# Introducció a AJAX

Jordi Sánchez Cano

PID_00172685

UOC
Universitat Oberta de Catalunya

# Índex

# Introducció

AJAX (*asynchronous Javascript and XML*) és un terme que engloba un conjunt de tecnologies i tècniques que permeten desenvolupar webs més rics i interactius. És un terme molt associat a d'altres com *RIA* (*rich Internet applications*), *Web 2.0* i *aplicacions web*.

En aquest mòdul s'explica què és AJAX, els seus beneficis i les tecnologies que empra. Entre les tecnologies utilitzades per AJAX, es troba JavaScript, un dels pilars fonamentals. Per aquest motiu, se'n fa un repàs breu.

# Objectius

Els objectius d'aquest mòdul didàctic són:

1. Conèixer què és AJAX i les diferents tecnologies que utilitza.

2. Entendre els beneficis de l'ús d'AJAX en aplicacions web sobre el model tradicional.

3. Repassar alguns conceptes bàsics de JavaScript.

# 1. Introducció a la programació web

En aquest apartat veurem alguns conceptes bàsics sobre la programació web.

## 1.1. El Web i la seva evolució

En els seus inicis, el Web es basava en simples pàgines HTML en què l'usuari navegava de l'una a l'altra mitjançant hipervincles. Cada acció que feia un usuari desencadenava la càrrega d'una nova pàgina i per això se'n diu *navegar*. Aquesta primera etapa del Web es va denominar *Web 1.0*, en què les pàgines eren estàtiques i eren actualitzades per l'entitat o persona que les publicava.

El Web 2.0 es refereix a una segona generació que engloba tant l'avenç tecnològic com el canvi de concepte que comporta. Aquest terme redefineix el Web com un conjunt de serveis i una plataforma, en què ara els usuaris són partícips dels seus continguts.

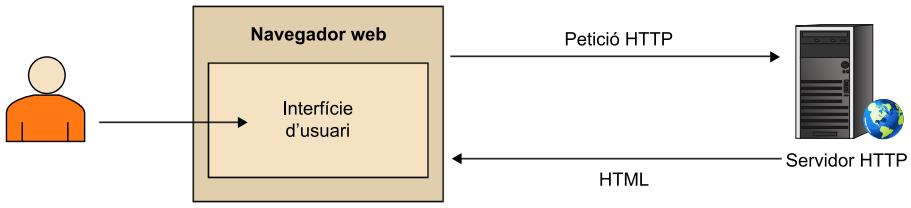
Aquest canvi sobre el concepte del Web no hauria estat possible sense una evolució tecnològica prèvia, en què AJAX té un paper molt important i és un dels termes que més acompanya el Web 2.0.

## 1.2. Què és AJAX

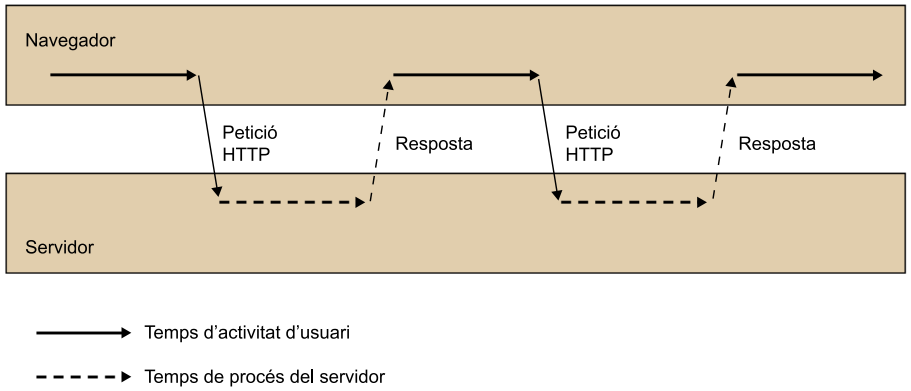
> Conceptualment podem definir AJAX com el conjunt de tècniques i tecnologies que permeten desenvolupar aplicacions web interactives o RIA i oferir una interacció, funcionalitat, accessibilitat i rendiment millors.

AJAX permet que els usuaris continuïn interactuant amb una mateixa pàgina mentre es manté la comunicació amb el servidor asíncronament, alhora que s'actualitzen els continguts. Això comporta que les dades s'obtenen i s'actualitzen en segon pla sense interferir amb l'ús normal de l'aplicació.

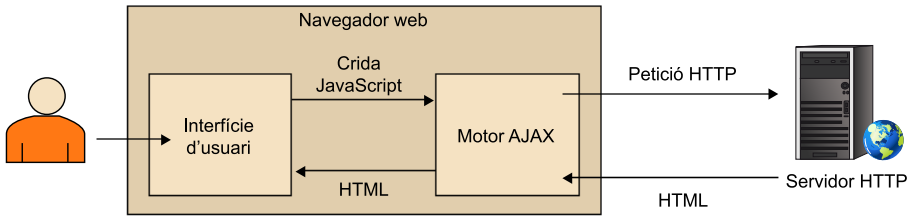
La majoria d'accions generades per l'usuari en el model tradicional d'aplicació web desencadenen una petició HTTP que sol·licita una pàgina nova al servidor. El servidor processa la pàgina i l'envia de tornada al navegador tal com es mostra en l'esquema següent.



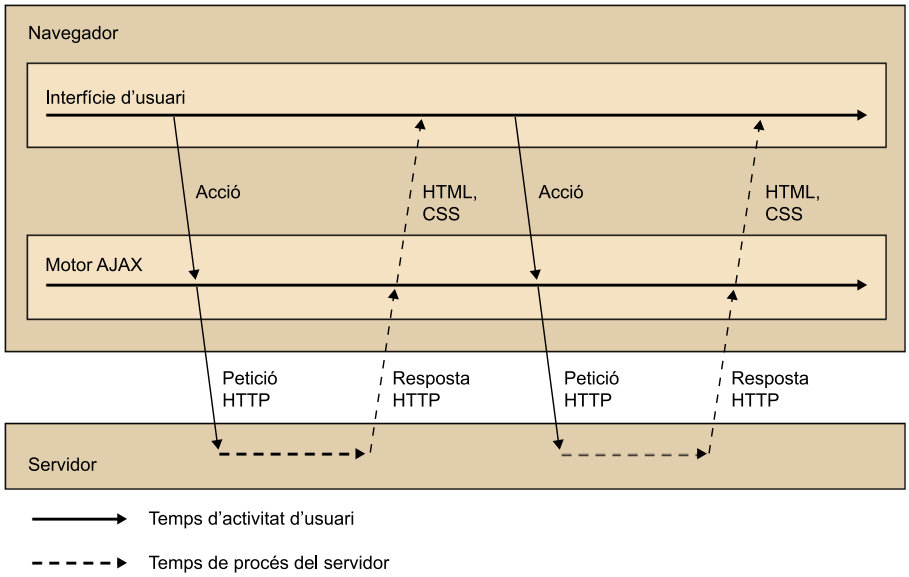

Des que s'envia la petició HTTP fins que es rep i renderitza la pàgina nova, l'usuari roman a l'espera sense poder fer ús de l'aplicació. El diagrama de seqüència següent mostra la disponibilitat del navegador entre cada canvi de pàgina.




La càrrega i espera es repeteix cada vegada que cal actualitzar una part de la pàgina. Les aplicacions que fan ús d'AJAX trenquen amb aquest model i permeten la comunicació amb el servidor i la recàrrega de la pàgina asíncronament, de manera que l'usuari pot continuar treballant amb l'aplicació sense interrupcions. La figura següent mostra interacció asíncrona entre l'usuari i el servidor.




El motor AJAX és compost per un conjunt de funcions JavaScript que gestiona la comunicació amb el servidor i efectua els canvis necessaris en la interfície d'usuari. Les accions que l'usuari duu a terme són delegades al motor AJAX i, mentre s'actualitzen els nous continguts, la interfície d'usuari continua disponible com es mostra en el diagrama següent.




En el passat ja es feia ús d'aquestes tècniques i tecnologies però no va ser fins al 18 de febrer del 2005 quan Jesse James Garrett va definir el terme AJAX per primera vegada en un article anomenat "AJAX: A New Approach to Web Applications".

**Lectura recomanada**

Es pot consultar l'article "AJAX: A New Approach to Web Applications" en la pàgina d'*Adaptive path*.

En aquest article es defineix AJAX com un model d'aplicació web que fa ús del conjunt de tecnologies següent:

- **XHTML** i **CSS**, utilitzades per a la presentació.
- **Document object model**, per a la modificació dinàmica de la presentació d'una pàgina.
- **XML** i **XSLT**, per a l'intercanvi i manipulació de dades.
- **JavaScript**, com a llenguatge per a l'execució i l'accés a les tecnologies anteriors.

Cadascuna d'aquestes tecnologies són estàndards oberts, per la qual cosa és possible usar AJAX en múltiples navegadors i plataformes.

#### Llocs web que usen AJAX

Cada vegada hi ha més llocs web per la Xarxa que fan ús d'AJAX. Uns dels primers i principals portals que el van utilitzar va ser Google al seu cercador i a Gmail.

#### Google Search

El cercador de Google fa ús d'AJAX en el quadre de text per a introduir els termes de cerca. Mentre hi escrivim, es desplega una llista amb suggeriments semblants a la part escrita. A mesura que continuem escrivint la llista s'actualitzarà amb suggeriments nous. Els suggeriments obtinguts s'obtenen d'una manera asíncrona sense interferir en la interacció de l'usuari.

#### Gmail

El client web de correu de Google fa ús d'AJAX en diferents parts de l'aplicació. Durant l'ús de tota l'aplicació, romandrem dins de la mateixa pàgina. A més, el motor AJAX de Gmail comprova el correu periòdicament i, si rep un correu nou, actualitza la vista amb els canvis necessaris.

### 1.3. Avantatges sobre l'ús d'AJAX

Alguns dels avantatges sobre l'ús d'AJAX en aplicacions web són els següents:

- Redueix el trànsit de les aplicacions web, atès que en cada moment envia només les dades necessàries.

- Redueix el temps d'espera, ja que cal enviar i rebre menys informació.

- En no recarregar la pàgina, l'usuari té la sensació de treballar sempre amb la mateixa aplicació, i no amb pàgines diferents.

- Permet millorar la usabilitat de llocs web, sobretot en llocs que contenen molts formularis i molt complexos (amb camps del formulari que depenen del valor d'altres camps del formulari: província, poble, codi postal...).

- Permet millorar la percepció que l'usuari té del lloc web: l'usuari té la impressió d'estar treballant en una aplicació d'escriptori.

- Permet reduir la dificultat a l'hora de crear pàgines web complexes, si es reutilitzen controls AJAX existents.

D'altra banda, desenvolupar pàgines que facin ús d'AJAX requereix més coneixements sobre les tecnologies que utilitza. Aquestes pàgines hauran de controlar els possibles errors que puguin ocórrer en dur a terme les operacions asíncrones. Un altre inconvenient és que alguns navegadors poden tenir JavaScript deshabilitat per la qual cosa les parts asíncrones quedaran inservibles.

# 2. Tecnologies utilitzades per AJAX

## 2.1. El protocol HTTP

HTTP és el protocol utilitzat en la comunicació web per a la transferència d'arxius, generalment HTML. Mitjançant HTTP, se sol·liciten i s'obtenen les diferents pàgines i els recursos necessaris entre el navegador i servidor.

Per a accedir a aquests recursos s'utilitza la seva adreça representada mitjançant URL[1] o localitzador uniforme de recurs.

(1) *URL* és la sigla d'*uniform resource locator*.

HTTP està orientat a transaccions, en què cadascuna és formada per una petició del client i una resposta del servidor. En cada transacció, s'estableix una connexió nova alhora que s'envia la petició. La connexió es tanca una vegada el servidor envia la resposta al client. També és possible intercanviar diferents peticions en la mateixa connexió, i establir així una **connexió persistent**.

El client que fa la petició s'anomena **agent d'usuari**[2]. Les dades que es transmeten a partir d'una petició, es coneixen com a **recursos**. Un recurs pot ser una pàgina, una imatge, un document XML, etc.

(2) En anglès, *user agent*.

### 2.1.1. Peticions HTTP

Les peticions HTTP s'utilitzen per a sol·licitar un recurs a un servidor i consta d'un conjunt de línies:

**1) Línia de sol·licitud**. Indica el mètode utilitzat, el recurs que s'ha d'obtenir i la versió del protocol. El mètode especifica el tipus d'operació sol·licitat pel client. El recurs s'especifica mitjançant el seu URL.

```
Mètode Recurs Versió
```

**2) Camps d'encapçalament de sol·licitud**. És compost per un conjunt de línies opcionals que aporten informació addicional sobre la petició, el navegador o el sistema operatiu utilitzat, etc. Cada línia és composta pel nom de l'encapçalament, dos punts (:) i el valor de l'encapçalament:

```
Encapçalament: valor
```

**3) Línia en blanc**. Utilitzada per a separar els encapçalaments del cos de petició.

**4) Cos de la petició**. Dades addicionals utilitzades en algunes peticions per a enviar paràmetres sobre la sol·licitud.

La petició següent obtindria la pàgina `index.html` del servidor `www.uoc.edu` i enviaria informació sobre el nostre navegador i sistema operatiu:

```
GET /index.html HTTP/1.1 Host:
User-Agent: Mozilla/5.0 (Windows; en-US) Firefox/1.0.1 [Lí-
nia en blanc]
```

Els mètodes per a fer peticions HTTP més interessants per a l'estudi d'AJAX són GET i POST.

**1) GET**

Permet obtenir un recurs ubicat en l'URL especificat en la línia de sol·licitud. Els paràmetres, si n'hi ha, s'indicaran en l'URL, concatenats a l'adreça del recurs. Un URL amb paràmetres té el format següent:

```
URL_del_recurs?
param1=valor1&param2=valor2&...&paramN=valorN
```

El símbol `?` separa l'URL del recurs que se sol·licita dels paràmetres. Els paràmetres se separen entre ells pel símbol `&` i el símbol = separa un paràmetre del seu valor. La línia de sol·licitud següent obté la pàgina `index.html` enviant dos paràmetres:

**Paràmetres en GET**

L'usuari pot veure els paràmetres dins de la barra de l'adreça del navegador una vegada hagi accedit al recurs.

```
GET /index.html?language=catalan&showImages=off HTTP/1.1
```

No hi ha limitacions en el nombre de paràmetres enviats per una sol·licitud, però sí en la longitud de l'URL, inclosos els paràmetres. En cas que calgui enviar una gran quantitat d'informació al servidor, es farà ús d'una petició de tipus POST, que s'explica a continuació.

**2) POST**

Permet obtenir un recurs i, a diferència del mètode GET, permet enviar més paràmetres dins del cos de la petició. Això ofereix dos avantatges: el primer és que la longitud de paràmetres pot ser molt més extensa (fins a 2 GB, aproximadament). El segon és que els paràmetres no estan a vista de l'usuari, cosa que fa que l'URL només contingui el recurs que es vol obtenir.

Les peticions POST s'utilitzen normalment en la tramesa de formularis HTML, en què la informació de tots els camps s'envia dins del cos de la petició en forma de paràmetres.

### 2.1.2. Respostes HTTP

Una resposta HTTP conté un conjunt de línies que el servidor envia al navegador com a resposta a una petició. L'estructura d'una resposta és la següent:

**1) Línia d'estat**. Dins d'aquesta línia s'especifica la versió del protocol utilitzada, el codi de resposta i el text explicatiu del codi de resposta.

**2) Camps de l'encapçalament de resposta**. Un conjunt de línies opcionals que aporten informació addicional sobre la resposta i el servidor: hora i data de la resposta, longitud del contingut, tipus de servidor, etc.

**3) Línia en blanc**. Separa els encapçalaments del cos de la petició.

**4) Cos de la resposta**. Conté el recurs sol·licitat.

Els codis de resposta continguts en la línia d'estat són formats per tres dígits. La taula següent mostra alguns dels codis de resposta HTTP/1.0.

| Grup d'operació | Codi | Significat |
|---|---|---|
| 2××<br>Operació correcta | 200<br>201<br>202<br>204 | OK<br>S'ha creat un recurs nou<br>Petició acceptada<br>Resposta buida |
| 3××<br>Readreçament a un altre URL | 301<br>304 | El recurs sol·licitat ha canviat d'adreça<br>El contingut del recurs no ha canviat |
| 4××<br>Error per part del client | 400<br>401<br>403<br>404 | Sol·licitud incorrecta<br>Usuari no autoritzat<br>Accés prohibit<br>No s'ha trobat el recurs sol·licitat |
| 5××<br>Error per part del servidor | 500<br>501<br>503 | Error intern del servidor<br>Operació no implementada<br>Servei no disponible |

## 2.2. XML

L'XML[3] és un llenguatge extensible de marques que permet estructurar i diferenciar els continguts d'un document. Tota la informació és etiquetada, estructurada i organitzada i pot ser interpretada i modificada de manera senzilla entre els sistemes involucrats. Per tant, és possible l'intercanvi d'informació sense que hi hagi dependències amb els sistemes involucrats.

L'exemple següent mostra el contingut d'un fitxer XML:

(3) *XML* és la sigla d'*extensible markup language*.

**W3C i XML**

El W3C (World Wide Web Consortium) és l'organisme encarregat de mantenir el sistema XML.

```
<?xml version="1.0" encoding="utf-8" ?>
<PC>
  <Name>PCAdmin1</Name>
  <CPU>Core2 Duo</CPU>
  <Ram>4GB</Ram>
</PC>

<PC>
  <Name>PCRRHH2</Name>
  <CPU>AMD Athlon</CPU>
  <Ram>2GB</Ram>
</PC>
```

Un fitxer DTD[4] permet definir l'estructura i la sintaxi de les etiquetes. D'aquesta manera, el contingut d'un fitxer XML pot ser validat per a comprovar si compleix la sintaxi del DTD a què fa referència.

(4)*DTD* és la sigla de *document type definition*.

### 2.3. XHTML

L'XHTML[5] és un llenguatge de marques basat en XML l'objectiu del qual era reemplaçar el llenguatge HTML 4 i convertir-se en estàndard de qualsevol pàgina web. L'XHTML engloba i amplia l'HTML 4 de manera que permet representar les mateixes pàgines i, a més, validar-ne l'estructura i la sintaxi.

(5)*XHTML* és la sigla d'*extensible hypertext markup language*, 'llenguatge extensible de marques d'hipertext'.

En tractar-se d'XML, els documents XHTML es poden mostrar, editar i validar amb eines XML convencionals.

### 2.4. CSS

El CSS[6] és un llenguatge formal utilitzat per a donar format a documents per a la seva visualització. S'utilitza en HTML, XHTML i XML i permet separar el contingut del document del seu format.

(6)*CSS* és la sigla de *cascading style sheets*, 'fulls d'estil en cascada'.

En un fitxer CSS, es declaren i defineixen estils amb un conjunt de propietats i els seus valors respectius.

### 2.5. DOM

DOM[7] és una API que permet als programes i *scripts* accedir al contingut, a l'estructura i a l'estil de documents XML, XHTML i HTML i manipular-los. Actualment la gran majoria de navegadors ofereix una implementació del DOM accessible des de JavaScript.

(7)*DOM* és la sigla de *document object model*.

**Significat de DOM**

*Document object model* es pot traduir com a model en objectes per a la representació de documents i també com a model d'objectes del document.

Aquest model ofereix una visió estructurada de les dades com una jerarquia de nodes en forma d'arbre. Cada node del document és representat mitjançant un objecte que conté les seves propietats i els seus mètodes. Per mitjà de JavaScript és possible navegar per aquesta estructura, accedir a cadascun dels objectes d'una manera independent i fer canvis dinàmicament.

**Importància de DOM en AJAX**

El DOM és una peça essencial d'AJAX, ja que permet la modificació de pàgines ja carregades d'una manera dinàmica mitjançant JavaScript.

El W3C va definir *DOM* com un estàndard per a facilitar la compatibilitat entre els diferents navegadors. Malgrat aquest estàndard, l'ús del DOM per a aplicacions que estiguin disponibles per a qualsevol navegador és tot un repte per als desenvolupadors. Encara que la majoria de navegadors ofereixen suport per a documents mitjançant DOM, aquestes implementacions poden diferir entre els uns i els altres. Per exemple, Mozilla fa esforços per a seguir l'estàndard, mentre que la implementació de Microsoft al seu navegador Internet Explorer fins a la versió 8 difereix de l'estàndard i no n'implementa tots els nivells.

## 2.6. JavaScript

***Script***

Un *script* és un programa de processament per lots. Generalment són llenguatges de programació simples i interpretats.

JavaScript és un llenguatge de *script* interpretat per la gran majoria de navegadors. Va ser creat per a oferir dinamisme i interacció dins de les pàgines HTML. Es pot emprar per a oferir efectes visuals avançats, validar camps d'un formulari, detectar el navegador de l'usuari, respondre a esdeveniments de controls web, etc.

No s'ha de confondre JavaScript amb el llenguatge Java: Java és un llenguatge orientat a objectes que necessita ser compilat i interpretat mitjançant una màquina virtual. JavaScript no necessita ser compilat i només s'executa dins d'una pàgina HTML, ja sigui tant en la part client com en la part servidor.

La sintaxi de JavaScript és semblant a C i Java. Està orientat a objectes però no permet la creació de classes. L'execució de codi JavaScript està restringida a l'àmbit web i per motius de seguretat no és possible executar codi extern ni fer crides al sistema operatiu.

**Vegeu també**

Els exemples numerats que es presenten al llarg de tot aquest subapartat 2.6 estan també disponibles en línia.

### 2.6.1. Primers passos amb JavaScript

Per a fer ús de JavaScript dins d'una pàgina, és necessari que el codi estigui entre les etiquetes `<script></script>`.

L'exemple següent (exemple 1) és una simple pàgina HTML que mostra el text "Hola món" mitjançant JavaScript.

**Vegeu també**

En aquest punt es fa un repàs general de JavaScript. Per a ferne un repàs més a fons, convé anar al material docent de l'assignatura *Programació web* .

**Exemple 1**

```
<html>
<head>
    <title>Primers passos amb JavaScript</title>
</head>
<body>
    <script type="text/ javascript">
        document.write('<b>Hola món des de JavaScript</b>');
    </script>
</body>
</html>
```

En l'exemple anterior s'accedeix al DOM de la pàgina actual mitjançant l'objecte `document`. El mètode `write` de `document` permet escriure codi HTML o JavaScript en el document.

També és possible incloure en una pàgina HTML codi JavaScript definit en un altre fitxer. Per a això s'especificarà l'URL del fitxer en l'atribut `src` de l'etiqueta `<script>` (exemple 2).

**Exemple 2**

```
<html>
<head>
<title>Hola món utilitzant un fitxer extern</title>
</head>
<body>
       <script src="holamon.js"></script>
</body>
</html>
```

El contingut del fitxer `holamon.js` és el següent:

```
document.write('<b>Hola món des de JavaScript</b>');
```

### 2.6.2. Tipus de dades

Els tipus de dades en JavaScript són: numèrics, flotants, booleans, cadenes de caràcters, objectes, *null* i *undefined* (no definits).

#### 1) Numèrics

Els tipus numèrics poden ser enters o reals. Els tipus enters poden ser representats en tres bases diferents: decimal, octal i hexadecimal. Per a especificar un valor octal caldrà anteposar un "0", i un "0x" per als hexadecimals. L'exemple 3 crea tres variables de tipus numèric i les inicialitza amb valors en base decimal, octal i hexadecimal.

**Exemple 3**

```
<script type="text/javascript">
    var var1 = 10; //S'assigna un valor decimal
    var var2 = 043; // S'assigna un valor en base octal
    var var3 = 0xFE; //S'assigna un valor en base hexadecimal
    document.write('El valor decimal de var1 és: '+var1+'<br/>');
    document.write('El valor decimal de var2 és: '+var2+'<br/>');
    document.write('El valor decimal de var3 és: '+var3+'<br/>');
</script>
```

### 2) Flotants

Poden representar un nombre en coma flotant.

### 3) Booleans

Representen un valor amb dos estats possibles: vertader (*true*) o fals (*false*). També s'interpreta com a vertader el valor enter més gran que zero, i com a fals el valor 0 per a tipus numèrics.

### 4) Cadenes de caràcters

Representen un vector de caràcters. Les cadenes es representen entre cometes simples `(' ')` o dobles `(" ")`.

```
var mensaje = 'Això és una cadena de text';
```

Podem definir alguns caràcters especials que es poden usar dins de les cadenes:

| Descripció | Codificació |
|---|---|
| Salt de línia | \n |
| Tabulador | \t |
| Retorn | \r |
| Contrabarra | \\ |
| Cometa simple | \' |
| Cometa doble | \" |

L'exemple 4 mostra l'ús d'alguns dels caràcters especials de la taula anterior.

**Exemple 4**

```
<script type="text/javascript">
    alert('Núm. Mòdul:\t1\nNombre:\t\tIntroducció a Ajax\n');
</script>
```

### 5) Objectes

Un objecte és un tipus de dada que pot contenir diferents valors i propietats per a accedir-hi.

**6) *Null***

És un valor buit que indica que una variable conté un valor desconegut.

**7) *Undefined***

Valor de variables creades però no inicialitzades. L'exemple 5 declara una variable sense inicialitzar-la. La sortida per pantalla serà "El valor de var1 és: undefined".

**Exemple 5**

```
var var1;
document.write('El valor de var1 és: '+var1);
```

### 2.6.3. Variables

Una variable és una referència a una zona de memòria on podem emmagatzemar informació.

Per a poder usar una variable, aquesta ha de ser declarada prèviament. En JavaScript no cal indicar el tipus de valor que contindrà una variable en la seva definició. Per a declarar una variable n'indicarem el nom i hi assignarem un valor inicial:

```
num1 = 5;
```

Durant la vida d'una variable, és possible canviar-ne el tipus simplement assignant-hi un valor d'un tipus diferent. Per exemple, la variable declarada anteriorment és de tipus numèric. És possible assignar-hi un valor d'un altre tipus (exemple 6).

**Exemple 6**

```
<script type="text/javascript">
    num1 = 5;
    document.write('El valor de num1 és: '+num1+'<br/>');
    num1 = false;
    document.write('El valor de num1 ara és: ' + num1);
</script>
```

#### Àmbit d'una variable

L'àmbit o vida d'una variable pot ser:

- **Global**: s'hi pot accadir des de qualsevol part de la pàgina HTML.

- **Local**: únicament s'hi pot accedir des de dins de la funció en què es declara.

Per defecte, les variables s'interpreten com a globals. Perquè l'àmbit sigui local, especificarem la paraula
en declarar-la.

L'exemple 7 mostra com es declaren una variable local i una de global.

**Exemple 7**

```
<script type="text/javascript">
       function CrearVariables(){
             var variableLocal = 'Variable local';
             variableGlobal = 'Variable global';
       }
       CrearVariables();
       alert('El valor de variableGlobal és: ' + variableGlobal);
       alert('El valor de variableLocal és: ' + variableLocal)
</script>
```

Les dues variables creades dins de la funció anterior tenen àmbits diferents. La primera es declara amb el modificador
i és accessible només des de dins de la funció. La segona és global. Si s'executa l'exemple, es comprovarà que es mostra correctament el valor de la variable global però es genera un error en mostrar la variable local.

### Matrius

Les matrius[8] són un tipus especial d'objectes que ens permet emmagatzemar una col·lecció de variables. Els elements continguts en una matriu poden ser de qualsevol tipus.

(8) En anglès, *arrays*.

Per a crear una matriu s'utilitzarà l'objecte `Array` mitjançant l'operador **`new`**:

```
var matriu = new Array(7);
```

Aquesta instrucció crea una variable que referenciarà una matriu amb una dimensió de set elements. També és possible crear una matriu inicialitzant-la amb diferents valors:

```
var matriu = new Array('Dilluns', 'Dimarts', 'Dimecres', 'Di-
jous', 'Divendres', 'Dissabte', 'Diumenge');
```

Aquesta segona matriu conté set variables de tipus cadena (*string*). Per a poder accedir a cadascun dels elements d'una matriu utilitzarem l'operador `[int]` i indicarem l'índex de la posició a la qual volem accedir. Mitjançant aquest operador, podem tant obtenir com establir la variable en la posició indicada. En qualsevol matriu en JavaScript, el primer element té com a índex 0.

L'exemple 8 recorre la matriu anterior i en mostra el contingut.

**Exemple 8**

```
<script type="text/javascript">
var matriu = new Array('Dilluns', 'Dimarts', 'Dimecres', 'Dijous',
    'Divendres', 'Dissabte', 'Diumenge');
for(var i = 0; i < matriz.length; i++){
    document.write(matriz[i]+'<br/>');
}
</script>
```

El mètode `length` de l'objecte `Array` torna el nombre d'elements que conté.

### 2.6.4. Operadors

Podem distingir tres tipus d'operadors: operadors aritmètics, lògics i condicionals.

#### Operadors aritmètics

Els operadors aritmètics permeten fer operacions matemàtiques amb un o dos operands.

| Operador | Descripció |
|---|---|
| + | Fa la suma entre dos operands o la concatenació de dues cadenes. |
| - | Torna la resta entre dos nombres. |
| ++ | Incrementa en una unitat un nombre. |
| -- | Fa decréixer en una unitat un nombre. |
| * | Multiplica dos nombres. |
| / | Fa la divisió entre dos nombres. |
| % | Torna el mòdul entre dos nombres. |

L'exemple 9 mostra els operadors aritmètics.

**Exemple 9**

```
<html>
 <head>
    <title>Operadors aritmètics</title>
 </head>
 <body>
    <script type="text/javascript">
    function operArit()
    {
       var num1 = document.getElementById("textFieldNum1").value;
       var num2 = document.getElementById("textFieldNum2").value;
       document.write('num1: ' + num1 + '<br/>');//6
       document.write('num2: ' + num2 + '<br/>');//2
       document.write('num1 * num2: ' + (num1*num2) + '<br/>');//12
       num1++;
       document.write('num1++: ' + num1 + '<br/>');//
       document.write('num1 % num2: ' + (num1%num2) + '<br/>');
    }
    </script>

    Num1: <input id="textFieldNum1" type="text"/><br/>
    Num2: <input id="textFieldNum2" type="text"/><br/>
 <button onclick="operArit();">Operadors aritmètics</button>
 </body>
</html>
```

Aquest exemple sol·licita dos nombres i mostra el resultat d'algunes operacions aritmètiques.

## Operadors lògics

Els operadors lògics indiquen les operacions que es poden utilitzar sobre expressions booleanes. Una expressió booleana és qualsevol operació, funció o comparació que torni un valor booleà. Els tres tipus d'operadors lògics a JavaScript apareixen en la taula següent:

| Operador | Símbol | Descripció |
|---|---|---|
| AND | && | Retorna vertader en cas que els dos operands siguin vertaders, i fals en cas contrari. |
| OR | ÷÷ | Retorna vertader si algun dels operands és vertader, i fals en cas que tots els operands siguin falsos. |
| NOT | ! | Nega el valor de l'operand. |

L'exemple 10 mostra l'ús dels tres operadors lògics.

**Exemple 10**

```
<script type="text/javascript">
     var a = true;
     var b = false;

     document.writeln(' a && b -> ' + (a && b) + '<br/>');
     document.writeln(' a || b -> ' + (a || b) + '<br/>');
     document.writeln(' !b -> ' + (!b));
</script>
```

### Operadors condicionals

Els operadors condicionals comparen dues expressions i retornen un valor booleà. S'empren dins de condicions per a determinar si s'ha d'executar una acció.

Els sis operadors condicionals de què disposa JavaScript són: més gran que (>), més petit que (<), més gran o igual que (> =), més petit o igual que (< =), igual que (= =), diferent que (! =).

L'exemple 11 mostra el resultat de diferents operacions booleanes a partir de dos nombres introduïts per l'usuari.

**Exemple 11**

```
<html>
 <head>
    <title>Operadors condicionals</title>
 </head>
 <body>
    <script type="text/javascript">
      function operCond()
      {
        var num1 = document.getElementById("textFieldNum1").value;
        var num2 = document.getElementById("textFieldNum2").value;
        document.write('num1: ' + num1 + '<br/>');
        document.write('num2: ' + num2 + '<br/>');
        document.write('num1 > num2: ' + (num1 > num2) + '<br/>');
        document.write('num1 < num2: ' + (num1 < num2) + '<br/>');
        document.write('num1 == num2: ' + (num1 == num2) + '<br/>');
        document.write('num1 != num2: ' + (num1 != num2) + '<br/>');
      }
    </script>

    Num1: <input id="textFieldNum1" type="text"/><br/>
    Num2: <input id="textFieldNum2" type="text"/><br/>
    <button onclick="operCond();">Op. condicionals</button>
 </body>
</html>
```

## 2.6.5. Sentències de control

Les sentències de control permeten avaluar mitjançant una condició si s'ha d'executar un bloc de codi i quantes vegades. Les diferents estructures de control es poden agrupar en condicionals i bucles.

### Condicionals

Les sentències de control condicionals són:

- La sentència `if..else`

Executa un bloc d'instruccions si es compleix una condició; aquesta condició serà qualsevol expressió que retorni un valor booleà. Es pot usar opcionalment la sentència `else` per a executar un altre bloc diferent en cas que la condició no es compleixi:

```
if (condició) {
    //instruccions
}
[else{
    //instruccions
}]
```

Dins d'una sentència `else`, es pot tornar a avaluar una altra condició, com es veu en l'exemple 12.

**Exemple 12**

```
<script type="text/javascript">
       var num1 = 10;
       var num2 = 20;
       if (num1 < num2) {
              document.writeln('num1 és més gran que num2');
       }
       else if (num1 == num2) {
              document.writeln('num1 és igual que num2');
       }
       else {
              document.writeln('num1 es més gran que num2');
}
</script>
```

- La sentència `switch`

El bloc `switch` avalua una expressió que retorna un valor i aquest es compara amb un conjunt de valors. Cadascun d'aquests, té associat un bloc de codi que serà executat en cas que coincideixin.

El format és el següent:

```
switch(expressió){
   case valor1:
      //Sentències
      break;
   case valor2:
      //Sentències
      break;
   default:
      //Sentències per defecte
}
```

L'expressió continguda entre parèntesis es compara amb cadascun dels possibles valors precedits per la sentència `case`. En cas que siguin iguals, s'executen les sentències que segueixen a continuació. La instrucció `break` és opcional i permet sortir de la sentència `switch`. És útil per a evitar que es continuï comparant una expressió per tots els altres valors de la sentència.

```
<script type="text/javascript">
```

```
    var dateObject=new Date();
    dia=dateObject.getDay();
    switch(dia){
    case 1:
              document.write('Avui és dilluns');
              break;
    case 2:
              document.write('Avui és dimarts');
              break;
    case 3:
              document.write('Avui és dimecres');
              break;
    case 4:
              document.write('Avui és dijous');
              break;
    case 5:
              document.write('Avui és divendres');

              break;
    default:
              document.write('Avui és cap de setmana');
    }
</script>
```

El cas `default` permet executar un bloc alternatiu en cas que l'avaluació no coincideixi amb cap valor.

### Bucles

Els bucles permeten executar repetidament un bloc de codi fins que una certa condició es compleixi.

- El bucle `for`

El bucle `for` s'utilitza per a executar un conjunt d'instruccions fins que es compleixi una condició definida en la mateixa sentència. L'estructura és la següent:

```
for(inicialització; condició; increment)
{
   //Instruccions
}
```

En *inicialització* es declara i s'inicialitza la variable que servirà de comptador per al bucle. La *condició* és un resultat booleà que indica en cada iteració si s'ha de tornar a executar el bloc de codi. L'increment modifica el valor de la variable en cada iteració. Tant la condició com l'*increment* s'avaluen en cada iteració.

Un exemple senzill de bucle `for` seria mostrar els deu primers nombres naturals (exemple 13).

**Exemple 13**

```
<script type="text/javascript">
       var i;
       for(i = 0; i <= 10; i++)
       {
           document.write('Iteració número ' + i + '<br/>'); }
</script>
```

- El bucle `while`

El bucle `while` permet executar un bloc d'instruccions fins que es compleixi una condició determinada:

```
while (condició)
   {
       //Instruccions
   }
```

L'exemple 14 mostra els nombres de l'1 al 10.

**Exemple 14**

```
<script type="text/javascript">
       var num = 1;
       while (num <= 10)
        {
              document.writeln('Iteració número: '+num+'<br/>');
              num++;
        }
</script>
```

- El bucle `Do..While`

A diferència de `while`, `Do..While` executa el bloc d'instruccions abans d'avaluar la condició i, per tant, el bloc s'executarà almenys una vegada:

```
do{
    //Instruccions
}while (condició)
```

L'exemple 15 mostra els deu primers nombres utilitzant el bucle `Do..While`.

**Exemple 15**

```
<script type="text/javascript">
       var num = 1;
       do
       {
           document.writeln('Iteració número: ' + num + '<br/>');
           num++;
       }while (num <= 10)
</script>
```

### 2.6.6. Funcions

Les funcions són blocs de codi representats per un nom als quals es pot accedir des de diferents parts d'una pàgina HTML.

L'estructura d'una funció és la següent:

```
function nomDeLaFuncio([param1,param2,...]){
      //Instruccions
      [return valor;]
}
```

Els paràmetres permeten passar diferents valors que poden ser utilitzats en el cos de la funció. Opcionalment, una funció pot retornar un valor com a resultat.

Les funcions se solen declarar dins de l'etiqueta `<head>`. Si són d'ús general, es recomana emmagatzemar-les en un fitxer extern.

L'exemple 16 mostra una funció amb dos paràmetres que retorna la suma d'aquests paràmetres.

**Exemple 16**

```
<html>
<head>
      <meta http-equiv="Content-Type" content="text/html;
      charset=ISO-8859-1">
      <title>Exemple de funcions</title>
      <script type="text/javascript">
             function Suma(num1, num2){
                 return num1 + num2;
             }
      </script>

</head>
<body>
    <script type="text/javascript">
        document.write("La suma de 2 + 3 és" + Suma(2,3));
    </script>
</body>
</html>
```