Document Object Model

Document Object Model

- 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.
- Representa el documento como un conjunto de nodos estructurados con sus propiedades y métodos.
- Mediante las API de DOM, dicha representación puede modificarse desde cualquier lenguaje de programación.
  - Ofrece métodos para navegar entre los elementos del documento, acceder a elementos determinados, a su contenido o propiedades y modificarlos.
- Se trata de un estándar del W3C (www.w3.org/DOM/) no plenamente implementado por todos los navegadores.

Document Object Model
El árbol del documento

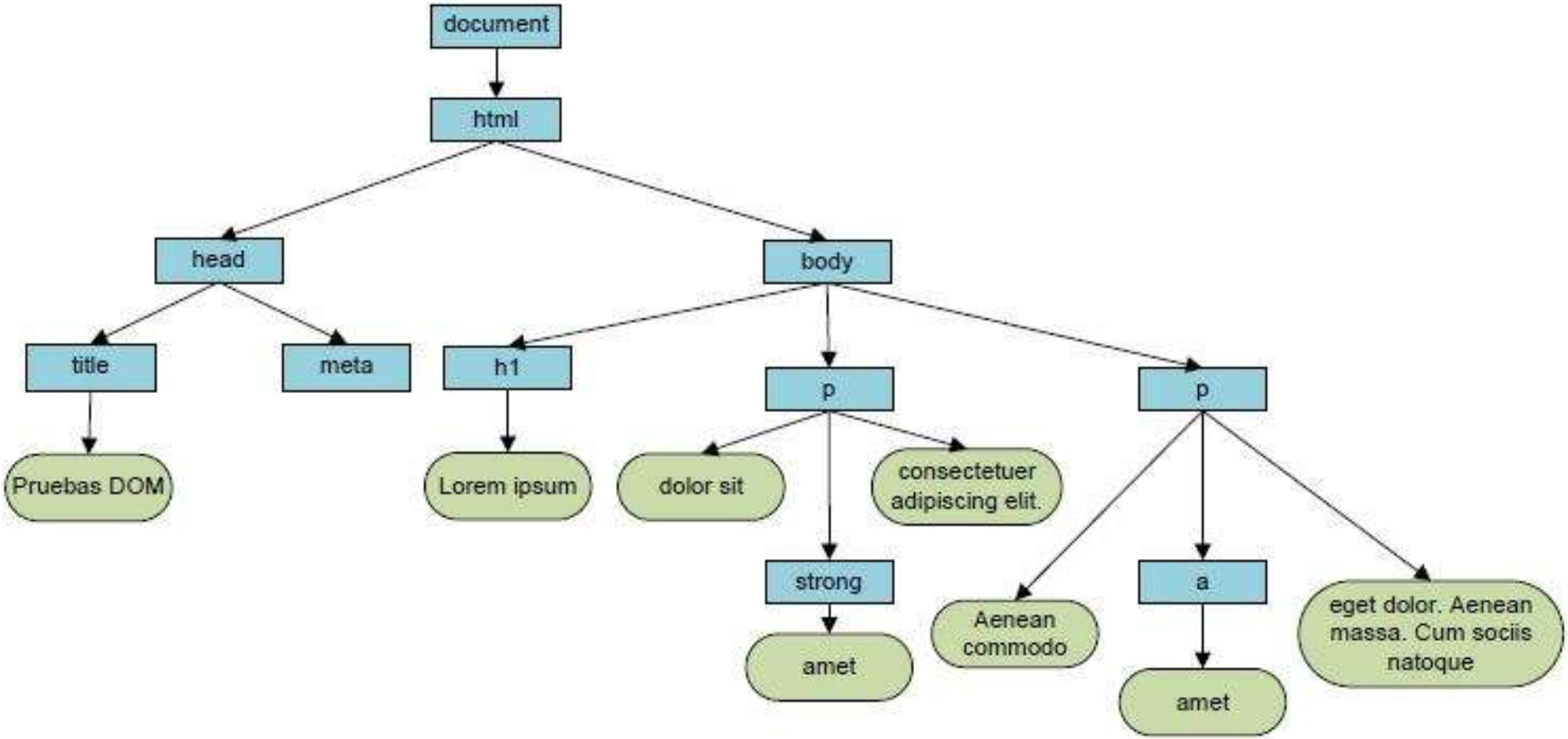
- DOM considera un documento xhtml (bien formado) como un árbol de nodos.
- Para el siguiente código html...

```
<?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>
```

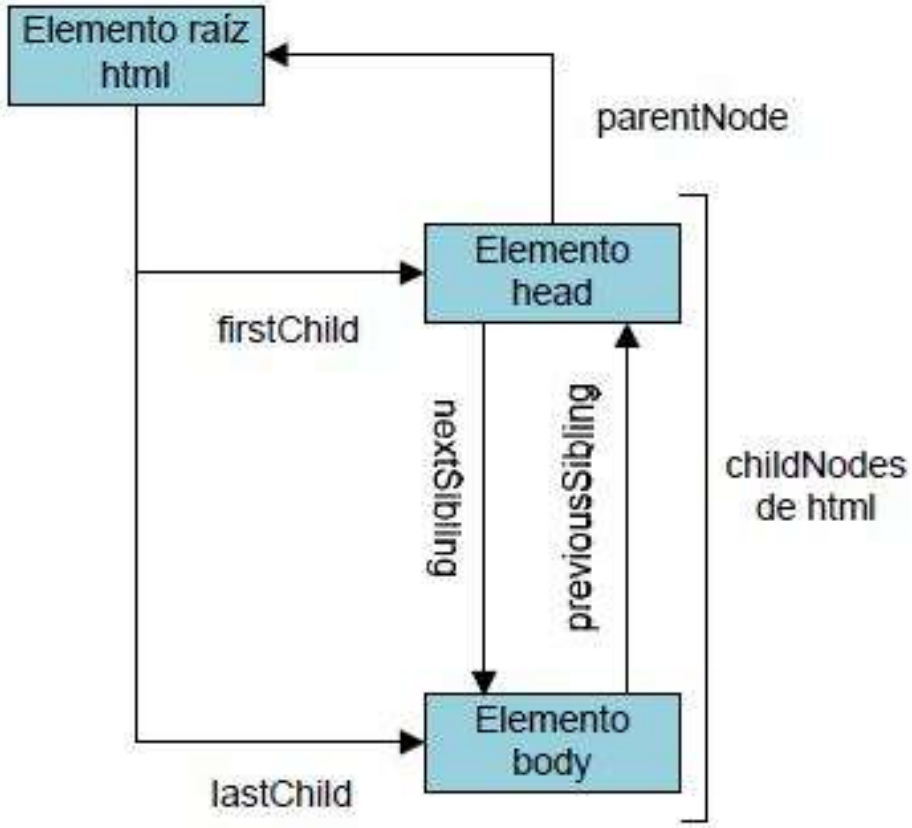
Document Object Model
El árbol del documento (II)

- Este sería el árbol de nodos resultante...



Document Object Model
El árbol del documento (IV)

- 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.



Document object model
Tipos de nodos

- DOM define 12 tipos de nodos. Los más importantes son:
  - `Document`. Hace referencia al nodo raíz del que derivan todos los demás nodos.
  - `Element`. Representa cada una de las etiquetas del documento.
    - ✓ Puede contener atributos y de él pueden derivar otros nodos.
  - `Attr`. Representa cada uno de los atributos de una etiqueta, representados por parejas *`nombreAtributo = valor`*.
  - `Text`. Representa el contenido de un elemento.
  - `Coment`. Representa un comentario.
- El resto de tipos de datos son `DocumentType`, `CDataSection`, `DocumentFragment`, `Entity`, `EntityReference`, `ProcessingInstruction` y `Notation`.

Document object model
Tipos de datos

- DOM utiliza los siguientes tipos de datos (clases):
  - `document`. Hace referencia al elemento raíz. Se trataría de un nodo de tipo documento.
  - `element`. Hace referencia a un nodo de tipo elemento.
  - `nodeList`. Se trata de un conjunto de nodos.
    - ✓ Para hacer referencia a cada uno de ellos se utiliza la sintaxis de arrays.
      - Si `lista` es un `nodeList`, se accedería al primer elemento mediante `lista[i]`.
  - `attribute`. Se corresponde a un atributo de un elemento.
  - `namedNodeMap`. Una lista de nodos especial a la que se puede acceder tanto a partir del índice como del nombre del elemento.

Document Object Model
Acceso a los nodos

- 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`.
- 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.
  - 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.
      - o `getElementsByTagName`, `getElementsByName` y `getElementById`.

Document Object Model
Acceso a los nodos (II)

- Método `getElementsByTagName`.
  - Devuelve un `nodeList` 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*.

      `var párrafos = document.getElementsByTagName("p")`

      o `párrafos` se cargaría con todos los elementos `p` del documento.

      `var enlaces = párrafos[0].getElementsByTagName("a")`

      o `enlaces` se cargaría con todos los elementos `a` del primer párrafo del documento.

```
<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 <a href="http://www.colimbo.net">massa</a>. Cum sociis natoque</p>
<script type="text/javascript">
//<![CDATA[
   var parrafos = document.getElementsByTagName("p");
   document.write("Num. de p&aacute;rrafos del documento:" + parrafos.length + "<br>");
   var enlaces = parrafos[1].getElementsByTagName("a");
   for(var i=0; i < enlaces.length;i++)
       document.write(enlaces[i].innerHTML + "<br>");
   // Escribe ligula y masa
//]]>
</script>
</body>
```

Document Object Model
Acceso a los nodos (III)

- Método `getElementById`.
  - Devuelve el nodo que tenga como valor del atributo id el dato que se pasa como argumento.

    nodo`.getElementById(`*valorID*`)`
    - ✓ Devuelve el elemento html, descendiente de *nodo* cuyo identificador sea igual a *valorID*.

```
<body>
<h1>Lorem ipsum</h1>
<p>dolor sit <strong>amet</strong>, consectetuer adipiscing elit.</p>
<p id="segundoparrafo">Aenean commodo <a href="http://www.upsam.com">ligula</a> eget dolor.
Aenean <a href="http://www.colimbo.net">massa</a>. Cum sociis natoque</p>
<script type="text/javascript">
//<![CDATA[
   var nodo = document.getElementById("segundoparrafo");
   var enlaces = nodo.getElementsByTagName("a");
   for(var i=0; i < enlaces.length;i++)
       document.write(enlaces[i].innerHTML + "<br>");
   // Escribe ligula y masa
//]]>
</script>
</body>
```

Document Object Model
Algunas 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".
    - ✓ Coment, "#coment".
- *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.
    - ✓ Coment, contenido del comentario.

Document Object Model
Algunas propiedades de los nodos (II)

- *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.
    - ✓ `Coment`, 8.
- *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 `getElementById`.

Element VS text node

```
<div>
   <p> Este parrafo tiene un
     <a href="page.html">link</a>.
   </p>
</div>
```

- Q: ¿Cuántos hijos tiene el div anterior?
- A: 3
  - Un nodo element <p>
  - Dos *text nodes* que son "\n\t" (antes/después del párrafo) debido a los enter
- Q: ¿Cuántos hijos tiene el <p>? ¿y el <a>?

Document Object Model
Modificar la estructura DOM

- El modelo de objetos DOM proporciona los métodos necesarios para modificar la estructura de DOM.
  - Algunos métodos...
    - ✓ Los métodos `createElement` y `createTextElement` permite 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.

Document Object Model
Crear nodos html

- 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`.
- 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 de otro nodo hijo.

Document Object Model
Crear nodos html (II)

- Método `createElement`.

  document`.createElement(`*etiquetaHTML*`)`
  - *etiquetaHTML* es una cadena con la etiqueta.
  - Devuelve un nodo de tipo `element` con la etiqueta especificada.
- Método `createTextElement`.

  document`.createTextElement(`*contenido*`)`
  - Devuelve un nodo de tipo `text` con el contenido especificado.
- Método `appendChild`.

  nodoPadre`.appendChild(`*nodoHijo*`)`
  - Hace que *nodoHijo* se coloque como último hijo de *nodoPadre*.
    - ✓ Si *nodoHijo* ya existe, lo elimina de dónde está y lo coloca en la nueva posición.
- Método `insertBefore`.

  nodoPadre`.insertBefore(`*nodoAñadido*`, `*nodoSiguiente*`)`
  - Inserta el *nodoAñadido* como hijo de *nodoPadre* antes del *nodoSiguiente* referenciado.

Document Object Model
Crear nodos html (III)

```
function insertarNodoAlFinal(){
     var elemento= document.createElement("p");
     var texto = document.createTextNode("Insertado al final...");
     elemento.appendChild(texto);
     document.body.appendChild(elemento);
}
funcion moverNodoAlFinal(nodo){
     document.body.appendChild(nodo);
}
function insertarNodoAntes(){
     var elemento= document.createElement("p");
     var texto = document.createTextNode("Insertado antes del último párrafo...");
     elemento.appendChild(texto);
     var nodo = document.getElementById("ultimoParrafo");
     document.body.insertBefore(elemento,nodo);
}
<body>
<p id="primerparrafo" onclick="insertarNodoAlFinal()">Lorem ipsum dolor sit amet,
consectetuer adipiscing elit. Aenean commodoligula eget dolor. Aenean massa. ...</p>

<p onclick="moverNodoAlFinal(this)">Donec pede justo, fringilla vel,... </p>

<p id="ultimoParrafo" onclick="insertarNodoAntes()">Etiam rhoncus. Maecenas
 tempus, tellus eget ... </p>
</body>
```

Document Objecto Model
Eliminación de nodos

- 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.
  - Para asegurarse de quién es *nodoPadre* se puede utilizar la propiedad `parentNode` de *nodo*.

```
function eliminarNodo(){
     nodoViejo = document.getElementById("enlace");
     //No se declara con var para que nodoViejo sea global
     nodoViejo.parentNode.removeChild(nodoViejo);
}
function recuperarEnlace(){
     document.getElementById("ultimoParrafo").appendChild(nodoViejo);
}
<p>Aliquam lorem ante <a href="#" id="enlace" onclick="eliminarNodo()">,dapibus in</a>,
viverraquis, feugiat a, tellus. Phasellus viverra nulla ut metus varius laoreet.Quisque
rutrum. Aenean imperdiet. Etiam ultricies nisi vel augue. Curabiturullamcorper ultricies
nisi. Nam eget <strong id="ultimapalabra" onclick="recuperarEnlace()">dui.</strong></p>
<p id="ultimoParrafo" onclick="insertarNodoAntes()">Etiam rhoncus. Maecenas tempus,
tellus eget condimentum rhoncus, sem quam ... </p>
```

Document Object Model
Reemplazar y copiar nodos

- El método `replaceChild` permite cambiar un nodo por otro.

  nodoPadre`.replaceChild(`*nodoViejo*`, `*nodoNuevo*`)`
  - Cambia *nodoViejo* por *nodoNuevo*.
  - Devuelve *nodoViejo*.
- El método `cloneNode` devuelve una copia de un nodo.

  nodo`.cloneNode(`*copiarHijos*`)`
  - 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.

```
function remplazarÚltimoPorPrimero(){
     var nodoViejo = document.getElementById("ultimoParrafo");
     var nodoNuevo = document.getElementById("primerparrafo").cloneNode(true);
     document.body.replaceChild(nodoNuevo,nodoViejo);
}
```

Document Object Document
Acceso a los atributos de un elemento

- La propiedad `attributes` permite acceder a los atributos de un elemento.

  elemento`.attributes`

  - Devuelve un dato de tipo `namedNodeMap`.
    - ✓ Una colección a la que se puede acceder tanto por el índice de cada elemento, como por el nombre del atributo.
      - DOM no determina el orden de los elementos.
  - Cada elemento de la colección es un nodo de tipo `Attr`.
    - ✓ La propiedad `name` del nodo devuelve el nombre del atributo.
    - ✓ La propiedad `value` del nodo devuelve el valor del atributo.

```
<p id="primerparrafo" class="normal" onclick="enmarcarPárrafo()">Lorem ipsum dolor sit
amet, consectetuer adipiscing elit. <a href="http://www.colimbo.net">Aenean commodo</a>
ligula eget dolor...</p>

var nodo = document.getElementById("primerparrafo");
alert("Número de atributos:" + nodo.attributes.length); //Devuelve 3
alert(nodo.attributes[1].name);                         //Puede devolver id,class,onclick
alert(nodo.attributes["id"].value);                     //Devuelve primerparrafo
```

Document Object Document
Acceso a los atributos de un elemento (II)

- 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 `attibutes`.

```
<style type="text/css">
  .recuadro {padding: 0.5em; background-color:#EBEBEB; border: #d6d6d6 solid 1px;}
</style>

//Sobre el párrafo de la diapositiva anterior
//Cambia el enlace
document.body.getElementsByTagName("a")[0].href = "http://www.upsam.com";

function enmarcarPárrafo(){
    var nodo = document.getElementById("primerparrafo");
    nodo.className="recuadro";
}
```

Document Object Document
Crear y eliminar atributos

- 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.

Document Object Model
Acceder a los estilos

- La propiedad `style` de los elementos permite acceder a los distintos estilos **en línea** de un nodo.
  - Devuelve un objeto de tipo `style` que representa el conjunto de las propiedades de estilos en línea establecidas para el elemento mediante el atributo html `style`.
    - ✓ Como los estilos en línea tienen mayor prioridad que el resto dentro de CSS, permite cambiar el estilo de los elementos.
  - El objeto `style` devuelto permite acceder a todos los atributos de estilo definidos en CSS.

    `nodo``.style.`*`nombreAtributoCSS`*
    - ✓ El nombre de los atributos varía ligeramente respecto a CSS.
      - En los atributos CSS con guiones, los guiones desaparecen.
      - Cuando hay palabras compuestas, las primera va en minúsculas y las siguientes en mayúsculas.
        - Por ejemplo, la propiedad `color` de CSS llamaría igual que la propiedad `color` del objeto `style` de DOM, la propiedad `background-color` de CSS se llamaría `backgroundColor` en DOM o la propiedad `border-top-color` de CSS se llamaría `borderTopColor` en DOM.
      - La especificación de DOM del W3C dar una correspondencia de todas las propiedades CSS con DOM en www.w3.org/TR/DOM-Level-2-Style/css.html#CSS-CSS2Properties.

Document Object Model
Modificar los estilos

- El objeto `style` es de sólo lectura por lo que no se puede modificar directamente.
  - Para modificar un estilo en línea hay que utilizar las propiedades del objeto `style` (los nombre de los atributos del estilo) que son de lectura escritura.

```
var nodo=document.getElementById("ultimoParrafo");
nodo.style.color = "blue"
```

- Si se desea cambiar varios atributos de estilo...
  - Crear una regla para una clase con todos los atributos que se desea modificar.
  - Cambiar el atributo `class` del elemento mediante *nodo*`.className`.

```
<style type="text/css">
  .recuadro {padding: 0.5em; background-color:#EBEBEB; border: #d6d6d6 solid 1px;}
</style>

function enmarcarPárrafo(){
   var nodo = document.getElementById("primerparrafo");
   if(nodo.className=="normal"){
      nodo.className="recuadro"; }
   else{
      nodo.className="normal"; }
}
```