Introducción a AJAX

Jordi Sánchez Cano

PID_00172705

UOC
Universitat Oberta de Catalunya

Índice

Introducción

AJAX (*asynchronous Java-Script and XML*) es un término que engloba un conjunto de tecnologías y técnicas que permiten desarrollar webs más ricas e interactivas. Es un concepto muy asociado a otros, como RIA (*rich Internet applications*), Web 2.0 y aplicaciones web.

En este módulo se explica qué es AJAX y cuáles son sus beneficios y las tecnologías que emplea. Entre las tecnologías utilizadas por AJAX, se halla Java-Script, uno de sus pilares fundamentales. Por este motivo se da un breve repaso a este lenguaje de programación.

Objetivos

Los objetivos de este módulo didáctico son los siguientes:

1. Conocer qué es AJAX y cuáles son las diferentes tecnologías que utiliza.

2. Entender los beneficios del uso de AJAX en aplicaciones web sobre el modelo tradicional.

3. Repasar algunos conceptos básicos de Java-Script.

1. Introducción a la programación web

En este apartado veremos algunos conceptos básicos sobre la programación web.

1.1. La Web y su evolución

En sus inicios, la Web se basaba en simples páginas HTML por las que el usuario navegaba de una a otra mediante hipervínculos. Cada acción que llevaba a cabo un usuario desencadenaba la carga de una nueva página, y de ahí el concepto de *navegar*. En esta primera etapa de la Web, a la que se denominó Web 1.0, las páginas eran estáticas y las actualizaba la entidad o la persona que las publicaba.

La Web 2.0 se refiere a una segunda generación que engloba tanto el avance tecnológico como el cambio de concepto que tenemos sobre ella. Este término redefine la Web como un conjunto de servicios y una plataforma en la que ahora los usuarios son partícipes de los contenidos.

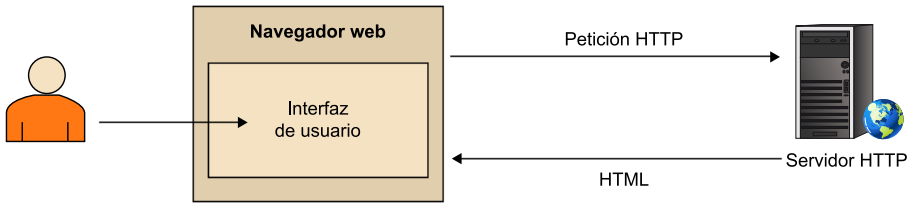
Este cambio sobre el concepto de la Web no hubiera sido posible sin una evolución tecnológica previa, en la que AJAX desempeña un papel muy importante, de manera que es uno de los términos que más acompañan a la Web 2.0.

1.2. Qué es AJAX

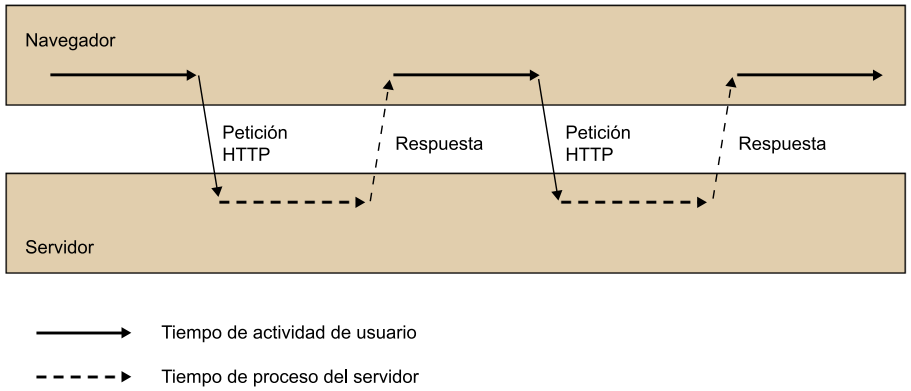
Conceptualmente, podemos definir AJAX como el conjunto de técnicas y tecnologías que permiten desarrollar aplicaciones web interactivas o RIA, que ofrecen mejor interacción, funcionalidad, accesibilidad y rendimiento.

AJAX permite que los usuarios sigan interactuando con una misma página mientras se mantiene la comunicación con el servidor asíncronamente, al mismo tiempo que se actualizan los contenidos. Esto comporta que los datos se obtengan y se actualicen en segundo plano, sin interferir en el uso normal de la aplicación.

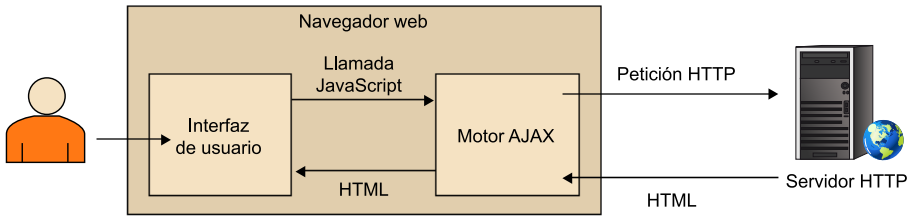
La mayoría de acciones generadas por el usuario en el modelo tradicional de aplicación web desencadenan una petición HTTP que solicita una nueva página al servidor. El servidor procesa la página y la envía de vuelta al navegador, tal y como se muestra en el esquema siguiente.



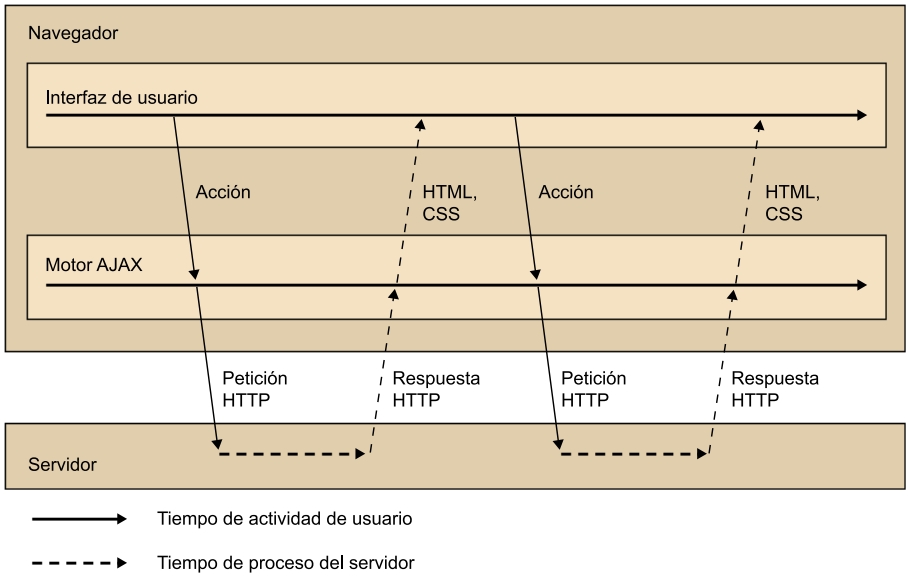
Desde que se envía la petición HTTP hasta que se recibe y se renderiza la nueva página, el usuario queda a la espera, sin poder hacer uso de la aplicación. El siguiente diagrama de secuencia muestra la disponibilidad del navegador entre cada cambio de página.



La carga y la espera se repiten cada vez que es necesario actualizar una parte de la página. Las aplicaciones que hacen uso de AJAX rompen con este modelo permitiendo la comunicación con el servidor y la recarga de la página de manera asíncrona, de forma que el usuario puede continuar trabajando con la aplicación sin interrupciones. La figura siguiente muestra interacción asíncrona entre el usuario y el servidor.



El motor AJAX está compuesto por un conjunto de funciones Java-Script que gestiona la comunicación con el servidor y efectúa los cambios necesarios en la interfaz de usuario. Las acciones que lleva a cabo el usuario son delegadas al motor AJAX y, mientras se actualizan los nuevos contenidos, la interfaz de usuario sigue disponible, como se muestra en el diagrama siguiente.



En el pasado ya se hacía uso de estas técnicas y tecnologías, pero no fue hasta el 18 de febrero del 2005 cuando Jesse James Garrett definió el término AJAX por primera vez en un artículo llamado "AJAX. A new approach to web applications".

Lectura recomendada

Se puede consultar el artículo "AJAX. A new approach to web applications" en la página de Adaptive Path.

En este artículo se define AJAX como un modelo de aplicación web que hace uso del siguiente conjunto de tecnologías:

- **XHTML** y **CSS**, que se utilizan para la presentación;
- **document object model**, que se utiliza para la modificación dinámica de la presentación de una página;
- **XML** y **XSLT**, que se utilizan para el intercambio y la manipulación de datos, y
- **Java-Script**, que se utiliza como lenguaje para la ejecución y el acceso a las tecnologías anteriores.

Todas estas tecnologías son estándares abiertos, por lo que es posible usar AJAX en múltiples navegadores y plataformas.

Sitios web que usan AJAX

Cada vez existen más sitios web en la Red que hacen uso de AJAX. Uno de los primeros y principales portales en utilizarlo fue Google, tanto en su buscador como en Gmail.

Google Search

El buscador de Google hace uso de AJAX en su cuadro de texto para introducir los términos de búsqueda. Mientras escribimos en él, se despliega una lista con sugerencias parecidas a la parte que hemos escrito. A medida que vamos escribiendo, la lista se actualiza con nuevas sugerencias. Las sugerencias obtenidas se obtienen de forma asíncrona, sin interferir en la interacción del usuario.

Gmail

El cliente web de correo de Google utiliza AJAX en diferentes partes de la aplicación. Durante el uso de toda la aplicación, permaneceremos dentro de la misma página. Además, el motor AJAX de Gmail comprueba el correo periódicamente, de manera que si se recibe un nuevo correo, se actualiza la vista con los cambios necesarios.

1.3. Ventajas sobre el uso de AJAX

Algunas de las ventajas sobre el uso de AJAX en aplicaciones web son las siguientes:

- Reduce el tráfico de las aplicaciones web, al enviar sólo los datos necesarios en cada momento.
- Reduce el tiempo de espera, puesto que hay que enviar y recibir menos información.
- Al no recargar la página, el usuario tiene la sensación de trabajar siempre con la misma aplicación, y no con páginas diferentes.
- Permite mejorar la usabilidad de sitios web, sobre todo en sitios que contienen muchos formularios y éstos son muy complejos (con campos del formulario que dependen del valor de otros campos: provincia, pueblo, código postal...).
- Permite mejorar la percepción que el usuario recibe del sitio web: el usuario tiene la impresión de estar trabajando en una aplicación de escritorio.
- Permite reducir la dificultad al crear páginas web complejas si se reutilizan controles AJAX existentes.

Por otro lado, desarrollar páginas que hagan uso de AJAX requiere más conocimientos sobre las tecnologías que éste utiliza. Estas páginas deberán controlar los posibles errores que puedan ocurrir al realizar las operaciones asíncronas. Otro inconveniente de AJAX es que algunos navegadores pueden tener Java-Script deshabilitado, por lo que las partes asíncronas quedarán inservibles.

2. Tecnologías utilizadas por AJAX

2.1. El protocolo HTTP

HTTP es el protocolo utilizado en la comunicación web para la transferencia de archivos, generalmente HTML. Mediante HTTP, se solicitan y se obtienen las diferentes páginas y los recursos necesarios entre el navegador y el servidor.

Para acceder a estos recursos, se utiliza su dirección, representada mediante URL[1].

(1)URL es la sigla de *uniform resource locator* ('localizador uniforme de recurso').

HTTP está orientado a transacciones, cada una de las cuales está formada por una petición del cliente y una respuesta del servidor. En cada transacción, se establece una nueva conexión al tiempo que se envía la petición. La conexión se cierra una vez que el servidor envía la respuesta al cliente. También es posible intercambiar diferentes peticiones en la misma conexión, con lo que se establece una **conexión persistente**.

Al cliente que realiza la petición se le llama **agente de usuario**[2]. A los datos que se transmiten a partir de una petición, se les conoce como **recursos**. Un recurso puede ser una página, una imagen, un documento XML, etc.

(2)En inglés, *user agent*.

2.1.1. Peticiones HTTP

Las peticiones HTTP se utilizan para solicitar un recurso a un servidor y constan de un conjunto de líneas:

1) Línea de solicitud. Indica el método utilizado, el recurso que se ha de obtener y la versión del protocolo. El método especifica el tipo de operación solicitada por el cliente. El recurso se especifica mediante su URL.

```
Método Recurso Versión
```

2) Campo de encabezado de solicitud. Está compuesto por un conjunto de líneas opcionales que aportan información adicional sobre la petición, el navegador, el sistema operativo utilizado, etc. Cada línea está formada por el nombre del encabezado, dos puntos (:) y el valor del encabezado.

```
Encabezado: valor
```

3) Línea en blanco. Se utiliza para separar los encabezados del cuerpo de la petición.

4) Cuerpo de la petición. Son datos adicionales que se utilizan en algunas peticiones para enviar parámetros sobre la solicitud.

La siguiente petición obtendría la página `index.html` del servidor `www.uoc.edu` enviando información sobre nuestro navegador y el sistema operativo:

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

Para el estudio de AJAX, los métodos más interesantes para llevar a cabo peticiones HTTP son GET y POST.

1) GET

Permite obtener un recurso ubicado en el URL especificado en la línea de solicitud. Los parámetros, en caso de que existan, se indicarán en el URL, de manera que se concatenarán a la dirección del recurso. Un URL con parámetros tiene el siguiente formato:

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

El símbolo `?` separa el URL del recurso que se ha de solicitar de los parámetros. Los parámetros se separan entre sí por el símbolo `&`, y el símbolo = separa un parámetro de su valor. La siguiente línea de solicitud obtiene la página `index.html` enviando dos parámetros:

Parámetros en GET

El usuario puede ver los parámetros dentro de la barra de dirección del navegador una vez que haya accedido al recurso.

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

No hay limitaciones en el número de parámetros enviados por una solicitud, pero sí en la longitud del URL, parámetros incluidos. En el caso de que sea necesario enviar una gran cantidad de información al servidor, se hará uso de una petición de tipo POST, que se explica a continuación.

2) POST

Permite obtener un recurso y, a diferencia del método GET, con POST se pueden enviar más parámetros dentro del cuerpo de la petición. Esto ofrece dos ventajas: la primera es que la longitud de parámetros puede ser mucho mayor

(hasta 2 GB aproximadamente); la segunda es que los parámetros no quedan a la vista del usuario, de manera que el URL sólo contiene el recurso que se ha de obtener.

Las peticiones POST se utilizan normalmente en el envío de formularios HTML, de manera que se envía la información de todos los campos dentro del cuerpo de la petición en forma de parámetros.

2.1.2. Respuestas HTTP

Una respuesta HTTP contiene un conjunto de líneas que el servidor envía al navegador como respuesta a una petición. La estructura de una respuesta es la siguiente:

1) Línea de estado. En esta línea se especifica la versión del protocolo utilizada, el código de respuesta y el texto explicativo del código de respuesta.

2) Campos del encabezado de respuesta. Son un conjunto de líneas opcionales que aportan información adicional sobre la respuesta y el servidor: hora y fecha de la respuesta, longitud del contenido, tipo de servidor, etc.

3) Línea en blanco. Se utiliza para separar los encabezados del cuerpo de la petición.

4) Cuerpo de la respuesta. Contiene el recurso solicitado.

Los códigos de respuesta contenidos en la línea de estado están formados por tres dígitos. La tabla siguiente muestra algunos de los códigos de respuesta HTTP/1.0.

Grupo de operación	Código	Significado
2×× Operación correcta	200 201 202 204	OK Se ha creado un nuevo recurso Petición aceptada Respuesta vacía
3×× Redirección a otro URL	301 304	El recurso solicitado ha cambiado de dirección El contenido del recurso no ha cambiado
4×× Error por parte del cliente	400 401 403 404	Solicitud incorrecta Usuario no autorizado Acceso prohibido No se ha encontrado el recurso solicitado
5×× Error por parte del servidor	500 501 503	Error interno del servidor Operación no implementada Servicio no disponible

2.2. XML

XML[3] es un lenguaje extensible de marcas que permite estructurar y diferenciar los contenidos de un documento. Toda la información queda etiquetada, estructurada y organizada, de manera que puede ser interpretada y modificada de modo sencillo entre los sistemas involucrados. Por consiguiente, es posible el intercambio de información sin que haya dependencias con los sistemas involucrados.

(3)XML es la sigla de *extensible markup language* ('lenguaje de marcas extensible').

W3C y XML

El W3C (World Wide Web Consortium) es el organismo encargado de mantener el sistema XML.

El ejemplo siguiente muestra el contenido de un fichero 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 fichero DTD[4] permite definir la estructura y la sintaxis de etiquetas. De esta manera, el contenido de un fichero XML puede ser validado para comprobar si cumple la sintaxis del DTD al que hace referencia.

(4)DTD es la sigla de *document type definition* ('definición de tipo de documento').

2.3. XHTML

XHTML[5] es un lenguaje de marcas basado en XML cuyo objetivo era reemplazar el lenguaje HTML 4 y convertirse en estándar de toda página web. XHTML engloba y extiende HTML 4, por lo que permite representar las mismas páginas, pero puede además validar su estructura y su sintaxis.

(5)XHTML es la sigla de *extensible hypertext markup language* ('lenguaje extensible de marcado de hipertexto').

Como se trata de XML, los documentos XHTML se pueden mostrar, editar y validar con herramientas XML convencionales.

2.4. CSS

CSS[6] es un lenguaje formal utilizado para dar formato a documentos para que sean visualizados. Se emplea en HTML, XHTML y XML, y permite separar el contenido del documento de su formato.

(6)CSS es la sigla de *cascading style sheets* ('hojas de estilo en cascada').

En un fichero CSS, se declaran estilos y se definen con un conjunto de propiedades y sus respectivos valores.

2.5. DOM

DOM[7] es una API que permite a los programas y a los *scripts* acceder y manipular el contenido, la estructura y el estilo de documentos XML, XHTML y HTML. Actualmente, la gran mayoría de navegadores ofrece una implementación del DOM a la que se puede acceder desde Java-Script.

[(7)]DOM es la sigla de *document object model*.

Significado de DOM

La expresión *document object model* puede traducirse al español como 'modelo en objetos para la representación de documentos' y también como 'modelo de objetos del documento'.

Este modelo ofrece una visión estructurada de los datos como una jerarquía de nodos en forma de árbol. Cada nodo del documento se representa mediante un objeto que contiene sus propiedades y métodos. Por medio de Java-Script es posible navegar por esta estructura, acceder a cada uno de los objetos de manera independiente y realizar cambios dinámicamente.

Importancia del DOM en AJAX

El DOM es una pieza esencial de AJAX, ya que permite la modificación de forma dinámica mediante Java-Script de páginas ya cargadas.

El W3C definió el DOM como un estándar para facilitar la compatibilidad entre los diferentes navegadores. Pese a este estándar, el uso del DOM para aplicaciones que estén disponibles para cualquier navegador es todo un reto para los desarrolladores. Aunque la mayoría de navegadores ofrecen soporte para documentos mediante el DOM, estas implementaciones pueden diferir entre unos y otros. Por ejemplo, Mozilla hace esfuerzos para seguir el estándar, mientras que la implementación de Microsoft en su navegador Internet Explorer hasta la versión 8 difiere del estándar y no implementa todos sus niveles.

2.6. Java-Script

Java-Script es un lenguaje de *script* interpretado por la gran mayoría de navegadores. Fue creado para ofrecer dinamismo e interacción dentro de las páginas HTML. Se puede emplear para ofrecer efectos visuales avanzados, validar campos de un formulario, detectar el navegador del usuario, responder a eventos de controles web, etc.

Script

Un *script* es un programa de procesamiento por lotes. Generalmente, son lenguajes de programación simples e interpretados.

No se debe confundir Java-Script con el lenguaje Java: Java es un lenguaje orientado a objetos que necesita ser compilado e interpretado mediante una máquina virtual. Java-Script no necesita ser compilado y sólo se ejecuta dentro de una página HTML, sea en la parte cliente o en la parte servidor.

La sintaxis de Java-Script es parecida a las de C y Java. Está orientado a objetos, pero no permite la creación de clases. La ejecución de código Java-Script está restringida al ámbito web y, por motivos de seguridad, no es posible ejecutar código externo ni hacer llamadas al sistema operativo.

Ved también

Los ejemplos numerados que se presentan a lo largo de todo este subapartado 2.6 están también disponibles en línea.

2.6.1. Primeros pasos con Java-Script

Para hacer uso de Java-Script dentro de una página, es necesario que el código esté entre las etiquetas `<script></script>`.

El ejemplo siguiente (ejemplo 1) es una simple página HTML que muestra el texto "Hola mundo" mediante Java-Script.

Ved también

En este punto se hace un repaso general de Java-Script. Para llevar a cabo un estudio más a fondo, conviene ir al material docente de la asignatura *Programación web*.

Ejemplo 1

```
<html>
<head>
   <title>Primeros pasos con Java-Script </title>
</head>
<body>
   <script type="text/javascript">
      document.write('<b>Hola mundo desde Java-Script</b>');
   </script>
</body>
</html>
```

En el ejemplo anterior se accede al DOM de la página actual mediante el objeto `document`. El método `write` de `document` permite escribir código HTML o Java-Script en el documento.

También es posible incluir en una página HTML código Java-Script definido en otro fichero. Para ello, se especificará el URL del fichero en el atributo `src` de la etiqueta `<script>` (ejemplo 2).

Ejemplo 2

```
<html>
<head>
<title>Hola mundo usando un fichero externo</title>
</head>
<body>
   <script src="holamundo.js"></script>
</body>
</html>
```

El contenido del fichero `holamundo.js` es el siguiente:

```
document.write('Hola mundo desde Java-Script');
```

2.6.2. Tipos de datos

Los tipos de datos en Java-Script son: numéricos, flotantes, booleanos, cadenas de caracteres, objetos, *null* y *undefined* (no definidos).

1) Numéricos

Los tipos numéricos pueden ser enteros o reales. Los tipos enteros se pueden representar en tres bases diferentes: decimal, octal y hexadecimal. Para especificar un valor octal, se deberá anteponer un '0', y un '0x' para los valores hexadecimales. El ejemplo 3 crea tres variables de tipo numérico que se inicializan con valores en base decimal, octal y hexadecimal.

Ejemplo 3

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

2) Flotantes

Pueden representar un número en coma flotante.

3) Booleanos

Representan un valor con dos estados posibles: verdadero (*true*) o falso (*false*). También se interpreta verdadero el valor entero mayor que cero y falso el valor 0 para tipos numéricos.

4) Cadenas de caracteres

Representan un vector de caracteres. Las cadenas se representan entre comillas simples (`' '`) o dobles (`" "`).

```
var mensaje = 'Esto es una cadena de texto';
```

Podemos definir algunos caracteres especiales que pueden usarse dentro de las cadenas:

Descripción	Codificación
Salto de línea	\n
Tabulador	\t
Retorno de carro	\r
Contrabarra	\\
Comilla simple	\'
Comilla doble	\"

El ejemplo 4 muestra el uso de algunos de los caracteres especiales de la tabla anterior.

Ejemplo 4

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

5) Objetos

Un objeto es un tipo de dato que puede contener diferentes valores y propiedades para acceder a éstos.

6) *Null*

Es un valor vacío que indica que una variable contiene un valor desconocido.

7) *Undefined*

Es un valor de variables creadas pero no inicializadas. El ejemplo 5 declara una variable sin inicializarla. La salida por pantalla será: "El valor de var1 es: undefined".

Ejemplo 5

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

2.6.3. Variables

Una variable es una referencia a una zona de memoria en la que podemos almacenar información.

Para poder usar una variable, ésta debe ser declarada previamente. En Java-Script no es necesario indicar el tipo de valor que contendrá una variable en su definición. Para declarar una variable, indicaremos su nombre y le asignaremos un valor inicial:

```
num1 = 5;
```

Durante la vida de una variable, es posible cambiar el tipo simplemente asignándole un valor de un tipo diferente. Por ejemplo, la variable declarada anteriormente es de tipo numérico. Sin embargo, es posible asignarle un valor de otro tipo (ejemplo 6).

Ejemplo 6

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

Ámbito de una variable

El ámbito o vida de una variable puede ser:

- **Global**. Se puede acceder a ellas desde cualquier parte de la página HTML.
- **Local**. Únicamente se accederá a ellas desde dentro de la función en la que se declara.

Por defecto, las variables se interpretan como globales. Para que el ámbito de una variable sea local, especificaremos la palabra `var` cuando la declaremos.

El ejemplo 7 muestra cómo se declaran una variable local y otra global.

Ejemplo 7

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

Las dos variables creadas dentro de la función anterior tienen ámbitos diferentes. La primera se declara con el modificador `var`, de manera que se podrá acceder a ella sólo dentro de la función. La segunda es global. Si se ejecuta el ejemplo, se comprobará que se muestra correctamente el valor de la variable global, pero que, al mostrar la variable local, se genera un error.

Matrices

Las matrices[8] son un tipo especial de objetos que nos permiten almacenar una colección de variables. Los elementos contenidos en una matriz pueden ser de cualquier tipo.

(8)En inglés, *arrays*.

Para crear una matriz se utilizará el objeto `Array` mediante el operador `new`:

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

Esta instrucción crea una variable referenciando a la matriz con una dimensión de siete elementos. También es posible crear una matriz inicializándola con diferentes valores:

```
var matriz = new Array('Lunes', 'Martes', 'Miércoles',
'Jueves', 'Viernes', 'Sábado', 'Domingo');
```

Esta segunda matriz contiene siete variables de tipo cadena (*string*). Para poder acceder a cada uno de los elementos de una matriz, utilizaremos el operador `[int]` indicando el índice de la posición a la que queramos acceder. Mediante este operador, podemos tanto obtener la variable como establecerla en la posición indicada. En toda matriz en Java-Script, el primer elemento tiene como índice 0.

El ejemplo 8 recorre la matriz anterior y muestra su contenido.

Ejemplo 8

```
<script type="text/javascript">
var matriz = new
Array('Lunes','Martes','Miércoles','Jueves','Viernes','Sábado',
    'Domingo');
for(var i = 0; i < matriz.length; i++){
   document.write(matriz[i]+'<br/>');
}
</script>
```

El método `length` del objeto `Array` devuelve el número de elementos que contiene.

2.6.4. Operadores

Podemos distinguir tres tipos de operadores: los operadores aritméticos, los lógicos y los condicionales.

Operadores aritméticos

Los operadores aritméticos permiten realizar operaciones matemáticas con uno o dos operandos.

Operador	Descripción
+	Realiza la suma de dos operandos o la concatenación de dos cadenas.
-	Devuelve la resta entre dos números.
++	Incrementa en una unidad un número.
--	Disminuye en una unidad un número.
*	Multiplica dos números.
/	Realiza la división entre dos números.

Operador	Descripción
%	Devuelve el módulo entre dos números.

El ejemplo 9 muestra los operadores aritméticos.

Ejemplo 9

```
<html>
  <head>
     <title>Operadores aritméticos</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();">Operadores aritméticos</button>
  </body>
</html>
```

Este ejemplo solicita dos números y muestra el resultado de algunas operaciones aritméticas.

Operadores lógicos

Los operadores lógicos indican las operaciones que se pueden utilizar sobre expresiones booleanas. Una expresión booleana es cualquier operación, función o comparación que devuelva un valor booleano. Los tres tipos de operadores lógicos en Javascript aparecen en la tabla siguiente:

Operador	Símbolo	Descripción
AND	&&	Devuelve “verdadero” en el caso de que los dos operandos sean verdaderos y “falso” en el caso contrario.
OR	||	Devuelve “verdadero” si alguno de los operandos es verdadero y “falso” en el caso de que todos los operandos sean falsos.
NOT	!	Niega el valor del operando.

El ejemplo 10 muestra el uso de los tres operadores lógicos:

Ejemplo 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>
```

Operadores condicionales

Los operadores condicionales comparan dos expresiones y devuelven un valor booleano. Se emplean dentro de condiciones para determinar si se debe ejecutar una acción.

Los seis operadores condicionales de los que dispone Javascript son los siguientes:

- mayor que (>),
- menor que (<),
- mayor o igual que (> =),
- menor o igual que (< =),
- igual que (= =) y
- diferente que (! =).

El ejemplo 11 muestra el resultado de diferentes operaciones booleanas a partir de dos números introducidos por el usuario.

Ejemplo 11

```
<html>
   <head>
      <title>Operadores condicionales</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. condicionales</button>
   </body>
</html>
```

2.6.5. Sentencias de control

Las sentencias de control permiten evaluar mediante una condición si se debe ejecutar un bloque de código y cuántas veces. Las estructuras de control pueden ser de dos tipos: condicionales y de bucles.

Condicionales

Las sentencias de control condicionales son:

- La sentencia `if..else`

Ejecuta un bloque de instrucciones si se cumple una condición; dicha condición será cualquier expresión que devuelva un valor booleano. Opcionalmente, puede usarse la sentencia `else` para ejecutar otro bloque distinto en el caso de que la condición no se cumpla:

```
if (condición) {
   //instrucciones
}
[else{
   //instrucciones
}]
```

Dentro de la sentencia `else`, se puede volver a evaluar otra condición, como se ve en el ejemplo 12.

Ejemplo 12

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

- La sentencia `switch`

El bloque `switch` evalúa una expresión que devuelve un valor, y éste se compara con un conjunto de valores. Cada uno de estos valores tiene asociado un bloque de código, que será ejecutado en el caso de que coincidan.

Su formato es el siguiente:

```
switch(expresión){
```

```
    case valor1:
       //Sentencias
       break;
    case valor2:
       //Sentencias
       break;
    default:
       //Sentencias por defecto
}
```

La expresión contenida entre paréntesis se compara con cada uno de los posibles valores precedidos por la sentencia `case`. En el caso de que sean iguales, se ejecutan las sentencias que siguen a continuación. La instrucción `break` es opcional y permite salir de la sentencia `switch`. Es útil para evitar que se siga comparando una expresión por todos los demás valores de la sentencia.

```
<script type="text/javascript">
   var dateObject=new Date();
   dia=dateObject.getDay();
   switch(dia){
   case 1:
      document.write('Hoy es Lunes');
      break;
   case 2:
      document.write('Hoy es Martes');
      break;
   case 3:
      document.write('Hoy es Miércoles');
      break;
   case 4:
      document.write('Hoy es Jueves');
      break;
   case 5:
      document.write('Hoy es Viernes');
      break;

   default:
      document.write('Hoy es fin de semana');
   }
</script>
```

El caso `default` permite ejecutar un bloque alternativo si la evaluación no coincide con ningún valor.

Bucles

Los bucles permiten ejecutar repetidamente un bloque de código hasta que una cierta condición se cumpla.

Son los siguientes:

- El bucle `for`

El bucle `for` se utiliza para ejecutar un conjunto de instrucciones hasta que se cumpla una condición definida en la misma sentencia.

Su estructura es la siguiente:

```
for(inicialización; condición; incremento)
{
   //Instrucciones
}
```

En la inicialización se declara y se inicializa la variable que servirá de contador para el bucle. La condición es un resultado booleano que indica en cada iteración si se debe volver a ejecutar el bloque de código. El incremento modifica el valor de la variable en cada iteración. Tanto la condición como el incremento se evalúan en cada iteración.

Un ejemplo sencillo de bucle `for` sería mostrar los diez primeros números naturales (ejemplo 13).

Ejemplo 13

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

- El bucle `while`

El bucle `while` permite ejecutar un bloque de instrucciones hasta que se cumpla una condición determinada.

```
while (condición)
   {
      //Instrucciones
   }
```

El ejemplo 14 muestra los números del 1 al 10.

Ejemplo 14

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

- El bucle `Do..While`

A diferencia del bucle `while`, `Do..While` ejecuta el bloque de instrucciones antes de evaluar la condición y, por lo tanto, el bloque se ejecutará al menos una vez.

```
do{
   //Instrucciones
}while (condición)
```

El ejemplo 15 muestra los diez primeros números utilizando el bucle `Do..While`.

Ejemplo 15

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

2.6.6. Funciones

Las funciones son bloques de código representados por un nombre a los que se puede acceder desde distintas partes de una página HTML.

La estructura de una función es la siguiente:

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

Los parámetros permiten pasar diferentes valores, que pueden ser utilizados en el cuerpo de la función. Opcionalmente, una función puede devolver un valor como resultado.

Las funciones suelen declararse dentro de la etiqueta <head>. Si son de uso general, se recomienda que se almacenen en un fichero externo.

El ejemplo 16 muestra una función con dos parámetros que devuelve la suma de éstos.

Ejemplo 16

```
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html;
            charset=ISO-8859-1">
      <title>Ejemplo de funciones</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 es " + Suma(2,3));
      </script>
   </body>
</html>
```