

## ■ Tema 3: JavaScript

### ■ Parte II: Modelo de Objetos del documento (DOM), nivel I.



Tecnológicas  Web

## Contenidos

- El DOM y el árbol del documento
- Acceso a los nodos. Obtener información del documento
- Cambiar el contenido y la estructura del documento
- DOM 1 HTML



Tecnológicas  Web

## 1.

### El DOM y el árbol del documento



### DOM (Document Object Model)

- API orientado a objetos que permite interactuar con el documento HTML
  - Cambiar/leer contenido y estructura
  - Cambiar/leer estilos CSS
- Niveles
  - 0: impuesto por Netscape y Microsoft en la generación anterior de navegadores (Explorer 4, Netscape 4)
  - 1 (W3C): contenido dinámico
  - 2 (W3C): estilos dinámicos, eventos



# DOM 1

- API que permite acceso/cambio a contenido del documento, por ejemplo, se puede
  - Insertar nuevas etiquetas en el documento (p.ej. crear un botón nuevo o una fila nueva en una tabla)
  - Leer/cambiar el contenido de cualquier etiqueta (p.ej. de un párrafo <p>)
  - Reordenar los componentes del documento (p.ej. reordenar las filas de una tabla)
- Está dividido en módulos
  - **Núcleo**: sirve para cualquier lenguaje de marcado (XML, HTML, ...)
  - **HTML**: objetos, propiedades y métodos que facilitan el trabajo con HTML
- Cada navegador puede implementar unos módulos y no otros



## Comprobar compatibilidad

- Objeto navigator

```
if (navigator.appName=="Netscape") {  
    //estamos en Netscape (o Firefox)
```

- En general, es más práctico comprobar si el método o el objeto existe viendo si es undefined (==false)

```
//getElementById es un método, aquí no lo estamos llamando,  
//ya que no aparecen los ( ), simplemente comprobando si existe  
if (document.getElementById) {  
    //aquí sí lo llamamos  
    t = document.getElementById("tabla")  
}
```



# Documentos en DOM

■ En DOM los documentos no se tienen como “texto plano”

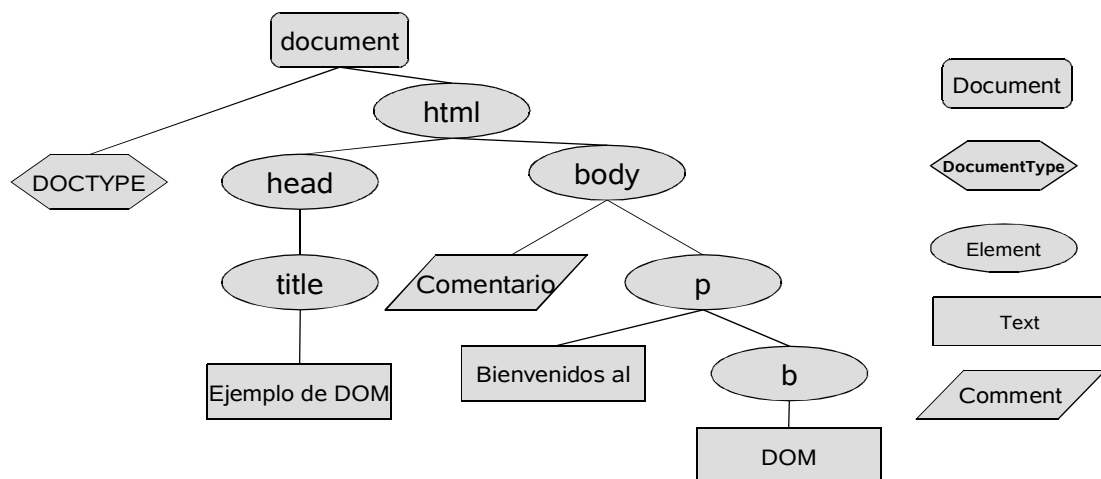
```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
  <title>Ejemplo de DOM</title>
</head>
<body>
  <!-- es un ejemplo un poco simple -->
  <p style="color:red">Bienvenidos al <b>DOM</b></p>
</body>
</html>
```



Tecnología Web 

## El árbol del documento

■ Sus nodos reflejan el contenido y la estructura del documento



■ **Nota:** Mozilla crea nodos de texto adicionales donde haya espacios en blanco, retornos de carro, etc. entre etiquetas. Explorer no



Tecnología Web 

## Los nodos del árbol

- Todos los nodos son del “tipo” **Node**, pero hay distintos “subtipos”: **Document**, **DocumentType**, **Element**, **Text**, **Comment**, ...
- Aunque los atributos de las etiquetas son nodos de tipo **Attr**, “no están” en el árbol, hay que acceder a ellos con métodos del nodo que los posee.



## ■ 2.

- Acceso a los nodos. Obtener información del documento



## Cómo acceder a los nodos: acceso directo

- Hay dos formas de acceso: directo al nodo o descendiendo desde la raíz (u otro nodo).
- Las dos dependen inicialmente del objeto predefinido **document** (que ya estaba en DOM 0)
- Acceso directo a un nodo en concreto
  1. El nodo debe tener un nombre, dado en HTML con el atributo **id**
  2. El método **document.getElementById(id)** devuelve el nodo buscado

Fragmento de HTML:

```
<p id="parrafo"> Esto es un párrafo </p>
```

Javascript en la misma página:

```
var par = document.getElementById("parrafo") //par es el objeto nodo <p>
```



Tecnologías  Web

## Obtener información de un nodo

- Una vez obtenida la referencia a un nodo podemos obtener sus propiedades.
- Algunas props. de **Node** (cualquier nodo)
  - **nodeType**: cte. entera que representa el tipo de nodo
  - **nodeName**: nombre, **nodeValue**: valor. Dependen del tipo de nodo

Tipo de nodo	nodeType	nodeName	nodeValue
Etiqueta	1 (Node.ELEMENT_NODE)	Nombre de la etiqueta sin los "<>" y en máyúsc.	null
Texto	3 (Node.TEXT_NODE)	#text	Texto del nodo
Comentario	8 (Node.COMMENT_NODE)	#comment	Texto del comentario
DOCTYPE	10(Node.DOCUMENT_TYPE_NODE)	Nombre de la etiq. raíz del DOCTYPE	null
Documento	9 (Node.DOCUMENT_NODE)	#document	null

**NOTA:** en Explorer, las ctes. Node.XXX\_NODE no están predefinidas, hay que utilizar el valor numérico



Tecnologías  Web

## Obtener información de un nodo (II)

### Algunas props. del tipo **Element** (nodo etiqueta)

- **tagName**: nombre de la etiqueta, sin los <> y en mayúsculas (idem a nodeName)
- **getAttribute(nombre)**: valor de un atributo

```
<p align="right" id="par">Párrafo alineado a la derecha</p>
```

```
<script language="JavaScript">
```

```
    alert(document.getElementById("par").getAttribute("align")); // "right"
```

```
</script>
```

### Algunas props. del tipo **Document** (nodo que representa al documento entero)

- **documentElement**: nodo con la etiqueta raíz

```
alert(document.documentElement.nodeName); // en HTML, "HTML"
```

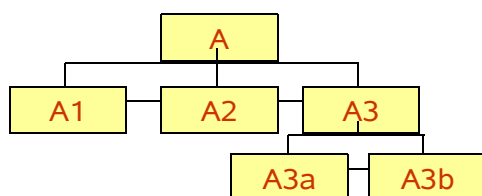


Tecnologías  Web

## Cómo acceder a un nodo desde otro

### Cada nodo tiene una serie de propiedades que reflejan el “parentesco” con otros, algunas de las cuales son

- **childNodes**: array con los nodos hijos
- **firstChild**: primer nodo hijo, **lastChild**: último nodo hijo
- **parentNode**: nodo padre
- **nextSibling**: siguiente hermano (nodo al mismo nivel) **prevSibling**: hermano anterior.



A.firstChild = A1

A1.nextSibling = A2

A.lastChild = A3

A3.prevSibling = A2

A.childNodes.length = 3

A3.nextSibling = null

A.childNodes[0] = A1

A.childNodes[1] = A2

A.lastChild.firstChild = A3a

A3b.parentNode.parentNode = A



Tecnologías  Web

## Cómo acceder a un nodo desde otro (II)

- Esta forma de acceso es problemática por
  - Dependencia de la estructura del árbol. Si cambia, acabaremos en otro nodo o generaremos un error
  - Incompatibilidades entre navegadores: como se ha visto, Mozilla interpreta los espacios en blanco entre etiquetas como nodos de texto
- No obstante, es necesaria
  - Para recorrer de manera sistemática todo el árbol
  - Para acceder a ciertos nodos. Por ejemplo, los nodos de texto no son accesibles con el método “directo”.



## Acceso directo a todos los nodos del mismo tipo

- Mediante el método `document.getElementsByTagName(nombreEtiqueta)`, obtenemos todas las etiquetas del mismo tipo.

```
//Cambia el color de todos los párrafos a rojo
var nodeList = document.getElementsByTagName("P");
for (var i = 0; i < nodeList.length; i++)
    //la propiedad style representa el estilo CSS, con subpropiedades
    //que son nombres de propiedades CSS
    nodeList[i].style.color = "red";
```

- Si `nombreEtiqueta="**"`, entonces accedemos a todas las etiquetas HTML del documento
- Si lo llama una etiqueta, obtenemos solo sus subetiquetas

```
var tabla = document.getElementById("tabla1");
var filasDeTabla1 = tabla.getElementsByTagName("tr");
```



### 3.

## Cambiar el contenido y la estructura del documento



## Cambiar los datos de un nodo

- Muchas propiedades son solo de lectura (**nodeName**, **firstChild**, **parentNode**,...) para cambiarlas hay que hacerlo de modo indirecto recurriendo a otros métodos.
- Cambiar el valor: cambiar la propiedad **nodeValue**
- Cambiar un atributo: **setAttribute(nombre,nuevoValor)**

```
<p id="p" align="left">Estoy alineado a la izquierda</p>
<script language="JavaScript">
  var p = document.getElementById("p");
  p.setAttribute("align","right");
  p.firstChild.nodeValue="ahora estoy alineado a la derecha";
</script>
```



## Crear nuevos nodos

- Distintos métodos del objeto predefinido **document**, según el tipo de nodo a crear
  - **document.createElement(nombre)**: crea nodo etiqueta. Se le pasa el nombre de la etiqueta sin <>.
  - **document.createTextNode(texto)**: crea nodo de texto, con el contenido especificado

- Hace falta **insertar los nodos creados** en el árbol

```
<body id="cuerpo">
<script language="JavaScript">
  var par = document.createElement("p");
  var texto = document.createTextNode("Yo antes no estaba aquí")
  par.appendChild(texto);
  document.getElementById("cuerpo").appendChild(par);
</script>
</body>
```



## Insertar/eliminar nodos

- Métodos de la clase **Node**, los llama el que va a ser “padre” del nodo a insertar / el padre del que se va a eliminar

- Insertar nodos

- **appendChild(nuevoHijo)**: Añade el hijo al final de todos los hijos actuales
- **insertBefore(nuevoHijo, hijoReferencia)**. Inserta el nuevo hijo justo antes del “hijo de referencia”
- **setAttribute(nuevoAtributo, nuevoValor)**. Si el atributo no existía, lo crea. Como ya se ha visto, si existía cambia su valor

- Eliminar nodos

- **removeChild(hijoABorrar)**: un nodo deja de ser hijo
- **replaceChild(nuevoHijo, hijoAntiguo)**: reemplaza un hijo por otro nuevo



## 4.

### DOM 1 HTML



### DOM 1 HTML

- Facilita el trabajo con documentos HTML, haciendo más directas algunas operaciones
- Proporciona compatibilidad con el DOM 0, predefiniendo objetos como los arrays **forms**, **images**, etc. o métodos como **document.write()**.

```
//Reducir el tamaño de todas las imágenes a la mitad, con DOM núcleo
imgs = document.getElementsByTagName("img");
for(var i=0; i<imgs.length; i++) {
    imgs[i].setAttribute("width",imgs[i].getAttribute("width")/2);
    imgs[i].setAttribute("height",imgs[i].getAttribute("height")/2);
}

//idem con DOM HTML
for(var i=0; i<document.images.length; i++) {
    document.images[i].width /= 2;
    document.images[i].height /= 2;
}
```



## Acceso a los atributos HTML

- Se dispone de propiedades predefinidas que representan los atributos HTML. El nombre de la propiedad es el del atributo HTML

```

<script language="JavaScript">
  var logo = document.getElementById("logo");
  alert(logo.getAttribute("width")); //DOM 1 núcleo
  alert(logo.width);               //DOM 1 HTML
  logo.setAttribute("tipo","jpg"); //esto solo se puede hacer con
                                     //núcleo, ya que no es un atributo
                                     //HTML
</script>
```

- Excepciones: cuando un atributo HTML es palabra reservada en JavaScript:
  - El nombre de la propiedad es el del atributo precedido de "html" (Ej: el atributo **for** es la propiedad **htmlFor**)
  - Excepción a la excepción: el atributo HTML **class** (palabra reservada en JavaScript) es la propiedad **className**



## Acceso a etiquetas por nombre

- El atributo **name** de HTML no tiene por qué ser único en un documento, se puede nombrar un grupo de etiquetas con el mismo **name** (típico en botones de radio o en *checkboxes*)
- document.getElementsByName(name)** obtiene un array con todas las etiquetas con un mismo **name**



## Trabajo con tablas: algunos métodos

- **rows**: propiedad de un nodo tabla que contiene todas sus filas
- **cells**: propiedad de un nodo fila que contiene todas sus celdas
- Insertar y borrar filas: los llama un nodo tabla
  - **insertRow(*pos*)**: insertar nueva fila vacía (nodo **tr**) en la posición *pos*. Comienzan por 0.
  - **deleteRow(*pos*)**: borrar la fila nº *pos*
- Insertar y borrar celdas: los llama un nodo fila
  - **insertCell(*pos*)**: insertar nueva celda vacía (nodo **td**) en la posición *pos*. Comienzan por 0
  - **deleteCell(*pos*)**: borrar la celda nº *pos*.

