

UT3_Modelo de Objetos predeterminados en Javascript

Modelo de objetos predefinidos en Javascript.

Unidad Trabajo nº 3

Autor: David García Fernández

Este obra está bajo una [licencia](http://creativecommons.org/licenses/by-nc-sa/4.0/) [de Creative Commons](http://creativecommons.org/licenses/by-nc-sa/4.0/) [Reconocimiento-NoComercial-CompartirIgual](http://creativecommons.org/licenses/by-nc-sa/4.0/) [4.0](http://creativecommons.org/licenses/by-nc-sa/4.0/) [Internacional](http://creativecommons.org/licenses/by-nc-sa/4.0/) [.](http://creativecommons.org/licenses/by-nc-sa/4.0/)

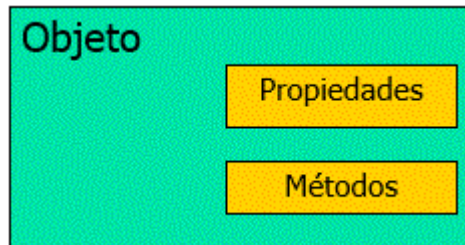


- Conocer los objetos predefinidos o nativos de Javascript.
- Saber generar texto y contenido HTML empleando lenguaje JavaScript.
- Saber crear nuevas ventanas desde JavaScript y ver cómo permitir la comunicación entre ellas.
- Conocer las diferentes opciones para modificar la apariencia de las ventanas de nueva creación desde JavaScript.
- Saber cómo ejecutar código entre marcos.

Introducción

Los objetos en Javascript.

- W3C define el DOM (Modelo de Objetos del Documento) como una interfaz de programación de aplicaciones (API), define la estructura lógica de los documentos y el modo en el que se acceden y se manipulan los objetos.
- **Objeto en javascript:** una entidad con una serie de propiedades que definen su estado, y unos métodos (funciones), que actúan sobre esas propiedades.
- Referencias al objeto:
 - Objeto.propiedad
 - Objeto.método([parámetros])



- La creación de objetos se realiza mediante el operador **new**.

```
var nuevo_objeto = new [Nombre_Objeto]([parámetros])
```

Objetos nativos de Javascript.

Objetos.

En JavaScript, la mayoría de elementos del lenguaje son objetos, tan solo los operadores y tipos simples quedan fuera de esta categoría.

Tanto las características del núcleo de JavaScript como arrays hasta el API del explorador está construido sobre objetos. Además, se pueden crear objetos propios de usuario para encapsular funciones y variables relacionadas.

En esta unidad, estudiaremos los siguientes objetos nativos básicos: String, Number, Date, Boolean y Math.

Objeto String

- El objeto String permite manipular las cadenas de texto.
- Una cadena (string) consta de uno o más caracteres de texto, rodeados de comillas simples (') o dobles (").

Propiedades del objeto String:

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

Propiedades del objeto String:

Métodos del objeto String	
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.

Debes saber:

- Plantillas literales

Las cadenas de las plantillas literales proporcionan interpolación de cadenas, esto es, sustitución y evaluación del dato que contienen.

Las plantillas literales están encerradas por la comilla invertida (`)

```
// plantilla literal. Interpolación de cadenas
var name = 'Pepe', time = 'hoy';
alert(`Hola ${name}, ¿cómo estás ${time}?`);
```

- **Expresiones Regulares: Objeto RegExp.**

Una expresión regular es una notación específica para comparar cadenas.

Metacaracteres para el uso de Expresiones Regulares: \ ^ \$. [] [^] | () * + ?

\ Carácter de escape, usado para \n, \t, etc.

^ Carácter para inicio de cadena, por ejemplo ^G (Se inicia con G)

\$ Carácter de fin de cadena, por ejemplo a\$ (que acaban en a)

. Cualquier carácter (solo uno).

[] Caracteres encerrados entre corchetes, cualquier carácter entre [] es válido. Por ejemplo: [ABC], [a-zA-Z] [0-9]

[^] Caracteres distintos a los encerrados entre corchetes. Por ejemplo: [^0-9], que no sean dígitos.

| Concordancia OR.

() Concordancia con un carácter entre paréntesis.

* Concuera con ninguna o más veces.

+ Concuera con una o más veces.

? Concuera con ninguna o una vez.

Pueden darse casos como el siguiente:

- ^[ABC] concuerda con A,B o C como primer carácter de la cadena.
- ^[^ABC] concuerda con cualquier carácter distinto a A,B o C como primer carácter de la cadena.
- ^[a-z]\$ concuerda con un carácter simple, excepto letra en minúscula.
- A|B Concordancia con A o B.
- (r) Concordancia con un carácter r
- A* Concuera con ninguna o más de una A.
- A+ Concuera con una o más A.
- A? Concuera con ninguna o una A.

Ejemplo: Validación de un número entre 0-10.

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  |   <meta charset="utf-8">
5  |   <title>JS input</title>
6  </head>
7  <body>
8  |   <h2>CAMPO 1</h2>
9  |   <input type="number" name="texto1" id="campo1">
10 |   <button onclick="enviar()">Prueba</button>
11 |   <script>
12 |   var valor1 = null;
13 |
14 |   //submit
15 |   function enviar(){
16 |       |
17 |       |
18 |       |
19 |       |
20 |       |
21 |   }
22 |   </script>
23 </body>
24 </html>
```

La siguiente web sirve para validar expresiones regulares: <https://regex101.com/>
<<https://regex101.com/>>




Objeto Number

- El objeto Number permite realizar operaciones y acciones sobre datos numéricos.
- Los números y sus valores están definidos internamente en JavaScript, como valores de doble precisión en coma flotante de 64 bits.
- 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.
- Suele utilizarse para operar con los métodos propios del objeto y/o propiedades, ya que con el manejo de variables numéricas y operadores aritméticos se satisface cualquier utilidad.

Propiedades del objeto Number:

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.

Métodos del objeto Number:

<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>	<p>Convierte un objeto <code>Number</code> en una cadena.</p> <ul style="list-style-type: none"> ✓ Si se pone 2 como parámetro se mostrará el número en  binario. ✓ Si se pone 8 como parámetro se mostrará el número en  octal. ✓ Si se pone 16 como parámetro se mostrará el número en  hexadecimal.
<code>valueOf()</code>	Devuelve el valor primitivo de un objeto <code>Number</code> .

Objeto Date

- Permite realizar operaciones con datos de fechas y horas.

Propiedades del objeto:

Propiedades del objeto Date	
Propiedad	Descripción
<code>constructor</code>	Devuelve la función que creó el objeto <code>Date</code> .
<code>prototype</code>	Te permitirá añadir propiedades y métodos a un objeto.

Métodos del objeto:

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).

Objeto Math

- Permite realizar operaciones matemáticas distintas a las ofrecidas por los operadores aritméticos (+, -, *, etc.)
- Además, incluye constantes de cálculo (PI) accesibles a través de sus propiedades.

Propiedades del objeto:

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).

Métodos del objeto:

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.

Objeto Boolean

- Permite trabajar con datos booleanos (true, false).
- Transforma datos numéricos o de cadena en datos booleanos (útil en expresiones de comparación).
- A la hora de crear un objeto con Boolean admite distintos parámetros que dan como resultado:
 - true. Si el valor de parámetro proporcionado está entrecomillado o es distinto de 0.
 - false. Si no se pasa valor al parámetro, o cadena vacía, o el número 0.
 - Ejemplos:

```
var b1 = new Boolean()
document.write(b1 + "<br>")
//muestra false
```

```
var b3
document.write(b3 + "<br>")
//muestra false
```

```
var b2 = new Boolean("")
document.write(b2 + "<br>")
//muestra false
```

```
var b4 = new Boolean(0)
document.write(b4 + "<br>")
//muestra false
```

```
var b25 = new Boolean(false)
```

```
var b4
document.write(b4 + "<br>")
//muestra false
```



```
document.write(b25 + "<br>")
//muestra false

var b5
document.write(b5)
```

Propiedades del objeto:

Propiedades del objeto Boolean	
Propiedad	Descripción
<code>constructor</code>	Devuelve la función que creó el objeto <code>Boolean</code> .
<code>prototype</code>	Te permitirá añadir propiedades y métodos a un objeto.

Métodos del objeto:

Métodos del objeto Boolean	
Método	Descripción
<code>toString()</code>	Convierte un valor <code>Boolean</code> a una cadena y devuelve el resultado.
<code>valueOf()</code>	Devuelve el valor primitivo de un objeto <code>Boolean</code> .



Debes saber:

Cuando evalúas una variable null, el valor nulo se comporta como 0 en contextos numéricos y como false en contextos booleanos.

Por ejemplo:

```
var n = null;
console.log(n * 32); // Registrará 0 en la consola
```

Investigación:

¿Sabes cómo javascript puede utilizar los datos introducidos en un campo de texto html?

Solución. Ejemplo.

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4  |   <meta charset="utf-8">
5  |   <title>JS input</title>
6  </head>
7  <body>
8  |   <h2>CAMPO 1</h2>
9  |   |   <input type="text" name="texto1" id="campo1">
10 |   <h2>CAMPO 2</h2>
11 |   |   <input type="text" name="numero2" id="campo2">
12 |   <button onclick="enviar()">Prueba</button>
13 |   <script>
14 |   |   var valor1 = null;
15 |   |   var valor2 = null;
16 |
17 |   |   //submit
18 |   |   function enviar(){
19 |   |   |   valor1 = document.getElementById("campo1").value;
20 |   |   |   valor2 = document.getElementById("campo2").value;
21 |   |   |   alert( `Los valores son: ${valor1} y ${valor2} ` );
22 |   |   |   valor1 = document.getElementsByName("texto1")[0].value;
23 |   |   |   valor2 = document.getElementsByName("numero2")[0].value;
24 |   |   |   alert(valor1 + " " + valor2);
25 |   |   }
26 |   }
27 |   </script>
28 </body>
29 </html>

```

Rellenar huecos

Convierte el texto en mayúsculas:

```
let txt = "Hello World!";
txt = txt.  ();
```

Rellenar huecos

Utiliza el método slice para devolver la palabra "todos".

```
let txt = "voy a aprobar todos los módulos";
let x = txt.slice(  ,  );
```

Rellenar huecos

¿Cómo se obtiene el mes de la fecha actual?

```
const d = new Date();  
mes=  ;
```

Rellenar huecos

El siguiente ejemplo realiza una validación para que se introduzca un número comprendido entre 0 y 100 a través de una expresión regular. Rellena las cajas de texto para que funcione correctamente.

```
function enviar(){  
    valor1 = document.getElementById(" campo1 ").value;  
    resultado =  .test(  );  
    alert( ` Los valores son: ${valor1} y ${resultado} ` );  
};
```

`^[0-9]$` Significa que empieza por un dígito comprendido entre 0 y 9 y termina.

`{1,2}` Significa que puede aparecer 1 o dos veces la secuencia anterior.

`|` Significa que puede coincidir con lo que hay a la izquierda del `|` o bien a su derecha.

`^100$` Significa que puede coincidir con el número 100. `^` empieza y `$` acaba, por lo que está limitado exclusivamente a que sea 100.

Funciones

Funciones en Javascript.

Las funciones como en cualquier lenguaje de programación permiten:

- Estructurar el código en bloques funcionales.

- Reutilizar partes de código.
- Optimizar la memoria almacenando el bloque de función y referenciándolo.

Declaración

Las funciones de JavaScript se definen con la palabra clave function.

Para su declaración se puede utilizar una declaración de función propiamente:

```
function nombreFunción (parámetro1, parámetro2, ... parámetroN) {  
  ...código de la función...  
}
```

O declaración por asignación:

```
var variable = function nombreFunción (parámetro1, parámetro2, ..  
  ...código de la función...  
}
```



O declaración por función flecha:

```
let func = (arg1, arg2, ..., argN) => expression;  
  
let func = function(arg1, arg2, ..., argN) {  
  return expression;  
};
```

Ejemplo:

```
let sum = (a, b) => a + b;  
  
/* Esta función de flecha es una forma más corta de:
```

```
let sum = function(a, b) {  
    return a + b;  
};  
*/  
  
alert( sum(1, 2) ); // 3
```

Visibilidad de las variables.

El ámbito o visibilidad de las variables de una función es el mismo que el estudiado en el tema anterior a nivel de bloques.

- Los valores pasados a una función como parámetros se copian a sus variables locales.
- Una función puede acceder a variables externas. Pero funciona solo de adentro hacia afuera. El código fuera de la función no ve sus variables locales.
- Una función puede devolver un valor. Si no lo hace, entonces su resultado es undefined.
- Las variables globales son visibles desde cualquier función (a menos que se les superpongan variables locales con el mismo nombre).

Buenas prácticas:

Es una buena práctica reducir el uso de variables globales. El código moderno tiene pocas o ninguna variable global. La mayoría de las variables residen en sus funciones. Aunque a veces puede justificarse almacenar algunos datos a nivel de proyecto.

Parámetros de una función.

Los valores pasados por parámetros (argumentos) se copian en variables locales a la función con el nombre que los identifica.

```
function saludar(a,b){  
    alert("Hola " + a + " y " + b + ".");  
}
```

```
function devolverMayor(a,b){  
  if (a > b) then  
    return a;  
  else  
    return b;  
}
```

Valores por defecto en los parámetros.

Si una función es llamada pero no se le proporciona un argumento, su valor correspondiente se convierte en undefined.

Se puede especificar un valor llamado “predeterminado” o “default” (que se usa si el argumento fue omitido) en la declaración de función usando =

```
function saludar(a="Juan",b="Luis"){  
  alert("Hola " + a + " y " + b + ".");  
}  
  
// saludar();    Se visualiza Hola Juan y Luis.
```

Funciones anidadas.

Podemos encapsular la accesibilidad de una función dentro de otra y hacer que esa función sea privada o local a la función principal.

Esta técnica se utiliza cuando tenemos una secuencia de instrucciones que necesitan ser llamadas desde múltiples sitios dentro de una función, y esas instrucciones sólo tienen significado dentro del contexto de esa función principal.

Ejemplo:

```
function hipotenusa(a, b){  
  function cuadrado(x){  
    return x*x;  
  }  
  return Math.sqrt(cuadrado(a) + cuadrado(b));  
}  
document.write("<br/>La hipotenusa de 1 y 2 es: "+hipotenusa(1,2));
```


Funciones predefinidas en Javascript.

Javascript dispone de algunos elementos que necesitan ser tratados a escala global y que no pertenecen a ningún objeto en particular (o que se pueden aplicar a cualquier objeto).

Propiedades globales:

Propiedad	Descripción
Infinity	Un valor numérico que representa el infinito positivo/negativo.
NaN	Valor que no es numérico "Not a Number".
undefined()	Indica que a esa variable no le ha sido asignado un valor.

Funciones globales:

Función	Descripción
decodeURI()	Decodifica los caracteres especiales de una URL excepto: , / ? : @ & = + \$ #
decodeURIComponent()	Decodifica todos los caracteres especiales de una URL.
encodeURI()	Codifica los caracteres especiales de una URL excepto: , / ? : @ & = + \$ #
encodeURIComponent()	Codifica todos los caracteres especiales de una URL.
escape()	Codifica caracteres especiales en una cadena, excepto: * @ - _ + . /
eval()	Evalúa una cadena y la ejecuta si contiene código u operaciones.
isFinite()	Determina si un valor es un número finito válido.
isNaN()	Determina cuando un valor no es un número.
Number()	Convierte el valor de un objeto a un número.
parseFloat()	Convierte una cadena a un número real.
parseInt()	Convierte una cadena a un entero.
unescape()	Decodifica caracteres especiales en una cadena, excepto: * @ - _ + . /

Ejemplo:

```
<script type="text/javascript">
eval("x=50;y=30;document.write(x*y)"); // Imprime 1500
document.write("<br />" + eval("8+6")); // Imprime 14
document.write("<br />" + eval(x+30)); // Imprime 80
</script>
```

UT3. Práctica1

UT3. Práctica 1.

Duración: 3 horas

Ejercicio 1:

Modifica el ejercicio de UT2.Práctica_2.Ejercicio3, para utilizar un método del objeto String que acceda a la posición de la cadena dada de letras del DNI y obtenga la letra del número DNI introducido.

Ejercicio 2:

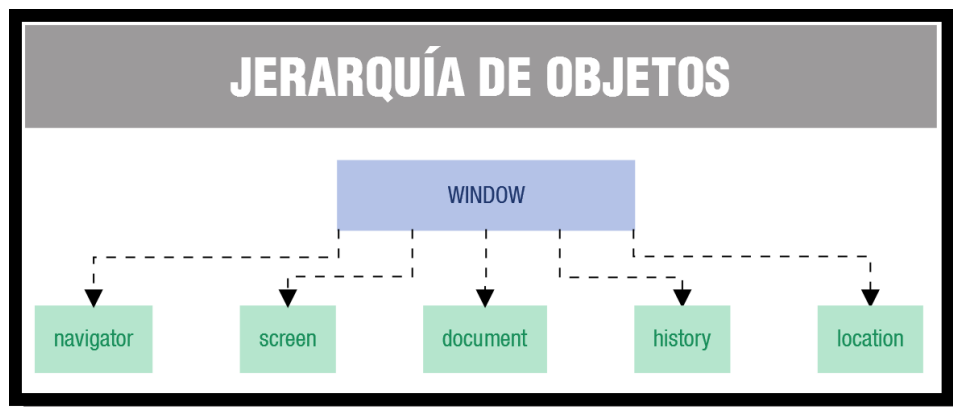
Realizar los cálculos de notas del alumno.

- Se introducirán por teclado las notas referentes a: 2 notas de prácticas y 1 nota de examen de evaluación.
- Los porcentajes de calificación serán: 45% las prácticas, 45% examen, 10% actitud.
- Para las notas de prácticas se hace la media entre las 2 notas de prácticas.
- La nota de actitud solo es aplicable si las notas de prácticas y exámenes son igual o superior a 5.
- Es necesario que las notas sean de 4 o superior para poder realizar la media.

Objetos de alto nivel de Javascript.

Objetos de alto nivel de Javascript.

- Los objetos de alto nivel en Javascript están jerarquizados.
- Son: window, navigator, screen, document, history y location.



- Estos objetos están incluidos en el BOM (Browser Object Model).

Objeto Window

- El objeto window es compatible en todos los navegadores.
- Representa la ventana del navegador.
- Todos los objetos, variables y funciones globales se convierten automáticamente en miembros del objeto window.
- El objeto window se crea cuando el usuario ejecuta "abrir un navegador".
- Con javascript se puede abrir subventanas desde el objeto principal window.

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

- Si un documento contiene marcos (<frame> o <iframe>), el navegador crea un objeto window para el documento HTML, y un objeto window adicional para cada marco.

Propiedades:

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.

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.

**Debes saber:**

Existen otros métodos del objeto window pero al no estar estandarizado el BOM, cada navegador admite unos y otros no. Para ello se debe consultar la referencia específica del navegador y su implementación de window.

Objeto Screen

- El objeto screen se utiliza para obtener información sobre la pantalla del usuario.
- Proporciona la resolución del monitor en el que se están visualizando las páginas. Los diseñadores de páginas web necesitan conocer las resoluciones más utilizadas por los usuarios para adaptar sus diseños a esas resoluciones.

Propiedades del objeto:

Propiedad	Descripción
availHeight	Altura de pantalla disponible para las ventanas
availWidth	Anchura de pantalla disponible para las ventanas
colorDepth	Profundidad de color de la pantalla (32 bits normalmente)
height	Altura total de la pantalla en píxel
width	Anchura total de la pantalla en píxel

Ejemplo: Redimensionar la ventana en función del tamaño de la pantalla.

```
window.moveTo(0, 0);  
window.resizeTo(screen.availWidth, screen.availHeight);
```

Objeto Navigator

- Contiene información sobre el navegador que estamos utilizando cuando abrimos una URL o un documento local.

Propiedades del objeto:

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> .

Métodos del objeto:

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

Objeto Location

- El objeto `location` contiene información referente a la URL actual.
- Este objeto, es parte del objeto `window` y accedemos a él a través de la propiedad `window.location`.

Propiedades del objeto:

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

Métodos del objeto:

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> .

Objeto Document

- Cada documento cargado en una ventana del navegador, será un objeto de tipo document.
- El objeto document proporciona a los scripts, el acceso a todos los elementos HTML dentro de una página.
- Este objeto forma parte además del objeto window, y puede ser accedido a través de la propiedad window.document o directamente document (ya que podemos omitir la referencia a la window actual).

Colecciones del objeto:

Colecciones del objeto Document	
Colección	Descripción
<code>anchors[]</code>	Es un array que contiene todos los hiperenlaces del documento.
<code>applets[]</code>	Es un array que contiene todos los applets del documento
<code>forms[]</code>	Es un array que contiene todos los formularios del documento.
<code>images[]</code>	Es un array que contiene todas las imágenes del documento.
<code>links[]</code>	Es un array que contiene todos los enlaces del documento.

Propiedades del objeto:

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.

Propiedades del objeto Document	
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.

Marcos (frame, iframe)

Marcos

- Un objeto frame, representa un marco HTML. La etiqueta <frame> identifica una ventana particular, dentro de un conjunto de marcos (frameset).
- Para cada etiqueta <frame> en un documento HTML, se creará un objeto frame.
- Todo lo anterior se aplicará también al objeto Iframe <iframe>.

Propiedades del objeto:

Propiedades del objeto Frame/Iframe	
Propiedad	Descripción
<code>align</code>	Cadena que contiene el valor del atributo <code>align</code> (alineación) en un <code>iframe</code> .
<code>contentDocument</code>	Devuelve el objeto documento contenido en un <code>frame/iframe</code> .
<code>contentWindow</code>	Devuelve el objeto <code>window</code> generado por un <code>frame/iframe</code> .
<code>frameBorder</code>	Cadena que contiene el valor del atributo <code>frameborder</code> (borde del marco) de un <code>frame/iframe</code> .
<code>height</code>	Cadena que contiene el valor del atributo <code>height</code> (altura) de un <code>iframe</code> .
<code>longDesc</code>	Cadena que contiene el valor del atributo <code>longdesc</code> (descripción larga) de un <code>frame/iframe</code> .
<code>marginHeight</code>	Cadena que contiene el valor del atributo <code>marginheight</code> (alto del margen) de un <code>frame/iframe</code> .
<code>marginWidth</code>	Cadena que contiene el valor del atributo <code>marginwidth</code> (ancho del margen) de un <code>frame/iframe</code> .
<code>name</code>	Cadena que contiene el valor del atributo <code>name</code> (nombre) de un <code>frame/iframe</code> .
<code>noResize</code>	Cadena que contiene el valor del atributo <code>noresize</code> de un <code>frame/iframe</code> .
<code>scrolling</code>	Cadena que contiene el valor del atributo <code>scrolling</code> (desplazamiento) de un <code>frame/iframe</code> .
<code>src</code>	Cadena que contiene el valor del atributo <code>src</code> (origen) de un <code>frame/iframe</code> .
<code>width</code>	Cadena que contiene el valor del atributo <code>width</code> (ancho) de un <code>iframe</code> .

Eventos:

Eventos del objeto Frame/Iframe	
Evento	Descripción
<code>onload</code>	Script que se ejecutará inmediatamente después a que se cargue el <code>frame/iframe</code> .

Debes saber:

Los marcos (frames) han dejado de utilizarse desde hace algunos años, el motivo es que los buscadores no pueden indexar información contenida en ellos.

Además HTML5 ya no soporta las etiquetas frame ni frameset. (HTML4 sí)

Como solución alternativa algunos diseñadores web incluyen las palabras claves importantes en las etiquetas del título y del meta del frameset.

Comunicación entre ventanas

Comunicación entre ventanas.

- Abrir una ventana desde otra:

Cada objeto window tiene una propiedad llamada opener. Esta propiedad contiene la referencia a la ventana o marco, que ha abierto ese objeto window empleando el método open(). Para la ventana principal el valor de opener será null.

En el siguiente ejemplo se muestra el código de cómo abrir una ventana utilizando el método open():

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta http-equiv="content-type" content="text/html; charset=utf-8">
5      <title>Apertura y Cierre de Ventanas</title>
6      <script type="text/javascript">
7          function inicializar() {
8              document.getElementById("crear-ventana").onclick = crearNueva;
9              document.getElementById("cerrar-ventana").onclick = cerrarNueva;
10         }
11         var nuevaVentana;
12         function crearNueva() {
13             nuevaVentana = window.open("http://www.google.es", "", "height=400,width=800");
14         }
15         function cerrarNueva() {
16             if (nuevaVentana) {
17                 nuevaVentana.close(); nuevaVentana = null;
18             }
19         }
20     </script>
21 </head>
22
23 <body onLoad="inicializar()">
24     <h1>Abrimos y cerramos ventanas</h1>
25     <form>
26         <p> <input type="button" id="crear-ventana" value="Crear Nueva Ventana">
27             <input type="button" id="cerrar-ventana" value="Cerrar Nueva Ventana">
28         </p>
29     </form>
30 </body>
31 </html>

```

La mayoría de los navegadores bloquean las ventanas emergentes si se llaman fuera de los controladores de eventos activados por el usuario, como onclick.

```

// popup blocked
window.open("https://javascript.info");
// popup allowed
button.onclick = () => { window.open("https://javascript.info")

```



- Acceso de la ventana principal a la sub-ventana:

La llamada `open` devuelve una referencia a la nueva ventana. Se puede usar para manipular sus propiedades, cambiar de ubicación y aún más.

```
let nuevaVentana = open("/", "ejemplo", "width=300,height=300")
nuevaVentana .focus(); alert(nuevaVentana .location.href); // (
nuevaVentana .onload = function() { let html = `<div style="for
    nuevaVentana .document.body.insertAdjacentHTML("afterbegin"
};
```



- Acceso a la ventana principal desde la sub-ventana:

```
let ventanaNueva= window.open("about:blank", "hola", "width=200
ventanaNueva.document.write( "<script>window.opener.document.bc
```



Debes conocer: La política de “Mismo origen” (mismo sitio) limita el acceso de ventanas y marcos entre sí. La idea es que si un usuario tiene dos páginas abiertas: una de `john-smith.com`, y otra es `gmail.com`, entonces no querrán que un script de `john-smith.com` lea nuestro correo de `gmail.com`. Por lo tanto, el propósito de la política de “Mismo origen” es proteger a los usuarios del robo de información.

UT3. Práctica 2

UT3. Práctica 2

Ejercicio 1:

- Crea una ventana que se abra con el aula virtual del curso DWEC en Educamadrid y posicionala con su esquina superior-izq en el centro exacto de tu pantalla.

Ejercicio 2:

- Investiga como utilizar el objeto `window.history` para crear en un documento html (con cualquier contenido) dos botones que vuelvan a la página anterior y a la siguiente.

Ejercicio 3:

Realizar una aplicación en JavaScript que realice lo siguiente:

- Abra una nueva ventana no redimensionable.
- Hacer una función y dentro de esa función:
- Escribir en la nueva ventana <h3>Ejemplo de Ventana Nueva</h3>
- URL Completa: XXXXX
- Protocolo utilizado: XXXXX
- Nombre en código del navegador: XXXXX
- Que detecte si está JAVA disponible en esa ventana del navegador y si es así que escriba:
 - Java SI disponible en esta ventana, o bien.
 - Java NO disponible en esta ventana.

Y ahora fuera del código de la función que siga haciendo lo siguiente:

- Que escriba en la ventana principal <h1>TAREA DWECE03</H2><HR />
- Que solicite: introduzca su nombre y apellidos.
- Que solicite: introduzca DIA de nacimiento.
- Que solicite: introduzca MES de nacimiento.
- Que solicite: introduzca AÑO de nacimiento.
- Una vez solicitados esos datos imprimirá en la ventana principal:
- Buenos días XXXXX
- Tu nombre tiene XX caracteres, incluidos espacios.
- La primera letra A de tu nombre está en la posición: X
- La última letra A de tu nombre está en la posición: X
- Tu nombre menos las 3 primeras letras es: XXXXXXXXX
- Tu nombre todo en mayúsculas es: XXXXXXXXX
- Tu edad es: XX años.
- Naciste un feliz XXXXXX del año XXXX.
- El coseno de 180 es: XXXXXXXXXXXX
- El número mayor de (34,67,23,75,35,19) es: XX
- Ejemplo de número al azar: XXXXXXXXXXXX

Donde aparecen las XXXX tendrá que aparecer el cálculo o texto que corresponda.

Ejercicio 4:

Desde una ventalla llamada padre se deberá abrir una ventana llamada hija. Cada ventana deberá poder enviar un mensaje a la otra y que se visualice dicho mensaje en la ventana destino. (comunicación bidireccional).

Nota: Será necesario utilizar xampp para alojar los ficheros html y js que crees.

Bibliografía y recursos

Tutoriales y Referencias de Javascript: <<https://www.w3schools.com/js/default.asp>>

- h <<https://www.w3schools.com/js/default.asp>> <https://www.w3schools.com/js/default.asp> <<https://www.w3schools.com/js/default.asp>>

- <https://developer.mozilla.org/es/docs/Learn/JavaScript>
<<https://developer.mozilla.org/es/docs/Learn/JavaScript>>
- <https://es.javascript.info/> <<https://es.javascript.info/>>

Expresiones Regulares:

- <https://regex101.com/> <<https://regex101.com/>>

Apuntes cedidos por el Ministerio de Educación y Formación Profesional.

Obra publicada con [Licencia Creative Commons Reconocimiento No comercial Compartir igual 4.0](https://creativecommons.org/licenses/by-nc-sa/4.0/)
<[http://creativecommons.org/licenses/by-nc-sa/4.0/](https://creativecommons.org/licenses/by-nc-sa/4.0/)>