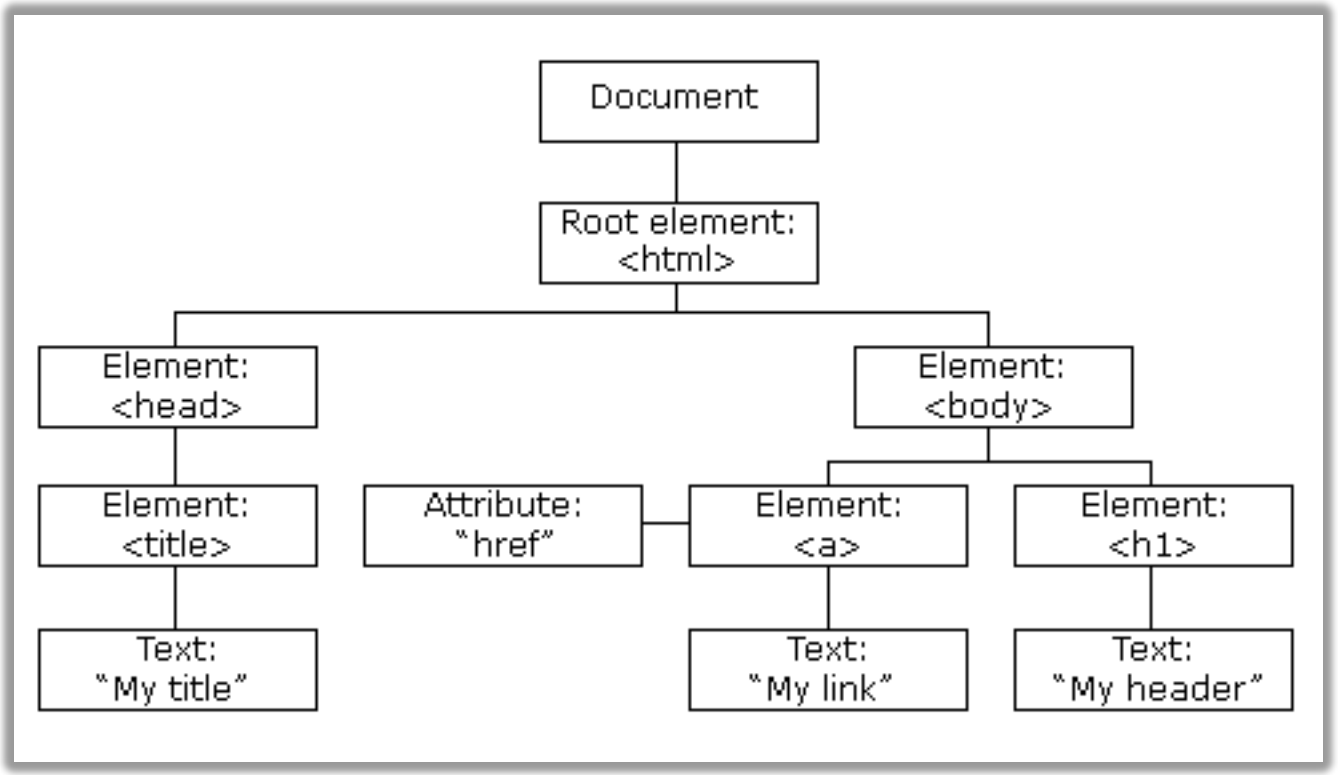
JS: W3C DOM

Objetos, Propiedades y Métodos

Antonio Espín Herranz

HTML DOM (1)

- El DOM (**Document Object Model**) de HTML define un conjunto estándar de objetos para HTML, y una manera estándar de acceder y manipular documentos HTML
- El DOM interpreta un documento HTML como una estructura en árbol (jerarquía de nodos), con elementos, atributos y texto

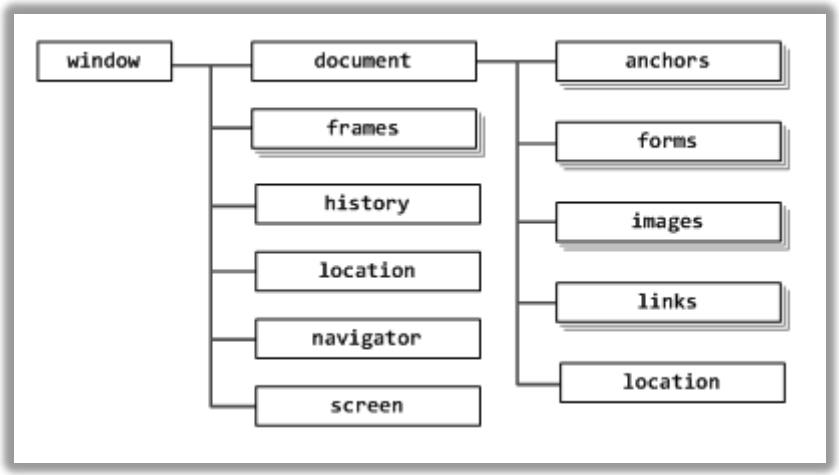


HTML DOM (2)

- Todos los elementos HTML se pueden acceder con el DOM
  - **Se pueden añadir, modificar y eliminar elementos HTML**
- DOM es un API independiente de la plataforma y lenguaje empleado
  - Puede ser usado por cualquier lenguaje de programación
  - Mantenido por el W3C
  - Adoptado por todos los navegadores
    - Problema de compatibilidad: Microsoft implementa una extensión al DOM en Internet Explorer

Objetos de HTML DOM

- **Anchor**
- **Document**
- **Event**
- **Form** y **Form Input** (Button, Checkbox, File, Hidden, Password, Radio, Reset, Submit, Text)
- **Frame**, **Frameset** e **IFrame**
- **Image**
- **Location***
- **Navigator***
- **Option** y **Select**
- **Screen***



- **Table**, **TableHeader**, **TableRow** y **TableData**
- **Window***

- ***Objetos del *runtime engine* de JavaScript, no forman parte del DOM sino del BOM (Browser Object Model)**
- **Referencia: http://www.w3schools.com/htmldom/**

Document Object: escribir texto en la salida

```
<html>
<body>
<script type="text/javascript">
    document.write("Hola mundo!")
</script>
</body>
</html>
```

Document Object: getElementById()

```
<html>
<head>
  <script type="text/javascript">
    function getElement() {
        var x=document.getElementById("cabecera");
        alert("I un elemento" + x.tagName);
    }
  </script>
</head>
<body>
  <h1 id="cabecera" onclick="getElement()">
        Click para saber qué elemento soy!
  </h1>
</body>
</html>
```

Document Object: getElementsByName()

```
<html>
<head>
  <script type="text/javascript">
    function getElements() {
         var x=document.getElementsByName("myInput");
         alert(x.length);
    }
  </script>
</head>

<body>
    <input id="a1" name="myInput" type="text" size="20" /><br />
    <input id="a2" name="myInput" type="text" size="20" /><br />
    <input id="a3" name="myInput" type="text" size="20" /><br />

    <input type="button" onclick="getElements()"
           value="¿Cuántos elementos 'myInput'?" />
</body>
</html>
```

Document Object: devolver el *innerHTML* del primer *anchor* del documento

```
<html>
<body>

    <a name="first">Primer anchor</a><br />
    <a name="second">Segundo anchor</a><br />
    <a name="third">Tercer anchor</a><br />

    "InnerHTML" del primer "anchor" del documento:
    <script type="text/javascript">
        document.write(document.anchors[0].innerHTML);
    </script>

</body>
</html>
```

Document Object: acceso a un ítem de una colección

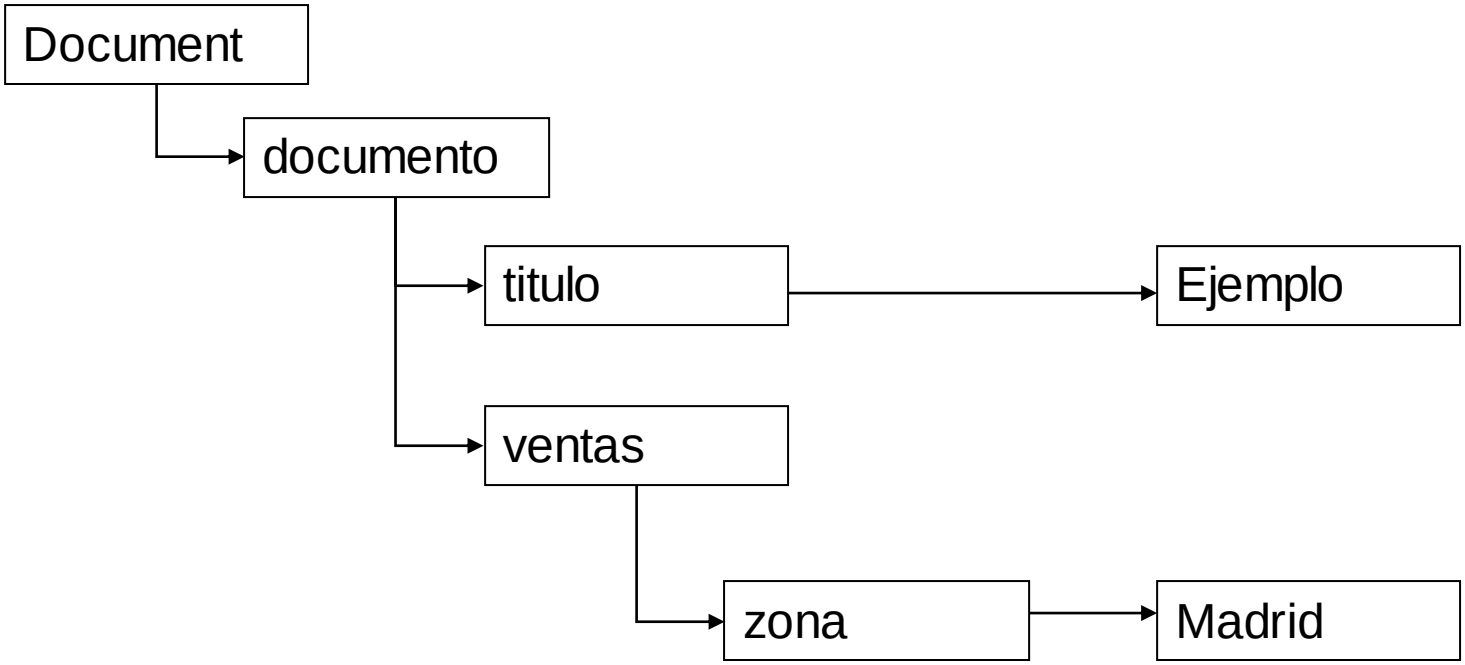
```
<html>
<body>
    <form id="form1" name="Form1">
        Ciudad: <input type="text">
    </form>
    <form id="form2" name="Form2">
        pais: <input type="text">
     </form>

    <p>Para acceder a un ítem de una colección se puede
    usar tanto el número como el nombre del ítem</p>

    <script type="text/javascript">
      document.write("<p>Nombre del primer form: " +
                  document.forms[0].name + "</p>");
      document.write("<p>Nombre del primer form: " +
                  document.getElementById("form1").name + </p>");
    </script>
</body>
</html>
```

El DOM de XML

- DOM representa el XML como árbol formado por nodos.
- Representa la relación entre elementos, atributos, …
- Se utilizará un árbol para representar la estructura, tal y como se definió en el XML.

```
<?xml version="1.0"?>
<documento id="3456">
    <titulo>Ejemplo</titulo>
    <ventas periodo="1">
        <zona>Madrid</zona>
    </ventas>
</documento>
```

Representación en Memoria



OJO. En la representación en memoria se hace una distinción entre los nodos etiqueta y los de texto. Los atributos también estarán a parte.

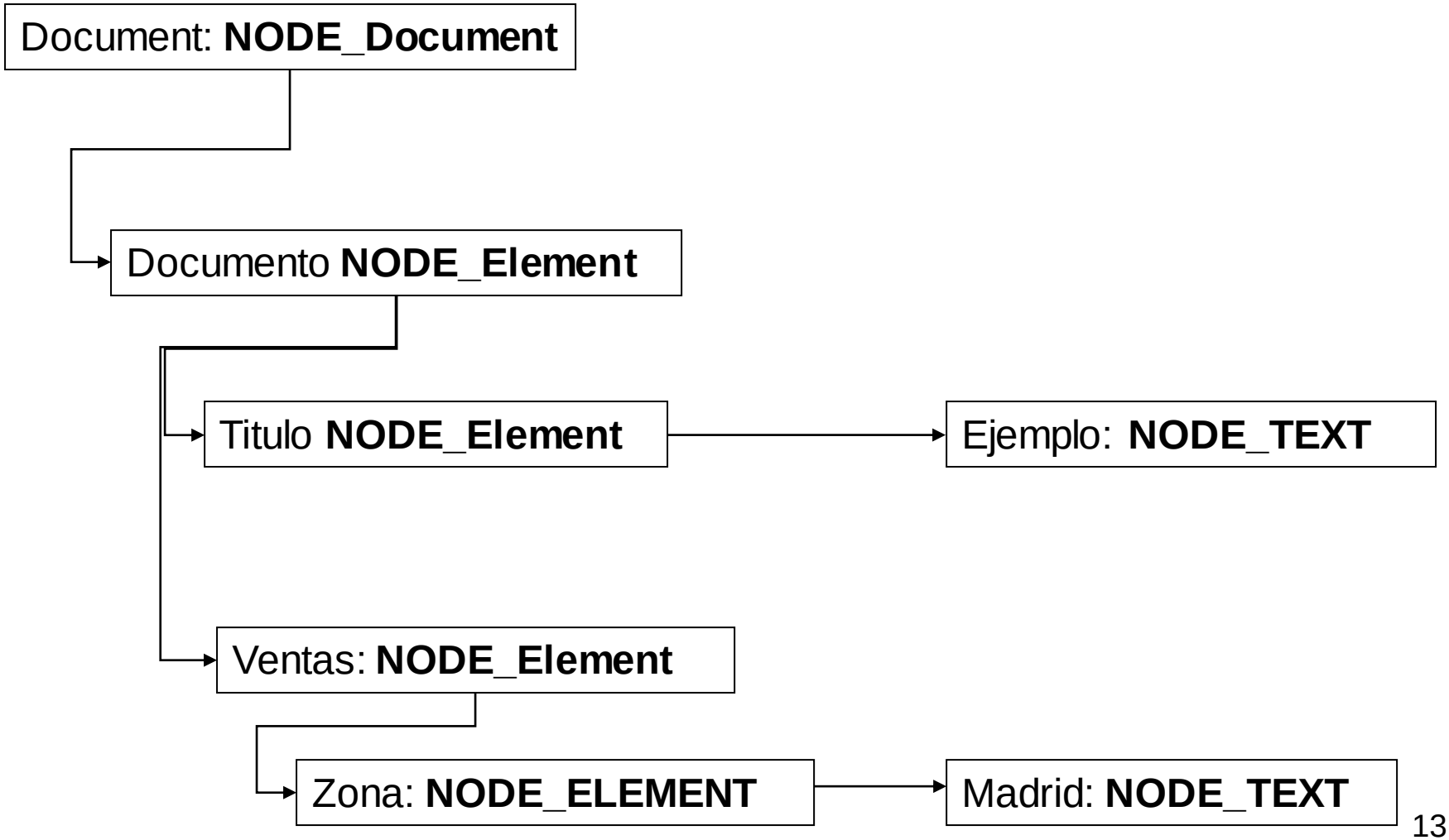
Esto ocurre tanto en un fichero XML como en un HTML / XHTML.

Tipos de Nodos

Cada tipo de nodo tiene asociado un número (una constante) ver valores NodeType.

- **Element** – un elemento XML
- **Attribute** – un attributo
- **Text** – texto contenido en un elemento o atributo
- CDATAsection – sección CDATA
- EntityReference – Referencia a una entidad
- Entity – Indicación de una entidad XML
- ProcessingInstruction – Una instrucción de procesamiento
- Comment – Contenido de un comentario de XML
- **Document** – El objeto documento
- DocumentType – Referencia al elemento DOCTYPE
- DocumentFragment – Referencia a fragmento de documento
- Notation – Contenedor de una anotación

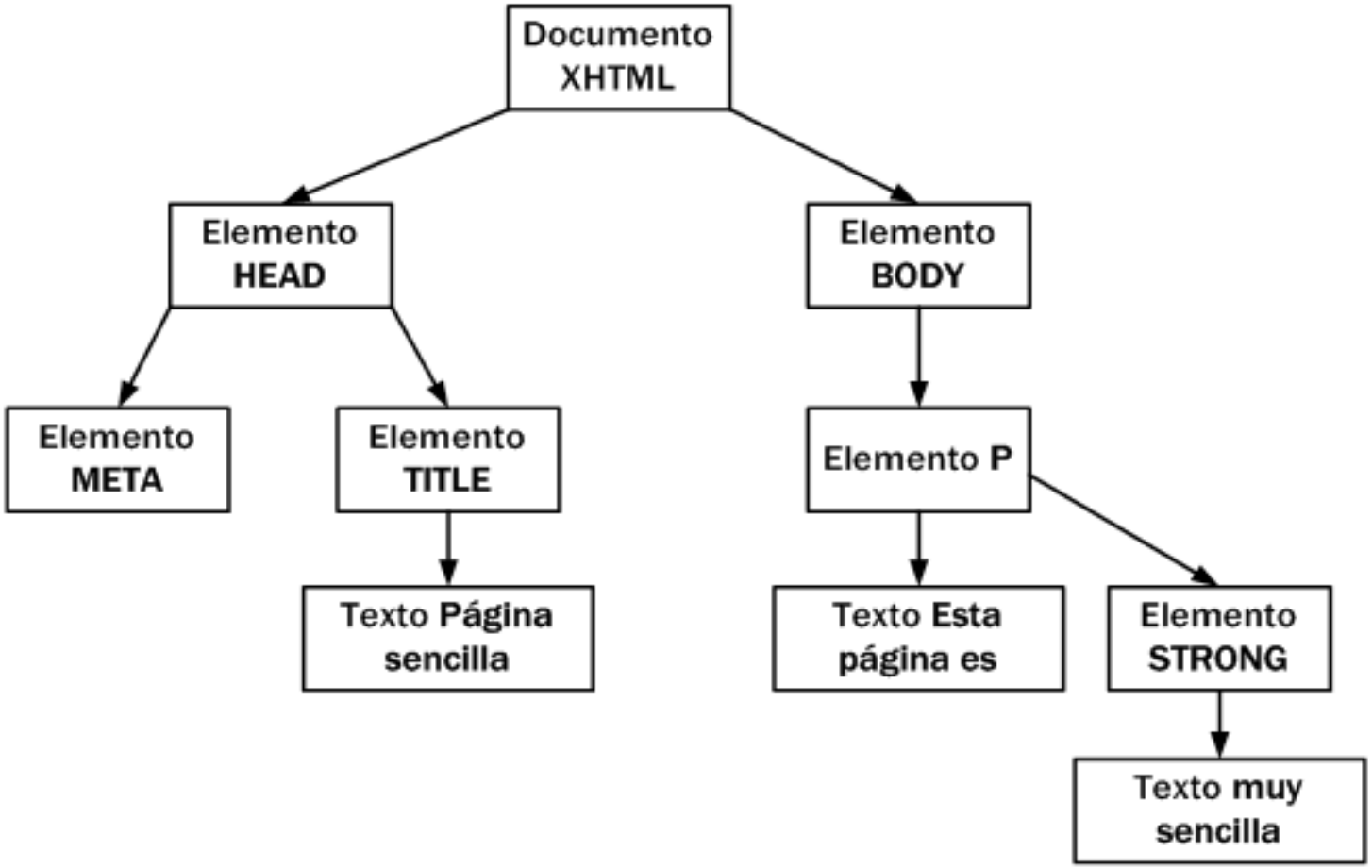
Tipos de nodos



Otro Ejemplo – La página

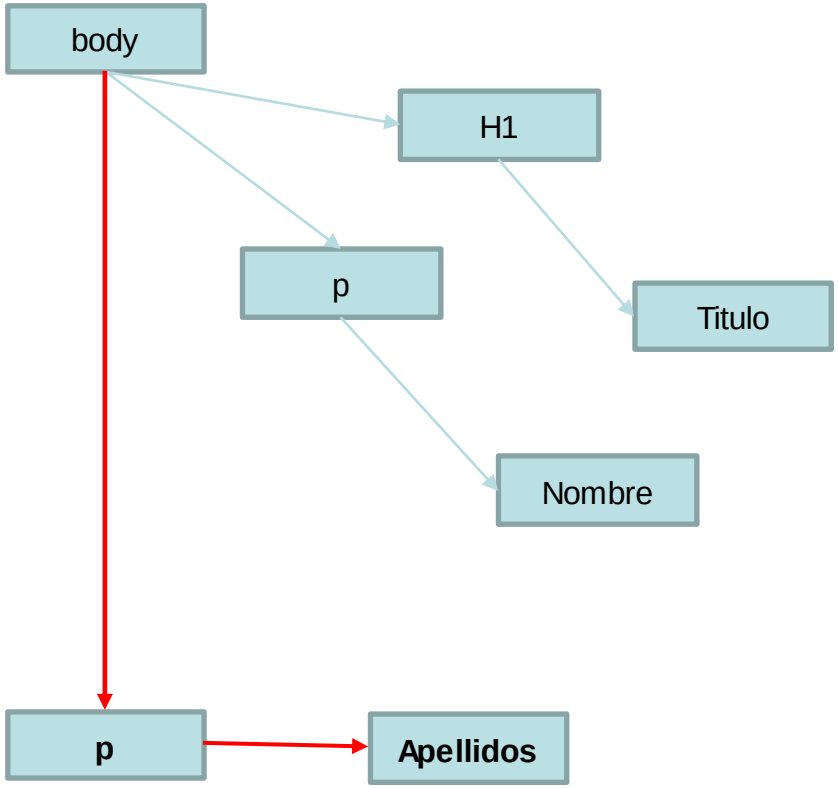
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-
    1" />
<title>Página sencilla</title>
</head>
<body>
<p>Esta página es <strong>muy sencilla</strong></p>
</body>
</html>
```

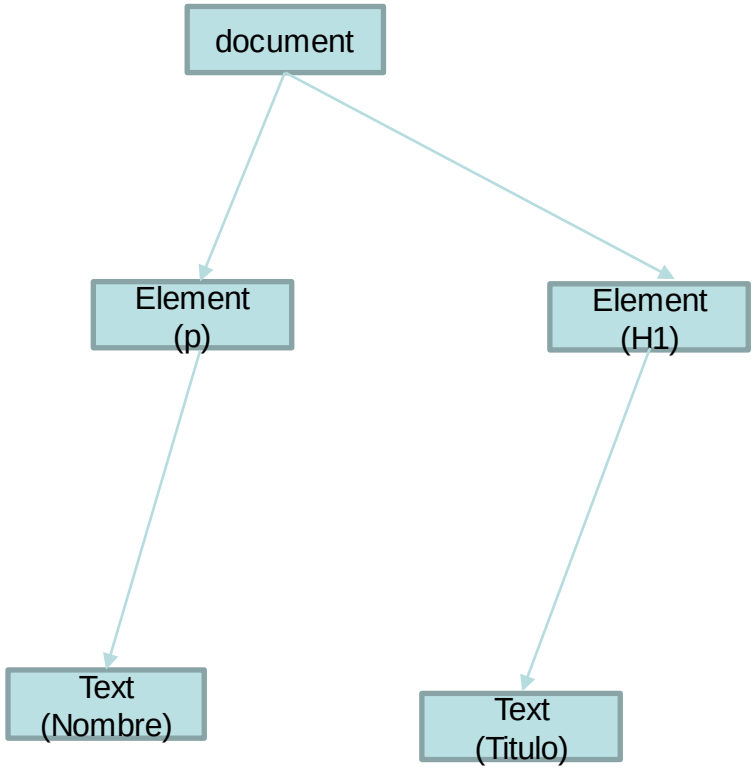
DOM EN MEMORIA



```
<body>
  <h1 class="rojo" id="principal">Titulo</h1>
  <p>Nombre</p>
  <p>Apellidos</p>
</body>
```

Otro ejemplo





Uso del DOM

- Los objetos mas utilizados:
  - document: Representa el propio documento. Y representa el DOM.
  - Window: El propio Navegador.
  - Element: permite acceder y manipular los elementos XML / HTML de un documento.

Node

- Representa cada uno de los nodos del árbol.

- Principales propiedades y métodos:
  - nodeName: El nombre del nodo.
  - nodeValue: El valor del nodo.
  - nodeType: El tipo del nodo. (La lista de antes).

Valores de NodeType

Valor	Descripción
1	Nodo elemento
2	Nodo atributo
3	Nodo de texto
4	Nodo de sección CDATA
5	Nodo de referencia a entidad
6	Nodo entidad
7	Nodo de instrucción de proceso
8	Nodo comentario
9	Nodo documento
10	Nodo tipo de documento
11	Nodo fragmento de documento
12	Nodo anotación

Node

- Mas propiedades de Node:
  - className: El nombre de la clase CSS del nodo.
  - parentNode: El nodo padre.
  - ownerDocument: El documento propietario del nodo con el que estamos trabajando.
  - **childNodes**: La lista de los hijos del Nodo. **Nodelist**.
  - firstChild: Retorna el primer hijo del nodo.
  - lastChild: Retorna el último hijo.
  - previousSibling: Retorna el nodo anterior pero en el mismo nivel que el nodo actual (el hermano).
  - nextSibling: Retorna el nodo posterior pero en el mismo nivel que el actual (el hermano).
  - **attributes**: retorna una lista (**NamedNodeMap**) con los atributos asociados.
  - tagName: El nombre del tag de un nodo.

Node

- insertBefore(newChild, refChild): inserta el primer nodo recibido como argumento antes del indicado en el 2º argumento.
- replaceChild(newChild, oldChild): reemplaza el nodo indicado en el 2º argumento por el especificado en el primero.
- removeChild(oldChild): elimina el nodo indicado.
- appendChild(newChild): inserta el nodo especificado tras el nodo actual.
- hasChildNodes(): indica si el nodo actual tiene o no nodos hijos.

```
var foo = document.getElementById(“foo”);
if (foo.hasChildNodes()){
    foo.removeChild(foo.childNodes[0]);
}
```

NodeList

- Nos permite trabajar con listas de nodos.

- Propiedades y métodos:
  - length: número de elementos de la lista.
  - item(index): para acceder a una posición concreta dentro de la lista.

```
// parg referencia a un objeto de tipo element <p>
if (parg.hasChildNodes()){
    // Children es de tipo NodeList:
    var children = parg.childNodes;
    for (var i=0 ; i < children.length ; i++){
      // Hacer algo con cada hijo children[i];
    }
}
```

NamedNodeMap

- Representa una colección de nodos que pueden ser accesibles por su nombre.

- Propiedades y métodos:
  - getNamedItem(name): obtiene el nodo según el nombre especificado.
  - setNamedItem(name): añade un nodo a la lista.
  - removeNamedItem(name): elimina el nodo especificado de la lista y lo devuelve.
  - item(index): permite acceder a un nodo en concreto.
  - length: nos dice el número de elementos de la colección.

Document

- En el DOM, el objeto Document ofrece el mecanismo para almacenar en memoria documentos HTML, XHTML y XML.

- El objeto **document** está contenido en el objeto **window**.
  - window.document → No es necesario, ocurre igual con alert().

Métodos de document

- document.**createAttribute**(name): Crea un nuevo nodo del tipo attribute y lo retorna.

- Ejemplo:

```
// Localizamos un nodo:
var node = document.getElementById("div1");

// Creamos el atributo:
var a = document.createAttribute("my_att");

// Damos valor al atributo:
a.nodeValue = "newVal";

// Asignamos el atributo al nodo:
node.setAttributeNode(a);

// Comprobamos que el nodo tiene asignado el atributo:
alert(node.getAttribute("my_att"); // "newVal"
```

Métodos de document

- document.**createElement**(name): Crea un nuevo elemento con el nombre dado.

- Ejemplo:

```
// Creamos una capa nueva:
newDiv = document.createElement("div");

// La asignamos contenido:
newDiv.innerHTML = "<h1>Hola mundo!</h1>";

// Se añade el nuevo elemento al árbol:
my_div = document.getElementById("org_div1");
document.body.insertBefore(newDiv, my_div);
```

Métodos de document

- document.**createTextNode**(data): Crea un nodo de tipo Texto con los datos específicos como argumento.

- Ejemplo:

```
function addTextNode(){
    var newText = document.createTextNode("Texto añadido al DOM");
    var para = document.getElementById("p1");
    para.appendChild(newText);
}

<!– En la página tendríamos algo así: - - >
<p id = "p1">Primera línea</p>
<button onclick = "addTextNode()">Añade Texto</button>
```

Métodos de document

- document.**evaluate**(xpathExpresion, contextNode, nameSpaceResolver, resultType, result):
  - Tendríamos que conocer el lenguaje XPATH.
  - Este método devuelve el resultado (los nodos seleccionados) de evaluar una exp. De XPATH.

- document.**getElementById**(id): Devuelve una referencia a un nodo de tipo elemento, realizando una búsqueda según el att Id.

Métodos de document

- document.**getElementsByName**(name): Retorna una lista de elementos (NodeList) que tienen el atributo fijado al valor buscado.

- document.**getElementsByTagName**(name) Retorna una lista de elementos (NodeList) según el nombre de los propios elementos (tags o etiquetas).

  var todosLosParrafos = document.getElementByTagName(“p”);

Métodos de document

- document.**querySelector**()
- document.**querySelectorAll**()

  – El método querySelector() devuelve el primer elemento que coincide con un selector CSS.

  – Para devolver todas las coincidencias (no solo la primera), use en su lugar querySelectorAll().

  – Ambos querySelector() y querySelectorAll() pueden lanzar una excepción SYNTAX_ERR si los selectores no son válidos.

Ejemplo

```
<p>This is a p element.</p>
<p>This is a p element.</p>

<script>
    document.querySelector("p").style.backgroundColor = "red";
</script>
```

- Añade el color rojo de fondo en el primer párrafo.

Ejemplo 2

- Obtener el primer párrafo de la clase example:

```
<p class="example">I am a paragraph.</p>
<p class="example">I am a paragraph.</p>

<script>
    document.querySelector(".example").style.backgroundColor = "red";
</script>
```

Ejemplo 3

```
<p id="demo">This is a p element with id="demo".</p>

<script>
        document.querySelector("#demo").innerHTML = "Hello World!";
</script>
```

Ejemplo 4

- `document.querySelector("h3, h4").style.backgroundColor = "red"`

Ejemplo 5

```html
<div id="prueba">
   <span id="id5" class="clase" title="Azul"></span>
   <span id="id4" class="clase" title="Verde"></span>
   <span id="id3" class="clase" title="Naranja"></span>
   <span id="id2" class="clase" title="Lila"></span>
   <span id="id1" class="clase" title="Rojo"></span>
</div>
```

```javascript
document.getElementById('id1').title // Rojo

document.querySelector('#prueba .clase').title // Azul

document.querySelector('#prueba #id3.clase').title // Naranja

document.querySelector('#prueba .clase + .clase').title // Verde

document.querySelector('#prueba .clase[title^=L]').title // Lila
```

querySelectorAll

- querySelectorAll() devuelve:
  - todos los elementos que coinciden con un selector CSS.
  - devuelve una NodeList .
  - El método lanza una excepción SYNTAX_ERR si los selectores no son válidos.

Ejemplos

- const nodeList= document.querySelectorAll("p");
  nodeList[0].style.backgroundColor = "red";

- const nodeList = document.querySelectorAll("p.example");
  nodeList[0].style.backgroundColor = "red";

- let numb = document.querySelectorAll(".example").length;

- const nodeList = document.querySelectorAll("h3, div, span");

- Establece el color de fondo de cada elemento <p> donde el padre es un elemento :
  - const nodeList = document.querySelectorAll("div > p");

Ejemplos 2

```html
<div id="prueba">
   <span id="id5" class="clase" title="Azul"></span>
   <span id="id4" class="clase" title="Verde"></span>
   <span id="id3" class="clase" title="Naranja"></span>
   <span id="id2" class="clase" title="Lila"></span>
   <span id="id1" class="clase" title="Rojo"></span>
</div>
```

```javascript
// Los 5 <span>
document.querySelectorAll('#prueba .clase')

// Todos los <span> de una página
document.querySelectorAll('span')

// Todos los <span> y <img> de una página
document.querySelectorAll('span, img')

// Todos los <span> hijos de <div>
document.querySelectorAll('div > span')
```

Búsqueda por nombre / att

- También con querySelector y querySelectorAll se puede buscar por el nombre del control y el valor de un atributo.
- Por ejemplo para buscar cual es el radio button seleccionado de un grupo:
  - document.querySelector('[name="your_name"]:checked').value;
  - Si no hay ninguno marcado, devuelve null.

Métodos de document

- document.**open**(): permite abrir el documento para escribir en él.
- document.**close**(): cierra el documento abierto.
- document.**write**("marca"): Escribe en el documento.
- document.**writeln**("marca"): Escribe con un salto de línea.
- Ejemplo:

```
document.open();
document.write("<h1>Nuevo contenido</h1>");
document.close();
```

Propiedades de document

- **characterSet**: Devuelve la codificación del documento.
- **contentType**: Devuelve el content type de la cabecera. El tipo del contenido de la página.
- **docType**: Retorna el DTD del documento.
- **documentElement**: Retorna el elemento que es hijo directo del nodo de tipo documento; para archivos HTML, suele ser el elemento <html>
- **forms**: Retorna la lista (NodeList) de elementos <form> del documento.

Propiedades de document

- **height / width**: Alto / Ancho del documento.
- **images**: Retorna la lista de imágenes del documento.
- **lastModified**: Fecha de modificación del documento.
- **links**: La lista de todos los enlaces que han sido incluidos en el documento.
- **URL**: Retorna la url del documento.
- **styleSheets**: la lista de hojas asociadas al documento.
- **title**: El título del documento.

Objeto element

- Permite acceder y manipular los elementos XML / XHTML de un documento.

- **Principales propiedades**:
  - **attributes**: Retorna un objeto NamedNodeMap con la lista de atributos asociados al elemento.
  - **childNodes**: Retorna el NodeList con todos los hijos del elemento.
  - **className**: Obtener / fijar el valor del atributo class del elemento.
  - **clientHeight**: Retorna un numero que especifica la altura del elemento. Introducida por IE y es soportada por Mozilla.
  - **clientWidth** / **clientLeft** / **clientTop**: Similares a las anterior para el ancho, separación por la izquierda y por arriba.

Objeto element: mas propiedades

- **firstChild**: Permite acceder al primer hijo, si no tiene será null.
- **lastChild**: idem del anterior pero con el último hijo.
- **id**: Fija u obtiene el valor del atributo id del elemento.
- **innerHTML**: Permite añadir el contenido HTML / XHTML a un elemento. Por ejemplo a una capa. Se permite incluir Tags de HTML.

Objeto element: mas propiedades

- **nextSibling**: Retorna un objeto de tipo Node (null si no tiene hermanos) al mismo nivel.
- **nodeName**: Devuelve el nombre del nodo en una cadena de caracteres.
- **nodeType**: Retorna un número que indica el tipo del nodo. Volver a ver la tabla de tipos de nodo.
- **nodeValue**: El valor del nodo. Para los nodos de tipo elemento es null. Para los de texto obtendremos respuesta.
- **parentNode**: Retorna un objeto Node que representa el nodo padre del actual, null en caso contrario.

Objeto element: mas propiedades

- **style**: Retorna un objeto con las declaraciones de estilo (CSS) asociadas al elemento.

```
var div = document.getElementsById("div1");
div.style.marginTop = "25px";
```

- **textContent**: Obtiene o fija el contenido de texto del elemento.

document.getElementById("**divA**").textContent = "***Algo de texto***";

// Se genera:

Algo de texto

Objeto element: Métodos

- **appendChild**(appendedNode): Inserta un nodo como último hijo del elemento.

```
var p = document.createElement("p");
// Se añade al final del Body:
document.body.appendChild(p);
```

- **cloneNode**(deep): Clona un nodo y opcionalmente sus contenidos.

```
p = document.getElementById("para1");
p_prime = p.cloneNode(true);
```

- **getAttribute**(name): Devuelve el valor del atributo especificado dentro del elemento.

- **getAttributeNode**(name): Devuelven un nodo de tipo Attr con la información del atributo.

```
var t = document.getElementById("top");
var idAttr = t.getAttributeNode("id");
alert(idAttr.value);
```

Objeto element: Métodos

- **getElementsByTagName**(name): Retorna la lista de elementos hijos del actual que tengan el nombre especificado. Similar a document.getElementsByTagName.

- **hasAttribute**(name): Permite conocer si el elemento posee el atributo cuyo nombre se especifica como argumento. Retorna un boolean.

Objeto element: Métodos

- **hasAttributes**(): Retorna un valor de verdad, indicando si el elemento tienen o no atributos.
- **hasChildNodes**(): Retorna un valor de verdad, indicando si el elemento tiene o no nodos hijos.
- **insertBefore**(insertedNode, referenceElement): Inserta el nodo indicando en el primer argumento tras el especificado en el segundo argumento.
- **removeAttribute**(name): Elimina el atributo especificado por su nombre del elemento actual.

Objeto element: Métodos

- **removeChild**(removedNode):Permite eliminar un nodo del hijo actual.

- **replaceChild**(insertedNode, replacedNode) Reemplaza el segundo nodo por el primero indicado en la lista de argumentos.

- **setAttribute**(name, value): Añade o modifica el atributo indicado en name, fijándole el valor especificado en value.

```
var d = document.getElementById(“d1”);
d.setAttribute(“align”, “center”);
```

CSS desde el DOM

- Es posible manipular los estilos de los elementos desde el árbol DOM, ofreciendo grandes posibilidades de dinamización.

- Para ello tenemos que utilizar la propiedad style y la subpropiedad que nos interese especificadas en W3C.

- Ejemplos:

```
elt.style.color = “blue”;  // Uso directo.

var estilo = elt.style;
estilo.color = “blue”;  // Uso indirecto.

// Indicando la propiedad style directamente:
c = document.getElementById(“tid”);
c.style = “padding-right: 20px”;
```