Tema 6. Modelo de Objetos de Documento (DOM) (I)

- DOM
- Árbol de nodos
- Tipos de nodos
- Nodos y etiquetas
- El Objeto *Node*
- Nodos padre, hijo y hermanos
- DOM básico
- La interfaz *Document*
- La interfaz *Element*
- Objetos colección
- La interfaz *NamedNodeMap*

Tema 6. Modelo de Objetos de Documento (DOM) (II)

- DOM HTML
- Interfaz *HTMLDocument*
- *links*
- *images*
- *forms*
- Acceso al formulario y a sus campos
- Propiedades comunes
- La interfaz *HTMLInputElement*
- El control *input*
- La interfaz *HTMLSelectElement*
- El control *select*
- Tablas
- Interfaz *TableElement*
- Interfaz *HTMLTableColElement*
- Interfaz *HTMLTableRowElement*
- Interfaz *HTMLTableCellElement*

DOM

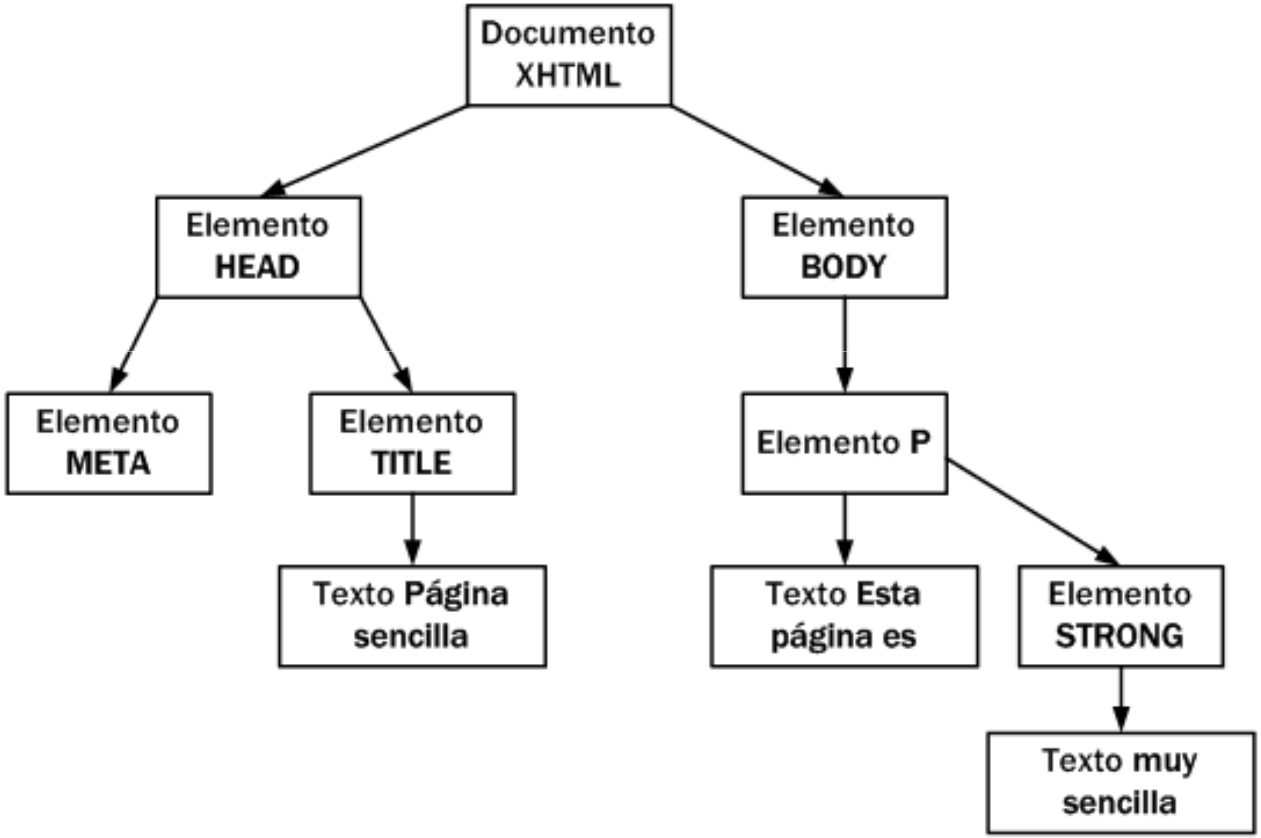
- DOM (*Document Object Model)* es un conjunto de utilidades diseñadas para manipular documentos XML.
- Por extensión, DOM también se puede utilizar para manipular documentos XHTML y HTML.
- Técnicamente, DOM es una API (Interfaz de Programación de Aplicaciones) que proporciona un conjunto estándar de objetos para representar documentos HTML y XML, un modelo estándar sobre cómo pueden combinarse dichos objetos, y una interfaz estándar para acceder a ellos y manipularlos.
- A través del DOM, los programas pueden acceder y modificar el contenido, estructura y estilo de los documentos HTML y XML.
- En primer lugar, DOM transforma internamente el archivo XML original en una estructura formada por una jerarquía de nodos en forma de árbol.
- Mediante los nodos del árbol, se representan los contenidos de la página Web y las relaciones entre ellos.
- DOM es una API independiente de cualquier lenguaje de programación. De hecho, existen interfaces DOM para distintos lenguajes de programación: Java, C++, PHP, etc.

Árbol de nodos (I)

- La conversión de etiquetas en nodos se realiza de forma jerárquica.
- Cada etiqueta se transforma en un nodo.
- Aquellas etiquetas que tienen texto, contienen como nodos hijos que además son hojas, el texto sobre el que se aplican.

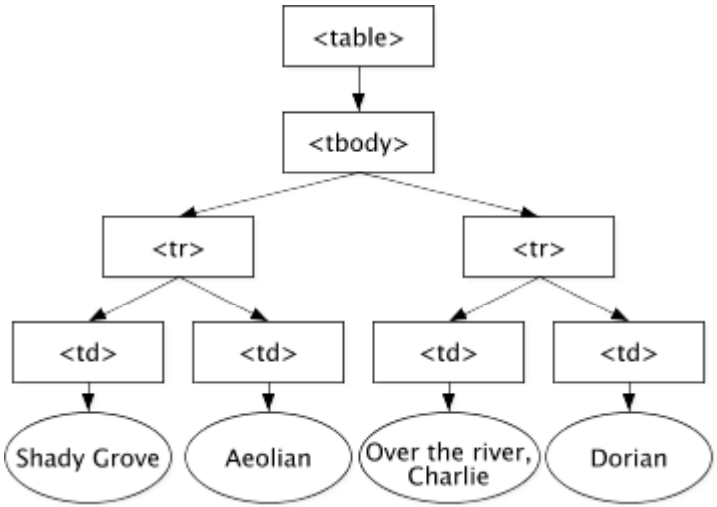
```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
   charset=iso-8859-1" />
<title>Página sencilla</title>
</head>
<body>
<p>Esta página es <strong>muy sencilla</strong></p>
</body>
</html>
```

Árbol de nodos (II)



Árbol de nodos (III)

```
<table>
<tbody>
 <tr>
   <td>Shady Grove</td>
   <td>Aeolian</td>
</tr>
<tr>
  <td>Over the river, Charlie</td>
  <td>Dorian</td>
</tr>
</tbody>
</table>
```



Tipos de nodos (I)

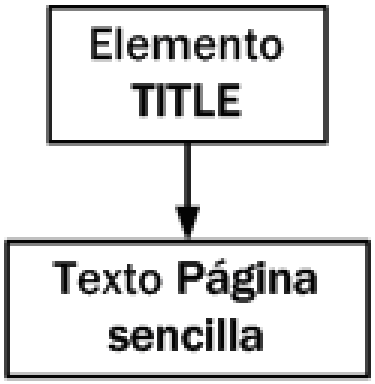
- La especificación completa de DOM define 12 tipos de nodos, aunque las páginas HTML habituales se pueden manipular manejando solamente cuatro o cinco tipos de nodos:
  - *Document* → Nodo raíz del que derivan todos los demás nodos del árbol.
  - *Element* → Representa cada una de las etiquetas HTML. 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 XHTML, es decir, uno por cada par *atributo=valor*.
  - *Text* → Nodo que contiene el texto encerrado por una etiqueta HTML.
  - *Comment* → Representa los comentarios incluidos en la página HTML.
- Los otros tipos de nodos existentes son
  - *ProcessingInstruction*, *DocumentType*, *CDataSection*, *DocumentFragment*, *Entity*, *Notation* y *EntityReference*.

Tipos de nodos (II)

Node	Namenode	Value	nodeType
Document	#document	null	9
Element	Nombre de etiqueta (tagName)	null	1
Attr	Nombre del atributo	Valor del atributo	2
Text	#text	Contenido del nodo de texto	3
Comment	#comment	Contenido del comentario	8

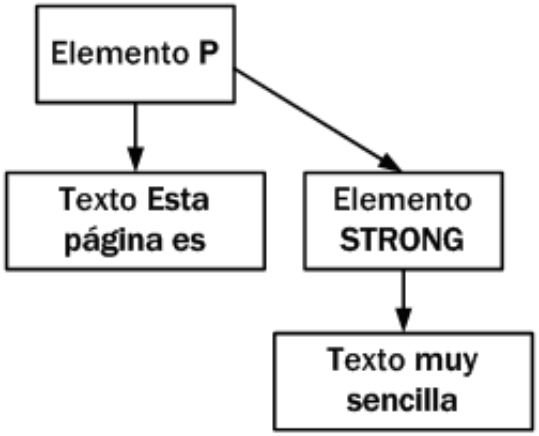
Nodos y etiquetas (I)

- La transformación de las etiquetas XHTML habituales genera dos nodos:
  - el primero es el nodo de tipo *Element* correspondiente a la propia etiqueta XHTML
  - el segundo es un nodo de tipo *Text* que contiene el texto encerrado por esa etiqueta XHTML.
- *<title>Página sencilla</title> g*enera los siguientes dos nodos:



Nodos y etiquetas (II)

- * Esta página es muy sencilla * genera los siguientes nodos:
  - Nodo de tipo *Element* correspondiente a la etiqueta * .*
  - Nodo de tipo *Text* con el contenido textual de la etiqueta * *.
  - Como el contenido de * * incluye en su interior otra etiqueta XHTML, la etiqueta interior se transforma en un nodo de tipo *Element* que representa la etiqueta ** y que deriva del nodo anterior
  - El contenido de la etiqueta ** genera a su vez otro nodo de tipo *Text que* deriva del nodo generado por **.



El Objeto *Node*

- DOM proporciona el objeto *Node* para definir las propiedades y métodos necesarios para procesar y manipular los elementos que forma parte de un documento.
- En primer lugar, el objeto *Node* define las siguientes constantes para la identificación de los distintos tipos de nodos:
  - Node.ELEMENT_NODE = 1
  - Node.ATTRIBUTE_NODE = 2
  - Node.TEXT_NODE = 3
  - Node.CDATA_SECTION_NODE = 4
  - Node.ENTITY_REFERENCE_NODE = 5
  - Node.ENTITY_NODE = 6
  - Node.PROCESSING_INSTRUCTION_NODE = 7
  - Node.COMMENT_NODE = 8
  - Node.DOCUMENT_NODE = 9
  - Node.DOCUMENT_TYPE_NODE = 10
  - Node.DOCUMENT_FRAGMENT_NODE = 11

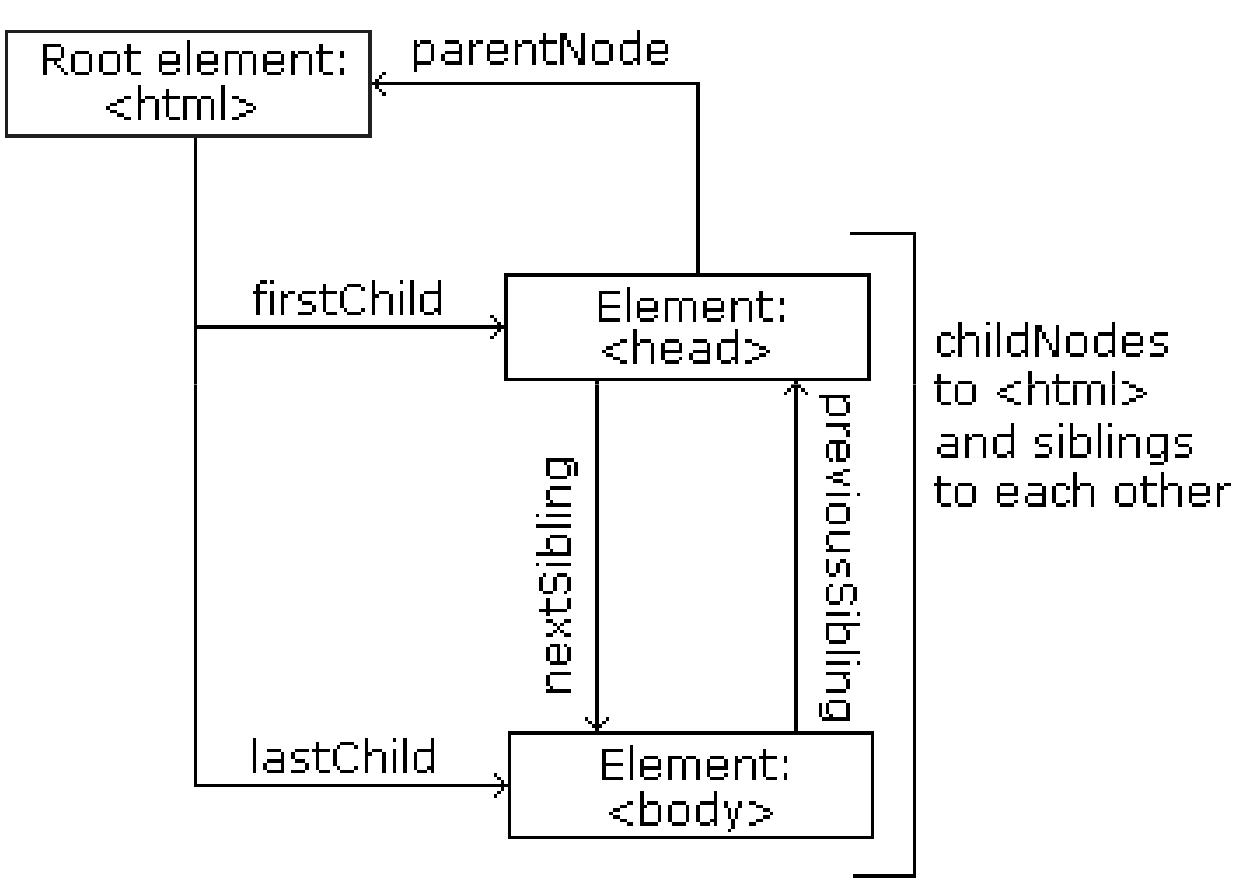
Atributos de *Node* (I)

Atributo	Tipo	Descripción
nodeName	*String*	El nombre de este nodo, dependiendo de su tipo (ver la tabla precedente)
nodeValue	*String*	El valor de este nodo, dependiendo de su tipo (ver la tabla precedente)
nodeType	*Number*	Una de las 12 constantes definidas anteriormente
parentNode	*Node*	Referencia al padre del nodo
firstChild	*Node*	Referencia del primer nodo de la lista *childNodes*
lastChild	*Node*	Referencia del último nodo de la lista *childNodes*

Atributos de *Node* (II)

Atributo	Tipo	Descripción
childNodes	*NodeList*	Lista de todos los nodos hijo del nodo actual
previousSibling	*Node*	Referencia del nodo hermano anterior o null si este nodo es el primer hermano
nextSibling	*Node*	Referencia del nodo hermano siguiente o null si este nodo es el último hermano
attributes	*NamedNodeMap*	Se emplea con nodos de tipo *Element*. Contiene objetos de tipo *Attr* que definen todos los atributos del elemento
ownerDocument	*Document*	Referencia del documento al que pertenece el nodo

Relación entre nodos



Métodos de *Node*

Método	Tipo devuelto	Descripción
hasChildNodes()	*Boolean*	Devuelve *true* si el nodo actual tiene uno o más nodos hijo
appendChild(nodo)	*Node*	Añade un nuevo nodo al final de la lista *childNodes*
removeChild(nodo)	*Node*	Elimina un nodo de la lista *childNodes*
replaceChild(new,ant)	*Node*	Reemplaza el nodo *ant* por el nodo *new*
insertBefore(new,ant)	*Node*	Inserta el nodo *new* antes de la posición del nodo *ant*, dentro de la lista *childNodes*

Nodos padre, hijo y hermanos (I)

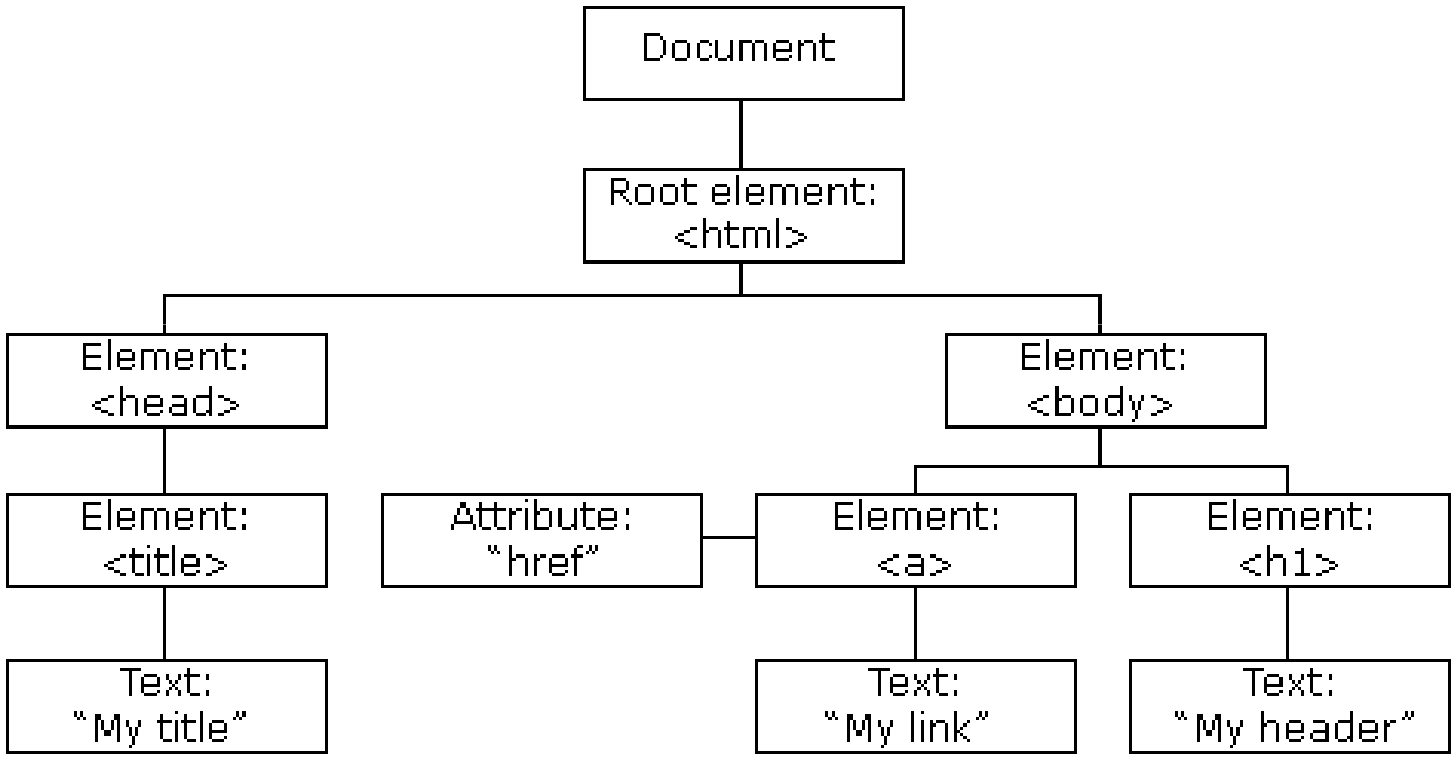
- HTML DOM visualiza un documento HTML como una estructura en forma de árbol (árbol de nodos).
- A través del árbol se puede acceder a todos los nodos. Además, se puede modificar o eliminar el contenido de cada nodo e incluso crear nuevos elementos.
- Los nodos del árbol poseen una relación jerárquica con los demás nodos.
- Los términos padre (*parent*), hijo (*child*), y hermano (*sibling*) se usan para describir estas relaciones.
  - Los nodos padre tienen hijos
  - Los hijos en el mismo nivel se llaman hermanos
- En un árbol, el nodo superior (el primero) se llama nodo raíz (*root*)
- Cada nodo, excepto la raíz, tiene exactamente un padre (definición de la estructura árbol).
- Un nodo hoja (*leaf*) es aquel que no tiene hijos.
- Los nodos hermanos (*siblings*) son aquellos que tienen el mismo padre.

Nodos padre, hijo y hermanos (II)

```
<html>
  <head>
   <title>DOM Tutorial</title>
  </head>
  <body>
   <h1>DOM Lesson one</h1>
   <p>Hello world!</p>
   </body>
</html>
```

- El nodo <*html*> es el nodo raíz
- El nodo padre de <*head*> y <*body*> es el nodo <*html*>
- El nodo padre del nodo *text* "Hello world!" es el nodo <*p*>
- El nodo <*html*> tiene dos hijos: <*head*> y <*body*>
- El nodo <*head*> tiene un hijo: el nodo <*title*>
- El nodo <*title*> también tiene un nodo hijo: el nodo *text* "DOM tutorial"
- Los nodos <*h1*> y <*p*> son hermanos, ambos son hijos del nodo <*body*>

Nodos padre, hijo y hermanos (III)



DOM básico

- Cualquier lenguaje basado en XML, como XHTML puede utilizar el DOM básico visto hasta ahora.
- Además, muchos lenguajes definen sus propios DOM, ampliando la funcionalidad básica.
- En un documento HTML, según el DOM básico todo son nodos:
  - El documento entero es un nodo *Document*
  - Cualquier elemento es un nodo *Element*
  - El texto de los elementos HTML son nodos *Text*
  - Todo atributo HTML es un nodo *Attr* (Attribute)
- W3C desarrolló un DOM más especifico para HTML, de modo que HTML DOM define los objetos y propiedades de todos los elementos HTML y los métodos para acceder a ellos.
- Con HTML DOM podemos acceder, cambiar, añadir o eliminar los elementos HTML

La interfaz *Document*

Método	Descripción
createAttribute(atr)	Crea un nodo de tipo atributo con el nombre indicado
createElement(tag)	Crea un nodo *Element* con la etiqueta *tag*
createTextNode(text)	Crea un nodo de tipo texto con el texto *text*
getElementsByTagName(tag)	Obtiene todos los elementos del documento cuya etiqueta sea igual a *tag*
getElementsByName(nom)	Obtiene todos los elementos del documento cuyo atributo *name* coincida con *nom*.
getElementById(id)	Devuelve el elemento cuyo atributo *id* coincide con el parámetro. Como el atributo *id* es único, la función devuelve únicamente el nodo deseado.

Atributo	Descripción
documentElement	Es el elemento raíz del documento. En HTML es el elemento con la etiqueta <*html*>

Ejemplo 1

```
<html>
<head><title>Exemple 1</title></head>
<body>
<h1>Aplicando el DOM</h1>
<p id="p1">El párrafo uno</p>
<p align="center">El párrafo dos</p>
<script type="text/javascript">
var Ohtml=document.documentElement;
alert(Ohtml.nodeName);
var hijos=Ohtml.childNodes;
for(i=0;i<hijos.length;i++) alert(hijos[i].nodeName);
var Obody=Ohtml.firstChild.nextSibling; //var Obody=hijos[1];
alert(Obody.nodeName);
//alert(Obody.firstChild.nextSibling.firstChild.nodeValue);
//En mozilla i safari, el primer node fill de body no es <h1> sino #text.
alert(document.getElementById("p1").firstChild.nodeValue);
</script></body>
</html>
```

La interfaz *Element*

Método	Descripción
getAttribute(nom)	Devuelve el valor del atributo *nom*.
setAttribute(nom, valor)	Añade o modifica el atributo *nom* con el valor *elem*
removeAttribute(nom)	Elimina el atributo *nom*
hasAttribute(nom);	Verdadero si existe el atributo, falso en caso contrario

```
<html>
<head><title>Exemple 1</title></head>
<body>
<h1>Aplicando el DOM</h1>
<p id="p1">El párrafo uno</p>
<p align="center">El párrafo dos</p>
<script type="text/javascript">
var Op2=document.getElementsByTagName("p")[1];
if(Op2.hasAttribute("align"))
  alert("Tiene align="+Op2.getAttribute("align"));
Op2.setAttribute("align","right");
</script>
```

Objetos colección

- Además de los nodos, DOM define objetos de ayuda que se utilizan para trabajar con nodos, pero que no formar parte de un documento DOM
  - *NodeList*: Vector de nodos indexado numéricamente, que se utiliza para representar los nodos hijo de un elemento
  - *NamedNodeMap*: Vector de nodos indexado numéricamente y por nombre que se usa para representar atributos de elementos

La interfaz *NamedNodeMap*

Método	Descripción
getNamedItem(nom)	Devuelve el nodo cuya propiedad *nodeName* contenga el valor *nom*.
removeNamedItem(nom)	Elimina el nodo cuya propiedad *nodeName* coincida con el valor nombre.
setNamedItem(nodo)	Añade el nodo a la lista *attributes*, indexándolo según su propiedad *nodeName*.
item(pos)	Devuelve el nodo que se encuentra en la posición indicada por el entero *pos*.

- Equivalencias entre los métodos de *NamedNodeMap* y de *Element*
  - *getAttribute(nom) ←→ attributes.getNamedItem(nom).*
  - *setAttribute(nom, valor) ←→ attributes.getNamedItem(nom).value= valor*
  - *removeAttribute(nom) ←→ attributes.removeNamedItem(nom)*

Ejemplo 2 (I)

```
<html>
<head>
<title>Exemple 2</title>
</head>
<body>
<h1>Aplicando el DOM</h1>
<p id="p1">El párrafo uno</p>
<p align="center">El párrafo dos</p>
<script type="text/javascript">
var llistaP=document.getElementsByTagName("p");
alert("Eliminando atributo align");
llistaP[1].attributes.removeNamedItem("align");
alert("Añado color rojo al primer párrafo");
var nAtr=document.createAttribute("style");
nAtr.nodeValue="color:red";
llistaP[0].attributes.setNamedItem(nAtr);
```

Ejemplo 2 (II)

```
...
alert("Centramos");
var nodoStyle=llistaP[0].attributes.getNamedItem("style");
nodoStyle.nodeValue+=";text-align:center";
alert("Lo rojo es azul?");
nAtr.nodeValue="text-align:center;color:blue";
alert("Un nuevo párrafo");
var nElement=document.createElement("p");
var nText=document.createTextNode("Mi nuevo párrafo al final");
nElement.appendChild(nText);
document.body.appendChild(nElement);
alert("Insertar antes de \"parráfo dos\"");
document.body.insertBefore(nElement,llistaP[1]);
```

Ejemplo 2 (III)

```
...
alert("Otro nodo después del nuevo");
var nElement1=document.createElement("p");
var nText1=document.createTextNode("Otro párrafo después de ...");
nElement1.appendChild(nText1);
nElement.parentNode.insertBefore(nElement1,nElement.nextSibling);
var otro=document.createElement("p");
var nText2=document.createTextNode("Reemplaza a Mi nuevo ...");
otro.appendChild(nText2);
alert("Reemplaza");
nElement.parentNode.replaceChild(otro,nElement);
</script>
</body>
</html>
```

DOM HTML

- Las propiedades y métodos del DOM básico son genéricas, diseñadas para funcionar con cualquier documento DOM.
- Las propiedades y métodos del DOM HTML son especificas para HTML y facilitan determinadas operaciones como el acceso a atributos y métodos propios de los elementos de HTML.
- Además, DOM HTML proporciona una conjunto de nodos adicional al DOM básico donde el tipo de nodo coincide con los elementos HTML
  - *HTMLDocument, HTMLHtmlElement, HTMLHeadElement, HTMLTitleElement, HTMLBodyElement, HTMLFormElement, HTMLSelectElement, HTMLOptionElement, HTMLTextAreaElement, HTMLButtonElement, HTMLImageElement,HTMLTableElement, HTMLTableColElement, HTMLTableRowElement, HTMLFrameSetElement, HTMLFrameElement,* etc.
- A su vez, estos nodos propios de HTML heredan de los nodos básicos de DOM.
- Más información en *http://www.w3.org/TR/DOM-Level-2-HTML*

Interfaz *HTMLDocument*

```
interface HTMLDocument : Document {
           attribute      DOMString      title;
  readonly attribute      DOMString      referrer;
  readonly attribute      DOMString      domain;
  readonly attribute      DOMString      URL;
           attribute      HTMLElement    body;
  readonly attribute      HTMLCollection images;
  readonly attribute      HTMLCollection applets;
  readonly attribute      HTMLCollection links;
  readonly attribute      HTMLCollection forms;
  readonly attribute      HTMLCollection anchors;
           attribute      DOMString      cookie;
  void     open();
  void     close();
  void     write(in DOMString text);
  void     writeln(in DOMString text);
  NodeList getElementsByName(in DOMString elementName);
};
```

links

- En el vector *links* se encuentran todos los enlaces del documento.
- Cada elemento de *links* engloba todas las propiedades que tienen los enlaces del documento actual.

Atributo	Descripción
target	Nombre de la ventana especificado en la etiqueta target
hash	Nombre del ancla dentro de la URL
host	Nombre del servidor y número de puerto, dentro de la URL
hostname	Nombre de dominio del servidor (o la dirección IP) dentro de la URL
href	Contiene la URL completa
pathname	Camino al recurso, dentro de la URL
port	Número de puerto, dentro de la URL
protocol	Protocolo usado, dentro de la URL

Ejemplo 3

```
<html>
<body>
<a href="http://www.yahoo.com" target="_blank">Yahoo!!</a><br><br>
<a href="http://www3.uji.es/~murgui/hcweb">La página</a><br><br>
<script type="text/javascript">
for(var i=0;i<document.links.length;i++){
   document.write("<p>Target : " + document.links[i].target + "<br>");
   document.write("Host : " + document.links[i].host + "<br>");
   document.write("Href : " + document.links[i].href + "<br>");
   document.write("Pathname : " + document.links[i].pathname +
   "<br>");
   document.write("</p>");
}
</script>
</body>
</html>
```

images

- Es una vector que contiene todas las imágenes del documento.
- Cada elemento del vector es un objeto *Image,* el cual posee los atributos de la tabla siguiente.
- *JavaScript* proporciona constructor para este objeto

Atributo	Descripción
border	Borde de la imagen
complete	Valor booleano que indica si la imagen se ha descargado completamente
height	Altura en píxeles de la imagen
hspace	Espacio horizontal desde el borde izquierdo de la ventana
name	Contiene el valor del atributo *name* de la imagen
src	Contiene el valor del parámetro *src* de la imagen
vspace	Espacio vertical desde el borde superior de la ventana
width	Anchura de la imagen

Ejemplo 4 (I)

```
<html>
<head>
<title>Exemple 4</title>
</head>
<body>
<h2>DOM básico y DOM HTML</h2>
<p>Probando con imágenes</p>
<img src="blue.jpg" width="68" height="68" border="0" name="imagen1">
</body>
<script type="text/javascript">
var Oim1=document.images[0];
alert("Tamaño x2");
Oim1.height*=2;
Oim1.width*=2;
alert("Cambio la imagen");
Oim1.src="green.jpg";
```

Ejemplo 4 (II)

```
alert("Añado otra imagen");
var Oim2=new Image();
Oim2.src="blue.jpg";
Oim2.height=Oim1.height;
Oim2.width=Oim1.width;
document.body.appendChild(Oim2);
alert("Añado otra más");
Oim3=document.createElement("img");
Oim3.src="green.jpg";
Oim3.height=Oim1.height;
Oim3.width=Oim1.width;
document.body.appendChild(Oim3);
</script>
</html>
```

forms

- *forms* es un atributo de *document* y contiene la colección de formularios del documento. A través de este vector podremos acceder a los elementos del formulario.
- Cada elemento de *forms* posse la interface *HTMLFormElement* para acceder al mismo. Las propiedades se listan en las tablas siguientes.

Atributo	Descripción
elements	Array con todos los elementos del formulario, en el mismo orden en el que se definen en el documento HTML.
length	Número de elementos del formulario
action, enctype, method	Valor de los atributos HTML
name	Valor del atributo *name*

Método	Descripción
reset()	Resetea el formulario. Equivalente al botón de reset
submit()	Envía el formulario. Equivalente al botón enviar

Acceso al formulario

- Mediante el vector forms
  - *var form1=document.forms[n]*
  - Accede al formulario que ocupa la posición *n* en el documento.
- Mediante el atributo *name*
  - *var form1=document.miformulario*
  - *var form1=document.forms["miformulario"]*
  - Accede al formulario cuyo valor del atributo *name* es *miformulario*
- Localización de elementos por métodos de búsqueda
  - *getElementById()*
  - *getElementsByTagName()*
  - *getElementsByName()*

Acceso a los campos del formulario

- Mediante el vector *elements*, atributo de *forms*, podemos acceder a los elementos del formulario, del mismo modo que en la página anterior.
- Mediante *elements*
  - *form1.elements[n]*
  - Accede al control del formulario que ocupa la posición *n*
- Mediante *name*
  - *form1.textbox1*
  - Accede al control cuyo atributo *name* es *textbox1*
  - *form1.elements["textbox1"]*
  - Como antes, aunque si el valor de *name* contiene espacios, es la única manera
- Localización de elementos por métodos de búsqueda

Propiedades comunes

- Todos los campos del formulario menos los campos ocultos cuentan con propiedades y métodos comunes.

Atributo	Descripción
form	Es un puntero al formulario que pertenece el elemento
name	Obtiene el valor del atributo *name* de HTML. No se puede modificar.
value	Permite leer y modificar el valor del atributo *value* de HTML. Para los campos de texto obtiene el texto introducido en la caja de texto
type	Indica el tipo de elemento (*text, button, checkbox*, ...)
disabled	Indica si el control está desactivado. Se puede modificar

Método	Descripción
blur()	Pierde el foco del ratón sobre el objeto especificado
focus()	Obtiene el foco del ratón sobre el objeto especificado.

La interface *HTMLInputElement*

```
interface HTMLInputElement : HTMLElement {
         attribute        DOMString        defaultValue;
         attribute        boolean          defaultChecked;
readonly attribute        HTMLFormElement  form;
         attribute        DOMString        alt;
         attribute        boolean          checked;
         attribute        boolean          disabled;
         attribute        long             maxLength;
         attribute        DOMString        name;
         attribute        boolean          readOnly;
         attribute        unsigned long    size;
         attribute        DOMString        type;
         attribute        DOMString        value;
...
```

El control *input*

```
interface HTMLInputElement : HTMLElement {
... //Métodos
void blur();
void focus();
void select();
void click();
};
```

- Mediante esta interfaz podemos manipular todos los controles de tipo *input:*
  - *text*
  - *textarea*
  - *password*
  - *checkbox*
  - *radio*

Atributos

- Atributos
  - *defaultValue* → Es una cadena que contiene el valor por defecto que se le ha dado a uno de estos objetos por defecto.
  - *maxlength* → Número máximo de caracteres que puede contener el campo de texto.
  - *checked* → Valor booleano que nos dice si el botón está seleccionado o no.
  - *defaultChecked* → Valor booleano que nos dice si el radio debe estar seleccionado por defecto o no.
- Métodos
  - *click()* → Realiza la acción de pulsado del botón.
  - *select()* → Selecciona el texto dentro del objeto dado
- Cuando se accede a un objeto de tipo *radio* o *checkbox* *(*si comparten *name)* mediante su *name*, éste representa un vector que contiene a todos los elementos *radio* o *checkbox* con el mismo *name*.

Ejemplo 5. *text* y *password*

```
<script type="text/javascript">
function mostrar(){
alert('Su nombre: ' + document.formulario.nombre.value);
alert('El password: ' + document.formulario.pass.value);
}
function llevarnom(){
document.formulario.nombre.blur();
}
function posarnom(){
document.formulario.nombre.focus();
}
function netejar(){
document.formulario.reset();
}
</script>
```

Ejemplo 5 (II)

```
<body>
<form action="procesa.php" name="formulario" id="formulario"
   method="POST" >
Nombre: <input type="text" name="nombre" value="El teu nom"
   maxlength="15" onClick="llevarnom();"><br>
Password: <input type="password" name="pass" maxlength="10"><br>
</form>
<a href="#" onclick="mostrar();return false;">Mostrar datos</a><br>
<a href="#" onclick="posarnom();return false;">Nombre</a><br>
<a href="#" onclick="netejar();return false;">Borrar datos</a><br>
</body>
</html>
```

Ejemplo 6. *radio*

```
function mostrar(boton){
with(document.formulario){
  var msg="Elementos: "+edad.length+"\n";
  msg+="Menor de 18 años: "+edad[0].checked+"\n";
  msg+="Entre 18 y 60 años: "+edad[1].checked+"\n";
  msg+="Mayor de 60 años: "+edad[2].checked+"\n";
}
alert(msg);
}
function seleccio(){
with(document.formulario){
  for(var i=0;i<edad.length && !edad[i].checked;i++);
    if(i<edad.length)
       alert("El valor seleccionad es "+edad[i].value);
    else
        alert("Debería seleccionar un valor");
}
}
```

Ejemplo 6. (II)

```
<html>
...
<body>
<form action="procesa.php" name="formulario" method="GET">
Edad:<br>
<input type="radio" name="edad" value="<18"> Menor de 18 años.<br>
<input type="radio" name="edad" value=">18 y <60" checked>
   Entre 18 y 60 años.<br>
<input type="radio" name="edad" value=">60"> Mayor de 60 años.<br>
<input type="button" value="Selección" onClick="seleccio();">
</form>
<A href="#" onclick="mostrar(); return false">Ver valores</A>
</body>
</html>
```

La interface *HTMLSelectElement*

```
interface HTMLSelectElement : HTMLElement {
readonly attribute        DOMString                type;
         attribute        long                     selectedIndex;
         attribute        DOMString                value;
         attribute        unsigned long            length;
readonly attribute        HTMLFormElement          form;
readonly attribute        HTMLOptionsCollection    options;
         attribute        boolean                  disabled;
         attribute        boolean                  multiple;
         attribute        DOMString                name;
         attribute        long                     size;
void     add(in HTMLElement element, in HTMLElement before)
void     remove(in long index);
void     blur();
void     focus();
};
```

El control *select*

- Este objeto representa una lista de opciones dentro de un formulario.
- Puede tratarse de una lista desplegable de la que podremos escoger alguna (o algunas) de sus opciones.
- Atributos
  - *length*. Valor numérico que nos indica cuántas opciones tiene la lista
  - *size* → Número de filas visibles
  - *selectedIndex* → Valor del índice de la opción seleccionada
  - *multiple* → Indica si se puede seleccionar más de una opción
  - *options*. Vector de elementos *option* que contiene cada una de las opciones de la lista.
- Métodos
  - *add(element, before)* → Añade un nuevo elemento al vector *options*. Es equivalente a *appendChild* si el parámetro *before* es null y equivalente *a insertBefore* en otro caso.
  - *remove(index)* → Elimina un elemento del vector *options.* Si *index* no es corrrecto no realiza ninguna acción.

Elemento *option*

- Cada elemento del vector *options* posee las siguientes propiedades:
  - *defaultSelected* → Booleano que indica si la opción está seleccionada por defecto.
  - *index* → Valor numérico que nos da la posición de la opción dentro de la lista.
  - *length* → Valor numérico que nos dice cuántas opciones tiene la lista.
  - *selected* → Valor booleano que indica si la opción está actualmente seleccionada.
  - *text* → Cadena con el texto mostrado en una opción concreta.
  - *value* → Es una cadena que contiene el valor del parámetro *value* de la opción concreta de la lista.

Ejemplo 7 (I)

```
<script type="text/javascript">
function mostrar(boton)
{
with(document.formulario){
  msg="Elementos: "+edad.length+"\n";
  msg+="Edad: "+edad.options[edad.selectedIndex].value+"\n";
   msg+=edad.options[edad.selectedIndex].text+"\n";
   msg+="Lista Múltiple: "+edad.multiple;
   msg+="Posición: "+edad.options[edad.selectedIndex].index+"\n";
   msg+="Posición: "+edad.selectedIndex;
}
alert(msg);
}
</script>
```

Ejemplo 7 (II)

```
<html>
...
<form action="procesa.php" name="formulario" method="GET">
<p>Edad:<br/>
<select name="edad">
   <option value="<18" selected>Menor de 18 años</option>
   <option value=">18 y <60">Entre 18 y 60 años</option>
   <option value=">60">Mayor de 60 años</option>
</select>
</p>
</form>
<a href="#" onclick="mostrar(); return false">Ver valores</a>
</body>
</html>
```

Tablas

- DOM proporciona métodos específicos para trabajar con tablas.
- Mediante las interfaces definidas en DOM para las tablas podemos acceder a los elementos a nivel de tabla, fila o celda:
  - HTMLTableElement
  - HTMLTableRowElement
  - HTMLTableColElement
  - HTMLTableCellElement,

Interfaz *TableElement* (I)

```
interface HTMLTableElement : HTMLElement {
          attribute      HTMLTableCaptionElement  caption;
          attribute      HTMLTableSectionElement  tHead;
          attribute      HTMLTableSectionElement  tFoot;
readonly  attribute      HTMLCollection           rows;
readonly  attribute      HTMLCollection           tBodies;
          attribute      DOMString                align;
          attribute      DOMString                bgColor;
          attribute      DOMString                border;
          attribute      DOMString                cellPadding;
          attribute      DOMString                cellSpacing;
          attribute      DOMString                frame;
          attribute      DOMString                rules;
          attribute      DOMString                summary;
          attribute      DOMString                width;
```

Interfaz *TableElement* (II)

```
...
//interface HTMLTableElement : HTMLElement
//Métodos
HTMLElement    createTHead();
void           deleteTHead();
HTMLElement    createTFoot();
void           deleteTFoot();
HTMLElement    createCaption();
void           deleteCaption();
HTMLElement    insertRow(in long index)
void           deleteRow(in long index)
};
```

Interfaz *HTMLTableColElement*

```
interface HTMLTableColElement : HTMLElement {
   attribute   DOMString      align;
   attribute   DOMString      ch;
   attribute   long           span;
   attribute   DOMString      vAlign;
   attribute   DOMString      width;
};
```

Interfaz *HTMLTableRowElement*

```
interface HTMLTableRowElement : HTMLElement {
readonly  attribute   long            rowIndex;
readonly  attribute   long            sectionRowIndex;
readonly  attribute   HTMLCollection  cells;
          attribute   DOMString       align;
          attribute   DOMString       bgColor;
          attribute   DOMString       ch;
          attribute   DOMString       chOff;
          attribute   DOMString       vAlign;
HTMLElement           insertCell(in long index) ;
void                  deleteCell(in long index);
 };
```

Interfaz *HTMLTableCellElement*

```
interface HTMLTableCellElement : HTMLElement {
readonly attribute        long              cellIndex;
         attribute        DOMString         abbr;
         attribute        DOMString         align;
         attribute        DOMString         axis;
         attribute        DOMString         bgColor;
         attribute        DOMString         ch;
         attribute        DOMString         chOff;
         attribute        long              colSpan;
         attribute        DOMString         headers;
         attribute        DOMString         height;
         attribute        boolean           noWrap;
         attribute        long              rowSpan;
         attribute        DOMString         scope;
         attribute        DOMString         vAlign;
         attribute        DOMString         width;
};
```

Ejemplo 8 (I)

```
<html>
<head>
<title>Exercici DOM</title>
<script type="text/javascript">
function appFila()
{
   var ncols=parseInt(prompt("Numero de columnas",""));
   var tabla=document.getElementsnByName("taula");
//  var nodotabla=document.getElementById("tab");
   var fila=document.createElement("tr");
   for(var i=0;i<ncols;i++){
     var col=document.createElement("td");
     col.appendChild(document.createTextNode("columna "+i));
     fila.appendChild(col);
   }
   tabla[0].appendChild(fila);
//  nodotabla.appendChild(fila);
}
```

Ejemplo 8 (II)

```
...
</script>
</head>
<body>
<table id="tab" name="taula" border="1">
</table>
<form name="formulari">
<input type="button" name="boton" value="Nueva Fila"
   onclick="appFila()">
</form>
</body>
</html>
```

Ejemplo 9

```
colors=new Array("Red","Orange","Yellow","Green","Cyan","Blue","BlueViolet");
function creaTabla()
{
var nfils=parseInt(prompt("Numero de filas",""));
var ncols=parseInt(prompt("Numero de columnas",""));
var Otabla=document.getElementById("tab");
for(var i=0;i<nfils;i++){
  Otabla.insertRow(i);
  for(var j=0;j<ncols;j++){
   Otabla.rows[i].insertCell(j);
   Otabla.rows[i].cells[j].appendChild(document.createTextNode("Celda "+i+"
   "+j));
    Otabla.rows[i].bgColor=colors[(i*ncols+j)%colors.length];
    //Otabla.rows[i].cells[j].bgColor=colors[(i*ncols+j)%colors.length];
  }
}
}
```