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.

- 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 o crear nuevos elementos.

DOM

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

plenamente implementado por todos los Navegadores.

DOM

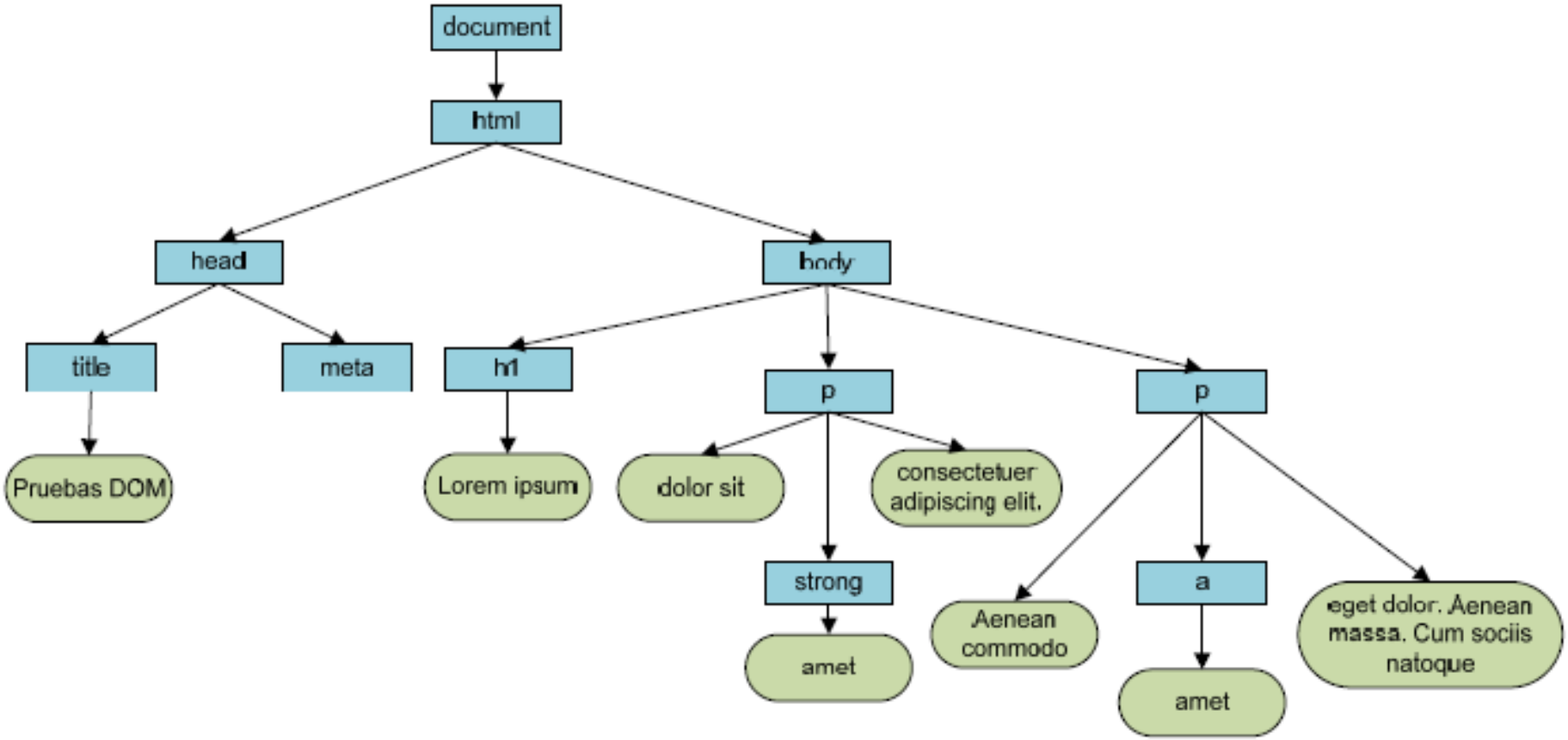
- DOM considera un documento xhtml (bien formado)como un árbol de nodos.
- Por ejemplo:

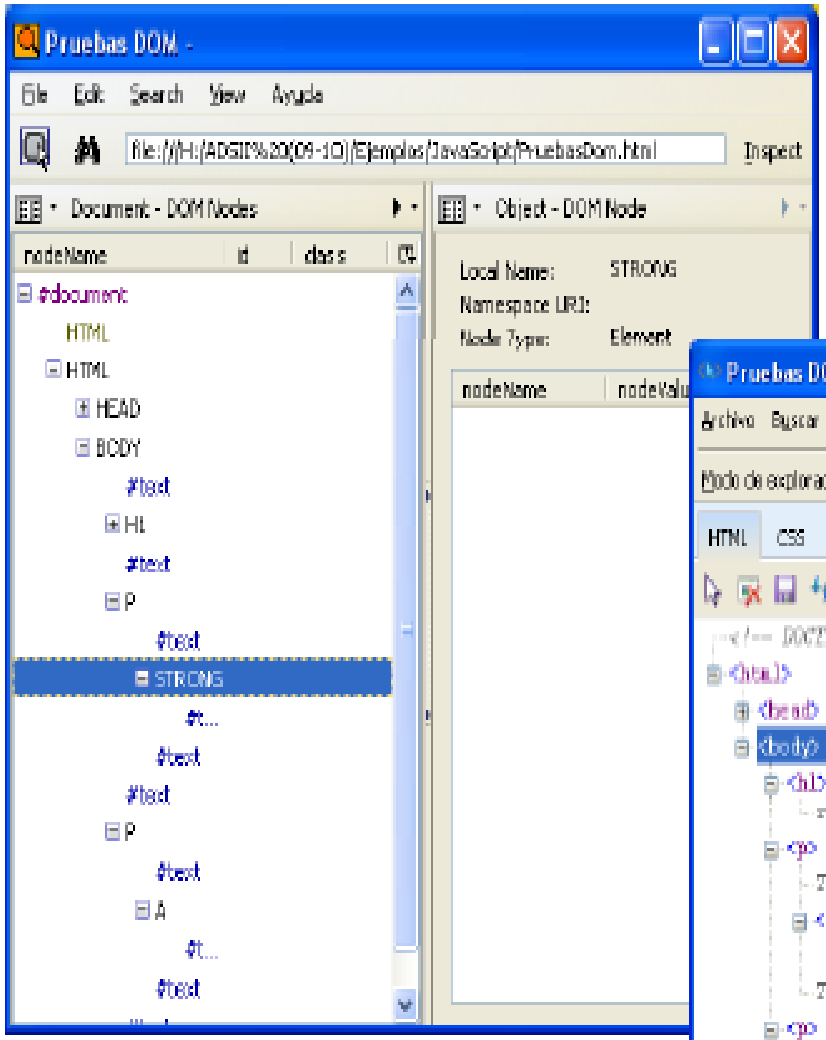
```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
        "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es">
<head>
  <title>Pruebas DOM</title>
  <meta http-equiv="content-type" content="text/html; 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>
```

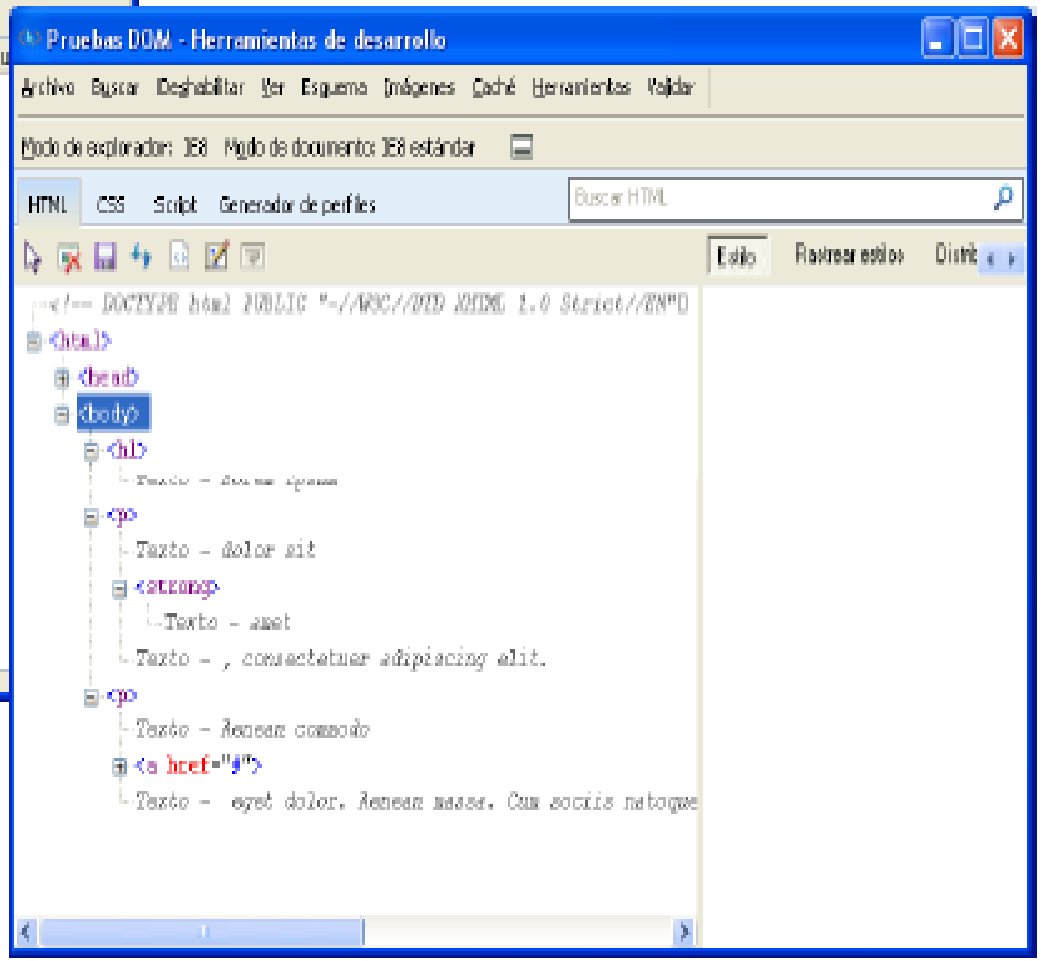
DOM

- Este sería el árbol de nodos resultante...





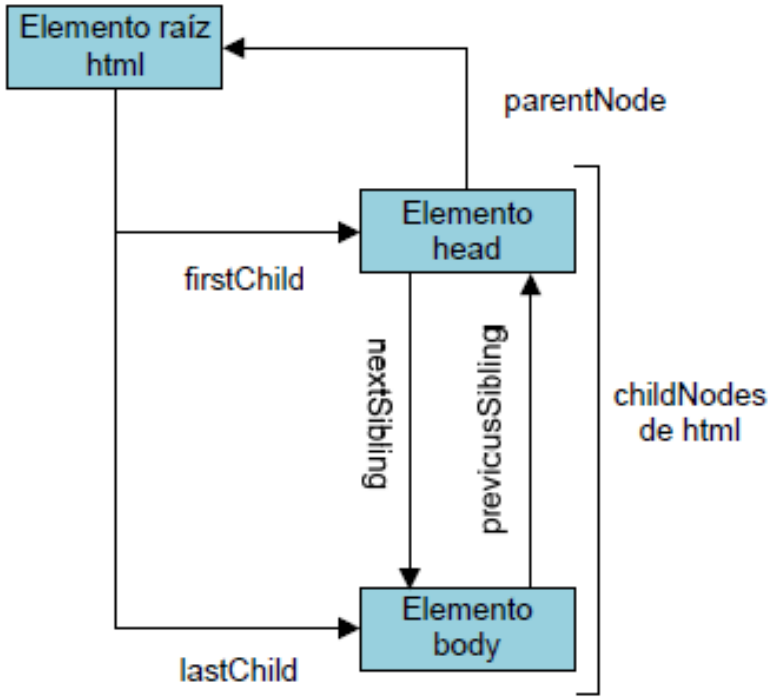
Árbol del documento con DOM Inspector de Web Developer para Firefox



Árbol del documento con las herramientas de desarrollo de Internet Explorer 8 (F12)

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

- 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.
- **Comment.** Representa un comentario.

DOM

- El resto de tipos de nodos son:

DocumentType

CDataSection

DocumentFragment

Entity

EntityReference ProcessingInstruction

Notation.

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**

DOM

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.

DOM

- 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

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.

prueba1

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 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")

párrafos se cargaría con todos los elementos p del documento. ejemplo

DOM

Método getElementoById

- 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.*

ejemplo

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 getElementById.

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.*

añadir

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);

Ejercicio Repaso

Crear una página con la siguiente estructura:

Usar las css adecuadas

1-Cabecera: Ejercicio de Manipulación de Nodos

2-En la parte derecha haremos un cuadro que contenga tres párrafos diferentes.

2-En la parte izquierda pondremos una lista de botones con las acciones a realizar:

Ejercicio (cont)

- Restaurar->restaurar el estado inicial de la caja
- Debajo->Insertar nuevo elemento debajo de todos
- Delante->Insertar nuevo elemento delante del segundo
- Reemplazar->reemplazar el primer elemento de la caja por otro nuevo
- Suprimir->Suprimir el tercer párrafo
- Cambiar->Poner el segundo texto en último lugar
- Copiar->Copia del segundo texto, se inserta al final