Introducció al DOM

Vicent Moncho Mas
Guillem Garcia Brustenga

PID_00191130

UOC
Universitat Oberta de Catalunya

Índex

1. Naixement i evolució del DOM

Els models d'objectes defineixen la interfície que descriu l'estructura lògica d'un objecte i la forma estàndard que permet manipular els objectes des del llenguatge de programació. En el cas particular de JavaScript, se sol fer referència a dos models:

- **BOM** (*browser object model*): defineix l'accés a les característiques del navegador.
- **DOM** (*document object model*): defineix l'accés al contingut del document que es mostra a la finestra del navegador, de manera que conté tots els objectes que formen part de la pàgina HTML.

BOM

Aquest model defineix característiques com la finestra, la pantalla i l'historial.

No és senzill de delimitar la frontera entre els dos models anteriors, ja que aspectes que teòricament formen part del BOM o del DOM apareixen entremesclats i en dificulten la identificació. Això implica que generalment es fa referència al DOM, en el cual s'inclouen els objectes relacionats amb el navegador.

Per tant, es pot definir el DOM com una **organització jeràrquica**, en què un objecte depèn de l'objecte que hi ha a sobre i regeix els que hi ha en nivells inferiors. La comunicació entre objectes depèn d'aquesta jerarquia: l'objecte que és en el nivell superior s'ha de comunicar amb els del nivell inferior mitjançant els que hi ha en els nivells intermedis.

Al llarg de la història de JavaScript, han aparegut fins a quatre models diferents d'objectes:

- Model tradicional d'objectes (Netscape 2 i Internet Explorer 3).
- Model tradicional estès (Netscape 3).
- Model d'objectes d'HTML dinàmic (Netscape 4 i Internet Explorer 4).
- Model d'objectes de navegador tradicional + DOM estàndard (a partir de Nestcape 6 i Internet Explorer 5).

L'objectiu d'aquest primer apartat del mòdul és la revisió històrica de l'evolució del DOM en cadascun dels navegadors principals, ja que aquesta visió ens ajudarà a entendre més bé el DOM actual i a veure com pot evolucionar. No es pretén estudiar cada objecte ni les propietats i els mètodes que tenen.

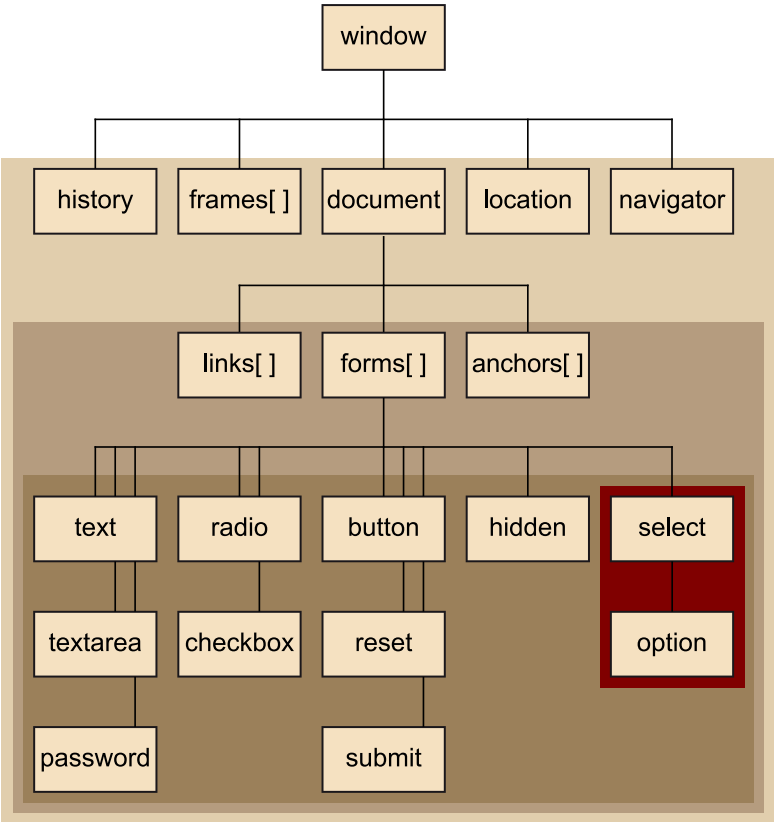
Vegeu també

L'estudi de cada objecte i de les propietats i els mètodes que tenen es plantejarà en l'apartat 3 d'aquest mòdul.

1.1. El model tradicional

Com és sabut, un dels primers objectius del llenguatge JavaScript era validar els continguts dels formularis. Si es té en compte això, era d'esperar que el primer model d'objectes es focalitzés en l'estructura dels formularis tenint característiques bàsiques del navegador i del document. Això va fer que el primer model d'objectes fos molt semblant en totes dues plataformes, Netscape 2 i Internet Explorer 3. La figura següent presenta aquest primer model:

Figura 1. Model d'objectes tradicional (Netscape 2 i Internet Explorer 3)



En el punt més alt de la jerarquia hi ha l'objecte Window, que representa l'àrea de la finestra del navegador en què apareixen els documents HTML. El nivell següent en la jerarquia és format per aquests objectes:

- Document: és l'objecte que conté els elements HTML i el text que hi ha en els elements anteriors.

- Frames[ ]: és una array que conté els marcs de la pàgina, de manera que cada marc, al seu torn, fa referència a un objecte Window.

- History: és un objecte que conté la col·lecció d'adreces URL que ha visitat l'usuari.

- Location: és un objecte que conté l'URL actual del navegador.

- Navigator: és un objecte que conté les característiques bàsiques del navegador (versió, tipus, etc.).

Tal com s'aprecia en la figura 1, l'objecte realment important és Document i el seu objecte fill Forms[ ], ja que la majoria d'elements del DOM tradicional són elements de formulari.

La simplicitat del model, però, feia que no suportés el maneig de text, imatges, miniaplicacions (*applets*) i objectes incrustats i que no es pogués accedir a les propietats de presentació de la majoria d'elements de la pàgina web.

Tot seguit es presentaran l'objecte Document i la gestió d'esdeveniments disponibles en el model, amb el simple objectiu de revisar-ne les limitacions, de manera que al llarg del mòdul se n'estudiarà l'evolució.

1.1.1. L'objecte Document

L'objecte Document proporciona accés a elements de la pàgina com les àncores, els enllaços i els elements dels formularis, però a més disposa d'un seguit de propietats que defineixen l'aspecte de la pàgina, com el color del fons o del text.

Les àncores

Les àncores en HTML permeten accedir a una posició del document HTML. Per a dur a terme aquest accés, abans que res s'ha d'especificar el lloc del document al qual es vol accedir:

```
<A id="ancora">
```

La crida a aquesta àncora es fa utilitzant l'etiqueta següent:

```
<A HREF="#ancora">Saltem a la posició del document marcada</A>
```

De la mateixa manera, es pot accedir a àncores situades en documents remots. Per a això s'afegeix el nom de l'àncora a l'URL de la manera següent:

```
<A HREF="enllaços.html#ancora">Una altra vegada</A>
```

Quan el navegador carrega una pàgina HTML, s'instancien objectes de JavaScript per a tots els elements HTML que es poden manipular des del llenguatge de programació. El conjunt d'elements que es poden codificar es limita en el model tradicional a les àncores, els enllaços, els formularis i els elements del formulari.

Si s'ha carregat una pàgina HTML que conté dos formularis:

```
.......
<form>
<input type="text">
</form>
<br>
<form>
```

```
<input type="text">
</form>
<br>
........
```

es pot accedir al primer formulari amb la sintaxi següent:

```
window.document.forms[0];
```

i a més, com que l'objecte Forms[ ] disposa d'una col·lecció elements[ ], es pot accedir al primer element del formulari amb la sintaxi següent:

```
window.document.forms[0].elements[0];
```

Per tant, l'accés als elements dels formularis es pot fer utilitzant les arrays forms[ ] i elements[ ], però aquesta sintaxi no es pot fer servir gaire, ja que implica saber la posició en l'array de l'element a què es vol accedir.

Per a resoldre l'inconvenient anterior, s'han d'identificar els elements de la pàgina HTML amb l'objectiu d'accedir-hi pel nom que els identifica. Per a això s'ha d'utilitzar l'atribut id de les etiquetes.

Tot seguit, es modifica l'exemple anterior afegint l'identificador en els elements form i input:

```
.......
<form id="registre">
<input type="text" id="nom">
</form>
<br>
<form id="encarrec">
<input type="text" id="article">
</form>
</br>
........
```

Amb el codi HTML anterior es pot accedir als mateixos elements que en l'exemple anterior també, però d'una manera més simple:

```
window.document.registre;
window.document.registre.nom;
```

A més, es pot no utilitzar la referència a Window, ja que se sobreentén i no és obligada, de manera que l'exemple queda així:

```
document.registre;
```

```
document.registre.nom;
```

1.1.2. Gestió d'esdeveniments

Els scripts, però, no solament s'executen quan es carrega la pàgina HTML, sinó que també ho fan com a resposta a certes accions que ha fet l'usuari de la pàgina HTML. Els esdeveniments més comuns són Click, MouseOver i MouseOut, que es produeixen quan l'usuari fa un clic amb el ratolí, quan el passa per sobre o quan el desplaça fora d'algun element del document HTML, respectivament.

Aquests esdeveniments estan associats a botons de formularis, camps de text, imatges, etc. S'ha de tenir en compte, però, que no tots els objectes són capaços de controlar tots els esdeveniments, sobretot en el model tradicional, ja que a mesura que el DOM va evolucionar es van anar incorporant esdeveniments als objectes del DOM.

La gestió dels esdeveniments s'implementa amb el controlador d'esdeveniments, que és un codi JavaScript associat a un objecte del document i a algun esdeveniment específic que suporta aquest objecte. Aquest codi s'executa quan l'esdeveniment ocorre en l'objecte del document en què s'ha definit.

L'exemple següent mostra un missatge d'alerta quan l'usuari fa un clic amb el ratolí al botó boto1 del formulari registre:

```
<form id="registre">
<input id="envia" type="button" value="Accepta" onclick="alert('Document enviat')">
</form>
```

La propietat onclick de l'objecte Button vincula el codi JavaScript a l'esdeveniment Click que hi ha disponible en l'objecte. D'aquesta manera, quan l'usuari fa un clic al botó, el navegador envia l'esdeveniment Click a l'objecte Button i, com que aquest objecte té especificada una funció en el controlador d'esdeveniments, executa el codi que deixa anar l'alerta.

El model d'esdeveniments defineix el conjunt d'interfícies i objectes que permeten el maneig d'esdeveniments. Els controladors d'esdeveniments es poden definir directament en l'objecte mateix com una propietat més en comptes d'utilitzar l'HTML per a portar a cap l'assignació, com s'ha fet en l'exemple anterior. L'exemple següent és equivalent a l'anterior:

```
<form id="registre">
<input id="envia" type="button" value="Accepta">
</form>
<script type="text/javascript">
        <!-
        document.registre.envia.onclick = new Function(alert('Document enviat'));
```

```
        //->
</script>
```

Fixem-nos que el controlador d'esdeveniments s'ha definit o s'ha enllaçat a la propietat onclick de l'objecte i les accions s'han definit a partir d'una funció anònima.

Vegeu també

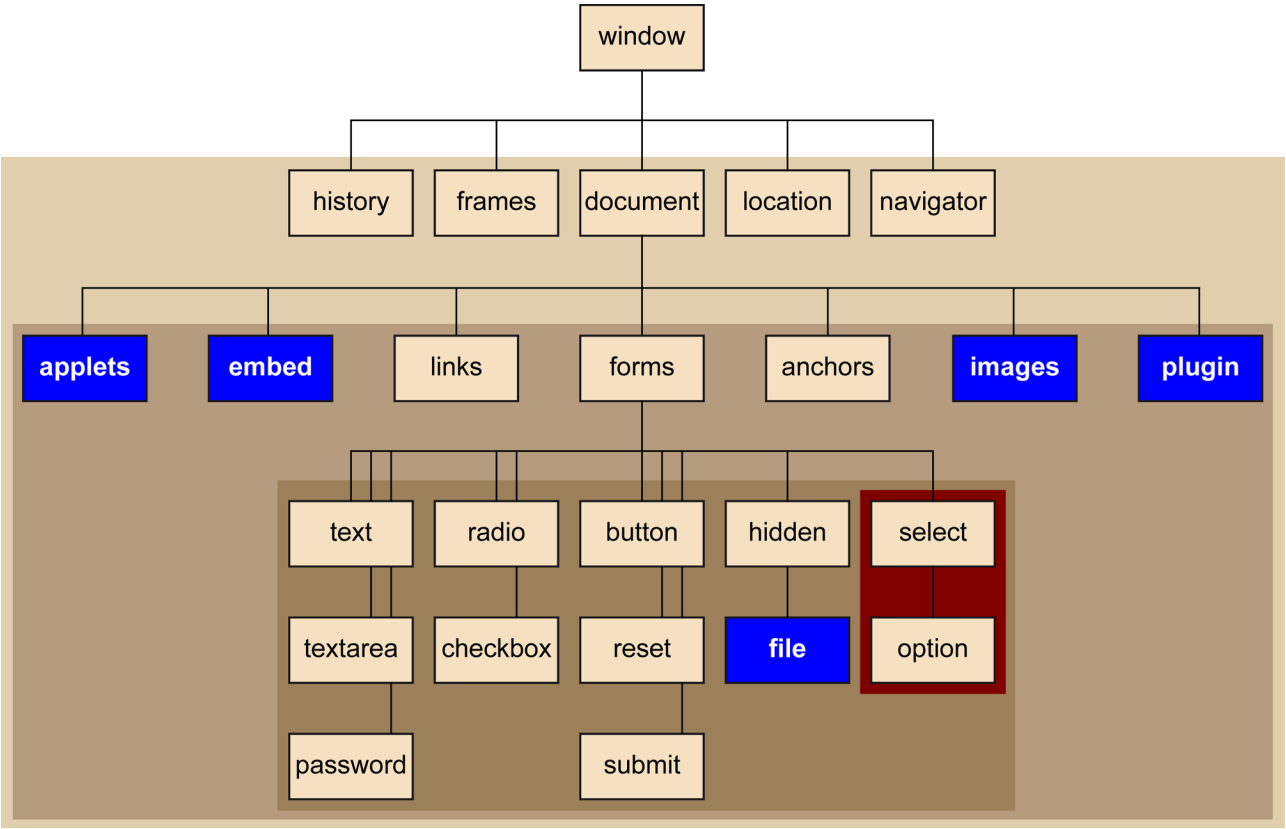
El model d'esdeveniments i el conjunt de controladors d'esdeveniments disponibles s'estudiaran en detall en l'apartat 4 d'aquest mòdul.

1.2. Model tradicional estès

La versió 3 de Netscape introdueix un conjunt nou d'objectes de document, cosa que proporciona capacitat d'accés a imatges, connectors (*plug-ins*), mini-aplicacions i objectes incrustats a la pàgina HTML.

L'estructura i els components d'aquest model es representen en la figura 2:

Figura 2. Model d'objectes tradicional estès (Netscape 3)



La novetat que va tenir més impacte va ser la inclusió de l'array images[ ], perquè encara que la majoria de propietats de l'objecte Image eren de lectura, la propietat src, que definia la ruta de la imatge, es podia modificar i això va permetre de canviar dinàmicament imatges en la pàgina web, com a resposta d'esdeveniments de l'usuari.

Una aplicació típica van ser els botons rollover, que canviaven d'aspecte quan el ratolí s'hi posava a sobre.

Tot seguit es mostra un exemple de creació de botó rollover:

```
<a href="#"
onmouseover="document.images[0].src='/images/botóON.gif' "
onmouseout="document.images[1].src='/images/botóOFF.gif' " >
<img src=/"images/botoOFF.gif">
</a>
```

Amb el codi anterior, quan el punter del ratolí se situa sobre la imatge, el controlador d'esdeveniments onmouseover de l'etiqueta canvia les imatges i, quan el punter surt de la imatge, és el controlador onmouseout el que torna a canviar la imatge.

Encara que van aparèixer més mètodes i propietats, no n'hi va haver cap que tingués l'impacte de l'objecte Image.

1.3. Model d'objectes d'HTML dinàmic

El model d'objectes d'HTML dinàmic va ser implementat en les versions 4.0, tant de Netscape Navigator com d'Internet Explorer, però cadascun dels navegadors va evolucionar d'una manera divergent.

Això darrer va crear un buit tan important en l'ús del DOM, en cadascun dels navegadors, que va derivar en un problema gros per als programadors.

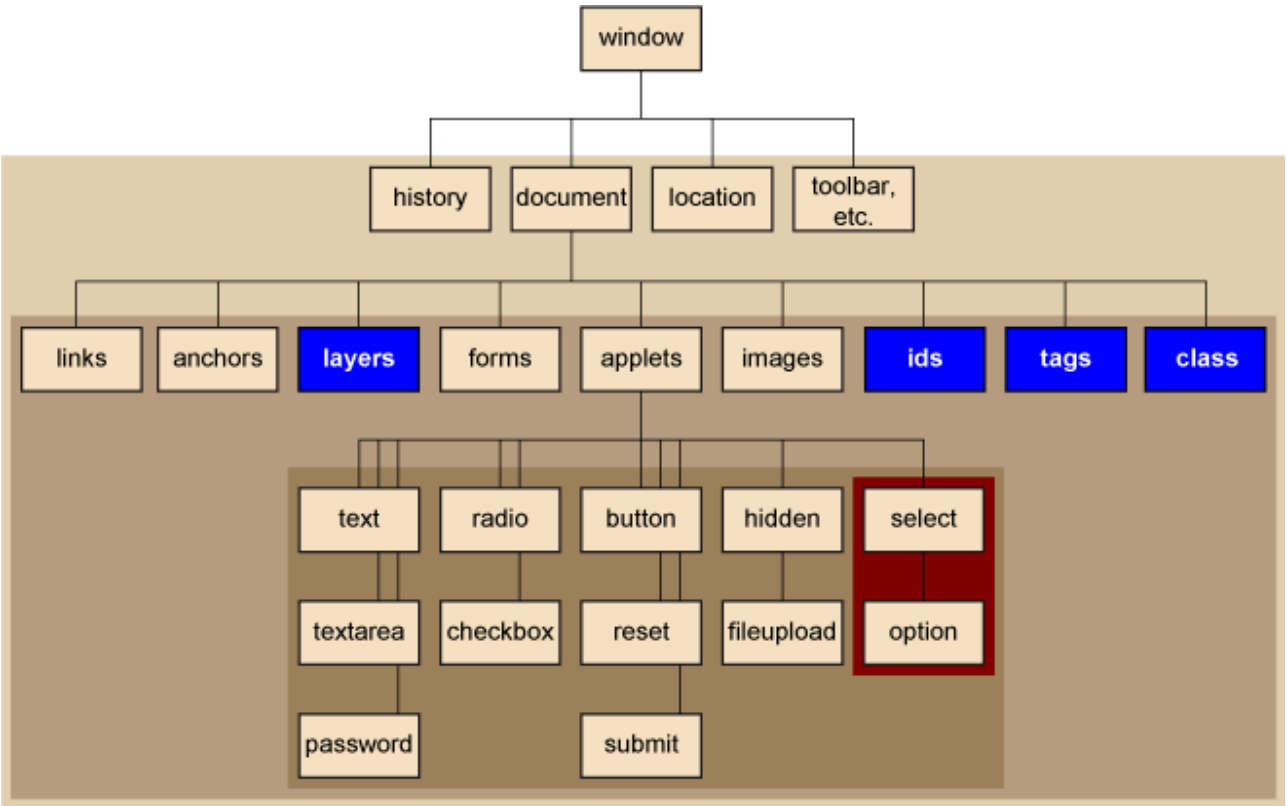
1.3.1. Netscape 4

L'única animació possible fins a la incorporació de la versió 4 del navegador era la permutació d'imatges que s'ha vist en el subapartat anterior; les pàgines web eren completament estàtiques.

El canvi principal que va aportar aquesta versió nova va ser el suport per a l'etiqueta propietària `<layer>`, algunes novetats en el model de gestió d'esdeveniments i l'objecte Style amb mètodes que permetien modificar els fulls d'estil.

En la figura següent es veuen les novetats que apareixen en DOM:

Figura 3. Model d'objectes HTML dinàmic de Netscape 4



Els objectes nous introduïts són els següents:

- ids: permet assignar fulls d'estil a elements HTML amb l'atribut id establert;
- classes: permet assignar fulls d'estil a elements HTML amb l'atribut class establert;
- tags: permet assignar fulls d'estil a elements HTML lliures;
- layers[ ]: array de capes del document.

Els tres primers objectes van permetre que els programadors creessin o manegessin atributs CSS per a tots els elements del document. Encara que va ser una característica molt potent, tenia una limitació, que es basava en el fet que només es podien crear o modificar elements d'estil abans que mostrés en pantalla el contingut a què s'aplicava, perquè no es podien modificar estils quan els objectes ja s'havien carregat a la pàgina web.

La introducció de l'etiqueta `<layer>` va permetre definir àrees de contingut que es podien col·locar en una posició exacta, moure-les, encavalcar-les i ocultar-les. Es podia escriure contingut nou en una capa amb el mètode write() i s'esperava que fos una de les bases de les aplicacions dinàmiques, però com que es tractava d'una etiqueta de propietat de Netscape que no es va incorporar a l'estàndard HTML de W3C, no la va implementar cap navegador de la competència.

També es va afegir una quantitat de propietats noves a l'objecte Window, que proporcionaven informació sobre la grandària de la pantalla, l'alçària, l'amplada, la configuració de la barra d'eines i un conjunt nou de mètodes.

Pel que fa al model d'esdeveniments:

- Van aparèixer controladors d'esdeveniments nous de ratolí i de teclat, com ara onmouseup/down i onkeyup/down.

- Va aparèixer un model nou de gestió d'esdeveniments que fa que els esdeveniments portin a cap un recorregut des dels objectes de la part superior de la jerarquia i descendeixin fins a arribar a l'objecte en què s'ha creat l'esdeveniment.

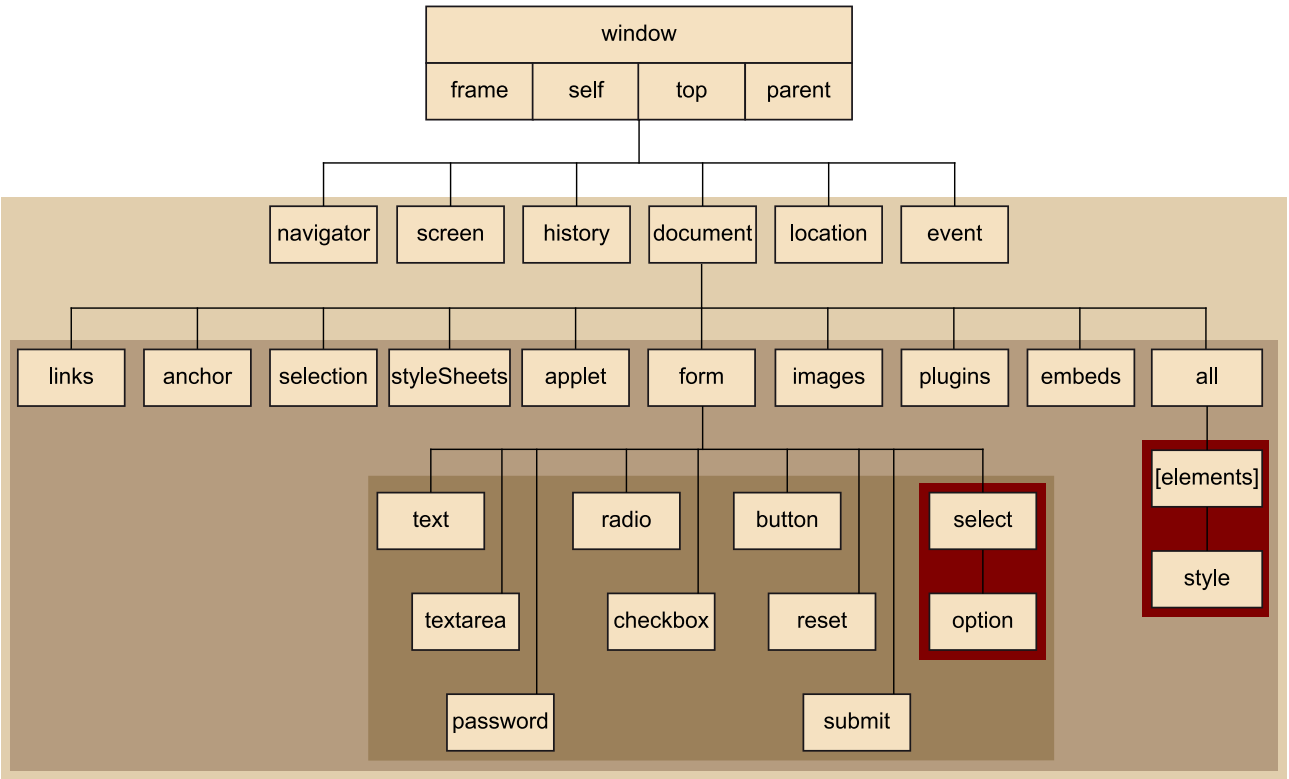
La captura dels esdeveniments es podia fer amb el mètode captureEvents(), que permetia tractar "al nivell més alt" un esdeveniment capturat. Aquesta tècnica podia simplificar la programació de pàgines en què un nombre elevat d'objectes de baix nivell, com els camps de formulari, requeria un mateix tractament davant d'un esdeveniment. D'aquesta manera, s'evitava haver de repetir en cadascun dels objectes el mateix controlador d'esdeveniments.

1.3.2. Internet Explorer 4

Microsoft, per la seva banda, va comercialitzar Internet Explorer 4 (IE4), i va anar més enllà del que va proporcionar Netscape, sobretot perquè va posar a l'abast de JavaScript "tots" els elements HTML, però amb l'inconvenient que aquest model era incompatible amb el de Netscape.

La figura 4 representa el model d'objectes d'Internet Explorer 4:

Figura 4. Model d'objectes HTML dinàmic d'Internet Explorer 4



Les propietats noves de l'objecte Document que s'hi van introduir van ser les següents:

- all[ ]: array que conté totes les etiquetes HTML del document;
- applets[ ]: array que conté totes les miniaplicacions del document;
- children[ ]: array de tots els elements fill de l'objecte;
- embeds[ ]: array formada pels objectes encastats del document;
- images[ ]: array d'imatges del document:
- scripts[ ]: array de scripts del document;
- styleSheets[ ]: array d'objectes Style del document.

De les característiques introduïdes, sense cap dubte la més important és l'array document.all[ ] perquè proporcionava accés a tots els elements del document HTML. Aquest accés es basa en l'índex, que pot ser mitjançant l'índex de l'objecte o mitjançant l'atribut id o name.

```
var element = document.all[2];     //element referencia el tercer objecte del
                                   //document HTML
var element2 = document.all["Enviar"];    //element2 referencia l'objecte del
                                          //document HTML amb id "Enviar"
```

Una característica interessant és que document.all[ ] situa en un mateix nivell tots els elements del document HTML (independentment de la posició que tenen en la jerarquia); d'aquesta manera permet un accés ràpid i senzill a qualsevol part del document HTML.

Una altra característica fonamental disponible en IE4 va ser que es podia modificar el disseny d'una manera dinàmica (després d'haver carregat la pàgina, a diferència de Netscape 4).

Això va implicar que, per primera vegada des de JavaScript, es podia manejar tota l'estructura del document, el contingut de les variables i l'estètica de tots els aspectes de la pàgina HTML d'una manera dinàmica.

Pel que fa al model d'esdeveniments implementat, va ser del tot oposat al de Netscape 4, perquè els esdeveniments comencen a l'objecte en què ocorren i pugen pels objectes ascendents fins a arribar a l'objecte Window. D'aquesta manera, qualsevol objecte pel qual passa l'esdeveniment en el camí que fa fins a Window pot capturar l'esdeveniment, processar-lo o cancel·lar-lo.

1.4. Model d'objectes del DOM estàndard

D'acord amb el que s'ha dit en els subapartats anteriors, el problema principal fins ara és que cada proveïdor decideix quines característiques HTML presenta al programador JavaScript i quina és la sintaxi que s'ha d'utilitzar. Aquesta situació, com era d'esperar, va provocar una divergència molt important en la interpretació del codi entre els diversos navegadors.

El Consorci World Wide Web (W3C, organització internacional que publica recomanacions per al WWW) va publicar, l'octubre de 1998, una especificació anomenada *DOM.Nivell 1*, en què es van considerar les característiques i els mecanismes de manipulació de tots els elements que hi havia en els arxius HTML i XML.

El novembre de l'any 2000, es va publicar l'especificació del DOM.Nivell 2. Aquesta especificació va incloure la manipulació d'esdeveniments en el navegador, la capacitat d'interacció amb CSS i la manipulació de parts del text a les pàgines del web.

Finalment, el DOM.Nivell 3 es va publicar l'abril de 2004 i utilitza la DTD (definició del tipus de document) i la validació de documents.

D'aquesta manera, el DOM que especifica el W3C proposa una interfície de programació d'aplicacions que posa tots els elements d'una pàgina web a disposició del llenguatge de programació i convida tots els proveïdors a adaptar-la als seus navegadors amb l'objectiu clar de convergir.

Web recomanat

Les especificacions completes de cadascun dels nivells del DOM que ha publicat el W3C són documents públics i es poden consultar en línia o baixar-los.

Una altra manera de visualitzar el DOM és classificar-lo en les categories següents:

- **Nucli del DOM:** especifica un model de gestió de document format per etiquetes amb una estructura jeràrquica.

- **DOM HTML:** especifica una extensió de l'anterior per fer-lo servir amb HTML, proporciona les característiques que fan falta per a gestionar documents HTML i utilitza una sintaxi semblant a la dels models d'objectes tradicionals.

- **Esdeveniments DOM:** especifica el control d'esdeveniments, tant d'esdeveniments d'interfície d'usuari com d'esdeveniments del DOM que es produeixen quan es modifiquen parts de l'estructura del document.

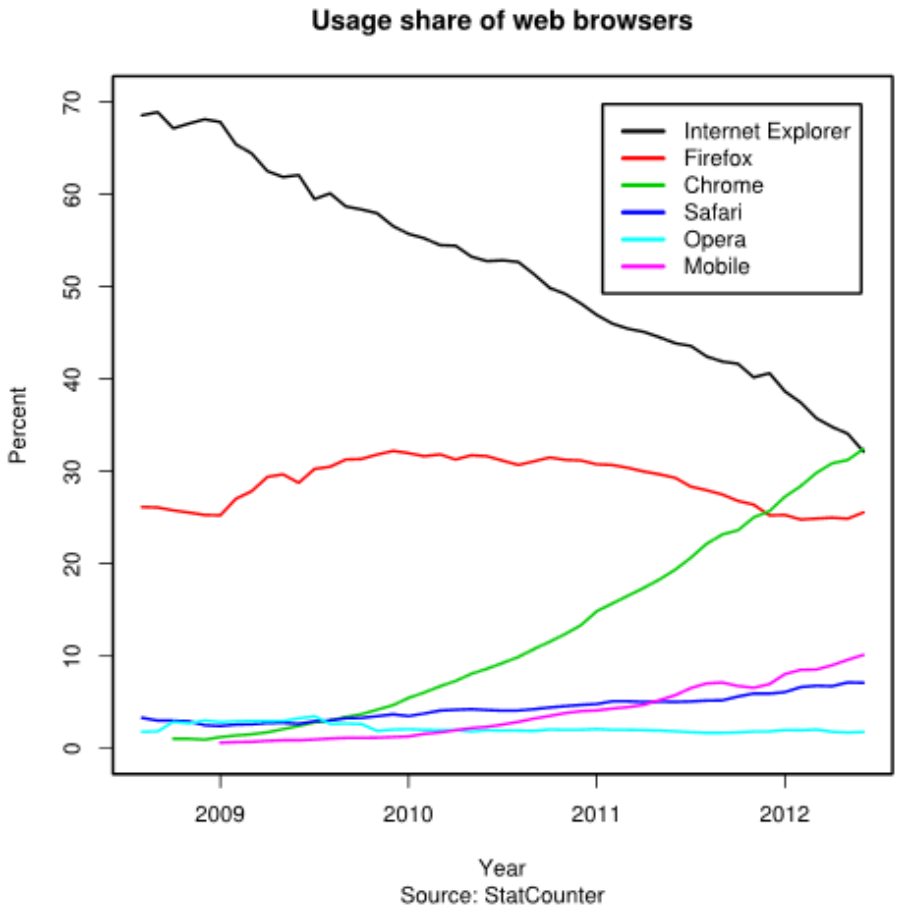
- **DOM CSS:** especifica les interfícies que permeten gestionar les regles CSS des del llenguatge de programació.

- **DOM XML:** és l'equivalent al DOM HTML, però en aquest cas es tracta de documents XML.

1.4.1. Nivell de compliment de l'estàndard

En el gràfic següent es pot observar l'evolució en els últims anys de l'ús dels diferents navegadors que hi ha en el mercat actualment.

Figura 5. Percentatge d'ús dels navegadors principals (2008-2009)



Font: http://en.wikipedia.org/wiki/Usage_share_of_web_browsers

En general el nivell de compliment del DOM és molt variat; en la taula següent es pot observar l'estat de l'art de manera molt general.

Taula 1. Nivell de compliment dels estàndards del DOM del W3C

	IE	Firefox	Safari	Chrome	Opera
DOM1	Sí	Sí	Sí	Sí	Sí
DOM2	Gairebé per complet	Gairebé per complet	Gairebé per complet	Gairebé per complet	Gairebé per complet
DOM3	No	Parcialment	Parcialment	Parcialment	Parcialment

Web recomanat

N'hi ha un de comparatiu amb un nivell de detall més gran i actualitzat al web: http://en.wikipedia.org/wiki/Comparison_of_layout_engines_%28Document_Object_Model%29

2. El model estàndard d'objectes

En aquest apartat es presentarà el model estàndard d'objectes que hi ha plantejats en el DOM Nivell 1 i Nivell 2 del W3C. Per tant, en els subapartats següents s'estudiarà l'estructura jeràrquica definida per l'estàndard i la tipologia d'objectes que hi ha en un document HTML, per a acabar amb els mètodes que permeten accedir, modificar i eliminar qualsevol element de la jerarquia.

2.1. Jerarquia d'elements

Tot seguit es presenta una pàgina HTML senzilla que servirà de base de l'anàlisi i presentació que es farà més endavant dels conceptes que defineixen el DOM.

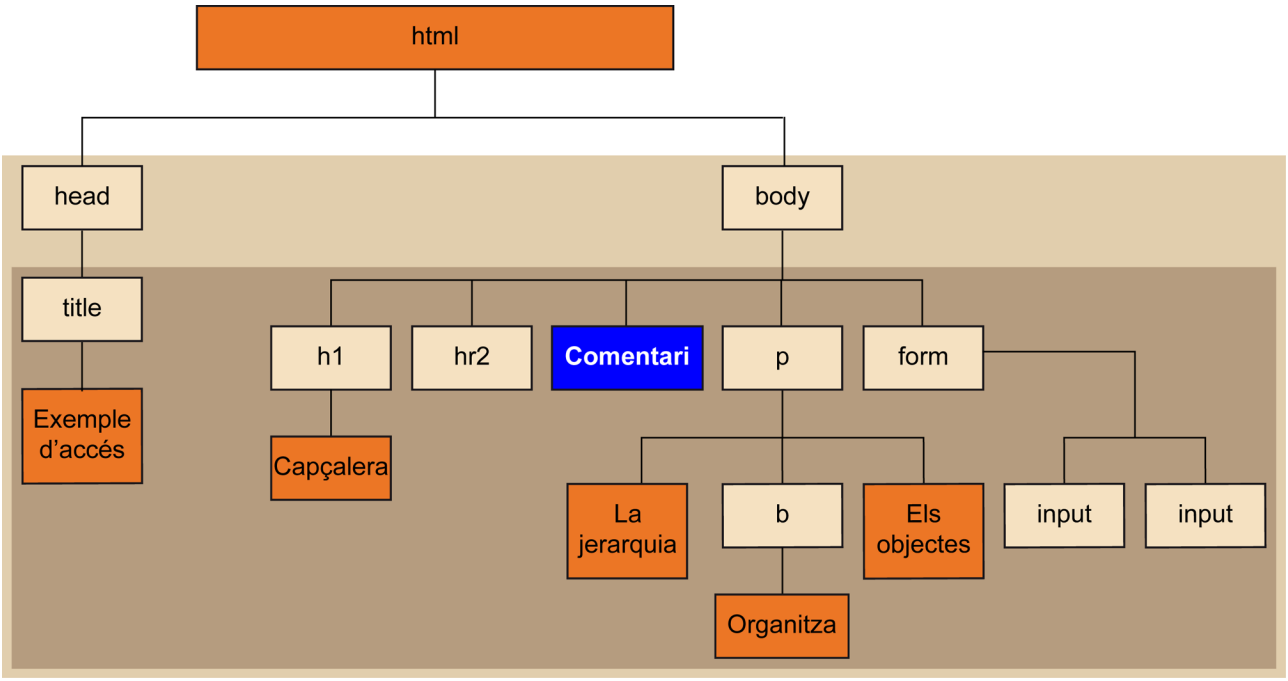
```
<html>
<head>
<!DOCTYPE HTML PUBLIC "//W3C//DTD HTML 4.01 Transitional//EN" http://www.w3.org/TR/html4/loose.dtd>
<title>Exemple d'accés</title>
</head>
<body>
<h1 id="cap">Capçalera</h1>
<hr>
<!--Comentari-->
<p id="p1">La jerarquia <b id="b1">organitza</b> els objectes</p>
<form name="form1" id="form1">
<input type="text" name="camp1" id="camp1">
<input type="button" name="executa" id="executa" value="Executa">
</form>
</body>
</html>
```

Quan el navegador carrega un document HTML, crea una estructura d'arbre a partir de les etiquetes HTML que defineixen el document. Aquesta estructura jeràrquica és senzilla d'intuir, ja que el codi HTML per si mateix segueix una certa estructura jeràrquica, és a dir:

- L'etiqueta `<html>` conté totes les etiquetes que defineixen la pàgina web.
- L'etiqueta `<head>` conté l'etiqueta `<title>`.
- L'etiqueta `<body>` conté les etiquetes `<h1>`, `<hr>`, `<!-->` i `<p>`, que, al seu torn, conté una etiqueta `<b>`.
- L'etiqueta `<form>` conté dues etiquetes `<input>`.

Si es representa l'estructura anterior en un gràfic, en el qual es mostrin les relacions a partir de línies que uneixen les etiquetes, s'obté l'estructura següent:

Figura 6



Cada element de l'arbre es coneix com a **node**; cada node pot contenir, al seu torn, altres nodes que poden ser de diferents tipus (en l'exemple n'hi ha tres tipus diferents: etiquetes, comentaris i text).

Estructura jeràrquica

Una pàgina web s'organitza d'una manera jeràrquica seguint l'estructura que defineixen les etiquetes HTML.

En l'estàndard, el DOM estableix dotze tipus de nodes, però la majoria només són útils en documents XML, cosa que redueix aquesta llista a la meitat en el cas de documents HTML, tal com es pot apreciar en la taula següent:

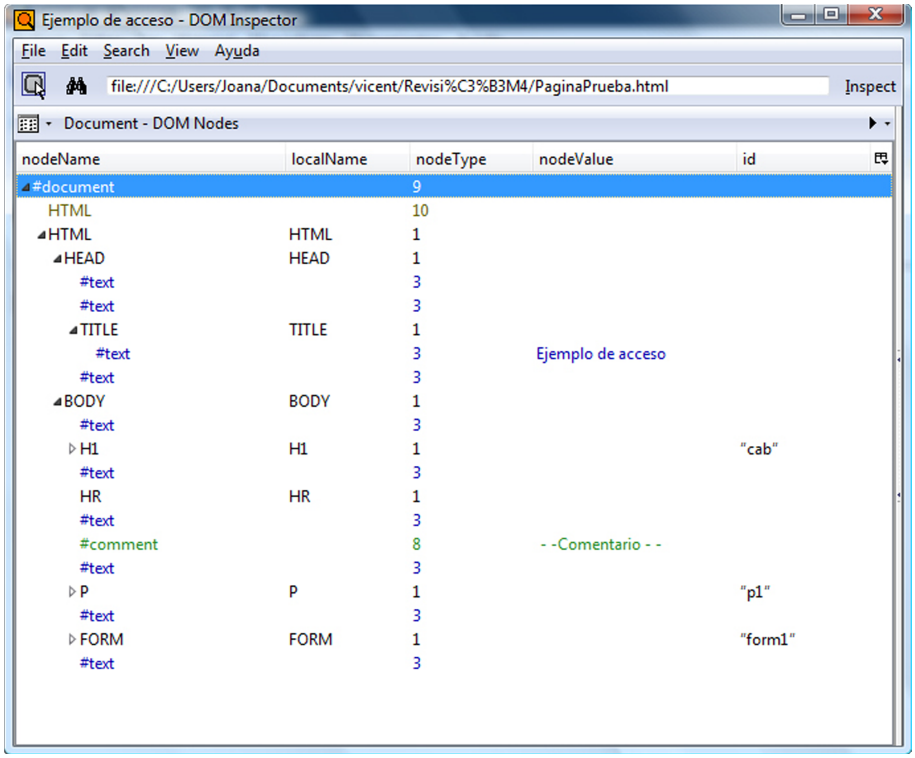
Taula 2. Tipus de nodes de l'estàndard DOM per a HTML

Identificador	Tipus	Descripció
1	Element	Una etiqueta HTML, per exemple `<b>`, `<input>`.
2	Atribut	Un atribut d'una etiqueta HTML, per exemple align="center".
3	Text	Un text inclòs en una etiqueta HTML; per exemple, en la nostra senzilla pàgina HTML, "Exemple d'accés".
8	Comentari	Un comentari HTML, per exemple "Comentari".
9	Document	Document arrel del document, és a dir, l'etiqueta `<html>`.
10	Tipus document	Una definició del tipus de document; es tracta de l'etiqueta `!DOCTYPE`, per exemple `<!DOCTYPE HTML PUBLIC "//W3C//DTD HTML 4.01 Transitional//EN" http://www.w3.org/TR/html4/loose.dtd>`.

Es pot analitzar l'estructura del document a partir d'unes eines que inspeccionen la pàgina HTML. Firefox té un complement que es diu *DOM Inspector* (es pot baixar d'https://addons.mozilla.org/ca/firefox/addon/dom-inspector-6622/), que mostra la jerarquia del document des del punt de vista de l'estàndard del W3C.

En el cas de l'exemple anterior, l'inspector de DOM torna l'estructura següent:

Figura 7. DOM inspector



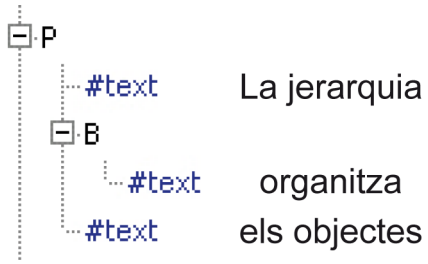
La configuració de les columnes de l'inspector del DOM es fa a partir de la icona:

seleccionant les columnes que es volen mostrar. En l'exemple anterior: nodeName, localName, nodeType, nodeValue i id.

Com es veu, les etiquetes HTML són nodes del tipus 1; el text del títol, de la capçalera i del paràgraf són nodes del tipus 3, i el comentari és un node del tipus 8. Si s'hagués definit algun atribut per a una de les etiquetes, apareixeria com un node del tipus 2.

Les relacions entre els nodes de la jerarquia són semblants a les relacions en els arbres genealògics. Per a explicar aquestes relacions, se selecciona el subarbre del paràgraf de text. L'estructura d'aquest arbre és la següent:

Figura 8. Estructura del subarbre del paràgraf de text



D'aquest subarbre s'observen les relacions següents:

- L'etiqueta `P` és el node pare i té tres fills: el node de text "La jerarquia", el node element "B" i el node de text "els objectes".

- L'ordre dels fills coincideix amb el que s'ha indicat en el punt anterior, és a dir, el primer fill de `P` és el node de text "La jerarquia", el següent és el node element "B" i el darrer, el node de text "els objectes".

- El node de text "organitza" és fill del node element "B", però no del node "P", encara que sí que n'és descendent (nét).

Aclarida la jerarquia d'elements o de nodes que es creen a partir d'una pàgina HTML, tot seguit s'estudiaran els mètodes del DOM que permeten accedir a cadascun dels nodes.

2.2. Accedir als nodes

Arribats a aquest moment, ja es tenen prou coneixements de la jerarquia que interpreta el DOM del W3C per a començar a estudiar l'accés als elements d'aquesta jerarquia. En primer lloc, es presenta un senzill script que utilitza un primer mètode d'accés:

```
var element= document.getElementById("camp1");
var msg = "Nom: "+element.nodeName+"\n";
msg += "Tipus: "+element.nodeType+"\n";
msg +="Valor: "+element.nodeValue+"\n";
alert(msg);
```

L'script utilitza el mètode de l'estàndard getElementById(). A partir d'un id que s'hi passa com a paràmetre, aquest mètode torna el node o element i s'assigna a la variable corresponent. Es tracta d'un mètode amb una funcionalitat semblant a la que va introduir IE4 amb l'array all, perquè permet buscar directament un element sense haver de fer una navegació per tota la jerarquia.

D'aquesta manera, l'script, una vegada localitzat l'element, emmagatzema en una cadena de text tres propietats del node, nodeName, nodeType i nodeValue, i acaba mostrant aquestes propietats en una finestra alert. El resultat és el següent:

Figura 9

Com es pot veure, el nom del node és INPUT, que és el tipus de l'etiqueta; el tipus és 1, que correspon a un element HTML, i el valor és null. Aquestes propietats ja eren en el DOM inspector.

L'objecte Node té un seguit de propietats que, a més de descriure les característiques principals que té, com el nom, el valor i el tipus, també mostra la relació amb la resta de nodes de l'estructura i permeten de navegar per l'estructura. La taula següent mostra les propietats de l'objecte Node:

Taula 4. Propietats de l'objecte Node

Propietats	Descripció
nodeName	Torna una cadena amb el nom del node.
nodeValue	Torna una cadena amb el valor del node, que només es pot aplicar a nodes de tipus text.
nodeType	Torna un nombre que especifica el tipus de node.
parentNode	Torna una referència al node pare.
firstChild	Torna una referència al primer fill.
lastChild	Torna una referència al darrer fill.
previousSibling	Torna una referència al germà gran.
nextSibling	Torna una referència al germà petit.
childNodes	Torna una array amb els nodes fills.
attributes	Torna una array amb els atributs del node.

Propietats	Descripció
ownerDocument	Torna l'objecte Document que el conté.

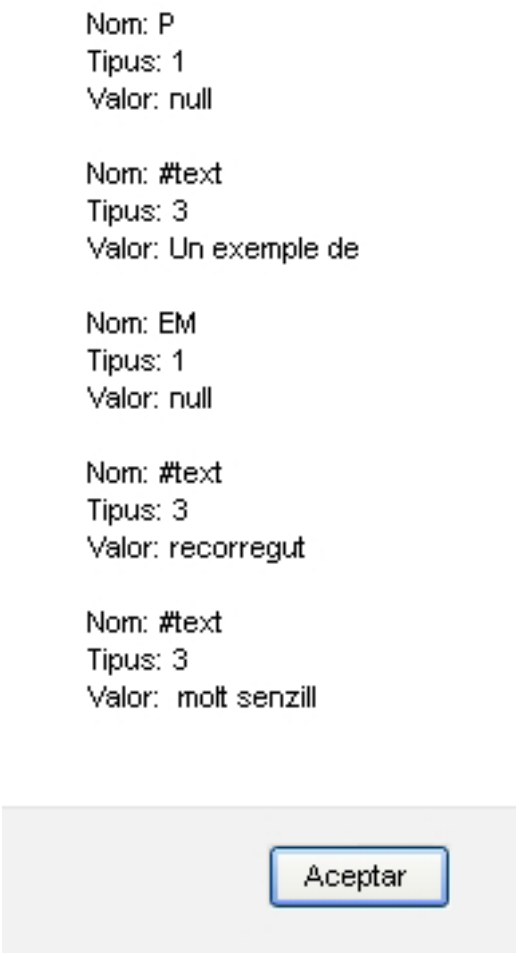
Amb aquestes propietats es poden fer recorreguts per l'arbre a partir d'un node concret.

Tot seguit es planteja un recorregut per la jerarquia a partir d'un exemple amb una estructura molt senzilla:

```
<html>
<head>
</head>
<body>
<p id="p1" align="center">Un exemple de <em>recorregut</em> molt senzill</p>
<script type="text/javascript">
function tipusNode(node){
       var temp = "";
       temp += "Nom: "+node.nodeName+"\n";
       temp += "Tipus: "+node.nodeType+"\n";
       temp +="Valor: "+node.nodeValue+"\n\n";
       return temp;
}
var nodeActual = document.getElementById("p1");
var msg = tipusNode(nodeActual);
nodeActual = nodeActual.firstChild;
msg += tipusNode(nodeActual);
nodeActual = nodeActual.nextSibling;
msg += tipusNode(nodeActual);
nodeActual = nodeActual.firstChild;
msg += tipusNode(nodeActual);
nodeActual = nodeActual.parentNode;
nodeActual = nodeActual.previousSibling;
nodeActual = nodeActual.parentNode;
nodeActual = nodeActual.lastChild;
msg += tipusNode(nodeActual);
alert(msg);
</script>
</body>
</html>
```

El resultat de l'exemple anterior és el següent:

Figura 10



El recorregut per la jerarquia, però, s'ha fet sense problemes, perquè se sabia l'estructura i, per tant, les relacions entre els diversos nodes. Evidentment, aquesta situació no es dóna en la realitat, ja que no es pot saber *a priori* l'estructura, ni tan sols si un node té fills o germans.

Per això, es pot fer una navegació completa utilitzant l'estratègia següent: el recorregut comença per un node X, es comprova si el node té fills i, si els té, s'accedeix al fill petit i, si no els té, només es pot buscar un germà petit o bé comprovar si té un node pare.

Per a establir un recorregut com l'anterior, es poden utilitzar les estructures següents:

```
if (nodeActual.hasChildNodes())
        nodeActual=nodeActual.firstChild;

if (nodeActual.parentNode)
        nodeActual=nodeActual.parentNode;

if (nodeActual.nextSibling)
```

```
nodeActual=nodeActual.nextSibling;
```

Tot seguit es presenta una funció que a partir d'un node fa un recorregut per l'estructura jeràrquica:

```
function mouNode(elementActual, direccio){
        switch (direccio){
                case "previousSibling":
                       if (elementActual.previousSibling)
                               elementActual=elementActual.previousSibling;
                       else
                               alert("No té germà anterior");
                break;
                case "nextSibling":
                       if (elementActual.nextSibling)
                               elementActual=elementActual.nextSibling;
                       else
                               alert("No té germà posterior");
                break;
                case "parent":
                       if (elementActual.parentNode)
                               elementActual=elementActual.parentNode;
                       else
                               alert("No té pare");
                break;
                case "firstChild":
                       if (elementActual.hasChildNodes())
                               elementActual=elementActual.firstChild;
                       else
                               alert("No té fills");
                break;
                case "lastChild":
                       if (elementActual.hasChildNodes())
                               elementActual=elementActual.lastChild;
                       else
                               alert("No té fills");
                break;
                default: alert("Crida a la direcció errònia");
        }
return (elementActual); (
}
```

La funció anterior parteix de dos paràmetres: el primer indica el node inicial sobre el qual es vol fer el recorregut, i el segon, en quina direcció es vol moure. La funció torna el node i, si no n'hi ha, avisa amb un missatge.

No és comú o no té massa utilitat el recorregut d'una estructura per a analitzar-la, però és fonamental saber localitzar un node (és senzill amb getElementById()) i poder accedir, a més, a un dels "familiars directes" que té (pare, fill o germans) per a modificar l'estructura del document, i per a això darrer la funció anterior pot ser de gran ajuda.

L'objecte Document no solament disposa del mètode getElementById(). En la taula següent es mostren els mètodes d'accés principals:

Taula 5. Mètodes d'accés principals de l'objecte Document

Mètode	Descripció
getElementById()	Torna l'objecte l'atribut id del qual coincideix amb el que s'ha passat com a paràmetre.
getElementsByName()	Torna una llista amb els objectes amb el valor en l'atribut name que es passa com a paràmetre.
getElementsByTagName()	Torna una llista amb els objectes l'etiqueta HTML dels quals s'hi passa com a paràmetre.
documentElement	Torna l'objecte Arrel, és a dir, l'etiqueta `<html>`.
body	Torna l'objecte corresponent a l'etiqueta `<body>`.

La diferència entre els mètodes getElementsByName() i getElementById() es basa en el fet que l'atribut name pot tenir el mateix valor en diverses etiquetes, mentre que l'especificació sobre l'atribut id indica que aquest identificador ha de ser únic en tot el document. Per aquest motiu, s'ha d'utilitzar l'atribut id davant el name.

2.3. Manipular els nodes

Arribats a aquest punt, es disposa d'una referència a un node, i ens podem moure per la jerarquia a partir dels mètodes anteriors, que permeten passar d'un node al seu node pare, germà o fill. Ara bé, com s'accedeix al contingut del document HTML i com es modifica?

La modificació del contingut d'una pàgina HTML es basa en els punts següents:

- **Modificar nodes:** modificar-ne el contingut o els atributs.

- **Inserir o eliminar nodes:** intercalar un node en una posició en la jerarquia o eliminar un dels nodes de l'estructura.

2.3.1. Modificar nodes

Per començar, exposarem un exemple curt amb un potencial gran, ja que a partir d'aquest exemple es podrà modificar el contingut de qualsevol pàgina HTML.

El contingut de la informació que es mostra en les pàgines HTML s'emmagatzema en nodes del tipus 3 i, tal com es veu en la figura 6, sobre el DOM inspector; aquests nodes no tenen un valor en la columna id, de manera que no s'hi pot accedir utilitzant el mètode getElementById().

Vegeu també

Podeu veure la figura 6 en el subapartat 2.1 d'aquest mòdul.

La solució és simple: s'accedeix al node pare del text que es vol modificar; des d'aquí s'accedeix al node de text mitjançant la propietat firstChild, i finalment es modifica el text utilitzant la propietat nodeValue.

En l'exemple següent es modifica el text "La jerarquia" per "L'estructura" de l'exemple que s'ha plantejat al començament d'aquest apartat:

```
var element= document.getElementById("p1").firstChild;
element.data ="L'estructura";
```

D'aquesta manera, s'accedeix al node pare que té id p1; d'aquest node es passa al primer fill que té, que és el node de text, i en la línia següent se substitueix el contingut del node.

Hi ha un seguit de mètodes que afegeixen funcionalitat a la modificació de nodes de text, ja que en l'exemple anterior només es mostra com s'ha de fer per a substituir tot el contingut d'un node de text. Els mètodes següents permeten la modificació parcial dels nodes de text:

Taula 6. Mètodes per a modificar nodes de text

Mètode	Descripció
appendData(cadena)	Afegeix la cadena al final del node de text.
deleteData(offset, quantitat)	Esborra la quantitat de caràcters començant per l'índex especificat per offset.
insertData(offset, cadena)	Insereix el valor de la cadena començant per l'índex especificat en offset.
replaceData(offset, quantitat, cadena)	Reemplaça la quantitat de caràcters de text en el node començant per offset amb els caràcters especificats a la cadena.
splitText(offset)	Divideix el node de text en dues peces en l'índex donat en offset. Torna la part dreta de la divisió en un node de text nou i de la part esquerra a l'original.
substringData(offset, quantitat)	Torna una cadena corresponent a la subcadena que comença per l'índex offset i continua fins a una quantitat de caràcters.

Els mètodes anteriors afegeixen prou funcionalitat perquè la manipulació del contingut de la pàgina HTML es gestioni d'una manera òptima. Una alternativa a l'ús dels mètodes anteriors és recuperar el contingut del node de text en una cadena, manipular aquesta cadena amb els mètodes que hi ha de l'objecte String utilitzant, si escau, expressions regulars i tornar a assignar el contingut de la cadena.

Tal com es pot intuir, però, després dels exemples anteriors, en realitat no es modifiquen els nodes, sinó els atributs dels nodes (data és un atribut o una propietat del node text).

Per exemple, es vol introduir el valor Introdueix text... en un quadre de text: per a fer-ho, s'ha de modificar l'atribut value del node camp de text camp1:

```
var element= document.getElementById("camp1");
element.setAttribute("value","Introdueix text...");
```

En l'exemple s'ha utilitzat el mètode setAttribute, que assigna un valor a un atribut del node sobre el qual actua. L'objecte Node té tres mètodes que permeten treballar amb els atributs dels nodes. La taula següent mostra la relació de mètodes:

Taula 7. Mètodes de l'objecte Node per a modificar nodes

Mètode	Descripció
getAttribute(nom)	Torna el valor de l'atribut passat com a paràmetre.
setAttribute(nom,valor)	Assigna l'atribut amb el nom i valor passats com a paràmetres.
removeAttribute(nom)	Elimina l'atribut passat com a paràmetre.

2.3.2. Crear i inserir nodes

En el subapartat anterior s'han vist els mètodes que permeten modificar nodes que ja hi havia, però no la inserció de nodes nous en algunes parts del document. El **procés d'inserir un node** en un document té les etapes següents:

- Crear el node que es vol inserir.
- Localitzar el lloc en la jerarquia en què s'ha d'inserir.
- Inserir el node nou al lloc adequat.

Crear nodes

La creació d'un node és molt simple: l'objecte Document té mètodes que permeten crear nodes. Per exemple, la línia de codi següent crea un node del tipus 1 Element, que representa una etiqueta HTML de paràgraf <p>:

```
nouParagraf = document.createElement("P");
```

D'aquesta manera, la variable *nouParagraf* conté una referència a un node de tipus 1 amb l'etiqueta `P`. D'una manera semblant, es crea un node de tipus text:

```
nouText = document.createTextNode("Node nou de text");
```

En l'exemple anterior, el mètode que s'ha utilitzat és createTextNode, en comptes de createElement.

La taula següent compila els mètodes que s'utilitzen per a crear nodes:

Taula 8. Mètodes per a crear nodes

Mètode	Descripció
createElement(etiqueta)	Crea un node del tipus element amb l'etiqueta passada com a paràmetre.
createTextNode(cadena)	Crea un node de text amb la cadena passada com a paràmetre.
createComment(cadena)	Crea un node de comentari amb la cadena passada com a paràmetre.
createAttribute(nombre)	Crea un atribut per a un element especificat per la cadena nom. S'utilitza rarament.

Com es veu en la taula, hi ha un mètode de creació de nodes per a cadascun dels tipus de nodes que hi ha, exceptuant els tipus 9 i 10, que es refereixen a les etiquetes `<html>` i `<body>`, respectivament.

Una alternativa a la creació d'un node és el **clonatge** o la **còpia** d'un que existeix; per a això, l'objecte Node té el mètode cloneNode(boolea), en què el paràmetre booleà és un argument lògic que indica si la còpia ha d'incloure tots els fills del node o només l'element per si mateix.

La característica anterior és molt interessant, perquè permet no solament copiar un node sinó també tots els fills que té.

Fixeu-vos en l'exemple següent:

```
function clonaIcopia(idNodeC,idNodeI){
        var nodeAClonar = document.getElementById(idNodeC);
        var nodeClonat = nodeAClonar.cloneNode(true);
        var puntInsercio = document.getElementById(idNodeI);
        puntInsercio.appendChild(nodeClonat);
}
```

La funció té les característiques següents:

- Rep dos paràmetres: el primer és l'id del node que s'ha de clonar i el segon és l'id del node en què s'ha d'inserir com a fill el clon.

- La variable *nodeAClonar* és la referència al node que es vol clonar, localitzat amb la funció getElementById.

- La variable *nodeClonat* és la referència al node nou que s'ha clonat i que, quan es passarà a la funció cloneNode el paràmetre true, es clonarà no solament l'element en si mateix, sinó també tots els fills que té.

- La variable *puntInsercio* és la referència del node pare al qual s'inserirà en la línia de codi següent el node clonat.

- La darrera línia de codi mostra un mètode nou, que es presentarà tot seguit, però és intuïtiu pensar que el mètode insereix el node clonat com a fill del node puntInsercio.

Inserir nodes

Tal com s'ha vist en l'exemple anterior, quan ja es té el node que es vol afegir al document, s'ha d'inserir en algun lloc de la jerarquia del document.

La inserció d'un node en la jerarquia sempre es duu a terme partint del node pare, amb els mètodes que es mostren en la taula següent:

Taula 9. Mètodes per a inserir nodes

Mètode	Descripció
insertBefore(nouNode, nodeFill)	S'especifica davant de quin fill nodeFill es vol inserir el node nouNode.
appendChild(nouNode)	Afegeix el node nouNode al final de la llista dels fills del node que ha fet la crida al mètode.

Per tant, la diferència entre els dos mètodes es basa en el fet que el primer mètode insereix el node que s'ha passat en el paràmetre nouNode, a l'esquerra del fill que s'ha passat com a segon paràmetre nodeFill; és a dir, el node nou és el germà gran del que es passa com a paràmetre.

Tanmateix, el mètode appendChild() afegeix el node nou al final de la llista dels fills i, per tant, són el fill petit o el que hi ha més a la dreta de tots els nodes fills.

Els mètodes anteriors es veuen en l'exemple següent:

```
var nouNode = document.getElementById("form1").cloneNode(true);
var pare = document.getElementById("cap");
```

```
pare.appendChild(nouNode);
```

en què:

- Es crea un clon d'un formulari identificat amb l'id form1, ja que no solament es clona l'etiqueta del formulari sinó totes les filles que té quan es passa com a paràmetre del mètode cloneNode el valor true.

- Es localitza el node pare sobre el qual s'ha d'inserir el formulari, que està identificat amb l'id cap.

- El codi s'acaba amb l'addició del formulari com a fill més petit del node cap, per al qual es fa servir el mètode appendChild.

Eliminar i reemplaçar nodes

A vegades és necessari eliminar nodes en l'estructura del document. Per a això l'objecte Node té el mètode removeChild(nodeFill), que s'utilitza per a eliminar un node especificat pel paràmetre nodeFill. Per exemple:

```
nodeActual.removeChild(nodeActual.lastChild);
```

El codi anterior elimina el darrer fill del node a què fa referència la variable nodeActual; és important saber que, a més d'eliminar-lo, el torna i es pot assignat a una variable.

A més d'eliminar un node, se'n pot reemplaçar un per un altre utilitzant el mètode replaceChild(fillNou, fillReemplacat), en què el node fillNou reemplaça el node fillReemplacat.

3. Els objectes del BOM

En aquest apartat es presentaran els objectes juntament amb les propietats i els mètodes principals que formen part del BOM, de manera que en l'apartat següent es presentarà el DOM, és a dir, tot el model d'objectes que descendeix des de l'objecte document.

Com era d'esperar, la llista d'objectes presentats és dinàmica i evoluciona en paral·lel a l'aparició de versions noves dels navegadors. La utopia de l'estandardització continua essent un objectiu pendent i no pas una realitat com caldria.

És important destacar que aquesta part del BOM té el nombre més gran de divergències, ja que està relacionada directament amb el navegador.

Per tant, l'objectiu de l'apartat és presentar els objectes i els components d'aquests objectes, que juntament amb alguns exemples ens ajudaran a comprendre el potencial del procés de manipulació del DOM des de JavaScript. En cap moment no es planteja que s'hagin d'aprendre "de memòria" tots els objectes i els mètodes i les propietats que tenen, perquè no té cap sentit, ja que sempre es poden consultar en l'apartat mateix o per Internet.

Amb referència a això, amb vista a solucionar problemes amb la compatibilitat del codi, les referències web següents mostren el model d'objectes en cadascun dels navegadors actuals, inclosos tant el BOM com el DOM:

- Especificació del DOM dels navegadors basat en Mozilla: http://developer.mozilla.org/en/Gecko_DOM_Reference.

- Especificació del DOM dels navegadors Internet Explorer: http://msdn.microsoft.com/en-us/default.aspx

3.1. L'objecte Window

L'objecte Window és el que hi ha en el nivell més alt en la jerarquia d'objectes i, per tant, és el contenidor de tot el que es pot veure en el navegador.

Tan aviat com s'executa el navegador, encara que no visualitzi cap document, es crea una instància de Window a què es pot accedir des de JavaScript.

Es poden referenciar totes les finestres que hi ha obertes a la pantalla i les pot manipular una instància de l'objecte (sempre que s'hagi creat des de la mateixa instància que les prova de manipular). Tal com es veurà en aquest subapartat, les propietats d'una instància de Window inclouen la mida, els components, la posició, etc. Els mètodes, per la seva banda, permeten crear i destruir finestres genèriques, finestres d'alerta o de confirmació, etc.

S'ha de tenir en compte que l'objecte Window pot representar la finestra del navegador o un frame (quan n'hi ha més d'un, cadascun d'aquests frames es considera un objecte Window independent).

Així, doncs, un objecte Window es pot crear amb alguna de les opcions següents:

- Les etiquetes `body` o `frameset` d'HTML.
- El mètode open de JavaScript.

L'accés a les propietats i els mètodes de l'objecte Window, per la seva banda, segueix la sintaxi que s'ha presentat en els apartats anteriors:

```
window.nom_propietat;
window.nom_metode([paràmetres]);
```

Quan es fa referència a la finestra que conté el document mateix, es pot fer servir la paraula *self* en substitució de la paraula *window*. Per exemple, per a tancar la finestra actual, es poden fer servir d'una manera indistinta les formes window.close() i self.close().

De tota manera, amb els mètodes open() i close() cal fer servir les formes window.open() i window.close(), ja que aquests mètodes es poden confondre amb els mètodes de l'objecte Document (document.open() i document.close()).

La paraula *self* es pot ometre en els casos més simples i reservar-la per als que impliquen diversos frames. Això és així perquè, quan es visualitza un document, s'assumeix el fet que ja hi ha una finestra, la que el conté. Per tant, es pot accedir a les propietats i els mètodes de la finestra actual sense especificar-la:

```
nom_propietat
nom_metode([paràmetres])
```

Tot seguit es mostrarà una taula amb les propietats principals disponibles de l'objecte Window:

Taula 10. Propietats de l'objecte Window

Nom	Descripció	Exemple

closed	És un valor booleà que indica si s'ha tancat una finestra. Quan es tanca una finestra, l'objecte Window hi continua essent, però amb el valor d'aquesta propietat true. Només és de lectura.	finestra = window.open('doc.htm','finestra1','width=400, height=100)'
		...
		if (**finestra.closed**)
		alert("tancada")
		else
		alert("oberta")

defaultStatus	És el missatge per defecte que es visualitza a la barra d'estat del navegador.	**window.defaultStatus**="Pàgina índex"
document	Fa referència a l'objecte Document contingut a la finestra.	
frames	És una array d'objectes frame. Els frames són els que genera l'etiqueta Frame del FRAMESET contingut en la pàgina i en l'array estan en l'ordre en què s'han creat en el codi HTML. Només és de lectura.	En una pàgina que contingui dos frames, es pot accedir a aquests frames de la manera següent: parent.frames["índex"] parent.frames["fcontingut"] o bé parent.frames[0] parent.frames[1]
history	Fa referència a l'objecte History contingut a la finestra.	

length	Conté el nombre de frames de la finestra. Només és de lectura.	nreframes = **window.length;**
		if (nreframes == 0)
		alert("no conté frames");
		else
		alert("conté "+numframes+" frames");

location	Fa referència a l'objecte Location contingut a la finestra.	
name	Cadena de text que especifica el nom de la finestra o marc. Només és de lectura.	finestra = window.open('doc.htm','finestra1','width=400, height=100)' alert(**finestra.name**)
navigator	Fa referència a l'objecte Navigator contingut a la finestra.	
opener	Fa referència a l'objecte Window que ha obert la finestra actual. Aquesta propietat persisteix encara que el document que ha fet la crida s'hagi baixat.	Tanca la finestra que ha obert la finestra actual: **window.opener**.close() Tanca la finestra del navegador: **top.opener**.close()
parent	Fa referència a l'objecte Window o Frame pare del marc actual. És la finestra que conté el frameset.	En una pàgina que contingui dos frames, es pot accedir al segon des del primer amb la sentència següent: **parent**.frames[1]
screen	Fa referència a l'objecte Screen del navegador.	
self	És sinònim de la finestra actual.	

status	Especifica el text que es visualitza a la barra d'estat del navegador.	function ini_status() {
		window.status = "Cliqui en aquest enllaç per accedir a l'índex"
		}
		Index

top	Fa referència a l'objecte Window superior en la jerarquia d'objectes del document. S'utilitza per a accedir a l'objecte Window contenidor des d'un marc.	top.close()
window	És la finestra o el frame actual.	

La taula següent mostra els mètodes principals de l'objecte Window. Igual que en el cas de les propietats, no es tracta d'una llista exhaustiva, sinó que es presenten els mètodes més utilitzats o que l'autor considera més interessants.

Taula 11. Mètodes de l'objecte Window

Nom	Descripció	Sintaxi	Paràmetres	Retorn
alert	Obre una caixa de diàleg amb un missatge i el botó Acceptar.	alert(cadena de text).	String (cadena de text)	
blur	Elimina el focus de la finestra o frame especificats.	blur()		
clearInterval	Cancel·la el timeout que s'ha inicialitzat amb el mètode setInterval.	clearInterval(intervalID)	intervalID s'inicialitza amb la crida prèvia al mètode setInterval.	
close	Tanca la finestra especificada.	close()		
confirm	Obre una caixa de diàleg amb un missatge i els botons Acceptar i Cancel·lar.	confirm("missatge")	String.	true si l'usuari prem Acceptar i false si prem Cancel·lar.
focus	Assigna el focus a la finestra o al frame especificat.	focus()		
moveBy	Mou la finestra a la posició relativa respecte a la seva posició actual el nombre de píxels especificats.	moveBy(horitzontal, vertical)	horitzontal: nombre de píxels per al desplaçament horitzontal. vertical: nombre de píxels per al desplaçament vertical.	
moveTo	Mou la cantonada superior esquerra de la finestra a la posició absoluta de la pantalla especificada en píxels.	moveTo(x, y)	*x*: coordenada *x* *y*: coordenada *y*	
open	Obre una finestra nova del navegador.	open(URL, nom, caracteristiques)	URL: cadena de text que especifica l'URL per a visualitzar en la finestra nova. nom: text que indica el nom de la finestra.	
print	Fa que aparegui el diàleg d'impressió.	print()		
prompt	Obre una caixa de diàleg amb un missatge i un camp d'entrada de text.	prompt(missatge, [text])	missatge: text que mostra la caixa de diàleg. text: opcional, text que per defecte surt en el camp d'entrada de text.	Dada que introdueix l'usuari.
resizeBy	Redimensiona la finestra movent la cantonada inferior dreta segons els valors relatius especificats.	resizeBy(horitzontal, vertical)	horitzontal: nombre de píxels que s'han de sumar o restar a la dimensió horitzontal. vertical: nombre de píxels que s'han de sumar o restar a la dimensió vertical.	
resizeTo	Redimensiona la finestra en els valors especificats.	resizeTo(ample, alt)	ample: amplària en píxels de la finestra. alt: alçària en píxels de la finestra.	

Nom	Descripció	Sintaxi	Paràmetres	Retorn
scroll	Fa un scroll de la finestra a les coordenades especificades.	scroll(x,y)		
scrollBy	Fa un scroll de l'àrea de la finestra, segons els valors relatius a la posició actual.	scrollBy(x,y)	x: nombre de píxels que s'han de sumar o restar per a l'scroll horitzontal. y: nombre de píxels que s'han de sumar o restar per a l'scroll vertical.	
scrollTo	Fa un scroll de l'àrea de la finestra, segons els valors absoluts indicats.	scrollTo(x, y)	x: coordenada *x* en píxels de l'àrea que s'ha de visualitzar. y: coordenada *y* en píxels de l'àrea que s'ha de visualitzar.	
setInterval	Executa una funció, cada vegada que transcorre el temps especificat i fins que s'executa el mètode clearInterval.	setInterval(expresió, temps) o bé setInterval(funció, temps, param1,..., paramN)	temps: nombre de mil·lisegons.	identificador.
setTimeout	Executa una funció, una vegada transcorregut el temps especificat.	setTimeout(expressió, temps) o bé setTimeout(funci, temps, param1,..., paramN)	temps: nombre de mil·lisegons.	identificador.

L'ús d'alguns dels mètodes anteriors necessita l'activació o desactivació d'algunes propietats relacionades amb la seguretat del navegador; això vol dir que pot ser que no funcionin per restriccions de seguretat.

3.1.1. El mètode open

Quan s'obre una finestra, es pot especificar l'adreça URL, el nom, la grandària, els botons i tot un seguit d'atributs que defineixen l'aspecte de la finestra.

La sintaxi del mètode open és aquesta:

```
open(url, nom, característiques, reemplaçar)
```

en què:

- url és l'adreça que indica el document que es carrega a la finestra;
- nom és el nom de la finestra (que s'utilitza per a referenciar-la després);
- característiques és una cadena delimitada per comes que indica un seguit de propietats de la finestra que es crea;
- reemplacar és un valor lògic i opcional que indica si l'URL especificat ha de reemplaçar el contingut de la finestra actual.

Un primer exemple d'aquest mètode és el següent:

```
finestraUoc = open("http://www.uoc.edu", "UOC", "height=300, width=200");
```

De la mateixa manera que es pot obrir una finestra, es pot tancar utilitzant el mètode close():

```
finestraUoc.close();
```

Aquest mètode, però, només pot tancar finestres que s'hagin creat abans des del mateix àmbit i, en alguns navegadors, per polítiques de seguretat, no es pot tancar la finestra principal.

La llista d'**opcions** del mètode és molt àmplia; tot seguit, es presenten les més "interessants":

- alwaysLowered: valor booleà que indica que la finestra nova quedarà per sota de la resta de finestres.
- alwaysRaised: valor booleà que indica que la finestra nova quedarà per sobre de la resta de finestres.
- dependent: valor booleà que indica que la finestra creada és dependent de la finestra pare. Això implica que, quan es tanca una finestra, es tanquen totes les que en depenen.
- directories: valor booleà que indica que la finestra nova contindrà els botons de directori estàndard.
- height: valor en píxels que especifica l'alçària de la finestra.
- hotkeys: valor booleà que indica si les tecles ràpides del navegador estan habilitades en la finestra nova.
- innerHeight: valor en píxels que especifica l'alçària del contingut de la finestra.
- innerWidth: valor en píxels que especifica l'amplària del contingut de la finestra.
- left: valor en píxels que especifica la coordenada *x* en què apareixerà la finestra nova.
- location: valor booleà que indica si la barra d'adreces s'ha de mostrar a la finestra.
- menubar: valor booleà que indica si la finestra nova contindrà la barra de menú.

- outerHeight: valor en píxels que especifica la dimensió vertical de les vores de la finestra nova.

- resizable: valor booleà que indica si l'usuari podrà canviar la mida de la finestra nova.

- scrollbars: valor booleà que indica si la finestra nova contindrà les barres de scroll quan el document superi els límits de la finestra.

- status: valor booleà que indica si la finestra nova contindrà la barra d'estat.

- titlebar: valor booleà que indica si la finestra nova contindrà la barra de títol.

- toolbar: valor booleà que indica si la finestra nova contindrà la barra d'eines estàndard.

- top: valor en píxels que especifica la coordenada *y* en què apareixerà la finestra nova.

- width: valor en píxels que especifica l'amplària de la finestra.

- z-lock: valor booleà que especifica si no es podrà canviar l'ordre de pila relatiu a altres finestres fins i tot si aquesta obté el focus.

L'exemple següent obre una finestra amb les característiques que es fan servir més comunament:

```
function obrir(){
       url= "www.uoc.edu.htm";
       caract = "left=50, top=50, status=yes, menubar=yes, toolbar=no,
       width=590, height=250, directory=no, resize=no, scrollbars=yes";
       return = window.open(url,"Exemple obertura",caract);
}
```

3.1.2. Caixes de diàleg

Hi ha **tres mètodes** que creen les clàssiques caixes de diàleg o missatges en finestres modals:

- alert(): crea una finestra amb un missatge i el botó Acceptar que tanca la finestra.

- confirm(): crea una finestra que mostra un missatge i espera que l'usuari respongui prement el botó Acceptar o Cancel·lar. Aquest missatge torna el valor true si s'ha pitjat el botó Acceptar i false si s'ha pitjat Cancel·lar.

- prompt(): crea una finestra que mostra un missatge i sol·licita dades a l'usuari a partir d'un camp de text. Igual que confirm(), té els botons Acceptar i Cancel·lar: si es prem Cancel·lar, el mètode torna el valor null; si es prem Acceptar, torna el contingut del camp de text que ha introduït l'usuari.

L'exemple següent mostra un cas d'ús dels tres mètodes:

```
var entrada = prompt("Entra una dada: ", 0);
alert("La dada introduïda és: " + entrada);
var resposta = confirm("Vols tancar la finestra?");
if (resposta==true) window.close();
```

El codi anterior s'interpreta perfectament si ens fixem en el resultat en les figures següents:

Figura 11. Caixa de diàleg prompt

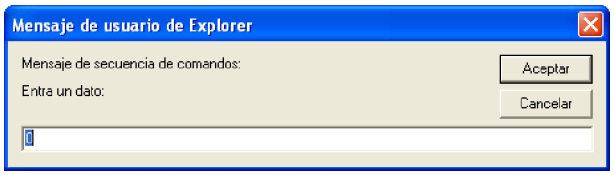


Figura 12. Caixa de diàleg alert

Figura 13. Caixa de diàleg confirm

3.1.3. Els mètodes setTimeout i clearTimeout

En l'exemple següent es crea un rellotge, per a la qual cosa es fan servir els mètodes setTimeout() i clearTimeout():

```
var timerID = null;
var timerRunning = false;
function stopRellotge(){
```

```
        if(timerRunning)
                clearTimeout(timerID);
        timerRunning = false;
}
function startRellotge(){
        stopRellotge();//S'ha d'assegurar que el rellotge és parat
        veure();
}
function veure(){
        var now = new Date();
        var hours = now.getHours();
        var minutes = now.getMinutes();
        var seconds = now.getSeconds();
        var timeValue = "" + ((hours > 12) ? hours - 12 : hours);
        timeValue += ((minutes < 10) ? ":0" : ":") + minutes;
        timeValue += ((seconds < 10) ? ":0" : ":") + seconds;
        timeValue += (hours >= 12) ? " pm" : " am";
        document.rellotge.face.value = timeValue;
        timerID = setTimeout("veure()",1000);
        timerRunning = true;
}
```

El document HTML només haurà d'implementar el següent:

- Fer una crida a la funció startRellotge().
- Tenir un formulari i un camp de text que es diguin rellotge i face, respectivament.

La base del funcionament de rellotge està en la crida recursiva cada segon de la funció veure(); d'aquesta manera, cada segon es va actualitzant la informació de l'hora en el camp de text del formulari.

3.1.4. El mètode moveBy

En l'exemple següent es pot veure l'ús dels mètodes moveBy i setInterval d'una manera coordinada, cosa que provoca el desplaçament per la pantalla de la finestra que s'acaba de crear:

```
var v1;
function obrir(){
        var caract = "left=200, top=10, width=400, height=210, statusbar=no,
        menubar=no, directory=no, resize=no, scrollbars=no";
        v1=window.open("Moviment.html","Finestra mòbil",caract);
        ini_moure()
}
```

```
function ini_moure(){
       setInterval('moure()',500);
}

function moure(){
       v1.moveBy(5,10);
}
```

La crida a la funció obrir() s'ha de fer des de la pàgina HTML, ja que en primer lloc crea una finestra nova i a continuació crida la funció que inicia el moviment.

3.1.5. Manipular marcs

Cada marc que es mostra a la pantalla és una finestra que es pot manipular. De fet, quan una finestra conté diversos marcs, es pot accedir a cadascun d'aquests marcs de manera separada utilitzant l'array window.frames[ ].

Com que un marc és una instància de l'objecte Window, comparteix les propietats i els mètodes d'aquest objecte, però s'han de remarcar algunes petites especificitats en el cas dels marcs:

- Per a la finestra del navegador, les propietats top i parent es refereixen a la finestra mateixa. Tanmateix, en un marc, la propietat top es refereix a la finestra del navegador i la propietat parent, a la finestra que el conté.

- El mètode close no es pot fer servir per a tancar un marc.

- Si l'etiqueta `frame` conté els atributs NAME i SRC, la resta de marcs han de fer referència a aquest marc usant parent.nom_marc o parent.frames[index]. Per exemple, si el marc és el quart que hi ha a la pàgina i es diu f1, des dels altres marcs s'hi fa referència posant parent.f1 o parent.frames[3].

El problema principal de manipular marcs en JavaScript és identificar de manera correcta els noms i les relacions entre els diversos marcs. Per tant, les recomanacions per a l'ús adequat de marcs són les següents:

- Identificar correctament cada marc amb l'id adequat, de manera que s'hi pugui accedir a partir del nom que l'identifica.

- En cas de dubte, es pot utilitzar la propietat length sobre l'array frames[ ] de Window per a comprovar el nombre de marcs que té una finestra en particular.

- Utilitzar top per a accedir a la finestra superior, de la qual hereten la resta en cas de dubte, i parent quan és clar que es vol accedir al nivell immediatament superior.

D'aquesta manera, si es té una finestra formada per un frameset de tres marcs (frame1, frame2 i frame3), s'hi pot accedir des de qualsevol d'aquests marcs amb la sintaxi següent:

```
window.top.frame2;
```

Hi és equivalent (perquè només hi ha un nivell de frames) la sintaxi següent:

```
window.parent.frame2;
```

3.2. L'objecte Location

L'objecte Location proporciona accés a l'adreça URL i a porcions de l'URL de manera estructurada. Assignar una cadena a un objecte Location fa que el navegador analitzi la cadena com una adreça d'Internet, actualitzi les propietats de l'objecte i estableixi la cadena com la propietat href de l'objecte.

Una adreça URL té l'estructura següent:

```
protocol//host:port/pathname#hash?search
```

Per exemple:

```
http://www.uoc.edu:8080/assist/extensions.html#topic1?x=7&y=2
```

El significat de cadascuna de les parts que componen l'URL és el següent:

- protocol: representa el començament de l'URL, és a dir, el protocol web que inclou els dos punts (http:);
- host: representa el domini o l'adreça IP de l'ordinador amfitrió a la Xarxa (www.uoc.edu);
- port: representa el port de comunicació que usa el servidor per a les comunicacions (8080);
- pathname: representa la ruta del document (assist/extensions/);
- hash: representa l'àncora en l'URL. S'hi inclou el signe # (només per a http);

- search: representa la informació per a una cerca (query). Inclou el signe ?, que n'indica el començament (només per a http). Cada element de la cerca és compost pel nom de la variable i el valor que té, i cada element se separa del següent pel signe &.

Quan s'assigna un valor a la propietat location d'un objecte, JavaScript crea un objecte Location i assigna aquest valor a la propietat href de l'objecte Location. Per tant, les dues formes que es mostren tot seguit són equivalents:

```
window.location.href="http://www.uoc.es";
window.location="http://www.uoc.edu";
```

L'objecte Location hereta directament de l'objecte Window. Si es fa referència a l'objecte Location, sense especificar una finestra, la informació d'aquest objecte s'associarà a la finestra actual.

Tot seguit es mostren en una taula les propietats principals de l'objecte Location:

Taula 12. Propietats de l'objecte Location

Nom	Descripció	Exemple
hash	Nom d'àncora en l'URL.	alert(window.**location.hash**)
host	Nom del servidor o del domini que forma part del paràmetre hostname.	window.**location.host** = "www.uoc.edu"
hostname	Conté el nom complet del servidor, l'ordinador central i el port.	window.**location.hostname** = "www.uoc.edu:8080"
href	Especifica l'URL complet.	window.**location.href**="http://www.uoc.es/"
pathname	Especifica la ruta del document.	window.**location.pathname** = "/js/exemples/rellotge.html"
port	Port de comunicacions que usa el servidor.	window.**location.port** =80
protocol	Començament de l'URL que especifica el mode d'accés.	window.**location.protocol** = "http:"
search	Especifica la cerca en l'URL.	window.**location.search** = "?cp=2&scp=4"

La taula següent mostra els mètodes principals de l'objecte Location:

Taula 13. Mètodes de l'objecte Location

Nom	Descripció	Sintaxi	Paràmetres
assign	És equivalent a modificar l'URL mitjançant location.href.	assign(url)	url: cadena de text
reload	Actualitza el document.	reload([forceGet])	Opcional. Si és true, força la càrrega del document amb el mètode GET.

Nom	Descripció	Sintaxi	Paràmetres
replace	Carrega l'adreça URL especificada a la finestra i substitueix la que hi havia.	replace(url)	url: cadena de text

Respecte als mètodes anteriors, és important tenir en compte que l'URL d'una pàgina carregada amb el mètode replace substitueix l'anterior en l'historial de pàgines visitades, de manera que quan es reemplaça una pàgina la referència a aquesta pàgina se substitueix en l'historial.

En l'exemple següent, es pot veure l'ús de propietats i mètodes d'aquest objecte:

```
<html>
<head>
<title>Ús de l'objecte location</title>
<script type="text/javascript">
<!-
function mostrarImatge(img) {
       document.images[0].src=img;
}
//->
</script>
</head>
<body bgcolor="#FFFFFF">
<form name="elmeuForm">
<p><b><font face="Arial, Helvetica, sans-serif">Triï un regal:</font></b><font
face="Arial, Helvetica, sans-serif"><br><br>
<INPUT type="radio" name="imatge" value="imatge1" CHECKED
onClick="mostrarImatge('img/regal1.gif')"> Verd <br>
<INPUT type ="radio" name ="imatge" value ="imatge2"
onClick="mostrarImatge('img/regal2.gif')"> Groc <br>
<INPUT type ="radio" name ="imatge" value ="imatge3"
onClick="mostrarImatge('img/regal3.gif')"> Vermell </font></p><p><br>
<Img name="lameva_imatge" SRC="img/regal1.gif" align="center"></p><p>
<INPUT type ="button" value="Actualitzar"
onClick="window.location.reload()"><p>
<INPUT type="button" value="Veure regal"
onClick="window.location.replace('regal.htm')" name="button">
</form>
</body>
</html>
```

En l'exemple anterior, s'ha inserit un botó al formulari que fa una recàrrega de la pàgina HTML a partir del mètode reload() i un botó nou, Veure regal, que substitueix la pàgina carregada per una de nova, regal.htm, que se suposa que ha de mostrar el regal que s'ha triat.

L'exemple anterior necessita tenir creada la pàgina regal.htm i les imatges regal1.gif, regal2.gif i regal3.gif en una subcarpeta amb el nom img.

3.3. L'objecte History

El navegador emmagatzema una array amb les adreces web visitades en l'objecte History, de manera que aquest objecte proporciona propietats i mètodes que permetran accedir a aquestes adreces des de JavaScript i visitar aquests webs.

A l'objecte History, s'hi accedeix amb la propietat History de Window (window.history), ja que és un objecte que hereta directament de Window.

Tal com s'ha dit, manté l'historial de pàgines visitades, guarda les adreces URL en una array d'objectes i, per tant, si s'accedeix a una posició de l'array, per exemple history[2], s'obté l'adreça URL visitada en tercer lloc (la primera posició d'una array en JavaScript és la 0). S'ha de tenir en compte que, per motius de seguretat, pot ser que hi hagi restringides per defecte algunes propietats en alguns navegadors.

Les propietats principals de l'objecte són les següents:

Taula 14. Propietats de l'objecte History

Nom	Descripció	Exemple

current	String que conté l'URL actual de l'historial. Només és de lectura.	if (history.current.indexOf(uoc.es) != -1)
		{
		lamevaFuncio(history.current)
		}

length	Nombre d'elements de l'array. Només és de lectura.	alert(history.length)
next	URL de la següent entrada en l'historial. Només és de lectura.	history.next
previous	URL de l'entrada anterior en l'historial. Només és de lectura.	history.previous

Els mètodes principals de l'objecte són els següents:

Taula 15. Mètodes de l'objecte History

Nom	Descripció	Sintaxi	Paràmetres
back	Carrega l'URL anterior de l'historial.	back()	
forward	Carrega l'URL següent de l'historial.	forward()	
go	Carrega un URL de l'historial.	go(pos) go(url)	pos: enter que representa la posició relativa en l'historial. url: string que representa un url de l'historial.

Respecte als mètodes anteriors, s'han de tenir en compte els detalls següents:

- El mètode back té el mateix efecte que fer servir el mètode go de la forma: history.go(-1).

- El mètode forward té el mateix efecte que fer servir el mètode go de la forma: history.go(1).

- La forma del mètode go, go(0) té l'efecte d'actualitzar la pàgina actual.

Tot seguit es presenta un exemple en què es veu l'ús dels mètodes de l'objecte:

```
<html>
<head>
</head>
<body leftmargin="0" topmargin="0" marginwidth="0" marginheight="0">
<table width="100%" border="0" cellspacing="0" cellpadding="0">
    <tr>
    <td bgcolor="#000000">
    <div align="center"><a href="javascript:history.back()"><b><font
    color="#FFFFFF" face="Arial, Helvetica, sans-serif">Anterior</font></b></a></div>
    </td>
    <td bgcolor="#000000">
    <div align="center"><a href="javascript:history.forward()"><b><font
    color="#FFFFFF" face="Arial, Helvetica, sans-serif">Siguiente</font></b></a></div>
    </td>
    <td bgcolor="#000000">
    <div align="center"><a href="javascript:history.go(0)"><b><font
    color="#FFFFFF" face="Arial, Helvetica, sans-serif">Actualizar</font></b></a></div>
    </td>
    </tr>
</table>
<p align="center"><font face="Arial, Helvetica, sans-serif" size="6">Página
    1</font>
<div align="center">
    <font face="Arial, Helvetica, sans-serif"><a href="http://www.uoc.edu">Página
    2</a></font>
    <a href="http://www.google.com"><font face="Arial, Helvetica, sans-serif">Página
    3</font></a>
</div>
<p align="center"> </p>
</body>
</html>
```

3.4. L'objecte Screen

L'objecte Screen conté propietats només de lectura que subministren informació de la pantalla de l'usuari. Es tracta d'un objecte fill de Window.

Tot seguit es presenten les propietats principals de l'objecte:

Taula 16. Propietats de l'objecte Screen

Nom	Descripció
availHeight	Alçària en píxels de la pantalla sense comptar les utilitats possibles que mostri el sistema en pantalla, com les barres d'eines.
availWidth	Amplària en píxels de la pantalla sense comptar les utilitats possibles que mostri el sistema en pantalla, com les barres d'eines.
colorDepth	Si hi ha una paleta en ús, indica la profunditat de color en bits per píxel; en un altre cas, deriva de screen.pixelDepth.
height	Alçària de la pantalla en píxels.
pixelDepth	Resolució de pantalla en bits per píxel.
width	Amplària de la pantalla en píxels.

L'objecte Screen no té mètodes.

3.5. L'objecte Navigator

L'objecte Navigator conté la informació del navegador que es fa servir. Aquestes propietats s'utilitzen generalment per a detectar el navegador, encara que també proporcionen un seguit de detalls sobre la configuració de l'usuari, com l'idioma de preferència i el sistema operatiu.

Aquest objecte és descendent directe de l'objecte Window i totes les propietats que té són només de lectura. Tot seguit es poden consultar les propietats principals de l'objecte en la taula següent:

Taula 17. Propietats de l'objecte Navigator

Nom	Descripció	Exemple
appCodeName	Nom del codi del navegador	document.write("El codi del seu navegador és" + navigator.appCodeName)
appName	Nom del navegador	document.write("El nom del seu navegador és" + navigator.appName)
appVersion	Versió del navegador	document.write("La versió del seu navegador és" + navigator.appVersion)
cookieEnabled	Indica si el navegador té actives les galetes.	if (navigator.cookieEnabled) alert("Les galetes estan activades en el navegador")
mimeTypes	Array de tots els tipus MIME que suporta el navegador.	
platform	Sistema operatiu sobre el qual s'executa el navegador.	alert(navigator.platform) Podria mostrar dades com: Win32, Win16, Mac68k, MacPPC...
plugins	Array de connectors instal·lats en el navegador.	
userAgent	Valor de la capçalera user-agent enviada en el protocol HTTP, del client al servidor.	document.write("La capçalera user-agent enviada en el protocol HTTP és" + navigator.userAgent)

Nom	Descripció	Exemple
vendor	Cadena que conté informació de la marca comercial del navegador.	

La taula següent mostra el mètode més usat dels que hi ha disponibles en l'objecte:

Taula 18. Mètodes de l'objecte Navigator

Nom	Descripció	Sintaxi	Retorn
javaEnabled	Verifica si l'opció Java està activada.	javaEnabled()	True si Java està activat i False si no ho està.

En l'exemple següent es mostra l'ús de les propietats de l'objecte:

```
<html>
<head>
</head>
<body bgcolor="#FFFFFF">
<h1>Propietats del navegador</h1>
<script type="text/javascript">
        document.write("Nom codi: <B>" + navigator.appCodeName + "</b><br>")
        document.write("Nom: <B>" + navigator.appName + "</b><br>")
        document.write("Versió: <B>" + navigator.appVersion + "</b><br>")
        document.write("Idioma: <B>" + navigator.language + "</b><br>")
        document.write("Tipus MIME: <B>" + navigator.mimeTypes + "</b><br>")
        document.write("Plataforma: <B>" + navigator.platform + "</b><br>")
        document.write("Connectors: <B>" + navigator.plugins + "</b><br>")
        document.write("Capçalera URL: <B>" + navigator.userAgent + "</b><br>")
</script>
</body>
</html>
```

3.6. L'objecte MimeType

L'objecte MimeType (*multipart Internet mail extension*) representa el tipus MIME que suporta el client. S'hi pot accedir amb l'array MimeType de l'objecte Navigator o des de l'objecte Plugin:

```
navigator.mimeTypes[index]
```

D'aquesta manera, el programador pot consultar si un tipus MIME en particular està suportat i, si ho està, extreure informació de l'objecte Plugin que el controla. Les propietats d'aquest objecte són només de lectura i es presenten en la taula següent:

Taula 19. Propietats de l'objecte MimeType

Nom	Descripció	Exemple
description	Descripció del tipus MIME.	navigator.mimeTypes["image/jpeg"].description El resultat és: JPEG Image
enabledPlugin	Referència a l'objecte Plugin configurat pel tipus MIME. Si no n'hi ha cap, la propietat val null.	navigator.mimeTypes[image/jpeg].enabledPlugins
suffixes	String que conté la llista d'extensions que accepta el tipus MIME.	navigator.mimeTypes[image/jpeg].suffixes El resultat és: jpeg, jpg, jpe, jfif, pjpeg, pjp
type	Nom del tipus MIME.	navigator.mimeTypes[image/jpeg].type El resultat és: image/jpeg

L'exemple següent mostra les propietats per a cada objecte MimeType del client:

```
<html>
<head>
</head>
<body bgcolor="#FFFFFF">
<b>Propietats de cada objecte mimeType del client:</b><p>
<script type="text/javascript">
document.writeln("<TABLE BORDER=1><TR VALIGN=TOP>"+  "<TH ALIGN=left>i"+  "<TH ALIGN=left>type"+
"<TH ALIGN=left>description"+  "<TH ALIGN=left>suffixes"+ "<TH ALIGN=left>enabledPlugin.name</TR>")
for (i=0 i < navigator.mimeTypes.length i++) {
  document.writeln("<TR VALIGN=TOP><td tipus='fuoc'>"+i+
"<td tipus='fuoc'>"+navigator.mimeTypes[i].type+
"<td tipus='fuoc'>"+navigator.mimeTypes[i].description+
"<td tipus='fuoc'>"+navigator.mimeTypes[i].suffixes)
  if (navigator.mimeTypes[i].enabledPlugin==null) {
   document.writeln(   "<td tipus='fuoc'>None"+   "</TR>")
  } else {
    document.writeln(    "<td tipus='fuoc'>"+navigator.mimeTypes[i].enabledPlugin.name+    "</TR>")
  }
}
document.writeln("</TABLE>")

</script>
</body>
</html>
```

3.7. L'objecte Plugin

Cada objecte Plugin correspon a un component instal·lat al navegador. Aquests objectes estan disponibles mitjançant la propietat enabledPlugin d'objectes MimeType. Cada connector proporciona informació del component, com la descripció, els tipus MIME que suporta, etc.

Aquest objecte se sol utilitzar per a determinar si el navegador suporta un component connector específic i la versió d'aquest component.

Tot seguit es presenten les propietats principals de l'objecte:

Taula 20. Propietats de l'objecte Plugin

Nom	Descripció
description	Descripció del connector. Només és de lectura.
filename	Nom del connector al disc. Només és de lectura.
length	Nombre d'elements de l'array d'objectes MimeType. Només és de lectura.
name	Nom del connector. Només és de lectura.

L'exemple següent mostra les propietats per a cada connector instal·lat en el client:

```
<html>
<head>
</head>
<body bgcolor="#FFFFFF">
<b>Propietats per a cada connector installat en el client:</b><p>
<script type="text/javascript">
document.writeln("<TABLE BORDER=1><TR VALIGN=TOP>"+ "<th ALIGN=left>i"+ "<TH ALIGN=left>name"+
"<TH ALIGN=left>filename"+ "<TH ALIGN=left>description"+ "<TH ALIGN=left># of types</TR>")
for (i=0; i < navigator.plugins.length; i++) {
 document.writeln("<TR VALIGN=TOP><td tipus='fuoc'>"+i+
"<td tipus='fuoc'>",navigator.plugins[i].name+  "<td tipus='fuoc'>"+navigator.plugins[i].filename+
"<td tipus='fuoc'>"+navigator.plugins[i].description+
"<td tipus='fuoc'>"+navigator.plugins[i].length+  "</TR>")
}
document.writeln("</TABLE>")
</script>
</body>
</html>
```

4. Els objectes del DOM

En aquest apartat es baixarà un nivell, fins a arribar a l'objecte Document, de manera que seguint la nomenclatura que s'ha plantejat es pot considerar que s'estudiarà el DOM. Igual que en l'apartat anterior, la llista d'objectes presentats és dinàmica i evoluciona en paral·lel a les diverses versions dels navegadors.

4.1. L'objecte Document

L'objecte Document proporciona accés al contingut del document HTML i proporciona mètodes per a manipular-lo. Per a cada pàgina HTML es genera un objecte Document quan aquesta pàgina es carrega en una finestra; d'aquesta manera, tot objecte Window conté un objecte Document a què pot accedir utilitzant la propietat document. L'objecte Document es construeix a partir de l'etiqueta `body` d'HTML.

En la taula següent es mostren les propietats principals de l'objecte:

Taula 21. Propietats de l'objecte Document

Nom	Descripció	Exemple
alinkColor	Color d'un enllaç actiu (per exemple, quan es passa el ratolí per sobre de l'enllaç).	document.alinkColor="red" o equivalentment: document.alinkColor="FF0000"
anchors	Array d'objectes Anchor del document. Només és de lectura.	
applets	Array d'objectes Applet del document. Només és de lectura.	
bgColor	Color de fons del document.	document.bgColor="FFFFFF"
cookie	String que representa les galetes associades al document.	
domain	Cadena que conté el nom del domini des del qual es va carregar el document.	document.domain="mon.com"
embeds	Array de tots els objectes encastats al document. Només és de lectura.	
fgColor	Color del text del document	document.fgColor="black"
forms	Array d'objectes Form del document. Només és de lectura.	document.forms[i] o equivalentment: document.form_i
images	Array d'objectes Image del document. Només és de lectura.	document.images[i] o equivalentment: document.image_i
lastModified	String que representa la data de la darrera modifcació del document. Només és de lectura.	document.write("Aquesta pàgina ha estat actualitzada " + document.lastModified)
linkColor	Color dels enllaços.	document.linkColor="blue"

Nom	Descripció	Exemple
links	Array d'objectes Link del document. Només és de lectura.	document.links[i]
plugins	Array d'objectes Plugin del document. Només és de lectura.	document.plugins[i]

referrer	URL del document que es crida quan es prem un enllaç. Només és de lectura.	function obtenerRef() {
		return document.referrer
		}

title	String que representa el títol del document. Només és de lectura.	var finestra1 = window.open("http://www.uoc.es") var titol = finestra1.document.title
URL	String que representa l'URL complet del document. Només és de lectura.	document.write("L'URL actual és" + document.URL)
vlinkColor	Color dels enllaços visitats.	document.vlinkColor="00FFFF"

Respecte a les propietats presentades, s'han de tenir en compte els detalls següents:

- L'ordre dels elements en les arrays, per a totes les propietats que en contenen, és el mateix ordre en què apareixen aquests elements en la pàgina.

- La propietat domain només es pot modificar de manera restringida. Al començament, conté el domini del servidor web des del qual s'ha carregat la pàgina. Es pot modificar aquesta propietat, però només per un domini amb el mateix sufix. Per exemple, un script que prové de tot.mon.com pot canviar el domini per mon.com, però un script que prové de stars.com no ho pot fer. Quan s'ha canviat el valor de la propietat, no es pot tornar al valor original.

- La propietat lastModified deriva de les dades de la capçalera HTTP que ha enviat el servidor. El servidor normalment obté aquesta dada examinant la data de la darrera modificació del fitxer. Aquesta informació no ha de ser necessàriament a la capçalera. En aquest cas, JavaScript rep un 0 que visualitza la data January 1, 1970 GMT.

- La propietat referrer és buida si el servidor no proveeix la variable d'entorn que conté aquesta informació.

Tot seguit, es mostra un subconjunt dels mètodes principals que hi ha disponibles en l'objecte:

Taula 22. Mètodes de l'objecte Document

Nom	Descripció	Sintaxi	Paràmetres
close	Acaba el flux de sortida al document i mostra el contingut escrit.	close()	

Nom	Descripció	Sintaxi	Paràmetres
open	Obre el document per a l'escriptura.	open([tipusMime, "replace"])	tipusMime: string que representa el tipus del document. Per defecte, és "text/html. Replace": si s'omet aquesta paraula, el tipus MIME és text/html. En cas contrari, el document nou no s'afegeix en l'historial.
write	Escriu una expressió HTML al document.	document.write(expr1,..., exprN)	
writeln	Escriu una expressió HTML acabada en salt de línia.	writeln(expr1,..., exprN)	

Dels mètodes presentats s'han de considerar els aspectes següents:

a) El mètode close acaba la càrrega del document que havia començat amb el mètode open; quan això passa, a la barra d'estat del navegador apareix la paraula *Listo*.

b) Els valors possibles per al primer paràmetre del mètode open són els següents:

- text/html: especifica que el document conté text ASCII en format HTML;
- text/plain: especifica que el document conté text ASCII pla amb caràcters de final de línia per al text que es visualitza;
- image/gif: especifica que el document conté bits codificats que constitueixen capçaleres GIF i dades de píxel;
- image/jpeg: especifica que el document conté bits codificats que constitueixen capçaleres JPEG i dades de píxel;
- image/x-bitmap: especifica que el document conté bits codificats que constitueixen capçaleres bitmap i dades de píxel;
- Valors per a connectors específics; per exemple, application/x-director carrega el connector de Macromedia Shockwave. Aquests només són vàlids si el connector està instal·lat.

L'exemple següent mostra el procés en què es crea una finestra, sobre la qual s'escriurà text a partir dels mètodes de Document:

```
<html>
<head>
<script type="text/javascript">
var lamevaFinestra
function escriure_a_la_finestra() {
       var text = "Exemple per a l'objecte Document"
```

```
        lamevaFinestra.document.open("text/html", "replace")
        lamevaFinestra.document.write("<p>" + text)
lamevaFinestra.document.write("<p>history.length és " + lamevaFinestra.history.length)
        lamevaFinestra.document.close()
}
</script>
</head>
<body bgcolor="#FFFFFF">
<script type="text/javascript">
        lamevaFinestra=window.open("","",'toolbar=yes,scrollbars=yes,width=400,height=300')
        escriure_a_la_finestra()
</script>
</body>
</html>
```

4.2. L'objecte A, Anchor, Link i Area

En els models tradicionals hi havia un objecte diferent per a elements `<a>` que especificaven una propietat name (anomenada *Anchor*) i un altre per als que especificaven una propietat href (anomenada *Link*). Aquesta nomenclatura és antiquada, i amb el DOM estàndard ja no hi ha cap distinció, perquè es fusionen Anchor i Link.

D'altra banda, l'objecte Area correspon a un element `<area>` d'HTML (que representa una àrea de mapa d'imatge); l'accés a aquest objecte es fa a partir de l'array links[ ] de l'objecte document.

Com que es tracta d'objectes que fan referència a una etiqueta HTML, comparteixen un seguit de propietats i mètodes amb la resta d'etiquetes. Les dues taules següents mostren les més interessants i, com es pot intuir, formen part de les propietats i els mètodes de totes les etiquetes HTML:

Taula 23. Propietats d'etiquetes HTML

Nom	Descripció	Exemple
attributes[ ]	Array amb els atributs de l'element.	Area2.attributes[2]; mostra el tercer atribut si el té.
disabled	Valor lògic que indica si l'element està deshabilitat.	Disponible = Area2.disabled;
id	Cadena que conté l'identificador únic de l'element.	Ident = Area2.id;
style	Fa referència a l'objecte Style inserit en línia de l'element.	
tabIndex	Valor numèric que indica l'ordre de tabulació de l'objecte.	

A les propietats anteriors cal afegir-hi les que s'han presentat en el segon apartat d'aquest mòdul, relacionades amb el DOM estàndard (per exemple, nodeValue, nodeType, nodeName i nextSibling).

La taula següent mostra els mètodes més usats que hi ha disponibles en l'objecte:

Taula 24. Mètodes d'etiquetes HTML

Nom	Descripció	Sintaxi	Retorn
blur()	Treu el focus de l'element.	Area2.blur():	
click()	Simula un clic del ratolí a l'objecte.	Area2.click();	
focus()	Assigna el focus a l'element.	Area2.focus();	

De la mateixa manera que en les propietats, als mètodes anteriors s'hi han d'afegir els mètodes del DOM estàndard (per exemple, cloneNode(), getAttribute() i hasChildNodes()).

A les propietats i els mètodes anteriors s'hi han d'afegir els següents, que són específics dels objectes Anchor, Link i Area:

Taula 25. Propietats i mètodes específics dels objectes Anchor, Link i Area

Nom	Descripció	Exemple
hash	String començada per #, que especifica un nom d'àncora en l'URL.	Si l'URL és: http://mon.com/europa.htm#italia document.links[0].hash és: #italia
host	String que especifica el domini i el subdomini del servidor.	
hostname	String que conté el nom complet de l'ordinador central, inclosos el nom del servidor, el subdomini, el domini i el port.	
ref.	String que especifica l'URL sencer.	
pathname	String que conté la porció d'URL referent a la ruta.	
port	String que representa el port de comunicacions que usa el servidor.	
protocol	String que conté la porció d'URL referent al protocol.	
search	String que especifica una recerca continguda en l'URL.	
target	Nom de la finestra que visualitzarà el contingut de l'enllaç.	document.aindice.target="finestra1"
text	(Només per a l'objecte Area.) Text que correspon a l'etiqueta A.	

Respecte als mètodes anteriors, cal fer una atenció especial a l'hora d'usar la propietat hash, perquè si l'URL és:

```
http://mon.com/europa.htm#italia
```

i es fa l'assignació següent:

```
h = document.links[0].hash;
document.links[0].hash = h;
```

el valor que agafa la propietat és el següent:

```
##italia
```

4.3. L'objecte Image

Un objecte Image correspon a una etiqueta <img> d'HTML. Aquest objecte té propietats que permeten la manipulació dinàmica de les imatges en el document. S'accedeix a un objecte Image a partir de la col·lecció images[ ] de l'objecte document.

L'objecte Image es construeix amb els mecanismes següents:

- L'etiqueta <img> d'HTML.
- La forma new Image(ample, alt), en què els paràmetres són opcionals i indiquen el valor d'ample i alt de la imatge en píxels.

Com que es tracta d'un objecte basat en una etiqueta HTML, comparteix les propietats i els mètodes presentats en el subapartat anterior i els següents, que li són específics:

Taula 26. Propietats de l'objecte Image

Nom	Descripció	Exemple

border	String que especifica l'amplària en píxels de la vora de la imatge. Només és de lectura.	function bordeImg(imatge) {
		if (imatge.border==0)
		alert('La imatge no té vora')
		else alert('La vora de la imatge és ' +
		imatge.border)
		}

complete	Valor booleà que indica quan ha acabat de carregar la imatge el navegador. Només és de lectura.	
height	String que especifica l'alçària en píxels de la imatge. Només és de lectura.	
hspace	String que especifica l'espai en píxels entre la vora dreta i esquerra de la imatge i el text que la segueix o la precedeix. Només és de lectura.	
lowscr	String que especifica l'URL de la imatge per mostrar en versions per a baixa resolució.	

Nom	Descripció	Exemple
name	String que especifica el nom de la imatge. Només és de lectura.	
src	String que especifica l'URL de la imatge.	
vspace	String que especifica l'espai en píxels entre l'aresta superior i inferior de la imatge i el text que la segueix o la precedeix. Només és de lectura.	
width	String que especifica l'amplària en píxels de la imatge. Només és de lectura.	

Les propietats referents a l'alçària i l'amplària d'una imatge es poden establir en la creació de l'objecte Image, però no es poden modificar des de JavaScript.

4.4. L'objecte Form

L'objecte correspon a l'etiqueta `<form>` d'HTML i és fill de l'objecte document. Aquest objecte s'utilitza per a incloure camps de text, elements de selecció (botons de selecció, llistes de selecció, etc.) i botons, i l'objectiu final que té és que l'usuari compili informació per processar-la i enviar-la al servidor.

Cada formulari en un document és, en si mateix, un objecte diferent. Per a fer referència als elements d'un formulari, cal utilitzar l'atribut id.

Igual que els objectes anteriors, es tracta d'un objecte relacionat amb una etiqueta HTML, de manera que comparteix les propietats i els mètodes dels objectes HTML i a, més, es plantegen els següents:

Taula 27. Propietats de l'objecte Form

Nom	Descripció	Exemple
action	String que especifica l'URL de destinació quan s'envien les dades del formulari.	document.elmeuForm.action = " mailto://pep@jet.es"
elements	Array d'objectes corresponents als elements del formulari. Només és de lectura.	

encoding	String que especifica la codificació MIME del formulari.	function obtenirCodif() {
		return document.elmeuForm.encoding
		}

length	Nombre d'elements del formulari. Només és de lectura.	nreele = elmeuForm.elements.length

method	String que especifica com s'envia la informació del formulari al servidor. Pot tenir els valors get i post.	function obtenirMetode() {
		return document.elmeuForm.method
		}

name	String que especifica el nom del formulari.	for (var i =0; i < document.elmeuForm.elements.length; i++)
		{ document.write(document.elmeuForm.elements[i].name +
		"
		")
		}

target	String que especifica la finestra en què es visualitzarà la resposta quan s'haurà enviat el formulari.	document.elmeuForm.target = "finestra1"

Es poden fer servir els mecanismes (tradicionals) següents per a accedir als elements d'un formulari:

```
nom_formulari.nom_element.value
nom_formulari.elements[posició].value
```

Taula 28. Mètodes de l'objecte Form

Nom	Descripció	Sintaxi
reset	Simula un clic de ratolí en un botó de tipus reset.	reset()
submit	Envia el formulari.	submit()

Respecte als mètodes anteriors, s'ha d'indicar que el mètode reset restaura els valors per defecte en els elements del formulari. Per a usar-lo no fa falta haver definit un botó en el formulari sinó que es pot cridar des d'una funció.

Tot seguit es presenta un exemple en què se sol·licita a l'usuari que introdueixi els valors A o B. Si la dada introduïda és correcta, es mostra un missatge que ho indica; si no ho és, el missatge mostra els valors possibles que es poden introduir i inicialitza el formulari.

```
<html>
<head>
<script type="text/javascript">
function verificaEntrada(lletra) {
       if (lletra.value == 'A' || lletra.value == 'B') alert('Entrada correcta')
       else document.elmeuForm.reset()
}
</script>
</head>
<body bgcolor="#FFFFFF">
<form name="elmeuForm" onReset="alert('Introdueix A o B')">
Introdueix A o B:
<input type="text" name="opció" size="5" onChange=verificaEntrada(this) maxlength="1">
</form>
</body>
</html>
```

Tot seguit es plantegen tots els objectes que pertanyen a un formulari o en formen part. Com que es tracta d'objectes basats en etiquetes HTML, comparteixen les propietats i els mètodes que s'han presentat abans.

4.4.1. Els objectes Hidden, Text, Textarea i Password

Es plantegen els objectes de manera conjunta, perquè comparteixen propietats i mètodes; concretament, els objectes Hidden, Text i Password corresponen a l'etiqueta HTML `<input>`, en què la diferència que tenen és l'atribut type. D'aquesta manera, es té el següent:

- L'objecte Hidden és un camp més del formulari, però no és visible quan es visualitza la pàgina al navegador. Com que no és visible, l'usuari no en pot modificar el valor, que només es pot modificar des de la programació canviant el valor de la propietat value. S'utilitza per a enviar parells de dades (nom/valor) quan es fa un submit del formulari.

- L'objecte Text és un camp del formulari en què l'usuari pot introduir text.

- L'objecte Password és un camp del formulari en què l'usuari pot introduir text. Cada caràcter introduït es visualitza a la pantalla amb un asterisc.

Els objectes Hidden, Text i Password es construeixen amb l'etiqueta `<input>` d'HTML, amb els valors hidden, text i password en l'atribut type.

- L'objecte Textarea és un camp del formulari en què l'usuari pot introduir text en diverses línies. Per a indicar en el text introduït un salt de línia, s'ha d'especificar el caràcter concret per a fer-ho. Aquest caràcter varia segons la plataforma: a Unix és \n; a Windows és \r, i a Macintosh és \n. JavaScript comprova aquests caràcters possibles i els trasllada a la plataforma de l'usuari.

L'objecte Textarea es construeix amb l'etiqueta `<textarea>` d'HTML.

Les taules següents mostren les propietats i els mètodes específics d'aquests objectes:

Taula 29. Propietats dels objectes Hidden, Text, Textarea i Password

Nom	Descripció	Exemple
form	Referència a l'objecte form que el conté. Només és de lectura.	<input type=hidden name="any" value= "2000"> ... <input name="b1" type="button" value="Canviar any" onClick=this.form.any.value ='2001' ">
name	String que especifica el nom de l'objecte.	document.elmeuForm.elements[i].name
type	String que especifica el tipus de l'objecte (sempre és Hidden). Només és de lectura.	document.elmeuForm.elements[i].type
value	String que especifica el valor de l'objecte.	document.write("El valor del camp és" + document.elmeuForm.any.value + " ")

Taula 30. Mètodes dels objectes Hidden, Text, Textarea i Password

Nom	Descripció	Sintaxi
blur	Elimina el focus del camp de text.	blur()
focus	Assigna el focus al camp de text.	focus()
select	Selecciona l'àrea d'entrada del camp de text.	select()

Respecte als mètodes anteriors, s'ha d'indicar que el mètode select il·lumina l'àrea de text i es pot fer servir per a situar el cursor al lloc en què l'usuari ha de fer l'entrada de dades. És més còmode per a l'usuari que quan comença a introduir una dada se substitueixi directament tota la dada que ja hi ha.

4.4.2. L'objecte FileUpload, File

Aquest objecte correspon a un element `<input>` amb l'atribut type file; la missió que té és incloure un fitxer en les dades que el formulari envia al servidor.

Dins un formulari, la línia que especifica el camp fileUpload és, per exemple, la següent:

```
<input type="file" size="50" name="fileu">
```

Figura 14

Per tant, quan es prem el botó Examinar, s'obre la finestra diàleg que permet de cercar i seleccionar el fitxer que es vol adjuntar.

Tot seguit es mostren les propietats principals de l'objecte:

Taula 31. Propietats de l'objecte FileUpload, File

Nom	Descripció
form	Referència a l'objecte Form que el conté. Només és de lectura.
name	String que especifica el nom de l'objecte.
type	String que especifica el tipus de l'objecte. Només és de lectura.
value	String que especifica el valor de l'objecte.

4.4.3. Els objectes Button, Reset i Submit

Aquests objectes corresponen a un element HTML `<input>` amb l'atribut type button, reset i submit, respectivament. Els tres objectes es representen com un botó, i les característiques que tenen són les següents:

- L'objecte Button representa un botó del formulari que no té cap acció predefinida.

- L'objecte Reset representa un botó del formulari que té la particularitat de restablir els valors que tenen per defecte els elements del formulari.

- L'objecte Submit representa un botó del formulari que té la particularitat d'enviar les dades del formulari.

Tot seguit es presenten les propietats i els mètodes principals d'aquests objectes:

Taula 32. Propietats dels objectes Button, Reset i Submit

Nom	Descripció
checked	Indica si el botó està seleccionat.
form	Referència a l'objecte Form que el conté. Només és de lectura.
name	String que especifica el nom de l'objecte.
type	String que especifica el tipus de l'objecte. Només és de lectura.
value	String que especifica el valor de l'objecte.

Taula 33. Mètodes dels objectes Button, Reset i Submit

Nom	Descripció	Sintaxi
clic	Simula la pulsació del botó.	click()

4.4.4. L'objecte Radio

Aquest objecte correspon a un camp d'entrada de formulari de `<input>` amb l'atribut type radio. L'objecte representa un botó d'opció individual dins d'un grup de botons d'opció del formulari.

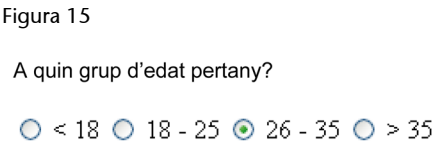
Per a entendre'n més bé l'ús, tot seguit es planteja un exemple en què hi ha un formulari que conté quatre objectes Radio dins d'un mateix grup. Això vol dir que el fet de tenir-ne un de seleccionat implica que la resta no ho estigui.

Per a indicar que pertanyen al mateix grup, el valor del paràmetre name ha de ser igual per a tots, i per a determinar quina opció estarà marcada per defecte, s'utilitza la paraula *checked* com a paràmetre.

```
A quin grup d'edat pertanys?<br><br>
<input type="radio" name="edat" value="1"> <18
<input type="radio" name="edat" value="2"> 18 - 25
<input type="radio" name="edat" value="3" checked> 26 - 35
<input type="radio" name="edat" value="4"> >35
```

El resultat a la finestra del navegador és el següent:

Figura 15



Tot seguit es presenten les propietats i els mètodes de l'objecte Radio:

Taula 34. Propietats de l'objecte Radio

Nom	Descripció	Exemple

checked	Valor booleà que especifica si el botó està seleccionat o no (true si ho està i false si no està).	if (document.elmeuForm.musica[0].checked == true) {
		alert("Hi ha seleccionada l'opció música clàssica")
		}

defaultChecked	Valor booleà que especifica l'estat per defecte del botó (true si està seleccionat per defecte i false si no ho està).	
form	Referència a l'objecte Form que el conté. Només és de lectura.	

length	Nombre d'objectes Radio en el grup.	for (i=0; i<elmeuform.edat.length; i++){
		if (elmeuform.edat[i].checked) alert(i)
		}

name	String que especifica el nom de l'objecte.	
type	String que especifica el tipus de l'objecte. Només és de lectura.	
value	String que especifica el valor associat al botó.	

Respecte a les propietats anteriors, s'ha de tenir en compte que la propietat value d'un botó d'opció no es visualitza en el document, i és el valor que s'envia amb la resta de dades del formulari al servidor.

Taula 35. Mètodes de l'objecte Radio

Nom	Descripció	Sintaxi
clic	Simula la pulsació del botó.	click()

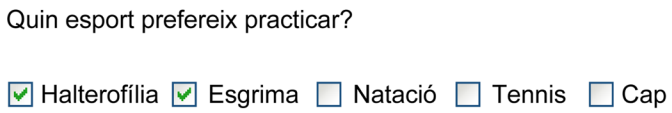
4.4.5. L'objecte Checkbox

Igual que l'objecte anterior, l'objecte Checkbox correspon a un element HTML `<input>`, amb l'atribut type a checkbox, i es representa com una casella que l'usuari pot marcar o desmarcar segons li convingui.

En l'exemple següent, el formulari conté cinc objectes Checkbox dins un mateix grup. En aquest cas, a diferència del grup d'objectes Radio, es permet seleccionar més d'una opció en paral·lel. Per a determinar quina opció quedarà marcada per defecte, s'utilitza la paraula *checked* com a paràmetre:

```
Quins esports prefereix practicar?<br><br >
<input type="checkbox" name="esports" value="1" checked> Halterofília
<input type="checkbox" name="esports" value="2" checked> Esgrima
<input type="checkbox" name="esports" value="3"> Natació
<input type="checkbox" name="esports" value="4"> Tenis
<input type="checkbox" name="esports" value="5"> Cap
```

Figura 16



Tot seguit es presenten les propietats i els mètodes principals de l'objecte Checkbox:

Taula 36. Propietats de l'objecte Checkbox

Nom	Descripció
checked	Valor booleà que especifica si el botó està seleccionat o no (true si ho està i false si no ho està).
defaultChecked	Valor booleà que especifica l'estat per defecte del botó (true si està seleccionat per defecte i false si no ho està).
form	Referència a l'objecte Form que el conté. Només és de lectura.
name	String que especifica el nom de l'objecte.
type	String que especifica el tipus de l'objecte. Només és de lectura.
value	String que especifica el valor associat al botó.

Respecte a les propietats anteriors, igual que en l'objecte Radio, la propietat value no es visualitza en el document, i és el valor que s'envia amb la resta de dades del formulari al servidor.

Taula 37. Mètodes de l'objecte Checkbox

Nom	Descripció	Sintaxi
clic	Simula la pulsació del botó.	click()

4.4.6. L'objecte Select

Aquest objecte correspon a l'element HTML `<select>`. Es tracta d'una llista de selecció del formulari, en la qual l'usuari pot seleccionar una opció o més d'una segons els atributs amb què s'ha creat.

Tot seguit es presenten les propietats i els mètodes principals de l'objecte Select:

Taula 38. Propietats de l'objecte Select

Nom	Descripció	Exemple
form	Referència a l'objecte Form que el conté. Només és de lectura.	
length	Nombre d'opcions en la llista.	nre = document.elmeuForm.llista.length
name	String que especifica el nom de l'objecte.	for (var i = 0; i < document.elmeuForm.elements.length; i++) { document.write(document.elmeuForm.elements[i].name + " ") }
options	Array que correspon a les opcions de la llista ordenades segons com apareixen.	lamevaLlista.options[1] = null
selectedIndex	Nombre que indica la posició de l'opció seleccionada.	function ObtenirIndex() { return document.elmeuForm.llista.selectedIndex }
type	String que especifica el tipus de l'objecte. Només és de lectura.	

De les propietats anteriors, és fonamental tenir en compte els detalls següents:

- Si en un objecte Select no hi ha cap element seleccionat, el valor de la propietat selectedIndex és -1.

- En cas de poder seleccionar diversos elements de la llista, la propietat selectedIndex conté la posició del primer element de la llista que està seleccionat.

- En el paràmetre type, s'especifica si en la llista es poden seleccionar diversos elements o un i prou. Els valors per a indicar-ho són select-multiple i select-one.

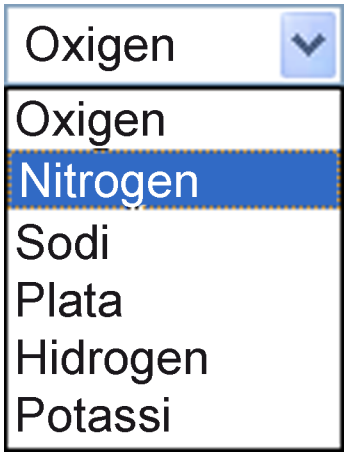
- Igual que els objectes anteriors, la propietat value no es visualitza en el document, i és el valor que s'envia amb la resta de dades del formulari.

En l'exemple següent es mostra com es construeix un objecte Select:

```
<select name="elements">
       <option value="O">Oxigen
       <option value="N">Nitrogen
       <option value="Na">Sodi
       <option value="Ag">Plata
       <option value="H">Hidrogen
       <option value="K">Potassi
</select>
```

Com es pot veure, cadascun dels elements de la llista és un objecte Option. El resultat en pantalla és el següent:

Figura 17



4.4.7. L'objecte Option

Aquest objecte correspon a una etiqueta HTML `<option>`. Representa un element en una llista de selecció del formulari.

Els valors option es poden construir utilitzant HTML o en JavaScript, tal com es descriu tot seguit:

a) Mitjançant l'etiqueta `<option>` d'HTML.

b) Mitjançant el constructor new option(text, value, defaultSelected, selected), els paràmetres del qual són opcionals i tenen el significat següent:

- text: especifica el text que es veurà en la llista;
- value: especifica el valor que s'envia amb les dades del formulari si l'opció està seleccionada;
- defaultSelected: especifica el valor inicial de l'opció: true si està seleccionat i false si no ho està;
- selected: especifica l'estat actual de l'opció, és a dir, si està seleccionada o no.

Tot seguit es presenten les propietats principals de l'objecte Option:

Taula 39. Propietats de l'objecte Option

Nom	Descripció	Exemple

defaultSelected	Valor booleà que indica si l'opció està seleccionada o no per defecte. El valor true indica que sí i el valor false, que no.	function restauraLlista() {
		for (var i = 0; i <
		document.elmeuForm.llista.length; i++) {
		if (
		document.elmeuForm.llista.options[i].defaultSelected
		== true)
		document.elmeuForm.llista.options[i].selected =
		true
		else
		document.elmeuForm.llista.options[i].selected =
		false
		}
		}

selected	Valor booleà que indica si l'opció està seleccionada. El valor true indica que sí i el valor false, que no.	function restauraLlista() {
		for (var i = 0; i <
		document.elmeuForm.llista.length; i++) {
		if (
		document.elmeuForm.llista.options[i].defaultSelected
		== true)
		document.elmeuForm.llista.options[i].selected =
		true
		else
		document.elmeuForm.llista.options[i].selected =
		false
		}
		}

text	Text que es mostra en la llista.	
value	Especifica el valor que s'envia amb les dades del formulari si l'opció està seleccionada.	

5. Gestió d'esdeveniments

Un **esdeveniment** és una acció que ocorre la majoria de vegades per alguna cosa que fa l'usuari, com, per exemple, prémer un botó o moure el ratolí. El controlador d'esdeveniments, per la seva banda, és el codi JavaScript, que s'associa a un esdeveniment, de manera que quan es dóna aquest esdeveniment, s'executa el codi.

No hi ha un sol model de gestió d'esdeveniments sinó que n'hi ha bàsicament dos que han evolucionat paral·lelament a l'evolució dels models d'objectes dels navegadors principals.

En el model tradicional del model d'objectes, tots dos navegadors oferien un suport molt bàsic per a controlar esdeveniments. Aquests es basaven pràcticament en esdeveniments generats en formularis i alguns esdeveniments de sistema o del navegador.

A partir de l'aparició de les versions 4.x dels navegadors, el model de gestió d'esdeveniments es va ampliar considerablement, i va afegir esdeveniments nous i més funcionalitat, però de la mateixa manera que els models d'objectes divergien, també ho van fer els models d'esdeveniments.

La convergència es produeix quan apareix el model d'esdeveniments estàndard del W3C. Aquest model es pot consultar en l'estàndard DOM 2, model d'esdeveniments. Tal com es veurà tot seguit, el model estàndard adquireix els avantatges de cadascun dels models anteriors i, d'aquesta manera, crea un model robust i molt complet de control d'esdeveniments.

En els subapartats següents es fa una revisió històrica del model d'esdeveniments, des del començament fins que va aparèixer el model estàndard del DOM 2 del W3C.

5.1. Model bàsic de control d'esdeveniments

El model bàsic d'esdeveniments és compost per un conjunt d'esdeveniments que es poden associar a un tipus d'objectes concret. La captura d'aquests esdeveniments es fa a partir de la vinculació que tenen amb els objectes en què es produeixen.

La taula següent mostra el conjunt bàsic d'esdeveniments disponibles en la majoria dels navegadors, i indica els elements que els admeten:

Taula 40. Conjunt bàsic d'esdeveniments

Controlador d'esdeveniment	Descripció	Objectes que l'admeten
onabort	Es produeix quan l'usuari interromp la càrrega de la imatge.	<Img>
onblur	Es produeix quan un element perd el focus.	<a>, <area>, <button>, <input>, <label>, <select>, <textarea> <applet>, , <embed>, <hr>, <img>, <marquee>, <object>, , <table>, <td>, <tr> <body> <frameset>, <ilayer>, <layer>
onchange	Es produeix quan un camp de formulari ha perdut el focus i el valor que té ha canviat.	<input>, <select>, <textarea>
onclick	Es produeix quan es pitja un objecte.	Tots els elements visibles
ondblclick	Es produeix quan es fa una doble pulsació a l'objecte.	Tots els elements visibles
onfocus	Es produeix quan un objecte rep el focus.	<a>, <area>, <button>, <input>, <label>, <select>, <textarea> <applet>, , <embed>, <hr>, <img>, <marquee>, <object>, , <table>, <td>, <tr> <body> <frameset>, <ilayer>, <layer>
onkeydown	Es produeix quan es pitja una tecla.	Tots els elements visibles
onkeypress	Es produeix quan es manté pitjada una tecla.	Tots els elements visibles
onkeyup	Es produeix quan s'allibera una tecla pitjada.	Tots els elements visibles
onload	Es produeix quan el navegador acaba de carregar una finestra o un conjunt de marcs.	<body>, <frameset> <applet>, <embed>, <link>, <script>, <style> <ilayer>, <img>, <layer>
onmousedown	Es produeix quan es pitja un botó del ratolí.	Tots els elements visibles
onmousemove	Es produeix quan es mou el ratolí mentre s'és a sobre de l'objecte.	Tots els elements visibles
onmouseout	Es produeix quan el punter surt de l'àrea d'un objecte.	Tots els elements visibles
onmouseover	Es produeix quan el punter entra en una àrea d'un objecte.	Tots els elements visibles
onmouseup	Es produeix quan s'allibera un botó pitjat del ratolí.	Tots els elements visibles
onreset	Es produeix quan es netegen els camps d'un formulari.	<form>
onselect	Es produeix quan se selecciona el contingut d'un camp de text o d'una àrea de text en un formulari.	<input>, <textarea>

Controlador d'esdeveniment	Descripció	Objectes que l'admeten
onsubmit	Es produeix quan s'envia un formulari.	<form>
onunload	Es produeix quan se surt de la pàgina.	<body>, <frameset>

A més dels esdeveniments anteriors, els fabricants han implementat un conjunt més ampli d'esdeveniments propietaris. Evidentment, la llista augmenta en cada versió nova i en aquest apartat només s'han mostrat els esdeveniments estàndard o els més importants.

Internet Explorer

Per exemple, Internet Explorer té un conjunt d'esdeveniments que li permeten capturar accions de ratolí més complexes, esdeveniments d'elements com el moviment del text de l'etiqueta `<marquee>` i esdeveniments de vinculació de dades que permeten carregar dades.

Tal com es veu en la taula, a més dels que tracten accions de ratolí, hi ha esdeveniments de navegador o de sistema com load i unload, dos esdeveniments bastant útils.

Load i unload

Per exemple, l'esdeveniment load s'utilitza per a precarregar imatges, de manera que aquestes imatges queden en la memòria cau del navegador, cosa que fa que es presentin més de pressa quan es necessiten. De la mateixa manera, l'esdeveniment unload es pot utilitzar per a netejar la memòria i, així, habilitar espai en la memòria cau del navegador.

La vinculació d'esdeveniments en el model tradicional és molt senzilla: es poden implementar controladors d'esdeveniments en les etiquetes HTML i mitjançant objectes en JavaScript. En tots dos casos, només es pot implementar el controlador si l'etiqueta o l'objecte al qual es vol aplicar admet l'esdeveniment en qüestió.

5.1.1. Vinculació en etiquetes HTML

Si s'aplica un esdeveniment a una etiqueta HTML, s'ha d'implementar un controlador d'esdeveniment a l'etiqueta. Per a implementar-lo, s'ha d'afegir un atribut a l'etiqueta que l'identificarà. La sintaxi general per a això és la següent:

```
<etiqueta controlador_d_esdeveniment="Codi JavaScript">
```

En l'exemple següent, s'implementa un controlador per a l'esdeveniment mouseover en l'etiqueta `<a>`:

```
<a href="http://www.uoc.es" onmouseover="lamevaFuncio()">
```

En l'exemple anterior, el pas del ratolí per sobre de l'enllaç fa que s'executi la funció lamevaFuncio().

Tot seguit es mostra l'exemple incloent-hi el codi HTML:

```
<html>
<head>
```

```
<script type="text/javascript">
        function la mevaFuncio() {
        alert("Enllaç a la pàgina inicial de la UOC");
        }
</script>
</head>
<body>
<a href="http://www.uoc.es" onmouseover="lamevaFuncio()">Poseu el punter sobre aquest enllaç.</a >
</body>
</html>
```

5.1.2. Vinculació mitjançant objectes en JavaScript

A partir de les versions 3 de Netscape i 4 d'Internet Explorer es va implementar la possibilitat de gestionar controladors d'esdeveniments d'un objecte amb JavaScript. L'avantatge principal d'aquest mecanisme és la separació entre l'estructura i la lògica del document de la presentació.

Quan s'implementa un controlador d'esdeveniments per a un objecte, l'objecte en qüestió adquireix una propietat, que es diu com el controlador de l'esdeveniment. Aquesta propietat nova és la que permet accedir al controlador de l'esdeveniment.

```
objecte.controlador_d_esdeveniment = funció_controladora;
```

En els exemples següents, es mostren dues maneres de respondre a l'esdeveniment load de la finestra del navegador. En el primer cas, es defineix el controlador de l'esdeveniment en l'etiqueta `<body>`, i en el segon, en l'objecte Window:

```
<html>
<head>
<script type="text/javascript">
    function direccion() {
        alert("El títol d'aquesta pàgina és: " + document.title );
    }
</script>
</head>
<body onload="direccion()">
</body>
</html>
```

En l'exemple següent, cal remarcar que el controlador es fa servir com una propietat més de l'objecte Window:

```
<html>
<head>
```

```
<script type="text/javascript">
    function direccion() {
        alert("El títol d'aquesta pàgina és: " + document.title);
    }
    window.onload = direccion; //nom del controlador en minúscules
</script>
</head>
<body>
</body>
</html>
```

5.1.3. Valors de retorn

Els controladors d'esdeveniments tenen una característica molt útil: els valors de retorn. Els controladors d'esdeveniments són funcions JavaScript i aquestes funcions poden retornar un valor amb la sentència return.

Els valors de retorn es poden utilitzar per a diversos objectius. Vegem-ne un parell d'exemples:

- Un formulari en què s'ha definit un controlador per a l'esdeveniment submit, de manera que aquest controlador valida els valors introduïts en el camp i torna true o false, depenent de si aquests valors són correctes o no. Si torna true, la tramesa del formulari es duu a terme, i si torna false, es cancel·la.

- L'esdeveniment click sobre un enllaç, de manera que si torna false la càrrega de la pàgina es cancel·la.

El codi següent és un exemple del que s'ha dit en el punt anterior:

```
<a href="http://www.uoc.edu" onclick="return confirm('Vols anar al Campus?')">Campus UOC</a >
```

La taula següent mostra els valors de retorn dels esdeveniments més útils:

Taula 41. Valors de retorn dels esdeveniments més útils

Esdeveniment	Retorn	Descripció
Click	false	Botó d'opció i casella de verificació: no s'estableix. Botó de tramesa: la tramesa del formulari es cancel·la. Botó de restablir: el formulari no es restableix. Enllaç: l'enllaç no es carrega.
Keydown	false	Cancel·la els esdeveniments keypress que continuen mentre l'usuari té pitjada la tecla.
Keypress	false	Cancel·la l'esdeveniment keypress.
Mousedown	false	Cancel·la l'acció predeterminada.

Esdeveniment	Retorn	Descripció
Submit	false	Cancel·la la tramesa del formulari.

5.2. Model HTML dinàmic de control d'esdeveniments

Els models d'esdeveniments que van sorgir a partir de les versions 4.x dels navegadors van proporcionar un objecte estàtic nou, Event. Aquest objecte es creava quan hi havia un esdeveniment i l'objecte Event recollia informació de l'esdeveniment, com la posició del ratolí, el botó del ratolí que estava pitjat i el tipus d'esdeveniment.

El que és interessant de debò és que aquest objecte es pot consultar pel controlador de l'esdeveniment i, d'aquesta manera, té una informació que descriu l'esdeveniment que ha ocorregut.

Com es pot suposar, la divergència sorgida en els models d'objectes a partir de les versions 4.x també va afectar el model d'esdeveniments, de manera que els models entre els dos navegadors eren incompatibles.

En els subapartats següents es presenten els dos models de manera independent.

5.2.1. Model d'esdeveniments de Netscape 4.x

En Netscape els esdeveniments es propagaven mitjançant la jerarquia del document: començava a la part superior i baixava fins a arribar a l'objecte en què s'havia generat l'esdeveniment.

D'aquesta manera, els objectes superiors com Window, Document i Layer podien capturar els esdeveniments abans que fossin processats per l'objecte que els havia generat. L'objectiu era simple: unificar la gestió d'esdeveniments en un nivell superior per a evitar la repetició de controladors idèntics en nivells inferiors.

L'esdeveniment click

Per exemple, l'esdeveniment click, en alguns casos, s'ha de capturar en l'àmbit de Document, perquè el que es vol és actuar quan l'usuari fa un clic a la pàgina, independentment del lloc en què el faci.

En Netscape, l'objecte Event tenia les propietats següents:

Taula 42. Propietats de l'objecte Event en Netscape

Propietat	Descripció
pageX	Conté la coordenada *X* en què ha ocorregut l'esdeveniment.
pageY	Conté la coordenada *Y* en què ha ocorregut l'esdeveniment.
screenX	Conté la coordenada *X* en què ha ocorregut l'esdeveniment respecte a la mida completa de la pantalla.
screenY	Conté la coordenada *Y* en què ha ocorregut l'esdeveniment respecte a la mida completa de la pantalla.

Propietat	Descripció
target	Conté la referència a l'objecte en què s'ha produït l'esdeveniment.
type	Conté una cadena que indica el tipus d'esdeveniment.
which	En esdeveniments de ratolí, indica el botó de ratolí que s'ha pitjat. En esdeveniments de teclat, indica el valor unicode de la tecla que s'ha pitjat.
modifiers	Conté una màscara de bits que indica les tecles de modificador que s'han pitjat durant l'esdeveniment: ALT_MASK, CONTROL_MASK, META_MASK i SHIFT_MASK.

Tot seguit se'n presenta un exemple d'ús:

```
<a href="http://www.uoc.edu">Campus UOC</a>
<script type="text/javascript">
function controla(e){
        alert("Has fet clic a: X="+ e.screenX +" Y="+ e.screenY);
}
document.links[0].onclick=controla;
</script>
```

Sobre l'exemple anterior, cal incidir en un parell de detalls:

- La funció controladora necessita un paràmetre d'entrada que representarà l'objecte Event.

- S'indica la captura de l'esdeveniment onclick de l'etiqueta `<a>` mitjançant la línia que segueix la definició de la funció, en la qual s'assigna la funció al controlador, i s'ha de tenir en compte que en la referència a aquesta funció no hi ha els típics parèntesis de crida de funció.

D'altra banda, tal com es deia al començament del subapartat, els objectes Window, Document i Layer poden capturar esdeveniments abans que arribin al seu objecte destinació. Aquesta captura s'habilita amb el mètode captureEvents() de Window i Document (l'objecte Layer utilitza el mètode del seu objecte fill Document).

Respecte al mètode anterior, s'ha de tenir en compte que ha de rebre com a paràmetre els esdeveniments que es volen capturar; la construcció del paràmetre es fa a partir de les màscares de bits següents:

Taula 43

ABORT	BLUR	CHANGE
CLICK	DBLCLICK	DRAGDROP
ERROR	FOCUS	KEYDOWN
KEYPRESS	KEYUP	LOAD

MOSE-DOWN	MOUSEMO-VE	MOUSEO-VER
MOUSEUP	MOVE	RESET
RESIZE	SELECT	SUBMIT
UNLOAD		

En l'exemple següent, l'script captura la pulsació d'una tecla i l'esdeveniment click al nivell de l'objecte Document:

```
function clickTecla(e){
        alert("Has fet clic o tecla!");
}
document.captureEvents(Event.CLICK | Event.KEYPRESS);
document.onclick=clickTecla;
document.onkeypress=clickTecla;
```

De l'exemple anterior cal tenir en compte els punts següents:

- S'utilitza una sola funció controladora per als dos esdeveniments, encara que es podria definir una funció per a cadascun d'aquests dos esdeveniments.

- El caràcter | indica un OR entre les màscares de bits, amb la qual cosa s'indica que es capturen tots dos esdeveniments.

- Les dues darreres línies assignen la funció que controlarà cadascun dels esdeveniments que s'han capturat.

Si amb la funció captureEvents() s'activa la captura d'esdeveniments, la funció releaseEvents() fa la funció contrària: desactivar una captura activada prèviament.

Per exemple, la desactivació dels esdeveniments anteriors es fa amb el codi següent:

```
document.releaseEvents(Event.CLICK | Event.KEYPRESS);
```

A vegades, pot ser que un objecte de nivell superior només vulgui capturar esdeveniments que compleixin certes condicions i, en cas contrari, passar el control als objectes de nivell inferior.

L'objecte Document

Per exemple, pot ser que l'objecte Document necessiti capturar tots els esdeveniments de Click, però que no li interessin aquells en què s'ha pitjat el botó dret del ratolí. En aquest cas, l'objecte Document captura l'esdeveniment, comprova que s'ha pitjat el botó dret i passa l'esdeveniment als objectes més petits en la jerarquia.

La funció que fa aquest pas de control és routeEvent(), que rep com a paràmetre l'esdeveniment que es processa en aquell moment. En el codi següent es pot veure l'exemple que s'ha plantejat:

```
function ratoliEsq(e) {
       if (e.type=="click" &&e.which="1"){
                alert("Click!")
       } else {
                routeEvent(e);
       }
}
document.captureEvents(Event.CLICK);
document.onclick = ratoliEsq;
```

Una funcionalitat que ofereix el mètode routeEvent() és que torna el valor que finalment processa l'esdeveniment, de manera que es pot actuar depenent del resultat que s'ha obtingut en nivells més baixos.

El mètode handleEvent() proporciona un mecanisme per a assignar directament el control d'un esdeveniment a un objecte. Aquest mètode rep en un paràmetre l'esdeveniment que es vol controlar, de manera que dins un controlador d'esdeveniments es pot passar l'esdeveniment directament a un altre objecte.

Per acabar el subapartat, és important assenyalar que les versions posteriors de Netscape van deixar de banda aquest model d'objectes, ja que van implementar el model estàndard del W3C.

5.2.2. Model d'esdeveniments d'Internet Explorer 4.x

En Internet Explorer 4.x, gairebé tots els objectes de la pàgina podien capturar esdeveniments. A més, tenia un conjunt més ampli d'esdeveniments que es podien aplicar a cada objecte.

Igual que en Netscape, quan ocorria un esdeveniment es creava un objecte Event, però en comptes de passar-lo al controlador d'esdeveniments com a paràmetre, l'objecte era global i, per tant, accessible directament des de qualsevol codi.

Les propietats de l'objecte Event en Internet Explorer es poden consultar en la taula següent:

Taula 44. Propietats de l'objecte Event en Internet Explorer

Propietat	Descripció
srcElement	Conté una referència a l'objecte a què es destina l'esdeveniment.
type	Conté una cadena que inclou el tipus d'esdeveniment.

Propietat	Descripció
clientX	Conté la coordenada *X* en què ha ocorregut l'esdeveniment.
clientY	Conté la coordenada *Y* en què ha ocorregut l'esdeveniment.
screenX	Conté la coordenada *X* en què ha ocorregut l'esdeveniment respecte a la pantalla completa.
screenY	Conté la coordenada *Y* en què ha ocorregut l'esdeveniment respecte a la pantalla completa.
button	Indica el botó del ratolí que s'ha pitjat.
keyCode	Indica el valor Unicode de la tecla que s'ha pitjat.
altKey	Conté un valor booleà que indica si s'ha pitjat la tecla Alt.
ctrlKey	Conté un valor booleà que indica si s'ha pitjat la tecla Control.
shiftKey	Conté un valor booleà que indica si s'ha pitjat la tecla Maj.
cancelBubble	Conté un valor booleà que indica si l'esdeveniment no hauria d'ascendir per la jerarquia.
returnValue	Conté un valor booleà que indica el valor de retorn del controlador.
fromElement	Conté una referència a l'element del qual es desplaça el ratolí en un esdeveniment MouseOver o MouseOut.
toElement	Conté una referència a l'element al qual es desplaça el ratolí des d'un esdeveniment MouseOver o MouseOut.

A diferència del model de Netscape, en el model d'esdeveniments d'Internet Explorer 4.x, l'esdeveniment es comença a propagar des de l'objecte en què s'ha creat i ascendeix per l'estructura de la jerarquia fins a l'objecte Window.

Tot seguit es presenta un exemple d'ús dels esdeveniments en Internet Explorer:

```
function clickTecla(){
        alert("Has pitjat la tecla" + event.keyCode);
}
document.onkeypress=clickTecla;
```

De l'exemple anterior s'han de tenir en compte els detalls següents:

- No fa falta declarar els esdeveniments que es capturen en els objectes com es feia en Netscape.

- Com que l'objecte Event es crea com a variable global, no fa falta passar-lo com a paràmetre de la funció; s'utilitza directament la variable global *event*.

Hi ha un seguit d'esdeveniments que per les característiques especials que tenen no ascendeixen per la jerarquia. En la resta d'esdeveniments es pot evitar l'ascensió assignant el valor true a la propietat cancelBubble de l'objecte Event.

Esdeveniments que no ascendeixen per la jerarquia

Un exemple d'aquests esdeveniments és la tramesa de formularis o la recepció del focus que fa un element.

Igual que a Netscape, els objectes d'Internet Explorer tenen un mètode que permet de transferir el control d'un esdeveniment fireEvent(). Aquest objecte rep dos paràmetres: l'un és el controlador que s'activarà (per exemple, onblur) i l'altre, l'esdeveniment mateix.

Per acabar, hem d'indicar que s'ha d'anul·lar l'ascensió de l'esdeveniment quan es crida el mètode fireEvent(), perquè realment es genera un esdeveniment nou, que es transfereix de manera que si en l'original no s'anul·la l'ascensió es produiria una bifurcació amb efectes no desitjats.

5.3. Model d'esdeveniments del DOM estàndard

El model d'esdeveniments del DOM2 és un híbrid entre el model de Netscape 4.x i el d'Internet Explorer 4.x. En aquest model, els esdeveniments comencen a la part superior de la jerarquia i hi descendeixen fins a arribar a l'objecte de destinació. Aquest procés es coneix com a **fase de captura** i és semblant al que ocorre en els navegadors Netscape 4.

La propagació, però, no s'ha acabat en l'objecte de destinació, sinó que quan hi sigui i el controlador (si se n'ha assignat un) acabi el tractament, comença la fase que es coneix com a *fase d'ascens*, en què l'esdeveniment implementa el camí invers fins a arribar a la part alta de la jerarquia. D'aquesta manera, un esdeveniment el pot capturar i controlar un objecte de la jerarquia, tant en la fase de captura com d'ascens.

En el DOM estàndard, l'objecte Event té un conjunt ampli de propietats que es detallen en la taula següent:

Taula 45. Propietats de l'objecte Event en el DOM estàndard

Propietat	Descripció
altKey	Conté un valor booleà que indica si s'ha pitjat la tecla Alt.
ctrlKey	Conté un valor booleà que indica si s'ha pitjat la tecla Control.
shiftKey	Conté un valor booleà que indica si s'ha pitjat la tecla Maj..
metaKey	Conté un valor booleà que indica si s'ha pitjat la tecla Meta.
bubbles	Conté un valor booleà que indica si l'esdeveniment ascendeix per la jerarquia.
button	Indica el botó del ratolí que s'ha pitjat.
cancelable	Conté un valor lògic que indica si l'esdeveniment no hauria d'ascendir per la jerarquia.
clientX	Conté la coordenada *X* en què ha ocorregut l'esdeveniment.

Propietat	Descripció
clientY	Conté la coordenada *Y* en què ha ocorregut l'esdeveniment.
screenX	Conté la coordenada *X* en què ha ocorregut l'esdeveniment respecte a la pantalla completa.
screenY	Conté la coordenada *Y* en què ha ocorregut l'esdeveniment respecte a la pantalla completa.
target	Conté una referència al node en què s'ha produït l'esdeveniment.
currentTarget	Conté una referència al node a què s'ha assignat el controlador.
relatedTarget	Conté una referència al node del qual ha sortit l'esdeveniment.
type	Conté una cadena que inclou el tipus d'esdeveniment.
eventPhase	Indica la fase del recorregut en què s'ha capturat l'esdeveniment: 1 captura, 2 en la destinació i 3 en l'ascens.

Els navegadors, per la seva banda, no compleixen l'estàndard al 100%; en realitat, tenen algunes diferències i afegeixen propietats o extensions a l'estàndard.

5.3.1. Esdeveniments de l'estàndard DOM

La taula següent mostra un subconjunt d'esdeveniments definits en el DOM2, que són bàsicament els que especifiquen HTML 4 i DOM0, més un conjunt nou d'esdeveniments relacionats amb la interfície d'usuari i la manipulació de l'estructura del document.

Taula 46. Subconjunt d'esdeveniments definits en el DOM2

Tipus esdeveniment	Esdeveniment	Ascen-deix?	Cancel·lable?
Ratolí			
	click	Sí	Sí
	mousedown	Sí	Sí
	mouseup	Sí	Sí
	mouseover	Sí	Sí
	mousemove	Sí	No
	mouseout	Sí	Sí
Navegador i HTML			
	load	No	No
	unload	No	No
	abort	Sí	No
	error	Sí	No
	select	Sí	No

Tipus esdeveniment	Esdeveniment	Ascendeix?	Cancel·lable?
	change	Sí	No
	submit	Sí	Sí
	reset	Sí	No
	focus	No	No
	blur	No	No
	resize	Sí	No
	scroll	Sí	No
D'interfície d'usuari			
	DOMFocusIn		
	DOMFocusOut		
	DOMActivate		
De canvi en la jerarquia			
	DOMSubtreeModified	Sí	No
	DOMNodeInserted	Sí	No
	DOMNodeRemoved	Sí	No
	DOMNodeRemovedFromDocument	No	No
	DOMNodeInsertedIntoDocument	No	No
	DOMAttrModified	Sí	No
	DOMCharacterDataModified	Sí	No

Com es veu en la taula, hi ha un tipus nou d'esdeveniments: **esdeveniments de canvi en la jerarquia**. Aquest tipus d'esdeveniments, tal com indica el nom, es produeixen quan es modifica l'estructura de la pàgina, ja que això es pot fer a partir dels mètodes del DOM estàndard.

La vinculació d'esdeveniments especificada en el DOM2 també és un híbrid de la vinculació especificada en els models propietaris d'Internet Explorer i Netscape. El DOM especifica que la funció controladora rebrà un paràmetre que apuntarà a l'objecte Event (tal com ocorria en Netscape).

En l'exemple següent, es veu que l'esdeveniment click es vincula a una etiqueta `<a>`, i s'utilitza la sintaxi del DOM en la localització de l'objecte:

```
<a href="http://www.uoc.edu" id="uoc">Campus UOC</a>
<script type="text/javascript">
function controla(e){
```

```
        alert("Has fet clic a: X="+ e.screenX +" Y="+ e.screenY);
}
document.getElementById("uoc").onclick=controla;
</script>
```

De l'exemple anterior, cal remarcar el següent:

- La funció controladora rep en el seu paràmetre l'objecte Event, que es pot consultar a dins de la funció.

- L'assignació de l'esdeveniment a l'objecte enllaç es fa utilitzant les propietats estàndard del DOM.

Les versions actuals d'Internet Explorer compleixen parcialment l'estàndard DOM2, de manera que hi pot haver en la gestió d'esdeveniments utilitzant l'estàndard si s'executen en aquests navegadors.

D'altra banda, el DOM proporciona un mecanisme molt interessant que permet vincular esdeveniments a objectes de la jerarquia. Aquest mecanisme aporta els avantatges següents:

- Permet especificar si l'esdeveniment es controlarà durant la fase de captura o durant la fase d'ascens, mentre que en el mètode que s'ha presentat abans l'esdeveniment es captura quan l'objecte és la destinació i durant la fase d'ascens, si és que n'hi ha per a l'esdeveniment.

- Permet vincular diversos controladors per a un mateix esdeveniment i un mateix objecte.

- Permet vincular un controlador a un node del tipus text.

Els mètodes afegeixen i eliminen el que es coneix com a **oients d'esdeveniments**, que són controladors d'esdeveniments vinculats a un node determinat de la jerarquia d'objectes, que s'activa durant una fase específica del cicle de vida de l'esdeveniment, sia la fase de captura o la d'ascens.

El mètode que afegeix un oient té la sintaxi següent:

```
node.addEventListener(tipus,controlador,sentit);
```

en què els paràmetres tenen l'objectiu següent:

- Tipus: és una cadena que indica l'esdeveniment que se sentirà.

- Controlador: és la funció que controlarà l'esdeveniment.

- Sentit: és un valor booleà que indica si es captura en la fase de captura o en la fase d'ascens.

La sintaxi que s'utilitza per al mètode removeEventListener (tipus, controlador, sentit) per a eliminar un oient que s'ha afegit abans és la mateixa que s'utilitza en el mètode anterior per a crear un oient.

A més, l'objecte Event té dos mètodes molt interessants:

- preventDefault(): cancel·la el comportament estàndard d'un esdeveniment;

- stopPropagation(): atura el flux de l'esdeveniment mitjançant la jerarquia.

En l'exemple següent, es pot veure el codi que fan servir les funcions anteriors:

```
<a href="http://www.uoc.edu" id="uoc">Campus UOC</a>
<script type="text/javascript">
function NoAltClick(e){
       if (e.altKey){
                e.preventDefault();
                e.stopPropagation();
       }
}
function MostraClick(){
       alert("S'ha fet clic");
}

document.addEventListener("clic",NoAltClick,true);
document.getElementById("uoc").addEventListener("clic",MostraClick,true);
</script>
```

De l'exemple, se n'ha de tenir en compte el detall següent: el controlador NoAltClick fa que els esdeveniments click no funcionin mentre es prem la tecla Alt. Això s'aconsegueix cancel·lant el comportament predeterminat de l'esdeveniment i aturant-ne la propagació mitjançant la jerarquia.

Per acabar, un darrer mètode disponible en tots els nodes del DOM, dispatchEvent(), que rep com a paràmetre un objecte Event i fa que el node que ha cridat el mètode es converteixi en el destinatari nou de l'esdeveniment.

Activitats

1. Creeu un script que, a partir d'una pàgina web formada per 2 paràgrafs amb 3 enllaços web, faci el següent:

a. Calculi el nombre d'enllaços que té la pàgina web.

b. Retorni l'adreça a la qual enllaça l'últim enllaç.

c. Calculi el nombre d'enllaços del primer paràgraf.

2. Creeu un script perquè, a partir d'una pàgina web amb un paràgraf que explica l'opinió sobre una recepta de cuina i un enllaç, en prémer aquest últim s'obri una caixa de text sol·licitant-ne l'opinió a l'usuari i l'annex al text anterior (similar a les opinions que s'adjunten en certs diaris web a les notícies publicades).

3. Creeu una pàgina inici.html, formada per camps que continguin el nom, cognoms, correu electrònic i DNI de l'usuari. Heu de programar la validació dels camps de la pàgina en un fitxer valida.js.

- Els controls que ha de validar el formulari són: el DNI és obligatori i numèric i la selecció de la lletra ha de ser correcta. En cas d'error, s'han de visualitzar els missatges següents segons el cas: "Completa el camp DNI", "Escriu un DNI (sense lletres, només nombres)" i "La lletra del NIF es incorrecta."
- L'algoritme per a calcular la lletra del NIF és el resultat de calcular la resta de dividir el nombre del DNI per 23 i la lletra del NIF correspon al caràcter obtingut de la cadena "TRWAGMYFP-DXBNJZSQVHLCKE" en funció del valor de la resta.
- Ha de controlar que l'adreça de correu existeix i que les dades introduïdes corresponen a una adreça de correu electrònic correcta.

4. Utilitzeu l'script explicat a la pàgina web:

http://www.desarrolloweb.com/articulos/1422.php,

per a crear una pàgina que contingui una galeria de fotos i que pugui mostrar de manera seqüencial un conjunt d'imatges.

5. Creeu una pàgina web que mostri en un camp de text les coordenades del punter del ratolí de manera dinàmica. Per a aquest objectiu s'han d'utilitzar els esdeveniments corresponents.