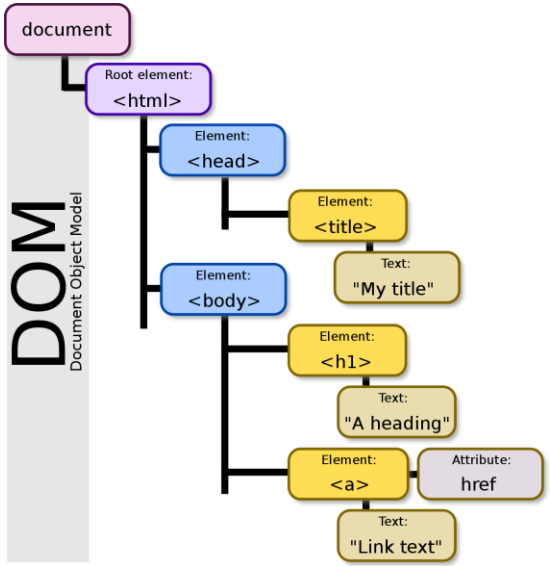


Utilización del modelo de objetos del documento

- DOM -

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.

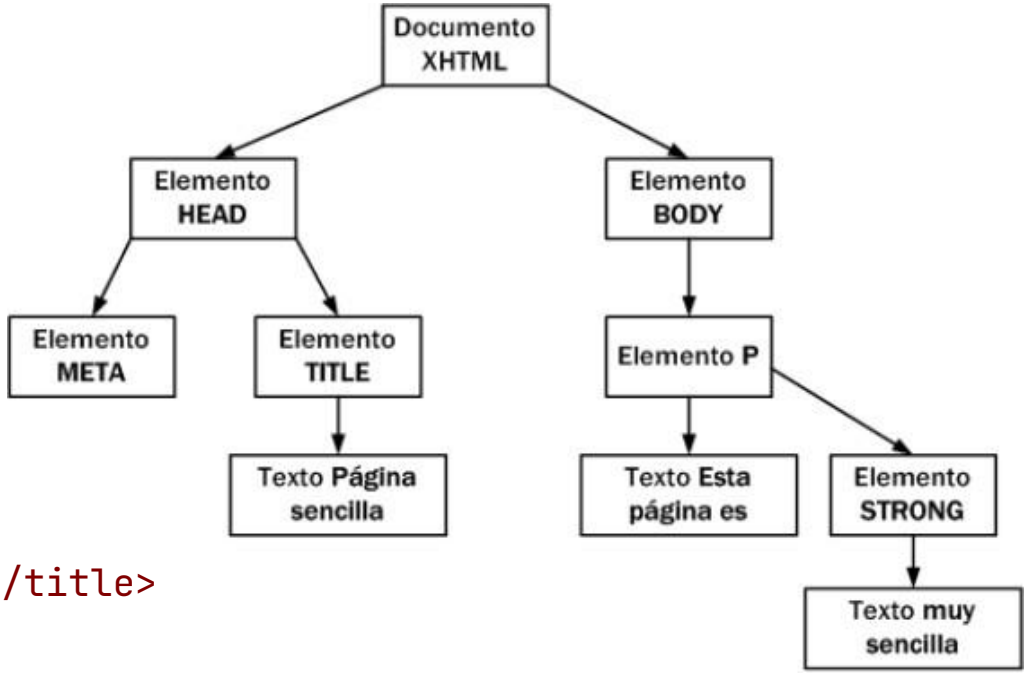
Tipos:

DOM CORE: modelo para cualquier documento estructurado.

XML DOM: modelo estándar para documentos XML.

HTML DOM: modelo estándar para documentos HTML

DOM




```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Texto página sencilla</title>
  </head>
  <body>
    <p>Esta es una <strong>muy</strong> página sencilla</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.

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.

Propiedades y métodos

Interfaz Node

- JavaScript crea un objeto llamado Node 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.

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.PROCESSING_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

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.

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.

Acceso al documento desde código

Ejemplo de código en HTML:

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

```javascript
// Nodo raíz:
let obj_html = document.documentElement;

// Primer y último hijo del nodo:
let obj_head = obj_html.firstChild;
var obj_body = obj_html.lastChild;

// Acceso a nodos desde el índice de un array:
let obj_head = obj_html.childNodes[0];
let obj_body = obj_html.childNodes[1];

// Acceso al número de hijos (longitud del array):
let numeroHijos = obj_html.childNodes.length;
```

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
```

Acceso al texto de un nodo de tipo texto:

Extraer el texto de un nodo:

```
let x = document.getElementById("miNodo").textContent;
```

Cambiar el texto de un nodo:

```
document.getElementById("miNodo").textContent = “¡Párrafo cambiado!";
```

Acceso al documento desde código

Acceso directo a los nodos:

getElememementsByTagName()

- **Recupera los elementos de la página HTML que le hayamos pasado por parámetro.**
- `let 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.**
- `let divPrimero= document.getElementsByName("primero");`

getElementById()

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

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.

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).

Acceso al documento desde código

EJERCICIO: U6T1 - Acceso al DOM desde código

- Dispones del código de una página web en html: municipios.html.
- Introduce en el apartado de script el código necesario para extraer:
  - El número de párrafos de la página.
  - El texto del segundo párrafo.
  - El número de enlaces de la página.
  - La dirección del primer enlace.
  - La dirección del penúltimo enlace.
  - El número de enlaces que apuntan a /wiki/Municipio
  - El número de enlaces del primer párrafo.
- Para agregar texto en la página deberás introducir una etiqueta div con el id=info y añadir en ella toda la información detallada mediante:
  - let info = document.getElementById("info");
  - Info.innerHTML = "Texto informativo";

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.

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.

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>
```

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

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

Resultado

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

Acceso al documento desde código

EJERCICIO – U6T2 – Generando elementos

- Crea una página web que tenga un listado de tipo <ul> con un <li> de muestra.
- Introduce un botón en la página que, cuando lo pulses, te muestre un prompt para que el usuario introduzca un texto.
- Una vez cerrado el prompt el valor se añadirá como un nuevo <li> a la lista creada.
- Añade dos botones más con texto “Borrar primer li” y “Borrar último li” de modo que cuando pulses el primer botón borre el primer elemento de la lista y cuando pulses el último borre el último elemento de la lista.

Acceso al documento desde código

EJERCICIO – U6T3 – Generando formularios dinámicamente

- Crea de manera dinámica (es decir, al cargarse la página) el formulario que definimos en el tema anterior (tarea U4T1).
- Ten en cuenta que el formulario deberá tener los atributos necesarios para que, al crearse, tenga la misma funcionalidad que el que creaste en html.
- No olvides añadir las etiquetas <label> a cada uno de los elementos.

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>
```

Programación de eventos

Comprobar que el árbol 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>
```

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.getElementsByTagName("div")[0].childNodes[0].nodeValue;
      }
      function ratonFuera() {
        document.getElementsByTagName("div")[0].childNodes[0].nodeValue;
      }
    </script>
  </head>
  <body>
    <div onmouseover="ratonEncima();" onmouseout="ratonFuera();">VALOR POR DEFECTO</div>
  </body>
</html>
```

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
```

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_NODE: 7,
    COMMENT_NODE: 8,
    DOCUMENT_NODE: 9,
    DOCUMENT_TYPE_NODE: 10,
    DOCUMENT_FRAGMENT_NODE: 11,
    NOTATION_NODE: 12,
  };
}
```

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.

Diferencias en las implementaciones del modelo

EJERCICIO – U6T4 Generador de formularios:

Desarrolla una aplicación que te permita generar tus propios formularios de manera dinámica. Para ello dibuja una tabla de una sola fila y varias columnas. En cada columna habrá un botón que realice lo siguiente:

- Crear un input de tipo texto. Le preguntará al usuario mediante un prompt qué nombre (atributo name) tiene el input.
- Crear un input de tipo password. Le preguntará al usuario mediante un prompt qué nombre (atributo name) tiene el input.
- Crear un textarea. Le preguntará al usuario el nombre y generará automáticamente un textarea de 40 columnas y 5 filas.
- Crear un label. Preguntará al usuario a qué input va referido (atributo for).
- Crear una imagen. Preguntará al usuario qué ruta tiene la imagen (atributo src).
- Crear un checkbox. Preguntará al usuario el nombre y el valor (atributos name y value).
- Crear un radio. Preguntará al usuario el nombre y el valor (atributos name y value).
- Crear un botón (submit). Preguntará al usuario el nombre y el valor (atributos name y value).

END

prof.jduran@iesalixar.org