo se representaría este árbol partiendo de un código HTML.

```
<table>
  <tbody>
    <tr>
      <td>Título:</td>
      <td>Presentación en la Web con HTML/XHTML </td>
    </tr>
    <tr>
      <td>Título:</td>
      <td>El Modelo de Objetos del Documento</td>
    </tr>
  </tbody>
</table>
```

Código HTML para representar una tabla.

El resultado de la creación del DOM representado sería:



# Tipos de nodos

 Editor Edix

---

---

**Como ya hemos comentado, las aplicaciones de visualización de lenguajes de marcas crean el DOM (Document Object Model) del documento después de su lectura.**

En el caso de las páginas web son los exploradores los encargados de esta tarea, creando una representación interna del código HTML en un árbol de nodos.

Los principales tipos de nodos en el DOM son:

## Document

El objeto *document* es el nodo raíz, a partir del cual derivan el resto de nodos. Es la frontera entre el navegador y nuestro contenido.

## Element

Son los nodos creados por las propias etiquetas de HTML. Un documento web está formado por una jerarquía de nodos. Lo habitual es aplicarles términos de parentesco; si dentro de una etiqueta contenedora *<div>* ponemos dos etiquetas de párrafo *<p>*, estas *<p>* serán las hijas de la etiqueta *<div>*.

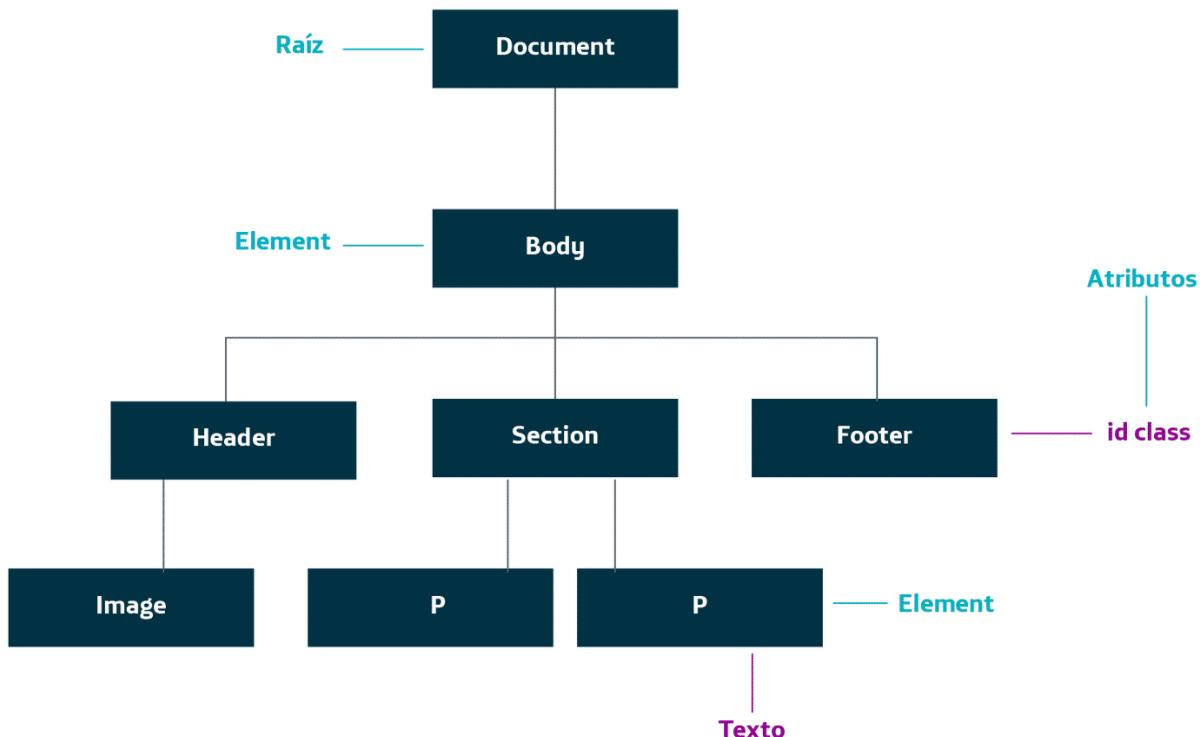
## Text

Es el contenido dentro de un nodo *element*. Este contenido, aunque no se representa dentro del árbol generado, se considera un nuevo nodo hijo de *element*.

## Comentarios

Existen otros elementos en el documento, como comentarios o el DOCTYPE entre otros, que también generan nodos. Aunque lo habitual en el desarrollo es ignorarlos, dado que rara vez interactuamos con ellos.

**Attribute:** existe otro elemento en el DOM que, aunque no genera un nodo, es de gran importancia. **Los atributos son elementos asociados a cada nodo**, imprescindibles en muchos casos para su localización o modificación.



Todos los nodos derivan del objeto base Node. A continuación, se recogen **las constantes asociadas a los diferentes tipos de nodos**, tal como están definidos por el W3C.

1. ELEMENT\_NODE: El nodo es del tipo *Element*
2. ATTRIBUTE\_NODE: El nodo es del tipo *Attr*
3. TEXT\_NODE: El nodo es del tipo *Text*
4. CDATA\_SECTION\_NODE: El nodo es del tipo *CDataSection*
5. ENTITY\_REFERENCE\_NODE: El nodo es del tipo *EntityReference*
6. ENTITY\_NODE: El nodo es del tipo *Entity*
7. PROCESSING\_INSTRUCTION\_NODE: El nodo es del tipo *ProcessingInstruction*
8. COMMENT\_NODE: El nodo es del tipo *Comment*
9. DOCUMENT\_NODE: El nodo es del tipo *Document*
10. DOCUMENT\_TYPE\_NODE: El nodo es del tipo *DocumentType*
11. DOCUMENT\_FRAGMENT\_NODE: El nodo es del tipo *DocumentFragment*

Fuente: W3C

Todos los nodos están organizados en forma de árbol jerárquico en un documento HTML, puedes ampliar la información sobre la jerarquía de los tipos del DOM en este enlace.

VER

## Jerarquía de nodos

---

**Una vez tenemos el DOM creado, debemos ser capaces de recorrerlo e identificar sus diferentes niveles de parentesco.**

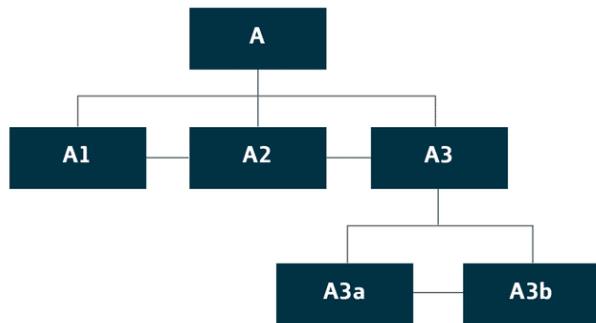
Para ello cada nodo (objeto) dispone de atributos y métodos para encontrar a sus parientes.

Uno de ellos, y el más utilizado, es el *childNodes*, que nos lista los elementos que contiene el nodo padre.

Una vez nos encontramos en un nodo podemos acceder a otros nodos aún cuando no sepamos sus identificadores.

- **childNodes**: es una matriz (*array*) que contiene todos los nodos hijos de un elemento.
- **getAttribute(id)**: devuelve el valor del atributo que identificamos por el parámetro *id*.
- **parentNode**: accede al nodo padre, o lo que es lo mismo, al nodo en el que se encuentra.
- **firstChild** y **lastChild**: contienen, respectivamente, el primer y último hijo de un nodo.
- **previousSibling** y **nextSibling**: para acceder a los hermanos del nodo en orden secuencial.
- **hasChildNodes()**: devuelve un valor booleano, indicando si el nodo tiene hijos.

Es importante entender que colocarnos en un nodo no es lo mismo que estar en su contenido. Para ir al contenido usaremos el método `nodeValue`.



<code>A.firstChild = A1</code>	<code>A1.nextSibling = A2</code>
<code>A.lastChild = A3</code>	<code>A3.prevSibling = A2</code>
<code>A.childNodes.length = 3</code>	<code>A3.nextSibling = null</code>
<code>A.childNodes[0] = A1</code>	
<code>A.childNodes[1] = A2</code>	
<code>A.lastChild.firstChild = A3a</code>	
<code>A3b.parentNode.parentNode = A</code>	

# Clasificación de objetos en HTML

Como ya hemos comentado, en la creación del árbol de nodos el navegador convierte cada etiqueta en un objeto. En esta conversión le asigna los atributos correspondientes según su tipo.

En HTML existen diferentes tipos de objetos, pero podemos hacer una clasificación de los tipos de objetos y atributos más significativos.

## Tipos de atributos

Para cada etiqueta de HTML tenemos atributos y aunque pueden ser específicos (como el *src* para un objeto *img*), tenemos algunos comunes que están en todas las etiquetas:

- Atributos básicos.
- Atributos de foco.
- Atributos de internacionalización.
- Atributos de eventos.

## Elementos

Los elementos son las propias etiquetas HTML. Se clasifican en dos grupos: **elementos en línea** o *inline* o **elementos en bloque** o *block*.

La diferencia está en el comportamiento que ejerce sobre su contenido; los elementos de bloque siempre empiezan en una nueva línea y ocupan todo el espacio disponible, mientras que los elementos en línea solo ocupan el espacio necesario para mostrar su contenido.

## Resumen

---

**Has terminado la lección, veamos los puntos más importantes que hemos tratado.**

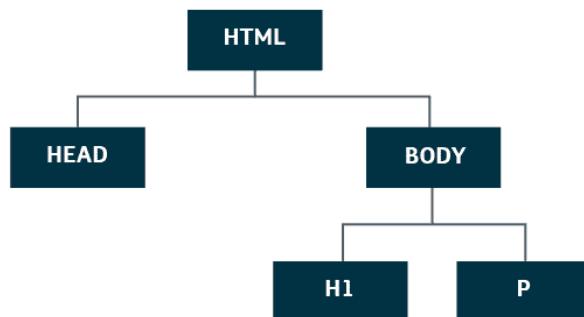
---

El Modelo de Objetos del Documento (DOM) es una interfaz de programación de aplicaciones para documentos válidos y bien construidos en HTML y XML.

Esta interfaz de programación (API) define la estructura lógica de los documentos y el modo en el que se accede y manipula un documento.

La palabra "documento" representa información que puede estar presentada en diferentes sistemas, pero el DOM permite crear documentos, navegar por su estructura, añadir, modificar y eliminar contenido.

Por lo tanto, esta especificación del W3C persigue que el Modelo de Objetos del Documento proporcione una interfaz estándar que pueda ser utilizada en una variedad amplia de entornos y aplicaciones.





**PROEDUCA**