Miguel Ángel Dominguez Llorente

[Dirección de correo electrónico]

Descripción breve

[Dibujar su lector con un resumen de la participación. Normalmente es un breve resumen del documento.   
Cuando esté listo para agregar contenido, haga clic aquí y empiece a escribir.]

UD 4 Modelos de objetos predefinidos en Javacript

[Subtítulo del documento]

Contenido

[1.- Objetos de más alto nivel en Javascript. 3](#_Toc187148756)

[1.1.- Objeto window. 5](#_Toc187148757)

[1.1.2.- Propiedades y métodos. 8](#_Toc187148758)

[2.- Objetos nativos en Javascript. 9](#_Toc187148759)

[2.3.- Objeto Number. 10](#_Toc187148760)

0. – Introducción  
  
En esta unidad de trabajo profundizaremos en los objetos de más alto nivel en JavaScript, los cuales estarán disponibles en todas nuestras aplicaciones. Veremos como gestionar los marcos o frames y la forma de comunicarnos entre múltiples ventanas que tengamos abiertas en el navegador.

# 1.- Objetos de más alto nivel en Javascript.

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:

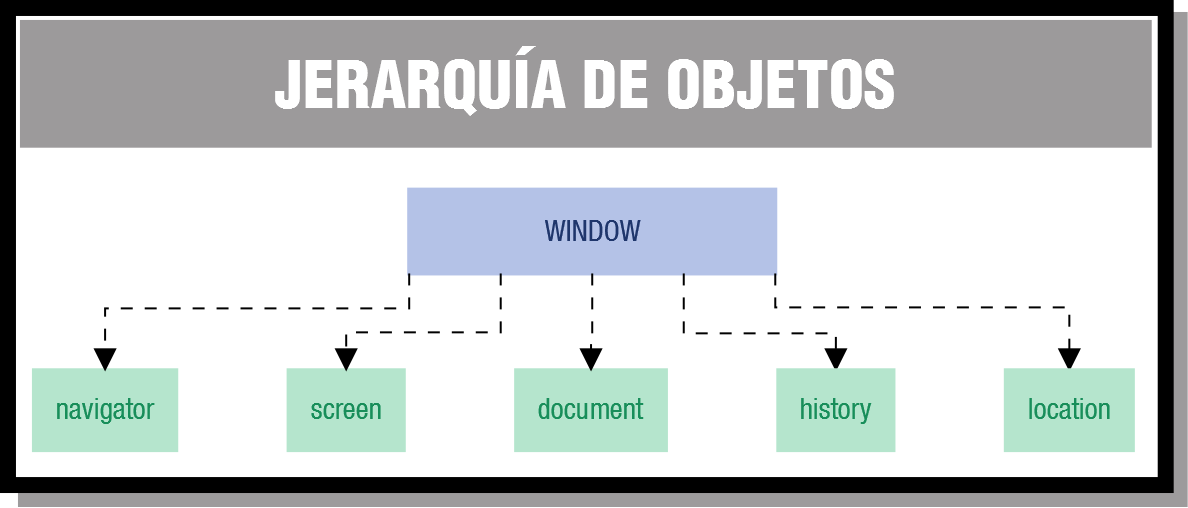
nombreObjeto.Propiedad  
  
La forma de acceder a un método de un objeto es la siguiente  
  
nombreObjeto.metodo ( parametros opcionales) ;

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.

En esta unidad no hablaremos del uso de **frames** ya que su uso está totalmente desaconsejado por motivos de accesibilidad y usabilidad, por lo que te recomendamos no usarlos.

# 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.nombreMetodo ( parametros)  
  
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.nombreMetodo ( parametros )  
  
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  
nombreMetodo ( parametro )   
  
  
  
  
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.

**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:  
  
let 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">

const inicializar = () => {

document.getElementById("crear-ventana").onclick=crearNueva;

document.getElementById("cerrar-ventana").onclick=cerrarNueva;

}

let nuevaVentana;

const crearNueva = () => {

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

}

const cerrarNueva = () => {

if (nuevaVentana) {

nuevaVentana.close(); nuevaVentana = null;

}

}

</script>

</head>

<body onLoad="inicializar()">

<h1>Abrimos y cerramos ventanas</h1>

<form>

<input type="button" id="crear-ventana" value="Crear Nueva Ventana">

<input type="button" id="cerrar-ventana" value="Cerrar Nueva Ventana">

</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.   
  
[tabla ]  
  
**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**.  
  
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**.  
  
[tabla ]  
  
1.3.- Objeto navigator.

[tabla ]  
  
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).  
  
[tabla]

# 2.- Objetos nativos en Javascript.

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**.

## 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.

Recuerda que de cómo definir las cadenas ya hemos hablado y que no se te olviden las plantillas de cadenas que te hacen muy fácil su composición en vez de estar concatenando que a veces resulta engorroso.

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:  
  
  
[tabla]  
  
**2.1.1.- Propiedades y métodos del objeto String.**

Para crear un objeto **String** lo podremos hacer de la ssiguientes formas:   
  
Es decir, cada vez que tengamos una cadena de texto, en realidad es un objeto **String** que tiene propiedades y métodos:   
  
[tabla]  
  
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 posee 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:   
  
[tabla]

# 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()**.  
  
[table]  
  
2.4.- Objeto Boolean.

El objeto **Boolean** se utiliza para convertir un valor no lógico, a un valor lógico (**true** o **false**).

[table]  
  
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**):

[Tabla]