3.A. Modelo de objetos predefinidos en JavaScript

1.- Objetos Javascript del navegador

1.3.- Objeto window.

En la jerarquía de objetos, tenemos en la parte superior los objetos navigator y window.

El objeto window es el contenedor principal de todo el contenido que se visualiza en una ventana o pestaña del 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] )
```

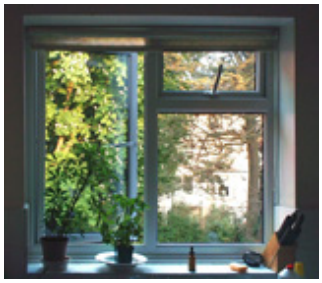
Podremos usar cualquiera de las dos referencias anteriores, pero intentaremos dejar la palabra reservada self, para scripts más complejos en los que tengamos múltiples marcos y ventanas.

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.



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>
```

Descarga del código del ejemplo. (0.01 MB)

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()**.

Método	Descripción
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.

Más información y ejemplos sobre el objeto Window.

Citas para pensar

> "Sólo cerrando las puertas detrás de uno se abren ventanas hacia el porvenir." *SAGAN, Françoise.*

Créditos de las imágenes

Imagen de ventana 1

Autoría: Dominic's pics.

Licencia: CC BY 2.0.

Imagen de ventana 2

Autoría: quinn.anya.

Licencia: CC BY-SA 2.0.