Tema 6 - MODELO DE OBJETOS PREDEFINIDOS EN JAVASCRIPT.

CASO PRÁCTICO.

Antonio ha completado correctamente la fase de introducción y fundamentos básicos del lenguaje JavaScript, y ahora comienza a investigar en las características de los objetos predefinidos en JavaScript.

Estos objetos le van a permitir gestionar ventanas, marcos, propiedades de los navegadores, de las URL, etc. en JavaScript.

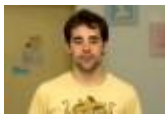
Además, también va a poder realizar operaciones matemáticas, de fecha y de cadenas, con otros tantos objetos nativos del lenguaje JavaScript.

Antonio tiene una pequeña reunión con Ada y con su tutor Juan, para comentar los progresos realizados hasta este momento y se pone manos a la obra con esta nueva sección.

1. Objetos de más alto nivel en Javascript

CASO PRÁCTICO.

Bajo la tutoría de Juan, Antonio se dispone a profundizar en los objetos básicos y de más alto nivel de JavaScript. Estos objetos, los encontrará en prácticamente la mayoría de aplicaciones que haga con JavaScript, por lo que será fundamental, que tenga muy claras las características y diferentes funcionalidades que estos objetos le van a aportar a sus aplicaciones.



Una página web, es un documento HTML que será interpretado por los navegadores de forma gráfica, pero que también va a permitir el acceso al código fuente de la misma.

El Modelo de Objetos del Documento (DOM), permite ver el mismo documento de otra manera, describiendo el contenido del documento como un conjunto de objetos, sobre los que un programa de Javascript puede interactuar.

Según el W3C, el Modelo de Objetos del Documento es una interfaz de programación de aplicaciones (API), para documentos válidos HTML y bien construidos XML. Define la estructura lógica de los documentos, y el modo en el que se acceden y se manipulan.

Ahora que ya has visto en la unidad anterior, los fundamentos de la programación, vamos a profundizar un poco más en lo que se refiere a los objetos, que podremos colocar en la mayoría de nuestros documentos.

Definimos como **objeto**, una entidad con una serie de **propiedades** que definen su estado, y unos **métodos** (funciones), que actúan sobre esas propiedades.

La forma de acceder a una propiedad de un objeto es la siguiente:

nombreobjeto.propiedad

La forma de acceder a un método de un objeto es la siguiente:

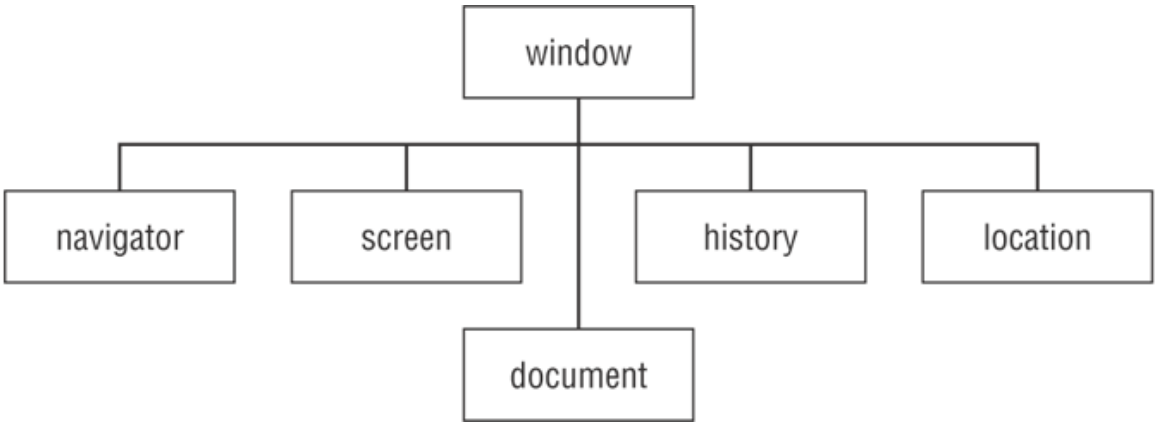
nombreobjeto.metodo([parámetros opcionales])

También podemos referenciar a una propiedad de un objeto, por su índice en la creación. Los índices comienzan por 0.

En esta unidad, nos enfocaremos en objetos de alto nivel, que encontrarás frecuentemente en tus aplicaciones de JavaScript: *window*, *location*, *navigator y document*. El objetivo, no es solamente indicarte las nociones básicas para que puedas comenzar a realizar tareas sencillas, sino también, el prepararte para profundizar en las propiedades y métodos, gestores de eventos, etc. que encontrarás en unidades posteriores.

En esta unidad, verás solamente las propiedades básicas, y los métodos de los objetos mencionados anteriormente.

Te mostramos aquí el gráfico del modelo de objetos de alto nivel, para todos los navegadores que permitan usar JavaScript.



Es muy importante que tengas este gráfico en mente porque va a ser la guía a lo largo de toda esta unidad.

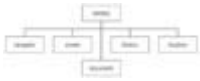
1.1 Objeto window

En la jerarquía de objetos, tenemos en la parte superior el objeto window.

Este objeto está situado justamente ahí, porque es el contenedor principal de todo el contenido que se visualiza en el navegador. Tan pronto como se abre una ventana (window) en el navegador, incluso aunque no se cargue ningún documento en ella, este objeto window ya estará definido en memoria.

Además de la sección de contenido del objeto window, que es justamente dónde se cargarán los documentos, el campo de influencia de este objeto, abarca también las dimensiones de la ventana, así como todo lo que rodea al área de contenido: las barras de desplazamiento, barra de herramientas, barra de estado, etc.

Cómo se ve en el gráfico de la jerarquía de objetos, debajo del objeto window tenemos otros objetos como el *navigator*, *screen*, *history*, *location* y el objeto *document*. Este objeto document será el que contendrá toda la jerarquía de objetos, que tengamos dentro de nuestra página html.



Atención: En los navegadores más modernos, los usuarios tienen la posibilidad de abrir las páginas tanto en nuevas pestañas dentro de un navegador, como en nuevas ventanas de navegador. Para JavaScript tanto las ventanas de navegador, como las pestañas, son ambos objetos window.

Acceso a propiedades y métodos

Para acceder a las propiedades y métodos del objeto window, lo podremos hacer de diferentes formas, dependiendo más de nuestro estilo, que de requerimientos sintácticos. Así, la forma más lógica y común de realizar esa referencia, incluiría el objeto window tal y como se muestra en este ejemplo:

```
window.nombrePropiedad
window.nombreMétodo( [parámetros] )
```

Como puedes ver, los parámetros van entre corchetes, indicando que son opcionales y que dependerán del método al que estemos llamando.

Un objeto window también se podrá referenciar mediante la palabra self, cuando estamos haciendo la referencia desde el propio documento contenido en esa ventana:

```
self.nombrePropiedad
self.nombreMétodo( [parámetros] )
```

Podremos usar cualquiera de las dos referencias anteriores, pero intentaremos dejar la palabra reservada self, para scripts más complejos en los que tengamos múltiples marcos y ventanas.

Debido a que el objeto window siempre estará presente cuando ejecutemos nuestros scripts, podremos omitirlo, en referencias a los objetos dentro de esa ventana. Así que, si escribimos:

```
nombrePropiedad
nombreMétodo( [parámetros] )
```

También funcionaría sin ningún problema, porque se asume que esas propiedades o métodos, son del objeto de mayor jerarquía (el objeto window) en el cuál nos encontramos.

CITA PARA PENSAR.
"Sólo cerrando las puertas detrás de uno se abren ventanas hacia el porvenir. "
de SAGAN, Françoise.

1.1.1 Gestión de ventanas

Un script no creará nunca la ventana principal de un navegador. Es el usuario, quien realiza esa tarea abriendo una URL en el navegador o un archivo desde el menú de abrir. Pero sin embargo, un script que esté ejecutándose en una de las ventanas principales del navegador, si que podrá crear o abrir nuevas sub-ventanas.

El método que genera una nueva ventana es **window.open()**. Este método contiene hasta tres parámetros, que definen las características de la nueva ventana: la URL del documento a abrir, el nombre de esa ventana y su apariencia física (tamaño, color,etc.).

Por ejemplo, si consideramos la siguiente instrucción que abre una nueva ventana de un tamaño determinado y con el contenido de un documento HTML:

var subVentana=window.open("nueva.html","nueva","height=800,width=600");

Lo importante de esa instrucción, es la asignación que hemos hecho en la variable subVentana. De esta forma podremos a lo largo de nuestro código, referenciar a la nueva ventana desde el script original de la ventana principal. Por ejemplo, si quisiéramos cerrar la nueva ventana desde nuestro script, simplemente tendríamos que hacer: subVentana.close();

Aquí si que es necesario especificar subVentana, ya que si escribiéramos window.close(), self.close() o close() estaríamos intentando cerrar nuestra propia ventana (previa confirmación), pero no la subVentana que creamos en los pasos anteriores.

Véase el siguiente ejemplo que permite abrir y cerrar una sub-ventana:

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="content-type" content="text/html;charset=utf-8">
<title>Apertura y Cierre de Ventanas</title>
<script type="text/javascript">
function inicializar()
{
	document.getElementById("crear-ventana").onclick=crearNueva;
	document.getElementById("cerrar-ventana").onclick=cerrarNueva;
}

var nuevaVentana;
function crearNueva()
{
	nuevaVentana = window.open("http://www.google.es","","height=400,width=800");
}

function cerrarNueva()
{
if (nuevaVentana)
{
	nuevaVentana.close(); nuevaVentana = null;
}
}
</script>
</head>
<body onLoad="inicializar()">
<h1>Abrimos y cerramos ventanas</h1>
<form>
<p> <input type="button" id="crear-ventana" value="Crear Nueva Ventana">
<input type="button" id="cerrar-ventana" value="Cerrar Nueva Ventana"> </p>
</form>
</html>
```

1.1.2 Propiedades y métodos

El objeto window representa una ventana abierta en un navegador. Si un documento contiene marcos (<frame> o <iframe>), el navegador crea un objeto window para el documento HTML, y un objeto window adicional para para cada marco.

Propiedades del objeto window:

Propiedad	Descripción
closed	Devuelve un valor Boolean indicando cuando una ventana ha sido cerrada o no.
defaultStatus	Ajusta o devuelve el valor por defecto de la barra de estado de una ventana.
document	Devuelve el objeto document para la ventana.
frames	Devuelve un array de todos los marcos (incluidos iframes) de la ventana actual.
history	Devuelve el objeto history de la ventana.
length	Devuelve el número de frames (incluyendo iframes) que hay en dentro de una ventana.
location	Devuelve la Localización del objeto ventana (URL del fichero).
name	Ajusta o devuelve el nombre de una ventana.
navigator	Devuelve el objeto navigator de una ventana.
opener	Devuelve la referencia a la ventana que abrió la ventana actual.
parent	Devuelve la ventana padre de la ventana actual.
self	Devuelve la ventana actual.
status	Ajusta el texto de la barra de estado de una ventana.

Métodos del objeto window:

Método	Descripción
alert()	Muestra una ventana emergente de alerta y un botón de aceptar.
blur()	Elimina el foco de la ventana actual.
clearInterval()	Resetea el cronómetro ajustado con setInterval().
setInterval()	Llama a una función o evalúa una expresión en un intervalo especificado (en milisegundos).
close()	Cierra la ventana actual.
confirm()	Muestra una ventana emergente con un mensaje, un botón de aceptar y un botón de cancelar.
focus()	Coloca el foco en la ventana actual.
open()	Abre una nueva ventana de navegación.
prompt()	Muestra una ventana de diálogo para introducir datos.

DEBES CONOCER.

El siguiente enlace amplía información sobre el objeto Window y todas sus propiedades y métodos.

Texto enlace: Más información y ejemplos sobre el objeto Window.

URL: http://www.webestilo.com/javascript/js17.phtml

Título: Enlace a la página de webestilo, con información ampliada del objeto Window.

1.2 Objeto location

El objeto location contiene información referente a la URL actual.

Este objeto, es parte del objeto window y accedemos a él a través de la propiedad window.location.

Propiedades del objeto location:

Propiedad	Descripción
hash	Cadena que contiene el nombre del enlace, dentro de la URL.
host	Cadena que contiene el nombre del servidor y el número del puerto, dentro de la URL.
hostname	Cadena que contiene el nombre de dominio del servidor (o la dirección IP), dentro de la URL.
href	Cadena que contiene la URL completa.
pathname	Cadena que contiene el camino al recurso, dentro de la URL.
port	Cadena que contiene el número de puerto del servidor, dentro de la URL.
protocol	Cadena que contiene el protocolo utilizado (incluyendo los dos puntos), dentro de la URL.
search	Cadena que contiene la información pasada en una llamada a un script, dentro de la URL.

Métodos del objeto location:

Método	Descripción
assign()	Carga un nuevo documento.
reload()	Vuelve a cargar la URL especificada en la propiedad href del objeto location.
replace()	Reemplaza el historial actual mientras carga la URL especificada en cadenaURL.

CITA PARA PENSAR.

"Mil rutas se apartan del fin elegido, pero hay una que llega a él."

de MONTAIGNE, Michel de.

DEBES CONOCER.

El siguiente enlace amplía información sobre el objeto Location y todas sus propiedades y métodos.

Texto enlace: Más información y ejemplos sobre el objeto Location.

URL: http://www.webestilo.com/javascript/js19.phtml

Título: Enlace a la página de webestilo, con más información y ejemplos del objeto Location.

1.3 Objeto navigator

Este objeto navigator, contiene información sobre el navegador que estamos utilizando cuando abrimos una URL o un documento local.

Propiedades del objeto navigator:

Propiedad	Descripción
appCodeName	Cadena que contiene el nombre en código del navegador.
appName	Cadena que contiene el nombre del cliente.
appVersion	Cadena que contiene información sobre la versión del cliente.
cookieEnabled	Determina si las cookies están o no habilitadas en el navegador.
platform	Cadena con la plataforma sobre la que se está ejecutando el programa cliente.
userAgent	Cadena que contiene la cabecera completa del agente enviada en una petición HTTP. Contiene la información de las propiedades appCodeName y appVersion.

Métodos del objeto navigator:

Método	Descripción
javaEnabled()	Devuelve true si el cliente permite la utilización de Java, en caso contrario, devuelve false.

DEBES CONOCER.

El siguiente enlace amplía información sobre el objeto Navigator y todas sus propiedades y métodos.

Texto enlace: Más información y ejemplos sobre el objeto Navigator.

URL: http://www.webestilo.com/javascript/js21.phtml

Título: Enlace a la página de webestilo con más información y ejemplos del objeto Navigator.

1.4 Objeto document

Cada documento cargado en una ventana del navegador, será un objeto de tipo document.

El objeto document proporciona a los scripts, el acceso a todos los elementos HTML dentro de una página.

Este objeto forma parte además del objeto window, y puede ser accedido a través de la propiedad window.document o directamente document (ya que podemos omitir la referencia a la window actual).

Colecciones dentro del objeto document:

Colección	Descripción
anchors[]	Es un array que contiene todos los hiperenlaces del documento.
forms[]	Es un array que contiene todos los formularios del documento.
images[]	Es un array que contiene todas las imágenes del documento.
links[]	Es un array que contiene todos los enlaces del documento.

Propiedades del objeto document:

Propiedad	Descripción
cookie	Devuelve todos los nombres/valores de las cookies en el documento.
domain	Cadena que contiene el nombre de dominio del servidor que cargó el documento.
referrer	Cadena que contiene la URL del documento desde el cuál llegamos al documento actual.
title	Devuelve o ajusta el título del documento.
URL	Devuelve la URL completa del documento.

Métodos del objeto document:

Método	Descripción
close()	Cierra el flujo abierto previamente con document.open().
getElementById()	Para acceder a un elemento identificado por el id escrito entre paréntesis.
getElementsByName()	Para acceder a los elementos identificados por el atributo name escrito entre paréntesis.
getElementsByTagName()	Para acceder a los elementos identificados por el tag o la etiqueta escrita entre paréntesis.
open()	Abre el flujo de escritura para poder utilizar document.write() o document.writeln en el documento.
write()	Para poder escribir expresiones HTML o código de JavaScript dentro de un documento.
writeln()	Lo mismo que write() pero añade un salto de línea al final de cada instrucción.

DEBES CONOCER.
El siguiente enlace amplía información sobre el objeto Document todas sus propiedades y métodos.
Texto enlace: Más información y ejemplos sobre el objeto Document.
URL: http://www.w3schools.com/jsref/dom_obj_document.asp
Título: Enlace a la página de W3Schools, con información y ejemplos completos del objeto Document.

1.5 Objeto screen

El objeto screen contiene información sobre la pantalla del cliente.

No existe un estándar que se pueda aplicar al objeto screen, pero la mayoría de los navegadores lo soportan.

Propiedades del objeto screen:

Propiedad	Descripción
availHeight	Devuelve la altura de la pantalla (sin incluir la barra de tareas de Windows)
availWidth	Devuelve el ancho de la pantalla (sin incluir la barra de tareas de Windows)
colorDepth	Devuelve la profundidad de bits de la paleta de colores.
height	Devuelve la altura total de la pantalla.
PixelDepth	Devuelve la resolución de color (en bits por pixel) de la pantalla.
width	Devuelve el ancho total de la pantalla.

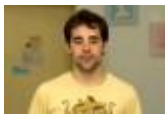
2. Objetos nativos en Javascript

CASO PRÁCTICO.

El lenguaje JavaScript es un lenguaje basado en objetos. A Antonio le suena un poco el tema de objetos aunque nunca trabajó intensivamente con ellos. Como todos los lenguajes que incorporan sus funciones para realizar acciones, conversiones, etc., en JavaScript también dispone de unos objetos nativos que le van a permitir a Antonio el realizar muchas de esas tareas.

Estos objetos, hacen referencia al trabajo con cadenas de texto, operaciones matemáticas, números, valores booleanos y trabajo con fechas y horas.

Ésto le va a ser muy útil para realizar su aplicación ya que tendrá que realizar diferentes tipos de conversiones de datos, trabajar intensivamente con cadenas y por supuesto con fechas y horas.



Aunque no hemos visto como crear objetos, sí que ya hemos dado unas pinceladas a lo que son los objetos, propiedades y métodos.

En esta sección vamos a echar una ojeada a objetos que son nativos en JavaScript, ésto es, aquello que JavaScript nos da, listos para su utilización en nuestra aplicación.

Echaremos un vistazo a los objetos String, Math, Number, Boolean y Date.

CITA PARA PENSAR.

"Si me hubieran hecho objeto sería objetivo, pero me hicieron sujeto."

de BERGAMÍN, José.

REFLEXIONA.

¿Te has parado a pensar alguna vez que nuestro mundo está rodeado de objetos por todas partes?

¿Sabes que prácticamente, todos esos objetos tienen algunas propiedades como pueden ser tamaño, color, peso, tipo de corriente que usan, temperatura, tipo de combustible, etc..?

¿Sabes que también podemos realizar acciones con esos objetos, como pueden ser encender, apagar, mover, abrir, cerrar, subir temperatura, bajar temperatura, marcar número, colgar, etc..?

2.1 Objeto String

Una cadena (string) consta de uno o más caracteres de texto, rodeados de comillas simples o dobles; da igual cuales usemos ya que se considerará una cadena de todas formas, pero en algunos casos resulta más cómodo el uso de unas u otras. Por ejemplo si queremos meter el siguiente texto dentro de una cadena de JavaScript:

```
<input type="checkbox" name="coche" />Audi A6
```

podremos emplear las comillas dobles o simples:

```
var cadena = '<input type="checkbox" name="coche" />Audi A6';
var cadena = "<input type='checkbox' name='coche' />Audi A6";
```

Si queremos emplear comillas dobles al principio y fin de la cadena, y que en el contenido aparezcan también comillas dobles, tendríamos que escaparlas con \", por ejemplo:

```
var cadena = "<input type=\"checkbox\" name=\"coche\" />Audi A6";
```

Cuando estamos hablando de cadenas muy largas, podríamos concatenarlas con +=, por ejemplo:

```
var nuevoDocumento = "";
nuevoDocumento += "<!DOCTYPE html>";
nuevoDocumento += "<html>" ;
nuevoDocumento += "<head>";
nuevoDocumento += '<meta http-equiv="content-type"';
nuevoDocumento += ' content="text/html;charset=utf-8">';
….
```

Si queremos concatenar el contenido de una variable dentro de una cadena de texto emplearemos el símbolo + :

```
nombreEquipo = prompt("Introduce el nombre de tu equipo favorito:","");
var mensaje= "El " + nombreEquipo + " ha sido el campeón de la Copa del Rey!";
alert(mensaje);
```

Caracteres especiales o caracteres de escape

La forma en la que se crean las cadenas en JavaScript, hace que cuando tengamos que emplear ciertos caracteres especiales en una cadena de texto, tengamos que escaparlos, empleando el símbolo \ seguido del carácter.

Vemos aquí un listado de los caracteres especiales o de escape en JavaScript:

Símbolos	Explicación
\"	Comillas dobles
\'	Comilla simple
\\	Barra inclinada
\b	Retroceso
\t	Tabulador
\n	Nueva Línea
\r	Salto de Línea
\f	Avance de página

DEBES CONOCER.

El siguiente enlace amplía información sobre el objeto String y todas sus propiedades y métodos.

Texto enlace: Más información y ejemplos sobre el objeto String.

URL: http://www.w3schools.com/jsref/jsref_obj_string.asp

Título: Enlace a la página de W3Schools, con información ampliada y ejemplos del objeto String.

2.1.1 Propiedades y métodos del objeto String

Para crear un objeto String lo podremos hacer de la siguiente forma:

var miCadena = new String("texto de la cadena");

O también se podría hacer:

var miCadena = "texto de la cadena";

Es decir, cada vez que tengamos una cadena de texto, en realidad es un objeto String que tiene propiedades y métodos:

cadena.propiedad;
cadena.metodo([parámetros]);

Propiedades del objeto String:

Propiedad	Descripción
length	Contiene la longitud de una cadena.

Métodos del objeto String:

Métodos	Descripción
charAt()	Devuelve el carácter especificado por la posición que se indica entre paréntesis.
charCodeAt()	Devuelve el Unicode del carácter especificado por la posición que se indica entre paréntesis.
concat()	Une una o más cadenas y devuelve el resultado de esa unión.
fromCharCode()	Convierte valores Unicode a caracteres.
indexOf()	Devuelve la posición de la primera ocurrencia del carácter buscado en la cadena.
lastIndexOf()	Devuelve la posición de la última ocurrencia del carácter buscado en la cadena.
match()	Busca una coincidencia entre una expresión regular y una cadena y devuelve las coincidencias o null si no ha encontrado nada.
replace()	Busca una subcadena en la cadena y la reemplaza por la nueva cadena especificada.
search()	Busca una subcadena en la cadena y devuelve la posición dónde se encontró.
slice()	Extrae una parte de la cadena y devuelve una nueva cadena.
split()	Divide una cadena en un array de subcadenas.
substr()	Extrae los caracteres de una cadena, comenzando en una determinada posición y con el número de caracteres indicado.
substring()	Extrae los caracteres de una cadena entre dos índices especificados.
toLowerCase()	Convierte una cadena en minúsculas.
toUpperCase()	Convierte una cadena en mayúsculas.

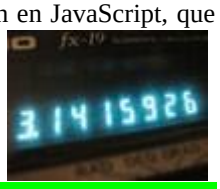
Ejemplos de uso:

```
var cadena="El parapente es un deporte de riesgo medio";

document.write("La longitud de la cadena es: "+ cadena.length + "<br/>");
document.write(cadena.toLowerCase()+ "<br/>");
document.write(cadena.charAt(3)+ "<br/>");
document.write(cadena.indexOf('pente')+ "<br/>");
document.write(cadena.substring(3,16)+ "<br/>");
```

2.2 Objeto Math

Ya vimos anteriormente algunas funciones, que nos permitían convertir cadenas a diferentes formatos numéricos (parseInt, parseFloat). A parte de esas funciones, disponemos de un objeto Math en JavaScript, que nos permite realizar operaciones matemáticas. El objeto Math no es un constructor (no nos permitirá por lo tanto crear o instanciar nuevos objetos que sean de tipo Math), por lo que para llamar a sus propiedades y métodos, lo haremos anteponiendo Math a la propiedad o el método. Por ejemplo:




```
var x = Math.PI;            // Devuelve el número PI.
var y = Math.sqrt(16);      // Devuelve la raíz cuadrada de 16.
```

Propiedades del objeto Math:

Propiedad	Descripción
E	Devuelve el número Euler (aprox. 2.718).
LN2	Devuelve el logaritmo neperiano de 2 (aprox. 0.693).
LN10	Devuelve el logaritmo neperiano de 10 (aprox. 2.302).
LOG2E	Devuelve el logaritmo base 2 de E (aprox. 1.442).
LOG10E	Devuelve el logaritmo base 10 de E (aprox. 0.434).
PI	Devuelve el número PI (aprox. 3.14159).
SQRT2	Devuelve la raíz cuadrada de 2 (aprox. 1.414).

Métodos del objeto Math:

Método	Descripción
abs(x)	Devuelve el valor absoluto de x.
acos(x)	Devuelve el arcocoseno de x, en radianes.
asin(x)	Devuelve el arcoseno de x, en radianes.
atan(x)	Devuelve el arcotangente de x, en radianes con un valor entre -PI/2 y PI/2.
atan2(y,x)	Devuelve el arcotangente del cociente de sus argumentos.
ceil(x)	Devuelve el número x redondeado al alta hacia el siguiente entero.
cos(x)	Devuelve el coseno de x (x está en radianes).
floor(x)	Devuelve el número x redondeado a la baja hacia el anterior entero.
log(x)	Devuelve el logaritmo neperiando (base E) de x.
max(x,y,z,...,n)	Devuelve el número más alto de los que se pasan como parámetros.
min(x,y,z,...,n)	Devuelve el número más bajo de los que se pasan como parámetros.
pow(x,y)	Devuelve el resultado de x elevado a y.
random()	Devuelve un número al azar entre 0 y 1.
round(x)	Redondea x al entero más próximo.
sin(x)	Devuelve el seno de x (x está en radianes).
sqrt(x)	Devuelve la raíz cuadrada de x.
tan(x)	Devuelve la tangente de un ángulo.

Ejemplos de uso:

```
document.write(Math.cos(3) + "<br />");
document.write(Math.asin(0) + "<br />");
document.write(Math.max(0,150,30,20,38) + "<br />");
document.write(Math.pow(7,2) + "<br />");
document.write(Math.round(0.49) + "<br />");
```

2.3 Objeto Number

El objeto Number se usa muy raramente, ya que para la mayor parte de los casos, JavaScript satisface las necesidades del día a día con los valores numéricos que almacenamos en variables. Pero el objeto Number contiene alguna información y capacidades muy interesantes para programadores más serios.

Lo primero, es que el objeto Number contiene propiedades que nos indican el rango de números soportados en el lenguaje. El número más alto es 1.79E + 308; el número más bajo es 2.22E-308. Cualquier número mayor que el número más alto, será considerado como infinito positivo, y si es más pequeño que el número más bajo, será considerado infinito negativo.

Los números y sus valores están definidos internamente en JavaScript, como valores de doble precisión y de 64 bits.

El objeto Number, es un objeto envoltorio para valores numéricos primitivos.
Los objetos Number son creados con **new Number()**.

Propiedades del objeto Number:

Propiedad	Descripción
constructor	Devuelve la función que creó el objeto Number.
MAX_VALUE	Devuelve el número más alto disponible en JavaScript.
MIN_VALUE	Devuelve el número más pequeño disponible en JavaScript.
NEGATIVE_INFINITY	Representa a infinito negativo (se devuelve en caso de overflow).
POSITIVE_INFINITY	Representa a infinito positivo (se devuelve en caso de overflow).
prototype	Permite añadir nuestras propias propiedades y métodos a un objeto.

Métodos del objeto Number:

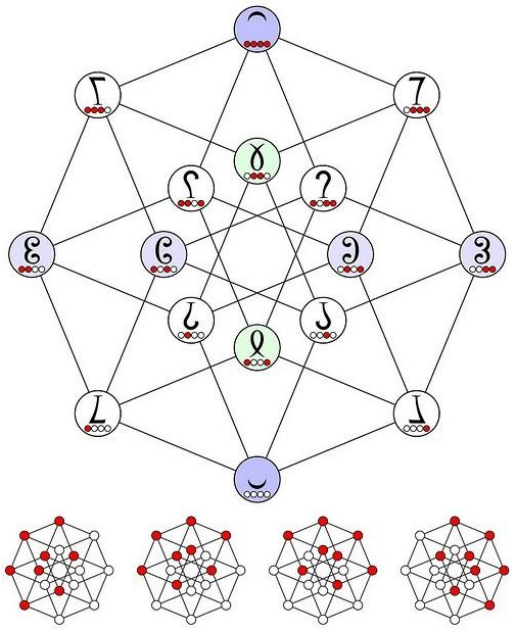
Método	Descripción
toExponential(x)	Convierte un número a su notación exponencial.
toFixed(x)	Formatea un número con x digitos decimales después del punto decimal.
toPrecision(x)	Formatea un número a la longitud x.
toString()	Convierte un objeto Number en una cadena. • Si se pone 2 como parámetro se mostrará el número en binario. • Si se pone 8 como parámetro se mostrará el número en octal. • Si se pone 16 como parámetro se mostrará el número en hexadecimal.
valueOf()	Devuelve el valor primitivo de un objeto Number.

Algunos ejemplos de uso:

```
var num = new Number(13.3714);
document.write(num.toPrecision(3)+"<br />");
document.write(num.toFixed(1)+"<br />");
document.write(num.toString(2)+"<br />");
document.write(num.toString(8)+"<br />");
document.write(num.toString(16)+"<br />");
document.write(Number.MIN_VALUE);
document.write(Number.MAX_VALUE);
```

2.4 Objeto Boolean

El objeto Boolean se utiliza para convertir un valor no Booleano, a un valor Booleano (true o false).
Propiedades del objeto Boolean:



Propiedad	Descripción
constructor	Devuelve la función que creó el objeto Boolean.
prototype	Te permitirá añadir propiedades y métodos a un objeto.

Métodos del objeto Boolean:

Método	Descripción
toString()	Convierte un valor Boolean a una cadena y devuelve el resultado.
valueOf()	Devuelve el valor primitivo de un objeto Boolean.

Algunos ejemplos de uso:

```
var bool = new Boolean(1);
document.write(bool.toString());
document.write(bool.valueOf());
```

PARA SABER MÁS.

Texto enlace: Más información y ejemplos sobre el objeto Boolean.
URL: http://www.desarrolloweb.com/articulos/777.php
Título: Enlace a la página de desarrollo web, con información y ejemplos del objeto Boolean.

2.5 Objeto Date

El objeto Date se utiliza para trabajar con fechas y horas. Los objetos Date se crean con new Date().
Hay 4 formas de instanciar (crear un objeto de tipo Date):

```
var d = new Date();
var d = new Date(milisegundos);
var d = new Date(cadena de Fecha);
var d = new Date(año, mes, día, horas, minutos, segundos, milisegundos);
        // (el mes comienza en 0, Enero sería 0, Febrero 1, etc.)
```

Propiedades del objeto Date:

Propiedad	Descripción
constructor	Devuelve la función que creó el objeto Date.
prototype	Te permitirá añadir propiedades y métodos a un objeto.

Algunos métodos del objeto Date:

Método	Descripción
getDate()	Devuelve el día del mes (de 1-31).
getDay()	Devuelve el día de la semana (de 0-6).
getFullYear()	Devuelve el año (4 dígitos).
getHours()	Devuelve la hora (de 0-23).
getMilliseconds()	Devuelve los milisegundos (de 0-999).
getMinutes()	Devuelve los minutos (de 0-59).
getMonth()	Devuelve el mes (de 0-11).
getSeconds()	Devuelve los segundos (de 0-59).
getTime()	Devuelve los milisegundos desde media noche del 1 de Enero de 1970.
getTimezoneOffset()	Devuelve la diferencia de tiempo entre GMT y la hora local, en minutos.
getUTCDate()	Devuelve el día del mes en base a la hora UTC (de 1-31).
getUTCDay()	Devuelve el día de la semana en base a la hora UTC (de 0-6).
getUTCFullYear()	Devuelve el año en base a la hora UTC (4 dígitos).
setDate()	Ajusta el día del mes del objeto (de 1-31).
setFullYear()	Ajusta el año del objeto (4 dígitos).
setHours()	Ajusta la hora del objeto (de 0-23).

Algunos ejemplos de uso:

```
var d = new Date();
document.write(d.getFullYear());
document.write(d.getMonth());
document.write(d.getUTCDay());
var d2 = new Date(5,28,2011,22,58,00);
d2.setMonth(0);
d.setFullYear(2020);
```

DEBES CONOCER.

El siguiente enlace amplía información sobre el objeto Date y todas sus propiedades y métodos.

Texto enlace: Más información y ejemplos sobre el objeto Date.

URL: http://www.w3schools.com/js/js_obj_date.asp

Título: Enlace a la página de W3Schools, con información y ejemplos completos de uso del objeto Date.