DESARROLLO WEB EN ENTORNO CLIENTE

6. UTILIZACIÓN DEL MODELO DE OBJETOS DEL DOCUMENTO (DOM)

UTILIZACIÓN DEL MODELO DE EVENTOS DEL DOCUMENTO (DOM)

EL MODELO DE OBJETOS DEL DOCUMENTO (DOM)

DOM:

- Estándar W3C que define cómo acceder a documentos (HTML, XML...).
- Interfaz de programación de aplicaciones (API) del W3C.
- Mediante scripts se puede acceder y actualizar el contenido dinámicamente.

Tipos:

DOM CORE: modelo para cualquier documento estructurado.

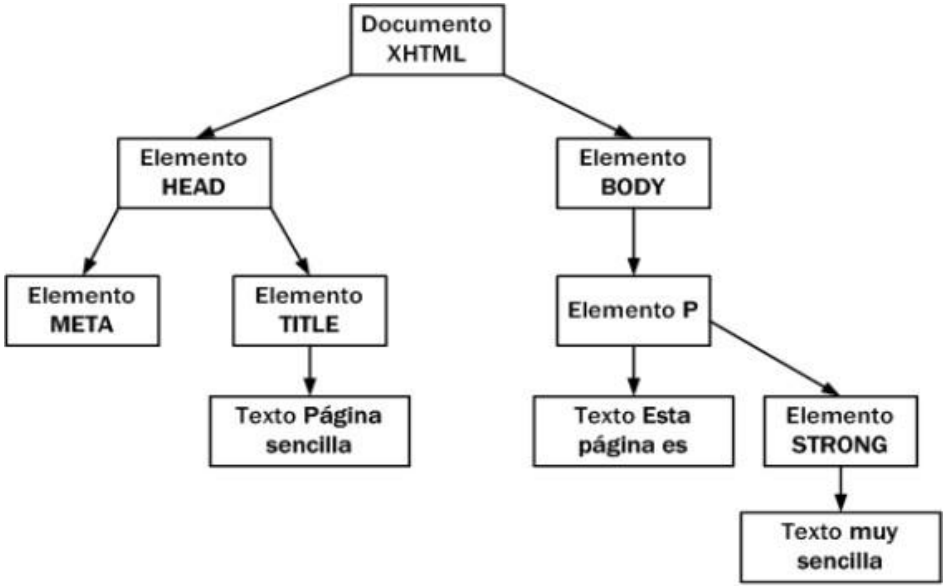
XML DOM: modelo estándar para documentos XML.

HTML DOM: modelo estándar para documentos HTML

UTILIZACIÓN DEL MODELO DE EVENTOS DEL DOCUMENTO (DOM)

EL MODELO DE OBJETOS DEL DOCUMENTO (DOM)

```
<!DOCTYPE html PUBLIC ”
//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1
-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<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>
```



Ordenación de la estructura del árbol:

- En el árbol de nodos el superior es la raíz.
- Cada nodo, salvo la raíz, tiene un padre.
- Un nodo puede tener cualquier número de hijos.
- Una hoja es un nodo sin hijos.
- Nodos con el mismo padre son hermanos.

UTILIZACIÓN DEL MODELO DE EVENTOS DEL DOCUMENTO (DOM)
OBJETOS DEL MODELO. PROPIEDADES Y MÉTODOS

Nodos más importantes

Elemento	Descripción
Document	Es el nodo raíz del documento HTML, todos los documentos del árbol cuelgan de él
DocumentType	Representación del DTD de la página. Un DTD es una definición de tipo de documento. Define la estructura y sintaxis de un documento XML. El DOCTYPE es el encargado de indicar el DocumentType.
Element	Representa el contenido de una pareja de etiquetas de apertura y cierre (<etiqueta>...</etiqueta>). También puede representar una etiqueta abreviada que se cierra a si misma ( ). Este es el único nodo que puede tener tantos nodos hijos como atributos.
Attr	Representa el nombre de un atributo o valor.
Text	Almacena la información que es contenida dentro de un nodo Element.
CDataSection	Representa una secuencia de código de tipo <![CDATA[]]>. Este texto solamente será analizado por un programa de análisis.
Comment	Representa un comentario XML.

Interfaz Node

- Javascript crea un objeto llamado Node para para manipular la interfaz de los nodos.
- Define una serie de constantes que identifican los tipos de nodo (diapositiva siguiente).
- También proporciona propiedades y métodos.

UTILIZACIÓN DEL MODELO DE EVENTOS DEL DOCUMENTO (DOM)

OBJETOS DEL MODELO. PROPIEDADES Y MÉTODOS

Constantes que identifican los tipos de nodo

- 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.PRO
- CESSING_INSTRUCTION_NODE = 7
- Node.COMMENT_NODE = 8
- Node.DOCUMENT_NODE = 9
- Node.DOCUMENT_TYPE_NODE = 10
- Node.DOCUMENT_FRAGMENT_NODE = 11
- Node.NOTATION_NODE = 12

UTILIZACIÓN DEL MODELO DE EVENTOS DEL DOCUMENTO (DOM)
OBJETOS DEL MODELO. PROPIEDADES Y MÉTODOS

Propiedades y métodos

Propiedad/Método	Valor devuelto	Descripción
nodeName	String	Nombre del nodo (no está definido para algunos tipos de nodo)
nodeValue	String	Valor del nodo (no está definido para algunos tipos de nodo). Ej. Para Texto , el nodeValue devolverá "Texto".
nodeType	Number	Una de las 12 constantes definidas anteriormente.
ownerDocument	Document	Referencia al documento al que pertenece el nodo.
firstChild	Node	Primer nodo de la lista de childNodes.
lastChild	Node	Último nodo de la lista de childNodes.
childNodes	NodeList	Lista de todos los hijos del nodo actual.

UTILIZACIÓN DEL MODELO DE EVENTOS DEL DOCUMENTO (DOM)
OBJETOS DEL MODELO. PROPIEDADES Y MÉTODOS

Propiedades y métodos

Propiedad/Método	Valor devuelto	Descripción
previousSibling	Node	Referencia del hermano anterior o null si es el primer hermano.
nextSibling	Node	Referencia del hermano siguienteo null si es el último hermano.
hasChildNodes()	Boolean	Devuelve true si el nodo actual tiene uno o más hijos.
attributes	NamedNodeMap	Con nodos de tipo Element. Contiene objetos de tipo Attr que definen 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(nuevo,anterior)	Node	Reemplaza el nuevo nodo por el nodo anterior.
insertBefore(nuevo, anterior)	Node	Inserta un nuevo nodo antes de la posición del nodo anterior.

UTILIZACIÓN DEL MODELO DE EVENTOS DEL DOCUMENTO (DOM)

ACCESO AL DOCUMENTO DESDE CÓDIGO

Ejemplo de código en HTML:

```
<html>
<head>
<title>TituloDOM</title>
</head>
<body>
<p>ParrafoDOM</p>
<p>ParrafoDOM segundo</p>
<p>ParrafoDOM tres</p>
</body>
</html>
```

Nodo raíz:

```
var obj_html= document.documentElement;
```

Primer y útlimo hijo del nodo:

```
var obj_head= obj_html.firstChild;
var obj_body= obj_html.lastChild;
```

Acceso a nodos desde el índice de un array:

```
var obj_head= obj_html.childNodes[0];
var obj_body= obj_html.childNodes[1];
```

Acceso al número de hijos (longitud del array):

```
var numeroHijos= obj_html.childNodes.length;
```

UTILIZACIÓN DEL MODELO DE EVENTOS DEL DOCUMENTO (DOM)
ACCESO AL DOCUMENTO DESDE CÓDIGO

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!“;

UTILIZACIÓN DEL MODELO DE EVENTOS DEL DOCUMENTO (DOM)
ACCESO AL DOCUMENTO DESDE CÓDIGO

Acceso directo a los nodos:

getElememementsByTagName()

- Recupera los elementos de la página HTML que le hayamos pasado por parámetro.
- var divs= document.getElementsByTagName("div");

getElementsByName()

- Recupera los datos de la página HTML donde "*name*" coincide con el name que le hayamos pasado en la función.
- var divPrimero= document.getElementsByName("primero");

getElementById()

- Recupera el elemento HTML cuyo ID coincida con el nombre pasado en la función. Ejemplo:
- var elemento1= document.getElementById("elemento1");

UTILIZACIÓN DEL MODELO DE EVENTOS DEL DOCUMENTO (DOM)
ACCESO AL DOCUMENTO DESDE CÓDIGO

Acceso a los atributos de un nodo de tipo element:

getNameItem(nomAttr)

- Devuelve el nodo de tipo attr cuya propiedad nodeName contenga el valor nomAttr.

removeNameItem(nomAttr)

- Elimina el nodo de tipo attr en el que la propiedad nodeName coincida con el valor nomAttr.

setNameItem(nodo)

- Añade el nodo attr a la lista de atributos del nodo element. Lo indexa según la propiedad nodeName del atributo.

item(pos)

- Devuelve el nodo correspondiente a la posición indicada por el valor numérico pos.

UTILIZACIÓN DEL MODELO DE EVENTOS DEL DOCUMENTO (DOM)
ACCESO AL DOCUMENTO DESDE CÓDIGO

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).

UTILIZACIÓN DEL MODELO DE EVENTOS DEL DOCUMENTO (DOM)
ACCESO AL DOCUMENTO DESDE CÓDIGO

Creación y eliminación de nodos

createAttribute(nomAtributo)

- Crea un nodo de tipo atributo con el nombre pasado a la función.

createCDataSection(texto)

- Crea una sección de tipo CDATA con un nodo hijo de tipo texto con el valor pasado.

createComment(texto)

- Crea un nodo de tipo comentario con el contenido del texto pasado.

createDocumentFragment():

- Crea un nodo de tipo DocumentFragment.

UTILIZACIÓN DEL MODELO DE EVENTOS DEL DOCUMENTO (DOM)
ACCESO AL DOCUMENTO DESDE CÓDIGO

Creación y eliminación de nodos

createElement(nomEtiqueta)

- Crea un elemento del tipo etiqueta, del tipo del parámetro pasado como nomEtiqueta.

createEntityReference(nomNodo)

- Crea un nodo de tipo EntityReference.

createProcessingInstrution(objetivo.dato)

- Crea un nodo de tipo ProcessingInstruction.

createTextNode(texto)

- Crea un nodo de tipo texto con el valor del parámetro pasado.

UTILIZACIÓN DEL MODELO DE EVENTOS DEL DOCUMENTO (DOM)
ACCESO AL DOCUMENTO DESDE CÓDIGO

Ejemplo de creación de nodos en un documento:

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

let 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>

UTILIZACIÓN DEL MODELO DE EVENTOS DEL DOCUMENTO (DOM)

ACCESO AL DOCUMENTO DESDE CÓDIGO

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>
```

UTILIZACIÓN DEL MODELO DE EVENTOS DEL DOCUMENTO (DOM)

PROGRAMACIÓN DE EVENTOS

Carga de la página HTML:

- Se utiliza el evento onload:

```
<html>
<head><title>TituloDOM</title></head>
<body onload="alert('Página cargada completamente');">
   <p>Primer parrafo</p>
</body>
</html>
```

UTILIZACIÓN DEL MODELO DE EVENTOS DEL DOCUMENTO (DOM)

PROGRAMACIÓN DE EVENTOS

Comprobar que el arbol DOM está cargado:

- Se puede utilizar el método window.onload:

```
<html>
<head><title>TituloDOM</title>
 <script>
  function cargada() {  window.onload= "true";
      if(window.onload){ return true; }
      return false;
  }
  function pulsar(){
    if(cargada()){ alert("Página cargada correctamente");}
  }
</script>
</head>
<body> <p onclick="pulsar();">Primer párrafo</p> </body>
</html>
```

UTILIZACIÓN DEL MODELO DE EVENTOS DEL DOCUMENTO (DOM)
PROGRAMACIÓN DE EVENTOS

Actuar sobre el DOM al desencadenarse eventos:

- Se puede utilizar el método window.onload:

```
<html>
<head><title>TituloDOM</title>
<script>
function ratonEncima(){
  document.getElementsByTa gNa me("div")[0].child Nodes[0].nodeValue    = "EL RATON ESTA ENCIMA";
}
function ratonFuera(){
  document.getElementsByTa gNa me("div")[0].child Nodes[0].nodeValue    = "NO ESTA EL RATON ENCIMA";
}
</script>
</head>
<body>
  <div onmouseover="ratonEncima(); "onmouseout="ratonFuera();">
     VALOR POR DEFECTO
       </div>
     </body>
     </html>
```

UTILIZACIÓN DEL MODELO DE EVENTOS DEL DOCUMENTO (DOM)
DIFERENCIAS EN LAS IMPLEMENTACIONES DEL MODELO

Adaptaciones de código para diferentes navegadores:

- Para mejorar la compatibilidad podemos crear de forma explícita las constantes predefinidas:

  - alert(Node.DOCUMENT_NODE); // Devolvería 9
  - alert(Node.ELEMENT_NODE); // Devolvería 1
  - alert(Node.ATTRIBUTE_NODE); // Devolvería 2

UTILIZACIÓN DEL MODELO DE EVENTOS DEL DOCUMENTO (DOM)

DIFERENCIAS EN LAS IMPLEMENTACIONES DEL MODELO

Adaptaciones de código para diferentes navegadores:

```
if(typeofNode == "undefined") {
  varNode  =  {
  ELEMENT_NODE: 1,
    ATTRIBUTE_NODE: 2,
    TEXT_NODE: 3,
    CDATA_SECTION_NODE: 4,
    ENTITY_REFERENCE_NODE: 5,
    ENTITY_NODE: 6,
      PROCESSING_INSTRUCTION_NO
      DE: 7,
    COMMENT_NODE: 8,
    DOCUMENT_NODE: 9,
    DOCUMENT_TYPE_NODE: 10,
    DOCUMENT_FRAGMENT_NODE:
      11,
    NOTATION_NODE: 12
  };
}
```

UTILIZACIÓN DEL MODELO DE EVENTOS DEL DOCUMENTO (DOM)

DIFERENCIAS EN LAS IMPLEMENTACIONES DEL MODELO

Uso de librerías de terceros

- Cross-browser: permite visualizar una página o aplicación igual en todos los navegadores. Para ello surgen utilidades que permiten unificar eventos y propiedades.
- Renderizar a través de una web: hay páginas que permiten introducir una dirección y elegir la versión del navegador para visualizar (ej. Netrenderer.com, para explorer).
- Programas para renderizar: permiten instalar varias versiones del mismo navegador, pero dan problemas de compatibilidad de versiones con los últimos navegadores.
- Instalar los navegadores en máquinas virtuales: consiste en instalar las versiones de los navegadores en máquinas virtuales acorde con los sistemas operativos para los que hay instalables de la versión del navegador. Hay que asegurarse que el navegador toma una decisión u otra en función de la respuesta.