Document Object Model

Tema 6

DOM

- El Document Object Model (DOM) es una interfaz de programación para documentos HTML y XML.
- Establece una representación estructurada del documento.
- Proporciona interfaces para acceder y modificar la estructura, el contenido y la presentación del documento.

DOM

- Representa el documento como un conjunto de nodos estructurados con sus propiedades y métodos.

- Ofrece métodos para navegar entre los elementos del documento, acceder a elementos determinados, a su contenido o propiedades y modificarlos o crear nuevos elementos.

DOM

- Se trata de un estándar del W3C www.w3.org/DOM/

plenamente implementado por todos los Navegadores.

DOM

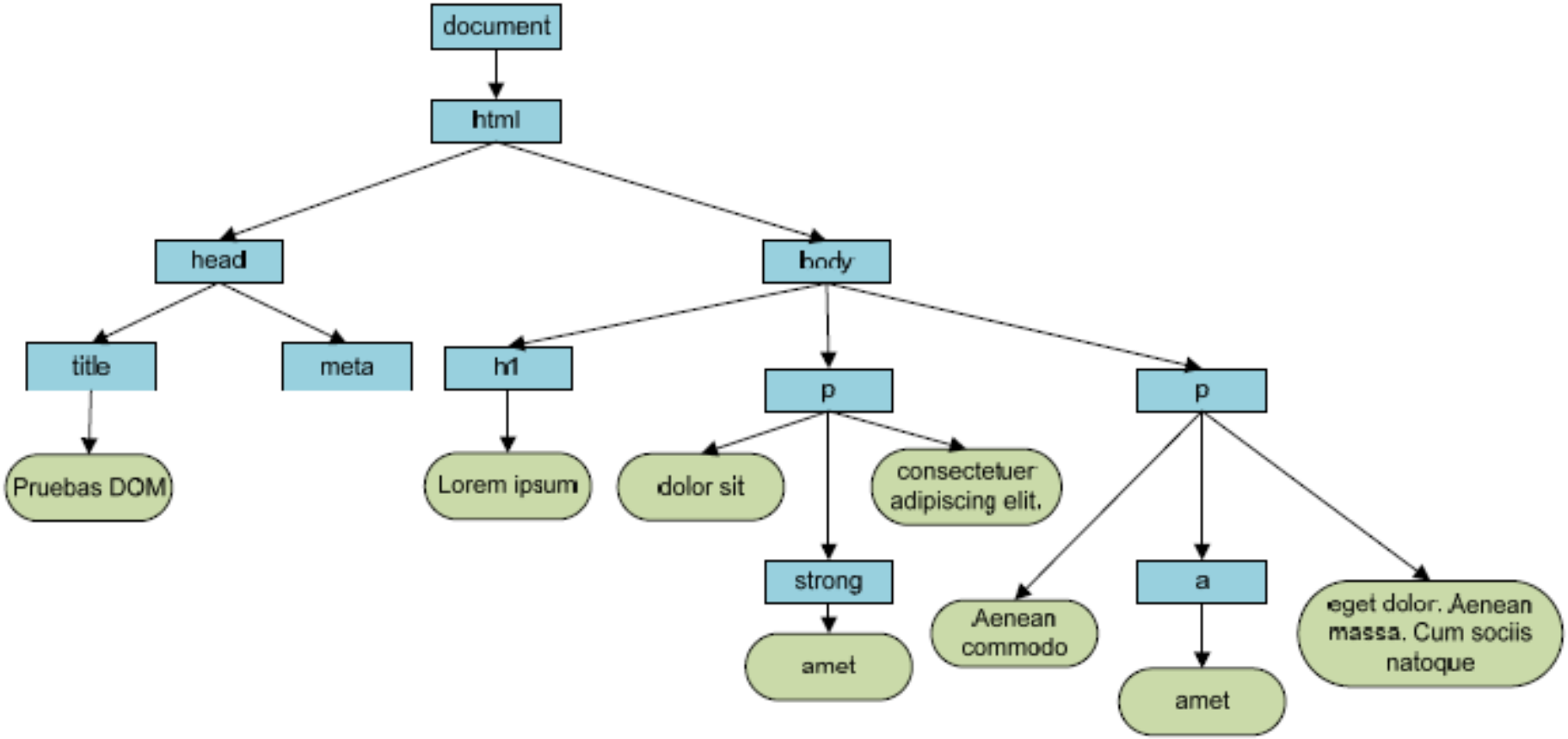
- DOM considera un documento xhtml (bien formado)como un árbol de nodos.
- Por ejemplo:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
        "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es">
<head>
  <title>Pruebas DOM</title>
  <meta http-equiv="content-type" content="text/html; charset=iso-8859-1" />
</head>
<body><h1>Lorem ipsum</h1>
<p>dolor sit <strong>amet</strong>, consectetuer adipiscing elit.</p>

<p>Aenean commodo <a href="http://www.upsam.com">ligula</a> eget dolor. Aenean massa.
   Cum sociisnatoque
</p>
</body>
</html>
```

DOM

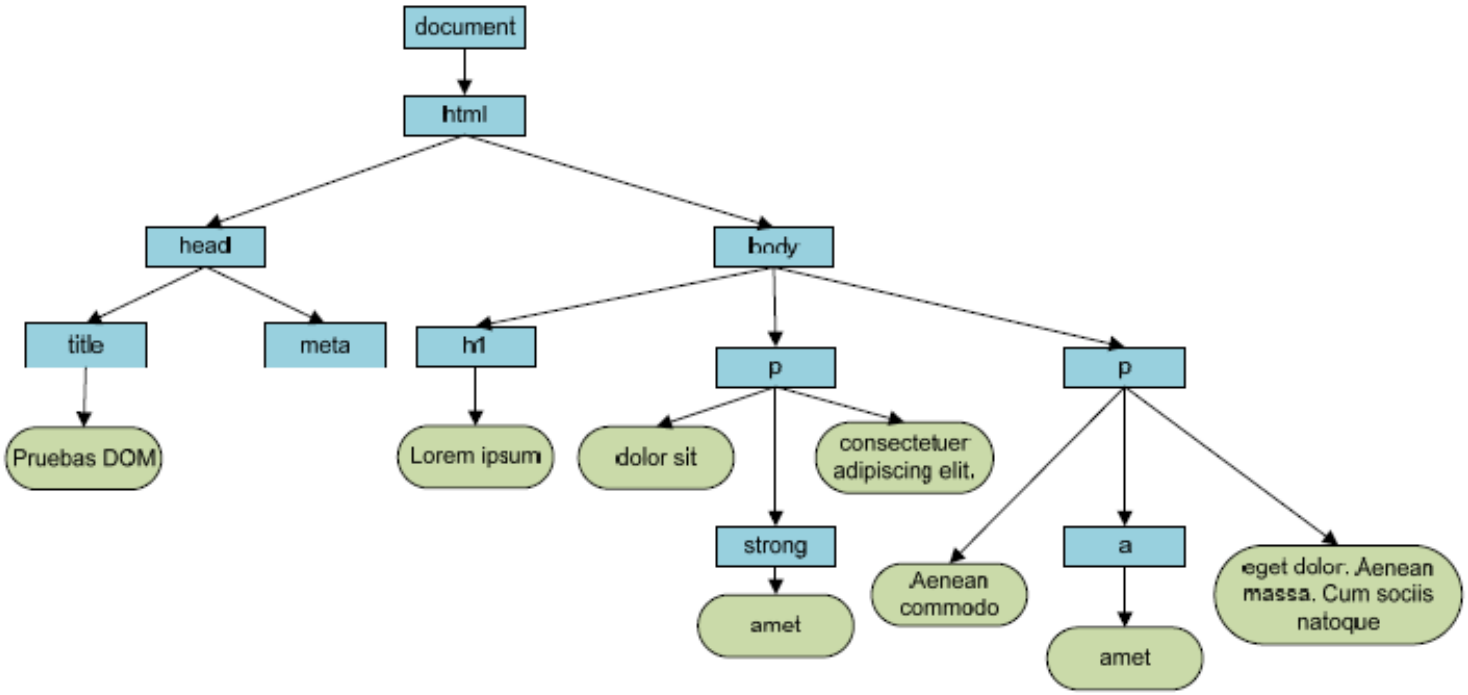
- Este sería el árbol de nodos resultante...



```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es">
<head>
  <title>Pruebas DOM</title>
  <meta http-equiv="content-type" content="text/html; charset=iso-8859-1" />
</head>
<body><h1>Lorem ipsum</h1>
<p>dolor sit <strong>amet</strong>, consectetuer adipiscing elit.</p>

<p>Aenean commodo <a href="http://www.upsam.com">ligula</a> eget dolor. Aenean massa.
   Cum sociisnatoque
</p>
</body>
</html>
```

- Este sería el árbol de nodos resultante...



ACCESO DIRECTO

Acceso directo a partir de las características de un nodo.

- Tanto el objeto document como el objeto element tienen métodos para acceder a un nodo a partir de la etiqueta html, el valor de la propiedad name (obsoleto) o mediante el id de un elemento.
- **getElementsByTagName, getElementsByName y getElementsById**.

DOM

Método getElementsByTagName.

- Devuelve un nodeList (array DE Nodos) con los nodos que correspondan a una etiqueta html.

nodo.getElementsByTagName(etiquetaHTML)

- Devuelve los nodos cuya etiqueta sea igual a *etiquetaHTML que se* encuentre dentro de *nodo.*

```
let parrafos = document.getElementsByTagName("p")
```

la variable parrafos se cargaría con todos los elementos p del documento.

Párrafos se convierte en una array de elementos.

- En el siguiente ejemplo, se obtienen todos los enlaces del primer párrafo de un document :

- var parrafos = document.getElementsByTagName("p");
- var primerParrafo = parrafos[0];
- var enlaces = primerParrafo.getElementsByTagName("a");

Función getElementsByName()

- La función getElementsByName() obtiene todos los elementos de la página XHTML cuyo atributo name coincida con el parámetro que se le pasa a la función.

DOM

Método getElementoById

- Devuelve el nodo que tenga como valor del atributo id del dato que se pasa como argumento.
- *nodo.getElementById(valorID)*

. Como el atributo `id` debe ser único para cada elemento de una misma página , la función devuelve únicamente el nodo deseado.

```
var cabecera = document.getElementById("cabecera");
 <div id="cabecera">
     <a href="/" id="logo">...</a>
 </div>
```

QUERYSELECTOR()

- La función querySelector() acepta como parámetro un selector que identifica el elemento (o elementos) a seleccionar.
- En el caso de esta función, únicamente es devuelto **el primer elemento** que cumple la condición.
- Si no existe el elemento, el valor retornado es null.

```
var logo = document.querySelector(".enlace");
```

```
<div id="cabecera">
  <a href="/" class="enlace">...</a>
</div>
<div id="cuerpo">
    <p>Loren ipsum <a href="enlace">...</a></p>
</div>
```

QUERYSELECTORALL()

- La función querySelectorAll() acepta como parámetro un selector que identifica el elemento (o elementos) a seleccionar.
- Esta función devuelve un objeto de tipo NodeList con los elementos que coincidan con el selector.

ejemplo

```
var enlaces = document.querySelectorAll(".enlace");
```

```
<div id="cabecera">
  <a href="/" class="enlace">...</a>
</div>
<div id="cuerpo">
    <p>Loren ipsum <a href="enlace">...</a></p>
</div>
```

```
for (var i=0; i<enlaces.length; i++) {
    var enlaces = enlaces[i];
}
```

Tipos de Nodos

- La especificación completa de DOM define 12 tipos de nodos, aunque las páginas XHTML 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 XHTML. 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 XHTML.
- **Comment,** representa los comentarios incluidos en la página XHTML.

Una vez que DOM ha creado de forma automática el árbol completo de nodos de la página, ya es posible utilizar sus funciones para obtener información sobre los nodos o manipular su contenido.

JavaScript crea el objeto **Node** para definir las propiedades y métodos necesarios para procesar y manipular los documentos.

En primer lugar, el objeto **Node** define las siguientes constantes para la identificación de los distintos tipos de nodos:

DOM

- ELEMENT_NODE: 1,
- ATTRIBUTE_NODE: 2,
- TEXT_NODE: 3,
- CDATA_SECTION_NODE: 4,
- ENTITY_REFERENCE_NODE: 5,
- ENTITY_NODE: 6,
- PROCESSING_INSTRUCTION_NODE: 7,
- COMMENT_NODE: 8,
- DOCUMENT_NODE: 9,
- DOCUMENT_TYPE_NODE: 10,
- DOCUMENT_FRAGMENT_NODE: 11,
- NOTATION_NODE: 12

- Para saber que tipo de nodo :propiedad **nodeType**

Extracción de los elementos del objeto Node

- for (var key in Node) {
- alert(key + ' = ' + Node[key]);
- };

Propiedades y métodos

Propiedad/Método	Valor devuelto	Descripción
`nodeName`	`String`	El nombre del nodo (no está definido para algunos tipos de nodo)
`nodeValue`	`String`	El valor del nodo (no está definido para algunos tipos de nodo)
`nodeType`	`Number`	Una de las 12 constantes definidas anteriormente
`ownerDocument`	`Document`	Referencia del documento al que pertenece el nodo
`firstChild`	`Node`	Referencia del primer nodo de la lista `childNodes`
`lastChild`	`Node`	Referencia del último nodo de la lista `childNodes`
`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
`hasChildNodes()`	`Bolean`	Devuelve `true` si el nodo actual tiene uno o más nodos hijo

Propiedades y métodos

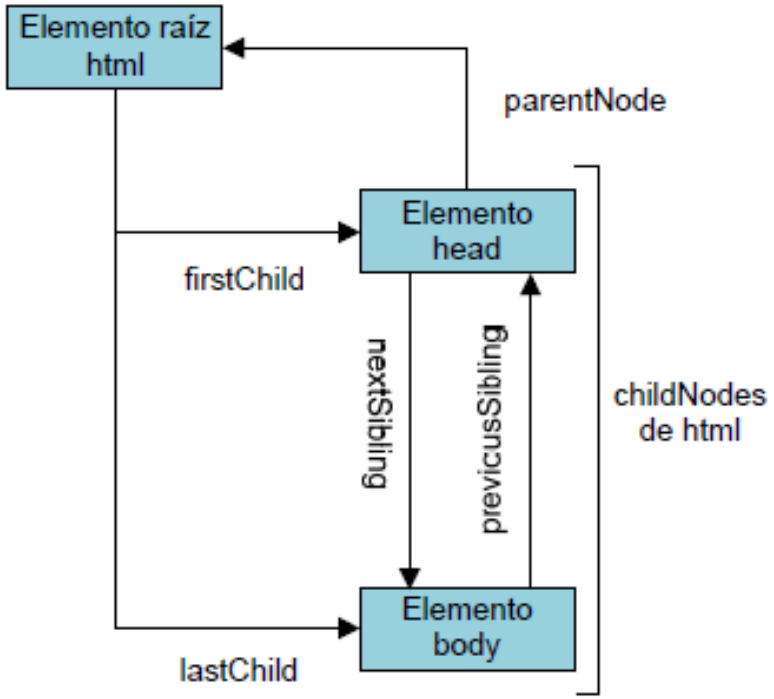
Propiedad/Método	Valor devuelto	Descripción
`attributes`	`NamedNodeMap`	Se emplea con nodos de tipo `Element`. Contiene objetos de tipo `Attr` que definen todos los atributos del elemento
`appendChild(nodo)`	`Node`	Añade un nuevo nodo al final de la lista `childNodes`
`removeChild(nodo)`	`Node`	Elimina un nodo de la lista `childNodes`
`replaceChild(nuevoNodo, anteriorNodo)`	`Node`	Reemplaza el nodo `anteriorNodo` por el nodo `nuevoNodo`
`insertBefore(nuevoNodo, anteriorNodo)`	`Node`	Inserta el nodo `nuevoNodo` antes que la posición del nodo `anteriorNodo` dentro de la lista `childNodes`

DOM(atajos)

- DOM proporciona distintas interfaces para acceder a los nodos.
- Acceso a los elementos raíz del documento.
- document.body, hace referencia al elemento body del documento.
- document.documentElement, hace referencia al elemento html.

DOM

- Relaciones entre elementos.
  - Los términos padre (parent), hijo (child) y hermanos (sibling) definen las relaciones entre nodos.
  - El nodo de jerarquía superior sería el nodo raíz.
  - Cada nodo, excepto el raíz tiene un nodo padre.
  - Un nodo puede tener cualquier número de hijos.
  - Una hoja es un nodo sin hijos.
  - Los nodos hermanos, son nodos del mismo padre.



DOM

Existen dos formas de acceder:

Acceso a partir de otros nodos.

- Los elementos del árbol tienen las propiedades **parentNode, firstChild, lastChild, nextSibling y previousSibling** que devuelven nodos a partir de un nodo dado.
- La propiedad **childNodes** devuelve un nodeList de los elementos hijos de un nodo dado.(lenght)
- **hasChildNodes()**

Nos devuelve true o false en caso de que tenga o no hijos el nodo.

Acceso directo a partir de las características de un nodo.

- Tanto el objeto document como el objeto element tienen métodos para acceder a un nodo a partir de la etiqueta html, el valor de la propiedad name (obsoleto) o mediante el id de un elemento.
- **getElementsByTagName, getElementsByName y getElementsById**.

DOM
Propiedades de los nodos

nodo.nodeName.

- Devuelve una cadena con el nombre de *nodo.*
- Propiedad de sólo lectura.
- Según el tipo del nodo devolverá:
  - Document, "#document".
  - Element, en un documento html, el nombre de la etiqueta html.
  - Attr, el nombre del atributo.
  - text, "#text".
  - Comment, "#coment".

DOM
Propiedades de los nodos

nodo.nodeValue.

- Devuelve o establece el valor de *nodo.*
- El valor será para los distintos tipos de nodos…
  - Document, null.
  - Element, null.
  - Attr, valor del atributo.
  - text, contenido del texto.
  - Comment, contenido del comentario.

DOM
Propiedades de los nodos

nodo.nodeType.

- Devuelve valor numérico con el tipo de *nodo.*
- Propiedad de sólo lectura.
- Según el tipo del nodo devolverá:
  - Document, 9.
  - Element, 1.
  - Attr, 2.
  - text, 3.
  - Comment, 8.

DOM
Propiedades de los nodos

nodo.innerHTML

- Devuelve o establece el contenido HTML de *nodo.*
  - Aunque no forma parte del estándar del W3C, la gran mayoría de los navegadores la utiliza.
  - Se emplea comúnmente para modificar de forma dinámica el código html de un documento.

nodo.childNodes

- Devuelve un nodeList con los nodos hijos de *nodo.*
- Funciona de forma distinta en Mozilla (FireFox y Chrome) e IE. En Mozilla cuenta como nodo los espacios entre elementos, mientras que IE sólo cuenta como elementos los elementos html.
  - Para acceder a los nodos es mejor utilizar el método directo.

Acceso a los tipos de nodo (extrayendo el valor):

- obj_tipo_document= document.nodeType; //9, o DOCUMENT_NODE
- obj_tipo_elemento= document.documentElement.nodeType; //1, o ELEMENT_NODE

Acceso a los tipos de nodo (comparando con el valor):

- alert(document.nodeType == Node.DOCUMENT_NODE); // true
- alert(document.documentEle

Acceso al texto de un nodo de tipo texto:

- Extraer el texto de un nodo: var x = document.getElementById("miNodo").textContent;
- Cambiar el texto de un nodo: document.getElementById("miNodo").textContent = "Paragraph changed!";

Ejercicio

```
<html>
<head>
  <title>Aprendiendo DOM</title>
</head>
<body>
  <p>Aprendiendo DOM</p>
  <p>DOM es sencillo de aprender</p>
  <p>Ademas, DOM es muy potente</p>
</body>
</html>
```

•Elemento raíz de la página:

```
var objeto_html = document.documentElement;
Visualizar objeto_html y el tipo de nodo.
```

•Obtener los elementos `<head>` y `<body>`

```
var objeto_head = objeto_html.firstChild;
var objeto_body = objeto_html.lastChild;
```

•Otra forma directa de obtener los dos nodos consiste en utilizar la propiedad `childNodes` del elemento `<html>`:

```
var objeto_head = objeto_html.childNodes[0];
var objeto_body = objeto_html.childNodes[1];
```

•Si se desconoce el número de nodos hijo que dispone un nodo, se puede emplear la propiedad `length` de `childNodes`:

```
var numeroDescendientes = objeto_html.childNodes.length;
 o bien se puede utilizar hasChildNodes.
```

•Además, el DOM de HTML permite acceder directamente al elemento `<body>` utilizando el atajo `document.body`:

```
var objeto_body = document.body;
```

Además de las propiedades anteriores, existen otras propiedades como **`previousSibling` y `parentNode`** que se pueden utilizar para acceder a un nodo a partir de otro.
Probad:

```
objeto_head.parentNode
objeto_body.parentNode
objeto_body.previousSibling
objeto_head.nextSibling
objeto_head.ownerDocument
```

DOM
Atributos

- Los elementos html de DOM, tienen propiedades para cada uno de los atributos definidos en los elementos html.
- Se accede mediante *elemento.nombreAtributo*.
- Los nombre de los atributos son los mismos que en html, con la excepción del atributo class que es className.
- Devuelve el valor del atributo *nombreAtributo de elemento.*
- Se puede utilizar para establecer el valor de un atributo.
- Para modificar el valor de un atributo, no se debe utilizar la colección **attributes.**

El método `setAttribute` permite crear un nuevo nodo de atributo en un elemento DOM.

nodo`.setAttribute(`*nombreAtributo*`, `*valorAttributo*`)`

- Crea o establece el valor de un atributo del elemento *nodo*.
  - ✓ Si *nombreAtributo* ya existe modifica su valor.
  - ✓ Si *nombreAtributo* no existe, crea un nuevo atributo.
- No devuelve nada.

- El método `removeAttribute` elimina un atributo de un elemento.

  nodo`.removeAttribute(`*nombreAtributo*`)`

  - Elimina *nombreAtributo* de nodo.
  - Si el atributo no existe genera una excepción.
    - ✓ Se puede utilizar el método *nodo*`.hasAttribute(`*nombreAtributo*`)` para determinar si un elemento tiene un atributo.

- El método `getAttribute` permite obtener el valor de un atributo.

  nodo`.getAttribute(`*nombreAtributo*`)`
  - Devuelve una cadena con el valor del atributo.
  - Si el atributo no existe, devuelve una cadena nula o el valor nulo.

Acceso a los nodos de tipo atributo:

getAttribute(nomAtributo)

- Equivale a attributes.getNameItem(nomAtributo)

setAttribute(nomAtributo, valorAtributo)

- Equivale a attributes.getNamedItem(nomAtributo).value=valor

removeAttribute(nomAtributo)

- Equivale a attributes.removeNameItem(nomAtributo).

DOM
Modificar la estructura

El modelo de objetos DOM proporciona los métodos necesarios para modificar la estructura de DOM.

- Algunos métodos…
  - Los métodos **createElement y createTextElement** permiten crear nuevos nodos.
  - Los métodos **appendChild e insertBefore** permiten insertar nuevos nodos en la estructura DOM.
  - El método **removeChild** permite eliminar nodos.
  - El método **replaceChild** permite sustituir un nodo por otro.
  - El método **cloneNode** permite copiar un nodo.

DOM
Modificar la estructura

- En el árbol de nodos, los elementos html con contenido presentan,al menos, dos nodos:

- Un **nodo Element** con la etiqueta.
- Un **nodo Text** con el contenido de la etiqueta que será hijo del nodo Element.

DOM
Modificar la estructura

Para añadir un nuevo nodo html habrá que…

- Crear un nuevo nodo de tipo Element que represente a la etiqueta mediante el método del objeto Document createElement.

- Crear un nuevo nodo de tipo Text que con el contenido del elemento mediante el método del objeto Document createTextNode.

- Añadir el nodo de tipo Text al elemento con el método appendChild.
- Añadir el elemento en la página en el lugar correspondiente.
  - El método appendChild inserta el elemento como último nodo del padre.
  - El método insertBefore inserta el elemento dentro del nodo padre, antes otro hijo.

DOM
Modificar la estructura

Método createElement:

document.createElement(etiquetaHTML)

- *etiquetaHTML es una cadena con la etiqueta.*
- Devuelve un nodo de tipo Element con la etiqueta especificada.

DOM
Modificar la estructura

Método createTextElement.

document.createTextNode(contenido)

- Devuelve un nodo de tipo Text con el contenido especificado.

DOM
Modificar la estructura

Método appendChild:

nodoPadre.appendChild(nodoHijo)

- Hace que *nodoHijo se coloque como último hijo de nodoPadre.*
- Si *nodoHijo ya existe, lo elimina de dónde esté y lo coloca en la nueva posición.*

Ejemplo de creación de nodos en un documento:

```
var h = document.createElement("h1") ; //crea el elemento h1
var t = document.createTextNode("Hola, mundo");  //crea el nodo de texto
h.appendChild(t); //añade el texto al elemento h1

var att = document.createAttribute("class"); //crea el atributo
att.value = "prueba"; // añade el valor del atributo
h.setAttributeNode(att); //añade el atributo al nodo h1 creado
```

RESULTADO: <h1 class="prueba">Hola, mundo</h1>

PARA AÑADIR UN ELEMENTO AL BODY:

```
document.body.appendChild(h);
```

RESULTADO:

```
<body>
    <h1 class="prueba">Hola, mundo</h1>
</body>
```

PARA AÑADIR UN ELEMENTO A OTRO ELEMENTO QUE YA ESTÁ EN EL HTML:

```
var miDiv = document.getElementById("miDiv");
miDiv.appendChild(h);
```

RESULTADO:

```
<body>
    <div id="miDiv">
        <h1 class="prueba">Hola, mundo</h1>
    </div>
</body>
```

DOM
Modificar la estructura

Método insertBefore:

nodoPadre.insertBefore(nodoAñadido, nodoSiguiente)

- Inserta el *nodoAñadido como hijo de nodoPadre antes del nodoSiguiente* referenciado.
- Si *nodoSiguiente no existe, lo inserta como último nodo de nodoPadre.*

DOM
Modificar la estructura

El método removeChild permite eliminar un nodo hijo de un nodo.

nodoPadre.removeChild(nodo)

- Elimina *nodo de nodoPadre.*
- Devuelve el nodo eliminado.
- Aunque *nodo no esté dentro de DOM se mantiene en* memoria, por lo que es posible reutilizarlo.
- Si *nodo no existe, se genera una excepción.*
- Para asegurarse de quién es *nodoPadre se puede* utilizar la propiedad parentNode de *nodo.*

DOM
Modificar la estructura

El método replaceChild permite cambiar un nodo por otro.

nodoPadre.replaceChild(nodoNuevo, nodoViejo)

- Cambia *nodoViejo por nodoNuevo.*
- Devuelve *nodoViejo.*
- Si *nodoNuevo ya existe, primero lo elimina.*
- Si *nodoViejo no existe, genera una excepción.*

DOM
Modificar la estructura

El método cloneNode devuelve una copia de un nodo.

nodo.cloneNode(copiarHijos)

● cloneNode no inserta nada, sólo devuelve una copia de *nodo con todos* sus atributos.

Si se desea incluir esa copia en el árbol de nodo habría que recurrir al método appendChild.

copiarHijos es un valor lógico.

- Si se pone a falso, no se clonan los nodos hijos, incluido el contenido del
- nodo.
- function remplazarÚltimoPorPrimero(){
- var nodoViejo = document.getElementById("ultimoParrafo");
- var nodoNuevo =
- document.getElementById("primerparrafo").cloneNode(true);
- document.body.replaceChild(nodoNuevo,nodoViejo);