

# UT3. Modelo de Objetos predefinidos en JavaScript

## 1)Objetos de más alto nivel en JavaScript

- 1)Objeto window
- 2)Objeto location
- 3)Objeto navigator
- 4)Objeto document

## 2)Objetos Nativos en JavaScript

- 1)Objeto String.
- 2)Objeto Math.
- 3)Objeto Number.
- 4)Objeto Boolean.
- 5)Objeto Date.

# 1. Objetos de más alto nivel en JavaScript .DOM

- 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.
- Define la **estructura lógica** de los documentos, y el modo en el que se acceden y se manipulan.

# 1. Objetos de más alto nivel en JavaScript (II).DOM

Algunas de las operaciones más comunes para las cuales se diseñó JavaScript son:

- Abrir una nueva ventana en el navegador
- Escribir un texto en un document
- Redirigir un navegador a otra ubicación
- Validar los datos de un formulario
- Etc

# 1. Objetos de más alto nivel en JavaScript (IV)

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

- Para acceder a una **propiedad** de un objeto

**nombreobjeto.propiedad**

- La forma de acceder a un **método** de un objeto:

**nombreobjeto.metodo([parámetros opcionales] )**

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

# 1. Objetos de más alto nivel en JavaScript (V)

- Objetos de alto nivel que vamos a ver:

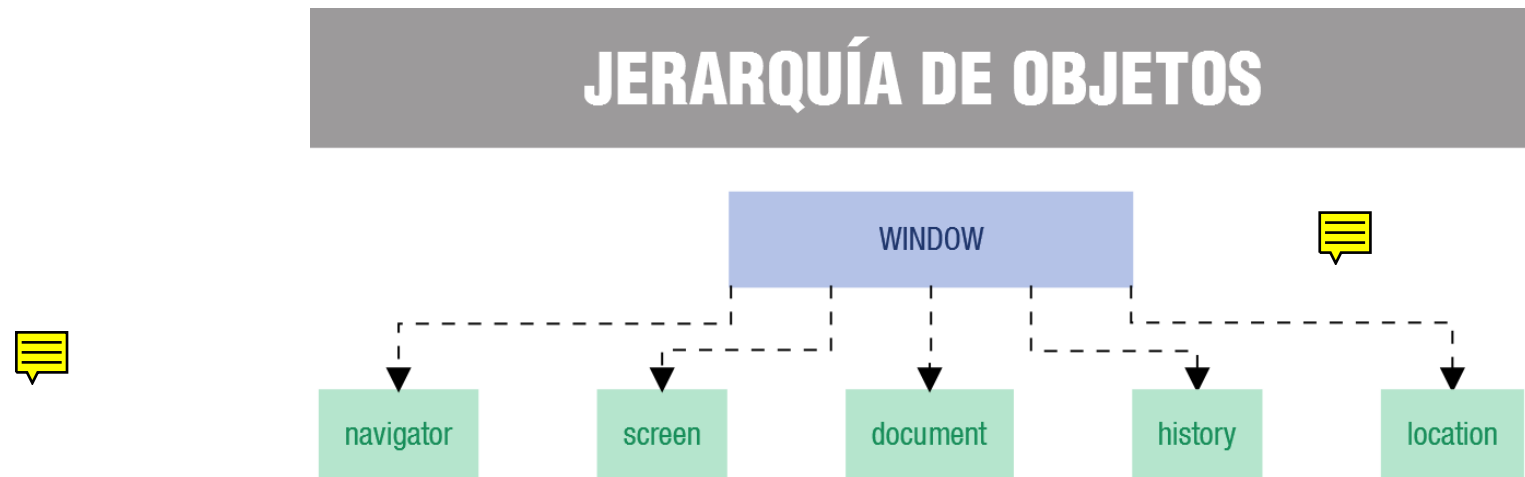
**window , location , navigator y document**

Se encuentran frecuentemente en las aplicaciones de JavaScript.


La mayoría de los navegadores web modernos han implementado la mayoría los mismos métodos y propiedades para la interacción JavaScript

# 1. Objetos de más alto nivel en JavaScript (VI)

- Gráfico del modelo de objetos de alto nivel, para todos los navegadores que permitan usar JavaScript:



# 1.1 Objeto window

- Situado en la parte superior de la jerarquía de objetos.
- Es el contenedor principal de todo el contenido que se visualiza en el navegador.
- El objeto **window** hace referencia a la ventana actual.
- 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. 

# 1.1 Objeto window

Campo de **influencia de Objeto window**:

- Contenido de dicho objeto (donde se cargarán los documentos)
- Dimensiones de la ventana
- Todo lo que rodea al área de contenido: las barras de desplazamiento, barra de herramientas, barra de estado, etc.




# 1.1 Objeto window

- El objeto **document** será el que contendrá toda la jerarquía de objetos que tengamos dentro de nuestra página HTML.

# 1.1 Objeto window

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** . 
- En los navegadores con pestañas, cada una contiene su propio objeto window.
- Esto significa que **el objeto window no se comparte entre diferentes pestañas de la misma ventana** del navegador

# 1.1 Objeto window

## Acceso a propiedades y métodos

Hay varias formas de acceder, la forma que se elija dependiendo más de nuestro estilo, que de requerimientos sintácticos.


**1.window.nombrePropiedad**  
**window.nombreMétodo( [parámetros] )**


Los parámetros van entre corchetes, indicando que son opcionales y que dependerán del método al que estemos llamando.

# 1.1 Objeto window

## Acceso a propiedades y métodos

2. **self.nombrePropiedad**  
**self.nombreMétodo( [parámetros]**

Se puede un objeto window desde el propio documento contenido en esa ventana. 

Mejor dejar la palabra reservada **self** para scripts más complejos en los que tengamos múltiples ventanas. 

# 1.1 Objeto window.

## Acceso a propiedades y métodos

**3. nombrePropiedad  
nombreMétodo( [parámetros]**

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

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

# 1.1 Objeto window.

## 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.
- Sin embargo, un script que esté ejecutándose en una de las ventanas principales del navegador, si que podrá crear o abrir nuevas **sub-ventanas**.

# 1.1 Objeto window

## Gestión de ventanas

Métodos para abrir y cerrar ventanas

**window.open()** 

- método que genera una nueva ventana
- Hasta 3 parámetros, que definen las características de la nueva ventana:

- la URL del documento a abrir
  - el nombre de esa ventana
  - su apariencia física (tamaño, color,etc.).

**window.close()**

# 1.1 Objeto window.

## Gestión de ventanas

- Para manejar **sub-ventanas**: hacer asignación a una **variable**.
- Con esta asignación podremos a lo largo de nuestro código, referenciar a la nueva ventana desde el script original de la ventana principal



# 1.1 Objeto window.

## Gestión de ventanas

Ejemplo abrir y cerrar sub-ventanas:

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

```
subVentana.close()
```

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

# 1.5.Comunicación entre múltiples ventanas

Objeto `window`: Método `open` y propiedad `opener`

Propiedad `opener` del objeto `window`: contiene la referencia a la ventana, que ha abierto ese objeto `window` empleando el método `open()` .

Para la ventana principal el valor de `opener` será `null`

- Debido a que `opener` es una referencia válida a la ventana padre que abrió las otras, podemos emplearlo para iniciar la referencia a objetos de la ventana original (padre) desde la ventana hija. Es semejante a lo que vimos con frames , pero en este caso es entre ventanas independientes del navegador.

## 1.5 Comunicación entre ventanas

Código HTML de padre.html:

**<p id="parPadre"></p>**

Código HTML de hijo1.html:

**<p id="parHijo1"></p>**

Código HTML de hijo2.html:

**<p id="parHijo2"></p>**

Ventana padre.html crea dos subventanas:

//utilizar var, let restringe la visibilidad del objeto al padre

**var v1 = window.open("hijo1.html","hijo1");**

**var v2 = window.open("hijo2.html","hijo2");**

# 1.5. Comunicación entre ventanas

Ventana padre.html modifica un elemento de hijo1.html:

**v1.parHijo1.innerHTML="Esto viene del padre"**

Ventana hijo1.html modifica un elemento de padre.html:

 **opener.parPadre.innerHTML="Esto viene del hijo1"**

Ventana hijo1.html modifica un elemento de hijo2.html:

**opener.v2.parHijo2.innerHTML="Esto viene del hijo1"** 

Para que este código funcione es necesario ejecutarlo desde un mismo servidor web, sino dará el error de directiva de seguridad **cross\_origin**, ya que ejecutándolas con **file://** desde el navegador considera que el origen de las 3 páginas es diferente, por lo que salta esta directiva de seguridad que no permite que una página modifique el elemento de la otra página.

# 1.5. Comunicación entre múltiples ventanas. Ejemplo

Ejemplo: página 11.

NOTA: Si no se abren las ventanas a lo mejor tienes que desactivar el bloqueador de pop-ups y volver a probar.

Si no traslada el texto de una página a otra prueba a ejecutar desde el editor Brackets directamente.

# **1.1 Objeto window.**

## **Propiedades y Métodos**

- El objeto `window` representa una ventana abierta en un navegador.

# 1.1 Objeto window.

## Propiedades y Métodos



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

# 1.1 Objeto window.

## Propiedades y Métodos

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

`alert()`



`confirm()`



`prompt()`





# 1.1 Objeto window.

## Ejercicio

**Ejercicio** página 20.



Otros métodos objeto Window:

MoveTo



MoveBy

SetTimeout



Indica qué realizan dichos métodos y el script propuesto.

NOTA: Algunos métodos, como `window.resizeTo` y `window.resizeBy` se aplican sobre toda la ventana del navegador y no sobre una pestaña específica a la que pertenece el objeto

# 1.2 Objeto location. (I)

## Propiedades y Métodos

- El objeto **location** contiene información referente a la URL actual.  
También nos proporciona detalles del servidor que nos la ha servido

Sintaxis de una URL :

*protocolo://maquina\_host[:puerto]/camino\_al\_recurso*

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



# 1.2 Objeto location.(II)

## Propiedades y Métodos

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.



# 1.2 Objeto location. (III)

## Propiedades y Métodos



### Métodos del objeto Location

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

# 1.2 Objeto location. (IV)

## Ejemplos

**[https://www.w3schools.com/js/tryit.asp?filename=tryjs\\_loc\\_href](https://www.w3schools.com/js/tryit.asp?filename=tryjs_loc_href)**

```
demo.innerHTML +=  
"The URL completa es: " + window.location.href+"<br>" +  
"The sitio web is: " + window.location.hostname+"<br>" +  
"The sitio web y el puerto is: " +  
window.location.host+"<br>" +  
"The ruta del fichero is: " +  
window.location.pathname+"<br>" +  
"El puerto es: " + window.location.port+"<br>" +  
"El protocolo es: " + window.location.protocol+"<br>" +  
"La cadena de busqueda es : " +  
window.location.search+"<br>" +  
"El hash es: " + window.location.hash+"<br>" ;
```

# 1.2 Objeto location. (IV)

## Ejemplos

Ejemplo de método `location.assign()`;

```
https://www.w3schools.com/js/tryit.asp?  
filename=tryjs_loc_assign
```

# 1.2 Objeto location. (IV)

## Ejercicio

**Ejercicio** página 21.

Indica qué componentes de una URL está mostrando el script propuesto.

# 1.2 Objeto location. (IV)

## Ejemplos

Ver ejemplos de los métodos `assign()`, `reload()` y `replace()` en:

[https://www.w3schools.com/jsref/obj\\_location.asp](https://www.w3schools.com/jsref/obj_location.asp)



# 1.3 Objeto navigator.



- Este objeto contiene información sobre el navegador que estamos utilizando cuando abrimos una URL o un documento local.
  - Tipo de navegador.
  - Versión del navegador.
  - Sistema operativo
- Se suele utilizar para obtener este tipo de información, y en base al resultado, tomar una decisión sobre qué tipo de código ejecutar.

# 1.3 Objeto navigator. (II)

## Propiedades y Métodos

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

# 1.3 Objeto navigator. (III)

## Propiedades y Métodos

Métodos del objeto Navigator	
Método	Descripción
<code>javaEnabled()</code>	Devuelve true si el cliente permite la utilización de Java, en caso contrario, devuelve false.

# 1.4 Objeto document. (I)

- El objeto document representa tu página web
- Es el elemento raíz del Document Object Model (DOM)
- Para acceder a cualquier elemento de una página HTML, siempre hay que acceder al objeto document.
- Desde aquí podemos
  - Encontrar elementos HTML
  - Cambiar elementos HTML
  - Crear elementos HTML
  - Eliminar elementos HTML
  - Añadir manejadores de eventos

## 1.4 Objeto document. (II)

- 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`
- Se accede :
  - a través de la propiedad `window.document` o
  - directamente **`document`** (ya que podemos omitir la referencia a la `window` actual)

# 1.4 Objeto document. (III)

## Propiedad `links[]`



- `links[]` es una colección de elementos `<a>` y `<area>` con un atributo `href` que hay en el objeto document.



Más información:

[https://www.w3schools.com/jsref/coll\\_doc\\_links.asp](https://www.w3schools.com/jsref/coll_doc_links.asp)

- Ejercicio: crea una página HTML con 2 enlaces. Desde JavaScript informa del número de enlaces que tiene tu documento utilizando la propiedad `document.links[]`

# 1.4 Objeto document. (III)

## Propiedad `forms[]`


- `forms[]` es una colección de todos los elementos `<form>` que hay en el objeto document.

Más información:

[https://www.w3schools.com/jsref/coll\\_doc\\_forms.asp](https://www.w3schools.com/jsref/coll_doc_forms.asp)

# 1.4 Objeto document. (III)

## Propiedad `images[]`

- `images[]` es una colección de todos los elementos `<img>` que hay en el objeto document. 

Más información:

[https://www.w3schools.com/jsref/coll\\_doc\\_images.asp](https://www.w3schools.com/jsref/coll_doc_images.asp)

Ejercicio:

Crea un documento HTML con 3 imágenes. Desde Javascript y a través de la propiedad `document.images[]` cambia el borde de todas las imágenes, asignando a la propiedad `border` de cada imagen el valor 8



# 1.4 Objeto document. (IV)

## Propiedades y Métodos

Propiedades del objeto Document	
Propiedad	Descripción
<code>cookie</code>	Devuelve todos los nombres/valores de las cookies en el documento.
<code>domain</code>	Cadena que contiene el nombre de dominio del servidor que cargó el documento.
<code>lastModified</code>	Devuelve la fecha y hora de la última modificación del documento
<code>readyState</code>	Devuelve el estado de carga del documento actual
<code>referrer</code>	Cadena que contiene la URL del documento desde el cuál llegamos al documento actual.
<code>title</code>	Devuelve o ajusta el título del documento.
<code>URL</code>	Devuelve la URL completa del documento.

- La propiedad `domain` es igual a `location.hostname`
- La propiedad `lastModified` indica la fecha del fichero que guarda el documento HTML. Si se salva el documento HTML y se vuelve a ejecutar se verá dicha fecha cambiada.
- La propiedad `referrer` devuelve vacío si el documento no ha sido abierto con un enlace.
- La propiedad `title` es el contenido del elemento HTML `<title>`
- La propiedad `URL` es igual a `location.href`



# 1.4 Objeto document. (V)

## Propiedades y Métodos

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

- Métodos `open()` se utiliza para abrir el documento para su escritura con `write()/writeln()`, y después se cierra con método `close()`
- El método `write()/writeln()` abre automáticamente el documento si no está abierto.

# 1.4 Objeto document. (V)

## Propiedades y Métodos

- **write() / writeln()** solo funcionan si la página no está completamente cargada cuando se ejecuta. Si la página ya está creada se va a borrar. Exactamente se borra cuando se ejecuta `document.open()`
- Ejemplo del problema:

```
<!DOCTYPE html>
<html>
<body>
<p>Comienzo</p>
<button onclick="loBorra()">prueba</button>
<script>
function loBorra () {
    document.write("He borrado todo");
}
</script>
<p>Fin</p>
</body>
</html>
```

# 1.4 Objeto document. (VI)

## Ejemplo

Crear elementos HTML desde cero



```
<script>  
var eleLogo= document.createElement("img");  
eleLogo.src = "https://www.madrid.org/img/logo.png";  
eleLogo.alt = "Logo madrid.org";  
document.body.appendChild(eleLogo);  
</script>
```

## 2. Objetos nativos en JavaScript

- JavaScript proporciona una serie de objetos definidos nativamente que no dependen del navegador (son objetos que JavaScript nos da listos para su utilización en nuestra aplicación)

- Los objetos que se verán son:

**String , Math , Number , Boolean y Date**

- NOTA: Si quisieramos crear un objeto nuevo → se utiliza la palabra clave new.

```
var miObjeto = new Object();
```

## 2.1. Objeto String (I)

- 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.
- **Concatenar el contenido de una variable dentro de una cadena:** con **+**

```
nombreEquipo = prompt("Introduce el nombre de tu equipo  
favorito:", "");
```

```
var mensaje= "El " + nombreEquipo + " ha sido el campeón de la  
Copa del Rey!";
```

```
alert(mensaje);
```

## 2.1.Objeto String (III)

Caracteres de escape y especiales 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.



Ejemplo: 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 \ " ,

```
var equipo = \"Real Madrid\";
alert (equipo);
```

## 2.1.Objeto String (IV)

Formas de crear un objeto String :

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

```
var miCadena = "texto de la cadena";
```

Cada vez que tengamos una cadena de texto, en realidad es un objeto String que tiene **propiedades y métodos**:

```
cadena.propiedad;
```

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



# 2.1.Objeto String (V)

Propiedades del objeto String	
Propiedad	Descripción
<code>length</code>	Contiene la longitud de una cadena.



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



## 2.1.Objeto String (VI)

### 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/>");
```

Para obtener más información :

[http://www.w3schools.com/js/js\\_obj\\_string.asp](http://www.w3schools.com/js/js_obj_string.asp)

[https://www.w3schools.com/js/js\\_string\\_methods.asp](https://www.w3schools.com/js/js_string_methods.asp)

## 2.1.Objeto String (VI)

- Ejemplo para recorrer todas las ocurrencias dentro de una cadena desde el principio hasta el final con `indexOf()` :

```
let text = "Hello world, welcome to the universe.";
var lugar= 0;
demo.innerHTML+="text vale: "+text+"<br>";
demo.innerHTML+="<BR>La e desde el comienzo aparece en pos:<br>";

while (lugar <text.length&&lugar!=-1){
    lugar= text.indexOf("e",lugar);
    // lugar se pone respecto a la posición del caracter en el
    // string contando desde el principio
    if (lugar!=-1){
        demo.innerHTML += lugar+"<br>";
        lugar++;
    }
}
```



## 2.1.Objeto String (VI)

- Ejemplo para recorrer todas las ocurrencias dentro de una cadena desde el final hasta el principio con de `lastIndexOf()` :

```
let text = "Hello world, welcome to the universe.";
var lugar= 0;
demo.innerHTML+="text vale: "+text+"<br>";
demo.innerHTML+="<BR>La e desde el final aparece en pos:<br>";

while (lugar >0&&lugar!=-1){
    lugar= text.lastIndexOf("e",lugar);
    // lugar se pone respecto a la posición del caracter en el
    // string contando desde el principio como en indexOf

    if (lugar!=-1){
        demo.innerHTML += lugar+"<br>";
        lugar--;
    }
}
```

- Ejemplo de `trim()` : 

```
let text="    mi movil es Android    ";
text.trim() → //devuelve "mi movil es Android"
```

## 2.1.Objeto String (VI)

- Los métodos `replace`, `match` y `search` funcionan también con expresiones regulares. Las expresiones regulares se verán con detalle en la UT6. Ejemplos:

```
let text = "Hello world, welcome to the universe.";
```



```
text.match("e") // devuelve e
```

```
text.match(/e/g) // devuelve e,e,e,e,e,e,e → cada vez que encuentra e
```



```
text.replace('e','3') //dev "H3llo world, welcome to the universe."
```

```
text.replace(/e/g,'3') //dev "H3llo world, w3lcome to th3 univ3rs3."
```

```
text.replace(/.o/, '5') //dev "Hel5 world, welcome to the universe."
```

```
text.replace(/.o/g, '5') //dev "Hel5 5rld, wel5me 5 the universe."
```

```
text.search(/w.rld/); → devuelve 6
```

- método `split` se indica por argumento el separador:

```
text.split(" ")[1] // devuelve "world"
```



```
text.split(" ") // devuelve Hello,world,,welcome,to,the,universe.
```

## 2.1. Objeto String (VII)

### Ejercicio

Realizar una aplicación en JavaScript que realice lo siguiente:

- Que escriba en la ventana principal `<h1>Ejercicio String</H2><HR />`
- Que solicite: introduzca su nombre y apellidos.

Una vez solicitados esos datos imprimirá en la ventana principal:

- o Buenos dias XXXXX
- o Tu nombre tiene XX caracteres, incluidos espacios.
- o La primera letra A de tu nombre está en la posición: X
- o La última letra A de tu nombre está en la posición: X
- o Tu nombre menos las 3 primeras letras es: XXXXXXXXX
- o Tu nombre todo en mayúsculas es: XXXXXXXXX

## 2.2. Objeto Math (I)

- El objeto **Math** es la calculadora de JavaScript
- Podremos realizar operaciones raíces cuadradas, logaritmos, operaciones trigonométricas y, muy importante, obtener números pseudoaleatorios.
- **Math no es un constructor** (no nos permitirá crear o instanciar nuevos objetos que sean de tipo **Math**)  
**Es directamente un objeto**
- Cómo en Java todos los métodos del objeto Math son estáticas

## 2.2. Objeto Math (II)

Formas de llamar a las propiedades y métodos de Math:

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

- Las propiedades de Math se utilizan para acceder a algunas constantes matemáticas de interés y son de sólo lectura.



## 2.2. Objeto Math (III)

Propiedades del objeto Math	
Propiedad	Descripción
<code>E</code>	Devuelve el número Euler (aproximadamente 2.718).
<code>LN2</code>	Devuelve el logaritmo neperiano de 2 (aproximadamente 0.693).
<code>LN10</code>	Devuelve el logaritmo neperiano de 10 (aproximadamente 2.302).
<code>LOG2E</code>	Devuelve el logaritmo base 2 de E (aproximadamente 1.442).
<code>LOG10E</code>	Devuelve el logaritmo base 10 de E (aproximadamente 0.434).
<code>PI</code>	Devuelve el número PI (aproximadamente 3.14159).
<code>SQRT2</code>	Devuelve la raíz cuadrada de 2 (aproximadamente 1.414).

## 2.2. Objeto Math (IV)



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

## 2.2. Objeto Math (V)

### 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 />");
```

## 2.2. Objeto Math (V)

### Generar números aleatorios:

```
Math.random() //devuelve un número decimal aleatorio de 0 a 1  
Math.random()*num //devuelve un número decimal aleatorio de 0 y num  
Math.random()*3 //devuelve un número decimal aleatorio de 0 y 3
```



```
Math.round(Math.random()*num) //devuelve un número aleatorio de 0 y num  
Math.round(Math.random()*3) //devuelve un número aleatorio entre 0,1,2,3
```

```
Math.round(Math.random()*10+20) //devuelve un número aleatorio 20 y 30  
(de 0 a 10, y luego sumándole 20)
```



## 2.3. Objeto Number (I)

- Los métodos del objeto **Number** ayudan a trabajar con números
- 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.
- Contiene alguna información y capacidades muy interesantes para programadores más serios.
- El objeto **Number** , es un objeto envoltorio para valores numéricos primitivos.

## 2.3. Objeto Number (II)

- 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  $2.22E-308$ .

Un número mayor que el número más alto, será considerado como infinito positivo

Un número 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**.
- Los objetos Number son creados con **new Number()** .Aunque es poco común.

```
numero = Number(123)
```

## 2.3. Objeto Number (III)

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

## 2.3. Objeto Number (IV)

Métodos del objeto Number	
Método	Descripción
<code>toExponential(x)</code>	Convierte un número a su notación exponencial.
<code>toFixed(x)</code>	Formatea un número con x dígitos decimales después del punto decimal.
<code>toPrecision(x)</code>	Formatea un número a la longitud x.
<code>toString()</code>	Convierte un objeto <code>Number</code> en una cadena. <ul style="list-style-type: none"><li>✓ Si se pone 2 como parámetro se mostrará el número en binario.</li><li>✓ Si se pone 8 como parámetro se mostrará el número en octal.</li><li>✓ Si se pone 16 como parámetro se mostrará el número en hexadecimal.</li></ul>
<code>valueOf()</code>	Devuelve el valor primitivo de un objeto <code>Number</code> .

```
19.656.toString();           // devuelve "19.656"
19.656.toExponential();      // devuelve 1.9656e+1
// siempre 1 dígito de parte entera
// por defecto se pone la parte decimal completa
19.656.toExponential(2);     // devuelve 1.97+1
// parte decimal con 2 dígitos redondeados
9.656.toFixed(2)             // devuelve 9.66 - 2 decimales redondeado
9.656.toPrecision(2)         // devuelve 9.7 - 2 dígitos en total
1209.656.toPrecision(6)      // devuelve 1209.66 - 6 dígitos
// Se fuerza la notación exponencial, devuelve 1.21e+3 - 3 dígitos
1209.656.toPrecision(3)
```



## 2.3. Objeto Number (V)

### 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);
```

## 2.3. Objeto Number (VI)

### Métodos de JavaScript para convertir números:

- **Number(valor)** // convierte valor en un número  
`Number(true);` // devuelve 1  
`Number(new Date());` // devuelve 1404568027739  
`Number("10");` // devuelve 10  
`Number("10 20");` // devuelve NaN
- **parseFloat(valor)** // convierte valor en un número float  
`parseFloat("10.33");` // devuelve 10.33
- **parseInt(valor)** // convierte valor en un número entero  
`parseInt("10.33");` // devuelve 10  
`parseInt("10 años");` // devuelve 10  
`parseInt("años 10");` // devuelve NaN

## 2.3. Objeto Number

`Number.isNaN(numero)`

- NaN es un valor que significa Not a Number, y es lo que devuelven las funciones que operan con números, y cuando no pueden realizar la operación porque uno de los operando no es un número.
- Método para saber si el valor guardado en numero es Not a Number o no. Este método es más robusto que `isNaN(numero)`

- Ejemplo:

```
let num;

while (Number.isNaN(num=parseInt(prompt("Escribe numero: "))))
{
    alert("Introduce un numero");
}
```

## 2.3. Objeto Boolean (I)

- El objeto Boolean se utiliza para **convertir un valor no Booleano a un valor Booleano** ( true o false ).
- Permite crear valores booleanos.

## 2.4. Objeto Boolean (III)

Dependiendo de lo que reciba el constructor de la clase Boolean el valor del objeto booleano que se crea será verdadero o falso:

Se inicializa a **false** cuando recibe:


- ningún valor al constructor o
- una cadena vacía,
- el número 0
- la palabra false sin comillas.

Se inicializa a **true** cuando recibe:

- cualquier valor entrecomillado
- cualquier número distinto de 0.

```
var b1 = new Boolean()  
document.write(b1 + "<br>")  
//muestra false  
var b2 = new Boolean("")  
document.write(b2 + "<br>")  
//muestra false  
var b25 = new Boolean(false)  
document.write(b25 + "<br>")  
//muestra false  
var b3 = new Boolean(0)  
document.write(b3 + "<br>")  
//muestra false  
var b35 = new Boolean("0")  
document.write(b35 + "<br>")  
//muestra true  
var b4 = new Boolean(3)  
document.write(b4 + "<br>")  
//muestra true  
var b5 = new Boolean("Hola")  
document.write(b5 + "<br>")  
//muestra true
```

## 2.5. Objeto Date (I)

- El objeto `Date` se utiliza para trabajar con fechas y horas. Permite realizar **controles relacionados con el tiempo en las aplicaciones web**.
- Los objetos `Date` se crean con `new Date()` .  
**Por defecto JavaScript crea un objeto `Date` con la hora local actual del sistema.**
- Las fechas en JavaScript se miden como el número de milisegundos transcurridos desde la época UNIX (1 de enero de 1970 00:00:00 UTC). 
- **El tiempo universal coordinado o UTC** es el principal estándar de tiempo por el cual el mundo regula los relojes y el tiempo. Es muy similar a la hora Greenwich. El tiempo local será el tiempo UTC con un desfase + o – de múltiplos de 1 hora dependiendo del huso horario local.
- El año debe expresarse con 4 dígitos,  
el mes entre 0 y 11,  
y el día entre 1 y 31.

## 2.5. Objeto Date (I)



- Hay 4 formas de instanciar (crear un objeto de tipo Date ):

```
var d = new Date ();
```

```
//Por defecto crea un objeto Date tomando la hora local actual del sistema.  
//esta hora no se actualiza en con el transcurso del tiempo
```

```
var d = new Date (milisegundosDesdeEpocaUNIX);
```




```
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.  
// Te crea una fecha local
```

## 2.5. Objeto Date (II)




- Cuenta con una serie de métodos divididos en tres subconjuntos: 

- Métodos de lectura. Empiezan por el prefijo get.

[https://www.w3schools.com/js/js\\_date\\_methods.asp](https://www.w3schools.com/js/js_date_methods.asp)

```
var d=new Date()
```



```
d.getDate()           // d.getUTCDate()
d.getDay()             // d.getUTCDay()
d.getFullYear()        // d.getUTCFullYear()
d.getHours()  // d.getUTCHours()
 d.getMilliseconds() // d.getUTCMilliseconds()
d.getMinutes()         // d.getUTCMinutes()
d.getMonth()           // d.getUTCMonth()
d.getTime()  // d.getUTCTime()
d.getTimezoneOffset()
d.getYear().
```



## 2.5. Objeto Date (II)

- Cuenta con una serie de métodos divididos en tres subconjuntos:

- **Métodos de escritura.** Empiezan por el prefijo `set`.

[https://www.w3schools.com/js/js\\_date\\_methods\\_set.asp](https://www.w3schools.com/js/js_date_methods_set.asp)

```
var d=new Date()
```

```
d.setDate()           // d.setUTCDate()
d.setDay()            // d.setUTCDay()
d.setFullYear()       // d.setUTCFullYear()
d.setHours()          // d.setUTCHours()
d.setMilliseconds()   // d.setUTCMilliseconds()
d.setMinutes()        // d.setUTCMinutes()
d.setMonth()          // d.setUTCMonth()
d.setTime()           // d.setUTCTime()
d.setTimezoneOffset()
d.setYear().
```

## 2.5. Objeto Date (II)

- Métodos de conversión. Convierte objetos de tipo Date en cadenas de texto o en milisegundos. `Date.parse()` 

[https://www.w3schools.com/js/js\\_date\\_formats.asp](https://www.w3schools.com/js/js_date_formats.asp)


.

## 2.5. Objeto Date (II)



- Métodos para mostrar la fecha y hora



```
var d=new Date()  
  
d.toString() // Thu Oct 13 2022  
d.toGMTString() //Thu, 13 Oct 2022 10:59:11 GMT  
d.toUTCString() //Thu, 13 Oct 2022 10:59:11 GMT  
d.toISOString() //2022-10-13T10:59:11.875Z  
d.toLocaleString() //13/10/2022, 12:59:11  
d.toLocaleDateString() //13/10/2022  
d.toLocaleTimeString() //12:59:11  
d.toString() // Thu Oct 13 2022 12:59:11 GMT+0200 (hora de verano de Europa central)
```

## 2.5. Objeto Date (III)

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

## 2.5. Objeto Date (IV)

### 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);
```

#### ---Otro---

```
var entrada = new Date(2016,12,22);
var salida = new Date(2017,1,6);
var noches =
    (salida.getTime()-entrada.getTime())/86400000;
// 86400000 es un día es milisegundos
alert(noches);
alert(entrada.toLocaleString());
alert(entrada);
```

#### ---Otro---

```
var d=new Date();
d.getHours() → devuelve la hora local. Pej:16
d.setHours(d.getHours()+2) → se incrementa en 2 horas
d.toLocaleString()→ la hora local es ahora 18
```

## 2.6. Ejercicio.

# Varios Objetos Nativos

Realizar una aplicación en JavaScript que realice lo siguiente:

- Que escriba en la ventana principal  
<h1>Ejercicio Objetos Nativo</H2><HR />
- Que solicite:
  - introduzca DIA de nacimiento.
  - introduzca MES de nacimiento.
  - introduzca AÑO de nacimiento.

Una vez solicitados esos datos imprimirá en la ventana principal:

- o Tu edad es: XX años.
- o Naciste un feliz XXXXXX del año XXXX.
- o El coseno de 180 es: XXXXXXXXXXXX
- o El número mayor de (34,67,23,75,35,19) es: XX
- o Ejemplo de número al azar: XXXXXXXXXXXX