



# Sintaxis del lenguaje JavaScript

# Índice

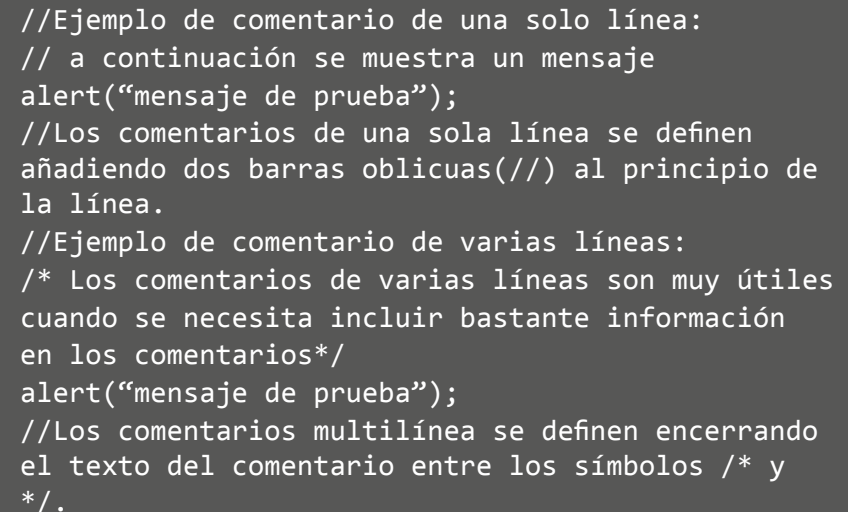


1   Sintaxis de lenguaje	3
1.1   Variables	4
1.2   Tipos de datos	6
Numéricos	6
Cadenas de texto	6
Arrays	7
Booleanos	8
1.3   Operadores	9
Asignación	9
Incremento y decremento	9
Lógicos	10
Negación	10
AND	11
OR	11
Matemáticos	11
Relacionales	11
1.4   Objetos del navegador	12
El Objeto window	13
El Objeto location	14
El Objeto screen	14
El Objeto document - La página en sí	15
El Objeto history	17
El Objeto navigator	17

# 1. Sintaxis de lenguaje

La sintaxis de un lenguaje de programación es el conjunto de reglas y parámetros que deben seguirse al redactar el código de los programas para que pasen a ser considerados como correctos y aceptables para ese lenguaje.

- **No importan las nuevas líneas y los espacios en blanco** al igual que sucede en HTML ya que el propio intérprete del lenguaje los ignora.
- **Es case sensitive:** Distingue entre las mayúsculas y minúsculas
- En contra de otros lenguajes de programación **no se definen el tipo de las variables (var)**. En JavaScript nunca sabemos el tipo de datos que va a contener una variable por lo que una misma variable puede almacenar diferentes tipos de datos.
- **Cada sentencia en JavaScript acaba con el carácter; (punto y coma)** Aunque no es necesario ya que el intérprete lee cada sentencia aunque no exista este carácter. Por convenio deberíamos incluirlo.
- **Existe la opción de incluir comentarios para añadir información en el código fuente del programa.** Estos comentarios suelen servir para dar información al propietario del código u otro desarrollador sobre el contenido del bloque de código en JavaScript. Los comentarios pueden ser de una sola línea o de varias líneas (en bloque)



```
//Ejemplo de comentario de una sola línea:  
// a continuación se muestra un mensaje  
alert("mensaje de prueba");  
//Los comentarios de una sola línea se definen  
añadiendo dos barras oblicuas(//) al principio de  
la línea.  
//Ejemplo de comentario de varias líneas:  
/* Los comentarios de varias líneas son muy útiles  
cuando se necesita incluir bastante información  
en los comentarios*/  
alert("mensaje de prueba");  
//Los comentarios multilínea se definen encerrando  
el texto del comentario entre los símbolos /* y  
*/.
```

## 1.1 | Variables

Las variables de los lenguajes de desarrollo siguen una lógica similar a las variables utilizadas en otras ciencias como las físicas o las matemáticas. Una variable es un contenedor que se usa para almacenar y hacer referencia a otro valor.

De la misma manera que si en Física no existieran las variables no se podrían definir las fórmulas, en los lenguajes de programación no se podrían redactar códigos útiles sin las variables.

Sin las variables sería imposible escribir y crear **“programas genéricos”**, es decir, códigos que funcionan de la misma manera independientemente de los valores concretos usados.

Si no se usaran variables, un código que suma dos números podría redactarse como:

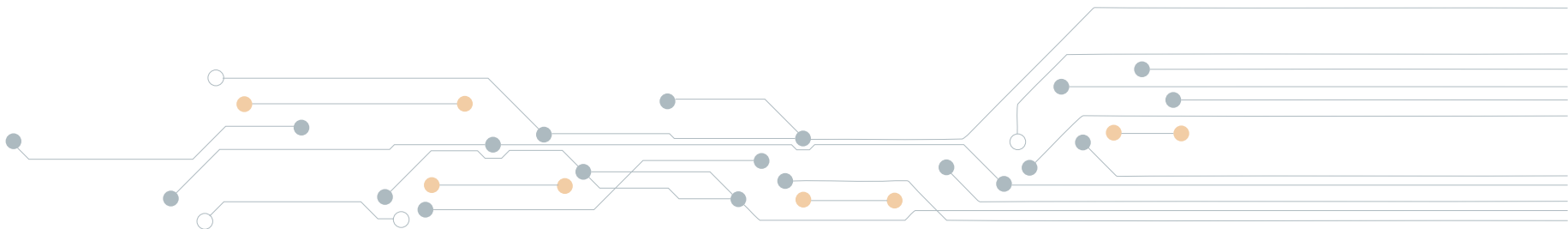
```
res = 4 + 2
```

El código anterior es inútil ya que sólo sirve para el caso en el que el primer número de la suma sea igual a 4 y el segundo número sea igual a 2. En otras opciones, el código obtiene un *res* incorrecto.

Por otro lado, el código se puede reescribir de la siguiente forma usando variables para almacenar y referenciarse a cada número:

Los elementos *num\_1* y *num\_2* son variables que retienen los valores que utiliza el código. El *res* se halla siempre en función del valor retenido por las variables, por lo cual, este código funciona de manera correcta para cualquier par de números que indiquemos. Si se varía el valor de las variables *num\_1* y *num\_2*, el código sigue trabajando correctamente.

```
num_1 = 4  
num_2 = 2  
res = num_1 + num_2
```



Las variables en JavaScript se utilizan mediante la palabra reservada 'var'.

```
var num_1 = 4;  
var num_2 = 2;  
var res = num_1 + num_2;
```

La palabra 'var' solamente se indica al definir por primera vez la variable, y a eso lo llamamos 'declarar' una variable. Cuando se declara una variable no hace falta declarar también el tipo de dato que va a almacenar esa variable.

Si en el momento de declarar una variable se le otorga también un valor, se dice que la variable ha sido inicializada. En JavaScript no es obligatorio inicializar las variables, por lo que pueden ser inicializadas posteriormente.

```
var num_1;  
var num_2;  
num_1 = 4;  
num_2 = 2;  
var res = num_1 + num_2;
```

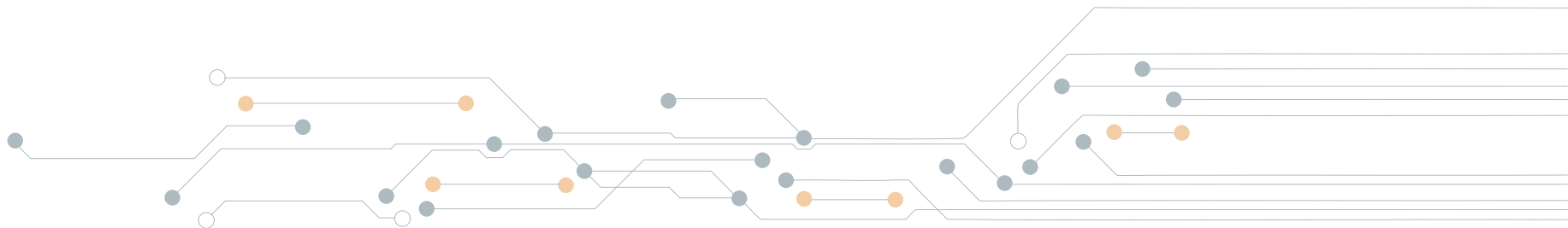
Podemos utilizar una variable no declarada en cualquier sentencia de código. Esta es una de las habilidades más sorprendentes de JavaScript y que muchos otros lenguajes de programación no tienen. JavaScript creará una variable global para esta variable no declarada y la asigna el valor que le corresponda por el código.

```
num_1 = 4;  
num_2 = 2;  
res = num_1 + num_2;
```

De cualquier otra forma, es recomendable declarar todas las variables que se vayan a usar.

El nombre de una variable también se le conoce como identificador y debe cumplir la siguiente normativa:

- El **identificador** únicamente puede estar formado por números, letras, y los símbolos '\$' y '\_' a lo sumo.
- El **primer** carácter del identificador no debe ser un número.



## 1.2 | Tipos de datos

Ya sabemos que todas las variables en JavaScript se crean a través de la palabra reservada *“var”* pero dependiendo de los valores que almacenen pueden ser de un tipo u otro. En otras palabras, el valor que almacena la variable otorga tipo a esa misma variable.

### NUMÉRICOS

Se usan para contener valores numéricos enteros (llamados integer) o decimales (llamados float).

De esta manera, el valor que se asigna a la variable se realiza indicando directamente el número entero o decimal. Los números decimales utilizan el carácter *“.”* en vez de *“,”* para realizar la separación entre la parte entera y la parte decimal:

```
var entero = 99;           // variable tipo entero
var decimal = 9384.23;    // variable tipo decimal
```

### CADENAS DE TEXTO

Se usan para contener caracteres, palabras y/o frases de texto. Para darle el valor a la variable, se encierran los valores entre comillas dobles o simples, que delimitan el inicio y el final de la frase, caracteres o palabras:

```
var sms = "Welcome to our city!";
var nomProducto = 'Escoba';
var letter = 'e';
```

A veces el texto que contienen estas variables no es tan fácil. Si el propio texto encerrado entre comillas dobles o simples tiene comillas dobles o simples que deben aparecer como parte del valor se haría así:

```
/* Comillas simples dentro de comillas dobles*/
var text1 = "Una frase con 'comillas simples' dentro";
/* Comillas dobles dentro de comillas simples*/
var text2 = 'Una frase con "comillas dobles" dentro';
```

A veces, hacen falta caracteres especiales para definir un cambio de línea dentro del texto de nuestra variable, o incluso, quizás queramos meter comillas simples y dobles a la vez dentro de nuestra sentencia.

Si se quiere incluir...	Se debe incluir...
Una nueva línea	\n
Un tabulador	\t
Una comilla simple	\'
Una comilla doble	\"
Una barra inclinada	\\

De esta manera, podemos rehacer el ejemplo anterior:

```
var text1 = 'Una frase con \'comillas simples\' dentro';  
var text2 = "Una frase con \"comillas dobles\" dentro";
```

Este mecanismo de JavaScript se denomina "mecanismo de escape" o "caracteres de escape".

## ARRAYS

También llamados vectores o matrices. Sin embargo, la denominación 'array' es la más utilizada y es un término comúnmente aceptado en el entorno de desarrollo.

Un array es una colección de variables, sin importar los tipos de los que sean cada una. Los arrays sirven para guardar colecciones de valores, de manera que serviría para agrupar diferentes variables. Por ejemplo tenemos esta sucesión de variables con los días de la semana:

```
var dia1 = "Lunes";  
var dia2 = "Martes";  
...  
var dia7 = "Domingo";
```

El código anterior es correcto aparte de poco eficiente y que complica el desarrollo de nuestra programación.

Variando el ejemplo anterior podemos disponerlo de la siguiente forma:

```
var diasSemana =  
["Lunes", "Martes", "Miércoles", "Jueves", "Viernes",  
"Sábado", "Domingo"];
```

Hemos definido una única variable (*diasSemana*) que agrupa un conjunto de valores de variables (los propios días de la semana, que son cadenas). Por tanto la manera de crear o escribir un array es con el contenido entre corchetes ([ ]) y separando cada valor de nuestra variable por una coma (no se pone una coma en el elemento final de nuestro array):

```
var nombre_array = [valor1, valor2, ..., valorN];
```

Ya que tenemos definido nuestro primer array, podemos sacar algunos de sus valores de su interior de esta manera:

```
var diaSeleccionado = diasSemana [0]; // diaSeleccionado  
    = "Lunes"  
var otroDia = diasSemana [5]; // otroDia = "Sábado"
```

La posición 0 del array sería el primer elemento del array (sea cual fuere) y el elemento del array que estaría en la sexta posición sería el elemento 5. Esto sucede porque los índices o posiciones dentro de un array empiezan con el elemento 0 y de ahí en adelante.

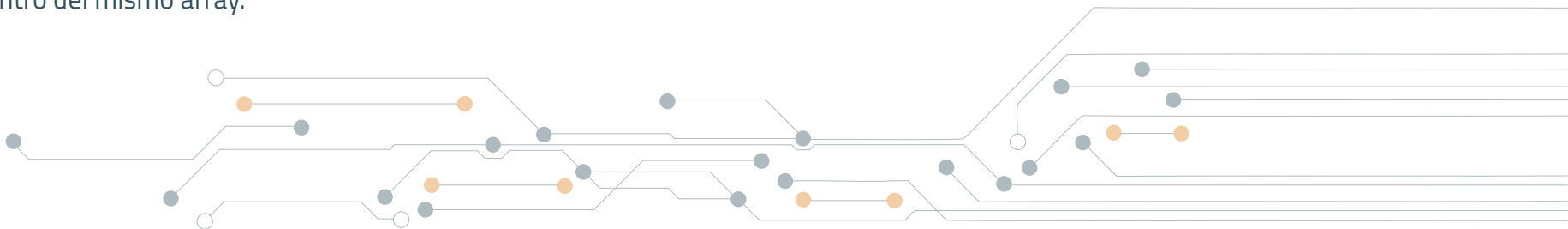
Finalmente debemos advertir que los array tampoco están tipados (no se les asigna un tipo) por lo que podemos meter elementos de cualquier tipo dentro del mismo array.

## BOOLEANOS

Las variables de tipo booleano también son llamadas o denominadas con el nombre de variables de tipo lógico. Estas variables suelen servir para condiciones o para la programación lógica.

Una variable este tipo solo puede almacenar dos valores: true (verdadero) o false (falso).

```
var register = false;  
var mayorEdad = true;
```





## 1.3 | Operadores

Los operadores manipulan los valores de las variables, realizan cálculos matemáticos y comparan los valores de diferentes variables.

### ASIGNACIÓN

Este operador es el principal y el más sencillo. Sirve para asignar un valor a una variable.

```
var num1 = 3;
```

### INCREMENTO Y DECREMENTO

Ambos operadores sirven para decrementar o incrementar el valor de una variable.

```
var num = 5;  
++num;  
alert(num); // num = 6
```

El primer operador (el de incremento) se indica mediante el prefijo ++ y aumenta en una unidad el valor de la variable. El segundo operador (el de decremento) se indica mediante el prefijo -- y disminuye en una unidad el valor de una variable.

```
var num = 5;  
--num;  
alert(num); // num = 4
```

Estos operadores también pueden ser indicados como sufijos en vez de prefijos.

```
var num = 5;  
num++;  
alert(num); // num = 6
```

Aunque el resultado en uno y otro caso es el mismo, no es lo mismo poner el operador como prefijo o sufijo:

```
var num1 = 5;
var num2 = 2;
num3 = num1++ + num2;
// num3 = 7, num1 = 6
```

```
var num1 = 5;
var num2 = 2;
num3 = ++num1 + num2;
// num3 = 8, num1 = 6
```

Cuando el operador ++ está como prefijo → Su valor se incrementa antes de la operación.

Cuando el operador ++ está como sufijo → Su valor se incrementa después de la operación.

## LÓGICOS

Los operadores lógicos son adecuados para realizar condiciones y lógica matemática.

El resultado de estas operaciones siempre da como resultado un valor lógico o de booleano.

## NEGACIÓN

El operador de negación se utiliza para dar el valor contrario a una variable. En el caso de una variable que tenga el valor de un booleano, se le asignará el valor contrario tras usarse el operador de negación.

```
var vis = true;
alert(!vis); // Muestra "false" y no "true"
```

El operador de negación también se puede usar cuando el valor de la variable es un texto o un número. Si es un número y su valor es 0 se transforma en false y si es otro número diferente de 0 su valor es true, por lo que al negarlo en los dos casos sería su valor contrario. En el caso de las cadenas, si es cadena vacía ("" ) se transforma en false y con cualquier otra cadena su valor sería true.

```
var cant = 0;
vac = !cant; // vac = true
cant = 2;
vac = !cant; // vac = false
var mens = "";
mensVac = !mens; // mensVac = true
mens = "Bienvenido";
mensVac = !mens; // mensVac = false
```

## AND

Este operador sirve para combinar los valores de dos variables, usando lógica matemática y solo dando true si ambos valores son true. En otro caso el valor final es false. El operador se define mediante el símbolo "&&".

## OR

Este operador sirve para combinar los valores de dos variables, usando lógica matemática y solo dando true si alguno de los valores es true. En otro caso el valor final es false. El operador se define mediante el símbolo "||".

## MATEMÁTICOS

Los operadores declarados son: *suma (+)*, *resta (-)*, *multiplicación (\*)* y *división (/)*. Estos operadores son todos matemáticos.

```
var num1 = 10;
var num2 = 5;
res = num1 / num2; // res = 2
res = 3 + num1;    // res = 13
res = num2 - 4;    // res = 1
res = num1 * num2; // res = 50
```

Aparte de estos anteriormente comentados, existe el operador de "módulo" que obtiene como valor el resto de una división. Este operador se indica mediante el símbolo "%".

```
var num1 = 10;
var num2 = 5;
res = num1 % num2; // res = 0
num1 = 9;
num2 = 5;
res = num1 % num2; // res = 4
```

## RELACIONALES

Los relacionales: *mayor que (>)*, *menor que (<)*, *mayor o igual (>=)*, *menor o igual (<=)*, *igual que (==)* y *distinto de (!=)*.

El resultado de ellos siempre es un valor de booleano.

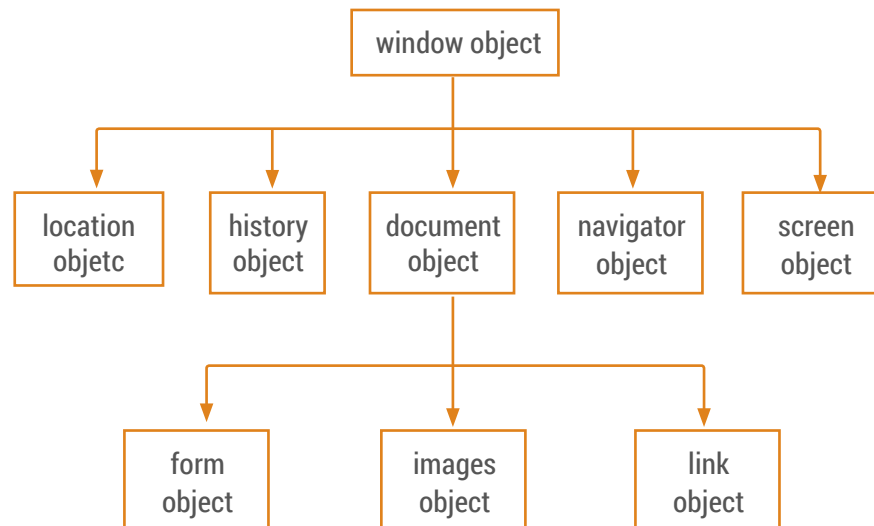
```
var num1 = 3;
var num2 = 5;
res = num1 > num2; // res = false
res = num1 < num2; // res = true

num1 = 5;
num2 = 5;
res = num1 >= num2; // res = true
res = num1 <= num2; // res = true
res = num1 == num2; // res = true
res = num1 != num2; // res = false
```

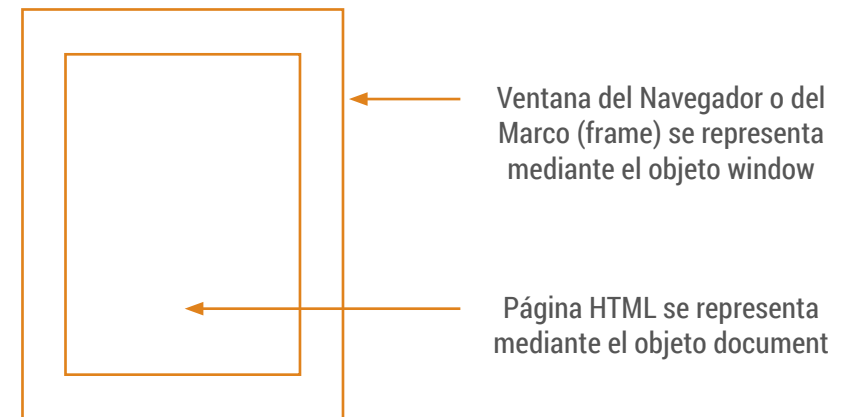
## 1.4 | Objetos del navegador

Cuando se carga una página en un navegador se crean un número de objetos característicos del navegador según el contenido de la página.

La siguiente figura muestra la jerarquía de clases del Modelo de Objetos del Documento (Document Object Model).



El objeto `window` es el de más alto nivel, contiene las propiedades de la ventana y en el supuesto de trabajar con marcos (frames), se genera un objeto `window` para cada uno. El objeto `document` contiene todas las propiedades del documento actual, como son: su color de fondo, enlaces, imágenes, etc.



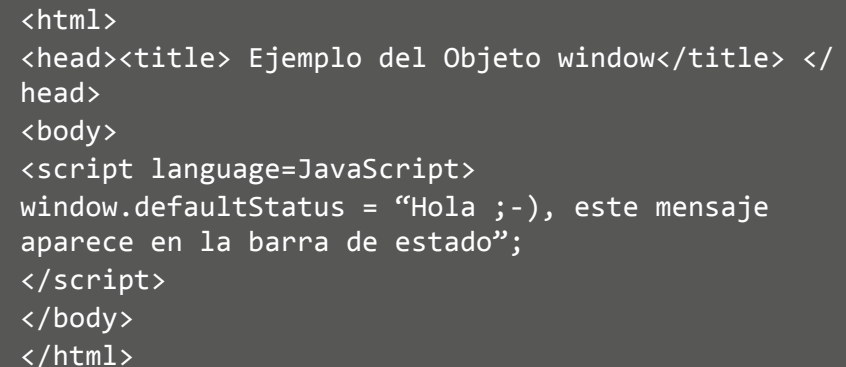
El objeto navigator contiene las propiedades del navegador. El objeto location contiene las propiedades de la URL activa. El objeto history contiene las propiedades que representan a las URL que el usuario ha visitado anteriormente. Es como una caché. El objeto screen contiene información referente a la resolución de la pantalla que muestra la URL.

## EL OBJETO WINDOW

Contiene las propiedades básicas de la ventana y sus componentes. Algunas de los datos más elementales son:

- **defaultStatus** contiene el mensaje que aparece en la barra de estado)
- **frames** es una matriz que representa todos los frames de la ventana
- **length** contiene el número de frames de la ventana
- **name** contiene el nombre de la ventana
- **self** hace referencia a la ventana activa

El siguiente ejemplo muestra cómo modificar el mensaje que aparece en la barra de estado del navegador.



```
<html>
<head><title> Ejemplo del Objeto window</title> </
head>
<body>
<script language=JavaScript>
window.defaultStatus = "Hola ;-), este mensaje
aparece en la barra de estado";
</script>
</body>
</html>
```



El objeto window también posee una serie de métodos que permiten ejecutar funciones específicas con las ventanas, como por ejemplo, crear ventanas y cuadros de diálogo.

También es posible determinar el aspecto que tendrá la nueva ventana del navegador mediante los campos de datos que permiten configurar el menú, la barra de herramientas, la barra de estado, etc. El siguiente ejemplo muestra cómo abrir una nueva ventana desde la ventana actual.

### EL OBJETO LOCATION

El objeto location contiene toda la información sobre la URL que se está visualizando, así como todos los detalles de esa dirección (puerto, protocolo, etc.).

### EL OBJETO SCREEN

Permite obtener información sobre la resolución de la pantalla. En el siguiente ejemplo, se establece el color de fondo de la página de acuerdo a la resolución que soporta la pantalla del usuario.



```
<html>
<head>
<title>Ejemplo de creación de ventana</title>
<script language="JavaScript">
function AbrirVentana() {
ventana=open("", "nueva", "toolbar=no, directories=no,
menubar=no, width=180, height=180");
ventana.document.write("<HEAD><TITLE>Nueva
Ventana </TITLE></HEAD><BODY>");
ventana.document.write("<FONT SIZE=4 COLOR=red>
Nueva Ventana</FONT><BR> <BR><BR>>");
ventana.document.write("<FORM><INPUT
TYPE='button' VALUE='Cerrar' onClick='self.
close()'></FORM>");
}
</script>
</head>
<body>
<form>
<input type="button" value="Abrir una ventana"
onClick="AbrirVentana();">
<br>
</form>
</body>
</html>
```

```

<html>
<head><title> Ejemplo del Objeto screen</title> </
head>
<body>
<script language=JavaScript>
switch (window.screen.colorDepth)
{
    case 1: case 4:
        document.bgColor = "white";
        break;
    case 8: case 15: case 16:
        document.bgColor = "blue";
        break;
    case 24: case 32:
        document.bgColor = "skyblue";
        break;
    default:
        document.bgColor = "white";
}
document.write("Su pantalla soporta color de " +
window.screen.colorDepth + " bit");
</script>
</body>
</html>

```

## EL OBJETO DOCUMENT - LA PÁGINA EN SÍ

El objeto document hace referencia a determinadas características de la página, como son su color de fondo (bgColor), el color de su enlaces, etc.

El código que se muestra a continuación carga una imagen dependiendo de la elección que haga el usuario.

```

<html>
<head><title> Ejemplo del Objeto document</title>
<!-- Se muestra un número diferente de imágenes
dependiendo
    -- del valor que introduzca el usuario
    -- dato: src
    -->
</head>
<body>
<IMG NAME=img1 SRC="" BORDER=0 WIDTH=200 HEIGHT=150>
<script language=JavaScript>
    var myImages = new Array("usa.gif","canada.
gif","jamaica.gif","mexico.gif");
    var imgIndex = prompt("Enter a number from 0 to
3","");
    document.images["img1"].src = myImages[imgIndex];
</script>
</body>
</html>

```

A continuación vemos un ejemplo que permite conectar código a los eventos de la página web. El primero de ellos simplemente muestra una ventana de alerta, mientras que el segundo va modificando de forma aleatoria la imagen que se carga.

### Ejemplo 1

```
<html>
<head><title> Ejemplo de Eventos</title>
</head>
<body>
<script language=JavaScript>
function linkSomePage_onclick() {
    alert('Este enlace no lleva a ninguna parte');
    return false;
}
</script>
<A HREF="somepage.htm" NAME="linkSomePage">
    Pincha Aquí
</A>
<script language=JavaScript>
    window.document.links[0].onclick = linkSomePage_
onclick;
</script>
</body>
</html>
```

### Ejemplo 2

```
<html>
<head><title> Ejemplo del Objeto document</title>
<!-- Se carga una imagen aleatoria
-->
<script language=JavaScript>
var myImages = new Array("usa.gif","canada.
gif","jamaica.gif","mexico.gif");
function changeImg(imgNumber) {
    var imgClicked = document.images[imgNumber];
    var newImgNumber = Math.round(Math.random() * 3);
    while (imgClicked.src.indexOf(myImages[newImgNumber])
!= -1) {
        newImgNumber = Math.round(Math.random() * 3);
    }
    imgClicked.src = myImages[newImgNumber];
    return false;
}
</script>
</head>
<body>
<A HREF="" NAME="linkImg1" onclick="return
changeImg(0)">
    <IMG NAME=img1 SRC="usa.gif" BORDER=0 >
</A>
<A HREF="" NAME="linkImg2" onclick="return
changeImg(1)">
    <IMG NAME=img1 SRC="mexico.gif" BORDER=0 >
</A>
</body>
</html>
```



## EL OBJETO HISTORY

El objeto history contiene información sobre los enlaces que el usuario ha visitado. Se utiliza principalmente para generar botones de avance y retroceso.

## EL OBJETO NAVIGATOR

El objeto navigator permite obtener información del navegador con el que se está visualizando el documento. El siguiente código JavaScript detecta el navegador que se está utilizando y abre la página específica del mismo.

```
<html>
<head><title> Ejemplo del Objeto navigator</title>
<!-- Se detecta el navegador con el que se ha abierto la
página
-->
<script language=JavaScript>
<!--
function getBrowserName() {
    var lsBrowser = navigator.appName;
    if (lsBrowser.indexOf("Microsoft") >= 0) {
        lsBrowser = "MSIE";
    }
    else if (lsBrowser.indexOf("Netscape") >= 0) {
        lsBrowser = "NETSCAPE";
    }
    else {
```

```
        lsBrowser = "UNKNOWN";
    }
    return lsBrowser;
}
function getOS() {
    var userPlat = "unknown";
    var navInfo = navigator.userAgent;
    if ((navInfo.indexOf("windows NT") != -1)
        || (navInfo.indexOf("windows 95") != -1 )
        || (navInfo.indexOf("windows 98") != -1 )
        || (navInfo.indexOf("WinNT") != -1 )
        || (navInfo.indexOf("Win95") != -1 )
        || (navInfo.indexOf("Win98") != -1 )) {
        userPlat = "Win32";
    }
    else if(navInfo.indexOf("Win16") != -1) {
        userPlat = "Win16";
    }
    else if(navInfo.indexOf("Macintosh") != -1) {
        userPlat = "PPC";
    }
    else if(navInfo.indexOf("68K") != -1) {
        userPlat = "68K";
    }
    return userPlat;
}
function getBrowserVersion() {
    var findIndex;
    var browserVersion = 0;
    var browser = getBrowserName();
```

```
if (browser == "MSIE") {
    browserVersion = navigator.userAgent;
    findIndex = browserVersion.indexOf(browser) + 5;
    browserVersion = parseInt(browserVersion.
substring(findIndex,findIndex + 1));
}
else {
    browserVersion = parseInt(navigator.appVersion.
substring(0,1));
}
return browserVersion;
}
-->
</script>
</head>
<body>
<script language=JavaScript>
<!--
var userOS = getOS();
var browserName = getBrowserName();
var browserVersion = getBrowserVersion();
if (browserVersion < 4 || browserName == "UNKNOWN" ||
userOS == "Win16") {
    document.write("<H2>Sorry this browser version is not
supported</H2>")
}
else if (browserName == "NETSCAPE") {
    location.replace("NetscapePage.html");
}
else {
```

```
location.replace("MSIEPage.html");
}
-->
</script>
<noscript>
    <H2>Esta página requiere un navegador que soporte
JavaScript</H2>
</noscript>
</body>
</html>
```

*Telefonica*

---

EDUCACIÓN DIGITAL