CFGS DAW

Mòdul 6: Desenvolupament web en entorn client

INS Joan d'Àustria

Document Object Model

El DOM es una interfície de programación para HTML que representa los documentos de forma jerárquica, como un árbol de nodos que podremos modificar.

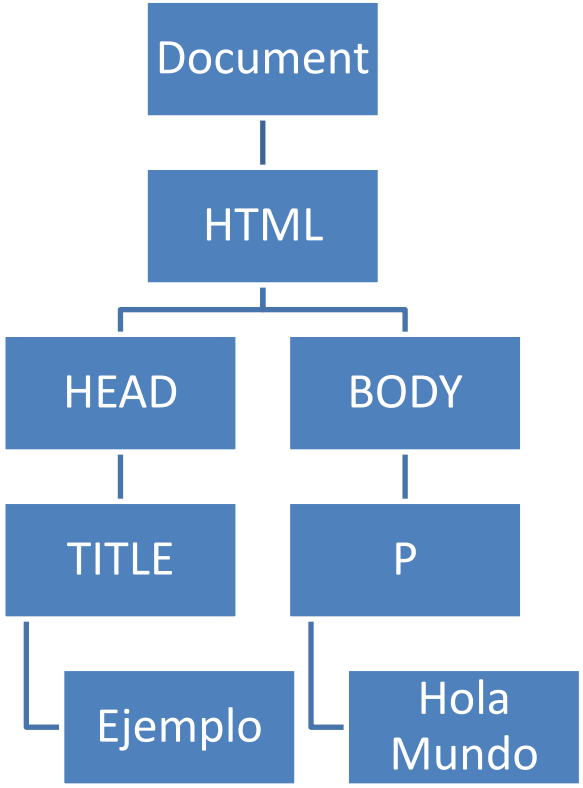
Todos los documentos HTML o XML se pueden representar de esta manera.

Existen diferentes tipos de nodos, cada uno de los cuales presenta diferentes características

Document Object Model

Así por ejemplo, podemos representar el siguiente documento como un árbol jerárquico:

```
<html>
<head>
<title>Ejemplo</title>
</head>
<body>
<p>Hola Mundo</p>
</body>
</html>
```



Document Object Model

Existen varios tipos de nodos, entre los que destacan:

- Document: Es el nodo raíz del que dependen el resto de nodos
- Element: Cada una de las etiquetas HTML se representa en el árbol con un nodo de tipo element. Puede contener atributos y de el pueden depender otros nodos
- Attr: Representan cada uno de los atributos que se pueden añadir a un elemento
- Text: Nodo que contiene el texto visible de una página o un elemento

Document Object Model

Los diferentes nodos tienen relaciones con otros nodos descritas en forma de parentesco (padre, hijo, hermanos).

Así, cada nodo tiene una propiedad childNodes que contiene una lista de nodos. Podemos ver esta lista de nodos como un array

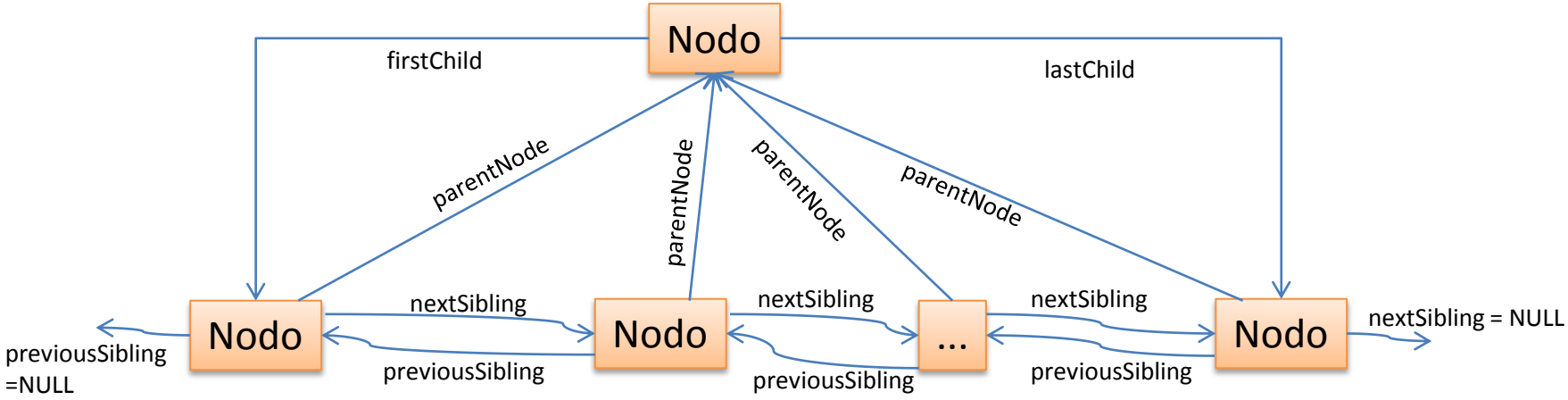
```
var primerHijo = unNodo.childNodes[0];
var primerHijo = unNodo.childNodes.item(0);
var segundoHijo = unNodo.childNodes.item(1);
var totalHijos = unNodo.childNodes.length;
```

Document Object Model

Cada nodo tiene un padre al que podemos acceder con parentNode.

Es posible acceder a los nodos hermanos mediante previousSibling y nextSibling.

Se puede acceder directamente al primer y último hijos mediante firstChild y lastChild



Document Object Model

Se pueden manipular los nodos de la estructura mediante los métodos:

- appendChild: añade un nuevo nodo al final de la lista de nodos hijos y actualiza las relaciones

```
unNodo.appendChild(nuevoNodo);
```

- insertBefore: añade un nuevo nodo como hermano anterior (previousSibling) de otro nodo

```
unNodo.insertBefore(nuevoNodo, unNodo.firstChild);
```

Document Object Model

- replaceChild: reemplaza un nodo por otro

```
unNodo.replaceBefore(nuevoNodo, unNodo.firstChild);
```

- removeChild: elimina un nodo determinado de la jerarquia

```
unNodo.removeChild(unNodo.firstChild);
```

Document Object Model

Para acceder a los nodos debemos partir de la raíz del árbol. Para ello, podemos acceder mediante el objeto document de cualquiera de las formas siguientes:

```
var html = document.documentElement;
```

```
var html  = document.childNodes(0);
```

```
var html = document.firstChild;
```

También disponemos de atajos a algunos elementos:

```
document.body
```

```
document.forms
```

```
document.images
```

```
document.links
```

Document Object Model

Aunque de esta manera podríamos llegar a recorrer todo el árbol y encontrar un nodo determinado, se hace necesario encontrar formas más eficientes y claras de hacerlo.

Así, el DOM nos ofrece dos métodos más claros:

- getElementsByTagName(): Le indicamos el nombre de un elemento y nos retorna una lista con los nodos que sean de este tipo

```
var imagenes= document.getElementsByTagName('img');
alert (imagenes[0].src);
```

Document Object Model

- getElementById(): Le indicamos el identificador de un elemento y nos retornará el elemento con ese id o null si no lo encuentra. El identificador introducido es case sensitive. Si hemos puesto más de un elemento con el mismo id en un documento nos retornará el que aparezca primero en la estructura jerárquica.

```
var imagen= document.getElementById('Myimg');
alert (imagen.src);
```

Document Object Model

- getElementsByName(): Le indicamos el nombre y nos retornará una lista con los nodos que tienen ese nombre. El ejemplo más clásico es para los radio buttons HTML.

Document Object Model

El tipo de **nodo Element** representa cualquier etiqueta HTML y presenta las siguientes características:

- nodeName: la etiqueta HTML
- nodeValue: Null
- parentNode: puede ser otro nodo tipo Element o tipo Document
- Child nodes: pueden ser otros nodos Element, Text, Comment…

Document Object Model

El tipo de **nodo HTMLElement** hereda todas las propiedades del nodo Element y le añade algunas más:

- id: Identificador único del elemento
- title: Información adicional representada mediante un tooltip
- className: Es el equivalente al atributo class de HTML

```
var div = document.getElementById(“myDiv”);
div.id = “OtroId”;
div.className = “otraClase”;
div.title = “Un texto de ayuda”;
```

Document Object Model

Podemos añadir nuevos elementos al árbol de forma dinámica. Los elementos que presentan texto se representan con el **nodo Text**, que tiene las siguientes características:

- nodeName: “#text”
- nodeValue: El texto que muestra
- parentNode: Un nodo tipo Element
- child Nodes: Este nodo no puede tener hijos

Document Object Model

Para acceder al valor de un nodo tipo text, disponemos entre otros de los siguientes métodos:

- appendData(text): añade el texto al final
- deleteData(offset, count): Borra count caracteres desde la posición offset
- insertData(offset, text): inserta el text en la posición offset

Document Object Model

También podemos crear nuevos elementos de texto y luego introducirlos dinámicamente en el documento

```
var element = document.createElement(“div”);
element.className = “texto”;

var textNode = document.createTextNode(“Hello world!”);
element.appendChild(textNode);

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

Document Object Model

Otro tipo de nodo es el **nodo Attr**. Este tipo de nodo presenta las siguientes propiedades:

- nodeName: nombre del atributo
- nodeValue: valor del atributo
- parentNode: Null
- Child Nodes: Este nodo no puede tener hijos

Document Object Model

Para crear nodos, y para consultar o modificar los valores de los atributos utilizaremos los métodos:

- getAttribute
- setAttribute
- removeAttribute
- createAttribute

```
var attr = document.createAttribute(“align”);
attr.value = “left”;
element.setAttributeNode(attr);
```

Document Object Model

El tipo de **nodo HTMLinputElement** es otra herencia del tipo de nodo Element, pero además añade las siguientes propiedades:

- Alt: descripción tipo tooltip
- Checked: booleano si el tipo de elemento es radio o checkbox
- defaultValue
- Disabled: booleano
- readOnly: booleano.
- Value: el valor del elemento

Document Object Model

Algunos ejemplos serian:

```
<div id="texto">Introduce los datos</div>
```

```
var x=document.getElementById("texto");
alert(x.firstChild.nodeValue);
x.firstChild.nodeValue="Otro texto";
```

```
<form>
<input type="text" id="nombre" />
<input type="text" id="apellido" />
<input type="button" onclick="comprovar();" />
</form>
```

```
var nombre=document.getElementById("nombre");
var apellido=document.getElementById("apellido");
 alert(nombre.value);
```