TEMA

Tema 2. Objetos nativos JS

Desarrollo de aplicaciones web

Desarrollo web en entorno cliente

Autor: Cristian Catalán

Linkia FP

Tema 2: Objetos nativos JS

¿Qué aprenderás?

- Utilizar los principales objetos nativos de JavaScript.
- Utilizar los principales objetos nativos del navegador.
- Crear cookies.
- Crear nuevas ventanas y comunicarse entre ellas.

¿Sabías que…?

- La última versión de ECMA Script fue publicada en junio de 2019
- En JavaScript NaN es del tipo Number
- NaN no es igual a NaN. Solo se puede comprobar si un valor es NaN con la función isNaN.
- Según JavaScript 0.1 + 0.3 = 0.30000000000000004

2.1. Objetos nativos JavaScript

JavaScript define de forma nativa un gran conjunto de objetos que pueden ser utilizados directamente sin tener importarlos. Siempre que queramos asignar a una variable una instancia de una clase de JavaScript utilizaremos la palabra clave *new* seguido del nombre de la clase y unos paréntesis. Por ejemplo, para declarar una variable del tipo *String* y una de tipo *Boolean* podremos hacer:

let miTexto = new String();

let sino = new Boolean();

No será necesario especificar el tipo de variable si directamente le asignamos un valor.

2.1.1. String

La clase *String* representa una cadena de texto.

Su propiedad principal es:

- lenght: retorna el número de caracteres que forman la cadena de texto

A continuación podemos ver una tabla con los principales métodos de *String* :

método	Uso	Descripción
charAt	`let str ="Hola que tal";` `str.charAt(1)` `// o`	Retorna la letra ubicada en la posición pasada como parámetro. En este caso retorna la letra “o”.
charCodeAt	`let str ="Hola que tal";` `str.charCodeAt(1)` `//111`	Retorna el código Unicode correspondiente con la letra en la posición pasada como parámetro. En este caso retorna 111.
fromCharCode	`let str ="Hola que tal";` `String.fromCharCode(111)` `//o`	Transforma el código Unicode pasado como parámetro a la letra correspondiente. En este caso una “o”.

indexOf	`let str ="Hola que tal";` `str.indexOf('o',0)` `//1`	En este caso indica en qué posición se encuentra la letra 'o' a partir de la posición 0 (es decir, el inicio del string). En este caso se encuentra en la posición 1.
substr	`let str ="Hola que tal";` `str.substr(1,3)` `//ola`	En este caso extrae 3 caracteres a partir de la posición 1. En éste caso retorna "ola".
substring	`let str ="Hola que tal";` `str.substring(1,3)` `//ol`	Retorna el texto entre la posición inicial (incluida) y la posición final (no incluida). En éste caso retorna "ol".
split	`let str ="Hola que tal";` `str.split("a")` `//["Hol", " que t", "l"]`	Corta la String por el carácter pasado por parámetro y con cada trozo genera un array de strings. En éste caso retorna un array de 3 strings: *["Hol", " que t", "l"]*
toLowerCase	`let str ="Hola que tal";` `str.toLowerCase();` `//hola que tal`	Retorna todas las letras de una string en minúsculas.
toLowerCase	`let str ="Hola que tal";` `str.toUpperCase()` `//HOLA QUE TAL`	Retorna todas las letras de una string en mayúsculas.

2.1.2. Number

En JavaSCript todos los números están en Formato binario de doble precisión de 64 bits IEEE 754 (son numero entre -(253 -1) and 253 -1). La clase Number permite contener variables de tipo numérico con o sin decimal y tres valores simbólicos: +Infinity, -Infinity, y NaN (not-a-number, no un número).

Sus atributos principales son:

atributo	Descripción
`Number.MAX_VALUE`	Retorna el número más largo que puede almacenar un variable del tipo Number
`Number.MIN_VALUE`	Retorna el número más pequeño que puede almacenar un variable del tipo Number
`Number.NEGATIVE_INFINITY`	Representa el infinito negativo
`Number.POITIVE_INFINITY`	Representa el infinito negativo
NaN	Representa un valor especial que se obtiene cuando se intenta operar matemáticamente con algún valor no numérico. Se puede comprobar si un valor es *NaN* con la función global *isNaN()*

Sus métodos principales son:

método	Uso	Descripción
isFinite	Number.isFinite(33) //true Number.isFinite(Infinity) //false Number.isFinite(0 / 0) //false	Retorna si el valor pasado por parámetro es finito o no.
isInteger()	Number.isInteger(33) //true Number.isInteger(0.5) //false Number.isInteger('33') //false	Retorna si el valor pasado por parámetro es entero o no.
toExponential ()	var num = 8.57839; var n = num.toExponential(); console.log(n); //8.57839e+0	Transforma un valor numérico a notación exponencial. En el siguiente ejemplo el resultado es : `8.57839e+0`
`toFixed()`	var num = 8.57839; var n = num.toFixed(2);	Corta los decimales redondeando. En el siguiente ejemplo el resultado es : `8.58`
toPrecision()	var num = 8.5; var n = num.toPrecision(4);	Corta o extiende el número añadiendo ceros en los decimales para que tenga la longitud indicada. En este caso el resultado es: 8.500

2.1.3. Date

La clase *Date* permite nos permite contener y manipular valores relacionados con fechas.

Date permite almacenar un año, mes, día, hora, minuto, segundo y milisegundo.

Vamos a ver las distintas formas de inicializar la clase *Date*:

Constructor	Descripción
`let d = new Date(0);`	Genera una variable del tipo Date que almacena la fecha pasados 0 segundos desde el 1 de Enero de 1970
`let d = new Date(2019,5,9,10,5,3,1);`	Genera una variable del tipo Date correspondiente con el año 2019, mes 5, día 9, hora 10, minuto 5, segundo 3 y milisegundo 1.
`let d = new Date(2019,5,9);`	Genera una variable del tipo Date correspondiente con el año 2019, mes 5, día 9, hora 0, minuto 0, segundo 0 y milisegundo 0.
`let d = new Date();`	Genera una variable del tipo Date que almacena la fecha actual del navegador en la que se ha ejecutado.

La clase *Date* tiene muchos métodos de los que vamos a destacar los principales:

método	Descripción
new Date(2019,5,9,10,5,3,1).**getTime**(); //1560067503001	Retorna el número de milisegundos transcurridos desde la referencia de tiempos (1 de Enero de 1970). En este caso: 1560067503001
new Date(2019,5,9,10,5,3,1).**getDate**(); //9	Retorna el número del día del mes.
new Date(2019,5,9,10,5,3,1).**getFullYear**(); // 2019	Retorna el año de la fecha como un número de 4 cifras.
new Date(2019,5,9,10,5,3,1). **getMonth** (); //5	Retorna el número del mes. En donde 0 es Enero.
new Date(2019,5,9,10,5,3,1).**getHours**(); //10	Retorna la hora almacenada.
new Date(2019,5,9,10,5,3,1). **getMinutes** (); //5	Retorna los minutos almacenados en la fecha.
new Date(2019,5,9,10,5,3,1). **getSeconds** (); //3	Retorna los segundos almacenados en la fecha.
new Date(2019,5,9,10,5,3,1). **getDay**(); //0	Retorna el número correspondiente al día de la semana correspondiente con la fecha, en donde el 0 es domingo y 6 es sábado.

2.1.4. Math

La clase *Math* nos ofrece un conjunto de constantes y métodos específicos para realizar operaciones matemáticas en JavaScript. Todos los métodos trigonométricos de *Math* trabajan en radiantes.

Sus atributos principales son:

atributo	Descripción
E	Retorna la constante de Euler (aproximadamente 2.718)
LN2	Retorna el logaritmo natural de 2 (aproximadamente 0.693)
LN10	Retorna el logaritmo natural de 10 (aproximadamente 2.302)
LOG2e	Retorna el logaritmo de E en base 2 (aproximadamente 1.442)
LOG10E	Retorna el logaritmo de E en base 10 (aproximadamente 0.434)
PI	Retorna el valor de Pi (aproximadamente 3.14)
SQRT1_2	Retorna la raíz cuadrada de 1/2 (aproximadamente 0.707)
SQRT2	Retorna la raíz cuadrada de 2 (aproximadamente 1.414)

Sus métodos principales son:

método	Descripción
Math.abs(x)	Retorna el valor absoluto el número pasado por parámetro.
Math.sin(x) Math.cos(x) Math.tan(x)	Retorna respectivamente el valor en radiantes del seno, coseno y tangente del número pasado por parámetro.
Math.asin(x) Math.acos(x) Math.atan(x)	Retorna respectivamente el valor en radiantes del arcoseno, arcocoseno y arcotangente del número pasado por parámetro.
Math.pow(x, y)	Retorna el resultado del primer parámetro elevado al segundo parámetro.
Math.sqrt(x)	Retorna la raíz cuadrada del número pasado por parámetro.
Math.exp(x)	Retorna el valor de E elevado al número pasado por parámetro.
Math.log(x)	Retorna el valor del logaritmo natural en base E del valor pasado por parámetro.
Math.max(x, y, z, …, n) Math.min(x, y, z, …, n)	Retorna respectivamente el valor máximo y mínimo entre todos los valores pasados por parámetro.
Math.trunc (x)	Retorna la parte entera del número pasado por parámetro sin hacer ningún tipo de redondeo.

Math.round(x)	Redondea un valor decimal hacia el entero más próximo. Por ejemplo: *Math.round(1.6); //es redondeado a 2* *Math.round(1.4); //es redondeado a 1*
Math.ceil(x)	Redondea un valor decimal hacia el entero más alto. Por ejemplo: *Math.round(1.6); //es redondeado a 2* *Math.round(1.4); //es redondeado a 2*
Math.floor(x)	Redondea un valor decimal hacia el entero más bajo. Por ejemplo: *Math.round(1.6); //es redondeado a 1* *Math.round(1.4); //es redondeado a 1*
Math.random()	Genera un valor decimal aleatorio entre 0 y 1. Se suele utilizar junto con floor para generar valores enteros aleatorios. Por ejemplo, podemos generar un número entero entre 1 y 10 con: *Math.floor((Math.random() * 10) + 1);*

2.2. Objetos nativos del Navegador

Hemos visto los principales objetos predefinidos de JavaScript. A continuación veremos los principales objetos predefinidos que nos permiten controlar características del navegador mediante JavaScript. Gracias a éstos objetos podremos manipular el HTML, crear nueva ventanas o modificar propiedades de las ventanas como su título o tamaños.

En este punto hay que recordar que el buen funcionamiento de estos métodos y propiedades va sujeto al navegador. Es decir, el navegador va a poder permitir o no que utilicemos los métodos y propiedades que ahora veremos. Por ello es posible que en muchas ocasiones alguna sentencia no funcione correctamente o directamente no funcione.

2.2.1. Navigator

Este objeto nos permite acceder a la información sobre el navegador en la que se está ejecutando el JavaScript. Es una información fácilmente manipulable por el usuario y el navegador. En muchas ocasiones ésta información es utilizada para adecuar el código JavaScript a utilizar ya que en algunos casos algunos intérpretes de JavaScript funcionan de forma distinta.

Sus atributos principales son:

atributo	Descripción
navigator.appCodeName navigator.appName navigator.product	Estas dos propiedades han dejado de ser útiles y se mantienen solamente por compatibilidad con versiones anteriores. Su valores siempre son respectivamente: “Mozilla” , “Netscape”, “Gecko”
navigator.appVersion	Retorna la versión del navegador. Por ejemplo: *(Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.110 Safari/537.36 Vivaldi/2.7.1628.30*

navigator.userAgent	Retorna la versión del navegador. Por ejemplo: *Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.110 Safari/537.36 Vivaldi/2.7.1628.30*
navigator.platform	Retorna información sobre el sistema operativo sobre el que se ejecuta el navegador. Por ejemplo: *"MacIntel", "Win32", "FreeBSD i386", "WebTV OS"*
navigator.cookieEnabled	Retorna *true* o *false* según si el usuario tiene habilitadas o no las cookies.
navigator.language	Retorna en base a la especificación BCP47 una string que identifica la lengua predefinida en el navegador. Por ejemplo: *“es”, “es-ES" ,"en", "en-US", "fr", "fr-FR"*
navigator.onLine	Retorna *true* o *false* según si el navegador tiene o no conexión a internet.

Sus métodos principales son:

método	Descripción
navigator.javaEnabled()	Retorna *true* o *false* para indicar si el navegador tiene habilitada la ejecución de Java o no.
navigator.vibrate(200); navigator.vibrate([100,30,50]);	Si el dispositivo y el navegador lo permiten, ejecuta una vibración que dura tantos milisegundos como los indicados por parámetro. Si por parámetro se pasa un arry de números se realizan tantas vibraciones como números se hayan indicado con una pausa entre ellos.

2.2.2. Location

El objeto *Location* nos ofrece información sobre la ubicación (URL) de la web en la que se ejecuta el JavaScript. El objeto *Window* y *Document* tienen un objeto *Location* asignado.

Sus atributos principales son:

Atributo	Descripción
location.href	Retorna la URL completa. Si se cambia ésta propiedad, automáticamente se navegará hacia la nueva dirección. Por ejemplo: *http://127.0.0.1:5500/ejemplos/ejemplo.html?param=1#hash*
location.origin	Retorna la forma canónica del origen de la URL. Por ejemplo: *http://127.0.0.1:5500*
location.pathname	Retorna la / inicial seguido por la ruta de la URL. Por ejemplo: */ejemplos/ejemplo.html*
location.protocol	Retorna el protocolo en la URL. Por ejemplo: *http:*
location.host	Retorna el nombre del dominio del servidor y el número de puerto en la URL. Por ejemplo: *127.0.0.1:5500*
location.hostname	Retorna el nombre del dominio del servidor en la URL. Por ejemplo: *127.0.0.1*
location.port	Retorna el número de puerto en la URL. Por ejemplo: *5500*
location.search	Retorna el listado de parámetros que contiene la URL. Por ejemplo: *¿param=1*

location.hash	Retorna la parte de la URL detrás del hashtag. Por ejemplo: *#hash*

Sus métodos principales son:

Método	Descripción
location.assign("https://linkiafp.es")	Carga el documento con la URL pasada como parámetro. En este caso cargaría la web ubicada en: https://linkiafp.es
location.replace("https://linkiafp.es");	Reemplaza el documento actual con la URL pasada como parámetro. A diferencia del método *assign()* , con *replace()* la página actual no va a ser guardada en History, por lo que el usuario no podrá usar el botón Atrás para navegar a esta.
location.reload(true)	Carga de nuevo el documento actual. El parámetro *true* obliga a que a cargue desde el servidor. El parámetro *false* obliga a que sea cargada desde la caché.

2.2.3. Screen

El objeto *Screen* nos permite acceder a información sobre la pantalla en la que se está visualizando la página web.

Sus atributos principales son:

Atributo	Descripción
screen.availHeight screen.availWidth	Retorna respectivamente la altura y anchura de la pantalla disponible. Tiene en cuenta los elementos permanentes o semipermanentes como puede ser la barra de tareas en Windows que quitan espacio a la visualización.
screen.height screen.width	Retorna la altura y anchura total de la pantalla. No tiene en cuenta los elementos permanentes o semipermanentes como puede ser la barra de tareas en Windows.
screen.colorDepth	Retorna el número de colores que puede representar la pantalla. En la mayoría de casos retorna 24 por compatibilidades.
screen.pixelDepth	Retorna la resolución de la pantalla expresada en bits por píxel. En la mayoría de casos retorna 24 por compatibilidades.
screen.orientation	Retorna un objeto con información sobre la orientación de la pantalla y si tiene algún evento asociado al hecho de girar la pantalla.

Su método principal es:

Método	Descripción
screen.lockOrientation; screen.unlockOrientation;	Bloquea o desbloquea respectivamente el giro de la pantalla.

2.2.4. History

El objeto *history* permite acceder al historial del navegador. Por seguridad en la mayoría de navegadores no es posible acceder al historia de todas las páginas que ha visitado el usuario, sino que se nos ofrecerá la posibilidad de navegar hacia la página anterior o página posterior. Por ello, *history* debería de ser pensado como una herramienta para facilitar que el usuario se desplace hacia delante y hacia tras en un mismo sitio web.

Su atributo principal es:

Atributo	Descripción
history.length	Retorna el número de URLs que se han visitado en una misma ventana del navegador.

Sus métodos principales son:

Método	Descripción
history.go(-3);	Carga la URL especificada por el índice pasado como parámetro. En éste caso carga la 3ª página anterior a la visitada.
history.back();	Carga la URL del recurso anterior al actual. Es equivalente a escribir: history.go(-1);
history.forward();	Carga la URL del recurso posterior al actual. Es equivalente a escribir: history.go(1);

2.2.4. Document

El objeto "*document"* hace referencia a la estructura de datos que representa la página web. A través del objeto "*document*" podremos acceder y modificar los elementos que forman la página web así como crear de nuevos o borrar. Es sin duda uno de los objetos más utilizados en JavaScript.

Sus atributos principales son:

Atributo	Descripción
document.title	Permite obtener o modificar el título del documento. Es decir, el texto que aparece en la pestaña del navegador.
document.activeElement	Retorna una referencia al elemento HTML que tenga el foco, es decir, que este seleccionado o con el cursor. Típicamente será un input o un elemento de formulario apunto para escribir en él.
document.URL	Retorna la URL completa del documento HTML.
document.referrer	Retorna la URL que nos ha llevado hasta el documento actual. Si el usuario accede a él directamente escribiendo la dirección, esta propiedad retornará una cadena de texto vacía.
document.lastModified	Retorna un Date con la fecha de la última vez que se modificó el documento.
document.cookie	Permite acceder y modificar las cookies asociadas al sitio web
document.forms document.images document.links document.scripts	Retorna respectivamente un array formado por las referencias a todos los formularios, imágenes, links y scripts que forman la página web.

Cuando veamos el acceso al DOM veremos más método asociados a *document*, por ahora sus métodos principales son:

Método	Descripción
document.getSelection();	Retorna el texto que el usuario haya seleccionado en la web.
document.hasFocus();	Retorna true o false según si el usuario ha situado el foco en algún sitio del documento. Es decir, si ha clicado en algún elemento del documento.
document.getElementById("idElemento");	Retorna una referencia a un elemento del documento accediendo a él a partir de su atributo Id pasado por parámetro.
document.querySelector("#idElemento");	Retorna una referencia al primer elemento que cumpla el selector CSS pasado por parámetro.
document.createElement("DIV");	Crea un nuevo elemento HTML y retorna una referencia al elemento creado.

2.2.5.1. Introducción a las Cookies

A partir de *document* podemos acceder a las cookies asociadas a un mismo sitio web. Las cookies son variables vinculadas a un sitio web que viajan entre el servidor y el navegador manteniendo su información. Las cookies nos sirven para poder pasar información entre JavaScripts de diferentes páginas web dentro del mismo sitio web. Si por ejemplo en la primera página web el usuario me ha indicado su nombre, si me guardo su nombre en una cookie cuando visite otra página de mi sitio web podré recuperar ese nombre.

Por seguridad no vamos a poder acceder a las cookies de otros sitios web.

Las cookies son cadenas de texto formado por como mínimo un nombre de propiedad y su valor. También pueden tener asociado una fecha de expiración para que el navegador la borre en superarla. Si no indicamos una fecha el navegador la borrará cuando le parezca oportuno.

Podemos guardar un valor en una cookie con la sintaxis:

document.cookie = "nombreCookie=Valor propiedad";

En donde “nombreCookie” es el nombre con el que queremos identificar el valor que almacenaremos.

Para añadir otra cookie basta con escribir de nuevo:

document.cookie = "otraCookie=Valor de otra propiedad";

Si el nombre de la cookie ya existía, entonces se actualizará el valor de la cookie.

Para acceder a las cookies almacenadas basta con escribir:

document.cookie;

Lo que retornará algo parecido a :

"nombreCookie=Valor propiedad;otraCookie=Valor de otra propiedad";

Y es que cada vez que creemos una cookie se añadirá en una única String detrás de las cookies que ya existan.

Acceder al valor de una cookie es algo más complejo, pues por desgracia directamente JavaSCript no ofrece un modo de acceder al valor de una única cookie. Más a delante veremos métodos para acceder a una cookie en concreto.

2.2.6. Window

El objeto window es una referencia a la pestaña en la que se está mostrando un documento y todo su contenido. El objeto window es un objeto global, ya que no es necesario especificarlo para acceder a sus atributos o métodos sino que estos pueden ser llamados directamente. Además el objeto window contiene las referencias a todos los objetos nativos del navegador, pudiendo acceder a ellos a través del objeto window.

Sus atributos principales son:

Atributo	Descripción
window.location window.history window.screen window.navigator window.document	Retorna la correspondiente referencia a los objetos vistos anteriormente correspondientes con *location*, *history*, *screen*, *navigator* y *location*
window.innerHeight window.innerWidth	Retorna respectivamente la altura y anchura utilizable de la ventana. Es decir, la altura y anchura de la página web contando la scrollbar pero sin contar los bordes ni menús del navegador.
window.outerHeight window.outerWidth	Retorna respectivamente la altura y anchura del navegador. Incluyendo los bordes y menús del navegador.
window.screenY window.screenTop	Ambas propiedades retornan la distancia desde el borde superior de la ventana hasta el borde superior de la pantalla.
window.screenX window.screenLeft	Ambas propiedades retornan la distancia desde el borde izquierdo de la ventana el borde izquierdo de la pantalla.
window.scrollbars.visible	Retorna un valor true o false según si las barras de desplazamiento (Scrollbars) están habilitadas o no en la ventana.
window.scrollX window.scrollY	Retornan respectivamente el número de píxeles que el documento mostrado en la ventana ha sido desplazado mediante un scroll horizontal y verticalmente.
window.sessionStorage	Retorna una referencia a la “sesión Storage”, un espacio de memoria en la que un mismo sitio web puede almacenar información para acceder a él posteriormente.
window.toolbar	Retorna un valor *true* o *false* que nos indica si la ventana se está mostrando con una barra de herramientas o no.

window.status	Retorna el texto que muestra la barra de estado del navegador.
window.name	Retorna el nombre de la ventana. Esto no hace referencia al título, sino al nombre que se puede dar a una ventana para indicar que se quiere abrir un contenido en ella.
window.opener	Retorna una referencia a la ventana que haya abierto la ventana actual. Si la ventana actual ha sido abierta por el usuario entonces retorna null.

Sus métodos principales son:

Método	Descripción
window.alert("mensaje"); window.prompt("mensaje"); window.confirm("mensaje");	Estos tres métodos generan un cuadro de diálogo emergente con el mensaje pasado por parámetro. Éste cuadro de diálogo va a detener la carga de la web y la ejecución del JavaScript hasta que el usuario lo cierre clicando en alguna de sus opciones. Muy comúnmente éstos tres métodos son bloqueados por los navegadores. El método *alert()* solo muestra el texto y un botón ACEPTAR que cierra el diálogo. El método *confirm*() muestra el texto, el botón ACEPTAR y otro botón CANCELAR. Retorna *true* si el usuario ha aceptado y *false* en cualquier otro caso. El método *prompt()* muestra el texto, el botón ACEPTAR ,el otro botón CANCELAR y una caja de texto para que el usuario escriba en ella. Si el usuario cica en ACEPTAR retorna el texto escrito, en cualquier otro caso retorna null.
window.print();	Abre el cuadro de diálogo para imprimir la página.

window.close();	Cierra la ventana actual siempre que haya sido abierta por un JavaScript.
window.resizeTo(300,400);	Redimensiona la ventana actual hasta que tenga la anchura y altura pasadas por parámetro en píxeles. En este ejemplo una anchura de 300px y altura de 400px.
window.scrollBy(20,50); window.scrollBy({ top:20 ,left:50 behavior:'smooth', });	Si recibe dos parámetros, desplaza el contenido del documento hacia la izquierda y la derecha tantos píxeles como indicados en los parámetros respectivamente. Si recibe un único parámetro en notación JSON, desplaza el contenido tantos píxeles como los indicados en la notación JSON con el comportamiento indicado.
window.scrollTo(20,50); window.scrollTo({ top:20 ,left:50 behavior:'smooth', });	El comportamiento es parecido a scrollBy, pero en ésta ocasión se desplaza el documento hasta situarlo en la posición indicada con píxeles en sus parámetros.
window.setInterval(function(){ //codigo a repetir } ,1500);	Permite definir un código que se va a ejecutar cada X tiempo. Recibe dos parámetros, en el primero podemos definir una función con el código a ir repitiendo y en segundo parámetro vamos a indicar cuantos milisegundos se esperará a volver a ejecutar el código. Retorna una referencia al *interval* que se puede utilizar con clearInterval() para detener el intervalo.
window.setTimeout(function(){ //codigo a repetir } ,1500);	Permite definir un código que se va a ejecutar pasado X tiempo. Recibe dos parámetros, en el primero podemos definir una función con el código a ejecutar y como segundo parámetro vamos a indicar cuantos milisegundos se esperará a ejecutar el código. Al ser creado retorna una referencia al Timeout que se puede utilizar con clearTimeout() para detener el *timeout* antes de que se haya ejecutado.

window.clearInterval(ref);	Si recibe como parámetro una referencia válida a un *interval*, lo detiene.
window.clearTimeout(ref);	Si recibe como parámetro una referencia válida a un *timeout*, lo detiene.
window.open(url, nombre, opciones);	Permite abrir una nueva ventana para que cargue el contenido indicado en el parámetro *url*. La nueva ventana se abrirá con el nombre y las opciones pasadas por parámetro.

2.2.6.1. **Window.open**

Como hemos visto window.open(url, nombre, opciones); permite abrir una nueva ventana. Vamos a ver más detenidamente como funciona ésta función y cómo podemos interaccionar desde JavaScript con la nueva ventana abierta.

Los tres parámetros de *window.open()* son:

- **URL de la ventana**. Si indicamos una dirección sin *http://* estaremos indicando una ruta relativa des de la ventana actual. Si indicamos una ruta fuera de nuestro dominio, entonces una vez abierta la página no podremos acceder a ella desde nuestro JavaScript por razones de seguridad.
- **Nombre del documento**: este nombre prácticamente solo tiene utilidad si en nuestra página web utilizamos *iframes*. Un *iframe* es un espacio del documento HTML en donde podemos insertar otra página web, y es a través del nombre que podríamos hacer referencia al contenido dentro de uno u otro *iframe*.
- **Opciones**: Con las opciones podremos especificar como queremos que se muestre la ventana. En muchos casos los navegadores no harán caso de estas opciones. Estas opciones las escribiremos como una string indicando el conjunto de *opción=valor* separado por comas. Por ejemplo:

  window.open("masinfo.html","","scrollbars=no,width=200");

Las principales opciones que podemos especificar son:

opcion	Descripción
height=500	altura de 500px
width=100	anchura de 100px
left=50,	situado 50 pixeles del borde izquierdo de la pantalla
top:40	situado 40 pixeles del borde superior de la pantalla
menubar:yes menubar:no	Si queremos que se muestre u oculte la barra del menú del navegador
status:yes status:no	Si queremos que se muestre u oculte la barra de estado del navegador

La función *window.open()* retorna una referencia al objeto *window* de la ventana abierta. A partir de esta referencia podemos acceder a todas las propiedad del objeto *window.* Pero hay que tener presente que hasta que no se haya cargado la nueva web no se podrá acceder a sus elementos HTML.
En el siguiente ejemplo se muestra un código JavaScript que abre una nueva ventana con una anchura de 500px y cargando dentro suyo el archivo HTML de nombre "masinfo.html". Posteriormente muestra su altura interior por consola y finalmente pasado un segundo redimensiona la nueva ventana para que tenga una anchura de 300px y altura de 400px.

```
let winOpen = window.open("masinfo.html","","scrollbars=yes,width=500");
console.log( winOpen.innerHeight );
window.setTimeout(function(){
    winOpen.resizeTo(300,400) ;
},1000);
```

Ejemplo en el que se abre una nueva ventana y se accede a ella para modificar sus propiedades

Desde la ventana creada también podemos acceder mediante JavaScript a la ventana padre con la referencia: *window.opener*.

En el siguiente ejemplo accedemos a la ventana padre, modificamos su título y la cerramos pasados 3 segundos.

```
window.opener.document.title = "Te voy a cerrar ";
window.setTimeout(function () {
    window.opener.close();
}, 3000);
```

Ejemplo de acceso a la window de la ventana padre

En el siguiente video podemos ver como crear ventanas, comunicarnos entre ellas y depurar errores derivados en JavaScript.

Video: Creación de ventanas y depuración de errores

Recursos y enlaces

- Web oficial de ECMA con la documentación sobre la especificación ECMASCript 6

- En la web de MDN web docs podéis encontrar una muy buena documentación de JavaScript (parcialmente traducida): https://developer.mozilla.org/en-US/docs/Web/JavaScript

- En W3Schools podéis encontrar una documentación muy amigable y bastante amplia sobre JavaScript: https://www.w3schools.com/jsref/default.asp

Enlace a la especificación BCP 47 sobre códigos representativos de lenguas

Conceptos clave

- **Objetos nativos JavaScript del navegador**: objetos predefinidos que nos ofrece JavaScript para interactuar con la pantalla, el navegador y el documento HTML.
- **Number:** es el único tipo de datos que define JavaScript
- **JS**: es el acrónimo de JavaScript
- **ES6**: es el acrónimo de ECMAScript 6. La especificación en la que se basa JavaScript.

Test de autoevaluación

¿Qué objeto nativo nos permite redireccionarnos a otra web?

a) Document
b) Por seguridad no podemos navegar a otra página web.
c) Navigator
d) Location

¿Qué objeto nativo JavaScript del Navegador nos permite abrir otra ventana?

a) Window
b) Screen
c) Navigator
d) Document

¿Cómo podemos obtener la fecha y hora del navegador?

a) new Date()
b) getDate()
c) date.getDate();
d) new Date(now);Window

Ponlo en práctica

Actividad 1

Programa un JavaScript para que al cargar la web se genere una tabla que indique si el navegador soporta cookies, el "User Agent" del navegador, el valor del número PI, la raíz cuadrada de 7.

Actividad 2

Crea una aplicación web en donde el usuario pueda buscar qué letra está situada en una posición en concreto. La palabra y la posición se han de indicar en un input.

Actividad 3

Crea una aplicación web con dos botones. El primero pida un número al usuario y lo guarde en una variable global.
El segundo abra una nueva ventana y pasado un segundo muestre el número.
En la nueva ventana añade un botón que al clicarlo muestre en la ventana padre el mismo número.

SOLUCIONARIOS

Test de autoevaluación

¿Qué objeto nativo nos permite redireccionarnos a otra web?

a) Document
b) Por seguridad no podemos navegar a otra página web.
c) Navigator
d) Location

¿Qué objeto nativo JavaScript del Navegador nos permite abrir otra ventana?

a) Window
b) Screen
c) Navigator
d) Document

¿Cómo podemos obtener la fecha y hora del navegador?

a) new Date()
b) getDate()
c) date.getDate();
d) new Date(now);Window

Ponlo en práctica

Actividad 1

Programa un JavaScript para que al cargar la web se genere una tabla que indique si el navegador soporta cookies, el "User Agent" del navegador, el valor del número PI, la raíz cuadrada de 7.

Los solucionarios están disponibles en la versión interactiva del aula.

Actividad 2

Crea una aplicación web en donde el usuario pueda buscar qué letra está situada en una posición en concreto. La palabra y la posición se han de indicar en un input.

Los solucionarios están disponibles en la versión interactiva del aula.

Actividad 3

Crea una aplicación web con dos botones. El primero pida un número al usuario y lo guarde en una variable global.
El segundo abra una nueva ventana y pasado un segundo muestre el número.
En la nueva ventana añade un botón que al clicarlo muestre en la ventana padre el mismo número.

Los solucionarios están disponibles en la versión interactiva del aula.