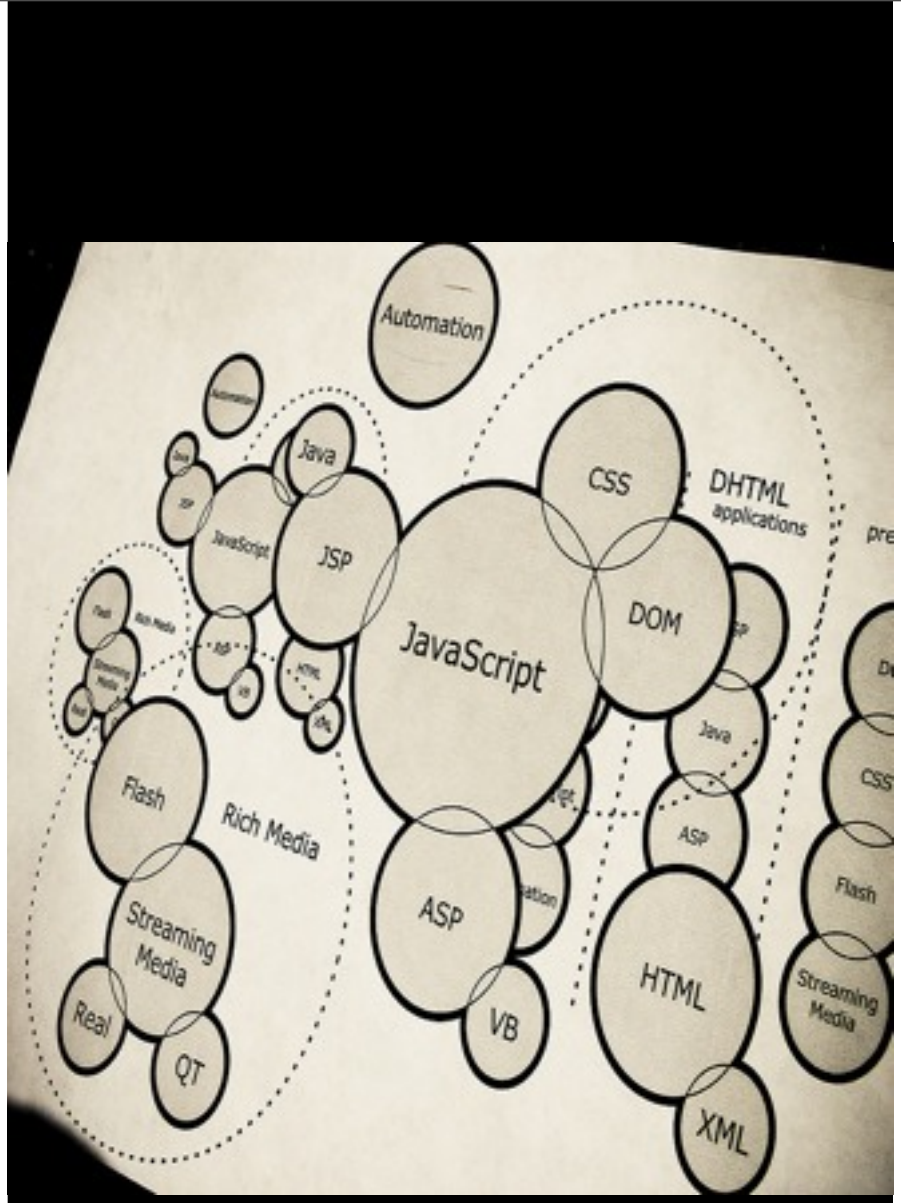


http://www.flickr.com/photos/jm3/362074444/

Tema 3, parte 4

Javascript: el DOM (Modelo de Objetos del Documento)

Javascript parte 4

El DOM

1. El árbol DOM

Conceptos básicos sobre el API DOM y la forma de representar las etiquetas HTML en dicho API

DOM (Document Object Model)

- API orientado a objetos que permite interactuar con el documento HTML
  - Cambiar/leer contenido y estructura
  - Cambiar/leer estilos CSS
  - Gestionar eventos con *listeners* de una forma mucho más sofisticada que con *handlers*

- Niveles (versiones)
  - 0: impuesto por Netscape y Microsoft a principios de la "guerra de navegadores" (finales de los 90)(Explorer 4, Netscape 4). En realidad no existe un "DOM 0 estándar" como tal, es una forma de hablar.
  - 1 (a partir de aquí, del W3C): contenido dinámico
  - 2: estilos dinámicos, eventos
  - 3: "serialización" de XML (permite p.ej. guardar documentos), eventos de teclado,...

DOM 1

- API que permite **acceso/cambio de contenido del documento**, por ejemplo, se puede
  - Insertar nuevas etiquetas en el documento (p.ej. crear un botón nuevo o una fila nueva en una tabla)
  - Leer/cambiar el contenido de cualquier etiqueta (p.ej. de un párrafo <p>)
  - Reordenar los componentes del documento (p.ej. reordenar las filas de una tabla)
- Está dividido en módulos, por ejemplo
  - **Núcleo** ("Core": sirve para cualquier lenguaje de marcado (XML, HTML, …)
  - **HTML**: objetos, propiedades y métodos que facilitan el trabajo con HTML
  - **Range**: manejar fragmentos de documentos

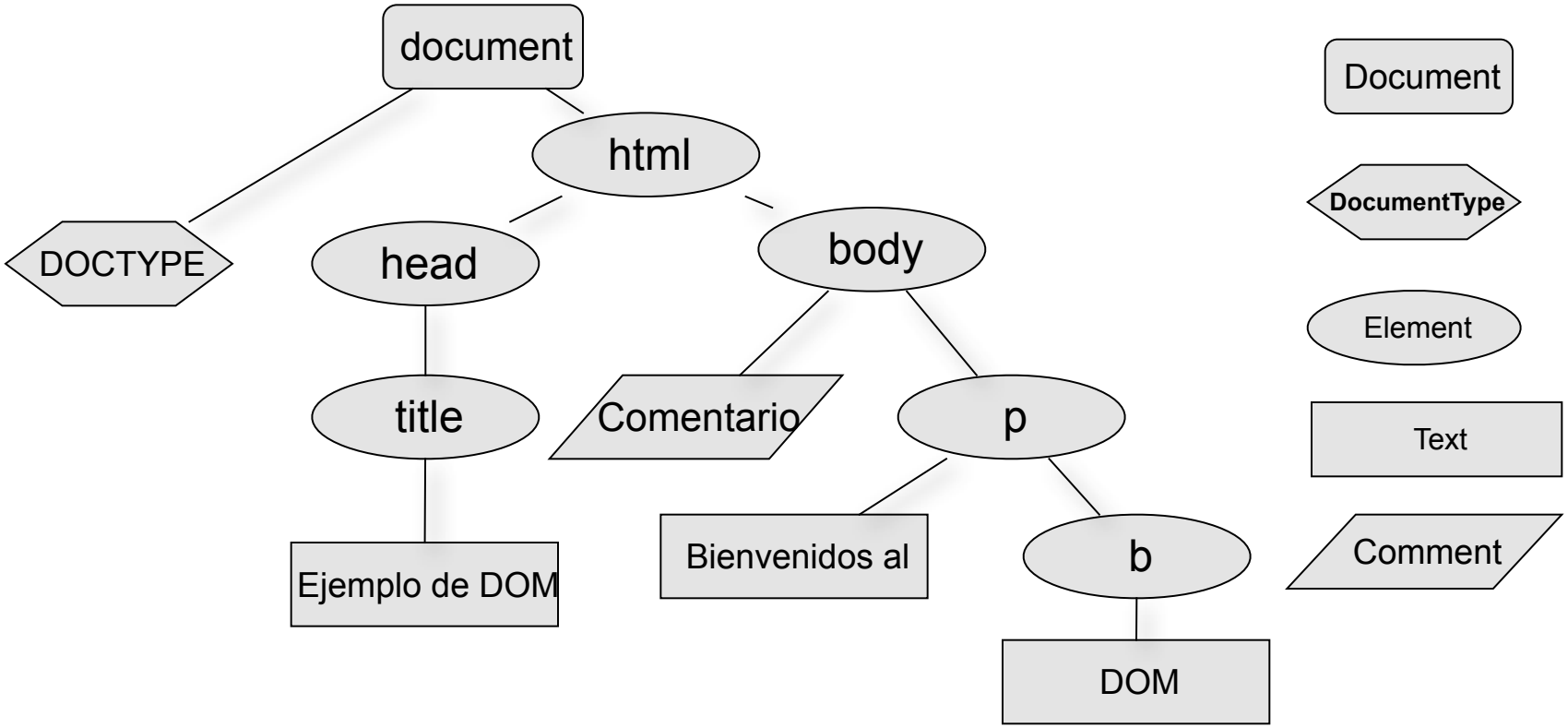
Documentos en DOM

- En DOM los documentos no se tienen representados como "texto plano"
  - El API no funciona como innerHTML

```
<!DOCTYPE html>
<html>
<head>
<title>Ejemplo de DOM</title>
</head>
<body>
<!-- es un ejemplo un poco simple -->
<p style="color:red">Bienvenidos al <b>DOM</b></p>
</body>
</html>
```

El árbol del documento

- Sus nodos reflejan el contenido y la estructura del documento



- El estándar exige nodos de texto "en blanco" adicionales (*whitespace nodes*) donde haya espacios en blanco, retornos de carro, etc. entre etiquetas. Explorer "pre-IE9" no los usa

Los nodos del árbol

- Para manipular el documento lo que tenemos que hacer es manipular los nodos
  - Por ejemplo, para mover un párrafo de sitio, hay que cortar la rama que lo une al sitio actual e insertar el nodo en otro lado
- Todos los nodos son del "tipo" Node, pero hay distintos "subtipos": Document, DocumentType, Element, Text, Comment, …
- Aunque los atributos de las etiquetas son nodos de tipo Attr, "no están" en el árbol, hay que acceder a ellos con métodos del nodo que los posee.

Javascript parte 4

El DOM

2. Obtener información del documento

Acceso a los nodos.
Obtener información de los nodos del documento

Obtener información de un nodo

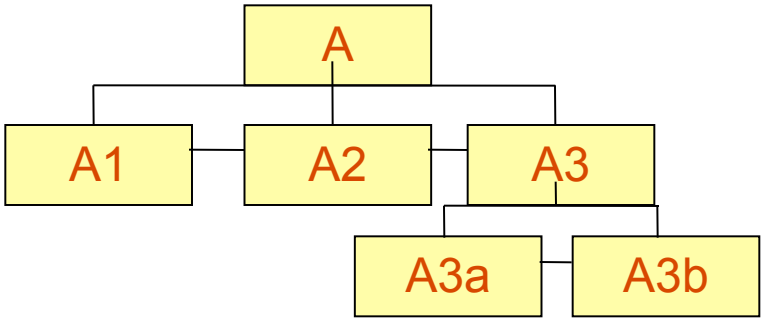
- Una vez obtenida la referencia a un nodo (p.ej. con document.getElementById()) podemos obtener sus propiedades.
- Algunas props. de Node (cualquier nodo)
  - nodeType: cte. entera que representa el tipo de nodo
  - nodeName: nombre, nodeValue: valor. Dependen del tipo de nodo

Tipo de nodo	nodeType	nodeName	nodeValue
Etiqueta	1 (Node.ELEMENT_NODE)	Nombre de la etiqueta sin los “<>” y en máyúsc.	null
Texto	3 (Node.TEXT_NODE)	#text	Texto del nodo
Comentario	8 (Node.COMMENT_NODE)	#comment	Texto del comentario
DOCTYPE	10(Node.DOCUMENT_TYPE_NODE)	Nombre de la etiq. raíz del DOCTYPE	null
Documento	9 (Node.DOCUMENT_NODE)	#document	null

NOTA: en Explorer, las ctes. Node.XXX_NODE no están predefinidas, hay que utilizar el valor numérico

Cómo acceder a un nodo desde otro

- Cada nodo tiene una serie de propiedades que reflejan el "parentesco" con otros, algunas de las cuales son
  - childNodes: array con los nodos hijos
  - firstChild: primer nodo hijo, lastChild: último nodo hijo
  - parentNode: nodo padre
  - nextSibling: siguiente hermano (nodo al mismo nivel) prevSibling: hermano anterior.



A.firstChild = A1
A.lastChild =A3
A.childNodes.length = 3
A.childNodes[0] = A1
A.childNodes[1] = A2
A.lastChild.firstChild = A3a
A3b.parentNode.parentNode = A

A1.nextSibling = A2
A3.prevSibling = A2
A3.nextSibling = null

Cómo acceder a un nodo desde otro (II)

- Esta forma de acceso es problemática por
  - Dependencia de la estructura del árbol. Si cambia, acabaremos en otro nodo o generaremos un error
  - Incompatibilidades entre navegadores: como se ha visto, en el estándar se interpretan los espacios en blanco entre etiquetas como nodos de texto
- No obstante, es necesaria
  - Para recorrer de manera sistemática todo el árbol
  - Para acceder a ciertos nodos. Por ejemplo, los nodos de texto no son accesibles con el método "directo".

Acceso directo a todos los nodos del mismo tipo

- Mediante el método **document.getElementsByTagName (nombreEtiqueta)**, obtenemos todas las etiquetas del mismo tipo.

```
//Cambia el color de todos los párrafos a rojo
var nodos = document.getElementsByTagName("P");
for (var i = 0; i < nodeList.length; i++)
    //la propiedad style representa el estilo CSS, con subpropiedades
    //que son nombres de propiedades CSS
    nodos[i].style.color = "red";
```

- Si nombreEtiqueta="*", entonces accedemos a todas las etiquetas HTML del documento
- Si lo llama una etiqueta, obtenemos solo sus subetiquetas

```
var tabla1 = document.getElementById("tabla1");
var filasDeTabla1 = tabla.getElementsByTagName("tr");
```

Javascript, parte 4

El DOM

3. Cambiar el contenido y la estructura del documento

Cambiar los datos de un nodo

- **Cambiar el valor:** cambiar la propiedad nodeValue
- **Cambiar un atributo:** setAttribute(nombre,nuevoValor)
- Otras muchas propiedades son **solo de lectura** (nodeName, firstChild, parentNode,…) para cambiarlas hay que hacerlo de modo indirecto recurriendo a otros métodos.

```
<p id="p" align="left">Estoy alineado a la izquierda</p>
<script language="JavaScript">
  var p = document.getElementById("p");
  p.setAttribute("align","right");
  p.firstChild.nodeValue="ahora estoy alineado a la derecha";
</script>
```

Crear nuevos nodos

- Distintos métodos del objeto predefinido document, según el tipo de nodo a crear
  - document.createElement(nombre): crea nodo etiqueta. Se le pasa el nombre de la etiqueta sin <>.
  - document.createTextNode(texto): crea nodo de texto, con el contenido especificado
- Hay que insertar los nodos creados en el lugar apropiado del árbol

```
<body id="cuerpo">
<script language="JavaScript">
  var par = document.createElement("p");
  var texto = document.createTextNode("Yo antes no existía!”);
  par.appendChild(texto);
  document.getElementById("cuerpo").appendChild(par);
</script>
</body>
```

Insertar/eliminar nodos

- Métodos de la clase Node, los llama el que va a ser "padre" del nodo a insertar / el padre del que se va a eliminar
- Insertar nodos
  - appendChild(nuevoHijo): Añade el hijo al final de todos los hijos actuales
  - insertBefore(nuevoHijo, hijoReferencia). Inserta el nuevo hijo justo antes del "hijo de referencia"
  - setAttribute(nuevoAtributo, nuevoValor). Si el atributo no existía, lo crea. Como ya se ha visto, si existía cambia su valor
- Eliminar nodos
  - removeChild(hijoABorrar): un nodo deja de ser hijo
  - replaceChild(nuevoHijo, hijoAntiguo): reemplaza un hijo por otro nuevo

Javascript, parte 4

El DOM

4. DOM HTML

Una extensión del API para documentos HTML

DOM 1 HTML

- Facilita el trabajo con documentos HTML, haciendo más directas algunas operaciones
  - Da compatibilidad con el DOM 0, definiendo objetos como los arrays forms, images, links, etc.
  - Por cada atributo HTML hay una propiedad javascript equivalente (como ya hemos visto muchas veces)

```
//Reducir el tamaño de todas las imágenes a la mitad, con DOM core
imags = document.getElementsByTagName("img");
for(i=0; i<imags.length; i++) {
      imags[i].setAttribute("width",imags[i].getAttribute("width")/2);
      imags[i].setAttribute("height",imags[i].getAttribute("height")/2);
}
//idem con DOM HTML
for(var i=0; i<document.images.length; i++) {
       document.images[i].width /= 2;
       document.images[i].height /= 2;
}
```

Trabajo con tablas: algunos métodos

- rows: propiedad de un nodo tabla que contiene todas sus filas
- cells: propiedad de un nodo fila que contiene todas sus celdas
- Insertar y borrar filas: los llama un nodo tabla
  - insertRow(pos): insertar nueva fila vacía (nodo tr) en la posición pos. Comienzan por 0.
  - deleteRow(pos): borrar la fila nº pos

- Insertar y borrar celdas: los llama un nodo fila
  - insertCell(pos): insertar nueva celda vacía (nodo td) en la posición pos. Comienzan por 0
  - deleteCell(pos): borrar la celda nº pos.

Acceso a los campos de formulario

- Array predefinido **document.forms**
- Por posición
  - Cada formulario tiene un array "elements" con los campos
  - Cada campo tiene un atributo "value" (campos de texto, textarea,...) o bien uno booleano "checked" en casillas de verificación o botones de radio

```
//Valor del primer campo del primer formulario
document.forms[0].elements[0].value
```

- Por nombre (name).
  - Automáticamente se define una propiedad con ese name. A los formularios también se les puede poner name

```
//Valor del campo login del formulario
document.miFormu.login.value
```

```
<form name="miFormu" action="log
    Login: <input type="text" name="login"> <br>
    ....
```

Algunas referencias

- Libros electrónicos accesibles solo **desde dentro de la UA**, en Safari O'Reilly (http://proquestcombo.safaribooksonline.com)
- **JavaScript: The Definitive Guide**, Sixth Edition, *David Flanagan*
  - *Referencia exhaustiva de Javascript, incluyendo DOM*
- **Pro Javascript Techniques**, *John Resig*
  - *DOM en Cap 5*
- **Javascript Cookbook**, *Shelley Powers*
  - *Poca teoría, básicamente ejemplos de código*
  - *DOM en Caps 11 y 12*