Índex

1. JavaScript

1.1 Definició

Què és un *script*? Un *script* és una successió de una o més sentències que poden ser executades en algun context sense necessitat de passos de compilació previs. És a dir, són interpretats no compilats. No són exclusius de la web però nosaltres només els farem servir en aquest context.

Quins llenguatges d'*script* hi podem trobar? Fonamentalment se'n fan servir dos:

- JavaScript
- Visual Basic Script.

Els llenguatges d'*script* aplicats a la web s'executen en el navegador i no en el servidor. Per aquesta raó, a vegades, és necessari programar un *scrip* de manera diferent en funció del navegador de l'usuari. La seva missió és donar dinamisme a les pàgines web.

Nosaltres farem servir JavaScript. Tòpics sobre JavaScript:

- JavaScript no és un llenguatge de programació propiament dit (no es compila). No es poden fer programes seriosos, només milloren la pàgina web.
- JavaScript és diferent del llenguatge del programació Java.
- JavaScript s'integra directament dins de la pàgina web:
  - És interpretat (no compilat)
  - No es declaren les variables que es fan servir.
  - No es pot escriure al disc dur. Això és degut a que els *scripts* tenen certes limitacions per a garantir la protecció del nostre PC i sistema operatiu.

1.2 Incloure JavaScript a la web:

1.2.1 Forma general:

```
<SCRIPT LANGUAGE="javascript + versió">
Sentències JavaScript
</SCRIPT>
```

En qualsevol part del codi HTML encara que sempre que sigui possible serà recomanable, per qüestions organitzatives, posar-ho dins del `HEAD`.

Generalment no s'inclou la versió sent el navegador el que la pren per defecte.

És possible que el navegador no accepti JavaScript, per aquesta raó la sintaxi que farem servir més freqüentment serà la següent:

```
<SCRIPT LANGUAGE=”javascript”>
<!--
   Sentències JavaScript
// -->
</SCRIPT>
```

I en el cas anterior ens podem recolzar en les etiquetes `<NOSCRIPT></NOSCRIPT>` que executaran el seu contingut només en el cas que el nostre navegador no admeti scripts.

1.2.2 Full a part:

Una altre forma d'incloure sentències JavaScript en el nostre document web és fer-ho des d'un fitxer a part. Per portar-ho a terme només hem que crear un fitxer amb extensió `js` i incloure'l a la web fent servir la següent sintaxi:

```
<SCRIPT SRC=”fitxer.js”></SCRIPT>
```

1.2.3 Etiquetes HTML

La majoria de les etiquetes HTML admeten incloure controladors d'events amb el seu respectiu codi JavaScript.

Veiem uns exemples:

```
<HTML>
<HEAD>
  <SCRIPT LANGUAGE=”javascript”>
  <!--
  function avisar()
  {
     alert(“Si cliques aniràs a GOOGLE”);
  }
  // -->
  </SCRIPT>
</HEAD>
<BODY>
<A HREF=”http://www.google.es”  onMouseOver=”avisar()”>Ir a GOOGLE</A>
<A HREF=”http://www.google.es”  onClick=”alert('Vas a GOOGLE')”>Ir a GOOGLE</A>
<A HREF=”http://www.google.es”  onClick=”return confirm('Vols anar a GOOGLE?')”>Ir a GOOGLE</A>
<H1 onClick=”alert('Acabes de clicar un títol')”>Hola que tal</H1>
</BODY>
</HTML>
```

Observeu les funcions `alert` i `confirm` que en JavaScript mostren un missatge en una finestra i la diferència d'ús de les cometes simples i dobles.

2. Events:

Un event és l'efecte que es produeix quan té lloc una acció determinada. Per exemple, un event pot ser:

- Clicar un botó.
- Clicar sobre un text.
- Col·locar-se en una caixa de text.
- Passar el ratolí sobre un link.
- Clicar un link.

2.1 Incorporar un event

L'esquema general de tractament d'un event és el següent:

```
<ETIQUETA_HTML CONTROLADOR_DE_EVENT="CODI JavaScript">
Per exemple:
<H1 onClick="alert('Acabes de clicar un títol')">Hola que tal</H1>
```

2.2 Tipus d'events

Els events que pot manipular JavaScript són els següents:

Event	Controlador de l'event	Té lloc quan l'usuari:
abort	onAbort	Cancel·la la carrega d'una imatge .
blur	onBlur	Fa inactiva la finestra, el frame o un element d'un formulari.
click	onClick	Clica sobre un element, un link o una part d'un formulari
doubleclick	onDblClick	Clica dues vegades sobre un element, un link o una part d'un formulari.
change	onChange	Canvia el valor d'un element.
error	onError	Obté un error en carregar la pàgina.
focus	onFocus	Fa activa la finestra, el frame o un element d'un formulari.
load	onLoad	Carrega una pàgina.
unload	onUnload	Abandona la pàgina.
mouseout	onMouseOut	Allunya el punter del ratolí d'un element HTML.
mouseover	onMouseOver	Passa el punter del ratolí sobre un element HTML.
mousedown	onMouseDown	Polsa algun botó del ratolí

mousemove	onMouseMove	Mou el ratolí.
keydown	onKeyDown	Polsa una tecla.
keypress	onKeyPress	Té polsada una tecla.
keyup	onKeyUp	Deixa de polsar una tecla.
reset	onReset	Reseteja un formulari (el deixa en blanc en polsar el botó corresponent).
select	onSelect	Selecciona un camp d'entrada d'un element d'un formulari.
dragdrop	onDragDrop	Bolca un objecte en una finestra del navegador.
move	onMove	Mou una finestra o frame.
resize	onResize	Canvia la mida d'una finestra o frame.
submit	onSubmit	Envia un formulari (polsant el botó corresponent).

3. Formularis HTML:

Un formulari és una àrea de la pàgina web en la que existeixen elements d'entrada d'informació, coneguts habitualment com a controls de formulari, en els quals l'usuari pot escriure una dada, triar una opció entre vàries, marcar o desmarcar una opció, ... El formulari en sí és un element `FORM` que tindrà com a contingut els elements que representen als controls: `INPUT`, `BUTTON`, `SELECT`, `TEXTAREA`, ...

3.1 Sintaxi general

La sintaxi general de l'element FORM és la següent:

```
<FORM ACTION="acció" METHOD="mètode d'enviament" ENCTYPE="codificació" ACCEPT="què acceptem">
</FORM>
```

Veiem ara a què fa referència cada element:

- `ACTION`: Se li assigna la URL de destí de la informació del formulari. Aquesta URL serà l'encarregada de processar el formulari.
- `METHOD`: És le mètode que farem servir per a enviar la informació. Hi han dos tipus:
  - `get`: Les dades del formulari s'afegeixen a la URL indicada a l'`ACTION` després d'un caràcter ? que actua de separador. Per tant les dades del formulari seran visibles a la barra d'adreces del navegador.
  - `post`: Mitjançant aquest mètode les dades s'introdueixen en el cos d'un bloc d'informació que serà enviat al servidor com a part de la solicitud, sense que existeixi cap reflex d'aquest fet a la URL del document. Permet enviar qualsevol tipus d'informació, fins i tot arxius locals.

- ENCTYPE: Permet determinar la codificació de l'enviament de les dades. Per defecte la codificació serà ASCII, però si volem enviar fitxers aquesta codificació no ens val, llavors farem servir ENCTYPE="multipart/from-data". Per tant, excepte en el cas que el formulari enviï fitxers, aquest paràmetre no es posarà.
- ACCEPT: Amb aquest atribut podem limitar els formats d'arxiu que s'acceptaran si l'usuari té la possibilitat d'enviar arxius mitjançant el formulari. Aquest atribut contindrà una llista de tipus MIME (Multipurpose Internet Mail Extensions – Protocol d'enviament d'informació no codificada en ASCII) separats per comes.

  Exemple:

```
<FORM ACTION="test.php" METHOD="post"
                              ENCTYPE="multipart/form-data" ACCEPT="image/jpeg,image/png">
```

  Els tipus MIME més comuns són:

Tipus MIME	Extensió
Imatges	
image/bmp	.bmp
image/x-windows-bmp	.bm
image/gif	.bmp
image/jpeg	.gif
image/jpeg	.jpe
image/png	.jpg / .png
So	
audio/basic	.au / .snd
audio/x-au	.au
audio/midi	.mid
audio/x-midi	.midi
audio/x-wav	.mid
audio/mod	.midi
audio/x-mod	.wav
audio/mpeg3	.mod
audio/x-mpeg3	.mod
audio/x-mpeg-3	.mp3

audio/x-pn-realaudio	.ra / .ram
Video	
video/avi	.avi
video/x-motion-jpeg	.mjpg
video/quicktime	.mov
video/mpeg	.mpg
application/x-shockwave-flash	.swf

Podeu consultar tots els tipus MIME a l'adreça:

http://www.utoronto.ca/webdocs/HTMLdocs/Book/Book-3ed/appb/mimetype.html

3.2 Elements dels formularis

Veiem quins elements es poden incorporar dins del formulari:

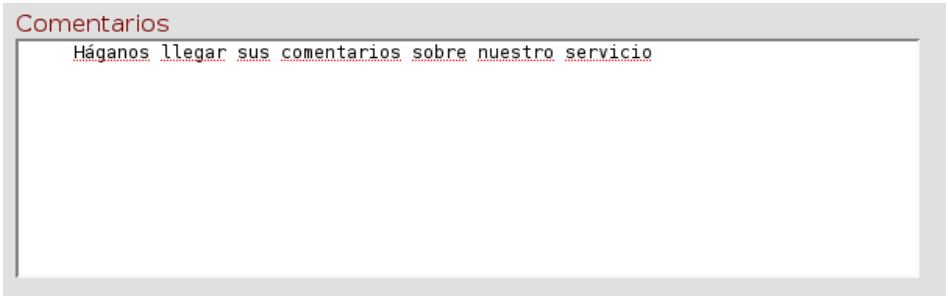
3.2.1 Elements de text

Introducció de text *(textbox)*. Hi ha dos controls que ens permeten incloure text.

- `INPUT`: Pensat per a introduir una línia de text. Els seus atributs són els següents

  - `type`: Ens permet determinar el tipus de caixa. És obligatori i pot tenir els valors següents:
    - `text`: Caixa normal.
    - `password`: Per a introduir contrasenyes.
    - `hidden`: La caixa quedarà oculta.
  - `readonly`: Només pot tenir un valor (`readonly`) i determina que la caixa de text només es pugui veure i no modificar.
  - `name`: Nom de la caixa de text. Important per a recuperar el valor una vegada enviat. És obligatori.
  - `size`: Amplada de la caixa de text en nombre de caràcters.
  - `maxlength`: Ens permet limitar l'extensió del text que podem introduir.
  - `value`: Ens permet donar un valor inicial al recuadre de text.
  - `title`: Permet crear la finestra flotant al voltant del requadre de text.
  - `tabindex`: Indica l'ordre en que l'objecte guanya el focus quan l'usuari tabula.

- `TEXTAREA`: Pensat per a introduir més d'un línia de text. Els seus atributs són:



  - `disabled`: Permet desactivar la capsa.
  - `cols`: Nombre de columnes que tindrà la caixa.
  - `rows`: Nombre de files que tindrà la caixa.
  - `name`, `readonly`, `tabindex`.

3.2.2 Caixes de sel·lecció

Caixes de selecció (*checkbox*): Són controls que ofereixen un requadre que pot marcar-se i desmarcar-se. Es construeixen amb `INPUT` però amb `TYPE="checkbox"`. A més pot contenir els següents atributs:

  - `checked`: No més pot tenir un valor (checked) i determina si la caixa de selecció està marcada o no.
  - `tabindex`, `readonly` i `disabled`.

3.2.3 Botons de ràdio

Botons de radio (*radiobutton*). És un tipus de control que opera sempre en conjunt, ha d'existir com a mínim un conjunt de dos botons de radio per grup, caracteritzant-se perquè la selecció d'un implica la no selecció dels altres. Així doncs, és apropiat per a oferir opcions exclusives. Es construeixen amb `INPUT` però amb `TYPE="radio"` i per a construir un grup de *radiobuttons* el que farem serà donar a cada membre del grup el mateix `name`. Els atributs són els següents:

  - `name`, `value` i `checked`.

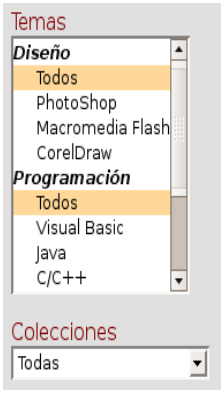
3.2.4 Llistes d'opcions

Llistes d'opcions (*listbox*): Els botons de radio no sempre són la millor opció per a triar entre diferents opcions, sobre tot si la llista d'opcions és molt extensa. Per a solucionar això existeix l'element `SELECT` en conjunció amb `OPTION` i, si és necessari, `OPTGROUP`. Aquests elements permeten crear llistes, desplegables o desplaçables, d'opcions entre les quals es pot triar una o vàries, segons ens interessi. Els atributs de `SELECT` són:

- `disabled`, `tabindex`, `name` i `size`.
- `multiple`: Només pot tenir un valor (`multiple`). Ens permet determinar que es pot triar més d'un element de la llista.

Els atributs de `OPTION` són:

- `selected`: Ens permet determinar quin element de la llista està triat.
- `value`: Ens permet donar un valor simbòlic a l'element de la llista.



Com portar-ho a terme? Veiem un codi il·lustratiu:

```
<select id="colecciones" name="colecciones">
    <option selected="selected">Todas</option>
    <option value="biblia">La biblia de</option>
    <option value="gp">Guía práctica</option>
    <option value="mi">Manual</option>
    <option value="ma">Manual avanzado</option>
</select>
```

```
<select id="categorias" name="categorias"
   multiple="multiple" size="10"
   title="Marque las de su interes">
<optgroup label="Diseño">
   <option value="disTodos" selected="selected">
   Todos
   </option>
   <option value="disPhoto">PhotoShop</option>
   <option value="disFlash">Macromedia Flash
   </option>
   <option value="disDraw">CorelDraw</option>
</optgroup>
<optgroup label="Programación">
   <option value="proTodos" selected="selected">
   Todos
   </option>
```

```
   <option value="proVB">Visual Basic</option>
   <option value="proJava">Java</option>
   <option value="proC">C/C++</option>
</optgroup>
<optgroup label="Sistemas">
   <option value="sisTodos" selected="selected">
   Todos
   </option>
   <option value="sisWindows">Windows</option>
   <option value="sisLinux">Linux</option>
   <option value="sisMac">Mac OS X</option>
</optgroup>
</select>
```

3.2.5 Botons d'acció

Per a afegir botons a un formulari contem amb les etiquetes `INPUT` i `BUTTON`. El primer està molt més limitat i ha de fer-se servir amb els següents atributs:

- `type="submit"` - És un botó que produeix l'enviament de la informació continguda en el formulari.
- `type="reset"` - Aquest botó reestablirà els controls del formulari al seu valor per defecte, és a dir l'esborrarà.
- `type="button"` - Crea un botó sense una funció específica i amb el títol que li assignem a value. Li donarem la funcionalitat amb JavaScript.

L'etiqueta `BUTTON`, en canvi, no està tan limitada i ens permet introduir en el botó contingut més amigable. Necessita tancament de la etiqueta. Els seus atributs són:

- `type="submit"`, `type="reset"` i `type="button"`.

Exemple:

```
<BUTTON TYPE="submit">
Suscriure's al <b>boletí nformatiu</b>
</BUTTON>
```

3.2.6 Altres controls

L'últim control que ens queda per veure és el que ens permet incloure un fitxer al formulari. Per fer-ho fem servir l'etiqueta `INPUT` amb `TYPE="file"`. Hem de tenir en compte dos factors:

- És imprescindible que l'etiqueta `FORM` tingui `ENCTYPE="multipart/form-data"`.

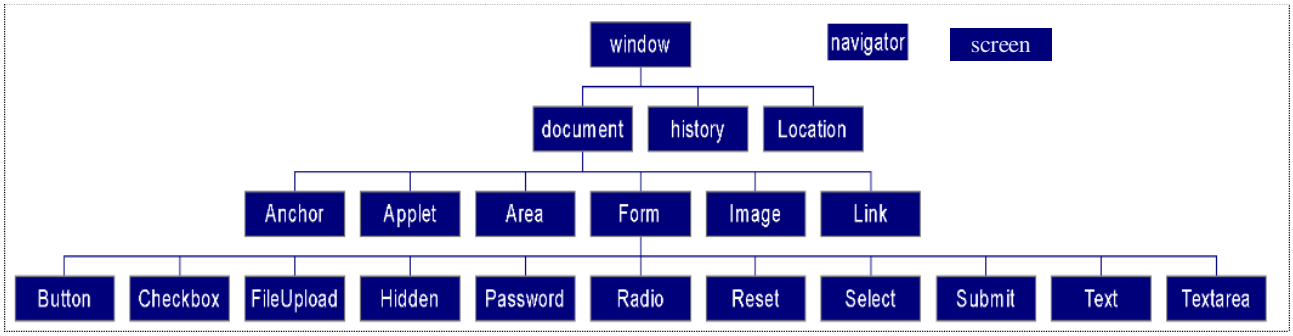
- Podem incloure l'atribut `ACCEPT` dins de l'`INPUT`. Això ens permet determinar diversos tipus de fitxers si estem pujant molts.

4. Objectes del navegador

En carregar una pàgina web es creen numerosos objectes, els valors dels quals estan basats en el document i en altres informacions. Aquests objectes són els que permeten el control de la pàgina de l'usuari.

Els objectes creats s'estructuren de forma jeràrquica. És molt important conèixer aquesta jerarquia per a comprendre bé la pàgina i poder accedir correctament als seus elements.

La jerarquia és la següent:



En la jerarquia tot el que hi ha per sota d'un element són propietats d'aquest elements.

Tota pàgina té els següents objectes:

- `navigator`: Propietats sobre el nom i la versió del navegador.
- `window`: Propietats aplicades a la finestra (mida, posició, ...)
- `Location`: Propietats basades en la URL actual.
- `history`: Propietats que representen les URL's que el client ha visitat.
- `document`: Propietats de tot el document (color de fons, de lletra, formularis, ...)

Sobre aquests objectes podem:

1. Modificar les seves propietats. Exemple: Canviar la propietat `SRC` de l'objecte `IMAGE`.
2. Realitzar accions sobre ells. Exemple: Tancar una finestra.

És important saber distinguir entre modificar un propietat o realitzar una acció, perquè la sintaxi que es fa servir JavaScript varia en funció de si fem una cosa o una altre.

4.1 Propietats dels objectes

Llistat de les propietats i accions més importants de cada objecte:

4.1.1 navigator

- `appName`: Dóna el nom del navegador.
- `appVersion`: Indica la versió del navegador.
- `platform`: Indica el nom del sistema operatiu.
- `onLine`: Informa si l'usuari a accedit a la pàgina web a través d'internet o en local.
- `connectionSpeed:` Indica la velocitat actual de la connexió.
- `cookieEnabled:` Indica mitjançant valors true/false si l'usuari té activada l'admissió de cookies.

4.1.2 window

- `close():` Tanca la finestra.
- `open():` Obre la finestra. Inclou els següents paràmetres:
  - `toolbar:` Indica si s'ha d'incloure una barra d'eines.
  - `location:` Indica si s'ha d'incloure la barra d'adreces.
  - `status:` Indica si s'ha d'incloure la barra d'estat.
  - `menubar:` Indica si s'ha d'incloure la barra de menú.
  - `scrollbars:` Indica si s'ha d'incloure les barres de desplaçament.
  - `resizable`: Determina si la mida de la finestra pot ésser modificat.
  - `width`: Estableix l'amplada de la finestra.
  - `height`: Estableix l'alçada de la finestra.
  - `top`: Estableix la distància respecte al costat superior de la finestra.
  - `left`: Estableix la distància respecte la costat esquerra de la finestra.

  Exemple:

```
window.open("http://pagina.com","Hola","toolbar=no, location=no,
                        top=100, left=100, width=200, height=200");
```

- `alert`(): Obre una finestra d'alerta amb un missatge que s'ha d'acceptar polsant OK.
- `confirm`: Obre una finestra de diàleg amb botons OK i Cancel.
- `prompt`: Obre una finestra a la que s'ha d'escriure text abans d'arribar a la pàgina.
- `blur`, `focus`: Deixa inactiva o activa la finestra (treu o dóna el focus).
- `scroll`: Desplaça una finestra fins a un coordenada específica.

- `setTimeout`: Avalua una expressió després d'un temps especificat.
- `back`: Torna a la pàgina carregada anteriorment.
- `disableExternalCapture`: Desactiva la captura d'events.
- `enableExternalCapture`: Activa la captura d'events.
- `find`: Troba el text especificat als continguts de la finestra.
- `forward`: Equival a polsar el botó de forward.
- `home`: Equival a polsar el botó de home.
- `moveBy`: Mou la finestra la quantitat especificada de píxels.
- `moveTo`: Mou la finestra a la coordenada especificada en píxels.
- `resizeBy`: Canvia la mida de la finestra la quantitat especificada en píxels.
- `resizeTo`: Canvia la mida de la finestra a la coordenada especificada en píxels.
- `scrollBy`: Desplaça el contingut de la finestra la quantitat especificada de píxels.
- `scrollTo`: Desplaça del contingut de la finestra a la coordenada especificada en píxels.
- `stop`: Deté la transferència actual de dades.

4.1.3 `document`

- `write()`: Escriu el text especificat al document.
- `open`(): Obre un document.
- `bgColor`: Estableix el color de fons.
- `fgColor`: Estableix el color del text.
- `referrer`: Proporciona la URL de la pàgina desde la qual he accedit al document.
- `title`: Retorna el títol del document.
- `URL`: Retorna la URL del document.

4.1.4 `history`

- `back()`: Carrega la pàgina anterior a la actual.
- `forward`(): Carrega la pàgina posterior a la actual.
- go(`num`): Avança (si `num` és positiu) o retrocedeix (si `num` és negatiu) en la history list el número indicat por `num`.

4.1.5 `location`

- `reload`: Equival a polsar el boto reload.
- `href`: Retorna la URL del document.

4.1.6 `screen`

- `height`: Dóna l'alçada de la pantalla en píxels.
- `width`: Dóna l'amplada de la pantalla en píxels.

4.2 Accés a les propietats i accions dels objectes:

Hi han, fonamentalment, dues formes d'accedir a les propietats:

1. Pel nom de l'objecte. Veiem-ho a través d'un exemple:

```
<FORM NAME="dades">
   <INPUT TYPE="text" NAME="caixa">
   ...
</FORM>
```

Amb JavaScript podríem accedir fent:

- `document.dades.caixa.value`
- `document.dades.caixa.class`

2. Per l'array: Tots els elements del mateix tipus s'organitzen al navegador a través d'arrays o vectors ordenats sempre per ordre d'aparició començant pel 0. Veiem-ho a través d'un exemple:

`<A HREF="http://www.google.es">Google</A>` - JavaScript: `document.links[0]`

`<A HREF="http://www.google.es">Google</A>` - JavaScript: `document.links[1]`

`<A HREF="http://www.google.es">Google</A>` - JavaScript: `document.links[2]`

Ens podem trobar en situacions bastant complicades com: `document.forms[5].elements[3]`

ATENCIÓ:

1. Encara que no és obligatori, és molt recomanable que totes les sentències en JavaScript acabin en `;`
2. Les acciones acaben en `([PARÀMETRES]);`
3. Les propietats:
   - O es modifiquen: (`="Nou valor"`)
   - O es fan servir: (`alert(....value);`)

4.3 Exemples

1. Fer un codi HTML amb JavaScript que mostri un missatge amb el nom i versió del navegador, el SO que s'està fent servir i si tenim les cookies activades o no.

```
<SCRIPT LANGUAGE="javascript">
alert("Navegador = " + navigator.appName + "\nVersio = " + navigator.appVersion +
               "\nSO = " + navigator.platform + "\nCookies = " + navigator.cookieEnabled);
</SCRIPT>
```

2. Creeu un paràgraf de text. Apliqueu l'etiqueta SPAN sobre un fragment donat d'aquest per a aplicar-li un color diferent de la resta del paràgraf. Feu, fent servir JavaScript, que en posar el ratolí sobre el text resaltat s'obri un finestra que contingui la pàgina web http://www.google.es i en treure'l la finestra es tanqui.

```
<P>
Aquest és un paràgraf amb un span
<SPAN style="color:blue">
onMouseOver="ventana=window.open('http://www.google.es','hola','top=100,left=100,width=200,
height=200')" onMouseOut="ventana.close()">aquí</SPAN>
per a obrir i tancar una finestra
</P>
```

3. Creeu un botó i un link que en clicar-los, fent servir JavaScript, us permetin tancar la finestra.

```
<input type="button" onclick="window.close();" value="Tanca aquesta finestra">
<a href="#" onClick="window.close();">Tancar</A>
```

4. Teniu dos imatges. Per defecte es carrega una d'elles. Feu un link que en posar el ratolí a sobre permeti veure l'altre imatge en lloc de la per defecte i quan el ratolí es retiri torni a mostrar la per defecte.

```
<IMG NAME="imatge1" SRC="f1m1p1.jpg">
<A href="#" onMouseOver="document.imatge1.src='f1m1p2.jpg'"
                                    onMouseOut="document.imatge1.src='f1m1p1.jpg'">Canviar</A>
```

5. Feu el mateix que en l'exemple anterior però canviant la imatge en passar el ratolí per sobre i tornar-la a canviar quan aquest desaparegui.

```
<A href="#" onMouseOver="document.imatge1.src='f1m1p2.jpg'"
                                              onMouseOut="document.imatge1.src='f1m1p1.jpg'">
          <IMG NAME="imatge1" SRC="f1m1p1.jpg">
</A>
```

6. Creeu un link que us permeti canviar el color de fons de la finestra.

```
<A href="#" onClick="document.bgColor='red'">Canviar color</A>
```

7. El mateix que l'anterior, però si deixeu el ratolí sobre el link es manté vermell i qual el treieu es torna un altre cop blanc.

```
<A href="#" onMouseOver="document.bgColor='red'"
                           onMouseOut="document.bgColor='white'">Canviar color</A>
```

8. Feu que la pàgina web us doni un missatge sobre la vostra URL i la vostra resolució.

```
<SCRIPT LANGUAGE="javascript">
   alert("URL = " + document.URL + "\nResolucio = " + screen.height + "x" + screen.width);
</SCRIPT>
```

5. Variables JavaScript

5.1 Definició i declaració

Les variables són elements del codi que permeten l'emmagatzemament temporal de dades. Per fer-les servir és necessari posar nom a aquestes variables. Pot ser qualsevol que comenci amb caràcter o _ i no sigui una paraula reservada.

JavaScript és sensible a majúscules i minúscules, fet que vol dir que una variable anomenada `mevaVariable` serà diferent a una variable anomenada `MevaVariable`.

Podem declarar les variables de les següents maneres:

1. Assignant-li un valor directament. Exemple: `mevaVariable = 555;`
2. Fent servir la paraula reservada `var` sense assignar cap valor. Exemple: `var mevaVariable;`
3. Fent servir els mètodes 1 i 2 a la vegada. Exemple: `var mevaVariable = 555;`

En declarar una variable i donar-li un valor ja diem de quin tipus és aquesta variable:

- `var numDecimal = 3.14;` -- Variable numèrica. Punt flotant.
- `var text = "Joan";` -- Variable de text. Cadena de caràcters.
- `var repetidor = false;` -- Variable booleana. Valors true o false.
- `var num = 4;` -- Variable numèrica. Valor enter.

5.2 Tipus de variables

- Numèriques: Enterns, decimals, punt flotant, ...
- Textuals: Caràcters i cadenes de caràcters.
- Booleanes: `true` / `false`.
- Nul·les: No tenen valor fins que no li assignem. Són les variables declarades de la forma `var mevaVariable;`
- D'objecte: Emmagatzemen objectes predefinits a JavaScript. Per exemple una variable podria contenir l'objecte finestra.

5.3 Conversions

A vegades ens interessa convertir un tipus de dada en una altre. Per a realitzar la conversió entre tipus de dades farem servir:

- La conversió implícita. Es fa servir l'operador + de JavaScript.

  Exemple: `var DNI = 42126126;`

  LLETRA_NIF = "A";

  `NIF = DNI + LLETRA_NIF;` -- La variable NIF ara és una llista de caràcters.

- La conversió explícita. Es realitza mitjançant ordres de JavaScript. Es poden fer conversions entre els següents tipus:
  - Passar a nombre enter: `parseInt(variable[,base])`. Retorna el nombre enter que conté la variable en decimal. Si li passem la base, considera que el número està en aquella base i retorna el seu equivalent decimal. Si hi ha un error en la conversió retorna `NaN` (Not a Number).
  - Passar a nombre flotant: `parseFloat(variable)`. Permet passar la variable a un valor en coma flotant.
  - `isNaN(variable)`. Aquesta funció és molt important. Ens permet determinar si una conversió s'ha realitzat satisfactòriament o no. Si `isNaN(variable)` retorna `true` vol dir que variable no és un nombre i per tant que la conversió ha fallat. En canvi si retorna `false` la conversió s'ha realitzat amb èxit.

Veiem uns exemples:

1. Feu un script que ens demani un valor, emmagatzemeu aquest valor a una variable i després feu un `alert` que mostri el contingut d'aquesta variable.

```
<SCRIPT LANGUAGE="javascript">
    var a = prompt("Introdueix un valor","");
    alert(a);
</SCRIPT>
```

2. Creeu tres variables una textual, una numèrica i una altre booleana. Mostreu en un única alert totes aquestes variables.

```
<SCRIPT LANGUAGE="javascript">
    var a = "prueba";
    var b = 123.34;
    var c = true;
    alert(a + " " + b + " " + c);
</SCRIPT>
```

3. Creeu dues variables textuals. Una contindrà un número i l'altre una cadena textual. Feu un conversió a nombre enter i comproveu, mitjançant la funció isNaN, si la conversió ha estat correcte. Mostreu el resultat a la pàgina web.

```
<SCRIPT LANGUAGE="javascript">
    var a = "prueba";
    var b = "123.34";
    document.write(isNaN(parseInt(a)));
    document.write(isNaN(parseInt(b)));
</SCRIPT>
```

5.4 Variables matrius (arrays)

Les variables poden emmagatzemar més d'una dada fent servir matrius o arrays en funció de l'estructura:

Cel·la	0	1	2	...
Valor	valor1	valor2	valor3	...

Per crear un array podem fer servir dos mètodes:

1. `var temperatura = new array(30,28,27);` (i així doncs tenim l'array amb els valors `temperatura[0]` que val `30`, `temperatura[1]` que val `28` i `temperatura[2]` que val `28`)
2. `var temperatura = new array();` (i després assignar nosaltres els valors manualment `temperatura[0]=30`, `temperatura[1]=28` i `temperatura[2]=27`)

6. Funcions a JavaScript

6.1 Definició

Les funcions són un dels elements fonamentals del JavaScript. Una funció no és res més que un conjunt de sentències que realitzen una tasca específica. Convé definir-les en el HEAD per a poder-les fer servir a qualsevol punt de la web.

Definició d'un funció:

- S'ha de fer servir la paraula `function` més el `nom` de la funció i, entre parèntesi, els arguments que passem a la funció separats per comes.
- Les sentències corresponents a una funció es posen entre claus.
- Un funció pot cridar a:
  - Altres funcions.
  - A sí mateixa. Això s'anomena recursivitat.

6.2 Sintaxi

La sintaxi general d'uns funció és:

```
function
nom_funcio([parametre1[,parametre2[,...]]])
{
       Sentències JavaScript
       return valor;
}
```

La instrucció `return` permet a la funció retornar un valor. Així doncs les funcions poden rebre molts valors i retornar només un.

Exemples de funcions:

1.
```
funcion suma(a,b) {
    var resultat = a+b;
    return resultat;
```

```
}
2. funcion missatge(text) {
    alert(text);
}
3. funcion validaForm() {
    document.form1.caixa.value = "Validat";
}
```

4. Feu un script que us demani dos valors numèrics. I després cridi a una funció que rep aquests dos valors i retorni la suma d'aquests. Mostreu el resultat al document HTML.

```
<HEAD>
<SCRIPT LANGUAGE="javascript">
function sumar(a,b) {
    return a+b;
}
</SCRIPT>
</HEAD>
A la web
<SCRIPT LANGUAGE="javascript">
    var a = prompt("Valor1","");
    var b = prompt("Valor2","");
    // Son valors textuals
    a = parseInt(a);
    b = parseInt(b);
    document.write(sumar(a,b));
</SCRIPT>
```

7. Àmbit de les variables

Les variables poden existir en dos àmbits:

- Àmbit global: són variables que es declaren fora de qualsevol funció. Aquestes variables poden ser utilitzades des de qualsevol lloc de la pàgina. La seva vida és global, existiran mentre la pàgina estigui carregada al navegador.
- Àmbit local: són variables que es declaren dins d'una funció. Aquestes variables existiran només dins d'aquesta funció i desapareixeran quan s'acabi d'executar el codi. Per tant la seva vida està determinada per la vida de la funció.

Exemple d'àmbit d'un variable:

```
function a()
{
	var salutacio = “Hola Mundo”;
	alert(salutacio);
}
function b()
{
	alert(salutacio);
}

<INPUT TYPE=”button” VALUE=”Botó 1” name=”boto1” onClick=”a();”>
<INPUT TYPE=”button” VALUE=”Botó 2” name=”boto2” onClick=”b();”>
```

L'última no funcionaria, perquè `salutacio` és una variable local de la funció `a()`. Si la variable estigués definida fora de les funcions l'exemple seria correcte.

8. Operadors

Els operadors que podem aplicar sobre les variables a JavaScript són els següents:

8.1 Operadors d'assignació

Operador	Descripció	**Exemple d'ús**
=	Assigna el valor de l'operand de la dreta a l'operand de l'esquerra.	t = r ;
+ =	Assigna la suma dels operands de la dreta i l'esquerra a l'operand de la dreta.	t += r ; (t = t + r)
- =	Assigna la resta dels operands de la dreta i l'esquerra a l'operand de la dreta.	t -= r ; (t = t - r)
* =	Assigna el producte dels operands de la dreta i l'esquerra a l'operand de la dreta.	t *= r; (t = t * r)
/ =	Assigna el quocient dels operands de la dreta i l'esquerra a l'operand de la dreta.	t /= r; (t = t / r)
% =	Assigna el residu del quocient dels operands de la dreta i l'esquerra a l'operand de la dreta.	t %= r ; (t = residu(t/r))

8.2 Operadors aritmètics

Operador	Descripció	**Exemple d'ús**
Binaris:		
+	Suma el valor de dos operands.	a + b;
-	Resta el valors de dos operands.	a - b;
*	Multiplica el valor de dos operands.	a * b ;
/	Divideix els valors de dos operands.	a / b ;
%	Calcula la resta de la divisió de dos operands.	a % b;
Monaris:		
–	Canvia el signe de l'operand de la seva dreta.	- preu ;
++	Incrementa en un unitat l'operand de la seva dreta o esquerra.	++a ; a++;
– –	Disminueix en una unitat l'operand de la seva dreta o esquerra.	--a; a--;

8.3 Operadors lògics

Operador	Descripció	**Exemple d'ús**
&&	Operador binari. Fa l'operació "i lògica", retorna true si els dos operands són true.	a && b;
||	Operador binari. Fa l'operació "o lògica", retorna true si un dels dos operands és true.	a | | b;
!	Operador monari. Fa l'operació "negació lògica", retorna true si l'operador és false.	! a; !esMenor;

8.4 Operadors de comparació

Operador	Descripció	**Exemple d'ús**
= =	Retorna true si els dos operadors són iguals.	a = = b;
! =	Retorna true si els dos operadors són diferents.	a ! =b;
>	Retorna true si l'operador de l'esquerra és més gran que el de la dreta.	a > b;
<	Retorna true si l'operador de l'esquerra és més petit que el de la dreta.	a < b ;
> =	Retorna true si l'operador de l'esquerra és més gran o igual que el de la dreta.	a > = b ;

< =	Retorna true si l'operador de l'esquerra és més petit o igual que el de la dreta.	a < = b;

ATENCIÓ: Heu de fer servir parèntesis per a delimitar l'ordre de les operacions, per exemple `b+c·d` és diferent de `(b+c)·d`.

9. Estructures de control de flux

En programació moltes vegades és necessari definir un fluxe no lineal depenent de determinades circumstàncies. Hi ha dues classes d'estructures:

- Presa de decisió: Es decideix si s'ha d'executar un fragment de codi o no en funció d'unes determinades circumstàncies:
  - `if – else`
  - `switch`
- Bucles: Permet l'execució d'un conjunt d'instruccions un nombre determinat o indeterminat de vegades, però sempre finit.
  - `for`
  - `while`

9.1 Estructures de decisió

if - else

```
if (condicio) {
  // Sentències si la condició és certa
} else {
  // Sentències si la condició és falsa
}
```

Es pot fer servir sense la clàusula else.

Exemple:

```
var a=prompt("Clau","");
if (a == "Lluis") {
     alert("Hola Lluis");
} else {
     alert("No ets benvingut");
}
```

if - else if - ... - else

```
if (condicio1) {
  // Sentències si la condicio1 és certa
} else if (condicio2){
  // Sentències si la condicio2 és certa
  ... (tants else if com es vulguin)
} else {
  // Sentències si totes les condicions
  // anteriors són falses
}
```

Es pot fer servir sense la clàusula else.

Exemple:

```
var a=prompt("Numero","");
if (a<5) {
     alert("És més petit que 5");
} else if (a<10) {
     alert("Més gran o igual que 5 i més petit que 10");
} else {
     alert("És més gra o igual que 10");
}
```

Exemple: Ús dels operadors lògics

```
var a=prompt("Nombre1","");
var b=prompt("Nombre2","");
```

```
if ((a>10) && (b<5)) {
     alert("Combinació correcte");
} else {
     alert("Equivocat");
}
```

```
if ((a>10) || (b<5)) {
     alert("Combinació correcte");
} else {
     alert("Equivocat");
}
```

switch ... case

```
switch (expressio)
{
   case valor1:
      // Sentències
      [break;]
   case valor2:
      // Sentències
      [break;]
   ...
   default:
      // Sentències
}
```

Exemple:

```
var a = prompt("Número 1-4:","");
switch(a) {
   case "1":
      alert("Has triat 1");
      break;
   case "2":
      alert("Has triat 2");
      break;
   case "3":
      alert("Has triat 3");
      break;
   case "4":
      alert("Has triat 4");
      break;
   default;
      alert("Has triat malament");
}
```

9.2 Estructures iteratives

for – Es sap el nombre de vegades que hem d'executar un conjunt d'instruccions.

```
for (contador=valor; condicio; increment)
{
     // Sentències JavaScript
}
```

Exemple:

1.

```
var ciutat
  = new array("Barcelona","Tarragona","Lleida");
var temperatura = new array(30,30,20);
for (i=0; i<4; i++) {
     alert(ciutat[i] + "=" + temperatura[i]);
}
```

2.

```
var suma=0;
for (i=1; i<=100; i++) {
     suma = suma + i;
}
document.write(suma);
```

while / do while – No es sap el nombre de vegades que hem d'executar un conjunt d'instruccions.

```
while (condició) {
   // Sentències JavaScript
}

do {
   // Sentències JavaScript
} while(condició);
```

Exemple

```
var i=0;
while(i<=10) {
     document.write(i);
     document.write("<BR>");
     i++;
}
```

Exemple

```
var i=0;
do {
     document.write(i);
     document.write("<BR>");
     i++;
} while(i<=10);
```

Trencament o continuació d'un bucle

- `break`: Trenca el bucle. Fa que s'acabi la seva execució.
- `continue`: Salta les instruccions actuals del bucle i passa a les següents.

Exemple:

```
var i=0;
while(i<=10) {
     if (i == 5) {
         break;
     }
     document.write(i);
     document.write("<BR>");
     i++;
}
```

Exemple:

```
var i=0;
while(i<=10) {
     if (i == 5) {
         i++;
         continue;
     }
     document.write(i);
     document.write("<BR>");
     i++;
}
```

10. Accés als elements d'un document

Una de les formes més àgils i fàcils d'accedir als diferents elements i manipular les seves característiques és fent servir la funció `getElementById('id_del_element')`. Aquesta funció de l'objecte `document`, ens permet crear una variable de tipus objecte amb l'element del document amb l'`id` especificat.

Per exemple si al HTML hem creat:

```
<div id="capa1" style="position:absolute; top=100px; left=20px; border:1px solid;">
   Capa de <b>prova</b>
</div>
```

Podrem crear una variable a JavaScript de tipus objecte que contingui tota aquesta capa fent:

```
var a = document.getElementById("capa1");
```

I ara podrem modificar propietats de la capa. Per a modificar propietats de l'estil de la capa haurem de fer:

```
a.style.propietat = "valor";
```

per exemple si volem canviar la posició haurem de fer:

```
a.style.top = "150px";
```

ATENCIÓ: Les propietats dels estils canvien de format quan les fem servir a JavaScript, per exemple:

- `background-color` passa a anomenar-se `backgroundColor`.
- `border-style` passa a anomenar-se `borderStyle`.
- `font-size` passa a anomenar-se `fontSize`.
- ...

Podem també obtenir i modificar el codi HTML que conté un element fent servir la propietat `innerHTML`, per exemple, si fem `alert(a.innerHTML);` el resultat serà una finestra que contindrà: `Capa de <b>prova</b>`.

Exemple:

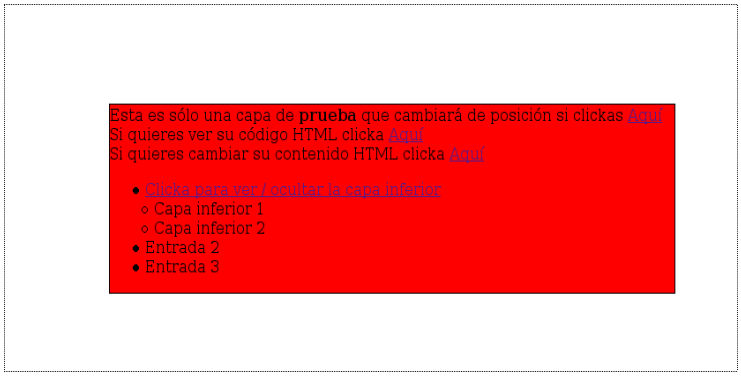
Codi HTML

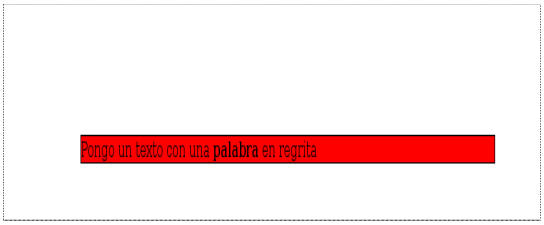
```
<html>
<head>
<title>Este es el titulo</title>
<script language="javascript">
   var seVe = true;
   function cambio() {
     var capa = document.getElementById("capa1");
     capa.style.backgroundColor="green";
     capa.style.top="200px";
```

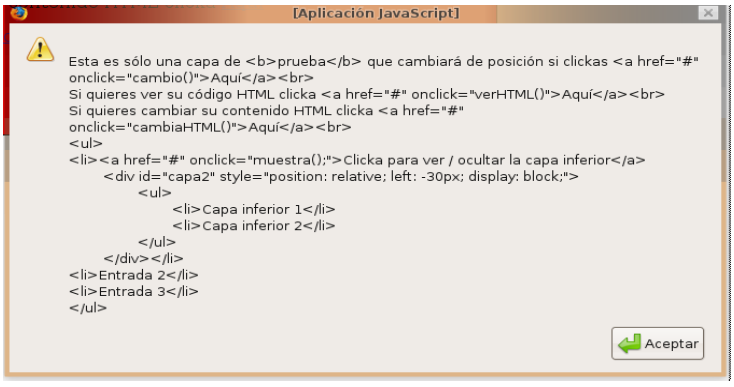
```
  }
  function verHTML() {
    var capa = document.getElementById("capa1");
    alert(capa.innerHTML);
  }
  function cambiaHTML() {
    var capa = document.getElementById("capa1");
    capa.innerHTML = "Pongo un texto";
    capa.innerHTML += "con una <b>palabra</b>";
    capa.innerHTML += "en regrita";
  }
  function muestra() {
    var subc = document.getElementById("capa2");
     if (seVe) {
       elemento.style.display="none";
    } else {
       elemento.style.display="block";
    }
    seVe = !seVe;
  }
</script>
</head>
<body>
<div id="capa1" style="position:absolute; top:100px; left:120px; width:50%;
                                                    border:1px solid; background-color:red;">
  Esta es s&oacute;lo una capa de <b>prueba</b> que
  cambiar&aacute; de posici&oacute;n si clickas
  <a href="#" onClick="cambio()"> Aqu&iacute; </a><br>

  Si quieres ver su c&oacute;digo HTML clicka
  <a href="#" onClick="verHTML()">Aqu&iacute;</a><br>

  Si quieres cambiar su contenido HTML clicka
  <a href="#" onClick="cambiaHTML()">Aqu&iacute;</a><br>



  <ul>
  <li>
    <a href="#" onClick="muestra();">Clicka para ver / ocultar la capa inferior</a>
```

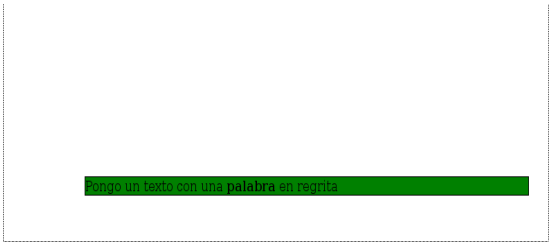
```
    <div id="capa2" style="position:relative; left:-30px; display:block;">
      <ul>
        <li>Capa inferior 1</li>
        <li>Capa inferior 2</li>
      </ul>
    </div>
  </li>
  <li>Entrada 2</li>
  <li>Entrada 3</li>
  </ul>
</div>
</body>
</html>
```

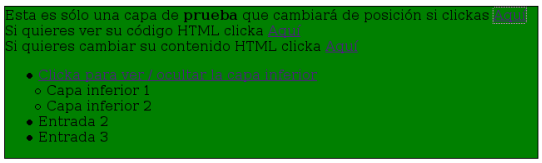
El resultat és el següent:

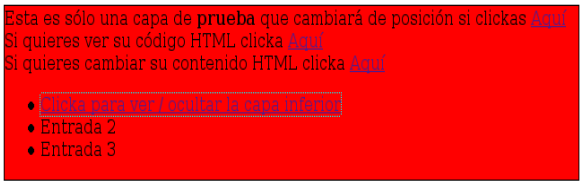












11.Classes importants de JavaScript

11.1 Classe `String`

Aquesta classe ens serveix per a manipular cadenes de caràcters. Estudiem les seves propietats i la llista de mètodes.

A JavaScript les variables de tipus text són objectes de la clase `String`. Això vol dir que una de les variables que creem de tipus text té una sèrie de propietats i mètodes. Recordem que les propietats són les característiques, com per exemple longitud de caràcters, i els mètodes són funcionalitats, com poden ser extreure una subcadena de text o posar un text a majúscules.

Per a crear un objecte de la classe `String` l'únic que hem de fer és assignar un text a una variable. El text va entre cometes. També es pot crear l'objecte `String` fent servir l'operador `new`.

11.1.1 Propietats de `String`

- `Length:` La classe `String` només té una propietat: `length`, que emmagatzema el nombre de caràcters del `String`.

11.1.2 Mètodes de `String`

Els objectes de la classe `String` tenen una bona quantitat de mètodes per a realitzar accions sobre ells.

- `charAt(index)`: Retorna el caràcter que hi ha en la posició indicada com a índex. La posició d'un `String` comença en 0.
- `indexOf(caràcter,des de)`: Retorna la posició de la primera vegada que apareix el caràcter indicat pel paràmetre. Si no troba el caràcter retorna -1. El segon paràmetre és opcional i serveix per a indicar a partir de quina posició es desitja que comenci la cerca.
- `lastIndexOf(caràcter,des de)`: Cerca la posició d'un caràcter exactament igual a com ho fa la funció `indexOf`, però des del final en lloc del principi. El segon paràmetre indica el nombre de caràcters des d'on es cerca.
- `replace(substring_a_cercar,novaStr)`: Implementat a JavaScript 1.2, serveix per a reemplaçar porcions del text d'un String per altre text, per exemple, podríem fer-lo servir per a reemplaçar totes les aparicions del Substring "xxx" por "yyy". El mètode no reemplaça en el String sinó que retorna una cadena resultant. Accepta expressions regulars.
- `split(separador)`: Només compatible amb JavaScript 1.1 en endavant. Serveix per a crear un vector a partir d'una String en el que cada element és la part del String que està separada pel separador indicat pel paràmetre.
- `substring(inicio,fin)`: Retorna la substring entre els dos índexs (posicions) proporcionats.
- `toLowerCase()`: Posa tots els caràcters del String en minúscules.
- `toUpperCase()`: Posa tots els caràcters del String en majúscules.
- `toString()`: Aquest mètode el tenen tots els objectes i es fa servir per a convertir-los en cadenes de text.

Exemple 1 de Strings:

Escriurem el contingut d'un String amb un caràcter separador ("-") entre cadascun dels seus caràcters.

```
var miString = "Hola Amigos"
var result = ""
for (i=0;i<miString.length-1;i++) {
  result += miString.charAt(i)
  result += "-"
}
result += miString.charAt(miString.length – 1)
document.write(result)
```

Exemple de Strings 2:

Farem un script que trenqui una cadena en dues meitats i les mostri per pantalla. Les meitats seran iguals sempre que la cadena tingui un nombre parell de caràcters. En cas que el nombre de caràcters sigui senar no es podrà fer per la meitat exactament i ho dividirem els més aproximadament possible.

```
var miString = "0123456789"
var mitad1,mitad2
posicion_mitad = miString.length / 2
mitad1 = miString.substring(0,posicion_mitad)
mitad2 = miString.substring(posicion_mitad,miString.length)
document.write(mitad1 + "<br>" + mitad2)
```

11.2 Classe `Date`

Explicarem la classe que es fa servir a JavaScript per a la manipulació de dates i hores.

Sobre aquest objecte recau tota la feina relacionada amb dates a JavaScript, com obtenir la data, el dia, l'hora i d'altres coses. Per a treballar amb dates necessitem instanciar un objecte de la classe `Date` i amb ell ja podrem realitzar les operacions que necessitem.

Un objecte de la classe `Date` es pot crear de dues maneres diferents. Per una banda podrem crear l'objecte amb el dia i l'hora actuals i per altre, podrem crear-lo amb un dia i una hora diferents a les actuals. Això depen dels paràmetres que passem en construir els objectes.

Per a crear un objecte data amb el dia i l'hora actuals col·locarem els parèntesis buits en cridar al constructor de la classe `Date`.

```
mevaData = new Date();
```

Per a crear un objecta data amb un dia i una hora diferents dels actuals hem d'indicar entre parèntesi el moment amb el qual inicialitzar l'objecte. Hi ha diferents maneres d'expressar un dia i una hora vàlides, per això podem construir una data guiant-nos per diferents esquemes. Aquests són dos, suficients per a crear tot tipus de dates i hores.

```
mevaData = new Date(any,mes,dia,hora,minuts,segons)
mevaData = new Date(any,mes,dia)
```

Els valors que ha de rebre els constructor són sempre numèrics. Un detall, el mes comença per 0, és a dir, gener és el mes 0. Si no indiquem l'hora, l'objecte data es crea amb hora 00:00:00.

Els objectes de la classe Date no tenen propietats però si un bon grapat de mètodes. Veiem-los:

11.2.1 Mètodes de `Date`

- `getDate()`: Retorna el dia del mes.
- `getDay()`: Retorna el dia de la setmana.
- `getHours()`: Retorna l'hora.
- `getMinutes()`: Retorna els minuts.
- `getMonth()`: Retorna el mes (atenció al mes que comença per 0).
- `getSeconds()`: Retorna els segons.
- `getTime()`: Retorna els milisegons transcorreguts entre el dia 1 de gener de 1970 i la data corresponent a l'objecte al que se li passa el missatge.
- `getYear()`: Retorna l'any, al que se li ha restat 1900. Per exemple, per al 1995 retorna 95, per al 2005 retorna 105. Aquest mètode està obsolet en Netscape a partir de la versió 1.3 de JavaScript i ara es fa servir `getFullYear()`.
- `getFullYear()`: Retorna l'any amb tots els dígits. És recomanable fer servir aquest mètode en lloc de l'anterior.
- `setDate()`: Actualitza el dia del mes.
- `setHours()`: Actualitza l'hora.
- `setMinutes()`: Canvia els minuts.
- `setMonth()`: Canvia el mes (atenció al mes que comença per 0).
- `setSeconds()`: Canvia els segons.
- `setTime()`: Actualitza la data completa. Rep un número de milisegons des del 1 de gener de 1970.
- `setYear()`: Canvia l'any. Rep un nombre, al que li suma 1900 abans de col·locar-lo com a any de la data. Per exemple, si rep 95 col·locarà l'any 1995. Aquest mètode està obsolet a partir de JavaScript 1.3 en Netscape. Ara es fa servir `setFullYear()`, indicant l'any amb tots els dígits.
- `setFullYear()`: Canvia l'any de la data al número que rep com a paràmetre. El nombre s'indica complet.

Exemples:

En aquest exemple crearem dues dates, una amb l'instant actual i l'altre amb data del passat. Després les imprimirem les dues i extraurem el seu any per a imprimir-lo també. Després actualitzarem l'any d'un de les data i la tornarem a escriure amb un format més llegible.

```
// Creem les dates
miFechaActual = new Date()
miFechaPasada = new Date(2006,4,23)
// Imprimim les dates
document.write (miFechaActual)
document.write ("<br>")
document.write (miFechaPasada)
// Prenem l'any de les dates
anoActual = miFechaActual.getFullYear()
anoPasado = miFechaPasada.getFullYear()
// Escrivim l'any a la pàgina
document.write("<br>El año actual es: " + anoActual)
document.write("<br>El año pasado es: " + anoPasado)
// Canviem l'any a la data actual
miFechaActual.setFullYear(2005)
// Prenem el dia el mes i l'any
dia = miFechaActual.getDate()
mes = parseInt(miFechaActual.getMonth()) + 1
ano = miFechaActual.getFullYear()
// Escrivim la data en un format llegible
document.write ("<br>")
document.write (dia + "/" + mes + "/" + ano)
```

11.2.2 Funcions de control del temps

Hi ha dues funcions que ens permetran executar funcions passat un cert interval de temps. Aquestes dues funcions són:

- `setTimeout()`: Executa una funció només una vegada passat un nombre determinat de segons. La seva sintaxi és `setTimeout("expressió", temps);` A on `expressió` és, en general, el nom d'una funció i `temps` és l'espera que hem de fer abans d'executar la funció, està expressat en milisegons.
- `setInterval()`: Executa una funció una vegada i una altre passat un nombre determinat de segons. La seva sintaxi és `setInterval("expressió", temps);` A on `expressió` és, en general, el nom d'una funció i `temps` és l'interval de temps que hem d'esperar abans de tornar a executar la funció, està expressat en milisegons.

Exemples:

```
alert("Hola");
setTimeout("alert('Adeu')", 5000);
```

Ens mostraria `Hola` amb un `alert` i, després de tancar la finestra anterior i esperar 5 segons, ens mostraria `Adeu` amb un altre `alert`.

```
setInterval("alert('Hola')",5000);
```

Ens mostrara `Hola` amb un `alert` cada 5 segons.

11.3 Classe `Math`

És la classe que farem servir per a realitzar càlculs matemàtics de tot tipus.

La classe `Math` proporciona els mecanismes per a realitzar operacions matemàtiques a JavaScript. Algunes operacions es resolen ràpidament amb els operadors aritmètics que ja coneixem, com la multiplicació o la suma, però hi ha una sèrie d'operacions matemàtiques addicionals que es tenen que realitzar fent servir la classe `Math` com poden ser calcular un sinus o fer una arrel quadrada.

Així doncs, per a qualsevol càlcul matemàtic complex farem servir la classe `Math`, amb una particularitat: fins ara cada vegada que volíem fer alguna cosa amb una classe havíem d'instanciar un objecte de la classe i treballar amb ell. En el cas de la classe `Math` es treballa directament amb la classe. Això es permet per que les propietats i mètodes de la classe `Math` són el que s'anomena propietats i mètodes de classe, i per a fer-los servir s'opera a través de la classe en lloc dels objectes. Dit d'una altre manera, per a treballar amb la classe `Math` no haurem de fer servir la instrucció `new` i farem servir el nom de la classe per a accedir a les seves propietats i mètodes.

11.3.1 Propietats de `Math`

Les propietats guarden valors que probablement necessitem en algun moment si estem fent càlculs matemàtics.

- `E`: Nombre E o constant d'Euler, la base dels logaritmes neperians.
- `LN2`: Logaritme neperià de 2.
- `LN10`: Logaritme neperià de 10.
- `LOG2E`: Logaritme en base 2 de E.
- `LOG10E`: Logaritme en base 10 de E.
- `PI`: Nombre PI.
- `SQRT1_2`: Arrel quadrada d'un mig.
- `SQRT2`: Arrel quadrada de 2.

11.3.1 Mètodes de `Math`

Així mateix, tenim una sèrie de mètodes per a realitzar operacions matemàtiques típiques, encara una mica més complexes.

- `abs()`: Retorna el valor absolut d'un nombre. (Valor sense signe).
- `acos()`: Retorna l'arccosinus d'un nombre en radians.
- `asin()`: Retorna l'arcsinus d'un nombre en radians.
- `atan()`: Retorna l'arctangent d'un número.
- `ceil()`: Retorna l'enter igual o inmediatament següent d'un número. Per exemple, ceil(3) és 3 i ceil(3.4)

és 4.

- `cos()`: Retorna el cosinus d'un nombre.
- `exp()`: Retorna el resultat d'elevar el nombre E a un número.
- `floor()`: El contrari de ceil() ja que retorna un nombre igual o inmediatament inferior.
- `log()`: Retorna el logaritme neperià d'un número.
- `max()`: Retorna el més gran de 2 números.
- `min()`: Retorna el més petit de 2 números.
- `pow()`: Rep dos nombre com a paràmetres i retorna el primer aixecat al segon.
- `random()`: Retorna un nombre aleatoria entre 0 i 1.
- `round()`: Arrodoneix a l'enter més pròxim.
- `sin()`: Retorna el sinus d'un nombre en radians.
- `sqrt()`: Retornar l'arrel quadrada d'un número.
- `tan()`: Calcula i retorna la tangent d'un número en radians.

Exemple:

```
document.write (Math.cos(2 * Math.PI))
document.write ("<br>")
document.write (Math.sin(2 * Math.PI))
```

12. Accés i control dels elements d'un formulari

12.1 Caixes i àrees de text

Les caixes i les àrees de text es tracten fàcilment i la seva propietat més important és `value`. Veiem un exemple, suposem que tenim el següent formulari:

```
<form name="dades" method="post" action="">
        Nom: <input type="text" name="nom"><br>
        Cognom: <input type="text" name="cognom"><br>
        Comentaris:<br>
        <textarea name="comentaris"></textarea><br>
        <input type="buttton" value="Veure valors" onClick="valors()">
</form>
```

Per accedir als valors introduits per l'usuari als camps de text haurem de fer la següent funció JavaScript:

```
function valors() {
	var nom = document.dades.nom.value;
	var cognom = document.dades.cognom.value;
	var comentaris = document.dades.comentaris.value;
	alert("El teu nom és: " + nom);
	alert("El teu cognom és: " + cognom);
	alert("Els teus comentaris són: " + comentaris);
}
```

12.1.1 Propietats

- `defaultValue`: Conté el valor per defecte de l'àrea de text.
- `value`: Que conté el text que hi ha escrit en l'àrea de text.

12.1.2 Mètodes

- `blur()`: Per a treure el focus de l'aplicació de l'àrea de text.
- `focus()`: Per a posar el focus de l'aplicació a l'àrea de text.
- `select()`: Selecciona el text de l'àrea de text.

  Si el que volem és obtenir la longitud del text farem servir la propietat `length` de la següent forma.

```
var longitud = document.dades.nom.value.length;
```

12.2 Caixes de selecció o `ckeckboxes`

Amb JavaScript, a partir de la jerarquia d'objectes del navegador, tenim accés als `checkboxes`, que depen de l'objecte `form` del formulari a on es troba inclós.

12.2.1 Propietats

- `checked`: Informa de si el `checkbox` està marcar o no. Pot ser `true` o `false`.
- `defaultChecked`: Informa si el checkbox està marcat per defecte o no.
- `value`: El valor actual del `checkbox`.

12.2.2 Mètodes

La llista dels mètodes d'un `checkbox` és la següent:

- `click()`: Es como si fesim un click sobre el `checkbox`, es a dir, canvia l'estat del checkbox.
- `blur()`: Retira el focus del `checkbox`.
- `focus()`: Col·loca el focus en el `checkbox`.

Per a il·lustrar el funcionament dels `checkbox` anem a veure una pàgina que té un `checkbox` i tres botons. Els dos primers per a mostrar les propietats `checked` i `value` i el tercer per a invocar al seu mètode `click()` amb l'objectiu de simular un click sobre el `checkbox`.

```
<html>
<head>
	<title>Exemple Checkbox</title>
<script>
function alertaChecked(){
	alert(document.miFormulario.miCheck.checked)
}
function alertaValue(){
	alert(document.miFormulario.miCheck.value)
}
function metodoClick(){
	document.miFormulario.miCheck.click()
}
</script>
</head>
<body>
	<form name="miFormulario" action="mailto:promocion@guiarte.com" enctype="text/plain">
	<input type="checkbox" name="miCheck" value="El meu checkbox">
	<br>
	<input type="button" value="informa de su propiedad checked" onclick="alertaChecked()">
	<input type="button" value="informa de su propiedad value" onclick="alertaValue()">
	<br>
	<input type="button" value="Simula un click" onclick="metodoClick()">
	</form>
</body>
</html>
```

12.3 Botons de ràdio o `radiobuttons`

Quan en una pàgina tenim un conjunt de botons de radio es crea un objecte radio per a cada botó. Els objectes radio depenen del formulari i es pot accedir a ells per l'array d'elements, però també es crea un array amb botons de radio. Aquest array depèn del formulari i té el mateix nom que els botons de radio.

12.3.1 Propietats

Veiem una llista de les propietats d'aquest element.

- `checked`: Indica si està marcat o no un botó de radio.
- `defaultChecked`: El seu estat per defecte.

- `value`: El valor del camp de radio, assignat per la propietat value del radio.
- `Length` (com a propietat de l'array de radios): El nombre de botons de radio que formen part en el grup. Accessible en el vector de radios.

12.3.2 Mètodes

Són els mateixos que tenia l'objecte `checkbox`.

Exemple:

Veiem amb un exemple el mètode de treball amb el `radio buttons` en el que col·locarem uns quants i cadascun tindrà associat un color. També hi haurà un botó i en pulsar-lo canviarem el color de fons de la pantalla al color que estigui seleccionat en el conjunt de botons de radio.

```
<html>
<head>
	<title>Exemple Radio Button</title>
<script>
function cambiaColor(){
	var i
	for (i=0;i<document.fcolores.colores.length;i++){
	if (document.fcolores.colores[i].checked)
		break;
	}
	document.bgColor = document.fcolores.colores[i].value
}
</script>
</head>
<body>
	<form name=”fcolores”>
	<input type="Radio" name="colores" value="ffffff" checked> Blanco
	<br>
	<input type="Radio" name="colores" value="ff0000"> Rojo
	<br>
	<input type="Radio" name="colores" value="00ff00"> Verde
	<br>
	<input type="Radio" name="colores" value="0000ff"> Azul
	<br>
	<input type="Radio" name="colores" value="ffff00"> Amarillo
	<br>
	<input type="Radio" name="colores" value="00ff00"> Turquesa
	<br>
	<input type="Radio" name="colores" value="ff00ff"> Morado
```

```
        <br>
        <input type="Radio" name="colores" value="000000"> Negro
        <br>
        <input type="Button" name="" value="Cambia Color" onClick="cambiaColor()">
        </form>
</body>
</html>
```

Primer ens fixem en el formulari i en la llista de botons de radio. Tots s'anomenen `colores`, així que estan associats a un mateix grup. A més, veiem que l'atribut `value` de cada botó canvia. També podem observar un botó a sota de tot.

Amb aquesta estructura tindrem un array de botons de radio que s'anomena `colores` y depen del formulari, accessible de la següent manera manera:

```
document.form.colores
```

Aquest array conté en cada posició un dels botons de radio. Així a la posició 0 es troba el botó de color blanc, en la posició 1 el de color vermell, ... Per a accedir a aquests botons de radio hem de fer servir l'índex corresponent:

```
document.fcolores.colores[i]
```

Si volem accedir, per exemple, a la propietat value de l'últim botó de radio hem d'escriure:

```
document.fcolores.colores[7].value
```

La propietat `length` de l'array de radios ens indica el nombre de botons de radio que formen part del grup.

```
document.fcolores.colores.length
```

En el nostre cas la propietat `length` val 8.

La funció `cambiaColor()` defineix una variable en la que introduirem l'índex del radio button que tenim seleccionat. Per a això anem recorrent l'array de botons de radio fins que trobem el que té la seva propietat `checked` a `true`. En aquest moment sortim del bucle, i per tant la variable `i` emmagatzema l'índex del botó de radio seleccionat. En la darrera línia canviem el color de fons al valor contingut a l'atribut value del radio button seleccionat.

12.4 Llistes de selecció

12.4.1 Propietats

Veiem la llista de les propietats d'aquest element del formulari.

- `length`: Emmagatzema la quantitat d'opcions del camp select, és a dir, la quantitat d'etiquetes `<OPTION>`.
- `option`: Fa referència a cadascuna de les opcions. Són per si mateixes, objectes.
- `options`: Un array amb cadascuna de les opcions del `select`.

- `selectedIndex`: És l'índex de la opció que es troba seleccionada.

12.4.2 Mètodes

Els mètodes són només 2:

- `blur()`: Per a treure el focus d'aquest element del formulari.
- `focus()`: Per a posar el focus en aquest element del formulari.

12.4.3 Propietats de l'objecte `option`

Ens hem de fixar en aquest objecte per entendre bé el camp select. Recordem que les option són les diferents opcions que té un select, expressades amb l'etiqueta `<OPTION>`.

Aquest objectes només tenen propietats, no tenen mètodes, i són les següents:

- `defaultSelected`: Indica amb un `true` o un `false` si aquesta opció és la opció per defecte. La opció per defecte s'aconsegueix afegint l'atribut `selected` a un `option`.
- `index`: És l'índex d'aquesta opció dins del `select`.
- `selected`: Indica si aquesta opció es troba sel·leccionada o no.
- `text`: És el text de la opció. El que pot veure l'usuari en el select, que s'escriu després de l'etiqueta `<OPTION>`.
- `value`: Indica el valor de la opció que s'introdueix a l'atribut `VALUE` de l'etiqueta `<OPTION>`.

Exemple d'accés a un `select`:

Anem a veure un exemple sobre com s'accedeix a un select amb JavaScript, a les seves propietats i a la opció seleccionada.

Comencem creant el formulari que té el `select` amb el que treballarem. És un `select` que serveix per a valorar el web.

```
<form name="fomul">
        Valoració sobre aquest web:
        <select name="miSelect">
        <option value="10">Muy bien
        <option value="5" selected>Regular
        <option value="0">Muy mal
        </select>
        <br>
        <input type=button value="Dime propiedades" onclick="dimePropiedades()">
</form>
```

Ara construim una funció que recull les propietats més significatives del camp `select` i les presenta en una caixa `alert`.

```
function dimePropiedades(){
	var texto
	texto = "El numero de opciones del select: " + document.formul.miSelect.length;
	var indice = document.formul.miSelect.selectedIndex;
	texto += "\nIndice de la opcion escogida: " + indice;
	var valor = document.formul.miSelect.options[indice].value;
	texto += "\nValor de la opcion escogida: " + valor;
	var textoEscogido = document.formul.miSelect.options[indice].text
	texto += "\nTexto de la opcion escogida: " + textoEscogido
	alert(texto)
}
```

Aquesta funció crea una variable de text a on va introduint cadascuna de les propietats del `select`. La primera conté el valor de la propietat `length` del `select`, la segona l'índex de la opció seleccionada i les dues següents contenen el valor i el text de la opció seleccionada. Per a accedir a la opció seleccionada fem servir l'array options amb l'índex recollir en la segona variable.

13. Expressions regulars per a validar formularis

Què és una expressió regular? Una expressió regular és un patró que pot concordar amb diferents cadenes de text. La seva utilitat a JavaScript és la de comprovar si l'usuari ha introduït valors correctes als camps de text dels formularis.

Per exemple, si demanem a l'usuari el seu email, és imprescindible que introdueixi una cadena de text de la forma xxxxxxx@xxxxxx.xxx. Evidentment si l'usuari no introdueix un símbol @ o no posa el domini al final de l'email aquest no serà vàlid.

Les expressions regulars ens permeten crear patrons que s'ajustin a un conjunt molt ampli de cadenes de caràcters i així poder-les comprovar totes de cop.

Quins són els caràcters especials per a la creació d'expressions regulars?

Oper.	Descripció	Oper.	Descripció	Oper.	Descripció
c	Caràcter *c* no especial. Coincideix amb si mateix. En expressions regulars bàsiques.	*c*	Caràcter especial *c* (\: caràcter de escapament). En expressions regulars bàsiques.	^*X*	Començar amb *X*. En expressions regulars bàsiques.
X$	Finalitzar amb *X*. En expressions regulars bàsiques.	*X**	*X* cero o més vegades. En expressions regulars bàsiques.	.	Un caràcter indivudual cualquiera. En expressions regulars bàsiques.
[*c1c2c3*]	Conjunto de caràcters. Coincideix si *c1* o *c2* o *c3*. En expressions regulars bàsiques.	[^*c1c2c3*]	Coincideix amb caràcters diferents de *c1*, *c2* o *c3*. En expressions	[*c1-c2*]	Rang de caràcters. Coincideix amb qualsevol caràcter entre *c1* i *c2*. En expressions

			regulars bàsiques.		regulars bàsiques.
[^*c1*-*c2*]	Coincideix amb caràcters no compresos entre *c1* y *c2*. En expressions regulars bàsiques.	*XY*	Concatenació. Coincideix si *X* va per davant de *Y*. En expressions regulars bàsiques.	*X*+	*X* una o més vegades. En expressions regulars exteses.
X?	*X* cero o una vegada. En expressions regulars exteses.	(*X*)	Agrupa *X*. (*A més, en JavaScript, grava en RegExp. $1...*). En expressions regulars exteses.	*X*|*Y*	Alternativa. Coincideix amb *X* o *Y*. En expressions regulars exteses.
X{*n*}	*X* exactamente *n* vegades. En expressions regulars exteses.	*X*{*n*,}	*X* al menys *n* vegades. En expressions regulars exteses.	*X*{*m*,*n*}	*X* de *m* a *n* vegades. En expressions regulars exteses.
"*cad*"	Literal. Ignora els caràcters especials de *cad*. (*No se puede fer servir a JavaScript*) En expressions regulars exteses.	(?:*X*)	Coincideix amb *X* pero no el captura. En expressions regulars JavaScript.	*X*(?=*Y*)	Coincideix anb *X* si va per davant de *Y*. En expressions regulars JavaScript.
X(?!=*Y*)	Coincideix amb *X* si no va per davant de *Y*. En expressions regulars JavaScript.	*X*\b	*X* al final d'una paraula. En expressions regulars JavaScript.	*X*\B	*X* no està al final d'una paraula. En expressions regulars JavaScript.
\c*tec*	Caràcter de control Ctrl+*tec*. En expressions regulars JavaScript.	\d	Caràcter numèric. Equival a [0-9]. En expressions regulars JavaScript.	\D	Caràcter no numèric. Equivale a [^0-9]. En expressions regulars JavaScript.
\s	Espai en blanc (separador). Equival a [\f\n\r\t\v]. En expressions regulars JavaScript.	\S	Espai no blanc (separador). Equival a [^\f\n\r\t\v]. En expressions regulars JavaScript.	\w	Caràcter alfanumèric. Equival a [A-Za-z0-9_]. En expressions regulars JavaScript.
\W	Caràcter no alfanumèric. Equival a [^A-Za-z0-9_]. En expressions regulars JavaScript.	*num*	Referència cap enradera a subcadenes capturades (*num* sencer). En expressions regulars JavaScript.	\x*hh*	Caràcter el codi en hexadecimal es *hh*. En expressions regulars JavaScript.

Expressions regulars bàsiques

Expressions regulars esteses

Expressions regulars adicionales de JavaScript (inclou també las prèvies).

Veiem alguns exemples:

Expressió regular	Cadena de caràcters
`1*234`	234, 1234, 11234, 111234, ... (o qualsevol cadena de caràcters que la contingui)
`p+sado`	psado, ppsado, pppsado, ... (o qualsevol cadena de caràcters que la contingui)
`p{2}sado`	ppsado (o qualsevol cadena de caràcters que la contingui)
`p{2,}sado`	ppsado, pppsado, ppppsado, ... (o qualsevol cadena de caràcters que la contingui)
`p{2,4}sado`	ppsado, pppsado, ppppsado (o qualsevol cadena de caràcters que la contingui)
`^p{2,4}sado$`	Estrictament: ppsado, pppsado, ppppsado
`^tal{1,2}a`	Les cadenes de caràcters que comencin amb tala o talla i que continuïn amb qualsevol altre cadena de caràcters

Si voleu validar les vostres expressions regulars feu servir el següent enllaç http://regexpal.com/

Veiem un exemple pràctic. Imaginem que demanem a l'usuari el seu DNI amb la seva lletra. Crearem una expressió regular per a validar les dades introduïdes per l'usuari.

Expressió regular: `^[0-9]{8}[a-zA-Z]$` Aquesta expressió ha de començar amb 8 caràcters numèrics i finalitzar amb un caràcter alfabètic (ja sigui en minúscules o majúscules).

Com fem servir això a JavaScript? Veiem el següent exemple:

```
<html>
<head>
<title>Hola que tal</title>
<script language="javascript">
function validar()
{
        var patro = /^[0-9]{8}[a-zA-Z]$/;
        var posibleDni = document.fexp.dni.value;
        if (posibleDni.search(patro) != -1) {
                alert ("El DNI sembla correcte");
        } else {
```

```
                alert ("El DNI sembla incorrecte");
        }
}
</script>
</head>
<body>
<form action="pepe.php" method="post" name="fexp">
        Introdueix el DNI: <input type="text" name="dni">
        <input type="button" name="valida1" value="Validar" onClick="validar()">
</form>
</body>
</html>
```

Com podem veure per a crear una expressió regular crearem una variable JavaScript i li assignarem l'expressió regular entre els símbols /.

Després, per a comprovar si un text s'ajusta a l'expressió regular farem servir el mètode `search`. Aquest mètode ens retornara -1 si l'expressió regular no s'ajusta al text o un nombre més gran o igual a 0 que indica la posició en la que el patró de l'expressió regular coincideix amb el nostre text.