Introducció al DOM

Guillem Garcia Brustenga
Vicent Moncho Mas
Jordi Ustrell Garrigós

PID_00220472

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 las característiques del navegador.

BOM

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

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

No és senzill delimitar la frontera entre els dos models anteriors, ja que aspectes que teòricament formen part del BOM o del DOM apareixen barrejats i en dificulten la identificació. Això implicarà que generalment es faci referència al DOM incloent-hi els objectes relacionats amb el navegador.

Per tant, es pot definir el DOM com una **organització jeràrquica**, en la qual un objecte depèn del que es troba per damunt, i regeix els que es troben en nivells inferiors. La comunicació entre objectes depèn d'aquesta jerarquia, l'objecte que es troba en el nivell superior ha de comunicar-se amb els del nivell inferior a través dels quals es troben en els nivells intermedis.

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

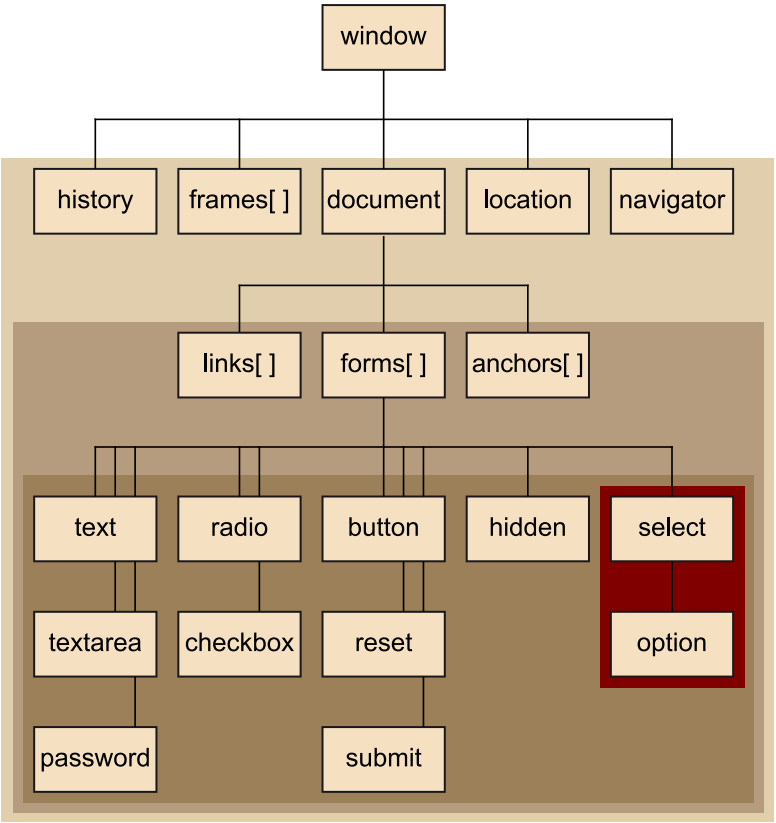
- Model tradicional d'objectes (Netscape 2 i Internet Explorer 3)
- Model tradicional estès (Netscape 3)
- Model d'objectes d'HTML dinàmic (Netscape 4 i Internet Explorer 4)
- Model d'objectes de navegador tradicional + DOM estàndard (a partir de Nestcape 6 i Internet Explorer 5)

L'objectiu d'aquest primer apartat del mòdul és la revisió històrica de l'evolució del DOM, ja que aquesta visió ajudarà entendre millor el DOM actual i com pot evolucionar en el futur. No es pretén estudiar cada objecte amb les seves propietats i mètodes, ja que això es plantejarà en el tercer apartat d'aquest mòdul.

1.1. El model tradicional

Com és sabut, un dels primers objectius del llenguatge JavaScript era la validació dels continguts dels formularis. Si es té en compte l'anterior, és 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 provocar que el primer model d'objectes fos molt similar 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 es troba l'objecte *Window*, que representa l'àrea de la finestra del navegador en la qual apareixen els documents HTML. El nivell següent en la jerarquia està format pels objectes següents:

- *Document*: és l'objecte que proporciona accés a elements HTML de la pàgina com les ancores, els enllaços i els elements dels formularis, però a més disposa d'un conjunt de propietats que defineixen l'aspecte de la pàgina, com el color del fons o del text.
- *Frames[]*: és un *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 visitades per 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, els objectes realment importants són *Document* i el seu objecte fill *Forms[]*, ja que la majoria d'elements del DOM tradicional són elements de formulari.

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

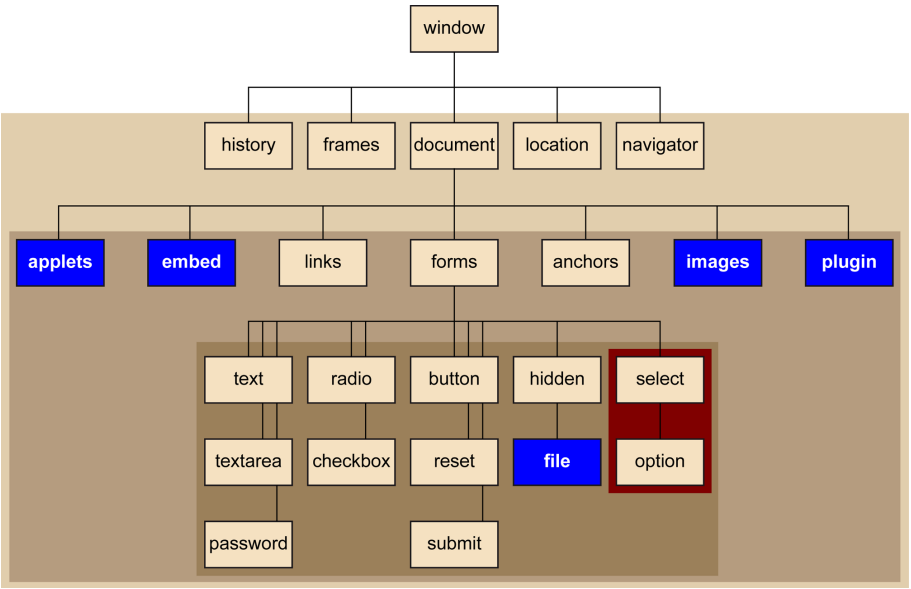
El model també incloïa la gestió d'esdeveniments, la qual cosa ens permetia executar *scripts* en resposta a certes accions de l'usuari en el document HTML, però solament a certs objectes.

1.2. Model tradicional estès

La versió 3 de Netscape introdueix un nou conjunt d'objectes de document, la qual cosa proporciona capacitat d'accés a imatges, *plugins*, miniaplicacions i objectes incrustats en 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[ ]", ja que encara que la majoria de propietats de l'objecte *Image* eren de lectura, la propietat "src", que definia la ruta de la imatge, era modificable i això va permetre 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 situava a sobre.

A continuació 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/botóOFF.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 canvia de nou la imatge.

Encara que van aparèixer més mètodes i propietats, cap no va tenir l'impacte de l'objecte *image*.

1.3. Model d'objectes d'HTML dinàmic

El model d'objectes d'HTML dinàmic es va implementar en les versions 4.0, tant de Netscape Navigator com d'Internet Explorer, però cada navegador va evolucionar de manera divergent.

Això últim va crear un buit tan important en l'ús del DOM, en cada navegador, que va derivar en un seriós problema per als programadors.

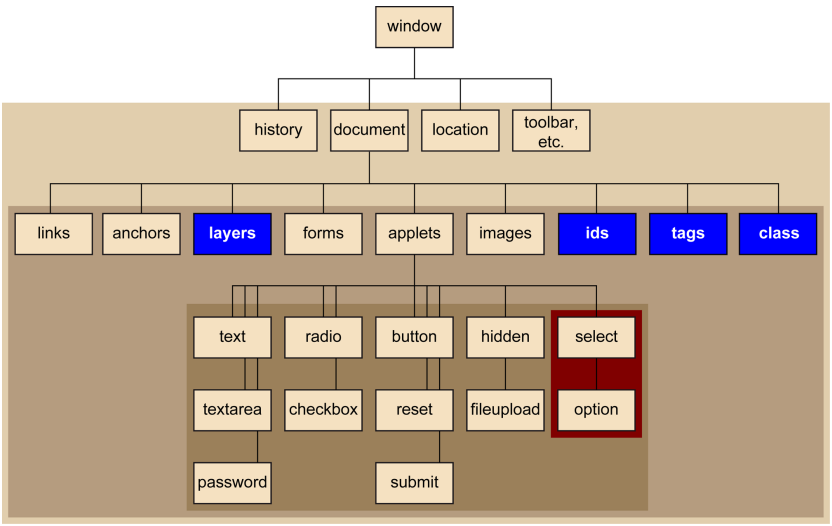
1.3.1. Netscape 4

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 nova versió 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 poden veure les novetats que apareixen en DOM:

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



Els nous objectes introduïts són:

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

Els tres primers objectes van permetre als programadors crear o manejar 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 fetque només era possible crear o modificar elements d'estil abans que mostrés en pantalla el contingut al qual s'aplicava, ja que no era possible modificar estils una vegada que els objectes ja havien estat carregats en 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, superposar-les i ocultar-les. Es podia escriure contingut nou en una capa amb el mètode *write()* i s'esperava que aquest fos una de les bases de les aplicacions dinàmiques, però com que es tractava d'una etiqueta propietària de Netscape que no es va incorporar a l'estàndard HTML de W3C, no va ser implementada per cap navegador de la competència.

També es va afegir una quantitat de propietats noves a l'objecte, que proporcionaven informació sobre la grandària de la pantalla, l'altura i l'amplària, la configuració de les barres d'eines i un conjunt nou de mètodes.

Quant al model d'esdeveniments:

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

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

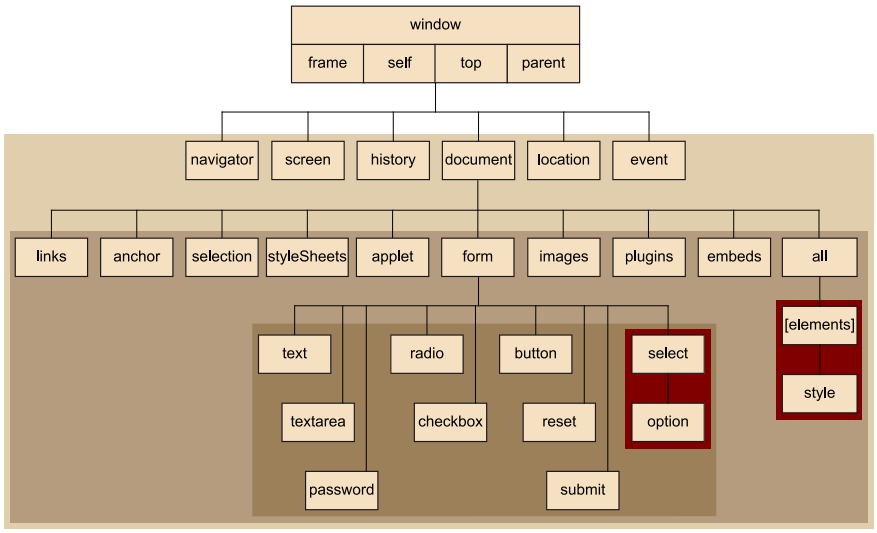
Els esdeveniments es podien capturar amb el mètode "captureEvents()", que permet tractar al "més alt nivell" un esdeveniment capturat. Aquesta tècnica podia simplificar la programació de pàgines en les quals un nombre elevat d'objectes de baix nivell, com els camps de formulari, requereix un mateix tractament davant d'un cert esdeveniment. D'aquesta manera, s'evitava repetir en cada objecte el mateix controlador d'esdeveniments.

1.3.2. Internet Explorer 4

Per la seva banda, Microsoft va llançar l'Internet Explorer 4, i va anar més enllà del proporcionat per Netscape, degut principalment al fet que 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 noves propietats de l'objecte *Document* introduïdes van ser:

- 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* format pels objectes incrustats 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[ ]", ja que 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 objete del
                        //document HTML
var element2 = document.all["Enviar"];   //element2 referencia l'objeto del
                           //document HTML con amb "Enviar"
```

Una característica interessant és que "document.all[]" situa en un mateix nivell tots els elements del document HTML (independentment de la seva posició en la jerarquia) i 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 de manera dinàmica (després que la pàgina hagués estat carregada, 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 de manera dinàmica.

Quant al model d'esdeveniments implementat, va anar totalment oposat al de Netscape 4, ja que els esdeveniments comencen en l'objecte en el qual ocorren i pugen pels objectes ascendents fins a arribar a l'objecte *Window*. D'aquesta manera, qualsevol objecte pel qual passi l'esdeveniment en el seu camí 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 comentat en els subapartats anteriors, fins ara, el problema principal és que cada proveïdor decideix quines característiques HTML presenta al programador JavaScript i quina sintaxi s'ha d'utilitzar. Aquesta situació, com era previsible, va provocar una divergència molt important en la interpretació del codi entre els diferents navegadors.

L'octubre de 1998, el World Wide Web Consortium (W3C, organització internacional que publica recomanacions per a la WWW) va publicar una especificació denominada *DOM Nivell 1*, en la qual 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 en les pàgines del web.

Per finalitzar, 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 especificat per la W3C proposa una interfície de programació d'aplicacions que posa tots els elements d'una pàgina web a la disposició del llenguatge de programació i convida a tots els proveïdors a adaptar-la en els seus navegadors amb l'objectiu clar de convergir.

Web recomanat

Les especificacions completes de cada nivell del DOM publicat pel W3C són documents públics i es poden consultar en línia o descarregar-los.

http://www.w3.org/DOM/

Una altra manera de visualitzar el DOM és classificant-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 al seu ús amb HTML, proporciona les característiques necessàries per a gestionar documents HTML i utilitza una sintaxi similar als 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 en modificar 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.

Web recomanat

Per a conèixer el compliment de l'estàndard en les diferents versions dels motors de disseny, podeu consultar la Viquipèdia, on trobareu una comparativa entre aquests.

http://en.wikipedia.org/wiki/Comparison_of_layout_engines_(Document_Object_Model)

2. El model estàndard d'objectes

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

2.1. Jerarquia d'elements

A continuació es presenta una senzilla pàgina HTML que servirà com a base de l'anàlisi i presentació posteriors dels conceptes que defineixen el DOM.

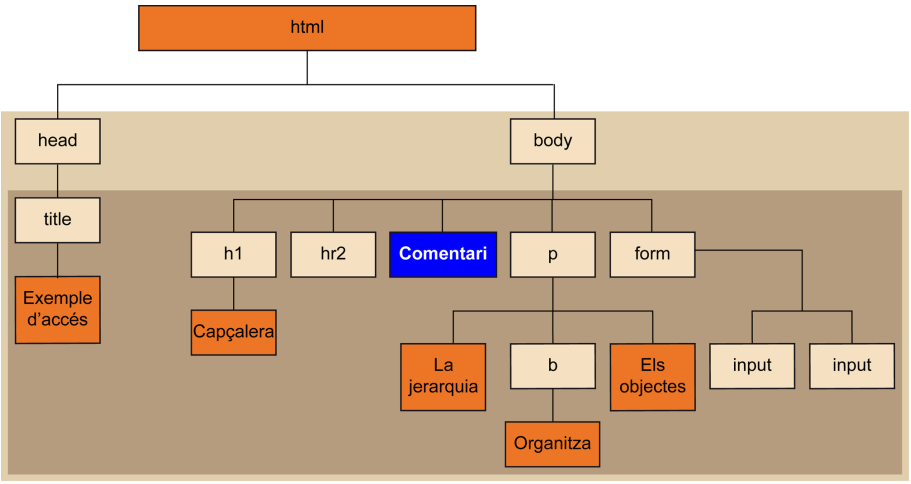
```
<html>
<head>
<!DOCTYPE HTML PUBLIC "//W3C//DTD HTML 4.01 Transitional//EN" http://www.w3.org/TR/html4/loose.dtd>
<title>Exemple d'accés</title>
</head>
<body>
<h1 id="cab">Capçalera</h1>
<hr>
<! - -Comentari - - >
<p id="p1">La jerarquia <b id="b1">organitza</b> els objectes</p>
<form name="form1" 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 fàcil d'intuir, ja que el codi HTML per si mateix segueix una certa estructura jeràrquica, és a dir:

- L'etiqueta <html> conté totes 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 què es mostren les relacions a partir de línies que uneixen les etiquetes, s'obté l'estructura següent:

Figura 5. Exemple de l'estructura d'un document HTML



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 es tenen tres tipus diferents: etiquetes, comentaris i text).

Estructura jeràrquica

Una pàgina web s'organitza de manera jeràrquica seguint l'estructura definida per les etiquetes HTML.

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

Taula 1. Tipus de nodes de l'estàndard DOM per a HTML

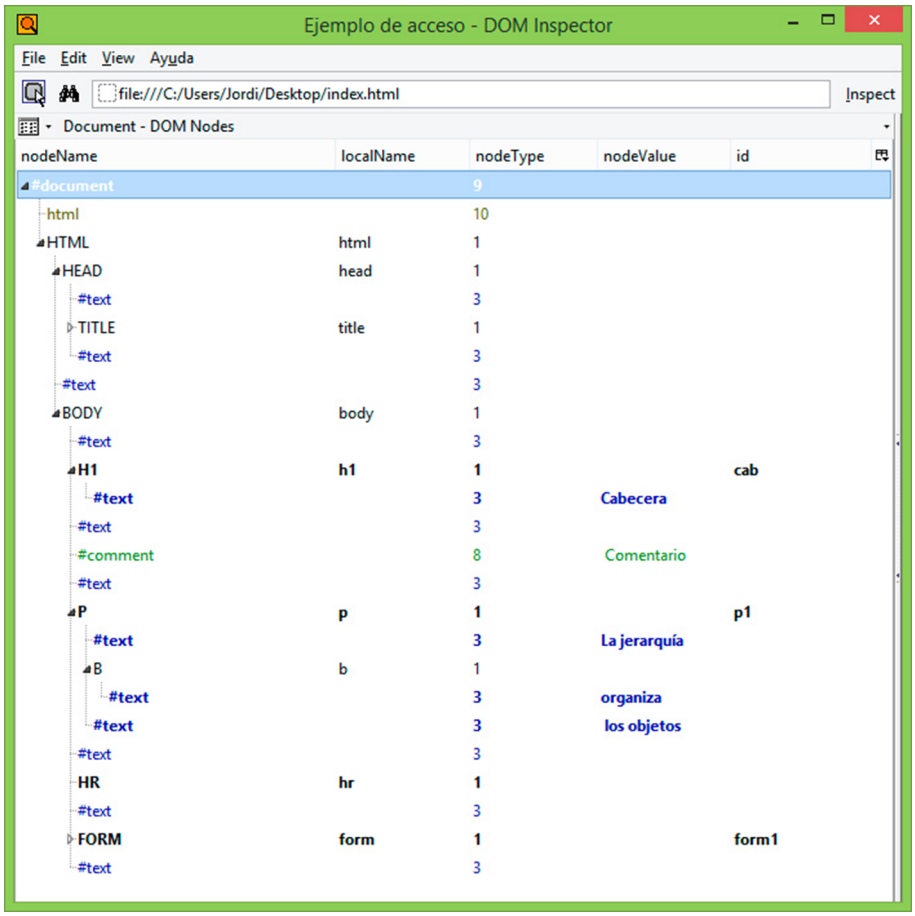
Identificador	Tipus	Descripció
1	Element	Una etiqueta HTML, per exemple <b>, <input>
2	Atribut	Un atribut d'una etiqueta HTML, per exemple align=”center”
3	Text	Un text inclòs en una etiqueta HTML, per exemple a la nostra senzilla pàgina HTML “Exemple d'accés”
8	Comentari	Un comentari HTML, per exemple: “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>

Estructura del DOM

És possible analitzar l'estructura del document a partir de certes eines que inspeccionen la pàgina HTML i que mostren la jerarquia del document des del punt de vista de l'estàndard de W3C. En el blog de l'assignatura en trobareu alguns exemples.

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

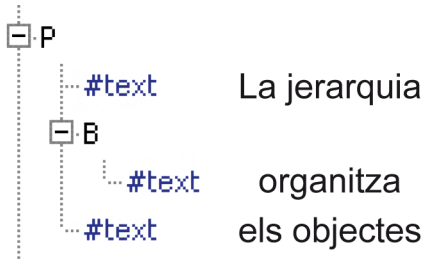
Figura 6. Exemple de l'estructura HTML amb un inspector del DOM



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

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

Figura 7. Estructura del subarbre del paràgraf de text



D'aquest subarbre s'observen les relacions següents:

- L'etiqueta P és el node pare i disposa de 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 l'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 l'últim, el node de text "els objectes".

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

Aclarida la jerarquia d'elements o de nodes que es creen a partir d'una pàgina HTML, a continuació s'estudiaran els mètodes del DOM que permetran accedir a cada node.

2.2. Accés als nodes

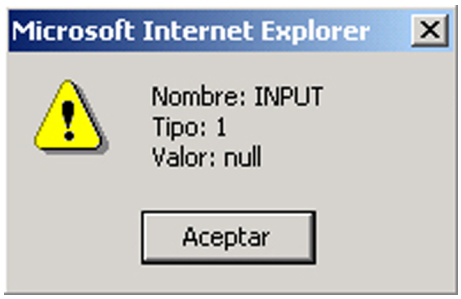
Arribats a aquest moment, ja es disposa de prou coneixements de la jerarquia que interpreta el DOM del W3C per a començar a estudiar l'accés als elements d'aquesta. En primer lloc, es presenta un senzill *script* que utilitza un primer mètode d'accés:

```
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 se li passa com a paràmetre, aquest mètode retorna el node o element i s'assigna a la variable corresponent. Es tracta d'un mètode amb una funcionalitat similar a la que va introduir IE4 amb l'*array* "all", ja que permet buscar directament un element sense navegar per tota la jerarquia.

D'aquesta manera, una vegada localitzat l'element, l'*script* 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 8



Com es pot observar, 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 s'observaven en el" DOM inspector".

L'objecte *Node* disposa d'un conjunt de propietats que, a més de descriure les seves característiques principals com el nom, valor i tipus, també mostren la seva relació amb la resta de nodes de l'estructura i permeten la navegació per l'estructura. La taula següent mostra el conjunt de propietats de l'objecte *Node*:

Taula 2. Propietats de l'objecte Node

Propietats	Descripció
nodeName	Retorna una cadena amb el nom del node.
nodeValue	Retorna una cadena amb el valor del node, solament aplicable a nodes de tipus text.
nodeType	Retorna un nombre que especifica el tipus de node.
parentNode	Retorna una referència al node pare.
firstChild	Retorna una referència al primer fill.
lastChild	Retorna una referència a l'últim fill.
previousSibling	Retorna una referència al germà gran.
nextSibling	Retorna una referència al germà petit.
childNodes	Retorna un *array* amb els nodes fills.
attributes	Retorna un *array* amb els atributs del node.
ownerDocument	Retorna l'objecte *Document* que el conté.

Mitjançant aquestes propietats és possible fer recorreguts per l'arbre a partir d'un node en concret.

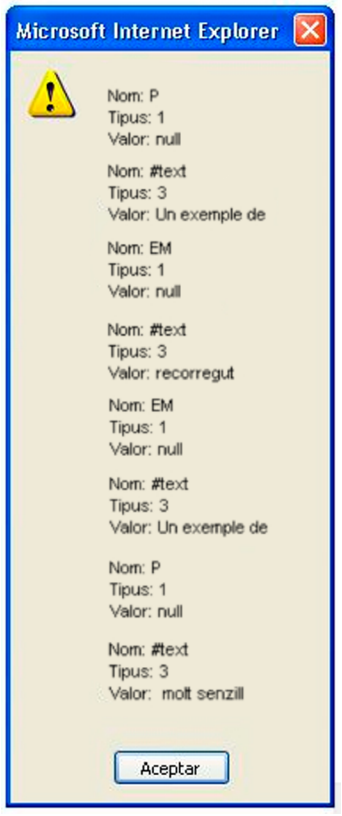
A continuació es va a plantejar un recorregut per la jerarquia a partir d'un exemple amb una estructura molt senzilla:

```
<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: "+nodo.nodeName+"\n";
   temp += "Tipus: "+nodo.nodeType+"\n";
   temp +="Valor: "+nodo.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;
msg += tipusNode(nodeActual);
nodeActual = nodeActual.previousSibling;
msg += tipusNode(nodeActual);
nodeActual = nodeActual.parentNode;
msg += tipusNode(nodeActual);
nodeActual = nodeActual.lastChild;
msg += tipusNode(nodeActual);
alert(msg);
</script>
</body>
</html>
```

El resultat de l'exemple anterior és el següent:

Figura 9. Un exemple de *recorregut* molt senzill



Però s'ha fet el recorregut per la jerarquia sense problemes, perquè es coneixia l'estructura i, per tant, les relacions entre els diferents nodes. Evidentment, aquesta situació no es dóna en la realitat, ja que no es pot conèixer *a priori* l'estructura, ni tan sols si un node disposa de fills o germans.

Per l'anterior, es podria fer una navegació completa utilitzant l'estratègia següent; el recorregut comença per un cert node X, comprovem si el node té fills. Si els té, s'accedeix al seu fill petit; si no els té, només es pot buscar un germà petit o comprovar si té un node pare.

Per a establir un recorregut com a 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;
```

A continuació 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=elementoActual.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 retorna el node, i si no n'hi ha avisa amb un missatge.

No és comú o no té gaire utilitat el recorregut d'una estructura per a la seva anàlisi, però és fonamental saber localitzar un node (és senzill amb getElementById()) i poder accedir, a més, a un dels seus "familiars directes" (pare, fill o germans) per a modificar l'estructura del document, i per a això últim 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 principals d'accés:

Taula 3. Mètodes principals d'accés de l'objecte *Document*

Mètode	Descripció
getElementById()	Retorna l'objecte l'atribut del qual "id" coincideix amb el passat com a paràmetre.
getElementsByName()	Retorna una llista amb els objectes amb el valor en l'atribut "name" que es passa com a paràmetre.
getElementsByTagName()	Retorna una llista amb els objectes l'etiqueta dels quals HTML se li passa com a paràmetre.
documentElement	Retorna l'objecte arrel, és a dir, l'etiqueta <html>.
body	Retorna 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 a tot el document. Per aquest motiu, s'ha d'utilitzar l'atribut "id" enfront del "name".

2.3. Manipulació dels nodes

Arribats a aquest punt, es disposa d'una referència a un node, i és possible moure's per la jerarquia a partir dels mètodes anteriors, que permeten passar d'un node al node pare, germà o fill. Però, 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:

- **Modificació de nodes**: modificant-ne el contingut o els atributs.
- **Inserció o eliminació de nodes**: intercalant un nou node en una certa posició en la jerarquia o eliminant un dels nodes de l'estructura.

2.3.1. Modificació de nodes

Per començar, exposarem un petit exemple amb un gran potencial, 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 com es veu en la figura 6, sobre el" DOM inspector". Aquests nodes no disposen d'un valor en la columna "id", per la qual cosa no és possible accedir-hi utilitzant el mètode getElementById().

Vegeu també

Podeu trobar 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'aquest s'accedeix al node de text mitjançant la propietat firstChild, i per acabar es modifica el text utilitzant la propietat "nodeValue".

En l'exemple següent, es modifica el text "La jerarquia" per "L'estructura" de l'exemple plantejat a l'inici de l'apartat actual:

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

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

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

Taula 4. 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 en la cadena.
splitText(*offset*)	Divideix el node de text en dues peces en l'índex donat en *offset*. Retorna la part dreta de la divisió en un nou node de text i de la part esquerra en l'original.
substringData(*offset*, *quantitat*)	Retorna una cadena corresponent a la subcadena que comença per l'índex *offset* i contiua fins a una quantitat de caràcters.

Els mètodes anteriors afegeixen la funcionalitat suficient perquè la manipulació del contingut de la pàgina HTML es gestioni de manera òptima. Una alternativa a l'ús dels mètodes anteriors és la recuperació del contingut del node de text en una cadena, la manipulació d'aquesta cadena amb els mètodes existents de l'objecte *String* utilitzant expressions regulars si fos necessari i l'assignació de nou del contingut de la cadena.

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

Per exemple, es vol introduir el valor "Introdueix text..." en un quadre de text: per a això 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* disposa de tres mètodes que permeten treballar amb els atributs dels nodes. La taula següent mostra la relació de mètodes:

Taula 5. Mètodes de l'objecte Node per a modificar nodes

Mètode	Descripció
getAttribute(*nom*)	Retorna 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. Creació i inserció de nodes

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

- Creació del nou node que es vol inserir.
- Localització de la ubicació en la jerarquia en què s'inserirà.
- Inserció del nou node en la ubicació adequada.

Creació de nodes

La creació d'un nou node és molt simple, l'objecte *Document* disposa de mètodes que permeten la creació de 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". De manera similar, es procedeix a crear un node de tipus text:

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

En l'exemple anterior, el mètode utilitzat és createTextNode en lloc de createElement.

La taula següent recopila els mètodes que s'utilitzen en la creació de nodes:

Taula 6. Mètodes per a la creació de 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(*nom*)	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 cada tipus de node, exceptuant els tipus 9 i 10, que es refereixen a les etiquetes <html> i <body> respectivament.

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

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

Vegeu 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 es clonarà, i el segon paràmetre és l'"id" del node al qual s'inserirà el clon com a fill.

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

- La variable *nodeClonat* és la referència al nou node que s'ha clonat. En passar-lo a la funció cloneNode, el paràmetre "true" clonarà no solament l'element en si, sinó també tots els seus fills.

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

- L'última línia de codi mostra un nou mètode que serà presentat a continuació, però és intuïtiu pensar que el mètode està inserint el node clonat com a fill del node *puntInsercio*.

Inserció de nodes

Tal com s'ha vist en l'exemple de l'apartat anterior, quan ja es disposa del node que es vol afegir al document, és necessari inserir-lo en algun lloc de la jerarquia del document.

La inserció d'un node en la jerarquia sempre es fa partint del node pare mitjançant els mètodes que es mostren en la taula següent:

Taula 7. Mètodes per a la inserció de 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 han cridat el mètode.

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

No obstant això, el mètode appendChild() afegeix el nou node al final de la llista dels fills i, per tant, serà el fill petit o situat més a la dreta de tots els nodes fills.

Els mètodes anteriors es poden observar en l'exemple següent:

```
var nouNode = document.getElementById("form1").cloneNode(true);
var pare = document.getElementById("cap");
padre.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 seves filles en passar com a paràmetre del mètode cloneNode el valor *true*.

- Es localitza el node pare sobre el qual s'inserirà el formulari; aquest està identificat amb l'id "cab".

- El codi finalitza amb l'addició del formulari com el més petit dels fills del node "cab", per a la qual cosa s'empra el mètode "appendChild".

Eliminació i reemplaçament de nodes

De vegades, és necessari poder eliminar nodes en l'estructura del document. Per a això, l'objecte *Node* disposa del 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 l'últim fill del node al qual fa referència la variable "nodeActual"; és important saber que, a més d'eliminar-lo, el retorna i pot ser assignat a una variable.

A més d'eliminar un node, és possible reemplaçar-ne un per un altre utilitzant el mètode replaceChild(fillNou, fillReemplaçat), en què el node “fillNou” reemplaçarà el node “fillReemplaçat”.

3. Els objectes del BOM

En aquest apartat, es presentaran els objectes al costat de les seves propietats i mètodes principals que formen part de 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*.

Web recomanat

Per a informació més detallada i actualitzada de les propietats i mètodes dels objectes del BOM podeu visitar la pàgina de w3schools

http://www.w3schools.com/jsref//

Com era previsible, la llista d'objectes que es presenta és dinàmica i evoluciona en paral·lel a l'aparició de noves versions dels navegadors. La utopia de l'estandardització encara és un objectiu pendent i no una realitat com seria desitjable.

És important destacar que aquesta part de BOM pateix el major nombre de divergències, ja que aquest està directament relacionat amb el navegador.

Per tant, l'objectiu de l'apartat és la presentació dels objectes i els seus components, que juntament amb alguns exemples ajudaran a comprendre el potencial del procés de manipulació del DOM des de JavaScript. En cap moment no es planteja que s'aprenguin "de memòria" tots els objectes amb els seus mètodes i propietats; no té cap sentit, sempre es poden consultar en l'apartat corresponent o en línia a Internet.

Quant a això últim, i per a solucionar problemes amb la compatibilitat del codi, les referències web següents mostren el model d'objectes en cadascun dels navegadors actuals, incloent-hi tant el BOM com el DOM:

- Especificació del DOM dels navegadors basat en Mozilla: https://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 es troba en el nivell més alt en la jerarquia d'objectes i, per tant, és el contenidor de tot allò que es pot veure en el navegador.

Tan aviat com el navegador s'executa, i encara que aquest no visualitzi cap document, es crea una instància de *Window* que és accessible des de JavaScript.

Poden referenciar-se totes les finestres que estan obertes en la pantalla i poden ser manipulades per una instància de l'objecte (sempre que hagin estat creades des de la pròpia instància que les intenta manipular). Tal com es veurà en aquest subapartat, les propietats d'una instància de *Window* inclouen la seva grandària, components, posició, etc. Per la seva banda, els mètodes permetran la creació i destrucció de finestres genèriques, de finestres d'alerta o de confirmació, etc.

Ha de tenir-se en compte que l'objecte *Window* pot representar la finestra del navegador o un *frame* (quan hi ha diversos *frames*, cadascun es considera un objecte *Window* independent).

Així doncs, un objecte *Window* es pot crear mitjançant alguna de les opcion següents s:

- L'etiquetes *body* o *frameset* d'HTML
- El mètode "open" de JavaScript.

Per la seva banda, l'accés a les propietats i mètodes de l'objecte *Window* segueixen 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 mateix document, pot usar-se la paraula *self* en substitució de la paraula *window*. Per exemple, per a tancar la finestra actual es poden usar indistintament les formes window.close() i self.close().

De totes maneres, amb els mètodes open() i close() és necessari usar les formes window.open() i window.close(), ja que aquells es podrien confondre amb els mètodes de l'objecte *Document* (document.open() i document.close()).

La paraula *self* pot ometre's en els casos més simples i reservar-se per als casos que impliquin diversos *frames*. Això és 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 mètodes de la finestra actual sense especificar-la:

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

A continuació es mostrarà una taula amb les propietats principals disponibles de l'objecte *Window*:

Taula 8. Propietats de l'objecte *Window*

Nom	Descripció	Exemple

closed	És un valor booleà que indica si una finestra ha estat tancada. Quan es tanca una finestra, l'objecte *Window* continua existint, però amb el valor d'aquesta propietat *true*.
	És de només 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 veu en la barra d'estat del navegador	**window.defaultStatus**="Pàgina índex"
document	Fa referència a l'objecte *Document* contingut en la finestra.	
frames	És un *array* d'objectes *frame*. Els *frames* són els generats per 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. És de només lectura.	En una pàgina que contingui dos *frames*, es pot accedir a aquests de la manera següent: parent.frames["findex"] parent.frames["fcontingut"] o bé parent.frames[0] parent.frames[1]
history	Fa referència a l'objecte *History* contingut en la finestra.	

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

location	Fa referència a l'objecte *Location* contingut en la finestra.	
name	Cadena de text que especifica el nom de la finestra o marc. És de només lectura.	finestra = window.open('doc.htm','finestra1','width=400, height=100') alert(finestra.name)
navigator	Fa referència a l'objecte *Navigator* contingut en 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 descarregat.	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 veu en la barra d'estat del navegador.	`function ini_status() {`
		`  window.status = "Cliqui en aquest enllaç per accedir a l'índex"`
		`}`
		`<a href="índex.htm" onMouseOver = "ini_status()">Índex</a>`

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()

Nom	Descripció	Exemple
window	És la finestra o el *frame* actual.	

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

Taula 9. Mètodes de l'objecte *Window*

Nom	Descripció	Sintaxi	Paràmetres	Retorn
alert	Obre un quadre 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 al mètode setInterval prèvia.	
clearTimeout	Cancel·la el *timeout* que s'ha inicialitzat amb el mètode setTimeout.	clearTimeout (intervalID)	intervalID s'inicialitza amb la crida al mètode setTimeout prèvia.	
close	Tanca la finestra especificada.	close()		
confirm	Obre un quadre 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 *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, i)	*x*: coordenada *x* *y*: coordenada *y*	
open	Obre una nova finestra del navegador.	open(URL, nom, característiques)	URL: cadena de text que especifica l'URL a visualitzar en la nova finestra. Nom: text que indica el nom de la finestra.	
print	Fa que aparegui el diàleg d'impressió.	print()		
prompt	Obre un quadre de diàleg amb un missatge i un camp d'entrada de text.	prompt(missatge, [text])	Missatge: text que mostra el quadre de diàleg. Text: opcional, text que per defecte surt en el camp d'entrada de text.	Dada introduïda per l'usuari.

Nom	Descripció	Sintaxi	Paràmetres	Retorn
resizeBy	Redimensiona la finestra movent la cantonada inferior dreta segons els valors relatius especificats.	resizeBy(horitzontal, vertical)	Horitzontal: nombre de píxels a sumar o restar a la dimensió horitzontal. Vertical: nombre de píxels a sumar o restar a la dimensió vertical.	
resizeTo	Redimensiona la finestra en els valors especificats.	resizeTo(amplada, altura)	Amplada: amplada en píxels de la finestra. Altura: altura en píxels de la finestra.	
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 a sumar o restar, per a l'*scroll* horitzontal. y: nombre de píxels a 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 a visualitzar. y: coordenada *y* en píxels, de l'àrea a visualitzar.	
setInterval	Executa una funció, cada vegada que transcorre el temps especificat i fins que s'executa el mètode clearInterval.	setInterval(expressió, 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(funció, temps, param1, ..., paramN)	Temps: nombre de mil·lisegons	Identificador

L'ús d'alguns dels mètodes anteriors necessita l'activació o desactivació de certes propietats relacionades amb la seguretat del navegador; això significa que és possible que no funcionin a causa de 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 conjunt d'atributs que en defineixen l'aspecte.

La sintaxi del mètode open és:

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

En què:

- *url* és l'adreça que indica el document que es carregarà en la finestra.
- *nom* és el nom de la finestra (que s'utilitzarà per a referenciar-la posteriorment).
- *característiques* és una cadena delimitada per comes que indica un conjunt de propietats de la finestra que es crearà.

- *reemplaçar* 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 seria:

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

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

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

Però aquest mètode només pot tancar finestres que hagin estat creades prèviament 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, a continuació es presenten les més "interessants":

- ***alwaysLowered***: valor booleà que indica que la nova finestra quedarà per sota de la resta de finestres.

- ***alwaysRaised***: valor booleà que indica que la nova finestra quedarà per sobre de la resta de finestres.

- ***dependent***: valor booleà que indica que la finestra creada és depenent de la finestra pare. Això implica que en tancar una finestra es tanquen totes les dependents d'aquesta.

- ***directories***: valor booleà que indica que la nova finestra contindrà els botons de directori estàndard.

- ***height***: valor en píxels que especifica l'altura de la finestra.

- ***hotkeys***: valor booleà que indica si les tecles ràpides del navegador estan habilitades en la nova finestra.

- ***innerHeight***: valor en píxels que especifica l'alt del contingut de la finestra.

- ***innerWidth***: valor en píxels que especifica l'amplada del contingut de la finestra.

- ***left***: valor en píxels que especifica la coordenada *x* en la qual apareixerà la nova finestra.

- ***location***: valor booleà que indica si la barra d'adreces ha de mostrar-se en la finestra.

- ***menubar***: valor booleà que indica si la nova finestra contindrà la barra de menú.

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

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

- ***scrollbars***: valor booleà que indica si la nova finestra contindrà les barres de desplaçament (o *scroll*) quan el document excedeixi els límits de la finestra.

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

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

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

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

- ***width***: valor en píxels que especifica l'amplada 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 s'usen més comunament:

```
function obrir(){
   url= "http://www.uoc.edu";
   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. Quadres de diàleg

Hi ha **tres mètodes** que creen els clàssics quadres 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 retorna el valor *true* si s'ha premut el botó Acceptar i *false* si s'ha premut 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() disposa dels botons Acceptar i Cancel·lar. En cas de pressionar Cancel·lar, el mètode retornarà el valor *null*, en cas contrari, retornarà el contingut del camp de text introduït per 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 s'observa el resultat en les figures següents:

Figura 10. Quadre de diàleg *prompt*

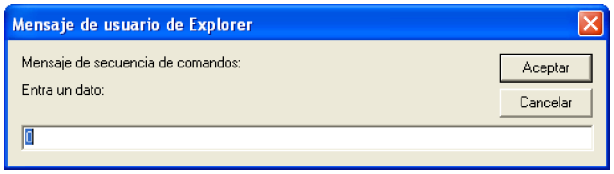


Figura 11. Quadre de diàleg *alert*

Figura 12. Quadre de diàleg *confirm*

3.1.3. Els mètodes setTimeout i clearTimeout

En l'exemple següent es crearà un rellotge, per a la qual cosa s'empraran 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
   ver();
}
function ver(){
   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) ? " P.M." : " A.M.";
   document.reloj.face.value = timeValue;
   timerID = setTimeout("ver()",1000);
   timerRunning = true;
}
```

El document HTML només necessitarà implementar el següent:

- Fer una crida a la funció "startRellotge()".
- Disposar d'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 s'actualitza la informació de l'hora en el camp de text del formulari.

3.1.4. El mètode moveBy

En l'exemple següent es pot observar l'ús dels mètodes moveBy i setInterval de manera coordinada, la qual cosa provoca el desplaçament per la pantalla de la finestra creada recentment:

```
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 nova finestra i, a continuació, crida la funció que inicia el moviment.

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* provoca 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.

Un 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 l'inici 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 el servidor fa servir per a les comunicacions (8080).

- *Pathname*: representa la ruta del document (assist/extensions/).

- *Hash*: representa l'ancora en l'URL. Inclòs el signe # (només per a http).

- *Search*: representa la informació per a una cerca (*query*). Inclou el signe ?, que indica l'inici (només per a http). Cada element de la cerca està compost pel nom de la variable i el seu valor, 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 a continuació són equivalents:

```
window.location.href="http://www.uoc.edu";
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 estarà associada a la finestra actual.

A continuació es mostren en una taula les propietats principals de l'objecte *Location*:

Taula 10. 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, el *host* i el port.	window.**location.host** = "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/examples/rellotge.html"
port	Port de comunicacions que usa el servidor.	window.location.port = 80
protocol	Inici de l'URL que especifica la forma 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 11. 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.
replace	Carrega l'URL especificat en la finestra i substitueix el que hi havia.	replace(url)	url: cadena de text

Respecte als mètodes anteriors, és important tenir en compte que l'URL d'una pàgina carregat amb el mètode *replace* substitueix l'anterior en l'historial de pàgines visitades, per la qual cosa, en reemplaçar una pàgina, la referència a aquesta és substituïda en l'historial.

En l'exemple següent es pot observar 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>
<form name="elmeuForm">
   <h2>Triï un regal:</h2>
   <p>
      <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')"> Rojo
   </p>
   <p>
      <img name="lameva_imatge" SRC="img/regal1.gif" align="center">
      <input type ="button" value="Actualitzar" onclick="window.location.reload()">
      <input type="button" value="Veure regal"
      onclick="window.location.replace('regal.htm')" name="button">
   </p>
</form>
</body>
```

```
</html>
```

En l'exemple anterior, s'ha inserit un botó en el formulari que fa una recàrrega de la pàgina HTML a partir del mètode "reload()" i un nou botó "Vegeu regal", que substitueix la pàgina carregada per una nova "regal.htm", que suposadament mostrarà el regal triat. També es 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 un *array* amb les adreces web visitades en l'objecte *History*, de manera que aquest proporciona propietats i mètodes que permetran accedir des de JavaScript a aquestes adreces i visitar aquests web.

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

Tal com s'ha comentat, manté l'historial de pàgines visitades, guardant els URL en un *array* d'objectes i, per tant, si s'accedeix a una posició de l'*array*, per exemple history[2], s'obtindrà l'URL visitat en tercer lloc (la primera posició d'un *array* en JavaScript és la 0). S'ha de tenir en compte que, per motius de seguretat, és possible que certes propietats estiguin restringides per defecte en alguns navegadors.

Les propietats principals de l'objecte són les següents:

Taula 12. Propietats de l'objecte *History*

Nom	Descripció	Exemple
length	Nombre d'elements de l'*array*. És de només lectura.	alert(history.length)
state	Retorna el valor de l'estat de l'objecte.	alert(history.state)

Els mètodes principals de l'objecte són els següents:

Taula 13. 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 usar el mètode *go* de la forma: history.go(-1).
- El mètode *forward* té el mateix efecte que usar el mètode *go* de la forma: history.go(1).
- La forma del mètode *go*, go(0), té com a efecte l'actualització de la pàgina actual.

A continuació es presenta un exemple en què es pot observar l'ús dels mètodes de l'objecte:

```
<html>
<head>
</head>
<body>
<table width="100%" border="0" cellspacing="0" cellpadding="0">
<tr>
      <td bgcolor="#000000">
         <div align="center"><a href="javascript:history.back()">Anterior</a></div>
      </td>
      <td bgcolor="#000000">
         <div align="center"><a href="javascript:history.forward()">Següent</a></div>
      </td>
      <td bgcolor="#000000">
         <div align="center"><a href="javascript:history.go(0)">Actualitzar</a></div>
      </td>
</tr>
</table>
<p align="center">Pàgina 1</p>
<div align="center">
   <p><a href="http://www.uoc.edu">Pàgina 2</a></p>
   <p><a href="http://www.google.com">Pàgina 3</a></p>
</div>
</body>
</html>
```

3.4. L'objecte *Screen*

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

A continuació es presenten les propietats principals de l'objecte:

Taula 14. Propietats de l'objecte *Screen*

Nom	Descripció
availHeight	Altura en píxels de la pantalla sense explicar les possibles utilitats que pugui mostrar el sistema en pantalla, com les barres d'eines.
availWidth	Amplada en píxels de la pantalla sense explicar les possibles utilitats que pugui mostrar 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	Altura de la pantalla en píxels.
pixelDepth	Resolució de pantalla en bits per píxel.
width	Amplada de la pantalla en píxels.

L'objecte *Screen* no disposa de mètodes.

3.5. L'objecte *Navigator*

L'objecte *Navigator* conté la informació del navegador que s'està usant. Aquestes propietats són utilitzades generalment per a la detecció del navegador, encara que també proporcionen una sèrie 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 seves propietats són de només lectura. A continuació, es poden consultar les principals propietats de l'objecte en la taula següent:

Taula 15. 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 suportats pel navegador.	
platform	Sistema operatiu sobre el qual s'executa el navegador.	alert(navigator.platform) Podria mostrar dades com Win32, Win16, Mac68k, MacPPC, etc.
plugins	*Array* de *plugins* 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 disponibles en l'objecte:

Taula 16. Mètodes de l'objecte *Navigator*

Nom	Descripció	Sintaxi	Retorn
javaEnabled	Comprova si l'opció Java està activada.	javaEnabled()	*True* si està actiu Java, i *False* en cas contrari.

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

```
<html>
<head>
</head>
<body>
<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 suportat pel client. Es pot accedir a aquest objecte mitjançant 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 és així, extreure informació sobre l'objecte *Plugin* que el controla. Les propietats d'aquest objecte són del tipus només lectura i es presenten en la taula següent:

Taula 17. Propietats de l'objecte *MimeType*

Nom	Descripció	Exemple
description	Descripció del tipus MIME.	navigator.mimeTypes["image/jpeg"].description El resultat seria: 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 acceptades pel tipus MIME.	navigator.mimeTypes["image/jpeg"].suffixes El resultat seria: jpeg, jpg, jpe, jfif, pjpeg, pjp
type	Nom del tipus MIME.	navigator.mimeTypes["image/jpeg"].type El resultat seria: image/jpeg

L'exemple següent mostra les propietats per a cada objecte mimeType del client.

```
<html>
<head>
</head>
<body>
<h1>Propietats de cada objecte mimeType del client:</h1>
<script type="text/javascript">
   document.writeln("<table border=1><tr valing=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 valing=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 en el navegador. Aquests objectes estan disponibles mitjançant la propietat enabledPlugin d'objectes MimeType. Cada *plugin* proporciona informació sobre el component, com la seva descripció, els tipus MIME que suporta, etc.

Aquest objecte se sol utilitzar per a determinar si el navegador suporta un component *plugin* específic i la seva versió.

A continuació es presenten les propietats principals de l'objecte:

Taula 18. Propietats de l'objecte *Plugin*

Nom	Descripció
description	Descripció del *plugin*. És de només lectura.
filename	Nom del *plugin* en el disc. És de només lectura.
length	Nombre d'elements de l'*array* d'objectes MimeType. És de només lectura.
name	Nom del *plugin*. És de només lectura.

L'exemple següent mostra les propietats per cada *plugin* instal·lat en el client.

```
<html>
<head>
</head>
<body>
<b>Propietats per a cada connector instal·lat en el client:</b><p>
<script type="text/javascript">
   document.writeln("<table botder=1><tr valign=top>"+ "<th valign=left>i" + "<th
   valign=left>name" + "<th valign=left>filename" + "<th valign=left>description"+
   "<th valign=left># of types</tr>")
   for (i=0; i < navigator.plugins.length; i++) {
      document.writeln("<th 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 de DOM

En aquest apartat es baixarà un nivell, fins a arribar a l'objecte *Document*, de manera que seguint la nomenclatura plantejada 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 diferents versions dels navegadors.

Web recomanat

Per a informació més detallada i actualitzada de les propietats i mètodes dels objectes del DOM, podeu visitar la pàgina de w3schools:

http://www.w3schools.com/jsref/

4.1. L'objecte *Document*

L'objecte *Document* proporciona accés al contingut del document HTML i proporciona mètodes per a la seva manipulació. Per a cada pàgina HTML es genera un objecte *Document* quan aquesta es carrega en una finestra; d'aquesta manera, tot objecte *Window* conté un objecte *Document* al qual pot accedir utilitzant la seva 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 19. Propietats de l'objecte *Document*

Nom	Descripció	Exemple
anchors	*Array* d'objectes *anchor* del document. És de només lectura.	
applets	*Array* d'objectes miniaplicació del document. És de només lectura.	
cookie	*String* que representa les galetes associades al document.	
doctype	Retorna un objecte del tipus *doctype* del document HTML actual.	alert(document.doctype.name);
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 incrustats en el document. És de només lectura.	
forms	*Array* d'objectes *form* del document. És de només lectura.	document.forms[i] o equivalentment: document.form_i
images	*Array* d'objectes *image* del document. És de només lectura.	document.images[i] o equivalentment: document.image_i
lastModified	*String* que representa la data de l'última modificació del document. És de només lectura.	document.write("Aquesta pàgina ha estat actualitzada " + document.lastModified)
links	*Array* d'objectes *link* del document. És de només lectura.	document.links[i]

Nom	Descripció	Exemple
plugins	*Array* d'objectes *plugin* del document. És de només lectura.	document.plugins[i]

referrer	URL del document que es crida en prémer un enllaç.
	És de només lectura.
		function obtenirRef() {
		return document.referrer
		}

title	*String* que representa el títol del document. És de només lectura.	var finestra1 = window.open("http://www.uoc.es") var títol = finestra1.document.title
URL	*String* que representa l'URL complet del document. És de només lectura.	document.write("L'URL actual és " + document.URL)

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

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

- La propietat *domain* només pot modificar-se de manera restringida. Inicialment, conté el domini del servidor web des del qual ha estat carregada la pàgina. Es pot modificar aquesta propietat, però només per un domini amb el mateix sufix. Per exemple, un *script* provinent de tot.mon.com pot canviar el domini per mon.com, però un *script* provinent de stars.com no pot. Una vegada s'ha canviat el valor de la propietat, no es pot retornar al seu valor original.

- La propietat *lastModified* es deriva de les dades de la capçalera HTTP enviada pel servidor. Normalment, el servidor obté aquesta dada examinant la data de l'última modificació del fitxer. Aquesta informació no ha d'existir necessàriament en la capçalera. En aquest cas, JavaScript rep un 0 que visualitzaria la data January 1, 1970 GMT.

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

A continuació, es mostra un subconjunt dels mètodes principals disponibles en l'objecte:

Taula 20. Mètodes de l'objecte *Document*

Nom	Descripció	Sintaxi	Paràmetres
close	Finalitza el flux de sortida al document i mostra el contingut escrit.	close()	
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 nou document no s'afegeix en l'historial.

Nom	Descripció	Sintaxi	Paràmetres
write	Escriu una expressió HTML en el 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:

- El mètode *close* finalitza la càrrega del document que havia començat amb el mètode *open*; quan això passa, en la barra d'estat del navegador apareix la frase "Llest".

- Els possibles valors per al primer paràmetre del mètode open són:
  - text/html: especifica que el document conté text ASCII en format HTML.
  - text/plain: especifica que el document conté text ASCII net amb caràcters de finalització 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 JPG 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 *plugins* específics, per exemple, "application/x-director" carrega el *plugin* de Macromedia Shockwave. Aquests només són vàlids si el *plugin* està instal·lat.

L'exemple següent mostra el procés en què es crea una nova 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 texto = "Exemple per a l'objecte Document"
      lamevaFinestra.document.open("text/html", "replace")
      lamevaFinestra.document.write("<p>" + texto)
      lamevaFinestra.document.write("<p>history.length es " + lamevaFinestra.history.length)
      lamevaFinestra.document.close()
   }
</script>
</head>
<body>
<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 <al fet que> especificaven una propietat name (cridada Anchor) i un altre pels quals especificaven una propietat href (anomenat Link). Aquesta nomenclatura és antiquada, i amb el DOM estàndard ja no existeix cap distinció, ja que es fusionen Anchor i Link.

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

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

Taula 21. 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 és necessari afegir totes aquelles que s'han presentat en el segon apartat d'aquest mòdul, relacionades amb el DOM estàndard (per exemple, nodeValue, nodeType, nodeName, nextSibling, etc.).

La taula següent mostra els mètodes mes usats dels disponibles en l'objecte:

Taula 22. Mètodes d'etiquetes HTML

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

Nom	Descripció	Sintaxi	Retorn
focus()	Assigna el focus a l'element.	Area2.focus();	

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

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

Taula 23. Propietats i mètodes específics dels objectes *Anchor*, *Link* i *Area*

Nom	Descripció	Exemple
hash	*String* començat 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, subdomini del servidor.	
hostname	*String* que conté el nom complet del *host* incloent el nom del servidor, el subdomini, el domini i el port.	
href	*String* que especifica l'URL sencer.	
pathname	*String* que conté la porció d'URL referent a la ruta.	
port	*Sring* 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 cerca continguda en l'URL.	
target	Nom de la finestra que visualitzarà el contingut de l'*hyperlink*.	document.aindex.target="finestra1"
text	(Només per a l'objecte *Area*). Text corresponent a l'etiqueta A.	

Respecte els mètodes anteriors hai que prestar especial atenció a l'hora d'usar la propietat *hash*, ja que si la URL és:

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

I es realitza la assignació següent:

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

Il valor que pren la propietat és el següent:

```
##italia
```

4.3. L'objecte *Image*

Un objecte *Image* correspon a una etiqueta <img> d'HTML. Aquest objecte disposa de 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), on els paràmetres són opcionals i indiquen el valor d'ample i alt de la imatge en píxels.

En tractar-se d'un objecte basat en una etiqueta HTML, compartirà les propietats i mètodes presentats en el subapartat anterior i els següents, que li són específics:

Taula 24. Propietats de l'objecte *Image*

Nom	Descripció	Exemple

border	*String* que especifica l'amplada en píxels de la vora de la imatge.
	És de només lectura
	(no suportat en HTML 5).
		`function bordeImg(imagen) {`
		`  if (imagen.border==0)`
		`  alert('La imatge no té vora')`
		`  else alert('La vora de la imatge és'`
		`  + imatge.border)`
		`}`

complete	Valor booleà que indica quan el navegador ha acabat de carregar la imatge. És de només lectura.	
height	*String* que especifica l'altura en píxels de la imatge. És de només lectura.	
hspace	*String* que especifica l'espai en píxels entre la vora dreta i esquerre de la imatge i el text que la segueix o la precedeix. És de només lectura (no suportat en HTML 5).	
name	*String* que especifica el nom de la imatge. És de només lectura (no suportat en HTML 5, utilitzar l'atribut "id").	
src	*String* que especifica l'URL de la imatge.	
vspace	*String* que especifica l'espai en píxels entre la vora superior i inferior de la imatge i el text que la segueix o la precedeix. És de només lectura (no suportat en HTML 5).	
width	*String* que especifica l'amplada en píxels de la imatge. És de només lectura.	

Les propietats referents a l'alt i l'ample d'una imatge poden establir-se en la creació de l'objecte *Image*, però no poden ser modificades 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 és utilitzat per incloure camps de text, elements de selecció (botons de selecció, llistes de selecció, etc.) i botons, i el seu objectiu final és la recopilació d'informació per part de l'usuari per ser processada i enviada al servidor.

Cada formulari en un document és, en si mateix, un objecte diferent. Per fer referència als elements d'un formulari, serà convenient utilitzar l'atribut aneu.

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

Taula 25. 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://pepe@jet.es"`
elements	*Array* d'objectes corresponents als elements del formulari. És de només lectura.	
encoding	*String* que especifica la codificació MIME del formulari.	`function obtenerCodif() { return document.elmeuForm.encoding }`
length	Nombre d'elements del formulari. És de solament lectura.	`numele = elmeuForm.elements.length`
method	*String* que especifica com i 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	`document.elmeuForm.target = "finestra1"`	*String* que especifica la finestra en la qual apareixerà la resposta, una vegada el formulari s'hagi enviat.

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

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

Taula 26. 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, és necessari apuntar que el mètode reset restaura els valors per defecte en els elements del formulari. Per usar-ho no és necessari haver definit un botó en el formulari, es pot cridar des d'una funció.

A continuació, es presenta un exemple en el qual se sol·licita a l'usuari la introducció dels valors A o B. Si la dada introduïda és correcte, es mostra un missatge que ho indica, en cas contrari, el missatge mostra els possibles valors que introduir i inicialitza el formulari.

```
<html>
<head>
<script language="JavaScript" 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"><P>
</form>
</body>
</html>
```

A continuació, es van a plantejar tots els objectes que pertanyen o formen part d'un formulari. En tractar-se d'objectes basat en etiquetes HTML, compartiran les propietats i mètodes presentats anteriorment.

4.4.1. Els objectes *Hidden*, *Text*, *Textarea*, *Password*, *URL*, *Search* i *Email*

Es plantegen els objectes de manera conjunta, ja que comparteixen propietats i mètodes; en particular, els objectes *Hidden*, *Text*, *Password*, *URL*, *Search* i *Email* corresponen a l'etiqueta HTML <input>, en la qual la seva diferència apareix per 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 en el navegador. Al no ser visible, l'usuari no pot modificar el seu valor, només es pot modificar des de la programació, can-

viant el valor de la propietat *value*. S'utilitza per enviar parells de dades (nomeni/valor) quan es fa un submit del formulari.

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

- Els objectes URL, Search i Email, són objectes similars a l'objecte Text, el fet de diferenciar-los de Text ens pot ajudar a l'hora de realitzar validacions, o fins i tot pot fer que el navegador treballi amb aquests camps de manera diferent de com ho faria amb un objecte Text, per exemple, en el navegador d'un dispositiu mòbil a fi del tipus URL adaptarà el seu teclat per posar més accessibles les tecles amb el contingut típoco d'una URL com podrien ser "/" o ".com".

- L'objecte *Password* és un camp del formulari en el qual l'usuari pot introduir text. Cada caràcter introduït es visualitza en la pantalla mitjançant un asterisc.

- L'objecte *Textarea* és un camp del formulari en el qual l'usuari pot introduir text en diverses línies. Per indicar en el text introduït un salt de línia, ha d'especificar-se el caràcter concret per a això. Aquest caràcter varia segons la plataforma: en Unix és \n; en Windows és \r; i en Macintosh és \n. JavaScript comprova aquests possibles carácteres i els trasllada a la plataforma de l'usuari. L'objecte Textarea es construeix mitjançant l'etiqueta <textarea> d'HTML.

Els objectes *Hidden*, *Text*, *Password*, *URL*, *Search* i *Email* es construeixen mitjançant l'etiqueta <input> d'HTML, con els valors "hidden", "text", "password","url", "search" i "email" en l'atribut "type".

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

Taula 27. Propietats dels objectes *Hidden*, *Text*, *Textarea*, *Password*, *URL*, *Search* i *Email*

Nom	Descripció	Exemple
form	Referència a l'objecte *form* que el conté. És de solament 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
placeholder	Cadena que apareixerà en l'objecte a tall d'exemple mentre el valor de l'objecte està buit (aquesta propietat no està disponible per a l'objecte *Hidden*).	document.elmeuForm.elements[i].placeholder = "Escrigui el seu nom"
readOnly	Valor booleà que indica si el contingut de l'objecte és editable o no (aquesta propietat no està disponible per a l'objecte *Hidden*).	document.elmeuForm.elements[i].readOnly = true
type	*String* que especifica el tipus de l'objecte. És de solament lectura.	document.elmeuForm.elements[i].type

Nom	Descripció	Exemple
value	*String* que especifica el valor de l'objecte.	document.write("El valor del camp és " + document.elmeuForm.any.value + " ")

Taula 28. Mètodes dels objectes *Hidden*, *Text*, *Textarea*, *Password*, *URL*, *Search* i *Email*

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, indicar que el mètode select il·lumina l'àrea de text i pot usar-se per situar el cursor en el lloc en el qual l'usuari ha de realitzar l'entrada de dades. És més còmode per a l'usuari quan en començar a introduir una dada se substitueix directament tota la dada ja existent.

4.4.2. L'objecte *Range* i *Number*

Estus objectes també es corresponen a un element <input> i es defineixen assignant con els valors "range", "number" en atribut "type". Aquests objectes comparteixes les propietats vistes en el punt anterior, exceptuant l'objecte Range que no disposa de la propietat readOnly.

Dins d'un formulari un camp Number, serveixen per introduir valors numèrics i pot fer que en navegador modifiqui la seva funcionalitat, per exemple, afegint al costat del camp dos botons per realitzar l'increment o decremente del valor o en el cas d'un dispositiu mòbil en intentar introduir un valor farà que aquest ens mostri directament el teclat numèric. La línia per especificar un camp del tipus number seria:

```
<input type="number" max="100" step="10">
```

Figura 13. Exemple d'un camp number

Els camps Range serveixen per triar un valor dins d'un rang utilitzant per a això un deslizador. La línia per espedicifar un camp del tipus number seria:

```
<input type="range" id="myRange" value="75">
```

La visualització d'aquest element dependrà del navegador, a continuació, un exemple de com es visualitza un camp Range en Internet Explorer i en Firefox:

Figura 14. Visualització d'un camp number en Firefox i Internet explorer.

Les propietats que afegeixen els objectes Range i Number són:

Taula 29. Propietats addicionals de Range i Number

Nom	Descripció	Sintaxi
max	Valor numèric que serveix per a indicar el valor màxim de l'objecte.	document.elmeuForm.elements[i].max = 100
min	Valor numèric que serveix per a indicar el valor mínim de l'objecte.	document.elmeuForm.elements[i].min = 10
step	Valor numèric que serveix per a especificar els intervals vàlids. Per exemple, si a la propietat *min* hi hem posat el valor de 2 i assignem un *step* amb el valor 3, el primer valor vàlid serà el 2, el següent el 5 i així successivament. En cas de no assignar un valor a *min*, es prendrà com a valor de partida el 0.	document.elmeuForm.elements[i].step = 5

4.4.3. L'objecte *FileUpload, File*

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

Dins d'un formulari, la línia especificant el camp fileUpload seria per exemple:

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

Figura 15

De manera que, en prémer el botó "Examinar", s'obre la finestra diàleg que permet buscar i seleccionar el fitxer que es desitja adjuntar.

A continuació, es mostren les principals propietats de l'objecte:

Taula 30. Propietats de l'objecte *FileUpload, File*

Nom	Descripció
form	Referència a l'objecte *form* que el conté. És de només lectura.
multiple	Valor booleà que indica si l'usuari pot seleccionar més d'un fitxer o no.
name	*String* que especifica el nom de l'objecte.
type	*String* que especifica el tipus de l'objecte. És de només lectura.

Nom	Descripció
value	*String* que especifica el valor de l'objecte.

4.4.4. 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 seves característiques 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 per defecte tenen els elements del formulari.
- L'objecte *Submit* representa un botó del formulari, que té la particularitat d'enviar les dades del formulari.

A continuació, es presenten les propietats i mètodes principals d'aquests objectes:

Taula 31. Propietats dels objectes *Button*, *Reset* i *Submit*

Nom	Descripció
form	Referència a l'objecte *form* que el conté. És de només lectura.
name	*String* que especifica el nom de l'objecte.
type	*String* que especifica el tipus de l'objecte. És de només lectura.
value	*String* que especifica el valor de l'objecte.

Taula 32. Mètodes dels objectes *Button*, *Reset* i *Submit*

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

4.4.5. L'objecte *Radio*

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

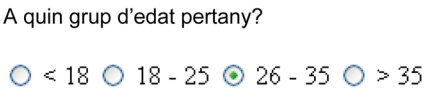
Per entener millor el seu ús, a continuació es planteja un exemple en el qual hi ha un formulari que conté 4 objectes Radio dins d'un mateix grup. Això significa que el fet de tenir un d'ells seleccionat implica que la resta no ho estigui.

Per indicar que pertanyen al mateix grup, el valor del paràmetre name ha de ser igual per a tots, i per determinar què 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 en la finestra del navegador és el següent:

Figura 16



A continuació, es presenten les propietats i mètodes principals de l'objecte *Radio*:

Taula 33. 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*, en cas contrari).	if (document.elmeuForm.musica[0].checked == true) { alert("Està seleccionada l'opció música clàssica") }
defaultChecked	Valor booleà que especifica l'estat per defecte del botó (*true* si està seleccionat per defecte, i *false*, en cas contrari).	
form	Referència a l'objecte *Form* que el conté. És de només 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. És de només lectura.	
value	*String* que especifica el valor associat al botó.	

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

Taula 34. Mètodes de l'objecte *Radio*

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

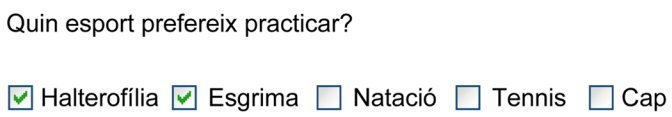
4.4.6. L'objecte *Checkbox*

Igual que l'objecte anterior, l'objecte *Checkbox* correspon a un element HTML <input>, amb l'atribut type en "checkbox", i es representa com una casella que l'usuari pot marcar o de la qual pot llevar la marca segons li interessi.

En l'exemple següent, el formulari conté 5 objectes Checkbox dins d'un mateix grup. En aquest cas, i a diferència del grup d'objectes Radio, es permet seleccionar més d'una opció en paral·lel. Per determinar què opció estarà 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 17.



A continuació, es presenten les principals propietats i mètodes de l'objecte *Checkbox*:

Taula 35. 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* en cas contrari).
defaultChecked	Valor booleà que especifica l'estat per defecte del botó (*true* si està seleccionat per defecte i *false* en cas contrari).
form	Referència a l'objecte *form* que el conté. És de només lectura.
name	*String* que especifica el nom de l'objecte.
type	*String* que especifica el tipus de l'objecte. És de només 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, és el valor que s'enviarà amb la resta de dades del formulari al servidor.

Taula 36. Mètodes de l'objecte *Checkbox*

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

4.4.7. 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 o més opcions en funció dels atributs amb els quals aquell ha estat creat.

A continuació, es presenten les principals propietats i mètodes de l'objecte Select:

Taula 37. Propietats de l'objecte Select

Nom	Descripció	Exemple
form	Referència a l'objecte *Form* que el conté. És de només lectura.	
length	Nombre d'opcions en la llista.	num = 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* corresponent a les opcions de la llista ordenades segons 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. És de només lectura.	

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

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

- En el 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 podran seleccionar diversos elements o solament un. Els valors per indicar-ho són: "select-multiple" i "select-one".

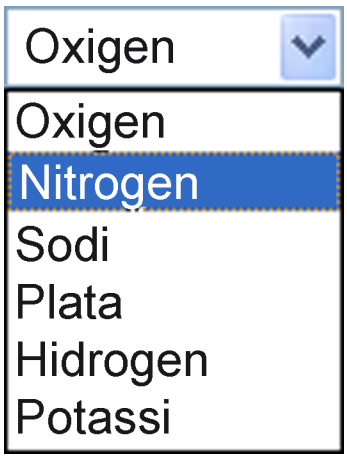
- Igual que els objectes anteriors, la propietat *value* no es visualitza en el document, és el valor que s'envia amb la resta de dades del formulari.

L'exemple següent es mostra com es realitza la construcció d'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 observar, cadascun dels elements de la llista és un objecte *Option*. El resultat en pantalla és el que s'observa a continuació:

Figura 18.



4.4.8. L'objecte *Option*

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

Els valors option es poden construir utilitzant HTML o en JavaScript tal com es descriu a continuació:

- Mitjançant la etiqueta <option> d'HTML

- Mitjançant el constructor: new option (text, value, defaultSelected, selected), els paràmetres del qual són opcionals i tenen el següent significat:
  - text: especifica el text que es veurà en la llista.
  - value: especifica el valor que s'enviarà amb les dades del formulari si l'opció està seleccionada.
  - defaultSelected: Especifica el valor inicial de l'opció, *true* si és seleccionat i *false* en cas contrari.
  - selected: especifica l'estat actual de l'opció, és a dir, si està seleccionada o no.

A continuació, es presenten les propietats principals de l'objecte option:

Taula 38. 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. lista.length; i++) { if ( document.elmeuForm.llista. options[i].defaultSelected == true ) document.elmeuForm.lista .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'enviarà amb les dades del formulari si l'opció està seleccionada.	

5. Gestió d'esdeveniments

Un esdeveniment és una acció que ocorre en la majoria de les ocasions per alguna cosa que fa l'usuari, per exemple, la pulsació d'un botó, el moviment del ratolí, etc. Per la seva banda, el manejador d'esdeveniments és el codi JavaScript, que s'associa a un esdeveniment, de manera que quan aquest es dóna, s'executa el codi.

No existeix un únic model de gestió d'esdeveniments, sinó bàsicament dos que han evolucionat paral·lelament a l'evolució dels models d'objectes dels principals navegadors.

En el model tradicional del model d'objectes, tots dos navegadors oferien un suport molt bàsic per manejar 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, afegint nous esdeveniments i una major funcionalitat, però de la mateixa manera que els models d'objectes divergien, també ho van fer els models d'esdeveniments.

La convergència es produeix amb l'aparició del model d'esdeveniments estàndard de W3C. Aquest model es pot consultar en l'estàndard "DOM2, model d'esdeveniments". Tal com es veurà a continuació, el model estàndard adquireix els avantatges de cadascun dels models anteriors, creant d'aquesta manera un model robust i molt complet de control d'esdeveniments.

En els següents subapartats es realitzarà una revisió històrica del model d'esdeveniments, des del seu principi fins a l'aparició del model estàndard del DOM2 del W3C.

5.1. Model bàsic de control d'esdeveniments

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

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

Taula 39. 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.	Tots els elements HTML excepte <base>, <bdo>, , <head>, <html>, <iframe>, <meta>, <param>, <script>, <style> i <title>
onchange	Es produeix quan un camp de formulari ha perdut el focus i el seu valor ha canviat.	<input>, <keygen>, <select> i <textarea>
onclick	Es produeix quan es prem sobre un objecte.	Tots els elements visibles
ondblclick	Es produeix quan es fa una doble pulsació sobre l'objecte.	Tots els elements visibles
onfocus	Es produeix quan un objecte rep el focus.	Tots els elements HTML excepte <base>, <bdo>, , <head>, <html>, <iframe>, <meta>, <param>, <script>, <style> i <title>
onkeydown	Es produeix quan es prem una tecla.	Tots els elements visibles
onkeypress	Es produeix quan es manté premuda una tecla.	Tots els elements visibles
onkeyup	Es produeix quan s'allibera una tecla premuda.	Tots els elements visibles
onload	Es produeix quan el navegador acaba de carregar una finestra o un conjunt de marcs.	<body>, <frame>, <iframe>, <img>, <input type="image">, <link>, <script> i <style>
onmousedown	Es produeix quan es prem un botó del ratolí.	Tots els elements visibles
onmousemove	Es produeix quan es mou el ratolí mentre s'està sobre l'objecte.	Tots els elements visibles
onmouseout	Es produeix quan el punter abandona 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 premut 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> i <keygen>
onsubmit	Es produeix quan s'envia un formulari.	<form>
onunload	Es produeix quan s'abandona la pàgina.	<body>, <frameset>

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

Internet Explorer

Per exemple, Internet Explorer disposa d'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 la càrrega de dades.

Com es veu en la taula, a més de tractar 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 fer una precàrrega d'imatges, de manera que aquestes queden en la memòria cau (*cache*) del navegador. Això fa que es presentin de manera més ràpida quan siguin requerides. De la mateixa manera, l'esdeveniment *unload* es pot utilitzar per a fer una neteja de la memòria i habilitar així 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 és possible implementar el controlador si l'etiqueta o l'objecte sobre el 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, és necessari implementar un controlador d'esdeveniment en 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:

```
<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 sobre l'enllaç provoca l'execució de la funció lamevaFuncio().

A continuació, es mostra l'exemple incloent el codi HTML:

```
<html>
<head>
<script type="text/javascript">
   function lamevaFuncio() {
      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 mitjançant JavaScript. L'avantatge principal d'aquest mecanisme és la separació entre l'estructura i la lògica del document de la seva presentació.

Quan s'implementa un controlador d'esdeveniment per a un objecte, l'objecte en qüestió adquireix una propietat, que pren el nom del controlador de l'esdeveniment. Aquesta nova propietat és la que permetrà 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 direccio() {
      alert("La URL d'aquesta pàgina és: " + document.URL );
   }
</script>
</head>
<body onload="direccio()">
</body>
</html>
```

En l'exemple següent, cal remarcar com el controlador s'usa com una propietat més de l'objecte *Window*:

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

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

5.1.3. Valors de tornada

Els controladors d'esdeveniments disposen d'una característica molt útil, els valors de tornada. Els controladors d'esdeveniments no són més que funcions JavaScript, i aquestes poden retornar un valor mitjançant la sentència *return*.

Els valors de tornada es poden utilitzar amb diversos objectius. Vegem-ne un parell d'exemples:

- Un formulari en el qual s'ha definit un controlador per a l'esdeveniment *submit*, de manera que aquest controlador valida els valors introduïts en el camp i retorna *true* o *false*, depenent de si aquests són correctes. En cas que es retorni *true*, l'enviament del formulari es fa, i si es retorna *false*, es cancel·la.

- Un altre exemple seria l'esdeveniment *clic* sobre un enllaç, de manera que si es retorna *false*, la càrrega de la pàgina es cancel·la.

El codi següent és un exemple del que s'ha comentat 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 tornada dels esdeveniments més útils.

Taula 40. Valors de retorn dels esdeveniments més útils

Esdeveniment	Retorn	Descripció
Click	false	Botó d'opció i casella de verificació: no s'estableix. Botó d'enviament: l'enviament 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 manté premuda la tecla.
Keypress	false	Cancel·la l'esdeveniment *keypress*.
Mousedown	false	Cancel·la l'acció predeterminada.
Submit	false	Cancel·la l'enviament del formulari.

5.2. Model HTML dinàmic de control d'esdeveniments

Els models d'esdeveniments sorgits a partir de les versions 4.x dels navegadors van proporcionar un nou objecte estàtic *Event*. Aquest objecte es creava quan ocorria un esdeveniment i l'objecte *Event* recollia informació sobre l'esdeveniment, com la posició del ratolí, el botó del ratolí premut, el tipus d'esdeveniment i altres informacions.

El realment interessant és que aquest objecte es pot consultar amb el controlador de l'esdeveniment i, d'aquesta manera, disposa d'un conjunt d'informació que descriu l'esdeveniment ocorregut.

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

5.2.1. Introducció al model d'esdeveniments d'Internet Explorer 4.x i Netscape 4.x

En Netscape 4.x, els esdeveniments es propagaven mitjançant la jerarquia del document, i començaven en la part superior per a descendir fins a arribar a l'objecte en el qual s'havia generat l'esdeveniment.

D'aquesta manera, els objectes superiors com *Window*, *Document* i *Layer* podien capturar els esdeveniments abans que aquests 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.

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

A diferència del model de Netscape, en el model d'esdeveniments d'Internet Explorer 4.x, l'esdeveniment comença a propagar-se des de l'objecte en el qual s'ha creat i va ascendint per l'estructura de la jerarquia fins a l'objecte *Window*.

Tant en Internet Explorer com en Netscape, quan ocorria un esdeveniment es creava un objecte *Event*. La diferència entre aquests és que Netscape l'enviava al controlador d'esdeveniments com a paràmetre, mentre que en Internet Explorer l'objecte era global i, per tant, accessible directament des de qualsevol codi per mitjà d'*Event*.

A continuació, es presenta un exemple del seu ús a Netscape:

```
<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>
```

En l'exemple veiem com la funció controladora necessita un paràmetre d'entrada que representarà l'objecte *Event*, i com al final del tot s'indica la captura de l'esdeveniment *onclick* de l'etiqueta <a> assignant la funció al controlador.

Vegem com es duia a terme la mateixa acció a Internet Explorer:

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

En el codi anterior podem veure com la funció *posicio()* no rep com a paràmetre l'objecte de l'esdeveniment i en el seu lloc fa referència a un objecte global *Event*.

Les diferències entre els navegadors en el maneig d'esdeveniments, la utilització de noms diferents per a les mateixes propietats o l'existència de certes propietats només en un d'aquests obligaven els programadors a detectar el navegador que estava utilitzant l'usuari per a poder actuar en conseqüència.

A continuació, un exemple de com es podia aconseguir que els exemples anteriors funcionessin en els dos navegadors:

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

En l'exemple anterior, veiem com en la primera línia de la funció *posició* es mira si realment la funció ha rebut com a paràmetre l'objecte esdeveniment que envia Netscape; en cas afirmatiu, assignarem el valor "e" a l'objecte rebut com a paràmetre, en cas negatiu, assignarem l'objecte global *Event* d'Internet Explorer.

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 en la part superior de la jerarquia i descendeixen per aquesta fins a arribar a l'objecte de destinació. Aquest procés es coneix com a **fase de captura** i és similar al que ocorre amb els navegadors Netscape 4.

Però la propagació no ha finalitzat en l'objecte de destinació, sinó que, una vegada es trobi en aquest i finalitzi el tractament pel controlador (si s'ha assignat), comença la fase que es coneix com a *fase d'ascens*, en la qual l'esdeveniment implementa el camí invers fins a arribar a la part alta de la jerarquia. D'aquesta manera, un esdeveniment pot ser capturat i manejat per un objecte de la jerarquia, tant en la fase de captura com d'ascens.

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

Taula 41. Propietats de l'objecte *Event* en l'el DOM estàndard

Propietat	Descripció
altKey	Conté un valor booleà que indica si s'ha premut la tecla ALT.
ctrlKey	Conté un valor booleà que indica si s'ha premut la tecla CTRL.
shiftKey	Conté un valor booleà que indica si s'ha premut la tecla SHIFT.
metaKey	Conté un valor booleà que indica si s'ha premut 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 pressionat.
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.
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 el qual s'ha produït l'esdeveniment.
currentTarget	Conté una referència al node al qual s'ha assignat el controlador.

Propietat	Descripció
relatedTarget	Conté una referència al node del qual ha sortit l'esdeveniment.
type	Conté una cadena que conté 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.

Per la seva banda, els navegadors no compleixen l'estàndard al 100%. En realitat, tenen certes 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 especificats per 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 42. Subconjunt d'esdeveniments definits en el DOM2

Tipus esdeveniment	Esdeveniment	Ascendeix?	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
	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		

Tipus esdeveniment	Esdeveniment	Ascendeix?	Cancel·lable?
	DOMFocusOut		
	DOMActivate		
De canvi jerarquia	DOMSubtreeModified	Sí	No
	DOMNodeInserted	Sí	No
	DOMNodeRemoved	Sí	No
	DOMNodeRemoved-FromDocument	No	No
	DOMNodeInsertedIn-toDocument	No	No
	DOMAttrModified	Sí	No
	DOMCharacterData-Modified	Sí	No

Com s'observa en la taula, apareix un nou tipus d'esdeveniments: **esdeveniments de canvi en la jerarquia**. Aquest tipus d'esdeveniments, com el seu nom indica, es produeixen quan l'estructura de la pàgina és modificada, ja que això és possible 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 passava a Netscape).

En l'exemple següent, es veu com es vincula l'esdeveniment *clic* a una etiqueta <a>, utilitzant 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, és necessari incidir en el següent:

- La funció controladora rep en el seu paràmetre a l'objecte *Event*, que podrà consultar-se a l'interior 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, per la qual cosa poden aparèixer problemes amb 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 la vinculació d'esdeveniments a objectes de la jerarquia. Aquest mecanisme aporta els avantatges següents:

- Permet especificar si l'esdeveniment es manejarà durant la fase de captura o durant la fase d'ascens, mentre que en el mètode presentat anteriorment, l'esdeveniment es captura quan l'objecte és la destinació i durant la fase d'ascens, en cas que n'hi hagi 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 **oïdors d'esdeveniments**, que no són més que controladors d'esdeveniments vinculats a un node determinat de la jerarquia d'objectes, que és activat durant una fase específica del cicle de vida de l'esdeveniment, la fase de captura o la d'ascens.

El mètode que afegeix un oïdor 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 s'escoltarà.
- Controlador: és la funció que manejarà l'esdeveniment.
- Sentit: és un valor booleà que indica si es captura en la fase de captura o en la fase d'ascens.

Igual que es disposa del mètode anterior per a la creació d'un oïdor, la mateixa sintaxi és utilitzada pel mètode removeEventListener(tipus,controlador,sentit) per a eliminar un oïdor afegit anteriorment.

A més, l'objecte *Event* disposa de dos mètodes molt interessants:

- preventDefault(): que cancel·la el comportament estàndard d'un esdeveniment.
- stopPropagation(): que atura el flux de l'esdeveniment per mitjà de la jerarquia.

En l'exemple següent es pot observar codi que utilitza 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 click");
   }
   document.addEventListener("click",NoAltClick,true);
   document.getElementById("uoc").addEventListener("click",MostraClick,true);
</script>
```

De l'exemple, és necessari tenir en compte el detall següent: el controlador NoAltClick provoca que els esdeveniments clic no funcionin mentre es prem la tecla Alt. Això s'aconsegueix en cancel·lar el comportament predeterminat de l'esdeveniment i en aturar-ne la propagació mitjançant la jerarquia.

Per acabar, un últim 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 nou destinatari 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.